



**N32G031 series**

**32-bit ARM Cortex<sup>®</sup>-M0 microcontroller**

User manual V2.2.0

## Contents

<b>1 Abbreviations in the text .....</b>	<b>27</b>
1.1 List of abbreviations for registers .....	27
1.2 Available peripherals .....	27
<b>2 Memory and bus architecture.....</b>	<b>28</b>
2.1 System architecture.....	28
2.1.1 Bus architecture .....	28
2.1.2 Bus address mapping .....	30
2.1.3 Boot management .....	32
2.2 Memory system .....	33
2.2.1 FLASH specification .....	33
2.2.2 SRAM .....	45
2.2.3 FLASH register description .....	46
<b>3 Power control (PWR) .....</b>	<b>53</b>
3.1 General description.....	53
3.1.1 Power supply.....	53
3.1.2 Power supply supervisor .....	54
3.1.3 NRST .....	56
3.2 Power modes .....	56
3.2.1 LPRUN mode .....	58
3.2.2 SLEEP mode.....	59
3.2.3 STOP mode.....	59
3.2.4 PD mode .....	60
3.3 Debug support .....	61
3.3.1 Low power mode debug support.....	61
3.3.2 Peripheral debug support .....	61
3.4 PWR registers .....	61
3.4.1 PWR register overview .....	61
3.4.2 Power control register (PWR_CTRL).....	62
3.4.3 Power control status register (PWR_CTRLSTS).....	63
3.4.4 Power control register 2 (PWR_CTRL2).....	65
3.4.5 Power control register 3 (PWR_CTRL3).....	65
3.4.6 Power control register 4 (PWR_CTRL4).....	65
3.4.7 Power control register 5 (PWR_CTRL5).....	66
3.4.8 Power control register 6 (PWR_CTRL6).....	67
3.4.9 Debug control register (DBG_CTRL) .....	67
<b>4 Reset and clock control (RCC) .....</b>	<b>70</b>
4.1 Reset Control Unit .....	70

4.1.1 Power reset.....	70
4.1.2 System reset .....	70
4.2 Clock control unit .....	72
4.2.1 Clock Tree Diagram.....	74
4.2.2 HSE clock .....	75
4.2.3 HSI clock .....	76
4.2.4 PLL clock.....	76
4.2.5 LSE clock.....	77
4.2.6 LSI clock.....	77
4.2.7 System clock (SYSCLK) selection.....	78
4.2.8 Clock security system (CLKSS) .....	78
4.2.9 RTC clock .....	78
4.2.10 Watchdog clock.....	78
4.2.11 LPUART clock .....	78
4.2.12 LPTIME clock .....	79
4.2.13 Clock output(MCO).....	79
4.3 RCC registers.....	79
4.3.1 RCC register overview.....	79
4.3.2 Clock control register (RCC_CTRL).....	81
4.3.3 Clock configuration register (RCC_CFG) .....	82
4.3.4 Clock interrupt register (RCC_CLKINT) .....	85
4.3.5 APB2 peripheral reset register (RCC_APB2PRST) .....	88
4.3.6 APB1 peripheral reset register (RCC_APB1PRST) .....	89
4.3.7 AHB peripheral clock enable register (RCC_AHBCLKEN).....	90
4.3.8 APB2 peripheral clock enable register (RCC_APB2PCLKEN).....	92
4.3.9 APB1 peripheral clock enable register (RCC_APB1PCLKEN).....	93
4.3.10 Low speed clock control register (RCC_LSCTRL).....	95
4.3.11 Control/status register (RCC_CTRLSTS) .....	96
4.3.12 AHB peripheral reset register (RCC_AHBPRST).....	98
4.3.13 Clock configuration register 2(RCC_CFG2) .....	99
4.3.14 EMC control register 3 (RCC_EMCTRL) .....	101
<b>5 GPIO and AFIO .....</b>	<b>104</b>
5.1 Summary.....	104
5.2 Function description .....	105
5.2.1 I/O mode configuration.....	105
5.2.2 Status after reset.....	110
5.2.3 Individual bit setting and bit clearing.....	110
5.2.4 External interrupt /wakeup line.....	110
5.2.5 Alternate function .....	110
5.2.6 I/O configuration of peripherals.....	120
5.2.7 GPIO Locking mechanism.....	122
5.3 GPIO Registers .....	123

5.3.1	GPIO register overview .....	123
5.3.2	GPIO port mode description register (GPIOx_PMODE).....	124
5.3.3	GPIO port type definition (GPIOx_POTYPE) .....	125
5.3.4	GPIO slew rate configuration register (GPIOx_SR).....	125
5.3.5	GPIO port pull-up/pull-down register (GPIOx_PUPD).....	126
5.3.6	GPIO port input data register (GPIOx_PID).....	127
5.3.7	GPIO port output data register (GPIOx_POD) .....	127
5.3.8	GPIO port bit set/clear register (GPIOx_PBSC).....	128
5.3.9	GPIO port configuration lock register (GPIOx_PLOCK).....	129
5.3.10	GPIO alternate function low register (GPIOx_AFL).....	129
5.3.11	GPIO alternate function high register (GPIOx_AFH) .....	130
5.3.12	GPIO port bit clear register (GPIOx_PBC) .....	131
5.3.13	GPIO driver strength configuration register (GPIOx_DS) .....	132
5.4	AFIO Registers .....	132
5.4.1	AFIO register overview .....	132
5.4.2	AFIO configuration register (AFIO_CFG) .....	133
5.4.3	AFIO external interrupt configuration register 1 (AFIO_EXTI_CFG1).....	134
5.4.4	AFIO external interrupt configuration register 2 (AFIO_EXTI_CFG2).....	134
5.4.5	AFIO external interrupt configuration register 3 (AFIO_EXTI_CFG3).....	135
5.4.6	AFIO external interrupt configuration register 4 (AFIO_EXTI_CFG4).....	136
<b>6</b>	<b>Interrupts and events.....</b>	<b>137</b>
6.1	Nested vectored interrupt controller .....	137
6.1.1	SysTick calibration value register.....	137
6.1.2	Interrupt and exception vectors.....	137
6.2	External interrupt/event controller (EXTI) .....	140
6.2.1	Introduction.....	140
6.2.2	Main features .....	140
6.2.3	Functional description.....	141
6.2.4	EXTI line mapping .....	143
6.3	EXTI Registers .....	145
6.3.1	EXTI register overview.....	145
6.3.2	Interrupt mask register(EXTI_IMASK).....	145
6.3.3	Event mask register(EXTI_EMASK) .....	146
6.3.4	Rising edge trigger selection register(EXTI_RT_CFG).....	146
6.3.5	Falling edge trigger selection register(EXTI_FT_CFG) .....	147
6.3.6	Software interrupt enable register(EXTI_SWIE).....	147
6.3.7	Interrupt request pending register(EXTI_PEND) .....	148
6.3.8	RTC Timestamp trigger source selection register (EXTI_TS_SEL).....	149
<b>7</b>	<b>DMA controller .....</b>	<b>150</b>
7.1	Introduction .....	150
7.2	Main features .....	150

7.3 Block diagram .....	151
7.4 Function description .....	151
7.4.1 DMA operation .....	151
7.4.2 Channel priority and arbitration .....	152
7.4.3 DMA channels and number of transfers.....	152
7.4.4 Programmable data bit width, alignment and endians .....	152
7.4.5 Peripheral/Memory address incrementation .....	154
7.4.6 Channel configuration procedure .....	154
7.4.7 Flow control.....	155
7.4.8 Circular mode .....	156
7.4.9 Error management.....	156
7.4.10 Interrupt .....	156
7.4.11 DMA request mapping.....	157
7.5 DMA registers .....	157
7.5.1 DMA register overview.....	157
7.5.2 DMA interrupt status register (DMA_INTSTS) .....	158
7.5.3 DMA interrupt flag clear register (DMA_INTCLR).....	159
7.5.4 DMA channel x configuration register (DMA_CHCFGx).....	160
7.5.5 DMA channel x transfer number register (DMA_TXNUMx) .....	162
7.5.6 DMA channel x peripheral address register (DMA_PADDRx) .....	162
7.5.7 DMA channel x memory address register (DMA_MADDRx) .....	163
7.5.8 DMA channel x channel request select register (DMA_CHSELx).....	163
<b>8 CRC calculation unit .....</b>	<b>165</b>
8.1 CRC introduction.....	165
8.2 CRC main features.....	165
8.2.1 CRC32 module .....	165
8.2.2 CRC16 module .....	165
8.3 CRC function description .....	166
8.3.1 CRC32 .....	166
8.3.2 CRC16 .....	166
8.4 CRC registers.....	167
8.4.1 CRC register overview.....	167
8.4.2 CRC32 data register (CRC_CRC32DAT).....	167
8.4.3 CRC32 independent data register (CRC_CRC32IDAT).....	167
8.4.4 CRC32 control register (CRC_CRC32CTRL).....	168
8.4.5 CRC16 control register (CRC_CRC16CTRL).....	168
8.4.6 CRC16 input data register (CRC_CRC16DAT) .....	169
8.4.7 CRC cyclic redundancy check code register (CRC_CRC16D) .....	169
8.4.8 LRC result register (CRC_LRC).....	170
<b>9 Advanced-control timers (TIM1 and TIM8) .....</b>	<b>171</b>
9.1 TIM1 and TIM8 introduction .....	171

9.2 Main features of TIM1 and TIM8.....	171
9.3 TIM1 and TIM8 function description .....	172
9.3.1 Time-base unit.....	172
9.3.2 Counter mode .....	173
9.3.3 Repetition counter .....	179
9.3.4 Clock selection.....	182
9.3.5 Capture/compare channels .....	185
9.3.6 Input capture mode .....	188
9.3.7 PWM input mode .....	189
9.3.8 Forced output mode .....	190
9.3.9 Output compare mode.....	191
9.3.10 PWM mode.....	192
9.3.11 One-pulse mode .....	195
9.3.12 Clearing the OCxREF signal on an external event .....	196
9.3.13 Complementary outputs with dead-time insertion .....	197
9.3.14 Break function.....	199
9.3.15 Debug mode.....	201
9.3.16 TIMx and external trigger synchronization .....	201
9.3.17 Timer synchronization .....	205
9.3.18 6-step PWM generation .....	205
9.3.19 Encoder interface mode .....	206
9.3.20 Interfacing with Hall sensor .....	209
9.4 TIMx register description(x=1, 8) .....	211
9.4.1 Register Overview .....	211
9.4.2 Control register 1 (TIMx_CTRL1) .....	212
9.4.3 Control register 2 (TIMx_CTRL2) .....	214
9.4.4 Slave mode control register (TIMx_SMCTRL).....	216
9.4.5 DMA/Interrupt enable registers (TIMx_DINTEN).....	219
9.4.6 Status registers (TIMx_STS).....	220
9.4.7 Event generation registers (TIMx_EVTGEN).....	222
9.4.8 Capture/compare mode register 1 (TIMx_CCMOD1) .....	223
9.4.9 Capture/compare mode register 2 (TIMx_CCMOD2) .....	227
9.4.10 Capture/compare enable registers (TIMx_CCEN).....	228
9.4.11 Counters (TIMx_CNT).....	231
9.4.12 Prescaler (TIMx_PSC) .....	231
9.4.13 Auto-reload register (TIMx_AR) .....	232
9.4.14 Repeat count registers (TIMx_REPCNT) .....	232
9.4.15 Capture/compare register 1 (TIMx_CCDAT1) .....	233
9.4.16 Capture/compare register 2 (TIMx_CCDAT2) .....	233
9.4.17 Capture/compare register 3 (TIMx_CCDAT3) .....	234
9.4.18 Capture/compare register 4 (TIMx_CCDAT4) .....	234
9.4.19 Break and Dead-time registers (TIMx_BKDT).....	235

9.4.20	DMA Control register (TIMx_DCTRL) .....	237
9.4.21	DMA transfer buffer register (TIMx_DADDR) .....	237
9.4.22	Capture/compare mode registers 3(TIMx_CCMOD3) .....	238
9.4.23	Capture/compare register 5 (TIMx_CC DAT5).....	239
9.4.24	Capture/compare register 6 (TIMx_CC DAT6).....	239
<b>10</b>	<b>General-purpose timers (TIM3) .....</b>	<b>240</b>
10.1	General-purpose timers introduction.....	240
10.2	Main features of General-purpose timers .....	240
10.3	General-purpose timers description .....	241
10.3.1	Time-base unit.....	241
10.3.2	Counter mode.....	242
10.3.3	Clock selection.....	247
10.3.4	Capture/compare channels .....	251
10.3.5	Input capture mode .....	254
10.3.6	PWM input mode .....	255
10.3.7	Forced output mode.....	256
10.3.8	Output compare mode .....	256
10.3.9	PWM mode.....	258
10.3.10	One-pulse mode .....	261
10.3.11	Clearing the OCxREF signal on an external event .....	262
10.3.12	Debug mode.....	263
10.3.13	TIMx and external trigger synchronization.....	263
10.3.14	Timer synchronization .....	263
10.3.15	Encoder interface mode .....	268
10.3.16	Interfacing with Hall sensor .....	270
10.4	TIMx register description(x=3) .....	270
10.4.1	Register Overview.....	270
10.4.2	Control register 1 (TIMx_CTRL1).....	272
10.4.3	Control register 2 (TIMx_CTRL2).....	274
10.4.4	Slave mode control register (TIMx_SMCTRL) .....	275
10.4.5	DMA/Interrupt enable registers (TIMx_DINTEN) .....	277
10.4.6	Status registers (TIMx_STS).....	278
10.4.7	Event generation registers (TIMx_EVTGEN).....	280
10.4.8	Capture/compare mode register 1 (TIMx_CCMOD1).....	281
10.4.9	Capture/compare mode register 2 (TIMx_CCMOD2) .....	284
10.4.10	Capture/compare enable registers (TIMx_CCEN).....	286
10.4.11	Counters (TIMx_CNT).....	287
10.4.12	Prescaler (TIMx_PSC) .....	288
10.4.13	Auto-reload register (TIMx_AR) .....	288
10.4.14	Capture/compare register 1 (TIMx_CC DAT1).....	288
10.4.15	Capture/compare register 2 (TIMx_CC DAT2).....	289
10.4.16	Capture/compare register 3 (TIMx_CC DAT3).....	289

10.4.17	Capture/compare register 4 (TIMx_CCDAT4).....	290
10.4.18	DMA Control register (TIMx_DCTRL) .....	290
10.4.19	DMA transfer buffer register (TIMx_DADDR) .....	291
<b>11 Basic timers (TIM6).....</b>		<b>293</b>
11.1	Basic timers introduction .....	293
11.2	Main features of Basic timers .....	293
11.3	Basic timers description .....	293
11.3.1	Time-base unit.....	293
11.3.2	Counter mode.....	294
11.3.3	Clock selection.....	297
11.3.4	Debug mode.....	297
11.4	TIMx register description(x=6) .....	297
11.4.1	Register overview .....	298
11.4.2	Control Register 1 (TIMx_CTRL1).....	298
11.4.3	DMA/Interrupt Enable Registers (TIMx_DINTEN) .....	299
11.4.4	Status Registers (TIMx_STS).....	300
11.4.5	Event Generation registers (TIMx_EVTGEN) .....	300
11.4.6	Counters (TIMx_CNT).....	301
11.4.7	Prescaler (TIMx_PSC).....	301
11.4.8	Automatic reload register (TIMx_AR) .....	302
<b>12 Low Power Timer (LPTIM).....</b>		<b>303</b>
12.1	Introduction .....	303
12.2	Main features .....	303
12.3	Block diagram .....	304
12.4	Function description .....	304
12.4.1	LPTIM clocks and on-off control .....	304
12.4.2	Prescaler.....	305
12.4.3	Glitch filter .....	305
12.4.4	Timer enable .....	306
12.4.5	Trigger multiplexer .....	306
12.4.6	Operating mode .....	307
12.4.7	Waveform generation.....	310
12.4.8	Register update .....	311
12.4.9	Counter mode.....	312
12.4.10	Encoder mode .....	312
12.4.11	Non-orthogonal encoder mode .....	314
12.4.12	Timeout function.....	315
12.4.13	LPTIM interrupts .....	316
12.5	LPTIM registers.....	316
12.5.1	LPTIM register overview.....	316
12.5.2	LPTIM interrupt and status register (LPTIM_INTSTS) .....	317

12.5.3	LPTIM interrupt clear register (LPTIM_INTCLR) .....	318
12.5.4	LPTIM interrupt enable register (LPTIM_INTEN).....	319
12.5.5	LPTIM configuration register (LPTIM_CFG).....	320
12.5.6	LPTIM control register (LPTIM_CTRL) .....	323
12.5.7	LPTIM compare register (LPTIM_COMP).....	323
12.5.8	LPTIM auto-reload register (LPTIM_ARR).....	324
12.5.9	LPTIM counter register (LPTIM_CNT).....	324
<b>13</b>	<b>Independent watchdog (IWDG) .....</b>	<b>326</b>
13.1	Introduction .....	326
13.2	Main features .....	326
13.3	Function description .....	327
13.3.1	Register access protection.....	327
13.3.2	Debug mode.....	328
13.4	User interface.....	328
13.4.1	Operate flow .....	328
13.5	IWDG registers.....	329
13.5.1	IWDG register overview.....	329
13.5.2	IWDG key register (IWDG_KEY) .....	329
13.5.3	IWDG pre-scaler register (IWDG_PREDIV) .....	330
13.5.4	IWDG reload register (IWDG_RELV) .....	330
13.5.5	IWDG status register (IWDG_STS) .....	331
<b>14</b>	<b>Window watchdog (WWDG) .....</b>	<b>332</b>
14.1	Introduction .....	332
14.2	Main features .....	332
14.3	Function description .....	332
14.4	Timing for refresh watchdog and interrupt generation .....	333
14.5	Debug mode.....	334
14.6	User interface.....	334
14.6.1	WWDG configuration flow .....	334
14.7	WWDG registers .....	335
14.7.1	WWDG register overview .....	335
14.7.2	WWDG control register (WWDG_CTRL).....	335
14.7.3	WWDG config register (WWDG_CFG) .....	335
14.7.4	WWDG status register (WWDG_STS) .....	336
<b>15</b>	<b>Analog to digital conversion (ADC) .....</b>	<b>337</b>
15.1	Introduction .....	337
15.2	Main features .....	337
15.3	Function Description.....	338
15.3.1	ADC clock .....	339
15.3.2	ADC switch control.....	340

15.3.3	Channel selection .....	340
15.3.4	Internal channel .....	341
15.3.5	Single conversion mode.....	341
15.3.6	Continuous conversion mode.....	341
15.3.7	Timing diagram.....	342
15.3.8	Analog watchdog .....	342
15.3.9	Scan mode .....	343
15.3.10	Injection channel management.....	343
15.3.11	Discontinuous mode .....	344
15.4	Data aligned .....	345
15.5	Programmable channel sampling time .....	345
15.6	Externally triggered conversion.....	346
15.7	DMA requests .....	347
15.8	Temperature sensor.....	347
15.8.1	Temperature sensor using flow .....	348
15.9	ADC interrupt .....	349
15.10	OPA channel control .....	349
15.11	ADC registers.....	351
15.11.1	ADC register overview .....	351
15.11.2	ADC status register (ADC_STS).....	352
15.11.3	ADC control register 1 (ADC_CTRL1) .....	353
15.11.4	ADC control register 2 (ADC_CTRL2) .....	355
15.11.5	ADC sampling time register 1 (ADC_SAMPT1).....	357
15.11.6	ADC sampling time register 2 (ADC_SAMPT2).....	358
15.11.7	ADC sampling time register 3 (ADC_SAMPT3).....	358
15.11.8	ADC injected channel data offset register x (ADC_JOFFSETx) (x=1...4).....	360
15.11.9	ADC watchdog high threshold register (ADC_WDGHIGH) .....	360
15.11.10	ADC watchdog low threshold register (ADC_WDGLOW) .....	360
15.11.11	ADC regular sequence register 1 (ADC_RSEQ1).....	361
15.11.12	ADC regular sequence register 2 (ADC_RSEQ2).....	362
15.11.13	ADC regular sequence register 3 (ADC_RSEQ3).....	362
15.11.14	ADC Injection sequence register (ADC_JSEQ).....	363
15.11.15	ADC injection data register x (ADC_JDATx) (x= 1...4) .....	364
15.11.16	ADC regulars data register (ADC_DAT) .....	364
15.11.17	ADC control register 3 (ADC_CTRL3) .....	365
15.11.18	ADC test register (ADC_TEST).....	366
15.11.19	ADC OPA control register (ADC_OPACTRL) .....	366
<b>16</b>	<b>Comparator (COMP) .....</b>	<b>368</b>
16.1	COMP system connection block diagram.....	368
16.2	COMP features .....	368
16.3	COMP configuration process.....	369
16.4	COMP working mode.....	369

16.4.1	Independent comparator .....	369
16.5	Comparator interconnection .....	370
16.6	Interrupt .....	370
16.7	COMP register .....	371
16.7.1	COMP register overview .....	371
16.7.2	COMP interrupt Enable register (COMP_INTEN).....	372
16.7.3	COMP interrupt register (COMP_INTSTS).....	372
16.7.4	COMP lock register(COMP_LOCK).....	372
16.7.5	COMP control register (COMP_CTRL).....	373
16.7.6	COMP Filter control register (COMP_FILC) .....	375
16.7.7	COMP Filter Frequency Division register (COMP_FILP) .....	375
16.7.8	COMP reference input compare voltage register (COMP_INVREF).....	376
<b>17</b>	<b>I2C interface.....</b>	<b>377</b>
17.1	Introduction .....	377
17.2	Main features .....	377
17.3	Function description .....	377
17.3.1	SDA and SCL line control .....	378
17.3.2	Software communication process .....	378
17.3.3	Error conditions description.....	388
17.3.4	DMA application .....	389
17.3.5	Packet error check.....	390
17.3.6	SMBus .....	391
17.4	Debug mode.....	393
17.5	Interrupt request.....	393
17.6	I2C registers.....	394
17.6.1	I2C register overview .....	394
17.6.2	I2C Control register 1 (I2C_CTRL1) .....	395
17.6.3	I2C Control register 2 (I2C_CTRL2) .....	397
17.6.4	I2C Own address register 1 (I2C_OADDR1).....	398
17.6.5	I2C Own address register 2 (I2C_OADDR2).....	399
17.6.6	I2C Data register (I2C_DAT) .....	399
17.6.7	I2C Status register 1 (I2C_STS1) .....	400
17.6.8	I2C Status register 2 (I2C_STS2) .....	403
17.6.9	I2C Clock control register (I2C_CLKCTRL).....	404
17.6.10	I2C Rise time register (I2C_TMRISE) .....	405
<b>18</b>	<b>Universal synchronous asynchronous receiver transmitter (USART) .....</b>	<b>407</b>
18.1	Introduction .....	407
18.2	Main features .....	407
18.3	Functional block diagram .....	408
18.4	Function description .....	408
18.4.1	USART frame format.....	409

18.4.2	Transmitter.....	410
18.4.3	Receiver .....	413
18.4.4	Generation of fractional baud rate .....	417
18.4.5	Receiver's tolerance clock deviation .....	418
18.4.6	Parity control .....	419
18.4.7	DMA application .....	420
18.4.8	Hardware flow control .....	422
18.4.9	Multiprocessor communication .....	424
18.4.10	Synchronous mode.....	426
18.4.11	Single-wire half-duplex mode .....	430
18.4.12	IrDA SIR ENDEC mode.....	430
18.4.13	LIN mode.....	432
18.4.14	Smartcard mode (ISO7816) .....	434
18.5	Interrupt request.....	436
18.6	Mode support.....	437
18.7	USART register .....	437
18.7.1	USART register overview.....	437
18.7.2	USART Status register (USART_STS).....	438
18.7.3	USART Data register (USART_DAT) .....	440
18.7.4	USART Baud rate register (USART_BRCF).....	441
18.7.5	USART control register 1 register (USART_CTRL1) .....	441
18.7.6	USART control register 2 register (USART_CTRL2) .....	443
18.7.7	USART control register 3 register (USART_CTRL3) .....	444
18.7.8	USART guard time and prescaler register (USART_GTP).....	446
<b>19</b>	<b>Low power universal asynchronous receiver transmitter (LPUART).....</b>	<b>448</b>
19.1	Introduction .....	448
19.2	Main features .....	448
19.3	Functional block diagram .....	449
19.4	Function description .....	449
19.4.1	LPUART frame format .....	450
19.4.2	Transmitter.....	450
19.4.3	Receiver .....	452
19.4.4	Fractional baud rate generation.....	454
19.4.5	Parity control .....	456
19.4.6	DMA application .....	456
19.4.7	Hardware flow control .....	458
19.4.8	Low power wake up.....	460
19.5	Interrupt request.....	460
19.6	LPUART registers .....	461
19.6.1	LPUART register overview .....	461
19.6.2	LPUART status register (LPUART_STS) .....	461
19.6.3	LPUART interrupt enable register (LPUART_INTEN) .....	462

19.6.4	LPUART control register (LPUART_CTRL).....	463
19.6.5	LPUART baud rate configuration register 1 (LPUART_BRCFG1) .....	464
19.6.6	LPUART data register (LPUART_DAT).....	465
19.6.7	LPUART baud rate configuration register 2 (LPUART_BRCFG2) .....	465
19.6.8	LPUART wake up data register (LPUART_WUDAT).....	466
<b>20</b>	<b>Serial peripheral interface/Inter-IC Sound (SPI/ I2S).....</b>	<b>467</b>
20.1	SPI introduction .....	467
20.2	SPI and I2S main features.....	467
20.2.1	SPI features.....	467
20.2.2	I2S features.....	467
20.3	SPI function description .....	468
20.3.1	General description.....	468
20.3.2	SPI work mode .....	471
20.3.3	Status flag .....	477
20.3.4	Turn off the SPI .....	478
20.3.5	SPI communication using DMA.....	479
20.3.6	CRC calculation.....	480
20.3.7	Error flag.....	481
20.3.8	SPI interrupt.....	482
20.4	I2S function description.....	483
20.4.1	Supported audio protocols .....	484
20.4.2	Clock generator.....	491
20.4.3	I2S send and receive sequence.....	492
20.4.4	Status flag .....	494
20.4.5	Error flag.....	495
20.4.6	I2S interrupt.....	495
20.4.7	DMA function.....	496
20.5	SPI and I2S register description .....	496
20.5.1	SPI register overview.....	496
20.5.2	SPI control register 1 (SPI_CTRL1) (not used in I2S mode) .....	496
20.5.3	SPI control register 2 (SPI_CTRL2).....	499
20.5.4	SPI status register (SPI_STS) .....	500
20.5.5	SPI data register (SPI_DAT).....	501
20.5.6	SPI CRC polynomial register (SPI_CRCPOLY) (not used in I2S mode) .....	501
20.5.7	SPI RX CRC register (SPI_CRCRDAT) (not used in I2S mode) .....	502
20.5.8	SPI TX CRC register (SPI_CRCTDAT) .....	502
20.5.9	SPI_I2S configuration register (SPI_I2SCFG) .....	503
20.5.10	SPI_I2S prescaler register (SPI_I2SPREDIV) .....	504
<b>21</b>	<b>Real-time clock (RTC).....</b>	<b>506</b>
21.1	Description .....	506
21.2	Specification .....	506

21.3 RTC function description.....	508
21.3.1    RTC block diagram.....	508
21.3.2    GPIOs of RTC.....	509
21.3.3    RTC register write protection .....	509
21.3.4    RTC clock and prescaler.....	509
21.3.5    RTC calendar .....	510
21.3.6    Calendar initialization and configuration.....	510
21.3.7    Calendar reading .....	511
21.3.8    Calibration clock output.....	512
21.3.9    Programmable alarms .....	512
21.3.10   Alarm configuration.....	512
21.3.11   Alarm output.....	512
21.3.12   Periodic automatic wakeup.....	513
21.3.13   Wakeup timer configuration .....	513
21.3.14   Timestamp function .....	513
21.3.15   Tamper detection .....	514
21.3.16   Daylight saving time configuration .....	515
21.3.17   RTC sub-second register shift.....	515
21.3.18   RTC digital clock precision calibration .....	515
21.3.19   RTC low power mode .....	516
21.4 RTC Registers.....	517
21.4.1    RTC register overview .....	517
21.4.2    RTC Calendar Time Register (RTC_TSH) .....	518
21.4.3    RTC Calendar Date Register (RTC_DATE) .....	519
21.4.4    RTC Control Register (RTC_CTRL) .....	519
21.4.5    RTC Initial Status Register (RTC_INITSTS) .....	522
21.4.6    RTC Prescaler Register (RTC_PRE) .....	524
21.4.7    RTC Wakeup Timer Register (RTC_WKUPT).....	524
21.4.8    RTC Alarm A Register (RTC_ALARMA).....	525
21.4.9    RTC Alarm B Register (RTC_ALARMB).....	526
21.4.10   RTC Write Protection register (RTC_WRP).....	527
21.4.11   RTC Sub-second Register (RTC_SUBS).....	527
21.4.12   RTC Shift Control Register (RTC_SCTRL).....	528
21.4.13   RTC Timestamp Time Register (RTC_TST) .....	528
21.4.14   RTC Timestamp Date Register (RTC_TSD).....	529
21.4.15   RTC Timestamp Sub-second Register (RTC_TSSS) .....	530
21.4.16   RTC Calibration Register (RTC_CALIB).....	530
21.4.17   RTC Tamper Configuration Register (RTC_TMPCFG) .....	531
21.4.18   RTC Alarm A sub-second register (RTC_ALRMASS).....	533
21.4.19   RTC Alarm B sub-second register (RTC_ALRMBSS).....	534
<b>22 Operational amplifier (OPAMP) .....</b>	<b>536</b>
22.1 OPAMP features .....	536

22.1.1	OPAMP function description .....	536
22.2	OPAMP working mode.....	537
22.2.1	OPAMP standalone operational amplifier mode .....	537
22.2.2	OPAMP follow mode.....	538
22.2.3	OPAMP internal programmable gain (PGA) mode .....	539
22.2.4	OPAMP independent write protection .....	540
22.2.5	OPAMP TIMER controls the switching mode.....	540
22.3	OPAMP registers .....	541
22.3.1	OPAMP register overview .....	541
22.3.2	OPAMP Control Status Register (OPAMP_CS) .....	541
22.3.3	OPAMP Lock Register (OPAMP_LOCK).....	542
<b>23</b>	<b>Beeper .....</b>	<b>544</b>
23.1	Introduction .....	544
23.2	Function description .....	544
23.3	Beeper registers .....	544
23.3.1	Beeper register overview .....	544
23.3.2	Beeper control register (BEEPER_CTRL) .....	544
<b>24</b>	<b>Arithmetic units (HDIV and SQRT).....</b>	<b>547</b>
24.1	Introduction to HDIV and SQRT .....	547
24.2	HDIV and SQRT function description.....	547
24.3	HDIV registers.....	547
24.3.1	HDIV register overview.....	547
24.3.2	HDIV control status register (HDIV_CTRLSTS).....	548
24.3.3	HDIV dividend register (HDIV_DIVIDEND).....	548
24.3.4	HDIV divisor register (HDIV_DIVISOR).....	549
24.3.5	HDIV quotient register (HDIV_QUOTIENT).....	549
24.3.6	HDIV remainder register (HDIV_REMAINDER) .....	549
24.3.7	HDIV divide by zero register (HDIV_DIVBY0).....	550
24.4	SQRT registers.....	550
24.4.1	SQRT register overview.....	550
24.4.2	SQRT control status register (SQRT_CTRLSTS).....	551
24.4.3	SQRT Radicand register (SQRT_RADICAND) .....	551
24.4.4	SQRT square root register (SQRT_ROOT) .....	552
<b>25</b>	<b>Debug support (DBG).....</b>	<b>553</b>
25.1	Overview .....	553
25.2	SWD function .....	554
25.2.1	Pin assignment .....	554
<b>26</b>	<b>Unique device serial number (UID).....</b>	<b>555</b>
26.1	Introduction .....	555

26.2 UID register .....	555
26.3 UCID register .....	555
26.4 DBGMCU_ID register .....	555
<b>27 Version history .....</b>	<b>557</b>
<b>28 Notice .....</b>	<b>558</b>

## List of Table

Table 2-1 List of peripheral register addresses .....	31
Table 2-2 List of boot mode.....	33
Table 2-3 Flash bus address list .....	34
Table 2-4 Option byte list .....	38
Table 2-5 Read protection configuration list .....	39
Table 2-6 Flash read-write-erase <sup>(1)</sup> permission control table .....	41
Table 2-7 FLASH register overview .....	46
Table 3-1 Power modes.....	56
Table 3-2 Peripheral running status .....	57
Table 3-3 PWR register overview.....	61
Table 4-1 RCC register overview .....	79
Table 5-1 I/O port configuration table .....	105
Table 5-2 I/O List of functional features of the pin .....	106
Table 5-3 I/O List of functional features of the pin .....	111
Table 5-4 TIM1 alternate function I/O remapping.....	111
Table 5-5 TIM8 alternate function I/O remapping.....	112
Table 5-6 TIM3 alternate function I/O remapping.....	112
Table 5-7 LPTIM alternate function I/O remapping.....	113
Table 5-8 USART1 alternate function I/O remapping .....	113
Table 5-9 USART2 alternate function I/O remapping .....	114
Table 5-10 LPUART alternate function I/O remapping.....	114
Table 5-11 I2C1 alternate function I/O remapping .....	115
Table 5-12 I2C2 alternate function I/O remapping .....	115
Table 5-13 SPI1/I2S alternate function I/O remapping .....	116
Table 5-14 SPI2 alternate function I/O remapping .....	117
Table 5-15 COMP alternate function I/O remapping.....	117
Table 5-16 BEEPER alternate function I/O remapping .....	117
Table 5-17 EVENTOUT alternate function I/O remapping.....	117
Table 5-18 RTC alternate function I/O remapping .....	118

Table 5-19 RCC alternate function I/O remapping .....	118
Table 5-20 OSC_IN/OSC_OUT alternate function I/O remapping .....	118
Table 5-21 OSC32 alternate function remapping .....	119
Table 5-22 OSC alternate function remapping .....	119
Table 5-23 ADC external trigger injection conversion alternate function remapping .....	119
Table 5-24 ADC external trigger regular conversion alternate function remapping .....	120
Table 5-25 ADC .....	120
Table 5-26 PVD .....	120
Table 5-27 TIM1/TIM8 .....	120
Table 5-28 TIM3 and LPTIM .....	120
Table 5-29 USART .....	120
Table 5-30 LPUART .....	121
Table 5-31 I2C .....	121
Table 5-32 SPI .....	121
Table 5-33 COMP .....	121
Table 5-34 BEEPER .....	122
Table 5-35 Other .....	122
Table 5-36 GPIO register overview .....	123
Table 5-37 AFIO register overview .....	132
Table 6-1 Vector table .....	137
Table 6-2 EXTI register overview .....	145
Table 7-1 Programmable data width and endian operation (when PINC = MINC = 1) .....	152
Table 7-2 Flow control table .....	155
Table 7-3 DMA interrupt request .....	156
Table 7-4 DMA request mapping .....	157
Table 7-5 DMA register overview .....	157
Table 8-1 CRC register overview .....	167
Table 9-1 Counting direction versus encoder signals .....	207
Table 9-2 Register overview .....	211
Table 9-3 TIMx internal trigger connection .....	218
Table 9-4 Output control bits of complementary OCx and OCxN channels with break function .....	230

Table 10-1 Counting direction versus encoder signals .....	268
Table 10-2 Register overview .....	270
Table 10-3 TIMx internal trigger connection.....	277
Table 10-4 Output control bits of standard OCx channel .....	287
Table 11-1 Register overview .....	298
Table 12-1 Pre-scaler division ratios.....	305
Table 12-2 6 trigger inputs corresponding to LPTIM_CFG.TRGSEL[2:0] bits.....	307
Table 13-1 IWDG counting maximum and minimum reset time .....	329
Table 13-2 IWDG register overview.....	329
Table 14-1 Maximum and minimum counting time of WWDG.....	334
Table 14-2 WWDG register overview .....	335
Table 15-1 ADC pins .....	338
Table 15-2 Analog watchdog channel selection.....	343
Table 15-3 Right-align data .....	345
Table 15-4 Left-align data.....	345
Table 15-5 ADC is used for external triggering of regular channels .....	346
Table 15-6 ADC is used for external triggering of injection channels.....	346
Table 15-7 ADC interrupt .....	349
Table 15-8 OPA channel selection .....	350
Table 15-9 ADC register overview .....	351
Table 16-1 COMP register overview .....	371
Table 17-1 Comparison between SMBus and I2C.....	391
Table 17-2 I <sup>2</sup> C interrupt request.....	393
Table 17-3 I2C register overview .....	394
Table 18-1 Stop bit configuration .....	411
Table 18-2 Data sampling for noise detection .....	416
Table 18-3 Error calculation when setting baud rate .....	418
Table 18-4 when DIV_Decimal = 0. Tolerance of USART receiver .....	419
Table 18-5 when DIV_Decimal != 0. Tolerance of USART receiver .....	419
Table 18-6 Frame format .....	419
Table 18-7 USART interrupt request.....	436

Table 18-8 USART mode setting <sup>(1)</sup> .....	437
Table 18-9 USART register overview.....	437
Table 19-1 Data sampling for noise detection .....	454
Table 19-2 Parity frame format.....	456
Table 19-3 LPUART interrupt requests .....	460
Table 19-4 LPUART register overview .....	461
Table 20-1 SPI interrupt request .....	482
Table 20-2 Use the standard 8MHz HSE clock to get accurate audio frequency.....	492
Table 20-3 I <sup>2</sup> S interrupt request.....	495
Table 20-4 SPI register overview.....	496
Table 21-1 RTC feature support.....	506
Table 21-2 RTC register overview.....	517
Table 22-1 OPAMP register overview .....	541
Table 23-1 Beeper register overview .....	544
Table 24-1 HDIV register overview .....	547
Table 24-2 SQRT register overview .....	550
Table 26-1 DBGMCU_ID bit description.....	555

## List of Figure

Figure 2-1 Bus architecture .....	29
Figure 2-2 Bus address map .....	30
Figure 3-1 Power supply block diagram.....	54
Figure 3-2 Power on reset/power down reset waveform .....	55
Figure 3-3 PVD threshold diagram.....	56
Figure 4-1 System reset generation .....	72
Figure 4-2 Clock Tree.....	74
Figure 4-3 HSE clock source .....	75
Figure 4-4 PLL clock configuration .....	77
Figure 5-1 Basic structure of I/O ports .....	105
Figure 5-2 Input floating / pull-up / pull-down configuration mode. ....	107
Figure 5-3 Output mode.....	108
Figure 5-4 Alternate function mode.....	109
Figure 5-5 Analog function mode with high impedance.....	109
Figure 6-1 Extenal interrupt/event controller block diagram.....	141
Figure 6-2 External interrupt generic I/O mapping .....	143
Figure 7-1 DMA block diagram.....	151
Figure 8-1 CRC calculation unit block diagram.....	166
Figure 9-1 Block diagram of TIM1 and TIM8 .....	172
Figure 9-2 Counter timing diagram with prescaler division change from 1 to 4.....	173
Figure 9-3 Timing diagram of up-counting. The internal clock divider factor = 2/N .....	175
Figure 9-4 Timing diagram of the up-counting, update event when ARPEN=0/1.....	176
Figure 9-5 Timing diagram of the down-counting, internal clock divided factor = 2/N.....	177
Figure 9-6 Timing diagram of the Center-aligned, internal clock divided factor =2/N .....	178
Figure 9-7 A center-aligned sequence diagram that includes counter overflows and underflows (ARPEN = 1) .....	179
Figure 9-8 Repeat count sequence diagram in down-counting mode .....	180
Figure 9-9 Repeat count sequence diagram in up-counting mode.....	181
Figure 9-10 Repeat count sequence diagram in center-aligned mode .....	181

Figure 9-11 Control circuit in normal mode, internal clock divided by 1.....	182
Figure 9-12 TI2 external clock connection example .....	183
Figure 9-13 Control circuit in external clock mode 1 .....	184
Figure 9-14 External trigger input block diagram .....	184
Figure 9-15 Control circuit in external clock mode 2.....	185
Figure 9-16 Capture/compare channel (example: channel 1 input stage).....	186
Figure 9-17 Capture/compare channel 1 main circuit.....	187
Figure 9-18 Output part of channelx (x= 1,2,3, take channel 1 as example).....	187
Figure 9-19 Output part of channelx (x= 4).....	188
Figure 9-20 PWM input mode timing.....	190
Figure 9-21 Output compare mode, toggle on OC1 .....	192
Figure 9-22 Center-aligned PWM waveform (AR=8).....	193
Figure 9-23 Edge-aligned PWM waveform (APR=8).....	194
Figure 9-24 Example of One-pulse mode.....	195
Figure 9-25 Clearing the OCxREF of TIMx.....	197
Figure 9-26 Complementary output with dead-time insertion.....	198
Figure 9-27 Output behavior in response to a break.....	201
Figure 9-28 Control circuit in reset mode.....	202
Figure 9-29 Control circuit in Trigger mode .....	203
Figure 9-30 Control circuit in Gated mode.....	204
Figure 9-31 Control circuit in Trigger Mode + External Clock Mode2.....	205
Figure 9-32 6-step PWM generation, COM example (OSSR=1) .....	206
Figure 9-33 Example of counter operation in encoder interface mode.....	207
Figure 9-34 Encoder interface mode example with IC1FP1 polarity inverted .....	208
Figure 9-35 Example of Hall sensor interface .....	210
Figure 10-1 Block diagram of TIMx (x=3) .....	241
Figure 10-2 Counter timing diagram with prescaler division change from 1 to 4.....	242
Figure 10-3 Timing diagram of up-counting. The internal clock divider factor = 2/N .....	243
Figure 10-4 Timing diagram of the up-counting, update event when ARDEN=0/1.....	244
Figure 10-5 Timing diagram of the down-counting, internal clock divided factor = 2/N.....	245
Figure 10-6 Timing diagram of the Center-aligned, internal clock divided factor =2/N .....	246

Figure 10-7 A center-aligned sequence diagram that includes counter overflows and underflows (ARPEN = 1)	247
Figure 10-8 Control circuit in normal mode, internal clock divided by 1	248
Figure 10-9 TI2 external clock connection example	249
Figure 10-10 Control circuit in external clock mode 1	250
Figure 10-11 External trigger input block diagram	250
Figure 10-12 Control circuit in external clock mode 2	251
Figure 10-13 Capture/compare channel (example: channel 1 input stage)	252
Figure 10-14 Capture/compare channel 1 main circuit	253
Figure 10-15 Output part of channelx (x = 1,2,3,4;take channel 4 as an example)	254
Figure 10-16 PWM input mode timing	256
Figure 10-17 Output compare mode, toggle on OC1	258
Figure 10-18 Center-aligned PWM waveform (AR=8)	259
Figure 10-19 Edge-aligned PWM waveform (APR=8)	260
Figure 10-20 Example of One-pulse mode	261
Figure 10-21 Control circuit in reset mode	263
Figure 10-22 Block diagram of timer interconnection	264
Figure 10-23 TIM3 gated by OC1REF of TIM1	265
Figure 10-24 TIM3 gated by enable signal of TIM1	266
Figure 10-25 Trigger TIM3 with an update of TIM1	267
Figure 10-26 Triggers timers 1 and 3 using the TI1 input of TIM1	268
Figure 10-27 Example of counter operation in encoder interface mode	269
Figure 11-1 Block diagram of TIMx (x = 6)	293
Figure 11-2 Counter timing diagram with prescaler division change from 1 to 4	294
Figure 11-3 Timing diagram of up-counting. The internal clock divider factor = 2/N	295
Figure 11-4 Timing diagram of the up-counting, update event when ARPEN=0/1	296
Figure 11-5 Control circuit in normal mode, internal clock divided by 1	297
Figure 12-1 LPTIM Diagram	304
Figure 12-2 Glitch filter timing diagram	306
Figure 12-3 LPTIM output waveform, Continuous counting mode configuration	308
Figure 12-4 PTIM output waveform, single counting mode configuration	309

Figure 12-5 LPTIM output waveform, Single counting mode configuration and Set-once mode activated ...	309
Figure 12-6 Waveform generation .....	311
Figure 12-7 Encoder mode counting sequence .....	314
Figure 12-8 Input waveforms of Input1 and Input2 when the decoder module is working normally .....	315
Figure 12-9 Input1 and Input2 input waveforms when decoder module is not working .....	315
Figure 13-1 Functional block diagram of the independent watchdog module.....	327
Figure 14-1 Watchdog block diagram.....	332
Figure 14-2 Refresh window and interrupt timing of WWDG .....	333
Figure 15-1 Block diagram of a single ADC .....	338
Figure 15-2 ADC clock.....	340
Figure 15-3 Timing diagram.....	342
Figure 15-4 Injection conversion delay .....	344
Figure 15-5 Temperature sensor and VREFINT Diagram of the channel.....	348
Figure 15-6 TIM1 CC4 triggers OPA channel switching ADC injection sampling.....	349
Figure 16-1 Comparator System Connection Diagram.....	368
Figure 17-1 I2C functional block diagram.....	379
Figure 17-2 I2C bus protocol.....	379
Figure 17-3 Slave transmitter transfer sequence diagram.....	382
Figure 17-4 Slave receiver transfer sequence diagram .....	383
Figure 17-5 Master transmitter transfer sequence diagram .....	385
Figure 17-6 Master receiver transfer sequence diagram.....	387
Figure 18-1 USART block diagram.....	408
Figure 18-2 word length = 8 setting.....	410
Figure 18-3 word length = 9 setting.....	410
Figure 18-4 configuration stop bit .....	411
Figure 18-5 TXC/TXDE changes during transmission.....	413
Figure 18-6 Start bit detection .....	414
Figure 18-7 Transmission using DMA .....	421
Figure 18-8 Reception using DMA .....	422
Figure 18-9 hardware flow control between two USART .....	423
Figure 18-10 RTS flow control.....	423

Figure 18-11 CTS flow controls .....	424
Figure 18-12 Mute mode using idle line detection .....	425
Figure 18-13 Mute mode detected using address mark .....	426
Figure 18-14 USART synchronous transmission example .....	427
Figure 18-15 USART data clock timing example (WL=0).....	428
Figure 18-16 USART data clock timing example (WL=1).....	429
Figure 18-17 RX data sampling / holding time .....	429
Figure 18-18 IrDASIRENDEC-Block diagram.....	431
Figure 18-19 IrDA data Modulation (3/16)-normal mode .....	432
Figure 18-20 Break detection in LIN mode (11-bit break length-the LINBSDL bit is set) .....	433
Figure 18-21 Break detection and framing error detection in LIN mode .....	434
Figure 18-22 ISO7816-3 Asynchronous Protocol.....	435
Figure 18-23 Use 1.5 stop bits to detect parity errors.....	436
Figure 19-1 LPUART block diagram .....	449
Figure 19-2 frame format.....	450
Figure 19-3 TXC changes during transmission .....	452
Figure 19-4 Data sampling for noise detection.....	454
Figure 19-5 Sending using DMA.....	457
Figure 19-6 Receiving with DMA .....	458
Figure 19-7 Hardware flow control between two LPUART .....	458
Figure 19-8 RTS flow control.....	459
Figure 19-9 CTS flow control.....	460
Figure 20-1 SPI block diagram.....	468
Figure 20-2 Selective management of hardware/software.....	469
Figure 20-3 Master and slave applications .....	470
Figure 20-4 Data clock timing diagram.....	471
Figure 20-5 Schematic diagram of the change of TE/RNE/BUSY when the host is continuously transmitting in full duplex mode.....	472
Figure 20-6 Schematic diagram of TE/BUSY change when the host transmits continuously in one-way only mode .....	473
Figure 20-7 Schematic diagram of RNE change when continuous transmission occurs in receive-only mode (BIDIRMODE = 0 and RONLY = 1).....	474

Figure 20-8 Schematic diagram of the change of TE/RNE/BUSY when the slave is continuously transmitting in full duplex mode.....	475
Figure 20-9 Schematic diagram of TE/BUSY change during continuous transmission in slave unidirectional transmit-only mode.....	475
Figure 20-10 Schematic diagram of TE/BUSY change when BIDIRMODE = 0 and RONLY = 0 are transmitted discontinuously .....	477
Figure 20-11 Transmission using DMA .....	480
Figure 20-12 Reception using DMA .....	480
Figure 20-13 I <sup>2</sup> S block diagram.....	483
Figure 20-14 I <sup>2</sup> S Philips protocol waveform (16/32-bit full precision, CLKPOL = 0) .....	485
Figure 20-15 I <sup>2</sup> S Philips protocol standard waveform (24-bit frame, CLKPOL = 0).....	485
Figure 20-16 I <sup>2</sup> S Philips protocol standard waveform (16-bit extended to 32-bit packet frame, CLKPOL = 0) .....	486
Figure 20-17 The MSB is aligned with 16-bit or 32-bit full precision, CLKPOL = 0.....	487
Figure 20-18 MSB aligns 24-bit data, CLKPOL = 0.....	487
Figure 20-19 MSB-aligned 16-bit data is extended to 32-bit packet frame, CLKPOL = 0 .....	488
Figure 20-20 LSB alignment 16-bit or 32-bit full precision, CLKPOL = 0 .....	488
Figure 20-21 LSB aligns 24-bit data, CLKPOL = 0 .....	489
Figure 20-22 LSB aligned 16-bit data is extended to 32-bit packet frame, CLKPOL = 0.....	489
Figure 20-23 PCM standard waveform (16 bits) .....	490
Figure 20-24 PCM standard waveform (16-bit extended to 32-bit packet frame).....	490
Figure 20-25 I <sup>2</sup> S clock generator structure .....	491
Figure 20-26 Audio sampling frequency definition.....	491
Figure 21-1 RTC block diagram.....	508
Figure 22-1 Block diagram of OPAMP connection diagram .....	536
Figure 22-2 OPAMP Standalone Operational Amplifier Mode .....	538
Figure 22-3 OPAMP follow mode .....	539
Figure 22-4 Internal programmable gain mode .....	540

# 1 Abbreviations in the text

## 1.1 List of abbreviations for registers

The following abbreviations are used in register descriptions:

read/write(rw)	Software can read and write these bits.
read-only(r)	Software can only read these bits.
write-only(w)	Software can only write this bit, and reading this bit will return the reset value.
read/clear(rc_w1)	Software can read this bit or clear it by writing' 1', and writing' 0' has no effect on this bit.
read/clear(rc_w0)	Software can read this bit or clear it by writing' 0', and writing' 1' has no effect on this bit.
read/clear by read(rc_r)	Software can read this bit. Reading this bit will automatically clear it to' 0'. Writing' 0' has no effect on this bit.
read/set(rs)	Software can read or set this bit. Writing' 0' has no effect on this bit.
read-only write trigger(rt_w)	Software can read this bit and write' 0' or' 1' to trigger an event, but it has no effect on this bit value.
toggle(t)	Software can only flip this bit by writing' 1', and writing' 0' has no effect on this bit.
Reserved(Res.)	Reserved bit, must be kept at reset value.

## 1.2 Available peripherals

For all models of N32G031 microcontroller series, the existence and number of a peripheral, please refer to the data sheet of the corresponding model.

## 2 Memory and bus architecture

### 2.1 System architecture

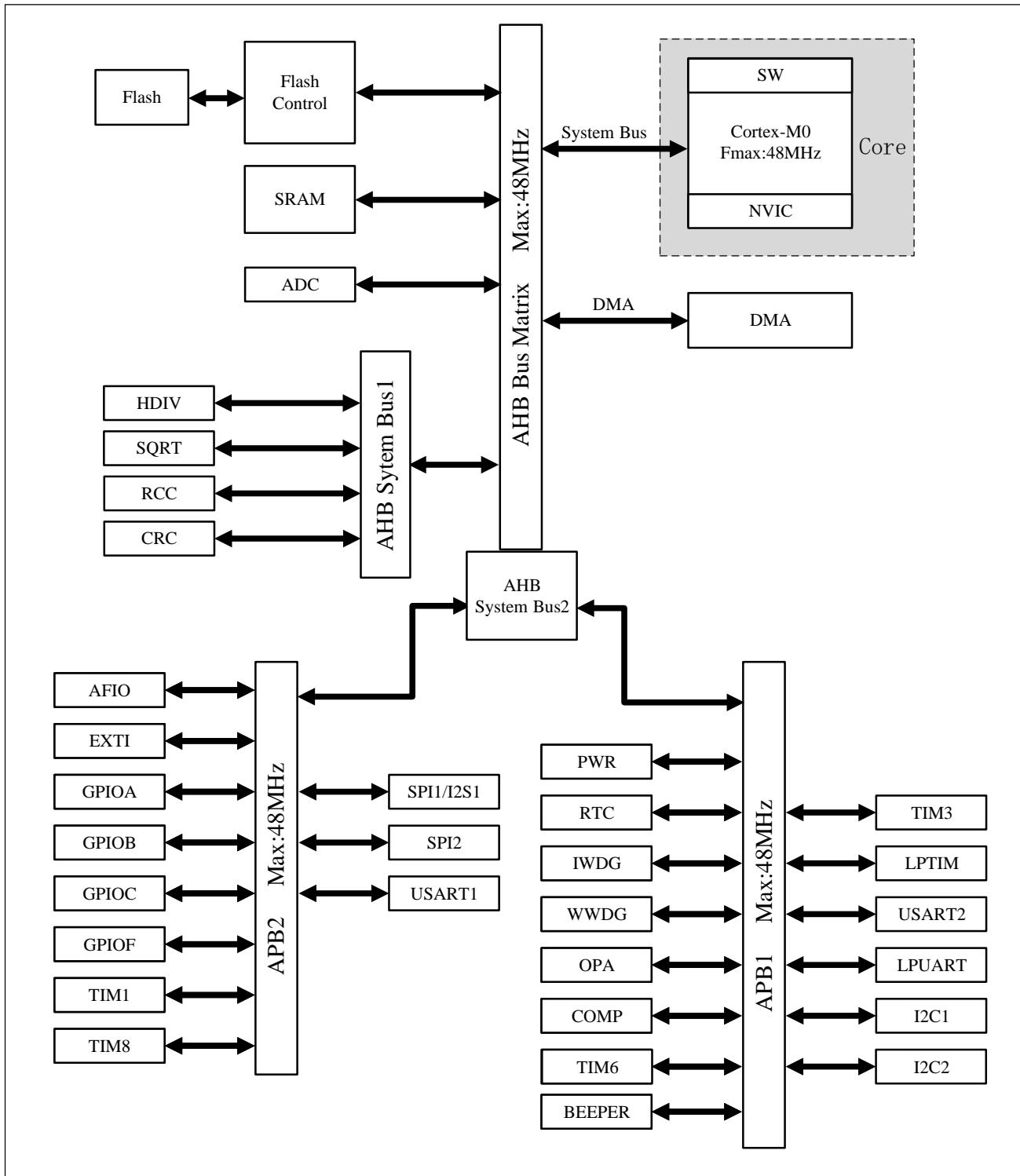
#### 2.1.1 Bus architecture

The main system consists of the following parts:

- Two main drive units:
  - Cortex®-M0 core system bus
  - General purpose DMA
- Six passive units
  - Internal SRAM
  - Internal flash memory
  - ADC
  - AHB to AHB bridge, it connects some AHB devices
  - AHB to APB bridge (AHB2APBx,x=1,2), which connects all APB devices

These are connected to each other through a multi-level AHB bus architecture, as shown in Figure 2-1:

Figure 2-1 Bus architecture



- CPU System bus: Connects to the Cortex®-M0 kernel Sbus to bus matrix, used for instruction pre-fetch, data load (constant load and debug access) and AHB/APB peripheral access.
- DMA bus: The DMA's AHB master interface is connected to the bus matrix, which coordinates the kernel and

DMA access to SRAM, flash and peripherals.

- The bus matrix coordinates access arbitration between the kernel system bus and the DMA master bus. The arbitration uses a Round Robin algorithm. The bus matrix consists of two driver components (CPU system bus, DMA bus) and six slave components (flash memory interface, SRAM, ADC, and AHB system bus 1/2). Some AHB peripherals are connected to system bus 1 through a bus matrix, and system bus 2 is connected to two AHB2APB Bridges.
- The system consists of two AHB2APB Bridges, i.e. AHB2APB1 and AHB2APB2. APB1 contains 14 APB peripherals and the maximum speed of PCLK is 48MHz. APB2 contains 11 APB peripherals with a maximum PCLK speed equal to 48MHz.

## 2.1.2 Bus address mapping

The address mapping includes all AHB and APB peripherals: AHB peripherals, APB1 peripherals, APB2 peripherals, Flash, SRAM, System Memory, etc. The specific mapping is as follows:

Figure 2-2 Bus address map

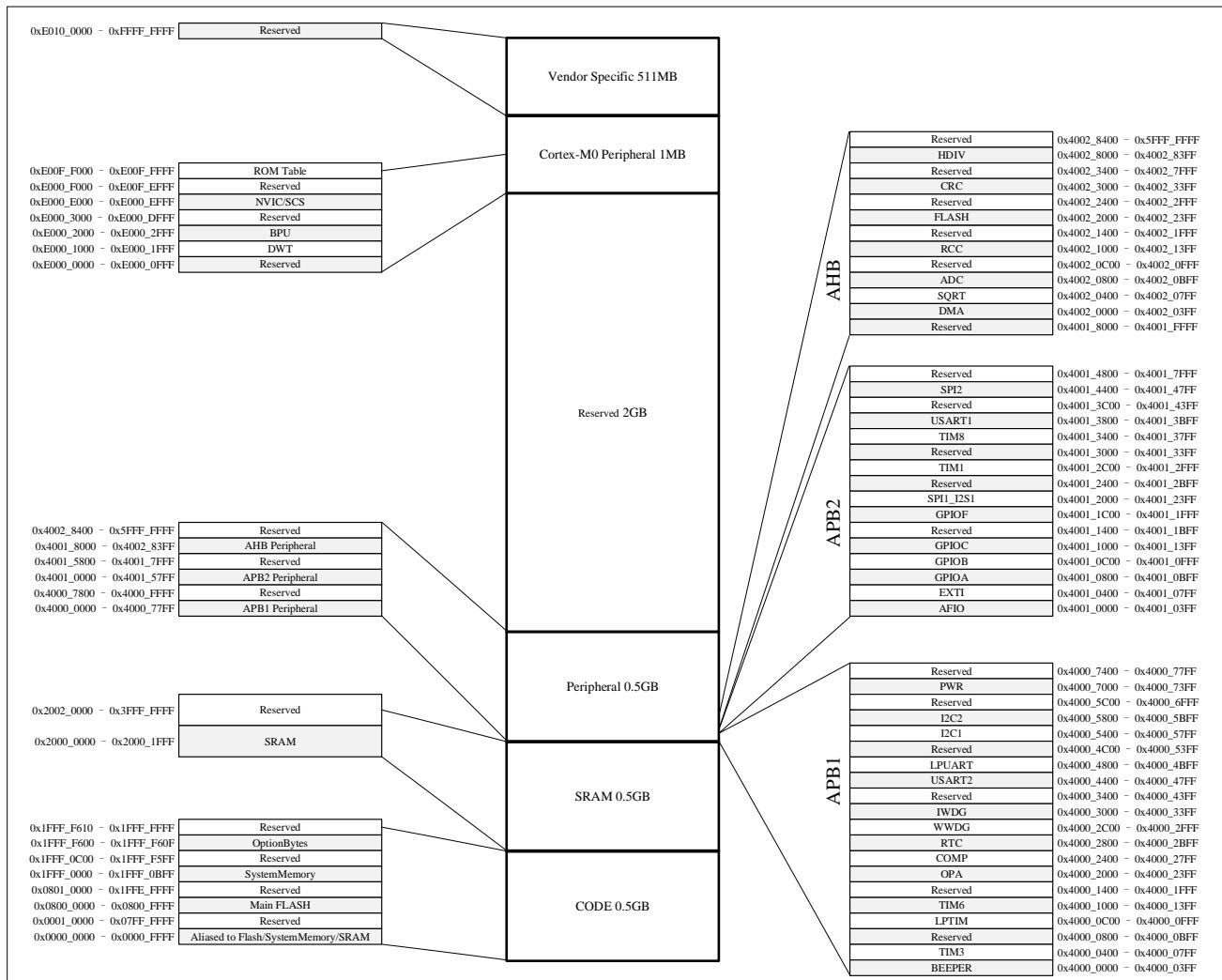


Table 2-1 List of peripheral register addresses

Address range	Peripherals	Bus
0x4002_8400 – 0x5FFF_FFFF	Reserved	AHB
0x4002_8000 – 0x4002_83FF	HDIV	
0x4002_3400 – 0x4002_7FFF	Reserved	
0x4002_3000 – 0x4002_33FF	CRC	
0x4002_2400 – 0x4002_2FFF	Reserved	
0x4002_2000 – 0x4002_23FF	FLASH	
0x4002_1400 – 0x4002_1FFF	Reserved	
0x4002_1000 – 0x4002_13FF	RCC	
0x4002_0C00 – 0x4002_0FFF	Reserved	
0x4002_0800 – 0x4002_0BFF	ADC	
0x4002_0400 – 0x4002_07FF	SQRT	
0x4002_0000 – 0x4002_03FF	DMA	
0x4001_8000 – 0x4001_FFFF	Reserved	
0x4001_4800 – 0x4001_7FFF	Reserved	APB2
0x4001_4400 – 0x4001_47FF	SPI2	
0x4001_3C00 – 0x4001_43FF	Reserved	
0x4001_3800 – 0x4001_3BFF	USART1	
0x4001_3400 – 0x4001_37FF	TIM8	
0x4001_3000 – 0x4001_33FF	Reserved	
0x4001_2C00 – 0x4001_2FFF	TIM1	
0x4001_2400 – 0x4001_2BFF	Reserved	
0x4001_2000 – 0x4001_23FF	SPI1_I2S	
0x4001_1C00 – 0x4001_1FFF	GPIOF	
0x4001_1400 – 0x4001_1BFF	Reserved	
0x4001_1000 – 0x4001_13FF	GPIOC	
0x4001_0C00 – 0x4001_0FFF	GPIOB	
0x4001_0800 – 0x4001_0BFF	GPIOA	
0x4001_0400 – 0x4001_07FF	EXTI	APB1
0x4001_0000 – 0x4001_03FF	AFIO	
0x4000_7400 – 0x4000_FFFF	Reserved	
0x4000_7000 – 0x4000_73FF	PWR	
0x4000_5C00 – 0x4000_6FFF	Reserved	
0x4000_5800 – 0x4000_5BFF	I2C2	APB1
0x4000_5400 – 0x4000_57FF	I2C1	
0x4000_4C00 – 0x4000_53FF	Reserved	
0x4000_4800 – 0x4000_4BFF	LPUART	

Address range	Peripherals	Bus
0x4000_4400 – 0x4000_47FF	USART2	
0x4000_3400 – 0x4000_43FF	Reserved	
0x4000_3000 – 0x4000_33FF	IWDG	
0x4000_2C00 – 0x4000_2FFF	WWDG	
0x4000_2800 – 0x4000_2BFF	RTC	
0x4000_2400 – 0x4000_27FF	COMP	
0x4000_2000 – 0x4000_23FF	OPAMP	
0x4000_1400 – 0x4000_1FFF	Reserved	
0x4000_1000 – 0x4000_13FF	TIM6	
0x4000_0C00 – 0x4000_0FFF	LPTIM	
0x4000_0800 – 0x4000_0BFF	Reserved	
0x4000_0400 – 0x4000_07FF	TIM3	
0x4000_0000 – 0x4000_03FF	BEEPER	

## 2.1.3 Boot management

### 2.1.3.1 Boot address

During system startup, you can select the BOOT mode after the reset through the BOOT0 pin and the user option byte BOOT configuration. The value of the BOOT pin will be re-locked after the system is reset or exits from the PD mode. After a startup delay, the CPU gets the address at the top of the stack from address 0x0000\_0000 and executes the code from the reset vector address indicated by address 0x0000\_0004. Because of the Cortex®-M0 always gets the stack top pointer and reset vector from addresses 0x0000\_0000 and 0x0000\_0004, so boot is only suitable for starting from the CODE area, and address remapping is designed for boot space. There are three boot modes to choose from:

- Boot from Main Flash:
  - ◆ Main flash memory is mapped to the boot space (0x0000\_0000);
  - ◆ Main flash memory is accessible in two address areas, 0x0000\_0000 or 0x0800\_0000;
- Boot from System Memory:
  - ◆ System memory is mapped to boot space (0x0000\_0000);
  - ◆ System memory can be accessed in two address areas, 0x0000\_0000 or 0x1FFF\_0000;
- Boot from the built-in SRAM:
  - ◆ The built-in SRAM is mapped to boot space (0x0000\_0000);
  - ◆ The built-in SRAM is accessible in two address areas, 0x0000\_0000 or 0x2000\_0000;

### 2.1.3.2 Boot configuration

Three different BOOT modes can be selected through the BOOT0 pin and the user option byte BOOT configuration.

Table 2-2 List of boot mode

Boot mode select pin				Boot mode	Specifies the start address for accessing memory space in boot mode		
nBOOT1	nBOOT0	BOOT0 pin	nSWBOOT0		Main Flash	System Memory	SRAM
X	X	0	1	Main Flash start	0x0000_0000 0x0800_0000	0x1FFF_0000	0x2000_0000
X	1	X	0		0x08000000	0x1FFF_0000	0x2000_0000
1	X	1	1	System Memory Start	0x0000_0000	0x0000_0000	0x2000_0000
1	0	X	0		0x08000000	0x1FFF_0000	0x2000_0000
0	X	1	1	SRAM start	0x08000000	0x1FFF_0000	0x0000_0000
0	0	X	0		0x08000000	0x1FFF_0000	0x2000_0000

Note: BOOT0 and GPIO are multiplexed, and input drop - down is used by default during power-on.

### 2.1.3.3 Embedded boot loader

Embedded boot loader programs are stored in System Memory for reprogramming flash Memory through USART1. The USART1 interface can run on an internal 8MHz oscillator (HSI). Please consult the bootstrap manual for further details. For further details, please refer to “The Embedded Boot Loader Manual”.

## 2.2 Memory system

The program memory, data memory, registers and I/O ports are organized in the same 4GB linear address space. Data bytes are stored in the memory in little endian format. The lowest address byte in a word is regarded as the least significant byte of the word, while the highest address byte is the most significant byte. The specifications of program memory and data memory are as follows.

### 2.2.1 FLASH specification

Flash consists of a main memory area and an information area, which are described separately below: (The capacity value in the following description does not include ECC)

- The maximum main memory area is 64KB, also known as main flash memory, which contains 128 Page for storing and running user programs and storing data.
- The information area is 5KB, including 10 Page, and consists of system storage area (3KB), system configuration area (1.5KB) and option byte area (0.5KB).
  - ◆ The System Memory area is 3KB, which contains 6 Page, also known as System Memory, and is used to store and run the BOOT program.
  - ◆ The system configuration area is 1.5KB, including 3 Page.
  - ◆ The Option Byte area is 0.5KB, containing 1 Page, also known as Option Byte, and the effective space is 14B, BOOT programs and user programs can be read, written or erased.

### 2.2.1.1 Flash memory module organization

Bus address space is allocated to the main storage area and the information area.

Table 2-3 Flash bus address list

Memory area	Page name	Address range	Size
The main memory area	Page 0	0x0800_0000 – 0x0800_01FF	0.5 KB
	Page 1	0x0800_0200 – 0x0800_03FF	0.5 KB
	Page 2	0x0800_0400 – 0x0800_05FF	0.5 KB
	:	:	:
	Page 127	0x0800_FE00 – 0x0800_FFFF	0.5 KB
Information area	System memory area	0xFFFF_0000 – 0xFFFF_0BFF	3KB
	System configuration area	0xFFFF_F000 – 0xFFFF_F5FF	1.5 KB
	Option byte area	0xFFFF_F600 – 0xFFFF_F60D	14B
Memory area interface register	FLASH_AC	0x4002_2000 – 0x4002_2003	4B
	FLASH_KEY	0x4002_2004 – 0x4002_2007	4B
	FLASH_OPTKEY	0x4002_2008 – 0x4002_200B	4B
	FLASH_STS	0x4002_200C – 0x4002_200F	4B
	FLASH_CTRL	0x4002_2010 – 0x4002_2013	4B
	FLASH_ADD	0x4002_2014 – 0x4002_2017	4B
	Reserved	0x4002_2018 – 0x4002_201B	4B
	FLASH_OB	0x4002_201C – 0x4002_201F	4B
	FLASH_WRP	0x4002_2020 – 0x4002_2023	4B
	FLASH_ECC	0x4002_2024 – 0x4002_2027	4B

Flash memory is organized into 32-bit wide memory units, which can store codes and data constants.

Information is divided into three parts:

- The system memory area is used for storing a boot program for boot loader mode of the system memory. The boot program uses USART1 serial interface to program the flash memory.
- System configuration area, which contains basic information of the chip.
- Option byte area.

Writing to main memory and information block is managed by embedded flash programming/erasing controller.

There are two ways to protect flash memory from illegal access (read, write and erase):

- Page write protection (WRP)
- Read protection (RDP)

When the flash memory write operation is executed, any read operation to the flash memory will lock the bus, and the read operation can only be carried out correctly after the write operation is completed. That is, when writing or erasing, cannot have any read access to the code or data.

The internal RC oscillator (HSI) must be turned on when the flash memory is programmed (written or erased).

*Note: In the low power consumption mode, all flash memory operations are suspended.*

### 2.2.1.2 Read and write operation

The Flash operation only supports 32-bit operation, and the Flash should be erased before the write operation, and the minimum block size for erasing is one Page 0.5KB. Write operation is divided into erasing and programming phases.

When reading Flash, the number of waiting cycles for reading can be configured by the register. When using, it needs to be calculated in combination with the clock frequency of SYSCLK interface. For example, when  $SYSCLK \leq 18MHz$ , the minimum number of waiting periods is 0; When  $18MHz < SYSCLK \leq 36MHz$ , the minimum number of waiting periods is 1; When  $36MHz < SYSCLK \leq 48MHz$ , the minimum number of waiting periods is 2.

*Note: Enable prefetch buffer whether number of wait periods is not zero can improve overall efficiency.*

### 2.2.1.3 Unlock Flash

After reset, the Flash module is protected and cannot be written into the FLASH\_CTRL register to prevent accidental operation of Flash due to electrical interference and other reasons. By writing a specific sequence of key values into the FLASH\_KEY register, you can open the operation authority of the FLASH\_CTRL register. The specific sequence is: Firstly, writing KEY1 = 0x45670123 in the FLASH\_KEY register. Secondly, write KEY2 = 0xCDEF89AB in the FLASH\_KEY register.

If there is an error in sequence or key value, a bus error will be returned and the FLASH\_CTRL register will be locked until the next reset. The software can check whether the Flash has been unlocked by looking at the FLASH\_CTRL.LOCK bit. If normal lock setting is needed, it can be realized by setting the FLASH\_CTRL.LOCK bit to 1 by software. After that, you can unlock the Flash by writing the correct key value series in FLASH\_KEY.

### 2.2.1.4 Erase and program

#### 2.2.1.4.1 Erase of main memory area

The main memory area can be erased page by page or whole.

##### Page Erase

Page Erase process:

- Check the FLASH\_STS.BUSY bit to confirm that there are no other flash operations in progress;
- Set the FLASH\_CTRL.PER bit to '1';
- Select the page to be erased with the FLASH\_ADD register;
- Set the FLASH\_CTRL.START bit to '1';
- Wait for the FLASH\_STS.BUSY bit to change to '0';
- Read out the erased page and verify it.

##### Mass Erase

Mass Erase process:

- Check the FLASH\_STS.BUSY bit to confirm that there are no other flash operations in progress;
- Set the FLASH\_CTRL.MER bit to '1';
- Set the FLASH\_CTRL.START bit to '1';
- Wait for the FLASH\_STS.BUSY bit to change to '0';
- Read out all pages and verify.

#### 2.2.1.4.2 Main memory area programming

The main memory area can be programmed with 32 bits at a time. When the FLASH\_CTRL.PG bit is '1', writing a word in a flash address will start programming once; Writing any half word of data will result in a bus error. During the programming process (the FLASH\_STS.BUSY bit is '1'), any operation of reading or writing the flash memory will cause the CPU to pause until the end of the flash programming.

Main memory programming process:

- Check the FLASH\_STS.BUSY bit to confirm that there are no other flash operations in progress;
- Set the FLASH\_CTRL.PG bit to '1';
- Write the word to be programmed at the specified address;
- Wait for the FLASH\_STS.BUSY bit to change to '0';
- Read the written address and verify the data.

*Note: When the FLASH\_STS.BUSY bit is '1', you cannot write to any register.*

#### 2.2.1.4.3 Option byte erase and programming

The option byte area is programmed differently from the main storage area. The number of option bytes is only 7 bytes (2 bytes for write protection, 2 bytes for read protection, 1 byte for configuration and 2 bytes for storing user data). After unlocking the Flash, you must write KEY1 and KEY2 respectively (see 2.2.1.3) to the FLASH\_OPTKEY register, and then set the FLASH\_CTRL.OPTWE bit to '1'. At this time, the option byte area can be programmed: set the FLASH\_CTRL.OPTPG bit to '1' and then write the word to the specified address.

When programming the word in the option byte area, use the low byte in the half-word and automatically calculate the high byte (the high byte is the complement of the low byte), and start the programming operation, which will ensure that the option byte and its complement are always correct.

Option byte erase process:

- Check the FLASH\_STS.BUSY bit to confirm that there are no other flash operations in progress;
- Unlock the FLASH\_CTRL.OPTWE bit;
- Set the FLASH\_CTRL.OPTER bit to '1';
- Set the FLASH\_CTRL.START bit to '1';
- Wait for the FLASH\_STS.BUSY bit to change to '0';
- Read the erased option byte and verify it.

Option byte area programming process:

- Check the FLASH\_STS.BUSY bit to confirm that there are no other flash operations in progress;
- Unlock the FLASH\_CTRL.OPTWE bit;
- Set the FLASH\_CTRL.OPTPG bit to '1';
- Writing the word to be programmed to the specified address;
- Wait for the FLASH\_STS.BUSY bit to change to '0';
- Read the written address and verify the data.

### 2.2.1.5 ECC function

The Flash module supports the ECC function to achieve 1-bit error detection and correction. ECC encoding and decoding (error correction and error detection) are performed automatically by the hardware. If an error is detected, an error is set and an interrupt is generated.

### 2.2.1.6 Instruction prefetching

The instruction prefetch function of Flash module supports the prefetch Buffer of 8B. Through instruction prefetching, the instruction execution efficiency of CPU can be improved. The instruction prefetch function can be configured to be enabled or disabled through the register, and it is enabled by default.

### 2.2.1.7 Option byte

Option byte block is mainly used to configure read-write protection, boot mode configuration, software/hardware watchdog and reset options when the system is in power-down or stop mode, and bus address space is allocated for read-write access. They consist of byte with 7 options: 2 byte for write protection, 2 bytes for read protection, 1 byte for configuration option, 2 bytes defined by user, These 7 bytes need to be written through the bus. The option byte block also contains the complement codes corresponding to these 7 option bytes. These complement codes need to be automatically calculated by hardware when the option bytes are written in the bus, and written into Flash together, and used for verification when the option bytes are read.

By default, the option byte block is always readable and write-protected. To write (program/erase) the option byte block, first unlock the Flash, then unlock the option byte: write the correct key-value sequence (KEY1 = 0x45670123, KEY2 = 0xCDEF89AB) in the FLASH\_OPTKEY, and then write the option byte block will be allowed. If the sequence is wrong or the key value is wrong, a bus error will be returned and the option byte will be locked until the next reset. If it is necessary to set the lock normally, it can be realized by writing 0 to the FLASH\_CTRL.OPTWE bit by software, and then the option byte can be unlocked by writing the correct key-value series in the FLASH\_OPTKEY.

After each system reset, the option byte data is read out from the option byte block of Flash and stored in the option byte register (FLASH\_OB/FLASH\_WRP) with read-only property. At the same time, the option byte complement data read out together will be used to verify whether the option byte data is correct. If it does not match, an option byte error flag (FLASH\_OB.OBERR) will be generated. When an option byte error occurs, the corresponding option byte is forced to 0xFF. When the option byte and its complement are both 0xFF (the state after erasing), the above verification steps are skipped and verification is not required.

Table 2-4 Option byte list

Address	[31:24] Corresponding complement code	[23:16] Option byte	[15:8] Corresponding complement code	[7:0] Option byte
0xFFFF_F600	nUSER	USER	nRDP1	RDP1
0xFFFF_F604	nData1	Data1	nData0	Data0
0xFFFF_F608	nWRP1	WRP1	nWRP0	WRP0
0xFFFF_F60C	Reserved	Reserved	nRDP2	RDP2

- Read protection L1 level option byte: RDP1
  - ◆ Protect the code stored in the flash memory;
  - ◆ When the correct value is written, it will be forbidden to read the flash memory;
  - ◆ The result of whether RDP1 is turned on or not can be inquired through FLASH\_OB[1];
- User configuration options: USER
  - ◆ The USER [7:6]: Reserved;
  - ◆ USER[5]: nSWBOOT0\_SEL configuration option, which can be queried by FLASH\_OB[7]
    - 0: nBOOT0 configuration option used for BOOT mode selection
    - 1: BOOT0 Pin is used to select the BOOT mode
  - ◆ USER[4]: nBOOT1 configuration option, which can be queried by FLASH\_OB[6]
  - ◆ USER[3]: nBOOT0 configuration option, which can be queried by FLASH\_OB[5]
  - ◆ USER[2]: nRST\_PD configuration option, which can be queried through FLASH\_OB[4]
    - 0: A reset occurs when PD mode is entered
    - 1: No reset occurs when entering PD mode
  - ◆ USER[1]: nRST\_STOP configuration option, which can be queried through FLASH\_OB[3]
    - 0: A reset occurs when entering STOP mode
    - 1: No reset occurs when entering the STOP mode
  - ◆ USER[0]: WDG\_SW configuration option, which can be queried by FLASH\_OB[2]
    - 0: Hardware watchdog
    - 1: Software watchdog
- 2 bytes of user data: Datax
  - ◆ Data1 (stored in FLASH\_OB[25:18]);
  - ◆ Data0 (stored in FLASH\_OB [17:10]);
- Write protection option byte: WRP0 ~ 1, which can be written through the register FLASH\_WRP [15:0] query

- ◆ WRP0: Write protection for pages 0 to 63, bit[0] corresponds to Page (0 to 7)..., bit[7] corresponds to Page (56~63);
- ◆ WRP1: Write protection for pages 64~127, bit[0] corresponds to Page (64~71)..., bit[7] corresponds to Page (120~127);
- Read protection L2 level option byte: RDP2
  - ◆ Add protection function on the basis of L1, see 2.2.1.9 detailed description of read protection;
  - ◆ Whether RDP2 is turned on or not can be determined by FLASH\_OB [31] query;

### 2.2.1.8 Write protect

Write protection can be configured for all pages in the flash main storage area (maximum 64KB) to prevent accidental write operations caused by program runaway or electrical interference. The basic unit of write protection is: for Page0 ~ 127, every 8 pages is a basic protection unit. Write protection can be configured by setting WRP0 ~ 1 in the option byte block; After each configuration, a system reset is required for the configured value to be reloaded to take effect. If an attempt is made to program or erase a protected page, a protection error flag will be returned in the FLASH\_STS.

The system memory block (3KB) in the system information area stores the boot program and cannot be changed.

The system configuration block (1.5KB) in the system information area stores the basic information of the chip and cannot be changed.

The option byte block (0.5KB) in the system information area stores the user-configurable option byte information. The write protection of the option byte block is achieved by writing 0 to the FLASH\_CTRL.OPTWE bit by software, and after that, you can write the correct key value series in FLASH\_OPTKEY to release the write protection of the option byte.

### 2.2.1.9 Read protection

The user code in flash can be protected from illegal reading by setting read protection. Read protection is set by configuring RDP bytes in the option byte block. Three different read protection levels can be configured, as shown in the following Table

Table 2-5 Read protection configuration list

Read protection status	RDP1	nRDP1	nRDP2	RDP2
L1 level	0xFF	0xFF	RDP2!=0xCC    nRDP2!=0x33	
Unprotected	0xA5	0x5A	RDP2!=0xCC    nRDP2!=0x33	
L2 level	0xXX	0xXX	0x33	0xCC
L1 level	Not the above three configurations			

#### ■ L0 level:

- ◆ In unprotected state, corresponding (RDP1 == 0xA5 & nRDP1 == 0x5A) && (RDP2!= 0xCC | nRDP2!= 0x33);
- ◆ The main memory area and option byte block can be read arbitrarily;
- ◆ The write protection property of each page can be configured for programming and erasing;

- L1 level:
  - ◆ The corresponding ~((RDP1 == 0xA5 & nRDP1 == 0x5A) && (RDP2!=0xCC | nRDP2!=0x33)) | (RDP2 == 0xCC & nRDP2 == 0x33));
  - ◆ Only the read operation of the main storage area from the user code is allowed, that is, when the program is started from the main flash memory in non debugging mode, the read operation of the main storage area is allowed;
  - ◆ All main memory pages can be programmed by code executing in main flash memory (for functions such as IAP or data storage)
  - ◆ All main memory pages do not allow write or erase operations in debug mode or after booting from internal SRAM (except for mass erase);
  - ◆ All pages are not allowed to write or erase in debug mode or after booting from internal SRAM (except for mass erase);
  - ◆ All functions of loading code into the built-in SRAM through JTAG/SWD and then execute it are still valid, or they can be started from the built-in SRAM through JTAG/SWD, which can be used to remove read protection;
  - ◆ When the read-protected option byte is rewritten to the unprotected L0 level, all the main storage areas will be automatically erased, and the process is as follows: (Erasing the option byte block will not result in automatic whole erasing operation, because the result of erasing is 0xFF, which is equivalent to still being in the protection state of L1 level)
    - Write the correct key value sequence in FLASH\_OPTKEY to unlock the option byte area;
    - The bus initiates a command to erase the entire option byte area (Page erase);
    - Bus write 0xA5 to read protection option byte;
    - Automatically erase all main storage areas internally;
    - Automatically write 0xA5 to read protection option byte internally;
    - When the system is reset (such as software reset, etc.), the option byte block (including the new RDP value 0xA5) will be reloaded into the system, and the read protection will be released;
- L2 level: Except that SRAM boot disabled, debug mode disabled, option byte write/page erase disabled and the protection level cannot be modified (irreversible), other features are the same as L1 level. The L2 level is realized by configuring another option byte, RDP2. No matter what the value of RDP1 is, as long as it satisfies (RDP2==0xCC & nRDP2==0x33), it is L2 level.

### 2.2.1.10 Permission Protection

- Flash main storage area permissions:
  - ◆ Under L0 level: the main storage area can be read; the main storage area can be configured with the write protection attribute of each Page for programming and page erasing;
  - ◆ Under L1/2 level:

- When executed in SWD debug mode or after booting from internal SRAM, all Pages do not allow (W/R/PE) operations;
  - The 0th to 7th pages are automatically write-protected, and other pages can be programmed through the code executed in the main flash memory area (implementing functions such as IAP or data storage);
  - Other Pages can configure the write protection property of each Page;
  - When the L1 level is rewritten to the L0 level, all Flash main storage areas will be automatically erased;
- Flash option byte area permission:
- ◆ Under the L0/L1 level, access is allowed (W/R/PE);
  - ◆ Under L2 level: except that the debug mode is disabled, read-only access to the Flash option byte area is allowed;
- Flash system storage area permissions:
- ◆ Only the code executed in the system memory area is allowed to access (W/R/PE); Flash and sram execution code or through the debugging interface are not allowed to access;
  - ◆ At the L1/L2 level, the sram boot jumps to the system memory execution code and does not allow access (W/R/PE). access otherwise allowed (W/R/PE);
- Flash system configuration area
- ◆ User information: The read operation can only be performed after the system memory is started or executed by jumping to the system memory;
  - ◆ ID information: readable at L0 level, access prohibited in SWD debug mode at L1/L2 level, access prohibited at L1 level after sram is started, see Table 2-6 for details;

Table 2-6 Flash read-write-erase<sup>(1)</sup> permission control table

protect level	Boot mode	Main Flash				Changing a Protection Level
	Perform user Access area	SWD	Main Flash	System Memory	SRAM	
L0 level	Before 4KB of flash main memory area	Read-Write- Erase	Read-Write- Erase	Read-Write-Erase	Read-Write- Erase	Change to L1 or L2 is allowed
	After 4KB of flash main memory area	Read-Write- Erase	Read-Write- Erase	Read-Write-Erase	Read-Write- Erase	
	Flash main memory area mass erase	Allow	Allow	Allow	Allow	
	Flash option byte area	Read-Write-	Read-Write-	Read-Write-Erase	Read-Write-	

		Erase	Erase		Erase	
	Flash system memory area	prohibit	prohibit	Read-Write-Erase	prohibit	
	SRAM (All)	Read and write	Read and write	Read and write	Read and write	
L1 level	Before 4KB of flash main memory area	Prohibit	Read-only	Read-only	Read-only	L0 or L2 is allowed. When changed to L0, the main memory area is automatically erased.
	After 4KB of flash main memory area	Prohibit	Read-Write-Erase	Read-Write-Erase	Read-Write-Erase	
	Flash main memory area mass erase	Allow	Allow	Allow	Allow	
	Flash option byte area	Read-Write-Erase	Read-Write-Erase	Read-Write-Erase	Read-Write-Erase	
	Flash system memory area	Prohibit	Prohibit	Read-write-erase	Prohibit	
	SRAM (All)	Read and write	Read and write	Read and write	Read and write	
L2 level	Before 4KB of flash main memory area	The interface is disabled.	Read-only	Read-only	Read-only	No modification is allowed.
	After 4KB of flash main memory area		Read-write-erase	Read-write-erase	Read-write-erase	
	Flash main memory area mass erase		Allow	Allow	Allow	
	Flash option byte area		Read-only	Read-only	Read-only	
	Flash system memory area		Prohibit	Read-write-erase	Prohibit	
	SRAM (All)		Read and write	Read and write	Read and write	

protect level	Boot mode	SRAM				Changing a Protection Level
	Perform user Access to areas	SWD	Main Flash	System Memory	SRAM	
L0 level	Before 4KB of flash main memory area	Read-write-erase	Read-write-erase	Read-write-erase	Read-write-erase	Change to L1 or L2 is allowed
	After 4KB of flash main memory area	Read-write-erase	Read-write-erase	Read-write-erase	Read-write-erase	
	Flash main memory area mass erase	Allow	Allow	Allow	Allow	
	Flash option byte area	Read-write-erase	Read-write-erase	Read-write-erase	Read-write-erase	
	Flash system memory area	Prohibit	Prohibit	Read-write-erase	Prohibit	
	SRAM (All)	Read and write	Read and write	Read and write	Read and write	
L1 level	Before 4KB of flash main memory area	Prohibit	Read-only	Read-only	Prohibit	L0 or L2 is allowed. When changed to L0, the main memory area is automatically erased.
	After 4KB of flash main memory area	Prohibit	Read-write-erase	Read-write-erase	Prohibit	
	Flash main memory area mass erase	Allow	Allow	Allow	Allow	
	Flash option byte area	Read-write-erase	Read-write-erase	Read-write-erase	Read-write-erase	
	Flash system memory area	Prohibit	Prohibit	Prohibit	Prohibit	
	SRAM (All)	Read and write	Read and write	Read and write	Read and write	
L2 level	Before 4KB of flash main memory area	L2 protection level, cannot boot from SRAM				No modification is allowed.

	After 4KB of flash main memory area					SWD is banned.
	Flash main memory area mass erase					
	Flash option byte area					
	Flash system memory area					
	SRAM (All)					
protect level	Boot mode	System Memory				Changing a Protection Level
	Perform user Access to areas	SWD	Jump to Flash	System Memory Start the	Jump to SRAM	
L0 level	Before 4KB of flash main memory area	Read-write-erase	Read-write-erase	Read-write-erase	Read-write-erase	Change to L1 or L2 is allowed
	After 4KB of flash main memory area	Read-write-erase	Read-write-erase	Read-write-erase	Read-write-erase	
	Flash main memory area mass erase	Allow	Allow	Allow	Allow	
	Flash option byte area	Read-write-erase	Read-write-erase	Read-write-erase	Read-write-erase	
	Flash system memory area	Prohibit	Prohibit	Read-write-erase	Prohibit	
	SRAM (All)	Read and write	Read and write	Read and write	Read and write	
L1 level	Before 4KB of flash main memory area	Prohibit	Read-only	Read-only	Read-only	L0 or L2 is allowed. When changed to L0,

	After 4KB of flash main memory area	Prohibit	Read-write-erase	Read-write-erase	Read-write-erase	the main memory area is automatically erased.
	Flash main memory area mass erase	Allow	Allow	Allow	Allow	
	Flash option byte area	Read-write-erase	Read-write-erase	Read-write-erase	Read-write-erase	
	Flash system memory area	Prohibit	Prohibit	Read-write-erase	Prohibit	
	SRAM (All)	Read and write	Read and write	Read and write	Read and write	
L2 level	Before 4KB of flash main memory area	SWD The interface is disabled.	Read-only	Read-only	Read-only	No modification allowed
	After 4KB of flash main memory area		Read-write-erase	Read-write-erase	Read-write-erase	
	Flash main memory area mass erase		Allow	Allow	Allow	
	Flash option byte area		Read-only	Read-only	Read-only	
	Flash system memory area		Prohibit	Read-write-erase	Prohibit	
	SRAM (All)		Read and write	Read and write	Read and write	

Note: 1. Erase here refers to flash page erase.

## 2.2.2 SRAM

SRAM is mainly used for code running, storing variables and data or stacks in the process of program execution, with a maximum capacity of 8KB.

SRAM supports read-write access of byte, half-word and word.

SRAM supports code running and can run programs at full speed in SRAM. The maximum address range of SRAM is 0x2000 0000 to 0x2000 1FFF.

SRAM cannot hold data in PD mode; other operating modes (Run, Lprun, Sleep, and Stop) Data retention normally.

The main features are as follows:

- The maximum total capacity is 8KB
- Support byte/half word/word reading and writing
- CPU/DMA access
- The CPU BUS can Remap to SRAM to run the program at full speed.

## 2.2.3 FLASH register description

These peripheral registers must be operated as words (32 bits).

### 2.2.3.1 FLASH register overview

Table 2-7 FLASH register overview

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0							
	FLASH_AC	Reserved																								LATENCY														
	Reset Value																										1	1	0	0	0									
004h	FLASH_KEY	FKEY																									11	12	0	0	0									
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
008h	FLASH_OPTKEY	OPTKEY																																						
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
00Ch	FLASH_STS	Reserved																																						
	Reset Value																																							
010h	FLASH_CTRL	Reserved																																						
	Reset Value																																							
014h	FLASH_ADD	FADD																																						
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						
018h	Reserved																																							
01Ch	FLASH_OB	RDPT2	Reserved	Data1								Data0								Not Used	ECCERR		0	PRFTBFS		5	PRFTBFE		4	Reserved		3	2		1		0			
	Reset Value	0		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	LOCK	0	START	0	EOP	0	WRPERR	0	0	MER	0	PGEERR	0	PER	0	PG	0	BUSY	0	
020h	FLASH_WRP	Reserved																WRPT												1		1		1		1		1		
	Reset Value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		
024h	FLASH_ECC	Reserved																																						
	Reset Value																																							

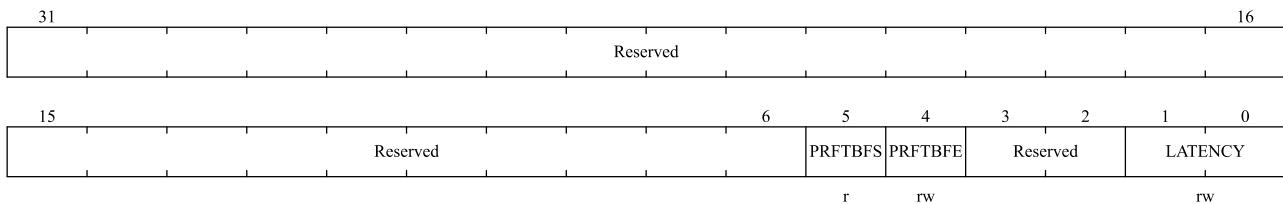
### 2.2.3.2 FLASH control and status registers

See for abbreviations in register descriptions 1.1 section.

### 2.2.3.2.1 FLASH access control register (FLASH\_AC)

Address offset: 0x00

Reset value: 0x0000 0030

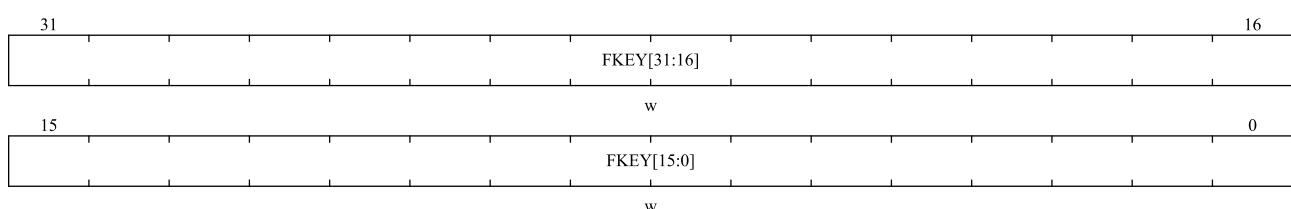


Bit field	Name	Description
31:6	Reserved	Reserved, the reset value must be maintained.
5	PRFTBFS	Pre-fetch buffer status This bit indicates the state of the pre-fetch buffer 0: The pre-fetch buffer is disabled. 1: The pre-fetch buffer is enabled.
4	PRFTBFE	The pre-fetch buffer is enabled 0: Disables the pre-fetch buffer. 1: Enables the pre-fetch buffer.
3:2	Reserved	Reserved, the reset value must be maintained
1:0	LATENCY	Time delay These bits represent the ratio of the SYSCLK (system clock) cycle to the flash access time 00: Zero periodic delay, when $0 < \text{SYSCLK} \leq 18\text{MHz}$ 01: A periodic delay, when $18\text{MHz} < \text{SYSCLK} \leq 36\text{MHz}$ 10: Two periodic delay, when $36\text{MHz} < \text{SYSCLK} \leq 48\text{MHz}$ 11: Reserved

### 2.2.3.2.2 FLASH key register (FLASH\_KEY)

Address offset: 0x04

Reset value: 0xFFFF FFFF

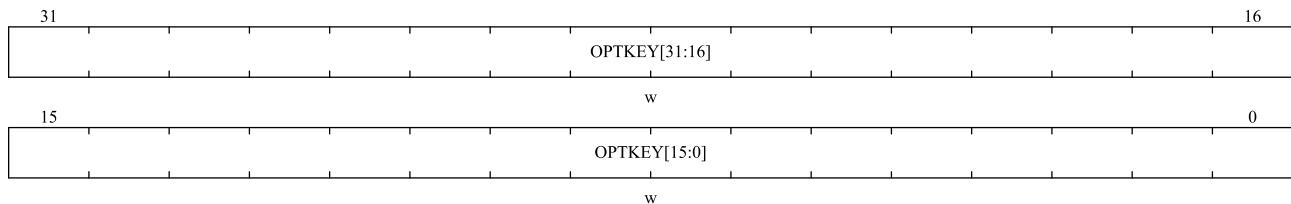


Bit field	Name	Description
31:0	FKEY	Used to unlock the FLASH_CTRL.LOCK bit.

### 2.2.3.2.3 FLASH OPTKEY register (FLASH\_OPTKEY)

Address offset: 0x08

Reset value: 0xFFFFFFFF

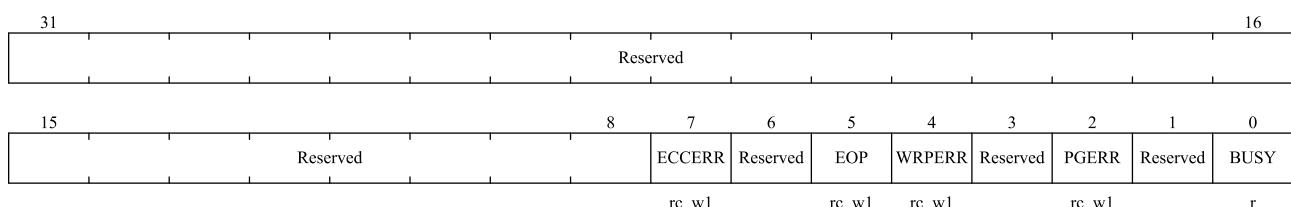


Bit field	Name	Description
31:0	OPTKEY	Used to unlock the FLASH_CTRL.OPTWE bit.

### 2.2.3.2.4 FLASH status register (FLASH\_STS)

Address offset: 0x0C

Reset value: 0x0000 0000



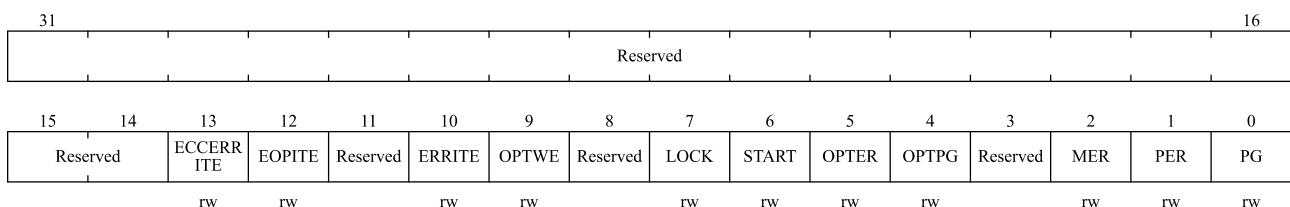
Bit field	Name	Description
31:8	Reserved	Reserved, the reset value must be maintained
7	ECCERR	ECC error Error reading FLASH, hardware set this to '1', write '1' to clear this state.
6	Reserved	Reserved, the reset value must be maintained
5	EOP	End of the operation When the flash operation (programming/erasing) is complete, the hardware sets this to '1' and writing '1' clears this state. <i>Note: EOP status is set for each successful programming or erasure.</i>
4	WRPERR	Write protection error When attempting to program a write protected flash address, hardware sets this to '1' and writing '1' clears this state.
3	Reserved	Reserved, the reset value must be maintained
2	PGERR	Programming errors When trying to program to an address whose content is not '0xFFFF_FFFF', the hardware sets this to '1'. Writing '1' clears this state. <i>Note: Before programming, the FLASH_CTRL.START bit must be cleared.</i>
1	Reserved	Reserved, the reset value must be maintained

Bit field	Name	Description
0	BUSY	Busy This bit indicates that a flash operation is in progress. This bit is set to '1' at the start of the flash operation; This bit is cleared to '0' at the end of the operation or when an error occurs.

### 2.2.3.2.5 FLASH control register (FLASH\_CTRL)

Address offset: 0x10

Reset value: 0x0000 0080



Bit field	Name	Description
31:14	Reserved	Reserved, the reset value must be maintained
13	ECCERRITE	ECC error interrupt This bit allows interrupts to occur when the FLASH_STS.ECCERR bit changes to '1'. 0: Forbid interruption. 1: Interrupts are allowed.
12	EOPITE	Allow operation completion interrupt. This bit allows an interrupt to be generated when the FLASH_STS.EOP bit becomes '1'. 0: Forbid interruption. 1: Interrupts are allowed.
11	Reserved	Reserved, the reset value must be maintained
10	ERRITE	Allow error status to be interrupted This bit allows interrupts in the event of Flash errors (when FLASH_STS.PGERR/ FLASH_STS.WRPERR is set to '1'). 0: Forbid interruption. 1: Interrupts are allowed.
9	OPTWE	Allows option bytes to be written When the bit is '1', programmatic manipulation of the option byte is allowed. When the correct key sequence is written to the FLASH_OPTKEY register, the bit is set to '1'. Software can clear this bit.
8	Reserved	Reserved, the reset value must be maintained
7	LOCK	Lock This bit can only be written as '1'. When the bit is '1', Flash and FLASH_CTRL

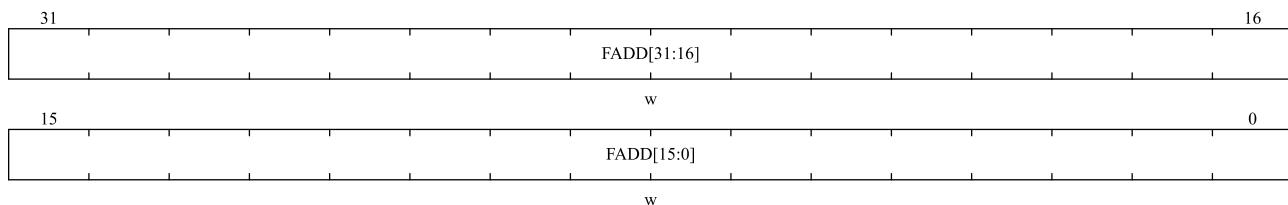
Bit field	Name	Description
		are locked. Hardware clears this bit to '0' after detecting a correct unlock sequence. After an unsuccessful unlock operation, this bit cannot be changed until the next system reset.
6	START	Start An erase operation is triggered when the bit is '1'. This bit can only be set by software to '1' and cleared to '0' when FLASH_STS.BUSY changes to '1'.
5	OPTER	Erase option bytes 0: Disable option bytes erase mode; 1: Enable option bytes erase mode.
4	OPTPG	Program option bytes 0: Disable option bytes program mode; 1: Enable option bytes program mode.
3	Reserved	Reserved, the reset value must be maintained
2	MER	Mass erase 0: Disable mass erase mode; 1: Enable mass erase mode.
1	PER	Page erase 0: Disable mass erase mode; 1: Enable mass erase mode.
0	PG	Program 0: Disable Program mode; 1: Enable Program mode.

Note: Please refer to Section 2.2.1.4 for programming and erasing.

### 2.2.3.2.6 FLASH address register (FLASH\_ADD)

Address offset: 0x14

Reset value: 0x0000 0000

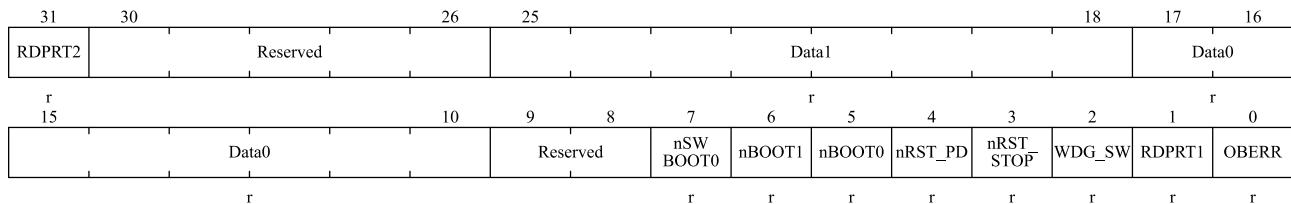


Bit field	Name	Description
31:0	FADD	Flash memory address Select the address to program when programming and the page to erase when erasing. <i>Note: When the FLASH_STS.BUSY bit is '1', this register cannot be written.</i>

### 2.2.3.2.7 FLASH option byte register (FLASH\_OB)

Address offset: 0x1C

Reset value: 0x03FF FFFF



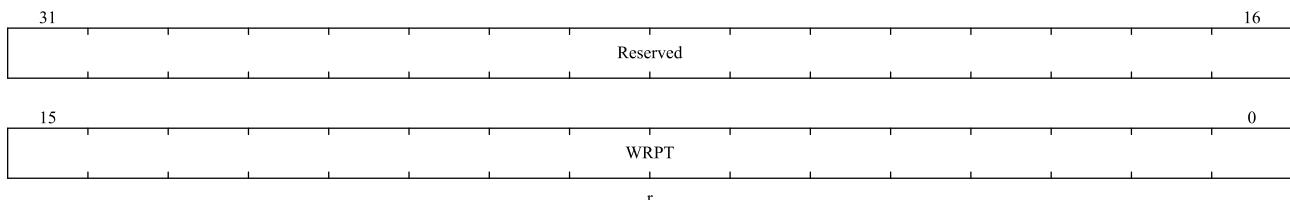
Bit field	Name	Description
31	RDPRT2	Read protection level L2 0: Read protection L2 is disabled. 1: Read protection L2 is enabled. <i>Note: This bit is read-only.</i>
30:26	Reserved	Reserved, the reset value must be maintained
25:18	Data1[7:0]	Data1 <i>Note: This bit is read-only.</i>
17:10	Data0[7:0]	Data0 <i>Note: This bit is read-only.</i>
9:8	Reserved	Reserved, the reset value must be maintained
7	nSWBOOT0	For the usage rules, see 2.1.3.2 Boot configuration.
6	nBOOT1	For the usage rules, see 2.1.3.2 Boot configuration.
5	nBOOT0	For the usage rules, see 2.1.3.2 Boot configuration.
4	nRST_PD	The Power Down mode is used to reset the configuration 0: A reset occurs immediately after the system enters the Power Down mode. Even if the system enters the Power Down mode, the system will be reset instead of entering the Power Down mode. 1: The system does not reset after entering the Power Down mode. <i>Note: This bit is read-only.</i>
3	nRST_STOP	Enter the STOP mode to reset the configuration 0: A reset occurs immediately after entering the STOP mode. Even if the process of entering the STOP mode is executed, the system will be reset instead of entering the STOP mode. 1: No reset is generated after the STOP mode is entered. <i>Note: This bit is read-only.</i>
2	WDG_SW	Watchdog Settings 0: Hardware watchdog. 1: Software watchdog. <i>Note: This bit is read-only.</i>
1	RDPRT1	Read protection level L1

Bit field	Name	Description
		0: The L1 level of read protection is disabled. 1: L1 read protection is enabled. <i>Note: This bit is read-only.</i>
0	OBERR	Option byte error When this bit is '1', the option byte does not match its inverse. <i>Note: This bit is read-only.</i>

### 2.2.3.2.8 FLASH write protection register (FLASH\_WRP)

Address offset: 0x20

Reset value: 0x0000 FFFF

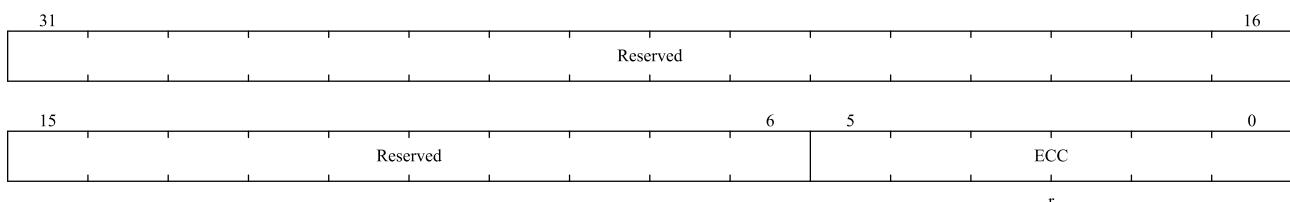


Bit field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained
15:0	WRPT	Write protect This register contains the write protection option byte loaded by option byte area. 0: Write protection takes effect. 1: Write protection is invalid. <i>Note: These bits are read-only.</i>

### 2.2.3.2.9 FLASH ECC register (FLASH\_ECC)

Address offset: 0x24

Reset value: 0x0000 0000



Bit field	Name	Description
31:6	Reserved	Reserved, the reset value must be maintained
5:0	ECC	After writing a word to a 32-bit Flash address, the corresponding higher 6-bit ECC value.

## 3 Power control (PWR)

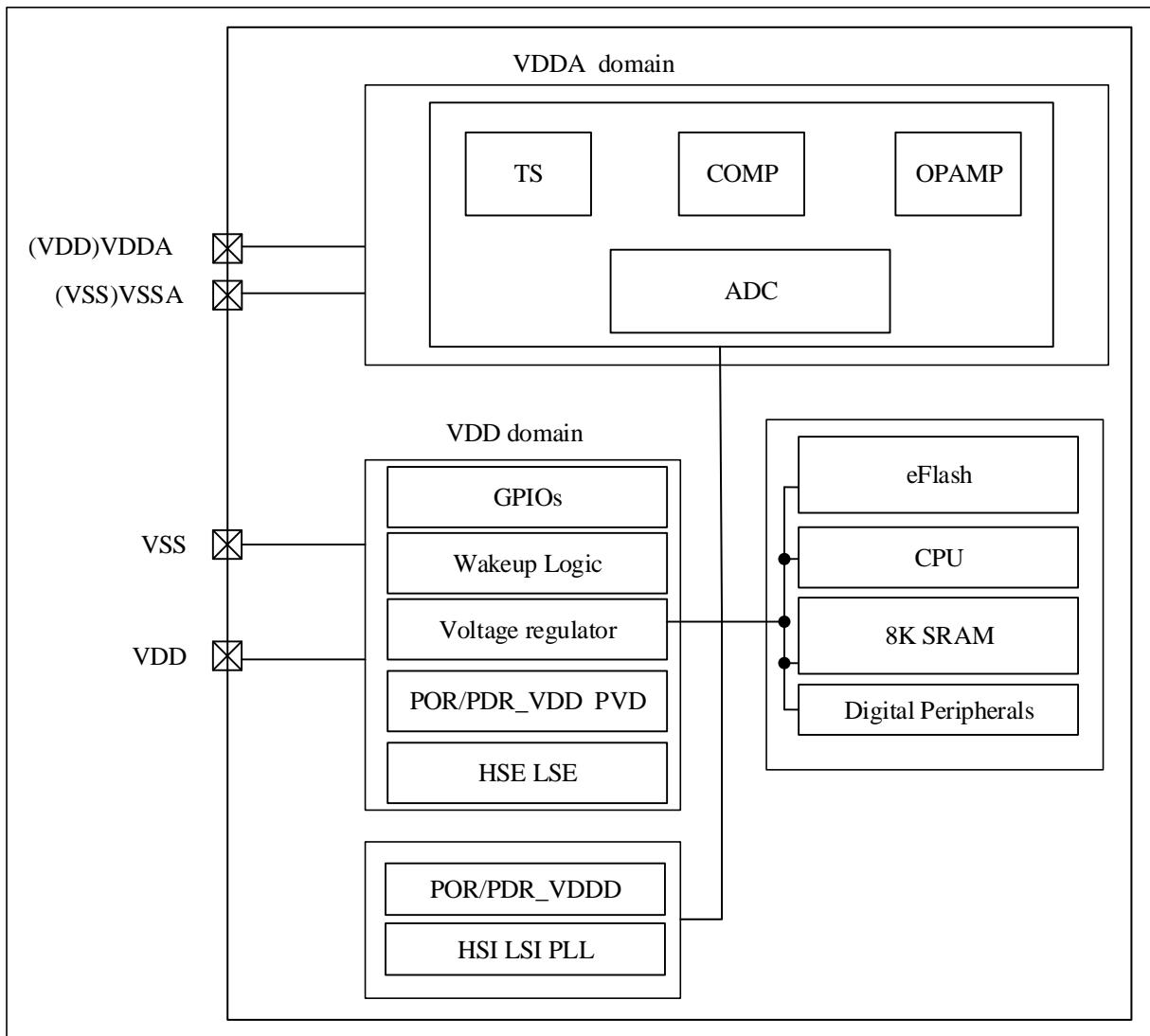
### 3.1 General description

PWR is power management unit to control status of different modules in different power modes. Its major function is to control MCU to enter different power modes and wakeup when events or interrupts happen. MCU supports RUN, LPRUN, SLEEP, STOP and PD (Power Down) mode. PWR controls voltage regulator, Clock sources, Resets and Flash/SRAM/GPIO status in different power modes.

#### 3.1.1 Power supply

- ❖ MCU has an external VDD supply. Embedded voltage regulator is used to supply the internal 1.5V digital power supplies. Voltage regulator has two modes, normal mode and low power mode.
  - VDD: 1.8V~5.5V, which mainly provides power input for MR, IO and clock reset system.
  - VDDA: 1.8V~5.5V, which powers most analog peripherals. For details, please refer to the electrical characteristics section of the relevant data sheet.
  - VDDA and VSSA must be connected to VDD and VSS respectively.
- ❖ Voltage regulator operates in several different modes, depending on the application:
  - RUN mode: The voltage regulator provides power in normal power mode.
  - LPRUN mode: The voltage regulator provides power in normal power mode.
  - SLEEP mode: The voltage regulator provides power in normal power mode.
  - STOP mode: The voltage regulator provides power in low power mode and the output voltage can be configured to 1.5V or 1.2V by software.
  - PD mode: The voltage regulator is turned off.

Figure 3-1 Power supply block diagram

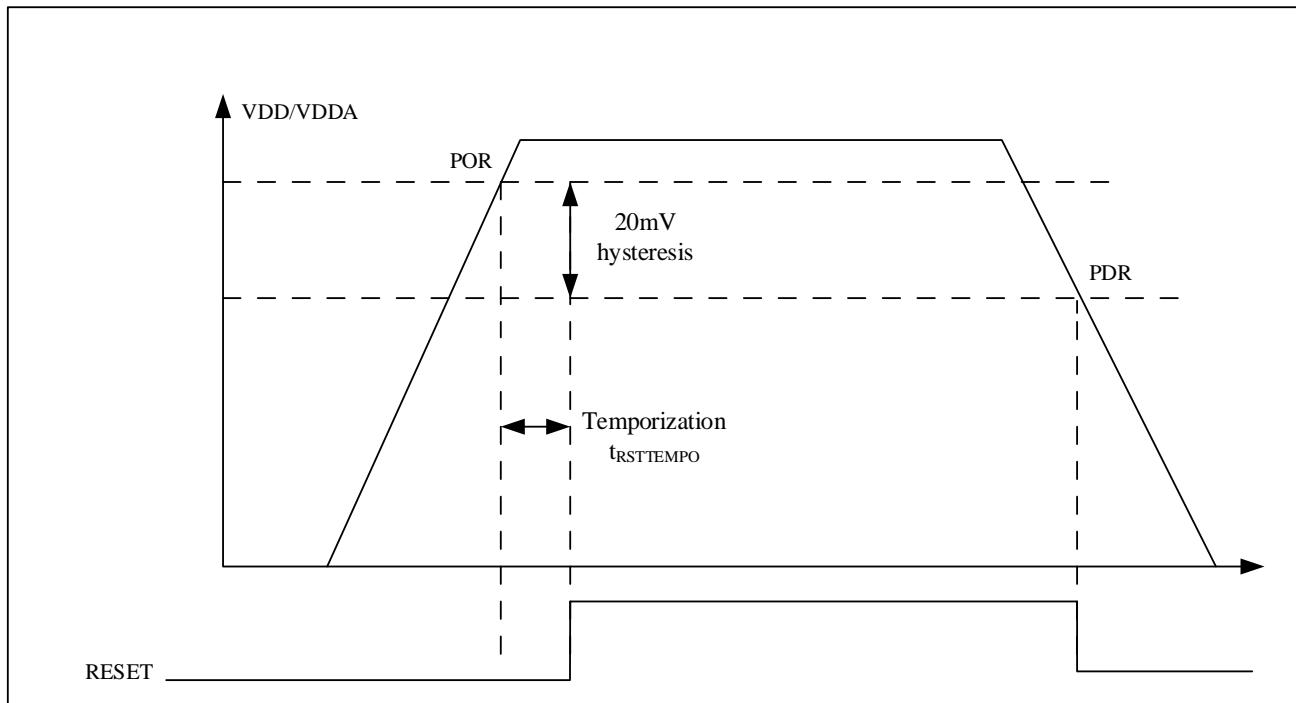


### 3.1.2 Power supply supervisor

#### 3.1.2.1 Power on reset (POR) and power down reset (PDR)

Power on reset (POR) and power down reset (PDR) circuits are integrated inside the chip. When VDD/VDDA is below the specified limit voltage VPOR/VPDR, the system remains in a reset state without the need for an external reset circuit. Refer to the electrical characteristics section of the data sheet for details on power-on and power-off resets.

Figure 3-2 Power on reset/power down reset waveform

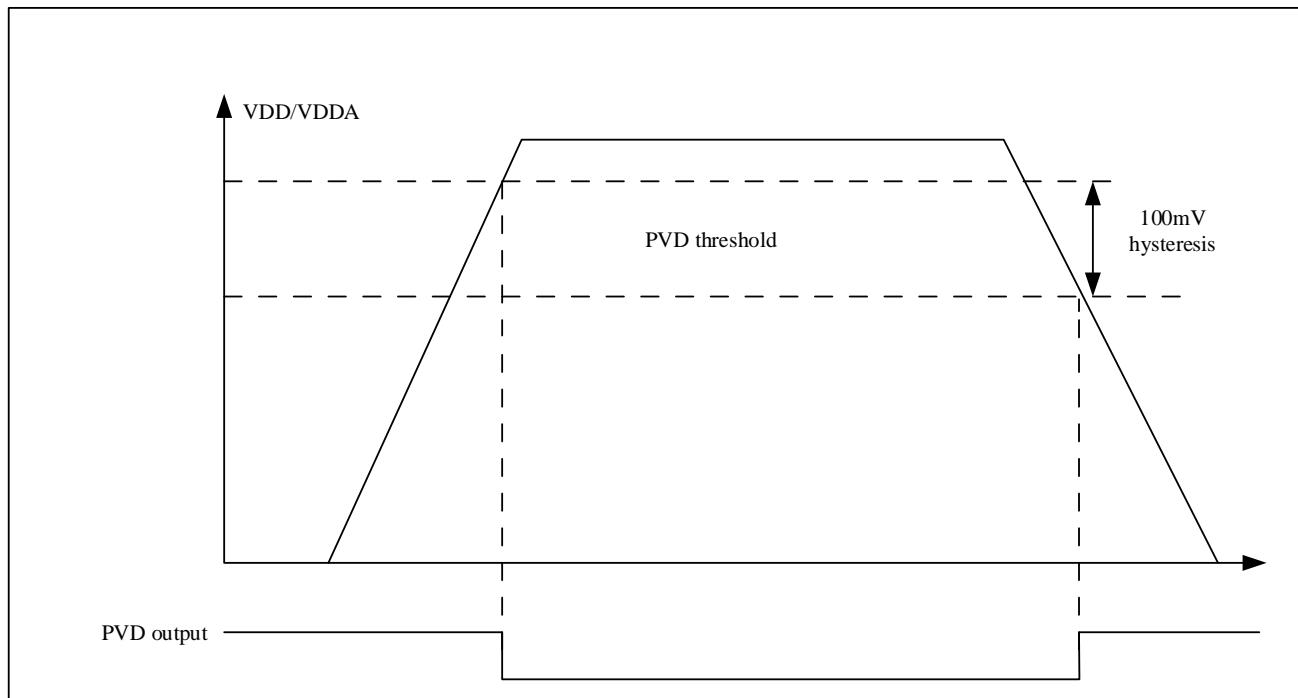


### 3.1.2.2 Programmable voltage detector (PVD)

PVD monitors the power supply by comparing the VDD voltage with the relevant bits in the power control register (PWR\_CTRL). PWR\_CTRL.PLS select the threshold of the monitoring voltage. Enable PVD by setting the PWR\_CTRL.PVDEN.

The PWR\_CTRLSTS.PVDO flag is used to indicate whether the VDD is above/below the PVD voltage threshold. This event is connected internally to EXTI line16 and produces an interrupt if the interrupt is enabled in the external interrupt register. A PVD break occurs when the VDD drops below the PVD threshold and/or when the VDD rises above the PVD threshold, according to the rise/fall edge trigger setting of EXTI line 16. For example, this feature can be used to perform emergency shutdown tasks.

Figure 3-3 PVD threshold diagram



### 3.1.3 NRST

NRST is an analog PAD. In STOP mode, PWR detects NRST reset event and asynchronously switches voltage regulator back to normal mode.

## 3.2 Power modes

Overall MCU has five power modes: RUN, LPRUN, SLEEP, STOP and PD. Different mode has different performance and power consumption. A summary of MCU power modes is shown below.

Table 3-1 Power modes

Mode	Condition	Enter	Exit
RUN	CPU running, all peripherals configurable.	Power up, system reset, or wakeup from other power modes.	Enter LPRUN, SLEEP, STOP and PD mode.
LPRUN from SRAM	CPU running at LSI or LSE, PLL off, all other peripherals configurable. Voltage regulator in normal mode. Flash enter deep standby mode.	software control	software control
LPRUN from FLASH	CPU running at LSI or LSE, PLL off, all other peripherals configurable. Voltage	software control	software control

Mode	Condition	Enter	Exit
	regulator in normal mode.		
SLEEP	CPU sleep, all peripherals configurable, regulator in normal mode, all digital peripherals powered. Interrupts & Events can wakeup CPU.	WFI  CPU returns from ISR  WFE	Any interrupts wakeup event.
STOP	CPU deep SLEEP, peripherals clock disabled. Voltage regulator in LP mode.  Flash enter deep standby mode.  HSE/HSI/PLL disabled. LSE/LSI configurable. RTC//LPUART/LPTIM/ COMP optional. SRAM/All register retention. All IO retention. After waking up, HSI is enabled.	WFI/WFE:  1 <input type="checkbox"/> SCB_SCR.SLEEPDEEP = 1, no pending interrupts/events.  2 <input type="checkbox"/> PWR_CTRL.PDSTP = 0	Any interrupts wakeup event through EXTI, NRST, IWDG.
PD	Voltage regulator OFF, all clocks OFF, most IO output High-Z.  NRST/PA0_WKUP0/PC13_WKUP1/PA2_WKUP2 can wake up the chip.	WFI/WFE <input type="checkbox"/>  1 <input type="checkbox"/> SCB_SCR.SLEEPDEEP = 1, no pending interrupt/event.  2 <input type="checkbox"/> PWR_CTRL.PDSTP = 1	WKUP0/1/2 rising or falling edge, NRST reset.

Note:

1. STOP mode, after wakeup, codes can resume and continue from stopped location.
2. CPU running, all peripherals in VDDD domain disabled.

The running enable conditions of different modules in different power consumption modes are shown in the following table:

Table 3-2 Peripheral running status

Peripheral	Run/Active	Sleep	Low power run	Stop mode		Power down mode	
				Status	Wakeup capability	Status	Wakeup capability
Cortex-M0	Y	-	Y	-	-	-	-
FLASH	O	O	O	O	-	-	-
SRAM8KB	Y	Y	Y	Y	-	-	-
POR/PDR	Y	Y	Y	Y	Y	-	-
PVD	O	O	O	O	O	-	-
DMA	O	O	O	-	-	-	-

USART/UART	O	O	O	-	-	-	-
LPUART	O	O	O	O	O	-	-
I2C	O	O	O	-	-	-	-
SPI	O	O	O	-	-	-	-
CAN	O	O	O	-	-	-	-
RTC	O	O	O	O	O	-	-
TIMER	O	O	O	-	-	-	-
LPTIMER	O	O	O	O	O		
HSE	O	O	-	-	-	-	-
HSI	O	O	-	-	-	-	-
LSE	O	O	O	O	-	-	-
LSI	O	O	O	O	-	-	-
PLL	O	O	-	-	-	-	-
IWDG	O	O	O	O	O	-	-
WWDG	O	O	O	-	-	-	-
ADC	O	O	-	-	-	-	-
Temperature Sensor	O	O	O	-	-	-	-
OPA	O	O	O	-	-	-	-
LPCOMP	O	O	O	O	O		
SysTick timer	O	O	O	-	-		
CRC	O	O	O	-	-	-	-
HDIV/SQRT	O	O	O	-	-	-	-
GPIOs	O	O	O	O	O	-	3 pins

Note:

1. Y: Yes (Enable), O: Optional (Disabled by default, Enabled by software), -: Not available.
2. The pins that can wake up from the PD are PA0 (WKUP0), PC13 (WKUP1), PA2 (WKUP2), and NRST.

### 3.2.1 LPRUN mode

In LPRUN mode, the system clock source can be configured as LSI or LSE by RCC\_LSCTRL.LPRUNCLKSEL; PLL is off; all peripherals are configurable; the voltage regulator operates in normal mode. If the code runs in SRAM, PWR\_CTRL4.LPRUNFLH can be configured to make the FLASH enter deep standby mode. It cannot be configured if running on FLASH.

After entering LPRUN mode, the user cannot configure RCC\_CFG.SCLKSW[2:0] to switch the system clock.

#### 3.2.1.1 Enter LPRUN mode

Before entering the LPRUN mode, user needs to turn on the LSI or LSE, and select the system clock source in the LPRUN mode through RCC\_LSCTRL.LPRUNCLKSEL, and then configure PWR\_CTRL4.LPRUNEN = 1, user can configure the FLASH to enter the deep standby mode as required. Note that a write key is required to enable

write protection before accessing the PWR\_CTRL4 register.

### 3.2.1.2 Exit LPRUN mode

Software sets PWR\_CTRL4.LPRUNEN = 0 to exit LPRUN mode, user can turn off LSI or LSE and switch system clock as needed. After exiting LPRUN mode, FLASH automatically returns to normal mode.

## 3.2.2 SLEEP mode

The CPU stops and all peripherals including peripherals around the Cortex®-M0 core (such as NVIC, SysTick, etc.) can run and wake up the CPU when an interrupt or event occurs.

### 3.2.2.1 Enter SLEEP mode

Enter SLEEP mode by executing WFI (wait for interrupt) or WFE (wait for event) instruction with SCB\_SCR.SLEEPDEEP = 0. Depending on the SCB\_SCR.SLEEPONEXIT, there are two options for SLEEP mode entry:

- SLEEP-NOW: If SCB\_SCR.SLEEPONEXIT = 0, then WFI or WFE instruction is executed immediately, and the system enters sleep mode immediately.
- SLEEP-ON-EXIT: If SCB\_SCR.SLEEPONEXIT = 1, the system immediately enters sleep mode when exiting from the lowest priority ISR.

### 3.2.2.2 Exit SLEEP mode

If WFI instruction is used to enter the SLEEP mode, any NVIC interrupts can wake up the device from the SLEEP mode.

If the WFE instruction is used to enter the SLEEP mode, MCU will exit the SLEEP mode immediately when the event occurs. Wake-up events can be generated in the following ways:

- Enable an interrupt in the peripheral control register instead of NVIC, and enable the SCB\_SCR.SEVONPEND. When MCU wakes up by WFE, the peripheral interrupt suspend bit and the peripheral NVIC interrupt channel suspend bit (in NVIC interrupt clear suspend register) must be cleared.
- Configure an external or internal EXTI event mode. When the MCU wakes up, it is not necessary to clear the peripheral interrupt suspend bit and the peripheral NVIC interrupt channel suspend bit (in the NVIC interrupt clear suspend register) because the suspend bit corresponding to the event line is not set. This mode provides the shortest wake-up time because there is no time spent on interrupt entry or exit.

## 3.2.3 STOP mode

STOP mode is based on the Cortex®-M0 deep sleep mode combined with peripheral clock gating. The voltage regulator operates in low power mode. The output voltage can be configured as 1.5V/1.2V. HSE/HSI/PLL is disabled. LSE/LSI can be configured to enable. All GPIO states, 8KB SRAM and all registers retention. All IO, IWDG and PVD can be used to wake up the CPU. Or other peripherals (RTC/LPUART/LPTIM/COMP) can be configured to wake up. After waking up, the HSI is turned on, and the code starts from where it hangs. FLASH is in deep sleep mode.

### 3.2.3.1 Enter STOP mode

When entering the STOP mode, user needs to set SCB\_SCR.SLEEPDEEP = 1 and PWR\_CTRL.PDSTP = 0.

If a FLASH operation is in progress, the time to enter STOP mode will be delayed until the memory access is completed.

If the access to the APB area is in progress, the time to enter the STOP mode will be delayed until the APB access is completed.

In STOP mode, the following peripherals are available:

- Independent Watchdog (IWDG) optional: Once enabled, it will keep counting until a reset is generated.
- RTC optional: It can be enable by RCC\_LSCTRL.RTCEN.
- LPUART/LPTIM/COMP/PVD peripheral can wake up.
- Internal RC oscillator (LSI RC) optional: It can be turned on by RCC\_LSCTRL.LSIEN.
- External 32.768kHz crystal oscillator (LSE OSC) optional: It can be turned on by RCC\_LSCTRL.LSEEN.

ADC should be disabled when entering STOP mode to avoid unnecessary power consumption.

*Note: If the application needs to disable the external clock before entering the stop mode, it must first switch the system clock to HSI and then deassert RCC\_CTRL.HSEEN bit. If RCC\_CTRL.HSEEN bit remains asserted and the external clock (external oscillator) is removed when entering the stop mode, the clock safety system (CSS) function must be enabled to detect any external oscillator failure.*

### 3.2.3.2 Exit STOP mode

When an interrupt or wake-up event wakes up STOP mode, the HSI RC oscillator is selected as the system clock, codes resumed from suspended location. Since the voltage regulator is in low-power mode in STOP mode, it consumes more startup time. In addition, users can configure PWR\_CTRL4.FLASHWKUP = 1 before entering STOP to shorten the wake-up time of FLASH.

## 3.2.4 PD mode

PD (Power Down) mode is based on Cortex®-M0 deep sleep mode, which can achieve lower power consumption. In this mode, the CPU, all peripherals, voltage regulator, HSE/HSI/PLL/LSE/LSI clock sources and all digital power supplies are turned off. Except for NRST/PA0/PC13/PA2, most IO ports output High-Z.

### 3.2.4.1 Enter PD mode

When entering the PD mode, user needs to set SCB\_SCR.SLEEPDEEP = 1 and PWR\_CTRL.PDSTP = 1.

If a FLASH operation is in progress, the time to enter STOP mode will be delayed until the memory access is completed.

If the access to the APB area is in progress, the time to enter the STOP mode will be delayed until the APB access is completed.

### 3.2.4.2 Exit PD mode

MCU exits PD mode when external reset (NRST pin), rising/falling edge of WKUP pin event occurs. And all registers will be reset after waking up.

After waking up from PD mode, code execution is same as power on (boot pin is detected, reset vector initialization, etc.).

## 3.3 Debug support

By default, if the application puts the MCU in STOP or PD mode while using the debug feature, the debug connection is lost. This is due to the Cortex®-M0 core losing its clock.

However, by setting some configuration bits in the DBG\_CTRL register, it is possible to debug the software even when using STOP and PD modes. If these register bits are configured, the voltage regulator and HSI will not be disabled or turned off.

### 3.3.1 Low power mode debug support

When debugging in low-power mode, ensure that the FCLK of the core is turned on to provide the necessary clock for core debugging. Users can debug the MCU in low power mode according to specific operations (guarantee the output of FCLK in low power mode). For specific operations and features, please refer to the description of DBG\_CTRL.PD and DBG\_CTRL.STOP in Chapter 3.4.9.

### 3.3.2 Peripheral debug support

In addition to supporting debug in low power mode, it also supports some peripherals to stop working in debug state (TIM1, TIM3, TIM6, TIM8, LPTIM, I2C1, I2C2, IWDG, WWDG). For specific operations and features, please refer to the description of the other bit fields of the DBG\_CTRL register in Chapter 3.4.9.

## 3.4 PWR registers

### 3.4.1 PWR register overview

Table 3-3 PWR register overview

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	PWR_CTRL	Reserved																		PDR	PLSI[3:0]				PVDEN	CLRBGPDF		CLRWKUPF	PDSTP		Reserved		
	Reset value																					1	0	0	0	0	0	0	0	0	0		
0x04	PWR_CTRLSTS	Reserved																		WKUPOL	Reserved				PVDO	DBGPDF		WKUPF	0		0		
	Reset value																					1	0	0	0	0	0	0	0	0	0		
0x08	PWR_CTRL2	Reserved																		IWDGRSTEN	Reserved												0
	Reset value																					1	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x14	PWR_CTRL3																																	
	Reset value																																	
0x20	PWR_CTRL4																																	
	Reset value																																	
0x24	PWR_CTRL5																																	
	Reset value																																	
0x28	PWR_CTRL6																																	
	Reset value																																	
0x30	DBG_CTRL																																	
	Reset value																																	

### 3.4.2 Power control register (PWR\_CTRL)

Address offset: 0x00

Reset value: 0x0000 0200

31	Reserved																													16	
15	Reserved								PDR		PLS[3:0]			PVDEN		CLR		CLR		PDSTP		Reserved									

Bit field	Name	Description
31:10	Reserved	Reserved, the reset value must be maintained.
9	PDR	Tune VDDD PDR trigger level during STOP mode. 0: VDDD PDR trigger at 1.0V 1: VDDD PDR trigger at 1.2V Only VDDD POR/PDR can reset this bit.
8:5	PLS[3:0]	PVD level selection. PVD threshold is controlled below:

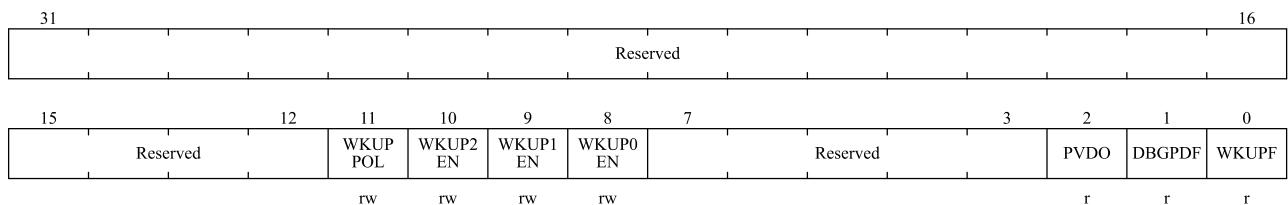
PWR_CTRL.PLS	Voltage
0000	1.8v
0001	2.0v
0010	2.2v
0011	2.4v

Bit field	Name	Description	
		0100	2.6v
		0101	2.8v
		0110	3.0v
		0111	3.2v
		1000	3.4v
		1001	3.6v
		1010	3.8v
		1011	4.0v
		1100	4.2v
		1101	4.6v
		1110	4.8v
		1111	5.0v
4	PVDEN	Enable of Power voltage detector. Software control. 0: PVD disabled 1: PVD enabled	
3	CLRDBGPDF	Clear DBG_PD mode flag. Always read as 0. 0: No effect. 1: Clear PWR_CTRLSTS.DBGPDF bit flag after 2 System clock cycles. (write)	
2	CLRWKUPF	Clear wake up flag. Always read as 0. 0: No effect. 1: Clear PWR_CTRLSTS.WKUPF bit flag after 2 System clock cycles. (write)	
1	PDSTP	Enter STOP/PD mode selection Software will set and clear this bit. 0: Enter STOP mode when CPU enters deep-sleep mode. 1: Enter PD mode when CPU enters deep-sleep mode.	
0	Reserved	Reserved, the reset value must be maintained.	

### 3.4.3 Power control status register (PWR\_CTRLSTS)

Address offset: 0x04

Reset value: 0x0000 0800

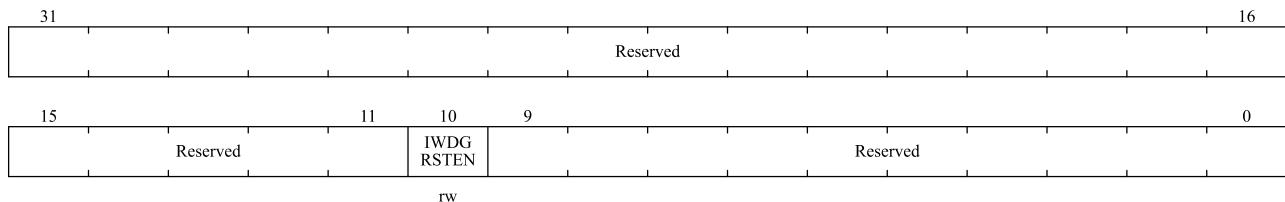


Bit field	Name	Description
31:12	Reserved	Reserved, the reset value must be maintained.
11	WKUPPOL	<p>Wakeup polarity for PA0/PA2/PC13. To wakeup PD mode by using rising edge or falling edge. Make sure disable wakeup enable before changing polarity value.</p> <p>0: Falling edge 1: Rising edge</p>
10	WKUP2EN	<p>Enable PA2_WKUP pin Software can set and clear this bit.</p> <p>0: WKUP pin is used for general purpose I/O. An event on the WKUP pin does not wakeup the device from PD mode.</p> <p>1: WKUP pin is used for wakeup from PD mode. <i>Note: This bit is reset by VDDD POR/PDR Reset only.</i></p>
9	WKUP1EN	<p>Enable PC13_WKUP pin Software can set and clear this bit.</p> <p>0: WKUP pin is used for general purpose I/O. An event on the WKUP pin does not wakeup the device from PD mode.</p> <p>1: WKUP pin is used for wakeup from PD mode. <i>Note: This bit is reset by VDDD POR/PDR Reset only.</i></p>
8	WKUP0EN	<p>Enable PA0_WKUP pin Software can set and clear this bit.</p> <p>0: WKUP pin is used for general purpose I/O. An event on the WKUP pin does not wakeup the device from PD mode.</p> <p>1: WKUP pin is used for wakeup from PD mode. <i>Note: This bit is reset by VDDD POR/PDR Reset only.</i></p>
7:3	Reserved	Reserved, the reset value must be maintained.
2	PVDO	<p>PVD output. Hardware will set and clear this bit. It is valid only if PWR_CTRL.PVDEN = 1.</p> <p>0: VDD/VDDA is higher than the PVD threshold selected with PWR_CTRL.PLS[3:0] 1: VDD/VDDA is lower than the PVD threshold selected with PWR_CTRL.PLS[3:0]</p>
1	DBGPDF	<p>DBGPD mode status bit. When entering DBGPD mode, hardware sets this bit to '1'. Hardware clears this bit when software sets PWR_CTRL.CLRDBGPDF = 1. Only VDDD POR/PDR can reset this bit.</p> <p>0: Chip never entered DBGPD mode 1: Chip has entered DBGPD mode</p>
0	WKUPF	<p>DBGPD mode wake-up status bit. This bit is set by hardware after WKUP pin wakes up DBGPD mode. Hardware clears this bit when software sets PWR_CTRL.CLRWKUPF = 1. Only VDDD POR/PDR can reset this bit.</p> <p>0: No wakeup event occurred 1: A wakeup event was received from the WKUP pin</p>

### 3.4.4 Power control register 2 (PWR\_CTRL2)

Address offset: 0x08

Reset value: 0x0000 0400

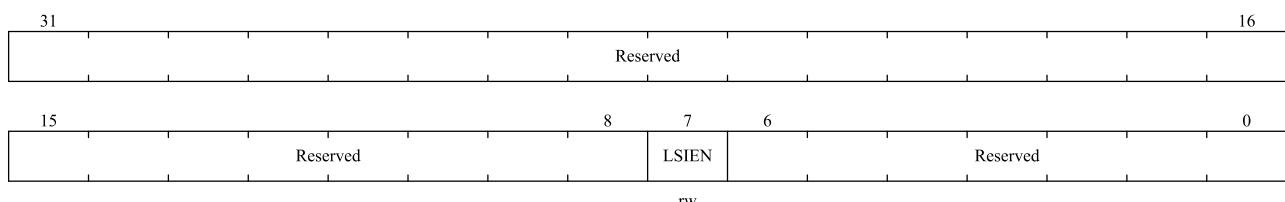


Bit field	Name	Description
31:11	Reserved	Reserved, the reset value must be maintained.
10	IWDGRSTEN	Independent watchdog reset enable. 0: Independent watchdog cannot generate reset to RCC. 1: Independent watchdog can generate reset to RCC.
9:0	Reserved	Reserved, the reset value must be maintained.

### 3.4.5 Power control register 3 (PWR\_CTRL3)

Address offset: 0x14

Reset value: 0x0000 037F



Bit field	Name	Description
31:8	Reserved	Reserved, the reset value must be maintained.
7	LSIEN	Control PWR to enable LSI. 0: PWR continues requesting LSI clock after system enter STOP mode 1: PWR stops requesting LSI clock after system enter STOP mode
6:0	Reserved	Reserved, the reset value must be maintained.

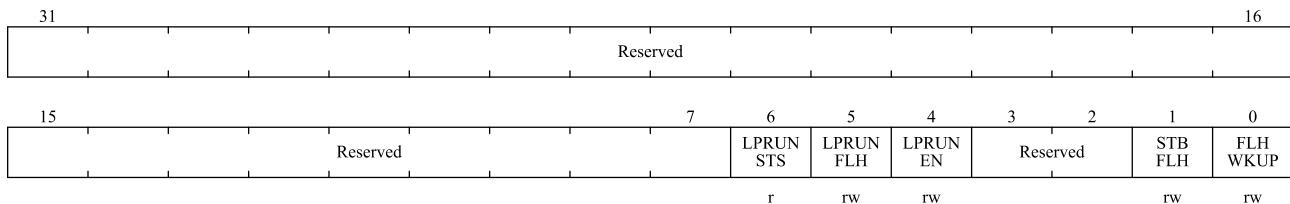
### 3.4.6 Power control register 4 (PWR\_CTRL4)

Address offset: 0x20

Reset value: 0x0000 0020

This register is write protected. Before each software write operation to this register, it must first write the key

0x0175\_3603 to this register to release the write protection.

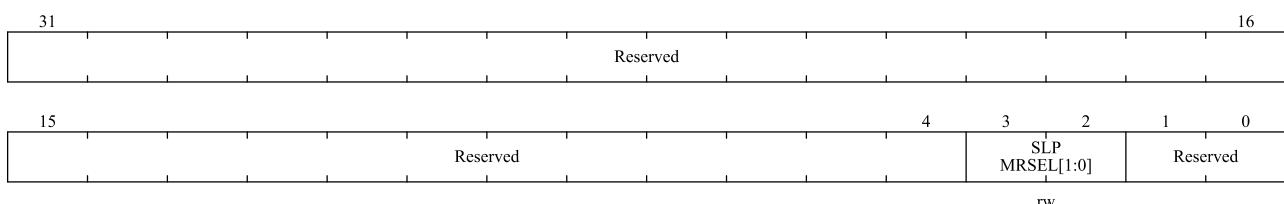


Bit field	Name	Description
31:7	Reserved	Reserved, the reset value must be maintained.
6	LPRUNSTS	LPRUN mode entry status. This bit is set and cleared by hardware. 0: System exited LPRUN mode 1: System in LPRUN mode
5	LPRUNFLH	Flash low-power control for LPRUN mode. 0: Put FLASH to deep standby after chip enters LPRUN mode 1: Keep FLASH to normal working state after chip enters LPRUN mode
4	LPRUNEN	LPRUN mode enable This bit is set and cleared by software and can also be cleared by hardware in STOP and PD modes. 0: System exits LPRUN mode 1: System enters LPRUN mode
3:2	Reserved	Reserved, the reset value must be maintained.
1	STBFLH	FLASH deep standby mode enable (RUN, can be configured in SLEEP mode) This bit is set and cleared by software and can also be cleared by hardware in STOP and PD modes. 0: FLASH back to normal mode 1: FLASH enters deep standby mode
0	FLHWKUP	FLASH fast wakeup enable 0: Disable fast wake-up when system exits from STOP (wake-up time ~10us) 1: Enable fast wake-up when system exits from STOP (wake-up time ~5us)

### 3.4.7 Power control register 5 (PWR\_CTRL5)

Address offset: 0x24

Reset value: 0x0000 0007

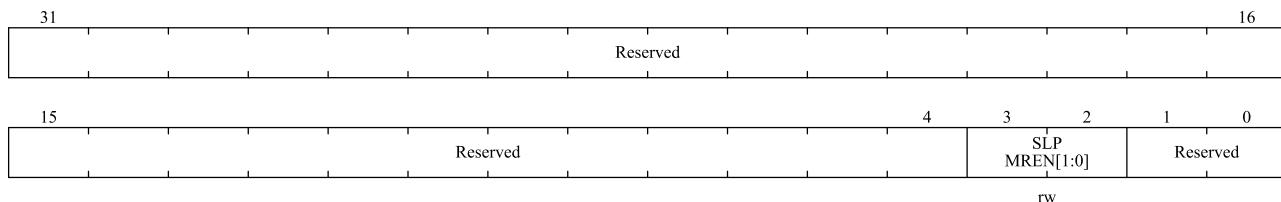


Bit field	Name	Description
31:4	Reserved	Reserved, the reset value must be maintained.
3:2	SLPMRSEL	VDDD output voltage selection after system enters STOP mode. Before configuring these bits, software must first configure PWR_CTRL6.SLPMREN = '11'. 00: Reserved 01: VDDD output voltage is 1.5V 10: Reserved 11: VDDD output voltage is 1.2V Only VDDD POR/PDR can reset this bit.
1:0	Reserved	Reserved, the reset value must be maintained.

### 3.4.8 Power control register 6 (PWR\_CTRL6)

Address offset: 0x28

Reset value: 0x0000 0000



Bit field	Name	Description
31:4	Reserved	Reserved, the reset value must be maintained.
3:2	SLPMREN	VDDD output voltage selection enable 00: After entering STOP mode, VDDD output voltage remains at 1.5V 01: Reserved 10: Reserved 11: After entering STOP mode, VDDD output voltage is controlled by PWR_CTRL5.SLPMRSEL Only VDDD POR/PDR can reset this bit.
1:0	Reserved	Reserved, the reset value must be maintained.

### 3.4.9 Debug control register (DBG\_CTRL)

Address offset: 0x30

Reset value: 0x0000 0000

Only VDDD POR/PDR can reset this register. Only after connecting to the Debugger, software can write access to this register.

31	Reserved										21	20	19	18	17	16
15	14	13	12	11	10	9	8	7			rw	3	rw	2	rw	0
I2C1 TIMOUT	Reserved	TIM3 STP	Reserved	TIM1 STP	WWDG STP	IWDG STP			Reserved			PD	STOP	SLEEP		
rw	rw	rw	rw	rw	rw	rw					rw	rw	rw			

Bit field	Name	Description
31:21	Reserved	Reserved, the reset value must be maintained.
20	TIM8STP	<p>TIM8 stops working when core enters debug state.            This bit is set or cleared by software.            0: The counter of TIM8 works normally            1: The counter of TIM8 stops working</p>
19	Reserved	Reserved, the reset value must be maintained.
18	TIM6STP	<p>TIM6 stops working when core enters debug state.            This bit is set or cleared by software.            0: The counter of TIM6 works normally            1: The counter of TIM6 stops working</p>
17	LPTIMSTP	<p>LPTIM stops working when core enters debug state.            This bit is set or cleared by software.            0: The counter of LPTIM works normally            1: The counter of LPTIM stops working</p>
16	I2C2TIMOUT	<p>I2C2 stops SMBUS timeout mode when core is stopped.            This bit is set or cleared by software.            0: Same as normal mode operation            1: Frozen the timeout control of SMBUS</p>
15	I2C1TIMOUT	<p>I2C1 stops SMBUS timeout mode when core is stopped.            This bit is set or cleared by software.            0: Same as normal mode operation            1: Frozen the timeout control of SMBUS</p>
14:13	Reserved	Reserved, the reset value must be maintained.
12	TIM3STP	<p>TIM3 stops working when core enters debug state.            This bit is set or cleared by software.            0: The counter of TIM3 works normally            1: The counter of TIM3 stops working</p>
11	Reserved	Reserved, the reset value must be maintained.
10	TIM1STP	<p>TIM1 stops working when core enters debug state.            This bit is set or cleared by software.            0: The counter of TIM1 works normally            1: The counter of TIM1 stops working</p>

Bit field	Name	Description
9	WWDGSTP	WWDG stops working when core enters debug state. This bit is set or cleared by software. 0: The counter of WWDG works normally 1: The counter of WWDG stops working
8	IWDGSTP	IWDG stops working when core enters debug state. This bit is set or cleared by software. 0: The counter of IWDG works normally 1: The counter of IWDG stops working
7:3	Reserved	Reserved, the reset value must be maintained.
2	PD	Debug PD mode control. This bit is set or cleared by software. 0: (FCLK off, HCLK off) system enters PD mode, digital circuit part is unpowered. From a software point of view, exiting PD mode is the same as a power-on reset. 1: (FCLK on, HCLK on) system enters DBGPD mode, digital circuit part is powered, and FCLK clock is provided by the internal RC oscillator. In addition, the microcontroller exits DBGPD mode by generating a system reset, which is the same as a system reset.
1	STOP	Debug STOP mode control. This bit is set or cleared by software. 0: (FCLK off, HCLK off) system enters STOP mode, clock controller disables all clocks (including HCLK and FCLK). When exiting STOP mode, the configuration of the clock is the same as after reset (Microcontroller is clocked by the 8MHz internal RC oscillator (HSI)). Therefore, software must reconfigure the clock control system to enable PLL, external oscillator, etc. 1: (FCLK on, HCLK on) system enters DBGSTOP mode, FCLK clock is provided by the internal RC oscillator. When exiting DBGSTOP mode, the software must reconfigure the clock control system to enable PLL, external oscillator, etc. (same operation as when configuring this bit to 0).
0	SLEEP	Debug SLEEP mode. Set or cleared by software. 0: (FCLK on, HCLK off) In SLEEP mode, FCLK is provided by the previously configured system clock, and HCLK is off. Since SLEEP mode does not reset the configured clock system, software does not need to reconfigure the clock system when exiting from SLEEP mode. 1: (FCLK on, HCLK on) In DBGSLEEP mode, both FCLK and HCLK clocks are provided by the previously configured system clock.

## 4 Reset and clock control (RCC)

### 4.1 Reset Control Unit

N32G031 supports the following two types of reset:

- Power Reset
- System Reset

#### 4.1.1 Power reset

A Power reset occurs in the following circumstances:

- Power-on/ Power-down reset (POR/PDR reset).
- When exiting PD mode.

A power reset sets all registers to their reset values (see Figure 3-1).

The reset source will act on the NRST pin and remain low during reset. The reset entry vector is fixed at address 0x0000\_0004. For more details, see Table 6-1 Vector table.

#### 4.1.2 System reset

Except for the following registers, a system reset will reset all registers to their reset states:

- RCC\_CTRL.HSEBP
- RCC\_CTRLSTS
- RCC\_EMCCTRL
- RCC\_LSCTRL.LSEBP
- PWR\_CTRL.STPPLSEN
- PWR\_CTRL.PDR
- PWR\_CTRL5
- PWR\_CTRL6
- DBG\_CTRL

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

- A low level on the NRST pin (external reset)
- Window watchdog end of count condition (WWDG reset)
- Independent watchdog end of count condition (IWDG reset)
- Software reset (SW reset)

- Low power management reset
- MMU protection reset
- RAM parity error reset
- EMC reset

The reset source can be identified by checking the reset flags in the Control/Status Register (RCC\_CTRLSTS).

#### 4.1.2.1 Software reset

A software reset can be generated by setting the SYSRESETREQ bit in Cortex®-M0 Application Interrupt and Reset Control Register. Refer to Cortex®-M0 technical reference manual for further information.

#### 4.1.2.2 Low-power management reset

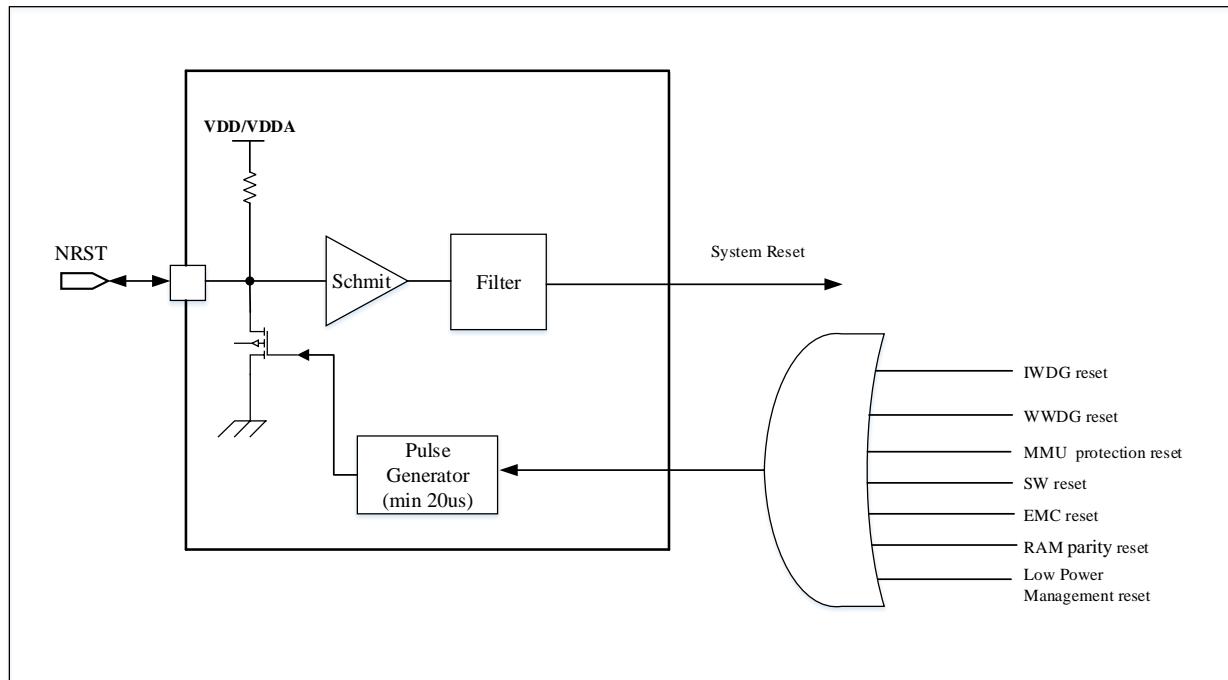
Low-power management reset can be generated by using the following methods:

- Low-power management reset generated when entering PD mode: This reset is enabled by resetting the nRST\_PD bit in User Option Bytes. In this case, whenever a PD mode entry sequence is successfully executed, the system is reset instead of entering PD mode.
- Low-power management reset generated when entering STOP modes: This reset is enabled by resetting the nRST\_STOP bit in User Option Bytes. In this case, whenever a STOP mode entry sequence is successfully executed, the system is reset instead of entering STOP modes.

The system reset signal provided to the chip is output on the NRST pin. The pulse generator guarantees a minimum reset pulse duration of 20μs for each reset source (external or internal). For external reset, the reset pulse is generated while the NRST pin is asserted low.

The Figure below shows the system reset generation circuit.

Figure 4-1 System reset generation



## 4.2 Clock control unit

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

- HSI oscillator clock
- HSE oscillator clock
- PLL clock
- LSI oscillator clock
- LSE oscillator clock

The devices have the following two secondary clock sources:

- LSI: 30 kHz low-speed internal RC can be used to drive independent watchdog (IWDG) and drive RTC,LPTIMER and LPUART through program selection. Used for Auto-wakeup from STOP mode.
- LSE: 32.768 kHz low-speed external crystal can also be used to drive RTC,LPTIMER and LPUART through program selection.

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

The frequency of the AHB, APB (APB1 and APB2) domains can be configured by the user through multiple prescalers. The maximum allowable frequency of the AHB domain, APB1 domain and APB2 domain is 48MHz.

All peripheral clocks are derived from the system clock (SYSCLK) except in the following cases:

- ADC clock is obtained by dividing the AHB/PLL clock.

- LPUART operating clock can come from one of the following six sources, which can be configured by software:
  - ◆ HSI clock
  - ◆ HSE Clock
  - ◆ LSI clock
  - ◆ LSE clock
  - ◆ SYSCLK system clock
  - ◆ APB1 clock (PCLK)
- LPTIMER working clock can come from one of the following six sources, which can be configured by software:
  - ◆ HSI clock
  - ◆ HSE Clock
  - ◆ LSI clock
  - ◆ LSE clock
  - ◆ COMP\_OUT
  - ◆ APB1 clock (PCLK)
- RTCCLK clock source can be provided by HSE/128, LSE or LSI clock.
  - ◆ LSE clock
  - ◆ LSI clock
  - ◆ HSE clock divided by 128
- IWDG clock source is LSI oscillator.
- Flash memory programming interface clock is always the HSI clock

RCC provides external clock for Cortex system timer (SysTick): AHB clock (HCLK) divided by 8. Either the above clock or the Cortex clock (HCLK) to drive the SysTick can be selected by programming the SysTick control and status registers.

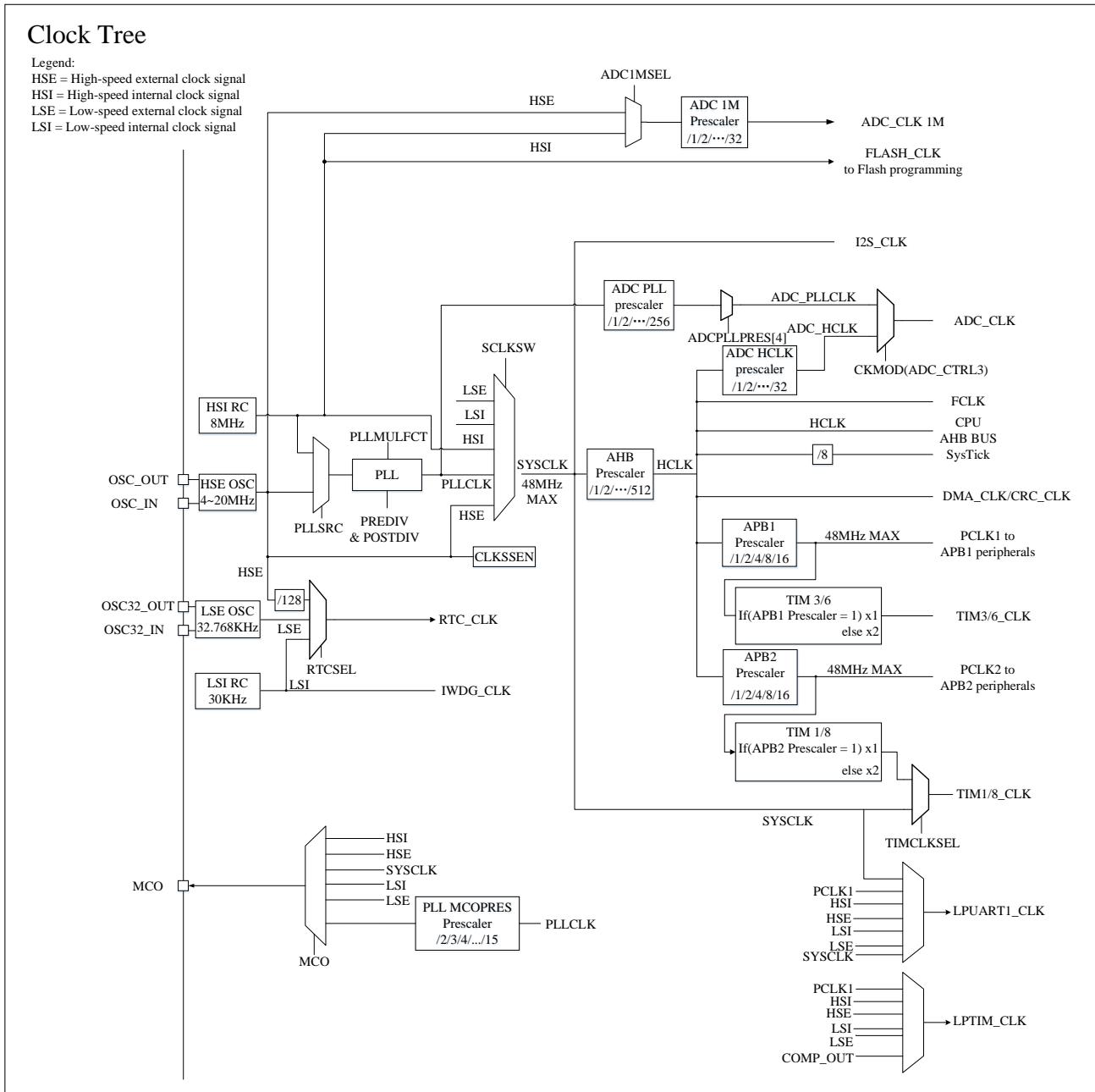
Timer clock frequency distribution is automatically set by hardware in the following 2 cases:

- ◆ If the APB prescaler is 1, the timer clock frequency is the same as the APB frequency where the timer resides.
- ◆ If the APB prescaler is not 1, the timer clock frequency is twice the APB frequency where the timer is located.

FCLK is the free running clock of the Cortex®-M0. See ARM's Cortex®-M0 Technical Reference Manual for details.

## 4.2.1 Clock Tree Diagram

Figure 4-2 Clock Tree



1. The maximum frequency available for the system clock is 48MHz.
2. For more details about the internal and external clock source characteristics, please refer to the "Electrical Characteristics" section in the product datasheet.

## 4.2.2 HSE clock

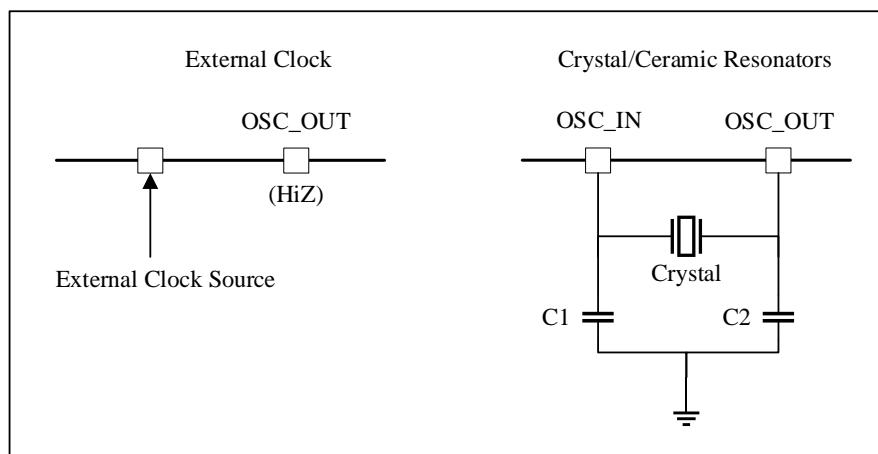
The high-speed external clock signal (HSE) can be generated from the following two clock sources:

- HSE external crystal/ceramic resonator
- HSE user external clock(Input through the PF0 pin)

In HSE bypass mode or crystal mode, RCC\_CTRL.HSEEN needs to be set to 1. If RCC\_CTRL.HSEEN=0, HSE will be turned off.

To reduce distortion of the clock output and shorten the start-up settling time, the crystal/ceramic resonator and load capacitor must be placed as close as possible to the oscillator pins. The load capacitance value must be adjusted according to the chosen oscillator.

Figure 4-3 HSE clock source



### 4.2.2.1 External clock source (HSE bypass)

In this mode, an external clock source must be provided. Its frequency can be up to 20MHz. Users can select this mode by setting the RCC\_CTRL.HSEBP and RCC\_CTRL.HSEEN bits. When PF0 is used as an external clock signal (square wave, sine wave or triangle wave with 50% duty cycle), it must be connected to the OSC\_IN pin, and the OSC\_OUT pin must be floating (Hi-Z). See Figure 4-3.

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

The 4 to 20MHz external oscillator has the advantage of producing a more accurate master clock for the system. The associated hardware configuration is shown in See Figure 4-3. For more details, please refer to the electrical characteristics section of the datasheet.

The RCC\_CTRL.HSERDF bit indicates whether the high-speed external oscillator is stable or not. At startup, the clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the Clock Interrupt Register (RCC\_CLKINT).

HSE clock can be switched on and off by setting the RCC\_CTRL.HSEEN bit.

If the user needs to change the configuration of RCC\_CTRL.HSEBP during operation, it should be configured before enabling RCC\_CTRL.HSEEN.

### 4.2.3 HSI clock

The HSI (High Speed Internal) clock signal is generated by an internal 8MHz RC oscillator and can be directly used as the system clock or PLL input. The HSI RC oscillator can provide a clock source without any external devices. It also has a shorter startup time than the HSE crystal oscillator. However, its frequency is less accurate even with calibration.

RC oscillator frequencies can vary from one chip to another due to manufacturing process variations. Therefore, the HSI clock frequency of the chip has been calibrated to 1% (25°C) before leaving the factory.

If the user application is subject to voltage or temperature variations, this may affect the accuracy of the RC oscillator. The HSI frequency can be trimmed by using the RCC\_CTRL.HSITRIM[4:0] bits.

The RCC\_CTRL.HSIRDF bit flag indicates if the HSI RC oscillator is stable. At startup, the HSI RC output clock is not released until this bit is set by hardware. HSI clock can be switched on and off using the RCC\_CTRL.HSIEN bit.

If the HSE crystal oscillator fails, the HSI clock can be used as a backup source. Refer to section 4.2.8 Clock security system (CLKSS).

### 4.2.4 PLL clock

The internal PLL can be used to multiply the HSI RC output clock or the HSE crystal output clock. Refer to Figure 4-4. The settings of the PLL (select HSI or HSE as the input clock of the PLL, select the multiplier, select the prescaler and the postscaler) must be completed before it is activated. Once the PLL is activated, these parameters cannot be changed. The PLL can be configured using control bits in RCC\_CTRL and RCC\_CFG registers.

When switching the input clock source of the PLL, the original clock source must be turned off after configuring the new clock source (through the clock configuration register bit RCC\_CFG.PLLSRC).

If the PLL interrupt is enabled in the clock interrupt register, an interrupt request can be generated when the PLL is ready.

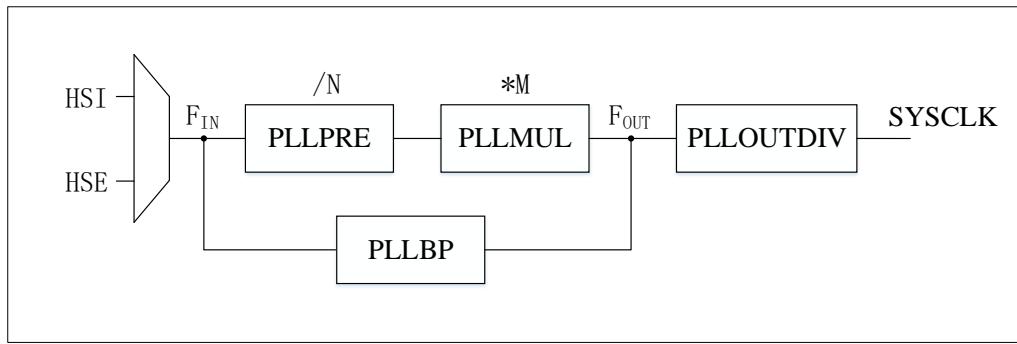
The input frequency  $F_{IN}$  range is 4~20MHz.

PLL VCO( $F_{OUT}$ ) frequency range requires  $48\text{MHz} \leq F_{IN} * M/N < 72\text{MHz}$ .

The system frequency is derived from the PLL VCO frequency divided by the postscale factor, and the software needs to be configured correctly to avoid SYSLCK exceeding 48MHz.

In addition, users can also configure RCC\_CTRL.PLLBP to bypass the PLL frequency division and frequency multiplication function.

Figure 4-4 PLL clock configuration



## 4.2.5 LSE clock

The Low Speed External clock signal (LSE) can be generated from the following two clock sources:

- LSE crystal/ceramic resonator
- LSE external clock(bypass)

### 4.2.5.1 LSE crystal clock source

The LSE crystal is a 32.768 kHz low-speed external crystal or ceramic resonator. It provides a low-power and accurate clock source for real-time clock or other timing functions.

The LSE clock can be turned on and off by setting the RCC\_LSCTRL.LSEEN bit.

The RCC\_LSCTRL.LSERD bit flag indicates if the LSE clock is stable. At startup, the LSE output clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the Clock Interrupt Register (RCC\_CLKINT).

### 4.2.5.2 LSE external clock source

In this mode, an external clock source with a frequency of up to 1 MHz can be provided. Users can select this mode by setting the RCC\_LSCTRL.LSEBP (when RCC\_LSCTRL.LSEEN disable) bits. The external clock signal(Square, Sine or Triangle) with 50% duty cycle must be connected to the OSC32\_IN pin while the OSC32\_OUT pin must be left floating (Hi-Z).

## 4.2.6 LSI clock

The LSI RC can provide clock the IWDG and AWU in STOP mode. The LSI clock frequency is about 30KHz. Please refer to the electrical characteristics section of the data sheet for further information.

The LSI clock can be turned on or off using the RCC\_CTRLSTS.LSIEN bit.

The RCC\_CTRLSTS.LSIRD bit flag indicates if the LSI clock is stable. At startup, the clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the Clock Interrupt Register (RCC\_CLKINT).

## 4.2.7 System clock (SYSCLK) selection

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

A switch from one clock source to another occurs only if the target clock source is ready (after startup delay or PLL locked). When the selected clock source is not ready, the switching of the system clock will not happen until the clock source is ready.

RCC\_CFG.SCLKSW[1:0] are used to select the system clock source. Status bits in RCC\_CTRL and RCC\_LSCTRL indicate which clock is ready, and RCC\_CFG indicates which clock is currently used as the system clock.

## 4.2.8 Clock security system (CLKSS)

Clock security system can be activated by software by setting the RCC\_CTRL.CLKSSEN bit. Once activated, the clock detector is enabled after the startup delay of the HSE oscillator, and disabled when the HSE clock is turned off.

If the HSE clock fails, the HSE oscillator will be automatically turned off, and a clock failure event will be sent to the break input of the advanced timers (TIM1 and TIM8), and the Clock Security System Interrupt CLKSSIF will be generated, allowing the software to execute rescue operations. The CLKSSIF interrupt is connected to the NMI (Non-Maskable Interrupt) interrupt of the Cortex®-M0.

Once the CSS is activated and the HSE clock fails, the CSS interrupt is generated and the NMI is automatically generated. The NMI will be executed continuously until the CSS interrupt pending bit is cleared. Therefore, it is necessary to clear the CSS interrupt by setting the RCC\_CLKINT.CLKSSICLR bit in the NMI handler.

If the HSE oscillator is directly or indirectly used as the system clock (indirectly means: it is used as the PLL input clock, and the PLL clock is used as the system clock), the clock failure will cause a switch of the system clock to the HSI oscillator and the disabling of the external HSE oscillator. If HSE clock (divided or not) is selected as PLL input clock then upon HSE clock failure, the PLL will be turned off.

## 4.2.9 RTC clock

By programming RCC\_LSCTRL.RTCSEL[1:0] bits, the RTCCLK clock source can be either the HSE/128, LSE, or LSI clocks.

## 4.2.10 Watchdog clock

If the independent watchdog has been started by hardware option or software, the LSI oscillator will be forced on and cannot be turned off. The clock is supplied to the IWDG after the LSI oscillator is stable.

## 4.2.11 LPUART clock

In normal working mode, the LPUART clock supports six clock sources: HSI, HSE, LSI, LSE, SYS\_CLK and LPUART\_PCLK. Since HSI, HSE, SYSCLK and PCLK will be turned off in low power mode, software should switch the LPUART clock to LSI or LSE before entering low power mode..

#### 4.2.12 LPTIME clock

In normal working mode, the LPTIME clock supports six clock sources: HSI, HSE, LSI, LSE, PCLK1 and COMP\_OUT. Since HSI, HSE and PCLK will be turned off in low power mode, software should switch the LPTIMER clock to LSI, LSE or COMP\_OUT before entering low power mode.

When HSI, HSE, LSI, LSE, and PCLK1 are switched between each other, software must ensure that both clocks are turned on to switch without glitches.

When switching from HSI, HSE, LSI, LSE, PCLK1 to COMP\_OUT, software configuration is as follows:

1. Make sure the clock is on before switching
  2. Set COMP\_CTRL.EN = 0, make sure COMP off
  3. Set RCC\_APB1PCLKEN.LPTIMEN = 1, turn on LPTIM
  4. Set RCC\_CFG2.LPTIMSEL = 5, select COMP\_OUT as clock source
  5. Set COMP\_CTRL.EN = 1, turn on COMP

#### 4.2.13 Clock output(MCO)

The microcontroller clock output (MCO) capability allows the clock signal to be output onto the external MCO pin.

The corresponding GPIO port register must be configured for the corresponding function. The following six clock signals can be selected as the MCO clock:

- SYSCLK
  - HSI
  - HSE
  - LSI
  - LSE
  - PLL clock division

The selection is controlled by RCC\_CFG.MCO[2:0] bits.

### 4.3 RCC registers

### 4.3.1 RCC register overview

Table 4-1 RCC register overview

		Reset Value					0	PLLSRC	0	1	0				
004h	RCC_CFG		MCORES [3:0]		MCO [2:0]		0	PLLOUTDIV [1:0]	PLLPRE [1:0]	PLLMULFCT [3:0]	SCLKSTS2	AHBPRES [3:0]			SCLKSW [2:0]
008h	RCC_CLKINT	Reset Value	0	0	1	0	0	0	0	0	0	PERRCLR	0	0	
00Ch	RCC_APB2PRST		Reserved				0	CLKSSICLR	PLLRCR	0	0	PLRIDICLR	0	0	
010h	RCC_APB1PRST	Reset Value	Reserved				0	PWRSTF	0	0	0	HSERDICLR	0	0	
014h	RCC_AHBCLKEN		Reserved				0	OPAEN	12C2RST	12C1RST	0	LPUART1ST	0	0	
018h	RCC_APB2PCLKEN	Reset Value	Reserved				0	PWREN	0	0	0	USART2RST	0	0	
01Ch	RCC_APB1PCLKEN	Reset Value	Reserved				0	0	12C2EN	12C1EN	0	LPUARTEN	0	0	
020h	RCC_LSCTRL		Reserved				0	0	0	0	0	USART1EN	0	0	
024h	RCC_CTRLSTS	Reset Value	Reserved				0	0	0	0	0	ADCEN	0	0	
028h	RCC_AHBPRT	Reset Value	Reserved				0	0	0	0	0	WWDGRT	0	0	
02Ch	RCC_CFG2	0	TIMCLK	Reserved	LPUARTSEL [2:0]		Reserved	LPTIMSEL [2:0]		Reserved	ADC1IMPRES [4:0]				
030h	RCC_EMCTRL	Reset Value	Reserved				0	0	0	0	0	EMCCLPRSTF	0	0	
					GBRST3	0	GBRST2	0	GBRST1	0	0	EMCGBNRSTF	0	0	
					GBRST0	0	GBRST3	0	GBRST1	0	0	LPUNCLKSEL	0	0	
					GBNRST0	0	GBNRST3	0	GBNRST2	0	0	LPWRSTF	0	0	
					GBNRST1	0	GBNRST1	0	GBNRST0	0	0	RTCSEL	0	0	
					CLPRST3	0	CLPRST2	0	CLPRST1	1	0	HDV_RST	0	0	
					CLPRST0	0	CLPRST2	0	CLPRST1	1	0	IWDGRSTF	0	0	
					GBDET3	0	GBDET2	0	GBDET1	0	0	ADCIMSEL	0	0	
					GBDET0	0	GBDET3	0	GBDET2	0	0	ADCIMSEL	0	0	
					HSQRTRST	0	HSQRTRST	0	HSQRTRST	0	0	BEEPFEN	0	0	
					PORRSTF	0	PORRSTF	0	PORRSTF	0	0	TOPCEN	0	0	
					FLITEN	0	FLITEN	0	FLITEN	0	0	LPTIMEN	0	0	
					TOPBEN	0	TOPBEN	0	TOPBEN	0	0	LPTIM1ST	0	0	
					SRAVEN	1	SRAVEN	1	SRAVEN	1	0	SRAMEN	0	0	
					MMURSTF	0	MMURSTF	0	MMURSTF	0	0	LPITIMECLKEN	0	0	
					RAMRSTF	0	RAMRSTF	0	RAMRSTF	0	0	TIME3EN	0	0	
					RMRSTF	0	RMRSTF	0	RMRSTF	0	0	DMAEN	0	0	

### 4.3.2 Clock control register (RCC\_CTRL)

Address offset: 0x00

Reset value: 0x0080 0083

31	Reserved	26	PLL RDF	25	PLLEN	24	PLL OUT EN	23	PLL BP	22	Reserved	21	CLK SSEN	20	HSE BP	19	HSE RD F	18	HSE EN	17		16
15			r		rw	8		rw	7		rw				rw	3	rw	2	r	1	rw	0
	Reserved															HSITRIM[4:0]			Reserved	HSIRDF	HSIEN	

Bit field	Name	Description
31:26	Reserved	Reserved, the reset value must be maintained
25	PLL RDF	PLL clock ready flag Set by hardware once PLL is ready. 0: PLL is not ready 1: PLL is ready
24	PLLEN	PLL enable Set and cleared by software. When entering the LPRUN,STOP or PD mode, it is cleared by hardware. This bit cannot be cleared when PLL is used as the system clock. 0: Disable PLL 1: Enable PLL
23	PLL OUTEN	PLL clock output enable bit. 0: PLL clock output is disabled 1: PLL clock output is enabled
22	PLL BP	PLL bypass mode 0: $F_{OUT} = F_{IN} * M/N$ (PLL VCO frequency) 1: $F_{OUT} = F_{IN}$ (bypass output)
21:20	Reserved	Reserved, the reset value must be maintained.
19	CLK SSEN	Clock security system enable Set and cleared by software. 0: Disable the clock detector 1: Enable the clock detector if the HSE oscillator is ready
18	HSE BP	External high-speed clock bypass enable Set and cleared by software. This bit can only be written when the HSE oscillator is disabled. 0: Disable the bypass function of HSE oscillator 1: Enable the bypass function of HSE oscillator
17	HSERDF	External high-speed clock ready flag Set by hardware once HSE is ready. This bit is cleared 6 HSE clock cycles after the

Bit field	Name	Description
		HSEEN bit is cleared. 0: HSE is not ready 1: HSE is ready
16	HSEEN	External high-speed clock enable Set and cleared by software. When entering the LPRUN,STOP or PD mode, it is cleared by hardware. This bit cannot be cleared when HSE is used as the system clock. 0: Disable HSE oscillator 1: Enable HSE oscillator
15:8	Reserved	Reserved, the reset value must be maintained.
7:3	HSITRIM[4:0]	Internal high-speed clock correction value written by software, used to calibrate the frequency of the internal HSI RC oscillator. Fine adjustment: current value + (measured frequency-target frequency:8M) / (8M * 0.33%) integer; The default value is 16, and the HSI can be adjusted to 8MHz±0.3%. According to application to obtain higher accuracy;
2	Reserved	Reserved, the reset value must be maintained.
1	HSIRDF	Internal high-speed clock ready flag Set by hardware once HSI is stable. After the HSIEN bit is cleared, it takes 6 internal 8 MHz oscillator clock cycles to go low.
0	HSIEN	Internal high-speed clock enable Set and cleared by software. This bit cannot be cleared when HSI is used as the system clock. When returning from LPRUN,STOP or PD mode or HSE failure occurs, set by hardware to enable the HSI oscillator. 0: Disable HSI oscillator 1: Enable HSI oscillator

### 4.3.3 Clock configuration register (RCC\_CFG)

Address offset: 0x04

Reset value: 0x2000 0000

31	MCOPRES[3:0]	28	MCO[2:0]	27	PLLSRC	25	PLLOUTDIV[1:0]	24	PLLPRE[1:0]	23	PLLMULFCT[3:0]	22		21		20		19		16
15	rw	14	rw	13	rw	11	rw	10	rw	8	rw	7	rw	4	rw	3	rw	2	rw	0
SCLKSTS[2:1]	APB2PRES[2:0]		rw		rw		rw		rw		rw		rw		rw		rw		rw	

Bit field	Name	Description
31:28	MCOPRES[3:0]	MCO prescaler Set and cleared by software.

Bit field	Name	Description
		0010: PLL clock divided by 2 0011: PLL clock divided by 3 0100: PLL clock divided by 4 0101: PLL clock divided by 5 0110: PLL clock divided by 6 0111: PLL clock divided by 7 1000: PLL clock divided by 8 1001: PLL clock divided by 9 1010: PLL clock divided by 10 1011: PLL clock divided by 11 1100: PLL clock divided by 12 1101: PLL clock divided by 13 1110: PLL clock divided by 14 1111: PLL clock divided by 15 Other values: not allowed
27:25	MCO[2:0]	Microcontroller clock output selection Set and cleared by software. 000: no clock 001: LSI clock 010: LSE clock 011: System clock (SYSCLK) 100: HSI clock 101: HSE clock 110: Clock after PLL frequency division <p><i>Note: This clock output may be truncated at startup or during MCO clock source switching. When the system clock is selected to output to the MCO pin, the output clock frequency must not exceed the maximum I/O speed (For details of the maximum frequency of the I/O port, see the data sheet).</i></p>
24	PLLSRC	PLL clock source. Set or cleared by software, this bit can be written only when the PLL is turned off. 0: HSI clock is used as PLL input clock 1: HSE clock is used as PLL input clock
23:22	PLLOUTDIV[1:0]	The PLL outputs the clock division value. Set and clear through software. 00: no frequency division 01: divided by 2 10: divided by 3 11: divided by 4
21:20	PLLPRE[1:0]	PLL prescaler $4\text{MHz} \leq F_{IN}/N \leq 20\text{MHz}$ This bit can only be written when the PLL is off

Bit field	Name	Description
		00: PLL input clock divided by 1 01: PLL input clock divided by 2 10: PLL input clock divided by 3 11: PLL input clock divided by 4
19:16	PLLMULFCT[3:0]	PLL multiplication factor. Set and clear through software. This bit can only be written when the PLL is off. $F_{OUT} = F_{IN} * M / N$ The actual PLL M value should be this register value + 3, $M = PLLMULFCT + 3$ . 0: $M = 3$ 1: $M = 4$ 2: $M = 5$ ... 15: $M = 18$
15:14	SCLKSTS2[1:0]	Use with SCLKSTS bit
13:11	APB2PRES[2:0]	APB high-speed (APB2) prescaler Set and cleared by software to configure the division factor of APB2 clock (PCLK2). Make sure that PCLK2 does not exceed 48MHz. 0xx: HCLK not divided 100: HCLK divided by 2 101: HCLK divided by 4 110: HCLK divided by 8 111: HCLK divided by 16
10:8	APB1PRES[2:0]	APB low-speed (APB1) prescaler Set and cleared by software to configure the division factor of the APB1 clock (PCLK1). Make sure that PCLK1 does not exceed 48MHz. 0xx: HCLK not divided 100: HCLK divided by 2 101: HCLK divided by 4 110: HCLK divided by 8 111: HCLK divided by 16
7:4	AHBPRES[3:0]	AHB prescaler Set and cleared by software to configure the division factor of the AHB clock (HCLK). 0xx: SYSCLK not divided 1000: SYSCLK divided by 2 1001: SYSCLK divided by 4 1010: SYSCLK divided by 8 1011: SYSCLK divided by 16 1100: SYSCLK divided by 64 1101: SYSCLK divided by 128 1110: SYSCLK divided by 256

Bit field	Name	Description
		1111: SYSCLK divided by 512
3	SCLKSTS	<p>System clock switch status, used together with SCLKSTS2[1:0] bit</p> <p>Set and cleared by hardware to indicate which clock source is used as system clock</p> <p>000: HSI oscillator used as system clock</p> <p>001: HSE oscillator used as system clock</p> <p>010: PLL used as system clock</p> <p>011: LSE used as system clock</p> <p>100: LSI used as system clock</p>
2:0	SCLKSW[2:0]	<p>System clock switch</p> <p>Set and cleared by software to select the system clock source.</p> <p>Set by hardware to force HSI selection when exiting from the STOP mode, or when the HSE oscillator fails and CLKSSEN is enabled.</p> <p>000: HSI selected as the system clock</p> <p>001: HSE selected as the system clock</p> <p>010: PLL selected as the system clock</p> <p>011: LSE selected as the system clock</p> <p>100: LSI selected as the system clock</p>

#### 4.3.4 Clock interrupt register (RCC\_CLKINT)

Address offset: 0x08

Reset value: 0x0000 0000

31	Reserved										24	23	22	21	20	19	18	17	16
											CLKSSI CLR	Reserved	PERR CLR	PLLRDI CLR	HSERDI CLR	HSIRDI CLR	LSERDI CLR	LSIRDI CLR	
15	14	13	12	11	10	9	8	w 7	w 6	w 5	w 4	w 3	w 2	w 1	w 0				
Reserved	RAMC ERRRST	RAMC ERRIEN	PLLRDI EN	HSERDI EN	HSIRDI EN	LSERDI EN	LSIRDI EN	CLKSSIF	Reserved	RAMCPIF	PLLRDIF	HSERDIF	HSIRDIF	LSERDIF	LSIRDIF				
	rw	rw	rw	rw	rw	rw	rw	r		r	r	r	r	r	r				

Bit field	Name	Description
31:24	Reserved	Reserved, the reset value must be maintained.
23	CLKSSICLR	<p>Clock security system interrupt clear</p> <p>Set by the software to clear the CLKSSIF flag.</p> <p>0: No effect</p> <p>1: Clear the CLKSSIF flag</p>
22	Reserved	Reserved, the reset value must be maintained.
21	PERRCLR	<p>PERRCLR: Clears PERR interrupts.</p> <p>This bit is set by the software to clear PERRF.</p> <p>0: No impact.</p> <p>1: PERRF cleared</p>
20	PLLRDICL	PLL ready interrupt clear

Bit field	Name	Description
		<p>Set by the software to clear the PLLRDIF flag.</p> <p>0: No effect</p> <p>1: Clear the PLLRDIF flag</p>
19	HSERDICLR	<p>HSE ready interrupt clear</p> <p>Set by the software to clear the HSERDIF flag.</p> <p>0: Not used</p> <p>1: Clear HSERDIF flag</p>
18	HSIRDICLR	<p>HSI ready interrupt clear</p> <p>Set by the software to clear the HSIRDIF flag.</p> <p>0: Not used</p> <p>1: Clear the HSIRDIF flag</p>
17	LSERDICLR	<p>LSE ready interrupt clear</p> <p>Set by the software to clear the LSERDIF flag.</p> <p>0: Not used</p> <p>1: Clear LSERDIF flag</p>
16	LSIRDICLR	<p>LSI ready interrupt clear</p> <p>Set by software to clear the LSIRDIF flag.</p> <p>0: Not used</p> <p>1: Clear the LSIRDIF flag</p>
15	Reserved	Reserved, the reset value must be maintained.
14	RAMCERRRST	<p>RAMC parity error reset enabled</p> <p>1: RAMC generates a reset when it detects a parity error</p> <p>0: The reset is not generated when RAMC detects a parity error</p>
13	RAMCERRIEN	<p>Enable RAMC parity error interrupt</p> <p>1: Interrupts when RAMC detects a parity error</p> <p>0: No interrupt is generated when RAMC detects a parity error</p>
12	PLLRDIEN	<p>PLL ready interrupt enable</p> <p>Set and cleared by software to enable and disable PLL ready interrupt</p> <p>0: Disable PLL ready interrupt</p> <p>1: Enable PLL ready interrupt</p>
11	HSERDIEN	<p>HSE ready interrupt enable</p> <p>Set and cleared by software to enable and disable HSE ready interrupt.</p> <p>0: Disable HSE ready interrupt</p> <p>1: Enable HSE Ready Interrupt</p>
10	HSIRDIEN	<p>HSI ready interrupt enable</p> <p>Set and cleared by software to enable and disable HSI ready interrupt.</p> <p>0: Disable HSI ready interrupt</p> <p>1: Enable HSI ready interrupt</p>
9	LSERDIEN	<p>LSE ready interrupt enable</p> <p>Set and cleared by software to enable and disable LSE ready interrupt.</p> <p>0: Disable LSE ready interrupt</p>

Bit field	Name	Description
		1: Enable LSE ready interrupt
8	LSIRDIEN	<p>LSI ready interrupt enable Set and cleared by software to enable and disable LSI ready interrupt.</p> <p>0: Disable LSI ready interrupt 1: Enable LSI ready interrupt</p>
7	CLKSSIF	<p>Clock security system interrupt flag Set by hardware when a failure is detected in the external HSE oscillator.</p> <p>0: No clock security system interrupt caused by HSE clock failure 1: Clock security system interrupt caused by HSE clock failure</p>
6	Reserved	Reserved, the reset value must be maintained.
5	RAMCPIF	<p>RAMC parity interrupt status. Set by hardware, set by software to clear PERRCLR</p> <p>1: RAMC parity error occurs 0: No RAMC parity error occurs</p>
4	PLLRDIF	<p>PLL ready interrupt flag This bit is set by hardware when PLLRDIEN is set and PLL clock is ready. This bit is cleared by software by setting the PLLRDICLRL bit.</p> <p>0: No clock ready interrupt caused by PLL lock 1: Clock ready interrupt caused by PLL lock</p>
3	HSERDIF	<p>HSE ready interrupt flag Set by hardware when HSERDIEN is set and the HSE clock is ready. This bit is cleared by software by setting the HSERDICLRL bit.</p> <p>0: No clock ready interrupt caused by HSE oscillator 1: Clock ready interrupt caused by HSE oscillator</p>
2	HSIRDIF	<p>HSI ready interrupt flag Set by hardware when HSIRDIE is set and the HSI clock is ready. This bit is cleared by software by setting the HSERDICLRL bit.</p> <p>0: No clock ready interrupt caused by HSI oscillator 1: Clock ready interrupt caused by HSI oscillator</p>
1	LSERDIF	<p>LSE ready interrupt flag Set by hardware when LSERDIEN is set and the LSE clock is ready. This bit is cleared by the software by setting the LSERDICLRL bit.</p> <p>0: No clock ready interrupt caused by LSE oscillator 1: Clock ready interrupt caused by LSE oscillator</p>
0	LSIRDIF	<p>LSI ready interrupt flag Set by the hardware when LSIRDIE is set and the LSI clock is ready. This bit is cleared by software by setting the LSERDICLRL bit.</p> <p>0: No clock ready interrupt caused by LSI oscillator 1: Clock ready interrupt caused by LSI oscillator</p>

### 4.3.5 APB2 peripheral reset register (RCC\_APB2PRST)

Address offset: 0x0c

Reset value: 0x0000 0000

RCC_APB2PRST Register Map															
31	Reserved														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	USART1 RST	TIM8RST	TIM1RST	Reserved	SPI2RST	SPI1RST	Reserved	IOPFRST	Reserved	IOPCRST	IOPBRST	IOPARST	Reserved	AFIGRST	rw

Bit field	Name	Description
31:15	Reserved	Reserved, the reset value must be maintained.
14	USART1RST	USART1 reset Set and cleared by software. 0: Clear reset 1: Reset USART1
13	TIM8RST	TIM8 reset Set and cleared by software. 0: Clear reset 1: Reset TIM8
12	TIM1RST	TIM1 reset Set and cleared by software. 0: Clear reset 1: Resets TIM1
11	Reserved	Reserved, the reset value must be maintained.
10	SPI2RST	SPI2 reset Set and cleared by software. 0: Clear reset 1: Reset SPI2
9	SPI1RST	SPI1 reset Set and cleared by software. 0: Clear reset 1: Reset SPI1
8	Reserved	Reserved, the reset value must be maintained.
7	IOPFRST	GPIO port F reset Set and cleared by software. 0: Clear reset 1: Reset GPIO port F
6:5	Reserved	Reserved, the reset value must be maintained.
4	IOPCRST	GPIO port C reset

Bit field	Name	Description
		Set and cleared by software. 0: Clear reset 1: Reset GPIO port C
3	IOPBRST	GPIO port B reset Set and cleared by software. 0: Clear reset 1: Reset GPIO port B
2	IOPARST	GPIO port A reset Set and cleared by software. 0: Clear reset 1: Reset GPIO port A
1	Reserved	Reserved, the reset value must be maintained.
0	AFIORST	Alternate function IO reset Set and cleared by software. 0: Clear reset 1: Reset alternate function IO

### 4.3.6 APB1 peripheral reset register (RCC\_APB1PRST)

Address offset: 0x10

Reset value: 0x0000 0000

31	Reserved	PWRRST	28	27	Reserved	23	22	I2C2RST	I2C1RST	21	20	19	18	17	16
15	rw	12	11	10		rw	6	rw	5	4	3	rw	2	rw	1
	Reserved	WWDG RST			Reserved		BEEPRST	TIM6RST	LPTIMRST	Reserved	TIM3RST	Reserved			0
		rw					rw		rw		rw		rw		rw

Bit field	Name	Description
31:29	Reserved	Reserved, the reset value must be maintained
28	PWRRST	Power interface reset Set and cleared by software. 0: Clear reset 1: Reset the power interface
27:23	Reserved	Reserved, the reset value must be maintained
22	I2C2RST	I2C2 reset Set and cleared by software. 0: Clear reset 1: Reset I2C2
21	I2C1RST	I2C1 reset Set and cleared by software.

Bit field	Name	Description
		0: Clear reset 1: Reset I2C1
20:19	Reserved	Reserved, the reset value must be maintained.
18	LPUARTRST	LPUART reset Set and cleared by software. 0: Clear reset 1: Reset LPUART
17	USART2RST	USART2 reset Set and cleared by software. 0: Clear reset 1: Reset USART2
16:12	Reserved	Reserved, the reset value must be maintained
11	WWDGRST	Window watchdog reset Set and cleared by software. 0: Clear reset 1: Reset window watchdog
10:6	Reserved	Reserved, the reset value must be maintained
5	BEEPRST	BEEPER reset Set and cleared by software. 0: Clear reset 1: Reset BEEPER
4	TIM6RST	<b>4.3.6.1 TIM6 timer reset</b>  Set and cleared by software. 0: Clear reset 1: Reset TIM6 timer
3	LPTIMRST	LPTIMR timer reset Set and cleared by software. 0: Clear reset 1: Reset LPTIM
2	Reserved	Reserved, the reset value must be maintained
1	TIM3RST	TIM3 timer reset Set and cleared by software. 0: Clear reset 1: Reset TIM3 timer
0	Reserved	Reserved, the reset value must be maintained.

### 4.3.7 AHB peripheral clock enable register (RCC\_AHBPCLKEN)

Address offset: 0x14

Reset value: 0x0000 0014

31													16
Reserved													
15	13	12	11	10	8	7	6	5	4	3	2	1	0
Reserved		ADCEN		Reserved		HDIVEN	CRCEN	HSQRTEN	FLITFEN	Reserved	SRAMEN	Reserved	DMAEN
		rw				rw	rw	rw	rw		rw		rw

Bit field	Name	Description
31:13	Reserved	Reserved, the reset value must be maintained.
12	ADCEN	ADC clock enable Set and cleared by software. 0: Disable ADC clock 1: Enable ADC clock
11:8	Reserved	Reserved, the reset value must be maintained.
7	HDIVEN	HDIV clock enable Set and cleared by software. 0: Disable the HDIV clock 1: Enable the HDIV clock.
6	CRCEN	CRC clock enable Set and cleared by software. 0: Disable CRC clock 1: Enable CRC clock
5	HSQRTEN	HSQRT clock enable Set and cleared by software. 0: Disable HSQRT clock 1: Enable HSQRT clock
4	FLITFEN	Flash interface clock enable Set and cleared by software. 0: Disable the flash interface clock 1: Enable the flash interface clock
3	Reserved	Reserved, the reset value must be maintained.
2	SRAMEN	SRAM clock enable Set and cleared by software. 0: SRAM clock disabled in sleep mode 1: SRAM clock enabled in sleep mode
1	Reserved	Reserved, the reset value must be maintained.
0	DMAEN	DMA clock enable Set and cleared by software. 0: Disable DMA clock 1: Enable DMA clock

### 4.3.8 APB2 peripheral clock enable register (RCC\_APB2PCLKEN)

Address offset: 0x18

Reset value: 0x0000 0000

RCC_APB2PCLKEN Register																16
31	Reserved															16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved	USART1EN	TIM8EN	TIM1EN	Reserved	SPI2EN	SPI1EN	Reserved	IOPFEN	Reserved	IOPCEN	IOPBEN	IOPAEN	Reserved	AFIOEN		RW

Bit field	Name	Description
31:15	Reserved	Reserved, the reset value must be maintained.
14	USART1EN	USART1 clock enable Set and cleared by software. 0: Disable USART1 clock 1: Enable USART1 clock
13	TIM8EN	TIM8 Clock Enable Set and cleared by software. 0: Disable TIM8 clock 1: Enable TIM8 clock
12	TIM1EN	TIM1 clock enable Set and cleared by software. 0: Disable TIM1 clock 1: Enable TIM1 clock
11	Reserved	Reserved, the reset value must be maintained.
10	SPI2EN	SPI2 clock enable Set and cleared by software. 0: Disable SPI2 clock 1: Enable SPI2 clock
9	SPI1EN	SPI1 clock enable Set and cleared by software. 0: Disable SPI1 clock 1: Enable SPI1 clock
8	Reserved	Reserved, the reset value must be maintained.
7	IOPFEN	GPIO port F clock enable Set and cleared by software. 0: Disable the clock of GPIO port F 1: Enable the clock of GPIO port F
6:5	Reserved	Reserved, the reset value must be maintained.
4	IOPCEN	GPIO port C clock enable

Bit field	Name	Description
		Set and cleared by software. 0: Disable the clock of GPIO port C 1: Enable the clock of GPIO port C
3	IOPBEN	GPIO port B clock enable Set and cleared by software. 0: Disable the clock of GPIO port B 1: Enable the clock of GPIO port B
2	IOPAEN	GPIO port A clock enable Set and cleared by software. 0: Disable the clock of GPIO port A 1: Enable the clock of GPIO port A
1	Reserved	Reserved, the reset value must be maintained.
0	AFIOEN	Alternate function IO clock enable Set and cleared by software. 0: Disable the alternate function IO clock 1: Enable the alternate function IO clock

### 4.3.9 APB1 peripheral clock enable register (RCC\_APB1PCLKEN)

Address offset: 0x1c

Reset value: 0x0000 0000

31	30	29	28	27	Reserved				23	22	21	20	19	18	17	16
OPAMP EN	Reserved	PWREN							I2C2EN	I2C1EN	Reserved	Reserved	LPUART EN	USART2 EN	Reserved	
rw 15		rw 12	rw 11	rw 10	rw 9	rw 8	rw 7	rw 6	rw 5	rw 4	rw 3	rw 2	rw 1	rw 0		
	Reserved		WWDG EN	Reserved	COMP FILTEN	COMPEN	Reserved	BEEPEN	TIM6EN	LPTIM EN	LPTIM PCLKEN	TIM3EN	Reserved			
			rw		rw		rw		rw	rw	rw	rw	rw	rw		

Bit field	Name	Description
31	OPAMPEN	OPAMP clock enable Set and cleared by software. 0: Disable OPAMP clock 1: Enable OPAMP clock
30:29	Reserved	Reserved, the reset value must be maintained
28	PWREN	Power interface clock enable Set and cleared by software. 0: Disable the power interface clock 1: Enable the power interface clock
27:23	Reserved	Reserved, the reset value must be maintained
22	I2C2EN	I2C2 clock enable Set and cleared by software.

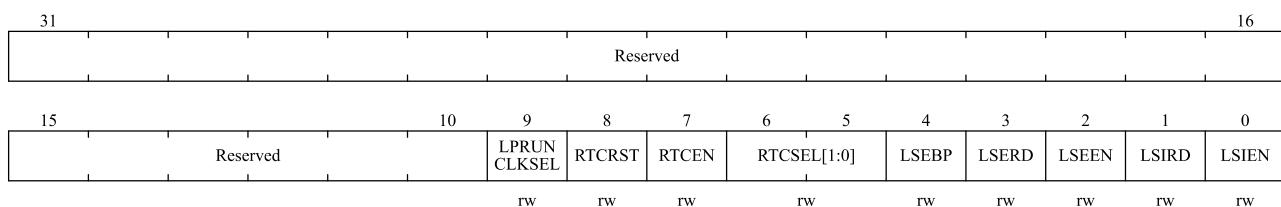
Bit field	Name	Description
		0: Disable I2C2 clock 1: Enable I2C2 clock
21	I2C1EN	I2C1 clock enable Set and cleared by software. 0: Disable I2C1 clock 1: Enable I2C1 clock
20:19	Reserved	Reserved, the reset value must be maintained
18	LPUARTEN	LPUART clock enable Set and cleared by software. 0: Disable LPUART clock 1: Enable LPUART clock
17	USART2EN	USART2 clock enable Set and cleared by software. 0: Disable USART2 clock 1: Enable USART2 clock
16:12	Reserved	Reserved, the reset value must be maintained
11	WWDGEN	Window watchdog clock enable Set and cleared by software. 0: Disable WWDG clock 1: Enable WWDG clock
10	Reserved	Reserved, the reset value must be maintained
9	COMPFILTEN	Comparator filter clock enable 0: Disable the comparator filter clock 1: Enable the comparator filter clock
8	COMPEN	Comparator clock enable 0: Disable the comparator clock 1: Enable the comparator clock
7:6	Reserved	Reserved, the reset value must be maintained
5	BEEPEN	BEEPER clock enable Set and cleared by software. 0: Disable BEEPER clock 1: Enable BEEPER clock
4	TIM6EN	TIM6 timer clock enable Set and cleared by software. 0: Disable TIM6 timer clock 1: Enable TIM6 timer clock
3	LPTIMEN	LPTIM timer clock enable Set and cleared by software. 0: Disable LPTIM timer clock 1: Enable LPTIM timer clock
2	LPTIMPCLKEN	LPTIM APB1 interface clock enable

Bit field	Name	Description
		<p>It needs to be enabled only when LPTIM selects APB1 as the timer clock source.</p> <p>When LPTIM selects other clock sources, this clock is turned off to save power.</p> <p>Set or cleared by software.</p> <p>0: Disable the LPTIM APB1 interface clock</p> <p>1: Enable the LPTIM APB1 interface clock</p>
1	TIM3EN	<p>TIM3 timer clock enable</p> <p>Set and cleared by software.</p> <p>0: Disable TIM3 timer clock</p> <p>1: Enable TIM3 timer clock</p>
0	Reserved	Reserved, the reset value must be maintained

#### 4.3.10 Low speed clock control register (RCC\_LSCTRL)

Address offset: 0x20

Reset value: 0x0000 0003



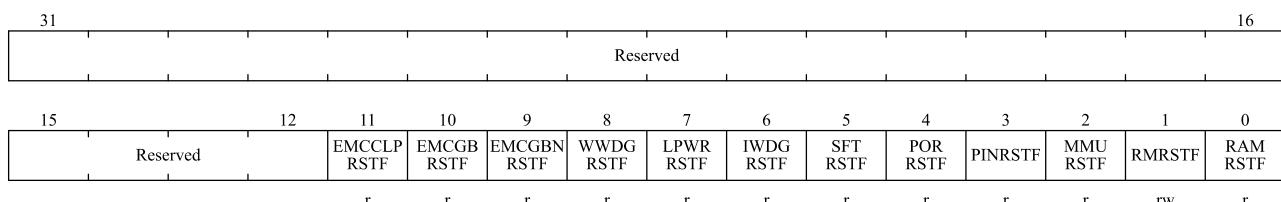
Bit field	Name	Description
31:10	Reserved	Reserved, the reset value must be maintained
9	LPRUNCLKSEL	<p>LPRUN clock selection</p> <p>Set and clear by software</p> <p>0: LPRUN mode selects LSI clock</p> <p>1: LPRUN mode selects LSE clock</p>
8	RTCRST	<p>RTC software reset</p> <p>0: Clear reset</p> <p>1: Reset RTC</p>
7	RTCEN	<p>RTC clock enable</p> <p>Set and clear by software</p> <p>0: RTC clock disabled</p> <p>1: RTC clock is enabled</p>
6:5	RTCSEL[1:0]	<p>RTC clock source selection.</p> <p>The software sets these bits to select the RTC clock source.</p> <p>00: no clock</p> <p>01: Select LSE oscillator as RTC clock</p> <p>10: Select LSI oscillator as RTC clock</p> <p>11: Select the HSE oscillator divided by 128 as the RTC clock</p>

Bit field	Name	Description
4	LSEBP	The external low-speed clock oscillator is bypassed. Set by software to bypass LSE. At this time LSEEN needs to be set 0 0: LSE clock is not bypassed 1: LSE clock is bypassed This bit cannot be reset by system reset
3	LSERD	External low-speed oscillator ready flag Set by hardware once LSE is ready. After LSIEN is cleared, LSIRD is cleared after 6 external low-speed oscillator clock cycles. 0: LSE is not ready 1: LSE is ready
2	LSEEN	External low-speed oscillator enable Set and cleared by software 0: Disable LSE oscillator 1: Enable LSE oscillator
1	LSIRD	Internal low-speed ready oscillator flag Set by hardware once LSI is ready. After LSIEN is cleared, LSIRD is cleared after 3 internal low-speed oscillator clock cycles. 0: LSI is not ready 1: LSI is ready
0	LSIEN	Internal low-speed oscillator enable Set and cleared by software 0: Disable LSI oscillator 1: Enable LSI oscillator

#### 4.3.11 Control/status register (RCC\_CTRLSTS)

Address offset: 0x24

Reset value: 0x0000 0018



Bit field	Name	Description
31:12	Reserved	Reserved, the reset value must be maintained
11	EMCCLPRSTF	EMCCLAMP reset flag Set by hardware when EMCCLAMP is reset. It is cleared by writing RMRSTF bit or por_rst_n reset. 0: No EMCCLAMP reset occurred

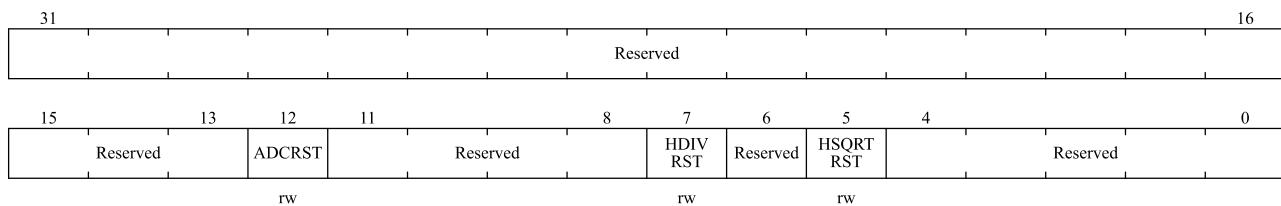
Bit field	Name	Description
		1: EMCCLAMP reset occurred
10	EMCGBRSTF	<p>EMCGB reset flag Set by hardware when EMCGB is reset. It is cleared by writing RMRSTF bit or por_rst_n reset.</p> <p>0: No EMCGB reset occurred 1: EMCGB reset occurred</p>
9	EMCGBNRSTF	<p>EMCGBN reset flag Set by hardware when EMCGBN is reset. It is cleared by writing RMRSTF bit or por_rst_n reset.</p> <p>0: No EMCGBN reset occurred 1: EMCGBN reset occurred</p>
8	LPWRRSTF	<p>Low-power reset flag Set by hardware at Low-power reset. It is cleared by writing RMRSTF bit or por_rst_n reset.</p> <p>0: No Low-power reset occurred 1: Low-power reset occurred</p>
7	WWDGRSTF	<p>Window watchdog reset flag Set by hardware when an Window watchdog reset occurs It is cleared by writing RMRSTF bit or por_rst_n reset.</p> <p>0: No Window watchdog reset occurred 1: Window watchdog reset occurred</p>
6	IWDGRSTF	<p>Independent watchdog reset flag Set by hardware when an independent watchdog reset occurs It is cleared by writing RMRSTF bit or por_rst_n reset.</p> <p>0: No independent watchdog reset occurred 1: Independent watchdog reset occurred</p>
5	SFTRSTF	<p>Software reset flag Set by hardware when a software reset occurs. It is cleared by writing RMRSTF bit or por_rst_n reset.</p> <p>0: No software reset occurred 1: Software reset occurred</p>
4	PORRSTF	<p>POR/PDR reset flag Set by hardware when POR/PDR is reset. Cleared by writing to the RMRSTF bit</p> <p>0: No POR/PDR reset occurred 1: POR/PDR reset occurred</p>
3	PINRSTF	<p>External pin reset flag Set by hardware when a reset from the NRST pin occurs. It is cleared by writing RMRSTF bit or por_rst_n reset.</p> <p>0: No NRST pin reset occurred 1: NRST pin reset occurred</p>

Bit field	Name	Description
2	MMURSTF	MMU reset flag Set by hardware when MMU reset occurs. It is cleared by writing RMRSTF bit or por_rst_n reset. 0: No MMU reset occurred 1: MMU reset occurred
1	RAMRSTF	RAM reset flag Set by hardware when RAM reset occurs. It is cleared by writing RMRSTF bit or por_rst_n reset. 0: No RAM reset occurred 1: RAM reset occurred
0	RMRSTF	REMOVE reset flag Set and clear by software 0: No effect 1: Clear these reset flags

#### 4.3.12 AHB peripheral reset register (RCC\_AHBPRST)

Address offset: 0x28

Reset value: 0x0000 0000



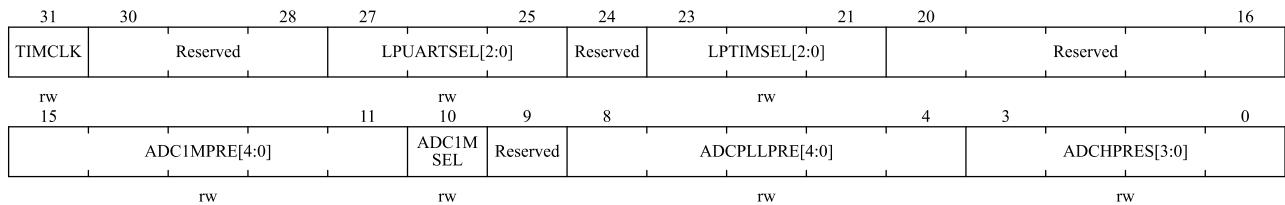
Bit field	Name	Description
31:13	Reserved	Reserved, the reset value must be maintained.
12	ADCRST	ADC reset Set and cleared by software. 0: Clear reset 1: Reset ADC
11:8	Reserved	Reserved, the reset value must be maintained.
7	HDIVRST	HDIV reset Set and cleared by software. 0: Clear reset 1: Reset HDIV
6	Reserved	Reserved, the reset value must be maintained.
5	HSQRTTRST	HSQRT reset Set and cleared by software. 0: Clear reset

Bit field	Name	Description
		1: Reset HSQRT
4:0	Reserved	Reserved, the reset value must be maintained.

### 4.3.13 Clock configuration register 2(RCC\_CFG2)

Address offset: 0x2c

Reset value: 0x0000 3800



Bit field	Name	Description
31	TIMCLK	TIM1/8 clock source selection  Set and cleared by software.  0: PCLK2 is selected as TIM1/8 clock source if APB2 prescaler is 1. Otherwise, PCLK2 × 2 is selected.  1: SYSCLK input clock is selected as TIM1/8 clock source.
30:28	Reserved	Reserved, the reset value must be maintained.
27:25	LPUARTSEL[2:0]	LPUART clock source selection  Set and cleared by software.  000: APB1 clock is selected 001: System clock is selected 010: HSI clock is selected 011: HSE clock is selected 100: LSI clock is selected 101: LSE clock is selected  Others: reserved.
24	Reserved	Reserved, the reset value must be maintained.
23:21	LPTIMSEL[2:0]	LPTIM clock source selection  Set and cleared by software.  000: select APB1 clock 001: Select HSI clock 010: Select HSE clock 011: Select LSI clock 100: Select LSE clock 101: Select COMP output  Others: Not allowed, and no clock will be generated
20:16	Reserved	Reserved, the reset value must be maintained.

Bit field	Name	Description
15:11	ADC1MPRE[4:0]	<p>ADC 1M clock prescaler Set and cleared by software to configure the division factor of ADC 1M clock source.</p> <p>00000: ADC 1M clock source not divided 00001: ADC 1M clock source divided by 2 00010: ADC 1M clock source divided by 3 ... 11110: ADC 1M clock source divided by 31 11111: ADC 1M clock source divided by 32</p>
10	ADC1MSEL	<p>ADC 1M clock source selection Set and cleared by software.</p> <p>0: HSI oscillator clock selected as the input clock of ADC 1M 1: HSE oscillator clock selected as the input clock of ADC 1M</p> <p><i>Note: When switching the ADC 1M clock source, you need to ensure that the HSI clock is turned on</i></p>
9	Reserved	Reserved, the reset value must be maintained.
8:4	ADCPPLLPRE[4:0]	<p>ADC PLL divider Set and cleared by software to configure the division factor from the PLL clock to the ADC.</p> <p>0xxxx: ADC PLL clock is disabled 10000: PLL clock not divided 10001: PLL clock divided by 2 10010: PLL clock divided by 3 10011: PLL clock divided by 4 10100: PLL clock divided by 6 10101: PLL clock divided by 8 10110: PLL clock divided by 10 10111: PLL clock divided by 12 11000: PLL clock divided by 16 11001: PLL clock divided by 32 11010: PLL clock divided by 64 11011: PLL clock divided by 128 Others: PLL clock divided by 256</p>
3:0	ADCHPRE[3:0]	<p>ADC HCLK prescaler Set and cleared by software to configure the division factor from the HCLK clock to the ADC.</p> <p>0000: HCLK clock divided by 1 0001: HCLK clock divided by 2 0010: HCLK clock divided by 3 0011: HCLK clock divided by 4 0100: HCLK clock divided by 6 0101: HCLK clock divided by 8</p>

Bit field	Name	Description
		0110: HCLK clock divided by 10 0111: HCLK clock divided by 12 1000: HCLK clock divided by 16 Others: HCLK clock divided by 32

#### 4.3.14 EMC control register 3 (RCC\_EMCTRL)

Address offset: 0x30

Reset value: 0x0000 0000

31	Reserved										24	23	22	21	20	19	18	17	16
15	CLPRST3	CLPRST2	CLPRST1	CLPRST0	GBDET3	GBDET2	GBDET1	GBDET0	GBNDETS	GBNDET2	GBNDET1	GBNDET0	CLPDET3	CLPDET2	CLPDET1	CLPDET0			
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bit field	Name	Description
31:24	Reserved	Reserved, the reset value must be maintained.
23	GBRST3	GB3 reset Set and cleared by software. and reset by por_rst_n 0: Disable reset request 1: Enable reset request
22	GBRST2	GB2 reset Set and cleared by software. and reset by por_rst_n 0: Disable reset request 1: Enable reset request
21	GBRST1	GB1 reset Set and cleared by software. and reset by por_rst_n 0: Disable reset request 1: Enable reset request
20	GBRST0	GB0 reset Set and cleared by software. and reset by por_rst_n 0: Disable reset request 1: Enable reset request
19	GBNRST3	GBN3 reset Set and cleared by software. and reset by por_rst_n 0: Disable reset request 1: Enable reset request
18	GBNRST2	GBN2 reset Set and cleared by software. and reset by por_rst_n 0: Disable reset request

Bit field	Name	Description
		1: Enable reset request
17	GBNRST1	GBN1 reset Set and cleared by software. and reset by por_rst_n 0: Disable reset request 1: Enable reset request
16	GBNRST0	GBN0 reset Set and cleared by software. and reset by por_rst_n 0: Disable reset request 1: Enable reset request
15	CLPRST3	EMC clamp3 reset Set and cleared by software. and reset by por_rst_n 0: Disable reset request 1: Enable reset request
14	CLPRST2	EMC clamp2 reset Set and cleared by software. and reset by por_rst_n 0: Disable reset request 1: Enable reset request
13	CLPRST1	EMC clamp1 reset Set and cleared by software. and reset by por_rst_n 0: Disable reset request 1: Enable reset request
12	CLPRST0	EMC clamp0 reset Set and cleared by software. and reset by por_rst_n 0: Disable reset request 1: Enable reset request
11	GBDET3	GB3 detection enable Set and cleared by software. and reset by por_rst_n 0: Disable detection 1: Enable detection
10	GBDET2	GB2 detection enable Set and cleared by software. and reset by por_rst_n 0: Disable detection 1: Enable detection
9	GBDET1	GB1 detection enable Set and cleared by software. and reset by por_rst_n 0: Disable detection 1: Enable detection
8	GBDET0	GB0 detection enable Set and cleared by software. and reset by por_rst_n 0: Disable detection 1: Enable detection

Bit field	Name	Description
7	GBNDET3	GBN3 detection enable Set and cleared by software. and reset by por_rst_n 0: Disable detection 1: Enable detection
6	GBNDET2	GBN2 detection enable Set and cleared by software. and reset by por_rst_n 0: Disable detection 1: Enable detection
5	GBNDET1	GBN1 detection enable Set and cleared by software. and reset by por_rst_n 0: Disable detection 1: Enable detection
4	GBNDET0	GBN0 detection enable Set and cleared by software. and reset by por_rst_n 0: Disable detection 1: Enable detection
3	CLPDET3	EMC Clamp3 detection enable Set and cleared by software. and reset by por_rst_n 0: Disable detection 1: Enable detection
2	CLPDET2	EMC Clamp2 detection enable Set and cleared by software. and reset by por_rst_n 0: Disable detection 1: Enable detection
1	CLPDET1	EMC Clamp1 detection enable Set and cleared by software. and reset by por_rst_n 0: Disable detection 1: Enable detection
0	CLPDET0	EMC Clamp0 detection enable Set and cleared by software. and reset by por_rst_n 0: Disable detection 1: Enable detection

## 5 GPIO and AFIO

### 5.1 Summary

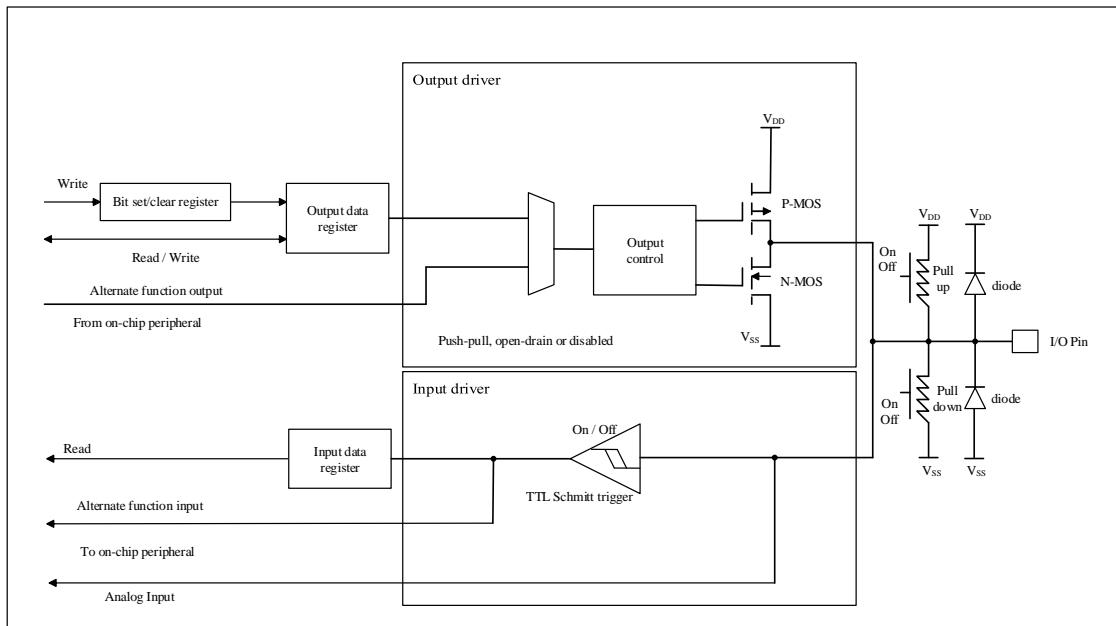
This design supports 40 GPIO, divided into 4 groups (GPIOA/GPIOB/GPIOC/ GPIOF), GPIOA and GPIOB each have 16 pins, GPIOC has 3 pins and GPIOF has 5 pins. Each GPIO pin can be configured by software as output (push-pull or open drain), input (with or without pull-up or pull-down) or alternate peripheral function ports (output/input), most GPIO pins are shared with digital or analog reuse peripherals, some IO pins are also reused with clock pins. Except for ports with analog input function, all GPIO pins have the ability to pass through a large current.

GPIO ports have the following characteristics:

- Each GPIO port can be individually configured into multiple modes by software
  - ◆ Input floating
  - ◆ Input pull-up
  - ◆ Input pull-down
  - ◆ Analog function
  - ◆ Open drain output and pull-up/pull-down can be configured
  - ◆ Push-pull output and pull-up/pull-down can be configured
  - ◆ Push-pull alternate function and pull-up/pull-down can be configured
  - ◆ Open-drain alternate function and pull-up/pull-down can be configured
- Individual bit set or bit clear function
- All I/O supports external interrupt function
- All I/O supports low power mode wake-up, rising or falling edge configurable
  - ◆ 16 EXTI can be used to wake up from SLEEP or STOP mode, and all I/Os can be reused as EXTI
  - ◆ PA0/PC13/PA2 three wake-up I/O can be used for PD mode wake-up, the maximum I/O filter time is 1us
- Support software remapping I/O alternate function
- Support GPIO lock mechanism, reset the lock state to clear

Each I/O port bit can be programmed arbitrarily, but I/O port registers must be accessed as 32-bit words (16-bit half-word or 8-bit byte access is not allowed). The following figure shows the basic structure of an I/O port.

Figure 5-1 Basic structure of I/O ports



## 5.2 Function description

### 5.2.1 I/O mode configuration

The mode control of I/O is set by the configuration registers GPIOx\_PMODE, GPIOx\_POTYPE and GPIOx\_PUPD (x=A,B,C,F). The configuration in different operation modes is shown in the following table:

Table 5-1 I/O port configuration table

PMODE[1:0]	POTYPE	PUPD[1:0]	I/O configuration
01	0	0	General-purpose output push-pull
	0	1	General-purpose output push-pull + pull-up
	1	0	General-purpose output push-pull + pull-down
	1	1	Reserved
	0	0	General-purpose output open-drain
	0	1	General-purpose output open-drain + pull-up
	1	0	General-purpose output open-drain + pull-down
	1	1	Reserved
10	0	0	Alternate function + push-pull
	0	1	Alternate function + push-pull + pull-up
	1	0	Alternate function + push-pull + pull-down
	1	1	Reserved
	0	0	Alternate function open-drain
	1	0	Alternate function open-drain + pull-up

P MODE[1:0]	P OTYPE	P UPD[1:0]		I/O configuration
	1	1	0	Alternate function open-drain + pull-down
	1	1	1	Reserved
00	x	0	0	Input floating
	x	0	1	Input pull-up
	x	1	0	Input pull-down
	x	1	1	Reserved
	x	0	0	Analog
11	x	0	1	Reserved
	x	1	0	
	x	1	1	

In addition, the GPIOx\_DS.DSy bit can be used to configure the high/low drive strength, and the GPIOx\_SR.SRy bit can be used to configure the high/low slew rate.

The input and output characteristics of I/O under different configurations are shown in the following table:

Table 5-2 I/O List of functional features of the pin

Feature	GPIO Input	GPIO Output	Analog function	Alternate function
Output buffer	Disabled	Enabled	Disabled	Configuration according to peripheral function
Schmitt trigger	Enabled	Enabled	Disabled, Output is forced to 0	Enable
PULL UP/DOWN/FLOATING	Configured	Configured	Disabled	Configuration according to peripheral function
OPEN DRAIN	Disabled	Can be configured, GPIO output 0 when output data is "0", high resistance of GPIO when "1"	Disabled	Can be configured, GPIO output 0 when output data is "0", high resistance of GPIO when "1"
PUSH PULL MODE	Disabled	Can be configured, GPIO output 0 when output data is "0", GPIO output 1 when output data is "1"	Disabled	Can be configured, GPIO output 0 when output data is "0", GPIO output 1 when output data is "1"
Input data register (I/O status)	Readable	Readable	Reads out 0	Readable
Output data register(Output value)	Invalid	Readable and written	Invalid	Readable

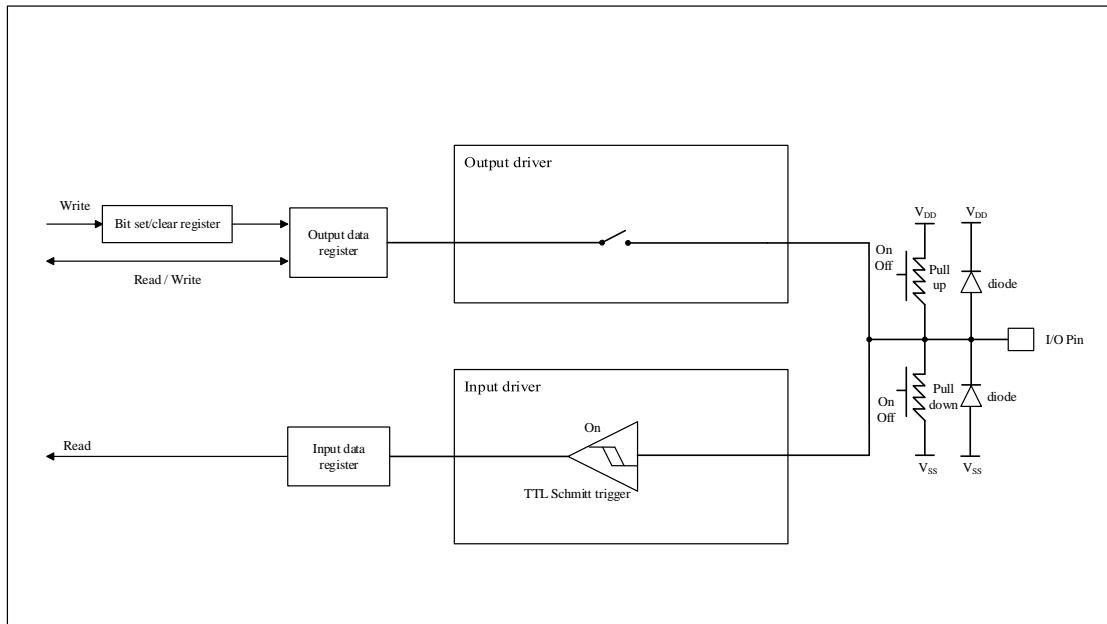
### 5.2.1.1 Input mode

When the I / O port is configured as input mode:

- Output buffer is disabled

- The schmitt trigger input is activated.
- Whether the pull-up and pull-down resistors are connected depends on the configuration of the GPIOx\_PUPD register.
- The data appearing on the I/O pins is sampled into the input data register on every APB2 clock
- I/O status is obtained by read access to the input data register

Figure 5-2 Input floating / pull-up / pull-down configuration mode.

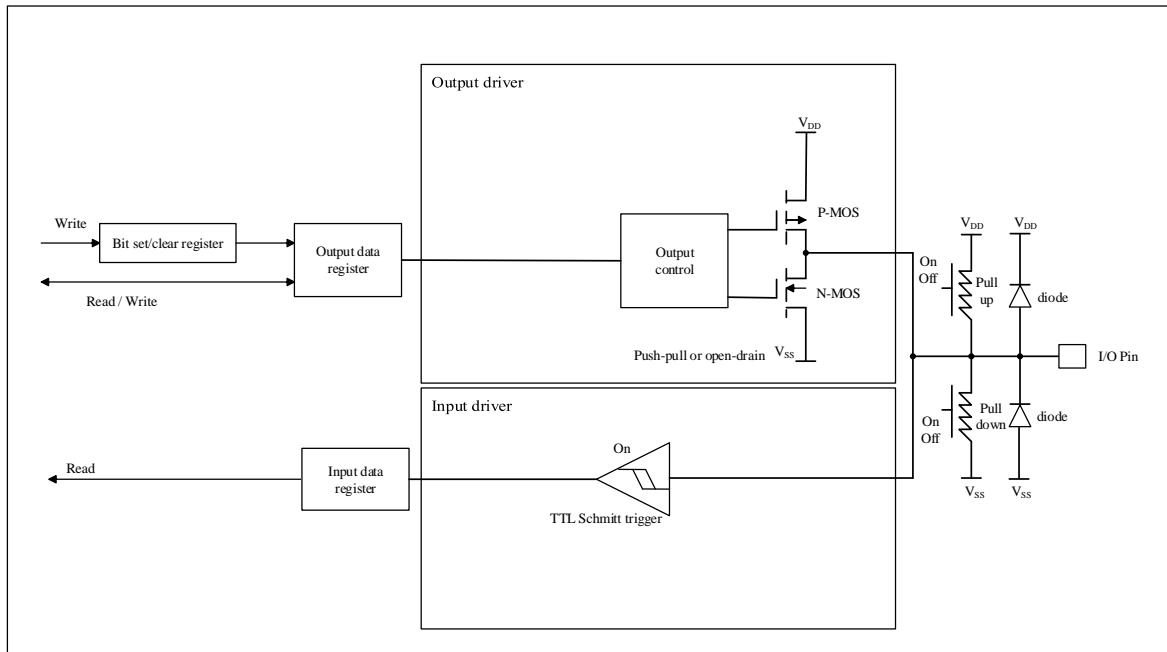


### 5.2.1.2 Output mode

When the I/O port is configured as output mode:

- The schmitt trigger input is activated.
- Whether the pull-up and pull-down resistors are connected depends on the configuration of the GPIOx\_PUPD register
- The output buffer is activated
  - Open drain mode: '0' on the output register activates the N-MOS, the pin outputs a low level. while '1' on the output register puts the port in a high resistance state (PMOS is never activated).
  - Push-pull mode: '0' on the output register activates the N-MOS, the pin outputs a low level. While '1' on the output register activates the P-MOS, the pin outputs a high level.
- The data appearing on the I/O pins is sampled into the input data register every APB2 clock.
- Read access to input data register to get I/O status.
- Read access to the output data register to get the last written value.

Figure 5-3 Output mode

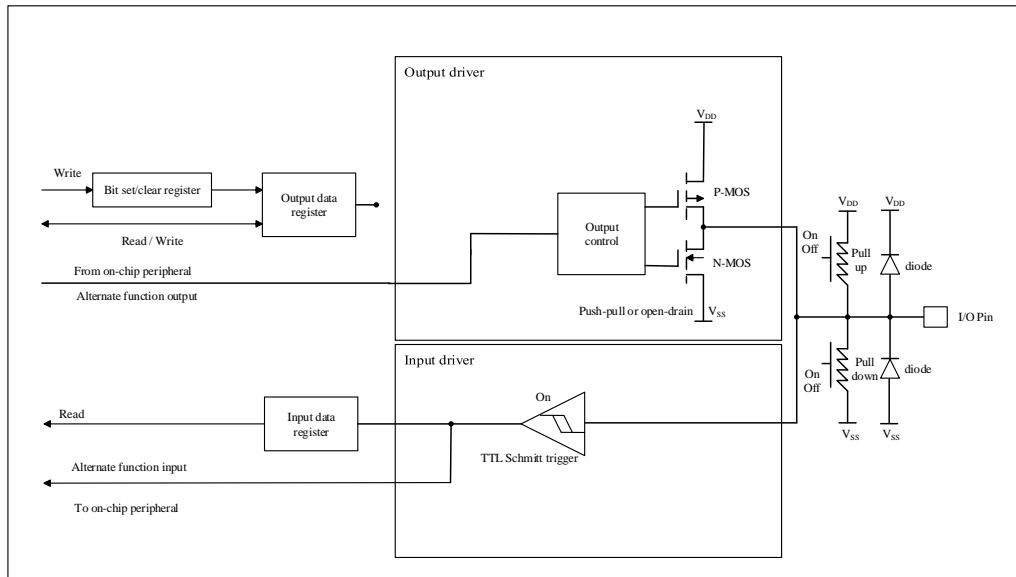


### 5.2.1.3 Alternate function mode

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

- The schmitt trigger input is activated.
- Whether the pull-up and pull-down resistors are connected, depending on the configuration of the GPIOx\_PUPD register.
- In open-drain or push-pull configuration, the output buffer is controlled by the peripheral.
- Signal-driven output buffers for built-in peripherals.
- At each APB2 clock cycle, the data appearing on the I/O pin is sampled into the input data register.
- Read access to input data register to get I/O status.
- Read access to the output data register to get the last written value.

Figure 5-4 Alternate function mode

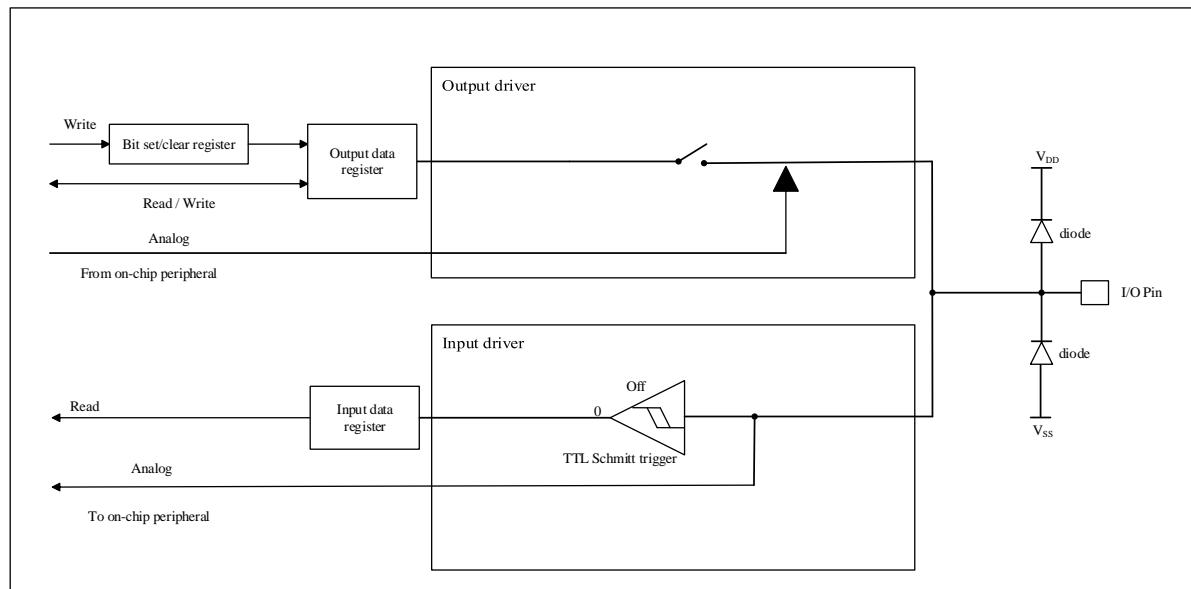


#### 5.2.1.4 Analog function mode

When the I/O Port is configured as analog function mode:

- The pull-up and pull-down resistors are disabled.
- Read access to the input data register gets the value “0”.
- The output buffer is disabled.
- Schmitt trigger input is disabled and output value is forced to '0' (achieves zero consumption on each analog I/O pin).

Figure 5-5 Analog function mode with high impedance



### 5.2.2 Status after reset

During and just after reset, the alternate function mapping is not turned on, and the I / O port is configured as analog function mode (GPIOx\_PMODE.PMODEEx [1:0] = 11b). However, there are the following exceptions to the signal.

- NRST default non-GPIO functions:
  - NRST pull-up input
- After reset, the default configuration of the pins related to the debugging system is the SWD interface I/O configuration:
  - PA14: SWCLK is placed in input pull-down mode
  - PA13: SWDIO is placed in input pull-up mode
- PA0 and PF0:
  - PA0 and PF0 default input floating mode.
  - PF0 are multiplexed to OSC\_IN.
- PF2/BOOT0:
  - During the chip startup process, PF2 as BOOT0 functions, and the default input pull-down mode. After the chip is initialized, it is used as a general-purpose GPIO and is configured as an analog function mode.

### 5.2.3 Individual bit setting and bit clearing

By writing '1' to the bit to be changed in the set register (GPIOx\_PBSC) and reset register (GPIOx\_PBC), the individual bit operation of the data register (GPIOx\_POD) can be realized, and one or more bits can be set/reset. The bit written with '1' is set or cleared accordingly, and the bit not written with '1' will not be changed. The software does not need to disable interrupts, and is completed in a single APB2 write operation.

### 5.2.4 External interrupt /wakeup line

All ports have external interrupt capability, which can be configured in the EXTI module:

- The port must be configured in input mode.
- All ports can be configured for SLEEP/STOP mode wake-up, supporting rising or falling edge configurable.
- PA0/PC13/PA2, can be used for PD mode wake-up, with independent wake-up enable, support rising edge / falling edge can be configured, need to configure before entering PD mode.
- General purpose I/O ports are connected to 16 external interrupt/event lines as shown in Figure 6-2, configured by registers AFIO\_EXTI\_CFGx.

### 5.2.5 Alternate function

When I/O ports are configured for alternate function mode. The port bit configuration register

(GPIOx\_AFL/GPIOx\_AFH, GPIOx\_PMODE, GPIOx\_POTYPE and GPIOx\_PUPD) must be configured before use, reuse input or output is determined by the peripheral.

### 5.2.5.1 Software remapping I/O alternate function

To expand the flexibility of alternate peripheral functions under different device packages, some peripheral alternate functions can be remapped to other pins. Each I/O has up to 16 alternate functions (AF0~AF15). After reset, except for PA13 and PA14, AFSELy is selected as AF15 by default. The I/O alternate function can be remapped by software configuring the corresponding registers (GPIOx\_AFL/ GPIOx\_AFH).

At this time, the alternate functions are no longer mapped to their original pins(For the I/O alternate function of the peripheral, if it is remapped to a different pin, then the input is remapping choose one of multiple, and the output will be connected to the remapped position, and the original position will be disconnected).

### 5.2.5.2 SWD alternate function I/O remapping

Table 5-3 I/O List of functional features of the pin

Alternate function	I/O	Remapping
SWDIO	PA13	AF0
SWCLK	PA14	AF0

### 5.2.5.3 TIMx alternate function I/O remapping

#### 5.2.5.3.1 TIM1 alternate function I/O remapping

Table 5-4 TIM1 alternate function I/O remapping

Alternate function	I/O	Remapping
TIM1_ETR	PA12	AF2
TIM1_BKIN	PA2	AF3
	PA6	AF1
	PB12	AF1
TIM1_CH1	PA4	AF3
	PA8	AF2
TIM1_CH2	PA3	AF3
	PA9	AF2
TIM1_CH3	PA5	AF3
	PA10	AF2
TIM1_CH4	PA11	AF2
TIM1_CH1N	PA7	AF5
	PB13	AF1
TIM1_CH2N	PA5	AF4
	PB0	AF1
	PB14	AF1
TIM1_CH3N	PB1	AF1
	PB15	AF1

### 5.2.5.3.2 TIM8 alternate function I/O remapping

Table 5-5 TIM8 alternate function I/O remapping

Alternate function	I/O	Remapping
TIM8_ETR	PA0	AF10
	PA4	AF13
	PA5	AF1
TIM8_BKIN	PA9	AF1
	PA10	AF1
	PB4	AF1
	PB5	AF1
TIM8_CH1	PA0	AF3
	PA5	AF2
	PA6	AF4
	PB8	AF5
	PB12	AF5
TIM8_CH2	PA1	AF2
	PA7	AF1
	PB9	AF5
	PB13	AF5
TIM8_CH3	PA2	AF2
	PB6	AF2
	PB11	AF5
	PB14	AF5
TIM8_CH4	PA3	AF2
	PB7	AF7
	PB15	AF7
TIM8_CH1N	PA9	AF5
	PB6	AF5
TIM8_CH2N	PA8	AF1
	PB7	AF5
TIM8_CH3N	PB5	AF5
	PB15	AF5

### 5.2.5.3.3 TIM3 alternate function I/O remapping

Table 5-6 TIM3 alternate function I/O remapping

Alternate function	I/O	Remapping
TIM3_ETR	PA1	AF10
	PB3	AF1
	PB10	AF1

Alternate function	I/O	Remapping
TIM3_CH1	PA4	AF2
	PA6	AF2
	PB4	AF2
TIM3_CH2	PA7	AF2
	PB5	AF2
TIM3_CH3	PB0	AF2
	PB1	AF5
TIM3_CH4	PB1	AF2
	PB2	AF5

#### 5.2.5.4 LPTIM alternate function I/O remapping

Table 5-7 LPTIM alternate function I/O remapping

Alternate function	I/O	Remapping
LPTIM _ETR	PA6	AF9
	PB6	AF8
LPTIM _IN1	PA0	AF7
	PB5	AF8
LPTIM _IN2	PA1	AF6
	PB7	AF8
LPTIM _OUT	PA9	AF9
	PB2	AF8
	PB4	AF8

#### 5.2.5.5 USARTx alternate function I/O remapping

##### 5.2.5.5.1 USART1 alternate function I/O remapping

Table 5-8 USART1 alternate function I/O remapping

Alternate function	I/O	Remapping
USART1_CTS	PA0	AF1
	PA11	AF4
USART1_RTS	PA1	AF1
	PA12	AF4
USART1_TX	PA2	AF1
	PA9	AF4
	PA13	AF6
	PA14	AF4
	PB6	AF4
	PB9	AF4
USART1_RX	PA3	AF1
	PA10	AF4

Alternate function	I/O	Remapping
	PA13	AF4
	PA15	AF4
	PB7	AF4
USART1_CK	PA4	AF1
	PA8	AF4
	PF1	AF2

### 5.2.5.5.2 USART2 alternate function I/O remapping

Table 5-9 USART2 alternate function I/O remapping

Alternate function	I/O	Remapping
USART2_CTS	PA0	AF4
	PA7	AF10
USART2_RTS	PA1	AF4
USART2_TX	PA2	AF4
	PA9	AF10
	PA14	AF1
USART2_RX	PA0	AF5
	PA3	AF4
	PA10	AF10
	PA13	AF1
	PA15	AF1
USART2_CK	PA4	AF4
	PB1	AF3
	PF1	AF5

### 5.2.5.6 LPUART alternate function I/O remapping

Table 5-10 LPUART alternate function I/O remapping

Alternate function	I/O	Remapping
LPUART_CTS	PA6	AF5
	PB7	AF2
	PB13	AF4
LPUART_RTS	PA15	AF6
	PB1	AF4
	PB14	AF4
LPUART_TX	PA0	AF6
	PA1	AF5
	PA4	AF10
	PA6	AF6
	PB3	AF5
	PB5	AF4

Alternate function	I/O	Remapping
	PB10	AF4
LPUART_RX	PA0	AF11
	PA3	AF6
	PA7	AF6
	PB4	AF5
	PB7	AF9
	PB11	AF4

### 5.2.5.7 I2Cx alternate function I/O remapping

#### 5.2.5.7.1 I2C1 alternate function I/O remapping

Table 5-11 I2C1 alternate function I/O remapping

Alternate function	I/O	Remapping
I2C1_SCL	PA4	AF7
	PA9	AF6
	PB6	AF6
	PB8	AF6
	PB10	AF6
	PF1	AF1
	PF6	AF1
I2C1_SDA	PA10	AF6
	PA13	AF7
	PB7	AF6
	PB9	AF6
	PB11	AF6
	PF0	AF1
	PF7	AF1
I2C1_SMBA	PA1	AF7
	PA14	AF7
	PB2	AF6
	PB5	AF6

#### 5.2.5.7.2 I2C2 alternate function I/O remapping

Table 5-12 I2C2 alternate function I/O remapping

Alternate function	I/O	Remapping
I2C2_SCL	PA6	AF7
	PA9	AF7
	PA11	AF7
	PB10	AF7
	PB13	AF7

Alternate function	I/O	Remapping
	PF6	AF0
I2C2_SDA	PA7	AF7
	PA10	AF7
	PA12	AF7
	PB11	AF7
	PB14	AF7
	PF7	AF0
I2C2_SMBA	PB2	AF7

### 5.2.5.8 SPIx/I2S alternate function I/O remapping

#### 5.2.5.8.1 SPI1/I2S alternate function I/O remapping

Table 5-13 SPI1/I2S alternate function I/O remapping

Alternate function	I/O	Remapping
SPI1_I2S_NSS_WS	PA1	AF0
	PA4	AF0
	PA15	AF0
	PB12	AF0
SPI1_I2S_SCLK_CK	PA0	AF0
	PA5	AF0
	PA13	AF5
	PB3	AF0
	PB13	AF0
SPI1_I2S_MISO_MCK	PA3	AF0
	PA4	AF5
	PA6	AF0
	PA14	AF5
	PB4	AF0
	PB14	AF0
SPI1_I2S莫斯SD	PA2	AF0
	PA5	AF5
	PA7	AF0
	PB1	AF0
	PB5	AF0
	PB10	AF0
	PB15	AF0

### 5.2.5.8.2 SPI2 alternate function I/O remapping

Table 5-14 SPI2 alternate function I/O remapping

Alternate function	I/O	Remapping
SPI2_NSS	PA7	AF9
	PA8	AF0
	PB9	AF0
	PB12	AF8
	PF7	AF4
SPI2_SCLK	PA9	AF0
	PB0	AF0
	PB10	AF2
	PB13	AF8
	PF6	AF4
SPI2_MISO	PA10	AF0
	PA12	AF0
	PB14	AF8
SPI2_MOSI	PA11	AF0
	PB1	AF8
	PB15	AF8

### 5.2.5.9 COMP alternate function I/O remapping

Table 5-15 COMP alternate function I/O remapping

Alternate function	I/O	Remapping
COMP_OUT	PA0	AF8
	PA6	AF8
	PA11	AF8
	PA12	AF8

### 5.2.5.10 BEEPER alternate function I/O remapping

Table 5-16 BEEPER alternate function I/O remapping

Alternate function	I/O	Remapping
BEEPER_OUT	PA6	AF11
BEEPER_N_OUT	PA7	AF11

### 5.2.5.11 EVENTOUT alternate function I/O remapping

Table 5-17 EVENTOUT alternate function I/O remapping

Alternate function	I/O	Remapping
EVENTOUT	PA1	AF3
	PA6~PA8	AF3
	PA11~PA12	AF3

Alternate function	I/O	Remapping
	PA15	AF3
	PB0	AF3
	PB3~PB4	AF3
	PB9	AF3
	PB11~PB12	AF3

### 5.2.5.12 RTC alternate function I/O remapping

PC13 can be used as RTC TAMPER1 intrusion detection pin, RTC Timestamp, RTC output (RTC alarm, Wakeup event or calibration output (256Hz or 1Hz)).

PA0 can be used as RTC TAMPER2 intrusion detection pin.

PA10 or PB15 can be used as RTC REFCLKIN reference clock input pin.

Table 5-18 RTC alternate function I/O remapping

Alternate function	I/O	Remapping
RTC_TS	PC13	AF1
RTC_TAMP1	PC13	AF2
RTC_TAMP2	PA0	AF9
RTC_REFIN	PA10	AF11
	PB15	AF9
RTC_OUT	PC13	AF3

### 5.2.5.13 RCC alternate function I/O remapping

Table 5-19 RCC alternate function I/O remapping

Alternate function	I/O	Remapping
MCO	PA8	AF5
	PA9	AF11

### 5.2.5.14 OSC\_IN/OSC\_OUT alternate function I/O remapping

Table 5-20 OSC\_IN/OSC\_OUT alternate function I/O remapping

Alternate function	I/O	Remapping
OSC_IN	PF0	AF0
OSC_OUT	PF1	AF0

### 5.2.5.15 Use OSC32\_IN/OSC32\_OUT pins as GPIO ports PC14/PC15

PC14~PC15 two pins can be used as GPIO mode or alternate function mode:

- PC14 and PC15 can be used for GPIO or LSE (OSC32\_IN, OSC32\_OUT) pins.
- Turn on the LSE function by setting the RCC\_LSCTRL.LSEEN and RCC\_LSCTRL.LSEBP bits.

Table 5-21 OSC32 alternate function remapping

PC14 and PC15	Condition	PAD mode configure
GPIO mode	It can only be used in GPIO mode when the LSE is turned off and the 1.5V power supply is turned off without entering the low power mode (PD).	The mode of the GPIO is determined by the application
LSE crystal mode	RCC_LSCTRL.LSEEN bit is enabled, alternate mode is on	Analog function mode
LSE external clock mode	RCC_LSCTRL.LSEEN bit is disabled, RCC_LSCTRL.LSEBP is enabled, alternate mode is enabled	Input pull-down

The default is analog function mode; PC14 and PC15 decide which mode and I/O function they are in according to RCC\_LSCTRL.LSEEN, RCC\_LSCTRL.LSEBP, chip mode signal, GPIOx\_PMODE, GPIOx\_POTYPE and GPIOx\_PUPD.

### 5.2.5.16 Remap OSC\_IN/OSC\_OUT alternate function

PF0 and PF1 can be used for GPIO or HSE (OSC\_IN, OSC\_OUT) pins and other analog or digital signal pins.

- Configure the mode of PF0/PF1 by setting registers such as GPIOF\_PMODE, GPIOF\_POTYPE and GPIOF\_PUPD.
- Remap the I/O alternate function of PF0/PF1 by setting the GPIOF\_AFL.AFSEL0 [3:0] bits and GPIOF\_AFL.AFSEL1[3:0] bits.
- Enable the HSE function by setting the RCC\_CTRL.HSEEN bit.

Table 5-22 OSC alternate function remapping

PF0	Condition	PAD mode configure
GPIO mode	Configure general purpose output or input mode and GPIOF_PID and GPIOF_POD registers.	The mode of the GPIO is determined by the application
HSE crystal mode or other analog alternate mode	Reset to analog input HSE, OPA alternate mode	Analog function mode
HSE external clock or other digital alternate mode	HSE clock, I2C1, USART1/2 alternate mode	Mode is determined by the application

### 5.2.5.17 ADC external trigger alternate function remapping

The external trigger source of ADC injection conversion and regular conversion supports remapping, see the AFIO\_CFG register.

Table 5-23 ADC external trigger injection conversion alternate function remapping

Alternate Function	ADC_ETRI = 0	ADC_ETRI = 1
ADC external trigger injection conversion	ADC external trigger injection conversion is connected to EXTI (0-15), EXTIX can be selected by AFIO_CFG.EXTI_ETRI[3:0] configuration	ADC external trigger injection conversion is connected to TIM8_CH4

Table 5-24 ADC external trigger regular conversion alternate function remapping

Alternate Function	ADC_ETRR = 0	ADC_ETRR = 1
ADC external trigger injection conversion	ADC external trigger regular conversion is connected to EXTI (0-15), EXTIx can be selected by AFIO_CFG.EXTI_ETRI[3:0] configuration	ADC external trigger regular conversion is connected to TIM8_TRGO

## 5.2.6 I/O configuration of peripherals

Table 5-25 ADC

ADC	PAD configuration
ADC	Analog function mode

Table 5-26 PVD

PVD	PAD configuration
PVD_IN	Analog function mode

Table 5-27 TIM1/TIM8

TIM pin	Configuration	PAD configuration mode
TIM1/8_CHx	Channel x input capture	Alternate function push-pull
	Output compare channel x	Alternate function push-pull
TIM1/8_CHxN	Complementary output channel x	Alternate function push-pull
TIM1/8_BKIN	Brake input	Alternate function push-pull
TIM1/8_ETR	External trigger clock input	Alternate function push-pull

Table 5-28 TIM3 and LPTIM

TIM3/LPTIM pin	Configuration	PAD configuration mode
TIM3/LPTIM_CHx	Input capture channel x	Alternate function push-pull
	Output compare channel x	Alternate function push-pull
TIM3/LPTIM_ETR	External trigger clock input	Alternate function push-pull

Table 5-29 USART

USART pin	Configuration	PAD configuration
USARTx_TX	full duplex mode	Alternate function push-pull
	Half duplex mode	Alternate function open-drain
USARTx_RX	Full duplex mode	Alternate function push-pull(no pull-down or pull-up/pull-up)
	Half duplex mode	Unused, can be used as general I/O.
USARTx_CK	Synchronous mode	Alternate function push-pull
USARTx RTS	Hardware flow control	Alternate function push-pull
USARTx CTS	Hardware flow control	Alternate function push-pull(no pull-down or pull-up/pull-up)

Table 5-30 LPUART

LPUART pin	Configuration	PAD configuration
LPUART_TX	Full duplex mode	Alternate function push-pull
LPUART_RX	Full duplex mode	Alternate function push-pull(no pull-down or pull-up/pull-up)
LPUART_RTS	Hardware flow control	Alternate function push-pull
LPUART_CTS	Hardware flow control	Alternate function push-pull(no pull-down or pull-up /pull-up)

Table 5-31 I2C

I2C pin	Configuration	PAD configuration
I2Cx_SCL	I2C clock	Alternate function open-drain
I2Cx_SDA	I2C data	Alternate function open-drain
I2Cx_SMBA	SMBA data	Alternate function open-drain

Table 5-32 SPI

SPI pin	Configuration	PAD configuration
SPIx_SCLK	Master mode	Alternate function push-pull
	Slave mode	Alternate function push-pull
SPIx_MOSI	Full duplex mode / Master mode	Alternate function push-pull
	Full duplex mode / Slave mode	Alternate function push-pull(no pull-down or pull-up/pull-up)
SPIx_MISO	Simple bidirectional data line / Master mode	Alternate function push-pull
	Simple bidirectional data line / Slave mode	Unused, can be used as general I/O.
SPIx_NSS	Full duplex mode / Master mode	Alternate function push-pull(no pull-down or pull-up/pull-up)
	Full duplex mode / Slave mode	Alternate function push-pull
	Simple bidirectional data line / Master mode	Unused, can be used as general I/O.
	Simple bidirectional data line / Slave mode	Alternate function push-pull
SPIx_NSS	Hardware Master/ Slave mode	Alternate function push-pull(no pull-down or pull-up/pull-up/pull-down)
	Hardware Master /NSS output enable	Alternate function push-pull (When acting as the master, NSS can choose idle high impedance or idle as 1)
	Software mode	Unused, can be used as general I/O.

Table 5-33 COMP

COMP pin	PAD configuration
COMP_OUT	Alternate function push-pull
COMP_IN	Analog function mode

Table 5-34 BEEPER

BEEPER pin	PAD configuration
BEEPER_OUT	Alternate function push-pull
BEEPER_N_OUT	Alternate function push-pull

Table 5-35 Other

Pin	Alternate function	PAD configuration
EVENTOUT	Event output	Alternate function push-pull
RTC_TAMP1/RTC_TAMP2	Intrusion event input	Alternate function push-pull Hardware forced pull-up
RTC_TS	RTC timestamp	Alternate function push-pull
RTC_REFIN	RTC reference clock input	Alternate function push-pull
RTC_OUT	RTC output	Alternate function push-pull
MCO	Clock output	Alternate function push-pull
EXTI input line	External interrupt input	Input floating or input with pull-up or input with pull-down

## 5.2.7 GPIO Locking mechanism

The locking mechanism is used to freeze the I/O configuration to prevent accidental changes. When a lock (LOCK) procedure is executed on a port bit, the configuration of the port cannot be changed until the next reset, refer to the port configuration lock register GPIOx\_PLOCK.

- Only after the GPIOx\_PLOCK.PLOCKKK is operated in the correct sequence w1->w0->w1->r0 (where r0 must be), it will become 1; after that, it will only become 0 after a system reset.
- GPIOx\_PLOCK.PLOCK[15:0] can only be modified when GPIOx\_PLOCK.PLOCKKK = 0 (that is, when unlocked).
- GPIOx\_PLOCK.PLOCK is only written simultaneously with non-zero GPIOx\_PLOCK.PLOCK[15:0], the sequence w1->w0->w1->r0 is valid; During the sequence writing process, GPIOx\_PLOCK.PLOCK[15:0] remains unchanged.
- As long as GPIOx\_PLOCK.PLOCKKK = 0, the bits of GPIOx\_PMODE / GPIOx\_POTYPE / GPIOx\_PUPD / GPIOx\_AFL / GPIOx\_AFH can be modified, they are not affected by GPIOx\_PLOCK.PLOCK[15:0] configuration.
- GPIOx\_PLOCK.PLOCKKK = 1, GPIOx\_PMODE/GPIOx\_POTYPE/GPIOx\_PUPD/GPIOx\_AFL/GPIOx\_AFH are controlled by GPIOx\_PLOCK.PLOCK[15:0]. Corresponding to GPIOx\_PLOCK.PLOCKy (y = 0...15) = 1, it is a lock configuration and cannot be modified; PLOCKy = 0, it can be modified.
- If the lock sequence operation is wrong, then it must be redone (w1-> w0-> w1-> r0) to initiate the lock operation again.

## 5.3 GPIO Registers

These peripheral registers must be operated as 32-bit words.

### 5.3.1 GPIO register overview

GPIOA base address: 0x40010800

GPIOB base address: 0x40010C00

GPIOC base address: 0x40011000

GPIOF base address: 0x40011C00

Table 5-36 GPIO register overview

Offset	Register	
000h	GPIOx_PMODE[x=A,B,C,F]	
	PMODE[5:0]	
	PMODE[15:10]	
	PMODE[14:10]	
	PMODE[13:10]	
	PMODE[12:10]	
	PMODE[11:10]	
	PMODE[10:10]	
	PMODE[9:10]	
	PMODE[8:10]	
	PMODE[7:10]	
	PMODE[6:10]	
	PMODE[5:10]	
	PMODE[4:10]	
	PMODE[3:10]	
	PMODE[2:10]	
	PMODE[1:10]	
	PMODE[0:10]	
004h	GPIOx_POTYPE[x=A,B,C,F]	
	POT[15:0]	
	POT[14:0]	
	POT[13:0]	
	POT[12:0]	
	POT[11:0]	
	POT[10:0]	
	POT[9:0]	
	POT[8:0]	
	POT[7:0]	
	POT[6:0]	
008h	GPIOx_SR[x=A,B,C,F]	
	SR[15:0]	
	SR[16:0]	
	SR[17:0]	
	SR[18:0]	
	SR[19:0]	
	SR[20:0]	
	SR[21:0]	
	SR[22:0]	
	SR[23:0]	
	SR[24:0]	
00Ch	GPIOx_PUPD[x=A,B,C,F]	
	PUP[5:0]	
	PUP[7:0]	
	PUP[13:0]	
	PUP[10:0]	
	PUP[9:0]	
	PUP[8:0]	
	PUP[7:0]	
	PUP[6:0]	
	PUP[5:0]	
	PUP[4:0]	
	PUP[3:0]	
	PUP[2:0]	
	PUP[1:0]	
	PUP[0:0]	
010h	GPIOx_PID[x=A,B,C,F]	
	PID[15:0]	
	PID[14:0]	
	PID[13:0]	
	PID[12:0]	
	PID[11:0]	
	PID[10:0]	
	PID[9:0]	
	PID[8:0]	
	PID[7:0]	
	PID[6:0]	
014h	GPIOx_POD[x=A,B,C,F]	
	POD[5:0]	
	POD[4:0]	
	POD[3:0]	
	POD[2:0]	
	POD[1:0]	
	POD[0:0]	
	SR25:1	
	SR26:1	
	SR27:1	
	SR28:1	
	SR29:1	
	SR30:1	

Offset	Register																	
018h	GPIOx_PBSC	x=A,B,C,F	PBC15	31	PBC15	30	PBC13	29	PBC12	28	PBC11	27	PBC10	26	PBC9	25	PBC8	24
		x=A,B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PBC6	22
		x=C	0	0	0												PBC5	21
		x=F																PBC4
01Ch	GPIOx_PLOCK	x=A,B,C,F	Reserved															
		x=A,B	Reserved															
		x=C	Reserved															
		x=F	Reserved															
020h	GPIOx_AFL	x=A,B,C,F	AFSEL7[3:0]	AFSEL6[3:0]	AFSEL5[3:0]	AFSEL4[3:0]	AFSEL3[3:0]	AFSEL2[3:0]	AFSEL1[3:0]	AFSEL0[3:0]	PLOCKK	PLOCK15	PLOCK14	PLOCK13	PLOCK12	PLOCK11	PLOCK10	PLOCK9
		x=A,B	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
		x=C																
		x=F	1	1	1	1	1	1	1	1								
024h	GPIOx_AFH	x=A,B,C,F	AFSEL15[3:0]	AFSEL14[3:0]	AFSEL13[3:0]	AFSEL12[3:0]	AFSEL11[3:0]	AFSEL10[3:0]	AFSEL9[3:0]	AFSEL8[3:0]	PLOCKK	PLOCK15	PLOCK14	PLOCK13	PLOCK12	PLOCK11	PLOCK10	PLOCK9
		x=A	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1
		x=B	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
		x=C	1	1	1	1	1	1	1	1								
028h	GPIOx_PBC	x=A,B,C,F	Reserved															
		x=A,B	Reserved															
		x=C	Reserved															
		x=F	Reserved															
02Ch	GPIOx_DS	x=A,B,C,F	Reserved															
		x=A,B	Reserved															
		x=C	Reserved															
		x=F	Reserved															

### 5.3.2 GPIO port mode description register (GPIOx\_PMODE)

Offset address : 0x00

Reset value : 0xEBFF FFFF (x=A); 0xFFFF FFFF (x=B); 0xFC00 0000 (x=C); 0x0000 F03F (x=F)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
PMODE15[1:0]	PMODE14[1:0]	PMODE13[1:0]	PMODE12[1:0]	PMODE11[1:0]	PMODE10[1:0]	PMODE9[1:0]	PMODE8[1:0]										
rw 15	rw 14	rw 13	rw 12	rw 11	rw 10	rw 9	rw 8	rw 7	rw 6	rw 5	rw 4	rw 3	rw 2	rw 1	rw 0		
PMODE7[1:0]	PMODE6[1:0]	PMODE5[1:0]	PMODE4[1:0]	PMODE3[1:0]	PMODE2[1:0]	PMODE1[1:0]	PMODE0[1:0]										
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit field	Name	Description
31:30	PMODEy[1:0]	Mode of port GPIOx (x = A,B,C,F) pin PINy: 00: Input mode 01: General output mode
29:28		
27:26		

Bit field	Name	Description
25:24		10: Alternate function mode
23:22		11: Analog function mode
21:20		<i>Note: when x = A,B, y = 0...15;</i>
19:18		<i>When x = C, y = 13, 14, 15, the remaining bits are reserved, and the reserved bits are read-only;</i>
17:16		<i>When x = F, y = 0, 1, 2, 6, 7, the remaining bits are reserved, and the reserved bits are read-only.</i>
15:14		
13:12		
11:10		
9:8		
7:6		
5:4		
3:2		
1:0		

### 5.3.3 GPIO port type definition (GPIOx\_POTYPE)

Offset address : 0x04

Reset value : 0x0000 0000 (x=A,B,C,F)

31	Reserved															16
15	POT15	POT14	POT13	POT12	POT11	POT10	POT9	POT8	POT7	POT6	POT5	POT4	POT3	POT2	POT1	POT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained
15:0	POTy	Output type of port GPIOx (x = A,B,C,F) pin PINy: 0: Output push-pull mode (state after reset) 1: Output open-drain mode <i>Note: when x = A,B, y = 0...15;</i> <i>When x = C, y = 13, 14, 15, the remaining bits are reserved, and the reserved bits are read-only;</i> <i>When x = F, y = 0, 1, 2, 6, 7, the remaining bits are reserved, and the reserved bits are read-only.</i>

### 5.3.4 GPIO slew rate configuration register (GPIOx\_SR)

Offset address : 0x08

Reset value : 0x0000 FFFF (x=A, B); 0x0000 E000 (x=C); 0x0000 00C7 (x=F)

31															16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SR15	SR14	SR13	SR12	SR11	SR10	SR9	SR8	SR7	SR6	SR5	SR4	SR3	SR2	SR1	SR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained
15:0	SRy	<p>Slew rate configuration bits for port GPIOx (<math>x = A, B, C, F</math>) pin PINy</p> <p>0: Fast slew rate 1: Slow slew rate</p> <p><i>Note: when <math>x = A, B, y = 0 \dots 15</math>;</i> <i>When <math>x = C, y = 13, 14, 15</math>, the remaining bits are reserved, and the reserved bits are read-only;</i> <i>When <math>x = F, y = 0, 1, 2, 6, 7</math>, the remaining bits are reserved, and the reserved bits are read-only.</i></p>

### 5.3.5 GPIO port pull-up/pull-down register (GPIOx\_PUPD)

Offset address : 0x0C

Reset value : 0x2400 0000 ( $x=A$ ); 0x0000 0000 ( $x=B,C,F$ )

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0]	PUPD14[1:0]	PUPD13[1:0]	PUPD12[1:0]	PUPD11[1:0]	PUPD10[1:0]	PUPD9[1:0]	PUPD8[1:0]								
15	rw	14	rw	13	rw	12	rw	11	rw	10	rw	9	rw	8	rw
PUPD7[1:0]	PUPD6[1:0]	PUPD5[1:0]	PUPD4[1:0]	PUPD3[1:0]	PUPD2[1:0]	PUPD1[1:0]	PUPD0[1:0]								
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

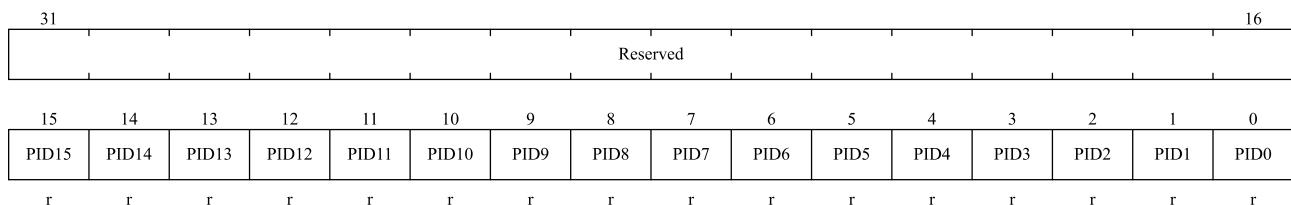
Bit field	Name	Description
31:30	PUPDy[1:0]	Pull-up and pull-down mode of port GPIOx ( $x = A, B, C, D, F$ ) pin PINy:
29:28		00: no pull-up/pull-down
27:26		01: Pull up
25:24		10: Pull down
23:22		11: Reserved
21:20		<i>Note: when <math>x = A, B, y = 0 \dots 15</math>;</i> <i>When <math>x = C, y = 13, 14, 15</math>, the remaining bits are reserved, and the reserved bits are read-only;</i>
19:18		<i>When <math>x = F, y = 0, 1, 2, 6, 7</math>, the remaining bits are reserved, and the reserved bits are read-only.</i>
17:16		
15:14		
13:12		
11:10		
9:8		
7:6		

Bit field	Name	Description
5:4		
3:2		
1:0		

### 5.3.6 GPIO port input data register (GPIOx\_PID)

Offset address : 0x10

Reset value : 0x0000 0000 (x=A,B,C,F)

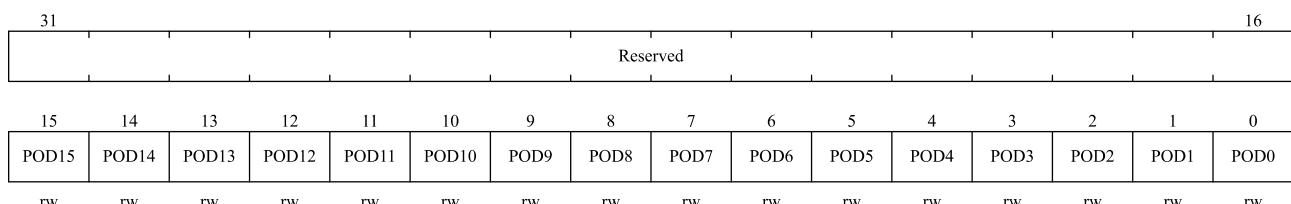


Bit field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained
15:0	PIDy	<p>Input data of port GPIOx (x = A,B,C,F) pin PINy</p> <p>These bits are read-only, and the read value is the state of the corresponding I/O port.</p> <p><i>Note: when x = A,B, y = 0...15;</i></p> <p><i>When x = C, y = 13, 14, 15, the remaining bits are reserved, and the reserved bits are read-only;</i></p> <p><i>When x = F, y = 0, 1, 2, 6, 7, the remaining bits are reserved, and the reserved bits are read-only.</i></p>

### 5.3.7 GPIO port output data register (GPIOx\_POD)

Offset address : 0x14

Reset value : 0x0000 0000 (x=A,B,C,F)



Bit field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained
15:0	PODy	<p>Output data of port GPIOx (x = A,B,C,F) pin PINy</p> <p>These bits are readable or writable by software, and the corresponding POD bits can be independently set/cleared.</p>

Bit field	Name	Description
		<p><i>Note: when <math>x = A, B, y = 0 \dots 15</math>;</i></p> <p><i>When <math>x = C, y = 13, 14, 15</math>, the remaining bits are reserved, and the reserved bits are read-only;</i></p> <p><i>When <math>x = F, y = 0, 1, 2, 6, 7</math>, the remaining bits are reserved, and the reserved bits are read-only.</i></p>

### 5.3.8 GPIO port bit set/clear register (GPIOx\_PBSC)

Offset address : 0x18

Reset value : 0x0000 0000 (x=A,B,C,F)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PBC15	PBC14	PBC13	PBC12	PBC11	PBC10	PBC9	PBC8	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
w 15	w 14	w 13	w 12	w 11	w 10	w 9	w 8	w 7	w 6	w 5	w 4	w 3	w 2	w 1	w 0
PBS15	PBS14	PBS13	PBS12	PBS11	PBS10	PBS9	PBS8	PBS7	PBS6	PBS5	PBS4	PBS3	PBS2	PBS1	PBS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit field	Name	Description
31:16	PBCy	<p>Clear bit y of port GPIOx (<math>x = A, B, C, F</math>)</p> <p>These bits can only be written.</p> <p>0: Does not affect the corresponding PODy bit</p> <p>1: Clear the corresponding PODy bit to 0</p> <p><i>Note: If the corresponding bits of PBSy and PBCy are set at the same time, the PBSy bit works.</i></p> <p><i>Note: when <math>x = A, B, y = 0 \dots 15</math>;</i></p> <p><i>When <math>x = C, y = 13, 14, 15</math>, the remaining bits are reserved, and the reserved bits are read-only;</i></p> <p><i>When <math>x = F, y = 0, 1, 2, 6, 7</math>, the remaining bits are reserved, and the reserved bits are read-only.</i></p>
15:0	PBSy	<p>Set bit y of port GPIOx (<math>x = A, B, C, F</math>)</p> <p>These bits can only be written.</p> <p>0: Does not affect the corresponding PODy bit</p> <p>1: Set the corresponding PODy bit to 1</p> <p><i>Note: when <math>x = A, B, y = 0 \dots 15</math>;</i></p> <p><i>When <math>x = C, y = 13, 14, 15</math>, the remaining bits are reserved, and the reserved bits are read-only;</i></p> <p><i>When <math>x = F, y = 0, 1, 2, 6, 7</math>, the remaining bits are reserved, and the reserved bits are read-only.</i></p>

### 5.3.9 GPIO port configuration lock register (GPIOx\_PLOCK)

Offset address : 0x1C

Reset value : 0x0000 0000 (x=A,B,C,F)

31	Reserved															17	16
																PLOCKK	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1		rw	0

Bit field	Name	Description
31:17	Reserved	Reserved, the reset value must be maintained
16	PLOCKK	<p>Lock key</p> <p>This bit can be read at any time, it can only be modified by the lock key write sequence.</p> <p>0: Port configuration lock key is not active</p> <p>1: The port configuration lock key bit is activated, and the GPIOx_PLOCK register is locked before the next system reset.</p> <p>Lock key write sequence:</p> <p>write 1 -&gt; write 0 -&gt; write 1 -&gt; read 0 -&gt; (read 1)</p> <p>The last read 1 can be omitted, but can be used to confirm that the lock key has been activated.</p> <p><i>Note: The value of PLOCK[15:0] cannot be changed while operating the lock key write sequence. Any errors in the operation lock key write sequence will not activate the lock key.</i></p>
15:0	PLOCKy	<p>Configuration lock bit for port GPIOx (x = A,B,C,F) pin PINy</p> <p>These bits are readable and writable but can only be written when the PLOCKK bit is 0.</p> <p>0: Do not lock the configuration of the port</p> <p>1: Lock the configuration of the port</p> <p><i>Note: when x = A,B, y = 0...15;</i></p> <p><i>When x = C, y = 13, 14, 15, the remaining bits are reserved, and the reserved bits are read-only;</i></p> <p><i>When x = F, y = 0, 1, 2, 6, 7, the remaining bits are reserved, and the reserved bits are read-only.</i></p>

### 5.3.10 GPIO alternate function low register (GPIOx\_AFL)

Offset address : 0x20

Reset value : 0xFFFF FFFF (x = A,B); 0x0000 0000 (x = C); 0xFF00 0FFF (x = F)

31	AFSEL7[3:0]	28	27	AFSEL6[3:0]	24	23	AFSEL5[3:0]	20	19	AFSEL4[3:0]	16
rw				rw			rw			rw	
15	AFSEL3[3:0]	12	11	AFSEL2[3:0]	8	7	AFSEL1[3:0]	4	3	AFSEL0[3:0]	0
rw				rw			rw			rw	

Bit field	Name	Description
31:28	AFSELy[3:0]	Alternate function configuration bits for port GPIOx (x = A,B,C,F) pins PINy (y = 0...7)
27:24		0000: AF0
23:20		0001: AF1
19:16		0010: AF2
15:12		0011: AF3
11:8		0100: AF4
7:4		0101: AF5
3:0		0110: AF6 0111: AF7 1000: AF8 1001: AF9 1010: AF10 1011: AF11 1100: AF12 1101: AF13 1110: AF14 1111: AF15
		<i>Note: when x = A, B, y = 0...7;</i>
		<i>When x = C, all bits are reserved and reserved bits are read-only. ;</i>
		<i>When x = F, y = 0, 1, 2, 6, 7, the remaining bits are reserved, and the reserved bits are read-only.</i>

### 5.3.11 GPIO alternate function high register (GPIOx\_AFH)

Offset address : 0x24

Reset value : 0xF00F FFFF (x = A); 0xFFFF FFFF (x = B); 0xFF0 0000 (x = C); 0x0000 0000 (x = F)

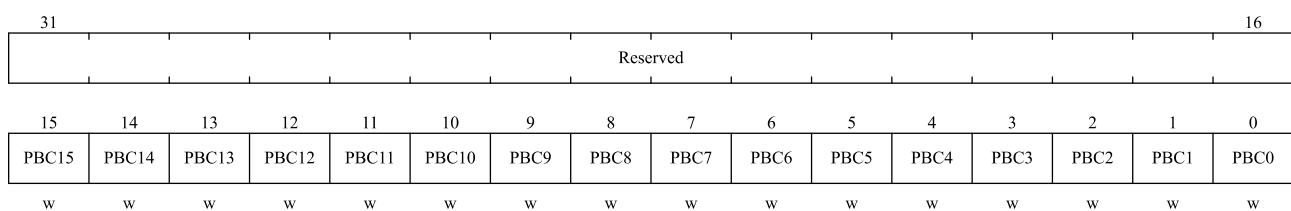
31	AFSEL15[3:0]	28	27	AFSEL14[3:0]	24	23	AFSEL13[3:0]	20	19	AFSEL12[3:0]	16
rw				rw			rw			rw	
15	AFSEL11[3:0]	12	11	AFSEL10[3:0]	8	7	AFSEL9[3:0]	4	3	AFSEL8[3:0]	0
rw				rw			rw			rw	

Bit field	Name	Description
31:28	AFSELy[3:0]	Alternate Function Configuration Bits for Port GPIOx (x = A,B,C,F) Pins PINy (y = 8...15)
27:24		0000: AF0
23:20		0001: AF1
19:16		0010: AF2
15:12		0011: AF3
11:8		0100: AF4
7:4		0101: AF5
3:0		0110: AF6 0111: AF7 1000: AF8 1001: AF9 1010: AF10 1011: AF11 1100: AF12 1101: AF13 1110: AF14 1111: AF15
		<i>Note: when x = A, B, y = 8...15; When x = C, y = 13, 14, 15, the remaining bits are reserved, and the reserved bits are read-only; When x = F, all bits are reserved and reserved bits are read-only.</i>

### 5.3.12 GPIO port bit clear register (GPIOx\_PBC)

Offset address : 0x28

Reset value : 0x0000 0000 (x=A,B,C,F)



Bit field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained
15:0	PBCy	Clear bit y of port GPIOx (x = A,B,C,F) These bits can only be written. 0: No effect on the corresponding PODy bit 1: Clear the corresponding PODy bit to 0 <i>Note: when x = A,B, y = 0...15;</i>

Bit field	Name	Description
		<p>When <math>x = C, y = 13, 14, 15</math>, the remaining bits are reserved, and the reserved bits are read-only;</p> <p>When <math>x = F, y = 0, 1, 2, 6, 7</math>, the remaining bits are reserved, and the reserved bits are read-only.</p>

### 5.3.13 GPIO driver strength configuration register (GPIOx\_DS)

Offset address : 0x2C

Reset value : 0x0000 0000 (x=A,B,C,F)

31															16	
Reserved																
15	DS15	DS14	DS13	DS12	DS11	DS10	DS9	DS8	DS7	DS6	DS5	DS4	DS3	DS2	DS1	0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained
15:0	DSy	<p>Drive capability configuration bits for port GPIOx (<math>x = A, B, C, F</math>) pins PINy</p> <p>0: High drive capability (16mA(5V)/8mA(3.3V)/4mA(1.8V))</p> <p>1: Low drive capability (8mA(5V)/4mA(3.3V)/2mA(1.8V))</p> <p>Note: when <math>x = A, B, y = 0 \dots 15</math>;</p> <p>When <math>x = C, y = 13, 14, 15</math>, the remaining bits are reserved, and the reserved bits are read-only;</p> <p>When <math>x = F, y = 0, 1, 2, 6, 7</math>, the remaining bits are reserved, and the reserved bits are read-only.</p>

## 5.4 AFIO Registers

### 5.4.1 AFIO register overview

AFIO base address: 0x40010000

Table 5-37 AFIO register overview

Offset	Register	31	30	29	28	27	26	25	24	SPI2_NSS	SPI1_NSS	22	21	ADC_ETRI	ADC_ATRR	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
000h	AFIO_CFG	Reserved								Reserved		Reserved		EXTI_ETRI [3:0]		EXTI_ETRR [3:0]		Reserved												Reserved						
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
008h	AFIO_EXTI_CFG1	Reserved								EXTI1_CFG [3:0]		EXTI2_CFG [3:0]		EXTI11_CFG [3:0]		EXTI10_CFG [3:0]		Reserved												Reserved						
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00Ch	AFIO_EXTI_CFG2	Reserved								EXTI7_CFG [3:0]		EXTI6_CFG [3:0]		EXTI5_CFG [3:0]		EXTI4_CFG [3:0]		Reserved												Reserved						

### 5.4.2 AFIO configuration register (AFIO\_CFG)

Offset address : 0x00

Reset value : 0x0000 0000

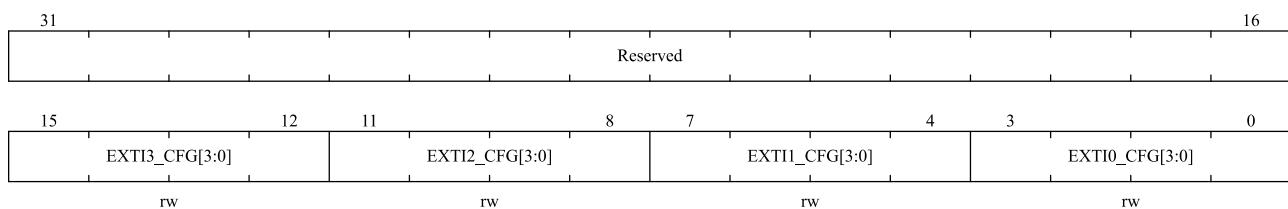
Bit field	Name	Description
31:24	Reserved	Reserved, the reset value must be maintained
23	SPI2_NSS	NSS mode selection bit of SPI2 (NSS is configured in AFIO push-pull mode). 0: NSS is high impedance when idle 1: NSS is high level when idle
22	SPI1_NSS	NSS mode selection bit of SPI1 (NSS is configured in AFIO push-pull mode). 0: NSS is high impedance when idle 1: NSS is high level when idle
21	Reserved	Reserved, the reset value must be maintained
20	ADC_ETRI	ADC injection conversion external trigger remapping This bit can be set to '1' or '0' by software. It controls the trigger input connected to the external trigger for ADC injection conversion. 0: ADC injection conversion external trigger is connected to EXTI (0-15) 1: ADC injection conversion external trigger is connected to TIM8_CH4.
19	ADC_ETRR	ADC regular conversion external trigger remapping This bit can be set to '1' or '0' by software. It controls the trigger input connected to the ADC regular conversion external trigger. 0: ADC regular conversion external trigger is connected to EXTI (0-15) 1: The ADC regular conversion external trigger is connected to TIM8_TRGO.
18:15	EXTI_ETRI[3:0]	Select EXTI Line injection to convert external trigger remapping 0000: select EXTI0 injection conversion external trigger 0001: Select EXTI1 injection to convert external trigger

Bit field	Name	Description
		... 1111: Select EXTI15 injection conversion external trigger
14:11	EXTI_ETRR[3:0]	Select EXTI Line regular to convert external trigger remapping 0000: Select EXTI0 regular to convert external trigger 0001: Select EXTI1 regular to convert external trigger ... 1111: Select EXTI15 regular to convert external trigger
10:0	Reserved	Reserved, the reset value must be maintained

### 5.4.3 AFIO external interrupt configuration register 1 (AFIO\_EXTI\_CFG1)

Offset address : 0x08

Reset value : 0x0000 0000

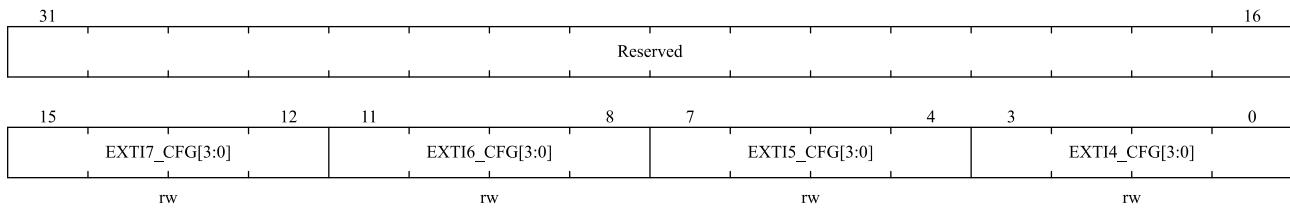


Bit field	Name	Description
31:30	Reserved	Reserved, the reset value must be maintained
15:0	EXTIx_CFG[3:0]	<b>EXTIx configuration (x = 0...3)</b> These bits are readable and writable by software and are used to select the input source for the EXTIx external interrupt. <b>EXTI0 configuration:</b> 0000: PA0 pin    0001: PB0 pin    0010: reserved 0101: PF0 pin    Other: reserved <b>EXTI1 configuration:</b> 0000: PA1 pin    0001: PB1 pin    0010: reserved 0101: PF1 pin    Other: reserved <b>EXTI2 configuration:</b> 0000: PA2 pin    0001: PB2 pin    0010: reserved 0101: PF2 pin    Other: reserved <b>EXTI3 configuration:</b> 0000: PA3 pin    0001: PB3 pin    0010: reserved 0101: reserved    Other: reserved

### 5.4.4 AFIO external interrupt configuration register 2 (AFIO\_EXTI\_CFG2)

Offset address : 0x0C

Reset value : 0x0000 0000

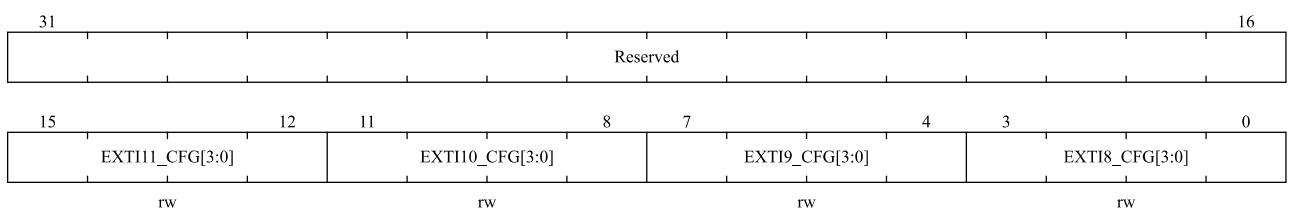


Bit field	Name	Description
31:30	Reserved	Reserved, the reset value must be maintained
15:0	EXTIx_CFG[3:0]	<p>EXTIx configuration (x = 4..7)</p> <p>These bits are readable and writable by software and are used to select the input source for the EXTIx external interrupt.</p> <p>EXTI4 configuration:</p> <ul style="list-style-type: none"> <li>0000: PA4 pin    0001: PB4 pin    0010: reserved</li> <li>0101: reserved    Other: reserved</li> </ul> <p>EXTI5 configuration:</p> <ul style="list-style-type: none"> <li>0000: PA5 pin    0001: PB5 pin    0010: reserved</li> <li>0101: reserved    Other: reserved</li> </ul> <p>EXTI6 configuration:</p> <ul style="list-style-type: none"> <li>0000: PA6 pin    0001: PB6 pin    0010: reserved</li> <li>0101: PF6 pin    Other: reserved</li> </ul> <p>EXTI7 configuration:</p> <ul style="list-style-type: none"> <li>0000: PA7 pin    0001: PB7 pin    0010: reserved</li> <li>0101: PF7 pin    Other: reserved</li> </ul>

#### 5.4.5 AFIO external interrupt configuration register 3 (AFIO\_EXTI\_CFG3)

Offset address : 0x10

Reset value : 0x0000 0000



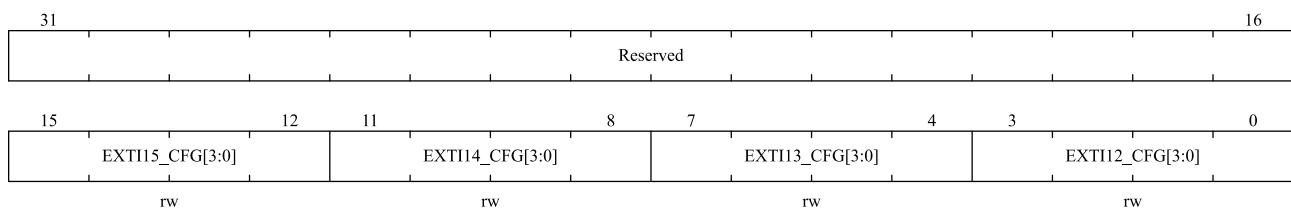
Bit field	Name	Description
31:30	Reserved	Reserved, the reset value must be maintained
15:0	EXTIx_CFG[3:0]	<p>EXTIx configuration (x = 8..11)</p> <p>These bits are readable and writable by software and are used to select the input source for the EXTIx external interrupt.</p>

Bit field	Name	Description
		<p>EXTI18 configuration:</p> <p>0000: PA8 pin    0001: PB8 pin    0010: reserved      0101: reserved    Other: reserved</p> <p>EXTI19 configuration:</p> <p>0000: PA9 pin    0001: PB9 pin    0010: reserved      0101: reserved    Other: reserved</p> <p>EXTI10 configuration:</p> <p>0000: PA10 pin    0001: PB10 pin    0010: reserved      0101: reserved    Other: reserved</p> <p>EXTI11 configuration:</p> <p>0000: PA11 pin    0001: PB11 pin    0010: reserved      0101: reserved    Other: reserved</p>

## 5.4.6 AFIO external interrupt configuration register 4 (AFIO\_EXTI\_CFG4)

Offset address : 0x14

Reset value : 0x0000 0000



Bit field	Name	Description
31:30	Reserved	Reserved, the reset value must be maintained
15:0	EXTIx_CFG[3:0]	<p>EXTIx configuration (x = 12...15)</p> <p>These bits are readable and writable by software and are used to select the input source for the EXTIx external interrupt.</p> <p>EXTI12 configuration:</p> <p>0000: PA12 pin    0001: PB12 pin    0010: reserved      0101: reserved    Other: reserved</p> <p>EXTI13 configuration:</p> <p>0000: PA13 pin    0001: PB13 pin    0010: PC13 pin      0101: reserved    Other: reserved</p> <p>EXTI14 configuration:</p> <p>0000: PA14 pin    0001: PB14 pin    0010: PC14 pin      0101: reserved    Other: reserved</p> <p>EXTI15 configuration:</p> <p>0000: PA15 pin    0001: PB15 pin    0010: PC15 pin      0101: reserved    Other: reserved</p>

## 6 Interrupts and events

### 6.1 Nested vectored interrupt controller

#### Features

- 32 maskable interrupt channels (not including 16 Cortex®-M0 neutral line);
- 4 programmable priorities (using 2 interrupt priorities);
- Low latency exception and interrupt handling;
- Power management control;
- The realization of system control register;

The nested vector interrupt Controller (NVIC) is closely linked to the processor core, enabling low latency interrupt processing and efficient processing of late interrupts. The nested vector interrupt controller manages interrupts including core exceptions.

#### 6.1.1 SysTick calibration value register

The system tick calibration value is fixed at 6000. When the system tick clock is set to 6MHz (the maximum value of HCLK/8), 1ms time reference is generated.

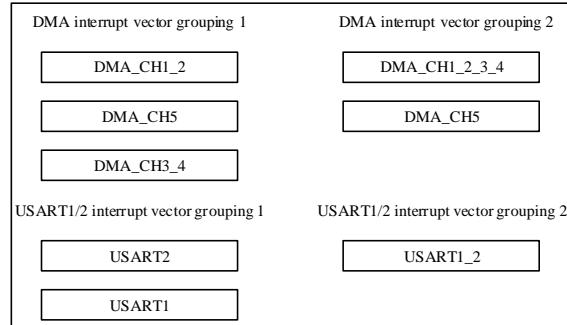
#### 6.1.2 Interrupt and exception vectors.

Table 6-1 Vector table

Position	Priority	Priority type	Name	Description	Address
-	-	-	-	Reserved	0x0000 0000
-	-3	Fixed	Reset	Reset	0x0000 0004
-	-2	Fixed	NMI	Non-maskable interrupt.RCC clock security system (CSS) is connected to the NMI vector.	0x0000 0008
-	-1	Fixed	HardFault	All types of errors (fault)	0x0000 000C
-	3	Settable	SVCall	System services invoked by SWI directives	0x0000 002C
-	5	Settable	PendSV	System service requests that can be pending	0x0000 0038
-	6	Settable	SysTick	System tick timer	0x0000 003C
0	7	Settable	WWDG	Window watchdog interrupt	0x0000 0040
1	8	Settable	PVD	PVD interrupt connected to EXTI line 16	0x0000 0044
2	9	Settable	RTC	RTC interrupt connected to EXTI lines 17, 19 and 20	0x0000 0048

Position	Priority	Priority type	Name	Description	Address
3	10	Settable	MMU	MMU global interrupt	0x0000 004C
4	11	Settable	FLASH	Flash global interrupt	0x0000 0050
5	12	Settable	RCC	RCC global interrupt	0x0000 0054
6	13	Settable	EXTI0_1	The EXTI line [1:0] interrupt	0x0000 0058
7	14	Settable	EXTI2_3	The EXTI line [3:2] interrupt	0x0000 005C
8	15	Settable	EXTI4_15	The EXTI line [15:4] interrupt	0x0000 0060
9	16	-	-	Reserved	0x0000 0064
10	17	Settable	DMA_CH1_2	DMA channel 1/2 interrupt	0x0000 0068
11	18	Settable	DMA_CH1_2_3_4	DMA channel 1/2/3/4 interrupt	0x0000 006C
12	19	Settable	DMA_CH5	DMA channel 5 interrupt	0x0000 0070
13	20	Settable	TIM1_BRK_UP_TRG_COM	TIM1 brakes, updates, triggers and communication interrupt	0x0000 0074
14	21	Settable	TIM1_CC	TIM1 capture comparison interrupt	0x0000 0078
15	22	Settable	DMA_CH3_4	DMA channel /3/4 interrupt	0x0000 007C
16	23	Settable	TIM3	TIM3 global interrupt	0x0000 0080
17	24	Settable	USART2	USART2 global interrupt	0x0000 0084
18	25	Settable	TIM8_BRK_UP_TRG_COM	TIM8 brakes, updates, triggers and communication interrupt	0x0000 0088
19	26	Settable	TIM8_CC	TIM8 capture comparison interrupt	0x0000 008C
20	27	Settable	LPTIM/TIM6	LPTIM (connected to EXTI line 23) /TIM6 global interrupt	0x0000 0090
21	28	Settable	ADC	ADC global interrupt	0x0000 0094
22	29	Settable	SPI2	SPI2 global interrupt	0x0000 0098
23	30	Settable	I2C1	I2C1 global interruption	0x0000 009C
24	31	Settable	I2C2	I2C2 global interrupt	0x0000 00A0
25	32	Settable	SPI1	SPI1 global interrupt	0x0000 00A4
26	33	Settable	HDIV/SQRT	Divider/square operator global interrupt	0x0000 00A8
27	34	Settable	RAMC_ERR	RAMC_ERR global interrupt	0x0000 00AC
28	35	Settable	USART1/USART2	USART1/USART2 global interrupt	0x0000 00B0
29	36	Settable	LPUART	LPUART global interrupt (connected to EXTI line 22)	0x0000 00B4
30	37	Settable	USART1	USART1 global interrupt	0x0000 00B8
31	38	Settable	COMP	COMP (connected to EXTI line 18)	0x0000 00BC

DMA and USART1/2 in the interrupt vector scale have several pairs of vectors with repeated functions. The repeated vectors are divided into two groups below:



When using DMA and USART1/2 interrupts, you can choose either group arbitrarily, but only one group at a time.

## 6.2 External interrupt/event controller (EXTI)

### 6.2.1 Introduction

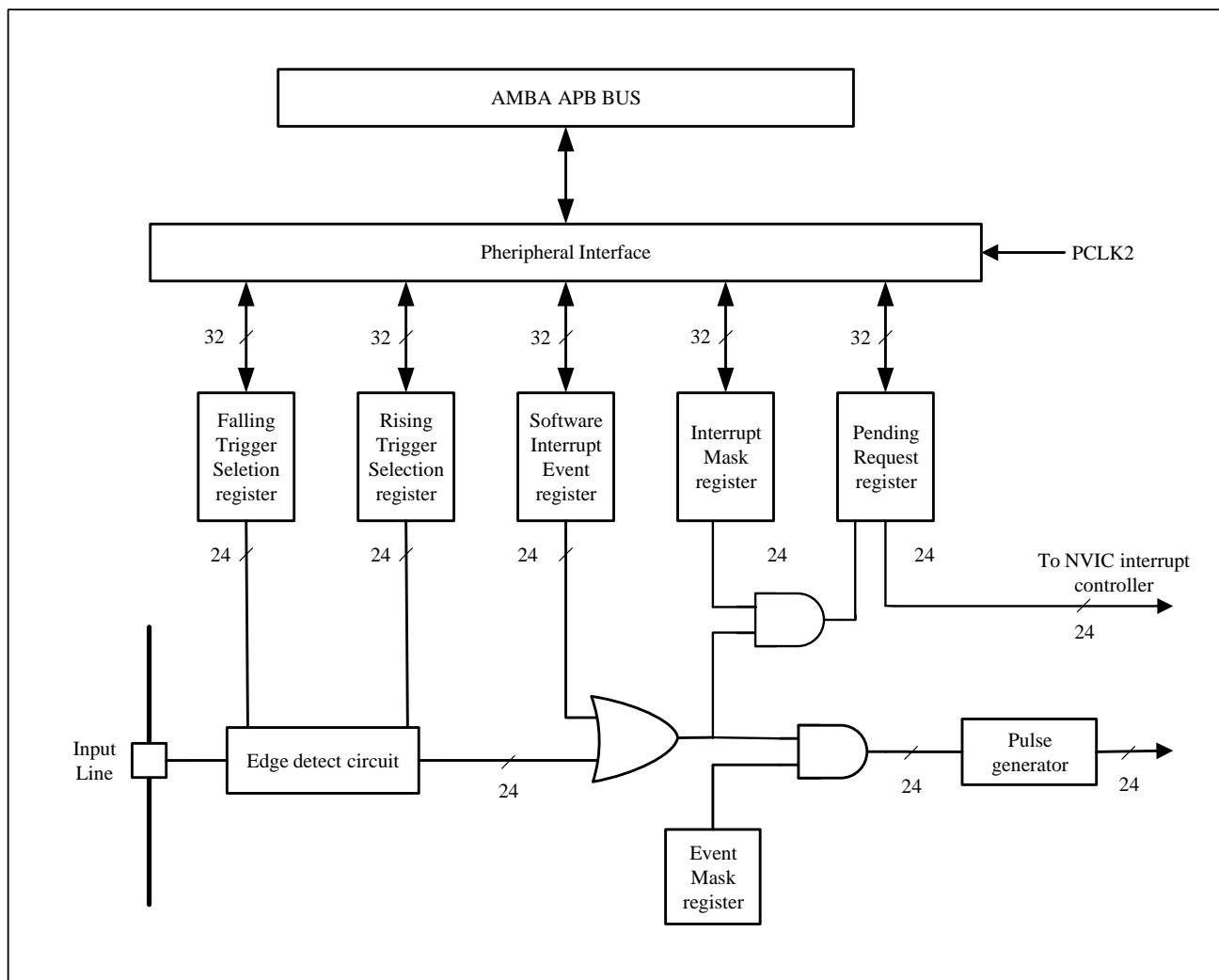
The external interrupt/event controller contains 24 edge detection circuits that generate interrupt/event triggers. Each input line can be independently configured with pulse or pending input types, and three trigger event types including rising edge, falling edge or double edge, which can also be independently shielded. Interrupt requests that hold the state line in the pending register can be cleared by writing '1' in the corresponding bit of the pending register.

### 6.2.2 Main features

The main features of EXTI controller are as follows:

- Supports 24 software interrupt/event requests
- Interrupts/events corresponding to each input line can be configured to trigger or mask independently
- Each interrupt line has an independent state bit
- Support for pulse or pending input types
- 3 trigger events are supported: rising edge, falling edge, and double edge
- Can wake up to exit low power mode

Figure 6-1 External interrupt/event controller block diagram



### 6.2.3 Functional description

EXTI contains 24 interrupts, 16 from I/O pins and 8 from internal modules. To generate interrupts, the NVIC interrupt channel of the external interrupt controller must be configured to enable the appropriate interrupt line. Select rising edge, falling edge, or double edge trigger event types by edge trigger configuration registers EXTI\_RT\_CFG and EXTI\_FT\_CFG, and write '1' to the corresponding bit of interrupt masking register EXTI\_IMASK to allow interrupt requests. When a preset edge trigger polarity is detected on the external interrupt line, an interrupt request is generated and the corresponding pending bit is set to '1'. Writing '1' to the corresponding bit of the pending register clears the interrupt request.

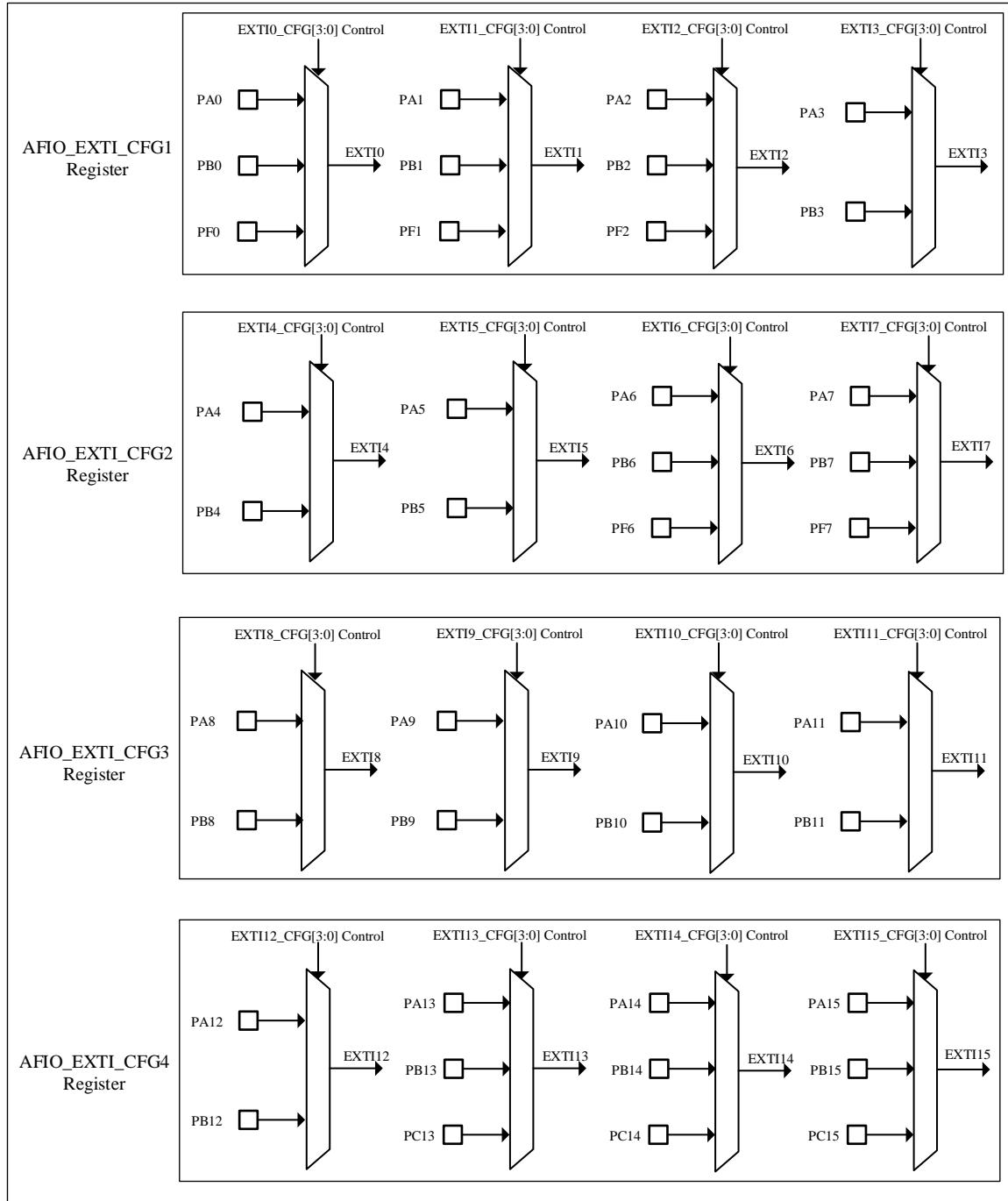
To generate events, the corresponding event line must be configured and enabled. According to the desired edge detection polarity, set up the rise/fall edge trigger configuration register, while writing '1' in the corresponding bit of the event masking register to allow interrupt requests. When a preset edge occurs on an event line, an event request pulse is generated and the corresponding pending bit is not set to '1'.

In addition, interrupt/event requests can also be generated by software by writing a '1' in the software interrupt/event register.

- Hardware interrupt configuration, select and configure 24 lines as interrupt sources as required:
  - ◆ Configure the mask bit (EXTI\_IMASK) for 24 interrupt lines.
  - ◆ Configure the selected disconnection trigger configuration bits (EXTI\_RT\_CFG and EXTI\_FT\_CFG);
  - ◆ Configure the enable and mask bits of the NVIC interrupt channel corresponding to the external interrupt controller so that the requests in the 24 interrupt lines can be correctly responded to.
- Hardware event configuration: Select 24 lines as event sources as required:
  - ◆ Configure the mask bit (EXTI\_EMASK) for 24 event lines.
  - ◆ Configure the trigger configuration bits for the selected event line (EXTI\_RT\_CFG and EXTI\_FT\_CFG).
- Software interrupt/event configuration, select 24 lines as software interrupt/event lines as required:
  - ◆ Configure 24 interrupt/event line mask bits (EXTI\_IMASK and EXTI\_EMASK).
  - ◆ Configure the request bit of the software interrupt event register (EXTI\_SWIE).

## 6.2.4 EXTI line mapping

Figure 6-2 External interrupt generic I/O mapping



To configure external interrupts/events on the GPIO line using AFIO\_EXTI\_CFG1~4, the AFIO clock must be enabled first. Universal I/O ports are connected to 16 external interrupt/event lines as shown above. The connection mode of the other 8 EXTI lines is as follows:

- EXTI line 16 is connected to the PVD output

- EXTI line 17 is connected to the RTC alarm
- EXTI line 18 is connected to the COMP wake up event
- EXTI line 19 is connected to the RTC tamper for detection or timestamp wake up event
- EXTI line 20 is connected to the RTC wake up event
- EXTI line 21 is reserved
- EXTI line 22 is connected to the LPUART wake up event
- EXTI line 23 is connected to the LPTIM wake up event

## 6.3 EXTI Registers

EXTI base address: 0x40010400

### **6.3.1 EXTI register overview**

Table 6-2 EXTI register overview

Offset	Register	TSSEL[3:0]			
000h	EXTI_IMASK	31	30	29	28
	Reset Value	Reserved	Reserved	Reserved	Reserved
004h	EXTI_EMASK	27	26	25	24
	Reset Value	Reserved	Reserved	Reserved	Reserved
008h	EXTI_RT_CFG	23	22	21	20
	Reset Value	RT_CFG23	RT_CFG22	RT_CFG21	RT_CFG20
00Ch	EXTI_FT_CFG	23	22	21	20
	Reset Value	FT_CFG23	FT_CFG22	FT_CFG21	FT_CFG20
010h	EXTI_SWIE	19	18	17	16
	Reset Value	SWE23	SWE22	SWE21	SWE20
014h	EXTI_PEND	15	14	13	12
	Reset Value	PEND23	PEND22	PEND21	PEND20
018h	EXTI_TS_SEL	11	10	9	8
	Reset Value	SWE11	SWE10	SWE9	SWE8
	Reserved	RT_CFG11	RT_CFG10	RT_CFG9	RT_CFG8
		EMASK11	EMASK10	EMASK9	EMASK8
		0	0	0	0

### 6.3.2 Interrupt mask register(EXTI\_IMASK)

Address offset : 0x00

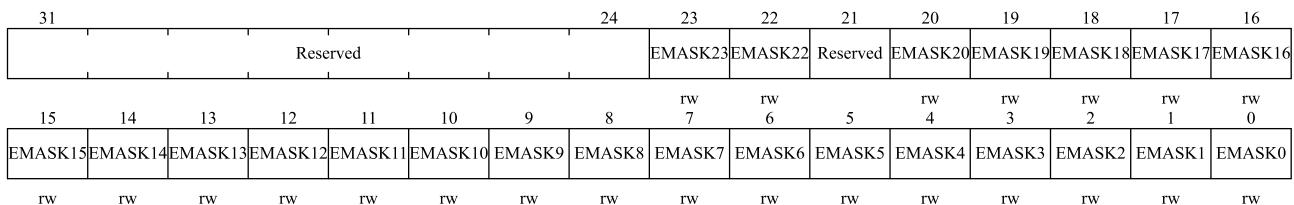
Reset value : 0x00000000

Bit field	name	describe
31:24	Reserved	Reserved, the reset value must be maintained.
23:22	IMASKx	Interrupt mask on line x. (x is 22,23) 0: Mask the interrupt request from line x; 1: open the interrupt request from line x
21	Reserved	Reserved, the reset value must be maintained.
20:0	IMASKx	Interrupt mask on line x. (x is 0,1,2...19,20) 0: Mask the interrupt request from line x; 1: open the interrupt request from line x

### 6.3.3 Event mask register(EXTI\_EMASK)

Address offset : 0x04

Reset value : 0x00000000



Bit field	name	describe
31:24	Reserved	Reserved, the reset value must be maintained.
23:22	EMASKx	Event masking on line x. (x is 22,23) 0: Masking the event request from line x; 1: open the event request from line x
21	Reserved	Reserved, the reset value must be maintained.
20:0	EMASKx	Event masking on line x. (x is 0,1,2...19,20) 0: Masking the event request from line x; 1: open the event request from line x

### 6.3.4 Rising edge trigger selection register(EXTI\_RT\_CFG)

Address offset : 0x08

Reset value : 0x00000000

31	Reserved								24	23	22	21	20	19	18	17	16
									RT_CFG23	RT_CFG22	Reserved	RT_CFG20	RT_CFG19	RT_CFG18	RT_CFG17	RT_CFG16	
15	14	13	12	11	10	9	8	rw 7	rw 6	rw 5	rw 4	rw 3	rw 2	rw 1	rw 0		
RT_CFG15	RT_CFG14	RT_CFG13	RT_CFG12	RT_CFG11	RT_CFG10	RT_CFG9	RT_CFG8	RT_CFG7	RT_CFG6	RT_CFG5	RT_CFG4	RT_CFG3	RT_CFG2	RT_CFG1	RT_CFG0		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit field	name	describe
31:24	Reserved	Reserved, the reset value must be maintained.
23:22	RT_CFGx	The rising edge on line x triggers the configuration bit. (x is 22,23) 0: Disables rising edge trigger (interrupts and events) on input line x. 1: Enable rising edge trigger (interrupts and events) on input line x.
21	Reserved	Reserved, the reset value must be maintained.
20:0	RT_CFGx	The rising edge on line x triggers the configuration bit. (x is 0,1,2...19,20) 0: Disables rising edge trigger (interrupts and events) on input line x. 1: Enable rising edge trigger (interrupts and events) on input line x.

### 6.3.5 Falling edge trigger selection register(EXTI\_FT\_CFG)

Address offset : 0x0C

Reset value : 0x00000000

31	Reserved								24	23	22	21	20	19	18	17	16
									FT_CFG23	FT_CFG22	Reserved	FT_CFG20	FT_CFG19	FT_CFG18	FT_CFG17	FT_CFG16	
15	14	13	12	11	10	9	8	rw 7	rw 6	rw 5	rw 4	rw 3	rw 2	rw 1	rw 0		
FT_CFG15	FT_CFG14	FT_CFG13	FT_CFG12	FT_CFG11	FT_CFG10	FT_CFG9	FT_CFG8	FT_CFG7	FT_CFG6	FT_CFG5	FT_CFG4	FT_CFG3	FT_CFG2	FT_CFG1	FT_CFG0		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit field	name	describe
31:24	Reserved	Reserved, the reset value must be maintained.
23:22	FT_CFGx	The falling edge on line x triggers the configuration bit. (x is 22,23) 0: Disables falling edge trigger (interrupts and events) on input line x. 1: Enable falling edge trigger (interrupts and events) on input line x is allowed.
21	Reserved	Reserved, the reset value must be maintained.
20:0	FT_CFGx	The falling edge on line x triggers the configuration bit. (x is 0,1,2...19,20) 0: Disables falling edge trigger (interrupts and events) on input line x. 1: Enable falling edge trigger (interrupts and events) on input line x is allowed.

### 6.3.6 Software interrupt enable register(EXTI\_SWIE)

Address offset : 0x10

Reset value : 0x00000000

31	Reserved								24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	rw	7	rw	6	5	rw	4	rw	3	rw
SWIE15	SWIE14	SWIE13	SWIE12	SWIE11	SWIE10	SWIE9	SWIE8	SWIE7	SWIE6	SWIE5	SWIE4	SWIE3	SWIE2	SWIE1	SWIE0	rw	0

Bit field	name	describe
31:24	Reserved	Reserved, the reset value must be maintained.
23:22	SWIEx	<p>Software interrupt on line x. (x is 22,23)</p> <p>When the bit is' 0 ', writing '1' sets the corresponding pending bit in EXTI_PEND. If this interrupt is allowed in EXTI_IMASK and EXTI_EMASK, an interrupt will be generated.</p> <p><i>Note: This bit can be cleared to '0' by writing '1' to clear the corresponding bit of EXTI_PEND.</i></p>
21	Reserved	Reserved, the reset value must be maintained.
20:0	SWIEx	<p>Software interrupt on line x. (x is 0,1,2...19,20)</p> <p>When the bit is' 0 ', writing '1' sets the corresponding pending bit in EXTI_PEND. If this interrupt is allowed in EXTI_IMASK and EXTI_EMASK, an interrupt will be generated.</p> <p><i>Note: This bit can be cleared to '0' by writing '1' to clear the corresponding bit of EXTI_PEND.</i></p>

### 6.3.7 Interrupt request pending register(EXTI\_PEND)

Address offset : 0x14

Reset value : 0x00000000

31	Reserved								24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	rc_w1	7	rc_w1	6	5	rc_w1	4	rc_w1	3	rc_w1
PEND15	PEND14	PEND13	PEND12	PEND11	PEND10	PEND9	PEND8	PEND7	PEND6	PEND5	PEND4	PEND3	PEND2	PEND1	PEND0	rc_w1	0

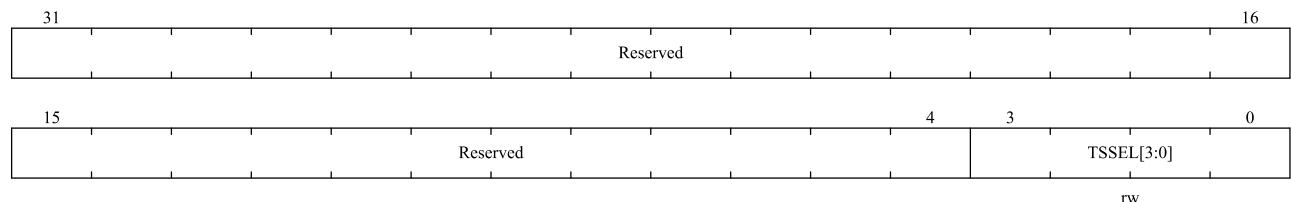
Bit field	name	describe
31:24	Reserved	Reserved, the reset value must be maintained.
23:22	PENDx	<p>Hang bit on line x. (x is 22,23)</p> <p>0: No pending request occurred.</p>

		<p>1: A pending trigger request has occurred.</p> <p>This bit is set to '1' when a selected edge trigger event occurs on the external interrupt line. It can be cleared by writing '1' to the bit, or by changing the polarity of the edge detection.</p>
21	Reserved	Reserved, the reset value must be maintained.
20:0	PENDx	<p>Hang bit on line x. (x is 0,1,2...19,20)</p> <p>0: No pending request occurred.</p> <p>1: A pending trigger request has occurred.</p> <p>This bit is set to '1' when a selected edge trigger event occurs on the external interrupt line. It can be cleared by writing '1' to the bit, or by changing the polarity of the edge detection.</p>

### 6.3.8 RTC Timestamp trigger source selection register (EXTI\_TS\_SEL)

Address offset : 0x18

Reset value : 0x00000000



Bit field	name	describe
31:4	Reserved	Reserved, the reset value must be maintained.
3:0	TSSEL[3:0]	<p>Select external interrupt input as trigger source of timestamp event.</p> <p>0: Select EXTI0 as the trigger source of timestamp event;</p> <p>1: select EXTI1 as the trigger source of timestamp event;</p> <p>.....</p> <p>15: Select EXTI15 as the trigger source of timestamp events.</p>

## 7 DMA controller

### 7.1 Introduction

The DMA controller can access totally 5 AHB slaves: Flash, SRAM, ADC, APB1 and APB2. DMA Controller is controlled by CPU to perform fast data movement from source to destination. After configuration, data can be transferred without CPU intervention. Thus, CPU can be released for other computation/control tasks or save overall system power consumption.

MCU's main backbone is a multi-layer AHB-Lite bus structure with round-robin arbitration scheme. DMA and CPU core can access different slaves in parallel or same slaves sequentially.

DMA controller has 5 logic channels. Each logic channel is to serve memory access requests from single or multiple peripherals. Internal arbiter controls the priority of different DMA channels.

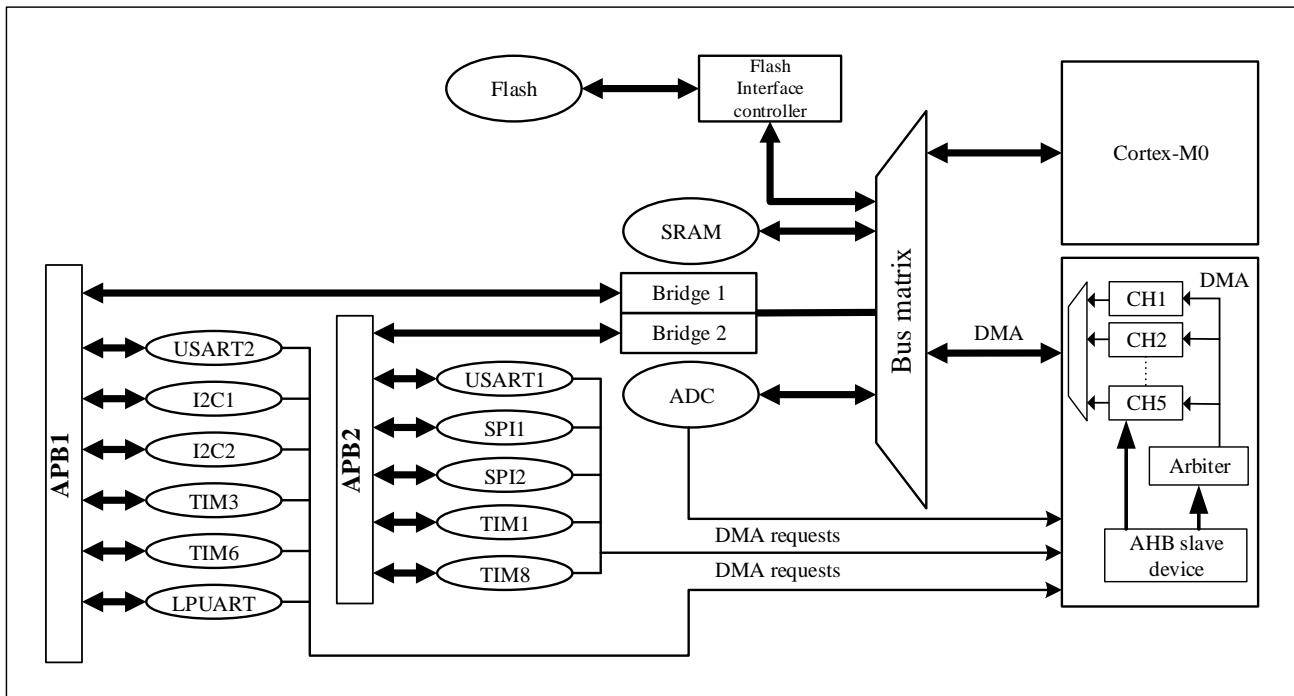
### 7.2 Main features

DMA main features:

- 5 DMA channels which can be configured independently.
- Each DMA channel supports hardware requests and software triggers to initiate transfer, and is configured by software.
- Each DMA channel has dedicated software priority level (DMA\_CHCFGx.PRIOLVL [1:0] bits, corresponding to 4 levels of priority) which can be configured individually. Channels with the same software priority level will further compare hardware index (channel number) to decide final priority (lower index number channel will have higher priority).
- Configurable source and destination size. Address setting should correspond to data size.
- Configurable circular transfer mode for each channel.
- Each channel has 3 independent event flags and interrupts (Transfer complete, Half transfer, Transfer error), and 1 global interrupt flag (set by logical OR of 3 events).
- Support three transfer types which are Memory-to-Memory, Memory-to-Peripheral and Peripheral-to-Memory.
- Access totally 5 AHB slaves: Flash, SRAM, ADC, APB1 and APB2.
- Configurable data transmit number (0~65535).

## 7.3 Block diagram

Figure 7-1 DMA block diagram



## 7.4 Function description

DMA controller and Cortex<sup>TM</sup>-M0 core share the same system data bus. When CPU and DMA access the same target (RAM or peripheral) at the same time, DMA request will suspend CPU from accessing the system bus for several cycles, and the bus arbiter will perform cyclic scheduling. This allows the CPU to get at least half of the system bus (memory or peripheral) bandwidth.

### 7.4.1 DMA operation

A DMA request can be triggered by hardware peripherals or software, and the DMA controller processes the request according to the priority level of the channel. The data is read from the source address according to the configured transfer address and bit width, and then the read data is stored in the destination address space. After one operation, the controller calculates the number of remaining transfers and updates the source address and the destination address of the next transfer.

Each DMA data transfer consists of three operations:

- Data access: determine the source address (DMA\_PADDRx or DMA\_MADDRx) according to the transfer direction and read data from the source address.
- Data storage: determine the destination address (DMA\_PADDRx or DMA\_MADDRx) according to the transfer direction and store the read data into the destination address space.
- Calculate the number of outstanding operations, perform a decrement operation of the DMA\_TXNUMx register,

and update the source and destination addresses of the next operation.

### 7.4.2 Channel priority and arbitration

The DMA uses an arbitration strategy to handle multiple requests from different channels. The priority of each channel is programmable in the channel control register (DMA\_CHCFGx).

4 levels of priority:

- ◆ Very high priority
- ◆ High priority
- ◆ Medium priority
- ◆ Low priority

By default, channel with lower index has higher priority if the programmed priority is the same.

For memory to memory transfer, re-arbitration is carried on after 4 transfer operations.

For transfer related to periphery, re-arbitration is carried on after each transfer operation.

### 7.4.3 DMA channels and number of transfers

Each channel can perform DMA transfer between the peripheral register at the specified address and the memory address. The number of data transferred by DMA is programmable, and the maximum supported value is 65535. The DMA\_TXNUM register is decremented after each transfer.

### 7.4.4 Programmable data bit width, alignment and endians

Peripheral and memory transfer data bit width supports byte, half-word and word, which can be programmed through DMA\_CHCFGx.PSIZE and DMA\_CHCFGx.MSIZE.

When DMA\_CHCFGx.PSIZE and DMA\_CHCFGx.MSIZE are different, the DMA module aligns the data according to the below.

Table 7-1 Programmable data width and endian operation (when PINC = MINC = 1)

Source width (bit)	Destina- tion width (bit)	Number of transfer (bit)	Source: Address / data	Transfer operations (R: Read, W: Write)	Destination: Address / data
8	8	4	0x0 / B0	1: R B0 [7:0] @0x0, W B0 [7:0] @0x0	0x0 / B0
			0x1 / B1	2: R B1 [7:0] @0x1, W B1 [7:0] @0x1	0x1 / B1
			0x2 / B2	3: R B2 [7:0] @0x2, W B2 [7:0] @0x2	0x2 / B2
			0x3 / B3	4: R B3 [7:0] @0x3, W B3 [7:0] @0x3	0x3 / B3
8	16	4	0x0 / B0	1: R B0 [7:0] @0x0, W 00B0 [15:0] @0x0	0x0 / 00B0
			0x1 / B1	2: R B1 [7:0] @0x1, W 00B1 [15:0] @0x2	0x2 / 00B1

Source width (bit)	Destina- tion width (bit)	Number of transfer (bit)	Source: Address / data	Transfer operations (R: Read, W: Write)	Destination: Address / data
			0x2 / B2 0x3 / B3	3: R B2 [7:0] @0x2, W 00B2 [15:0] @0x4 4: R B3 [7:0] @0x3, W 00B3 [15:0] @0x6	0x4 / 00B2 0x6 / 00B3
8	32	4	0x0 / B0	1: R B0 [7:0] @0x0, W 000000B0 [31:0] @0x0	0x0 / 000000B0
			0x1 / B1	2: R B1 [7:0] @0x1, W 000000B1 [31:0] @0x4	0x4 / 000000B1
			0x2 / B2	3: R B2 [7:0] @0x2, W 000000B2 [31:0] @0x8	0x8 / 000000B2
			0x3 / B3	4: R B3 [7:0] @0x3, W 000000B3 [31:0] @0xC	0xC / 000000B3
16	8	4	0x0 / B1B0	1: R B1B0 [15:0] @0x0, W B0 [7:0] @0x0	0x0 / B0
			0x2 / B3B2	2: R B3B2 [15:0] @0x2, W B2 [7:0] @0x1	0x1 / B2
			0x4 / B5B4	3: R B5B4 [15:0] @0x4, W B4 [7:0] @0x2	0x2 / B4
			0x6 / B7B6	4: R B7B6 [15:0] @0x6, W B6 [7:0] @0x3	0x3 / B6
16	16	4	0x0 / B1B0	1: R B1B0 [15:0] @0x0, W B1B0 [15:0] @0x0	0x0 / B1B0
			0x2 / B3B2	2: R B3B2 [15:0] @0x2, W B3B2 [15:0] @0x2	0x2 / B3B2
			0x4 / B5B4	3: R B5B4 [15:0] @0x4, W B5B4 [15:0] @0x4	0x4 / B5B4
			0x6 / B7B6	4: R B7B6 [15:0] @0x6, W B7B6 [15:0] @0x6	0x6 / B7B6
16	32	4	0x0 / B1B0	1: R B1B0 [15:0] @0x0, W 0000B1B0 [31:0] @0x0	0x0 / 0000B1B0
			0x2 / B3B2	2: R B3B2 [15:0] @0x2, W 0000B3B2 [31:0] @0x4	0x4 / 0000B3B2
			0x4 / B5B4	3: R B5B4 [15:0] @0x4, W 0000B5B4 [31:0] @0x8	0x8 / 0000B5B4
			0x6 / B7B6	4: R B7B6 [15:0] @0x6, W 0000B7B6 [31:0] @0xC	0xC / 0000B7B6
32	8	4	0x0 / B3B2B1B0	1: R B3B2B1B0 [31:0] @0x0, W B0 [7:0] @0x0	0x0 / B0
			0x4 / B7B6B5B4	2: R B7B6B5B4 [31:0] @0x4, W B4 [7:0] @0x1	0x1 / B4
			0x8 / BBBAB9B8	3: R BBBAB9B8 [31:0] @0x8, W B8 [7:0] @0x2	0x2 / B8
			0xC / BFBEBDBC	4: R BFBEBDBC [31:0] @0xC, W BC [7:0] @0x3	0x3 / BC
32	16	4	0x0 / B3B2B1B0	1: R B3B2B1B0 [31:0] @0x0, W B1B0 [15:0] @0x0	0x0 / B1B0
			0x4 / B7B6B5B4	2: R B7B6B5B4 [31:0] @0x4, W B5B4 [15:0] @0x2	0x2 / B5B4
			0x8 / BBBAB9B8	3: R BBBAB9B8 [31:0] @0x8, W B9B8 [15:0] @0x4	0x4 / B9B8
			0xC / BFBEBDBC	4: R BFBEBDBC [31:0] @0xC, W BDBC [15:0] @0x6	0x6 / BDBC
32	32	4	0x0 / B3B2B1B0	1: R B3B2B1B0 [31:0] @0x0, W B3B2B1B0 [31:0] @0x0	0x0 / B3B2B1B0
			0x4 / B7B6B5B4	2: R B7B6B5B4 [31:0] @0x4, W B7B6B5B4 [31:0] @0x4	0x4 / B7B6B5B4
			0x8 / BBBAB9B8	3: R BBBAB9B8 [31:0] @0x8, W BBBAB9B8 [31:0] @0x8	0x8 / BBBAB9B8
			0xC / BFBEBDBC	4: R BFBEBDBC [31:0] @0xC, W BFBEBDBC [31:0] @0xC	0xC / BFBEBDBC

Notice:

DMA always provide full 32-bits data to HWDATA[31:0] no matter what destination size it is (HSIZE still follows destination size setting for device supports byte/half-word operation). The HWDATA[31:0] it provides follow rules as follow:

- When source size is smaller than destination size, DMA pads the MSB with 0 until their sizes match and duplicates it to be 32 bits. E.g., source is 8 bits data 0x55 and destination size is 16 bits. DMA pads the source

data with 0 to make it 16 bits and become 0x0055, then duplicate it to 32-bit data 0x0055\_0055 and provide to HWDATA[31:0]; (if destination size is 32-bit then DMA will only pad source data with 0).

- When source size is larger or equal to destination size and smaller than 32 bits, DMA duplicates source data to 32 bits data. E.g., source data is 8 bits data 0x1F, HWDATA[31:0] = 0x1F1F\_1F1F. if source data is 16 bits data 0x2345, then HWDATA[31:0] = 0x2345\_2345.

This guarantees peripherals that only support word operation won't generate bus error and the desired data can still move to the place we want with extra bits i.e. 0 padding. If user wants to configure an 8-bit register but is aligned to a 32-bit address boundary, the source size should be set to 8 bits and destination to 32 bits so extra bits will be padded with 0.

#### 7.4.5 Peripheral/Memory address incrementation

DMA\_CHCFGx.PINC and DMA\_CHCFGx.MINC respectively control whether the peripheral address and memory address are enabled in auto-increment mode. The software cannot write (can read) the address register during transfer.

- In auto-increment mode, the next address to be transferred is automatically increased according to the data bit width (1, 2 or 4) after each transfer. The address of the first transfer is stored in DMA\_PADDRx or DMA\_MADDRx register.
- In fixed mode, the address is always fixed to the initial address.

At the end of transfer (i.e. the transfer count changes to 0), different processes will be carried out according to whether the current work is under circular mode or not.

- In acyclic mode, DMA stops after the transfer is completed. To start a new DMA transfer, need to rewrite the transfer number in the DMA\_TXNUMx register with the DMA channel disabled.
- In circular mode, at the end of a transfer, the content of the DMA\_TXNUMx register will be automatically reloaded to its initial value, and the current internal peripheral or memory address register will also be reloaded to the initial base address set by the DMA\_PADDRx or DMA\_MADDRx register.

#### 7.4.6 Channel configuration procedure

The detail configuration flow is as below:

- Configure interrupt mask bits, 1: enable interrupts, 0 disable interrupts.
- Configure channel peripheral address and memory address and transfer direction.
- Configure channel priority, 0: lowest, 3: highest.
- Configure peripheral and memory address increment.
- Configure channel transfer block size.
- If necessary, configure circular mode.
- If it is memory to memory, configure MEM2MEM mode (Note: to configure DMA to work in M2M mode, user

needs to set corresponding channel select value to reserved value, e.g., 47).

8. Repeat step 1~8 on channel 1~8 and finally.
9. Enable corresponding channel.

If software is used to serve interrupt, software must enquire interrupt status register to check which interrupt occurred (software needs to write 1 to interrupt flag clear bit to clear the corresponding interrupt). Before enable channel, all interrupts corresponding to the channel should be cleared.

If the interrupt is transfer complete interrupt, software can configure the next transfer, or report to user this channel transformation is done.

*Note: DMA user privilege management only supports that the user of the DMA configuration register is the same user as the DMA-enabled user, otherwise it will cause DMA transfer errors to occur.*

#### 7.4.7 Flow control

Three major flow controls are supported:

- Memory to memory
- Memory to peripheral
- Peripheral to memory

Flow control is controlled by two register bits in each DMA channel configuration register. Flow control is used to control source/destination and direction of DMA channel.

Table 7-2 Flow control table

DMA_CHCFGx.MEM2MEM	DMA_CHCFGx.DIR	Source	Destination	Transfer
1	x	Memory	Memory	AHB read to AHB write, can do back2back transfer
0	1	Memory	AHB Peripheral	AHB read to AHB write, single transfer
			APB Peripheral	AHB read to APB write, single transfer
0	0	AHB Peripheral	Memory	AHB read to AHB write, single transfer
		APB Peripheral		APB read to AHB write, single transfer

## 7.4.8 Circular mode

The circular mode is used to process circular buffers and continuous data transmission (such as ADC scan mode). The DMA\_CHCFGx.CIRC is used to enable this function. When the cyclic mode is activated, if the number of data to be transferred becomes 0, it will automatically be restored to the initial value when configuring the channel, and the DMA operation will continue.

If the user wants to turn off the circular mode, the user needs to write 0 to DMA\_CHCFGx.CHEN to disable the DMA channel, and then write 0 to DMA\_CHCFGx.CIRC (when DMA\_CHCFGx.CHEN is 1, other bits in the DMA\_CHCFGx register cannot be rewritten).

## 7.4.9 Error management

DMA access to a reserved address area will cause DMA transmission errors. When an error occurs, the transfer error flag is set, and the hardware automatically clears the current DMA channel enable bit (DMA\_CHCFGx.CHEN), and the channel operation is stopped. If the transfer error interrupt enable bit is set in the DMA\_CHCFGx register, an interrupt will be generated.

## 7.4.10 Interrupt

- Transfer complete interrupt:

An interrupt is generated when channel data transfer is complete. Interrupt is a level signal. Each channel has its dedicated interrupt, interrupt mask control and interrupt status bit. interrupt status bit is cleared when interrupt flag clear bit is set.

- Half transfer interrupt:

An interrupt is generated when half of the channel data is transferred. Interrupt is a level signal. Each channel has its dedicated interrupt, interrupt mask control and interrupt status bit. interrupt status bit is cleared when interrupt flag clear bit is set.

- Transfer error interrupt:

An interrupt is generated when bus returned error. Interrupt is a level signal. Each channel has its dedicated interrupt, interrupt mask control and interrupt status bit. interrupt status bit is cleared when interrupt flag clear bit is set.

Table 7-3 DMA interrupt request

Interrupt event	Event flag bit	Enable control bit
Half transfer	HTXF	HTXIE
Transfer complete	TXCF	TXCIE
Transfer error	ERRF	ERRIE

#### 7.4.11 DMA request mapping

Totally there are 35 DMA requests from all the peripherals. To have better support with full flexibility, register bits can be used to select which DMA request is mapped to which DMA channel. The table below show the mapping scheme of peripherals' DMA request to DMA controller's DMA channels.

Table 7-4 DMA request mapping

DMA channel select	Peripheral DMA request	DMA channel select	Peripheral DMA request
sel = 0	adc_dma	sel = 24	Tim1_ch2
sel = 1	Usart1_tx	sel = 25	Tim1_ch3
sel = 2	Usart1_rx	sel = 26	Tim1_ch4
sel = 3	Usart2_tx	sel = 27	Tim1_com
sel = 4	Usart2_rx	sel = 28	Tim1_up
sel = 5	Lpuart_tx	sel = 29	Tim1_trig
sel = 6	Lpuart_rx	sel = 30	Tim8_ch1
sel = 7	Reserved	sel = 31	Tim8_ch2
sel = 8	Reserved	sel = 32	Tim8_ch3
sel = 9	Reserved	sel = 33	Tim8_ch4
sel = 10	Reserved	sel = 34	Tim8_com
sel = 11	Reserved	sel = 35	Tim8_up
sel = 12	Reserved	sel = 36	Tim8_trig
sel = 13	Spi1_tx	sel = 37	Tim3_ch1
sel = 14	Spi1_rx	sel = 38	Tim3_ch3
sel = 15	Spi2_tx	sel = 39	Tim3_ch4
sel = 16	Spi2_rx	sel = 40	Tim3_up
sel = 17	Reserved	sel = 41	Tim3_trig
sel = 18	Reserved	sel = 42	Reserved
sel = 19	I2c1_tx	sel = 43	Reserved
sel = 20	I2c1_rx	sel = 44	Reserved
sel = 21	I2c2_tx	sel = 45	Reserved
sel = 22	I2c2_rx	sel = 46	TIM6
sel = 23	Tim1_ch1		

## 7.5 DMA registers

### 7.5.1 DMA register overview

Table 7-5 DMA register overview

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0
004h	DMA_INTCLR	Reserved				CERRF5	0	19	CHTXF5	0	18	CTXCF5	0
	Reset Value					0	0	0	0	0	0	0	0
008h	DMA_CHCFG1	Reserved				CTXCF5	0	17	CTXCF5	0	16	CGLBF5	0
	Reset Value					0	0	0	0	0	0	0	0
00Ch	DMA_TXNUM1	Reserved				MEM2MEM	0	15	CHTXF4	0	14	CHTXF4	0
	Reset Value					0	0	0	0	0	0	0	0
010h	DMA_PADDR1	Reserved				PRIOLVL[1:0]	0	13	CTXCF4	0	12	CGLBF4	0
	Reset Value					0	0	0	0	0	0	0	0
014h	DMA_MADDR1	Reserved				MSIZE[1:0]	0	11	CERRF3	0	10	CHSEL[1:0]	0
	Reset Value					0	0	0	0	0	0	0	0
018h	DMA_CHSEL1	Reserved				PSIZE[1:0]	0	9	CHSEL[5:0]	0	8	CHSEL[5:0]	0
	Reset Value					0	0	0	0	0	0	0	0
01Ch	DMA_CHCFG2	Reserved				MINC	0	7	CIRRC	0	6	CIRRC	0
	Reset Value					0	0	0	0	0	0	0	0
020h	DMA_TXNUM2	Reserved				DIR	0	5	DIR	0	4	CGLBF2	0
	Reset Value					0	0	0	0	0	0	0	0
024h	DMA_PADDR2	Reserved				ERRIE	0	3	ERRIE	0	2	HTXIE	0
	Reset Value					0	0	0	0	0	0	0	0
028h	DMA_MADDR2	Reserved				HTXIE	0	2	TXCIE	0	1	TXCIE	0
	Reset Value					0	0	0	0	0	0	0	0
02Ch	DMA_CHSEL2	Reserved				CHEN	0	0	CHEN	0	0	CHEN	0
	Reset Value					0	0	0	0	0	0	0	0
030h	DMA_CHCFG3	Reserved				PSIZE[1:0]	0	9	PSIZE[1:0]	0	8	PSIZE[1:0]	0
	Reset Value					0	0	0	0	0	0	0	0
034h	DMA_TXNUM3	Reserved				MINC	0	7	CIRRC	0	6	CIRRC	0
	Reset Value					0	0	0	0	0	0	0	0
038h	DMA_PADDR3	Reserved				DIR	0	5	DIR	0	4	DIR	0
	Reset Value					0	0	0	0	0	0	0	0
03Ch	DMA_MADDR3	Reserved				ERRIE	0	3	ERRIE	0	2	HTXIE	0
	Reset Value					0	0	0	0	0	0	0	0
040h	DMA_CHSEL3	Reserved				TXCIE	0	0	TXCIE	0	0	TXCIE	0
	Reset Value					0	0	0	0	0	0	0	0
044h	DMA_CHCFG4	Reserved				CHEN	0	0	CHEN	0	0	CHEN	0
	Reset Value					0	0	0	0	0	0	0	0
048h	DMA_TXNUM4	Reserved				PSIZE[1:0]	0	9	PSIZE[1:0]	0	8	PSIZE[1:0]	0
	Reset Value					0	0	0	0	0	0	0	0
04Ch	DMA_PADDR4	Reserved				MINC	0	7	CIRRC	0	6	CIRRC	0
	Reset Value					0	0	0	0	0	0	0	0
050h	DMA_MADDR4	Reserved				DIR	0	5	DIR	0	4	DIR	0
	Reset Value					0	0	0	0	0	0	0	0
054h	DMA_CHSEL4	Reserved				ERRIE	0	3	ERRIE	0	2	HTXIE	0
	Reset Value					0	0	0	0	0	0	0	0
058h	DMA_CHCFG5	Reserved				TXCIE	0	0	TXCIE	0	0	TXCIE	0
	Reset Value					0	0	0	0	0	0	0	0
05Ch	DMA_TXNUM5	Reserved				CHEN	0	0	CHEN	0	0	CHEN	0
	Reset Value					0	0	0	0	0	0	0	0
060h	DMA_PADDR5	Reserved				PSIZE[1:0]	0	9	PSIZE[1:0]	0	8	PSIZE[1:0]	0
	Reset Value					0	0	0	0	0	0	0	0
064h	DMA_MADDR5	Reserved				MINC	0	7	CIRRC	0	6	CIRRC	0
	Reset Value					0	0	0	0	0	0	0	0
068h	DMA_CHSEL5	Reserved				DIR	0	5	DIR	0	4	DIR	0
	Reset Value					0	0	0	0	0	0	0	0

## 7.5.2 DMA interrupt status register (DMA\_INTSTS)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r 15	r 14	r 13	r 12	r 11	r 10	r 9	r 8	r 7	r 6	r 5	r 4	r 3	r 2	r 1	r 0
ERRF4	HTXF4	TXCF4	GLBF4	ERRF3	HTXF3	TXCF3	GLBF3	ERRF2	HTXF2	TXCF2	GLBF2	ERRF1	HTXF1	TXCF1	GLBF1
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit field	Name	Description
19/15/11/7/3	ERRFx	Transfer error flag for channel x (x=1...5). Hardware sets this bit when transfer error happen. This bit is cleared by software by writing ‘1’ to DMA_INTCLR.CERRFx bit. 0: Transfer error no happened on channel x. 1: Transfer error happened on channel x.
18/14/10/6/2	HTXFx	Half transfer flag for channel x (x=1...5). Hardware sets this bit when half transfer is done. This bit is cleared by software by writing ‘1’ to DMA_INTCLR.CHTXFx bit. 0: Half transfer not yet done on channel x. 1: Half transfer was done on channel x.
17/13/9/5/1	TXCFx	Transfer complete flag for channel x (x=1...5). Hardware sets this bit when transfer is done. This bit is cleared by software by writing ‘1’ to DMA_INTCLR.CTXCFx bit. 0: Transfer not yet done on channel x. 1: Transfer was done on channel x.
16/12/8/4/0	GLBFx	Global flag for channel x (x=1...5). Hardware sets this bit when any interrupt events happen in this channel. This bit is cleared by software by writing ‘1’ to DMA_INTCLR.CGLBFx bit. 0: No transfer error, half transfer or transfer done event happen on channel x. 1: One of transfer error, half transfer or transfer done event happen on channel x.

### 7.5.3 DMA interrupt flag clear register (DMA\_INTCLR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
RW 15	RW 14	RW 13	RW 12	RW 11	RW 10	RW 9	RW 8	RW 7	RW 6	RW 5	RW 4	RW 3	RW 2	RW 1	RW 0
CERRF4	CHTF4	CTXCF4	CGLBF4	CERRF3	CHTF3	CTXCF3	CGLBF3	CERRF2	CHTF2	CTXCF2	CGLBF2	CERRF1	CHTF1	CTXCF1	CGLBF1
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

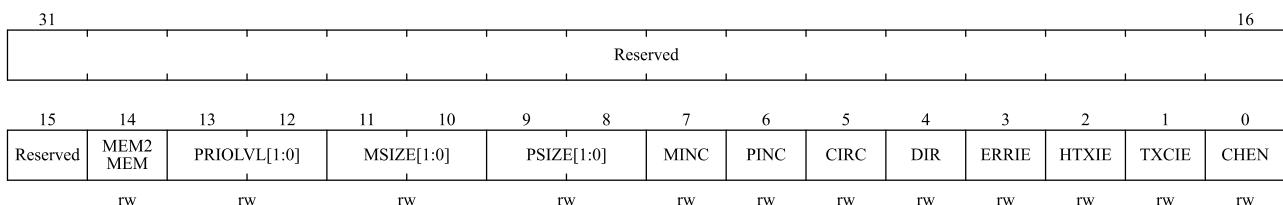
Bit field	Name	Description
19/15/11/7/3	CERRFx	Clear transfer error flag for channel x (x=1...5). Software can set this bit to clear ERRF of corresponding channel. 0: No action. 1: Reset DMA_INTSTS.ERRF bit of corresponding channel.
18/14/10/6/2	CHTDXFx	Clear half transfer flag for channel x (x=1...5). Software can set this bit to clear HTXF of corresponding channel. 0: No action. 1: Reset DMA_INTSTS.HTXF bit of corresponding channel.
17/13/9/5/1	CTXCFx	Clear transfer complete flag for channel x (x=1...5). Software can set this bit to clear TXCF of corresponding channel. 0: No action. 1: Reset DMA_INTSTS.TXCF bit of corresponding channel.
16/12/8/4/0	CGLBFx	Clear global event flag for channel x (x=1...5). Software can set this bit to clear GLBF of corresponding channel. 0: No action. 1: Reset DMA_INTSTS.GLBF bit of corresponding channel.

### 7.5.4 DMA channel x configuration register (DMA\_CHCFGx)

Note: The x is channel number, x = 1...5

Address offset: 0x08+20 \* (x-1)

Reset value: 0x0000 0000



Bit field	Name	Description
31:15	Reserved	Reserved, the reset value must be maintained.
14	MEM2MEM	Memory to memory mode. Software can configure this channel to memory to memory transfer when it is not yet enabled. 0: Channel transfer between memory and peripheral. 1: Channel set to memory to memory transfer.
13:12	PRIOLVL[1:0]	Channel priority. Software can program channel priority when channel is not enable. 00: Low 01: Medium 10: High

Bit field	Name	Description
		11: Very high
11:10	MSIZE[1:0]	<p>Memory data size.</p> <p>Software can configure data size read/write from/to memory address.</p> <p>00: 8-bits 01: 16-bits 10: 32-bits 11: Reserved</p>
9:8	PSIZE[1:0]	<p>Peripheral data size.</p> <p>Software can configure data size read/write from/to peripheral address.</p> <p>00: 8-bits 01: 16-bits 10: 32-bits 11: Reserved</p>
7	MINC	<p>Memory increment mode.</p> <p>Software can enable/disable memory address increment mode.</p> <p>0: Memory address won't increase with each transfer. 1: Memory address increase with each transfer.</p>
6	PINC	<p>Peripheral increment mode.</p> <p>Software can enable/disable peripheral address increment mode.</p> <p>0: Peripheral address won't increase with each transfer. 1: Peripheral address increase with each transfer.</p>
5	CIRC	<p>Circular mode.</p> <p>Software can set/clear this bit.</p> <p>0: Channel will stop after one round of transfer. 1: Channel configure as circular mode.</p>
4	DIR	<p>Data transfer direction</p> <p>Software can set/clear this bit.</p> <p>0: Data transfer from Peripheral to Memory 1: Data transfer from Memory to Peripheral.</p>
3	ERRIE	<p>Transfer error interrupt enable.</p> <p>Software can enable/disable transfer error interrupt.</p> <p>0: Disable transfer error interrupt of channel x. 1: Enable transfer error interrupt of channel x.</p>
2	HTXIE	<p>Half transfer interrupt enable.</p> <p>Software can enable/disable half transfer interrupt.</p> <p>0: Disable half transfer interrupt of channel x. 1: Enable half transfer interrupt of channel x.</p>
1	TXCIE	<p>Transfer complete interrupt enable.</p> <p>Software can enable/disable transfer complete interrupt.</p> <p>0: Disable transfer complete interrupt of channel x. 1: Enable transfer complete interrupt of channel x.</p>

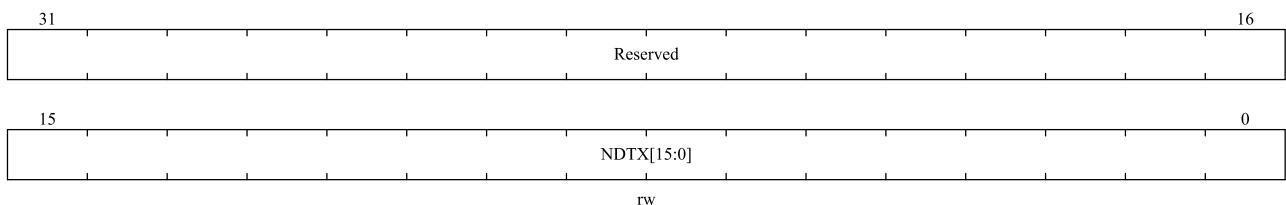
Bit field	Name	Description
0	CHEN	Channel enable. Software can set/reset this bit. 0: Disable channel. 1: Enable channel.

### 7.5.5 DMA channel x transfer number register (DMA\_TXNUMx)

Note: The x is channel number, x = 1...5

Address offset: 0x0C+20 \* (x-1)

Reset value: 0x0000 0000



Bit field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained.
15:0	NDTX	Number of data to transfer. Number of data to be transferred (0~65535). Software can read/write the number of transfers when channel is disable and it will be read only after channel enable. Every successful transfer of corresponding DMA channel will decrease this register by 1. If circular mode is enable, it will automatically reload pre-set value when it reach zero. Otherwise it will keep at zero and reset channel enable.

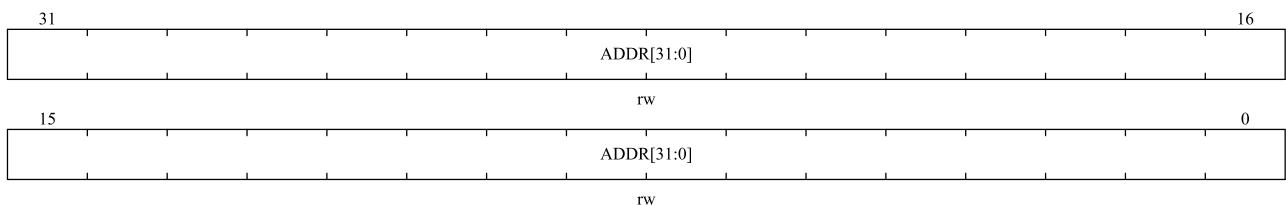
### 7.5.6 DMA channel x peripheral address register (DMA\_PADDRx)

Note: The x is channel number, x = 1...5

Address offset: 0x10+20 \* (x-1)

Reset value: 0x0000 0000

This register can only be written if the channel is disabled (DMA\_CHCFGx.CHEN = 0).



Bit field	Name	Description
31:0	ADDR	Peripheral address. Peripheral starting address for DMA to read/write from/to. Increment of address will be decided by DMA_CHCFGx.PSIZE. With DMA_CHCFGx.PSIZE equal to 01, DMA ignores bit 0 of PADDR and if DMA_CHCFGx.PSIZE equal to 10 DMA will ignore bit [1:0] of PADDR.

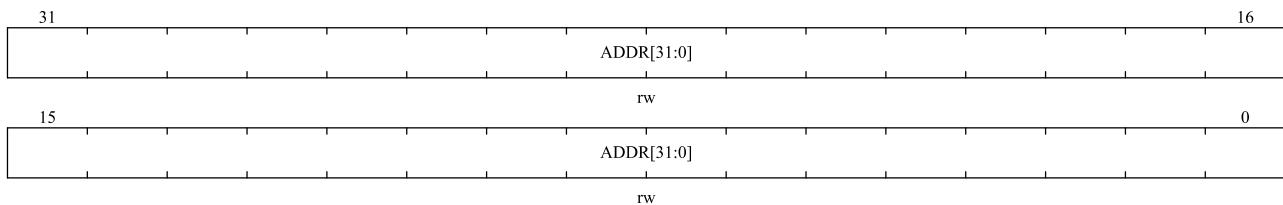
### 7.5.7 DMA channel x memory address register (DMA\_MADDRx)

Note: The x is channel number, x = 1...5

Address offset: 0x14+20 \* (x-1)

Reset value: 0x0000 0000

This register can only be written if the channel is disabled (DMA\_CHCFGx.CHEN = 0).



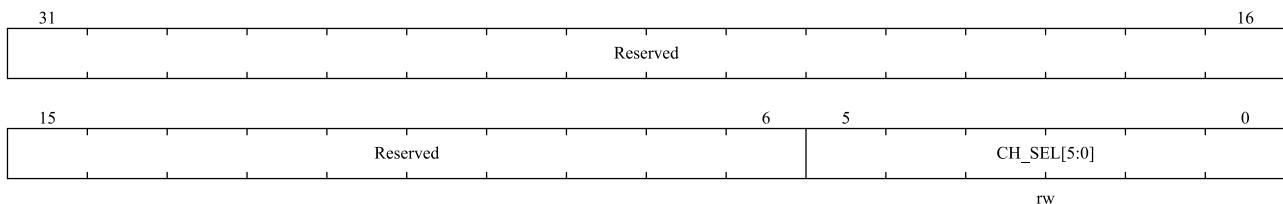
Bit field	Name	Description
31:0	ADDR	ADDR Memory address. Memory starting address for DMA to read/write from/to. Increment of address will be decided by DMA_CHCFGx.MSIZE. With DMA_CHCFGx.MSIZE equal to 01, DMA ignores bit 0 of MADDR and if DMA_CHCFGx.MSIZE equal to 10 DMA will ignore bit [1:0] of MADDR.

### 7.5.8 DMA channel x channel request select register (DMA\_CHSELx)

Note: The x is channel number, x = 1...5

Address offset: 0x18+20 \* (x-1)

Reset value: 0x0000 0000



Bit field	Name	Description
31:6	Reserved	Reserved, the reset value must be maintained.

Bit field	Name	Description
5:0	CH_SEL[5:0]	DMA channel request selection 0x00: adc_dma ..... 0x2E: TIM6 For the mapping of peripheral DMA requests to DMA input request channel numbers, please refer to Table 7-4

## 8 CRC calculation unit

### 8.1 CRC introduction

This module integrates the functions of CRC32 and CRC16, and the cyclic redundancy check (CRC) calculation unit obtains any CRC calculation result according to a fixed generator polynomial. In other applications, CRC technology is mainly used to verify the correctness and integrity of data transmission or data storage. EN/IEC 60335-1 provides a method to verify the integrity of flash memory. CRC calculation unit can calculate the identifier of the software when the program is running, then compare it with the reference identifier generated during connection, and then store it in the specified memory space.

### 8.2 CRC main features

#### 8.2.1 CRC32 module

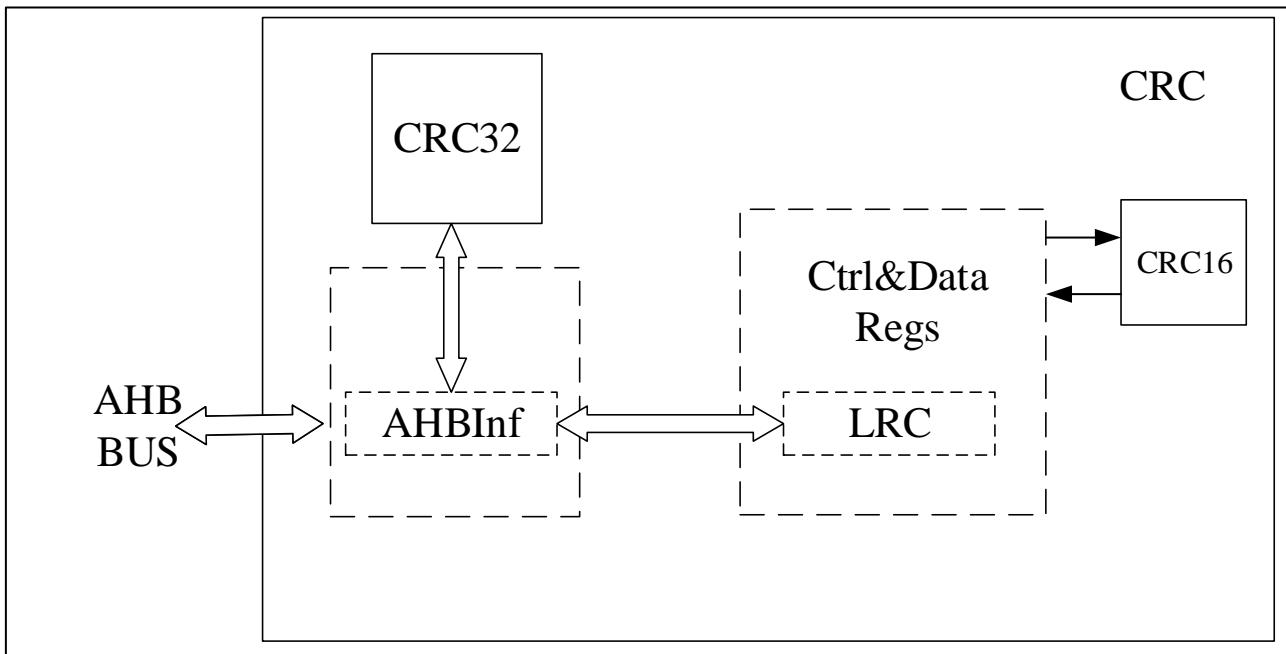
- CRC32( $X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^8+X^7+X^5+X^4+X^2+X+1$ )
- 32 bits of data to be checked and 32 bits of output check code.
- CRC calculation time: 1 AHB clock cycles (HCLK)
- General-purpose 8-bit register (can be used to store temporary data)

#### 8.2.2 CRC16 module

- CRC16( $X^{16}+X^{15}+X^2+1$ )
- There are 8 bits of data to be checked and 16 bits of output check code.
- CRC calculation time: 1 AHB clock cycle (HCLK)
- The verification initial value can be configured, and the size end of the data to be verified can be configured.
- Support 8bit LRC check value generation

The following figure is the block diagram of CRC unit.

Figure 8-1 CRC calculation unit block diagram



## 8.3 CRC function description

### 8.3.1 CRC32

CRC alculuation unit contains one 32-bit data register:

- Writing this register to input CRC data.
- Reading this register to get the calculated CRC result.

Every writing operation of this data register triggers the calculation of this new data with the previous calculation result (CRC calculation is performed on the whole 32-bit word rather than byte by byte).

Supports back-to-back writes or sequential write-read operations.

`CRC_CRC32DAT` can be re-initialized to `0xFFFFFFFF` by setting `CRC_CRC32CTRL.RESET`. This operation does not affect the data in register `CRC_CRC32IDAT`.

### 8.3.2 CRC16

`CRC_CRC16CTRL.ENDHDL` controls little endian or big endian.

To clear the result of the last CRC operation, set `CRC_CRC16CTRL.CLR` to 1 or `CRC_CRC16D` to 0.

The initial value of CRC calculation can be configured by writing the `CRC_CRC16D` register. By default, the initial value is the result of the last calculation.

LRC calculation is the same as CRC calculation. Both are carried out at the same time. CRC or LRC can be read out depending on needs. If the initial value needs to be set, the LRC register should be configured first.

## 8.4 CRC registers

### 8.4.1 CRC register overview

The following table lists the registers and reset values of CRC.

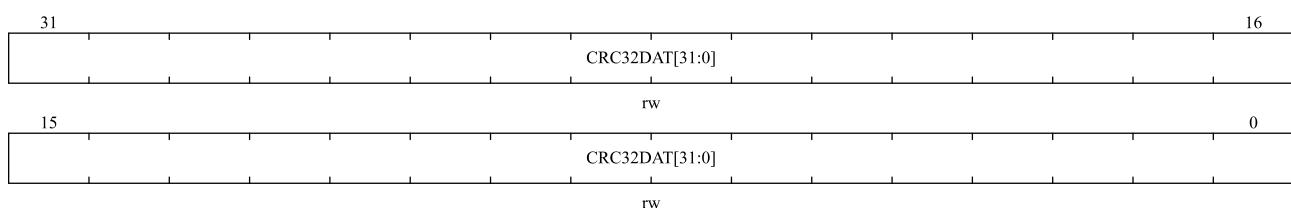
Table 8-1 CRC register overview

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	CRC32DAT																																
	Reset Value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		
004h	CRC32IDAT																																
	Reset Value																																
008h	CRC32CTRL																														RESET	0	
	Reset Value																																
00Ch	CRC16CTRL																												CLR	ENDHL	Reserved	0	
	Reset Value																																
010h	CRC16DAT																																
	Reset Value																																
014h	CRC16D																																
	Reset Value																																
018h	LRC																																
	Reset Value																																

### 8.4.2 CRC32 data register (CRC\_CRC32DAT)

Address offset: 0x00

Reset value: 0xFFFF FFFF

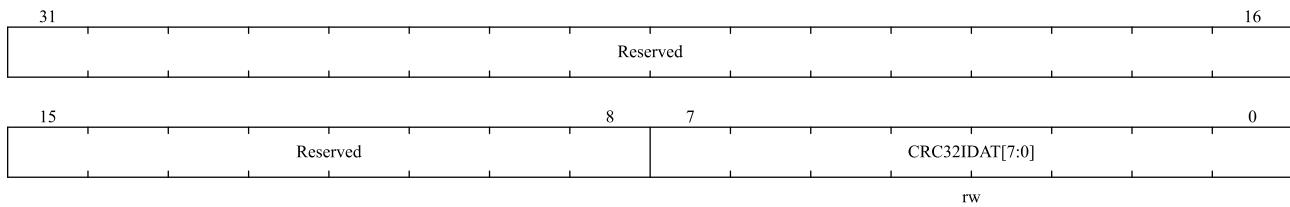


Bit field	Name	Description
31:0	CRC32DAT[31:0]	CRC32 Data register. The written data is the CRC value to be checked. The read data is the CRC calculation result. Only 32-bit operations are supported.

### 8.4.3 CRC32 independent data register (CRC\_CRC32IDAT)

Address offset: 0x04

Reset value: 0x0000 0000



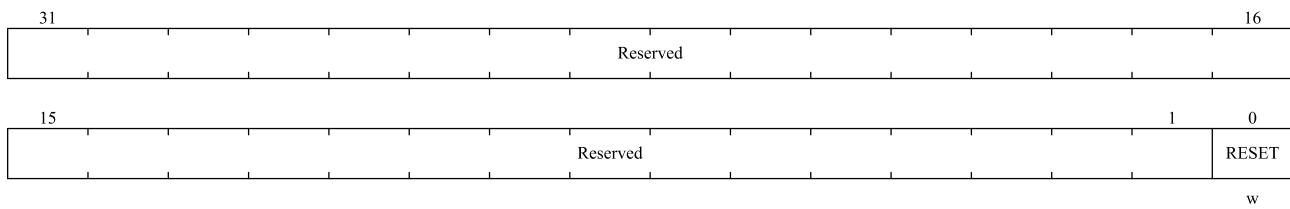
Bit field	Name	Description
31:8	Reserved	Reserved, the reset value must be maintained
7:0	CRC32IDAT[7:0]	Independent 8-bit data register. General 8 bits data register. It is for temporary stored 1-byte data. CRC_CRC32CTRL.RESET bit reset signal will not impact this register.

*Note: This register is not a part of CRC calculation and can be used to store any data.*

#### 8.4.4 CRC32 control register (CRC\_CRC32CTRL)

Address offset: 0x08

Reset value: 0x0000 0000

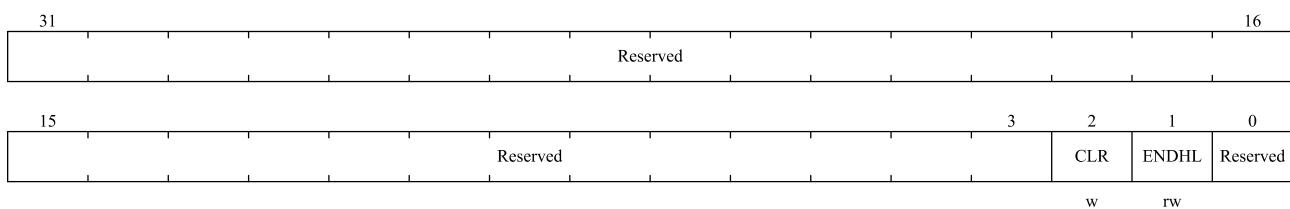


Bit field	Name	Description
31:1	Reserved	Reserved, the reset value must be maintained
0	RESET	RESET signal. It can reset CRC32 module and set data register to be 0xFFFF_FFFF. This reset can only write 1, and hardware will clear to 0 automatically.

#### 8.4.5 CRC16 control register (CRC\_CRC16CTRL)

Address offset: 0x0C

Reset value: 0x0000 0000



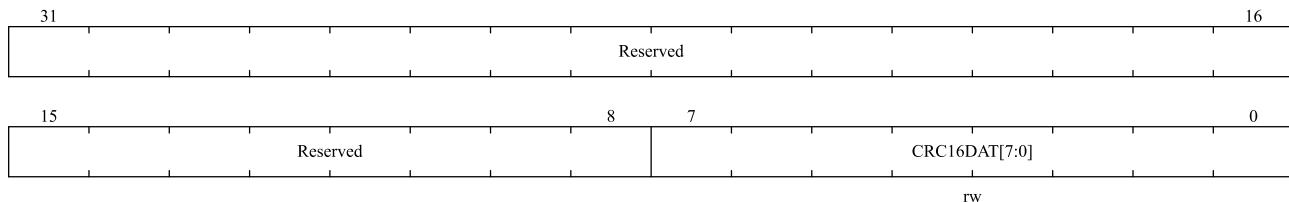
Bit field	Name	Description
31:3	Reserved	Reserved,the reset value must be maintained
2	CLR	Clear CRC16 results. 0: Not clear. 1: Clear to default value 0x0000. Set this bit to 1 will only maintain 1 clock cycle, hardware will clear automatically. (Software read always 0).
1	ENDHL	Data to be verified start to calculate from MSB or LSB. 0: From MSB to LSB 1: From LSB to MSB This bit is only for data to be verified.
0	Reserved	Reserved,the reset value must be maintained

*Note: 8-bits, 16-bits and 32-bits operations are supported.*

#### 8.4.6 CRC16 input data register (CRC\_CRC16DAT)

Address offset: 0x10

Reset value: 0x0000 0000



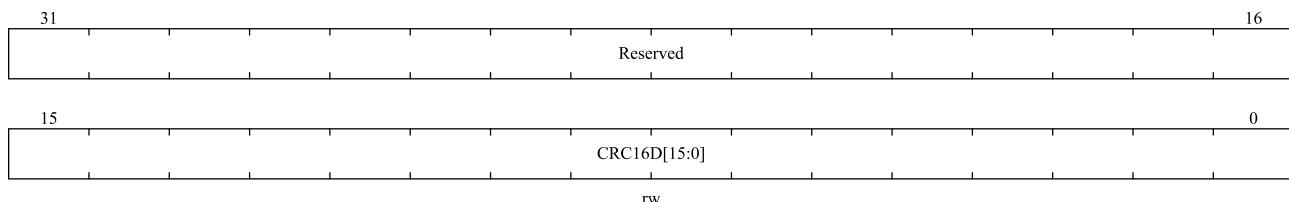
Bit field	Name	Description
31:8	Reserved	Reserved,the reset value must be maintained
7:0	CRC16DAT[7:0]	Data to be verified.

*Note: 8-bits, 16-bits and 32-bits operations are supported.*

#### 8.4.7 CRC cyclic redundancy check code register (CRC\_CRC16D)

Address offset: 0x14

Reset value: 0x0000 0000



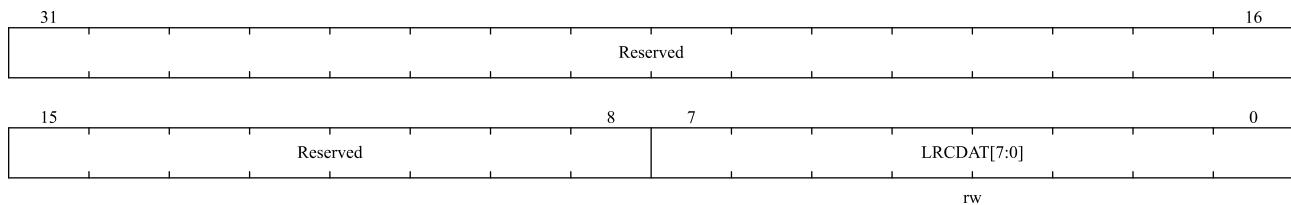
Bit field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained
15:0	CRC16D[15:0]	16-bit value of cyclic redundancy result data. Every time the software writes the CRC16DAT register, the 16-bit calculated data from CRC16 is updated in this register.

*Note: 8-bits, 16-bits and 32-bits operations are supported (8-bit operations must be performed twice in a row to ensure that 16-bit initial values are configured properly)*

## 8.4.8 LRC result register (CRC\_LRC)

Address offset: 0x18

Reset value: 0x0000 0000



Bit field	Name	Description
31:8	Reserved	Reserved, the reset value must be maintained
7:0	LRCDAT[7:0]	LRC check value register. Software need to write initial value before use. And then each writing data to CRCDR will be XOR with LCR register value. The result will be stored in LRC. Software read the result. It should be cleared before next use.

## 9 Advanced-control timers (TIM1 and TIM8)

### 9.1 TIM1 and TIM8 introduction

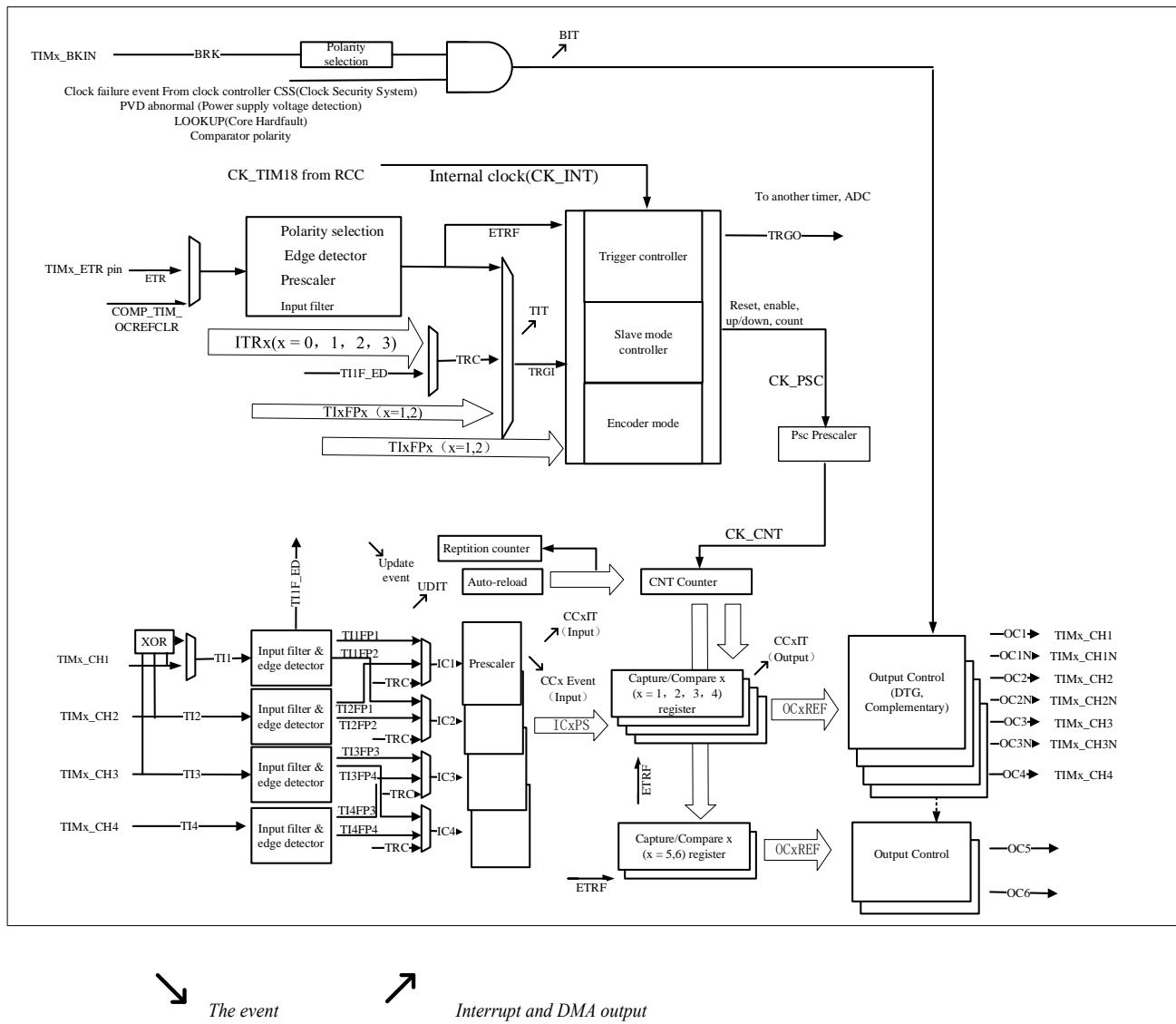
The advanced control timers (TIM1 and TIM8) is mainly used in the following occasions: counting the input signal, measuring the pulse width of the input signal and generating the output waveform, etc.

Advanced timers have complementary output function with dead-time insertion and break function. Suitable for motor control.

### 9.2 Main features of TIM1 and TIM8

- 16-bit auto-reload counters. (It can realize up-counting, down-counting, up/down counting)
- 16-bit programmable prescaler. (The frequency division factor can be configured with any value between 1 and 65536)
- Programmable Repetition Counter
- TIM1 up to 6 channels, TIM8 up to 6 channels.
- 4 capture/compare channels, the working modes are PWM output, output compare, one-pulse mode output, input capture.
- The events that generate the interrupt/DMA are as follows:
  - ◆ Update event
  - ◆ Trigger event
  - ◆ Input capture
  - ◆ Output compare
  - ◆ Break input
- Complementary outputs with adjustable dead-time
  - For TIM1 and TIM8, channel 1,2,3 support this feature
- Timer can be controlled by external signal
- Timers are linked internally for timer synchronization or chaining TIM1\_CC5 and TIM8\_CC5 for COMP blanking
- TIM1\_CC6 for OPAMP switch
- Incremental (quadrature) encoder interface: used for tracking motion and resolving rotation direction and position
- Hall sensor interface: used to do three-phase motor control

Figure 9-1 Block diagram of TIM1 and TIM8



## 9.3 TIM1 and TIM8 function description

### 9.3.1 Time-base unit

The advanced-control's time-base unit mainly includes: prescaler, counter, auto-reload and repetition counter. When the time base unit is working, the software can read and write the corresponding registers (TIMx\_PSC, TIMx\_CNT, TIMx\_AR and TIMx\_REPCNT) at any time.

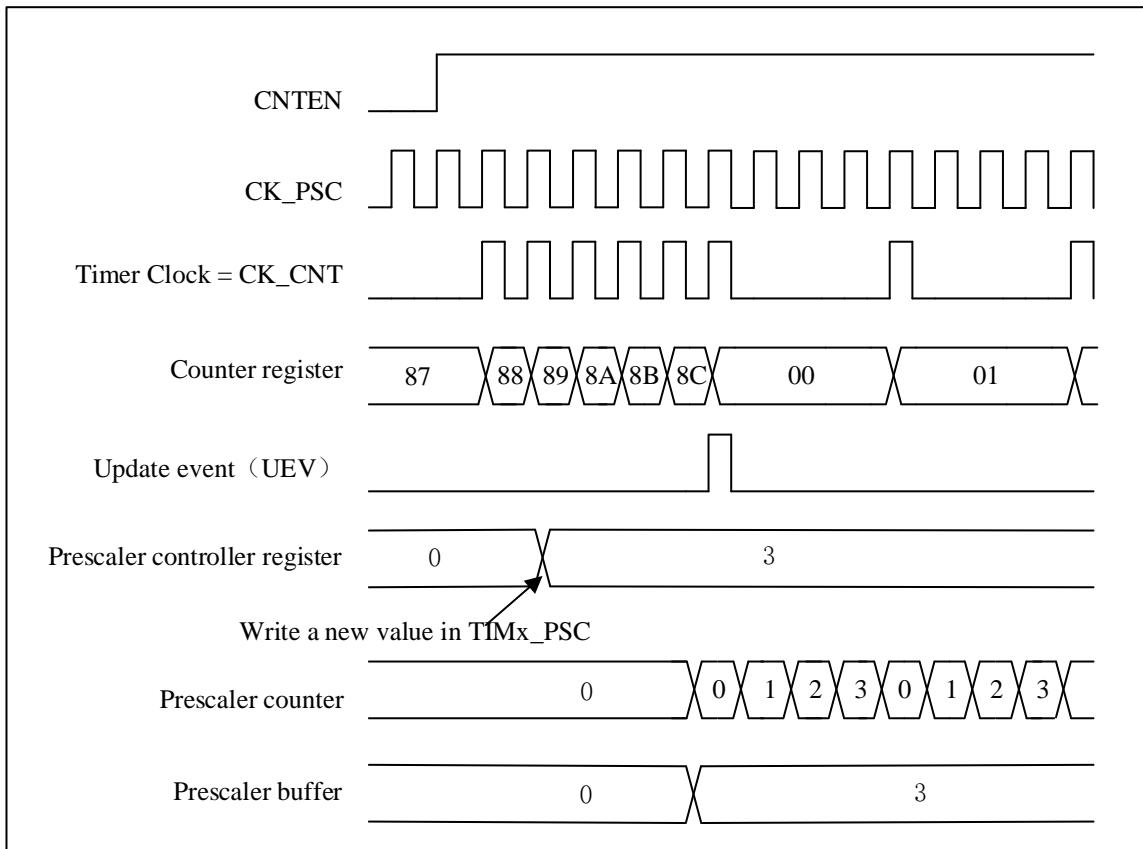
Depending on the setting of the auto-reload preload enable bit (TIMx\_CTRL1.ARREN), the value of the preload register is transferred to the shadow register immediately or at each update event UEV. An update event is generated when the counter reaches the overflow/underflow condition and it can be generated by software when TIMx\_CTRL1.UPDIS=0. The counter CK\_CNT is valid only when the TIMx\_CTRL1.CNTEN bit is set. The counter

starts counting one clock cycle after the TIMx\_CTRL1.CNTEN bit is set.

### 9.3.1.1 Prescaler description

The TIMx\_PSC register consists of a 16-bit counter that can be used to divide the counter clock frequency by any factor between 1 and 65536. It can be changed on the fly as it is buffered. The prescaler value is only taken into account at the next update event.

Figure 9-2 Counter timing diagram with prescaler division change from 1 to 4



## 9.3.2 Counter mode

### 9.3.2.1 Up-counting mode

In up-counting mode, the counter will count from 0 to the value of the register TIMx\_AR, then it resets to 0. And a counter overflow event is generated.

If the TIMx\_CTRL1.UPRS bit (select update request) and the TIMx\_EVTGEN.UDGN bit are set, an update event (UEV) will generate. And TIMx\_STS.UDITF will not be set by hardware, therefore, no update interrupts or update DMA requests are generated. This setting is used in scenarios where you want to clear the counter but do not want to generate an update interrupt.

Depending on the update request source is configured in the TIMx\_CTRL1.UPRS, When an update event occurs, all registers are updated and the TIMx\_STS.UDITF is set:

- The repetition counter reloads the contents of the TIMx\_REPCNT

- Update auto-reload shadow registers with preload value(TIMx\_AR), when TIMx\_CTRL1.ARPE = 1.
- The prescaler shadow register is reloaded with the preload value(TIMx\_PSC).

To avoid updating the shadow registers when new values are written to the preload registers, you can disable the update by setting TIMx\_CTRL1.UPDIS=1.

When an update event occurs, the counter will still be cleared and the prescaler counter will also be set to 0 (but the prescaler value will remain unchanged).

The figure below shows some examples of the counter behavior and the update flags for different division factors in the up-counting mode.

Figure 9-3 Timing diagram of up-counting. The internal clock divider factor = 2/N

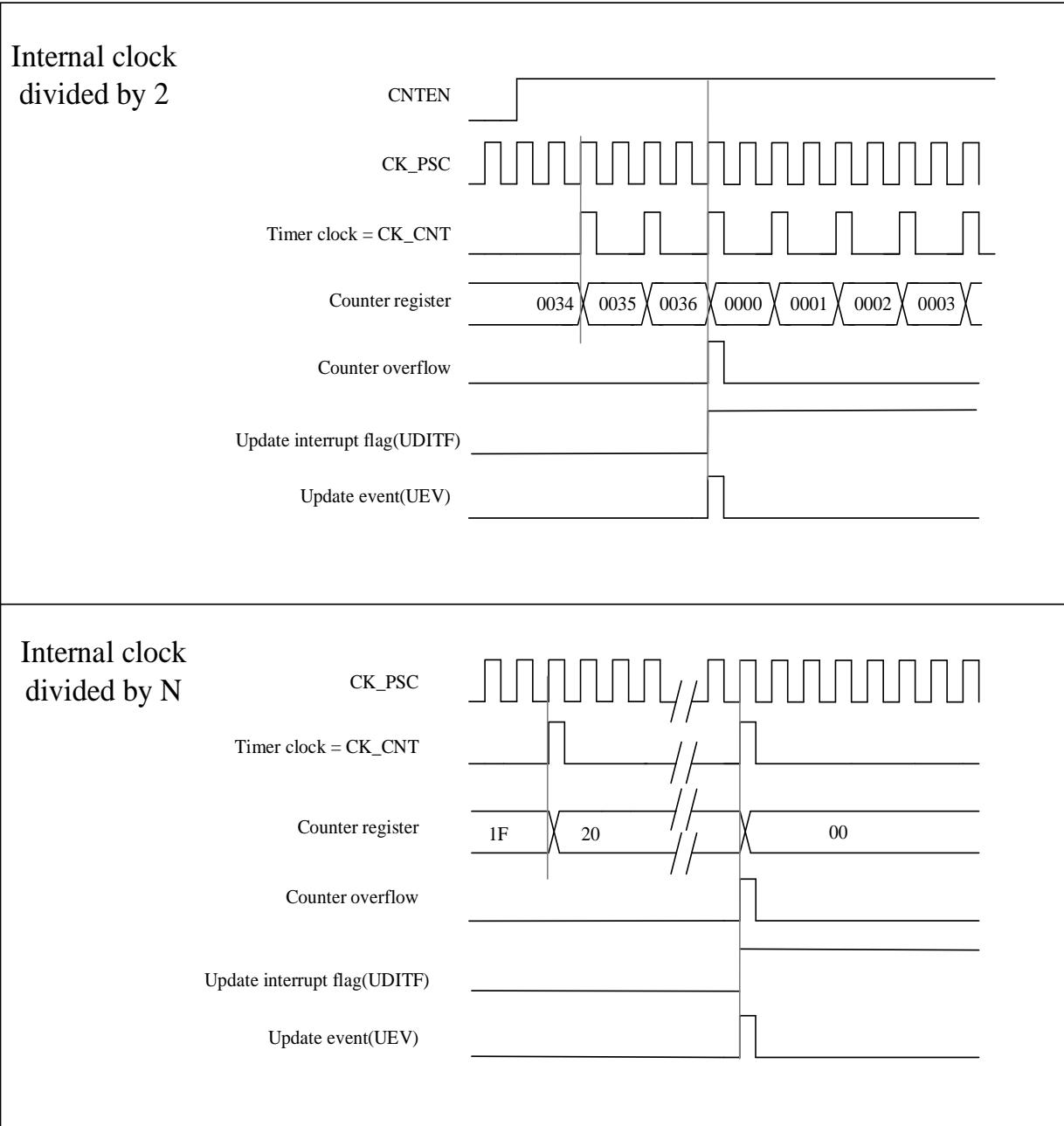
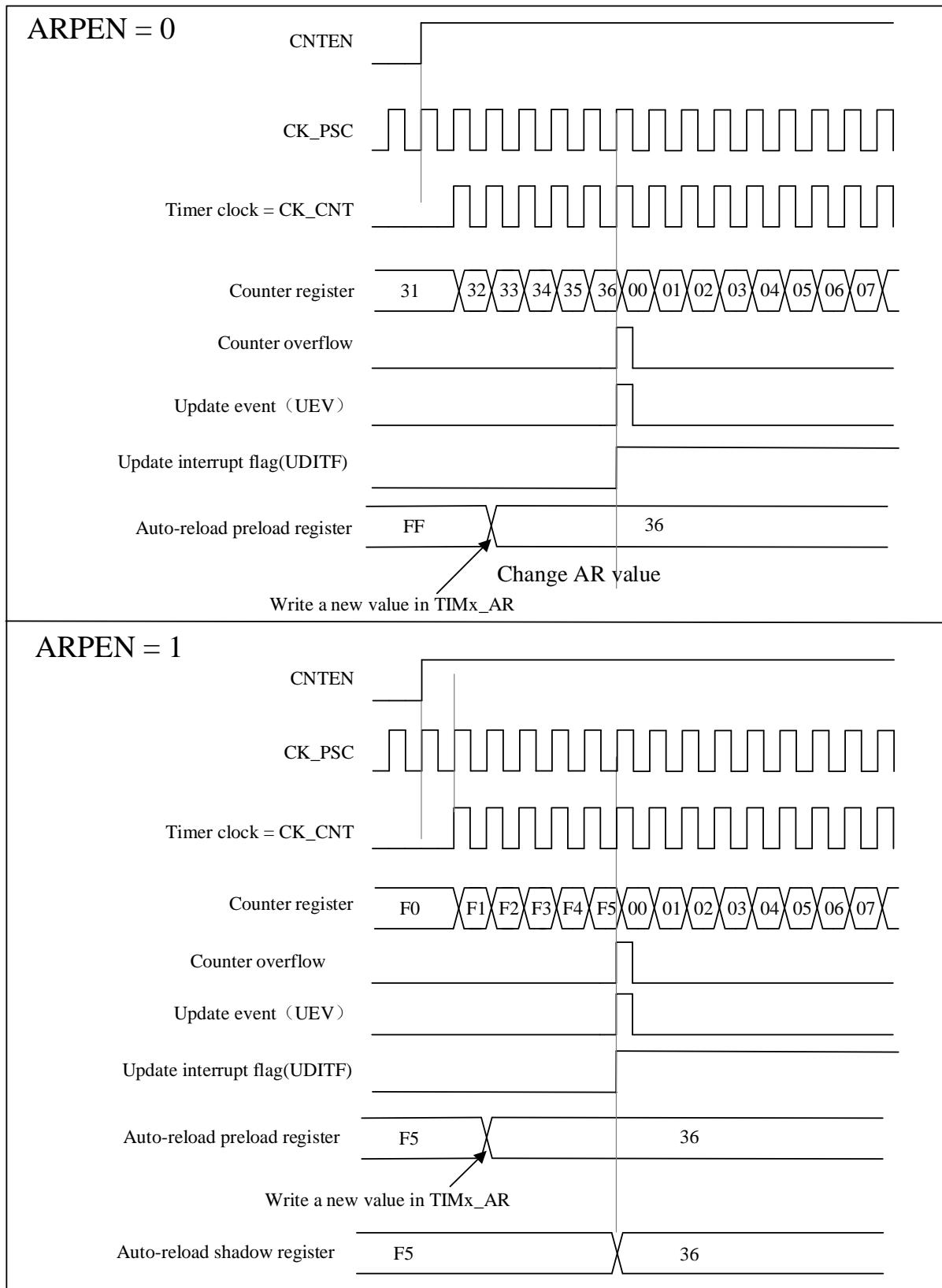


Figure 9-4 Timing diagram of the up-counting, update event when ARPEN=0/1



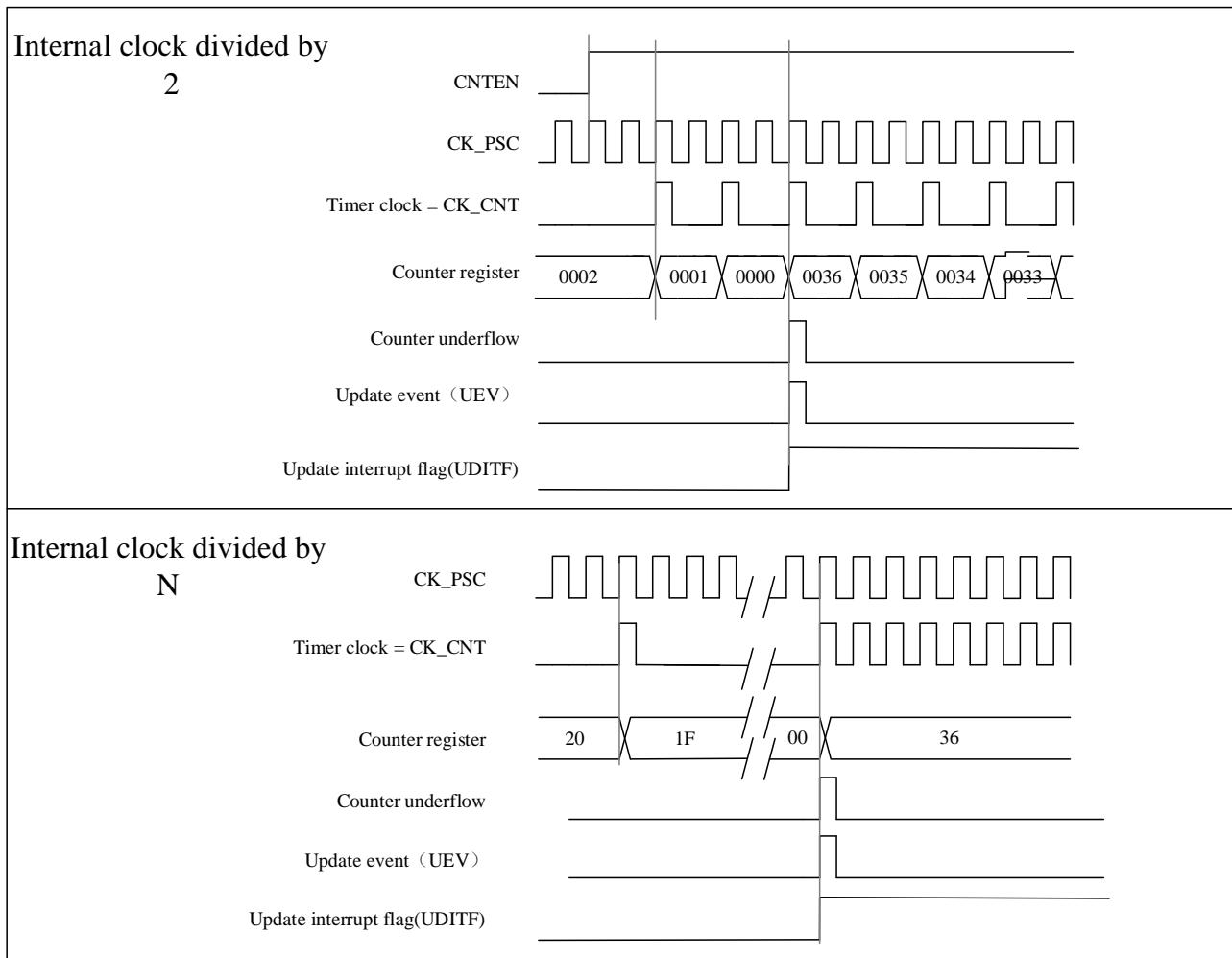
### 9.3.2.2 Down-counting mode

In down-counting mode, the counter will decrement from the value of the register `TIMx_AR` to 0, then restart from the auto-reload value and generate a counter underflow event.

The process of configuring update events and updating registers in down-counting mode is the same as in up-counting mode, see 9.3.2.1.

The figure below shows some examples of the counter behavior and the update flags for different division factors in the down-counting mode.

Figure 9-5 Timing diagram of the down-counting, internal clock divided factor = 2/N



### 9.3.2.3 Center-aligned mode

In center-aligned mode, the counter increments from 0 to the value (`TIMx_AR`) – 1, a counter overflow event is generated. It then counts down from the auto-reload value (`TIMx_AR`) to 1 and generates a counter underflow event. Then the counter resets to 0 and starts counting up again.

In this mode, the `TIMx_CTRL1.DIR` direction bits have no effect and the count direction is updated and specified by hardware. Center-aligned mode is valid when the `TIMx_CTRL1.CAMSEL` bit is not equal to "00".

The update events can be generated each time the counter overflows and each time the counter underflows. Alternatively, an update event can also be generated by setting the TIMx\_EVTGEN. UDGN bit (either by software or using a slave mode controller). In this case, the counter restarts from 0, as does the prescaler's counter.

Please note: if the update source is a counter overflow, auto-reload update before reloading the counter.

Figure 9-6 Timing diagram of the Center-aligned, internal clock divided factor =2/N

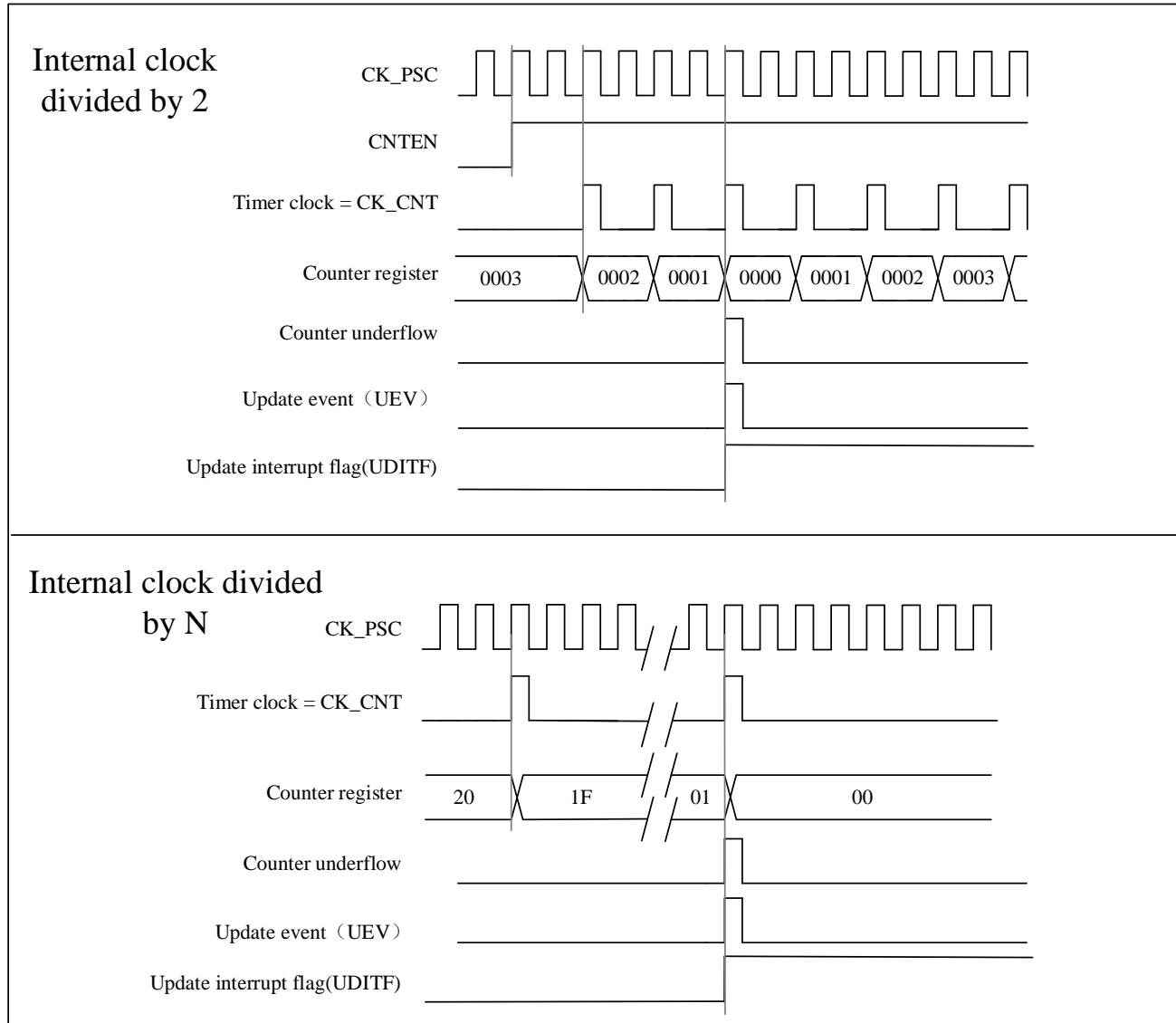
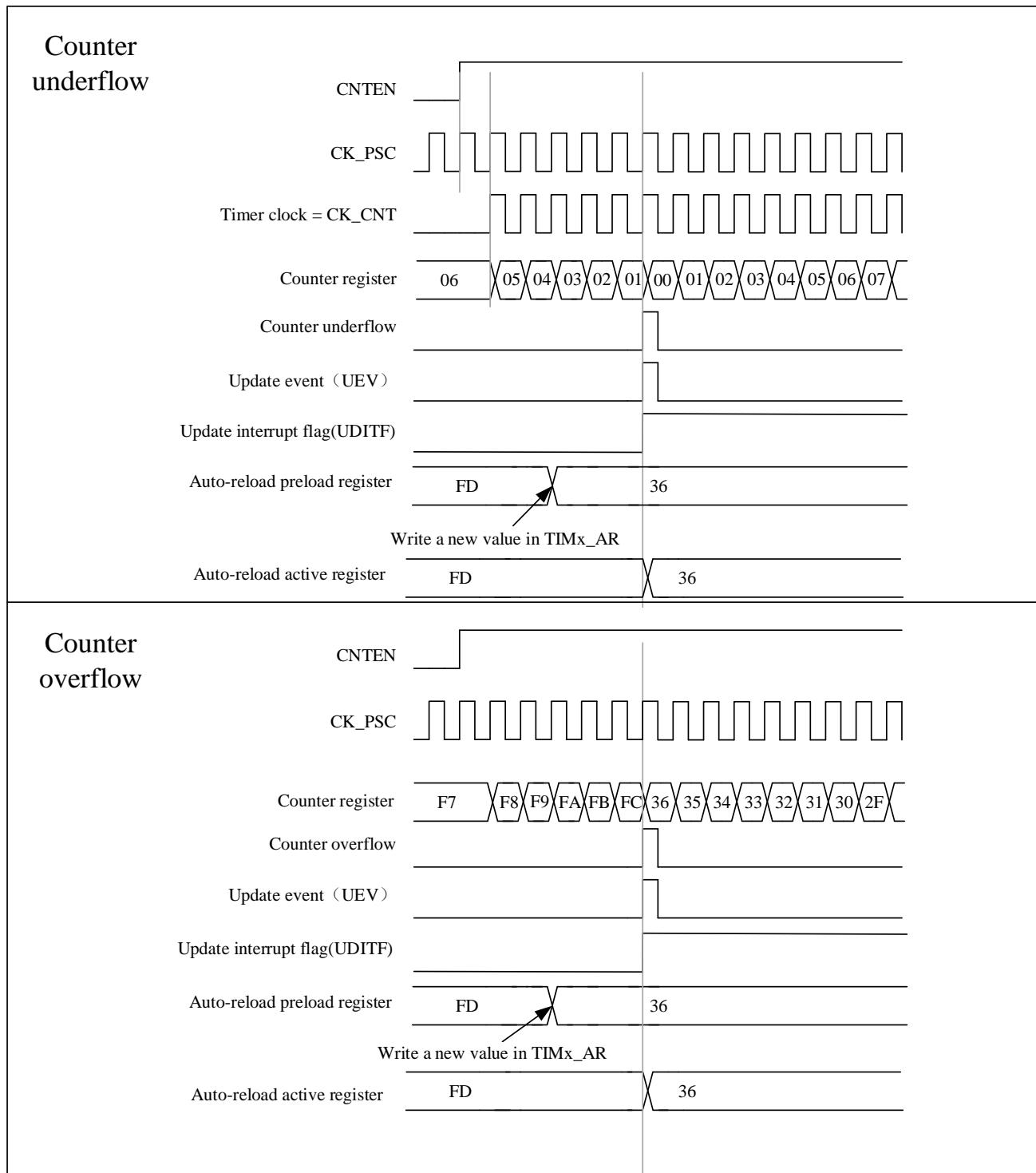


Figure 9-7 A center-aligned sequence diagram that includes counter overflows and underflows (ARPEN = 1)



### 9.3.3 Repetition counter

The basic unit of Section 9.3.1 describes the conditions for generating an update event (UEV). An update event (UEV) is actually only generated when the repeat counter reaches zero, which is valuable for generating PWM signals.

This means that data is transferred from the preload registers to the shadow registers every N+1 counter overflow or underflow, where N is the value in the TIMx\_REPCNT.

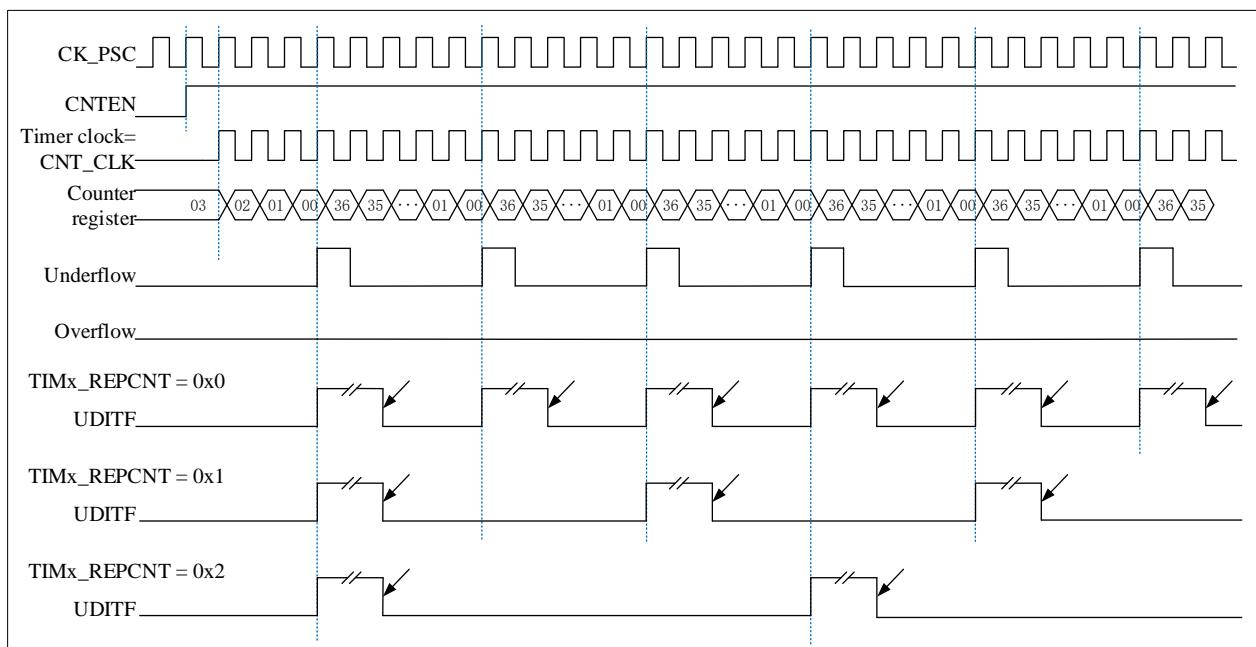
The repetition counter is decremented:

- In the up-counting mode, each time the counter reaches the maximum value, an overflow occurs.
- In down-counting mode, each time the counter decrements to the minimum value, an underflow occurs.
- In center-aligned mode, each time the counter overflows or underflows.

Its repetition rate is defined by the value of the TIMx\_REPCNT register.

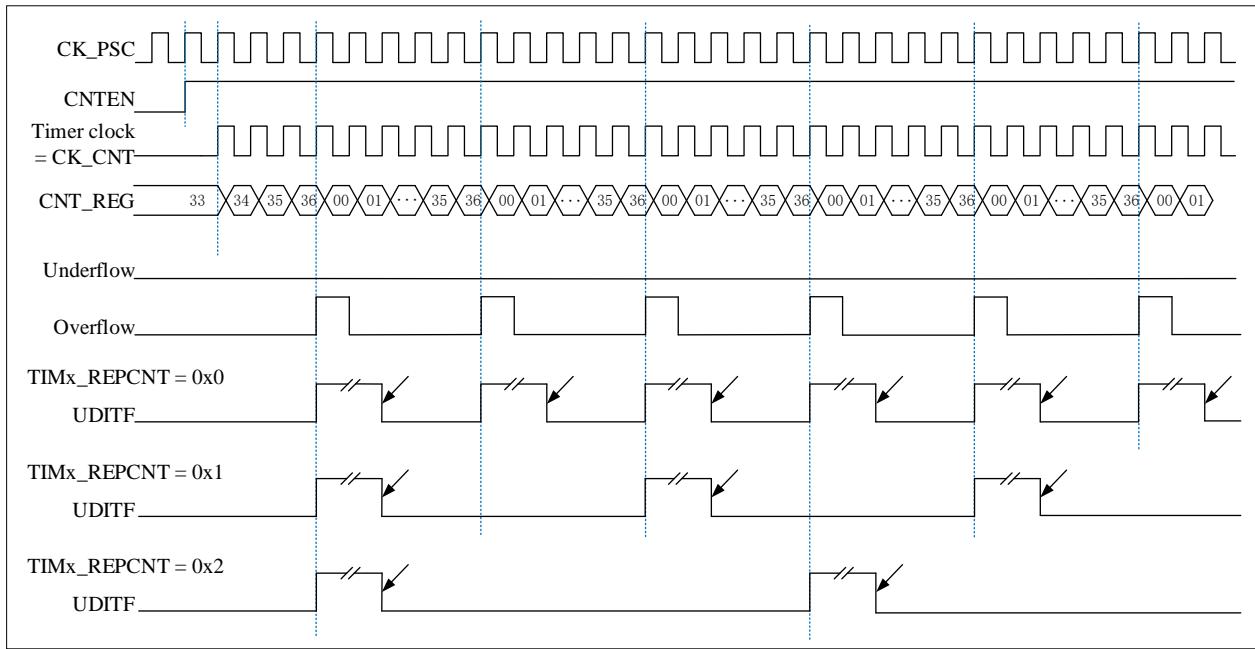
Repetition counters feature automatic reloading. The update event ( generated by setting TIMx\_EVTGEN.UDGN or hardware through slave mode controller) occurs immediately, regardless of the value of the repeat counter.

Figure 9-8 Repeat count sequence diagram in down-counting mode



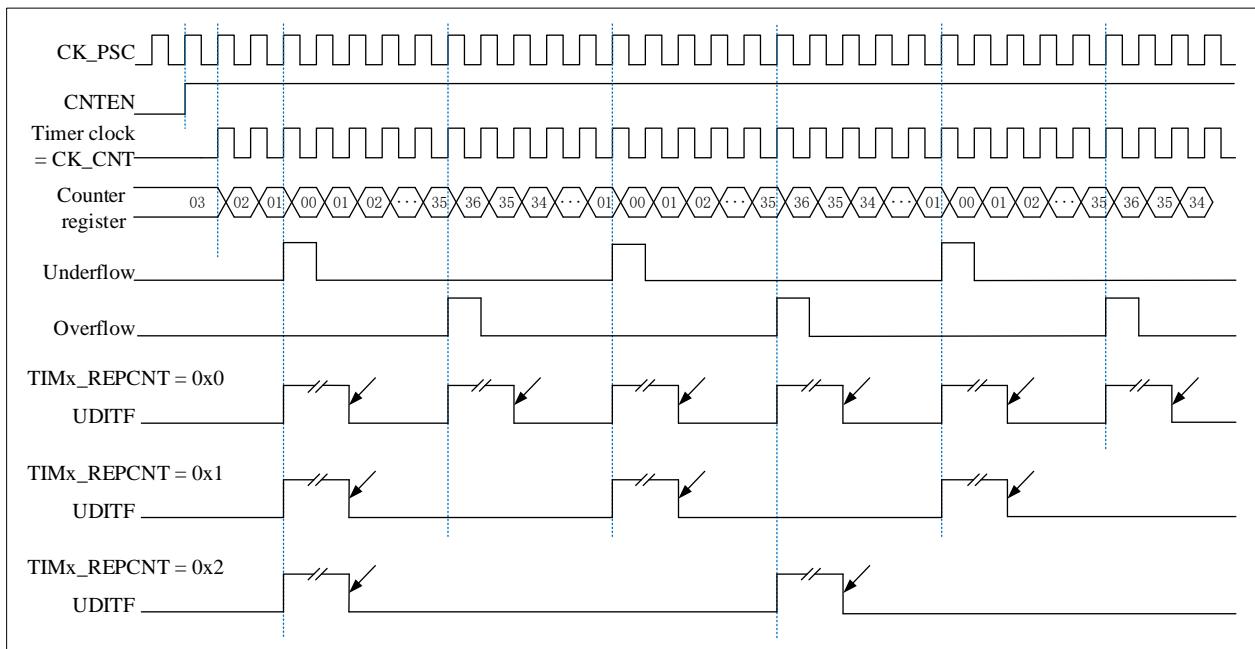
 Software clear

Figure 9-9 Repeat count sequence diagram in up-counting mode



*Software clear*

Figure 9-10 Repeat count sequence diagram in center-aligned mode



*Software clear*

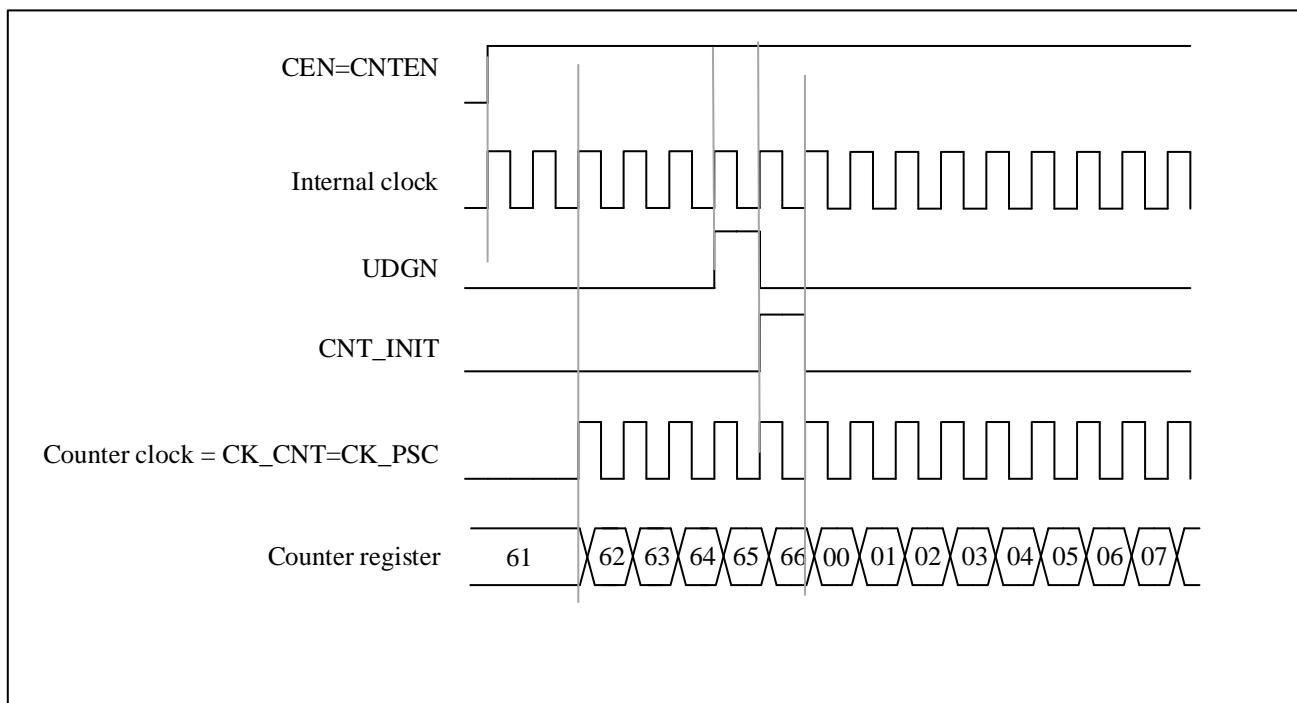
### 9.3.4 Clock selection

- The internal clock of Advanced-control timers : CK\_INT
- Two kinds of external clock mode :
  - external input pin
  - external trigger input ETR
- Internal trigger input (ITRx): one timer is used as a prescaler for another timer.

#### 9.3.4.1 Internal clock source (CK\_INT)

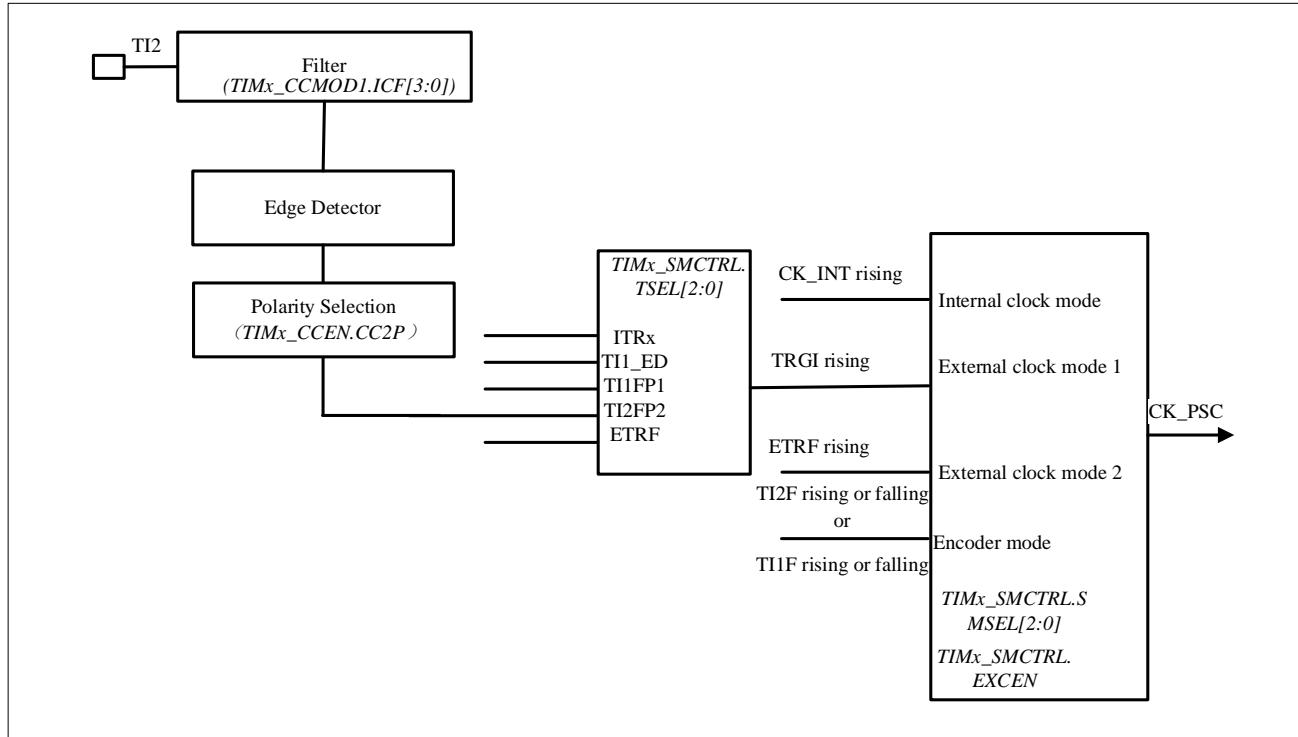
When the TIMx\_SMCTRL.SMSEL is equal to “000”, the slave mode controller is disabled. The three control bits (TIMx\_CTRL1.CNTEN □ TIMx\_CTRL1. DIR □ TIMx\_EVTGEN. UDGN) can only be changed by software (except TIMx\_EVTGEN. UDGN, which remains cleared automatically). It is provided that the TIMx\_CTRL1.CNTEN bit is written as '1' by soft, the clock source of the prescaler is provided by the internal clock CK\_INT.

Figure 9-11 Control circuit in normal mode, internal clock divided by 1



### 9.3.4.2 External clock source mode 1

Figure 9-12 TI2 external clock connection example



This mode is selected by configuring `TIMx_SMCTRL.SMSEL=111`. The counter can be configured to count on the rising or falling edge of the clock at the selected input.

For example, to configure up-counting mode to count on the rising edge of the clock at the TI2 input, the configuration steps are as follows:

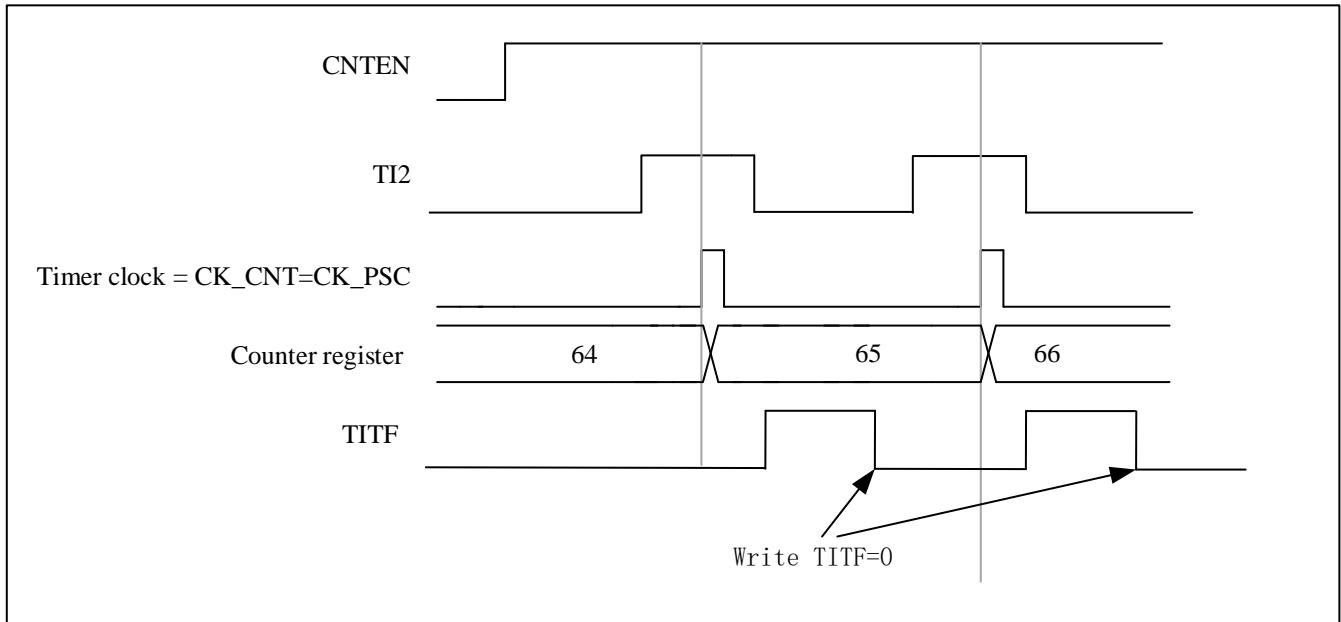
- Configure `TIMx_CCMOD1.CC2SEL` equal to ‘01’, CC2 channel is configured as input, IC2 is mapped to TI2
- Configure `TIMx_CCEN.CC2P` equal to ‘0’, select clock rising edge polarity
- To select input filter bandwidth by configuring `TIMx_CCMOD1.IC2F[3:0]` (if filter is not needed, keep IC2F bit at ‘0000’)
- Configure `TIMx_SMCTRL.SMSEL` equal to ‘111’, select timer external clock mode 1
- Configure `TIMx_SMCTRL.TSEL` equal to ‘110’, select TI2 as the trigger input source
- Configure `TIMx_CTRL1.CNTEN` equal to ‘1’ to start the counter

*Note: The capture prescaler is not used for triggering, so it does not need to be configured*

When the rising edge of the timer clock occurs at `TI2=1`, the counter counts once and the `TIMx_STS .TITF` flag is pulled high.

The delay between the rising edge of TI2 and the actual clock of the counter depends on the resynchronization circuit at the input of TI2.

Figure 9-13 Control circuit in external clock mode 1

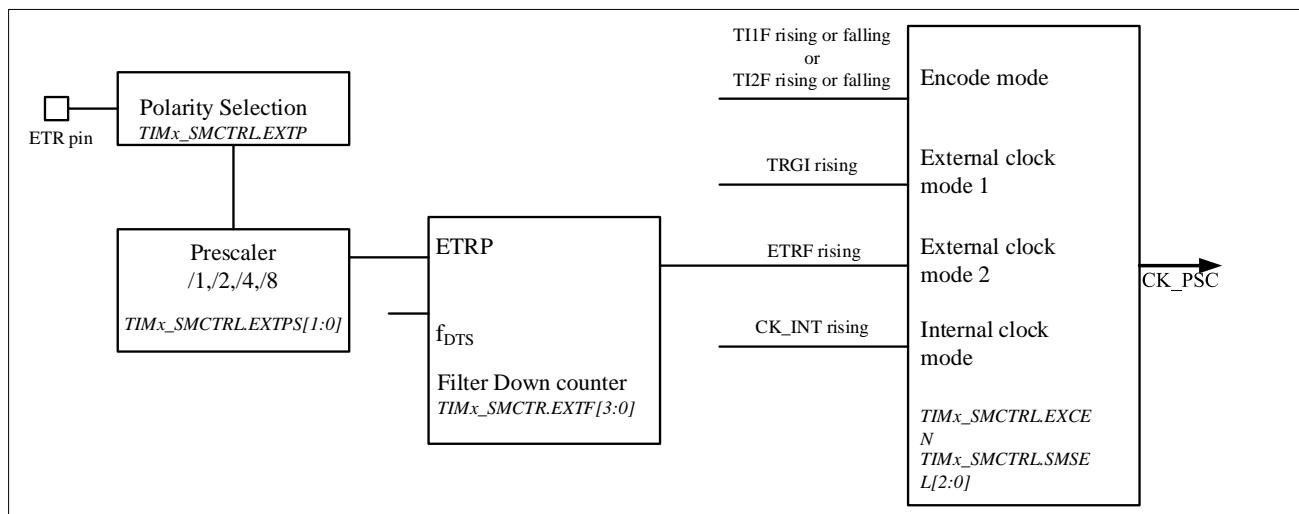


#### 9.3.4.3 External clock source mode 2

This mode is selected by `TIMx_SMCTRL.EXCEN` equal to 1. The counter can count on every rising or falling edge of the external trigger input ETR.

The following figure is a schematic diagram of the external trigger input module in External clock source mode 2

Figure 9-14 External trigger input block diagram



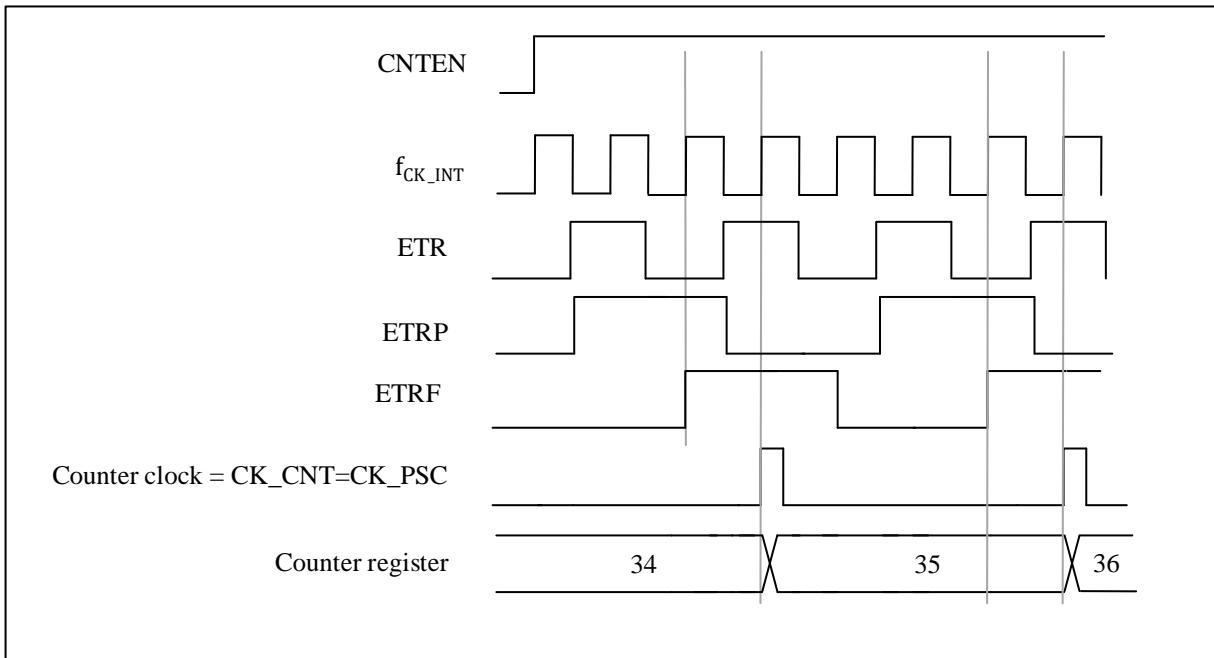
For example, use the following configuration steps to make the up counter count every 2 rising edges on ETR.

- Since no filter is needed in this case, make `TIMx_SMCTRL.EXTF[3:0]` equal to '0000'
- Configure the prescaler by making `TIMx_SMCTRL.EXTPS[1:0]` equal to '01'
- Select the polarity on ETR pin by setting `TIMx_SMCTRL.EXTP` equal to '0', The rising edge of ETR is valid

- External clock mode 2 is selected by setting TIMx\_SMCTRL .EXCEN equal to ‘1’
- Turn on the counter by setting TIMx\_CTRL1. CNTEN equal to ‘1’

The counter counts every 2 rising edges of ETR. The delay between the rising edge of ETR and the actual clock to the counter is due to a resynchronization circuit on the ETRP signal.

Figure 9-15 Control circuit in external clock mode 2

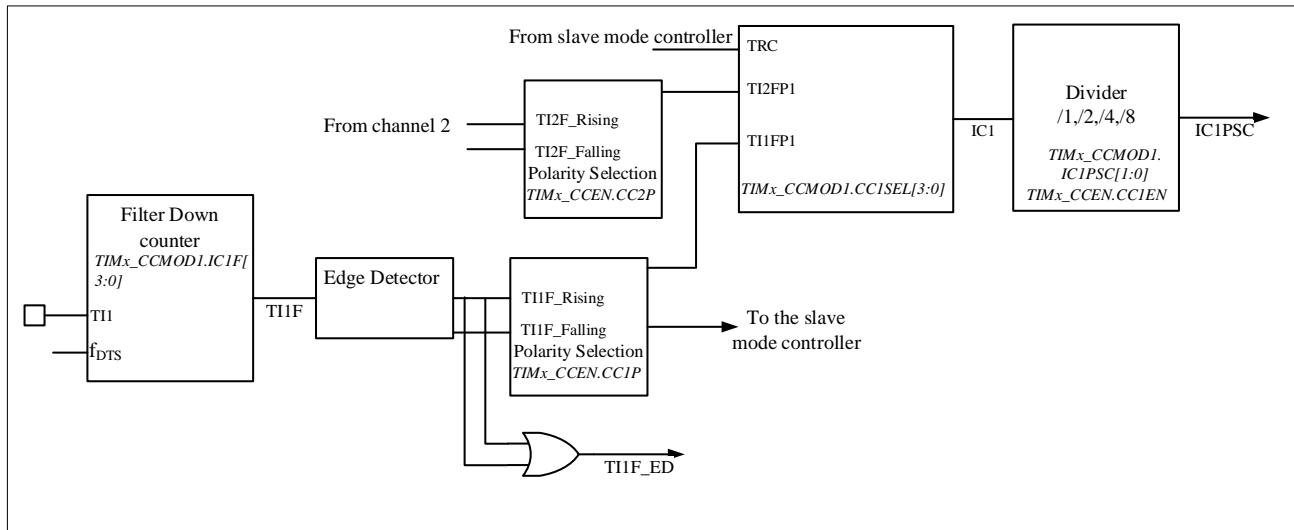


### 9.3.5 Capture/compare channels

Capture/compare channels include capture/compare registers and shadow registers. The input section consists of digital filters, multiplexers and prescalers. The output section includes comparators and output controls.

The input signal TI<sub>x</sub> is sampled and filtered to generate the signal TI<sub>x</sub>F. A signal (TI<sub>x</sub>F\_rising or TI<sub>x</sub>F\_falling) is then generated by the edge detector of the polarity select function, the polarity of which is selected by the TIMx\_CCEN.CCxP bits. This signal can be used as a trigger input for the slave mode controller. At the same time, the signal IC<sub>x</sub> is sent to the capture register after frequency division. The following figure shows a block diagram of a capture/compare channel.

Figure 9-16 Capture/compare channel (example: channel 1 input stage)



The output part generates an intermediate waveform OCxRef (active high) as reference. The polarity acts at the end of the chain.

Figure 9-17 Capture/compare channel 1 main circuit

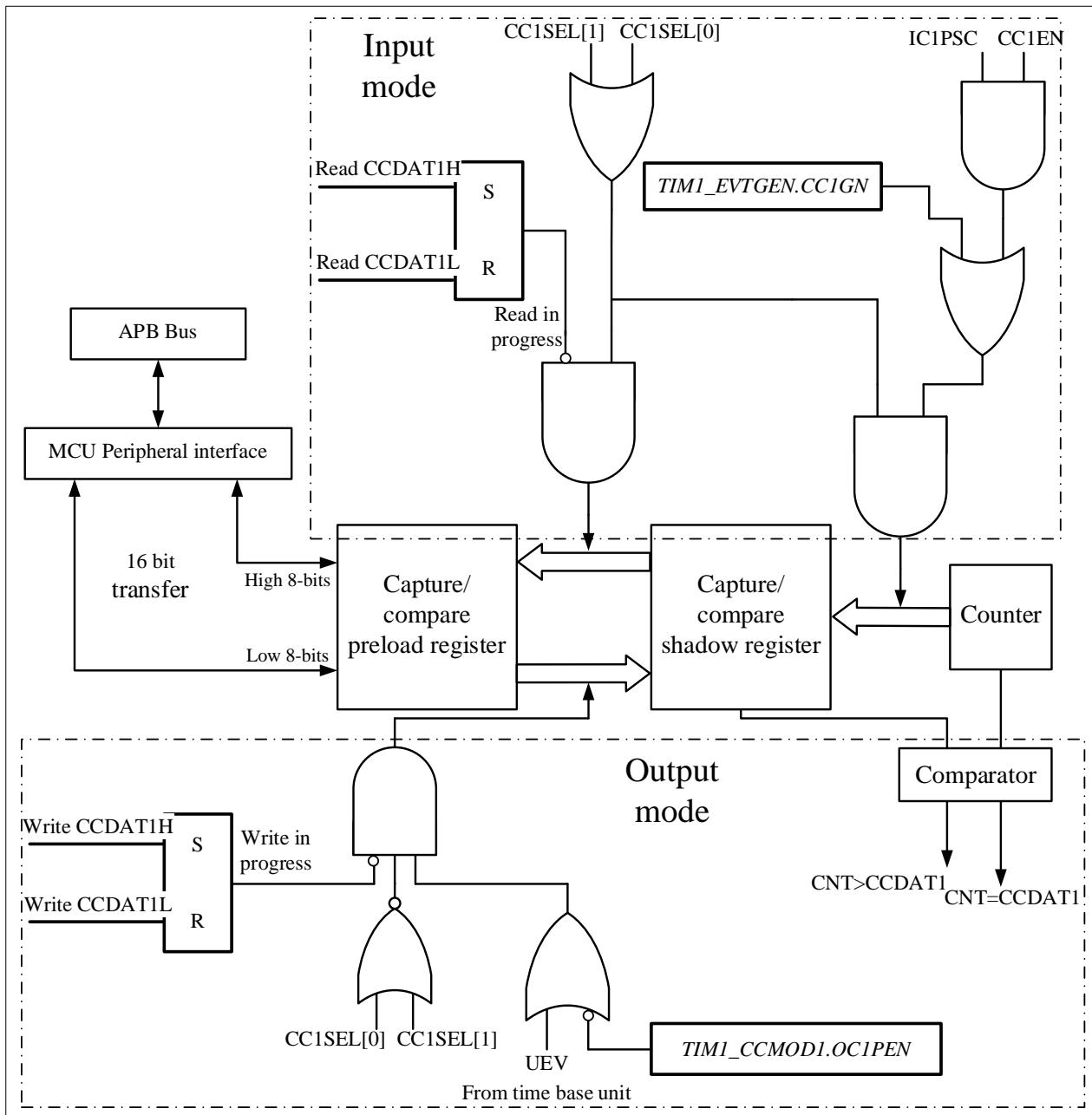


Figure 9-18 Output part of channelx (x= 1,2,3, take channel 1 as example)

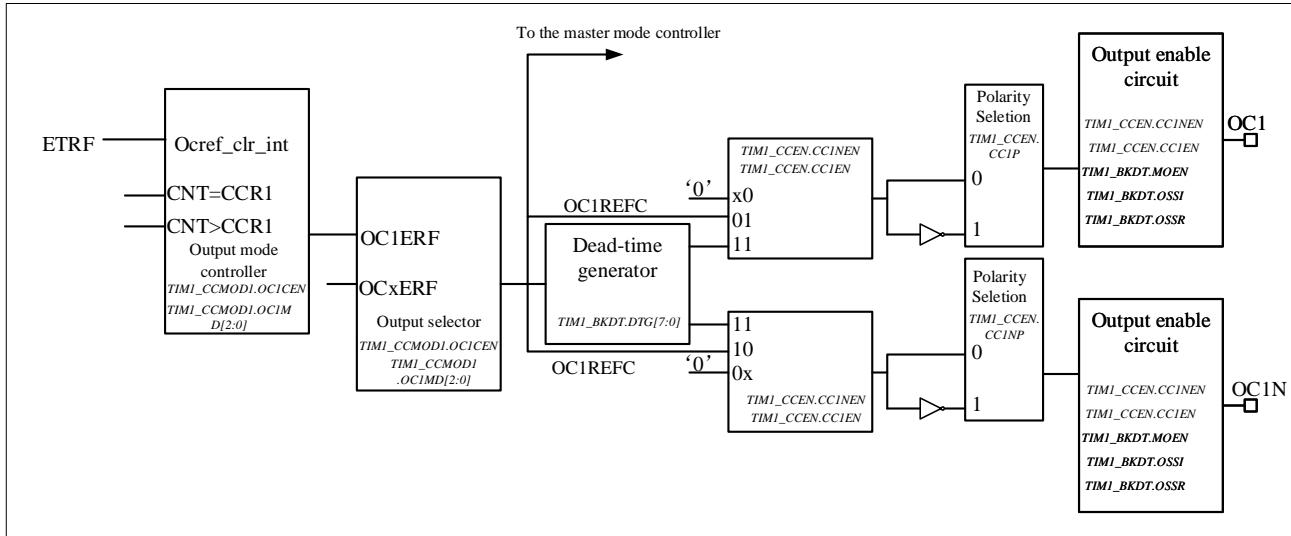
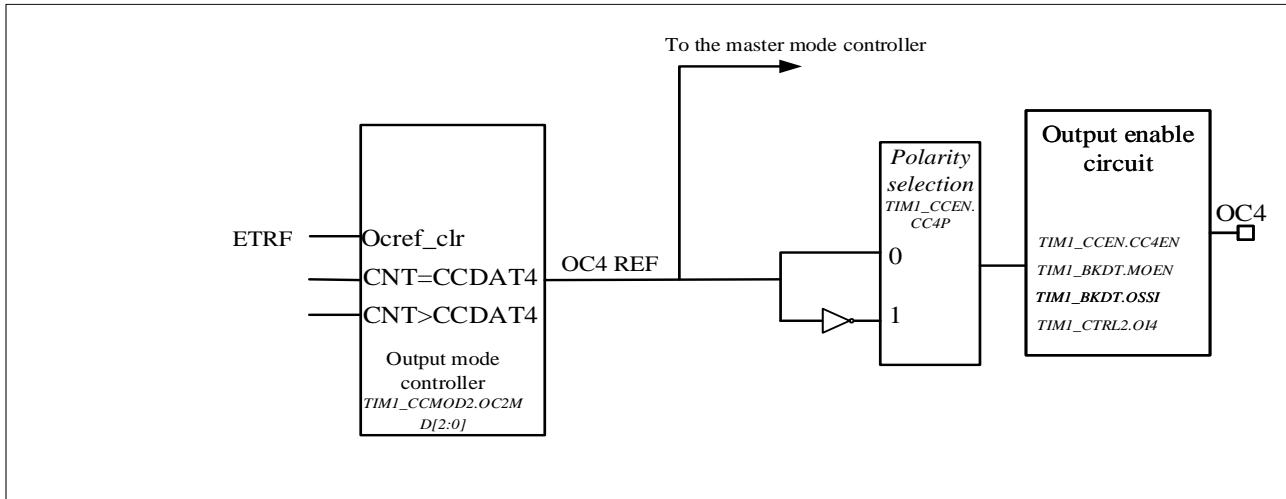


Figure 9-19 Output part of channelx (x=4)



Reads and writes always access preloaded registers when capturing/comparing. The two specific working processes are as follows:

In capture mode, the capture is actually done in the shadow register, and then the value in the shadow register is copied into the preload register.

In compare mode, as opposed to capture mode, the value of the preload register is copied into the shadow register, which is compared with the counter.

### 9.3.6 Input capture mode

In capture mode, the TIMx\_CCDATx registers are used to latch the counter value after the ICx signal detects.

There is a capture interrupt flag TIMx\_STS.CCxITF, which can issue an interrupt or DMA request if the corresponding interrupt enable is pulled high.

The TIMx\_STS.CCxITF bit is set by hardware when a capture event occurs and is cleared by software or by reading the TIMx\_CCDATx register.

The overcapture flag TIMx\_STS.CCxOCF is set equal to 1 when the counter value is captured in the TIMx\_CCDATx register and TIMx\_STS.CC1ITF is already pulled high. Unlike the former, TIMx\_STS.CCxOCF is cleared by writing 0 to it.

To achieve a rising edge of the TI1 input to capture the counter value into the TIMx\_CCDAT1 register, the configuration flow is as follows:

- To select a valid input:

Configure TIMx\_CCMOD1.CC1SEL to ‘01’. At this time, the input is the CC1 channel, and IC1 is mapped to TI1.

- Program the desired input filter duration:

Define the sampling frequency of the TI1 input and the length of the digital filter by configuring the TIMx\_CCMODx.ICxF bits. Example: If the input signal jitters up to 5 internal clock cycles, we must choose a filter duration longer than these 5 clock cycles. When 8 consecutive samples (sampled at  $f_{DTS}$  frequency) with the new level are detected, we can validate the transition on TI1. Then configure TIMx\_CCMOD1. IC1F to ‘0011’.

- By configuring TIMx\_CCEN.CC1P=0, select the rising edge as the valid transition polarity on the TI1 channel.
- Configure the input prescaler. In this example, configure TIMx\_CCMOD1.IC1PSC= ‘00’ to disable the prescaler because we want to capture every valid transition.
- Enable capture by configuring TIMx\_CCEN.CC1EN = ‘1’.

If you want to enable DMA request, you can configure TIMx\_DINTEN.CC1DEN=1. If you want enable related interrupt request, you can configure TIMx\_DINTEN.CC1IEN bit=1

### 9.3.7 PWM input mode

There are some differences between PWM input mode and normal input capture mode, including:

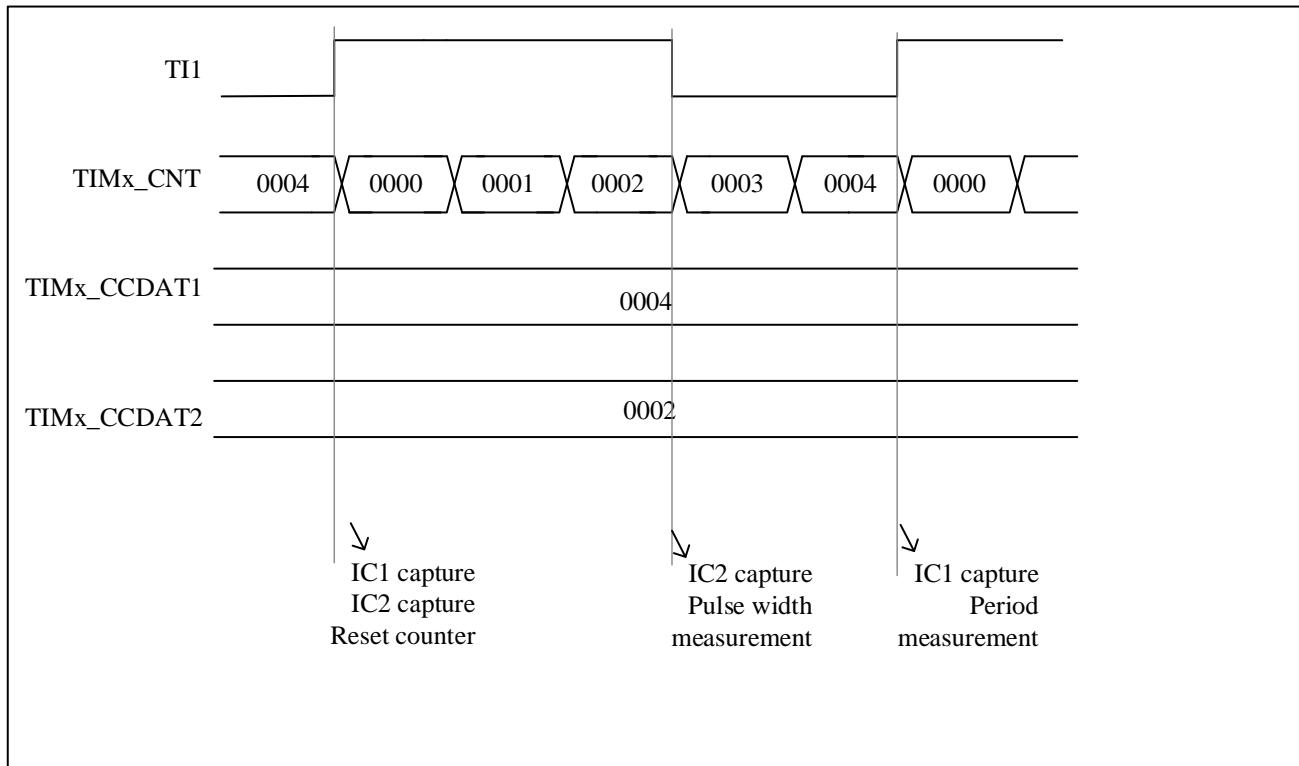
- Two ICx signals are mapped to the same TIx input.
- The two ICx signals are active on edges of opposite polarity.
- Select one of two TIxFP signals as trigger input.
- The slave mode controller is configured in reset mode.

For example, the following configuration flow can be used to know the period and duty cycle of the PWM signal on TI1 (It depends on the frequency of CK\_INT and the value of the prescaler).

- Configure TIMx\_CCMOD1.CC1SEL equal to ‘01’ to select TI1 as valid input for TIMx\_CCDAT1.
- Configure TIMx\_CCEN.CC1P equal to ‘0’ to select the active polarity of filtered timer input 1(TI1FP1), valid on the rising edge.
- Configure TIMx\_CCMOD1.CC2SEL equal to ‘10’ select TI1 as valid input for TIMx\_CCDAT2.

- Configure TIMx\_CCEN.CC2P equal to 1 to select the valid polarity of filtered timer input 2(TI1FP2), valid on the falling edge.
- Configure TIMx\_SMCTRL.TSEL=101 to select Filtered timer input 1 (TI1FP1) as valid trigger input.
- Configure TIMx\_SMCTRL.SMSEL=100 to configure the slave mode controller to reset mode.
- Configure TIMx\_CCEN.CC1EN=1 and TIMx\_CCEN.CC2EN=1 to enable capture.

Figure 9-20 PWM input mode timing



Because of only filter timer input 1 (TI1FP1) and filter timer input 2 (TI2FP2) are connected to the slave mode controller, the PWM input mode can only be used with the TIMx\_CH1/TIMx\_CH2 signals.

### 9.3.8 Forced output mode

Software can force output compare signals to active or inactive level directly, in output mode (TIMx\_CCMODx.CCxSEL=00).

User can set TIMx\_CCMODx. OCxMD=101 to force the output compare signal to active level. And the OCxREF will be forced high, OCx get opposite value to CCxP polarity bit. On the other hand, user can set TIMx\_CCMODx. OCxMD=100 to force the output compare signal to inactive level.

The values of the TIMx\_CCDATx shadow register and the counter still comparing with each other in this mode.

The comparison between the output compare register TIMx\_CCDATx and the counter TIMx\_CNT has no effect on OCxREF. And the flag still can be set. Therefore, the interrupt and DMA requests still can be sent.

### 9.3.9 Output compare mode

User can use this mode to control the output waveform, or to indicate that a period of time has elapsed.

When the capture/compare register and the counter have the same value, the output compare function's operations are as follow:

- TIMx\_CCMODx.OCxMD is for output compare mode, and TIMx\_CCEN.CCxP is for output polarity. When the compare matches, if set TIMx\_CCMODx.OCxMD=000, the output pin will keep its level;if set TIMx\_CCMODx.OCxMD=001, the output pin will be set active;if set TIMx\_CCMODx.OCxMD=010, the output pin will be set inactive;if set TIMx\_CCMODx.OCxMD=011, the output pin will be set to toggle.
- Set TIMx\_STS.CCxITF.
- If user set TIMx\_DINTEN.CCxIEN, a corresponding interrupt will be generated.
- If user set TIMx\_DINTEN.CCxDEN and set TIMx\_CTRL2.CCDSEL to select DMA request, and DMA request will be sent.

User can set TIMx\_CCMODx.OCxPEN to choose capture/compare shawdow regisete using capture/compare preload registers(TIMx\_CCDATx) or not.

The time resolution is one count of the counter.

In one-pulse mode, the output compare mode can also be used to output a single pulse.

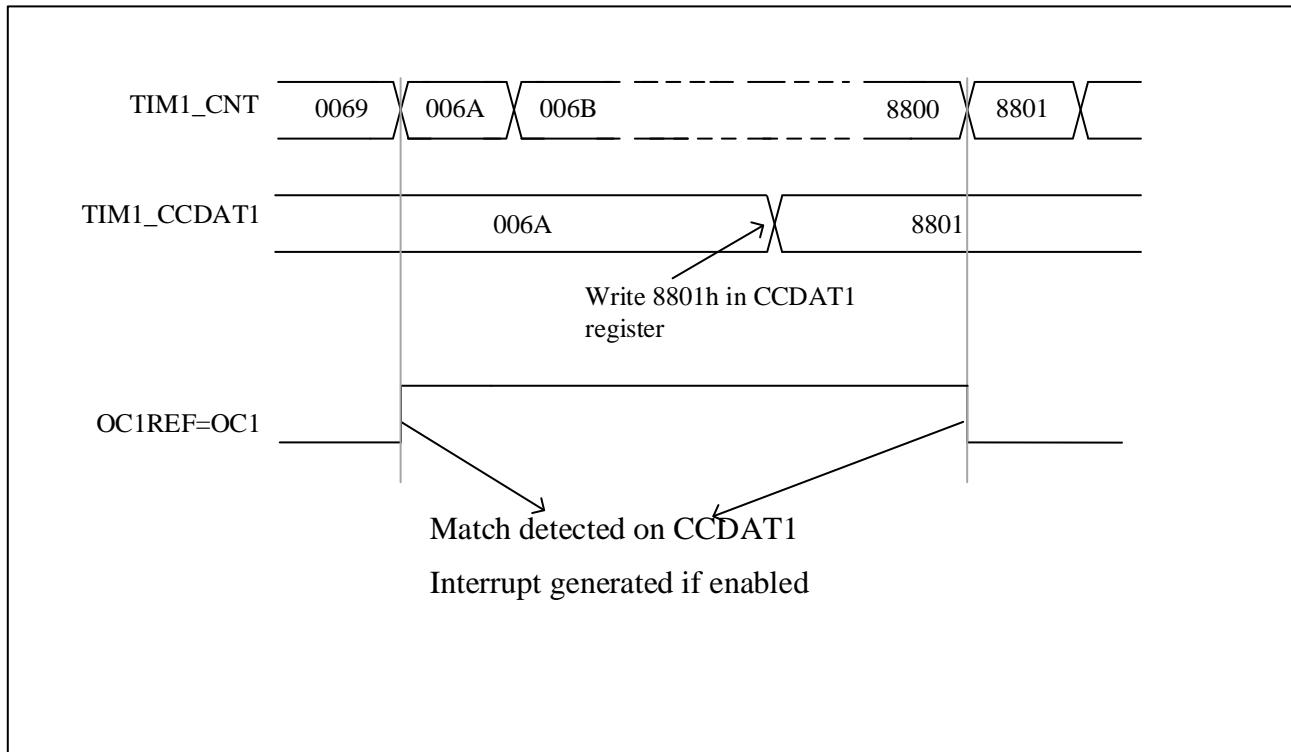
Here are the configuration steps for output compare mode:

- First of all, user should select the counter clock.
- Secondly, set TIMx\_AR and TIMx\_CCDATx with desired data.
- If user need to generate an interrupt, set TIMx\_DINTEN.CCxIEN.
- Then select the output mode by set TIMx\_CCEN.CCxP, TIMx\_CCMODx.OCxMD, TIMx\_CCEN.CCxEN, etc.
- At last, set TIMx\_CTRL1.CNTEN to enable the counter.

User can update the output waveform by setting TIMx\_CCDATx at any time, as long as the preload register is not enabled. Otherwise the TIMx\_CCDATx shadow register will be updated at the next update event.

Here is an example.

Figure 9-21 Output compare mode, toggle on OC1



### 9.3.10 PWM mode

User can use PWM mode to generate a signal whose duty cycle is determined by the value of the TIMx\_CCDATx register and whose frequency is determined by the value of the TIMx\_AR register. And depends on the value of TIMx\_CTRL1.CAMSEL, the TIM can generate PWM signal in edge-aligned mode or center-aligned mode.

User can set PWM mode 1 or PWM mode 2 by setting TIMx\_CCMODx. OCxMD=110 or setting TIMx\_CCMODx. OCxMD=111. To enable preload register, user must set corresponding TIMx\_CCMODx. OCxPEN. And then set TIMx\_CTRL1.AR PEN to auto-reload preload register eventually.

User can set polarity of OCx by setting TIMx\_CCEN.CCxP. On the other hand, to enable the output of OCx, user need to set the combination of the value of CCxEN, CCxNEN, MOEN, OSS1, and OSSR in TIMx\_CCEN and TIMx\_BKDT.

The values of TIMx\_CNT and TIMx\_CCDATx are always compared with each other when the TIM is under PWM mode.

Only if an update event occurs, the preload register will transfer to the shadow register. Therefore user must reset all the registers by setting TIMx\_EVTGEN. UDGN before the counter starts counting..

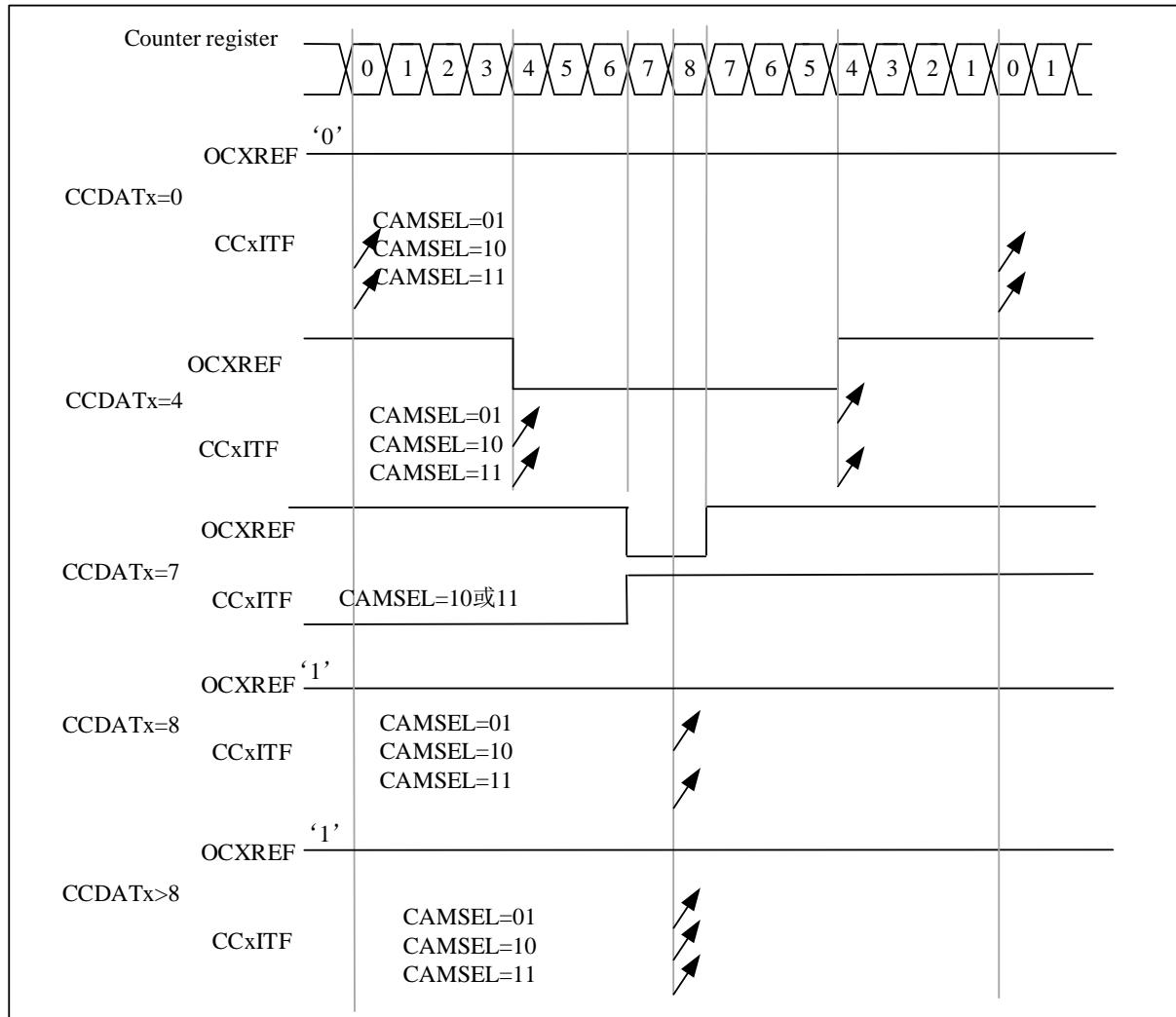
#### 9.3.10.1 PWM center-aligned mode

If user set TIMx\_CTRL1.CAMSEL equal 01, 10 or 11, the PWM center-aligned mode will be active. The setting of the compare flag depends on the value of TIMx\_CTRL1.CAMSEL. There are three kinds of situation that the compare flag is set, only when the counter counts up, only when the counter counts down, or when the counter counts

up and counts down. User should not modified TIMx\_CTRL1.DIR by software, it is updated by hardware.

Examples of center-aligned PWM waveforms is as follow, and the setting of the waveform are: TIMx\_AR=8, PWM mode 1, the compare flag is set when the counter counts down corresponding to TIMx\_CTRL1.CAMSEL=01.

Figure 9-22 Center-aligned PWM waveform (AR=8)



Notes that user should know when using center-aligned mode are as follow:

- It depends on the value of TIMx\_CTRL1.DIR that the counter counts up or down. Cautions that the DIR and CAMSEL bits should not be changed at the same time.
- User should not write the counter while running in center-aligned mode, otherwise it will cause unexpected results. Here are some example:
  - ◆ If the value written into the counter is 0 or is the value of TIMx\_AR, the direction will be updated but the update event will not be generated.
  - ◆ If the value written into the counter is greater than the value of auto-reload, the direction will not be updated.
- To be on the safe side, user is suggested setting TIMx\_EVTGEN.UDGN to generate an update by software

before starting the counter, and not writing the counter while it is running.

### 9.3.10.2 PWM edge-aligned mode

There are two kinds of configuration in edge-aligned mode, up-counting and down-counting.

- **Up-counting**

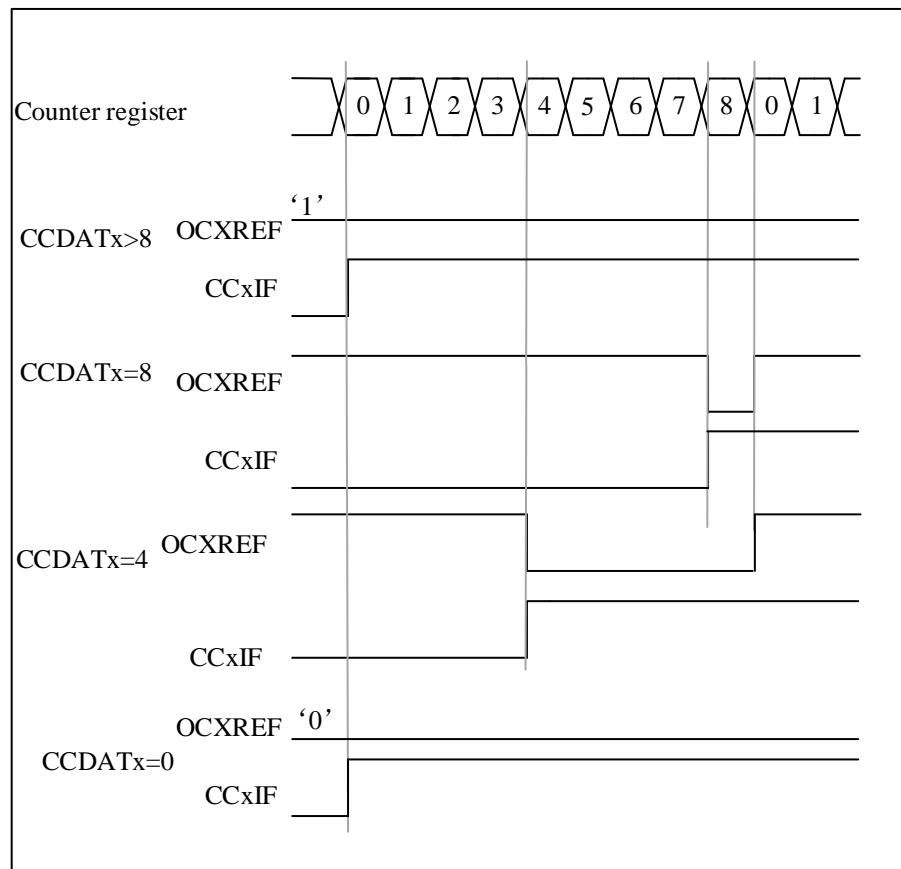
User can set `TIMx_CTRL1.DIR=0` to make counter counts up.

Here is an example for PWM mode1.

When  $\text{TIMx\_CNT} < \text{TIMx\_CCDATx}$ , the reference PWM signal OCxREF is high. Otherwise it will be low. If the compare value in `TIMx_CCXDATx` is greater than the auto-reload value, the OCxREF will remains 1. Conversely, if the compare value is 0, the OCxREF will remains 0.

When `TIMx_AR=8`, the PWM waveforms are as follow.

Figure 9-23 Edge-aligned PWM waveform (APR=8)



- **Down-counting**

User can set `TIMx_CTRL1.DIR=1` to make counter counts down.

Here is an example for PWM mode1.

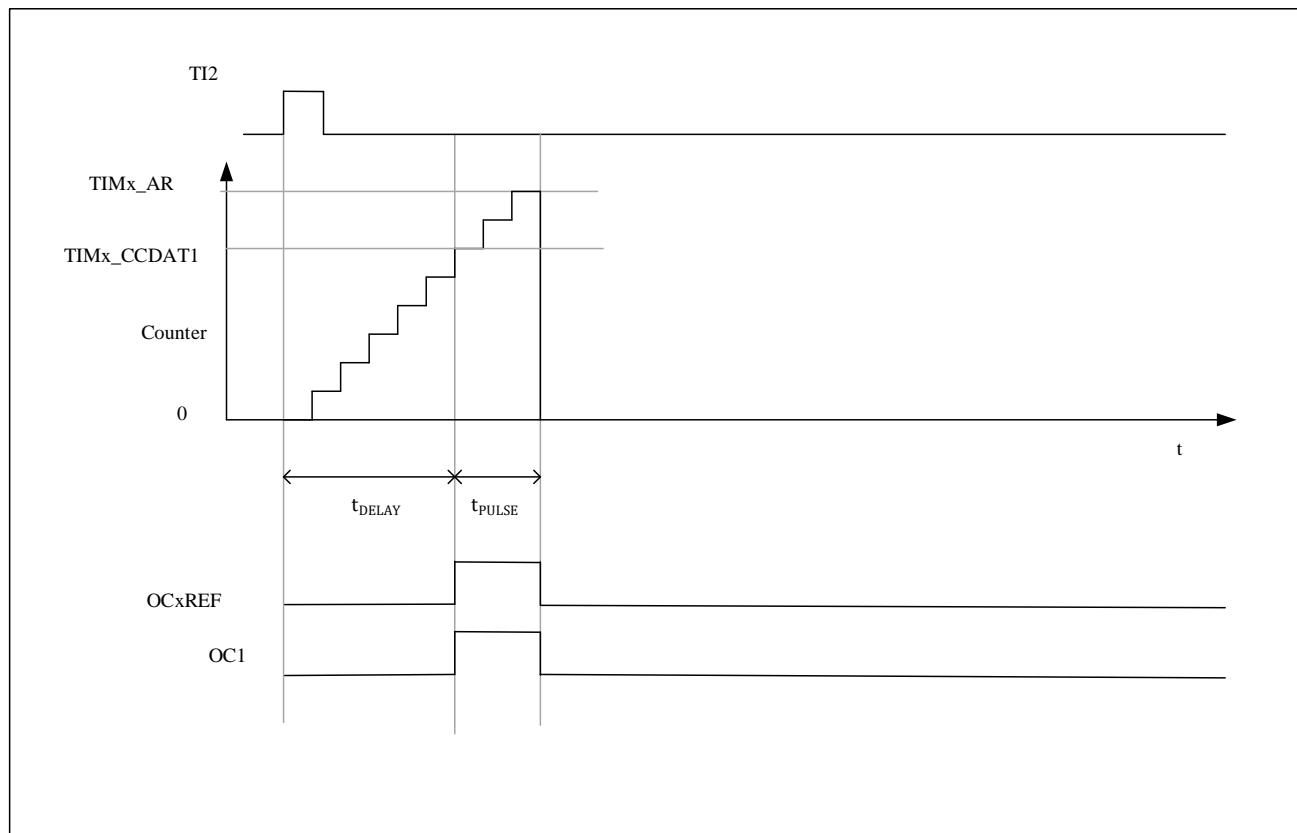
When  $\text{TIMx\_CNT} > \text{TIMx\_CCDATx}$ , the reference PWM signal OCxREF is low. Otherwise it will be high. If the compare value in `TIMx_CCXDATx` is greater than the auto-reload value, the OCxREF will remains 1. Conversely, if the compare value is 0, the OCxREF will remains 0.

Note: If the nth PWM cycle  $CCDATx$  shadow register  $\geq AR$  value, the shadow register value of  $CCDATx$  in the  $(n+1)$ th PWM cycle is 0. At the moment when the counter is 0 in the  $(n+1)$ th PWM cycle, although the value of the counter =  $CCDATx$  shadow register = 0 and  $OCxREF$  = '0', no compare event will be generated.

### 9.3.11 One-pulse mode

In the one-pulse mode (ONEPM), a trigger signal is received, and a pulse  $t_{PULSE}$  with a controllable pulse width is generated after a controllable delay  $t_{DELAY}$ . The output mode needs to be configured as output compare mode or PWM mode. After selecting one-pulse mode, the counter will stop counting after the update event UEV is generated.

Figure 9-24 Example of One-pulse mode



The following is an example of a one-pulse mode:

A rising edge trigger is detected from the TI2 input, and a pulse with a width of  $t_{PULSE}$  is generated on OC1 after a delay of  $t_{DELAY}$ .

1. Counter configuration: count up, counter  $TIMx\_CNT < TIMx\_CCDAT1 \leq TIMx\_AR$ ;
2. TI2FP2 is mapped to TI2,  $TIMx\_CCMOD1.CC2SEL = '01'$ ; TI2FP2 is configured for rising edge detection,  $TIMx\_CCEN.CC2P = '0'$ ;
3. TI2FP2 acts as the trigger (TRGI) of the slave mode controller and starts the counter,  $TIMx\_SMCTRL.TSEL = '110'$ ,  $TIMx\_SMCTRL.SMSEL = '110'$  (trigger mode);
4.  $TIMx\_CCDAT1$  writes the count value to be delayed ( $t_{DELAY}$ ),  $TIMx\_AR - TIMx\_CCDAT1$  is the count value of

the pulse width  $t_{PULSE}$ ;

5. Configure TIMx\_CTRL1.ONEPM=1 to enable single pulse mode, configure TIMx\_CCMOD1.OC1MD = '111' to select PWM2 mode;

6. Wait for an external trigger event on TI2, and a one pulse waveform will be output on OC1;

### 9.3.11.1 Special case: OCx fast enable:

In one-pulse mode, an edge is detected through the TIx input, and triggers the start of the counter to count to the comparison value and then output a pulse. These operations limit the minimum delay  $t_{DELAY}$  that can be achieved.

You can set TIMx\_CCMODx.OCxFEN=1 to turn on OCx fast enable, after triggering the rising edge, the OCxREF signal will be forced to be converted to the same level as the comparison match occurs immediately, regardless of the comparison result. OCxFEN fast enable only takes effect when the channel mode is configured for PWM1 and PWM2 modes.

### 9.3.12 Clearing the OCxREF signal on an external event

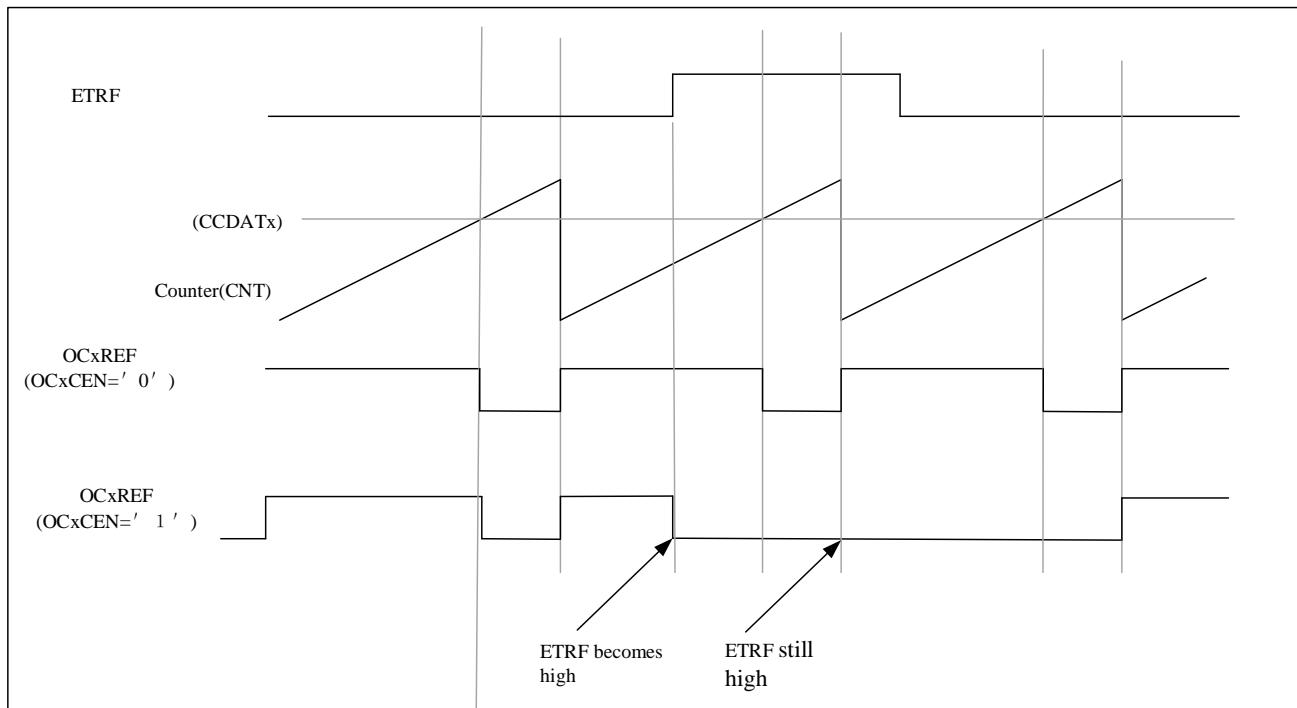
If user set TIMx\_CCMODx.OCxCEN=1, high level of ETRF input can be used to driven the OCxREF signal to low, and the OCxREF signal will remains low, until the next UEV happens. Only output compare and PWM modes can use this function. This cannot be used when it is in forced mode.

Here is an example for it. To control the current, user can connect the ETR signal to the output of a comparator, and the operation for ETR should be as follow:

- Set TIMx\_SMCTRL.EXTPS=00 to disable the external trigger prescaler.
- Set TIMx\_SMCTRL.EXCEN=0 to disable the external clock mode 2.
- Set TIMx\_SMCTRL.EXTP and TIMx\_SMCTRL.EXTF to configure the external trigger polarity and external trigger filter according to the need.

Here is an example for the case that when ETRF input becomes high, the behavior of OCxREF signal for different value of OCxCEN. Timer is set to be in PWM mode in this case.

Figure 9-25 Clearing the OCxREF of TIMx



### 9.3.13 Complementary outputs with dead-time insertion

Advanced-control timer can output two complementary signals, and manage the switching-off and switching-on of outputs. This is called dead-time. User should adjust dead-time depending on the devices connected to the outputs and their characteristics.

User can select the polarity of outputs by setting TIMx\_CCEN.CCxP and TIMx\_CCEN.CCxNP. And this selection is independently for each output.

User can control the complementary signals OCx and OCxN by setting the combination of several control bits, which are TIMx\_CCEN.CCxEN, TIMx\_CCEN.CCxNEN, TIMx\_BKDT.MOEN, TIMx\_CTRL2.OIx, TIMx\_CTRL2.OIxN, TIMx\_BKDT.OSSI, and TIMx\_BKDT.OSSR. When switching to the IDLE state, the dead-time will be activated.

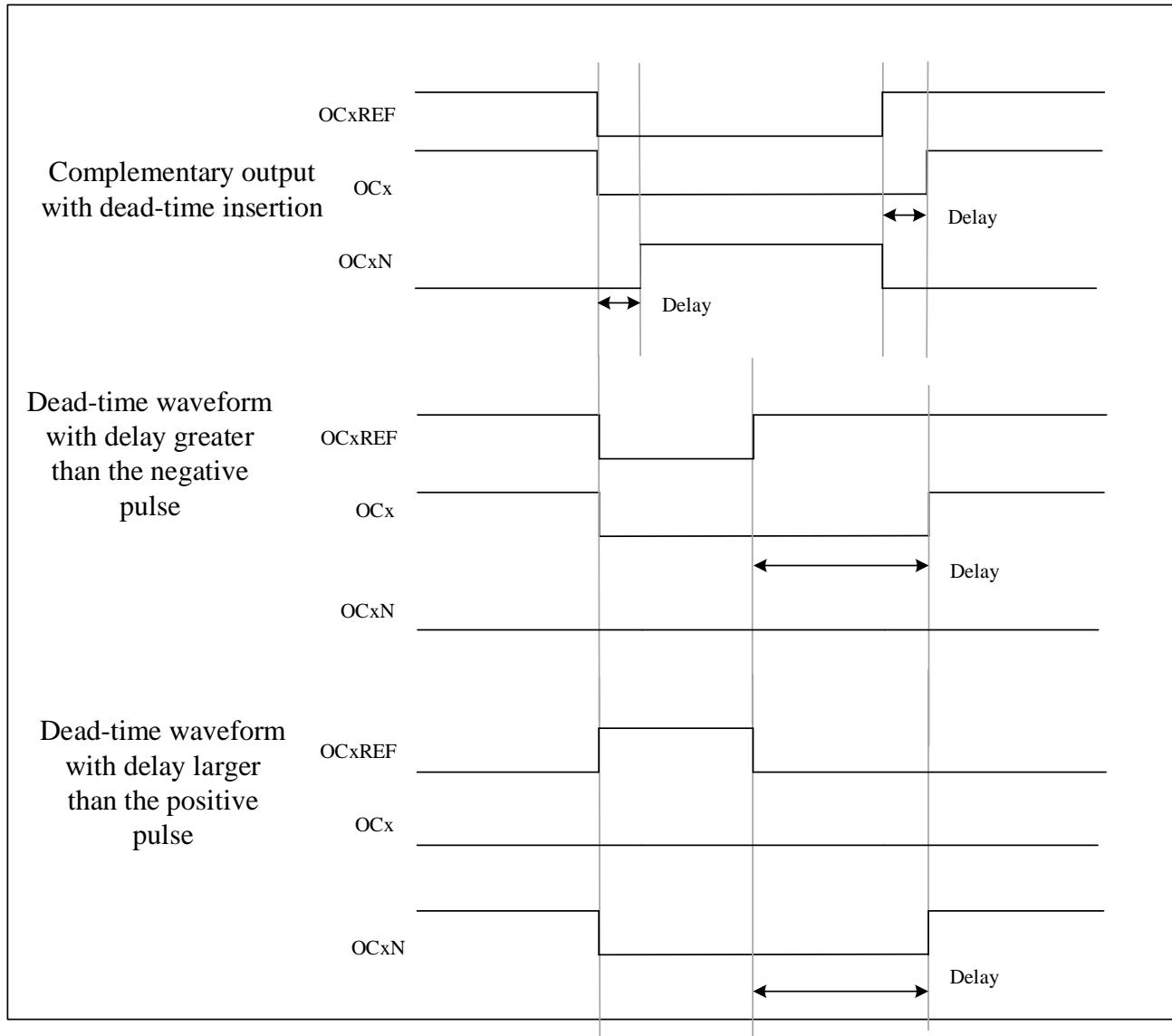
If user set TIMx\_CCEN.CCxEN and TIMx\_CCEN.CCxNEN at the same time, a dead-time will be insert. If there is a break circuit, the TIMx\_BKDT.MOEN should be set too. There are 10-bit dead-time generators for each channel.

Reference waveform OCxREF can generates 2 outputs OCx and OCxN. And if OCx and OCxN are active high, the OCx ouput signal is the same as the reference signal and the OCxN output signal is the opposite of the reference signal. However, OCx output signal will be delayed relative to the reference rising edge and the OCxN output signal will be delayed relative to the reference falling edge. If the delay is greater than the width of the active OCx or OCxN output, the corresponding pulse will not generated.

The relationships between the output signals of the dead-time generator and the reference signal OCxREF are as follow.

Assume that  $\text{TIMx\_CCEN.CCxP}=0$ ,  $\text{TIMx\_CCEN.CCxNP}=0$ ,  $\text{TIMx\_BKDT.MOEN}=1$ ,  $\text{TIMx\_CCEN.CCxEN}=1$ ,  $\text{TIMx\_CCEN.CCxNEN}=1$ .

Figure 9-26 Complementary output with dead-time insertion



User can set  $\text{TIMx\_BKDT.DTGN}$  to programme the dead-time delay for each of the channels.

### 9.3.13.1 Redirecting OCxREF to OCx or OCxN

User can set  $\text{TIMx\_CCEN.CCxEN}$  and  $\text{TIMx\_CCEN.CCxNEN}$  to re-directed OCxREF to the OCx output or to OCxN output, in output mode.

Here are two ways to use this function. When the complementary remains at its inactive level, user can use this function to send a specific waveform, such as PWM or static active level. User can also use this function to set both outputs in their inactive level or both outputs active and complementary with dead-time.

If user set  $\text{TIMx\_CCEN.CCxEN}=0$  and  $\text{TIMx\_CCEN.CCxNEN}=1$ , it will not complemented, and OCxN will become active when OCxREF is high. On the other hand, if user set  $\text{TIMx\_CCEN.CCxEN}=1$  and

TIMx\_CCEN.CCxNEN=1, OCx will become active when OCxREF is high. On the contrary, OCxN will become active when OCxREF is low.

### 9.3.14 Break function

The output enable signals and inactive levels will be modified when setting the corresponding control bits when using the break function. However, the output of OCx and OCxN cannot be at the active level at the same time no matter when, that is,  $(CCxP \wedge OIx) \wedge (CCxNP \wedge OIxN) \neq 0$ .

When multiple break signals are enabled, each break signal constitutes an OR logic. Here are some signal which can be the source of breaking.

- The break input pin
- A clock failure event, generated by the clock security system in the clock controller.
- A PVD failure event.
- Core Hardfault event.
- The output signal of the comparator (configured in the comparator module, high level break).
- By software through the TIMx\_EVTGEN.BGN.

The break circuit will be disable after reset. And the MOEN bit will be low. User can set TIMx\_BKDT.BKEN to enable the break function. The polarity of break input signal can be selected by setting TIMx\_BKDT.BKP. User can modify the TIMx\_BKDT.BKEN and TIMx\_BKDT.BKP at the same time. After user set the TIMx\_BKDT.BKEN and TIMx\_BKDT.BKP, there is 1 APB clock cycle delay before the option take effect. Therefore, user need to wait 1 APB clock cycle to read back the value of the written bit.

The falling edge of MOEN can be asynchronous, so between the actual signal and the synchronous control bit, there set a resynchronization circuit. This circuit will cause a delay between the asynchronous and the synchronous signal. When user set TIMx\_BKDT.MOEN while it is low, user need to insert a delay before reading the value. Because an asynchronous signal was written but user read the synchronous signal.

The behaviors that after a break occurs are as follow:

- TIMx\_BKDT.MOEN will be cleared asynchronously, and then the outputs will be put in inactive state, idle state or reset state. The state of output is selected by setting TIMx\_BKDT.OSSI. This will take effect even if the MCU oscillator is off.
- Once TIMx\_BKDT.MOEN=0, the output of each output channel will be driven with the level programmed in TIMx\_CTRL2.OIx. Timer will release the enable outputs(taken over by GPIO controller) if TIMx\_BKDT.OSSI=0, otherwise it will remains high.
- If user choose to use complementary outputs, the behaviors of TIM are as follow
  - ◆ Depends on the polarity, the outputs will be set in reset state first. It is an asynchronous option so it still works even if there is no clock provided to the timer.
  - ◆ The dead-time generator will reactivated if the timer clock is still provided, and drive the outputs according to the value of TIMx\_CTRL2.OIx and TIMx\_CTRL2.OIxN after the dead-time when  $(CCxP \wedge OIx) \wedge$

$(CCxNP^OIxN)! = 0$ , that is, the OCx and OCxN still cannot be driven to active level at the same time.

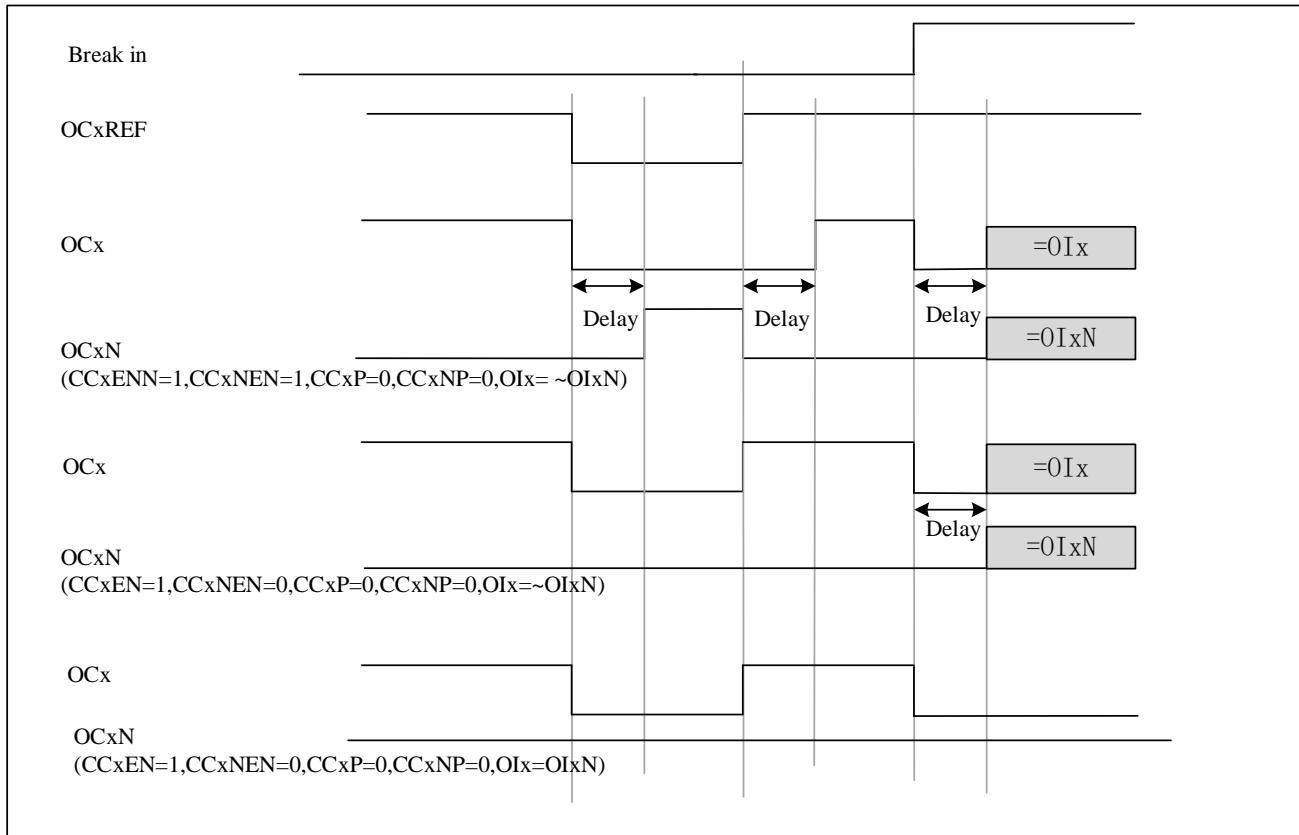
Note that the dead-time will be longer than usual because of the resynchronization on MOEN (almost 2 cycles of ck\_tim).

- ◆ Timer will release the output control if TIMx\_BKDT.OSSI=0. Otherwise, if the enable output was high, it will remain high. If it was low, it will become high when TIMx\_CCEN.CCxEN or TIMx\_CCEN.CCxNEN is high.
- If TIMx\_DINTEN.BIEN=1, when TIMx\_STS.BITF=1, an interrupt will be generated.
- If user set TIMx\_BKDT.AOEN, the TIMx\_BKDT.MOEN will be set automatically when the next UEV happened. User can use this to regulate. If user did not set TIMx\_BKDT.AOEN, the TIMx\_BKDT.MOEN will remain low until been set 1 again. At this situation, user can use this for security. User can connect the break input to thermal sensors, alarm for power drivers, or other security components.
- When the break input is active, TIMx\_BKDT.MOEN cannot be set automatically or by software at the same time, and the TIMx\_STS.BITF cannot be cleared. Because the break inputs are active on level.

To insure the security of application, the break circuit has the write protection function, and there is break input and output management too. It allow user to freeze some parameters, such as dead-time duration, OCx/OCxN polarities and state when disabled, OCxMD configurations, break enable and polarity. User can choose one of the 3 levels of protection to use by setting TIMx\_BKDT.LCKCFG. However, the TIMx\_BKDT.LCKCFG can only be written once after an MCU reset.

An example for output behavior in response to a break is as follow

Figure 9-27 Output behavior in response to a break



### 9.3.15 Debug mode

When the microcontroller is in debug mode (the Cortex-M0 core halted), depending on the `DBG_CTRL.TIMx_STOP` configuration in the PWR module, the TIMx counter can either continue to work normally or stop. For more details, see 3.3.2.

### 9.3.16 TIMx and external trigger synchronization

TIMx timers can be synchronized by a trigger in slave modes (reset, trigger and gated).

#### 9.3.16.1 Slave mode: Reset mode

In reset mode, the trigger event can reset the counter and the prescaler updates the preload registers `TIMx_AR`, `TIMx_CCDATx`, and generates the update event `UEV` (`TIMx_CTRL1.UPRS=0`).

The following is an example of a reset mode:

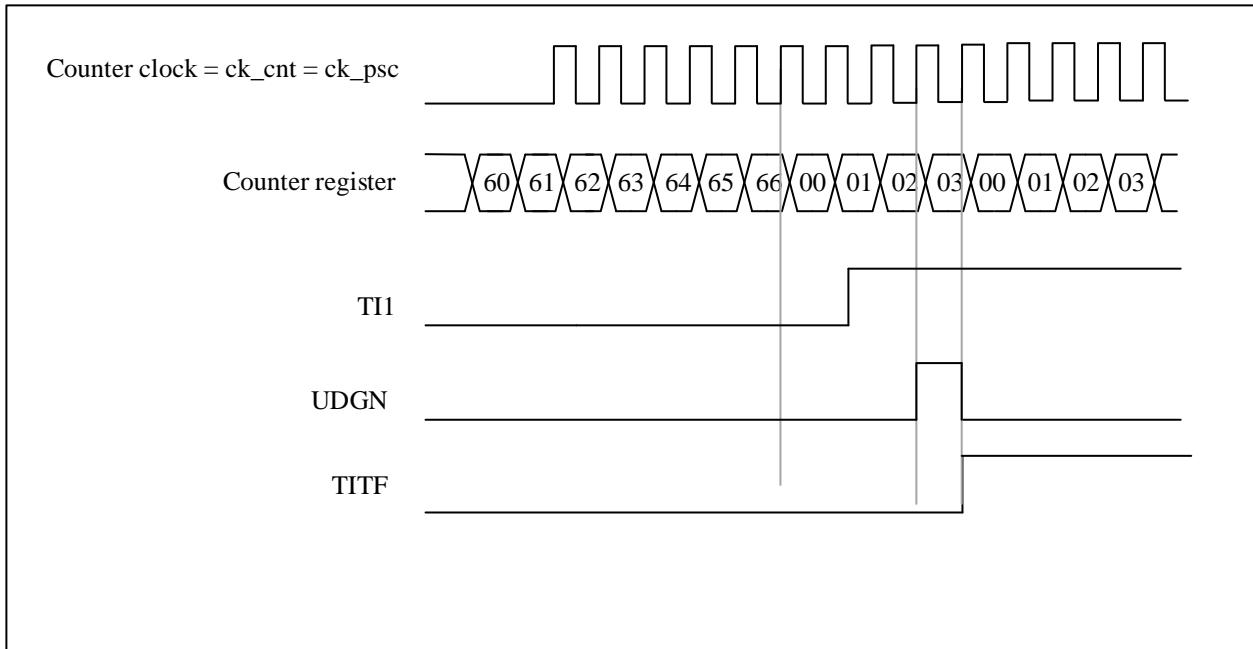
1. Channel 1 is configured as input to detect the rising edge of TI1 (`TIMx_CCMOD1.CC1SEL=01`, `TIMx_CCEN.CC1P=0`);
2. The slave mode is selected as reset mode (`TIMx_SMCTRL.SMSEL=100`), and the trigger input is selected as TI1 (`TIMx_SMCTRL.TSEL=101`);

3. Start counter( $\text{TIMx\_CTRL1.CNTEN} = 1$ ).

After starting the timer, when TI1 detects a rising edge, the counter resets and restarts counting, and the trigger flag is set ( $\text{TIMx\_STS.TITF}=1$ );

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 9-28 Control circuit in reset mode



### 9.3.16.2 Slave mode: Trigger mode

In trigger mode, the trigger event (rising edge/falling edge) of the input port can trigger the counter to start counting.

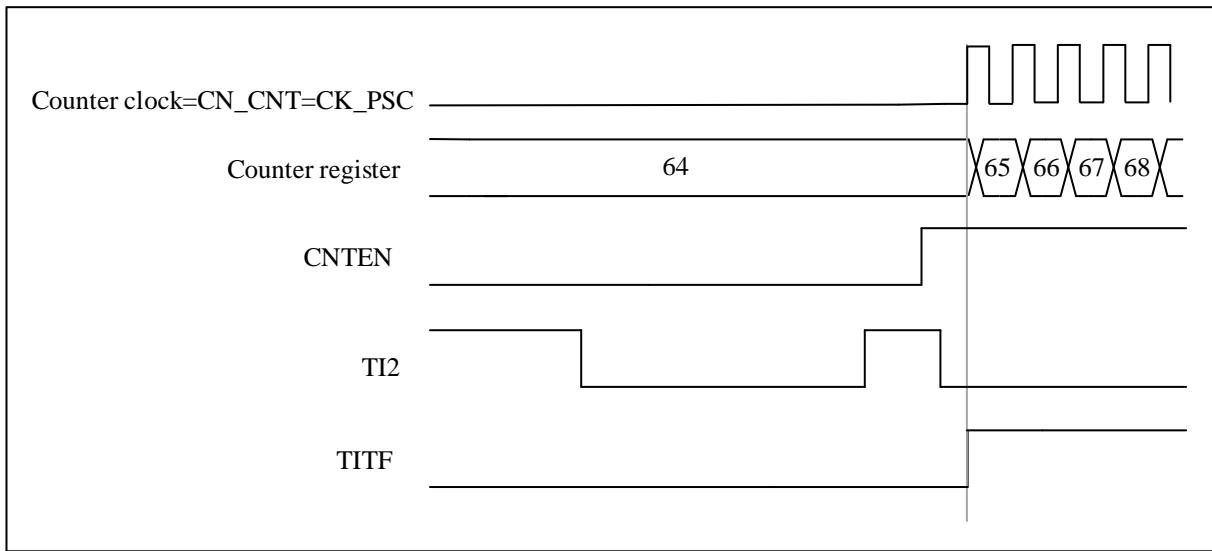
The following is an example of a trigger pattern:

1. Channel 2 is configured as input to detect the rising edge of TI2 ( $\text{TIMx_CCMOD1.CC2SEL}=01$ ,  $\text{TIMx_CCEN.CC2P}=0$ );
2. Select from mode to trigger mode ( $\text{TIMx_SMCTRL.SMSEL}=110$ ), select TI2 for trigger input ( $\text{TIMx_SMCTRL.TSEL}=110$ );

When TI2 detects a rising edge, the counter starts counting, and the trigger flag is set ( $\text{TIMx_STS.TITF}=1$ );

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 9-29 Control circuit in Trigger mode



### 9.3.16.3 Slave mode: Gated mode

In gate control mode, the level polarity of the input port can control whether the counter counts.

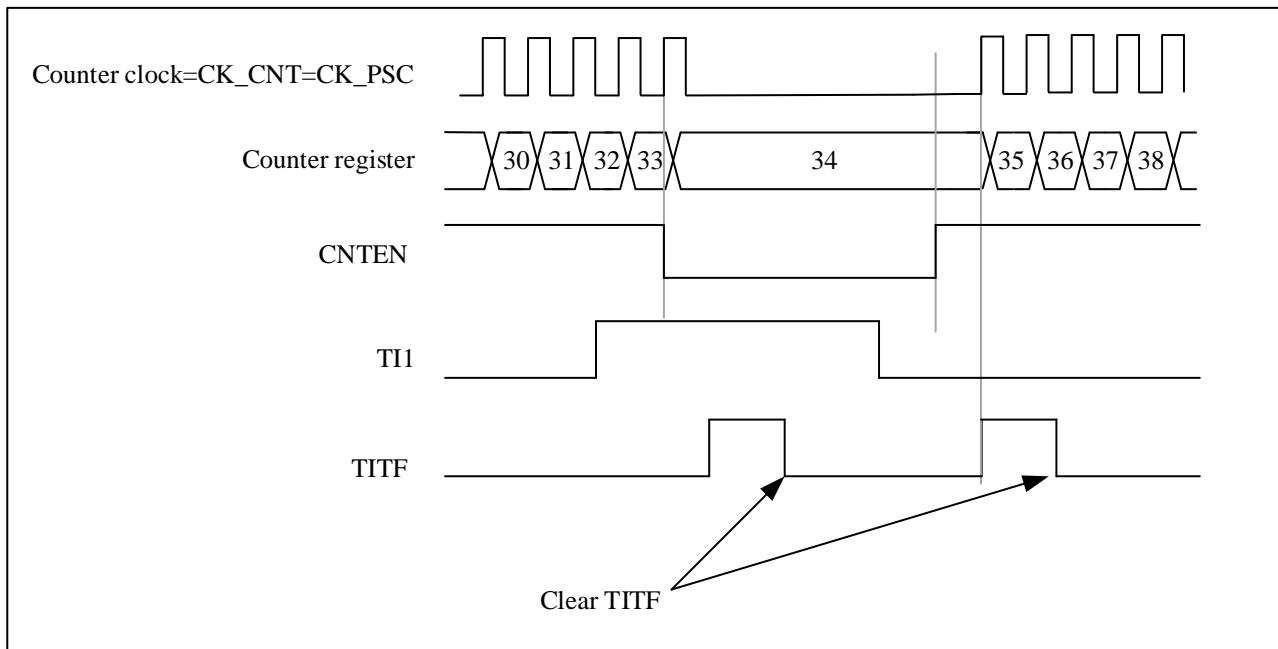
The following is an example of a gated pattern:

1. Channel 1 is configured as input detection active low on TI1 (TIMx\_CCMOD1.CC1SEL=01, TIMx\_CCEN.CC1P=1);
2. Select the slave mode as the gated mode (TIMx\_SMCTRL.SMSEL=101), and select TI1 as the trigger input (TIMx\_SMCTRL.TSEL=101);
3. Start counter(TIMx\_CTRL1.CNTEN=1).

When TI1 detects that the level changes from low to high, the counter stops counting, and when TI1 detects that the level changes from high to low, the counter starts counting, and the trigger flag will be set when it starts or stops counting (TIMx\_STS.TITF=1);

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 9-30 Control circuit in Gated mode



### 9.3.16.4 Slave mode: Trigger Mode + External Clock Mode 2

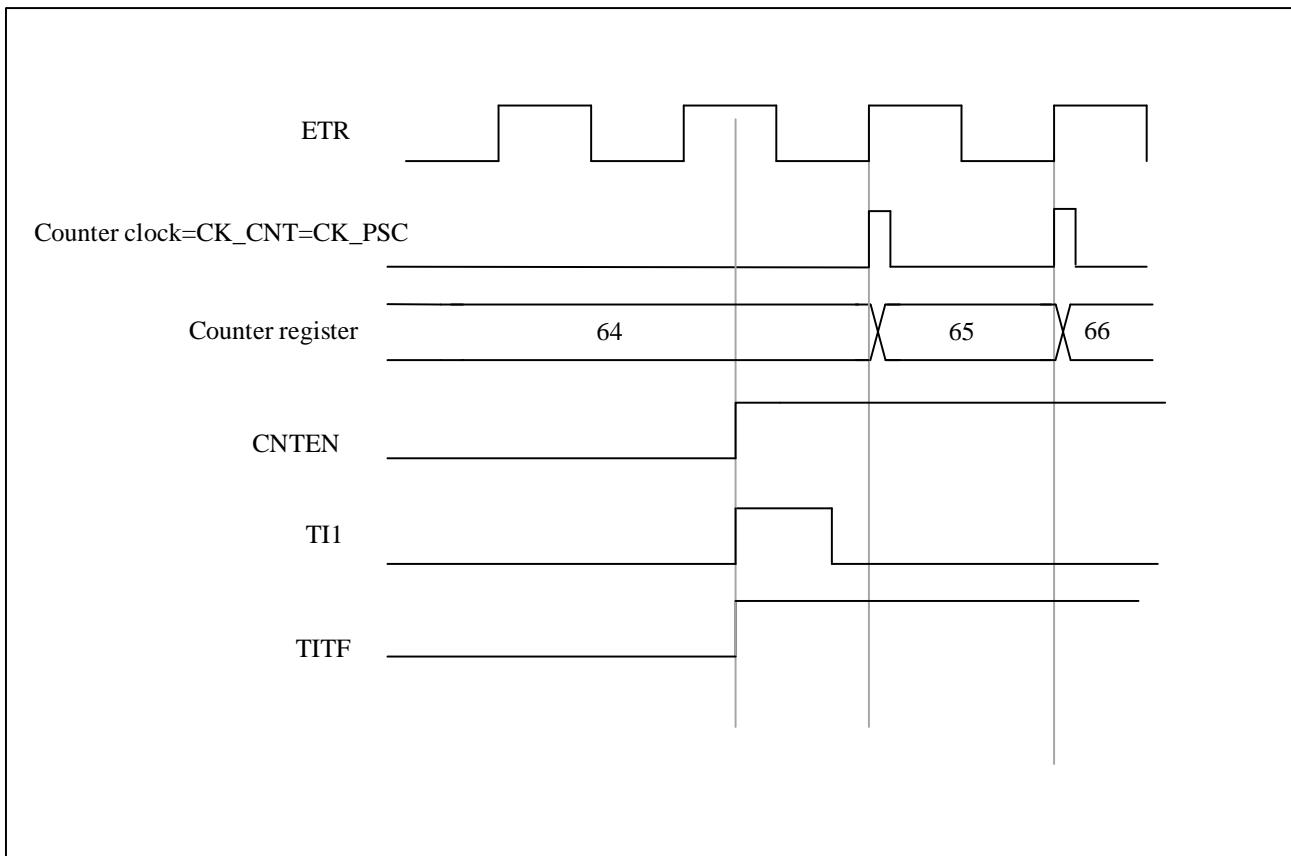
In reset mode, trigger mode and gate control mode, the counter clock can be selected as external clock mode 2, and the ETR signal is used as the external clock source input. At this time, the trigger selection needs to select non-ETRF (TIMx\_SMCTRL.TSEL=111).

Here is an example:

1. Channel 1 is configured as input to detect the rising edge of TI1 (TIMx\_CCMOD1.CC1SEL=01, TIMx\_CCEN.CC1P=0);
2. Enable external clock mode 2 (TIMx\_SMCTRL.EXCEN=1), select rising edge for external trigger polarity (TIMx\_SMCTRL.EXTP=0), select slave mode as trigger mode (TIMx\_SMCTRL.SMSEL=110), select TI1 for trigger input (TIMx\_SMCTRL.TSEL=101);

When TI1 detects a rising edge, the counter starts counting on the rising edge of ETR, and the trigger flag is set (TIMx\_STS.TITF=1);

Figure 9-31 Control circuit in Trigger Mode + External Clock Mode2



### 9.3.17 Timer synchronization

All TIM timers are internally connected for timer synchronization or chaining. For more details, see 10.3.14.

### 9.3.18 6-step PWM generation

In order to modify the configuration of all channels at the same time, the configuration of the next step can be set in advance (the preloaded bits are OCxMD, CCxEN and CCxNEN). When a COM commutation event occurs, the OCxMD, CCxEN, and CCxNEN preload bits are transferred to the shadow register bits.

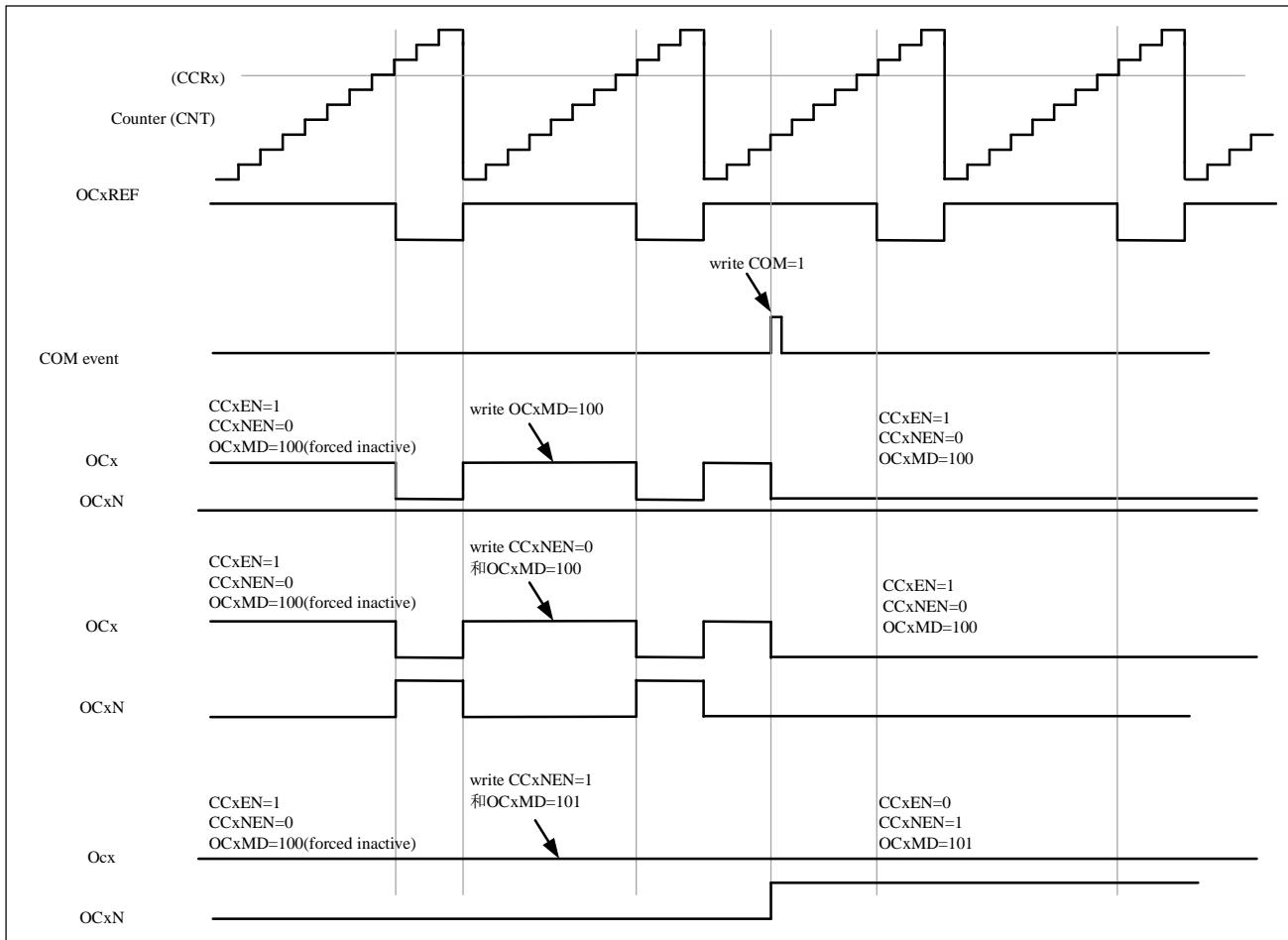
COM commutation event generation method:

1. The software sets TIMx\_EVTGEN.CCUDGN;
2. Generated by hardware on the rising edge of TRGI;

When a COM commutation event occurs, the TIMx\_STS.COMITF flag will be set, enabling interrupts (TIMx\_DINTEN.COMIEN) will generate interrupts, and enabling DMA requests (TIMx\_DINTEN.COMDEN) will generate DMA requests.

The following figure shows the output timing diagram of OCx and OCxN when a COM commutation event occurs in three different configurations:

Figure 9-32 6-step PWM generation, COM example (OSSR=1)



### 9.3.19 Encoder interface mode

The encoder uses two inputs TI1 and TI2 as an interface and the counter counts on every edge change on TI1FP1 or TI2FP2. The counting direction is automatically controlled by hardware TIMx\_CTRL1.DIR. There are three types of encoder counting modes:

1. The counter only counts on the edge of TI1, TIMx\_SMCTRL.SMSEL = ‘001’;
2. The counter only counts on the edge of TI2, TIMx\_SMCTRL.SMSEL =‘010’;
3. The counter counts on the edges of TI1 and TI2 at the same time, TIMx\_SMCTRL.SMSEL = ‘011’;

The encoder interface is equivalent to using an external clock with direction selection, and the counter only counts continuously between 0 and the auto-reload value (TIMx\_AR.AR [15:0]). Therefore, it is necessary to configure the auto-reload register TIMx\_AR in advance.

*Note: Encoder mode and external clock mode 2 are not compatible and must not be selected together.*

The relationship between the counting direction and the encoder signal is shown in **Table 9-1 Counting direction versus encoder signals:**

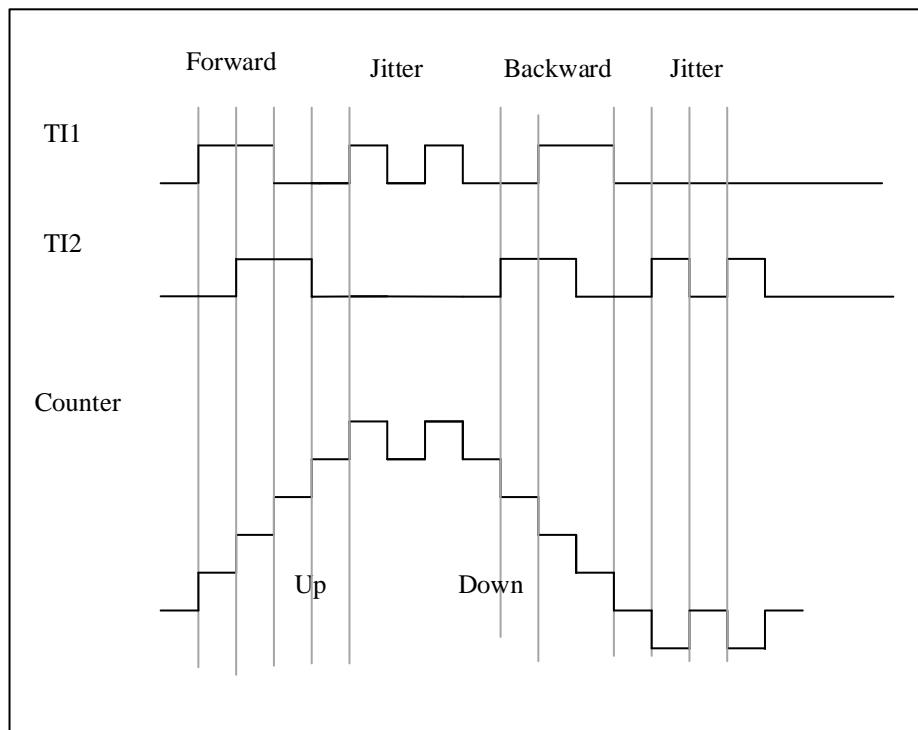
Table 9-1 Counting direction versus encoder signals

Active edge	Level on opposite signals (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI2FP2 signal	
		Rising	Falling	Rising	Falling
Counting only at TI1	High	Counting down	Counting up	Don't count	Don't count
	Low	Counting up	Counting down	Don't count	Don't count
Counting only at TI2	High	Don't count	Don't count	Counting up	Counting down
	Low	Don't count	Don't count	Counting down	Counting up
Counting on TI1 and TI2	High	Counting down	Counting up	Counting up	Counting down
	Low	Counting up	Counting down	Counting down	Counting up

Here is an example of an encoder with dual edge triggering selected to suppress input jitter:

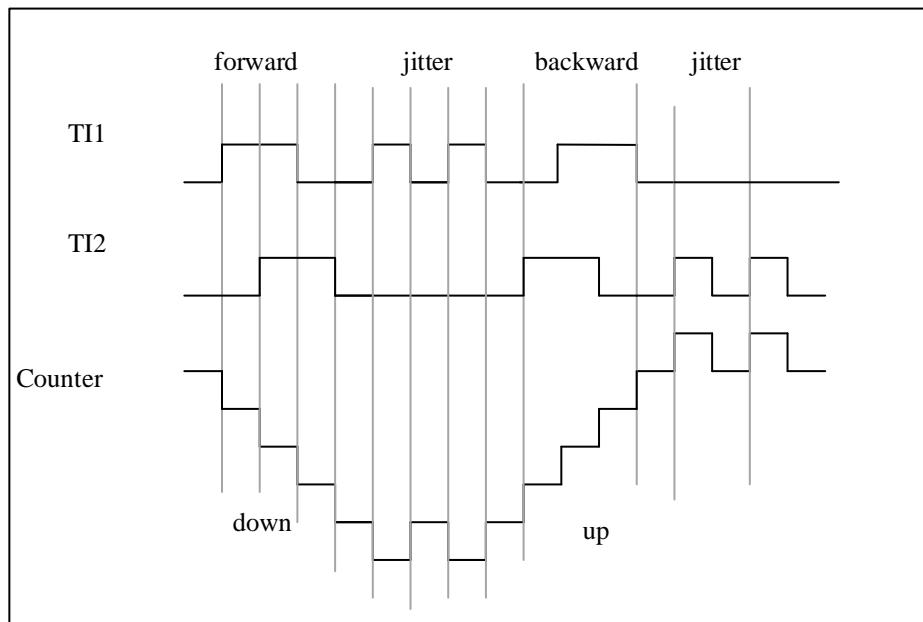
1. IC1FP1 is mapped to TI1 (TIMx\_CCMOD1.CC1SEL= '01'), IC1FP1 is not inverted (TIMx\_CCEN.CC1P= '0');
2. IC1FP2 is mapped to TI2 (TIMx\_CCMOD2.CC2SEL= '01'), IC2FP2 is not inverted (TIMx\_CCEN.CC2P= '0');
3. The input is valid on both rising and falling edges (TIMx\_SMCTRL.SMSEL = '011');
4. Enable counter TIMx\_CTRL1.CNTEN= '1';

Figure 9-33 Example of counter operation in encoder interface mode



The following figure shows the example of counter behavior when IC1FP1 polarity is inverted (CC1P= '1', other configurations are the same as above)

Figure 9-34 Encoder interface mode example with IC1FP1 polarity inverted



### 9.3.20 Interfacing with Hall sensor

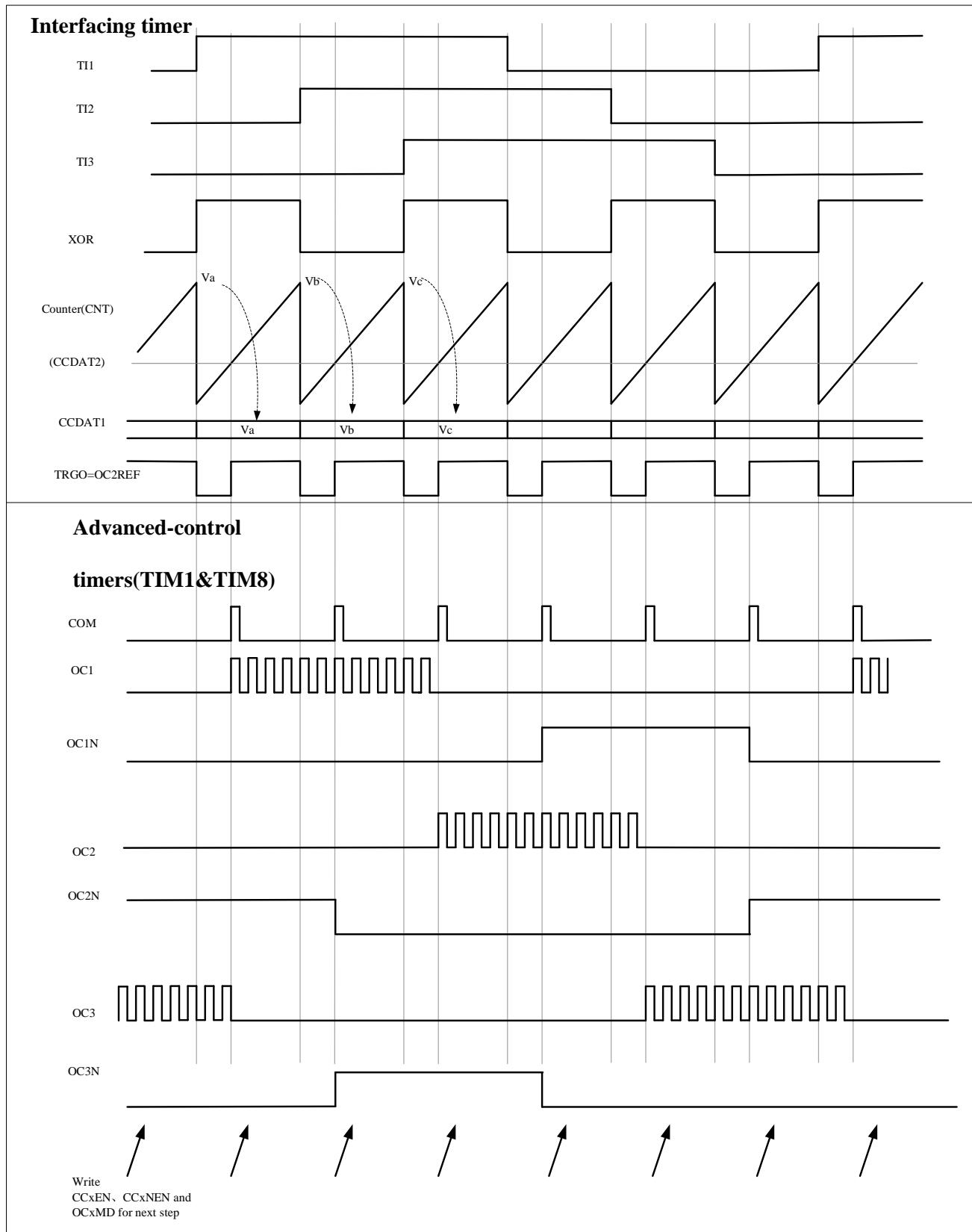
Connect the Hall sensor to the three input pins (CC1, CC2 and CC3) of the timer, and then select the XOR function to pass the inputs of TIMx\_CH1, TIMx\_CH2 and TIMx\_CH3 through the XOR gate as the output of TI1 to channel 1 for capture signal.

The timer needs to be configured as the reset mode in slave mode (TIMx\_SMCTRL.SMSEL=‘100’); the edge of the trigger select TI1 triggers TI1F\_ED (TIMx\_SMCTRL.TSEL=‘100’), any change in the Hall 3 inputs will trigger the counter to recount, so it is used as a time reference; the capture/compare channel 1 is configured to capture the TRC signal in capture mode (TIMx\_CCMOD1.CC1SEL=‘11’), which is used to calculate the two input time intervals, thereby reflecting the motor speed.

Select timer channel 2 to output pulses to the advanced timer to trigger the COM event of the advanced timer to update the control bits of the output PWM. The trigger selection of the advanced timer needs to select the corresponding internal trigger signal (TIMx\_SMCTRL.TSEL="ITRx"), the capture/compare preload control bit needs to be configured to support preload (TIMx\_CTRL2.CCPCTL=1) and support the rising edge of TRGI Trigger an update (TIMx\_CTRL2.CCUSEL=1).

This example is shown in the following figure.

Figure 9-35 Example of Hall sensor interface



## 9.4 TIMx register description(x=1, 8)

For abbreviations used in registers, see section 1.1

These peripheral registers can be operated as half word (16-bits) or one word (32-bits).

### 9.4.1 Register Overview

Table 9-2 Register overview

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18
000h	TIMx_CTRL1														
	Reset Value														
004h	TIMx_CTRL2														
	Reset Value														
008h	TIMx_SMCTRL														
	Reset Value														
00Ch	TIMx_DINTEN														
	Reset Value														
010h	TIMx_STS														
	Reset Value														
014h	TIMx_EVTGEN														
	Reset Value														
018h	TIMx_CCMD1 Output compare mode														
	Reset Value														
	TIMx_CCMD1 Input capture mode														
	Reset Value														
01Ch	TIMx_CCMD2 Output compare mode														
	Reset Value														
01Ch	TIMx_CCMD2 Input capture mode														
	Reset Value														

	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0													
020h	TIMx_CCEN																																													
	Reset Value																																													
024h	TIMx_CNT																																													
	Reset Value																																													
028h	TIMx_PSC																																													
	Reset Value																																													
02Ch	TIMx_AR																																													
	Reset Value																																													
030h	TIMx_REPCNT																																													
	Reset Value																																													
034h	TIMx_CCDAT1																																													
	Reset Value																																													
038h	TIMx_CCDAT2																																													
	Reset Value																																													
03Ch	TIMx_CCDAT3																																													
	Reset Value																																													
040h	TIMx_CCDAT4																																													
	Reset Value																																													
044h	TIMx_BKDT																																													
	Reset Value																																													
048h	TIMx_DCTRL																																													
	Reset Value																																													
04Ch	TIMx_DADDR																																													
	Reset Value																																													
054h	TIMx_CCMD3																																													
	Reset Value																																													
058h	TIMx_CCDAT5																																													
	Reset Value																																													
05Ch	TIMx_CCDAT6																																													
	Reset Value																																													

#### 9.4.2 Control register 1 (TIMx\_CTRL1)

Offset address: 0x00

Reset value: 0x0000 0000

31	Reserved																18	17	16
15	CLRSEL	Reserved	rw	C1SEL	IOM BKPN	rw	CLKD[1:0]	ARPEN	rw	CAMSEL[1:0]	DIR	rw	ONEPM	rw	UPRS	rw	UPDIS	CNTEN	
14																			
12																			
11																			
10																			
9																			
8																			
7																			
6																			
5																			
4																			
3																			
2																			
1																			
0																			

Bit field	Name	Description
31:18	Reserved	Reserved, the reset value must be maintained
17	PBKPen	PVD as BKP enable 0: Disable 1: Enable
16	LBKPen	LockUp as BKP enable 0: Disable 1: Enable
15	CLRSEL	OCxREF clear selection 0: Select the external OCxREF clear from ETR 1: Select the internal OCxREF clear from comparator
14:12	Reserved	Reserved, the reset value must be maintained
11	C1SEL	Channel 1 selection 0: Select external CH1 signal from IOM 1: Select internal CH1 signal from COMP
10	IOMBKPen	Enabling IOM as BKP 0: Enable 1: Disable
9:8	CLKD[1:0]	Clock division CLKD[1:0] indicates the division ratio between CK_INT (timer clock) and DTS (clock used for dead-time generator and digital filters (ETR, TIx)) 00: tDTS = tCK_INT 01: tDTS = 2 × tCK_INT 10: tDTS = 4 × tCK_INT 11: Reserved, do not use this configuration
7	ARPEN	ARPEN: Auto-reload preload enable 0: Shadow register disable for TIMx_AR register 1: Shadow register enable for TIMx_AR register
6:5	CAMSEL[1:0]	Center-aligned mode selection 00: Edge-aligned mode. TIMx_CTRL1.DIR specifies up-counting or down-counting. 01: Center-aligned mode 1. The counter counts in center-aligned mode, and the output compare interrupt flag bit is set to 1 when down-counting. 10: Center-aligned mode 2. The counter counts in center-aligned mode, and the output compare interrupt flag bit is set to 1 when up-counting. 11: Center-aligned mode 3. The counter counts in center-aligned mode, and the output compare interrupt flag bit is set to 1 when up-counting or down-counting. <i>Note: Switching from edge-aligned mode to center-aligned mode is not allowed when the counter is still enabled (TIMx_CTRL1.CNTEN = 1).</i>
4	DIR	Direction 0: Up-counting 1: Down-counting

Bit field	Name	Description
		<i>Note: This bit is read-only when the counter is configured in center-aligned mode or encoder mode.</i>
3	ONEPM	<p>One-pulse mode</p> <p>0: Disable one-pulse mode, the counter counts are not affected when an update event occurs.</p> <p>1: Enable one-pulse mode, the counter stops counting when the next update event occurs (clearing TIMx_CTRL1.CNTEN bit)</p>
2	UPRS	<p>Update request source</p> <p>This bit is used to select the UEV event sources by software.</p> <p>0: If update interrupt or DMA request is enabled, any of the following events will generate an update interrupt or DMA request:</p> <ul style="list-style-type: none"> <li>– Counter overflow/underflow</li> <li>– The TIMx_EVTGEN.UDGN bit is set</li> <li>– Update generation from the slave mode controller</li> </ul> <p>1: If update interrupt or DMA request is enabled, only counter overflow/underflow will generate update interrupt or DMA request</p>
1	UPDIS	<p>Update disable</p> <p>This bit is used to enable/disable the Update event (UEV) events generation by software.</p> <p>0: Enable UEV. UEV will be generated if one of following condition been fulfilled:</p> <ul style="list-style-type: none"> <li>– Counter overflow/underflow</li> <li>– The TIMx_EVTGEN.UDGN bit is set</li> <li>– Update generation from the slave mode controller</li> </ul> <p>Shadow registers will update with preload value.</p> <p>1: UEV disabled. No update event is generated, and the shadow registers (AR, PSC, and CCDDATx) keep their values. If the TIMx_EVTGEN.UDGN bit is set or a hardware reset is issued by the slave mode controller, the counter and prescaler are reinitialized.</p>
0	CNTEN	<p>Counter Enable</p> <p>0: Disable counter</p> <p>1: Enable counter</p> <p><i>Note: external clock, gating mode and encoder mode can only work after TIMx_CTRL1.CNTEN bit is set in the software. Trigger mode can automatically set TIMx_CTRL1.CNTEN bit by hardware.</i></p>

#### 9.4.3 Control register 2 (TIMx\_CTRL2)

Offset address: 0x04

Reset value: 0x0000 0000

31	Reserved												19	18	17	16
													OI6	Reserved	OI5	
15	14	13	12	11	10	9	8	7	6	4	3	rw	2	1	rw	
Reserved	OI4	OI3N	OI3	OI2N	OI2	OI1N	OI1	TI1SEL	MMSEL[2:0]	CCDSEL	CCUSEL	Reserved	CCPCTL			rw 0

rw      rw

Bit field	Name	Description
31:19	Reserved	Reserved, the reset value must be maintained
18	OI6	Output idle state 6 (OC6 output). See TIMx_CTRL2.OI1 bit.
17	Reserved	Reserved, the reset value must be maintained
16	OI5	Output idle state 5 (OC5 output). See TIMx_CTRL2.OI1 bit.
15	Reserved	Reserved, the reset value must be maintained
14	OI4	Output idle state 4 (OC4 output). See TIMx_CTRL2.OI1 bit.
13	OI3N	Output idle state 3 (OC3N output). See TIMx_CTRL2.OI1N bits.
12	OI3	Output idle state 3 (OC3 output). See TIMx_CTRL2.OI1 bit.
11	OI2N	Output idle state 2 (OC2N output). See TIMx_CTRL2.OI1N bits.
10	OI2	Output idle state 2 (OC2 output). See TIMx_CTRL2.OI1 bit.
9	OI1N	Output Idle state 1 (OC1N Output) 0: When TIMx_BKDT.MOEN = 0, after dead-time OC1N = 0 1: When TIMx_BKDT.MOEN = 0, after dead-time OC1N = 1
8	OI1	Output Idle state 1 0: When TIMx_BKDT.MOEN = 0, if OC1N is implemented, after dead-time OC1 = 0 1: When TIMx_BKDT.MOEN = 0, if OC1N is implemented, after dead-time OC1 = 1
7	TI1SEL	TI1 selection 0: TIMx_CH1 pin connected to TI1 input. 1: TIMx_CH1, TIMx_CH2, and TIMx_CH3 pins are XOR connected to the TI1 input.
6:4	MMSEL[2:0]	Master Mode Selection These 3 bits (TIMx_CTRL2.MMSEL [2:0]) are used to select the synchronization information (TRGO) sent to the slave timer in the master mode. Possible combinations are as follows: 000: Reset -When the TIMx_EVTGEN.UDGN is set or a reset is generated by the slave mode controller, a TRGO pulse occurs. And in the latter case, the signal on TRGO is delayed compared to the actual reset. 001: Enable - The TIMx_CTRL1.CNTEN bit is used as the trigger output (TRGO). Sometimes you need to start multiple timers at the same time or enable slave timer for a period of time. The counter enable signal is set when TIMx_CTRL1.CNTEN bit is set or the trigger input in gated mode is high. 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 description of the TIMx_SMCTRL.MSMD bit). 010: Update - The update event is selected as the trigger output (TRGO). For example, a master timer clock can be used as a slave timer prescaler.

Bit field	Name	Description
		011: Compare pulse - Triggers the output to send a positive pulse (TRGO) when the TIMx_STS.CC1ITF is to be set (even if it is already high), when a capture or a comparison succeeds. 100: Compare - OC1REF signal is used as the trigger output (TRGO). 101: Compare - OC2REF signal is used as the trigger output (TRGO). 110: Compare - OC3REF signal is used as the trigger output (TRGO). 111: Compare - OC4REF signal is used as the trigger output (TRGO).
3	CCDSEL	Capture/compare DMA selection 0: When a CCx event occurs, a DMA request for CCx is sent. 1: When an update event occurs, a DMA request for CCx is sent.
2	CCUSEL	Capture/compare control update selection 0: If TIMx_CTRL2.CCPCTL = 1, they can only be updated by setting CCUDGN bits 1: If TIMx_CTRL2.CCPCTL = 1, they can be updated by setting CCUDGN bits or a rising edge on TRGI. <i>Note: This bit only applied to channels with complementary outputs.</i>
1	Reserved	Reserved, the reset value must be maintained
0	CCPCTL	Capture/ Compare preloaded control 0: No preloading of CCxEN, CCxNEN and OCxMD bits occurs. 1: Preloading of CCxEN, CCxNEN and OCxMD bits occurs. they are updated only when a commutation event COM occurs (TIMx_EVTGEN.CCUDGN bit set or rising edge on TRGI depending on CCUSEL bit) <i>Note: This bit only applied to channels with complementary outputs.</i>

#### 9.4.4 Slave mode control register (TIMx\_SMCTRL)

Offset address: 0x08

Reset value: 0x0000

15	14	13	12	11	EXTF[3:0]	8	7	6	4	3	2	0
EXTP	EXCEN	EXTPS[1:0]			EXTF[3:0]	MSMD		TSEL[2:0]	Reserved		SMSEL[2:0]	
RW	RW	RW		RW		RW		RW			RW	

Bit field	Name	Description
15	EXTP	External trigger polarity This bit is used to select whether the trigger operation is to use ETR or the inversion of ETR. 0: ETR active at high level or rising edge. 1: ETR active at low level or falling edge.
14	EXCEN	External clock enable This bit is used to enable external clock mode 2, and the counter is driven by any active edge on the ETRF signal in this mode. 0: External clock mode 2 disable.

Bit field	Name	Description
		<p>1: External clock mode 2 enable.</p> <p><i>Note 1: When external clock mode 1 and external clock mode 2 are enabled at the same time, the input of the external clock is ETRF.</i></p> <p><i>Note 2: The following slave modes can be used simultaneously with external clock mode 2: reset mode, gated mode and trigger mode; However, TRGI cannot connect to ETRF (<math>TIMx\_SMCTRL.TSEL \neq '111'</math>).</i></p> <p><i>Note 3: Setting the <math>TIMx\_SMCTRL.EXCEN</math> bit has the same effect as selecting external clock mode 1 and connecting TRGI to ETRF (<math>TIMx\_SMCTRL.SMSEL = 111</math> and <math>TIMx\_SMCTRL.TSEL = 111</math>).</i></p>
13:12	EXTPS[1:0]	<p>External trigger prescaler</p> <p>The frequency of the external trigger signal ETRP must be at most 1/4 of TIMxCLK frequency. When a faster external clock is input, a prescaler can be used to reduce the frequency of ETRP.</p> <p>00: Prescaler disable</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>
11:8	EXTF[3:0]	<p>External trigger filter</p> <p>These bits are used to define the frequency at which the ETRP signal is sampled and the bandwidth of the ETRP digital filtering. In effect, the digital filter is an event counter that generates a validate output after consecutive N events are recorded.</p> <p>0000: No filter, sampling at f<sub>DTS</sub></p> <p>0001: f<sub>SAMPLING</sub> = f<sub>CLOCK</sub>, N = 2</p> <p>0010: f<sub>SAMPLING</sub> = f<sub>CLOCK</sub>, N = 4</p> <p>0011: f<sub>SAMPLING</sub> = f<sub>CLOCK</sub>, N = 8</p> <p>0100: f<sub>SAMPLING</sub> = f<sub>DTS</sub>/2, N = 6</p> <p>0101: f<sub>SAMPLING</sub> = f<sub>DTS</sub>/2, N = 8</p> <p>0110: f<sub>SAMPLING</sub> = f<sub>DTS</sub>/4, N = 6</p> <p>0111: f<sub>SAMPLING</sub> = f<sub>DTS</sub>/4, N = 8</p> <p>1000: f<sub>SAMPLING</sub> = f<sub>DTS</sub>/8, N = 6</p> <p>1001: f<sub>SAMPLING</sub> = f<sub>DTS</sub>/8, N = 8</p> <p>1010: f<sub>SAMPLING</sub> = f<sub>DTS</sub>/16, N = 5</p> <p>1011: f<sub>SAMPLING</sub> = f<sub>DTS</sub>/16, N = 6</p> <p>1100: f<sub>SAMPLING</sub> = f<sub>DTS</sub>/16, N = 8</p> <p>1101: f<sub>SAMPLING</sub> = f<sub>DTS</sub>/32, N = 5</p> <p>1110: f<sub>SAMPLING</sub> = f<sub>DTS</sub>/32, N = 6</p> <p>1111: f<sub>SAMPLING</sub> = f<sub>DTS</sub>/32, N = 8</p>
7	MSMD	<p>Master/ Slave mode</p> <p>0: No action</p> <p>1: Events on the trigger input (TRGI) are delayed to allow a perfect synchronization between the current timer (via TRGO) and its slaves. This is useful when several timers are required to be synchronized to a single external event.</p>

Bit field	Name	Description
6:4	TSEL[2:0]	<p>Trigger selection</p> <p>These 3 bits are used to select the trigger input of the synchronous counter.</p> <p>000: Internal trigger 0 (ITR0) 100: TI1 edge detector (TI1F_ED)      001: Internal trigger 1 (ITR1) 101: Filtered timer input 1 (TI1FP1)      010: Internal trigger 2 (ITR2) 110: Filtered timer input 2 (TI2FP2)      011: Internal trigger 3 (ITR3) 111: External triggered Input (ETRF)</p> <p>For more details on ITRx, see Table 9-3 below.</p> <p><i>Note: These bits must be changed only when not in use (e. g. TIMx_SMCTRL.SMSEL=000) to avoid false edge detection at the transition.</i></p>
3	Reserved	Reserved, the reset value must be maintained
2:0	SMSEL[2:0]	<p>Slave mode selection</p> <p>When an external signal is selected, the active edge of the trigger signal (TRGI) is linked to the selected external input polarity (see input control register and control register description)</p> <p>000: Disable slave mode. If TIMx_CTRL1.CNTEN = 1, the prescaler is driven directly by the internal clock.</p> <p>001: Encoder mode 1. According to the level of TI2FP2, the counter up-counting or down-counting on the edge of TI1FP1.</p> <p>010: Encoder mode 2. According to the level of TI1FP1, the counter up-counting or down-counting on the edge of TI2FP2.</p> <p>011: Encoder mode 3. According to the input level of another signal, the counter up-counting or down-counting on the edges of TI2FP1 and TI2FP2.</p> <p>100: Reset mode. On the rising edge of the selected trigger input (TRGI), the counter is reinitialized and the shadow register is updated.</p> <p>101: Gated mode. When the trigger input (TRGI) is high, the clock of the counter is enabled. Once the trigger input becomes low, the counter stops counting, but is not reset. In this mode, the start and stop of the counter are controlled.</p> <p>110: Trigger mode. When a rising edge occurs on the trigger input (TRGI), the counter is started but not reset. In this mode, only the start of the counter is controlled.</p> <p>111: External clock mode 1. The counter is clocked by the rising edge of the selected trigger input (TRGI).</p> <p><i>Note: Do not use gated mode if TI1F_ED is selected as the trigger input (TIMx_SMCTRL.TSEL=100). This is because TI1F_ED outputs a pulse for each TI1F transition, whereas gated mode checks the level of the triggered input.</i></p>

Table 9-3 TIMx internal trigger connection

Slave timer	ITR0 (TSEL = 000)	ITR1 (TSEL = 001)	ITR2 (TSEL = 010)	ITR3 (TSEL = 011)
<b>TIM1</b>	NA	NA	TIM3	NA
<b>TIM8</b>	TIM1	NA	NA	NA

## 9.4.5 DMA/Interrupt enable registers (TIMx\_DINTEN)

Offset address: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	TDEN	COMDEN	CC4DEN	CC3DEN	CC2DEN	CC1DEN	UDEN	BIEN	TIEN	COMIEN	CC4IEN	CC3IEN	CC2IEN	CC1IEN	UIEN

rw      rw

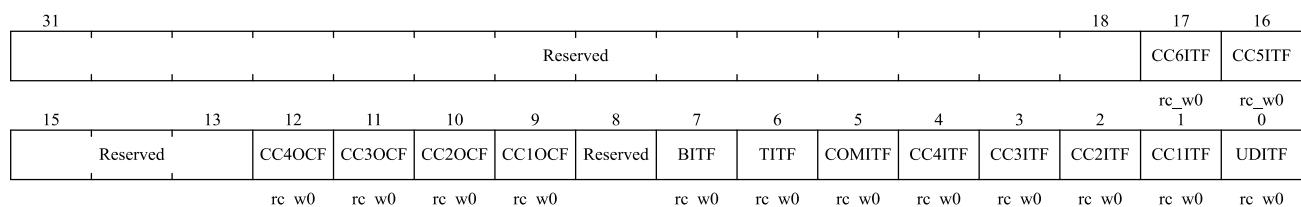
Bit field	Name	Description
15	Reserved	Reserved, the reset value must be maintained
14	TDEN	Trigger DMA request enable 0: Disable trigger DMA request 1: Enable trigger DMA request
13	COMDEN	COM DMA request enable 0: Disable COM DMA request 1: Enable COM DMA request
12	CC4DEN	Capture/Compare 4 DMA request enable 0: Disable capture/compare 4 DMA request 1: Enable capture/compare 4 DMA request
11	CC3DEN	Capture/Compare 3 DMA request enable 0: Disable capture/compare 3 DMA request 1: Enable capture/compare 3 DMA request
10	CC2DEN	Capture/Compare 2 DMA request enable 0: Disable capture/compare 2 DMA request 1: Enable capture/compare 2 DMA request
9	CC1DEN	Capture/Compare 1 DMA request enable 0: Disable capture/compare 1 DMA request 1: Enable capture/compare 1 DMA request
8	UDEN	Update DMA request enable 0: Disable update DMA request 1: Enable update DMA request
7	BIEN	Break interrupt enable 0: Disable break interrupt 1: Enable break interrupt
6	TIEN	Trigger interrupt enable 0: Disable trigger interrupt 1: Enable trigger interrupt
5	COMIEN	COM interrupt enable 0: Disable COM interrupt 1: Enable COM interrupt

Bit field	Name	Description
4	CC4IEN	Capture/Compare 4 interrupt enable 0: Disable capture/compare 4 interrupt 1: Enable capture/compare 4 interrupt
3	CC3IEN	Capture/Compare 3 interrupt enable 0: Disable capture/compare 3 interrupt 1: Enable capture/compare 3 interrupts
2	CC2IEN	Capture/Compare 2 interrupt enable 0: Disable capture/compare 2 interrupt 1: Enables capture/compare 2 interrupts
1	CC1IEN	Capture/Compare 1 interrupt enable 0: Disable capture/compare 1 interrupt 1: Enables capture/comparing 1 interrupt
0	UIEN	Update interrupt enable 0: Disable update interrupt 1: Enables update interrupt

#### 9.4.6 Status registers (TIMx\_STS)

Offset address: 0x10

Reset value: 0x0000 0000



Bit field	Name	Description
31: 18	Reserved	Reserved, the reset value must be maintained
17	CC6ITF	Capture/Compare 6 interrupt flag See TIMx_STS.CC1ITF description.
16	CC5ITF	Capture/Compare 5 interrupt flag See TIMx_STS.CC1ITF description.
15: 13	Reserved	Reserved, the reset value must be maintained
12	CC4OCF	Capture/Compare 4 overcapture flag See TIMx_STS.CC1OCF description.
11	CC3OCF	Capture/Compare 3 overcapture flag See TIMx_STS.CC1OCF description.
10	CC2OCF	Capture/Compare 2 overcapture flags See TIMx_STS.CC1OCF description.

Bit field	Name	Description
9	CC1OCF	<p>Capture/Compare 1 overcapture flag</p> <p>This bit is set by hardware only when the corresponding channel is configured in input capture mode. Cleared by software writing 0.</p> <p>0: No overcapture occurred</p> <p>1: TIMx_STS.CC1ITF was already set when the value of the counter has been captured in the TIMx_CCDAT1 register.</p>
8	Reserved	Reserved, the reset value must be maintained
7	BITF	<p>Break interrupt flag</p> <p>This bit is set by hardware once the brake input is active. This bit is cleared by software when the brake input becomes inactive.</p> <p>0: No break event occurred</p> <p>1: An active level has been detected</p>
6	TITF	<p>Trigger interrupt flag</p> <p>This bit is set by hardware when an active edge is detected on the TRGI input when the slave mode controller is in a mode other than gated. This bit is set by hardware when any edge in gated mode is detected. This bit is cleared by software.</p> <p>0: No trigger event occurred</p> <p>1: Trigger interrupt occurred</p>
5	COMITF	<p>COM interrupt flag</p> <p>This bit is set by hardware once a COM event is generated (when TIMx_CCEN.CCxEN, TIMx_CCEN.CCxNEN, TIMx_CCMOD1.OCxMD have been updated). This bit is cleared by software.</p> <p>0: No COM event occurred</p> <p>1: COM interrupt pending</p>
4	CC4ITF	<p>Capture/Compare 4 interrupt flag</p> <p>See TIMx_STS.CC1ITF description.</p>
3	CC3ITF	<p>Capture/Compare 3 interrupt flag</p> <p>See TIMx_STS.CC1ITF description.</p>
2	CC2ITF	<p>Capture/Compare 2 interrupt flag</p> <p>See TIMx_STS.CC1ITF description.</p>
1	CC1ITF	<p>Capture/Compare 1 interrupt flag</p> <p><b>When the corresponding channel of CC1 is in output mode:</b></p> <p>Except in center-aligned mode, this bit is set by hardware when the counter value is the same as the compare value (see TIMx_CTRL1.CAMSEL bit description). This bit is cleared by software.</p> <p>0: No match occurred.</p> <p>1: The value of TIMx_CNT is the same as the value of TIMx_CCDAT1.</p> <p>When the value of TIMx_CCDAT1 is greater than the value of TIMx_AR, the TIMx_STS.CC1ITF bit will go high if the counter overflows (in up-counting and up/down-counting modes) and underflows in down-counting mode.</p> <p><b>When the corresponding channel of CC1 is in input mode:</b></p>

Bit field	Name	Description
		<p>This bit is set by hardware when the capture event occurs. This bit is cleared by software or by reading TIMx_CCDAT1.</p> <p>0: No input capture occurred.</p> <p>1: Input capture occurred. Counter value has captured in the TIMx_CCDAT1. An edge with the same polarity as selected has been detected on IC1.</p>
0	UDITF	<p>Update interrupt flag</p> <p>This bit is set by hardware when an update event occurs under the following conditions:</p> <ul style="list-style-type: none"> <li>– When TIMx_CTRL1.UPDIS = 0, and repeat counter value overflow or underflow (An update event is generated when the repeat counter equals 0).</li> <li>– When TIMx_CTRL1.UPRS = 0, TIMx_CTRL1.UPDIS = 0, and set the TIMx_EVTGEN.UDGN bit by software to reinitialize the CNT.</li> <li>– When TIMx_CTRL1.UPRS = 0, TIMx_CTRL1.UPDIS = 0, and the counter CNT is reinitialized by the trigger event. (See TIMx_SMCTRL Register description)</li> </ul> <p>This bit is cleared by software.</p> <p>0: No update event occurred</p> <p>1: Update interrupt occurred</p>

#### 9.4.7 Event generation registers (TIMx\_EVTGEN)

Offset address: 0x14

Reset values: 0x0000

15	Reserved	8	7	6	5	4	3	2	1	0
			BGN	TGN	CCUDGN	CC4GN	CC3GN	CC2GN	CC1GN	UDGN

Bit field	Name	Description
15: 8	Reserved	Reserved, the reset value must be maintained
7	BGN	<p>Break generation</p> <p>This bit can generate a brake event when set by software. And at this time TIMx_BKDT.MOEN = 0, TIMx_STS.BITF = 1, if the corresponding interrupt and DMA are enabled, the corresponding interrupt and DMA will be generated. This bit is automatically cleared by hardware.</p> <p>0: No action</p> <p>1: Generated a break event</p>
6	TGN	<p>Trigger generation</p> <p>This bit can generate a trigger event when set by software. And at this time TIMx_STS.TITF = 1, if the corresponding interrupt and DMA are enabled, the corresponding interrupt and DMA will be generated. This bit is automatically cleared by hardware.</p> <p>0: No action</p> <p>1: Generated a trigger event</p>

Bit field	Name	Description
5	CCUDGN	<p>Capture/Compare control update generation</p> <p>This bit is set by software. And if TIMx_CTRL2.CCPCTL = 1 at this time, the CCxEN, CCxNEN and OCxMD bits are allowed to be updated. This bit is automatically cleared by hardware.</p> <p>0: No action 1: Generated a COM event</p> <p><i>Note: This bit is only valid for channels with complementary outputs.</i></p>
4	CC4GN	<p>Capture/Compare 4 generation</p> <p>See TIMx_EVTGEN.CC1GN description.</p>
3	CC3GN	<p>Capture/Compare 3 generation</p> <p>See TIMx_EVTGEN.CC1GN description.</p>
2	CC2GN	<p>Capture/Compare 2 generation</p> <p>See TIMx_EVTGEN.CC1GN description.</p>
1	CC1GN	<p>Capture/Compare 1 generation</p> <p>This bit can generate a capture/compare event when set by software. This bit is automatically cleared by hardware.</p> <p>When the corresponding channel of CC1 is in output mode: The TIMx_STS.CC1ITF flag will be pulled high, if the corresponding interrupt and DMA are enabled, the corresponding interrupt and DMA will be generated.</p> <p>When the corresponding channel of CC1 is in input mode: TIMx_CCDAT1 will capture the current counter value, and the TIMx_STS.CC1ITF flag will be pulled high, if the corresponding interrupt and DMA are enabled, the corresponding interrupt and DMA will be generated. If The TIMx_STS.CC1ITF is already pulled high, pull TIMx_STS.CC1OCF high.</p> <p>0: No action 1: Generated a CC1 capture/compare event</p>
0	UDGN	<p>Update generation</p> <p>This bit can generate an update event when set by software. And at this time the counter will be reinitialized, the prescaler counter will be cleared, the counter will be cleared in center-aligned or up-counting mode, but take TIMx_AR in down-counting mode the value of the register. This bit is automatically cleared by hardware.</p> <p>0: No action 1: Generated an update event</p>

#### 9.4.8 Capture/compare mode register 1 (TIMx\_CCMOD1)

Offset address: 0x18

Reset value: 0x0000

Channels can be used for input (capture mode) or output (compare mode), and the direction of the channel is defined by the corresponding CCxSEL bit. The other bits of the register act differently in input and output modes. OCx describes the function of a channel in output mode, ICx describes the function of a channel in input mode. Hence,

please note that the same bit can have different meanings for output mode and for input mode.

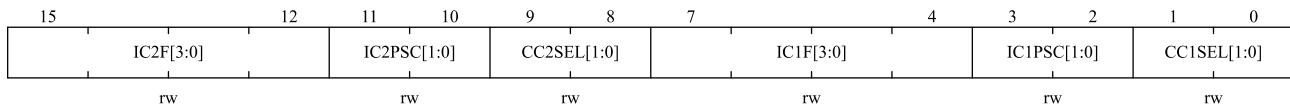
Output compare mode:

15	14	12	11	10	9	8	7	6	4	3	2	1	0
OC2CEN	OC2M[2:0]	OC2PEN	OC2FEN	CC2SEL[1:0]	OC1CEN		OC1M[2:0]	OC1PEN	OC1FEN	CC1SEL[1:0]			
RW	RW	RW	RW	RW	RW		RW	RW	RW	FW	FW		RW

Bit field	Name	Description
15	OC2CEN	Output Compare 2 clear enable
14:12	OC2MD[2:0]	Output Compare 2 mode
11	OC2PEN	Output Compare 2 preload enable
10	OC2FEN	Output Compare 2 fast enable
9:8	CC2SEL[1:0]	<p>Capture/compare 2 selection</p> <p>These bits are used to select the input/output and input mapping of the channel</p> <p>00: CC2 channel is configured as output</p> <p>01: CC2 channel is configured as input, IC2 is mapped on TI2</p> <p>10: CC2 channel is configured as input, IC2 is mapped on TI1</p> <p>11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is only active when the internal trigger input is selected by TIMx_SMCTRL.TSEL.</p> <p><i>Note: CC2SEL is writable only when the channel is off (TIMx_CCEN.CC2EN = 0).</i></p>
7	OC1CEN	<p>Output Compare 1 clear enable</p> <p>0: OC1REF is not affected by ETRF input level</p> <p>1: OC1REF is cleared immediately when the ETRF input level is detected as high</p>
6:4	OC1MD[2:0]	<p>Output Compare 1 mode</p> <p>These bits are used to manage the output reference signal OC1REF, which determines the values of OC1 and OC1N, and is valid at high levels, while the active levels of OC1 and OC1N depend on the TIMx_CCEN.CC1P and TIMx_CCEN.CC1NP bits.</p> <p>000: Frozen. Comparison between TIMx_CCDAT1 register and counter TIMx_CNT has no effect on OC1REF signal.</p> <p>001: Set channel 1 to the active level on match. When TIMx_CCDAT1 = TIMx_CNT, OC1REF signal will be forced high.</p> <p>010: Set channel 1 as inactive level on match. When TIMx_CCDAT1 = TIMx_CNT, OC1REF signal will be forced low.</p> <p>011: Toggle. When TIMx_CCDAT1 = TIMx_CNT, OC1REF signal will be toggled.</p> <p>100: Force to inactive level. OC1REF signal is forced low.</p> <p>101: Force to active level. OC1REF signal is forced high.</p> <p>110: PWM mode 1 - In up-counting mode, if TIMx_CNT &lt; TIMx_CCDAT1, OC1REF signal of channel 1 is high, otherwise it is low. In down-counting mode, if TIMx_CNT &gt; TIMx_CCDAT1, OC1REF signal of channel 1 is low, otherwise it is high.</p> <p>111: PWM mode 2 - In up-counting mode, if TIMx_CNT &lt; TIMx_CCDAT1, OC1REF signal of channel 1 is low, otherwise it is high. In down-counting mode, if TIMx_CNT &gt; TIMx_CCDAT1, OC1REF signal of channel 1 is high, otherwise it is low.</p> <p><i>Note 1: In PWM mode 1 or PWM mode 2, the OC1REF level changes only when the</i></p>

Bit field	Name	Description
		<i>comparison result changes or when the output compare mode is switched from frozen mode to PWM mode.</i>
3	OC1PEN	<p>Output Compare 1 preload enable</p> <p>0: Disable preload function of TIMx_CCDAT1 register. Supports write operations to TIMx_CCDAT1 register at any time, and the written value is effective immediately.</p> <p>1: Enable preload function of TIMx_CCDAT1 register. Only read and write operations to preload registers. When an update event occurs, the value of TIMx_CCDAT1 is loaded into the active register.</p> <p><i>Note 1: Only when TIMx_CTRL1.ONEPM = 1 (In one-pulse mode), PWM mode can be used without verifying the preload register, otherwise no other behavior can be predicted.</i></p>
2	OC1FEN	<p>Output Compare 1 fast enable</p> <p>This bit is used to speed up the response of the CC output to the trigger input event.</p> <p>0: CC1 behaves normally depending on the counter and CCDAT1 values, even if the trigger is ON. The minimum delay for activating CC1 output when an edge occurs on the trigger input is 5 clock cycles.</p> <p>1: An active edge of the trigger input acts like a comparison match on CC1 output. Therefore, OC is set to the comparison level regardless of the comparison result. The delay time for sampling the trigger input and activating the CC1 output is reduced to 3 clock cycles.</p> <p>OCxFEN only works if the channel is configured in PWM1 or PWM2 mode.</p>
1: 0	CC1SEL[1:0]	<p>Capture/compare 1 selection</p> <p>These bits are used to select the input/output and input mapping of the channel</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 channels are configured as inputs and IC1 is mapped to TRC. This mode is only active when the internal trigger input is selected by TIMx_SMCTRL.TSEL.</p> <p><i>Note: CC1SEL is writable only when the channel is off (TIMx_CCEN.CC1EN = 0).</i></p>

#### Input capture mode:



Bit field	Name	Description
15:12	IC2F[3:0]	Input Capture 2 Filter
11:10	IC2PSC[1:0]	Input Capture 2 Prescaler

Bit field	Name	Description
9:8	CC2SEL[1:0]	<p>Capture/Compare 2 selection</p> <p>These bits are used to select the input/output and input mapping of the channel</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 only active when the internal trigger input is selected by TIMx_SMCTRL.TSEL.</li> </ul> <p><i>Note: CC2SEL is writable only when the channel is off (TIMx_CCEN.CC2EN = 0).</i></p>
7:4	IC1F[3:0]	<p>Input Capture 1 filter</p> <p>These bits are used to define sampling frequency of TI1 input and the length of digital filter. The digital filter is an event counter that generates an output transition after N events are recorded.</p> <ul style="list-style-type: none"> <li>0000: No filter, sampling at f<sub>DTS</sub> frequency</li> <li>0001: f<sub>SAMPLING</sub> = f<sub>CLOCK</sub>, N = 2</li> <li>0010: f<sub>SAMPLING</sub> = f<sub>CLOCK</sub>, N = 4</li> <li>0011: f<sub>SAMPLING</sub> = f<sub>CLOCK</sub>, N = 8</li> <li>0100: f<sub>SAMPLING</sub> = f<sub>DTS</sub>/2, N = 6</li> <li>0101: f<sub>SAMPLING</sub> = f<sub>DTS</sub>/2, N = 8</li> <li>0110: f<sub>SAMPLING</sub> = f<sub>DTS</sub>/4, N = 6</li> <li>0111: f<sub>SAMPLING</sub> = f<sub>DTS</sub>/4, N = 8</li> <li>1000: f<sub>SAMPLING</sub> = f<sub>DTS</sub>/8, N = 6</li> <li>1001: f<sub>SAMPLING</sub> = f<sub>DTS</sub>/8, N = 8</li> <li>1010: f<sub>SAMPLING</sub> = f<sub>DTS</sub>/16, N = 5</li> <li>1011: f<sub>SAMPLING</sub> = f<sub>DTS</sub>/16, N = 6</li> <li>1100: f<sub>SAMPLING</sub> = f<sub>DTS</sub>/16, N = 8</li> <li>1101: f<sub>SAMPLING</sub> = f<sub>DTS</sub>/32, N = 5</li> <li>1110: f<sub>SAMPLING</sub> = f<sub>DTS</sub>/32, N = 6</li> <li>1111: f<sub>SAMPLING</sub> = f<sub>DTS</sub>/32, N = 8</li> </ul>
3:2	IC1PSC[1:0]	<p>Input Capture 1 prescaler</p> <p>These bits are used to select the ratio of the prescaler for IC1 (CC1 input).</p> <p>When TIMx_CCEN.CC1EN = 0, the prescaler will be reset.</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	CC1SEL[1:0]	<p>Capture/Compare 1 selection</p> <p>These bits are used to select the input/output and input mapping of the channel</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 to TRC. This mode is only active when the internal trigger input is selected by TIMx_SMCTRL.TSEL.</li> </ul>

Bit field	Name	Description
<i>Note: CCISEL is writable only when the channel is off (TIMx_CCEN.CC1EN = 0).</i>		

### 9.4.9 Capture/compare mode register 2 (TIMx\_CCMOD2)

Offset address: 0x1C

Reset value: 0x0000

See the description of the CCMOD1 register above

Output comparison mode:

15	14	12	11	10	9	8	7	6	4	3	2	1	0
OC4CEN	OC4MD[2:0]	OC4PEN	OC4FEN	CC4SEL[1:0]	OC3CEN	OC3MD[2:0]	OC3PEN	OC3FEN	CC3SEL[1:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit field	Name	Description
15	OC4CEN	Output compare 4 clear enable
14:12	OC4MD[2:0]	Output compare 4 mode
11	OC4PEN	Output compare 4 preload enable
10	OC4FEN	Output compare 4 fast enable
9:8	CC4SEL[1:0]	<p>Capture/Compare 4 selection</p> <p>These bits are used to select the input/output and input mapping of the channel</p> <p>00: CC4 channel is configured as output</p> <p>01: CC4 channel is configured as input, IC4 is mapped on TI4</p> <p>10: CC4 channel is configured as input, IC4 is mapped on TI3</p> <p>11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is only active when the internal trigger input is selected by TIMx_SMCTRL.TSEL.</p> <p><i>Note: CC4SEL is writable only when the channel is off (TIMx_CCEN.CC4EN = 0).</i></p>
7	OC3CEN	Output compare 3 clear enable
6:4	OC3MD[2:0]	Output compare 3 mode
3	OC3PEN	Output compare 3 preload enable
2	OC3FEN	Output compare 3 fast enable
1:0	CC3SEL[1:0]	<p>Capture/Compare 3 selection</p> <p>These bits are used to select the input/output and input mapping of the channel</p> <p>00: CC3 channel is configured as output</p> <p>01: CC3 channel is configured as input, IC3 is mapped to TI3</p> <p>10: CC3 channel is configured as input, IC3 is mapped on TI4</p> <p>11: CC3 channel is configured as input, IC3 is mapped to TRC. This mode is only active when the internal trigger input is selected by TIMx_SMCTRL.TSEL.</p> <p><i>Note: CC3SEL is writable only when the channel is off (TIMx_CCEN.CC3EN = 0).</i></p>

Input capture mode:

15	IC4F[3:0]	rw	12	IC4PSC[1:0]	rw	11	10	9	8	7	rw	IC3F[3:0]	rw	4	3	2	1	0
----	-----------	----	----	-------------	----	----	----	---	---	---	----	-----------	----	---	---	---	---	---

Bit field	Name	Description
15:12	IC4F[3:0]	Input Capture 4 filter
11:10	IC4PSC[1:0]	Input Capture 4 Prescaler
9:8	CC4SEL[1:0]	<p>Capture/Compare 4 selection</p> <p>These bits are used to select the input/output and input mapping of the channel</p> <p>00: CC4 channel is configured as output</p> <p>01: CC4 channel is configured as input, IC4 is mapped on TI4</p> <p>10: CC4 channel is configured as input, IC4 is mapped on TI3</p> <p>11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is only active when the internal trigger input is selected by TIMx_SMCTRL.TSEL.</p> <p><i>Note: CC4SEL is writable only when the channel is off (TIMx_CCEN.CC4EN = 0).</i></p>
7:4	IC3F[3:0]	Input Capture 3 filter
3:2	IC3PSC[1:0]	Input Capture 3 Prescaler
1:0	CC3SEL[1:0]	<p>Capture/compare 3 selection</p> <p>These bits are used to select the input/output and input mapping of the channel</p> <p>00: CC3 channel is configured as output</p> <p>01: CC3 channel is configured as input, IC3 is mapped to TI3</p> <p>10: CC3 channel is configured as input, IC3 is mapped on TI4</p> <p>11: CC3 channel is configured as input, IC3 is mapped to TRC. This mode is only active when the internal trigger input is selected by TIMx_SMCTRL.TSEL.</p> <p><i>Note: CC3SEL is writable only when the channel is off (TIMx_CCEN.CC3EN = 0).</i></p>

#### 9.4.10 Capture/compare enable registers (TIMx\_CCEN)

Offset address: 0x20

Reset value: 0x0000 0000

31	Reserved												22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	rw	rw	rw	CC6P	CC6EN	Reserved	CC5P	CC5EN		
									5	4	3	2	rw	rw	rw	rw	rw	rw	

Bit field	Name	Description
31:22	Reserved	Reserved, the reset value must be maintained
21	CC6P	<p>Capture/Compare 6 output polarity</p> <p>See TIMx_CCEN.CC1P description.</p>

Bit field	Name	Description
20	CC6EN	Capture/Compare 6 output enable See TIMx_CCEN.CC1EN description.
19: 18	Reserved	Reserved, the reset value must be maintained
17	CC5P	Capture/Compare 5 output polarity See TIMx_CCEN.CC1P description.
16	CC5EN	Capture/Compare 5 output enable See TIMx_CCEN.CC1EN description.
15:14	Reserved	Reserved, the reset value must be maintained
13	CC4P	Capture/Compare 4 output polarity See TIMx_CCEN.CC1P description.
12	CC4EN	Capture/Compare 4 output enable See TIMx_CCEN.CC1EN description.
11	CC3NP	Capture/Compare 3 Complementary output polarity See TIMx_CCEN.CC1NP description.
10	CC3NEN	Capture/Compare 3 complementary output enable See TIMx_CCEN.CC1NEN description.
9	CC3P	Capture/Compare 3 output polarity See TIMx_CCEN.CC1P description.
8	CC3EN	Capture/Compare 3 output enable See TIMx_CCEN.CC1EN description.
7	CC2NP	Capture/Compare 2 complementary output polarity See TIMx_CCEN.CC1NP description.
6	CC2NEN	Capture/Compare 2 complementary output enable See TIMx_CCEN.CC1NEN description.
5	CC2P	Capture/Compare 2 output polarity See TIMx_CCEN.CC1P description.
4	CC2EN	Capture/Compare 2 output enable See TIMx_CCEN.CC1EN description.
3	CC1NP	Capture/Compare 1 complementary output polarity 0: OC1N active high 1: OC1N active low
2	CC1NEN	Capture/Compare 1 complementary output enable 0: Disable - Disable output OC1N signal. The level of OC1N depends on the value of these bits TIMx_BKDT.MOEN, TIMx_BKDT.OSSI, TIMx_BKDT.OSSR, TIMx_CTRL2.OI1, TIMx_CTRL2.OI1N and TIMx_CCEN.CC1EN. 1: Enable - Enable output OC1N signal. The level of OC1N depends on the value of these bits TIMx_BKDT.MOEN, TIMx_BKDT.OSSI, TIMx_BKDT.OSSR, TIMx_CTRL2.OI1, TIMx_CTRL2.OI1N and TIMx_CCEN.CC1EN.

Bit field	Name	Description
1	CC1P	<p>Capture/Compare 1 output polarity</p> <p>When the corresponding channel of CC1 is in output mode:</p> <p>0: OC1 active high 1: OC1 active low</p> <p>When the corresponding channel of CC1 is in input mode:</p> <p>At this time, this bit is used to select whether IC1 or the inverse signal of IC1 is used as the trigger or capture signal.</p> <p>0: non-inverted: Capture action occurs when IC1 generates a rising edge. When used as external trigger, IC1 is non-inverted. 1: inverted: Capture action occurs when IC1 generates a falling edge. When used as external trigger, IC1 is inverted.</p> <p><i>Note: If TIMx_BKDT.LCKCFG = 3 or 2, these bits cannot be modified.</i></p>
0	CC1EN	<p>Capture/Compare 1 output enable</p> <p>When the corresponding channel of CC1 is in output mode:</p> <p>0: Disable - Disable output OC1 signal. The level of OC1 depends on the value of these bits TIMx_BKDT.MOEN, TIMx_BKDT.OSSI, TIMx_BKDT.OSSR, TIMx_CTRL2.OI1, TIMx_CTRL2.OI1N and TIMx_CCEN.CC1NEN.</p> <p>1: Enable - Enable output OC1 signal. The level of OC1N depends on the value of these bits TIMx_BKDT.MOEN, TIMx_BKDT.OSSI, TIMx_BKDT.OSSR, TIMx_CTRL2.OI1, TIMx_CTRL2.OI1N and TIMx_CCEN.CC1NEN.</p> <p>When the corresponding channel of CC1 is in input mode:</p> <p>At this time, this bit is used to disable/enable the capture function.</p> <p>0: Disable capture 1: Enable capture</p>

Table 9-4 Output control bits of complementary OCx and OCxN channels with break function

Control bits					Output state <sup>1)</sup>	
MOEN	OSSI	OSSR	CCxEN	CCxNEN	OCx Output state	OCxN Output state
1	X	0	0	0	Output disabled □ not driven by timer □ OCx=0,OCx_EN=0	Output disabled □ not driven by timer □ OCxN=0,OCxN_EN=0
		0	0	1	Output disabled □ not driven by 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 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 timer □ OCx=CCxP,OCx_EN=0	Output disabled □ not driven by timer □ OCxN=CCxNP,OCxN_EN=0
		1	0	1	Off-state (Output enabled with inactive	OCxREF + polarity,

Control bits					Output state <sup>1)</sup>	
MOEN	OSSI	OSSR	CCxEN	CCxNEN	OCx Output state	OCxN Output state
1	1	1	0		state) OCx=CCxP, OCx_EN=1	OCxN= OCxREF xor CCxNP, OCxN_EN=1
					OCxREF + polarity, OCx= OCxREF xor CCxP, OCx_EN=1	Off-state (Output enabled with inactive state) OCxN=CCxNP, OCxN_EN=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 timer □	
	0		0	1	Asynchronously: OCx=CCxP, OCx_EN=0, OCxN=CCxNP, OCxN_EN=0 □	
	0		1	0	Then if the clock is present: OCx=OIx and OCxN=OIxN after a dead-time, when (CCxP ^ OIx) ^ (CCxNP^OIxN)! = 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=OIx and OCxN=OIxN after a dead-time, when (CCxP ^ OIx) ^ (CCxNP^OIxN)! = 0	
	1		1	1		

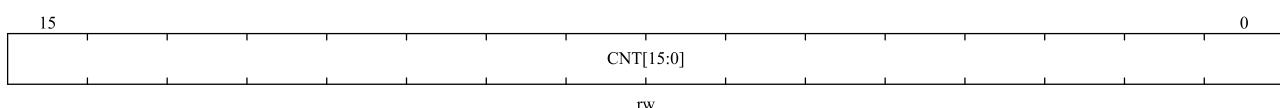
1. If both outputs of a channel are not used (CCxEN = CCxNEN = 0), OIx, OIxN, CCxP and CCxNP must all be cleared.

*Note: The status of external I/O pins connected to complementary OCx and OCxN channels depends on the OCx and OCxN channel states and GPIO and AFIO registers.*

#### 9.4.11 Counters (TIMx\_CNT)

Offset address: 0x24

Reset value: 0x0000

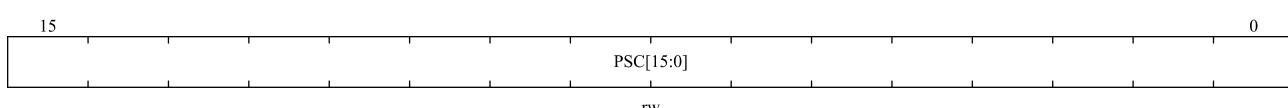


Bit field	Name	Description
15:0	CNT[15:0]	Counter value

#### 9.4.12 Prescaler (TIMx\_PSC)

Offset address: 0x28

Reset value: 0x0000

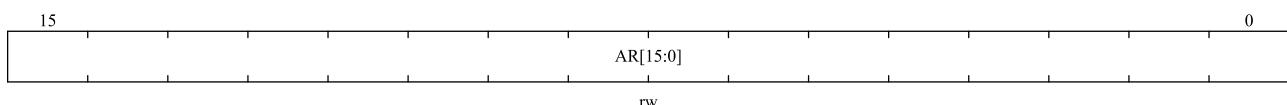


Bit field	Name	Description
15:0	PSC[15:0]	<p>Prescaler value</p> <p>Counter clock <math>f_{CK\_CNT} = f_{CK\_PSC} / (PSC [15:0] + 1)</math>.</p> <p>Each time an update event occurs, the PSC value is loaded into the active prescaler register.</p>

#### 9.4.13 Auto-reload register (TIMx\_AR)

Offset address: 0x2C

Reset values: 0xFFFF

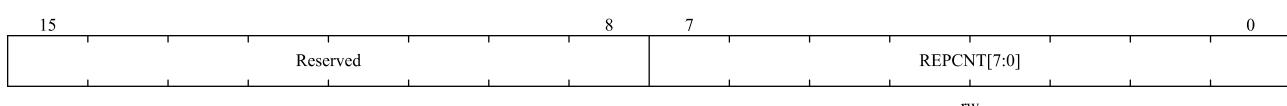


Bit field	Name	Description
15:0	AR[15:0]	<p>Auto-reload value</p> <p>These bits define the value that will be loaded into the actual auto-reload register.</p> <p>See Section 9.3.1 for more details.</p> <p>When the TIMx_AR.AR [15:0] value is null, the counter does not work.</p>

#### 9.4.14 Repeat count registers (TIMx\_REPCNT)

Offset address: 0x30

Reset value: 0x0000

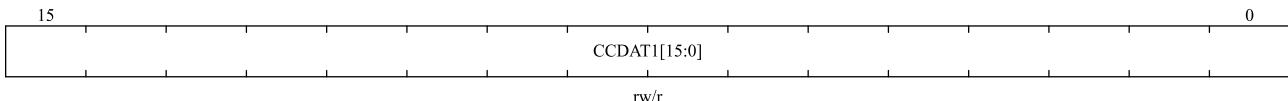


Bit field	Name	Description
15:8	Reserved	Reserved, the reset value must be maintained
7:0	REPCNT[7:0]	<p>Repetition counter value</p> <p>Repetition counter is used to generate the update event or update the timer registers only after a given number (<math>N+1</math>) cycles of the counter, where <math>N</math> is the value of TIMx_REPCNT.REPCNT .</p> <p>The repetition counter is decremented at each counter overflow in up-counting mode, at each counter underflow in down-counting mode or at each counter overflow and at each counter underflow in center-aligned mode. Setting the TIMx_EVTGEN.UDGN bit will reload the content of TIMx_REPCNT.REPCNT and generate an update event.</p>

### 9.4.15 Capture/compare register 1 (TIMx\_CC DAT1)

Offset address: 0x34

Reset value: 0x0000

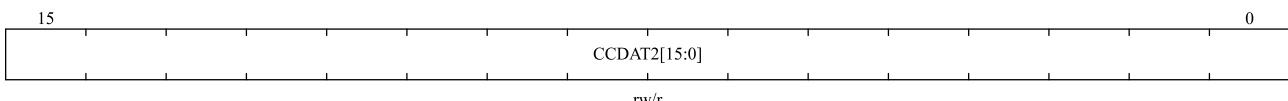


Bit field	Name	Description
15:0	CCDAT1[15:0]	<p>Capture/Compare 1 value</p> <ul style="list-style-type: none"> <li>■ CC1 channel is configured as output: CCDAT1 contains the value to be compared to the counter TIMx_CNT, signaling on the OC1 output.</li> <li>If the preload feature is not selected in TIMx_CCMOD1.OC1PEN bit, the written value is immediately transferred to the active register. Otherwise, this preloaded value is transferred to the active register only when an update event occurs.</li> <li>■ CC1 channel is configured as input: CCDAT1 contains the counter value transferred by the last input capture 1 event (IC1).</li> <li>When configured as input mode, register CCDAT1 and CCDDAT1 are only readable.</li> <li>When configured as output mode, register CCDAT1 and CCDDAT1 are readable and writable.</li> </ul>

### 9.4.16 Capture/compare register 2 (TIMx\_CC DAT2)

Offset address: 0x38

Reset value: 0x0000

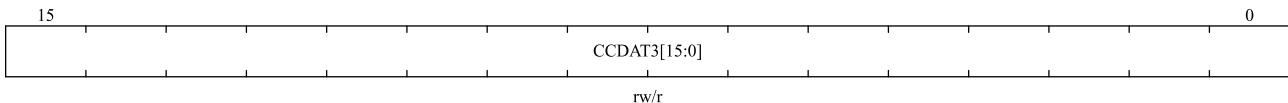


Bit field	Name	Description
15:0	CCDAT2[15:0]	<p>Capture/Compare 2 values</p> <ul style="list-style-type: none"> <li>■ CC2 channel is configured as output: CCDAT2 contains the value to be compared to the counter TIMx_CNT, signaling on the OC2 output.</li> <li>If the preload feature is not selected in TIMx_CCMOD1.OC2PEN bit, the written value is immediately transferred to the active register. Otherwise, this preloaded value is transferred to the active register only when an update event occurs.</li> <li>■ CC2 channel is configured as input: CCDAT2 contains the counter value transferred by the last input capture 2 event (IC2).</li> <li>When configured as input mode, register CCDAT2 and CCDDAT2 are only readable.</li> <li>When configured as output mode, register CCDAT2 and CCDDAT2 are readable and writable.</li> </ul>

#### 9.4.17 Capture/compare register 3 (TIMx\_CC DAT3)

Offset address: 0x3C

Reset value: 0x0000

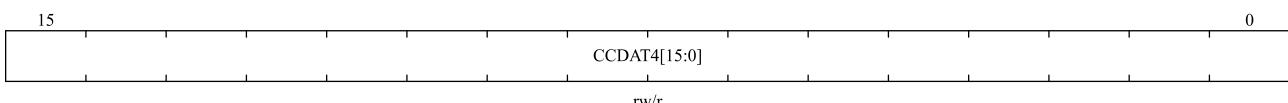


Bit field	Name	Description
15:0	CC DAT3[15:0]	<p>Capture/Compare 3 value</p> <ul style="list-style-type: none"> <li>■ CC3 channel is configured as output: CCDAT3 contains the value to be compared to the counter TIMx_CNT, signaling on the OC3 output.</li> <li>If the preload feature is not selected in TIMx_CCMOD2.OC3PEN bit, the written value is immediately transferred to the active register. Otherwise, this preloaded value is transferred to the active register only when an update event occurs.</li> <li>■ CC3 channel is configured as input: CCDAT3 contains the counter value transferred by the last input capture 3 event (IC3).</li> <li>When configured as input mode, register CC DAT3 and CCDDAT3 are only readable.</li> <li>When configured as output mode, register CC DAT3 and CCDDAT3 are readable and writable.</li> </ul>

#### 9.4.18 Capture/compare register 4 (TIMx\_CC DAT4)

Offset address: 0x40

Reset value: 0x0000



Bit field	Name	Description
15:0	CC DAT4[15:0]	<p>Capture/Compare 4 value</p> <ul style="list-style-type: none"> <li>■ CC4 channel is configured as output: CCDAT4 contains the value to be compared to the counter TIMx_CNT, signaling on the OC4 output.</li> <li>If the preload feature is not selected in TIMx_CCMOD2.OC4PEN bit, the written value is immediately transferred to the active register. Otherwise, this preloaded value is transferred to the active register only when an update event occurs.</li> <li>■ CC4 channel is configured as input: CCDAT4 contains the counter value transferred by the last input capture 4 event (IC4).</li> </ul>

		When configured as input mode, register CC DAT4 and CC D DAT4 are only readable. When configured as output mode, register CC DAT4 and CC D DAT4 are readable and writable.
--	--	---

### 9.4.19 Break and Dead-time registers (TIMx\_BKDT)

Offset address: 0x44

Reset value: 0x0000

15 MOEN	14 AOEN	13 BKP	12 BKEN	11 OSSR	10 OSSI	9 LCKCFG[1:0]	8	7	DTGN[7:0]	0
rw	rw	rw	rw	rw	rw	rw			rw	

*Note:* AOEN, BKP, BKEN, OSSR, and DTGN [7:0] bits can all be write protected depending on the LOCK configuration, and it is necessary to configure all of them on the first write to the TIMx\_BKDT register.

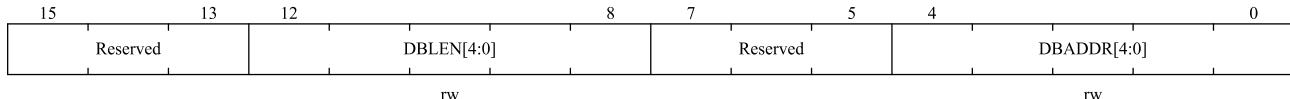
Bit field	Name	Description
15	MOEN	Main Output enable  This bit can be set by software or hardware depending on the TIMx_BKDT.AOEN bit, and is asynchronously cleared to '0' by hardware once the brake input is active. It is only valid for channels configured as outputs.  0: OC and OCN outputs are disabled or forced to idle state. 1: OC and OCN outputs are enabled if TIMx_CCEN.CCxEN or TIMx_CCEN.CCxNEN bits are set. For more details, see Section 9.4.10 Capture/Compare enable registers (TIMx_CCEN).
14	AOEN	Automatic output enable  0: Only software can set TIMx_BKDT.MOEN; 1: Software sets TIMx_BKDT.MOEN; or if the break input is not active, when the next update event occurs, hardware automatically sets TIMx_BKDT.MOEN.
13	BKP	Break input polarity  0: Low level of the brake input is valid 1: High level of the brake input is valid  <i>Note:</i> Any write to this bit requires an APB clock delay to take effect.
12	BKEN	Break enable  0: Disable brake input (BRK and CCS clock failure events) 1: Enable brake input (BRK and CCS clock failure events)  <i>Note:</i> Any write to this bit requires an APB clock delay to take effect.
11	OSSR	Off-state Selection for Run Mode  This bit is used when TIMx_BKDT.MOEN=1 and the channel is a complementary output. The OSSR bit does not exist in timer without complementary outputs.  0: When inactive, OCx/OCxN outputs are disabled (OCx/OCxN enable output signal = 0) 1: When inactive, OCx/OCxN outputs are enabled with their inactive level as soon as CCxEN = 1 or CCxNEN = 1. Then, OCx/OCxN enable output signal = 1  For more details, See Section 9.4.10, capture/compare enablement registers (TIMx_CCEN).

Bit field	Name	Description
10	OSSI	<p>Off-state Selection for Idle Mode</p> <p>This bit is used when TIMx_BKDT.MOEN=0 and the channels configured as outputs.</p> <p>0: When inactive, OCx/OCxN outputs are disabled (OCx/OCxN enable output signal = 0)</p> <p>1: When inactive, OCx/OCxN outputs are enabled with their idle level as soon as CCxEN = 1 or CCxNEN = 1. Then, OCx/OCxN enable output signal = 1</p> <p>For more details, See Section 9.4.10, capture/compare enablement registers (TIMx_CCEN).</p>
9:8	LCKCFG[1:0]	<p>Lock Configuration</p> <p>These bits offer a write protection against software errors.</p> <p>00:</p> <ul style="list-style-type: none"> <li>– No write protected.</li> </ul> <p>01:</p> <ul style="list-style-type: none"> <li>– LOCK Level 1</li> </ul> <p>TIMx_BKDT.DTGN, TIMx_BKDT.BKEN, TIMx_BKDT.BKP, TIMx_BKDT.AOEN, TIMx_CTRL2.OIx, TIMx_CTRL2.OIxN bits enable write protection.</p> <p>10:</p> <ul style="list-style-type: none"> <li>– LOCK Level 2</li> </ul> <p>Except for register write protection in LOCK Level 1 mode, TIMx_CCEN.CCxP and TIMx_CCEN.CCxNP (If the corresponding channel is configured in output mode), TIMx_BKDT.OSSR and TIMx_BKDT.OSSI bits also enable write protection.</p> <p>11:</p> <ul style="list-style-type: none"> <li>– LOCK Level 3</li> </ul> <p>Except for register write protection in LOCK Level 2, TIMx_CCMODx.OCxMD and TIMx_CCMODx.OCxPEN bits (If the corresponding channel is configured in output mode) also enable write protection.</p> <p><i>Note: After the system reset, the LCKCFG bit can only be written once. Once written to the TIMx_BKDT register, LCKCFG will be protected until the next reset.</i></p>
7:0	DTGN [7:0]	<p>Dead-time Generator</p> <p>These bits define the dead-time duration between inserted complementary outputs. The relationship between the DTGN value and the dead time is as follows::</p> <p>DTGN[7:5] = 0xx: dead time = DTGN[7:0] ×(t<sub>DTS</sub>)</p> <p>DTGN[7:5] = 10x: dead time =(64+DTGN[5:0]) ×(2 × t<sub>DTS</sub>)</p> <p>DTGN[7:5]=110: dead time =(32+DTGN[4:0]) ×(8 × t<sub>DTS</sub>)</p> <p>DTGN [then] = 111: dead time =(32 + DTGN [4:0]) ×(16 × t<sub>DTS</sub>)</p> <p>t<sub>DTS</sub> value see TIMx_CTRL1.CLKD [1:0].</p>

### 9.4.20 DMA Control register (TIMx\_DCTRL)

Offset address: 0x48

Reset value: 0x0000

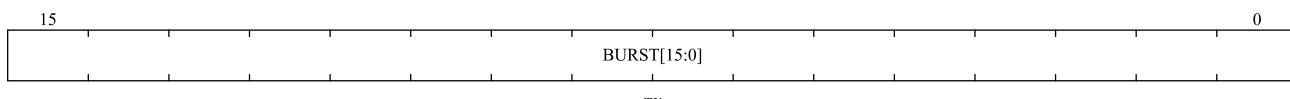


Bit field	Name	Description
15:9	Reserved	Reserved, the reset value must be maintained, kept at 0.
12:8	DBLEN[4:0]	<p>DMA Burst Length</p> <p>This bit field defines the number DMA will access (write/read) TIMx_DADDR register.</p> <p>00000: 1 time transfer          00001: 2 times transfers          00010: 3 times transfers          ...          10001: 18 times transfers</p>
7:5	Reserved	Reserved, the reset value must be maintained.
4:0	DBADDR[4:0]	<p>DMA Base Address</p> <p>This bit field defines the first address where the DMA accesses the TIMx_DADDR register.</p> <p>When access is done through the TIMx_DADDR first time, this bit-field specifies the address you just access. And then the second access to the TIMx_DADDR, you will access the address of “DMA Base Address + 4”</p> <p>00000: TIMx_CTRL1,          00001: TIMx_CTRL2,          00010: TIMx_SMCTRL,          ...          10001: TIMx_BKDT          10010: TIMx_DCTRL</p>

### 9.4.21 DMA transfer buffer register (TIMx\_DADDR)

Offset address: 0x4C

Reset value: 0x0000



Bit field	Name	Description
15:0	BURST[15:0]	<p>DMA access buffer.</p> <p>When a read or write operation is assigned to this register, the register located at the address range (DMA base address + DMA burst length × 4) will be accessed.</p> <p>DMA base address = The address of TIM_CTRL1 + TIMx_DCTRL.DBADDR * 4;</p> <p>DMA burst len = TIMx_DCTRL.DBLEN + 1.</p> <p>Example:</p> <p>If TIMx_DCTRL.DBLEN = 0x3(4 transfers), TIMx_DCTRL.DBADDR = 0xD (TIMx_CCDAT1), DMA data length = half word, DMA memory address = buffer address in SRAM, DMA peripheral address = TIMx_DADDR address.</p> <p>When an event occurs, TIMx will send requests to the DMA, and transfer data 4 times.</p> <p>For the first time, DMA access to the TIMx_DADDR register will be mapped to access TIMx_CCDAT1 register;</p> <p>For the second time, DMA access to the TIMx_DADDR register will be mapped to access TIMx_CCDAT2 register;</p> <p>....</p> <p>For the fourth time, DMA access to the TIMx_DADDR register will be mapped to access TIMx_CCDAT4 register;</p>

#### 9.4.22 Capture/compare mode registers 3(TIMx\_CCMOD3)

Offset address: 0x54

Reset value: 0x0000

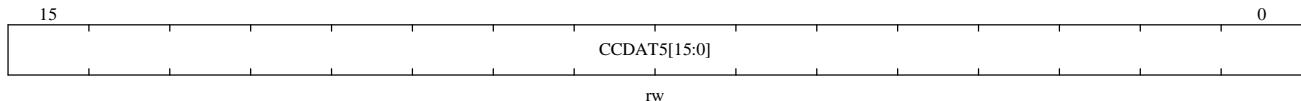
15	OC6CEN	rw	14	OC6MD[2:0]	rw	12	OC6PEN	rw	11	OC6FEN	rw	9	Reserved		8	OC5CEN	rw	7	OC5MD[2:0]	rw	6	OC5PEN	rw	4	OC5FEN	rw	3	Reserved		2	Reserved		1	Reserved		0	Reserved	
----	--------	----	----	------------	----	----	--------	----	----	--------	----	---	----------	--	---	--------	----	---	------------	----	---	--------	----	---	--------	----	---	----------	--	---	----------	--	---	----------	--	---	----------	--

Bit field	Name	Description
15	OC6CEN	Output compare 6 clear enable
14:12	OC6MD[2:0]	Output compare 6 mode
11	OC6PEN	Output compare 6 preload enable
10	OC6FEN	Output compare 6 fast enable
9:8	Reserved	Reserved, the reset value must be maintained
7	OC5CEN	Output compare 5 clear enable
6:4	OC5MD[2:0]	Output compare 5 mode
3	OC5PEN	Output compare 5 Preload enable
2	OC5FEN	Output compare 5 fast enable
1: 0	Reserved	Reserved, the reset value must be maintained

### 9.4.23 Capture/compare register 5 (TIMx\_CC DAT5)

Offset address: 0x58

Reset value: 0x0000

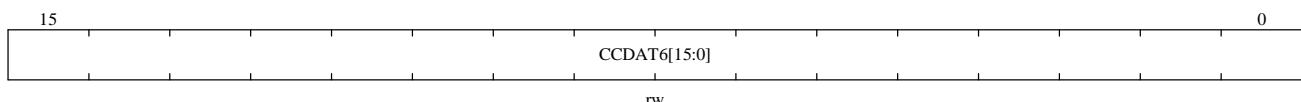


Bit field	Name	Description
15:0	CCDAT5[15:0]	<p>Capture/Compare 5 value</p> <ul style="list-style-type: none"> <li>■ CC5 channel can only configured as output:</li> </ul> <p>CCDAT5 contains the value to be compared to the counter TIMx_CNT, signaling on the OC5 output.</p> <p>If the preload feature is not selected in TIMx_CCMOD3.OC5PEN bit, the written value is immediately transferred to the active register. Otherwise, this preloaded value is transferred to the active register only when an update event occurs.</p> <p>TIM1_CC5 and TIM8_CC5 is used for comparator blanking.</p>

### 9.4.24 Capture/compare register 6 (TIMx\_CC DAT6)

Offset address: 0x5C

Reset value: 0x0000



Bit field	Name	Description
15:0	CCDAT6[15:0]	<p>Capture/Compare 6 value</p> <ul style="list-style-type: none"> <li>■ CC6 channel can only configured as output:</li> </ul> <p>CCDAT6 contains the value to be compared to the counter TIMx_CNT, signaling on the OC6 output.</p> <p>If the preload feature is not selected in TIMx_CCMOD3.OC6PEN bit, the written value is immediately transferred to the active register. Otherwise, this preloaded value is transferred to the active register only when an update event occurs.</p> <p>TIM1_CC6 for OPAMP switch.</p>

## 10 General-purpose timers (TIM3)

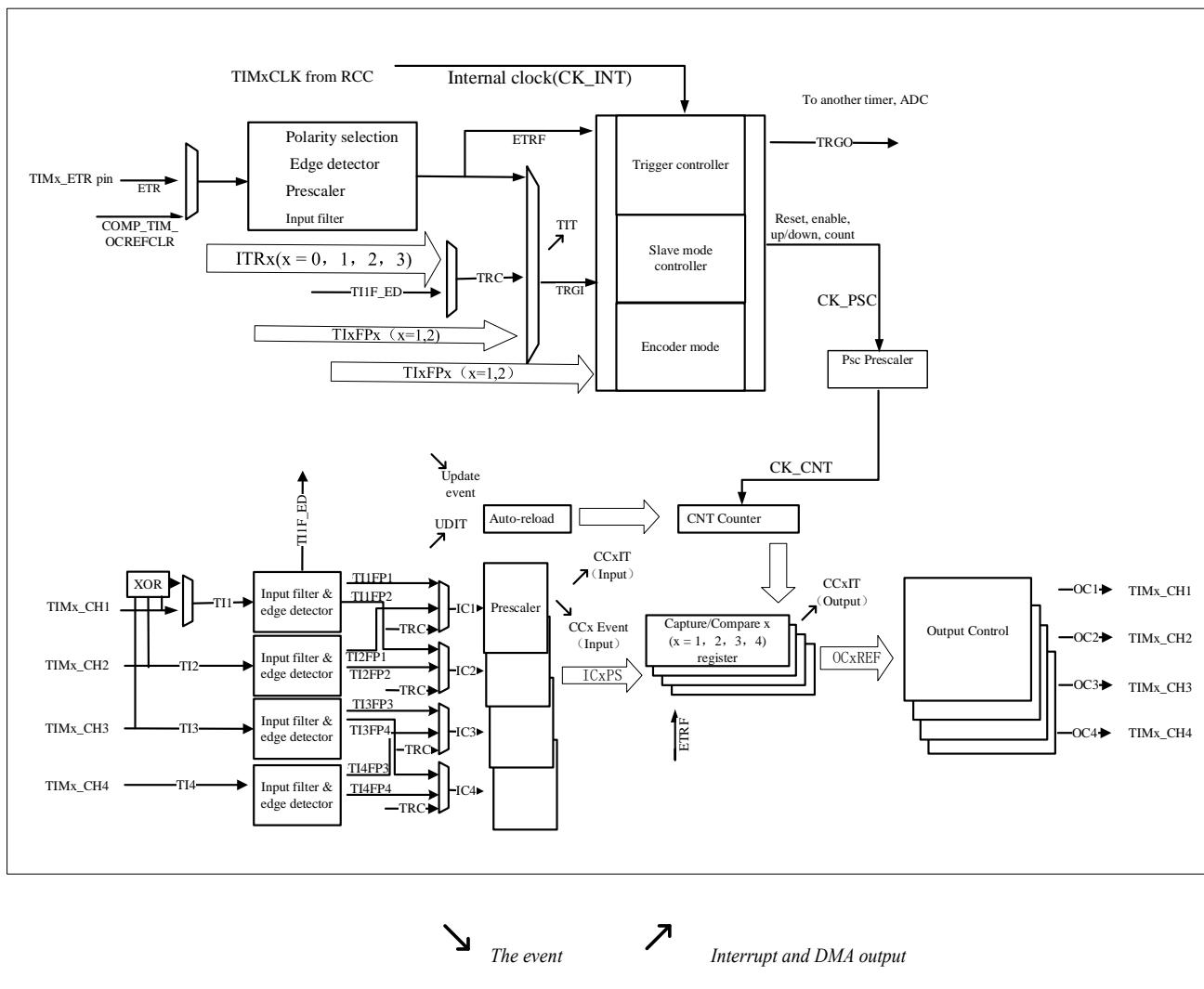
### 10.1 General-purpose timers introduction

The general-purpose timers (TIM3) is mainly used in the following occasions: counting the input signal, measuring the pulse width of the input signal and generating the output waveform, etc.

### 10.2 Main features of General-purpose timers

- 16-bit auto-reload counters. (It can realize up-counting, down-counting, up/down counting)
- 16-bit programmable prescaler. (The frequency division factor can be configured with any value between 1 and 65536)
- TIM3 up to 4 channels
- Channel's working modes: PWM output, ouput compare, one-pulse mode output, input capture
- The events that generate the interrupt/DMA are as follows:
  - ◆ Update event
  - ◆ Trigger event
  - ◆ Input capture
  - ◆ Output compare
- Timer can be controlled by external signal
- Timers are linked internally for timer synchronization or chaining
- Incremental (quadrature) encoder interface: used for tracking motion and resolving rotation direction and position
- Hall sensor interface: used to do three-phase motor control

Figure 10-1 Block diagram of TIMx (x=3)



## 10.3 General-purpose timers description

### 10.3.1 Time-base unit

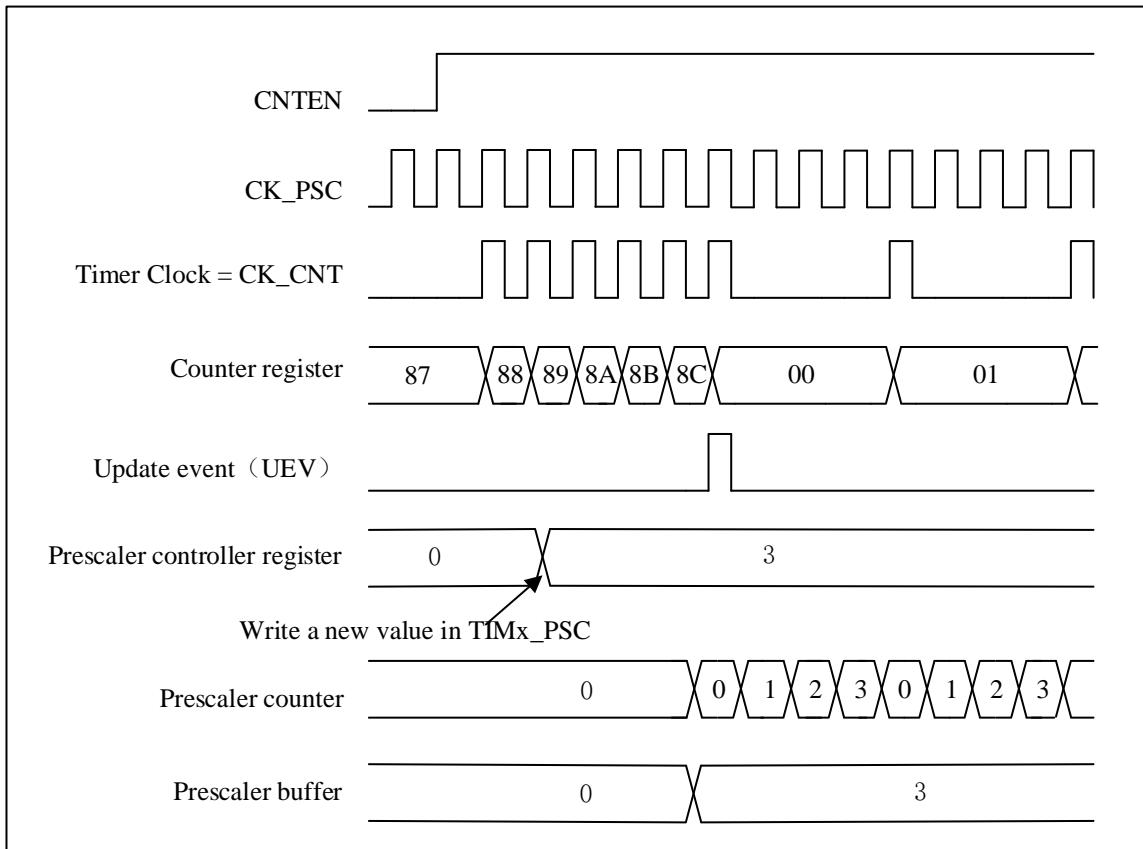
The time-base unit mainly includes: prescaler, counter and auto-reload. When the time base unit is working, the software can read and write the corresponding registers (TIMx\_PSC, TIMx\_CNT and TIMx\_AR) at any time.

Depending on the setting of the auto-reload preload enable bit (TIMx\_CTRL1.ARREN), the value of the preload register is transferred to the shadow register immediately or at each update event UEV. An update event is generated when the counter reaches the overflow/underflow condition and it can be generated by software when TIMx\_CTRL1.UPDIS=0. The counter CK\_CNT is valid only when the TIMx\_CTRL1.CNTEN bit is set. The counter starts counting one clock cycle after the TIMx\_CTRL1.CNTEN bit is set.

### 10.3.1.1 Prescaler description

The TIMx\_PSC register consists of a 16-bit counter that can be used to divide the counter clock frequency by any factor between 1 and 65536. It can be changed on the fly as it is buffered. The prescaler value is only taken into account at the next update event.

Figure 10-2 Counter timing diagram with prescaler division change from 1 to 4



## 10.3.2 Counter mode

### 10.3.2.1 Up-counting mode

In up-counting mode, the counter will count from 0 to the value of the register TIMx\_AR, then it resets to 0. And a counter overflow event is generated.

If the TIMx\_CTRL1.UPRS bit (select update request) and the TIMx\_EVTGEN.UDGN bit are set, an update event (UEV) will generate And TIMx\_STS.UDITF will not be set by hardware, therefore, no update interrupts or update DMA requests are generated. This setting is used in scenarios where you want to clear the counter but do not want to generate an update interrupt.

Depending on the update request source is configured in TIMx\_CTRL1.UPRS, When an update event occurs, all registers are updated and the TIMx\_STS.UDITF is set:

- Update auto-reload shadow registers with preload value(TIMx\_AR), when TIMx\_CTRL1.AR PEN = 1.
- The prescaler shadow register is reloaded with the preload value(TIMx\_PSC).

To avoid updating the shadow registers when new values are written to the preload registers, you can disable the update by setting TIMx\_CTRL1.UPDIS=1.

When an update event occurs, the counter will still be cleared and the prescaler counter will also be set to 0 (but the prescaler value will remain unchanged).

The figure below shows some examples of the counter behavior and the update flags for different division factors in the up-counting mode.

Figure 10-3 Timing diagram of up-counting. The internal clock divider factor = 2/N

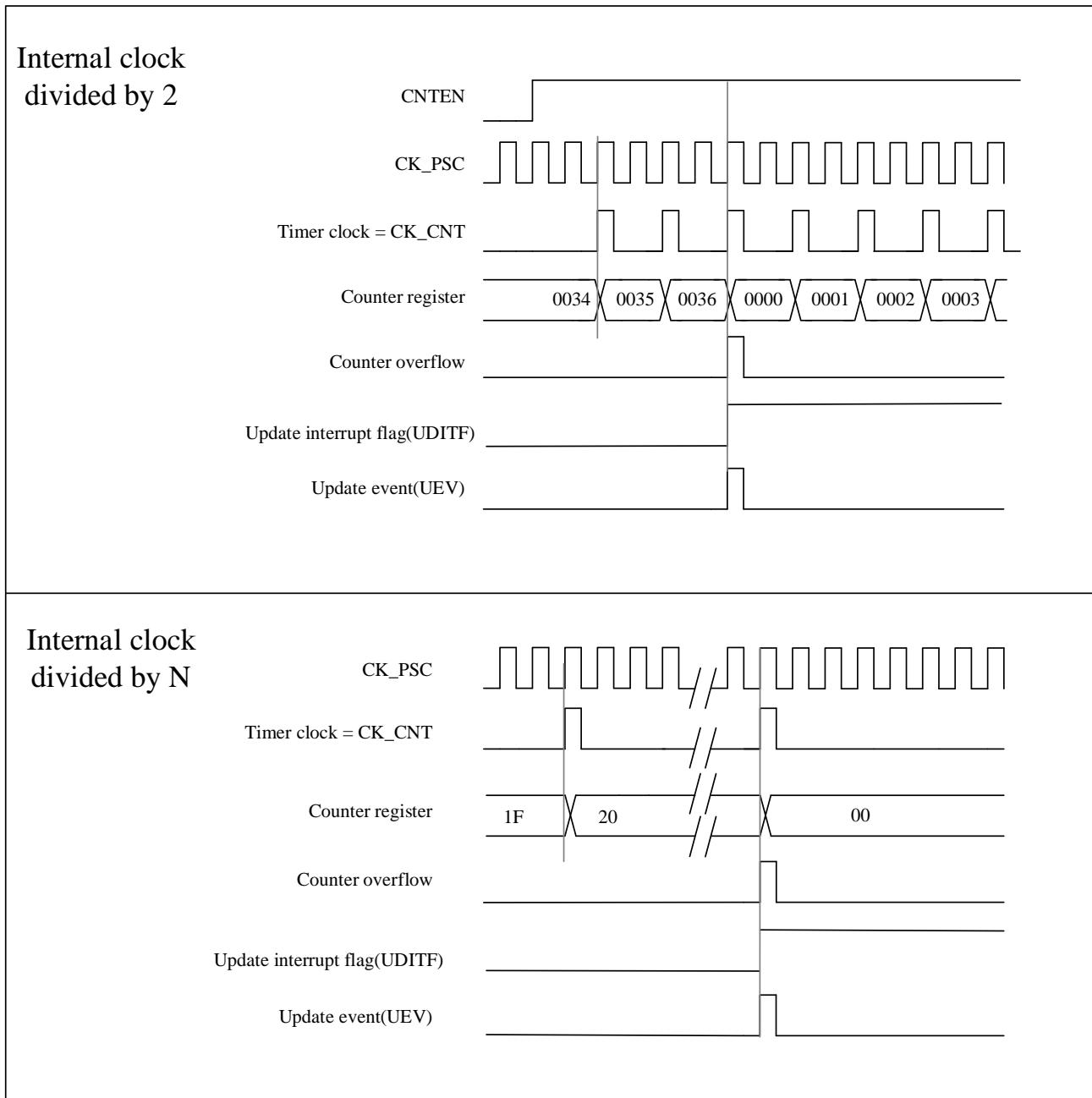
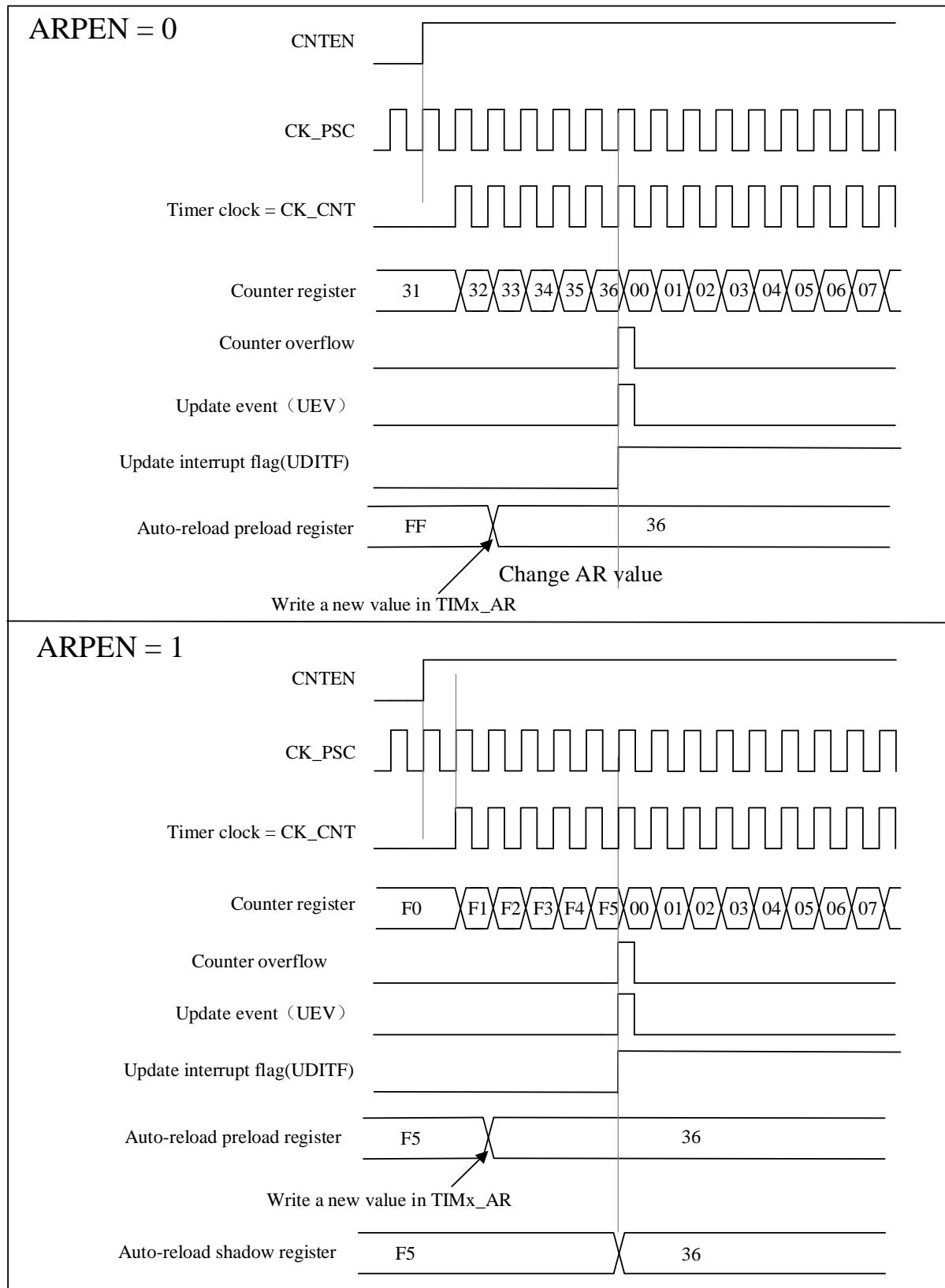


Figure 10-4 Timing diagram of the up-counting, update event when ARPEN=0/1



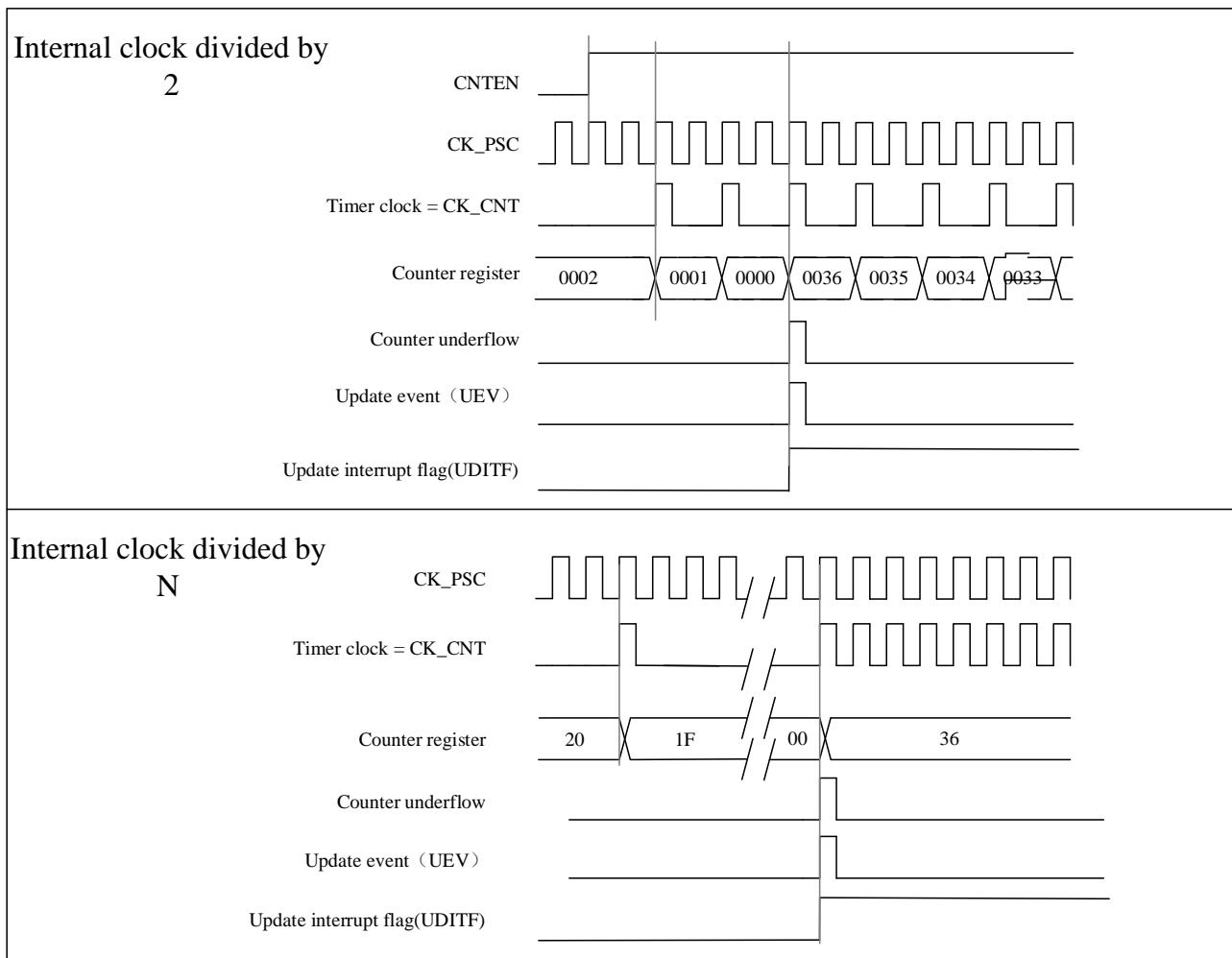
### 10.3.2.2 Down-counting mode

In down-counting mode, the counter will decrement from the value of the register `TIMx_AR` to 0, then restart from the auto-reload value and generate a counter underflow event.

The process of configuring update events and updating registers in down-counting mode is the same as in up-counting mode, see 10.3.2.1.

The figure below shows some examples of the counter behavior and the update flags for different division factors in the down-counting mode.

Figure 10-5 Timing diagram of the down-counting, internal clock divided factor =  $2/N$



### 10.3.2.3 Center-aligned mode

In center-aligned mode, the counter increments from 0 to the value (`TIMx_AR`) – 1, a counter overflow event is generated. It then counts down from the auto-reload value (`TIMx_AR`) to 1 and generates a counter underflow event. Then the counter resets to 0 and starts counting up again.

In this mode, the `TIMx_CTRL1.DIR` direction bits have no effect and the count direction is updated and specified by hardware. Center-aligned mode is valid when the `TIMx_CTRL1.CAMSEL` bit is not equal to "00".

The update events can be generated each time the counter overflows and each time the counter underflows. Alternatively, an update event can also be generated by setting the TIMx\_EVTGEN. UDGN bit (either by software or using a slave mode controller). In this case, the counter restarts from 0, as does the prescaler's counter.

Please note: if the update source is a counter overflow, auto-reload update before reloading the counter.

Figure 10-6 Timing diagram of the Center-aligned, internal clock divided factor =2/N

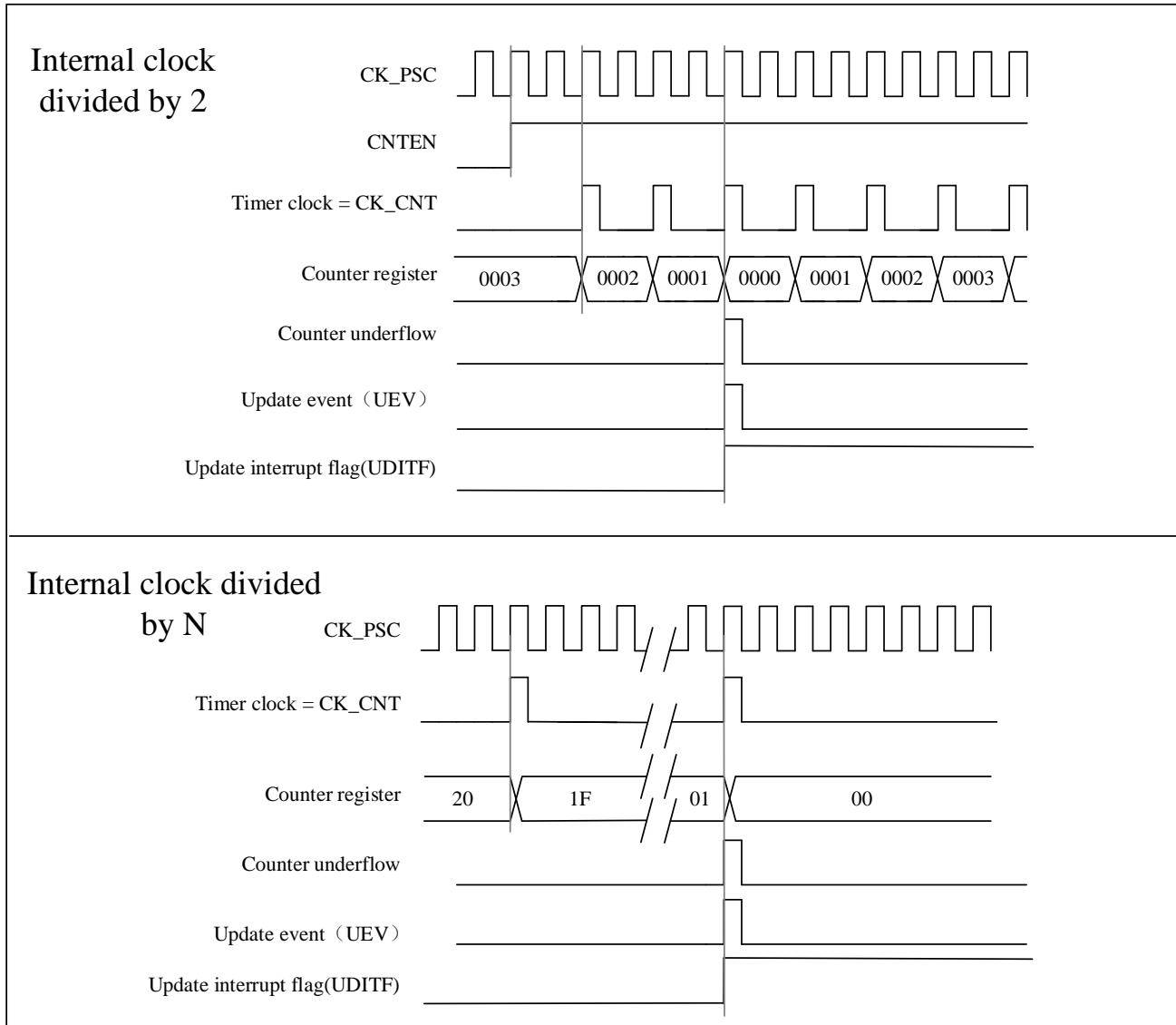
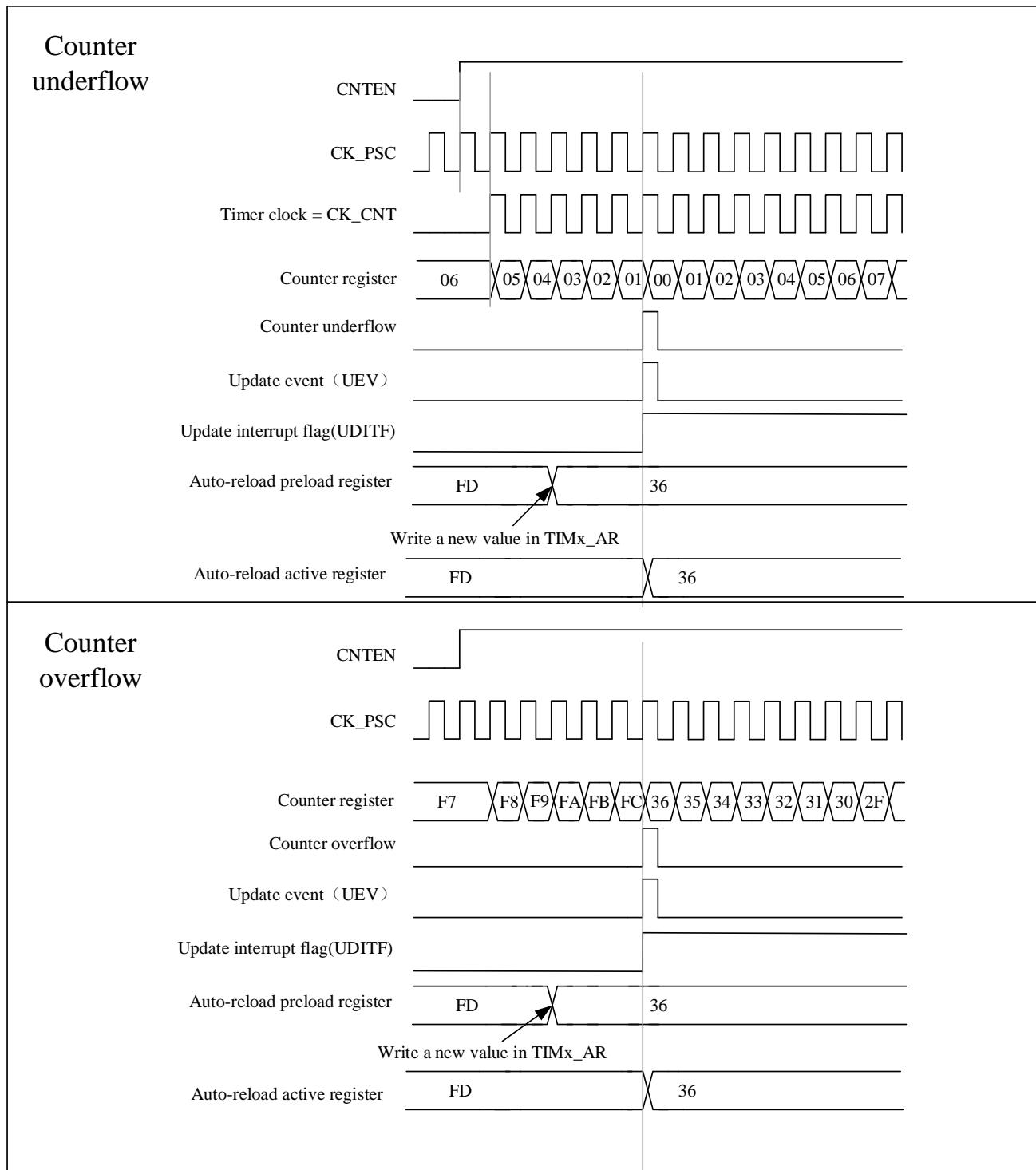


Figure 10-7 A center-aligned sequence diagram that includes counter overflows and underflows (ARPEN = 1)



### 10.3.3 Clock selection

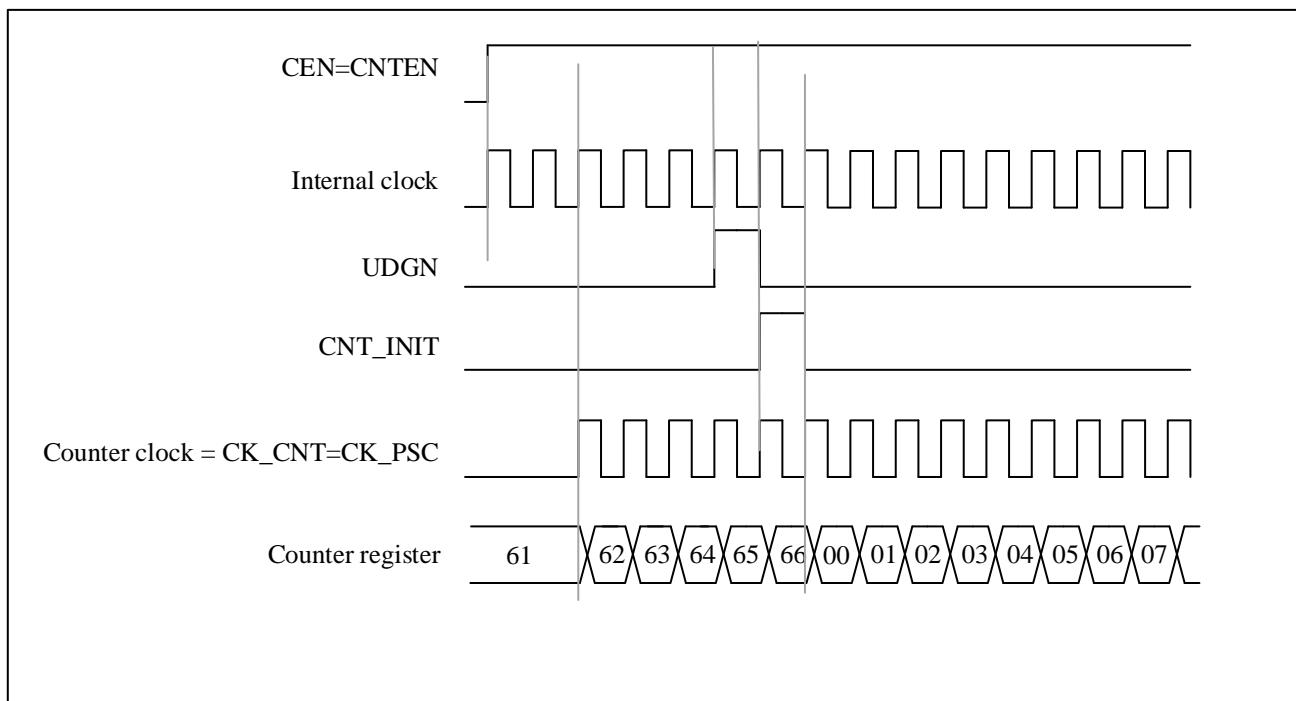
- The internal clock of timers : CK\_INT
- Two kinds of external clock mode :

- external input pin
- external trigger input ETR
- Internal trigger input (ITRx): one timer is used as a prescaler for another timer.

#### 10.3.3.1 Internal clock source (CK\_INT)

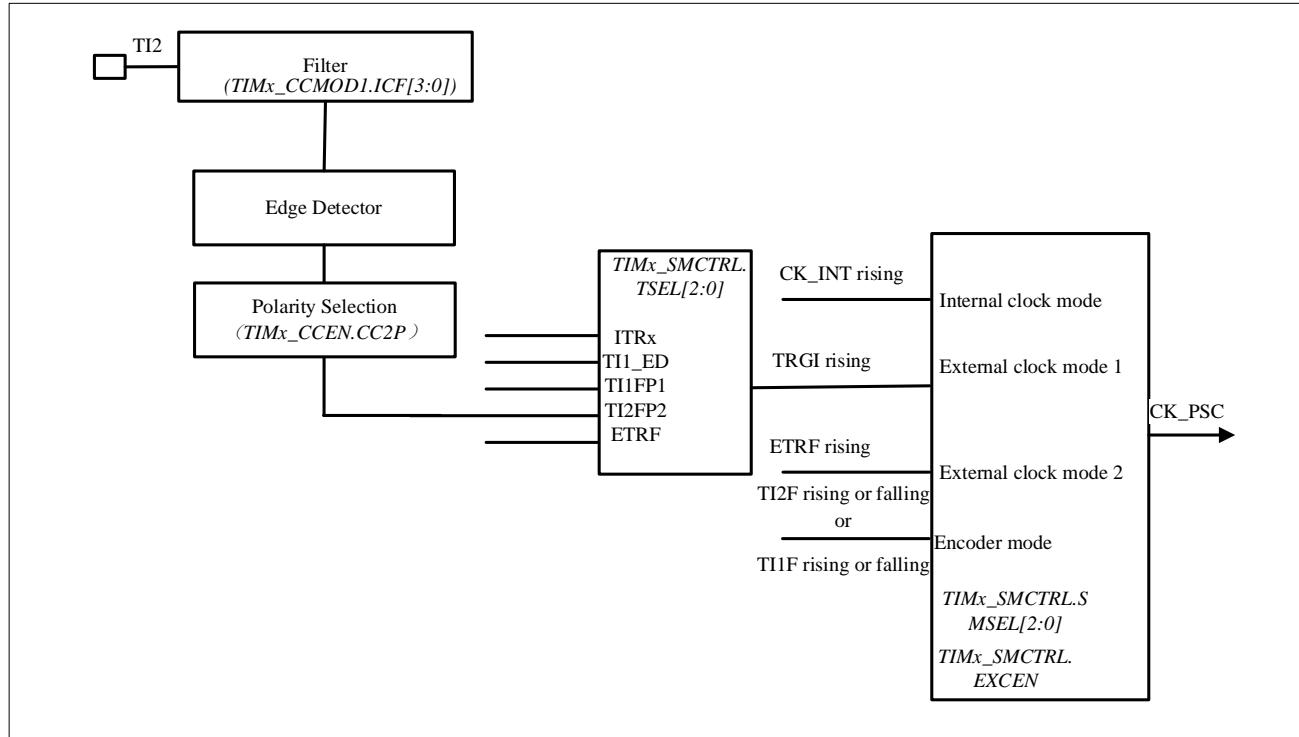
When the TIMx\_SMCTRL.SMSEL is equal to “000”, the slave mode controller is disabled. The three control bits (TIMx\_CTRL1.CNTEN □ TIMx\_CTRL1. DIR □ TIMx\_EVTGEN. UDGN) can only be changed by software (except TIMx\_EVTGEN. UDGN, which remains cleared automatically). It is provided that the TIMx\_CTRL1.CNTEN bit is written as '1' by soft, the clock source of the prescaler is provided by the internal clock CK\_INT.

Figure 10-8 Control circuit in normal mode, internal clock divided by 1



### 10.3.3.2 External clock source mode 1

Figure 10-9 TI2 external clock connection example



This mode is selected by configuring `TIMx_SMCTRL.SMSEL=111`. The counter can be configured to count on the rising or falling edge of the clock at the selected input.

For example, to configure up-counting mode to count on the rising edge of the clock at the TI2 input, the configuration steps are as follows:

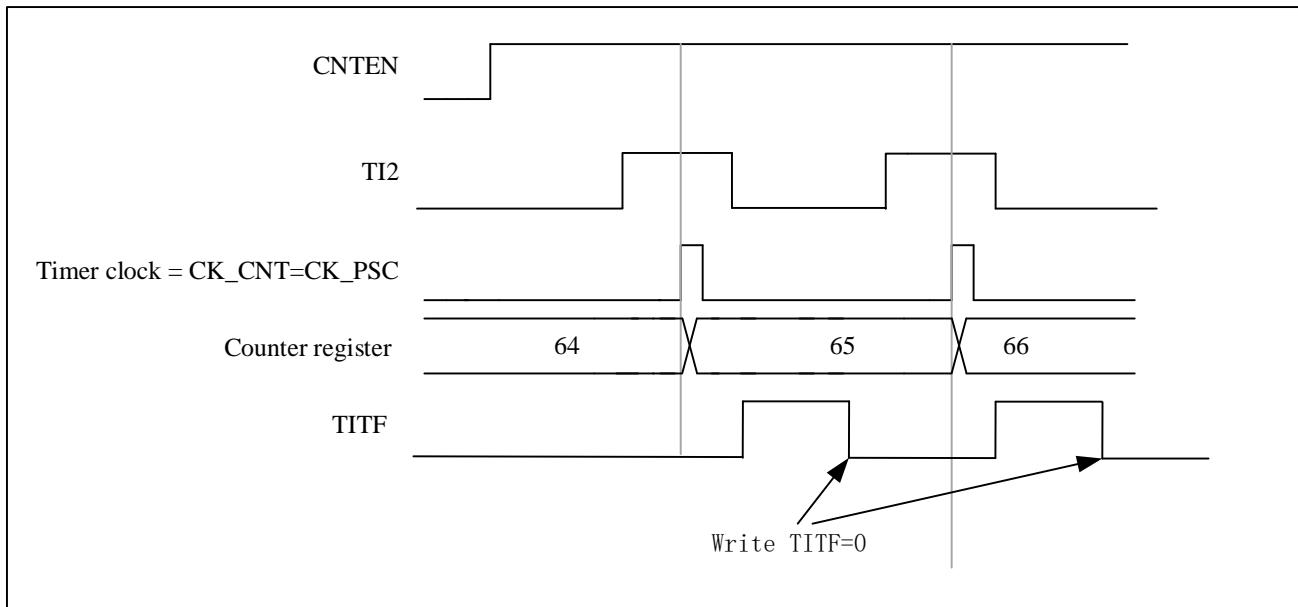
- Configure `TIMx_CCMOD1.CC2SEL` equal to ‘01’, CC2 channel is configured as input, IC2 is mapped to TI2
- Configure `TIMx_CCEN.CC2P` equal to ‘0’, select clock rising edge polarity
- To select input filter bandwidth by configuring `TIMx_CCMOD1.IC2F[3:0]` (if filter is not needed, keep IC2F bit at ‘0000’)
- Configure `TIMx_SMCTRL.SMSEL` equal to ‘111’, select timer external clock mode 1
- Configure `TIMx_SMCTRL.TSEL` equal to ‘110’, select TI2 as the trigger input source
- Configure `TIMx_CTRL1.CNTEN` equal to ‘1’ to start the counter

*Note: The capture prescaler is not used for triggering, so it does not need to be configured*

When the rising edge of the timer clock occurs at `TI2=1`, the counter counts once and the `TIMx_STS .TITF` flag is pulled high.

The delay between the rising edge of TI2 and the actual clock of the counter depends on the resynchronization circuit at the input of TI2.

Figure 10-10 Control circuit in external clock mode 1

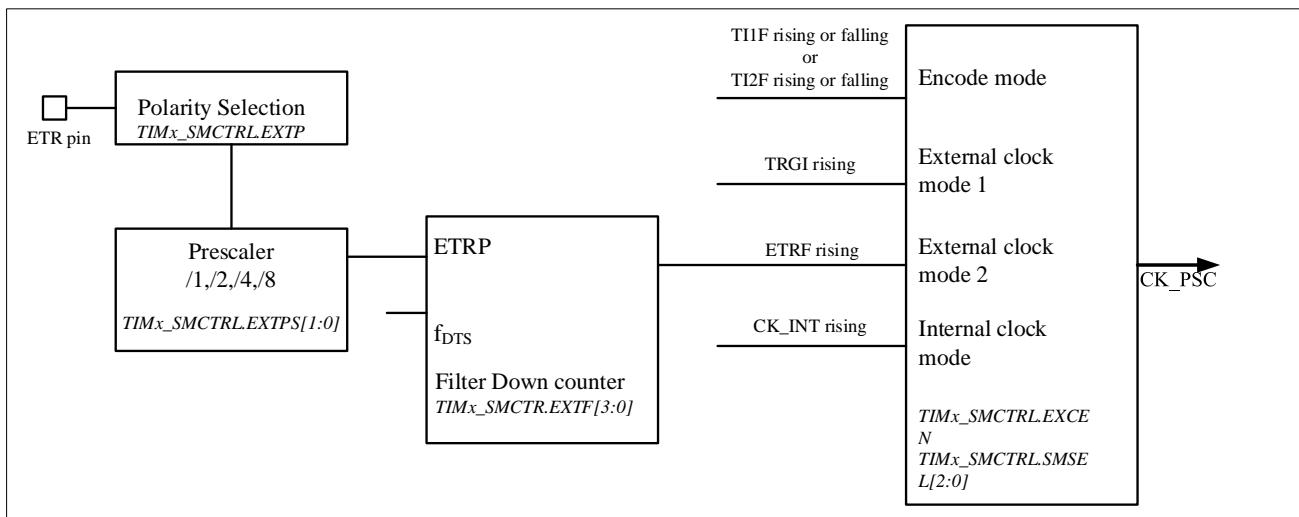


### 10.3.3.3 External clock source mode 2

This mode is selected by `TIMx_SMCTRL.EXCEN` equal to 1. The counter can count on every rising or falling edge of the external trigger input ETR.

The following figure is a schematic diagram of the external trigger input module in External clock source mode 2

Figure 10-11 External trigger input block diagram



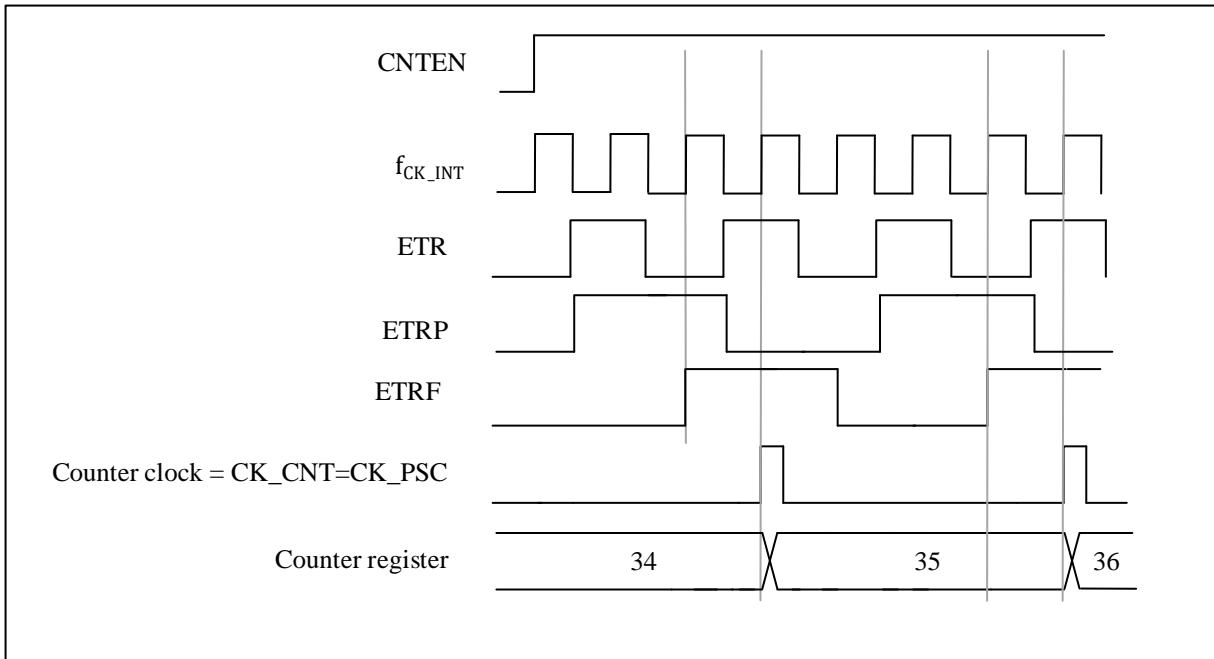
For example, use the following configuration steps to make the up counter count every 2 rising edges on ETR.

- Since no filter is needed in this case, make `TIMx_SMCTRL.EXTF[3:0]` equal to '0000'
- Configure the prescaler by making `TIMx_SMCTRL.EXTPS[1:0]` equal to '01'
- Select the polarity on ETR pin by setting `TIMx_SMCTRL.EXTP` equal to '0', The rising edge of ETR is valid

- External clock mode 2 is selected by setting TIMx\_SMCTRL .EXCEN equal to '1'
- Turn on the counter by setting TIMx\_CTRL1. CNTEN equal to '1'

The counter counts every 2 rising edges of ETR. The delay between the rising edge of ETR and the actual clock to the counter is due to a resynchronization circuit on the ETRP signal.

Figure 10-12 Control circuit in external clock mode 2

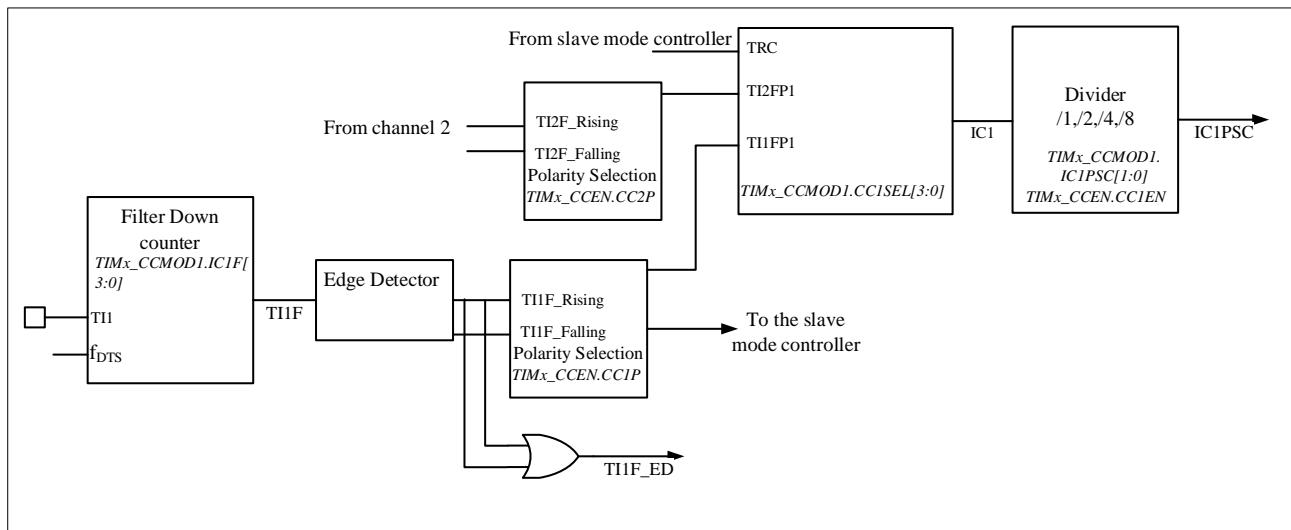


#### 10.3.4 Capture/compare channels

Capture/compare channels include capture/compare registers and shadow registers. The input section consists of digital filters, multiplexers and prescalers. The output section includes comparators and output controls.

The input signal TI<sub>x</sub> is sampled and filtered to generate the signal TI<sub>x</sub>F. A signal (TI<sub>x</sub>F\_rising or TI<sub>x</sub>F\_falling) is then generated by the edge detector of the polarity select function, the polarity of which is selected by the TIMx\_CCEN.CCxP bits. This signal can be used as a trigger input for the slave mode controller. At the same time, the signal IC<sub>x</sub> is sent to the capture register after frequency division. The following figure shows a block diagram of a capture/compare channel.

Figure 10-13 Capture/compare channel (example: channel 1 input stage)



The output part generates an intermediate waveform OCxRef (active high) as reference. The polarity acts at the end of the chain.

Figure 10-14 Capture/compare channel 1 main circuit

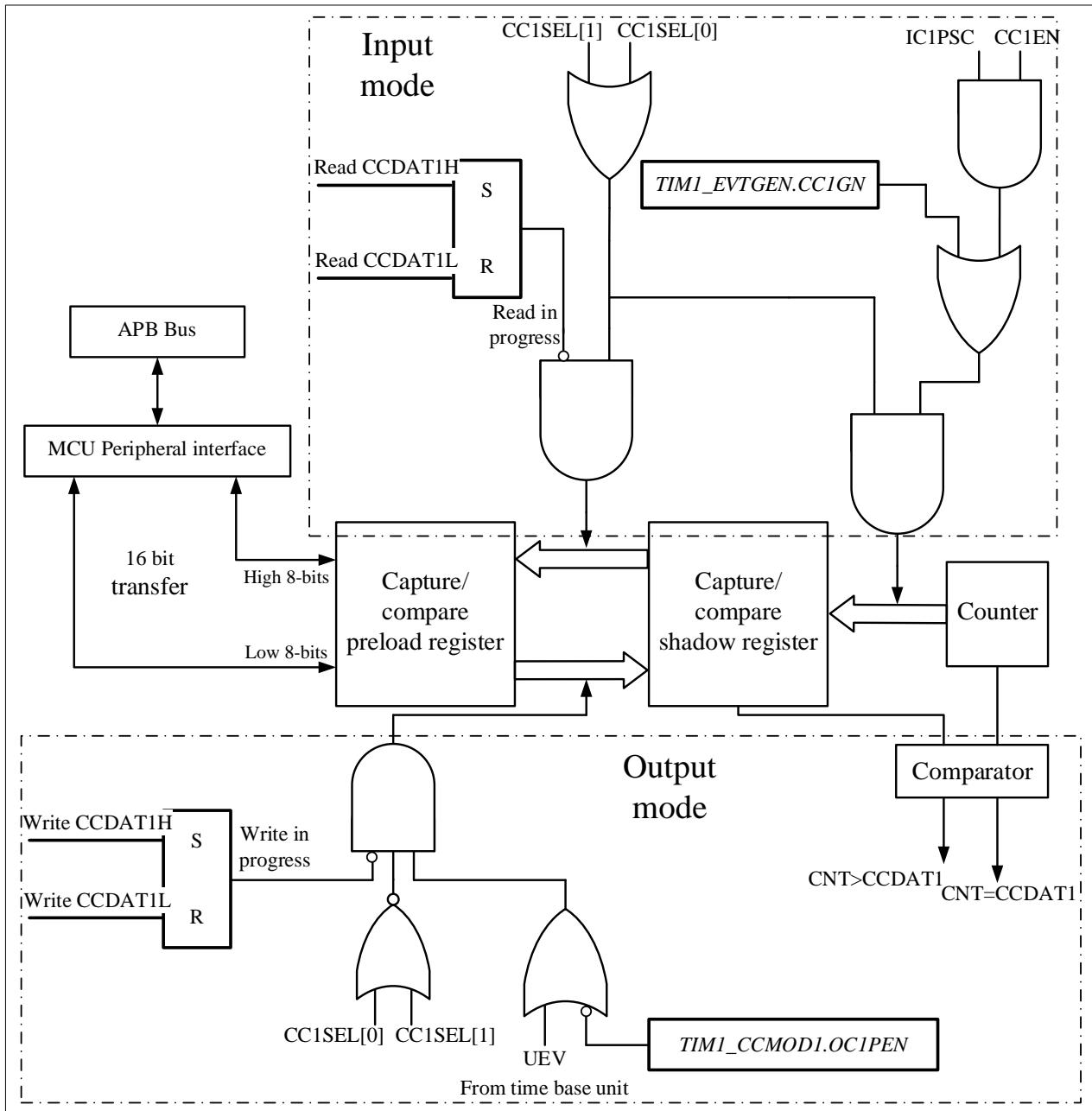
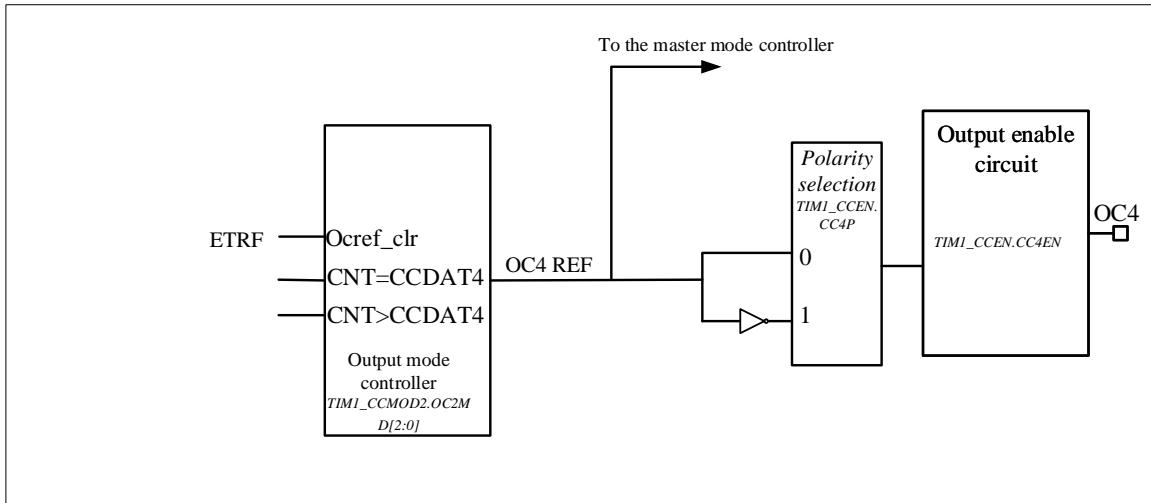


Figure 10-15 Output part of channelx ( $x = 1,2,3,4$ ; take channel 4 as an example)



Reads and writes always access preloaded registers when capturing/comparing. The two specific working processes are as follows:

In capture mode, the capture is actually done in the shadow register, and then the value in the shadow register is copied into the preload register.

In compare mode, as opposed to capture mode, the value of the preload register is copied into the shadow register, which is compared with the counter.

### 10.3.5 Input capture mode

In capture mode, the TIMx\_CCDATx registers are used to latch the counter value after the ICx signal detects.

There is a capture interrupt flag TIMx\_STS.CCxITF, which can issue an interrupt or DMA request if the corresponding interrupt enable is pulled high.

The TIMx\_STS.CCxITF bit is set by hardware when a capture event occurs and is cleared by software or by reading the TIMx\_CCDATx register.

The overcapture flag TIMx\_STS.CCxOCF is set equal to 1 when the counter value is captured in the TIMx\_CCDATx register and TIMx\_STS.CC1ITF is already pulled high. Unlike the former, TIMx\_STS.CCxOCF is cleared by writing 0 to it.

To achieve a rising edge of the TI1 input to capture the counter value into the TIMx\_CCDAT1 register, the configuration flow is as follows:

- To select a valid input:

Configure TIMx\_CCMOD1.CC1SEL to '01'. At this time, the input is the CC1 channel, and IC1 is mapped to TI1.

- Program the desired input filter duration:

Define the sampling frequency of the TI1 input and the length of the digital filter by configuring the

TIMx\_CCMODx.ICxF bits. Example: If the input signal jitters up to 5 internal clock cycles, we must choose a filter duration longer than these 5 clock cycles. When 8 consecutive samples (sampled at  $f_{DTS}$  frequency) with the new level are detected, we can validate the transition on TI1. Then configure TIMx\_CCMOD1. IC1F to '0011'.

- By configuring TIMx\_CCEN.CC1P=0, select the rising edge as the valid transition polarity on the TI1 channel.
- Configure the input prescaler. In this example, configure TIMx\_CCMOD1.IC1PSC= '00' to disable the prescaler because we want to capture every valid transition.
- Enable capture by configuring TIMx\_CCEN.CC1EN = '1'.

If you want to enable DMA request, you can configure TIMx\_DINTEN.CC1DEN=1. If you want enable related interrupt request, you can configure TIMx\_DINTEN.CC1IEN bit=1

### 10.3.6 PWM input mode

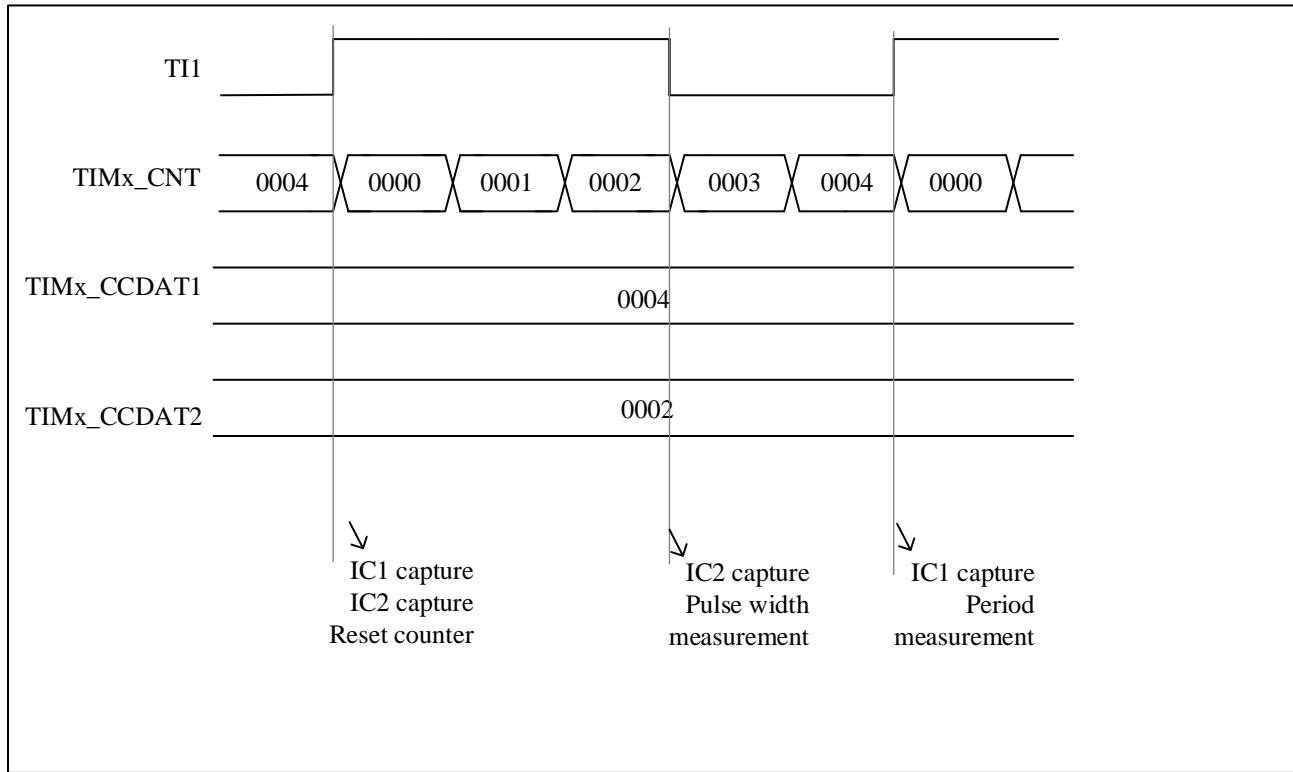
There are some differences between PWM input mode and normal input capture mode, including:

- Two ICx signals are mapped to the same TIx input.
- The two ICx signals are active on edges of opposite polarity.
- Select one of two TIxFP signals as trigger input.
- The slave mode controller is configured in reset mode.

For example, the following configuration flow can be used to know the period and duty cycle of the PWM signal on TI1 (It depends on the frequency of CK\_INT and the value of the prescaler).

- Configure TIMx\_CCMOD1.CC1SEL equal to '01' to select TI1 as valid input for TIMx\_CCDAT1.
- Configure TIMx\_CCEN.CC1P equal to '0' to select the active polarity of filtered timer input 1(TI1FP1), valid on the rising edge.
- Configure TIMx\_CCMOD1.CC2SEL equal to '10' select TI1 as valid input for TIMx\_CCDAT2.
- Configure TIMx\_CCEN.CC2P equal to 1 to select the valid polarity of filtered timer input 2(TI1FP2), valid on the falling edge.
- Configure TIMx\_SMCTRL.TSEL=101 to select Filtered timer input 1 (TI1FP1) as valid trigger input.
- Configure TIMx\_SMCTRL.SMSEL=100 to configure the slave mode controller to reset mode.
- Configure TIMx\_CCEN.CC1EN=1 and TIMx\_CCEN.CC2EN=1 to enable capture.

Figure 10-16 PWM input mode timing



Because of only filter timer input 1 (TI1FP1) and filter timer input 2 (TI2FP2) are connected to the slave mode controller, the PWM input mode can only be used with the TIMx\_CH1/TIMx\_CH2 signals.

### 10.3.7 Forced output mode

Software can force output compare signals to active or inactive level directly, in output mode (TIMx\_CCMODx.CCxSEL=00).

User can set TIMx\_CCMODx. OCxMD=101 to force the output compare signal to active level. And the OCxREF will be forced high, OCx get opposite value to CCxP polarity bit. On the other hand, user can set TIMx\_CCMODx. OCxMD=100 to force the output compare signal to inactive level.

The values of the TIMx\_CCDatax shadow register and the counter still comparing with each other in this mode. And the flag still can be set. Therefore, the interrupt and DMA requests still can be sent.

The comparison between the output compare register TIMx\_CCDatax and the counter TIMx\_CNT has no effect on OCxREF. And the flag still can be set. Therefore, the interrupt and DMA requests still can be sent.

### 10.3.8 Output compare mode

User can use this mode to control the output waveform, or to indicate that a period of time has elapsed.

When the capture/compare register and the counter have the same value, the output compare function's operations are as follow:

- TIMx\_CCMODx.OCxMD is for output compare mode, and TIMx\_CCEN.CCxP is for output polarity. When the compare matches, if set TIMx\_CCMODx.OCxMD=000, the output pin will keep its level; if set TIMx\_CCMODx.OCxMD=001, the output pin will be set active; if set TIMx\_CCMODx.OCxMD=010, the output pin will be set inactive; if set TIMx\_CCMODx.OCxMD=011, the output pin will be set to toggle.
- Set TIMx\_STS.CCxITF.
- If user set TIMx\_DINTEN.CCxIEN, a corresponding interrupt will be generated.
- If user set TIMx\_DINTEN.CCxDEN and set TIMx\_CTRL2.CCDSEL to select DMA request, and DMA request will be sent.

User can set TIMx\_CCMODx.OCxPEN to choose capture/compare shadow register using capture/compare preload registers(TIMx\_CCDATx) or not.

The time resolution is one count of the counter.

In one pulse mode, the output compare mode can also be used to output a single pulse.

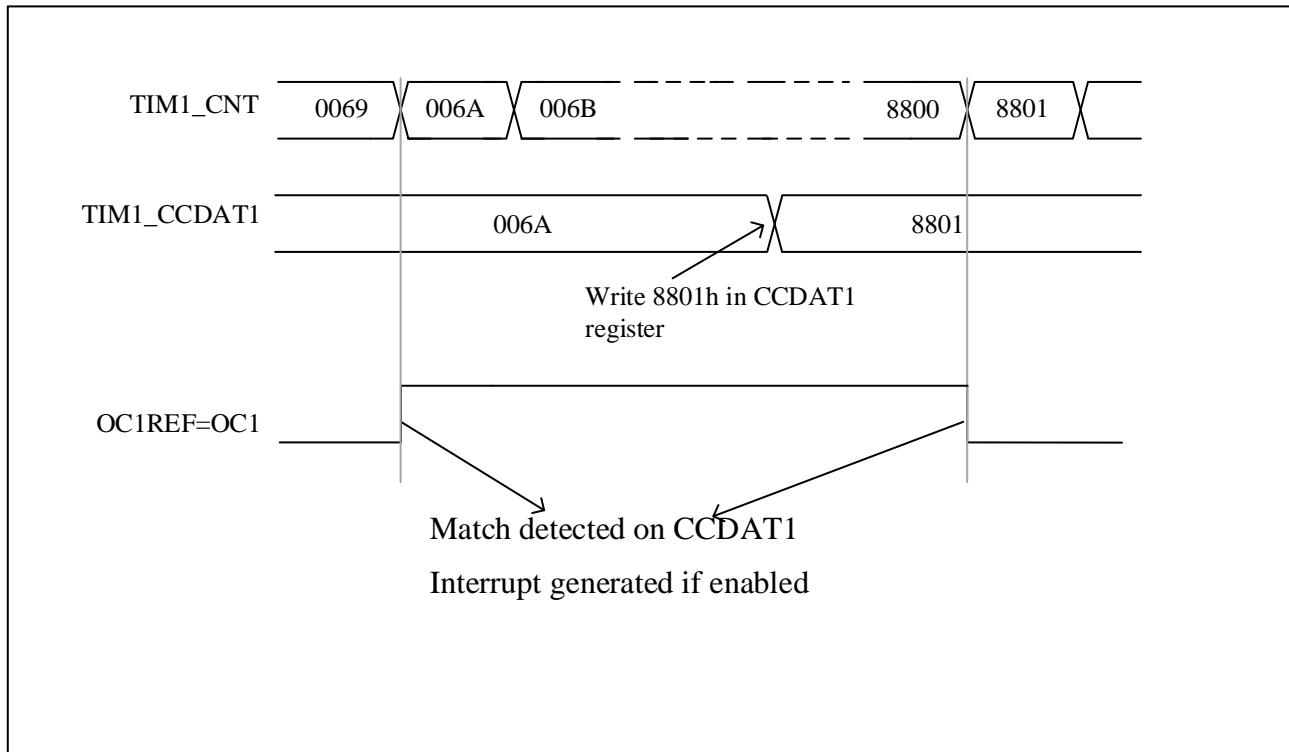
Here are the configuration steps for output compare mode:

- First of all, user should select the counter clock.
- Secondly, set TIMx\_AR and TIMx\_CCDATx with desired data.
- If user need to generate an interrupt, set TIMx\_DINTEN.CCxIEN.
- Then select the output mode by set TIMx\_CCEN.CCxP, TIMx\_CCMODx.OCxMD, TIMx\_CCEN.CCxEN, etc.
- At last, set TIMx\_CTRL1.CNTEN to enable the counter.

User can update the output waveform by setting TIMx\_CCDATx at any time, as long as the preload register is not enabled. Otherwise the TIMx\_CCDATx shadow register will be updated at the next update event.

Here is an example.

Figure 10-17 Output compare mode, toggle on OC1



### 10.3.9 PWM mode

User can use PWM mode to generate a signal whose duty cycle is determined by the value of the TIMx\_CC DATx register and whose frequency is determined by the value of the TIMx\_AR register. And depends on the value of TIMx\_CTRL1.CAMSEL, the TIM can generate PWM signal in edge-aligned mode or center-aligned mode.

User can set PWM mode 1 or PWM mode 2 by setting TIMx\_CCMODx. OCxMD=110 or setting TIMx\_CCMODx. OCxMD=111. To enable preload register, user must set corresponding TIMx\_CCMODx. OCxPEN. And then set TIMx\_CTRL1.AR PEN to auto-reload preload register eventually.

User can set polarity of OCx by setting TIMx\_CCEN.CCXP. To enable the output of OCx, user need to set the combination of the value of CCXEN.

The values of TIMx\_CNT and TIMx\_CC DATx are always compared with each other when the TIM is under PWM mode.

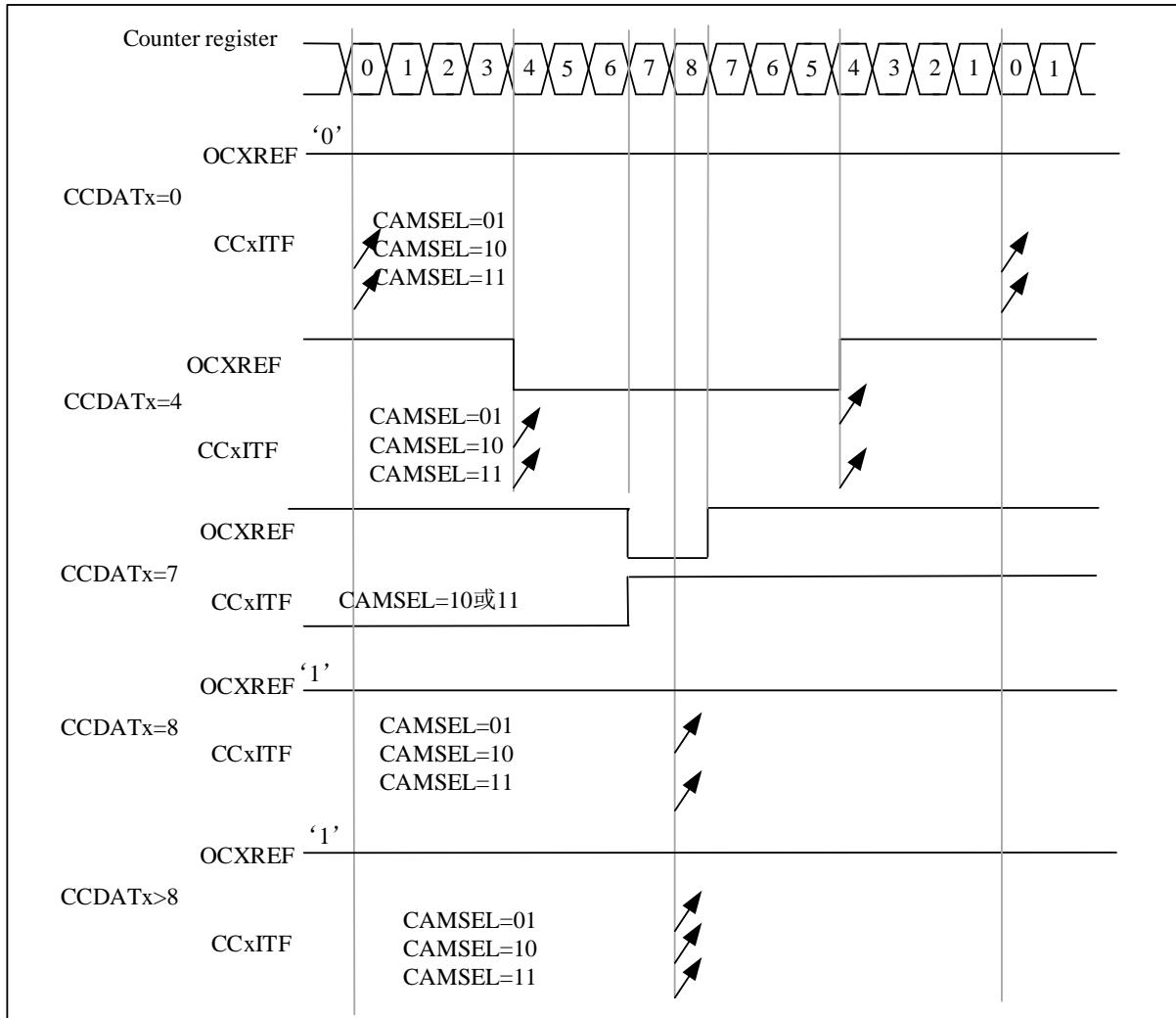
Only if an update event occurs, the preload register will transfer to the shadow register. Therefore user must reset all the registers by setting TIMx\_EVTGEN.UDGN before the counter starts counting.

#### 10.3.9.1 PWM center-aligned mode

If user set TIMx\_CTRL1.CAMSEL equal 01, 10 or 11, the PWM center-aligned mode will be active. The setting of the compare flag depends on the value of TIMx\_CTRL1.CAMSEL. There are three kinds of situation that the compare flag is set, only when the counter counts up, only when the counter counts down, or when the counter counts up and counts down. User should not modified TIMx\_CTRL1.DIR by software, it is updated by hardware.

Examples of center-aligned PWM waveforms is as follow, and the setting of the waveform are: TIMx\_AR=8, PWM mode 1, the compare flag is set when the counter counts down corresponding to TIMx\_CTRL1.CAMSEL=01.

Figure 10-18 Center-aligned PWM waveform (AR=8)



Notes that user should know when using center-aligned mode are as follow:

- It depends on the value of TIMx\_CTRL1.DIR that the counter counts up or down. Cautions that the DIR and CAMSEL bits should not be changed at the same time.
- User should not write the counter while running in center-aligned mode, otherwise it will cause unexpected results. Here are some example:
  - If the value written into the counter is 0 or is the value of TIMx\_AR, the direction will be updated but the update event will not be generated.
  - If the value written into the counter is greater than the value of auto-reload, the direction will not be updated.
- To be on the safe side, user is suggested setting TIMx\_EVTGEN.UDGN to generate an update by software before starting the counter, and not writing the counter while it is running.

### 10.3.9.2 PWM edge-aligned mode

There are two kinds of configuration in edge-aligned mode, up-counting and down-counting.

- **Up-counting**

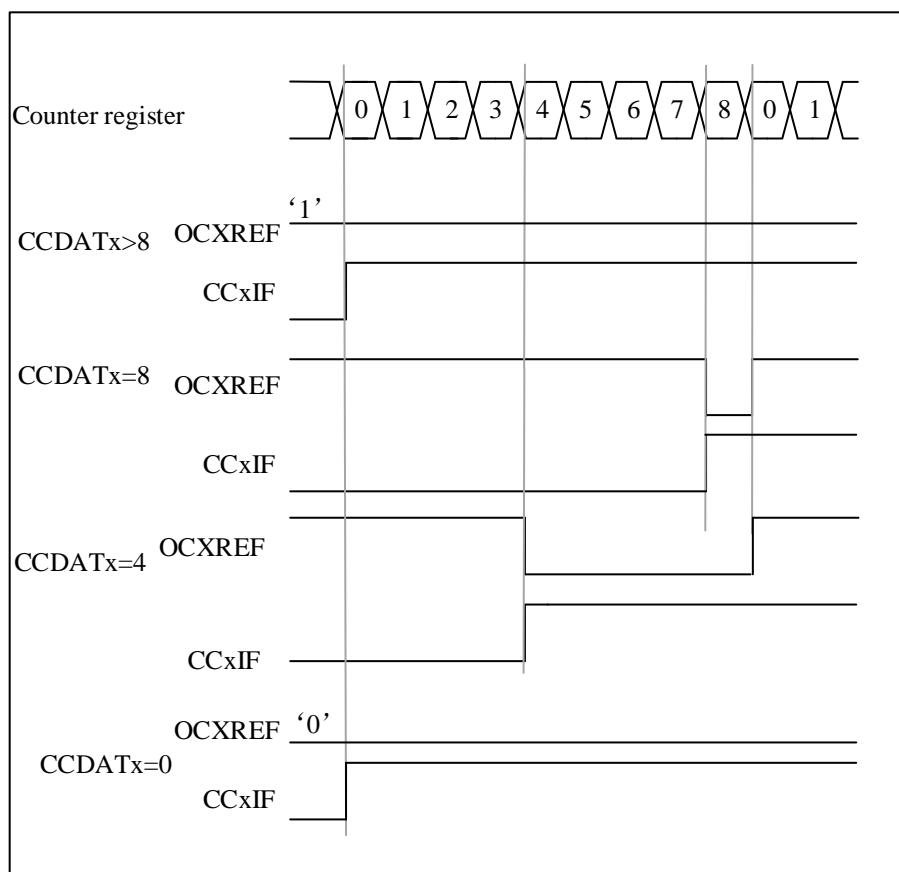
User can set `TIMx_CTRL1.DIR=0` to make counter counts up.

Here is an example for PWM mode1.

When  $\text{TIMx\_CNT} < \text{TIMx\_CCDATx}$ , the reference PWM signal `OCxREF` is high. Otherwise it will be low. If the compare value in `TIMx_CCDATx` is greater than the auto-reload value, the `OCxREF` will remains 1. Conversely, if the compare value is 0, the `OCxREF` will remains 0.

When  $\text{TIMx\_AR}=8$ , the PWM waveforms are as follow.

Figure 10-19 Edge-aligned PWM waveform (APR=8)



- **Down-counting**

User can set `TIMx_CTRL1.DIR=1` to make counter counts down.

Here is an example for PWM mode1.

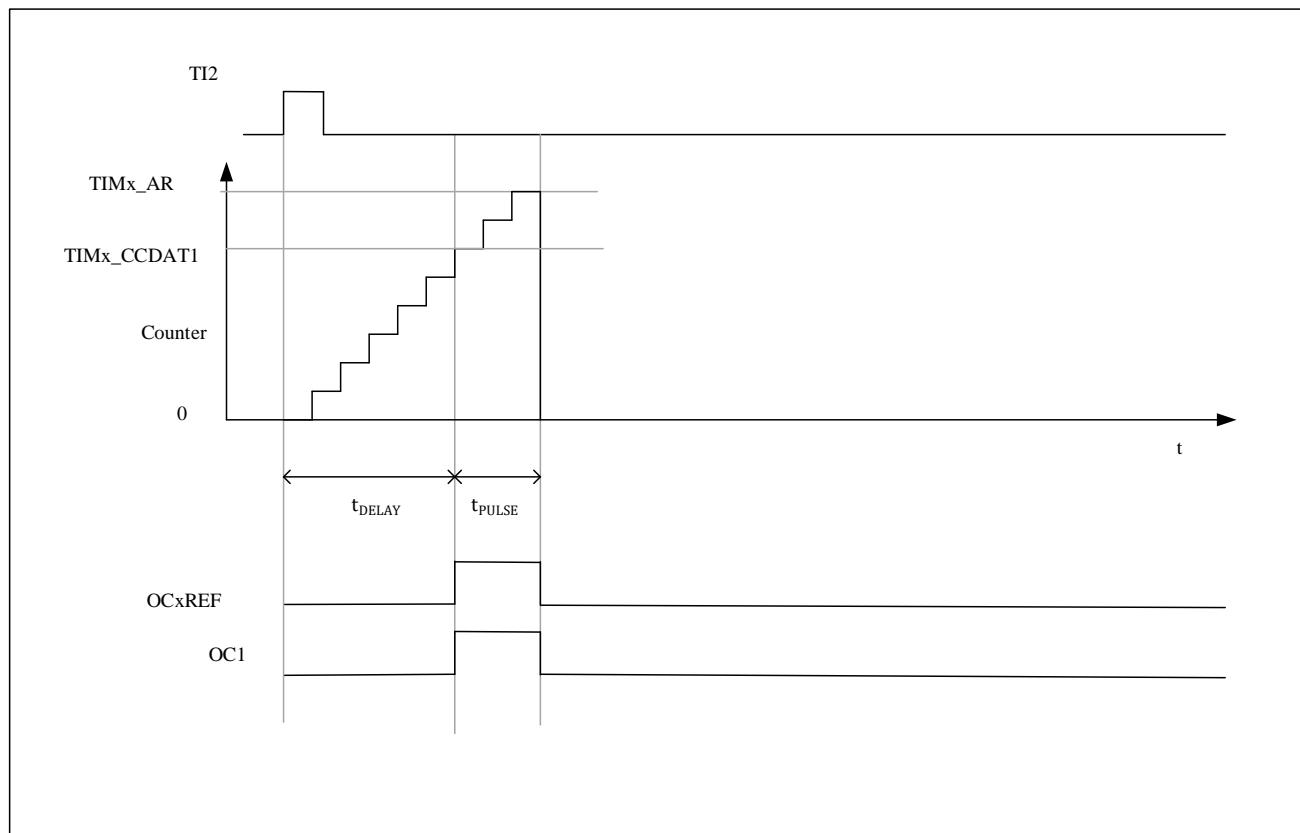
When  $\text{TIMx\_CNT} > \text{TIMx\_CCDATx}$ , the reference PWM signal `OCxREF` is low. Otherwise it will be high. If the compare value in `TIMx_CCDATx` is greater than the auto-reload value, the `OCxREF` will remains 1.

Note: If the nth PWM cycle  $CCDATx$  shadow register  $\geq AR$  value, the shadow register value of  $CCDATx$  in the  $(n+1)$ th PWM cycle is 0. At the moment when the counter is 0 in the  $(n+1)$ th PWM cycle, although the value of the counter =  $CCDATx$  shadow register = 0 and  $OCxREF$  = '0', no compare event will be generated.

### 10.3.10 One-pulse mode

In the one-pulse mode (ONEPM), a trigger signal is received, and a pulse  $t_{PULSE}$  with a controllable pulse width is generated after a controllable delay  $t_{DELAY}$ . The output mode needs to be configured as output compare mode or PWM mode. After selecting one-pulse mode, the counter will stop counting after the update event UEV is generated.

Figure 10-20 Example of One-pulse mode



The following is an example of a one-pulse mode:

A rising edge trigger is detected from the TI2 input, and a pulse with a width of  $t_{PULSE}$  is generated on OC1 after a delay of  $t_{DELAY}$ .

1. Counter configuration: count up, counter  $TIMx\_CNT < TIMx\_CCDAT1 \leq TIMx\_AR$ ;
2. TI2FP2 is mapped to TI2,  $TIMx\_CCMOD1.CC2SEL = '01'$ ; TI2FP2 is configured for rising edge detection,  $TIMx\_CCEN.CC2P = '0'$ ;
3. TI2FP2 acts as the trigger (TRGI) of the slave mode controller and starts the counter,  $TIMx\_SMCTRL.TSEL = '110'$ ,  $TIMx\_SMCTRL.SMSEL = '110'$  (trigger mode);
4.  $TIMx\_CCDAT1$  writes the count value to be delayed ( $t_{DELAY}$ ),  $TIMx\_AR - TIMx\_CCDAT1$  is the count value of

the pulse width  $t_{PULSE}$ ;

5. Configure TIMx\_CTRL1.ONEPM=1 to enable single pulse mode, configure TIMx\_CCMOD1.OC1MD = '111' to select PWM2 mode;

6. Wait for an external trigger event on TI2, and a one pulse waveform will be output on OC1;

#### 10.3.10.1 Special case: OCx fast enable:

In one-pulse mode, an edge is detected through the TIx input, and triggers the start of the counter to count to the comparison value and then output a pulse. These operations limit the minimum delay  $t_{DELAY}$  that can be achieved.

You can set TIMx\_CCMODx.OCxFEN=1 to turn on OCx fast enable, after triggering the rising edge, the OCxREF signal will be forced to be converted to the same level as the comparison match occurs immediately, regardless of the comparison result. OCxFEN fast enable only takes effect when the channel mode is configured for PWM1 and PWM2 modes.

#### 10.3.11 Clearing the OCxREF signal on an external event

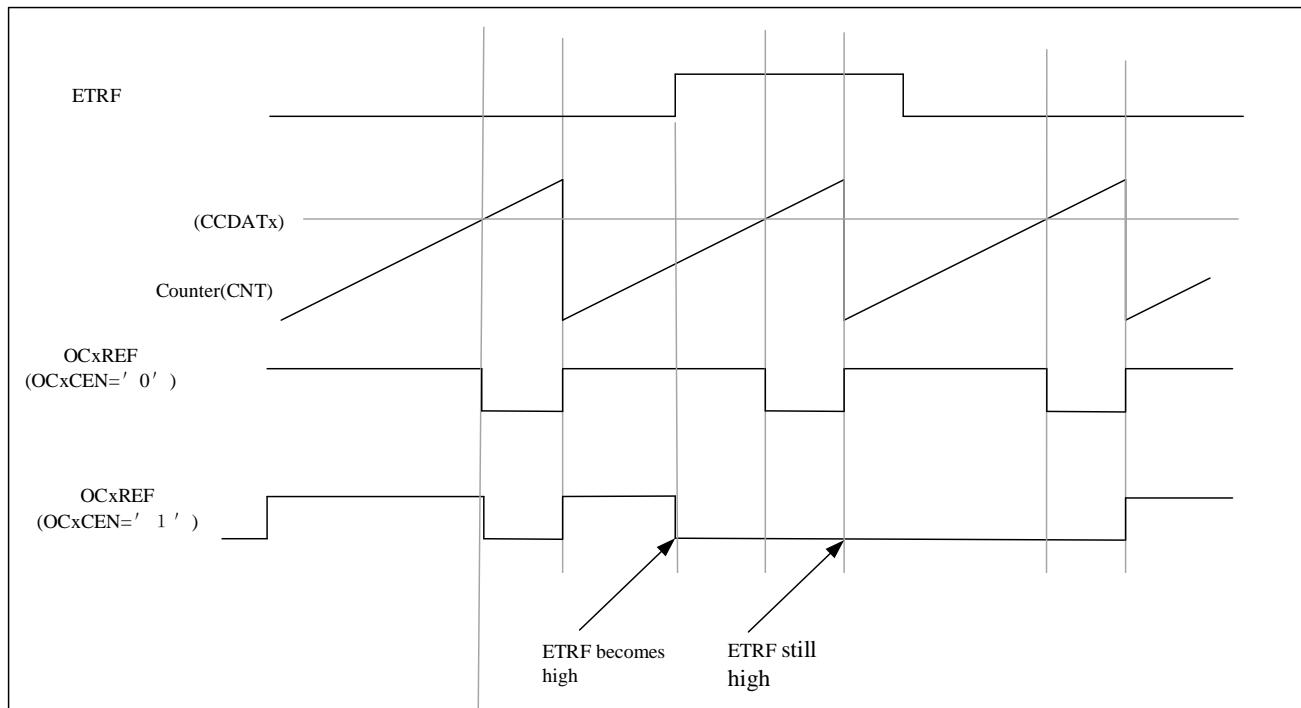
If user set TIMx\_CCMODx.OCxCEN=1, high level of ETRF input can be used to driven the OCxREF signal to low, and the OCxREF signal will remains low, until the next UEV happens. Only output compare and PWM modes can use this function. This cannot be used when it is in forced mode.

Here is an example for it. To control the current, user can connect the ETR signal to the output of a comparator, and the operation for ETR should be as follow:

- Set TIMx\_SMCTRL.EXTPS=00 to disable the external trigger prescaler.
- Set TIMx\_SMCTRL.EXCEN=0 to disable the external clock mode 2.
- Set TIMx\_SMCTRL.EXTP and TIMx\_SMCTRL.EXTF to configure the external trigger polarity and external trigger filter according to the need.

Here is an example for the case that when ETRF input becomes high, the behavior of OCxREF signal for different value of OCxCEN. Timer is set to be in PWM mode in this case.

Figure 10-21 Control circuit in reset mode



### 10.3.12 Debug mode

When the microcontroller is in debug mode (the Cortex-M0 core halted), depending on the `DBG_CTRL.TIMx_STOP` configuration in the PWR module, the TIMx counter can either continue to work normally or stop. For more details, see 3.3.2.

### 10.3.13 TIMx and external trigger synchronization

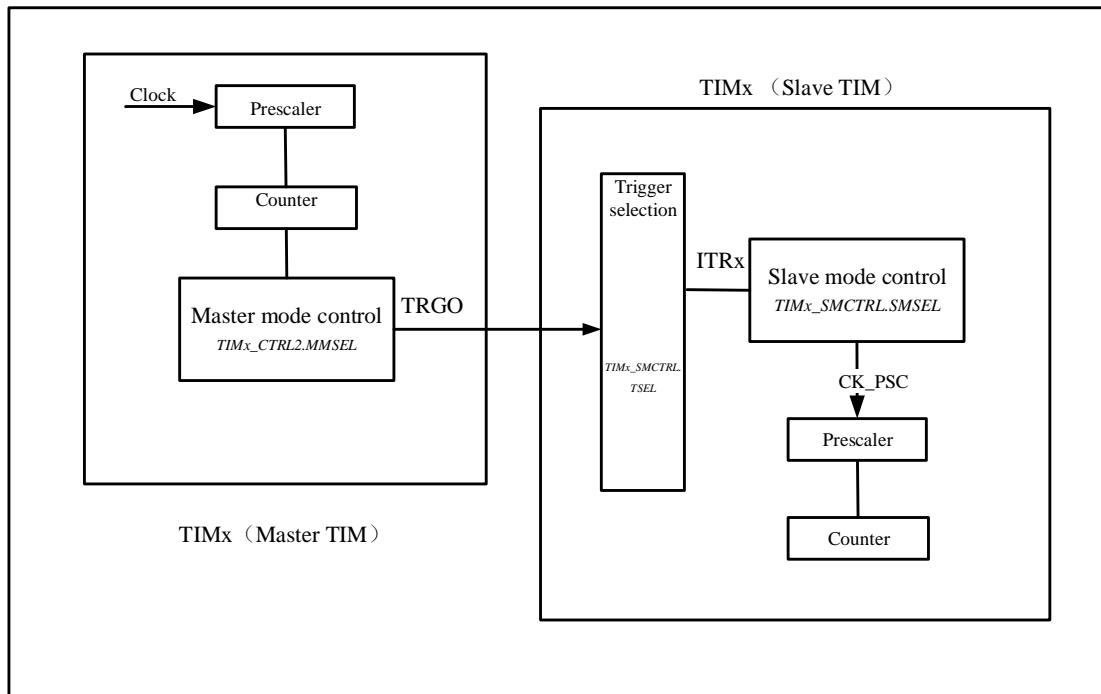
Same with advanced-control timer, see 9.3.16

### 10.3.14 Timer synchronization

All TIMx timers are internally connected to each other. This implementation allows any master timer to provide trigger to reset, start, stop or provide a clock for the other slave timers. The master clock is used for internal counter and can be prescaled. Below figure shows a Block diagram of timer interconnection.

The synchronization function does not support dynamic change of the connection. User should configure and enable the slave timer before enable the master timer's trigger or clock.

Figure 10-22 Block diagram of timer interconnection



#### 10.3.14.1 Master timer as a prescaler for another timer

TIM1 as a prescaler for TIM3. TIM1 is master, TIM3 is slave.

User need to do the following steps for this configuration.

- Setting TIM1\_CTRL2.MMSEL='010' to use the update event of TIM1 as trigger output.
- Configure TIM3\_SMCTRL.TSEL='000', connect the TRGO of TIM1 to TIM3.
- Configure TIM3\_SMCTRL.SMSEL = '111', the slave mode controller will be configured in external clock mode 1.
- Start TIM3 by setting TIM3\_CTRL1.CNTEN = '1'.
- Start TIM1 by setting TIM1\_CTRL1.CNTEN = '1'.

*Note: If user select OCx as the trigger output of TIM1 by configuring MMSEL = '1xx', OCx rising edge will be used to drive TIM2.*

#### 10.3.14.2 Master timer to enable another timer

In this example, TIM3 is enabled by the output compare of TIM1. TIM3 counter will start to count after the OC1REF output from TIM1 is high. Both counters are clocked based on CK\_INT via a prescaler divide by 3 is performed ( $f_{CK\_CNT} = f_{CK\_INT}/3$ ).

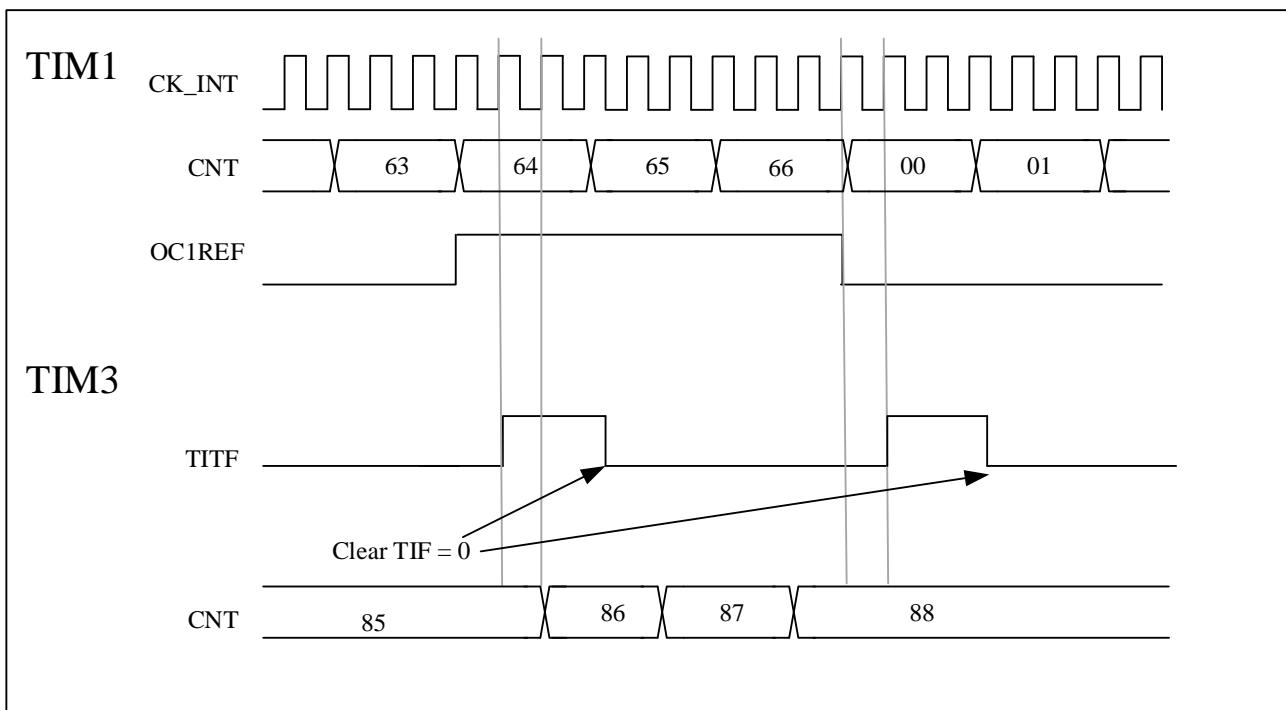
The configuration steps are shown as below.

- Setting TIM1\_CTRL2.MMSEL='100' to use the OC1REF of TIM1 as trigger output.

- Configure TIM1\_CCMOD1 register to configure the OC1REF output waveform.
- Setting TIM3\_SMCTRL.TSEL = '000' to connect TIM1 trigger output to TIM3.
- Setting TIM3\_SMCTRL.SMSEL = '101' to set TIM3 to gated mode.
- Setting TIM3\_CTRL1.CNTEN = '1' to start TIM3.
- Setting TIM1\_CTRL1.CNTEN = '1' to start TIM1.

*Note: The TIM3 clock is not synchronized with the TIM1 clock, this mode only affects the TIM3 counter enable signal.*

Figure 10-23 TIM3 gated by OC1REF of TIM1

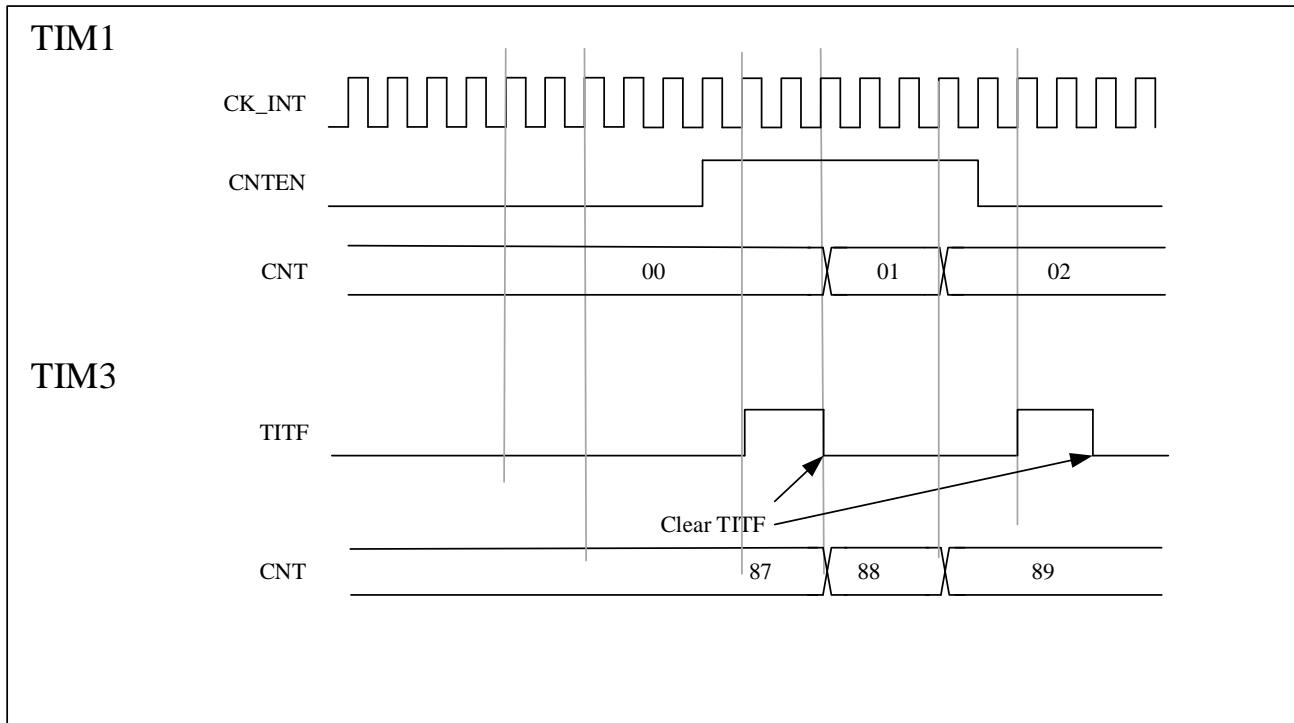


In the next example, Gated TIM3 with enable signal of TIM1, Setting TIM1 CTRL1.CNTEN = '0' to stop TIM1. TIM3 counts on the divided internal clock only when TIM1 is enable. Both counters are clocked based on CK\_INT via a prescaler divide by 3 is performed ( $f_{CK\_CNT} = f_{CK\_INT}/3$ ).

The configuration steps are shown as below

- Setting TIM1\_CTRL2.MMSEL='001' to use the enable signal of TIM1 as trigger output
- Setting TIM3\_SMCTRL.TSEL = '000' to configure TIM3 to get the trigger input from TIM1
- Setting TIM3\_SMCTRL.SMSEL = '101' to configure TIM3 in gated mode.
- Setting TIM3\_CTRL1.CNTEN = '1' to start TIM3.
- Setting TIM1\_CTRL1.CNTEN = '1' to start TIM1.
- Setting TIM1\_CTRL1.CNTEN = '0' to stop TIM1.

Figure 10-24 TIM3 gated by enable signal of TIM1



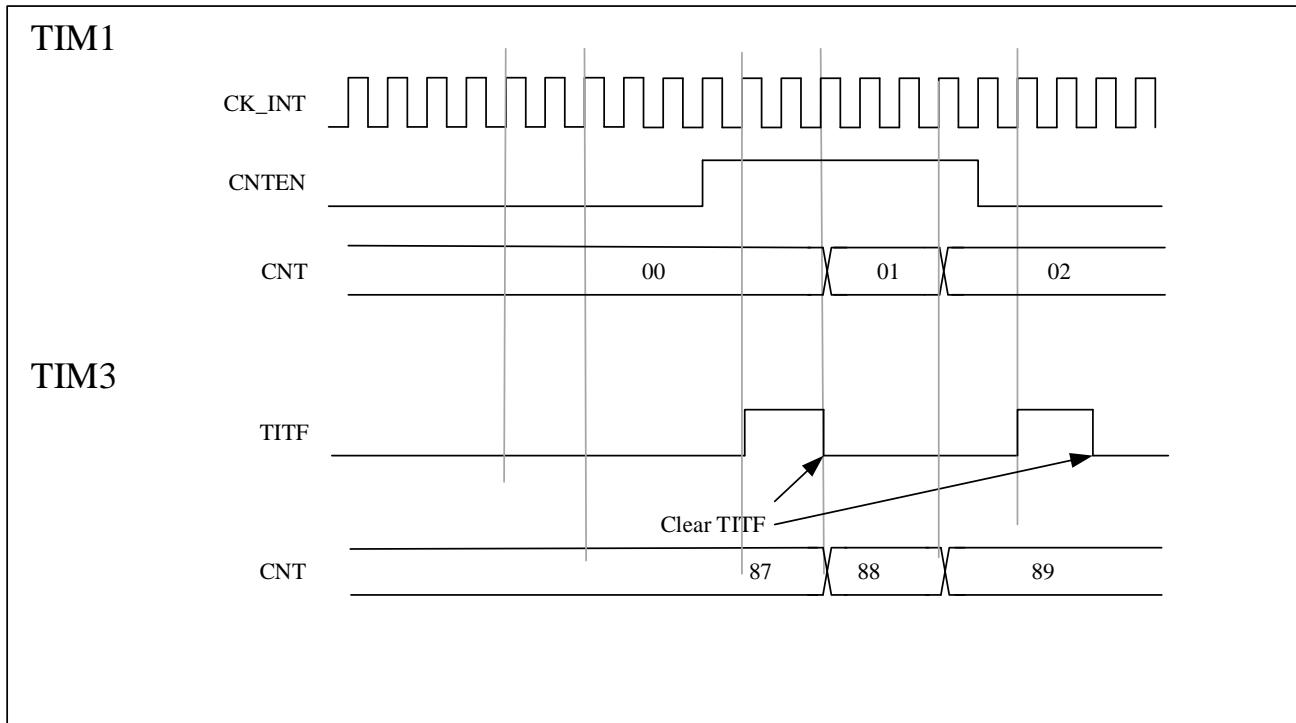
### 10.3.14.3 Master timer to start another timer

In this example, we can use update event as trigger source. TIM1 is master, TIM3 is slave.

The configuration steps are shown as below:

- Setting TIM1\_CTRL2.MMSEL='010' to use the update event of TIM1 as trigger output
- Configure TIM1\_AR register to set the output period.
- Setting TIM3\_SMCTRL.TSEL='000' to connect TIM1 trigger output to TIM3.
- Setting TIM3\_SMCTRL.SMSEL='110' to set TIM3 to trigger mode.
- Setting TIM1\_CTRL1.CNTEN=1 to start TIM1.

Figure 10-25 Trigger TIM3 with an update of TIM1



#### 10.3.14.4 Start 2 timers synchronously using an external trigger

In this example, TIM1 is enabled when TIM1's TI1 input rises, and TIM3 is enabled when TIM1 is enabled. To ensure the alignment of counters, TIM1 must be configured in master/slave mode. For TI1, TIM1 is the slave; for TIM3, TIM1 is the master.

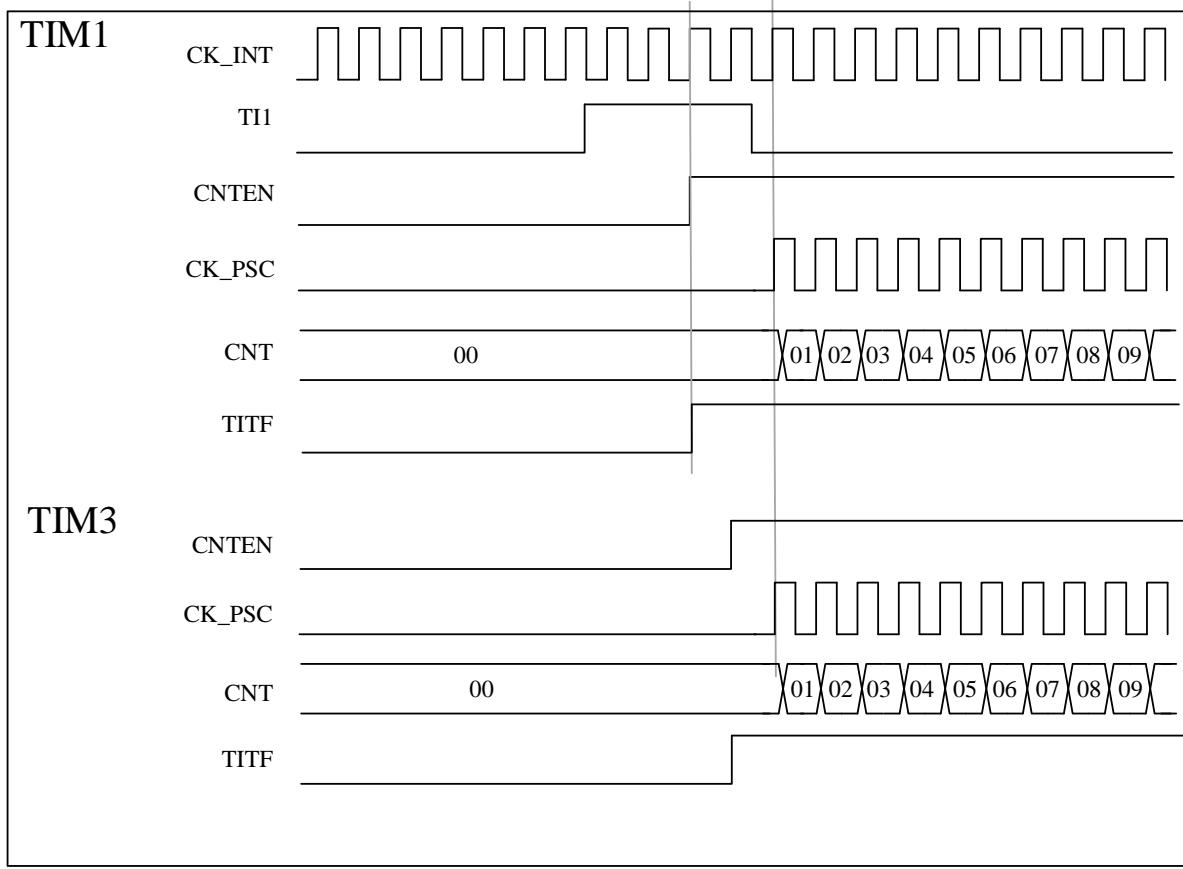
The configuration steps are shown as below:

- Setting TIM1.MMSEL = '001' to use the enable signal as trigger output
- Setting TIM1\_SMCTRL.TSEL = '100' to configure the TIM1 to slave mode and receive the trigger input of TI1.
- Setting TIM1\_SMCTRL.SMSEL = '110' to configure TIM1 to trigger mode.
- Setting TIM1\_SMCTRL.MSMD = '1' to configure TIM1 to master/slave mode.
- Setting TIM3\_SMCTRL.TSEL = '000' to connect TIM1 trigger output to TIM3.
- Setting TIM3\_SMCTRL.SMSEL = '110' to configure TIM3 to trigger mode.

When TI1 rising edge arrives, both timers start counting synchronously according to the internal clock, and both TITF flags are set simultaneously.

*The following figure shows a delay between CNTEN and CK\_PSC of TIM1 in master/slave mode.*

Figure 10-26 Triggers timers 1 and 3 using the TI1 input of TIM1



### 10.3.15 Encoder interface mode

The encoder uses two inputs TI1 and TI2 as an interface and the counter counts on every edge change on TI1FP1 or TI2FP2. The counting direction is automatically controlled by hardware TIMx\_CTRL1.DIR. There are three types of encoder counting modes:

1. The counter only counts on the edge of TI1, TIMx\_SMCTRL.SMSEL = '001';
2. The counter only counts on the edge of TI2, TIMx\_SMCTRL.SMSEL = '010';
3. The counter counts on the edges of TI1 and TI2 at the same time, TIMx\_SMCTRL.SMSEL = '011';

The encoder interface is equivalent to using an external clock with direction selection, and the counter only counts continuously between 0 and the auto-reload value (TIMx\_AR.AR [15:0]). Therefore, it is necessary to configure the auto-reload register TIMx\_AR in advance.

*Note: Encoder mode and external clock mode 2 are not compatible and must not be selected together.*

The relationship between the counting direction and the encoder signal is shown in **Table 10-1**:

Table 10-1 Counting direction versus encoder signals

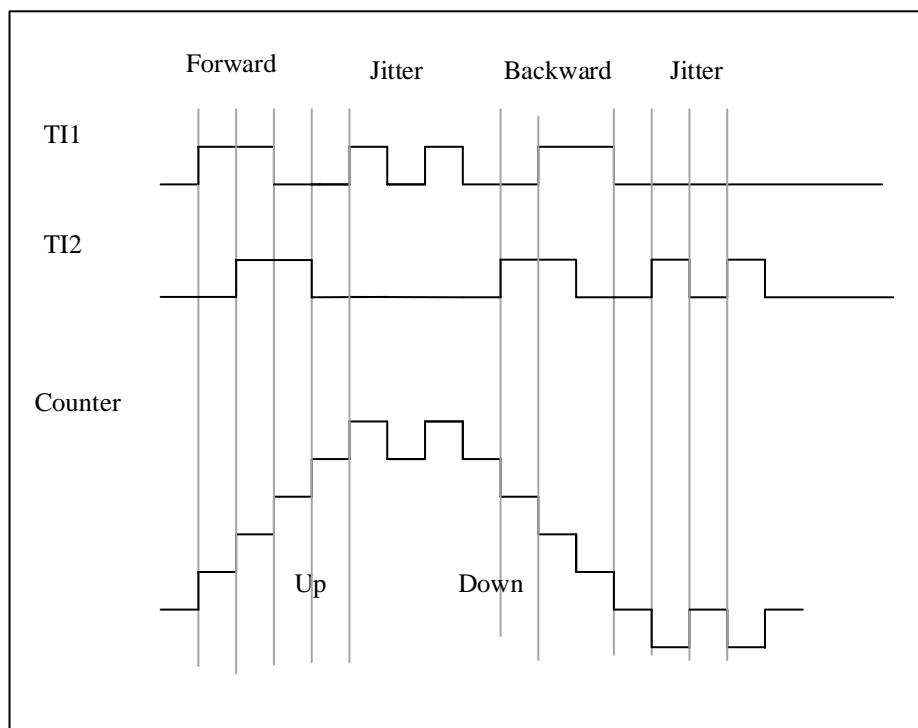
	Level on opposite signals	TI1FP1 signal	TI2FP2 signal
--	---------------------------	---------------	---------------

Active edge	(TI1FP1 for TI2, TI2FP2 for TI1)	Rising	Falling	Rising	Falling
Counting only at TI1	High	Counting down	Counting up	Don't count	Don't count
	Low	Counting up	Counting down	Don't count	Don't count
Counting only at TI2	High	Don't count	Don't count	Counting up	Counting down
	Low	Don't count	Don't count	Counting down	Counting up
Counting on TI1 and TI2	High	Counting down	Counting up	Counting up	Counting down
	Low	Counting up	Counting down	Counting down	Counting up

Here is an example of an encoder with dual edge triggering selected to suppress input jitter:

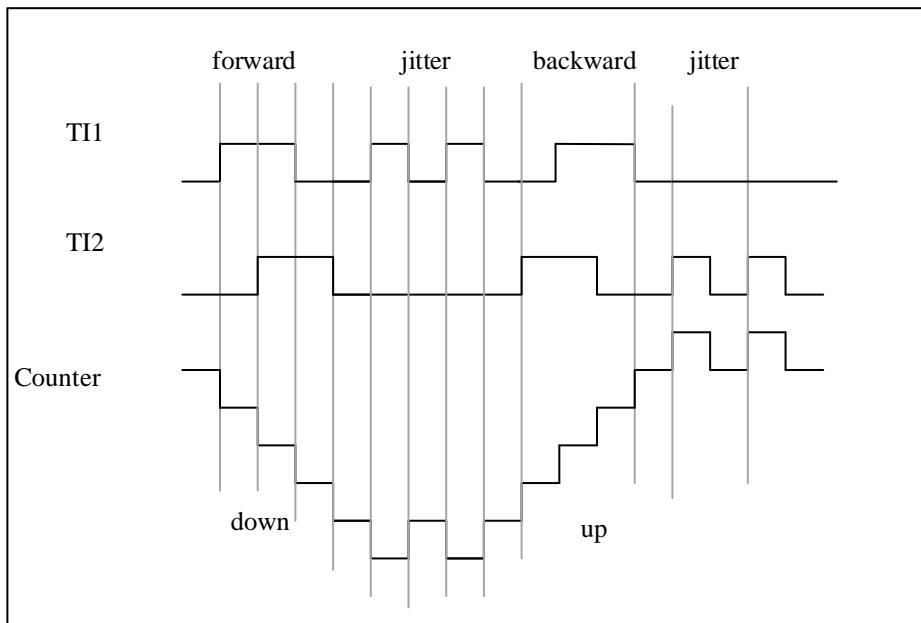
1. IC1FP1 is mapped to TI1 (TIMx\_CCMOD1.CC1SEL= '01'), IC1FP1 is not inverted (TIMx\_CCEN.CC1P= '0');
2. IC1FP2 is mapped to TI2 (TIMx\_CCMOD2.CC2SEL= '01'), IC2FP2 is not inverted (TIMx\_CCEN.CC2P= '0');
3. The input is valid on both rising and falling edges (TIMx\_SMCTRL.SMSEL = '011');
4. Enable counter TIMx\_CTRL1.CNTEN= '1';

Figure 10-27 Example of counter operation in encoder interface mode



The following figure shows the example of counter behavior when IC1FP1 polarity is inverted (CC1P= '1', other configurations are the same as above)

Figure 9-43 Encoder interface mode example with IC1FP1 polarity inverted



### 10.3.16 Interfacing with Hall sensor

Please refer to 9.3.20

## 10.4 TIMx register description(x=3)

For abbreviations used in registers, see section 1.1

These peripheral registers can be operated as half word (16-bits) or one word (32-bits).

### 10.4.1 Register Overview

Table 10-2 Register overview

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0												
000h	TIMx_CTRL1																0	CLRSFL	Reserved	C3SEL	C2SEL	C1SEL	Reserved	CLKD[1:0]	ARPEN	CAMSFL[1:0]	DIR	ONPM	URPS	UPDIS	CNTEN														
	Reset Value																0	EXTP	EXCEN	EXPSL[0]	ETRSEL	TISEL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
004h	TIMx_CTRL2																																												
	Reset Value																																												
008h	TIMx_SMCTRL																																												
	Reset Value																																												



Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
04Ch	TIMx_DADDR																																		
	Reset Value	Reserved												BURST[15:0]												0	0	0	0	0	0	0	0	0	0

## 10.4.2 Control register 1 (TIMx\_CTRL1)

Offset address: 0x00

Reset value: 0x0000 0000

31																																	16
15	CLRSEL	Reserved	C3SEL	C2SEL	C1SEL	Reserved	CLKD[1:0]	ARPEN	CAMSEL[1:0]	DIR	ONEPM	UPRS	UPDIS	CNTEN																			
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw																			

Bit field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained
15	CLRSEL	OCxREF clear selection 0: Select the external OCxREF clear from ETR 1: Select the internal OCxREF clear from comparator
14	Reserved	Reserved, the reset value must be maintained
13	C3SEL	Channel 3 Selection 0: Select external CH3 (from IOM) signal 1: Reserved
12	C2SEL	Channel 2 Selection 0: Select external CH2 (from IOM) signal 1: Reserved
11	C1SEL	Channel 1 selection 0: Select external CH1 signal from IOM 1: Select internal CH1 signal from COMP
10	Reserved	Reserved, the reset value must be maintained
9:8	CLKD[1:0]	Clock division CLKD[1:0] indicates the division ratio between CK_INT (timer clock) and tDTS (clock used for dead-time generator and digital filters (ETR, TIx)) 00: tDTS = tCK_INT 01: tDTS = 2 × tCK_INT 10: tDTS = 4 × tCK_INT 11: Reserved, do not use this configuration
7	ARPEN	ARPEN: Auto-reload preload enable 0: Shadow register disable for TIMx_AR register 1: Shadow register enable for TIMx_AR register

Bit field	Name	Description
6:5	CAMSEL[1:0]	<p>Center-aligned mode selection</p> <p>00: Edge-aligned mode. TIMx_CTRL1.DIR specifies up-counting or down-counting.</p> <p>01: Center-aligned mode 1. The counter counts in center-aligned mode, and the output compare interrupt flag bit is set to 1 when down-counting.</p> <p>10: Center-aligned mode 2. The counter counts in center-aligned mode, and the output compare interrupt flag bit is set to 1 when up-counting.</p> <p>11: Center-aligned mode 3. The counter counts in center-aligned mode, and the output compare interrupt flag bit is set to 1 when up-counting or down-counting.</p> <p><i>Note: Switching from edge-aligned mode to center-aligned mode is not allowed when the counter is still enabled (TIMx_CTRL1.CNTEN = 1).</i></p>
4	DIR	<p>Direction</p> <p>0: Up-counting</p> <p>1: Down-counting</p> <p><i>Note: This bit is read-only when the counter is configured in center-aligned mode or encoder mode.</i></p>
3	ONEPM	<p>One-pulse mode</p> <p>0: Disable one-pulse mode, the counter counts are not affected when an update event occurs.</p> <p>1: Enable one-pulse mode, the counter stops counting when the next update event occurs (clearing TIMx_CTRL1.CNTEN bit)</p>
2	UPRS	<p>Update request source</p> <p>This bit is used to select the UEV event sources by software.</p> <p>0: If update interrupt or DMA request is enabled, any of the following events will generate an update interrupt or DMA request:</p> <ul style="list-style-type: none"> <li>– Counter overflow/underflow</li> <li>– The TIMx_EVTGEN.UDGN bit is set</li> <li>– Update generation from the slave mode controller</li> </ul> <p>1: If update interrupt or DMA request is enabled, only counter overflow/underflow will generate update interrupt or DMA request</p>
1	UPDIS	<p>Update disable</p> <p>This bit is used to enable/disable the Update event (UEV) events generation by software.</p> <p>0: Enable UEV. And UEV will be generated if one of following condition been fulfilled:</p> <ul style="list-style-type: none"> <li>– Counter overflow/underflow</li> <li>– The TIMx_EVTGEN.UDGN bit is set</li> <li>– Update generation from the slave mode controller</li> </ul> <p>Shadow registers will update with preload value.</p> <p>1: UEV disabled. No update event is generated, and the shadow registers (AR, PSC, and CC DATx) keep their values. If the TIMx_EVTGEN.UDGN bit is set or a hardware reset is issued by the slave mode controller, the counter and prescaler are reinitialized.</p>
0	CNTEN	<p>Counter Enable</p> <p>0: Disable counter</p> <p>1: Enable counter</p>

Bit field	Name	Description
		<i>Note: external clock, gating mode and encoder mode can only work after TIMx_CTRL1.CNTEN bit is set in the software. Trigger mode can automatically set TIMx_CTRL1.CNTEN bit by hardware.</i>

### 10.4.3 Control register 2 (TIMx\_CTRL2)

Offset address: 0x04

Reset value: 0x0000

15	Reserved	9	8	7	6	4	3	2	0
			ETRSEL	TI1SEL	MMSEL[2:0]	CCDSEL		Reserved	

rw                    rw                    rw                    rw                    rw

Bit field	Name	Description
15:9	Reserved	Reserved, the reset value must be maintained
8	ETRSEL	External Triggered Selection storage (ETR Selection) 0: Select external ETR (from IOM) signal; 1: Reserved
7	TI1SEL	TI1 selection 0: TIMx_CH1 pin connected to TI1 input. 1: TIMx_CH1, TIMx_CH2, and TIMx_CH3 pins are XOR connected to the TI1 input.
6:4	MMSEL[2:0]	Master Mode Selection These 3 bits (TIMx_CTRL2. MMSEL [2:0]) are used to select the synchronization information (TRGO) sent to the slave timer in the master mode. Possible combinations are as follows: 000: Reset -When the TIMx_EVTGEN.UDGN is set or a reset is generated by the slave mode controller, a TRGO pulse occurs. And in the latter case, the signal on TRGO is delayed compared to the actual reset. 001: Enable - The TIMx_CTRL1.CNTEN bit is used as the trigger output (TRGO). Sometimes you need to start multiple timers at the same time or enable slave timer for a period of time. The counter enable signal is set when TIMx_CTRL1.CNTEN bit is set or the trigger input in gated mode is high. 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 description of the TIMx_SMCTRL.MSMD bit). 010: Update - The update event is selected as the trigger output (TRGO). For example, a master timer clock can be used as a slave timer prescaler. 011: Compare pulse - Triggers the output to send a positive pulse (TRGO) when the TIMx_STS.CC1ITF is to be set (even if it is already high), when a capture or a comparison succeeds. 100: Compare - OC1REF signal is used as the trigger output (TRGO). 101: Compare - OC2REF signal is used as the trigger output (TRGO). 110: Compare - OC3REF signal is used as the trigger output (TRGO).

Bit field	Name	Description
		111: Compare - OC4REF signal is used as the trigger output (TRGO).
3	CCDSEL	Capture/compare DMA selection 0: When a CCx event occurs, a DMA request for CCx is sent. 1: When an update event occurs, a DMA request for CCx is sent.
2:0	Reserved	Reserved, the reset value must be maintained

#### 10.4.4 Slave mode control register (TIMx\_SMCTRL)

Offset address: 0x08

Reset value: 0x0000

15	14	13	12	11	EXTF[3:0]	8	7	6	4	3	2	0
rw	rw	rw		rw		rw						

Bit field	Name	Description
15	EXTP	External trigger polarity This bit is used to select whether the trigger operation is to use ETR or the inversion of ETR. 0: ETR active at high level or rising edge. 1: ETR active at low level or falling edge.
14	EXCEN	External clock enable This bit is used to enable external clock mode 2, and the counter is driven by any active edge on the ETRF signal in this mode. 0: External clock mode 2 disable. 1: External clock mode 2 enable. <i>Note 1: When external clock mode 1 and external clock mode 2 are enabled at the same time, the input of the external clock is ETRF.</i> <i>Note 2: The following slave modes can be used simultaneously with external clock mode 2: reset mode, gated mode and trigger mode; However, TRGI cannot connect to ETRF (TIMx_SMCTRL.TSEL ≠ '111').</i> <i>Note 3: Setting the TIMx_SMCTRL.EXCEN bit has the same effect as selecting external clock mode 1 and connecting TRGI to ETRF (TIMx_SMCTRL.SMSEL = 111 and TIMx_SMCTRL.TSEL = 111).</i>
13:12	EXTPS[1:0]	External trigger prescaler The frequency of the external trigger signal ETRP must be at most 1/4 of TIMxCLK frequency. When a faster external clock is input, a prescaler can be used to reduce the frequency of ETRP. 00: Prescaler disable 01: ETRP frequency divided by 2 10: ETRP frequency divided by 4 11: ETRP frequency divided by 8

Bit field	Name	Description
11:8	EXTF[3:0]	<p>External trigger filter</p> <p>These bits are used to define the frequency at which the ETRP signal is sampled and the bandwidth of the ETRP digital filtering. In effect, the digital filter is an event counter that generates a validate output after consecutive N events are recorded.</p> <p>0000: No filter, sampling at <math>f_{DTS}</math>      0001: <math>f_{SAMPLING} = f_{CK\_INT}</math>, <math>N = 2</math>      0010: <math>f_{SAMPLING} = f_{CK\_INT}</math>, <math>N = 4</math>      0011: <math>f_{SAMPLING} = f_{CK\_INT}</math>, <math>N = 8</math>      0100: <math>f_{SAMPLING} = f_{DTS}/2</math>, <math>N = 6</math>      0101: <math>f_{SAMPLING} = f_{DTS}/2</math>, <math>N = 8</math>      0110: <math>f_{SAMPLING} = f_{DTS}/4</math>, <math>N = 6</math>      0111: <math>f_{SAMPLING} = f_{DTS}/4</math>, <math>N = 8</math>      1000: <math>f_{SAMPLING} = f_{DTS}/8</math>, <math>N = 6</math>      1001: <math>f_{SAMPLING} = f_{DTS}/8</math>, <math>N = 8</math>      1010: <math>f_{SAMPLING} = f_{DTS}/16</math>, <math>N = 5</math>      1011: <math>f_{SAMPLING} = f_{DTS}/16</math>, <math>N = 6</math>      1100: <math>f_{SAMPLING} = f_{DTS}/16</math>, <math>N = 8</math>      1101: <math>f_{SAMPLING} = f_{DTS}/32</math>, <math>N = 5</math>      1110: <math>f_{SAMPLING} = f_{DTS}/32</math>, <math>N = 6</math>      1111: <math>f_{SAMPLING} = f_{DTS}/32</math>, <math>N = 8</math></p>
7	MSMD	<p>Master/ Slave mode</p> <p>0: No action      1: Events on the trigger input (TRGI) are delayed to allow a perfect synchronization between the current timer (via TRGO) and its slaves. This is useful when several timers are required to be synchronized to a single external event.</p>
6:4	TSEL[2:0]	<p>Trigger selection</p> <p>These 3 bits are used to select the trigger input of the synchronous counter.</p> <p>000: Internal trigger 0 (ITR0) 100: TI1 edge detector (TI1F_ED)      001: Internal trigger 1 (ITR1) 101: Filtered timer input 1(TI1FP1)      010: Internal trigger 2 (ITR2) 110: Filtered timer input 2 (TI2FP2)      011: Internal trigger 3 (ITR3) 111: External triggered Input (ETRF)</p> <p>For more details on ITRx, see Table 10-3 below.</p> <p><i>Note: These bits must be changed only when not in use (e. g. TIMx_SMCTRL.SMSEL=000) to avoid false edge detection at the transition.</i></p>
3	Reserved	Reserved, the reset value must be maintained
2:0	SMSEL[2:0]	<p>Slave mode selection</p> <p>When an external signal is selected, the active edge of the trigger signal (TRGI) is linked to the selected external input polarity (see input control register and control register description)</p> <p>000: Disable slave mode. If TIMx_CTRL1.CNTEN = 1, the prescaler is driven directly by the internal clock.</p>

Bit field	Name	Description
		<p>001: Encoder mode 1. According to the level of TI2FP2, the counter up-counting or down-counting on the edge of TI1FP1.</p> <p>010: Encoder mode 2. According to the level of TI1FP1, the counter up-counting or down-counting on the edge of TI2FP2.</p> <p>011: Encoder mode 3. According to the input level of another signal, the counter up-counting or down-counting on the edges of TI2FP1 and TI2FP2.</p> <p>100: Reset mode. On the rising edge of the selected trigger input (TRGI), the counter is reinitialized and the shadow register is updated.</p> <p>101: Gated mode. When the trigger input (TRGI) is high, the clock of the counter is enabled. Once the trigger input becomes low, the counter stops counting, but is not reset. In this mode, the start and stop of the counter are controlled.</p> <p>110: Trigger mode. When a rising edge occurs on the trigger input (TRGI), the counter is started but not reset. In this mode, only the start of the counter is controlled.</p> <p>111: External clock mode 1. The counter is clocked by the rising edge of the selected trigger input (TRGI).</p> <p><i>Note: Do not use gated mode if TI1F_ED is selected as the trigger input (TIMx_SMCTRL.TSEL=100). This is because TI1F_ED outputs a pulse for each TI1F transition, whereas gated mode checks the level of the triggered input.</i></p>

Table 10-3 TIMx internal trigger connection

Slave timer	ITR0 (TSEL = 000)	ITR1 (TSEL = 001)	ITR2 (TSEL = 010)	ITR3 (TSEL = 011)
TIM3	TIM1	NA	NA	NA

#### 10.4.5 DMA/Interrupt enable registers (TIMx\_DINTEN)

Offset address: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	TDEN	Reserved	CC4DEN	CC3DEN	CC2DEN	CC1DEN	UDEN	Reserved	TIEN	Reserved	CC4IEN	CC3IEN	CC2IEN	CC1IEN	UIEN

Bit field	Name	Description
15	Reserved	Reserved, the reset value must be maintained
14	TDEN	Trigger DMA request enable 0: Disable trigger DMA request 1: Enable trigger DMA request
13	Reserved	Reserved, the reset value must be maintained

Bit field	Name	Description
12	CC4DEN	Capture/Compare 4 DMA request enable 0: Disable capture/compare 4 DMA request 1: Enable capture/compare 4 DMA request
11	CC3DEN	Capture/Compare 3 DMA request enable 0: Disable capture/compare 3 DMA request 1: Enable capture/compare 3 DMA request
10	CC2DEN	Capture/Compare 2 DMA request enable 0: Disable capture/compare 2 DMA request 1: Enable capture/compare 2 DMA request
9	CC1DEN	Capture/Compare 1 DMA request enable 0: Disable capture/compare 1 DMA request 1: Enable capture/compare 1 DMA request
8	UDEN	Update DMA request enable 0: Disable update DMA request 1: Enable update DMA request
7	Reserved	Reserved, the reset value must be maintained
6	TIEN	Trigger interrupt enable 0: Disable trigger interrupt 1: Enable trigger interrupt
5	Reserved	Reserved, the reset value must be maintained
4	CC4IEN	Capture/Compare 4 interrupt enable 0: Disable capture/compare 4 interrupt 1: Enable capture/compare 4 interrupt
3	CC3IEN	Capture/Compare 3 interrupt enable 0: Disable capture/compare 3 interrupt 1: Enable capture/compare 3 interrupts
2	CC2IEN	Capture/Compare 2 interrupt enable 0: Disable capture/compare 2 interrupt 1: Enables capture/compare 2 interrupts
1	CC1IEN	Capture/Compare 1 interrupt enable 0: Disable capture/compare 1 interrupt 1: Enables capture/comparing 1 interrupt
0	UIEN	Update interrupt enable 0: Disable update interrupt 1: Enables update interrupt

#### 10.4.6 Status registers (TIMx\_STS)

Offset address: 0x10

Reset value: 0x0000

15	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	CC4OCF	CC3OCF	CC2OCF	CC1OCF	Reserved		TITF	Reserved	CC4ITF	CC3ITF	CC2ITF	CC1ITF	UDITF	

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

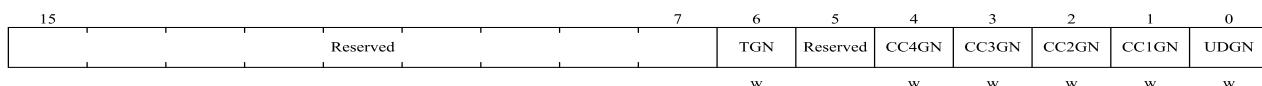
Bit field	Name	Description
15:13	Reserved	Reserved, the reset value must be maintained
12	CC4OCF	Capture/Compare 4 overcapture flag See TIMx_STS.CC1OCF description.
11	CC3OCF	Capture/Compare 3 overcapture flag See TIMx_STS.CC1OCF description.
10	CC2OCF	Capture/Compare 2 overcapture flags See TIMx_STS.CC1OCF description.
9	CC1OCF	Capture/Compare 1 overcapture flag This bit is set by hardware only when the corresponding channel is configured in input capture mode. Cleared by software writing 0. 0: No overcapture occurred 1: TIMx_STS.CC1ITF was already set when the value of the counter has been captured in the TIMx_CCDAT1 register.
8:7	Reserved	Reserved, the reset value must be maintained
6	TITF	Trigger interrupt flag This bit is set by hardware when an active edge is detected on the TRGI input when the slave mode controller is in a mode other than gated. This bit is set by hardware when any edge in gated mode is detected. This bit is cleared by software. 0: No trigger event occurred 1: Trigger interrupt occurred
5	Reserved	Reserved, the reset value must be maintained
4	CC4ITF	Capture/Compare 4 interrupt flag See TIMx_STS.CC1ITF description.
3	CC3ITF	Capture/Compare 3 interrupt flag See TIMx_STS.CC1ITF description.
2	CC2ITF	Capture/Compare 2 interrupt flag See TIMx_STS.CC1ITF description.
1	CC1ITF	Capture/Compare 1 interrupt flag <b>When the corresponding channel of CC1 is in output mode:</b> Except in center-aligned mode, this bit is set by hardware when the counter value is the same as the compare value (see TIMx_CTRL1.CAMSEL bit description). This bit is cleared by software. 0: No match occurred. 1: The value of TIMx_CNT is the same as the value of TIMx_CCDAT1. When the value of TIMx_CCDAT1 is greater than the value of TIMx_AR, the TIMx_STS.CC1ITF bit will go high if the counter overflows (in up-counting and up/down-counting modes) and underflows in down-counting mode.

Bit field	Name	Description
		<p><b>When the corresponding channel of CC1 is in input mode:</b></p> <p>This bit is set by hardware when the capture event occurs. This bit is cleared by software or by reading TIMx_CCDAT1.</p> <p>0: No input capture occurred.</p> <p>1: Input capture occurred. Counter value has captured in the TIMx_CCDAT1. An edge with the same polarity as selected has been detected on IC1.</p>
0	UDITF	<p>Update interrupt flag</p> <p>This bit is set by hardware when an update event occurs under the following conditions:</p> <ul style="list-style-type: none"> <li>- When TIMx_CTRL1.UPDIS = 0, overflow or underflow (An update event is generated).</li> <li>- When TIMx_CTRL1.UPRS = 0, TIMx_CTRL1.UPDIS = 0, and set the TIMx_EVTGEN.UDGN bit by software to reinitialize the CNT.</li> <li>- When TIMx_CTRL1.UPRS = 0, TIMx_CTRL1.UPDIS = 0, and the counter CNT is reinitialized by the trigger event. (See TIMx_SMCTRL Register description)</li> </ul> <p>This bit is cleared by software.</p> <p>0: No update event occurred</p> <p>1: Update interrupt occurred</p>

#### 10.4.7 Event generation registers (TIMx\_EVTGEN)

Offset address: 0x14

Reset values: 0 x0000



Bit field	Name	Description
15: 7	Reserved	Reserved, the reset value must be maintained.
6	TGN	<p>Trigger generation</p> <p>This bit can generate a trigger event when set by software. And at this time TIMx_STS.TITF = 1, if the corresponding interrupt and DMA are enabled, the corresponding interrupt and DMA will be generated. This bit is automatically cleared by hardware.</p> <p>0: No action</p> <p>1: Generated a trigger event</p>
5	Reserved	Reserved, the reset value must be maintained
4	CC4GN	<p>Capture/Compare 4 generation</p> <p>See TIMx_EVTGEN.CC1GN description.</p>
3	CC3GN	<p>Capture/Compare 3 generation</p> <p>See TIMx_EVTGEN.CC1GN description.</p>
2	CC2GN	<p>Capture/Compare 2 generation</p> <p>See TIMx_EVTGEN.CC1GN description.</p>

Bit field	Name	Description
1	CC1GN	<p>Capture/Compare 1 generation</p> <p>This bit can generate a capture/compare event when set by software. This bit is automatically cleared by hardware.</p> <p>When the corresponding channel of CC1 is in output mode:</p> <p>The TIMx_STS.CC1ITF flag will be pulled high, if the corresponding interrupt and DMA are enabled, the corresponding interrupt and DMA will be generated.</p> <p>When the corresponding channel of CC1 is in input mode:</p> <p>TIMx_CCDAT1 will capture the current counter value, and the TIMx_STS.CC1ITF flag will be pulled high, if the corresponding interrupt and DMA are enabled, the corresponding interrupt and DMA will be generated. If The IMx_STS.CC1ITF is already pulled high, pull TIMx_STS.CC1OCF high.</p> <p>0: No action 1: Generated a CC1 capture/compare event</p>
0	UDGN	<p>Update generation</p> <p>This bit can generate an update event when set by software. And at this time the counter will be reinitialized, the prescaler counter will be cleared, the counter will be cleared in center-aligned or up-counting mode, but take TIMx_AR in down-counting mode the value of the register. This bit is automatically cleared by hardware.</p> <p>0: No action 1: Generated an update event</p>

#### 10.4.8 Capture/compare mode register 1 (TIMx\_CCMOD1)

Offset address: 0x18

Reset value: 0x0000

Channels can be used for input (capture mode) or output (compare mode), and the direction of the channel is defined by the corresponding CCxSEL bit. The other bits of the register act differently in input and output modes. OCx describes the function of a channel in output mode, ICx describes the function of a channel in input mode. Hence, please note that the same bit can have different meanings for output mode and for input mode.

Output compare mode:

15	14	12	11	10	9	8	7	6	4	3	2	1	0
OC2CEN	OC2MD[2:0]	OC2PEN	OC2FEN	CC2SEL[1:0]	OC1CEN	OC1MD[2:0]	OC1PEN	OC1FEN	CC1SEL[1:0]				

rw                    rw

Bit field	Name	Description
15	OC2CEN	Output Compare 2 clear enable
14:12	OC2MD[2:0]	Output Compare 2 mode
11	OC2PEN	Output Compare 2 preload enable
10	OC2FEN	Output Compare 2 fast enable

Bit field	Name	Description
9:8	CC2SEL[1:0]	<p>Capture/compare 2 selection</p> <p>These bits are used to select the input/output and input mapping of the channel</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 only active when the internal trigger input is selected by TIMx_SMCTRL.TSEL.</li> </ul> <p><i>Note: CC2SEL is writable only when the channel is off (TIMx_CCEN.CC2EN = 0).</i></p>
7	OC1CEN	<p>Output Compare 1 clear enable</p> <p>0: OC1REF is not affected by ETRF input level</p> <p>1: OC1REF is cleared immediately when the ETRF input level is detected as high</p>
6:4	OC1MD[2:0]	<p>Output Compare 1 mode</p> <p>These bits are used to manage the output reference signal OC1REF, which determines the values of OC1 and OC1N, and is valid at high levels, while the active levels of OC1 and OC1N depend on the TIMx_CCEN.CC1P and TIMx_CCEN.CC1NP bits.</p> <p>000: Frozen. Comparison between TIMx_CCDAT1 register and counter TIMx_CNT has no effect on OC1REF signal.</p> <p>001: Set channel 1 to the active level on match. When TIMx_CCDAT1 = TIMx_CNT, OC1REF signal will be forced high.</p> <p>010: Set channel 1 as inactive level on match. When TIMx_CCDAT1 = TIMx_CNT, OC1REF signal will be forced low.</p> <p>011: Toggle. When TIMx_CCDAT1 = TIMx_CNT, OC1REF signal will be toggled.</p> <p>100: Force to inactive level. OC1REF signal is forced low.</p> <p>101: Force to active level. OC1REF signal is forced high.</p> <p>110: PWM mode 1 - In up-counting mode, if TIMx_CNT &lt; TIMx_CCDAT1, OC1REF signal of channel 1 is high, otherwise it is low. In down-counting mode, if TIMx_CNT &gt; TIMx_CCDAT1, OC1REF signal of channel 1 is low, otherwise it is high.</p> <p>111: PWM mode 2 - In up-counting mode, if TIMx_CNT &lt; TIMx_CCDAT1, OC1REF signal of channel 1 is low, otherwise it is high. In down-counting mode, if TIMx_CNT &gt; TIMx_CCDAT1, OC1REF signal of channel 1 is high, otherwise it is low.</p> <p><i>Note 1: In PWM mode 1 or PWM mode 2, the OC1REF level changes only when the comparison result changes or when the output compare mode is switched from frozen mode to PWM mode.</i></p>
3	OC1PEN	<p>Output Compare 1 preload enable</p> <p>0: Disable preload function of TIMx_CCDAT1 register. Supports write operations to TIMx_CCDAT1 register at any time, and the written value is effective immediately.</p> <p>1: Enable preload function of TIMx_CCDAT1 register. Only read and write operations to preload registers. When an update event occurs, the value of TIMx_CCDAT1 is loaded into the active register.</p> <p><i>Note 1: Only when TIMx_CTRL1.ONEPM = 1 (In one-pulse mode), PWM mode can be used without verifying the preload register, otherwise no other behavior can be predicted.</i></p>

Bit field	Name	Description
2	OC1FEN	<p>Output Compare 1 fast enable</p> <p>This bit is used to speed up the response of the CC output to the trigger input event.</p> <p>0: CC1 behaves normally depending on the counter and CCDAT1 values, even if the trigger is ON. The minimum delay for activating CC1 output when an edge occurs on the trigger input is 5 clock cycles.</p> <p>1: An active edge of the trigger input acts like a comparison match on CC1 output. Therefore, OC is set to the comparison level regardless of the comparison result. The delay time for sampling the trigger input and activating the CC1 output is reduced to 3 clock cycles.</p> <p>OCxFEN only works if the channel is configured in PWM1 or PWM2 mode.</p>
1: 0	CC1SEL[1:0]	<p>Capture/compare 1 selection</p> <p>These bits are used to select the input/output and input mapping of the channel</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 channels are configured as inputs and IC1 is mapped to TRC. This mode is only active when the internal trigger input is selected by TIMx_SMCTRL.TSEL.</p> <p><i>Note: CC1SEL is writable only when the channel is off (TIMx_CCEN.CC1EN = 0).</i></p>

#### Input capture mode:

15	IC2F[3:0]	rw	12	IC2PSC[1:0]	rw	11	CC2SEL[1:0]	rw	10	IC1F[3:0]	rw	9	IC1PSC[1:0]	rw	8	CC1SEL[1:0]	rw	7		4		3		2		1		0	
----	-----------	----	----	-------------	----	----	-------------	----	----	-----------	----	---	-------------	----	---	-------------	----	---	--	---	--	---	--	---	--	---	--	---	--

Bit field	Name	Description
15:12	IC2F[3:0]	Input Capture 2 Filter
11:10	IC2PSC[1:0]	Input Capture 2 Prescaler
9:8	CC2SEL[1:0]	<p>Capture/Compare 2 selection</p> <p>These bits are used to select the input/output and input mapping of the channel</p> <p>00: CC2 channel is configured as output</p> <p>01: CC2 channel is configured as input, IC2 is mapped on TI2</p> <p>10: CC2 channel is configured as input, IC2 is mapped on TI1</p> <p>11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is only active when the internal trigger input is selected by TIMx_SMCTRL.TSEL.</p> <p><i>Note: CC2SEL is writable only when the channel is off (TIMx_CCEN.CC2EN = 0).</i></p>
7:4	IC1F[3:0]	<p>Input Capture 1 filter</p> <p>These bits are used to define sampling frequency of TI1 input and the length of digital filter. The digital filter is an event counter that generates an output transition after N events are recorded.</p> <p>0000: No filter, sampling at f<sub>DTS</sub> frequency</p> <p>0001: f<sub>SAMPLING</sub> = f<sub>CLOCK</sub>, N = 2</p> <p>0010: f<sub>SAMPLING</sub> = f<sub>CLOCK</sub>, N = 4</p>

Bit field	Name	Description
		0011: f <sub>SAMPLING</sub> = f <sub>CLOCK</sub> , N = 8 0100: f <sub>SAMPLING</sub> = f <sub>DTS</sub> /2, N = 6 0101: f <sub>SAMPLING</sub> = f <sub>DTS</sub> /2, N = 8 0110: f <sub>SAMPLING</sub> = f <sub>DTS</sub> /4, N = 6 0111: f <sub>SAMPLING</sub> = f <sub>DTS</sub> /4, N = 8 1000: f <sub>SAMPLING</sub> = f <sub>DTS</sub> /8, N = 6 1001: f <sub>SAMPLING</sub> = f <sub>DTS</sub> /8, N = 8 1010: f <sub>SAMPLING</sub> = f <sub>DTS</sub> /16, N = 5 1011: f <sub>SAMPLING</sub> = f <sub>DTS</sub> /16, N = 6 1100: f <sub>SAMPLING</sub> = f <sub>DTS</sub> /16, N = 8 1101: f <sub>SAMPLING</sub> = f <sub>DTS</sub> /32, N = 5 1110: f <sub>SAMPLING</sub> = f <sub>DTS</sub> /32, N = 6 1111: f <sub>SAMPLING</sub> = f <sub>DTS</sub> /32, N = 8
3:2	IC1PSC[1:0]	Input Capture 1 prescaler These bits are used to select the ratio of the prescaler for IC1 (CC1 input). When TIMx_CCEN.CC1EN = 0, the prescaler will be reset. 00: No prescaler, capture is done each time an edge is detected on the capture input 01: Capture is done once every 2 events 10: Capture is done once every 4 events 11: Capture is done once every 8 events
1:0	CC1SEL[1:0]	Capture/Compare 1 selection These bits are used to select the input/output and input mapping of the channel 00: CC1 channel is configured as output 01: CC1 channel is configured as input, IC1 is mapped on TI1 10: CC1 channel is configured as input, IC1 is mapped on TI2 11: CC1 channel is configured as input, IC1 is mapped to TRC. This mode is only active when the internal trigger input is selected by TIMx_SMCTRL.TSEL. <i>Note: CC1SEL is writable only when the channel is off (TIMx_CCEN.CC1EN = 0).</i>

#### 10.4.9 Capture/compare mode register 2 (TIMx\_CCMOD2)

Offset address: 0x1C

Reset value: 0x0000

See the description of the CCMOD1 register above

Output comparison mode:

15	14	12	11	10	9	8	7	6	4	3	2	1	0
OC4CEN	OC4MD[2:0]		OC4PEN	OC4FEN	CC4SEL[1:0]		OC3CEN		OC3MD[2:0]		OC3PEN	OC3FEN	CC3SEL[1:0]

rw                    rw

Bit field	Name	Description
15	OC4CEN	Output compare 4 clear enable
14:12	OC4MD[2:0]	Output compare 4 mode
11	OC4PEN	Output compare 4 preload enable
10	OC4FEN	Output compare 4 fast enable
9:8	CC4SEL[1:0]	<p>Capture/Compare 4 selection</p> <p>These bits are used to select the input/output and input mapping of the channel</p> <p>00: CC4 channel is configured as output</p> <p>01: CC4 channel is configured as input, IC4 is mapped on TI4</p> <p>10: CC4 channel is configured as input, IC4 is mapped on TI3</p> <p>11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is only active when the internal trigger input is selected by TIMx_SMCTRL.TSEL.</p> <p><i>Note: CC4SEL is writable only when the channel is off (TIMx_CCEN.CC4EN = 0).</i></p>
7	OC3CEN	Output compare 3 clear enable
6:4	OC3MD[2:0]	Output compare 3 mode
3	OC3PEN	Output compare 3 preload enable
2	OC3FEN	Output compare 3 fast enable
1:0	CC3SEL[1:0]	<p>Capture/Compare 3 selection</p> <p>These bits are used to select the input/output and input mapping of the channel</p> <p>00: CC3 channel is configured as output</p> <p>01: CC3 channel is configured as input, IC3 is mapped to TI3</p> <p>10: CC3 channel is configured as input, IC3 is mapped on TI4</p> <p>11: CC3 channel is configured as input, IC3 is mapped to TRC. This mode is only active when the internal trigger input is selected by TIMx_SMCTRL.TSEL.</p> <p><i>Note: CC3SEL is writable only when the channel is off (TIMx_CCEN.CC3EN = 0).</i></p>

#### Input capture mode:

15	IC4F[3:0]	12	IC4PSC[1:0]	11	CC4SEL[1:0]	10	IC3F[3:0]	9	IC3PSC[1:0]	8	CC3SEL[1:0]	7		4		3		2		1		0
	rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw	

Bit field	Name	Description
15:12	IC4F[3:0]	Input Capture 4 filter
11:10	IC4PSC[1:0]	Input Capture 4 Prescaler
9:8	CC4SEL[1:0]	<p>Capture/Compare 4 selection</p> <p>These bits are used to select the input/output and input mapping of the channel</p> <p>00: CC4 channel is configured as output</p> <p>01: CC4 channel is configured as input, IC4 is mapped on TI4</p> <p>10: CC4 channel is configured as input, IC4 is mapped on TI3</p> <p>11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is only active when the internal trigger input is selected by TIMx_SMCTRL.TSEL.</p>

Bit field	Name	Description
<i>Note: CC4SEL is writable only when the channel is off (TIMx_CCEN.CC4EN = 0).</i>		
7:4	IC3F[3:0]	Input Capture 3 filter
3:2	IC3PSC[1:0]	Input Capture 3 Prescaler
1:0	CC3SEL[1:0]	<p>Capture/compare 3 selection</p> <p>These bits are used to select the input/output and input mapping of the channel</p> <p>00: CC3 channel is configured as output</p> <p>01: CC3 channel is configured as input, IC3 is mapped to TI3</p> <p>10: CC3 channel is configured as input, IC3 is mapped on TI4</p> <p>11: CC3 channel is configured as input, IC3 is mapped to TRC. This mode is only active when the internal trigger input is selected by TIMx_SMCTRL.TSEL.</p> <p><i>Note: CC3SEL is writable only when the channel is off (TIMx_CCEN.CC3EN = 0).</i></p>

## 10.4.10 Capture/compare enable registers (TIMx\_CCEN)

Offset address: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	CC4P	CC4EN	Reserved	CC3P	CC3EN	Reserved	CC2P	CC2EN	Reserved	CC1OP	CC1EN				
rw	rw			rw	rw		rw	rw		rw	rw				

Bit field	Name	Description
15:14	Reserved	Reserved, the reset value must be maintained.
13	CC4P	Capture/Compare 4 output polarity See TIMx_CCEN.CC1P description.
12	CC4EN	Capture/Compare 4 output enable See TIMx_CCEN.CC1EN description.
11:10	Reserved	Reserved, the reset value must be maintained
9	CC3P	Capture/Compare 3 output polarity See TIMx_CCEN.CC1P description.
8	CC3EN	Capture/Compare 3 output enable See TIMx_CCEN.CC1EN description.
7:6	Reserved	Reserved, the reset value must be maintained
5	CC2P	Capture/Compare 2 output polarity See TIMx_CCEN.CC1P description.
4	CC2EN	Capture/Compare 2 output enable See TIMx_CCEN.CC1EN description.
3:2	Reserved	Reserved, the reset value must be maintained
1	CC1P	Capture/Compare 1 output polarity When the corresponding channel of CC1 is in output mode:

Bit field	Name	Description
		<p>0: OC1 active high 1: OC1 active low</p> <p>When the corresponding channel of CC1 is in input mode: At this time, this bit is used to select whether IC1 or the inverse signal of IC1 is used as the trigger or capture signal.</p> <p>0: non-inverted: Capture action occurs when IC1 generates a rising edge. When used as external trigger, IC1 is non-inverted. 1: inverted: Capture action occurs when IC1 generates a falling edge. When used as external trigger, IC1 is inverted.</p> <p><i>Note: If TIMx_BKDT.LCKCFG = 3 or 2, these bits cannot be modified.</i></p>
0	CC1EN	<p>Capture/Compare 1 output enable</p> <p>When the corresponding channel of CC1 is in output mode: 0: Disable - Disable output OC1 signal. 1: Enable - Enable output OC1 signal.</p> <p>When the corresponding channel of CC1 is in input mode: At this time, this bit is used to disable/enable the capture function. 0: Disable capture 1: Enable capture</p>

Table 10-4 Output control bits of standard OCx channel

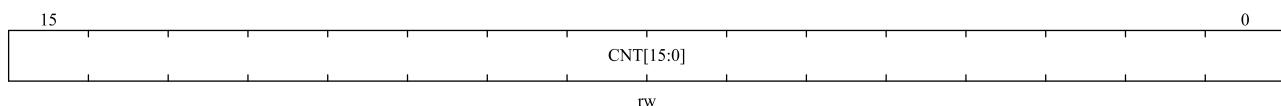
CCxEN	OCx output status
0	Disable output (OCx=0)
1	OCx = OCxREF + polarity

*Note: The state of external I/O pins connected to standard OCx channels depends on the OCx channel state and GPIO and AFIO registers.*

#### 10.4.11 Counters (TIMx\_CNT)

Offset address: 0x24

Reset value: 0x0000

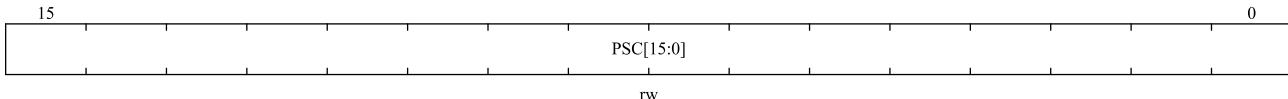


Bit field	Name	Description
15:0	CNT[15:0]	Counter value

### 10.4.12 Prescaler (TIMx\_PSC)

Offset address: 0x28

Reset value: 0x0000

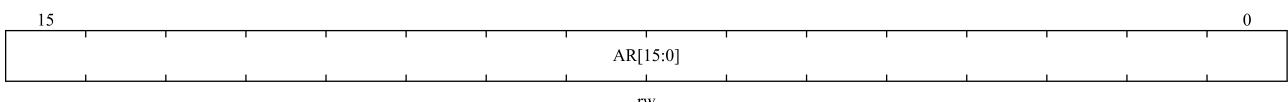


Bit field	Name	Description
15:0	PSC[15:0]	Prescaler value Counter clock $f_{CK\_CNT} = f_{CK\_PSC} / (PSC[15:0] + 1)$ . Each time an update event occurs, the PSC value is loaded into the active prescaler register.

### 10.4.13 Auto-reload register (TIMx\_AR)

Offset address: 0x2C

Reset values: 0xFFFF

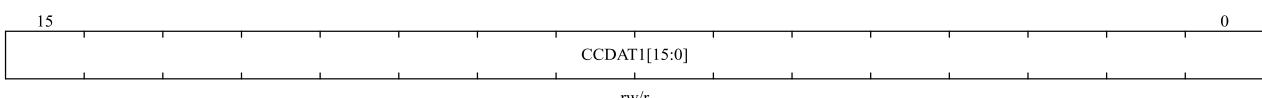


Bit field	Name	Description
15:0	AR[15:0]	Auto-reload value These bits define the value that will be loaded into the actual auto-reload register. See Section 9.3.1 for more details. When the TIMx_AR.AR [15:0] value is null, the counter does not work.

### 10.4.14 Capture/compare register 1 (TIMx\_CC DAT1)

Offset address: 0x34

Reset value: 0x0000



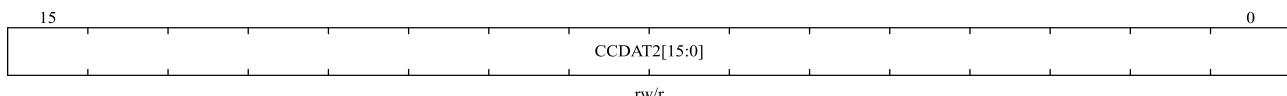
Bit field	Name	Description
15:0	CC DAT1[15:0]	Capture/Compare 1 value <ul style="list-style-type: none"> <li>■ CC1 channel is configured as output:</li> </ul>

Bit field	Name	Description
		<p>CCDAT1 contains the value to be compared to the counter TIMx_CNT, signaling on the OC1 output.</p> <p>If the preload feature is not selected in TIMx_CCMOD1.OC1PEN bit, the written value is immediately transferred to the active register. Otherwise, this preloaded value is transferred to the active register only when an update event occurs.</p> <ul style="list-style-type: none"> <li>■ CC1 channel is configured as input:</li> </ul> <p>CCDAT1 contains the counter value transferred by the last input capture 1 event (IC1).</p> <p>When configured as input mode, register CCDAT1 is only readable.</p> <p>When configured as output mode, register CCDAT1 is readable and writable.</p>

#### 10.4.15 Capture/compare register 2 (TIMx\_CCDAT2)

Offset address: 0x38

Reset value: 0x0000

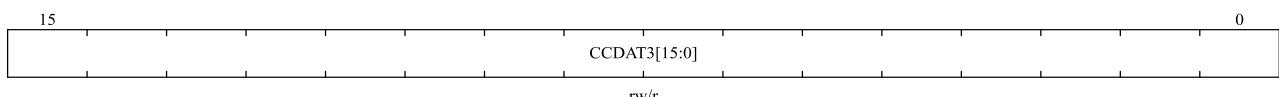


Bit field	Name	Description
15:0	CCDAT2[15:0]	<p>Capture/Compare 2 values</p> <ul style="list-style-type: none"> <li>■ CC2 channel is configured as output:</li> </ul> <p>CCDAT2 contains the value to be compared to the counter TIMx_CNT, signaling on the OC2 output.</p> <p>If the preload feature is not selected in TIMx_CCMOD1.OC2PEN bit, the written value is immediately transferred to the active register. Otherwise, this preloaded value is transferred to the active register only when an update event occurs.</p> <ul style="list-style-type: none"> <li>■ CC2 channel is configured as input:</li> </ul> <p>CCDAT2 contains the counter value transferred by the last input capture 2 event (IC2).</p> <p>When configured as input mode, register CCDAT2 is only readable.</p> <p>When configured as output mode, register CCDAT2 is readable and writable.</p>

#### 10.4.16 Capture/compare register 3 (TIMx\_CCDAT3)

Offset address: 0x3C

Reset value: 0x0000

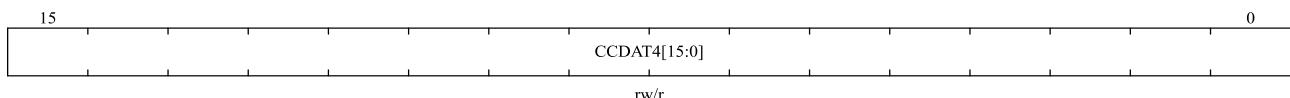


Bit field	Name	Description
15:0	CCDAT3[15:0]	<p>Capture/Compare 3 value</p> <ul style="list-style-type: none"> <li>■ CC3 channel is configured as output: CCDAT3 contains the value to be compared to the counter TIMx_CNT, signaling on the OC3 output.</li> </ul> <p>If the preload feature is not selected in TIMx_CCMOD2.OC3PEN bit, the written value is immediately transferred to the active register. Otherwise, this preloaded value is transferred to the active register only when an update event occurs.</p> <ul style="list-style-type: none"> <li>■ CC3 channel is configured as input: CCDAT3 contains the counter value transferred by the last input capture 3 event (IC3).</li> </ul> <p>When configured as input mode, register CCDAT3 is only readable.</p> <p>When configured as output mode, register CCDAT3 is readable and writable.</p>

#### 10.4.17 Capture/compare register 4 (TIMx\_CC DAT4)

Offset address: 0x40

Reset value: 0x0000

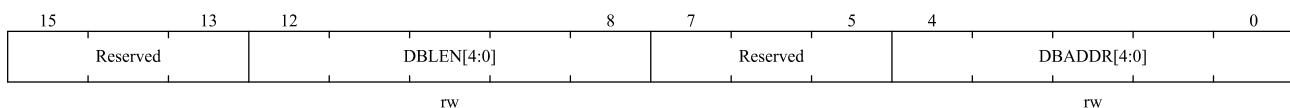


Bit field	Name	Description
15:0	CCDAT4[15:0]	<p>Capture/Compare 4 value</p> <ul style="list-style-type: none"> <li>■ CC4 channel is configured as output: CCDAT4 contains the value to be compared to the counter TIMx_CNT, signaling on the OC4 output.</li> </ul> <p>If the preload feature is not selected in TIMx_CCMOD2.OC4PEN bit, the written value is immediately transferred to the active register. Otherwise, this preloaded value is transferred to the active register only when an update event occurs.</p> <ul style="list-style-type: none"> <li>■ CC4 channel is configured as input: CCDAT4 contains the counter value transferred by the last input capture 4 event (IC4).</li> </ul> <p>When configured as input mode, register CCDAT4 is only readable.</p> <p>When configured as output mode, register CCDAT4 is readable and writable.</p>

#### 10.4.18 DMA Control register (TIMx\_DCTRL)

Offset address: 0x48

Reset value: 0x0000

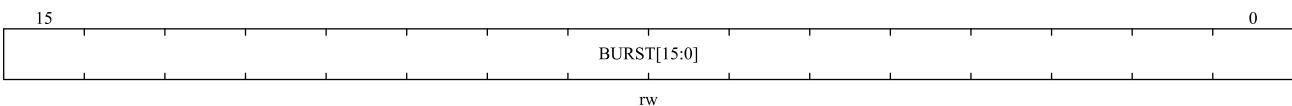


Bit field	Name	Description
15:13	Reserved	Reserved, the reset value must be maintained
12:8	DBLEN[4:0]	<p>DMA Burst Length</p> <p>This bit field defines the number DMA will access (write/read) TIMx_DADDR register.</p> <p>00000:1 time transfer      00001: 2 times transfers      00010: 3 times transfers      ...      10001: 18 times transfers</p>
7:5	Reserved	Reserved, the reset value must be maintained.
4:0	DBADDR[4:0]	<p>DMA Base Address</p> <p>This bit field defines the first address where the DMA accesses the TIMx_DADDR register.</p> <p>When access is done through the TIMx_DADDR first time, this bit-field specifies the address you just access. And then the second access to the TIMx_DADDR, you will access the address of "DMA Base Address + 4"</p> <p>00000: TIMx_CTRL1,      00001: TIMx_CTRL2,      00010: TIMx_SMCTRL,      ...      01011: TIMx_AR,      01100: Reserved,      01101: TIMx_CCDAT1,      .....      10000: TIMx_CCDAT4,      10001: Reserved,      10010: TIMx_DCTRL</p>

#### 10.4.19 DMA transfer buffer register (TIMx\_DADDR)

Offset address: 0x4C

Reset value: 0x0000



Bit field	Name	Description
15:0	BURST[15:0]	<p>DMA access buffer.</p> <p>When a read or write operation is assigned to this register, the register located at the address range (DMA base address + DMA burst length × 4) will be accessed.</p> <p>DMA base address = The address of TIM_CTRL1 + TIMx_DCTRL. DBADDR * 4;      DMA burst len = TIMx_DCTRL.DBLEN + 1.</p>

Bit field	Name	Description
		<p>Example:</p> <p>If TIMx_DCTRL.DBLEN = 0x3(4 transfers), TIMx_DCTRL.DBADDR = 0xD          (TIMx_CCDAT1), DMA data length = half word, DMA memory address = buffer address in SRAM, DMA peripheral address = TIMx_DADDR address.</p> <p>When an event occurs, TIMx will send requests to the DMA, and transfer data 4 times.</p> <p>For the first time, DMA access to the TIMx_DADDR register will be mapped to access TIMx_CCDAT1 register;</p> <p>For the second time, DMA access to the TIMx_DADDR register will be mapped to access TIMx_CCDAT2 register;</p> <p>.... .</p> <p>For the fourth time, DMA access to the TIMx_DADDR register will be mapped to access TIMx_CCDAT4 register;</p>

## 11 Basic timers (TIM6)

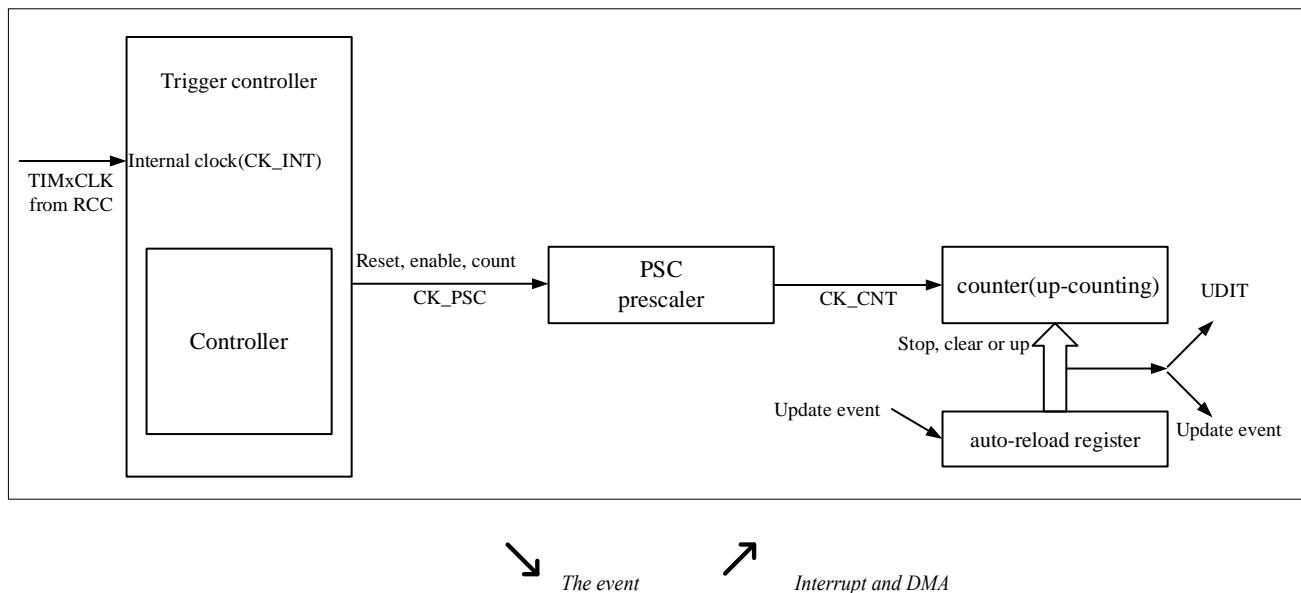
### 11.1 Basic timers introduction

The basic timer contains a 16-bit counter.

### 11.2 Main features of Basic timers

- 16-bit auto-reload up-counting counters.
- 16-bit programmable prescaler. (The frequency division factor can be configured with any value between 1 and 65536)
- The events that generate the interrupt/DMA are as follows:
  - ◆ Update event

Figure 11-1 Block diagram of TIMx ( $x = 6$ )



### 11.3 Basic timers description

#### 11.3.1 Time-base unit

The time-base unit mainly includes: prescaler, counter and auto-reload. When the time base unit is working, the software can read and write the corresponding registers (TIMx\_PSC, TIMx\_CNT and TIMx\_AR) at any time.

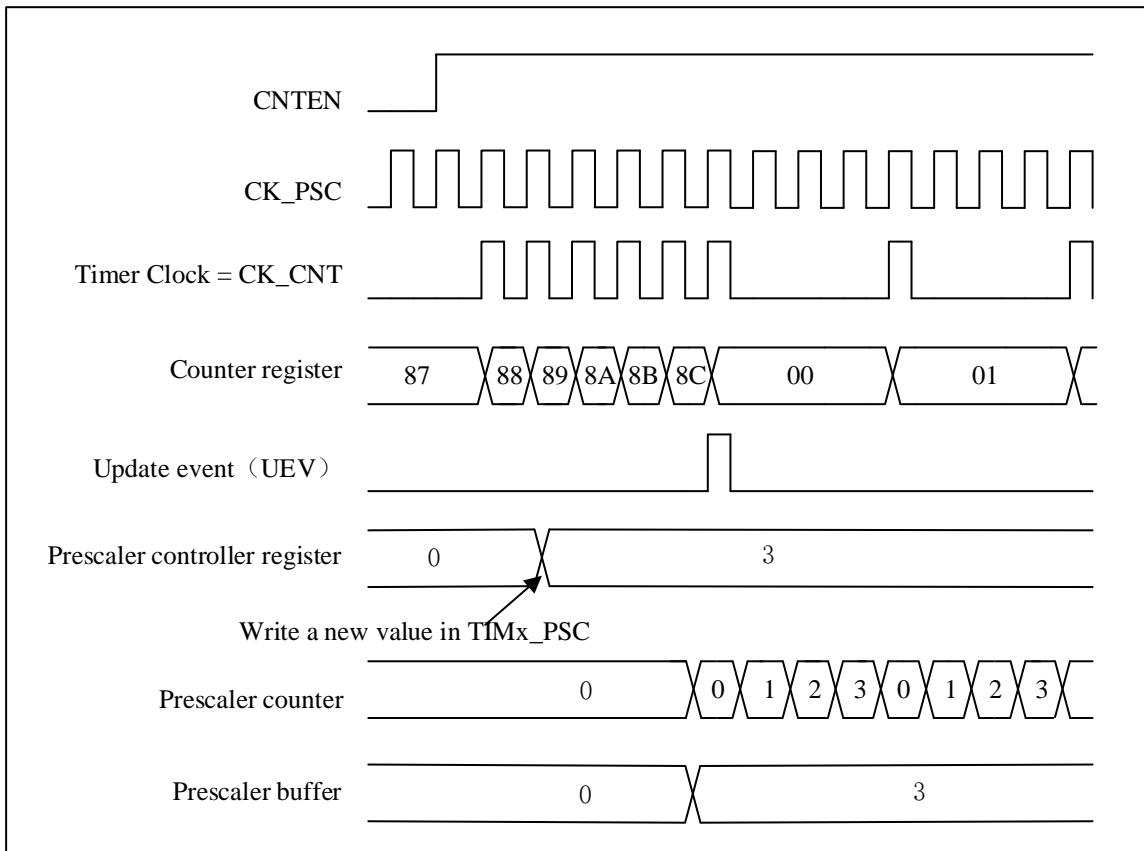
Depending on the setting of the auto-reload preload enable bit (TIMx\_CTRL1.ARREN), the value of the preload register is transferred to the shadow register immediately or at each update event UEV. An update event is generated when the counter reaches the overflow condition and it can be generated by software when TIMx\_CTRL1.UPDIS=0. The counter CK\_CNT is valid only when the TIMx\_CTRL1.CNTEN bit is set. The counter starts counting one clock

cycle after the TIMx\_CTRL1.CNTEN bit is set.

### 11.3.1.1 Prescaler description

The TIMx\_PSC register consists of a 16-bit counter that can be used to divide the counter clock frequency by any factor between 1 and 65536. It can be changed on the fly as it is buffered. The prescaler value is only taken into account at the next update event.

Figure 11-2 Counter timing diagram with prescaler division change from 1 to 4



### 11.3.2 Counter mode

#### 11.3.2.1 Up-counting mode

In up-counting mode, the counter will count from 0 to the value of the register TIMx\_AR, then it resets to 0. And a counter overflow event is generated.

If the TIMx\_CTRL1.UPRS bit (select update request) and the TIMx\_EVTGEN.UDGN bit are set, an update event (UEV) will generate, and TIMx\_STS.UDITF will not be set by hardware. Therefore, no update interrupts or update DMA requests are generated. This setting is used in scenarios where you want to clear the counter but do not want to generate an update interrupt.

Depending on the update request source is configured in the TIMx\_CTRL1.UPRS, When an update event occurs, all registers are updated and the TIMx\_STS.UDITF is set:

- Update auto-reload shadow registers with preload value(TIMx\_AR), when TIMx\_CTRL1.ARPE = 1.

- The prescaler shadow register is reloaded with the preload value(TIMx\_PSC)

To avoid updating the shadow registers when new values are written to the preload registers, you can disable the update by setting TIMx\_CTRL1.UPDIS=1.

When an update event occurs, the counter will still be cleared and the prescaler counter will also be set to 0 (but the prescaler value will remain unchanged).

The figure below shows some examples of the counter behavior and the update flags for different division factors in the up-counting mode.

Figure 11-3 Timing diagram of up-counting. The internal clock divider factor = 2/N

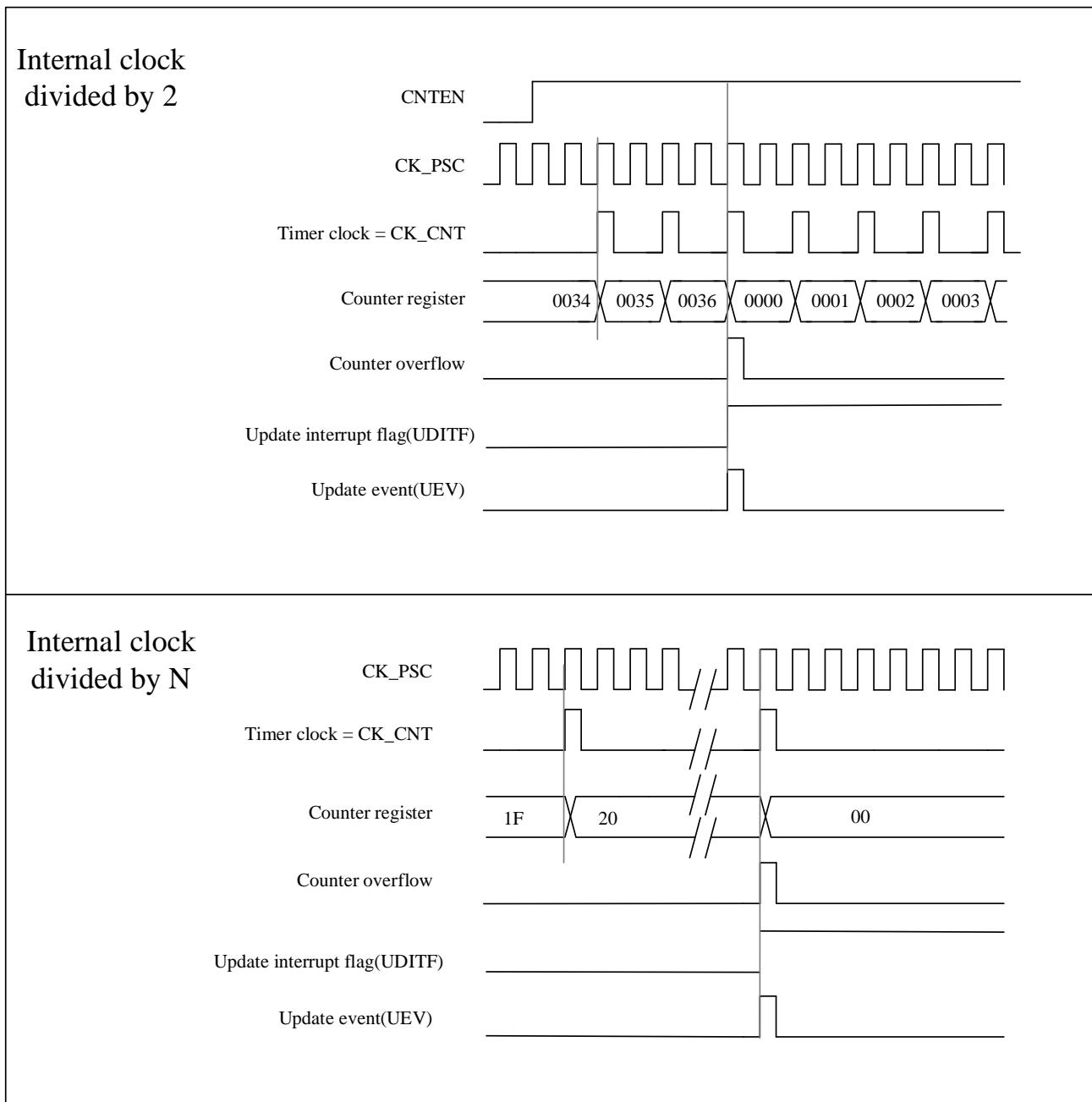
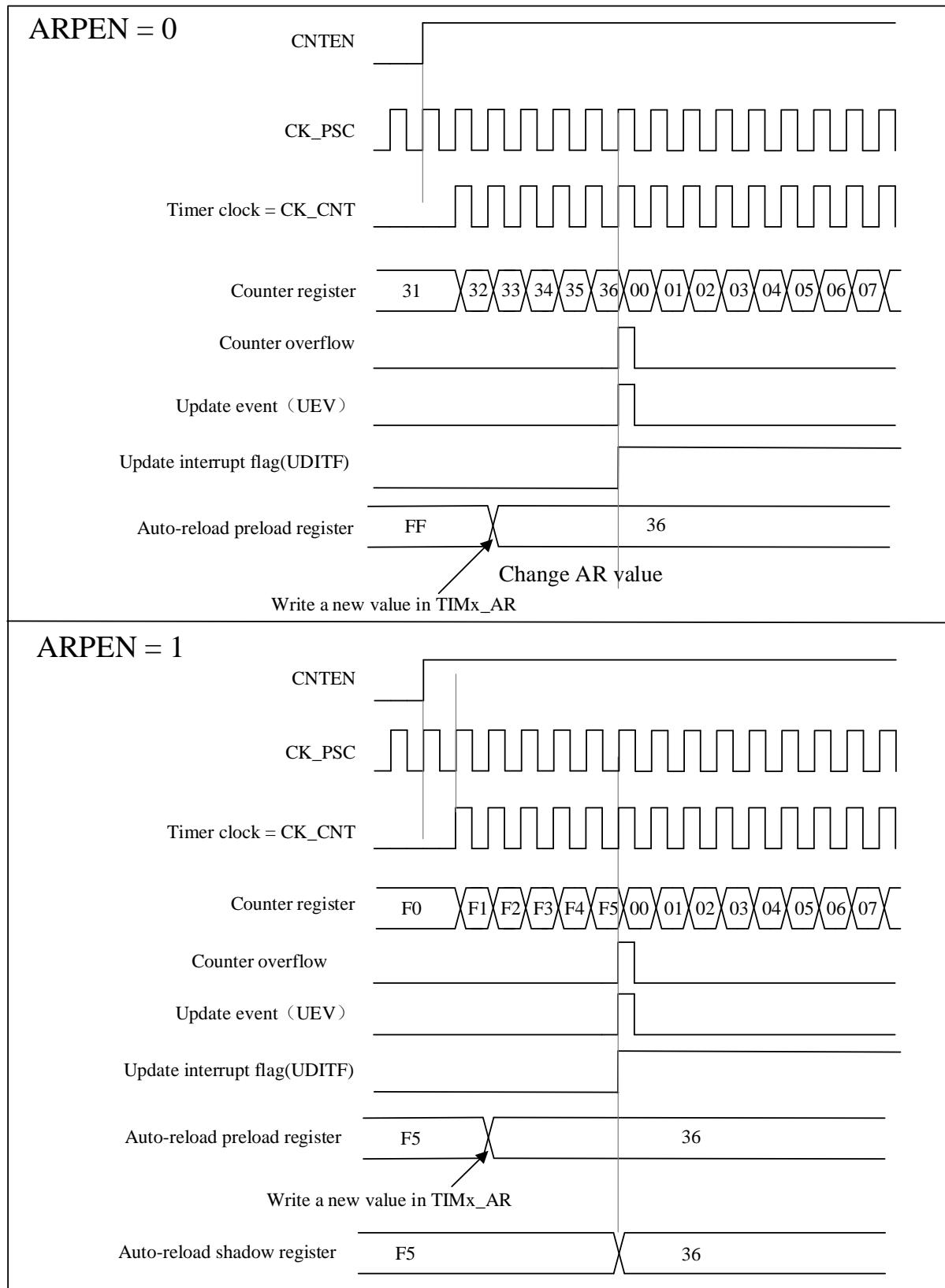


Figure 11-4 Timing diagram of the up-counting, update event when ARPEN=0/1



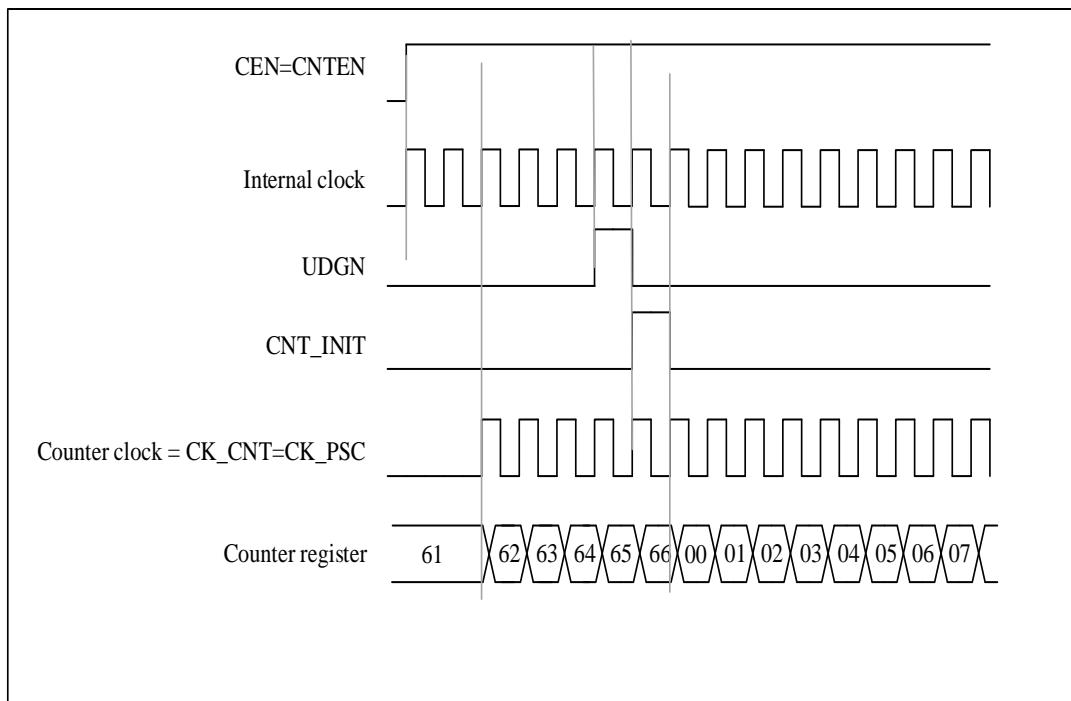
### 11.3.3 Clock selection

- The internal clock of timers: CK\_INT

#### 11.3.3.1 Internal clock source (CK\_INT)

It is provided that the TIMx\_CTRL1.CNTEN bit is written as '1' by software, the clock source of the prescaler is provided by the internal clock CK\_INT.

Figure 11-5 Control circuit in normal mode, internal clock divided by 1



### 11.3.4 Debug mode

When the microcontroller is in debug mode (the Cortex-M0 core halted), depending on the DBG\_CTRL.TIMx\_STOP configuration in the PWR module, the TIMx counter can either continue to work normally or stop. For more details, see 3.3.2.

## 11.4 TIMx register description(x=6)

For abbreviations used in registers, see section 1.1

These peripheral registers can be operated as half word (16-bits) or one word (32-bits).

## 11.4.1 Register overview

Table 11-1 Register overview

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
000h	TIMx_CTRL 1 Reset Value																																				
004h																																					
008h																																					
00Ch	TIMx_DINT EN Reset Value																																				
010h	TIMx_STS Reset Value																																				
014h	TIMx_EVTG EN Reset Value																																				
018h																																					
01Ch																																					
020h																																					
024h	TIMx_CNT Reset Value																																				
028h	TIMx_PSC Reset Value																																				
02Ch	TIMx_AR Reset Value																																				

## 11.4.2 Control Register 1 (TIMx\_CTRL1)

Offset address: 0x00

298 / 558

Reset value: 0x0000

15	Reserved	8	7	6	Reserved	4	3	2	1	0
				ARPEN			ONEPM	UPRS	UPDIS	CNTEN
					rw		rw	rw	rw	rw

Bit field	Name	Description
15:8	Reserved	Reserved, the reset value must be maintained
7	ARPEN	ARPEN: Auto-reload preload enable 0: Shadow register disable for TIMx_AR register 1: Shadow register enable for TIMx_AR register
6:4	Reserved	Reserved, the reset value must be maintained
3	ONEPM	One-pulse mode 0: Disable one-pulse mode, the counter counts are not affected when an update event occurs. 1: Enable one-pulse mode, the counter stops counting when the next update event occurs (clearing TIMx_CTRL1.CNTEN bit)
2	UPRS	Update request source This bit is used to select the UEV event sources by software. 0: If update interrupt or DMA request is enabled, any of the following events will generate an update interrupt or DMA request: Counter overflow The TIMx_EVTGEN.UDGN bit is set 1: If update interrupt or DMA request is enabled, only counter overflow will generate update interrupt or DMA request
1	UPDIS	Update disable This bit is used to enable/disable the Update event (UEV) events generation by software. 0: Enable UEV. UEV will be generated if one of following condition been fulfilled: Counter overflow The TIMx_EVTGEN.UDGN bit is set Shadow registers will update with preload value. 1: UEV disabled. No update event is generated, and the shadow registers (AR, PSC) keep their values. If the TIMx_EVTGEN.UDGN bit is set, the counter and prescaler are reinitialized.
0	CNTEN	Counter Enable 0: Disable counter 1: Enable counter

### 11.4.3 DMA/Interrupt Enable Registers (TIMx\_DINTEN)

Offset address: 0x0C

Reset value: 0x0000

15	Reserved	9	UDEN	8	7	Reserved	1	0
							rw	rw

Bit field	Name	Description
15:9	Reserved	Reserved, the reset value must be maintained
8	UDEN	Update DMA Request enable 0: Disable update DMA request 1: Enable update DMA request
7:1	Reserved	Reserved, the reset value must be maintained
0	UIEN	Update interrupt enable 0: Disable update interrupt 1: Enables update interrupt

#### 11.4.4 Status Registers (TIMx\_STS)

Offset address: 0x10

Reset value: 0x0000

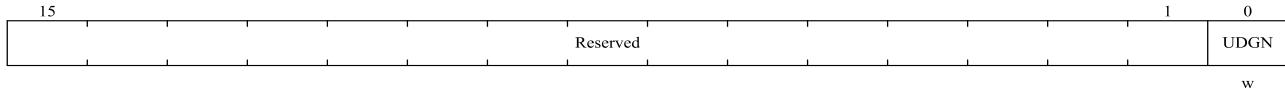
15	Reserved	1	0
			rc_w0

Bit field	Name	Description
15:1	Reserved	Reserved, the reset value must be maintained
0	UDITF	Update interrupt flag This bit is set by hardware when an update event occurs under the following conditions: When TIMx_CTRL1.UPDIS = 0, and counter value overflow. When TIMx_CTRL1.UPRS = 0, TIMx_CTRL1.UPDIS = 0, and set the TIMx_EVTGEN.UDGN bit by software to reinitialize the CNT. This bit is cleared by software. 0: No update event occurred 1: Update interrupt occurred

#### 11.4.5 Event Generation registers (TIMx\_EVTGEN)

Offset address: 0x14

Reset values: 0 x0000

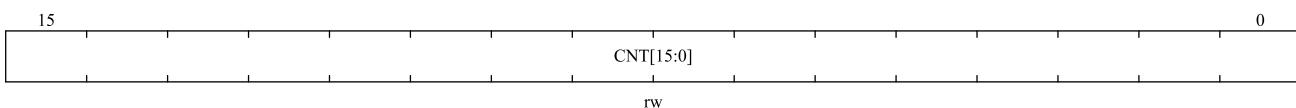


Bit field	Name	Description
15: 1	Reserved	Reserved, the reset value must be maintained.
0	UDGN	UDGN: Update generation Software can set this bit to update configuration register value and hardware will clear it automatically. 0: No effect. 1: Timer counter will restart and all shadow register will be updated. It will restart prescaler counter also.

#### 11.4.6 Counters (TIMx\_CNT)

Offset address: 0x24

Reset value: 0x0000

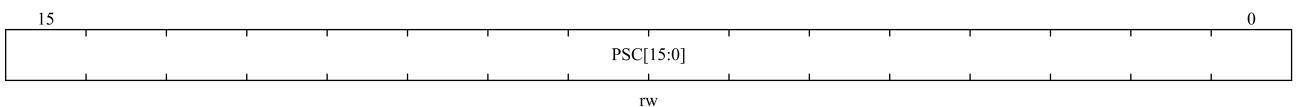


Bit field	Name	Description
15:0	CNT[15:0]	Counter value

#### 11.4.7 Prescaler (TIMx\_PSC)

Offset address: 0x28

Reset value: 0x0000

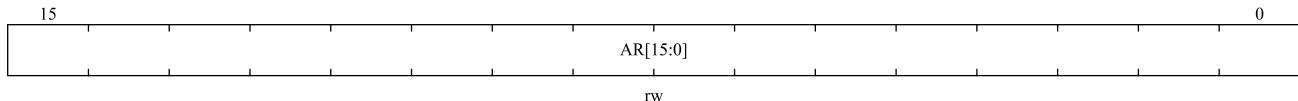


Bit field	Name	Description
15:0	PSC[15:0]	Prescaler value PSC register value will be updated to prescaler register at update event. Counter clock frequency is input clock frequency divide PSC + 1.

### 11.4.8 Automatic reload register (TIMx\_AR)

Offset address: 0x2C

Reset values: 0xFFFF



Bit field	Name	Description
15:0	AR[15:0]	<p>Auto-reload value</p> <p>These bits define the value that will be loaded into the actual auto-reload register.</p> <p>See 11.3.1 for more details.</p> <p>When the TIMx_AR.AR [15:0] value is null, the counter does not work.</p>

## 12 Low Power Timer (LPTIM)

### 12.1 Introduction

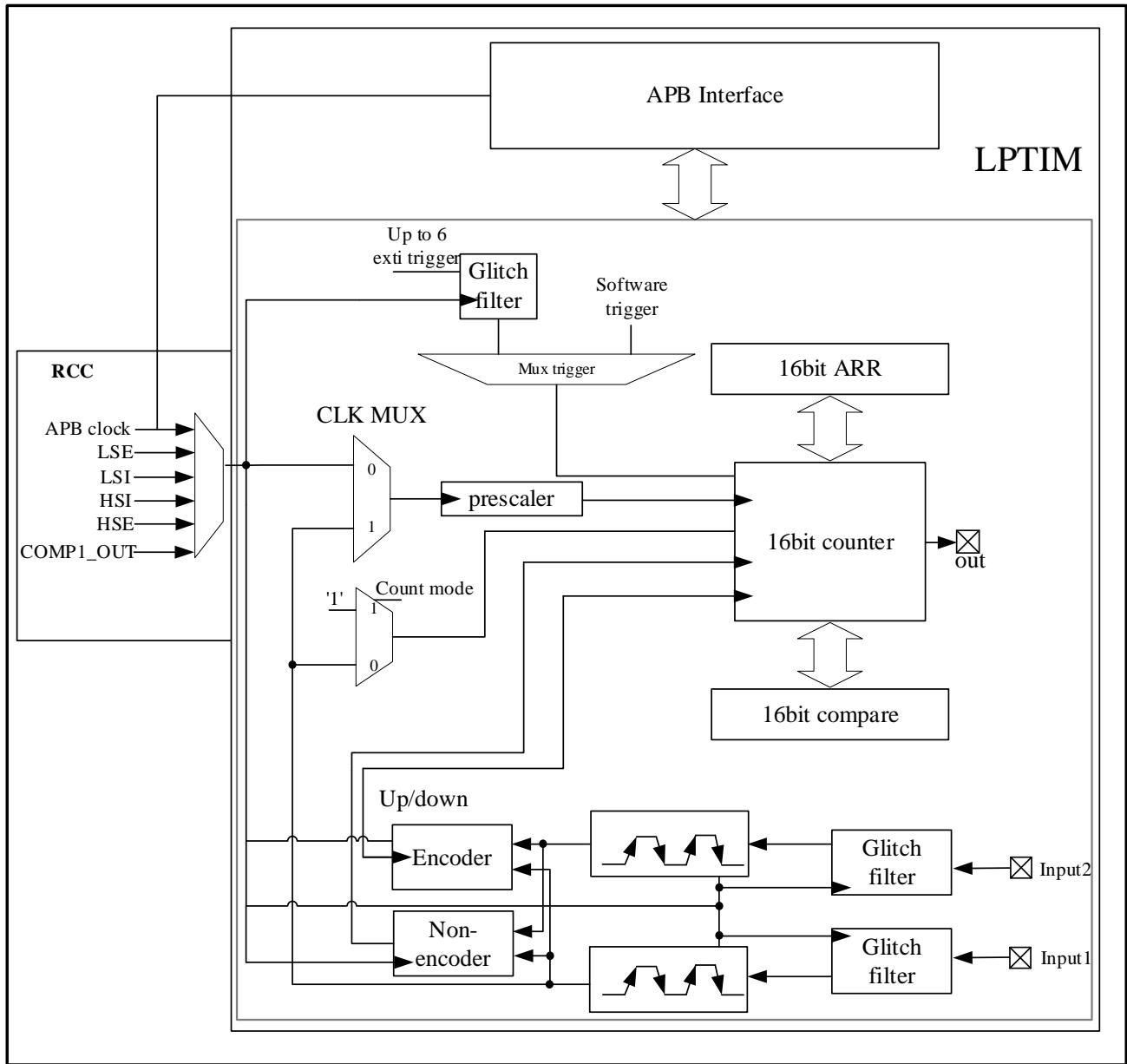
The LPTIM is a 16-bit timer with multiple clock sources, it can keep running in all power modes except for PD mode. LPTIM can run without internal clock source, it can be used as a “Pulse Counter”. Also, the LPTIM can wake up the system from low-power modes, to realize “Timeout functions” with extreme low power consumption.

### 12.2 Main features

- 16-bit up-counter
- 3-bit clock prescaler, 8 dividing factors (1, 2, 4, 8, 16, 32, 64, 128)
- Multiple clock sources
  - Internal: HSI, HSE, LSI, LSE, APB1 and COMP\_OUT clock
  - External: LPTIM Input1 (working with no LP oscillator running, used by Pulse Counter application)
- 16-bit auto-reload register
- 16-bit compare register
- Continuous/One-shot counting mode
- Programmable software and hardware input trigger
- Programmable digital filter for filtering glitch
- Configurable output: Pulse, PWM
- Configurable I/O polarity
- Encoder mode

## 12.3 Block diagram

Figure 12-1 LPTIM Diagram



## 12.4 Function description

### 12.4.1 LPTIM clocks and on-off control

The LPTIM can use either internal clock source or external clock source.

The LPTIM can use an internal clock source or an external clock source. The internal clock source can be selected between APB, LSI, LSE, HSE, HSI or COMP1 by configuring the RCC\_CFG2.LPTIMSEL[2:0] bits. The external

clock source can be selected from comparator or GPIO. For external clock source, the LPTIM has two configurations:

- The LPTIM uses both external clock and internal clock.
- The LPTIM only use external clock from comparator or external input1. This configuration is for LOW POWER application.

LPTIM\_CFG.CLKSEL and LPTIM\_CFG.CNTMEN bits are for the clock source configuration. The active clock edge is configured through LPTIM\_CFG.CLKPOL[1:0] bits.

When the LPTIM only uses external clock source. It can only select one active clock edge. LPTIM can select both active clock edges only when it is using internal clock source or both external and internal clock sources.

*Note: When both active edges for external clock, LPTIM needs to use an internal clock to oversample the external clock. The internal clock frequency should be at least 4 times higher than the external clock frequency.*

### 12.4.2 Prescaler

The LPTIM counter is preceded by a configurable power-of-2 prescaler. The prescaler ratio is controlled by the LPTIM\_CFG.CLKPREG[2:0] field. The table below lists all the possible division ratios:

Table 12-1 Pre-scaler division ratios

Control bits	The corresponding frequency division factor
000	/1
001	/2
010	/4
011	/8
100	/16
101	/32
110	/64
111	/128

### 12.4.3 Glitch filter

LPTIM has glitch filters for inputs to remove glitches and prevent unexpected counts or triggers.

Glitch filter needs an internal clock source to operate. And the clock source should be provided before the glitch filter is enabled. This is necessary to guarantee the proper operation of the filters.

The glitch filters has two major purposes:

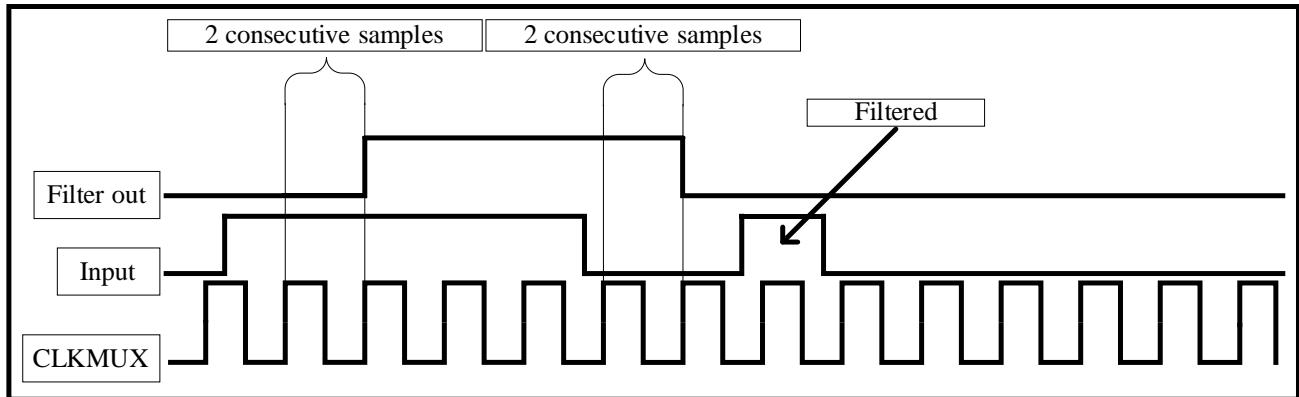
- For the external inputs: The filter sensitivity is configured through the LPTIM\_CFG.CLKFLT[1:0] bits.
- For the internal trigger inputs: The filter sensitivity is configured through the LPTIM\_CFG.RIGFLT[1:0] bits.

*Note: The detection configuration is only applicable for its corresponding inputs.*

The filter sensitivity acts on the number of consecutive equal samples that should be detected on one of the LPTIM inputs to consider a signal level change as a valid transition.

Shows an example of glitch filter behavior when detected a 2 consecutive samples.

Figure 12-2 Glitch filter timing diagram



*Note: If no internal clock is used, the glitch filter needs to be turned off by clearing LPTIM\_CFG.CLKFLT[1:0] and LPTIM\_CFG.TRIGFLT[1:0] bits. If glitch filter is not used, user can use digital filter in comparator or external analog filter to remove glitches.*

#### 12.4.4 Timer enable

The LPTIM\_CTRL.LPTIMEN bit is used to enable/disable the LPTIM kernel logic. After setting the LPTIM\_CTRL.LPTIMEN bit, a delay of two counter clock is needed before the LPTIM is turned on.

The LPTIM\_CFG and LPTIM\_INTEN registers must be modified only when the LPTIM is turned off.

#### 12.4.5 Trigger multiplexer

The LPTIM counter can be triggered either by software or by an active edge on one of the 6 trigger inputs.

The trigger source is configured through LPTIM\_CFG.TRGGEN[1:0] bits. LPTIM\_CFG.TRGGEN[1:0] = ‘00’, the trigger is selected as LPTIM\_CTRL.TSTCM or LPTIM\_CTRL.SNGMST bit, which can be set by software. The other values of LPTIM\_CFG.TRGGEN[1:0] are for the active edge configuration of the trigger. The internal counter will start once an active edge is detected.

LPTIM\_CFG.TRGSEL[2:0] is used to select one of the 6 trigger inputs only when LPTIM\_CFG.TRGGEN[1:0] is not equal to ‘00’.

If LPTIM is using external trigger, which will be considered as asynchronous triggers. For asynchronous triggers, the LPTIM needs two counter clock cycles latency for synchronization.

When timeout function is disabled, new trigger event will be ignored if the LPTIM is already started.

*Note: Any write to the LPTIM\_CTRL.SNGMST/ LPTIM\_CTRL.TSTCM bit will be discarded if the LPTIM is not*

*enabled.*

Table 12-2 6 trigger inputs corresponding to LPTIM\_CFG.TRGSEL[2:0] bits

Control bits	Corresponding trigger input
000	PB6 or PA6
001	RTC alarm A
010	RTC alarm B
011	RTC_TAMP1
100	RTC_TAMP2
110	COMP_OUT

## 12.4.6 Operating mode

The LPTIM has two operating modes:

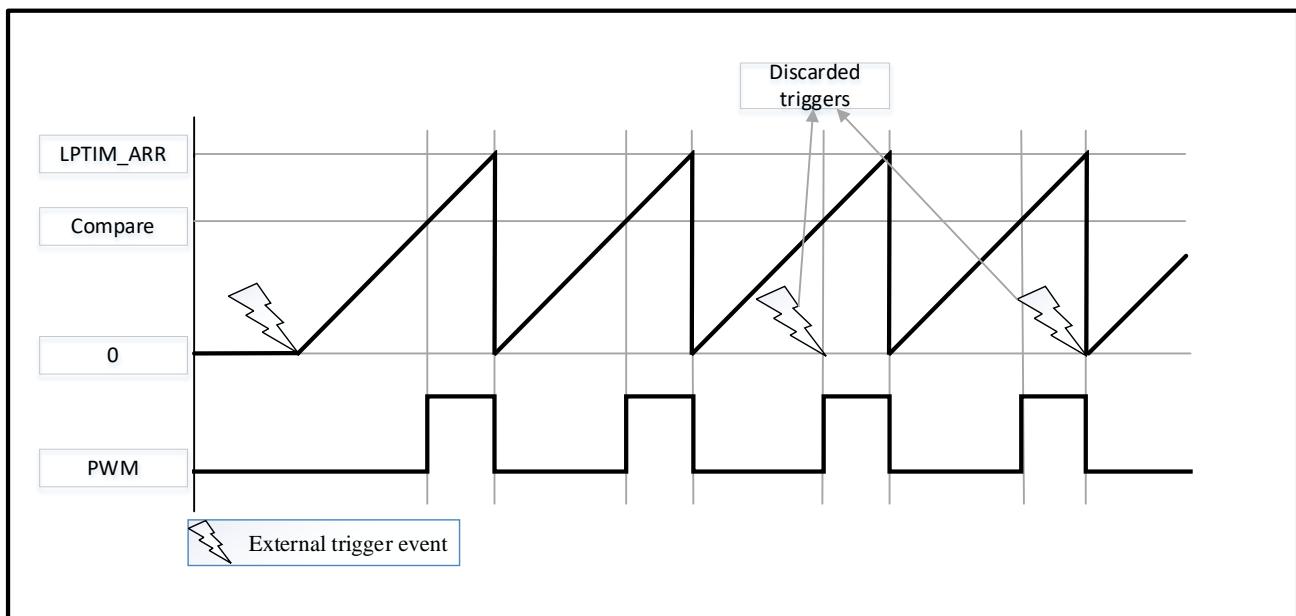
- Continuous mode: A trigger event will start the LPTIM and it will continue running until the user switched off the LPTIM.
- One-shot mode: A trigger event will start the LPTIM and it will stop when the counter value reached LPTIM\_ARR.ARRVAL[15:0].

### Continuous mode:

LPTIM\_CTRL.TSTCM bit must be set to enable the continuous mode. If LPTIM uses external trigger, the internal counter will start when an external trigger event arrives after LPTIM\_CTRL.TSTCM bit is set. After the continuous mode starts, hardware will discard any subsequent external trigger event.

If software trigger is used, setting LPTIM\_CTRL.TSTCM bit will start the internal counter for continuous mode. Any subsequent external trigger event will be discarded as shown in Figure 12-3.

Figure 12-3 LPTIM output waveform, Continuous counting mode configuration



LPTIM\_CTRL.SNGMST and LPTIM\_CTRL.TSTCM bits can only be set when the timer is enabled (The LPTIM\_CTRL.LPTIMEN bit is set to ‘1’).

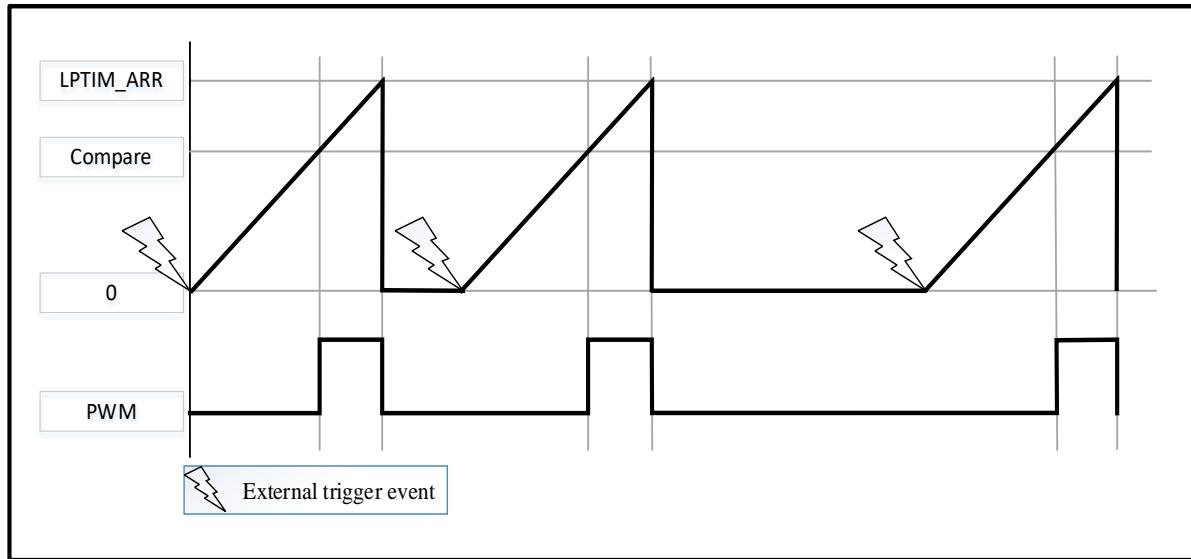
It is possible to switch from one-shot mode to continuous mode. Setting LPTIM\_CTRL.SNGMST bit will switch the LPTIM to one-shot counting mode if continuous counting mode was previously selected. The counter stops as soon as it reaches the LPTIM\_ARR register value if timer enable. If the one-shot counting mode was previously selected, setting LPTIM\_CTRL.TSTCM bit to 1 will switch the LPTIM to continuous counting mode. Counter will restart as soon as LPTIM\_ARR register value is reached if timer enable.

#### One-shot mode:

LPTIM\_CTRL.SNGMST bit must be set to enable the one-shot mode. A trigger event will re-start the LPTIM. Hardware will abandon all the trigger events after the internal counter starts and before the counter value equal to LPTIM\_ARR.ARRVAL[15:0] value.

If an external trigger is selected, after each external trigger event that arrives after the LPTIM\_CTRL.SNGMST bit is set, and after the timer register is stopped (containing a zero value), the timer is restarted for a new count cycle, as shown in Figure 12-4.

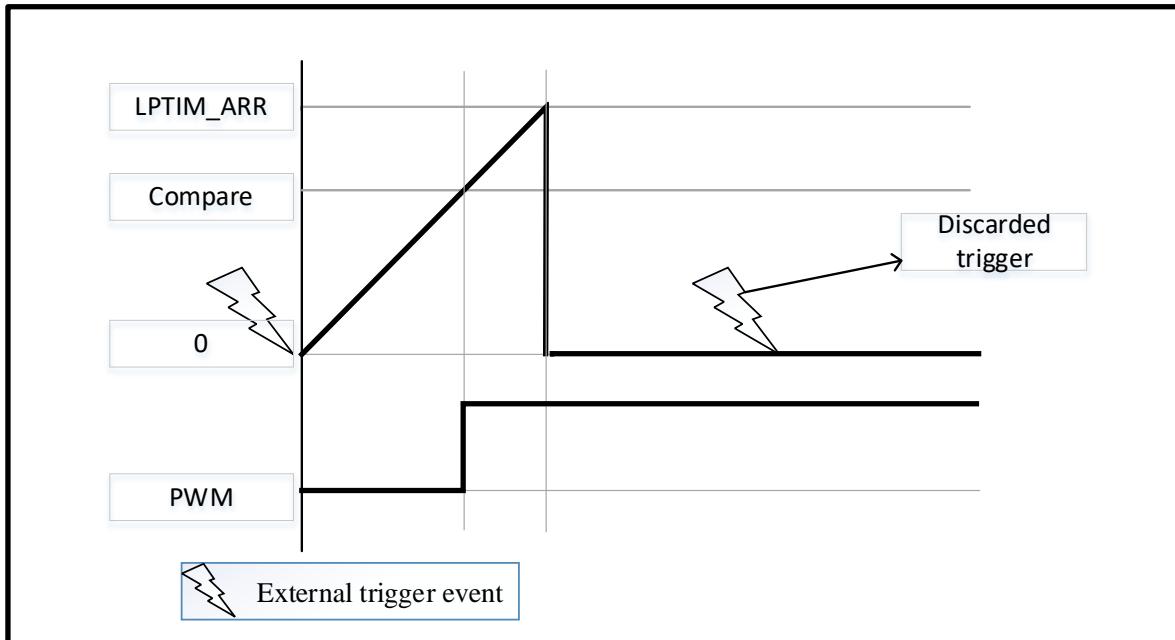
Figure 12-4 PTIM output waveform, single counting mode configuration



One-time mode activated:

The one-time mode is used when the LPTIM\_CFG.WAVE bit is set. In one-time mode, the counter is started once when the first trigger event happens, the hardware will discard any subsequent trigger event. As shown Figure 12-5.

Figure 12-5 LPTIM output waveform, Single counting mode configuration and Set-once mode activated



In case of software start (LPTIM\_CFG.TRGEN[1:0] = '00'), the LPTIM\_CTRL.SNGMST setting will start the counter for one-shot counting.

## 12.4.7 Waveform generation

The LPTIM auto-reload register (LPTIM\_ARR) and compare register (LPTIM\_COMP) are used for generating LPTIM output waveforms.

LPTIM supported waveforms are shown as below:

- PWM waveform: LPTIM output is set when a COMP match event happens. (I.E. the LPTIM\_CNT register value matched the LPTIM\_COMP register value.) The LPTIM output is reset when an ARR match happens. (I.E. the LPTIM\_CNT register value matched the LPTIM\_ARR register value.)
- One-pulse waveform: The first pulse is triggered same as PWM waveform, then the output is permanently reset when the ARR match happens.
- Set-once mode: the output waveform is similar to the One-pulse mode except that the output is kept to the last signal level (depends on the output configured polarity).

Above waveform configuration require that LPTIM\_ARR register value must be configured bigger than the LPTIM\_COMP register value.

The LPTIM output waveform can be configured through the LPTIM\_CFG.WAVE bit as follow:

- Clearing the LPTIM\_CFG.WAVE bit will force the LPTIM to generate a PWM waveform or a single-pulse waveform depending on the set bit (LPTIM\_CTRL.TSTCM or LPTIM\_CTRL.SNGMST).
- LPTIM\_CTRL.WAVE bit equals to '1' forces the LPTIM to generate a Set-once mode waveform.

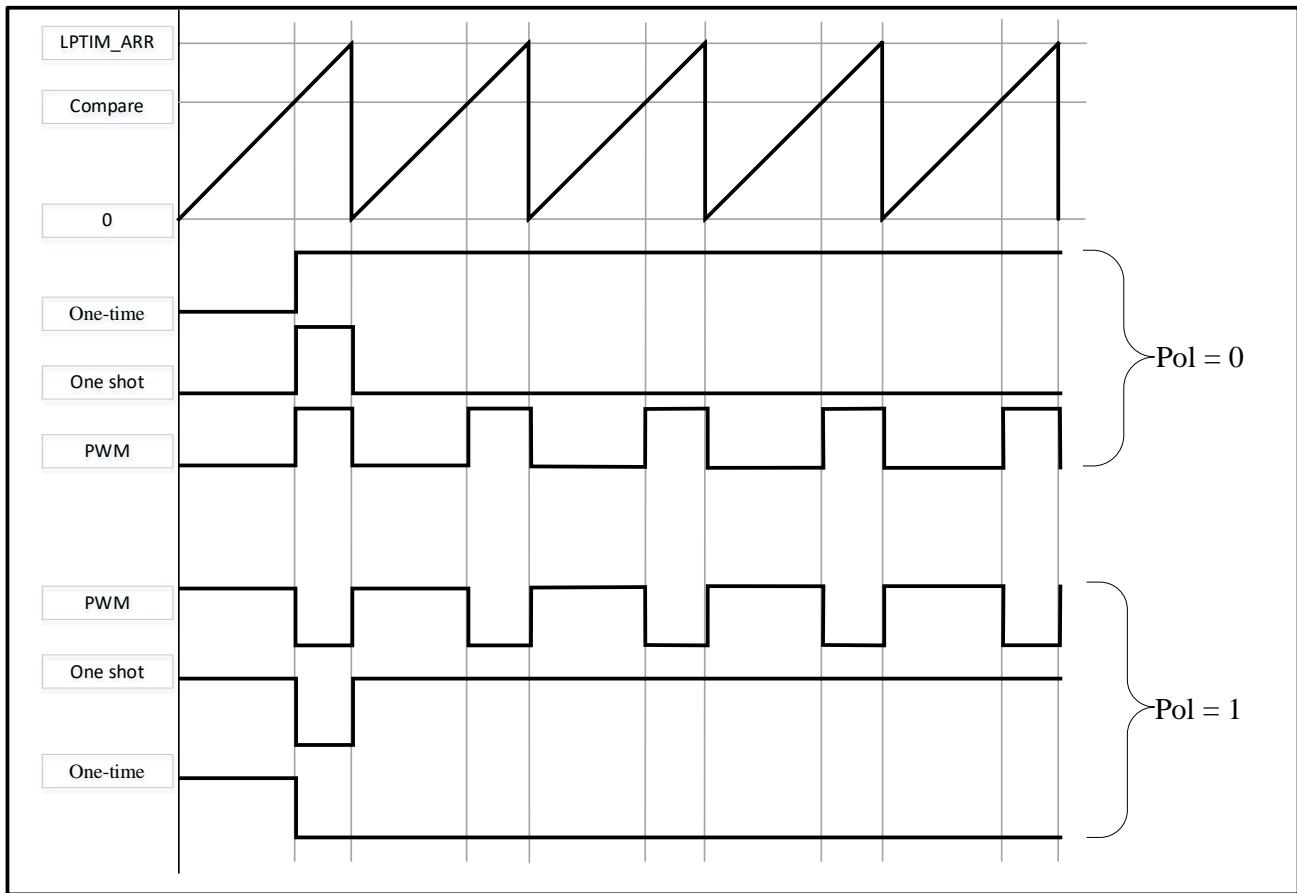
The LPTIM\_CFG.WAVEPOL bit controls LPTIM output polarity. The output idle steady level will change immediately after the user configured the polarity, even when the timer is disabled.

Signals with frequencies up to the LPTIM clock frequency divided by 2 can be generated. Only when LPTIM counter counting internal clock active edge can achieve clock frequency divided by 2.

(I.E. LPTIM\_CFG.CLKSEL = 0, LPTIM\_CFG.CLKPOL[1:0] = 10, LPTIM\_COMP.CMPVAL[15:0] = 'd1 (50% duty cycle) / 'd2, LPTIM\_ARR.ARRVAL[15:0] = 'd2. d1 and d2 means decimal 1, 2)

Figure 12-6 below shows the three possible waveforms that can be generated on the LPTIM output. Also, it shows the effect of the polarity change using the LPTIM\_CFG.WAVEPOL bit.

Figure 12-6 Waveform generation



#### 12.4.8 Register update

The LPTIM\_ARR register and LPTIM\_COMP register can be updated after the software writes. If the LPTIM is started, the LPTIM\_ARR register and LPTIM\_COMP register can be updated when counter overflow.

The LPTIM APB interface and the LPTIM kernel logic use different clocks, so there is some latency between the software write through APB bus and the moment when these values are available to the kernel logic. During this latency period, any additional write into these registers must be avoided.

The update method of LPTIM\_ARR and LPTIM\_COMP registers is determined by the LPTIM\_CFG.RELOAD bit:

- LPTIM CFG.RELOAD bit equals to '1': LPTIM ARR and LPTIM COMP registers are updated when counter overflow, if the LPTIM already started. When counter overflow, latency = 2~3 APB clock period.
- LPTIM CFG.RELOAD bit equals to '0': LPTIM ARR and LPTIM COMP registers are updated after any software write access. Latency = 2~3 APB clock period + 2~3 LPTIM internal pre-scaled clock period.

The LPTIM\_INTSTS.ARRUPD flag and the LPTIM\_INTSTS.CMPUPD flag indicate when the write operation is completed to respectively the LPTIM ARR register and the LPTIM COMP register.

After a write to the LPTIM ARR register or the LPTIM COMP register, any successive write before respectively the LPTIM\_INTSTS.ARRUPD flag or the LPTIM\_INTSTS.CMPUPD flag be set, will lead to unpredictable results. So

a new write operation to the same register can only be performed when the previous write operation is completed.

### 12.4.9 Counter mode

The internal counter can count external trigger events from LPTIM input1 or internal clock cycles. This can be configured through LPTIM\_CFG.CLKSEL and LPTIM\_CFG.CNTMEN bits.

If LPTIM is counting external triggers, user can configure LPTIM\_CFG.CLKPOL[1:0] bits to select the active edge from rising edge, falling edge or both edges.

The count modes below can be selected, depending on LPTIM\_CFG.CLKSEL and LPTIM\_CFG.CNTMEN bits values:

- LPTIM\_CFG.CLKSEL = 0: the LPTIM use an internal clock source to clock.
  - LPTIM\_CFG.CNTMEN = 0, The LPTIM is configured to be clocked by an internal clock source and the LPTIM counter is configured to be updated following each internal clock pulse.
  - LPTIM\_CFG.CNTMEN = 1, The LPTIM external Input1 is sampled with the internal clock provided to the LPTIM. In order to not miss any event, the frequency of the changes on the external Input1 signal should never exceed the frequency of the internal clock provided to the LPTIM. Also, the internal clock provided to the LPTIM must not be pre-scaled (LPTIM\_CFG.CLKPREG[2:0] = 000).
- LPTIM\_CFG.CLKSEL = 1: the LPTIM use an external clock source to clock.
  - LPTIM\_CFG.CNTMEN bit value is don't care. In this configuration, the LPTIM has no need for an internal clock source (except if the glitch filters are enabled). The signal injected on the LPTIM external Input1 is used as system clock for the LPTIM. This configuration is suitable for operation modes where no embedded oscillator is enabled.
  - For this configuration, the LPTIM counter can be updated either on rising edges or falling edges of the input1 clock signal but not on both rising and falling edges.
  - Since the signal injected on the LPTIM external Input1 is also used to clock the LPTIM kernel logic, there is some initial latency (after the LPTIM is enabled) before the counter is incremented. More precisely, the first two to five active edges on the LPTIM external Input1 (after LPTIM is enable) are lost.

### 12.4.10 Encoder mode

The Encoder mode can handle signals from quadrature encoders which used to detect angular position of rotary elements. The encoder mode allows the counter counts the events within 0 and LPTIM\_ARR.ARRVAL[15:0] value. (0 up to LPTIM\_ARR.ARRVAL[15:0] or LPTIM\_ARR.ARRVAL[15:0] to 0). In this case, user must configure LPTIM\_ARR.ARRVAL[15:0] before enable the counter. From external input1 and input2, a clock is generated for the counter. The counting direction depends on the phase between these two input signals.

The Encoder mode is only available when the LPTIM use an internal clock source to clock. The signals frequency on both Input1 and Input2 inputs must not exceed the LPTIM internal clock frequency divided by 4. This is mandatory in order to guarantee a proper operation of the LPTIM.

The change of counting direction is updated by LPTIM\_INTSTS.DOWN and LPTIM\_INTSTS.UP flags. Also, an

interrupt can be generated for both direction change events if enabled through the LPTIM\_INTEN register.

User can enable Encoder mode by setting LPTIM\_CFG.ENC bit. And the LPTIM need to be configured in continuous mode first.

When Encoder mode is active, the LPTIM counter is modified automatically following the speed and the direction of the incremental encoder. Therefore, its content always represents the encoder's position. The count direction, signaled by LPTIM\_INTSTS.DOWN and LPTIM\_INTSTS.UP flags, correspond to the rotation direction of the encoder rotor.

According to the edge polarity configured using the LPTIM\_CFG.CLKPOL[1:0] bits, different counting scenarios are possible. The following table summarizes the possible combinations, assuming that Input1 and Input2 do not switch at the same time.

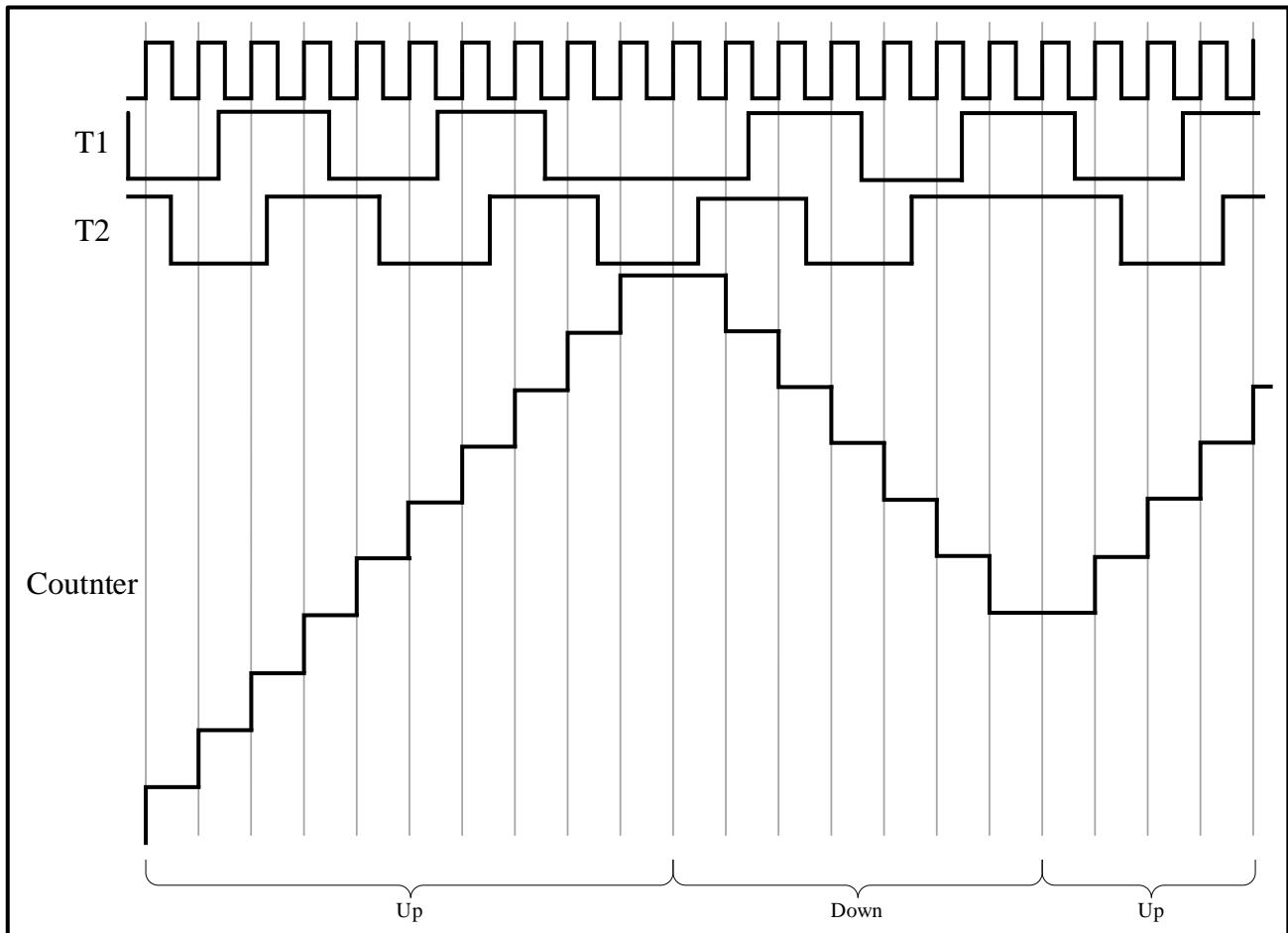
Table 12-3 Encoder counting scenarios

Trigger edge	The signal is opposite (Input1 for Input2, Input2 for Input1)	Input1 signal		Input2 signal	
		Rising	Falling	Rising	Falling
Rising Edge	High	Down	No count	Up	No count
	Low	Up	No count	Down	No count
Falling Edge	High	No count	Up	No count	Down
	Low	No count	Down	No count	Up
Both Edges	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

The following figure shows a counting sequence for Encoder mode where both-edge polarity is configured.

**Caution:** In this mode the LPTIM must be clocked by an internal clock source, so the LPTIM\_CFG.CLKSEL bit must be maintained to its reset value which is equal to '0'. Also, the prescaler division ratio must be equal to its reset value which is 1 (LPTIM\_CFG.CLKPREG[2:0] bits must be '000').

Figure 12-7 Encoder mode counting sequence



### 12.4.11 Non-orthogonal encoder mode

This mode allows handling signals from non-quadrature encoders, which is used to detect subsequent positive pulses from external interface. Non-Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value programmed into the LPTIM\_ARR register (0 up to ARR or ARR down to 0 depending on the direction). Therefore you must configure LPTIM\_ARR before starting. From the two external input signals, Input1 and Input2, a clock signal is generated to clock the LPTIM counter. The order between those two signals determines the counting direction.

The Non-Encoder mode is only available when the LPTIM is clocked by an internal clock source. The signals frequency on both Input1 and Input2 inputs must not exceed the LPTIM internal clock frequency divided by 4. This is mandatory in order to guarantee a proper operation of the LPTIM.

Direction change is signalized by LPTIM\_INTSTS.DOWN and LPTIM\_INTSTS.UP flags. Also, an interrupt can be generated for both direction change events if enabled through the LPTIM\_INTEN register.

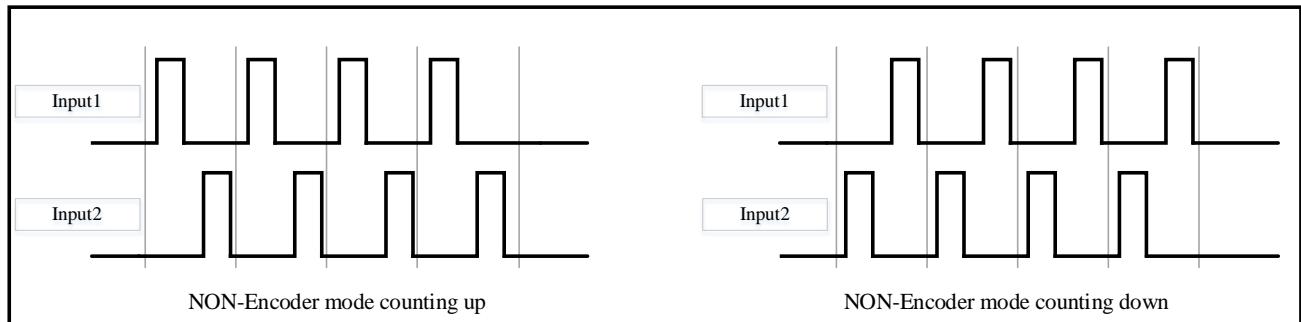
To activate the Non-Encoder mode the LPTIM\_CFG.NENC bit has to be set to '1'. The LPTIM must first be configured in Continuous mode.

When Non-Encoder mode is active, the LPTIM counter is modified automatically following the speed and the

direction of the incremental encoder. Therefore, its content always represents the encoder's position. The count direction, signaled by LPTIM\_INTSTS.DOWN and LPTIM\_INTSTS.UP flags, correspond to the rotation direction of the encoder rotor.

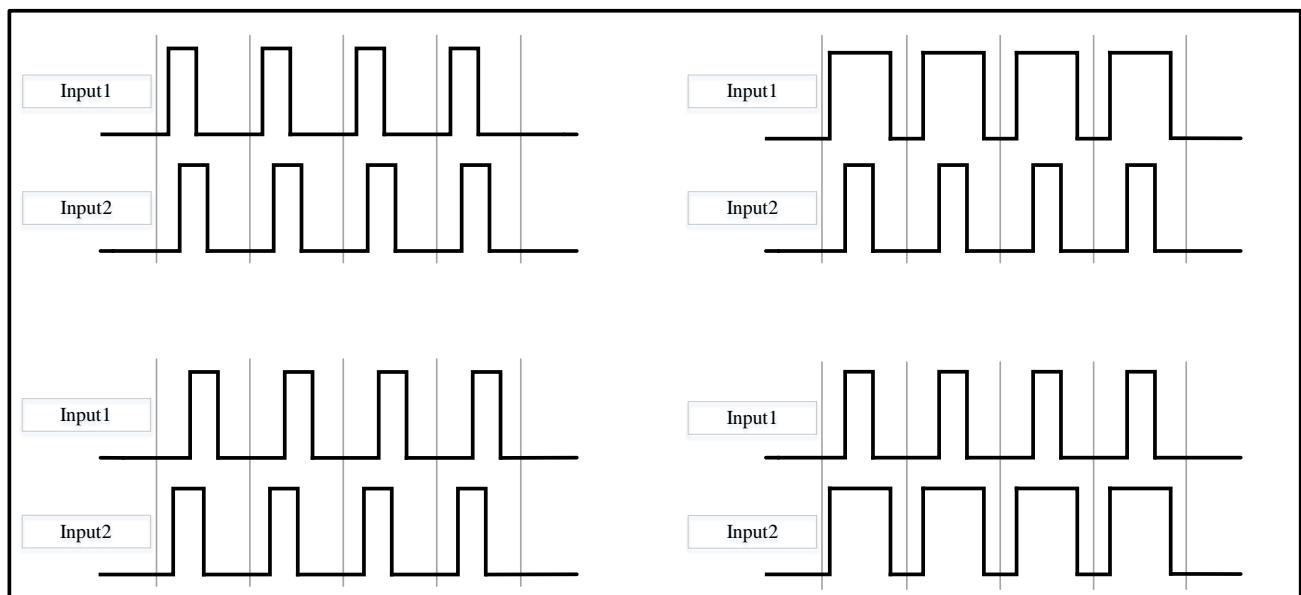
The following two waveforms, the decoder module can work properly, when there is no case that both Input1 and Input2 are high.

Figure 12-8 Input waveforms of Input1 and Input2 when the decoder module is working normally



If the Input1 and Input2 waveform is as following, the decoder module can't work properly. The counter will ignore these waveforms and keep the previous value.

Figure 12-9 Input1 and Input2 input waveforms when decoder module is not working



### 12.4.12 Timeout function

When LPTIM\_CFG.TIMOUTEN bit is enable, the LPTIM counter will be reset by an active edge from one selected trigger input.

When timeout function is used, the LPTIM counter will be reset and re-start by a selected trigger input event. If no trigger occurs within the configured time, the compare match event will happen. The waiting time is configured through the timeout value.

### 12.4.13 LPTIM interrupts

The following events generate an interrupt/wake-up event, if they are enabled through the LPTIM\_INTEN register:

- Compare match
- Auto-reload match (whatever the direction if encoder mode)
- External trigger event
- Auto-reload register write completed
- Compare register write completed
- Direction change (encoder mode), programmable(up / down / both).

*Note: If any bit in the LPTIM\_INTEN register (Interrupt Enable Register) is set after that its corresponding flag in the LPTIM\_INTSTS register (Status Register) is set, the interrupt is not asserted.*

Table 12-4 Interruption events

Corresponding interrupt event	Describe
Compare match	Interrupt flag is set when LPTIM_CNT (counter register value) = LPTIM_COMP (compare register value).
Auto reload match	Interrupt flag is set when LPTIM_CNT (counter register value) = LPTIM_ARR (auto-reload register value).
External trigger event	Interrupt flag is set when an external trigger event is detected.
Auto-reload register update OK	Interrupt flag is set when the write operation to the LPTIM_ARR register is complete.
Compare register update OK	Interrupt flag is set when the write operation to the LPTIM_COMP register is complete.
Direction change	Used in Encoder mode. Two interrupt flags are embedded to signal direction change: –UP flag indicated that the count direction is changed to count up. –DOWN flag indicated that the count direction is changed to count down.

## 12.5 LPTIM registers

### 12.5.1 LPTIM register overview

Table 12-5 LPTIM register overview

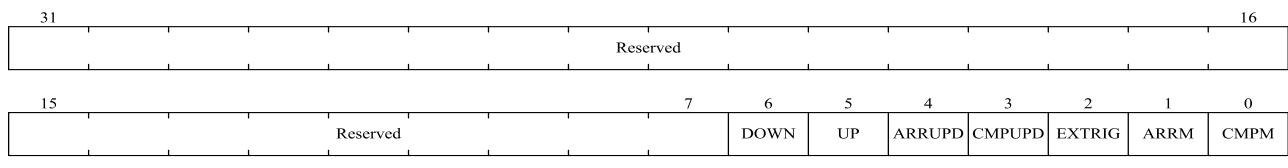
Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	DOWN	UP	ARRUPD	CMPUPD	EXTTRIG	ARRM	CMPM		
000h	LPTIM_INTSTS																										0	0	0	0	0	0	0		
	Reset Value																																		

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
004h	LPTIM_INTCLR																																			
	Reset Value																																			
008h	LPTIM_INTEN																																			
	Reset Value																																			
00Ch	LPTIM_CFG																																			
	Reset Value																																			
010h	LPTIM_CTRL																																			
	Reset Value																																			
014h	LPTIM_CMP																																			
	Reset Value																																			
018h	LPTIM_ARR																																			
	Reset Value																																			
01Ch	LPTIM_CNT																																			
	Reset Value																																			

## 12.5.2 LPTIM interrupt and status register (LPTIM\_INTSTS)

Address offset: 0x00

Reset value: 0x0000 0000



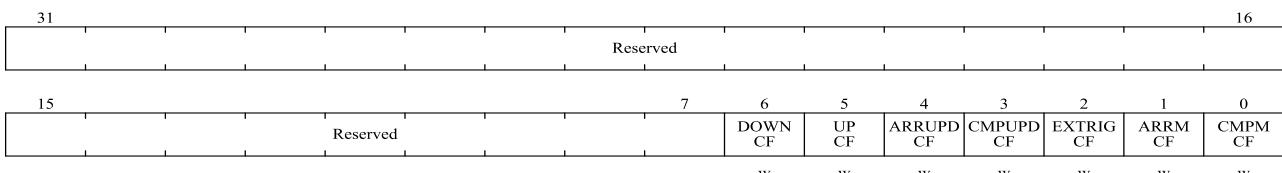
Bit Field	Name	Description
31:7	Reserved	Reserved, the reset value must be maintained.

Bit Field	Name	Description
6	DOWN	Change counter direction to down. In Encoder mode, hardware will set DOWN bit to inform the application the counter direction.
5	UP	Change counter direction to up. In Encoder mode, hardware will set UP bit to inform the application the counter direction.
4	ARRUPD	Auto-reload value updated to register. Hardware sets ARRUPD to inform application that LPTIM_ARR register has been written by the APB1 bus successfully. For more details, see 12.4.8.
3	CMPUPD	Compare value updated to register. Hardware sets COMPUPD to inform application that LPTIM_COMP register has been written by the APB1 bus successfully. For more details, see 12.4.8.
2	EXTRIG	External trigger valid event. Hardware sets EXTRIG to inform application that a valid external trigger edge has occurred. If the trigger is discarded when timer has already started, then this flag is not set.
1	ARRM	Auto-reload match. Hardware set this to inform application that LPTIM_CNT register value reached the LPTIM_ARR register's value.
0	CMPM	Compare match. Hardware set this to inform application that LPTIM_CNT register value reached the LPTIM_COMP register's value.

### 12.5.3 LPTIM interrupt clear register (LPTIM\_INTCLR)

Address offset: 0x04

Reset value: 0x0000 0000



Bit Field	Name	Description
31□7	Reserved	Reserved, the reset value must be maintained.
6	DOWNCF	Direction change to down Clear Flag Writing 1 to this bit clear the DOWN flag in the LPTIM_INTSTS register
5	UPCF	Direction change to UP Clear Flag Writing 1 to this bit clear the UP flag in the LPTIM_INTSTS register

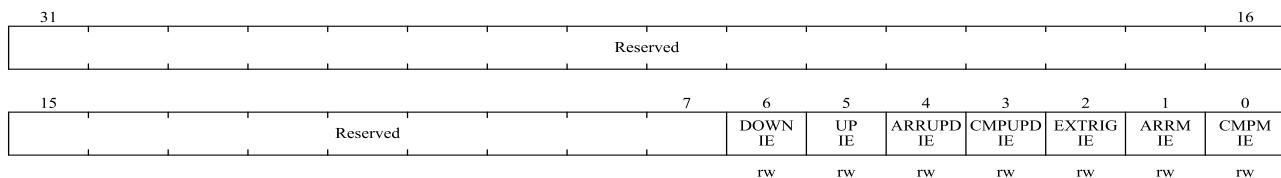
Bit Field	Name	Description
4	ARRUPDCF	Auto-reload register update OK Clear Flag Writing 1 to this bit clears the ARRUPD flag in the LPTIM_INTSTS register
3	CMPUPDCF	Compare register update OK Clear Flag Writing 1 to this bit clears the CMPUPD flag in the LPTIM_INTSTS register
2	EXTRIGCF	External trigger valid edge Clear Flag Writing 1 to this bit clears the EXTRIG flag in the LPTIM_INTSTS register
1	ARRMCF	Auto-reload match Clear Flag Writing 1 to this bit clears the ARRM flag in the LPTIM_INTSTS register
0	CMPMCF	Compare match Clear Flag Writing 1 to this bit clears the CMPM flag in the LPTIM_INTSTS register

## 12.5.4 LPTIM interrupt enable register (LPTIM\_INTEN)

Address offset: 0x08

Reset value: 0x0000 0000

*Note: The LPTIM\_INTEN register must only be modified when the LPTIM is disabled (LPTIM\_CTRL.LPTIMEN bit reset to '0')*



Bit Field	Name	Description
31:7	Reserved	Reserved, the reset value must be maintained.
6	DOWNIE	Direction change to down interrupt enable 0: DOWN interrupt disabled 1: DOWN interrupt enabled
5	UPIE	Direction change to up interrupt enable 0: UP interrupt disabled 1: UP interrupt enabled
4	ARRUPDIE	Auto reload register update succeeded interrupt enable bit. 0: ARRUPD interrupt disable 1: ARRUPD interrupt enable
3	CMPUPDIE	Compare register update OK interrupt enable 0: CMPUPD interrupt disabled 1: CMPUPD interrupt enabled
2	EXTRIGIE	External trigger valid edge interrupt enable 0: EXTRIG interrupt disabled 1: EXTRIG interrupt enabled
1	ARRMIE	Auto reload match interrupt enable bit. 0: ARRM interrupt disabled

Bit Field	Name	Description
		1: ARRM interrupt enabled
0	CMPMIE	Compare match interrupt enable 0: CMPM interrupt disabled 1: CMPM interrupt enabled

## 12.5.5 LPTIM configuration register (LPTIM\_CFG)

Address offset: 0x0C

Reset value: 0x0000 0000

*Note: The LPTIM\_CFG register must only be modified when the LPTIM is disabled (LPTIM\_CTRL.LPTIMEN bit reset to '0')*

31	Reserved				26	NENC	25	ENC	24	CNTMEN	23	RELOAD	22	WAVEPOL	21	WAVE	20	TIMOUT	19	EN	18	TRGEN[1:0]	17	Reserved	16
15	13	12	11		rw	9	rw	8	rw	7	rw	6	rw	5	rw	4	rw	3	rw	2	rw	1	0		
	TRGSEL[2:0]	Reserved		CLKPRE[2:0]	rw	Reserved		TRIGFLT[1:0]	rw	Reserved		CLKFLT[1:0]	rw	CLKPOL[1:0]	rw	CLKSEL	rw								

Bit Field	Name	Description
31:26	Reserved	Reserved, the reset value must be maintained.
25	NENC	Non-Orthogonal mode enable 0: Non-Orthogonal mode disabled 1: Non-Orthogonal mode enabled
24	ENC	Encoder mode enable 0: Encoder mode disabled 1: Encoder mode enabled
23	CNTMEN	Counter mode enabled The CNTMEN bit selects clock source for the LPTIM counter: 0: Counter is incremented following each internal clock pulse 1: Counter is incremented following each valid clock pulse on the LPTIM external Input1
22	RELOAD	Registers update mode The RELOAD bit controls the LPTIM_ARR and the LPTIM_COMP registers update mode 0: Registers are updated after each APB1 bus write access 1: Registers are updated at the end of the current LPTIM period
21	WAVEPOL	Waveform shape polarity The WAVEPOL bit controls the output polarity 0: The LPTIM output reflects the compare results between LPTIM_ARR and LPTIM_COMP registers 1: The LPTIM output reflects the inverse of the compare results between LPTIM_ARR and LPTIM_COMP registers

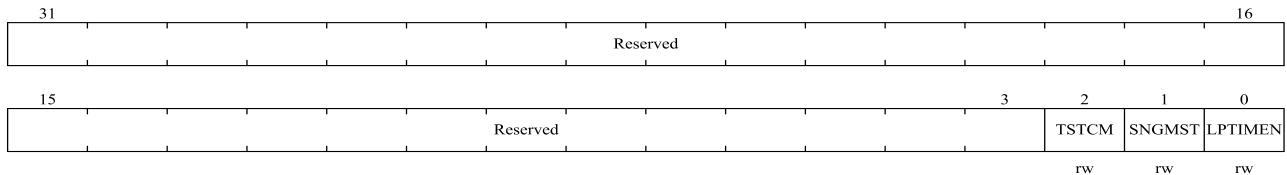
Bit Field	Name	Description
20	WAVE	<p>Waveform shape</p> <p>The WAVE bit controls the output shape</p> <p>0: Deactivate Set-once mode, PWM/One Pulse waveform (depending on LPTIM_CTRL.TSTCM or LPTIM_CTRL.SNGMST bit)</p> <p>1: Activate the Set-once mode</p>
19	TIMOUTEN	<p>Timeout enable</p> <p>0: A trigger event arriving when the timer is already started will be ignored</p> <p>1: A trigger event arriving when the timer is already started will reset and restart the counter</p>
18:17	TRGEN[1:0]	<p>Trigger enable and polarity</p> <p>The TRGEN bits controls whether the LPTIM counter is started by an external trigger or not. If the external trigger option is selected, three configurations are possible for the trigger active edge:</p> <p>00: Software trigger (counting start is initiated by software)</p> <p>01: Rising edge is the active edge</p> <p>02: Falling edge is the active edge</p> <p>03: Both edges are active edges</p>
16	Reserved	Reserved, the reset value must be maintained.
15:13	TRGSEL[2:0]	<p>Trigger selector</p> <p>The trigger source is selected from the following 6 available sources as the trigger event for LPTIM:</p> <ul style="list-style-type: none"> <li>000: PB6 or PA6</li> <li>001: RTC alarm A</li> <li>010: RTC alarm B</li> <li>011: RTC_TAMP1</li> <li>100: RTC_TAMP2</li> <li>101: Reserved</li> <li>110: COMP_OUT</li> <li>111: Reserved</li> </ul>
12	Reserved	Reserved, the reset value must be maintained.
11:9	CLKPRE[2:0]	<p>Clock division factor bit.</p> <ul style="list-style-type: none"> <li>000: / 1</li> <li>001: / 2</li> <li>010: / 4</li> <li>011: / 8</li> <li>100: / 16</li> <li>101: / 32</li> <li>110: / 64</li> <li>111: / 128</li> </ul>
8	Reserved	Reserved, the reset value must be maintained.
7:6	TRIGFLT[1:0]	Configure the data filter trigger bit.

Bit Field	Name	Description
		<p>The TRIGFLT value sets the number of consecutive equal samples that should be detected when a level change occurs on an internal trigger before it is considered as a valid level transition. An internal clock source must be present to use this feature</p> <p>00: Any trigger active level change is considered as a valid trigger.</p> <p>01: Trigger active level change must be stable for at least 2 clock periods before it is considered as valid trigger.</p> <p>10: Trigger active level change must be stable for at least 4 clock periods before it is considered as valid trigger.</p> <p>11: Trigger active level change must be stable for at least 8 clock periods before it is considered as valid trigger.</p>
5	Reserved	Reserved, the reset value must be maintained.
4:3	CLKFLT[1:0]	<p>Digital filter external clock input configuration</p> <p>The CLKFLT value sets the number of consecutive equal samples that should be detected when a level change occurs on an external clock signal before it is considered as a valid level transition. An internal clock source must be present to use this feature</p> <p>00: Any external clock signal level change is considered as a valid transition.</p> <p>01: External clock signal level change must be stable for at least 2 clock periods before it is considered as valid transition.</p> <p>10: External clock signal level change must be stable for at least 4 clock periods before it is considered as valid transition.</p> <p>11: External clock signal level change must be stable for at least 8 clock periods before it is considered as valid transition.</p>
2:1	CLKPOL[1:0]	<p>Clock polarity</p> <p>When the LPTIM is clocked by an external clock source, CLKPOL bits is used to configure the active edge or edges used by the counter:</p> <p>00: The rising edge is the active edge used for counting</p> <p>01: The falling edge is the active edge used for counting</p> <p>10: Both edges are active edges. When both external clock signal edges are considered active ones, the LPTIM must also be clocked by an internal clock source with a frequency equal to at least four time the external clock frequency.</p> <p>11: Not allowed</p> <p>If the LPTIM is configured in Encoder mode (LPTIM_CFG.ENC bit is set):</p> <p>00: The encoder rising edge counting mode.</p> <p>01: The encoder falling edge counting mode.</p> <p>02: The encoder both edges counting mode.</p>
0	CLKSEL	<p>Clock selector</p> <p>The CLKSEL bit selects which clock source the LPTIM will use:</p> <p>0: LPTIM is clocked by internal clock source (APB1 clock or any of the embedded oscillators)</p> <p>1: LPTIM is clocked by an external clock source through the LPTIM external Input1</p>

## 12.5.6 LPTIM control register (LPTIM\_CTRL)

Address offset: 0x10

Reset value: 0x0000 0000



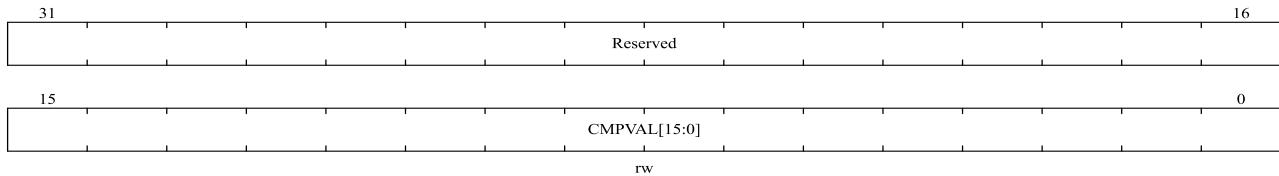
Bit Field	Name	Description
31:3	Reserved	Reserved, the reset value must be maintained.
2	TSTCM	<p>Timer start in Continuous mode This bit is set by software and cleared by hardware. In case of software start (LPTIM_CFG.TRGEN[1:0] = '00'), setting this bit starts the LPTIM in Continuous mode. If the software start is disabled (TRGEN[1:0] ≠ '00'), setting this bit starts the timer in Continuous mode as soon as an external trigger is detected. If this bit is set when a single pulse mode counting is ongoing, then the timer will not stop at the next match between the LPTIM_ARR and LPTIM_CNT registers and the LPTIM counter keeps counting in Continuous mode. This bit can be set only when the LPTIM is enabled. It will be automatically reset by hardware.</p>
1	SNGMST	<p>LPTIM start in Single mode This bit is set by software and cleared by hardware. In case of software start (LPTIM_CFG.TRGEN[1:0] = '00'), setting this bit starts the LPTIM in single pulse mode. If the software start is disabled (LPTIM_CFG.TRGEN[1:0] ≠ '00'), setting this bit starts the LPTIM in single pulse mode as soon as an external trigger is detected. If this bit is set when the LPTIM is in continuous counting mode, then the LPTIM will stop at the following match between LPTIM_ARR and LPTIM_CNT registers. This bit can only be set when the LPTIM is enabled. It will be automatically reset by hardware.</p>
0	LPTIMEN	<p>LPTIM enable The LPTIMEN bit is set and cleared by software. 0: LPTIM is disabled 1: LPTIM is enabled</p>

## 12.5.7 LPTIM compare register (LPTIM\_COMP)

Address offset: 0x14

Reset value: 0x0000 0000

*Note: The LPTIM\_COMP register must only be modified when the LPTIM is enabled (LPTIM\_CTRL.LPTIMEN bit reset to '1')*



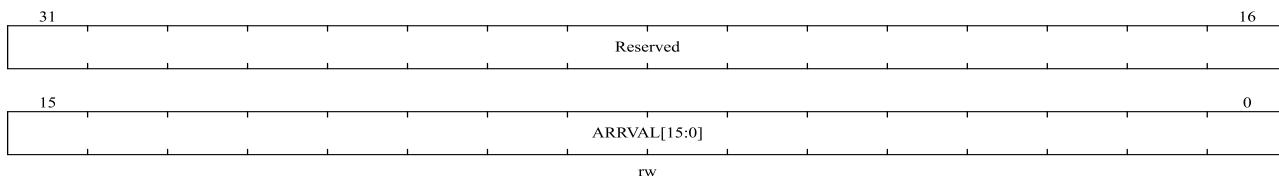
Bit Field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained.
15:0	CMPVAL[15:0]	Compare value CMPVAL is the compare value used by the LPTIM.

## 12.5.8 LPTIM auto-reload register (LPTIM\_ARR)

Address offset: 0x18

Reset value: 0x0000 0001

*Note: The LPTIM\_ARR register must only be modified when the LPTIM is enabled (LPTIM\_CTRL.LPTIMEN bit reset to '1')*

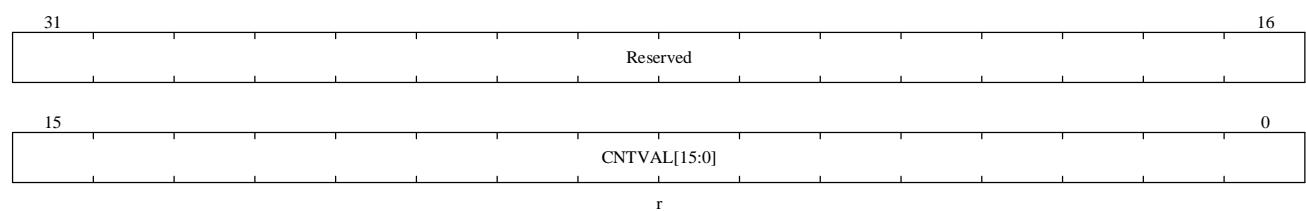


Bit Field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained.
15:0	ARRVAL[15:0]	Auto reload value ARRVAL is the auto-reload value for the LPTIM. This value must be strictly greater than the LPTIM_COMP.CMPVAL[15:0] value.

## 12.5.9 LPTIM counter register (LPTIM\_CNT)

Address offset: 0x1C

Reset value: 0x0000 0000



Bit Field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained.

Bit Field	Name	Description
15:0	CNTVAL[15:0]	<p>Counter value</p> <p>When the LPTIM is running with an asynchronous clock, reading the LPTIM_CNT register may return unreliable values. So in this case it is necessary to perform two consecutive read accesses and verify that the two returned values are identical.</p> <p>If identical, the reading is reliable.</p>

## 13 Independent watchdog (IWDG)

### 13.1 Introduction

The N32G031 has built-in independent watchdog (IWDG) and window watchdog (WWWDG) timers to solve the problems caused by software errors. Watchdog timer is very flexible to use, which improves the security of the system and the accuracy of timing control.

Independent Watchdog (IWDG) is driving by Low-speed internal clock (LSI clock) running at 30 KHz, which will still running event dead loop or MCU stuck is happening. This can provide higher safety level, timing accuracy and flexibility of watchdog. It can reset and resolve system malfunctions due to software failure. The IWDG is best suited for applications that require the watchdog to run as a totally independent process outside the main application, but have lower timing accuracy constraints.

When the power control register PWR\_CTRL2.IWDGRSTEN bit is '1' and the IWDG counter reaches 0, a system reset will be generated (if this bit is '0', the IWDG will count but not reset). IWDG reset can also be used for low power wake up.

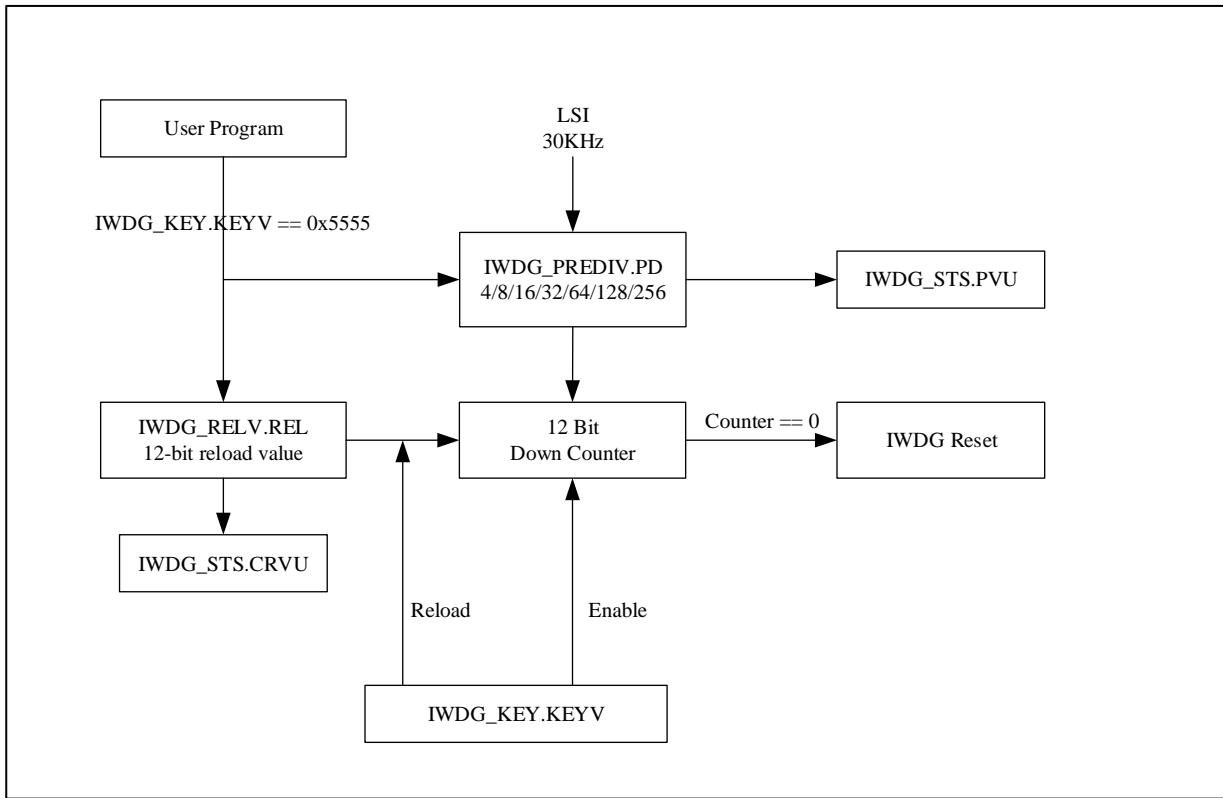
*Note: This chapter is based on the system default value IWDGRSTEN = 1.*

### 13.2 Main features

- A 12-bit down-counter that runs independently
- RC oscillator provides an independent clock source and can work normally in STOP mode
- Reset and low-power wake-up can be matched.
- A system reset occurs when the down counter reaches 0x0000(if watchdog activated)

## 13.3 Function description

Figure 13-1 Functional block diagram of the independent watchdog module



To enable IWDG, we need to write 0xCCCC to IWDG\_KEY.KEYV[15:0] bits. Counter starts counting down from reset value (0xFFFF). When counter count to 0x000, it generates a reset signal (IWDG\_RESET) to MCU. Other than that, as long as 0xAAAA (reload request) is write to IWDG\_KEY.KEYV[15:0] bits before reset, the counter value is set to the reload value in the IWDG\_RELV.REL[11:0] bits and prevents the watchdog from resetting the entire device.

If the "hardware watchdog timer" function is enabled through the selection byte, the watchdog will automatically start running after the system is powered on and will generate a system reset, unless the software reloads the counter before it reaches '0'.

### 13.3.1 Register access protection

IWDG\_PREDIV and IWDG\_RELV register are write protected. To modify the value of those two register, user needs to write 0x5555 to IWDG\_KEY.KEYV[15:0] bits. Writing other value enables write protections again. IWDG\_STS.PVU indicates whether the pre-scaler value update is on going. IWDG\_STS.CRVU indicates whether the IWDG is updating the reload value. The hardware sets the IWDG\_STS.PVU bit and/or IWDG\_STS.CRVU bit when the pre-scaler value and/or reload value is updating. After the pre-scaler value and/or reload value update is complete, the hardware clears the IWDG\_STS.PVU bit and/or IWDG\_STS.CRVU bit.

The reload operation (IWDG\_KEY.KEYV[15:0] configured with value of 0xAAAA) will also cause the registers to become write protected again.

### 13.3.2 Debug mode

In debug mode (Cortex-M0 core stops), IWDG counter will either continue to work normally or stops, depending on `DBG_CTRL.IWDG_STOP` bit in debug module. If this bit is set to ‘1’, the counter stops. The counter works normally when the bit is ‘0’. For details, see 3.3.2 Peripheral Debugging Support.

## 13.4 User interface

IWDG module user interface contains 4 registers: Key Register (`IWDG_KEY`), Pre-scale Register (`IWDG_PREDIV`), Reload Register (`IWDG_RELV`) and Status Register (`IWDG_STS`).

### 13.4.1 Operate flow

When IWDG is enable from reset from software (write 0xAAAA to `IWDG_KEY.KEYV[15:0]` bits) or hardware (clear `WDG_SW` bit). It starts counting down from 0xFFFF. Down counting gap is determined by pre-scale LSI clock. Once the counter is reloaded, each new round will start from the value in `IWDG_RELV.REL[11:0]` instead of 0xFFFF.

When program is running normally, software needs to feed IWDG before counter reaches 0 and start a new round of down counting. When counter reach 0, this indicates program malfunction. IWDG generates reset signal under this circumstance.

If user wants to configure IWDG pre-scale and reload value register, it needs to write 0x5555 to `IWDG_KEY.KEYV[15:0]` first. Then confirm `IWDG_STS.CRVU` bit and `IWDG_STS.PVU` bit. `IWDG_STS.CRVU` bit indicates reload value update is ongoing, `IWDG_STS.PVU` indicates Pre-scale divider ratio is updating. Only when those two bit are 0 then user can update corresponding value. When update is on-going, hardware sets corresponding bit to 1. At this time, reading `IWDG_PREDIV.PD[2:0]` or `IWDG_RELV.REL[11:0]` is invalid since data needs sync to LSI clock domain. The value read from `IWDG_PREDIV.PD[2:0]` or `IWDG_RELV.REL[11:0]` will be valid after hardware clears the `IWDG_STS.PVU` bit or `IWDG_STS.CRVU` bit.

If the application uses more than one reload value or pre-scaler value, it must wait until the `IWDG_STS.CRVU` bit is reset before changing the reload value, the same as changing the pre-scaler value. However, after updating the pre-scale and/or the reload value, it is not necessary to wait until `IWDG_STS.CRVU` bit or `IWDG_STS.PVU` bit are reset before continuing code execution (even in case of low-power mode entry, the write operation is taken into account and will complete).

Pre-scale register and reload register controls the time that generates reset, as shown in Table 13-1. Table 13-1

Table 13-1 IWDG counting maximum and minimum reset time

Pre-scale factor	PD[2:0]	Min timeout (ms) REL [11:0]=0x000	Max timeout (ms) REL [11:0]=0xFFFF
/ 4	000	0.13	546.13
/ 8	001	0.26	1092.27
/ 16	010	0.53	2184.53
/ 32	011	1.07	4369.07
/ 64	100	2.13	8738.13
/ 128	101	4.26	17476.27
/ 256	11x	8.53	34952.53

## 13.5 IWDG registers

### 13.5.1 IWDG register overview

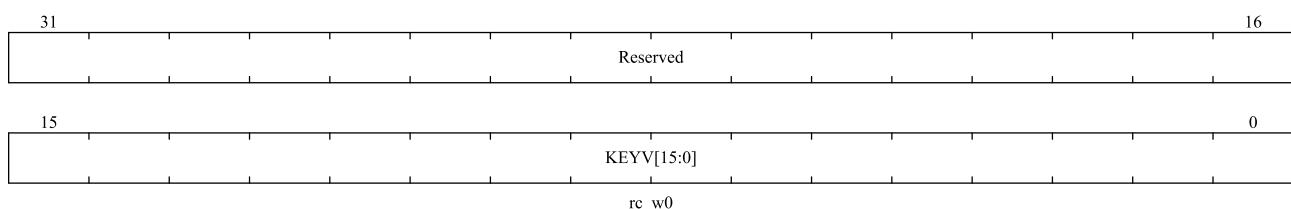
Table 13-2 IWDG register overview

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	IWDG_KEY	Reserved	KEYV[15:0]															KEYV[15:0]															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x04	IWDG_PREDIV	Reserved																									PD[2:0]						
	Reset value																										0	0	0				
0x08	IWDG_RELV	Reserved												REL[11:0]																			
	Reset value													1	1	1	1	1	1	1	1	1	1	1	1	1	1						
0x0C	IWDG_STS	Reserved																										CRVU PVU					
	Reset value	0	0																														

### 13.5.2 IWDG key register (IWDG\_KEY)

Address offset: 0x00

Reset value: 0x00000000

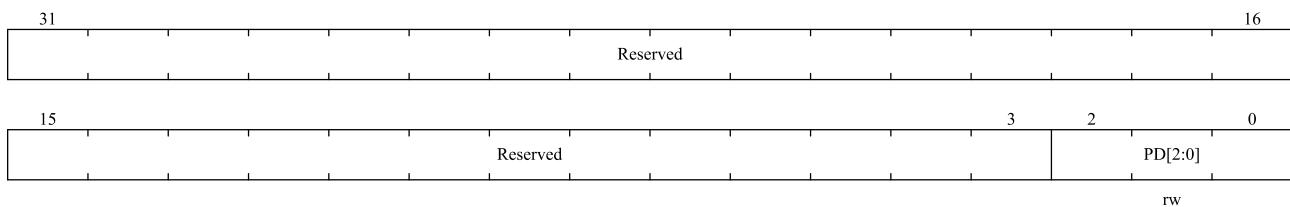


Bit field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained.
15:0	KEYV[15:0]	Key value register: only certain value will serve particular function 0xCCCC: Start watch dog counter, does not have any effect if hardware watchdog is enable, (if hardware watchdog is selected, it is not limited by this command word) 0xAAAA: Reload counter with REL value in IWDG_RELV register to prevent reset. 0x5555: Disable write protection of IWDG_PREDIV and IWDG_RELV register

### 13.5.3 IWDG pre-scaler register (IWDG\_PREDIV)

Address offset: 0x04

Reset value: 0x00000000

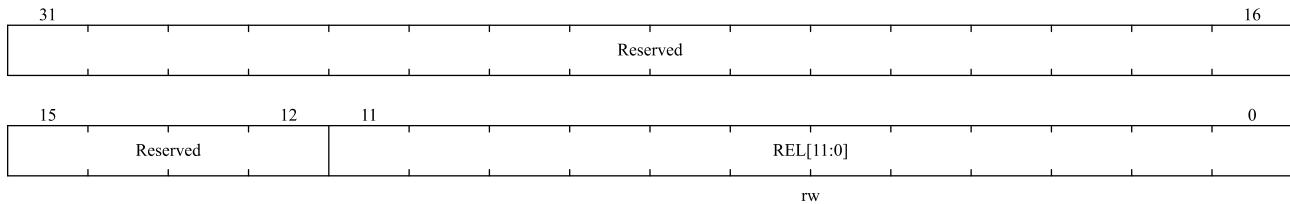


Bit field	Name	Description
31:3	Reserved	Reserved, the reset value must be maintained.
2:0	PD[2:0]	Pre-frequency division factor Pre-scaler divider: with write access protection when IWDG_KEY.KEYV[15:0] is not 0x5555. The IWDG_STS.PVU bit must be 0 otherwise PD [2:0] value cannot be changed. Divide number is as follow: 000: divider /4 001: divider /8 010: divider /16 011: divider /32 100: divider /64 101: divider /128 Other : divider /256 <i>Note: Reading this register will return the pre-divided value from the VDD voltage domain. If a write operation is in progress, the read-back value may be invalid. Therefore, the read value is valid only when the IWDG_STS.PVU bit is '0'.</i>

### 13.5.4 IWDG reload register (IWDG\_RELV)

Address offset: 0x08

Reset value: 0x00000FFF

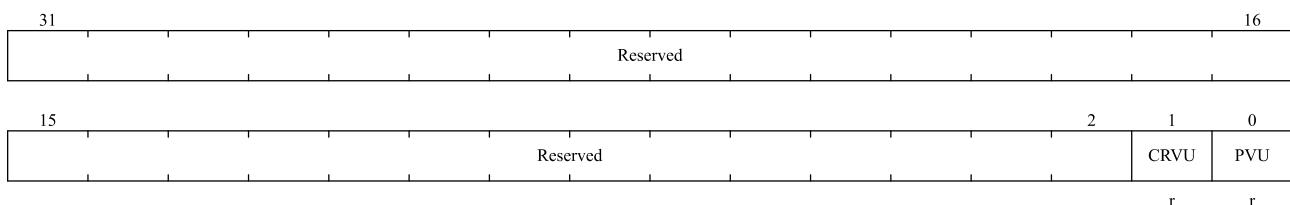


Bit field	Name	Description
31:12	Reserved	Reserved, the reset value must be maintained.
11:0	REL[11:0]	<p>Watchdog counter reload value.</p> <p>With write protection. Defines the reload value of the watchdog counter, which is loaded to the counter every time 0xAAAA is written to IWDG_KEY.KEYV[15:0] bits. The counter then starts to count down from this value. The watchdog timeout period can be calculated from this reloading value and the clock pre-scaler value, refer to Table 13-1.</p> <p>This register can only be modified when the IWDG_STS.CRVU bit is '0'.</p> <p><i>Note: Reading this register will return the reload value from the VDD voltage domain. If a write operation is in progress, the read-back value may be invalid. Therefore, the read value is valid only when the IWDG_STS.CRVU bit is '0'.</i></p>

### 13.5.5 IWDG status register (IWDG\_STS)

Address offset: 0x0C

Reset value: 0x00000000



Bit field	Name	Description
31:2	Reserved	Reserved, the reset value must be maintained.
1	CRVU	<p>Watchdog reload value update</p> <p>Reload Value Update: this bit indicates an update of reload value is ongoing. Set by hardware and clear by hardware. Software can only try to change IWDG_RELV.REL[11:0] value when IWDG_KEY.KEYV[15:0] bits' value is 0x5555 and this bit is 0.</p>
0	PVU	<p>Watchdog pre-scaler value update</p> <p>Pre-scaler Value Update: this bit indicates an update of pre-scaler value is ongoing. Set by hardware and clear by hardware. Software can only try to change IWDG_PREDIV.PD[2:0] value when IWDG_KEY.KEYV[15:0] bits' value is 0x5555 and this bit is 0.</p>

## 14 Window watchdog (WWDG)

### 14.1 Introduction

The clock of the window watchdog (WWDG) is obtained by dividing the APB1 clock frequency by 4096, and whether the program operation is abnormal is detected through the configuration of the time window. Therefore, WWDG is suitable for precise timing, and is often used to monitor software failures caused by external disturbances or unforeseen logic conditions that cause an application to deviate from its normal operating sequence. A system reset occurs when the WWDG down counter is refreshed before reaching the window register value or after the WWDG\_CTRL.T6 bit becomes 0.

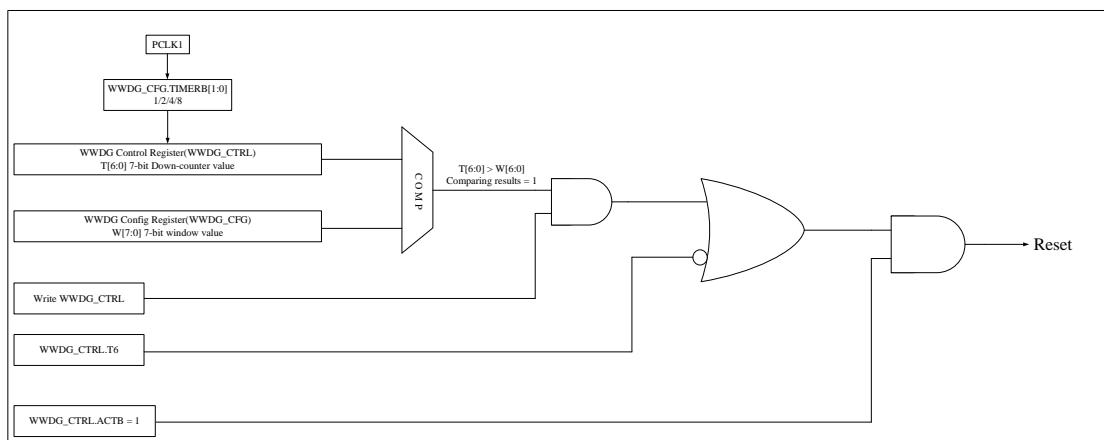
### 14.2 Main features

- 7-bit independent running down counter programmable
- After WWDG is enabled, a reset occurs under the following conditions
  - ◆ The value of the decremented counter is less than 0x40.
  - ◆ When the decremented counter value is greater than the value of the window register, it is reloaded.
- Early wake-up interrupt: If the watchdog is started and the interrupt is enabled, wake-up interrupt (WWDG\_CFG.EWINT) will be generated when the count value reaches 0x40.

### 14.3 Function description

If the watchdog is activated (the WWDG\_CTRL.ACTB bit), when the 7-bit (WWDG\_CTRL.T[6:0]) down-counter reaches 0x3F(WWDG\_CTRL.T6 bit is cleared), or the software reloads the counter when the counter value is greater than the value of the window register, a system reset will be generated. In order to avoid system reset, the software must periodically refresh the counter value in the window during normal operation.

Figure 14-1 Watchdog block diagram



Set the WWDG\_CTRL.ACTB bit to enable the watchdog, and thereafter, the WWDG will remain on until reset occurs. The 7-bit down-counter runs independently, and the counter keeps counting down whether WWDG is enabled or not. Therefore, before enabling the watchdog, you need to set WWDG\_CTRL.T [6] bit to 1, preventing reset right

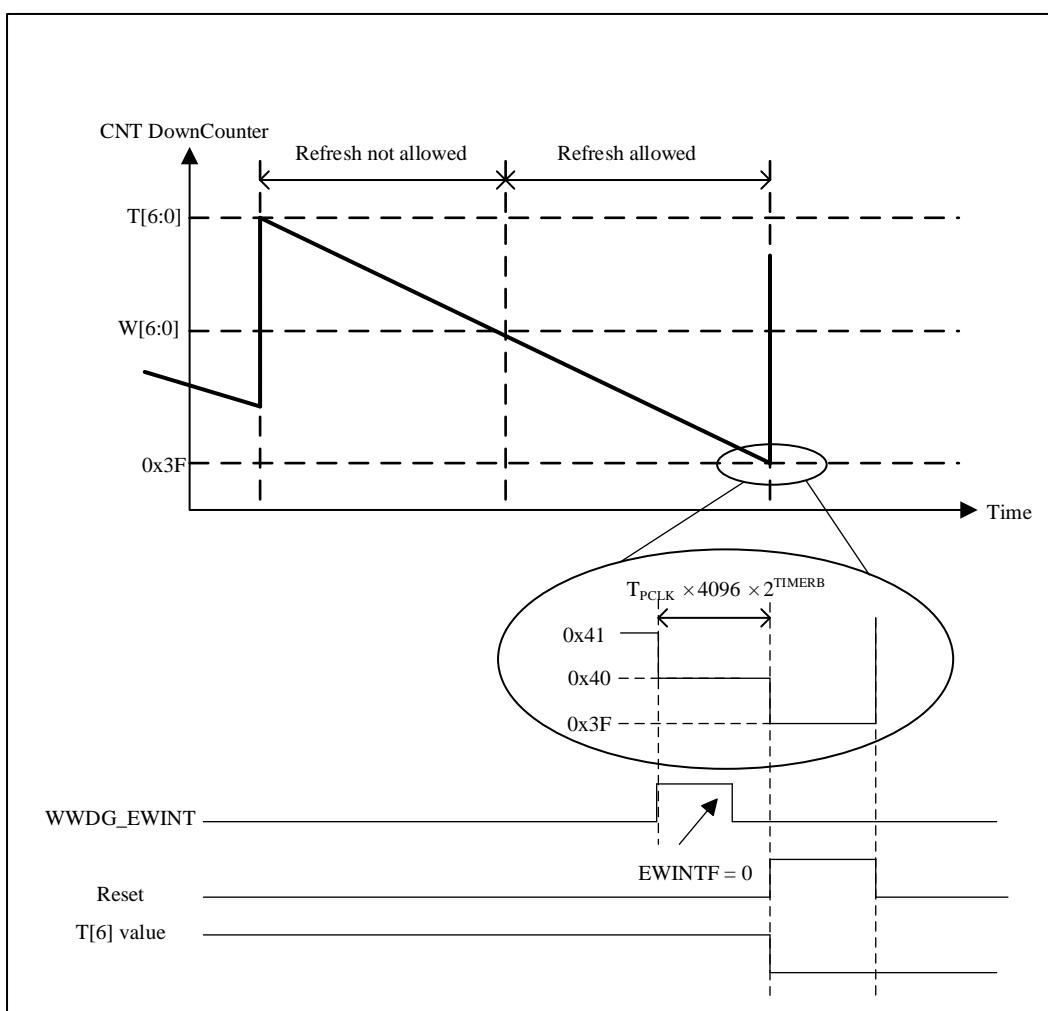
after enable. The pre-scaler value set by the clock APB1 and WWDG\_CFG.TIMERB[1:0] bits determine the decrement speed of the counter. WWDG\_CFG.W[6:0] bits set the upper limit of the window.

When the down-counter is refreshed before reaching the window register value or after WWDG\_CTRL.T6 bit becomes 0, a system reset will be generated. Figure 14-2 describes the working process of the window register.

Set the WWDG\_CFG.EWINT bit to enable early wake-up interrupt. When the count-down counter reaches 0x40, an interrupt will be generated. You can analyze the cause of software failure or save important data in the corresponding interrupt service routine (ISR), and reload the counter to prevent WWDG from resetting. Write '0' to the WWDG\_STS.EWINTF bit to clear the interrupt.

## 14.4 Timing for refresh watchdog and interrupt generation

Figure 14-2 Refresh window and interrupt timing of WWDG



Watchdog refreshing window is between WWDG\_CFG.W[6:0] value (maximum value 0x7F) and 0x3F, refresh outside this window will generates reset request to MCU. Counter count down from 0x7F to 0x3F using scaled APB1

clock, the maximum counting time and minimum counting time is shown in Table 14-1 (assuming APB1 clock 48 MHz) with calculate equation:

$$T_{WWDG} = T_{PCLK1} \times 4096 \times 2^{TIMERB} \times (T[5:0] + 1)$$

In which:

T<sub>WWDG</sub>: WWDG timeout

T<sub>PCLK1</sub>: APB1 clock interval in ms

Minimum-maximum timeout value at PCLK1 = 48MHz

Table 14-1 Maximum and minimum counting time of WWDG

TIMERB	Min timeout value(μs)	Max timeout value(ms)
	T[5:0] = 0x00	T[5:0] = 0x3F
0	85.33	5.46
1	170.67	10.92
2	341.33	21.85
3	682.67	43.68

## 14.5 Debug mode

In debug mode (Cortex-M0 core stops), WWDG counter will either continue to work normally or stops, depending on DBG\_CTRL.WWDG\_STOP bit in debug module. If this bit is set to ‘1’, the counter stops. The counter works normally when the bit is ‘0’. For details, see 3.3.2 Peripheral Debug Support.

## 14.6 User interface

### 14.6.1 WWDG configuration flow

- 1) Configure RCC\_APB1PCLKEN.WWDGEN[11] bit to enable the clock of WWDG module;
- 2) Software setting WWDG\_CFG.TIMERB[8:7] bits to configure pre-scale factor for WWDG.
- 3) Software configure WWDG\_CTRL.T[6:0] bits, setting starting value of counter. Need to set WWDG\_CTRL.T[6] bit to 1, preventing reset right after enable.
- 4) Configure WWDG\_CFG.W[6:0] bits to configure upper boundary window value;
- 5) Setting WWDG\_CTRL.ACTB[7] bit to enable WWDG;
- 6) Software operates WWDG\_STS.EWINTF[0] bit to clear wake-up interrupt flag;
- 7) Configure WWDG\_CFG.EWINT[9] bit to enable early wake-up interrupt.

## 14.7 WWDG registers

### 14.7.1 WWDG register overview

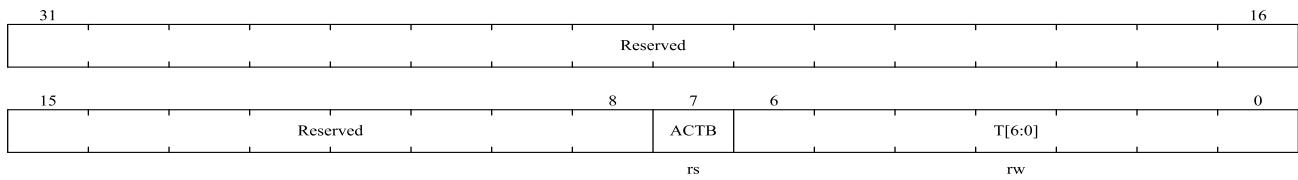
Table 14-2 WWDG register overview

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	WWDG_CTRL	Reserved																		T[6:0]													
	Reset Value																			ACTB	0	1	1	1	1	1	1	1	1				
004h	WWDG_CFG	Reserved																		EWINT	TIMERB [1:0]	W[6:0]											
	Reset Value																			0	0	0	1	1	1	1	1	1	1				
008h	WWDG_STS	Reserved																		EWINTF									0				
	Reset Value																													0			

### 14.7.2 WWDG control register (WWDG\_CTRL)

Address offset : 0x00

Reset value : 0x00000007F



Bit field	Name	Description
31:8	Reserved	Reserved, the reset value must be maintained.
7	ACTB	Activation bit When ACTB=1, the watchdog can generate a reset. This bit is set by software and only cleared by hardware after a reset. When ACTB = 1, the watchdog can generate a reset. 0: Disable watchdog 1: Enable watchdog
6:0	T[13:0]	These bits contain the value of the watchdog counter. It is decremented every (4096x2 <sup>TIMERB</sup> ) PCLK1 cycles. A reset is produced when it rolls over from 0x40 to 0x3F (T6 becomes cleared).

### 14.7.3 WWDG config register (WWDG\_CFG)

Address offset: 0x04

Reset value : 0x00000007F

31	Reserved										16
15	Reserved					10	9	8	7	6	0
						rs	rw				rw

Bit field	Name	Description
31:10	Reserved	Reserved, the reset value must be maintained.
9	EWINT	Early wake-up interrupt When set, an interrupt occurs whenever the counter reaches the value 0x40. This interrupt is only cleared by hardware after a reset.
8:7	TIMERB[1:0]	Timer base. The time base of the pre-scaler 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	W[6:0]	7-bit window value These bits contain the window value to be compared to the down counter.

#### 14.7.4 WWDG status register (WWDG\_STS)

Address offset: 0x08

Reset value : 0x0000

31	Reserved										16
15	Reserved										1 0
											EWINTF

rc\_w0

Bit field	Name	Description
31:1	Reserved	Reserved, the reset value must be maintained.
0	EWINTF	Early wake-up interrupt flag This bit is set by hardware when the counter has reached the value 0x40. It must be cleared by software by writing ‘0’. A write of ‘1’ has no effect. This bit is also set if the interrupt is not enabled.

## 15 Analog to digital conversion (ADC)

### 15.1 Introduction

The 12-bit ADC is a high-speed analog-to-digital converter using successive approximation. It has multiple channels, 16 channels, measuring 12 external and 4 internal signal sources. The A/D conversion of each channel has four execution modes: single, continuous, scan or discontinuous. ADC measurements are stored (left-aligned/ right-aligned) in 16-bit data registers. The application can detect that the input voltage is within user-defined high/low thresholds by analog watchdog and the maximum frequency of the input clock to the ADC is 18MHz. The ADC injection sampling channel supports automatic switching of the positive input of the OPA.

### 15.2 Main features

- Supports 1 ADC, supports 16 single-ended inputs, and can measure up to 12 external and 4 internal sources
- Support 12-bit resolution. The highest sampling rate is 1MSPS
- ADC clock source is divided into working clock source, sampling clock source and timing clock source
  - ◆ Only AHB\_CLK can be configured as the working clock source, Up to 48MHz.
  - ◆ PLL can be configured as a sampling clock source, up to 18MHz, support frequency division 1,2,3,4,6,8,10,12,16,32,64,128,256
  - ◆ The AHB\_CLK can be configured as the sampling clock source, up to 18MHz, and supports frequency division 1,2,3,4,6,8,10,12,16,32
  - ◆ The timing clock is used for internal timing functions and the frequency must be configured to 1MHz
- Supports timer trigger ADC sampling
- Sampling interval can be programmed individually by channel
- Support auto scan mode
- Support 2 conversion modes
  - ◆ Single conversion
  - ◆ Continuous conversion
- Support discontinuous mode
- Support DMA
- Interrupt generation
  - ◆ At the end of conversion
  - ◆ At the end of injection conversion
  - ◆ Analog watchdog event
- Data alignment with embedded data consistency

- Both regular conversions and injection conversions have external triggering options
- ADC power requirements: 2.4V to 5.5V
- ADC input voltage range:  $0 \leq V_{IN} \leq V_{DDA}$
- ADC injection channel supports software configurable OPA forward voltage input channel switching

## 15.3 Function Description

Figure 15-1 is a functional block diagram of an ADC.

Figure 15-1 Block diagram of a single ADC

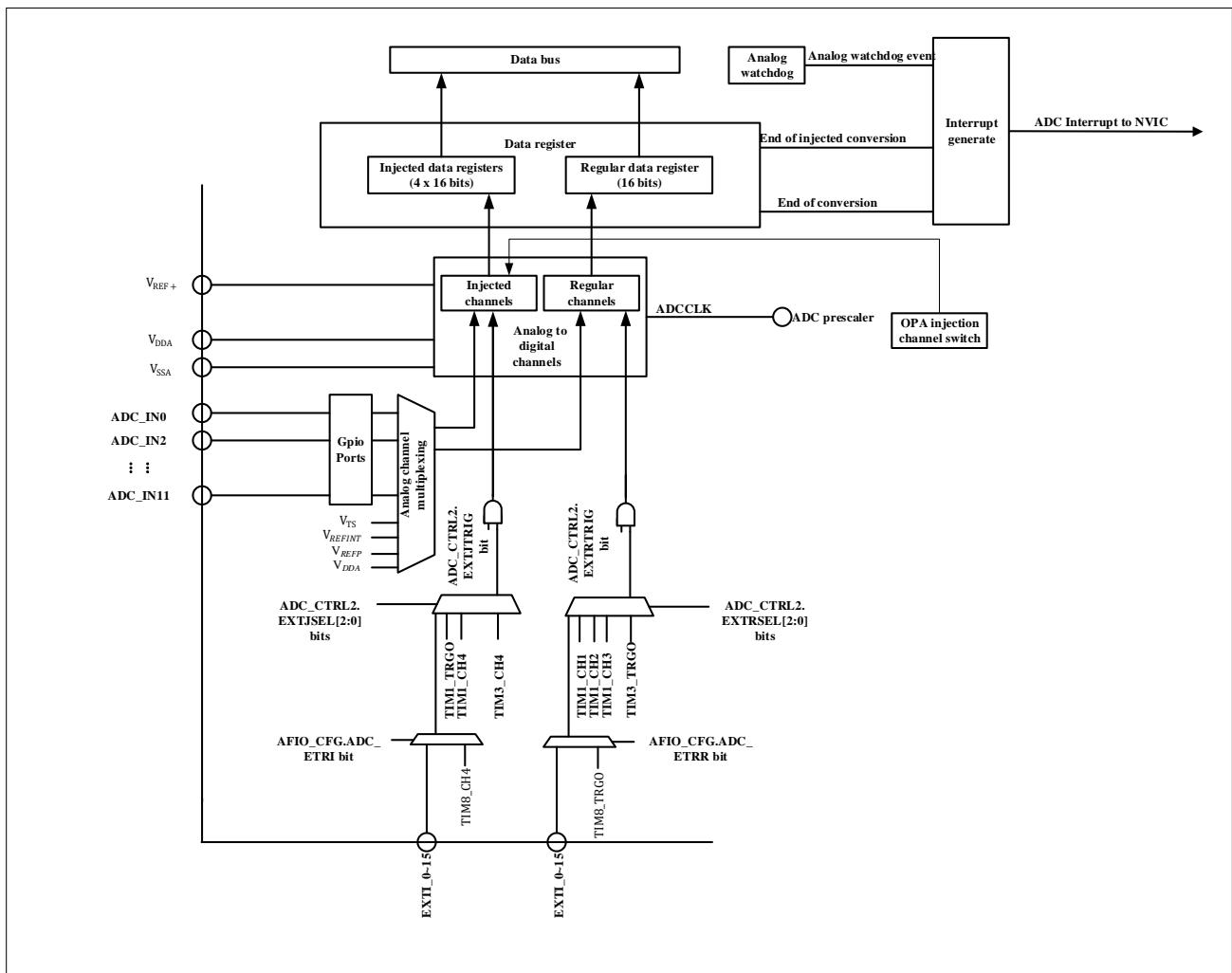


Table 15-1 ADC pins

Name	Signal types	Annotations
V <sub>DDA</sub>	Input, analog power supply	Equivalent to V <sub>DD</sub> analog power supply and: $1.8V \leq V_{DDA} \leq$

		V <sub>DD</sub> (5.5V)
V <sub>SSA</sub>	Input, analog power supply ground	Equivalent to V <sub>ss</sub> Analog power supply ground
V <sub>REF+</sub>	Input, analog reference positive	Positive reference voltage used by ADC, $2.4V \leq V_{REF+} \leq V_{DDA}$
ADCx_IN[11:0]	Analog input signal	12 analog external input channels

Note:  $V_{DDA}$  and  $V_{SSA}$ . They should be separately connected to  $V_{DD}$  and  $V_{SS}$ .

### 15.3.1 ADC clock

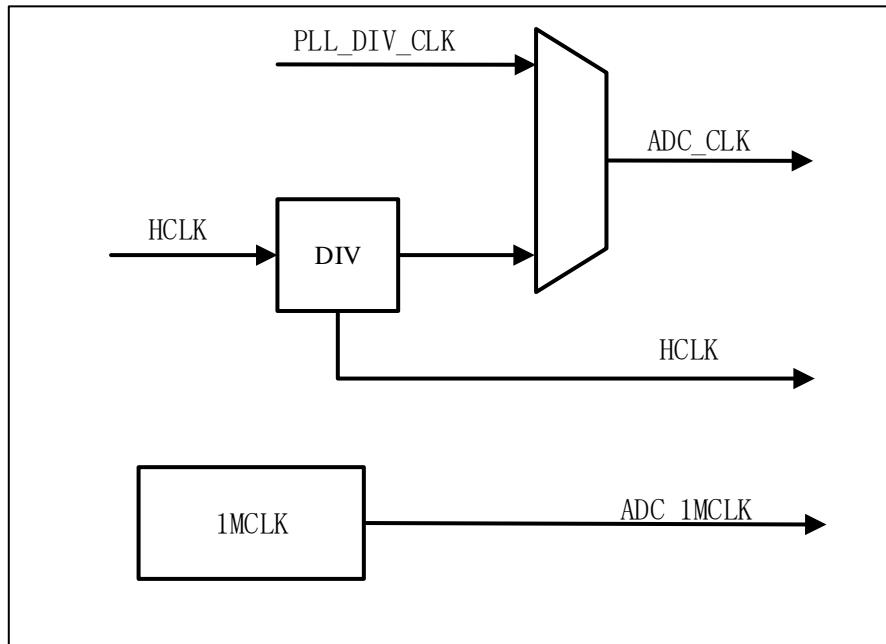
An ADC requires three clocks, HCLK, ADC\_CLK and ADC\_1MCLK.

- HCLK is used for the register access.
- ADC\_CLK is the working clock of ADC. ADC\_CLK has two sources (HCLK divider or PLL divider). HCLK divider and system are synchronous clock, while PLL divider and system are asynchronous clock. The advantage of using a synchronous clock is that there is no uncertainty when triggering the ADC to respond to the trigger. The advantage of using PLL's divider clock is that the ADC's working clock can be handled independently without affecting other modules attached to the HCLK
- ADC\_1MCLK for internal timing function, configured in RCC, frequency size must be configured to 1MHz

Note:

1. Configuration PLL as a clock source, up to 18MHz, support frequency division 1,2,3,4,6,8,10,12,16,32, 64,128,256
2. The AHB\_CLK frequency division can be configured as a working clock up to 18MHz. The AHB\_CLK frequency division can be 1,2,3,4,6,8,10,12,16,32

Figure 15-2 ADC clock



### 15.3.2 ADC switch control

You can proceed to the next step only after the power-up process is complete. You can check if the power-up is complete by polling the ADC\_CTRL3.RDY bit.

You can set the ADC\_CTRL2.ON bit to turn on the ADC. When the ADC\_CTRL2.ON bit is set for the first time, it wakes up the ADC from the power-off state. After a power-on delay of ADC ( $t_{STAB}$ ), and the conversion begins when the ADC\_CTRL2.ON bit is set again.

The conversion can be stopped by clearing the ADC\_CTRL2.ONbit and placing the ADC in power-off mode. In this mode, the ADC consumes almost no power (just a few  $\mu$ A). Power-down can be checked by polling the ADC\_CTRL3.PDRDY bit.

When the ADC is disabled, the default mode is power-down.

### 15.3.3 Channel selection

Each channel can be configured as a regular sequence and an injection sequence. A series of conversions in any order on any number of channels.

Injection sequence consists of multiple conversions, up to a maximum of 4. The ADC\_JSEQ register specifies the injection channel and the conversion order of the injection channel. The ADC\_JSEQ.JLEN[21:20] bits specified injection sequence length.

Regular sequence consists of multiple conversions, up to a maximum of 16. The ADC\_RSEQx registers specify the regular channels and the conversion order of the regular channels. The ADC\_RSEQ1.LEN[23:20] bits specified regular channel sequence length.

*Note: During conversion, changes to the ADC\_RSEQx or ADC\_JSEQ registers are prohibited; the ADC\_RSEQx or ADC\_JSEQ registers can only be changed when the ADC is idle.*

### 15.3.4 Internal channel

- The temperature sensor connects to channel ADC\_IN12
- The internal reference voltage V<sub>REFINT</sub> is connected to the channel ADC\_IN13
- Internal reference voltage V<sub>REFP</sub> is connected to channel ADC\_IN14
- VDDA pin voltage is connected to channel ADC\_IN15
- ADC\_IN6 can be connected to internal OPAMP\_OUT besides IO

Internal channels can be converted by injection or regular channels.

*Note: V<sub>REFINT</sub> needs to be enabled by configuring ADC\_CTRL3.VREFEN. After enabling, it is necessary to confirm that V<sub>REFINT</sub> is ready by ADC\_CTRL3.VREFRDY.*

### 15.3.5 Single conversion mode

The ADC can enter the single conversion mode by configuring ADC\_CTRL2.CTU to 0. In this mode, external triggering(for regular channels or injection channels) or setting ADC\_CTRL2.ON=1 (for regular channels only) can start the ADC to start conversion, and the ADC only performs one conversion.

After the conversion starts, when an injection channel conversion is completed, the injection channel conversion end flag(ADC\_STS.JENDC) will be set to 1. If the injection channel conversion end interrupt enable (ADC\_CTRL1.JENDCIEN) bit is set to 1, an interrupt will be generated at this time, and the converted data will be stored in the ADC\_JDATx register.

After the conversion starts, when a regular channel conversion is completed, the regular channel conversion end flag(ADC\_STS.ENDC) will be set to 1. If the regular channel conversion end interrupt enable (ADC\_CTRL1.ENDCIEN) bit is set to 1, an interrupt will be generated at this time, and the converted data will be stored in the ADC\_DAT register.

After single conversion, the ADC stops.

### 15.3.6 Continuous conversion mode

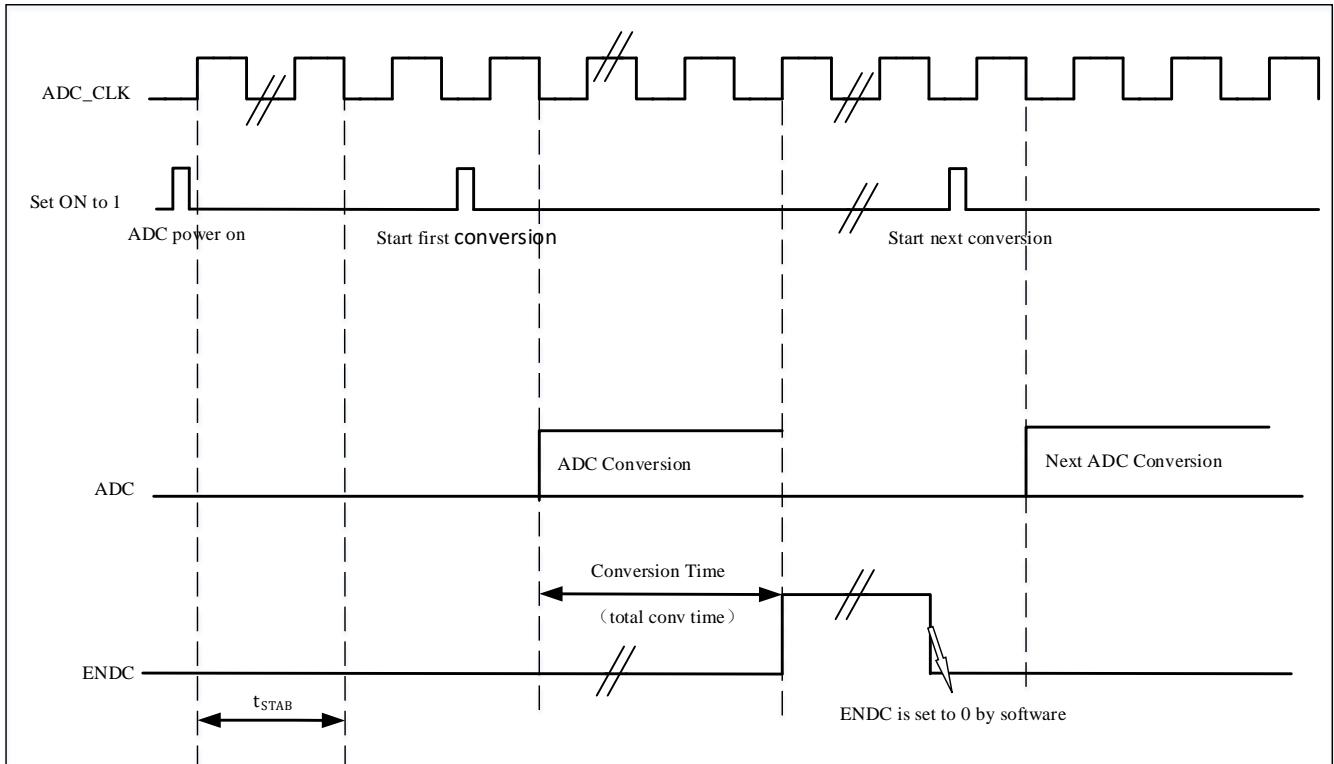
The ADC can enter the continuous conversion mode by configuring ADC\_CTRL2.CTU to 1. In this mode, external triggering or setting ADC\_CTRL2.ON to 1 can start the ADC to start conversion, and the ADC will continuously convert the selected channel. Continuous mode is only valid for regular channels, not for injection channels.

After the conversion starts, when a regular channel conversion is completed, the regular channel end of conversion flag bit (ADC\_STS.ENDC) will be set to 1. If the regular channel conversion end interrupt enable bit (ADC\_CTRL1.ENDCIEN) is set to 1 at this time, an interrupt will be generated . The converted data will be stored in the ADC\_DAT register.

### 15.3.7 Timing diagram

When ADC\_CTRL2.ON is set to 1 for the first time, the ADC is powered on. After the ADC is powered on, the ADC needs a certain time( $t_{STAB}$ ) to ensure its stability. After the ADC is stabilized, ADC\_CTRL2.ON is set to 1. At this time, set ADC\_CTRL2.ON to 1 again through software. To start the conversion and after 14 cycles, the end of conversion flag bit will be set to 1 after the conversion is completed.

Figure 15-3 Timing diagram



### 15.3.8 Analog watchdog

The analog watchdog can be enabled on the regular channel by setting ADC\_CTRL1.AWDGERCH to 1, or the analog watchdog on the injection channel can be enabled by setting ADC\_CTRL1.AWDGEJCH to 1. The high threshold of the analog watchdog can be set by configuring ADC\_WDGHIGH.HTH, and the low threshold of the analog watchdog can be set by configuring ADC\_WDGLOW.LTH. The threshold of the analog watchdog has nothing to do with the way of data alignment, because the comparison of the ADC's conversion value with the threshold is done before the alignment . When the value of ADC analog conversion is higher than the high threshold of the analog watchdog or lower than the low threshold of the analog watchdog, the analog watchdog flag (ADC\_STS.AWDG) will be set to 1, if ADC\_CTRL1.AWDGIEN has been configured to 1, an interrupt will be generated at this time. The analog watchdog can be controlled for one or more channels by configuring ADC\_CTRL1.AWSGSGLEN and ADC\_CTRL1.AWDGCH[3:0].

Table 15-2 Analog watchdog channel selection

Channel	ADC_CTRL1 register control bit		
	AWDGSGLEN	AWDGERCH	AWDGEJCH
There is none	Any value	0	0
All injection channels	0	0	1
All regular channels	0	1	0
All injection and regular channels	0	1	1
A single injection channel	1	0	1
A single regulars of the channel	1	1	0
A single injection or regular channel	1	1	1

### 15.3.9 Scan mode

By configuring ADC\_CTRL1.SCAMD to 1, the scan conversion mode can be turned on, and by configuring the four registers ADC\_RSEQ1, ADC\_RSEQ2, ADC\_RSEQ3, ADC\_JSEQ, the conversion sequence can be selected, and the ADC will scan and convert all the selected channels. After the conversion is started, the channels will be converted one by one. If ADC\_CTRL2.CTU is 1 at this time, the conversion will be restarted from the first channel of the conversion sequence after the conversion of all selected channels is completed. The DMA function can be turned on by setting ADC\_CTRL2.ENDMA to 1, and the DMA will transfer the data to the SRAM after the regular channel conversion is completed.

### 15.3.10 Injection channel management

#### 15.3.10.1 Automatic injection

If ADC\_CTRL1.AUTOJC bit is set, then the Injected channels are automatically converted following the regular channels mentioned by ADC\_RSEQx and ADC\_JSEQx. A single trigger can conver up to 16+ 4 channels. Setting ADC\_CTRL2.CTU the conversion sequence will be converted continuously.

When this function is turned on, the external trigger of the injection channel needs to be turned off.

This function cannot be used with the discontinuous mode at the same time.

When the ADC clock prescale factor is 2, there is a delay of two ADC clock intervals when the conversion sequence changes from regular to injection or injection to regular. When the ADC clock prescale factor is 4 to 8, there is a delay of one ADC clock intervals when the conversion sequence changes from regular to injection or injection to regular.

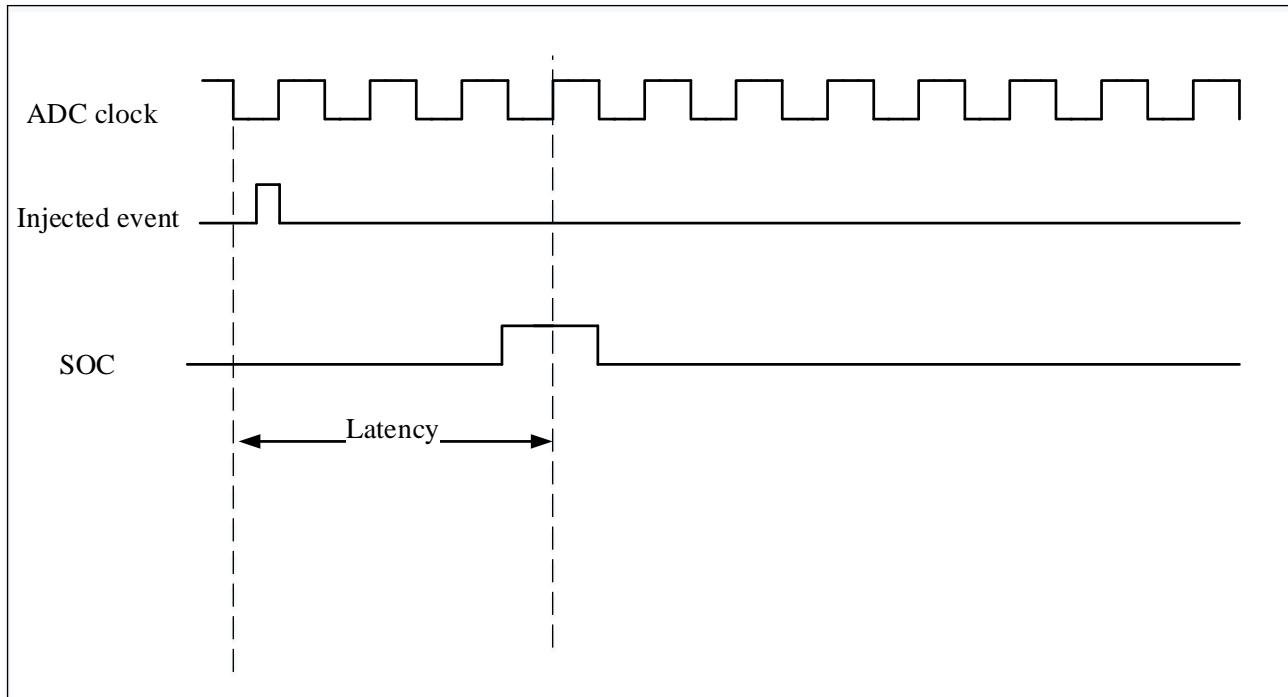
#### 15.3.10.2 Trigger injection

Set ADC\_CTRL1.AUTOJC to 0 and ADC\_CTRL1.SCAMD to 1 to enable the trigger injection function. In this function, the conversion on regular channels either by setting the ADC\_CTRL2.ON or by external trigger in continuous mode. When the regular channel is converted, if an external injection trigger is generated, the current conversion will be suspended, and the injection sequence channel will start conversion. When the injection sequence channel conversion is completed, the interrupted conversion of regular sequence channel will be resumed. If a regular event is generated during the injection conversion, the regular sequence channel will start conversion after the

injection sequence channel conversion is completed.

When using this function, the time interval between Injected channel triggers are fired needs to be greater than the time it takes for the injection sequence to complete the transition.

Figure 15-4 Injection conversion delay



*Note:For the maximum delay value, please refer to the electrical characteristics section in the data manual.*

### 15.3.11 Discontinuous mode

#### 15.3.11.1 Regular channels

Configure ADC\_CTRL1.DREGCH to 1 to enable the discontinuous mode on the regular channel, obtain the regular sequence by configuring ADC\_RSEQ1, ADC\_RSEQ2, ADC\_RSEQ3, and configure ADC\_CTRL1.DCTU[2:0] to control the conversion of n channels each time a trigger signal is generated. The total sequence length is defined by ADC\_RSEQ1.LEN[3:0].

When the trigger signal is generated, it will convert n channels of the regular sequence and then stop, until the next trigger signal is generated, it will continue to convert n channels from the point where the previous conversion stopped, until all channels of the regular sequence are converted (If the last trigger occurs and the remaining channels in the conversion sequence are less than n, only the remaining channels will be converted and the conversion will be stopped), and the end of conversion flag bit will also be set to 1. When the conversion of all channels in the conversion sequence is completed, when the next trigger signal occurs ,the conversion starts from the first channel of the regular sequence again.

#### 15.3.11.2 Injection channels

Configure ADC\_CTRL1.DJCH to 1 to enable the discontinuous mode on the injection channel, obtain the injection

sequence by configuring ADC\_JSEQ.

When the trigger signal is generated, it will convert 1 channel of the injection sequence and then stop. Until the next trigger signal is generated, it will continue to convert 1 channel from the point where the previous conversion stopped until all channels of the injection sequence are converted, and the end of conversion flag bit will also be set to 1. When the conversion of all channels in the conversion sequence is completed, when the next trigger signal occurs, the conversion starts from the first channel of the injection sequence again.

Only one of injection conversion and regular conversion can be set to discontinuous mode at the same time, and the automatic injection function and discontinuous mode cannot be set at the same time.

## 15.4 Data aligned

There are two alignment methods for data storage after conversion: left-aligned and right-aligned. The alignment can be set by the ADC\_CTRL2.ALIG bit. ADC\_CTRL2.ALIG = 0 is right-aligned, as shown in **Table 15-3**, ADC\_CTRL2.ALIG = 1 is left-aligned, as shown in **Table 15-4**.

For injection sequence, the SYM bit is the extended sign value, and the data stored in the register is the conversion result minus the user-defined offset in the ADC\_JOFFSETx register, so the result can be a negative value; for regular sequence, there is no need to subtract offset value.

Table 15-3 Right-align data

The Injection sequence

SYM	SYM	SYM	SYM	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
-----	-----	-----	-----	-----	-----	----	----	----	----	----	----	----	----	----	----

The regular sequence

0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
---	---	---	---	-----	-----	----	----	----	----	----	----	----	----	----	----

Table 15-4 Left-align data

Injection sequence

SYM	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0
-----	-----	-----	----	----	----	----	----	----	----	----	----	----	---	---	---

The regular sequence

D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0
-----	-----	----	----	----	----	----	----	----	----	----	----	---	---	---	---

## 15.5 Programmable channel sampling time

Specify the number of sampling cycles of ADC in ADC\_SAMPT1.SAMPx[2:0] and ADC\_SAMPT2.SAMPx[2:0], and then the ADC samples the input voltage in the specified sampling cycle. For different channels, you can select different sampling time. The total conversion time is calculated as follows:

$$T_{CONV} = \text{Sampling time} + 12 \text{ cycles}$$

Example:

ADCCLK=16MHz, the sampling time is 6 cycles and resolution is 12bit, the total conversion time is "6 + 12" ADCCLK Cycles, that is:

$$T_{CONV} = 6 + 12 = 18 \text{ cycle} = 1.125\mu\text{s}$$

## 15.6 Externally triggered conversion

For the regular sequence , software sets the ADC\_CTRL2.EXTRTRIG bit to 1, then the regular channel can use the rising edge of the external event to trigger the start conversion, and then the software sets the ADC\_CTRL2.EXTRSEL[2:0] bits to select the external trigger source of the regular sequence. The external trigger source selection is shown in the table below. If you select EXTI line 0~15 or TIM8\_TRGO as the external trigger source, you can set the AFIO\_CFG.ADC\_ETRR and AFIO\_CFG EXTI\_ETRR[3:0] bits to implement; if you select SWSTRRCH as the external trigger source, you can start the regular channel conversion by setting ADC\_CTRL2.SWSTRRCH to 1.

Table 15-5 ADC is used for external triggering of regular channels

EXTSEL[2:0]	Trigger source	Type
000	TIM1_CC1 event	Internal signal from the on-chip timer
001	TIM1_CC2 event	
010	TIM1_CC3 event	
011	N/A	
100	TIM3_TRGO event	
101	N/A	
110	EXTI line 0~15/TIM8_TRGO event	External pin/internal signal from on-chip timer
111	SWSTRRCH	Software control bit

For the injection sequence , the software sets the ADC\_CTRL2.EXTJTRIG bit to 1, then the injection channel can use the rising edge of the external event to trigger the start conversion, and the software sets the ADC\_CTRL2.EXTJSEL[2:0] bits to select the external trigger source of the injection sequence. The external trigger source selection is shown in the table below. If you select EXTI line 0~15 or TIM8\_CC4 as the external trigger source, you can set the AFIO\_CFG.ADC\_ETRI and AFIO\_CFG EXTI\_ETRI[3:0] bits to implement; if you select SWSTRJCH as the external trigger source, you can start the injection channel conversion by setting ADC\_CTRL2.SWSTRJCH to 1.

Table 15-6 ADC is used for external triggering of injection channels

EXTJSEL[2:0]	Trigger source	Type
000	TIM1_TRGO event	Internal signal from the on-chip timer
001	TIM1_CC4 event	
010	N/A	
011	N/A	
100	TIM3_CC4 event	
101	N/A	

EXTJSEL[2:0]	Trigger source	Type
110	EXTI line 0~15/TIM8_CC4 event	External pin/internal signal from on-chip timer
111	SWSTRJCH	Software control bit

*Note: Injection triggers can interrupt conversion of the regular sequence.*

## 15.7 DMA requests

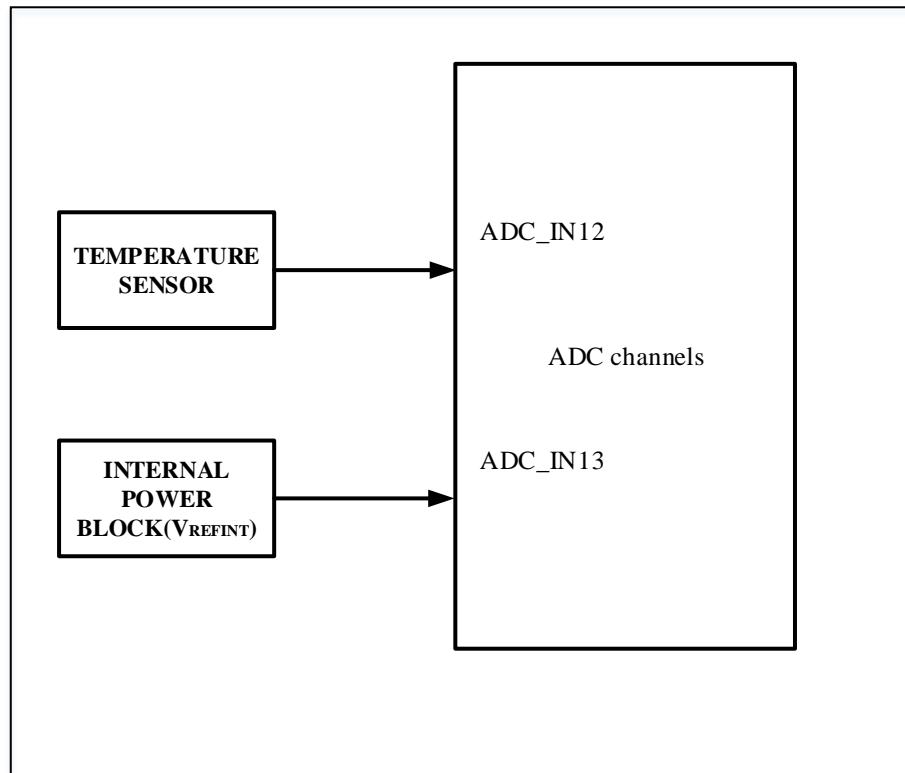
In order to avoid excessive data load when converting multiple regular channels, resulting in errors in the regular channel conversion result stored in the ADC\_DAT register, you can set the ADC\_CTRL2.ENDMA bit to 1 to use the DMA mode. When the ADC regular channel conversion ends, it will be A DMA request is generated. After receiving the request, the DMA will transfer the converted data from the ADC\_DAT register to the destination address specified by the user.

## 15.8 Temperature sensor

Set the ADC\_CTRL2.TEMPEN bit to 1, enable the temperature sensor, and use the temperature sensor to detect the ambient temperature when the device is working. The output voltage sampled by the temperature sensor is converted into a digital value by the ADC\_IN12 channel. When the temperature sensor is working, the recommended sampling time is more than 10us; when the temperature sensor is not working, The ADC\_CTRL2.TEMPEN bit can be cleared by software to turn off the temperature sensor to reduce power consumption. **Figure 15-5** is a block diagram of a temperature sensor.

The output voltage of the temperature sensor changes linearly with temperature. Different chips will have different offsets in the temperature curve due to different production processes. Through testing, it is found that the maximum offset is 3°C. This characteristic makes the internal temperature sensor more suitable for detecting temperature changes. Not suitable for measuring absolute temperature. When accurate temperature measurement is required, an external temperature sensor should be used.

Figure 15-5 Temperature sensor and VREFINT Diagram of the channel



### 15.8.1 Temperature sensor using flow

- 1) Configure the channel (ADC\_IN12) and sampling time greater than 10us
- 2) Set ADC\_CTRL2.TEMPEN bit to 1 to enable temperature sensor and set ADC\_CTRL3.VREFEN bit to enable VREFINT
- 3) Set ADC\_CTRL2.ON bit to 1 to start ADC conversion (or through external trigger)
- 4) Read the temperature data in the ADC data register, and calculate the temperature value by the following formula:

$$\text{Temperature } (^{\circ}\text{C}) = \{(V_{25} - V_{\text{SENSE}}) / \text{Avg\_Slope}\} + 25$$

In which:

$V_{25}$  =  $V_{\text{SENSE}}$  at 25 degrees Celsius

Avg\_Slope = temperature and  $V_{\text{SENSE}}$  Average slope of a curve (mV/°C or  $\mu$ V/°C)

Refer to the values of  $V_{25}$  and Avg\_Slope in the electrical characteristics chapter of the datasheet.

*Note: There is a settling time before the sensor wakes up from the power-off mode to the correct output of  $V_{\text{SENSE}}$ ; there is also a settling time after the ADC is powered on, so in order to shorten the delay, the ADC\_CTRL2.TEMPEN and ADC\_CTRL2.ON bits should be set at the same time.*

## 15.9 ADC interrupt

ADC interrupts can be from an end of regular or injection sequence conversion, an analog watchdog event when input voltage exceeds the threshold, any end of regular or injection channel conversion. These interrupts have independent interrupt enable bits.

There are 2 status flags in the ADC\_STS register: injection sequence channel conversion started (JSTR) and regular sequence channel conversion started (STR). But there are no interrupts associated with these two flags in the ADC.

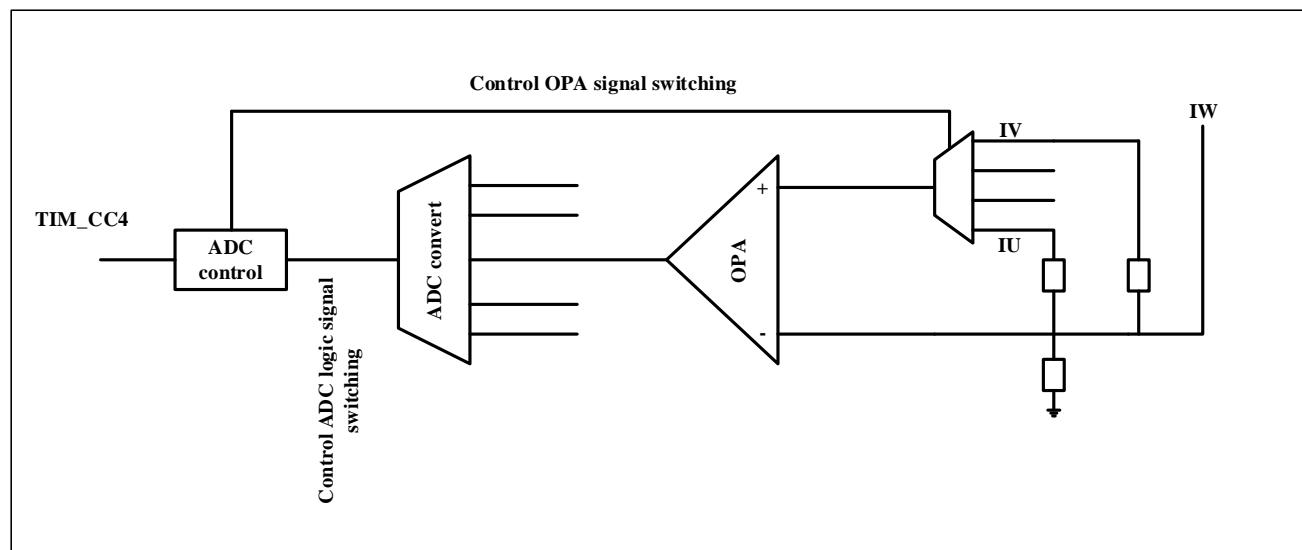
Table 15-7 ADC interrupt

Interrupt event	Event flags	Enable control bit
Regular sequence or injection conversion is complete	ENDC	ENDCIEN
Injection sequence conversion is complete	JENDC	JENDCIEN
Analog watchdog status bit is set	AWDG	AWDGIEN
Any regular channel interruption is enabled	ENDCA	ENDCAIEN
Any injection channel interruption is enabled	JENDCA	JENDCAIEN

## 15.10 OPA channel control

Figure 15-6 is a schematic block diagram of ADC switching OPA channel. Before sampling the injection channel, ADC\_OPACTRL register will control the OPA positive channel selection signal, wait for the OPA setup time, and then start sampling. When the sampling is completed, the ADC\_OPACTRL register release the channel control of the OPA. Figure 15-6 shows TIM1\_CC4 as the trigger, but the actual trigger source is not limited to TIM1\_CC4, all trigger sources can support.

Figure 15-6 TIM1 CC4 triggers OPA channel switching ADC injection sampling



The software sets the ADC\_OPACTRL.JSQx\_OPAEN register, and cooperates with ADC\_JSEQ.JLEN [1:0] to select

which sample of the 4 samples enables the control of the OPA channel. Software selects the OPA channel corresponding to each sample by setting the ADC\_OPACTRL.JSQ1\_OPASEL, ADC\_OPACTRL.JSQ2\_OPASEL, ADC\_OPACTRL.JSQ3\_OPASEL and ADC\_OPACTRL.JSQ4\_OPASEL. As shown in Table 15-6, if ADC\_JSEQ.JLEN [1:0] = 0, only the channel corresponding to JSQ4 is sampled, ADC\_OPACTRL.JSQ4\_OPAEN = 1 enables the control of the OPA input, and switches OPAMP\_CS.VPSEL = JSQ4\_OPASEL. If ADC\_JSEQ.JLEN [1:0] = 1, the first to convert the JSQ3 channel, ADC\_OPACTRL.JSQ3\_OPAEN = 1 will toggle and OPAMP\_CS.VPSEL = ADC\_OPACTRL.JSQ3\_OPASEL, the second converts the JSQ4 channel, ADC\_OPACTRL.JSQ4\_OPAEN = 1 will toggle OPAMP\_CS.VPSEL = JSQ4\_OPASEL. ADC\_OPACTRL.JSQ3\_OPASEL.

Table 15-8 OPA channel selection

LEN	1st conv	2nd conv	3rd conv	4th conv
0	JSQ4	-	-	-
1	JSQ3	JSQ4	-	-
2	JSQ2	JSQ3	JSQ4	-
3	JSQ1	JSQ2	JSQ3	JSQ4

Since the OPA channel needs a certain setup time after switching, the software can configure the ADC\_OPACTRL.OPA\_SETUP\_TIME. After switching the OPA channel, the ADC will wait for the corresponding setup time before starting sampling. The setup time is calculated as follows:

$$T = \text{ADC\_OPACTRL.OPA\_SETUP\_TIME} / \text{adc clock frequency}$$

If ADC clock frequency=16MHz, setup time is 5us, then OPA\_SETUP\_TIME=80 should be set.

## 15.11 ADC registers

### 15.11.1 ADC register overview

Table 15-9 ADC register overview

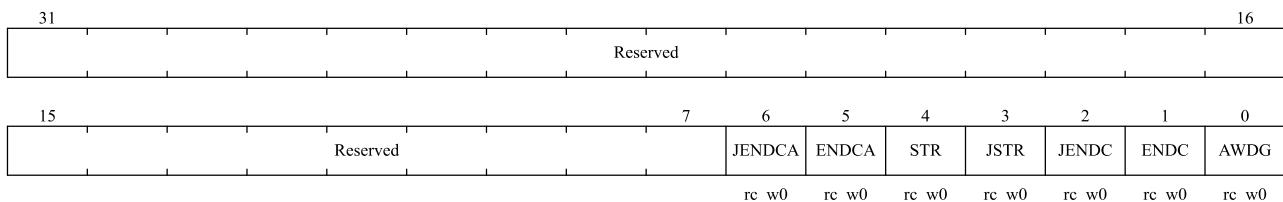
Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0										
000h	ADC_STS																																										
	Reset Value																																										
004h	ADC_CTRL1																																										
	Reset Value																																										
008h	ADC_CTRL2																																										
	Reset Value																																										
00Ch	ADC_SAMPT1																																										
	Reset Value																																										
010h	ADC_SAMPT2		SAMP15[3:0]				SAMP14[3:0]			SAMP13[3:0]			SAMP12[3:0]			SAMP11[3:0]			SAMP10[3:0]			SAMP9[3:0]			SAMP8[3:0]																		
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
014h	ADC_SAMPT3		SAMP7[3:0]				SAMP6[3:0]			SAMP5[3:0]			SAMP4[3:0]			SAMP3[3:0]			SAMP2[3:0]			SAMP1[3:0]			SAMP0[3:0]																		
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
018h	ADC_JOFFSET1																																										
	Reset Value																																										
01Ch	ADC_JOFFSET2																																										
	Reset Value																																										
020h	ADC_JOFFSET3																																										
	Reset Value																																										
024h	ADC_JOFFSET4																																										
	Reset Value																																										
028h	ADC_WDGHIGH																																										
	Reset Value																																										
02Ch	ADC_WDGLOW																																										
	Reset Value																																										
030h	ADC_RSEQ1																																										
	Reset Value																																										
034h	ADC_RSEQ2																																										
	Reset Value																																										
038h	ADC_RSEQ3																																										
	Reset Value																																										
03Ch	ADC_JSEQ																																										
	Reset Value																																										
040h	ADC_JDAT1																																										
	Reset Value																																										

044h	ADC_JDAT2	Reserved	JDAT2[15:0]															
	Reset Value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
048h	ADC_JDAT3	Reserved	JDAT3[15:0]															
	Reset Value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
04Ch	ADC_JDAT4	Reserved	JDAT4[15:0]															
	Reset Value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
050h	ADC_DAT	Reserved	DAT[15:0]															
	Reset Value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
054h	ADC_CTRL3	Reserved	JENDCAEN ENDCAEN															
	Reset Value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
058h	ADC_TEST	Reserved	PDRDY RDY CKM0D VREFEN REFSEL															
	Reset Value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
05Ch	ADC_OPACTRL	Reserved	OPA_SETUP_TIME[9:0]								ISQ1_OPASEL[2:0]							
	Reset Value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

### 15.11.2 ADC status register (ADC\_STS)

Address offset: 0x00

Reset value: 0x0000 0000



Bit field	Name	Description
31:7	Reserved	Reserved, the reset value must be maintained
6	JENDCA	Any injected channel end of conversion flag This bit is set by hardware at the end of any injection channel conversion and cleared by software. 0: Conversion is not complete; 1: Conversion is complete.
5	ENDCA	Any regular channel end of conversion flag This bit is set by hardware at the end of any channel (regular or injection) conversion and cleared by software. 0: Conversion is not complete; 1: Conversion is complete.
4	STR	Regular channel start flag This bit is set by hardware at the start of regular channel conversion and cleared by software. 0: Regular channel conversion has not started.

Bit field	Name	Description
		1: Regular channel conversion has started.
3	JSTR	Injected channel start flag This bit is set by hardware at the start of the injection channel conversion and cleared by software. 0: Injection sequence channel conversion has not started. 1: Injection sequence channel conversion has started.
2	JENDC	Injected channel end of conversion This bit is set by hardware at the end of all injection sequence channel conversions and cleared by software 0: Conversion is not complete. 1: Conversion is complete.
1	ENDC	Conversion sequence channel end of conversion This bit is set by hardware at the end of all regular( or injection) sequence channel conversion and cleared by software 0: Conversion is not complete. 1: Conversion is complete.
0	AWDG	Analog watchdog flag This bit is set by hardware and cleared by software when converted voltage values are outside the range defined by the ADC_LTR and ADC_HTR registers 0: Analog watchdog event not occurs; 1: Analog watchdog event occurs.

### 15.11.3 ADC control register 1 (ADC\_CTRL1)

Address offset: 0x04

Reset value: 0x0000 0000

31	Reserved								24	23	22	21	Reserved				16
15	DCTU[2:0]	DJCH	DREGCH	AUTOJC	AWDG SGLEN	SCANMD	JENDC IEN	AWD GIEN	ENDCIEN	Reserved				AWDGCH[3:0]			0

rw                    rw

Bit field	Name	Description
31:24	Reserved	Reserved,the reset value must be maintained
23	AWDGERCH	Analog watchdog enable on regular channels This bit is set and cleared by the software. 0: Disables analog watchdog on regular channel. 1: Use analog watchdog on regular channels.

Bit field	Name	Description
22	AWDGEJCH	Analog watchdog enable on injected channels This bit is set and cleared by the software. 0: Disables analog watchdog on injection channel. 1: Use analog watchdog on the injection channel.
21:16	Reserved	Reserved, the reset value must be maintained
15:13	DCTU[2:0]	Discontinuous mode channel count The software uses these bits to define the number of channels for converting regulars after receiving an external trigger in intermittent mode 000: 1 channel 001: 2 channels ... 111: 8 channels
12	DJCH	Discontinuous mode on injected channels This bit is set and cleared by the software. It is used to turn on or off discontinuous mode on injected channels. 0: Disable discontinuous mode on injection sequence channel 1: Enable discontinuous mode on injection sequence channel
11	DREGCH	Discontinuous mode is on regular channels. This bit is set and cleared by the software. It is used to turn on or off discontinuous mode on regular channels. 0: Disable discontinuous mode on regular sequence channel 1: Enable discontinuous mode on regular sequence channel
10	AUTOJC	Automatic injected sequence conversion This bit is set and cleared by the software to enable or disable automatic injection sequence channel conversion after regular sequence channel conversion is complete 0: Disable automatic injection channel conversion. 1: Enable automatic injection channel conversion.
9	AWDGSGLEN	Enable the watchdog on a single channel in scan mode This bit is set and cleared by software to enable or disable analog watchdog functions on channels specified by ADC_CTRL1.AWDGCH[3:0] 0: Use watchdog on all channels. 1: Use watchdog on single channel.
8	SCANMD	Scan mode This bit is set and cleared by the software to enable or disable scan mode. In scan mode, the conversion is made by ADC_RSEQx or the selected channel of the ADC_JSEQ register. 0: Disable scan mode. 1: Enable scan mode. <i>Note: If the ADC_CTRL1.ENDCIEN or ADC_CTRL1.JENDCIEN bits are set separately, ADC_STS.ENDC or ADC_STS.JENDC interrupts occur only after the last channel has been converted.</i>

Bit field	Name	Description
7	JENDCIEN	<p>Interrupt enable for injected channels</p> <p>This bit is set and cleared by the software to disallow or allow interrupts after all injection channel conversions have finished.</p> <p>0: Disable JENDC interruption.</p> <p>1: Enable JENDC interruption. An interrupt occurred when hardware set ADC_STS.JENDC bit.</p>
6	AWDGIEN	<p>Analog watchdog interrupt enable</p> <p>This bit is set and cleared by software to disallow or allow interrupt generated by analog watchdog. In scan mode, if the watchdog detects an out-of-range value, the scan is aborted only when that bit is set.</p> <p>0: Disable analog watchdog interruption.</p> <p>1: Enable analog watchdog interruption.</p>
5	ENDCIEN	<p>Interrupt enable for any channels</p> <p>This bit is set and cleared by the software to disallow or allow interrupts to occur after the regular channel conversion ends.</p> <p>0: Disable ENDC interruption.</p> <p>1: Enable ENDC interruption.</p>
4	Reserved	Reserved, the reset value must be maintained
3:0	AWDGCH[3:0]	<p>Analog watchdog channel select bits</p> <p>These bits are set and cleared by software to select input channels that analog watchdog protection.</p> <p>0000: ADC analog input channel 0</p> <p>0001: ADC analog input channel 1</p> <p>...</p> <p>1110: ADC analog input channel 14</p> <p>1111: ADC analog input channel 15</p> <p>Reserved all other values.</p>

#### 15.11.4 ADC control register 2 (ADC\_CTRL2)

Address offset: 0x08

Reset value: 0x0000 0000

31	Reserved							24	23	22	21	20	19	17	16
15	EXTJTRIG	EXTJSEL[2:0]	14	ALIG	12	11	10	9	8	rw 7	rw 7	rw 7	rw 7	rw 7	rw 7
14	rw	rw	13	rw	12	rw	11	rw	10	rw	rw	rw	rw	rw	rw
13	rw	rw	12	rw	11	rw	10	rw	9	rw	rw	rw	rw	rw	rw

Bit field	Name	Description
31:24	Reserved	Reserved, the reset value must be maintained

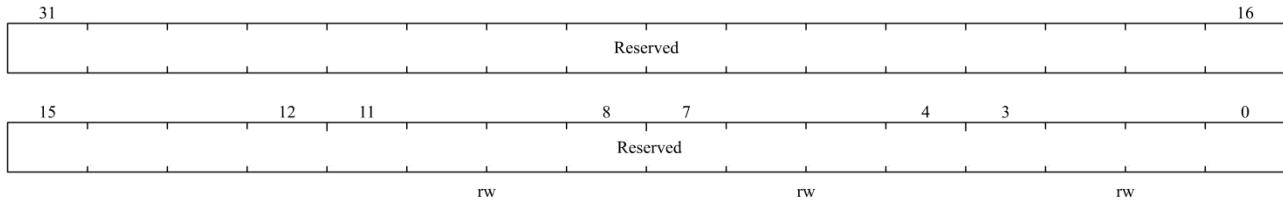
Bit field	Name	Description								
23	TEMPEN	<p>Temperature sensor Enable</p> <p>This bit is set and cleared by the software to enable or disable the temperature sensor and VREFINT Channel.</p> <p>0: Disables the temperature sensor.</p> <p>1: Enable the temperature sensor.</p>								
22	SWSTRRCH	<p>Start conversion of regular channels</p> <p>This bit is set by the software to start the conversion and cleared by the hardware as soon as the conversion begins. If SWSTRRCH is selected as the trigger event in the ADC_CTRL2.EXTRSEL[2:0] bit, which is used to initiate the conversion of a set of regular channels</p> <p>0: Reset state.</p> <p>1: Starts converting the regular channel.</p>								
21	SWSTRJCH	<p>Start conversion of injected channels</p> <p>This bit is set by the software to initiate the conversion and can be cleared by the software or by the hardware as soon as the conversion begins. If SWSTRJCH is selected as the trigger event in the ADC_CTRL2.EXTJSEL[2:0] bit, which is used to initiate a conversion of a set of injected channels</p> <p>0: Reset state.</p> <p>1: Starts converting the injection channel.</p>								
20	EXTRTRIG	<p>External trigger conversion mode for regular channels</p> <p>This bit is set and cleared by software to enable or disable external triggering events that can start regular sequence conversion.</p> <p>0: Start conversion without external events.</p> <p>1: Use an external event to start the conversion.</p>								
19:17	EXTRSEL[2:0]	<p>External event select for regular sequence</p> <p>These bits select external events to start the regular sequence conversion</p> <p>The triggering configuration of ADC is as follows</p> <table> <tbody> <tr> <td>000: indicates the CC1 event of timer 1</td> <td>100: indicates the TRGO event of timer 3</td> </tr> <tr> <td>001: indicates the CC2 event of timer 1</td> <td>101: Reserved</td> </tr> <tr> <td>010: indicates the CC3 event of timer 1</td> <td>110: EXTI line 0~15/TIM8_TRGO event</td> </tr> <tr> <td>011: Reserved</td> <td>111: SWSTRRCH</td> </tr> </tbody> </table>	000: indicates the CC1 event of timer 1	100: indicates the TRGO event of timer 3	001: indicates the CC2 event of timer 1	101: Reserved	010: indicates the CC3 event of timer 1	110: EXTI line 0~15/TIM8_TRGO event	011: Reserved	111: SWSTRRCH
000: indicates the CC1 event of timer 1	100: indicates the TRGO event of timer 3									
001: indicates the CC2 event of timer 1	101: Reserved									
010: indicates the CC3 event of timer 1	110: EXTI line 0~15/TIM8_TRGO event									
011: Reserved	111: SWSTRRCH									
16	Reserved	Reserved, the reset value must be maintained								
15	EXTJTRIG	<p>External trigger conversion mode for injected channels</p> <p>This bit is set and cleared by software to enable or disable external triggering events that can start injection sequence conversion.</p> <p>0: Start conversion without external events.</p> <p>1: Use an external event to start the conversion.</p>								

Bit field	Name	Description								
14:12	EXTJSEL[2:0]	<p>External event select for injected sequence These bits select the External event used to trigger the injected sequence conversion. The triggering configuration of ADC is as follows</p> <table> <tr> <td>000: indicates the TRGO event of timer 1</td><td>100: indicates the CC4 event of timer 3</td></tr> <tr> <td>001: indicates the CC4 event of timer 1</td><td>101: Reserved</td></tr> <tr> <td>010: Reserved</td><td>110: EXTI line 0~15/TIM8_CC4 event</td></tr> <tr> <td>011: Reserved</td><td>111: SWSTRJCH</td></tr> </table>	000: indicates the TRGO event of timer 1	100: indicates the CC4 event of timer 3	001: indicates the CC4 event of timer 1	101: Reserved	010: Reserved	110: EXTI line 0~15/TIM8_CC4 event	011: Reserved	111: SWSTRJCH
000: indicates the TRGO event of timer 1	100: indicates the CC4 event of timer 3									
001: indicates the CC4 event of timer 1	101: Reserved									
010: Reserved	110: EXTI line 0~15/TIM8_CC4 event									
011: Reserved	111: SWSTRJCH									
11	ALIG	<p>Data alignment This bit is set and cleared by the software. Refer to Table 15-3 and Table 15-4.</p> <p>0: Right-aligned. 1: Left-aligned.</p>								
10:9	Reserved	Reserved, the reset value must be maintained								
8	ENDMA	<p>Direct memory access mode This bit is set and cleared by the software. See the DMA Controller chapter for details.</p> <p>0: Do not use DMA mode. 1: Use DMA mode.</p>								
7:2	Reserved	Reserved, the reset value must be maintained								
1	CTU	<p>Continuous conversion This bit is set and cleared by the software. If this bit is set, the conversion continues until the bit is cleared.</p> <p>0: Single conversion mode. 1: Continuous conversion mode.</p>								
0	ON	<p>A/D converter ON/OFF This bit is set and cleared by the software. When the bit is '0', writing '1' will wake the ADC from power-off mode.</p> <p>When the bit is '1', writing '1' starts the conversion. The application should note that there is a delay tSTAB between the time the converter is powered on and the time the conversion begins, see Figure 15-3.</p> <p>0: Close ADC conversion/calibration and enter power-down mode. 1: Start ADC and start conversion.</p> <p>Note: If there are other bits changed in this register along with ON, the conversion will not be triggered. This is to prevent the wrong conversion from being triggered.</p>								

### 15.11.5 ADC sampling time register 1 (ADC\_SAMPT1)

Address offset: 0x0C

Reset value: 0x0000 0000

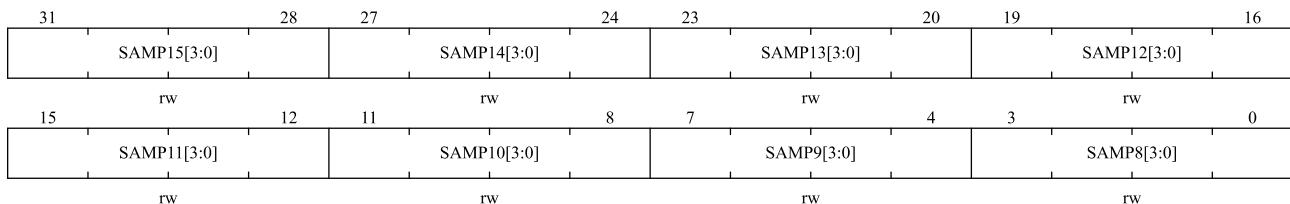


Bit field	Name	Description
31:0	Reserved	Reserved, the reset value must be maintained

### 15.11.6 ADC sampling time register 2 (ADC\_SAMPT2)

Address offset: 0x10

Reset value: 0x0000 0000

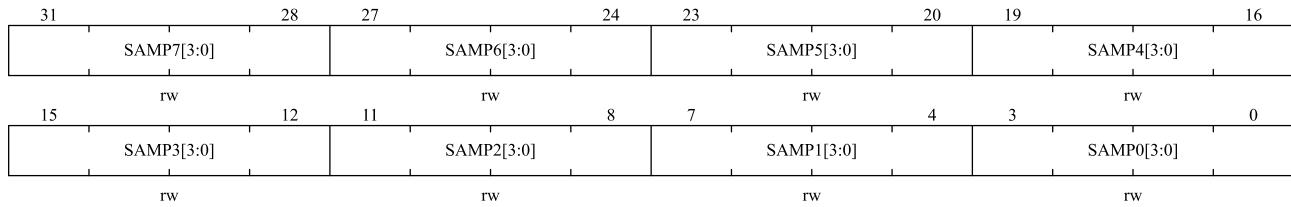


Bit field	Name	Description																
31:0	SAMPx[3:0] x=8~15	<p>Channel x sample time selection</p> <p>These bits are used to independently select the sampling time for each channel. The channel selection bit must remain constant during the sampling period.</p> <table> <tbody> <tr> <td>0000: 6 cycles</td> <td>1000: 88 cycles</td> </tr> <tr> <td>0001: 8 cycles</td> <td>1001: 120 cycles</td> </tr> <tr> <td>0010: 14 cycles</td> <td>1010: 182 cycles</td> </tr> <tr> <td>0011: 20 cycles</td> <td>1011: 240 cycles</td> </tr> <tr> <td>0100: 29 cycles</td> <td>1100: 300 cycles</td> </tr> <tr> <td>0101: 42 cycles</td> <td>1101: 400 cycles</td> </tr> <tr> <td>0110: 56 cycles</td> <td>1110: 480 cycles</td> </tr> <tr> <td>0111: 72 cycles</td> <td>1111: 600 cycles</td> </tr> </tbody> </table> <p><i>NOTE: ADC analog channel 13 and channel 14 are internally connected to the temperature sensor and VREFINT, respectively.</i></p>	0000: 6 cycles	1000: 88 cycles	0001: 8 cycles	1001: 120 cycles	0010: 14 cycles	1010: 182 cycles	0011: 20 cycles	1011: 240 cycles	0100: 29 cycles	1100: 300 cycles	0101: 42 cycles	1101: 400 cycles	0110: 56 cycles	1110: 480 cycles	0111: 72 cycles	1111: 600 cycles
0000: 6 cycles	1000: 88 cycles																	
0001: 8 cycles	1001: 120 cycles																	
0010: 14 cycles	1010: 182 cycles																	
0011: 20 cycles	1011: 240 cycles																	
0100: 29 cycles	1100: 300 cycles																	
0101: 42 cycles	1101: 400 cycles																	
0110: 56 cycles	1110: 480 cycles																	
0111: 72 cycles	1111: 600 cycles																	

### 15.11.7 ADC sampling time register 3 (ADC\_SAMPT3)

Address offset: 0x14

Reset value: 0x0000 0000

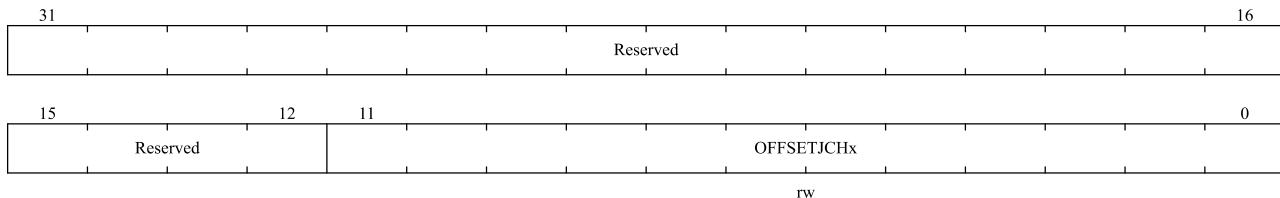


Bit field	Name	Description																
31: 0	SAMPx[3:0] x=0~7	<p>Channel x sample time selection</p> <p>These bits are used to independently select the sampling time for each channel. The channel selection bit must remain constant during the sampling period.</p> <table> <tr> <td>0000: 6 cycles</td><td>1000: 88 cycles</td></tr> <tr> <td>0001: 8 cycles</td><td>1001: 120 cycles</td></tr> <tr> <td>0010: 14 cycles</td><td>1010: 182 cycles</td></tr> <tr> <td>0011: 20 cycles</td><td>1011: 240 cycles</td></tr> <tr> <td>0100: 29 cycles</td><td>1100: 300 cycles</td></tr> <tr> <td>0101: 42 cycles</td><td>1101: 400 cycles</td></tr> <tr> <td>0110: 56 cycles</td><td>1110: 480 cycles</td></tr> <tr> <td>0111: 72 cycles</td><td>1111: 600 cycles</td></tr> </table>	0000: 6 cycles	1000: 88 cycles	0001: 8 cycles	1001: 120 cycles	0010: 14 cycles	1010: 182 cycles	0011: 20 cycles	1011: 240 cycles	0100: 29 cycles	1100: 300 cycles	0101: 42 cycles	1101: 400 cycles	0110: 56 cycles	1110: 480 cycles	0111: 72 cycles	1111: 600 cycles
0000: 6 cycles	1000: 88 cycles																	
0001: 8 cycles	1001: 120 cycles																	
0010: 14 cycles	1010: 182 cycles																	
0011: 20 cycles	1011: 240 cycles																	
0100: 29 cycles	1100: 300 cycles																	
0101: 42 cycles	1101: 400 cycles																	
0110: 56 cycles	1110: 480 cycles																	
0111: 72 cycles	1111: 600 cycles																	

### 15.11.8 ADC injected channel data offset register x (ADC\_JOFFSETx) (x=1...4)

Address offset: 0x18-0x24

Reset value: 0x0000 0000

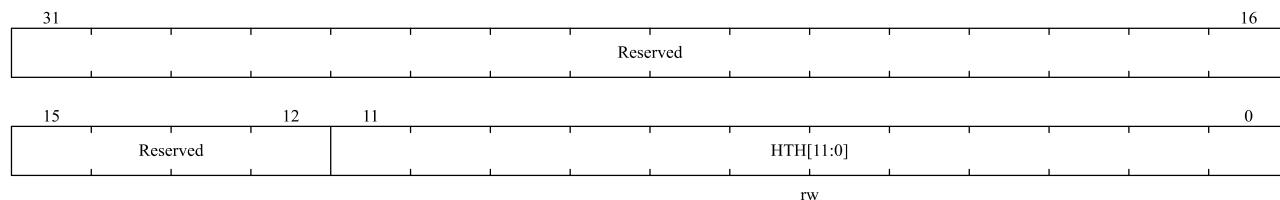


Bit field	Name	Description
31:12	Reserved	Reserved, the reset value must be maintained
11:0	OFFSETJCHx[11:0]	Data offset for injected channel x These bits define the values used to subtract from the original conversion data when the conversion is injected into the channel. The result of the conversion can be read in the ADC_JDATx register.

### 15.11.9 ADC watchdog high threshold register (ADC\_WDGHIGH)

Address offset: 0x28

Reset value: 0x0000 0FFF

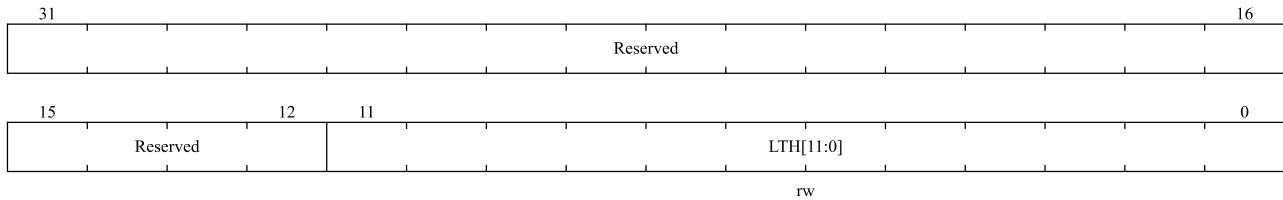


Bit field	Name	Description
31:12	Reserved	Reserved, the reset value must be maintained
11:0	HTH[11:0]	Analog watchdog high threshold These bits define the high thresholds for analog watchdog.

### 15.11.10 ADC watchdog low threshold register (ADC\_WDGLOW)

Address offset: 0x2C

Reset value: 0x0000 0000

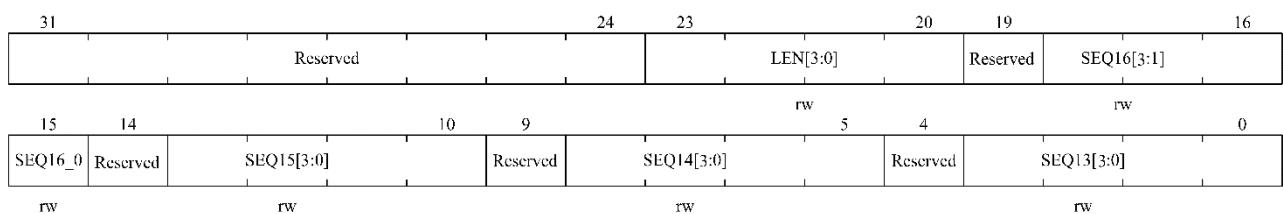


Bit field	Name	Description
31:12	Reserved	Reserved, the reset value must be maintained
11:0	LTH[11:0]	Analog watchdog low threshold These bits define the low thresholds for analog watchdog.

### 15.11.11 ADC regular sequence register 1 (ADC\_RSEQ1)

Address offset: 0x30

Reset value: 0x0000 0000



Bit field	Name	Description
31:24	Reserved	Reserved, the reset value must be maintained
23:20	LEN[3:0]	Regular channel sequence length These bits are software-defined as the number of channels in the regular sequence channel conversion. 0000: 1 conversion 0001: 2 conversions ... 1111: 16 conversions
19	Reserved	Reserved, the reset value must be maintained
18:15	SEQ16[3:0]	16th conversion in regular sequence These bits are software-defined as the number (0 to 15) of the 16th conversion channel in the conversion sequence.
14	Reserved	Reserved, the reset value must be maintained
13:10	SEQ15[3:0]	15th conversion in regular sequence
9	Reserved	Reserved, the reset value must be maintained
8:5	SEQ14[3:0]	14th conversion in regular sequence
4	Reserved	Reserved, the reset value must be maintained
3:0	SEQ13[3:0]	13th conversion in regular sequence

### 15.11.12 ADC regular sequence register 2 (ADC\_RSEQ2)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29		25	24		20	19	16
Reserved			SEQ12[3:0]		Reserved	SEQ11[3:0]		Reserved	SEQ10[3:1]
			rw			rw			
15	14		10	9		5	4		0
SEQ10_0	Reserved		SEQ9[3:0]		Reserved	SEQ8[3:0]		Reserved	SEQ7[3:0]
			rw			rw			

Bit field	Name	Description
31:29	Reserved	Reserved, the reset value must be maintained
28:25	SEQ12[3:0]	12th conversion in regular sequence These bits are software-defined as the number (0 to 15) of the 12th conversion channel in the conversion sequence.
24	Reserved	Reserved, the reset value must be maintained
23:20	SEQ11[3:0]	11th conversion in regular sequence
19	Reserved	Reserved, the reset value must be maintained
18:15	SEQ10[3:0]	10th conversion in regular sequence
14	Reserved	Reserved, the reset value must be maintained
13:10	SEQ9[3:0]	9th conversion in regular sequence
9	Reserved	Reserved, the reset value must be maintained
8:5	SEQ8[3:0]	8th conversion in regular sequence
4	Reserved	Reserved, the reset value must be maintained
3:0	SEQ7[3:0]	7th conversion in regular sequence

### 15.11.13 ADC regular sequence register 3 (ADC\_RSEQ3)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29		25	24		20	19	16
Reserved			SEQ6[3:0]		Reserved	SEQ5[3:0]		Reserved	SEQ4[3:1]
			rw			rw			
15	14		10	9		5	4		0
SEQ4_0	Reserved		SEQ3[3:0]		Reserved	SEQ2[3:0]		Reserved	SEQ1[3:0]
			rw			rw			

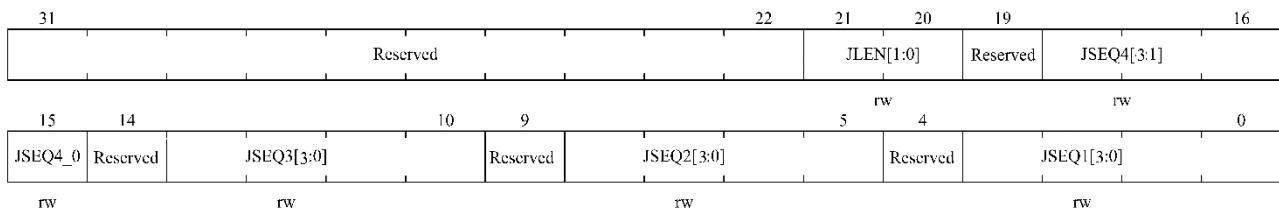
Bit field	Name	Description
31:29	Reserved	Reserved, the reset value must be maintained

Bit field	Name	Description
28:25	SEQ6[3:0]	6th conversion in regular sequence These bits are software-defined as the number (0 to 15) of the 6th transition channel in the conversion sequence.
24	Reserved	Reserved, the reset value must be maintained
23:20	SEQ5[3:0]	5th conversion in regular sequence
19	Reserved	Reserved, the reset value must be maintained
18:15	SEQ4[3:0]	4th conversion in regular sequence
14	Reserved	Reserved, the reset value must be maintained
13:10	SEQ3[3:0]	3rd conversion in regular sequence
9	Reserved	Reserved, the reset value must be maintained
8:5	SEQ2[3:0]	2nd conversion in regular sequence
4	Reserved	Reserved, the reset value must be maintained
3:0	SEQ1[3:0]	1st conversion in regular sequence

### 15.11.14 ADC Injection sequence register (ADC\_JSEQ)

Address offset: 0x3C

Reset value: 0x0000 0000



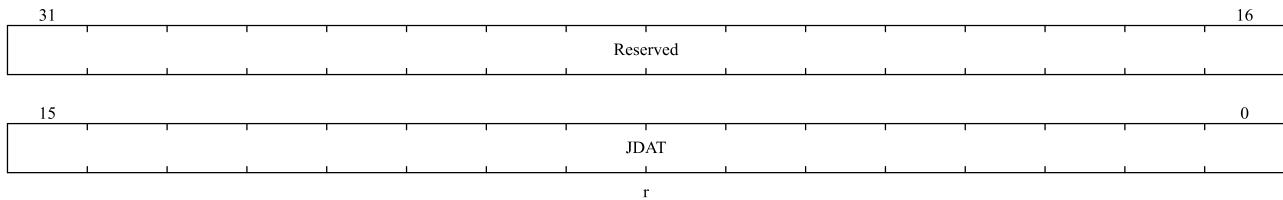
Bit field	Name	Description
31:22	Reserved	Reserved, the reset value must be maintained
21:20	JLEN[1:0]	Injected sequence length These bits are software-defined as the number of channels in the injected channel conversion sequence. 00: 1 conversion 01: 2 conversions 10: 3 conversions 11: 4 conversions
19	Reserved	Reserved, the reset value must be maintained
18:15	JSEQ4[3:0]	This is the 4th conversion in the injected sequence. These bits are software-defined as the number (0 to 15) of the fourth transition channel in the conversion sequence. <i>Note: Different from regular conversion sequences, if the length of ADC_JSEQ.JLEN[1:0] is less than 4, the sequence of conversion starts from (4-JLEN). For example, ADC_JSEQ[21:0] = 10</i>

Bit field	Name	Description
		<i>00011 00011 00111 00010 means that the scan conversion will be converted in the following channel order: 7, 3, 3 instead of 2, 7, 3.</i>
14	Reserved	Reserved, the reset value must be maintained
13:10	JSEQ3[3:0]	3rd conversion in injected sequence
9	Reserved	Reserved, the reset value must be maintained
8:5	JSEQ2[3:0]	2nd conversion in injected sequence
4	Reserved	Reserved, the reset value must be maintained
3:0	JSEQ1[3:0]	1st conversion in injected sequence

### 15.11.15 ADC injection data register x (ADC\_JDATx) (x= 1...4)

Address offset: 0x40 - 0x4C

Reset value: 0x0000 0000

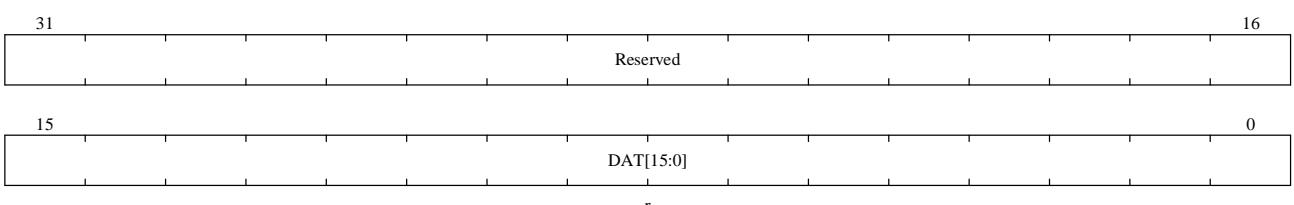


Bit field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained
15:0	JDATx[15:0]	Injected data for conversions These bits are read-only and contain the conversion results of the injected channel. The data is left-aligned or right-aligned

### 15.11.16 ADC regulars data register (ADC\_DAT)

Address offset: 0x50

Reset value: 0x0000 0000

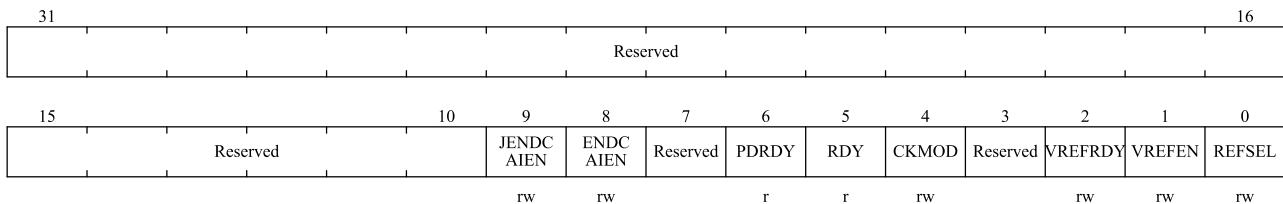


Bit field	Name	Description
32:16	Reserved	Reserved, the reset value must be maintained
15:0	DAT[15:0]	Regular data for conversion These bits are read-only and contain the conversion results of the regular channel. The data is left-aligned or right-aligned.

### 15.11.17 ADC control register 3 (ADC\_CTRL3)

Address offset: 0x54

Reset value: 0x0000 0040



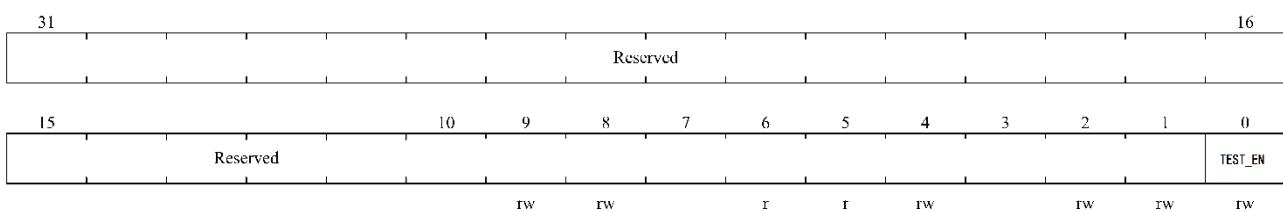
Bit field	Name	Description
31:10	Reserved	Reserved, the reset value must be maintained
9	JENDCAIEN	<p>Interrupt enable for any injected channels</p> <p>This bit is set and cleared by the software to enable/disable the injection channel conversion end interrupt</p> <p>0: ADC_STS.JENDCA interrupt is disabled</p> <p>1: ADC_STS.JENDCA interrupt is enabled</p>
8	ENDCAIEN	<p>Interrupt enable for any channels</p> <p>This bit is set and cleared by the software to enable/disable any channel conversion end the interrupt</p> <p>0: ADC_STS.ENDCA interrupt is disabled</p> <p>1: ADC_STS.ENDCA interrupt is enabled</p>
7	Reserved	Reserved, the value is forcibly set to 1.
6	PDRDY	<p>ADC power ready</p> <p>0: ADC is powered on</p> <p>1: ADC is powered down</p>
5	RDY	<p>ADC ready</p> <p>0: Not ready</p> <p>1: Get ready</p>
4	CKMOD	<p>Clock mode</p> <p>0: Select AHB for synchronization clock</p> <p>1: Select PLL for asynchronous clock</p>
3	Reserved	Reserved, the reset value must be maintained
2	VREFRDY	<p>VREFINT_READY</p> <p>ADC internal input buffer ready status, software must check this status bit before measuring</p>

Bit field	Name	Description
		VREFINT 0: VREFINT not ready 1: VREFINT is ready
1	VREFEN	VREFINT Enable ADC internal input buffer is enabled, software must enable this bit before measuring VREFINT 0: Disable VREFINT measurement 1: Enable VREFINT measurement
0	REFSEL	ADC reference source selset 0: The reference source is the external reference VDD 1: The reference source is the internal voltage 2.4V

### 15.11.18 ADC test register (ADC\_TEST)

Address offset: 0x58

Reset value: 0x0000 0000

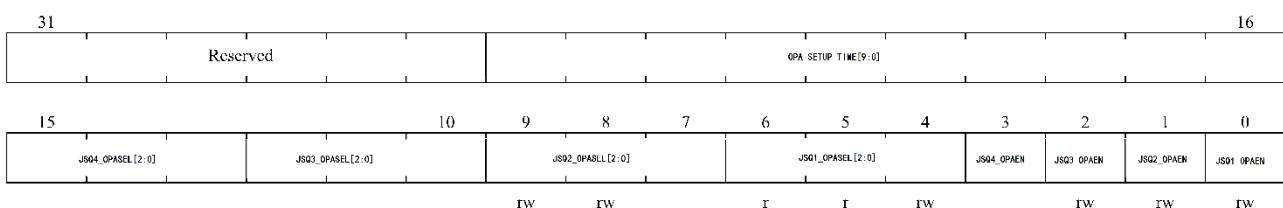


Bit field	Name	Description
31:1	Reserved	Reserved, the reset value must be maintained
0	TEST_EN	ADC test mode enable 0: ADC works in normal sampling mode 1: ADC works in test mode

### 15.11.19 ADC OPA control register (ADC\_OPACTRL)

Address offset: 0x5C

Reset value: 0x0000 0000



Bit field	Name	Description
31:26	Reserved	Reserved, the reset value must be maintained
25:16	OPA_SETUP_TIME[9:0]	Setup time for OPA mux 0: 0 ADC clock cycles 1: 1 ADC clock cycles ... 1023: 1023 ADC clock cycles
15:13	JSQ4_OPASEL[2:0]	Injected channel 4 for OPA mux selection
12:10	JSQ3_OPASEL [2:0]	Injected channel 3 for OPA mux selection
9:7	JSQ2_OPASEL [2:0]	Injected channel 2 for OPA mux selection
6:4	JSQ1_OPASEL [2:0]	Injected channel 1 for OPA mux selection
3	JSQ4_OPAEN	Injected channel 4 for OPA mux enable 0: do not switch 1: Switch to the channel indicated by the JSQ4_OPASEL register
2	JSQ3_OPAEN	Injected channel 3 for OPA mux enable 0: do not switch 1: Switch to the channel indicated by the JSQ3_OPASEL register
1	JSQ2_OPAEN	Injected channel 2 for OPA mux enable 0: do not switch 1: Switch to the channel indicated by the JSQ2_OPASEL register
0	JSQ1_OPAEN	Injected channel 1 for OPA mux enable 0: do not switch 1: Switch to the channel indicated by the JSQ1_OPASEL register

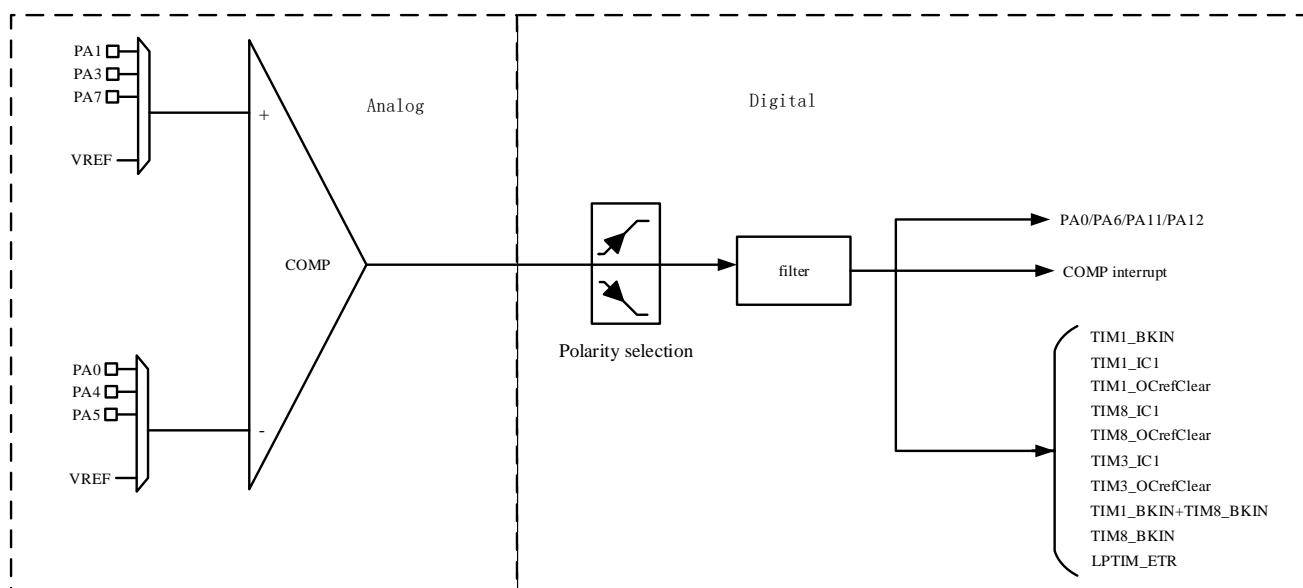
## 16 Comparator (COMP)

The COMP module is used to compare the magnitude of the two input analog voltages, and output high/low levels according to the comparison result. When the "INP" input voltage is higher than the "INM" input voltage, the comparator output is high, and when the "INP" input voltage is lower than the "INM" input voltage, the comparator output is low.

### 16.1 COMP system connection block diagram

The COMP module supports an independent comparator, which is connected to the APB1 bus.

Figure 16-1 Comparator System Connection Diagram



### 16.2 COMP features

- An independent comparator COMP, which is a low-power comparator (can work in LPRUN, SLEEP and STOP modes)
- Built-in a 64-level programmable reference input comparison voltage source VREF
- Support filter clock, filter reset
- Output polarity can be configured high and low
- Hysteresis configuration configurable none, low, medium, high
- Comparison results can be output to I/O ports or trigger timers for capture events, OCREF\_CLR events, brake events, and interrupt generation
- Input channel can select I/O port, VREF
- Can be equipped with read-only or read-write, in the case of locking, it needs to be reset to unlock

- Support blanking (Blanking), the blanking source that can be configured to generate Blanking can wake up the system from low-power mode by generating interrupts, and COMP has the ability to wake up from STOP
- Configurable filter window size
- Configurable filter threshold size
- Configurable sampling frequency for filtering

## 16.3 COMP configuration process

Complete configuration items are as follows. If the default configuration is used, skip the corresponding configuration items.

1. Configurable hysteresis level COMP\_CTRL.HYST[1:0]
2. Configure the output polarity COMP\_CTRL.POL
3. Configuration input selection, comparator non-inverting input COMP\_CTRL.INPSEL[3:0], inverting input COMP\_CTRL.INMSEL [2:0]
4. Select COMP\_CTRL.OUTSEL[3:0] for configuration output
5. Configure the blanking source COMP\_CTRL.BLKING[2:0]
6. Configure the filter sampling window COMP\_FILC.SAMPW[4:0]
7. Configure the threshold COMPx\_FILC.THRESH[4:0] (threshold should be greater than COMPx\_FILC.SAMPW[4:0]/2)
8. Configure the filter sampling frequency (for timer applications, sampling frequency should be greater than 5MHz)
9. Enable COMPx\_FILC.FILEN filter
10. Enable COMPx\_CTRL.EN on the comparator

*Note: For the above steps, you need to turn on the filter enable first, and then turn on the comparator enable. The comparator enable needs to be enabled after the filter (if enabled) configuration and enable are completed. In addition, when the comparator control register is locked to LOCK, The lock can only be canceled by resetting.*

## 16.4 COMP working mode

### 16.4.1 Independent comparator

One comparator can be configured independently to complete the comparator function. The output of the comparator can be output to the IO port, the comparator supports different remapping ports, and the output of the comparator can be selected and connected to the corresponding port through configuration.

Comparator output, support trigger events, for example, it can be configured as timer 1, timer 8 brake function.

*Note: Refer to the comparator interconnection for specific configuration.*

## 16.5 Comparator interconnection

For the interconnection of the comparator output ports, please refer to the AFIO remapping chapter, Remap IO multiplexing function in GPIOx\_AFL/AFH.

COMP\_OUT can be mapped to PA0/PA6/PA11/PA12 Comparator INP pins have the following configuration:

INPSEL	COMP
00	PA1
01	PA3
10	VREF
11	PA7

The comparator INM pins have the following configuration.

INMSEL	COMP
00	VREF
01	PA0
10	PA4
11	PA5

Comparator output TRIG signal has the following interconnection.

TRIG	COMP
0000	NC
0001	TIM1_BKIN
0010	TIM1_IC1
0011	TIM1_OCrefclear
0100	TIM8_IC1
0101	TIM8_OCrefclear
0110	TIM3_IC1
0111	TIM3_OCrefclear
1000	--
1001	--
1010	--
1011	TIM1_BKIN + TIM8_BKIN
1100	TIM8_BKIN
1101	LPTIM_ETR
Other	--

## 16.6 Interrupt

COMP supports interrupt response. There are two cases of interrupt generation as follows:

- COMP\_CTRL register sets the polarity of the POL bit is not reversed and the interrupt is enabled. When INPSEL > INMSEL, the comparator interrupt will be generated when the OUT bit of the COMP\_CTRL register is set to 1

by hardware.

- COMP\_CTRL register sets the polarity of the POL bit is reversed and the interrupt is enabled. When INPSEL < INMSEL, the comparator interrupt will be generated when the OUT bit of the COMP\_CTRL register is set to 1 by hardware.

*Note: The use of COMP interrupt needs to configure the EXTI line first, refer Table 6-1 Vector table.*

## 16.7 COMP register

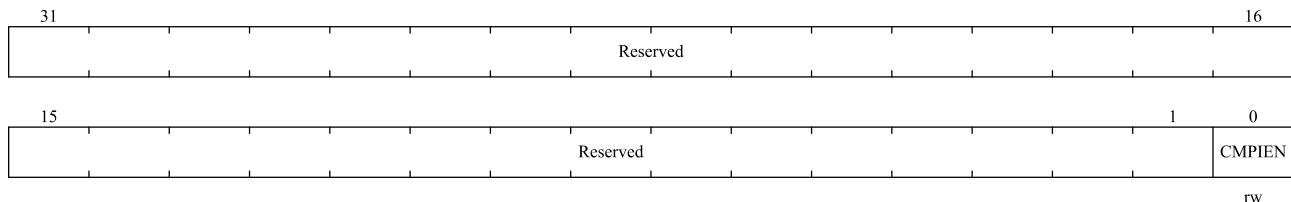
### **16.7.1 COMP register overview**

Table 16-1 COMP register overview

### 16.7.2 COMP interrupt Enable register (COMP\_INTEN)

Address offset : 0x00

Reset value : 0x0000 0000

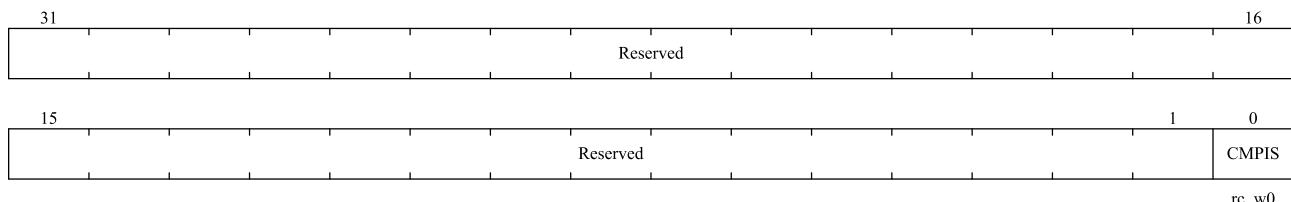


Bit field	Name	Description
31:1	Reserved	Reserved□the reset value must be maintained
0	CMPIEN	COMP interrupt enable 0: disable 1: enable

### 16.7.3 COMP interrupt register (COMP\_INTSTS)

Address offset : 0x04

Reset value : 0x0000 0000



Bit field	Name	Description
31:1	Reserved	Reserved□the reset value must be maintained
0	CMPIS	interrupt status of COMP Write 0 to clear

### 16.7.4 COMP lock register(COMP\_LOCK)

Address offset : 0x0C

Reset value : 0x0000 0000

31	Reserved															16															
15	Reserved															1 0															
CMPLK																															
rw																															

Bit field	Name	Description
31:1	Reserved	Reserved□the reset value must be maintained
0	CMPLK	This bit can only be reset then written once by software. If software is set to 1, the COMP_CTRL register will become a read-only register 0: COMP_CTRL can be read and written 1: COMP_CTRL read only

### 16.7.5 COMP control register (COMP\_CTRL)

Address offset : 0x10

Reset value : 0x0000 0000

31	Reserved										21	20	19	18	17	16
											CLKSEL	PWRMD	Reserved	OUT	BLKING[2]	
15	14	13	12	11	10	7	6				rw	rw	rw	r	rw	
BLKING[1:0]	HYST[1:0]	POL		OUTTRG[3:0]		Reserved	INPSEL[1:0]	Reserved	INMSEL[1:0]	EN	rw	rw	rw	rw	rw	

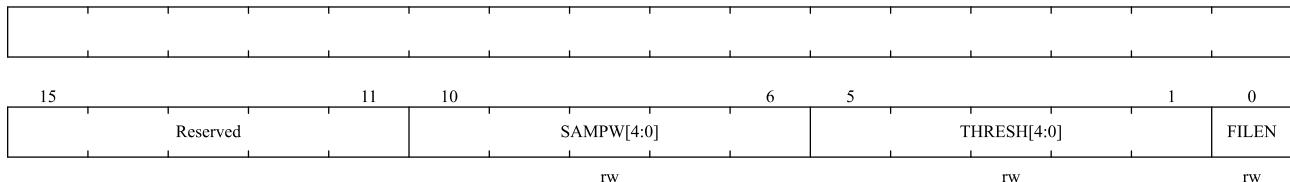
Bit field	Name	Description
31:21	Reserved	Reserved□the reset value must be maintained
20	CLKSEL	COMP operating clock selection 0: System clock (SYSCLK) 1: Low-speed working clock, can work in STOP mode or LPRUN mode.
19	PWRMD	COMP power select 0: normal mode 1: Low power mode
18	Reserved	Reserved□the reset value must be maintained
17	OUT	This read-only bit is COMP output state. 0: Output is low 1: Output is high
16:14	BLKING[2:0]	These bits select which Timer output controls the COMP output blanking. 000: No blanking 001: TIM1 OC5 selected as blanking source 010: TIM8 OC5 selected as blanking source Other values: reserved
13:12	HYST[1:0]	These bits control the hysteresis level.

Bit field	Name	Description
		00: No hysteresis 01: Low hysteresis 10: Medium hysteresis 11: High hysteresis
11	POL	This bit is used to invert the COMP output. 0: Output is not inverted 1: Output is inverted
10:7	OUTTRG[3:0]	These bits select which Timer input must be connected with the COMP output. 0000: Reserved 0001: TIM1_BKIN 0010: TIM1_IC1 0011: TIM1_OCrefclear 0100: TIM8_IC1 0101: TIM8_OCrefclear 0110: TIM3_IC1 0111: TIM3_OCrefclear 1000: Reserved 1001: Reserved 1010: Reserved 1011: TIM1_BKIN+TIM8_BKIN 1100: TIM8_BKIN 1101: LPTIM_ETR 1110: Reserved 1111: Reserved
6	Reserved	Reserved, the reset value must be maintained
5:4	INPSEL[2:0]	COMP positive select 00: PA1 01: PA3 10: VREF 11: PA7
3	Reserved	Reserved, the reset value must be maintained
2:1	INMSEL[2:0]	COMP negative input select 00: VREF 01: PA0 10: PA4 11: PA5
0	EN	This bit switches COMP ON/OFF. 0: disabled 1: enabled

## 16.7.6 COMP Filter control register (COMP\_FILC)

Address offset : 0x14

Reset value : 0x0000 0000

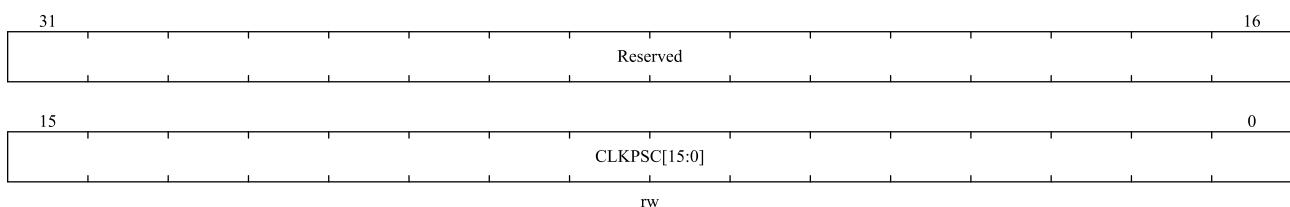


Bit field	Name	Description
31:11	Reserved	Reserved, the reset value must be maintained
10:6	SAMPW[4:0]	Filter sampling window size, sampling window = SAMPW + 1.
5:1	THRESH[4:0]	The filter threshold is set. At least the sampling threshold of the opposite state in the sample window can change the output state. This value is required to be greater than SAMPW / 2.
0	FILEN	Filter enable. 0: Disable 1: Enable

## 16.7.7 COMP Filter Frequency Division register (COMP\_FILP)

Address offset : 0x18

Reset value : 0x0000 0000

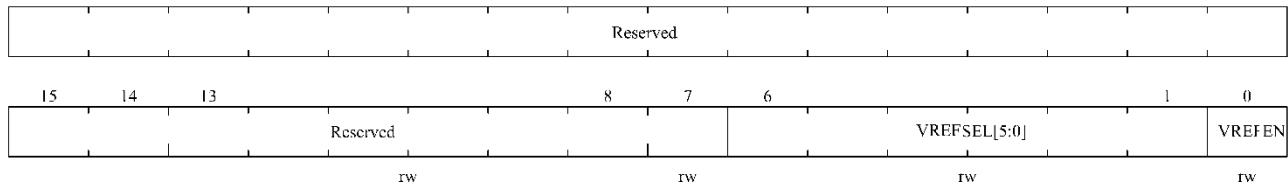


Bit field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained
15:0	CLKPSC[15:0]	Low filter sample clock prescale. System clock divider = CLK_PRE_CYCLE + 1, e.g. 0: Every cycle 1: Every 2 cycle 2: Every 3 cycle ...

### 16.7.8 COMP reference input compare voltage register (COMP\_INVREF)

Address offset : 0x40

Reset value : 0x0000 0000



<b>Bit field</b>	<b>Name</b>	<b>Description</b>
31:7	Reserved	Reserved, the reset value must be maintained
6:1	VREFSEL[5:0]	Comparator reference input comparison voltage VREF gear selection 0~VDDA, a total of 64 gears
0	VREFEN	Comparator reference input compare voltage VREF enable 0: disable 1: enable

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

### 17.1 Introduction

I2C(Inter-Integrated Circuit) bus is a widely used bus structure, it has only two bidirectional lines, namely data bus SDA and clock bus SCL. All devices compatible with I2C bus can communicate directly with each other through I2C bus with these two lines.

I2C interface connects microcontroller and serial I2C bus, and can be used for communication between MCU and external I2C devices. It supports standard speed mode and fast mode, it supports CRC calculation and verification, SMBus (System Management Bus) and PMBus (Power Management Bus), it also provides multi-host function to control all I2C bus specific timing, protocol, arbitration. I2C interface module also supports DMA mode, which can effectively reduce the CPU overload.

### 17.2 Main features

- Same interface can have both master function and slave function
- Parallel-bus to I<sup>2</sup>C protocol converter
- Supports 7-bit/10-bit address mode and broadcast addressing
- As I<sup>2</sup>C master, it can generate clock, start and stop signal
- As I<sup>2</sup>C slave, it supports address detection, stop bit detection function
- Support standard speed mode(up to 100 kHz) and fast mode(up to 400 kHz,1MHz)
- Support interrupt vector, Event interrupt and error interrupt share one interrupt vector
- Optional clock extending function
- Support DMA mode
- Optional PEC (Packet Error Check) generation and verification
- Compatible with SMBus 2.0 and PMBus

*Note: not all of the above features are included in all products. Please refer to the relevant data manual to confirm the I<sup>2</sup>C functions supported by the product.*

### 17.3 Function description

I2C interface is connected to I2C bus through data pin (SDA) and clock pin (SCL) to communicate with external devices. It can be connected to standard (up to 100kHz) or fast (up to 400kHz,1MHz) I2C bus. I2C module converts data from serial to parallel when receiving, and converts data from parallel to serial when sending. It support interrupt mode, users can enable or disable interrupt according to their needs.

### 17.3.1 SDA and SCL line control

I<sup>2</sup>C module has two interface lines: serial data line (SDA) and serial clock line (SCL). Devices connected to the bus and transmit information to each other through these two wires. SDA and SCL are two-way wires, it should be connected to a current source or the positive of the power supply with a pull-up resistor. When the bus is idle, both lines are high level. The output of device which is connected to the bus must have open drain or open collector to provide wired-AND functionality. The data on I<sup>2</sup>C bus can reach 100 kbit/s in standard mode and 1000 kbit/s in fast mode. Since devices of different processors may be connected to the I<sup>2</sup>C bus, the levels of logic '0' and logic '1' are not fixed and depend on the actual level of VDD.

If the clock extending is allowed, the SCL line is pulled down which can be avoided the overload error during receiving and the under load error during transmission.

For example, when in the transmission mode, if the transmit data register is empty and the byte transmit end bit is set (I<sup>2</sup>C\_STS1.TXDATE = 1, I<sup>2</sup>C\_STS1.BSF = 1), the I<sup>2</sup>C interface keeps the clock line low before transmission to wait for the software to read STS1 and write the data into the data register (both buffer and shift register are empty); when In the receive mode, if the data register is not empty and the byte sending end bit is set (I<sup>2</sup>C\_STS1.RXDATNE = 1, I<sup>2</sup>C\_STS1.BSF = 1), the I<sup>2</sup>C interface keeps the clock line low after receiving the data byte, waiting for the software to read STS1, and then read the data register(buffer and shift register are full).

If clock extending is disable in slave mode, if the receive data register is not empty (I<sup>2</sup>C\_STS1.RXDATNE = 1) in the receive mode, and the data has not been read before receiving the next byte, an overrun error will issue and the last word byte will be discarded. In transmit mode, if the transmit data register is empty (I<sup>2</sup>C\_STS1.TXDATE = 1), no new data is written into the data register before the next byte must be sent, an underrun error will issue. The same byte will be send repeatedly. In this case, duplicate write conflicts are not controlled.

### 17.3.2 Software communication process

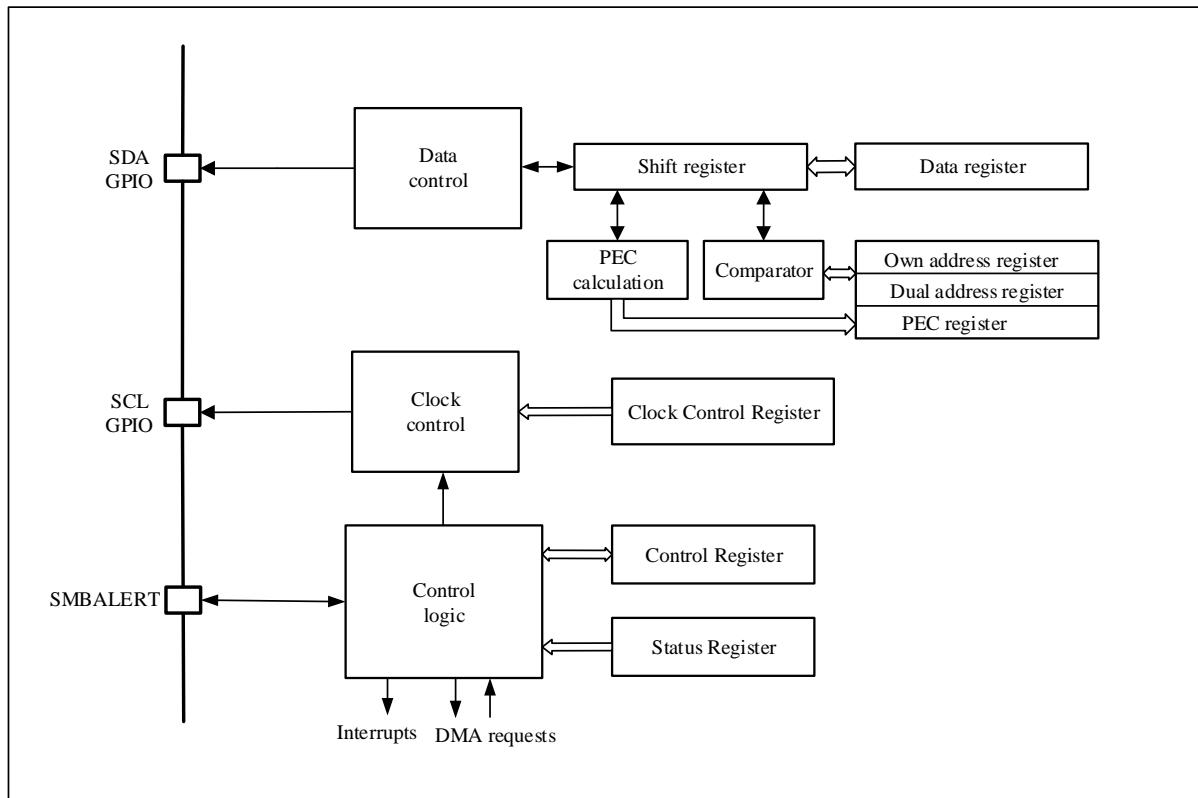
The data transmission of I<sup>2</sup>C device is divided into master and slave. Master is the device responsible for initializing the transmission of data on the bus and generating clock signal. At this time, any addressed device is a slave. Whether the I<sup>2</sup>C device is a master or a slave, it can send or receive data. Therefore, the I<sup>2</sup>C interface supports four operation modes:

- Slave transmitter mode
- Slave receiver mode
- Master transmitter mode
- Master receiver mode

After system reset, I<sup>2</sup>C works in slave mode by default. The I<sup>2</sup>C interface is configured by software to send a start bit on the bus, and then the interface automatically switches from the slave mode to the master mode. When arbitration is lost or a stop signal is generated, the interface will switch to the slave mode from the receive mode.

The block diagram of I<sup>2</sup>C interface is shown in the figure below.

Figure 17-1 I2C functional block diagram

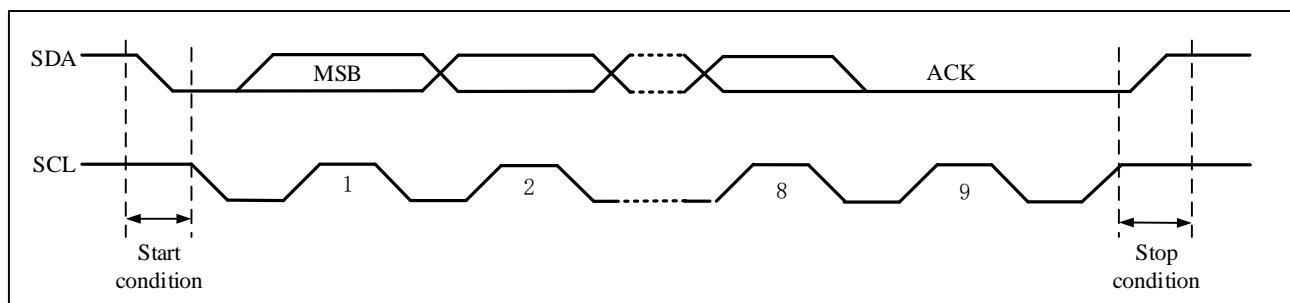


Note: in SMBus mode, SMBALERT is an optional signal. If SMBus is disabled, the signal cannot be used

### 17.3.2.1 Start and stop conditions

All data transfers always start with the start bit and end with the stop bit. The start and stop conditions are generated by software in the master mode. Start bit is a level conversion from high to low on SDA line when SCL is high. Stop bit is a level transition from low to high on SDA line when SCL is high. as shown in the figure below.

Figure 17-2 I2C bus protocol



### 17.3.2.2 Clock synchronization and Arbitration

The I2C module supports multi-master arbitration, which means two masters can initiate an I2C START operation concurrently when the bus is inactive. So some mechanisms are needed to grant a master the access to the bus. This process is generally named Clock Synchronization and Arbitration.

I2C module has two key features:

- SDA and SCL are drain open circuit structures, and the signal "wire-and" logic is realized through an external pull-up resistor.
- SDA and SCL pins will also detect the level on the pin while outputting the signal to check whether the output is consistent with the previous output. This provides the hardware basis for "Clock Synchronization" and "Bus Arbitration".

The I2C device on the bus is to output logic 0 by "grounding the line". Based on the characteristics of the I2C bus, if one device sends logic 0 and the other sends logic 1, then the line sees only logic 0, so there is no possibility of level conflicts on the line.

The physical connection of the bus allows the master to read data while writing data to the bus. In this way, when two masters are competing for the bus, the one that sends logic 0 does not know the occurrence of the competition. Only the one that sends logic 1 will find the conflict (when writing a logic 1, but read 0) and exit the competition.

### Clock synchronization

The high-to-low switching of the SCL line causes the devices to begin counting their low-level periods, and once the device's clock goes low, it keeps the SCL line in this state until the high-level of the clock is reached. However, if another clock is still in the low period, the low-to-high switch of this clock will not change the state of the SCL line. Therefore, the SCL line is kept low by the device with the longest low-level period. A device with a short low-level period will enter a high-level wait state.

When all related devices have counted their low-level periods, the clock line is released and goes high-level, after which there is no difference in the state of the device clock and SCL lines, and all devices will begin counting their high-level periods, the device that completes the high-level period first will pull the SCL line low again.

In this way, the low-level period of the generated synchronous SCL clock is determined by the device with the longest low-level clock period, and the high-level period is determined by the device with the shortest high-level clock period.

### Arbitration

Arbitration, like synchronization, is to resolve bus control conflicts in the case of multiple masters. The arbitration process has nothing to do with the slave. When the two masters both produce a valid start bit when the bus is idle, in this case, it is necessary to decide which master will complete the data transmission. This is the process of arbitration.

Each master controller does not have the priority level of controlling the bus, which is all determined by arbitration. The bus control is determined and carried out bit by bit. They follow the principle of "low level first", that is, whoever sends the low level first will control the bus. During the arbitration of each bit, when SCL is high, each host checks whether its own SDA level is the same as that sent by itself. In theory, if the content transmitted by two hosts is exactly the same, then they can successfully transmit without errors. If a host sends a high level but detects that the SDA line is low, it considers that it has lost arbitration and shuts down its SDA output driver, while the other host continues to complete its own transmission.

#### 17.3.2.3 I2C data communication flow

Each I2C device is identified by a unique address. According to the device function, they can be either a transmitter or a receiver.

The I2C host is responsible for generating the start bit and the end bit in order to start and end a transmission. And is responsible for generating the SCL clock.

The I2C module supports 7-bit and 10-bit addresses, and the user can configure the address of the I2C slave through software. After the I2C slave detects the start bit on the I2C bus, it starts to receive the address from the bus, and compares the received address with its own address. Once the two addresses are matched, the I2C slave will send an acknowledgement (ACK) and respond to subsequent commands on the bus: send or receive the requested data. In addition, if the software opens a broadcast call, the I2C slave always sends a confirmation response to a broadcast address (0x00).

Data and address are transmitted in 8-bit width, with the most significant bit first. The 1 or 2 bytes following the start condition is the address (1 byte in 7-bit mode, 2 bytes in 10-bit mode). The address is only sent in master mode. During the 9th clock period after 8 clocks of a byte transmission, the receiver must send back an acknowledge bit (ACK) to the transmitter, as shown in the Figure 17-2 I2C bus protocol.

Software can enable or disable acknowledgement (ACK), and can set the I2C interface address (7-bit, 10-bit address or general call address).

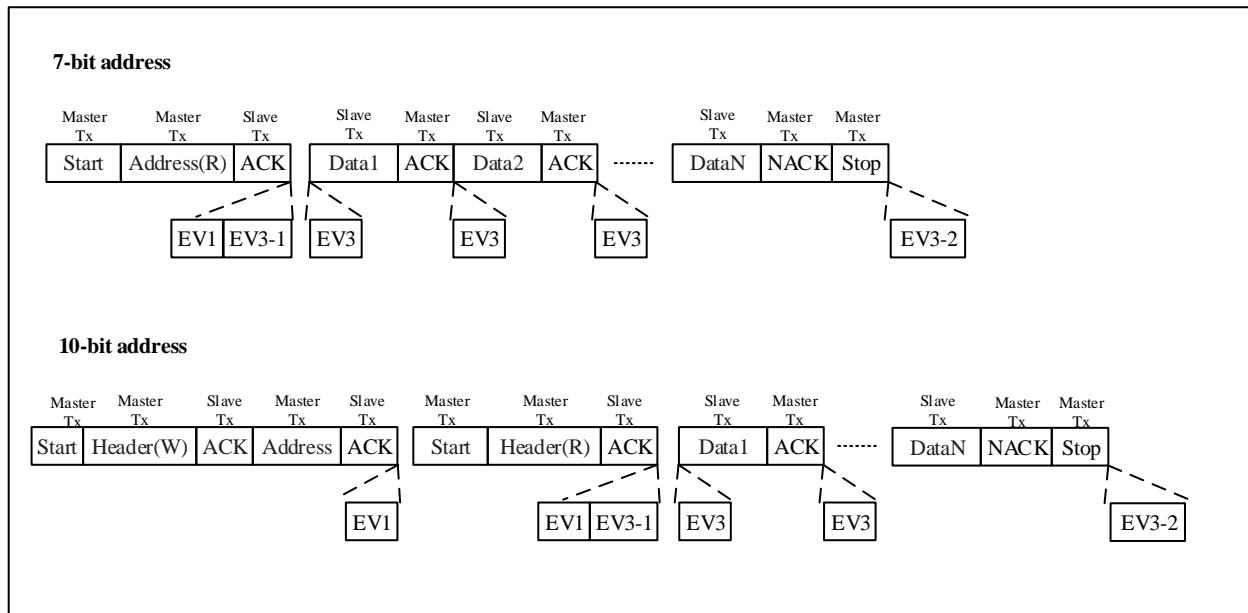
#### 17.3.2.4 I2C slave transmission mode

In slave mode, the transmission reception flag bit (I2C\_STS2.TRF) indicates whether it is currently in receiver mode or transmission mode. When sending data to I2C bus in transmission mode, the software should follow the following steps:

1. First, enable I2C peripheral clock and configure the clock related register in I2C\_CTRL1, ensuring the correct I2C timing. After these two steps are completed, I2C runs in slave mode, waiting for receiving start bit and address.
2. I2C slave receives a start bit first, and then receives a matching 7-bit or 10-bit address. I2C hardware will set the I2C\_STS1.ADDRF(received address and matched its own address). The software should monitor this bit regularly or have an interrupt to monitor this bit. After this bit is set, the software reads I2C\_STS1 register and then read I2C\_STS2 register to clear the I2C\_STS1.ADDRF bit. If the address is in 10 bit format, the I2C master should then generate a START and send an address header to the I2C bus. After detecting START and the following address header, the slave will continue to set I2C\_STS1.ADDRF bit. The software continues to read I2C\_STS1 register and read I2C\_STS2 register to clear the I2C\_STS1.ADDRF bit a second time.
3. I2C enters the data sending state, and now shift register and data register I2C\_DAT are all empty, so the hardware will set the I2C\_STS1.TXDATE(send data empty). At this time, the software can write the first byte data to I2C\_DAT register, however, because the byte of the I2C\_DAT register is immediately moved into the internal shift register, the I2C\_STS1.TXDATE bit is not cleared to zero. When the shift register is not empty, I2C starts to send data to I2C bus.
4. During the sending of the first byte, the software writes the second byte to I2C\_DAT, neither the I2C\_DAT register nor the shift register is empty. The I2C\_STS1.TXDATE bit is cleared to 0.
5. After the first byte is sent, I2C\_STS1.TXDATE is set again, and the software writes the third byte to I2C\_DAT, the same time, the I2C\_STS1.TXDATE bit is cleared. After that, as long as there is still data to be sent and I2C\_STS1.TXDATE is set to 1, the software can write a byte to I2C\_DAT register.

6. During the sending of the second last byte, the software writes the last data to the I2C\_DAT register to clear the I2C\_STS1.TXDATE flag bit, and then the I2C\_STS1.TXDATE status is no longer concerned. The I2C\_STS1.TXDATE bit is set after the second last byte is sent until the stop end bit is detected.
7. According to the I2C protocol, the I2C master will not send a ACK to the last byte received. Therefore, after the last byte is sent, the I2C\_STS1.ACKFAIL bit (acknowledge fail) of the I2C slave will be set to notify the software of the end of sending. The software writes 0 to the I2C\_STS1.ACKFAIL bit to clear this bit.

Figure 17-3 Slave transmitter transfer sequence diagram



#### Instructions □

1. EV1: I2C\_STS1.ADDRF = 1, read STS1 and then STS2 to clear the event.
2. EV3-1: I2C\_STS1.TXDATE=1, shift register is empty, data register is empty, write DAT.
3. EV3: I2C\_STS1.TXDATE=1, shift register is not empty, data register is empty, write DAT will clear the event.
4. EV3-2: I2C\_STS1.ACKFAIL=1, ACKFAIL bit of STS1 register write "0" to clear the event.

*Note: a) EV1 and EV3\_1 event prolongs the low SCL time until the end of the corresponding software sequence.*

*b) The software sequence of EV3 must be completed before the end of the current byte transfer.*

#### 17.3.2.5 I2C slave receiving mode

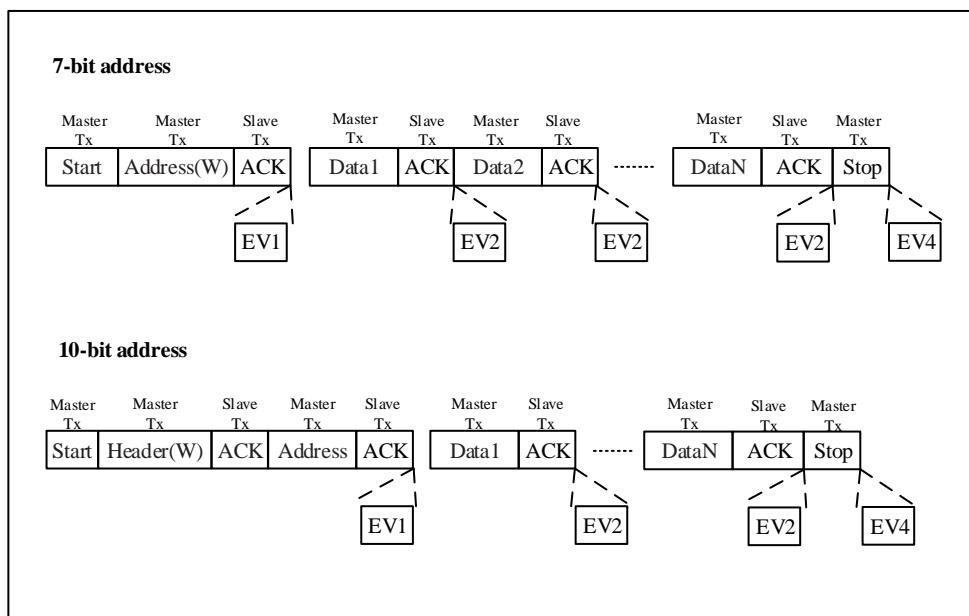
When receiving data in slave mode, the software should operate as follows:

1. First, enable I2C peripheral clock and configure the clock related register in I2C\_CTRL1 ensuring the correct I2C timing. After these two steps are completed, I2C runs in slave mode, waiting for receiving start bit and address.
2. After receiving the START condition and the matched 7-bit or 10-bit address, I2C hardware will set I2C\_STS1.ADDRF bit(the address received and matched with its own address) to 1. This bit should be detected by software polling or interrupt. After it is found that it is set, the software clears the I2C\_STS1.ADDRF bit by

reading I2C\_STS1 register first and then I2C\_STS2 register. Once the I2C\_STS1.ADDRF bit is cleared, The I2C slave starts to receive data from the I2C bus.

3. When the first byte is received, the I2C\_STS1.RXDATNE bit (the received data is not empty) is set to 1 by hardware. If the I2C\_CTRL2.EVTINTEN and I2C\_CTRL2.BUFINTEN bits are set, an interrupt is generated. The software should check this bit by polling or interrupt. Once it is found that it is set, the software can read the first byte of I2C\_DAT register, and then the I2C\_STS1.RXDATNE bit is cleared to 0. Note that if the I2C\_CTRL1.ACKEN bit is set, after receiving a byte, the slave should generate a response pulse.
4. At any time, as long as the I2C\_STS1.RXDATNE bit is set to 1, the software can read a byte from the I2C\_DAT register. When the last byte is received, I2C\_STS1.RXDATNE is set to 1 and the software reads the last byte.
5. When the slave detects the STOP bit on I2C bus, set I2C\_STS1.STOPF to 1, and if the I2C\_CTRL2.EVTINTEN bit is set, an interrupt will be generated. The software clears the I2C\_STS1.STOPF bit by reading the I2C\_STS1 register before writing the I2C\_CTRL1 register (see EV4 in the following figure).

Figure 17-4 Slave receiver transfer sequence diagram



Instructions:

1. EV1: I2C\_STS1.ADDRF = 1, read STS1 and then STS2 to clear the event.
2. EV2: I2C\_STS1.RXDATNE = 1, reading DAT will clear this event.
3. EV4: I2C\_STS1.STOPF = 1, reading STS1 and then writing the CTRL1 register will clear this event.

*Note: a) EV1 event prolongs the time when SCL is low until the end of the corresponding software sequence.*

*b) The software sequence of EV2 must be completed before the end of the current byte transmission.*

### 17.3.2.6 I2C master transmission mode

In the master mode, the I2C interface starts data transmission and generates a clock signal. Serial data transmission always starts with a start condition and ends with a stop condition. When the START condition is generated on the

bus through the start bit, the device enters the master mode.

When sending data to I2C bus in master mode, the software should operate as follows:

1. First, enable the I2C peripheral clock, and configure the clock-related registers in I2C\_CTRL1 to ensure the correct I2C timing. When these two steps are completed, I2C runs in the slave mode by default, waiting for receiving the start bit and address.
2. When BUSY=0, I2C\_CTRL1.STARTGEN bit set to 1, and the I2C interface will generate a start condition and switch to the master mode (I2C\_STS2.MSMODE=1).
3. Once the start condition is issued, I2C hardware will set I2C\_STS1.STARTBF bit(START bit flag)and then enters the master mode. If the I2C\_CTRL2.EVTINTEN bit is set, an interrupt will be generated. Then the software reads the I2C\_STS1 register and then writes a 7-bit address bit or a 10-bit address bit with an address header to the I2C\_DAT register to clear the I2C\_STS1.STARTBF bit. After the I2C\_STS1.STARTBF bit is cleared to 0, I2C starts sending addresses or address headers to I2C bus.

In 10-bit address mode, sending a header sequence will generate the following events:

- ◆ I2C\_STS1.ADDR10F bit is set by hardware, and if I2C\_CTRL2.EVTINTEN bit is set, an interrupt is generated. Then the master reads the STS1 register, and then writes the second address byte into the DAT register.
- ◆ I2C\_STS1.ADDRF bit is set by hardware, and if I2C\_CTRL2.EVTINTEN bit is set, an interrupt is generated. Then the master reads the STS1 register, followed by the STS2 register.

*Note: In the transmitter mode, the master device first sends the header byte (11110xx0) and then sends the lower 8 bits of the slave address. (where xx represents the highest 2 bits of the 10-bit address).*

In the 7-bit address mode, only one address byte needs to be sent out. Once the address byte is sent out:

- ◆ I2C\_STS1.ADDRF bit is set by hardware, and if I2C\_CTRL2.EVTINTEN bit is set, an interrupt is generated. Then the master device waits for reading the STS1 register once, followed by reading the STS2 register.

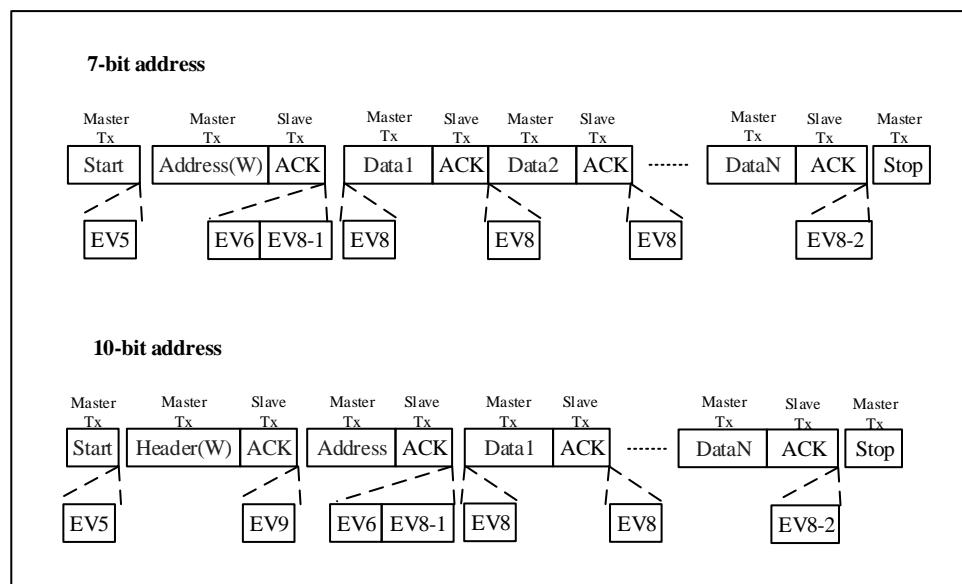
*Note: in the transmitter mode, when the master sends the slave address, set the lowest bit to "0".*

*Note: In the 7-bit address mode, do not configure the slave address as 0xF0 to prevent inadvertent hardware setting of the I2C\_STS1.ADDR10F bit.*

4. After the 7-bit or 10-bit address bit is sent, the I2C hardware sets the I2C\_STS1.ADDRF bit (address has been sent) to 1, if the I2C\_CTRL2.EVTINTEN bit is set, an interrupt is generated, and the software is cleared by reading the I2C\_STS1 register and then the I2C\_STS2 register I2C\_STS1.ADDRF.
5. I2C enters the data transmission state. Because the shift register and the data register (I2C\_DAT) are empty, the hardware sets the I2C\_STS1.TXDATE bit (transmission data empty) to 1, and then the software writes the first byte of data to the I2C\_DAT register, but because the byte written into the I2C\_DAT register is immediately moved into the internal shift register, the I2C\_STS1.TXDATE bit will not be cleared at this time. Once the shift register is not empty, I2C starts sending data to the bus.
6. During the transmission of the first byte, the software writes the second byte to I2C\_DAT, and I2C\_STS1.TXDATE is cleared at this time. At any time, as long as there is data waiting to be sent and the I2C\_STS1.TXDATE bit is set to 1, the software can write a byte to the I2C\_DAT register.

7. In the process of sending the penultimate byte, the software writes the last byte of data to I2C\_DAT to clear the I2C\_STS1.TXDATE flag bit. After that, there is no need to care about the status of the I2C\_STS1.TXDATE bit. The I2C\_STS1.TXDATE bit will be set after the penultimate byte is sent, and will be cleared when the stop bit (STOP) is sent.
8. After the last byte is sent, because the shift register and the I2C\_DAT register are empty at this time, the I2C host sets the I2C\_STS1.BSF bit (byte transmission end), and the I2C interface will keep SCL low before clearing the I2C\_STS1.BSF bit. After reading I2C\_STS1, writing to the I2C\_DAT register will clear the I2C\_STS1.BSF bit. The software sets the I2C\_CTRL1.STOPGEN bit at this time to generate a stop condition, and then the I2C interface will automatically return to the slave mode (I2C\_STS2.MSMODE bit is cleared).

Figure 17-5 Master transmitter transfer sequence diagram



Instructions:

1. EV5: I2C\_STS1.STARTBF = 1, reading STS1 and writing the address to the DAT register will clear the event.
2. EV6: I2C\_STS1.ADDRF = 1, read STS1 and then STS2 to clear the event.
3. EV8\_1: I2C\_STS1.TXDATE = 1, shift register is empty, data register is empty, write DAT register.
4. EV8: I2C\_STS1.TXDATE = 1, shift register is not empty, data register is empty, write to DAT register will clear the event.
5. EV8\_2: I2C\_STS1.TXDATE = 1, I2C\_STS1.BSF = 1, request to set stop bit. These two events are cleared by the hardware when a stop condition is generated.
6. EV9: I2C\_STS1.ADDR10F = 1, read STS1 and then write to DAT register to clear the event.

Note: a) EV5, EV6, EV9, EV8\_1 and EV8\_2 event prolonged the low SCL time until the end of the corresponding software sequence.

b) The software sequence of EV8 must be completed before the end of the current byte transfer.

c) When I2C\_STS1.TXDATE or I2C\_STS1.BSF bit is set, stop condition should be arranged when EV8\_2 occurs.

### 17.3.2.7 I2C master receiving mode

In master mode, software receiving data from I2C bus should follow the following steps:

1. First, enable the I2C peripheral clock and configure the clock-related registers in I2C\_CTRL1, in order to ensure that the correct I2C timing is output. After enabling and configuring, I2C runs in slave mode by default, waiting to receive the start bit and address.
2. When BUSY=0, set the I2C\_CTRL.STARTGEN bit, and the I2C interface will generate a start condition and switch to the master mode (I2C\_STS2.MSMODE bit is set to 1).
3. Once the start condition is issued, the I2C hardware sets I2C\_STS1.STARTBF(start bit flag) and enters the host mode. If the I2C\_CTRL2.EVTINTEN bit is set to 1, an interrupt will be generated. Then the software reads the I2C\_STS1 register and then writes a 7-bits address or a 10-bits address with an address header to the I2C\_DAT register, in order to clear the I2C\_STS1.STARTBF bit. After the I2C\_STS1.STARTBF bit is cleared to 0, I2C begins to send the address or address header to the I2C bus.

In 10-bits address mode, sending a header sequence will generate the following events:

- ◆ The I2C\_STS1.ADDR10F bit is set to 1 by hardware, and if the I2C\_CTRL2.EVTINTEN bit is set to 1, an interrupt will be generated. Then the master device reads the STS1 register, and then writes the second byte of address into the DAT register.
- ◆ The I2C\_STS1.ADDRF bit is set to 1 by hardware, and if the I2C\_CTRL2.EVTINTEN bit is set to 1, an interrupt will be generated. Then the master device reads the STS1 register and the STS2 register in sequence.

*Note: In the receiver mode, the master device sends the header byte (11110xx0) firstly, then sends the lower 8 bits of the slave address, and then resends a start condition followed by the header byte (11110xx1) (where xx represents the highest 2 digits of the 10-bits address).*

In the 7-bits address mode, only one address byte needs to be sent, once the address byte is sent:

- ◆ The I2C\_STS1.ADDRF bit is set to 1 by hardware, and if the I2C\_CTRL2.EVTINTEN bit is set to 1, an interrupt will be generated. Then the master device waits to read the STS1 register once, and then reads the STS2 register.

*Note: In the receiving mode, the master device sets the lowest bit as '1' when sending the slave address.*

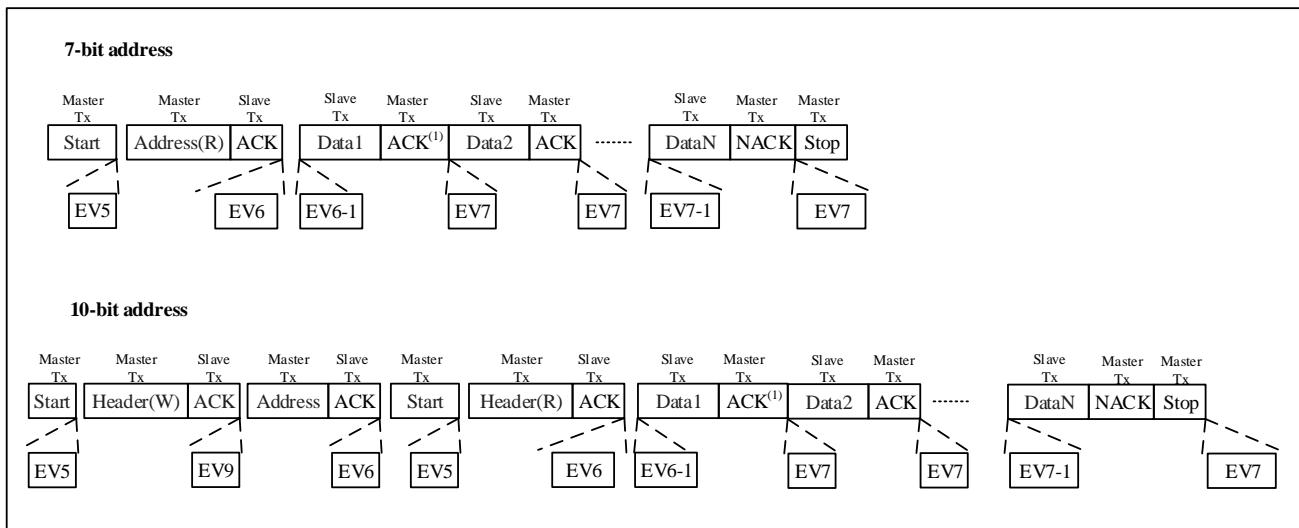
4. After the 7-bits or 10-bits address is sent, the I2C hardware sets the I2C\_STS1.ADDRF bit (address has been sent) to 1. If the I2C\_CTRL2.EVTINTEN bit is set to 1, an interrupt will be generated. The software clears the I2C\_STS1.ADDRF bit by reading the I2C\_STS1 register and the I2C\_STS2 register in sequence. If in the 10-bit address mode, software should set the I2C\_CTRL1.STARTGEN bit again to regenerate a START. After the START is generated, the I2C\_STS1.STARTBF bit will be set. The software should clear the I2C\_STS1.STARTBF bit by reading I2C\_STS1 firstly and then writing the address header to I2C\_DAT, and then the address header is sent to the I2C bus, I2C\_STS1.ADDRF is set to 1 again. The software should clear the I2C\_STS1.ADDRF bit again by reading I2C\_STS1 and I2C\_STS2 in sequence.
5. After sending the address and clearing the I2C\_STS1.ADDRF bit, the I2C interface enters the host receiving mode. In this mode, the I2C interface receives data bytes from the SDA line and sends them to the DAT register through the internal shift register. Once the first byte is received, the hardware will set the I2C\_STS1.RXDATNE bit (not empty flag bit of received data) to 1, and if the I2C\_CTRL1.ACKEN bit is set to 1, an acknowledge pulse will be sent. At this time, the software can read the first byte from the I2C\_DAT

register, and then the I2C\_STS1.RXDATNE bit is cleared to 0. After that, as long as I2C\_STS1.RXDATNE is set to 1, the software can read a byte from the I2C\_DAT register.

6. The master device sends a NACK after receiving the last byte from the slave device. After receiving the NACK, the slave device releases the control of SCL and SDA lines; the master device can send a stop/restart condition. In order to generate a NACK pulse after receiving the last byte, the software should clear the I2C\_CTRL1.ACKEN bit immediately after receiving the penultimate byte (N-1). In order to generate a stop/restart condition, the software must set the I2C\_CTRL1.STOPGEN bit or I2C\_CTRL1.STARTGEN to 1 after reading the penultimate data byte. This process needs to be completed before the last byte is received to ensure that the NACK is sent for the last byte.
7. After the last byte is received, the I2C\_STS1.RXDATNE bit is set to 1, and the software can read the last byte. Since I2C\_CTRL1.ACKEN has been cleared to 0 in the previous step, I2C no longer sends ACK for the last byte, and generates a STOP bit after the last byte is sent.

*Note: The above steps require the number of bytes  $N > 1$ . If  $N=1$ , step 6 should be executed after step 4, and it needs to be completed before the reception of byte is completed.*

Figure 17-6 Master receiver transfer sequence diagram



Instructions:

1. EV5: I2C\_STS1.STARTBF=1, reading STS1 and then writing the address into the DAT register will clear this event.
2. EV6: I2C\_STS1.ADDRF=1, reading STS1 and STS2 in sequence will clear this event. In the 10-bits master receiving mode, the I2C\_CTRL1.STARTGEN should be set to 1 after this event.
3. EV6\_1: There is no corresponding event flag, only suitable for receiving 1 byte. Just after EV6 (that is after clearing I2C\_STS1.ADDRF), the generation bits for acknowledge and stop condition should be cleared.
4. EV7: I2C\_STS1.RXDATNE=1, read the DAT register to clear this event.
5. EV7\_1: I2C\_STS1.RXDATNE =1, read the DAT register to clear this event. Set I2C\_CTRL1.ACKEN=0 and I2C\_CTRL1.STOPGEN=1.
6. EV9: I2C\_STS1.ADDR10F=1, reading STS1 and then writing to the DAT register will clear this event.

*Note:*

- a) If a single byte is received, it is NA.
- b) EV5, EV6, and EV9 events extend the low level of SCL until the corresponding software sequence ends.
- c) The EV7 software sequence shall be completed before the end of the current byte transmission.
- d) The software sequence of EV6\_1 or EV7\_1 shall be completed before the ACK pulse of the current transmission byte.

### 17.3.3 Error conditions description

I2C errors mainly include bus error, acknowledge error, arbitration loss, overload\ underload error. These errors may cause communication failure.

#### 17.3.3.1 Acknowledge Failure(ACKFAIL)

The interface have a acknowledge bit is detected that does not match the expectation, it will occurs acknowledge fail error, I2C\_STS1.ACKFAIL bit is set. An interrupt occurs, when I2C\_CTRL2.ERRINTEN bit is set to 1.

When transmitter receives a NACK, The communication must be reset: Device in slave mode, hardware release the bus; Device in master mode, it must generate a stop condition from software.

#### 17.3.3.2 Bus Error(BUSERR)

when address or data is transmitting,I2C interface receive external stop or start condition,it will happen a bus error, I2C\_STS1. BUSERR bit is set. An interrupt occurs, when I2C\_CTRL2.ERRINTEN bit is set to 1.

I2C device as master, the hardware does not release bus, as the same time it done not affect the current status of transfer,The current transfer will determined by software whether suspend.

I2C device as slave, when data is discarded in transmission and the bus releases by hardware, it will have two situation: If an error start condition is detected, the slave device considers a restart condition and waits for an address or a stop condition. If an error stop condition is detected, the slave device operates as a normal stop condition and the hardware releases the bus.

#### 17.3.3.3 Arbitration Lost(ARLOST)

The interface have arbitration lost is detected, hardware release the bus, it will occurs arbitration lost error, I2C\_STS1.ARLOST bit is set. An interrupt occurs, when I2C\_CTRL2.ERRINTEN bit is set to 1.

I2C interface will go to slave mode automatically(I2C\_STS2.MSMODE bit is cleared). When the I2C interface lost the arbitration, in the same communication, it can not respond to its slave address, but it can respond when master win the bus retransmits a start signal.

#### 17.3.3.4 Overrun/Underrun Error(OVERRUN)

In slave mode, disable clock extend prone to Overrun/Underrun Error:

When I2C interface is receiving data (I2C\_STS1.RXDATNE=1, data have received in register), and I2C\_DAT register still have previous byte has not been read, it will occurs an overrun error. In this situation, the last received data is discarded. And software should clear I2C\_STS1.RXDATNE bit, transmitter retransmit last byte.

When I2C interface is sending data (I2C\_STS1.TXDATE=1, new data have not sending to register), and I2C\_DAT

register still empty, it will occurs an underrun error. In this situation, the previous byte in the I2C\_DAT register is sending repeatedly. And User make sure that in the event of an underrun error, the receiver discard repeatedly byte, and transmitter should update the I2C\_DAT register at the specified time according to the I2C bus standard.

In sending the first byte, I2C\_DAT register must be written after I2C\_STS1.ADDRF bit is cleared and the before the first SCL rising edge. If cannot make sure do that, the first byte should be discard by receiver.

### 17.3.4 DMA application

DMA can generate a requests when transfer data register empty or full. DMA can oprate write data to I2C or read data from I2C reduce burden of CPU.

Before transfer current byte at the end DMA requests must be answered. If set the DMA channel transfer data is done, DMA will send EOT(End Of Transmission) to I2C, and occurs a interrupt when enable interrupt bit.

In the master transmit mode, in EOT interrupt handler DMA request need to be disbale, and set stop condition after waiting for I2C\_STS1.BSF event.

In the master receive mode, the data of received is great than or equal to 2, DMA will send a hardware signal EOT\_1 in DMA transmission(byte number-1). If set I2C\_CTRL2.DMALAST bit, when hardware have send the EOT\_1 next byte it will send a NACK automatically. The user can set a stop condition in the interrupt handler after the DMA transfer is completed if interrupt enable.

#### 17.3.4.1 Transmit process

If use the DMA mode need set the I2C\_CTRL2.DMAEN bit. When I2C\_STS1.TXDATE bit is set, the data will send to I2C\_DAT from storage area by the DMA. DMA assign a channle for I2C transmission, (x is the channel number) the following step must be opreate:

1. In the DMA\_PADDRx register set the I2C\_DAT register address. Data will be send to address in every I2C\_STS1.TXDATE event.
2. In the DMA\_MADDRx register set the memory address. Data will send to I2C\_DAT address in every I2C\_STS1.TXDATE event.
3. In the DMA\_TXNUMx register set the number of need to be transferred.In every I2C\_STS1.TXDATE event this number-1 until 0.
4. In the DMA\_CHCFGx register set PRIOLVL[1:0] bit to configure the priority of channel.
5. In the DMA\_CHCFGx register set DIR bit to configure when occurs an interrupt whether send a half data or all completed.
6. In the DMA\_CHCFGx register set CHEN bit to enable transfer channel.
7. When DMA transfer data is done, DMA need send a EOT/EOT\_1 signal to I2C indicate this transfer is done. If interrupt is enable, DMA occurs a interrupt.

*Note: if DMA is used for transmission, do not set I2C\_CTRL2.BUFINTEN bit.*

### 17.3.4.2 Receive process

If use DMA mode need set I2C\_CTRL2.DMAEN bit. When data byte is received,DMA will send I2C data to storage area, set DMA channel for I2C reception. The following steps must be opreate:

1. In DMA\_PADDRx register set the address of the I2C\_DAT register. In every I2C\_STS1.RXDATE event, data will send from address to storage area.
2. In DMA\_MADDRx register set the memory area address. In every I2C\_STS1.RXDATE event,data will send from I2C\_DAT register to storage area.
3. In DMA\_TXNUMx register set the number of need to be transferred. In every I2C\_STS1.RXDATE event the number-1 until 0.
4. In DMA\_CHCFGx register set PRIOLVL[0:1] to configure the priority of channel.
5. In DMA\_CHCFGx register clear DIR to configure when occurs a interrupt request whether received half data or all data is received.
6. In the DMA\_CHCFGx register set CHEN bit to activate the channle.
7. When DMA tansfer data is done, DMA need to send EOT/EOT\_1 signal to I2C indicate this transfer is done, if interrupt is enbale, DMA occurs a interrupt.

*Note: If DMA is used for receiving, do not set I2C\_CTRL2.BUFINTEN bit.*

### 17.3.5 Packet error check

Setting the I2C\_CTRL1.PECEN bit to 1 enables the PEC function. PEC uses CRC-8 algorithm to calculate all information bytes including address and read/write bits.it can improve the reliability of communication. The CRC-8 polynomial uses by the PEC calculator is  $C(x) = x^8 + x^2 + x + 1$ .

In transfer mode, software sets I2C\_CTRL1.PEC transfer bit in the last I2C\_STS1.TXDATE event, and then PEC will be transferred in the last byte. While in receiving mode, software sets I2C\_CTRL1.PEC transfer bit after the last I2C\_STS1.RXDATENE event, and then receives the PEC byte and compares the received PEC byte to the internally calculated PEC value. If it is not equal to the internally calculated PEC value, the receiver needs to send a NACK. If it is host receiver mode, NACK will be sent after PEC regardless of the calculated result. It should pay attention that I2C\_CTRL1.PEC bit has to be set before receiving.

If both DMA and PEC calculator are activated, I2C will automatically send or check the PEC value.

In transmit mode, when I2C interface receives EOT signal from DMA controller, it will automatically send PEC following the last byte. In receiving mode, when I2C interface receives an EOT\_1 signal from DMA, it will automatically consider the next byte as PEC and compare it with the internally calculated PEC. It will happen a DMA request after receiving PEC.

In order to allow intermediate PEC transfer, I2C\_CTRL2.DMALAST bit is used to determine whether it is the last DMA transfer. And if it does the last DMA request of the master receiver, NACK will be sent automatically after receiving the last byte.

When arbitration is lost, PEC calculation is invalid.

## 17.3.6 SMBus

### 17.3.6.1 Introduction

The System Management Bus(SMBus or SMB) is a dual-wire bus interface. Using SMBus can communicate with other device or other parts of the system, it able to commnicate with multiple devices without other independent control wire. SMBus base on I2C comminicate standard. SMBus have a control bus for system and power management related tasks. If you want browse more information, please refer to the SMBus specification v2.0(<http://smbus.org/specs/>).

SMBus have three types of device standard.

- Master: device send command,generate clocks and stop transmmissions;
- Slave: device receive,respond to commands;
- Host: system have only one host. a device provides a master to system CPU. host have function of master and slave, it support SMBus alert protocol.

SMBus is a subset of the data transmission format of the I2C specification.

Similarities between SMBus and I2C:

- Both bus protocols contain of 2 wires (a clock wire SCL and a data wire SDA), with an optional SMBus alert wire.
- The data format is similar. SMBus data format is similar to 7-bit address format of I2C(See Figure 17-2).
- Both are master-slave communication modes, and the master device provides the clock.
- Both support multi master

Differences between SMBus and I2C:

Table 17-1 Comparison between SMBus and I2C

SMBus	I <sup>2</sup> C
Maximum transmission speed 100kHz	Maximum transmission speed 1MHz
Minimum transmission speed 10kHz	No minimum transmission speed
Low clock timeout 35ms	No clock timeout
Fixed logic level	VDD determined logic level
Different address types (reserved, dynamic, etc.)	7-bit, 10-bit, and broadcast call slave address types
Different bus protocols (quick command, call handling, etc.)	No bus protocol

### 17.3.6.2 SMBus usage

SMBus uses the system management bus to meet lightweight communication requirements. In general, SMBus is commonly used on the computer motherboard. It is mainly used to transmit ON/OFF instructions for power unit and provide a control bus for system and power management-related tasks.

### 17.3.6.3 Device identification

On the SMBus, as a slave have a only address for any device, named slave address.

In order to distribute address for each devices, it must have a unique device identifier(UDID) to distinguish devices.

### 17.3.6.4 Bus protocol

SMBus specification include eight bus protocols. If want browse the details on protocols or SMBus address types,it can refer to the SMBus specification v2.0(<http://smbus.org/specs/>). User's software can device what protocols are implemented.

*Note: SMBus does not support Quick command protocol.*

Every packet through the SMBus complies with the SMBus protocol predefined format. SMBus is a subset of the data transfer format of I2C specification. As long as an I2C device can be accessed through one of the SMBus protocols, it is considered to be SMBus compliant.

### 17.3.6.5 Address resolution protocol

The SMBus resolves address conflicts by dynamically assigning a new unique address to each slave device. This is the address resolution protocol(ARP) .

Any master device can connected bus to access all devices.

SMBus physical layer arbitration enable to distribute addresses.When device power on, the device's distribute address is not change, the protocol allows address retain when device power off.

When address is distributed, there is no extra SMBus packaging cost(the cost time that access distribute address device and access fixed address device is same).

### 17.3.6.6 Timeout function

A kind of feature related to timeout on SMBus: if it has taken too long time during the communication, it automatically resets the device. This is the reason why SMBus has a minimum transmission rate limitation -- to prevent the bus from locking up for a long time after the timeout occurs. I2C bus is essentially a "DC" bus, that is to say, if the slave is executing some subroutines and cannot respond in time while the master is accessing the slave, it can hold the clock. That can remind the host that the slave is busy but does not want to give up the current communication. This session can continue after the current task of the slave is over. I2c doesn't have a maximum limitation for the delay, but it is limited to 35ms in the SMBus system. According to the SMBus protocol, if a session takes too long, it means something is wrong with the bus, and all devices should be reset to eliminate this state. Like this, the slave device is not allowed to pull the clock down for too long. I2C\_STS1.TIMOUT bit indicates the status of this feature.

### 17.3.6.7 SMBus alter mode

SMBus offer a optional interrupt signal SMBALERT(like SCL and SDA,is a wired-and signal) that devices uses to extend their control capabilities at expense of a pin. SMBus broadcast call address often combine with SMBALERT. There is 2 bytes message about SMBus.

A device which only has slave function can set I2C\_CTRL1.SMBALERT bit to indicate it want to communicate with host. The host handles the interrupt and accesses all SMBALERT devices through the ARA (Alert Response Address,

address value 0001100x). Only those devices that pull SMBALERT low can respond to ARA. This state is identified by the I2C\_STS1.SMBALERT. The 7-bit device address provided from the sending device is placed on the 7 most significant bits of the byte, the eighth bit can be either '0' or '1'.

When more than one device's SMBALERT is low, the highest priority(The smaller the address, the higher the priority) can win bus communication through the standard arbitration during address transmission. If confirming the slave address, device's SMBALERT is no longer pulled low. If message transmitted completely,device's SMBALERT still is low,it mean host will read ARA again. The host can periodically access the ARA when the SMBALERT signal is not used.

### 17.3.6.8 SMBus communication process

The communication process on SMBus is similar to that on I2C.To use the SMBus mode, you need to configure SMBus specific registers in the program, respond and process SMBus specific flag, to implement the upper-layer protocols described in the SMBus manual.

- 1.At first, set I2C\_CTRL1.SMBMODE bit, and configure I2C\_CTRL1.SMBTYPE bit and I2C\_CTRL1.AR PEN bit according to the application requirements. If I2C\_CTRL1.AR PEN=1 and I2C\_CTRL1.SMBTYPE=0, use the default address of the SMB device. If I2C\_CTRL1.AR PEN=1 and I2C\_CTRL1.SMBTYPE=1, use the SMB master header field.
- 2.In order to support ARP (I2C\_CTRL1.AR PEN=1), in SMBus host mode (I2C\_CTRL1.SMBTYPE=1), software needs to respond to the I2C\_STS2.SMBHADDR bit (in SMBus slave mode, respond to I2C\_STS2.SMBDADDR bit) and implement the functions according to the ARP protocol.
- 3.To support the SMBus warning mode, software should respond to the I2C\_STS1.SMBALERT bit and implement the corresponding functions.

## 17.4 Debug mode

When the microcontroller enters the debug mode (Cortex®-M0 core is in the stop state), configure the DBG\_CTRL.I2CxTIMOUT bit in the PWR module, Select SMBUS timeout to continue normal work or stop. See section 3.3.2 for details.

## 17.5 Interrupt request

All I2C interrupt requests are listed in the following table.

Table 17-2 I<sup>2</sup>C interrupt request

Interrupt function	Interrupt event	Event flag	Set control bit
I2C global interrupt	Start bit sent (master)	STARTBF	EVTINTEN
	Address sent (master) or address matched (slave)	ADDRF	
	10-bit header sent (master)	ADDR10F	
	Received stop (slave)	STOPF	
	Data byte transfer completed.	BSF	
	Receive buffer is not empty.	RXDATNE	EVTINTEN and BUFINTEN

Interrupt function	Interrupt event	Event flag	Set control bit
	Send buffer is empty.	TXDATE	ERRINTEN
	Bus error	BUSERR	
	Lost arbitration (master)	ARLOST	
	Acknowledge fail	ACKFAIL	
	Overrun/underrun	OVERRUN	
	PEC error	PECERR	
	Timeout /Tlow error	TIMOUT	
	SMBus Alert	SMBALERT	

*Note:* 1. *STARTBF, ADDR1, ADDR10F, STOPF, BSF, RXDATNE and TXDATE* are merged into a interrupt channel through logical OR.

2. *BUSERR, ARLOST, ACKFAIL, OVERRUN, PECERR, TIMEOUT and SMBALERT* are merged into a interrupt channel through logical OR.

3. Event interrupts and error interrupts are logically ORed into the global interrupt channel.

## 17.6 I2C registers

These peripheral registers can be operated by half word (16 bits) or word (32 bits)

### 17.6.1 I2C register overview

Table 17-3 I2C register overview

### 17.6.2 I2C Control register 1 (I2C\_CTRL1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SW RESET	Reserved	SMB ALERT	PEC	ACK POS	ACKEN	STOP GEN	START GEN	NO EXTEND	GCEN	PECEN	ARPEN	SMB TYPE	Reserved	SMB MODE	EN
RW		RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW		RW	RW

Bit field	Name	Description
15	SWRESET	<p>Software reset Make sure the I2C bus is idle before resetting this bit. 0:I2C not reset; 1:I2C reset.</p> <p><i>Note: This bit can be used when the I2C_STS2.BUSY bit is set to 1 and no stop condition is detected on the bus.</i></p>
14	Reserved	Reserved, the reset value must be maintained
13	SMBALERT	<p>SMBus alert It can be set or cleared by software. When I2C_CTRL1.EN=0, it will be cleared by hardware. 0: SMBAlert pin go high. The response address header is followed by the NACK signal; 1: SMBAlert pin go low. The response address header is followed by the ACK signal.</p>
12	PEC	<p>Packet error checking It can be set or cleared by software. It will be cleared by hardware when PEC has been transferred, or by start or stop condition, or when I2C_CTRL1.EN=0. 0: No PEC transfer 1: PEC transfer.</p> <p><i>Note: When arbitration is lost, the calculation of PEC is invalid.</i></p>
11	ACKPOS	<p>Acknowledge/PEC Position (for data reception) It can be set or cleared by software. Or when I2C_CTRL1.EN=0, it will be cleared by hardware. 0: I2C_CTRL1.ACKEN bit determines whether to send an ACK to the byte currently being received; I2C_CTRL1.PEC bit indicates that the byte in the current shift register is PEC.</p>

Bit field	Name	Description
		<p>1: I2C_CTRL1.ACKEN bit determines whether to send an ACK to the next received byte;  I2C_CTRL1.PEC bit indicates that the next byte received in the shift register is PEC.</p> <p><i>Note:</i>  <i>ACKPOS bit can only be used in 2-byte receiving configuration and must be configured before receiving data.</i>  <i>For the second byte of NACK, the I2C_CTRL1.ACKEN bit must be cleared after the I2C_STS1.ADDRF bit is cleared.</i>  <i>To detect the PEC of the second byte, the I2C_CTRL1.PEC bit must be set after the ACKPOS bit is configured and when the ADDR event is extended.</i></p>
10	ACKEN	<p>Acknowledge enable  It can be set or cleared by software. Or when I2C_CTRL1.EN equals to 0, it will be cleared by hardware.  0: No acknowledge send;  1: Send an acknowledge after receiving a byte</p>
9	STOPGEN	<p>Stop generation  It can be set or cleared by software. Or it will be cleared by hardware when a stop condition is detected. Or it will be set by hardware when SMBus timeout error is detected.,  In the master mode:  0: No stop condition generates;  1: Generate a stop condition.  In the slave mode:  0: No stop condition generates;  1: Release SCL and SDA lines after the current byte.</p> <p><i>Note: When the STOPGEN, STARTGEN or PEC bit is set, the software should not take any write operation to I2C_CTRL1 until this bit is cleared by hardware. Otherwise, the STOPGEN, STARTGEN or PEC bits may be set twice.</i></p>
8	STARTGEN	<p>Start generation  It can be set or cleared by software. Or it will be cleared by hardware when the start condition is transferred or I2C_CTRL1.EN=0.  0: No start condition generates;  1: Generate a start conditions.</p>
7	NOEXTEND	<p>Clock extending disable (Slave mode)  This bit determines whether to pull SCL low when the data is not ready(I2C_STS1.ADDRF or I2C_STS1.BSF flag is set) in slave mode, and is cleared by software reset  0: Enable Clock extending.  1: Disable Clock extending.</p>
6	GCEN	<p>General call enable  0: Disable General call. not respond(NACK) to the address 00h;  1: Enable General call. respond(ACK) the address 00h.</p>
5	PECEN	<p>PEC enable  0: Disable PEC module;</p>

Bit field	Name	Description
		1: Enable PEC module.
4	ARPEN	ARP enable 0: Disable ARP; 1: Enable ARP. If I2C_CTRL1.SMBTYPE=0, the default address of SMBus device is used. If I2C_CTRL1.SMBTYPE=1, the host address of SMBus is used.
3	SMBTYPE	SMBus type 0: Device 1: Host
2	Reserved	Reserved, the reset value must be maintained.
1	SMBMODE	SMBus mode 0: I2C mode; 1: SMBus mode.
0	EN	I2C Peripheral enable 0: Disable I2C module; 1: Enable I2C module <i>Note: If this bit is cleared when communication is in progress, the I2C module is disabled and returns to the idle state after the current communication ends, all bits will be cleared.</i> <i>In master mode, this bit must never be cleared until the communication has ended.</i>

### 17.6.3 I2C Control register 2 (I2C\_CTRL2)

Address offset: 0x04

Reset value: 0x0000

15	13	12	11	10	9	8	7	6	5	0
Reserved	DMA LAST	DMA EN	BUFINTEEN	EVTINT EN	ERRINT EN	Reserved			CLKFREQ[5:0]	rw

Bit field	Name	Description
15:13	Reserved	Reserved, the reset value must be maintained.
12	DMALAST	DMA last transfer 0: Next DMA EOT is not the last transfer 1: Next DMA EOT is the last transfer <i>Note: This bit is used in the master receiving mode, so that a NACK can be generated when data is received for the last time.</i>
11	DMAEN	DMA requests enable 0: Disable DMA 1: Enable DMA
10	BUFINTEEN	Buffer interrupt enable 0: When I2C_STS1.TXDATE=1 or I2C_STS1.RXDATNE=1, any interrupt is not generated.

Bit field	Name	Description
		1: If I2C_CTRL2.EVTINTEN= 1, When I2C_STS1.TXDATE=1 or I2C_STS1.RXDATNE=1, interrupt will be generated.
9	EVTINTEN	<p>Event interrupt enable</p> <p>0: Disable event interrupt;</p> <p>1: Enable event interrupt</p> <p>This interrupt is generated when:</p> <ul style="list-style-type: none"> <li>I2C_STS1.STARTBF = 1 (Master)</li> <li>I2C_STS1.ADDR F = 1 (Master/Slave)</li> <li>I2C_STS1.ADD10F = 1 (Master)</li> <li>I2C_STS1.STOPF = 1 (Slave)</li> <li>I2C_STS1.BSF = 1 with no I2C_STS1.TXDATE or I2C_STS1.RXDATNE event</li> <li>I2C_STS1.TXDATE = 1 if I2C_CTRL2.BUFINTEN = 1</li> <li>I2C_STS1.RXDATNE = 1 if I2C_CTRL2.BUFINTEN = 1</li> </ul>
8	ERRINTEN	<p>Error interrupt enable</p> <p>0: Disable error interrupt;</p> <p>1: Enable error interrupt.</p> <p>This interrupt is generated when:</p> <ul style="list-style-type: none"> <li>I2C_STS1.BUSERR = 1;</li> <li>I2C_STS1.ARLOST = 1;</li> <li>I2C_STS1.ACKFAIL = 1;</li> <li>I2C_STS1.OVERRUN = 1;</li> <li>I2C_STS1.PECERR = 1;</li> <li>I2C_STS1.TIMOUT = 1;</li> <li>I2C_STS1.SMBALERT = 1.</li> </ul>
7:6	Reserved	Reserved, the reset value must be maintained.
5:0	CLKFREQ[5:0]	<p>I2C Peripheral clock frequency</p> <p>CLKFREQ[5:0] should be the APB1 clock frequency to generate the correct timing.</p> <ul style="list-style-type: none"> <li>00000: Disable</li> <li>00001: Disable</li> <li>00010: 2MHz</li> <li>00011: 3MHz</li> <li>...</li> <li>110000: 48MHz</li> <li>110001~111111: Disable.</li> </ul>

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

Address offset: 0x08

Reset value: 0x0000

15	14	13		10	9	8	7		1	0
ADDR MODE	Reserved		Reserved		ADDR[9:8]			ADDR[7:1]		ADDR0
rw				rw				rw		rw

Bit field	Name	Description
15	ADDRMODE	Addressing mode (slave mode) 0: 7-bit slave address 1: 10-bit slave address
14	Reserved	Must always be kept as '1' by the software.
13:10	Reserved	Reserved, the reset value must be maintained.
9:8	ADDR[9:8]	Interface address 9~8 bits of the address. <i>Note: don't care these bits in 7-bit address mode</i>
7:1	ADDR[7:1]	Interface address 7~1 bits of the address.
0	ADDR0	Interface address 0 bit of the address. <i>Note: don't care these bits in 7-bit address mode</i>

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

Address offset: 0x0C

Reset value: 0x0000

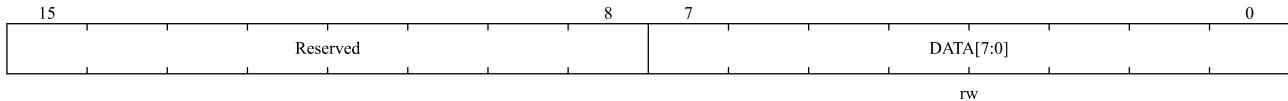
15		8	7		1	0
	Reserved			ADDR2[7:1]		
				rw		rw

Bit field	Name	Description
15:8	Reserved	Reserved, the reset value must be maintained.
7:1	ADDR2[7:1]	Interface address 7~1 bits of address in dual address mode.
0	DUALEN	Dual addressing mode enable 0: Disable dual address mode, only OADDR1 is recognized; 1: Enable dual address mode, both OADDR1 and OADDR2 are recognized. <i>Note: Valid only for 7-bit address mode</i>

## 17.6.6 I2C Data register (I2C\_DAT)

Address offset: 0x10

Reset value: 0x0000

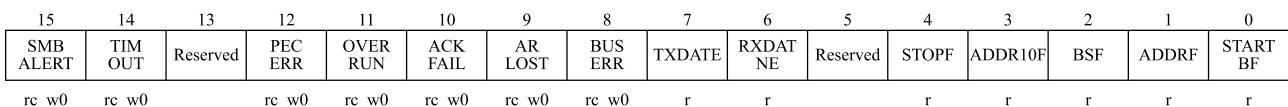


Bit field	Name	Description
15:8	Reserved	Reserved, the reset value must be maintained.
7:0	DATA[7:0]	8-bit data register Send or receive data buffer. <i>Note: In the slave mode, the address will not be copied into the data register;</i> <i>Note: if I2C_STS1.TXDATE =0, data can still be written into the data register;</i> <i>Note: If the ARLOST event occurs when processing the ACK pulse, the received byte will not be copied into the data register, so it cannot be read.</i>

### 17.6.7 I2C Status register 1 (I2C\_STS1)

Address offset: 0x14

Reset value: 0x0000



Bit field	Name	Description
15	SMBALERT	SMBus alert Writing ‘0’ to this bit by software can clear it, or it is cleared by hardware when I2C_CTRL1.EN=0. 0: No SMBus alert(host mode) or no SMB alert response address header sequence(slave mode); 1: SMBus alert event is generated on the pin(host mode) or receive SMBAlert response address(slave mode)
14	TIMOUT	Timeout or Tlow error Writing ‘0’ to this bit by software can clear it, or it is cleared by hardware when I2C_CTRL1.EN=0. 0: No Timeout error; 1: A timeout error occurred Error in the following cases: <ul style="list-style-type: none"> <li>■ SCL has kept low for 25ms (Timeout).</li> <li>■ Master cumulative clock low extend time more than 10 ms (Tlow:mext).</li> <li>■ Slave cumulative clock low extend time more than 25 ms (Tlow:sext).</li> </ul> Timeout in slave mode: slave device resets the communication and hardware frees the bus. Timeout in master mode: hardware sends the stop condition.
13	Reserved	Reserved, the reset value must be maintained.
12	PECERR	PEC Error in reception Writing ‘0’ to this bit by software can clear it, or it is cleared by hardware when I2C_CTRL1.EN=0.

Bit field	Name	Description
		0: No PEC error 1: PEC error: receiver will returns NACK Whether the I2C_CTRL1.ACKEN bit is enabled
11	OVERRUN	Overrun/Underrun Writing '0' to this bit by software can clear it, or it is cleared by hardware when I2C_CTRL1.EN=0. 0: No Overrun/Underrun 1: Overrun/Underrun Set by hardware in slave mode when I2C_CTRL1.NOEXTEND=1, and when receiving a new byte in receiving mode, if the data within DAT register has not been read yet, over-run occurs, the new received byte will be lost. When transferring a new byte in transfer mode, but there is not new data that has not been written in DAT register, under-run occurs which leads that the same byte will be send twice.
10	ACKFAIL	Acknowledge failure Writing '0' to this bit by software can clear it, or it is cleared by hardware when I2C_CTRL1.EN=0. 0: No acknowledge failed; 1: Acknowledge failed.
9	ARLOST	Arbitration lost (master mode) Writing '0' to this bit by software can clear it, or it is cleared by hardware when I2C_CTRL1.EN=0. 0: No arbitration lost; 1: Arbitration lost. When the interface loses control of the bus to another host, the hardware will set this bit to '1', and the I2C interface will automatically switch back to slave mode (I2C_STS2.MSMODE=0). <i>Note: In SMBUS mode, the arbitration of data in slave mode only occurs in the data stage or the acknowledge transfer interval (excluding the address acknowledge).</i>
8	BUSERR	Bus error Writing '0' to this bit by software can clear it, or it is cleared by hardware when I2C_CTRL1.EN=0. 0: No start or stop condition error 1: Start or stop condition error
7	TXDATE	Data register empty (transmitters) Writing data to DAT register by software can clear this bit; Or after a start or stop condition occurs, or automatically cleared by hardware when I2C_CTRL1.EN=0. 0: Data register is not empty; 1: Data register is empty. When sending data, this bit is set to '1' when the data register is empty, and it is not set at the address sending stage. If a NACK is received, or the next byte to be sent is PEC(I2C_CTRL1.PEC=1), this bit will not be set.

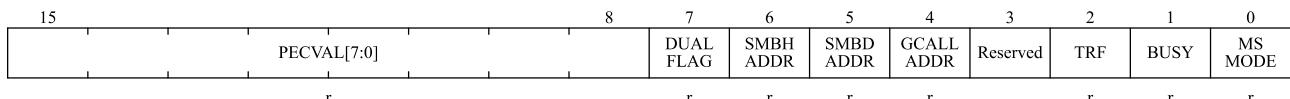
Bit field	Name	Description
		<p><i>Note: After the first data to be sent is written, or data is written when BSF is set, the TXDATE bit cannot be cleared, because the data register is still empty.</i></p>
6	RXDATNE	<p>Data register not empty(receivers)</p> <p>This bit is cleared by software reading and writing to the data register, or cleared by hardware when I2C_CTRL1.EN=0.</p> <p>0: Data register is empty; 1: Data register is not empty.</p> <p>During receiving data, this bit is set to' 1' when the data register is not empty, and it is not set at the address receiving stage.</p> <p>RXDATNE is not set when the ARLOST event occurs.</p> <p><i>Note: When BSF is set, the RXDATNE bit cannot be cleared when reading data, because the data register is still full.</i></p>
5	Reserved	Reserved, the reset value must be maintained.
4	STOPF	<p>Stop detection (slave mode)</p> <p>After the software reads the STS1 register, the operation of writing to the CTRL1 register will clear this bit, or when I2C_CTRL1.EN=0, the hardware will clear this bit.</p> <p>0: No stop condition is detected; 1: Stop condition is detected.</p> <p>After a ACK, the hardware sets this bit to' 1' when the slave device detects a stop condition on the bus.</p> <p><i>Note: I2C_STS1.STOPF bit is not set after receiving NACK.</i></p>
3	ADDR10F	<p>10-bit header sent (Master mode)</p> <p>After the software reads the STS1 register, the operation of writing to the CTRL1 register will clear this bit, or when I2C_CTRL1.EN=0, the hardware will clear this bit.</p> <p>0: No ADDR10F event; 1: Received has sent the first address byte.</p> <p>In 10-bit address mode, when the master device has sent the first byte, the hardware sets this bit to' 1'.</p> <p><i>Note: After receiving a NACK, the I2C_STS1.ADDR10F bit is not set.</i></p>
2	BSF	<p>Byte transfer finished</p> <p>After the software reads the STS1 register, reading or writing the data register will clear this bit; Or after sending a start or stop condition in sending mode, or when I2C_CTRL1.EN=0, this bit is cleared by hardware.</p> <p>0: Byte transfer does not finish. 1: Byte transfer finished.</p> <p>When I2C_CTRL1.NOEXTEND =0, the hardware sets this bit to' 1' in the following cases: In receiving mode, when a new byte (including ACK pulse) is received and the data register has not been read (I2C_STS1.RXDATNE=1). In sending mode, when a new data is to be transmitted and the data register has not been written with the new data (I2C_STS1.TXDATE=1).</p> <p><i>Note: After receiving a NACK, the BSF bit will not be set.</i></p>

Bit field	Name	Description
		<p>If the next byte to be transferred is PEC (I2C_STS2.TRF is '1' and I2C_CTRL1.PEC is '1'), the BSF bit will not be set.</p>
1	ADDRF	<p>Address sent (master mode) / matched (slave mode)</p> <p>After the STS1 register is read by software, reading the STS2 register will clear this bit, or when I2C_CTRL1.EN=0, it will be cleared by hardware.</p> <p>0: Address mismatch or no address received(slave mode) or Address sending did not end(master mode);</p> <p>1: Received addresses matched(slave mode) or Address sending ends(master mode)</p> <p>In master mode:</p> <p>In 7-bit address mode, this bit is set to '1' after receiving the ACK of the address.In 10-bit address mode, this bit is set to '1' after receiving the ACK of the second byte of the address.</p> <p>In slave mode:</p> <p>Hardware sets this bit to '1' (when the corresponding setting is enabled) when the received slave address matches the content in the OADDR register, or a general call or SMBus device default address or SMBus host or SMBus alter is recognized.</p> <p><i>Note: After receiving NACK, the I2C_STS1.ADDRF bit will not be set.</i></p>
0	STARTBF	<p>Start bit (Master mode)</p> <p>After the STS1 register is read by software, writing to the data register will clear this bit, or when I2C_CTRL1.EN=0, the hardware will clear this bit.</p> <p>0: Start condition was not sent;</p> <p>1: Start condition has been sent.</p> <p>This bit is set to '1' when the start condition is sent.</p>

### 17.6.8 I2C Status register 2 (I2C\_STS2)

Address offset: 0x18

Reset value: 0x0000



Bit field	Name	Description
15:8	PECVAL[7:0]	<p>Packet error checking register</p> <p>Stores the internal PEC value When I2C_CTRL1.PECEN =1.</p>
7	DUALFLAG	<p>Dual flag(Slave mode)</p> <p>Hardware clears this bit when a stop condition or a repeated start condition is generated, or when I2C_CTRL1.EN=0.</p> <p>0: Received address matches the content in OADDR1;</p> <p>1: Received address matches the content in OADDR2.</p>
6	SMBHADDR	<p>SMBus host header□Slave mode□</p> <p>Hardware clears this bit when a stop condition or a repeated start condition is generated, or</p>

Bit field	Name	Description
		when I2C_CTRL1.EN=0. 0: SMBus host address was not received; 1: when I2C_CTRL1.SMBTYPE=1 and I2C_CTRL1.AR PEN=1, the SMBus host address is received.
5	SMBDADDR	SMBus device default address□Slave mode□ Hardware clears this bit when a stop condition or a repeated start condition is generated, or when I2C_CTRL1.EN=0. 0: The default address of SMBus device has not been received; 1: when I2C_CTRL1.AR PEN=1, the default address of SMBus device is received.
4	GCALLADDR	General call address(Slave mode) Hardware clears this bit when a stop condition or a repeated start condition is generated, or when I2C_CTRL1.EN=0. 0: No general call address was received; 1: when I2C_CTRL1.GCEN=1, general call address was received.
3	Reserved	Reserved, the reset value must be maintained.
2	TRF	Transmitter/receiver After detecting the stop condition (I2C_STS1.STOPF=1), repeated start condition or bus arbitration loss (I2C_STS1.ARLOST=1), or when I2C_CTRL1.EN=0, the hardware clears it. 0: Data receiving mode 1: Data transmission mode; At the end of the whole address transmission stage, this bit is set according to the R/W bit of the address byte.
1	BUSY	Bus busy Hardware clears this bit when a stop condition is detected. 0: No data communication on the bus; 1: Data communication on the bus. When detecting that SDA or SCL is low level, the hardware sets this bit to '1'; <i>Note: This bit indicates the bus communication currently in progress, and this information is still updated when the interface is disabled (I2C_CTRL1.EN=0).</i>
0	MSMODE	Master/slave mode Hardware clears this bit when a stop condition is detected on the bus, arbitration is lost (I2C_STS1.ARLOST=1), or when I2C_CTRL1.EN=0. 0: In slave mode; 1: In master mode. When the interface is in the master mode (I2C_STS1.STARTBF=1), the hardware sets this bit;

### 17.6.9 I2C Clock control register (I2C\_CLKCTRL)

Address offset: 0x1c

Reset value: 0x0000

*Note: 1. F\_PCLKI is required to be an integer multiple of 10 MHz, so that a fast clock of 400KHz can be generated*

correctly.

2. The CLKCTRL register can only be set when I<sup>2</sup>C is turned off (I2C\_CTRL1.EN=0)

FSMODE	DUTY	Reserved	CLKCTRL[11:0]	
RW	RW			RW

Bit field	Name	Description
15:14	DUTY	<p>SCL duty cycle</p> <p>00: Tlow/Thigh = 1 □</p> <p>01: Tlow/Thigh = 1 □</p> <p>10: Tlow/Thigh = 2 □</p> <p>11: Tlow/Thigh = 16/9 □</p> <p><i>Note: 00 or 01 configuration is recommended for SCL 100K or 1M. 10 or 11 configuration recommended for SCL 400K.</i></p>
13:12	Reserved	Reserved, the reset value must be maintained.
11:0	CLKCTRL[11:0]	<p>Clock control register in Fast/Standard mode (Master mode)</p> <p>This division factor is used to set the SCL clock in the master mode.</p> <ul style="list-style-type: none"> <li>■ If duty cycle = Tlow/Thigh = 1/1:  <math>CLKCTRL = f_{PCLK1}(\text{Hz})/100000/2</math>  <math>T_{low} = CLKCTRL \times T_{PCLK1}</math>  <math>T_{high} = CLKCTRL \times T_{PCLK1}</math></li> <li>■ If duty cycle = Tlow/Thigh = 2/1:  <math>CLKCTRL = f_{PCLK1}(\text{Hz})/100000/3</math>  <math>T_{low} = 2 \times CLKCTRL \times T_{PCLK1}</math>  <math>T_{high} = CLKCTRL \times T_{PCLK1}</math></li> <li>■ If duty cycle = Tlow/Thigh = 16/9:  <math>CLKCTRL = f_{PCLK1}(\text{Hz})/100000/25</math>  <math>T_{low} = 16 \times CLKCTRL \times T_{PCLK1}</math>  <math>T_{high} = 9 \times CLKCTRL \times T_{PCLK1}</math></li> </ul> <p>For example, if <math>f_{PCLK1}(\text{Hz}) = 8\text{MHz}</math>, duty cycle = 1/1, <math>CLKCTRL = 8000000/100000/2 = 0x28</math>.</p> <p><i>Note: 1. The minimum setting value is 0x04 in standard mode and 0x01 in fast mode; 2. <math>T_{high} = T_{f(SCL)} + T_{w(SCL)}</math>. See the definitions of these parameters in the data sheet for details. 3. <math>T_{low} = T_{f(SCL)} + T_{w(SCL)}</math>, see the definitions of these parameters in the data sheet for details; 4. These delays have no filters;</i></p>

### 17.6.10 I<sup>2</sup>C Rise time register (I2C\_TMRISE)

Address offset: 0x20

Reset value: 0x0002

15	Reserved	6	5	0
				RW

Bit field	Name	Description
15:6	Reserved	Reserved, the reset value must be maintained.
5:0	TMRISE[5:0]	<p>Maximum rise time in fast/standard mode (master mode).</p> <p>These bits must be set to the maximum SCL rising time given in the I2C bus specification, and incremented step is 1.</p> <p>For example, the maximum allowable SCL rise time in standard mode is 1000ns. if the value in I2C_CTRL2.CLKFREQ [5:0] is equal to 0x08(8MHz) and <math>T_{PCLK1}=125\text{ns}</math>, <math>09h(1000\text{ns}/125\text{ ns} + 1)</math> must be written in TMRISE[5:0].</p> <p>If the result is not an integer, write the integer part to TMRISE[5:0] to ensure the <math>t_{HIGH}</math> parameter.</p> <p><i>Note: TMRISE[5:0] can only be set when I2C is disabled (EN=0).</i></p>

## 18 Universal synchronous asynchronous receiver transmitter (USART)

### 18.1 Introduction

USART is a full-duplex universal synchronous/asynchronous serial transceiver module. This interface is a highly flexible serial communication device that can perform full-duplex data exchange with external devices.

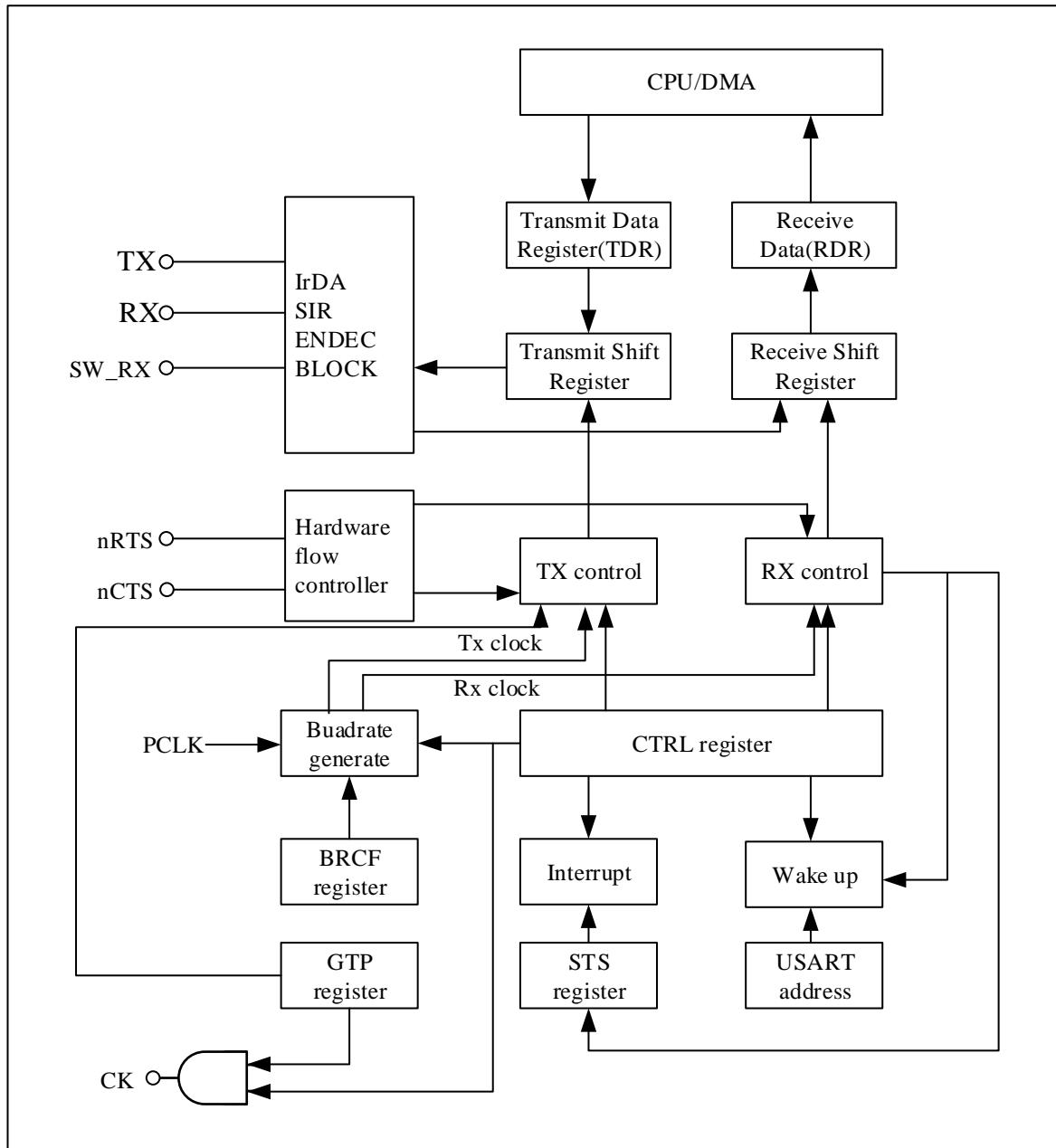
The USART has programmable transmit and receive baud rates and can communicate continuously using DMA. It also supports multiprocessor communication, LIN mode, synchronous mode, single-wire half-duplex communication, smart card asynchronous protocol, IrDA SIR ENDEC function, and hardware flow control function.

### 18.2 Main features

- Full-duplex operation
- Single-wire half-duplex operation
- Baud rate generator, the highest baud rate can reach 3Mbit/s
- Support serial data frame structure with 8 or 9 data bits, 1 or 2 stop bits
- Generation and checking of supported parity bits
- Support hardware flow control: RTS flow control and CTS flow control
- Support DMA receiving and sending
- Support multi-processor communication mode, can enter mute mode, wake up by idle detection or address mark detection
- Synchronous mode, allowing users to control bidirectional synchronous serial communication in master mode
- Comply with ISO7816-3 standard, support smart card asynchronous protocol
- IrDA SIR ENDEC function: IrDA normal mode and IrDA low power mode
- LIN (Local Area Network) mode
- Support data overflow error detection, frame error detection, noise error detection, parity error detection
- Interrupt requests include: transmit data register empty, CTS flag, transmit complete, receive data ready to read, data overflow detected, idle line detected, parity error, LIN break frame detection, noise flag/overflow error/frame error in multi-buffer communication

## 18.3 Functional block diagram

Figure 18-1 USART block diagram



## 18.4 Function description

As shown in the Figure 18-1, the bidirectional communication of any USART needs to use the RX and TX pins of the external connection. Among them, TX is the output pin for serial data transmission. When the transmitter is active

and not sending data, the TX pin is pulled high. When the transmitter is inactive, the TX pin reverts to the I/O port configuration. RX is an input pin for serial data reception, data is recovered by oversampling technique.

The data packets of serial communication are transmitted from the sending device to the RX interface of the receiving device through its own TX interface, and the bus is in an idle state before sending or receiving. Frame format is: 1 start bit + 8 or 9 data bits (least significant bit first) + 1 parity bit (optional) + 0.5,1,1.5 or 2 stop bit.

Use the fractional baud rate generator to configure transmit and receive baud rates .

According to the block diagram, when using the hardware flow control mode, the nRTS output and nCTS input pins are required. When the USART receiver is ready to receive new data, nRTS becomes low level. If nCTS is valid (pulled to a low level), the next data is sent, otherwise the next frame of data is not sent.

When using synchronous mode, the CK pin is required. The CK pin is used for clock output for synchronous transfers. Clock phase and polarity are software programmable. During the start and stop bits, the CK pin does not output clock pulses.The CK pin is also used when using smart card mode.

### 18.4.1 USART frame format

The start bit of the data frame is low.

The word length can be selected as 8 or 9 bits by programming the USART\_CTRL1.WL bits, least significant bit first.

The stop bit of the data frame is high.

An idle frame is a complete data frame consisting of '1's, including the start bit. followed by the start bit of a data frame containing the data .

A break frame is a complete data frame consisting of '0's, including the stop bit. at the end of the break frame, the transmitter inserts 1 or 2 more stop bits ('1') to acknowledge the start bit.

Figure 18-2 word length = 8 setting

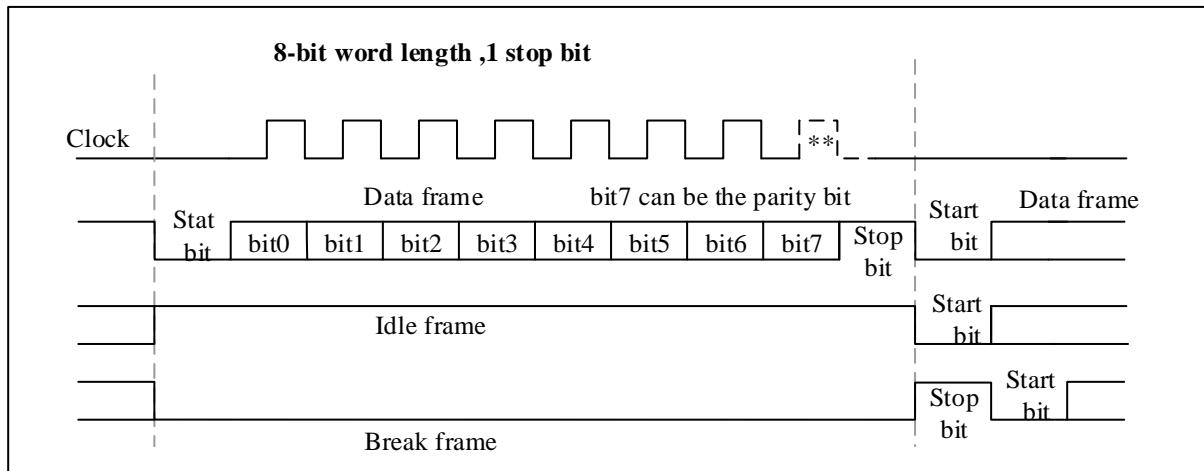
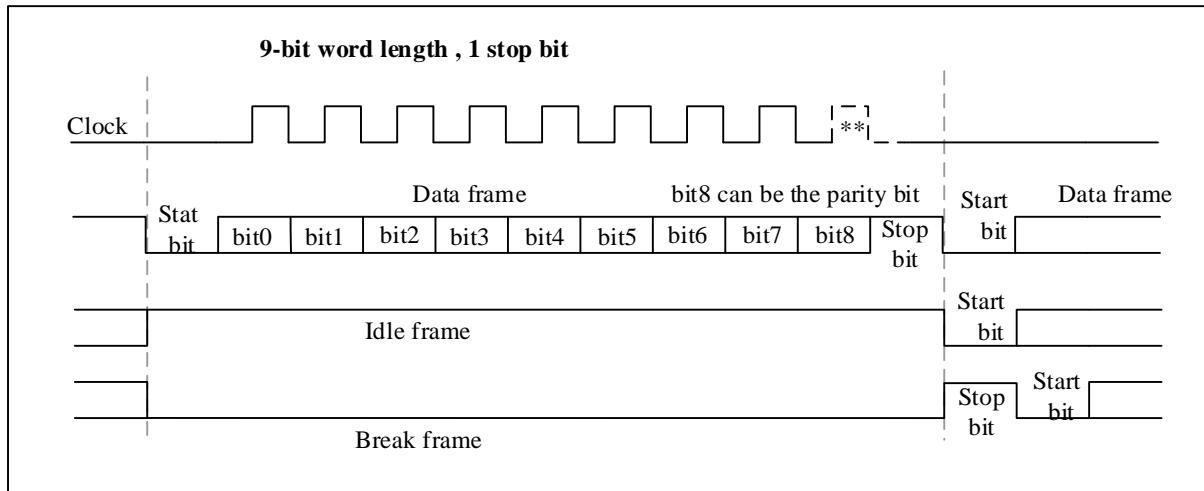


Figure 18-3 word length = 9 setting



## 18.4.2 Transmitter

After the transmitter is enabled, the data entered into the transmit shift register is sent out through the TX pin.

### 18.4.2.1 Idle frame

Setting USART\_CTRL1.TXEN will cause the USART to transmit an idle frame before the first data frame.

### 18.4.2.2 Character send

Idle frames are followed by characters sent. Each character is preceded by a low start bit. The transmitter sends 8-bit or 9-bit data according to the configuration of the data bit length, with the least significant bit first. If USART\_CTRL1.TXEN is reset during a data transfer, it will cause the baud rate counter to stop counting and the data being transferred will be corrupted.

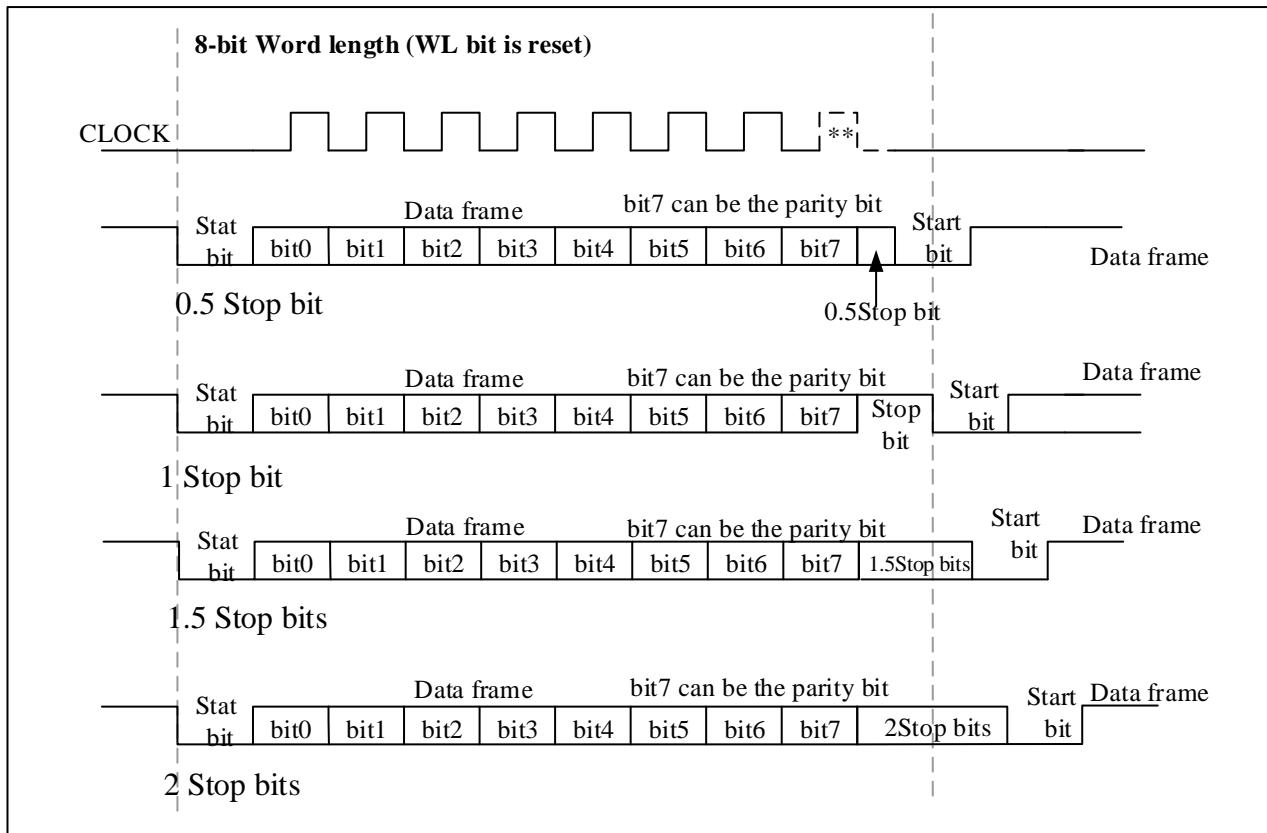
### 18.4.2.3 Stop bit

The characters are followed by stop bits, the number of which can be configured by setting USART\_CTRL2.STPB[1:0].

Table 18-1 Stop bit configuration

USART_CTRL2.STPB[1:0]	Stop bit length (bits)	functional description
00	1	default
01	0.5	Receiving in Smartcard mode
10	2	General USART mode, single-wire mode and modem mode.
11	1.5	Transmitting and receiving in Smartcard mode

Figure 18-4 configuration stop bit



### 18.4.2.4 Break frame

Use USART\_CTRL1.SDBRK to send the break character. When there is 8-bit data, the break frame consists of 10 bits of low level, followed by a stop bit; when there is 9-bit data, the break frame consists of 11 bits of low level,

followed by a stop bit.

After the break frame is sent, USART\_CTRL1.SDBRK is cleared by hardware, and the stop bit of the break frame is being sent. Therefore, to send a second break frame, USART\_CTRL1.SDBRK should be set after the stop bit of the previous break frame has been sent.

If software resets the USART\_CTRL1.SDBRK bit before starting to send the break frame, the break frame will not be sent.

#### 18.4.2.5 Transmitter process

1. Enable USART\_CTRL1.UEN to activate USART;
2. Configure the transmitter's baud rate, data bit length, parity bit (optional), the number of stop bits or DMA configuration;
3. Activate the transmitter (USART\_CTRL1.TXEN);
4. Send each data to be sent to the USART\_DAT register through the CPU or DMA, and the write operation to the USART\_DAT register will clear USART\_STS.TXDE;
5. After writing the last data word in the USART\_DAT register, wait for USART\_STS.TXC =1, which indicates the end of the transmission of the last data frame.

#### 18.4.2.6 Single byte communication

A write to the USART\_DAT register clears the USART\_STS.TXDE bit.

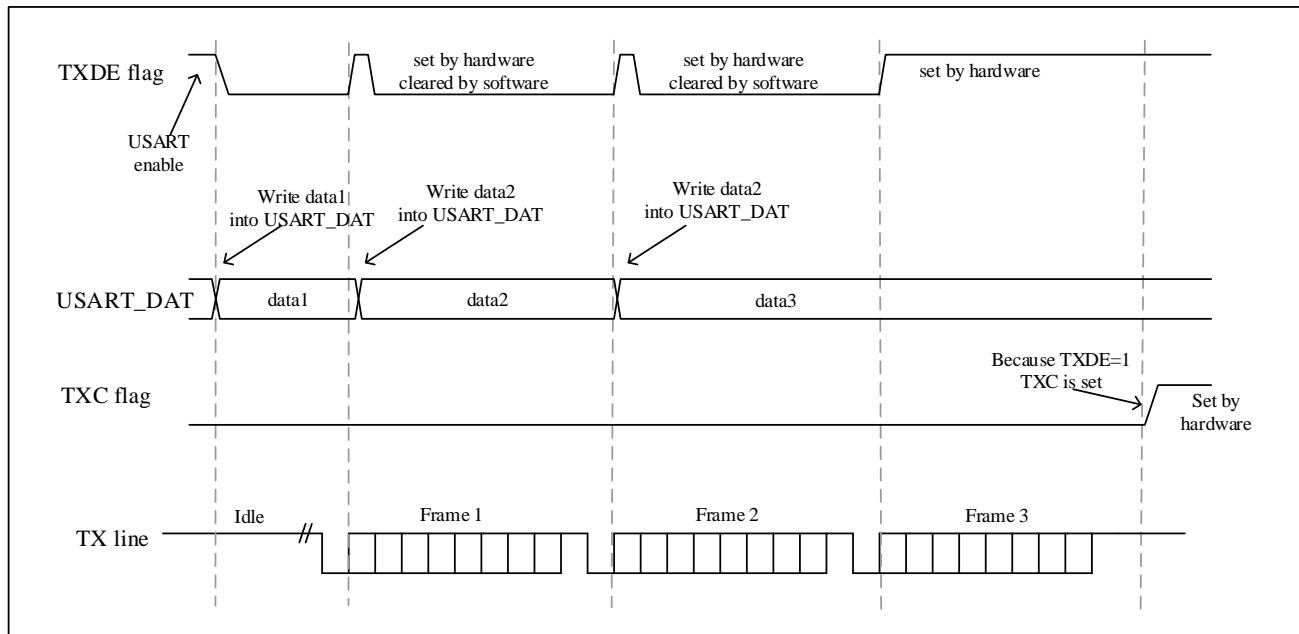
The USART\_STS.TXDE bit is set by hardware when the data in the TDR register is transferred to the transmit shift register (indicating that data is being transmitted). An interrupt will be generated if USART\_CTRL1.TXDEIEN is set. At this point, the next data can be sent to the USART\_DAT register because the TDR register has been cleared and will not overwrite the previous data.

Write operation to USART\_DAT register:

- When the transmit shift register is not sending data and is in an idle state, the data is directly put into the shift register for transmission, and the USART\_STS.TXDE bit is set by hardware;
- When the transmit shift register is sending data, the data is stored in the TDR register, and after the current transmission is completed, the data is put into the shift register.

When a frame containing data is sent and USART\_STS.TXDE=1, the USART\_STS.TXC bit is set to '1' by hardware. An interrupt is generated if USART\_CTRL1.TXCIEN is '1'. USART\_STS.TXC bit is cleared by a software sequence (read USART\_STS register first, then write USART\_DAT register).

Figure 18-5 TXC/TXDE changes during transmission



## 18.4.3 Receiver

### 18.4.3.1 Start bit detection

When the received sampling sequence is: 1 1 1 0 X 0 X 0 X 0 0 0 0, it is considered that a start bit is detected.

The samples at the 3rd, 5th, and 7th bits, and the samples at the 8th, 9th, and 10th bits are all '0' (that is, 6 '0'), then confirm the receipt of the start bit, the USART\_STS.RXDNE flag bit is set, and if USART\_CTRL1.RXDNEIEN=1, an interruption occurs and will not Set the NEF noise flag.

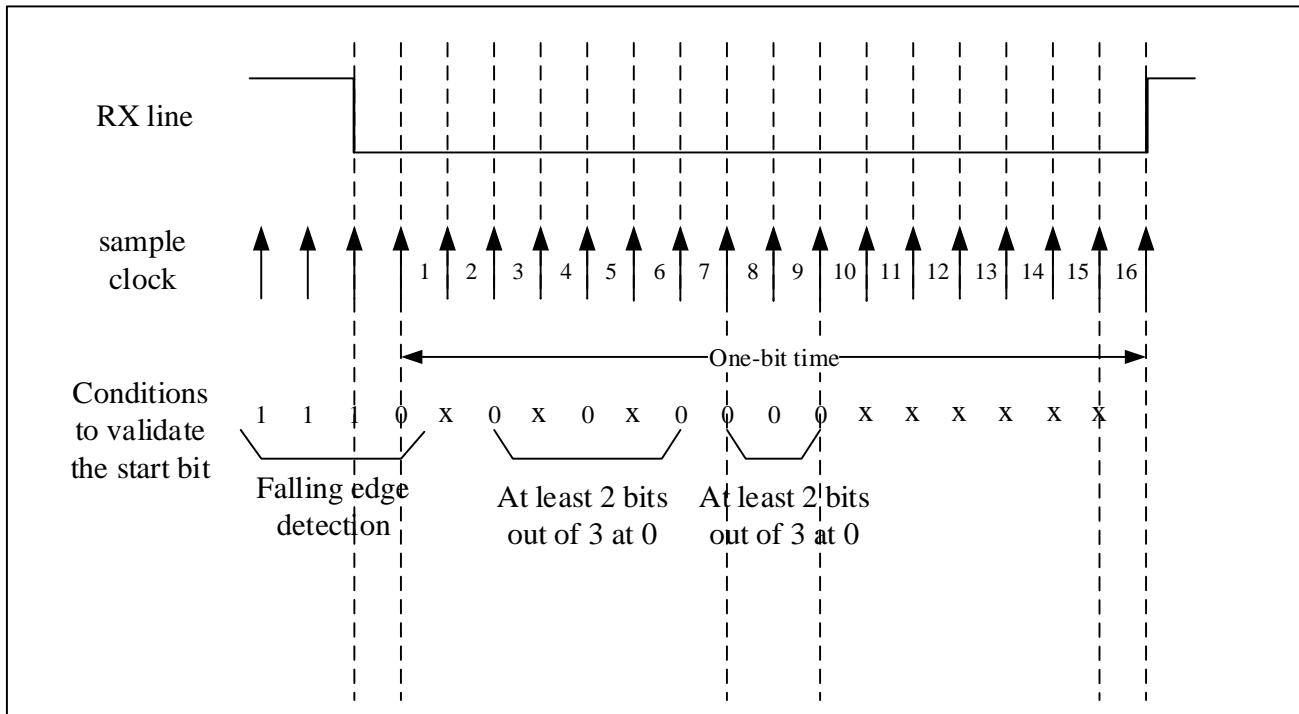
The samples of the 3rd, 5th, and 7th bits have two '0' points, and at the same time, the samples of the 8th, 9th, and 10th bits have three '0' points, then the start bit is confirmed, but it will be set NEF noise flag.

The samples of the 3rd, 5th, and 7th bits have three '0' points, and at the same time, the samples of the 8th, 9th, and 10th bits have two '0' points, then the start bit is confirmed, but it will be set NEF noise flag.

The samples of the 3rd, 5th, and 7th bits have two '0' points, and at the same time, the samples of the 8th, 9th, and 10th bits have two '0' points, then it is confirmed that the start bit is received, but it will be set bit NEF noise flag.

If the sampling values in the 3rd, 5th, 7th, 8th, 9th and 10th bits cannot meet the above four requirements, the USART receiver thinks that it has not received the correct start bit, and will exit the start bit detection and Return to idle state and wait for falling edge.

Figure 18-6 Start bit detection



#### 18.4.3.2 Stop bit description

During data reception, the number of data stop bits can be configured by the USART\_CTRL2.STPB[1:0]. In normal mode, 1 or 2 stop bits can be selected. In Smartcard mode, 0.5 or 1.5 stop bits can be selected.

1. 0.5 stop bits (receive in smartcard mode): 0.5 stop bits are not sampled. Therefore, if 0.5 stop bits is selected, framing errors and broken frames cannot be detected.
2. 1 stop bit: the sampling of one stop bit is carried out through three points, and the 8th, 9th and 10th sampling bits are selected.
3. 1.5 stop bit (Smartcard mode): when sending in Smartcard mode, the device must check whether the data is sent correctly. So the receiver function block must be activated (USART\_CTRL1.RXEN=1) and sample the signal on the data line during the transmission of the stop bit. If a parity error occurs, the smartcard will pull down the data line when the transmitter samples the NACK signal, that is, within the time corresponding to the stop bit on the bus, indicating that a framing error has occurred. The USART\_STS.FEF is set together with the

USART\_STS.RXDNE at the end of the 1.5th stop bit. The 1.5 stop bits were sampled at points 16, 17 and 18.

The 1.5 stop bits can be divided into two parts: one is 0.5 clock cycles, during which nothing is done. This is followed by the stop bit of 1 clock cycle, which is sampled at the midpoint of this period of time. For details, see 18.4.14 Smartcard mode.

4. 2 stop bits: the sampling of the 2 stop bits is completed at the 8th, 9th and 10th sampling points of the first stop position. If a frame error is detected during the first stop bit, the frame error flag is set. The second stop bit does not detect framing error. The USART\_STS.RXNE flag will be set at the end of the first stop bit.

#### 18.4.3.3 Receiver process

1. Enable USART\_CTRL1.UEN to activate USART;
2. Configure the receiver's baud rate, data bit length, parity bit (optional), stop bit number or DMA configuration;
3. Activate the receiver (USART\_CTRL1.RXEN) and start looking for the start bit;
4. The receiver receives 8-bit or 9-bit data according to the configuration of the data bit length, and the least significant bit of the data is first shifted from the RX pin into the receive shift register;
5. When the data of the received shift register is moved to the RDR register, USART\_STS.RXDNE is set, and the data can be read out. If USART\_CTRL1.RXNEIEN is 1, an interrupt will be generated;
6. When an overflow error, noise error, or frame error is detected in the received frame, the corresponding error flag status bit will be set. If USART\_CTRL1.RXEN is reset during data transmission, the data being received will be lost;
7. USART\_STS.RXDNE is set after receiving data, and a read operation of USART\_DAT can clear this bit:
  - During multi-buffer communication, the data register is cleared by the DMA read operation;
  - During single-buffer communication, it is cleared by software reading the USART\_DAT register.

#### 18.4.3.4 Idle frame detection

The receiver of the USART can detect idle frames. An interrupt is generated if USART\_CTRL1.IDLEIEN is '1'. USART\_STS.IDLEF bit is cleared by a software sequence (read USART\_STS register first, then read USART\_DAT register).

#### 18.4.3.5 Break frame detection

The frame error flag(USART\_STS.FEF) is set by hardware when the receiver detects a break frame. It can be cleared by a software sequence (read USART\_STS register first, then read USART\_DAT register).

#### 18.4.3.6 Framing error

A framing error occurs when a stop bit is not received and recognized at the expected time. At this time, the frame error flag USART\_STS.FEF will be set by hardware, and the invalid data will be transferred from the shift register to the USART\_DAT register. During single-byte communication, no framing error interrupt will be generated because it occurs with USART\_STS.RXDNE and the hardware will generate an interrupt when the USART\_STS.RXDNE flag is set. In multi-buffer communication mode, an interrupt will be generated if the USART\_CTRL3.ERRIEN bit is set.

#### 18.4.3.7 Overrun error

When USART\_STS.RXDNE is still '1', when the data currently received in the shift register needs to be transferred to the RDR register, an overflow error will be detected, and the hardware will set USART\_STS.OREF. When this bit is set, the value in the RDR register is not lost, but the data in the shift register is overwritten. It is cleared by a software sequence (read USART\_STS register first, then write USART\_DAT register).

When an overflow error occurs, USART\_STS.RXDNE is '1', and an interrupt is generated. If the USART\_CTRL3.ERRIEN bit is set, an interrupt will be generated when the USART\_STS.OREF flag is set in multi-buffer communication mode.

#### 18.4.3.8 Noise error

USART\_STS.NEF is set by hardware when noise is detected on a received frame. It is cleared by software sequence (read USART\_STS register first, then write USART\_DAT register). During single-byte communication, no noise interrupt generated because it occurs with USART\_STS.RXDNE and the hardware will generate an interrupt when the USART\_STS.RXDNE flag is set. In multi-buffer communication mode, an interrupt is generated when the USART\_STS.NEF flag is set if the USART\_CTRL3.ERRIEN bit is set.

Table 18-2 Data sampling for noise detection

Sample value	NE status	Received bits	Data validity
000	0	0	Effective
001	1	0	be invalid

010	1	0	be invalid
011	1	1	be invalid
100	1	0	be invalid
101	1	1	be invalid
110	1	1	be invalid
111	0	1	Effective

#### 18.4.4 Generation of fractional baud rate

The baud rate of the USART can be configured in the USART\_BRCF register. This register defines the integer and fractional parts of the baud rate divider. The baud rate of the transmitter and receiver should be configured to the same value. Be careful not to change the value of the USART\_BRCF register during communication, because the baud rate counter will be replaced by the new value of the baud rate register.

$$\text{TX / RX baud rate} = f_{\text{PCLK}} / (16 * \text{USARTDIV})$$

where  $f_{\text{PCLK}}$  is the clock provided to the peripheral:

- PCLK1 is used for USART2, up to 48MHz;
- PCLK2 is used for USART1, up to 48 MHz.

USARTDIV is an unsigned fixed-point number.

##### 18.4.4.1 USARTDIV and USART\_BRCF register configuration

Example 1:

If DIV\_Integer = 27, DIV\_Decimal = 12 (USART\_BRCF = 0x1BC), then

$$\text{DIV_Integer(USARTDIV)} = 27$$

$$\text{DIV_Decimal(USARTDIV)} = 12/16 = 0.75$$

$$\text{So USARTDIV} = 27.75$$

Example 2:

Requirements USARTDIV = 25.62, there are:

$$\text{DIV_Decimal} = 16 * 0.62 = 9.92$$

Closest integer is: 10 = 0x0A

$$\text{DIV_Integer} = \text{DIV_Integer}(25.62) = 25 = 0x19$$

$$\text{So, USART_BRCF} = 0x19A$$

Example 3:

Requirements USARTDIV = 50.99, there are:

$$\text{DIV\_Decimal} = 16 * 0.99 = 15.84$$

Closest integer:  $16 = 0x10 \Rightarrow \text{DIV\_Decimal}[3:0]$  overrun  $\Rightarrow$  carry must be added to the fractional part

$$\text{DIV\_Integer} = \text{DIV\_Integer} (0d50.990 + \text{carry}) = 51 = 0x33$$

So: USART\_BRCF= 0x330, USARTDIV = 0d51.00

Table 18-3 Error calculation when setting baud rate

Baud rate		$f_{PCLK} = 36\text{MHz}$			$f_{PCLK} = 48\text{MHz}$		
serial number	Kbps	Reality	Set value in register	Error(%)	Reality	Set value in register	Error(%)
1	2.4	2.4	937.5	0%	2.4	1250	0%
2	9.6	9.6	234.375	0%	9.6	312.5	0%
3	19.2	19.2	117.1875	0%	19.2	156.25	0%
4	57.6	57.6	39.0625	0%	57.623	52.0625	0.04%
5	115.2	115.384	19.5	0.15%	115.1	26.0625	0.08%
6	230.4	230.769	9.75	0.16%	230.769	13	0.16%
7	460.8	461.538	4.875	0.16%	461.538	6.5	0.16%
8	921.6	923.076	2.4375	0.16%	923.076	3.25	0.16%
9	2250	2250	1	0%	2285.714	1.3125	1.58%
10	3000	impossible	impossible	impossible	3000	1	0%

Notes: The lower the clock frequency of the CPU, the lower the error for a particular baud rate.

### 18.4.5 Receiver's tolerance clock deviation

Variations due to transmitter errors (including transmitter side oscillator variations), receiver side baud rate rounding errors, receiver side oscillator variations, variations due to transmission lines (usually due to The inconsistency between the low-to-high transition timing of the transceiver and the high-to-low transition timing of the transceiver), these factors will affect the overall clock system variation. Only when the sum of the above four changes is less than

the tolerance of the USART receiver, the USART asynchronous receiver can work normally.

When receiving data normally, the tolerance of the USART receiver depends on the selection of the data bit length and whether it is generated using a fractional baud rate. The tolerance of the USART receiver is equal to the maximum tolerable variation.

Table 18-4 when DIV\_Decimal = 0. Tolerance of USART receiver

WL bit	NF is an error	NF is don't care
0	3.75%	4.375%
1	3.41%	3.97%

Table 18-5 when DIV\_Decimal != 0. Tolerance of USART receiver

WL bit	NF is an error	NF is don't care
0	3.33%	3.88%
1	3.03%	3.53%

## 18.4.6 Parity control

Parity can be enabled by configuring the USART\_CTRL1.PCEN bit.

When the parity bit is enabled for transmission, A parity bit is generated, parity check is performed on reception.

Table 18-6 Frame format

WL bit	PCEN bit	USART frame
0	0	Start bit   8-bit data   Stop bit
0	1	Start bit   7 bits of data   Parity bit   Stop bit
1	0	Start bit   9-bit data   Stop bit
1	1	start bit   8-bit data   parity bit   stop bit

### Even parity

Configure USART\_CTRL1.PSEL to 0, and even parity can be selected.

Make the number of '1' in the transmitted data (including parity bit) be an even number. That is: if Data=11000101, there are 4 '1's, then the parity bit will be '0' (4 '1' in total). After the data and check digit are sent to the receiver, the receiver calculates the number of 1s in the data again. If it is an even number, the check is passed, indicating that no

errors occurred during the transmission process. If it is not even, it means that an error has occurred, the USART\_STS.PEF flag is set to '1', and if USART\_CTRL1.PEIEN is enabled, an interrupt is generated.

### Odd parity

Configure USART\_CTRL1.PSEL to 1, you can choose odd parity.

Make the number of '1' in the transmitted data (including parity bit) be an odd number. That is: if Data=11000101, there are 4 '1's, then the parity bit will be '1' (5 '1' in total). After the data and check digit are sent to the receiver, the receiver calculates the number of 1s in the data again. If it is an odd number, the check is passed, indicating that no errors occurred during the transmission process. If it is not an odd number, it means that an error has occurred, the USART\_STS.PEF flag is set to '1', and if USART\_CTRL1.PEIEN is enabled, an interrupt is generated.

## 18.4.7 DMA application

The USART supports the DMA mode using multi-buffer configuration, which can realize high-speed data communication.

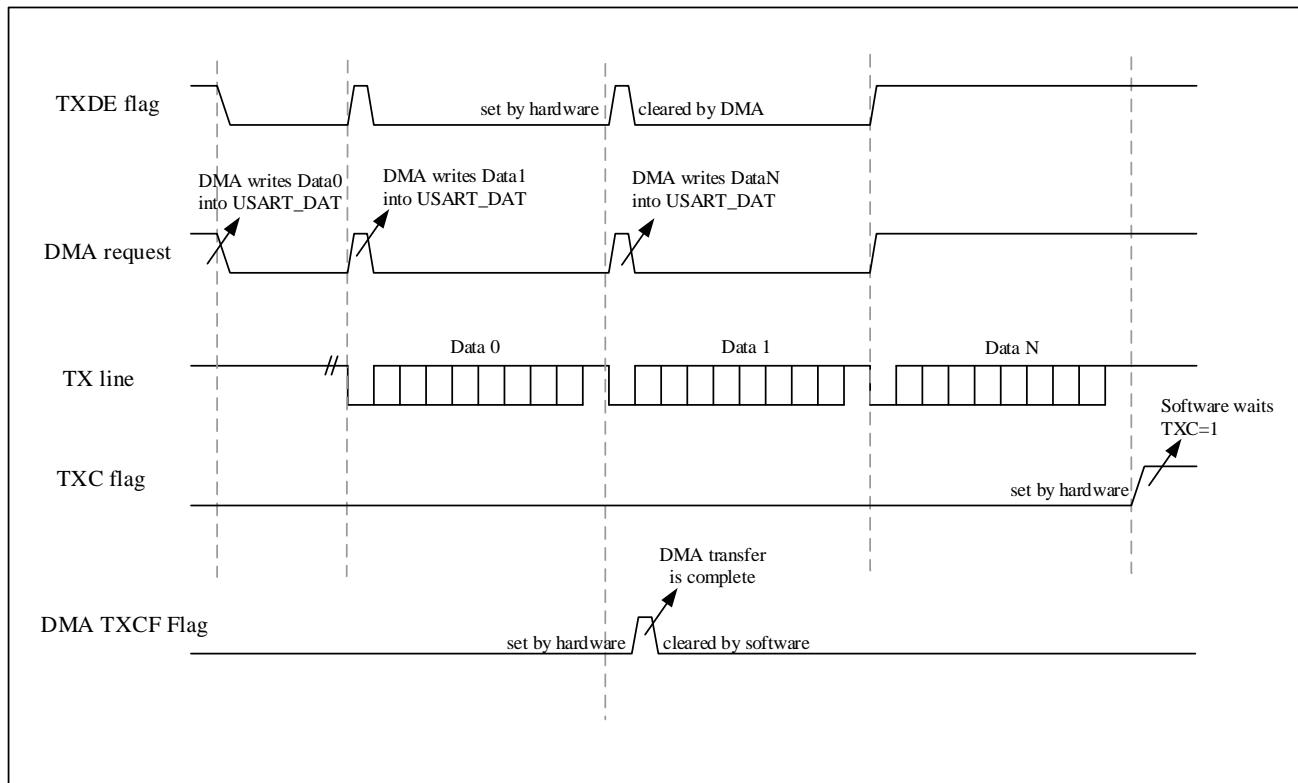
### 18.4.7.1 Using DMA transmission

Set USART\_CTRL3.DMATXEN to enable DMA mode when transmitting. When the USART's transmit shift register is empty (USART\_STS.TXDE=1), the DMA will transfer the data from the SRAM to the USART\_DAT register of the USART.

When using DMA transmission, the process of configuring the DMA channel is as follows:

1. Set the address of the data memory. When a data transfer request occurs, the transferred data will be read from this address.
2. Set the address of the USART\_DAT register. When a data transfer request occurs, this address will be the destination address of the data transfer.
3. Set the amount of data to transfer.
4. Set the priority of the channel, set whether to use the cyclic mode, the incremental mode of peripherals and memory, the data width of peripherals and memory, the interrupt generated by half of the transfer or the interrupt when the transfer is completed.
5. Start the channel.
6. After the data transfer is completed, the transfer complete flag (DMA\_INTSTS.TXCFx) is set to 1.

Figure 18-7 Transmission using DMA



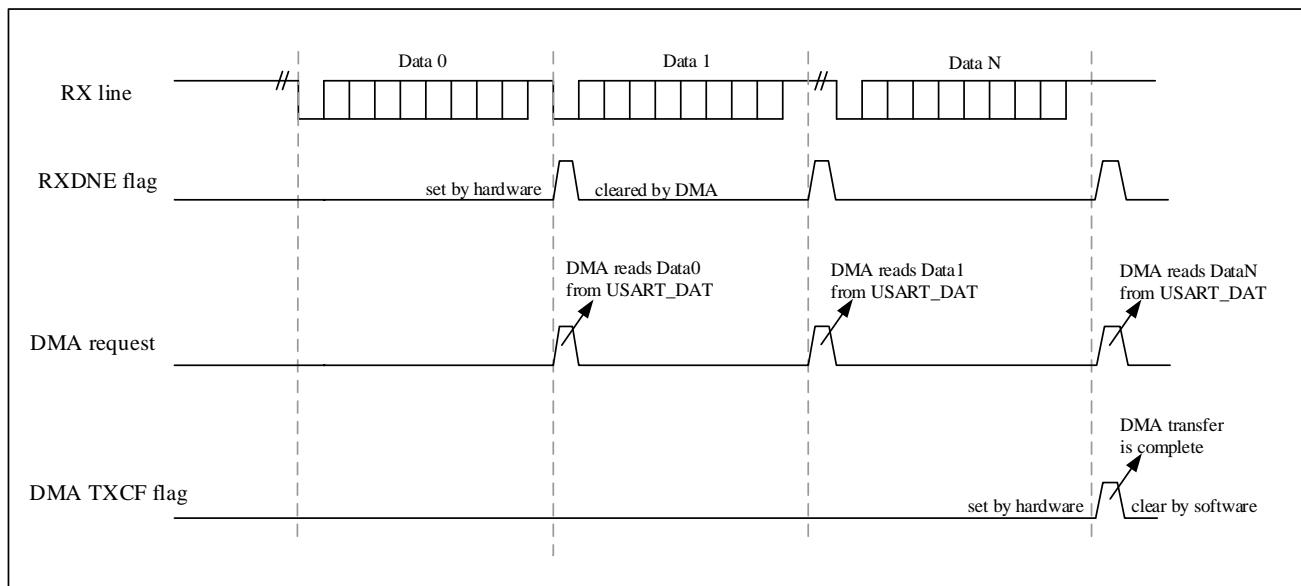
#### 18.4.7.2 Using DMA reception

Set `USART_CTRL3.DMARXEN` to enable DMA mode when receiving. When a byte is received (`USART_STS.RXDNE=1`), the DMA will transfer the data from the `USART_DAT` register of the USART to the SRAM.

When using DMA reception, the process of configuring the DMA channel is as follows:

1. Set the address of the `USART_DAT` register. When a data transfer request occurs, this address will be the source address of the data transfer.
2. Set the address of the data memory. When a data transfer request occurs, the transferred data will be written to this address.
3. Set the amount of data to transfer.
4. Set the priority of the channel, set whether to use the cyclic mode, the incremental mode of peripherals and memory, the data width of peripherals and memory, the interrupt generated by half of the transfer or the interrupt when the transfer is completed.
5. Start the channel.

Figure 18-8 Reception using DMA

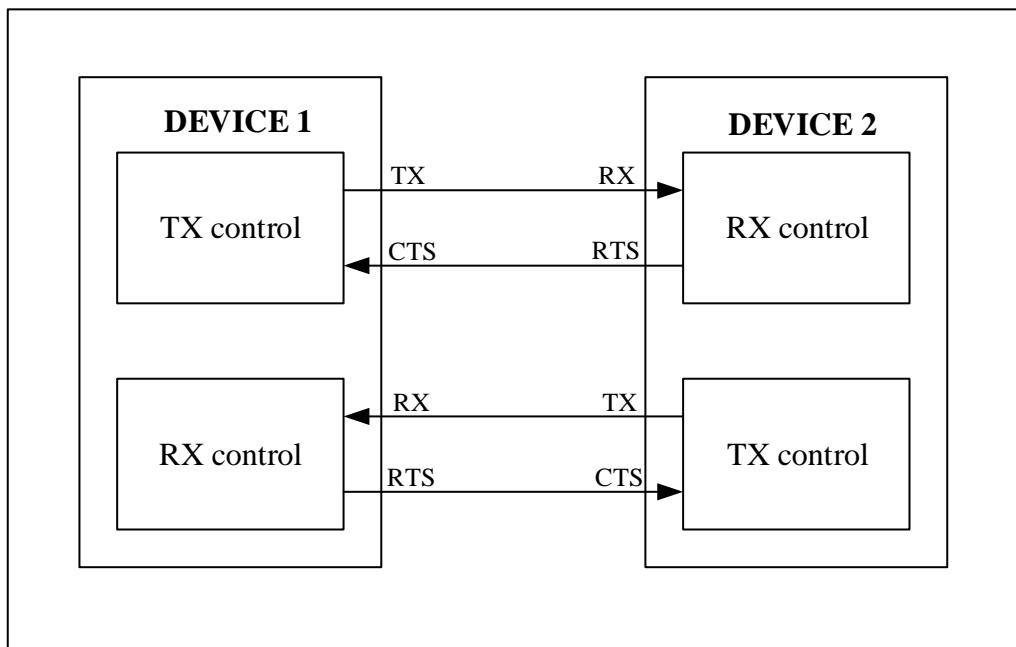


In multi-buffer communication mode, the error flag will be set when there is a frame error, overrun or noise error. An interrupt will be generated if the error interrupt is enabled (USART\_CTRL3.ERRIEN=1).

#### 18.4.8 Hardware flow control

USART supports hardware flow control. The purpose is to coordinate the sending and receiving parties so that the data will not be lost. The connection method is shown in the following figure.

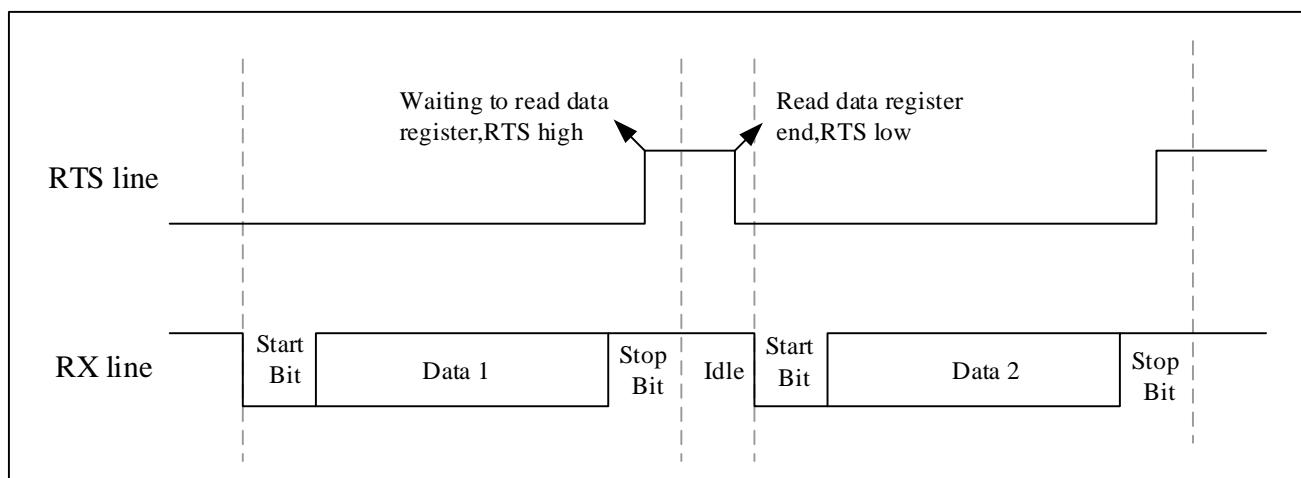
Figure 18-9 hardware flow control between two USART



#### 18.4.8.1 RTS flow control

Set USART\_CTRL3.RTSEN to enable RTS. RTS is the output signal used to indicate that the receiver is ready. When data arrives in RDR, pull high nRTS output, notifying the sender to stop data transmission at the end of the current frame. when receiver is ready to receive new data, assert (pull low) the nRTS output.

Figure 18-10 RTS flow control



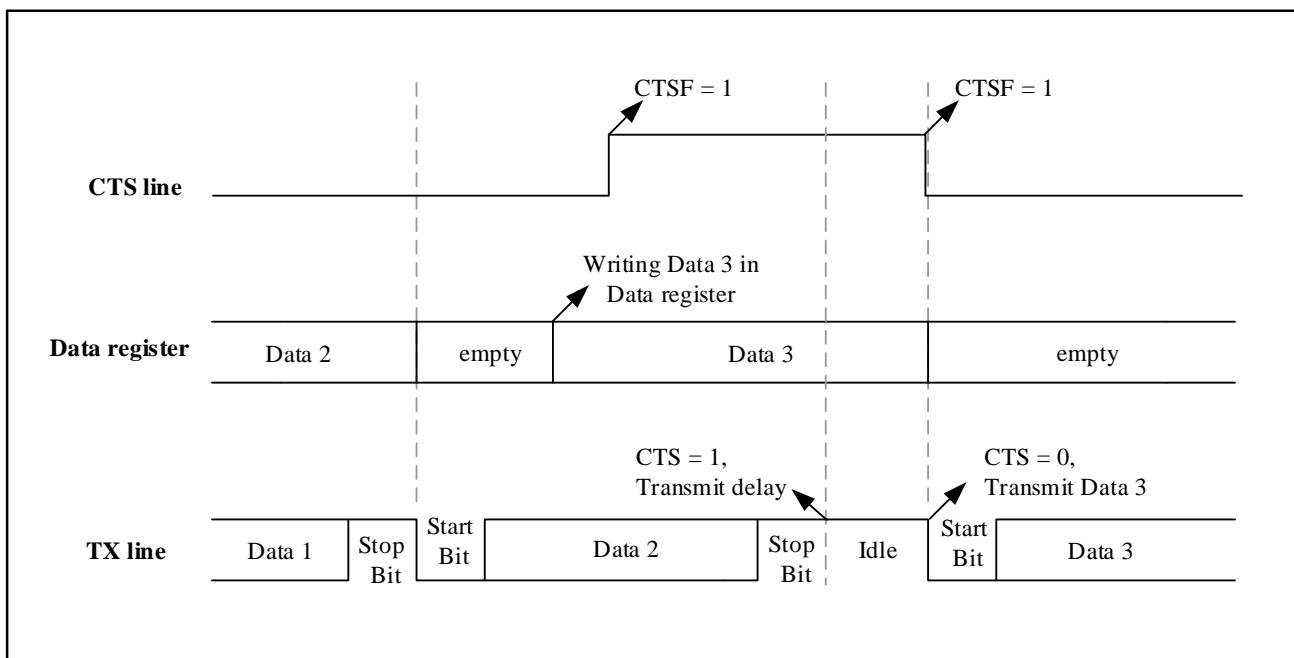
#### 18.4.8.2 CTS flow control

Set USART\_CTRL3.CTSEN to enable CTS. CTS is an input signal, used to judge whether data can be sent to the other device. The low level is valid, and the low level indicates that the device can send data to the other device. If

the nCTS signal becomes invalid during data transmission, the transmission will stop after sending the data. If you write data to the data register when nCTS is invalid, the data will not be sent until nCTS is valid.

If the USART\_CTRL3.CTSEN bit is set, the USART\_STS.CTSF bit will be set high by hardware when the nCTS input changes state. An interrupt will be generated if USART\_CTRL3.CTSIEN is enabled.

Figure 18-11 CTS flow controls



### 18.4.9 Multiprocessor communication

USART allows multiprocessor communication. The principle is: multiple processors communicate through USART, and it is necessary to determine who is the master device, and the remaining processors are all slave devices. The TX output of the master device is directly connected to the RX port of all slave device. The TX outputs of the slaves are logically AND together and connected to the RX inputs of the master.

When multi-processor communication is performed, the slave devices are all in mute mode, and the host uses a specific method to wake up a slave device to be communicated when needed, so that the slave device is in an active state and transmits data with the master device.

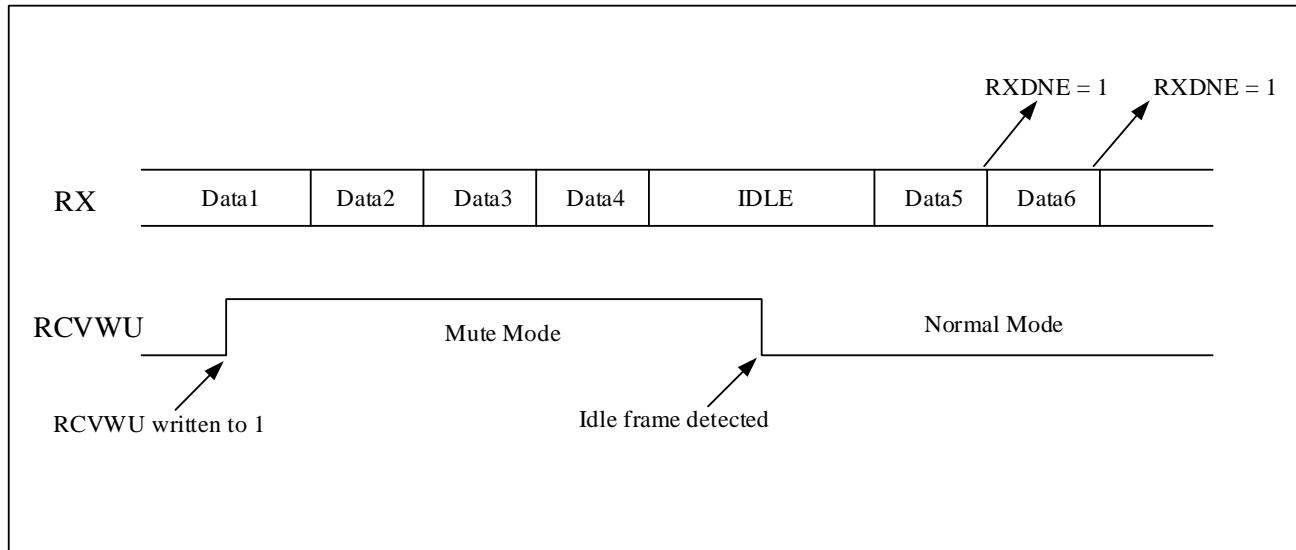
The USART can wake up from mute mode by idle line detection or address mark detection.

#### 18.4.9.1 Idle line detection

The idle line detection configuration process is as follows:

1. Configure the USART\_CTRL1.WUM bit to 0, and the USART performs idle line detection;
2. When USART\_CTRL1.RCVWU is set (which can be automatically controlled by hardware or written by software under certain conditions), USART enters mute mode. In mute mode, none of the receive status bits are set, and all receive interrupts are disabled;
3. As shown in the Figure 18-12 below, when an idle frame is detected, USART is woken up, and then USART\_CTRL1.RCVWU is cleared by hardware. At this time, USART\_STS.IDLEF is not set.

Figure 18-12 Mute mode using idle line detection



#### 18.4.9.2 Address mark detection

By configuring the USART\_CTRL1.WUM bit to 1, the USART performs address mark detection. The address of the receiver is programmable through the USART\_CTRL2.ADDR[3:0] bits. If the MSB is 1, the byte is considered an address, otherwise it is considered data.

In this mode, the USART can enter mute mode by:

- When the receiver does not contain data, USART\_CTRL1.RCVWU can be written to 1 by software, and USART enters mute mode;

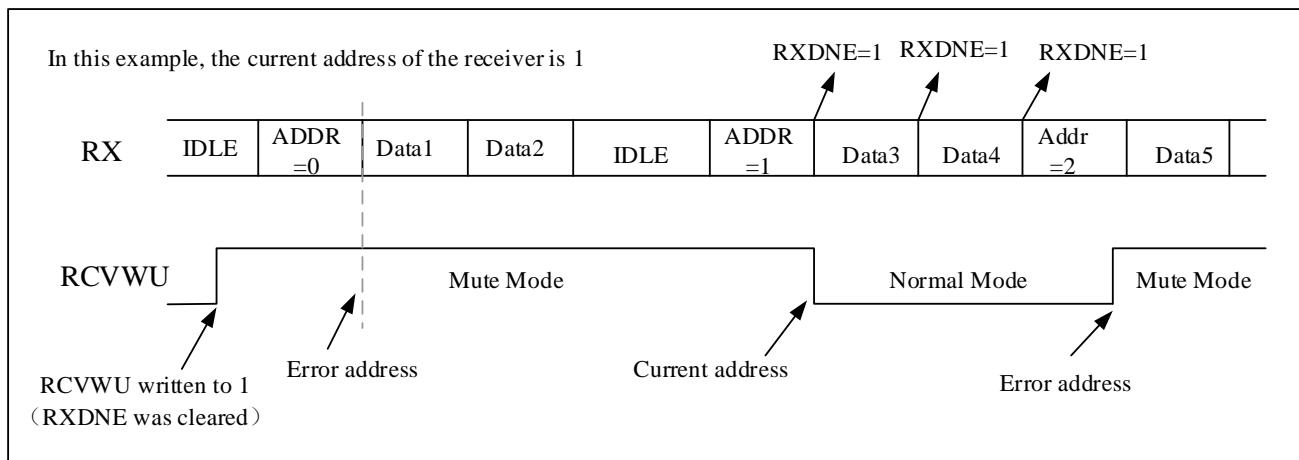
*Note: When the receive buffer contains no data (RXNE=0 in USART\_SR), the USART\_CTRL1.RCVWU bit can be written to 0 or 1. Otherwise, the write operation is ignored.*

- When the received address does not match the address of the USART\_CTRL2.ADDR[3:0] bits, USART\_CTRL1.RCVWU is written to 1 by hardware.

In mute mode, none of the receive status bits are set and all receive interrupts are disabled.

When the received address matches the address of the USART\_CTRL2.ADDR[3:0] bits, the USART is woken up and USART\_CTRL1.RCVWU is cleared. The USART\_STS.RXDNE bit will be set when this matching address is received. Data can then be transmitted normally.

Figure 18-13 Mute mode detected using address mark



### 18.4.10 Synchronous mode

USART supports synchronous serial communication. The USART only supports the master mode, and cannot use the input clock from other devices to receive and transmit data. Synchronous mode can be enabled by configuring the USART\_CTRL2.CLKEN bit.

*Note: When using synchronous mode, USART\_CTRL2.LINMEN, USART\_CTRL3.SCMEN, USART\_CTRL3.HDMEN, USART\_CTRL3.IRDAMEN, these bits need to be kept clear.*

#### 18.4.10.1 Synchronized clock

The CK pin is the output of the USART transmitter clock. During the bus idle period, before the actual data arrives and when the break symbol is sent, the clock not output.

Clock phase and polarity are software programmable and need to be configured when both the transmitter and receiver are disabled. When the clock polarity is 0 (USART\_CTRL2.CLKPOL=0), the default level of CLK is low; when the clock polarity is 1 (USART\_CTRL2.CLKPOL=1), the default level of CLK is high. When the phase polarity is 0 (USART\_CTRL2.CLKPHA=0), the data is sampled on the first edge of the clock; when the phase polarity is 1 (USART\_CTRL2.CLKPHA=1), the data is sampled on the second edge.

During the start and stop bits, the CK pin does not output clock pulses.

A sync data cannot be received when no data is sent. Because the clock is only available when the transmitter is activated and data is written to the USART\_DAT register.

The USART\_CTRL2.LBCLK bit controls whether to output the clock pulse corresponding to the last data byte (MSB) sent on the CK pin. This bit needs to be configured when both the transmitter and receiver are disabled. If USART\_CTRL2.LBCLK is 1, the clock pulse of the last bit of data will be output from CK. If USART\_CTRL2.LBCLK is 0, the clock pulse of the last bit of data is not output from CK.

#### 18.4.10.2 Synchronized transmitting

The transmitter in synchronous mode works the same as in asynchronous mode. Data on the TX pin is sent out synchronously with CK.

#### 18.4.10.3 Synchronized receiving

The receiver in synchronous mode works differently than in asynchronous mode. Data is sampled on CK without any oversampling. But setup time and hold time (depending on baud rate, 1/16 bit time) must be considered.

Figure 18-14 USART synchronous transmission example

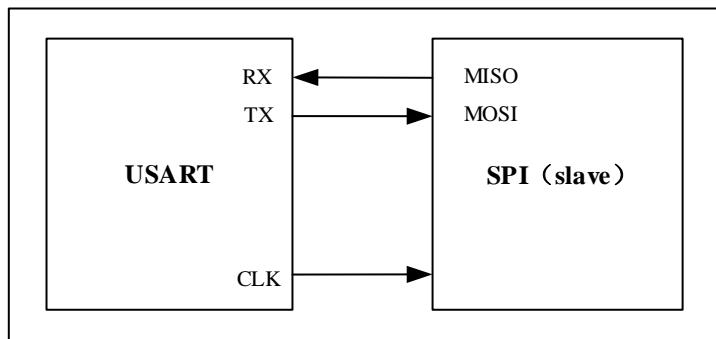


Figure 18-15 USART data clock timing example (WL=0)

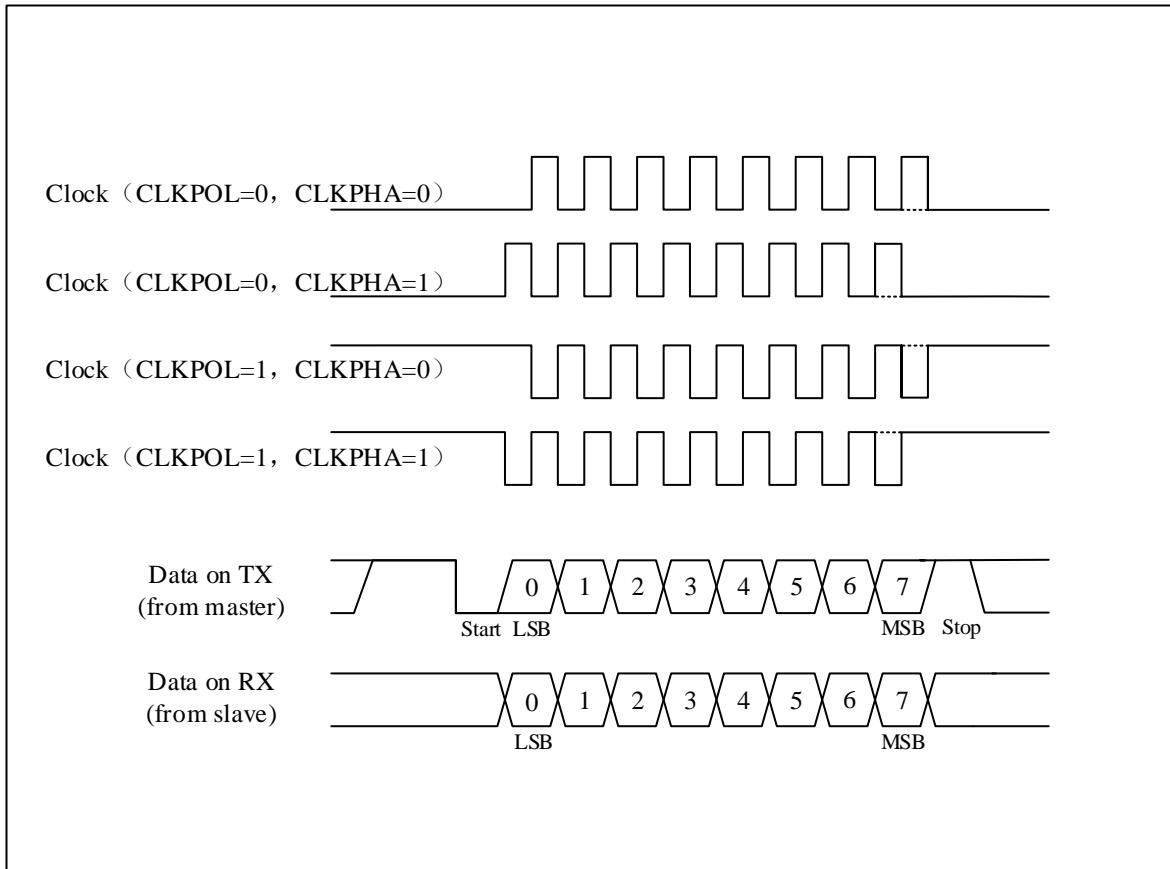


Figure 18-16 USART data clock timing example (WL=1)

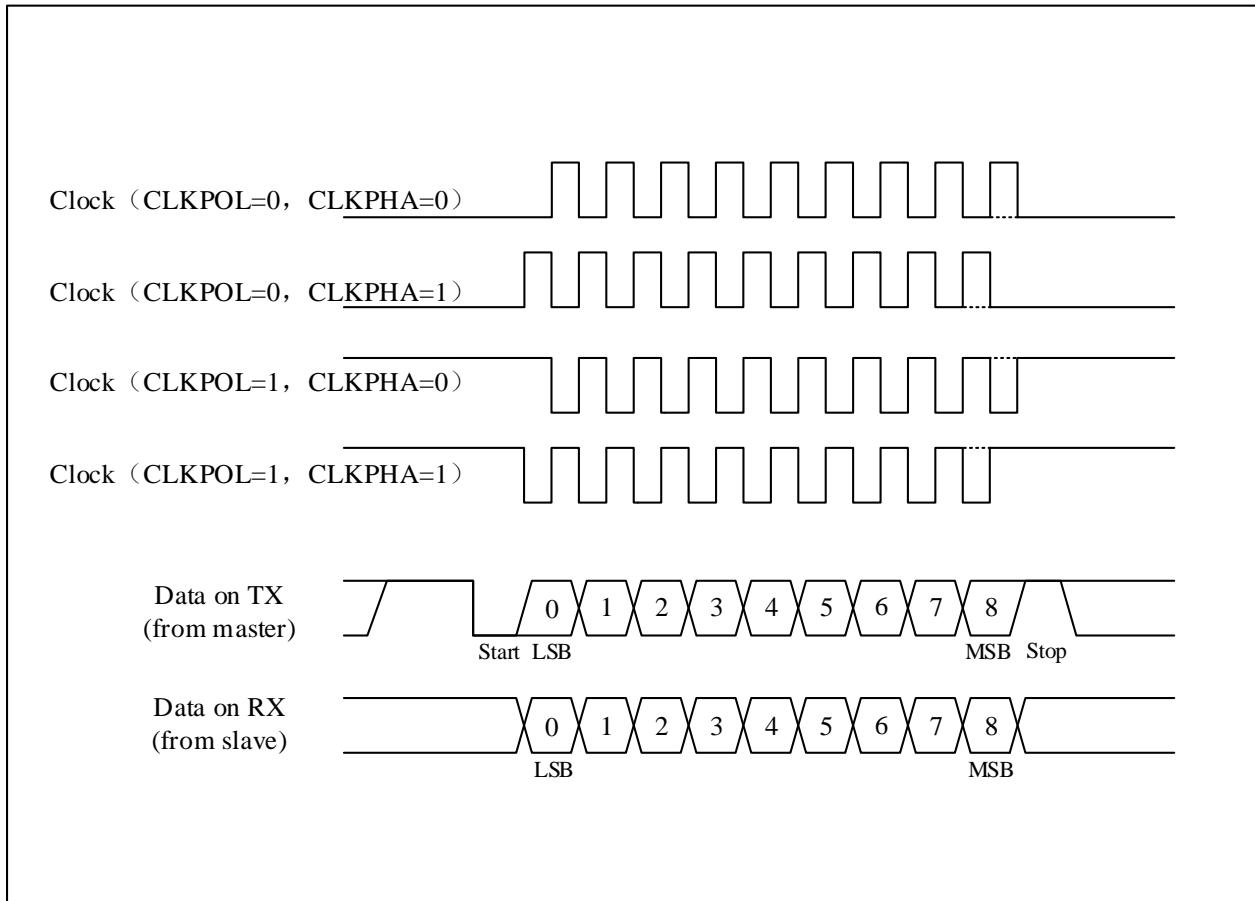
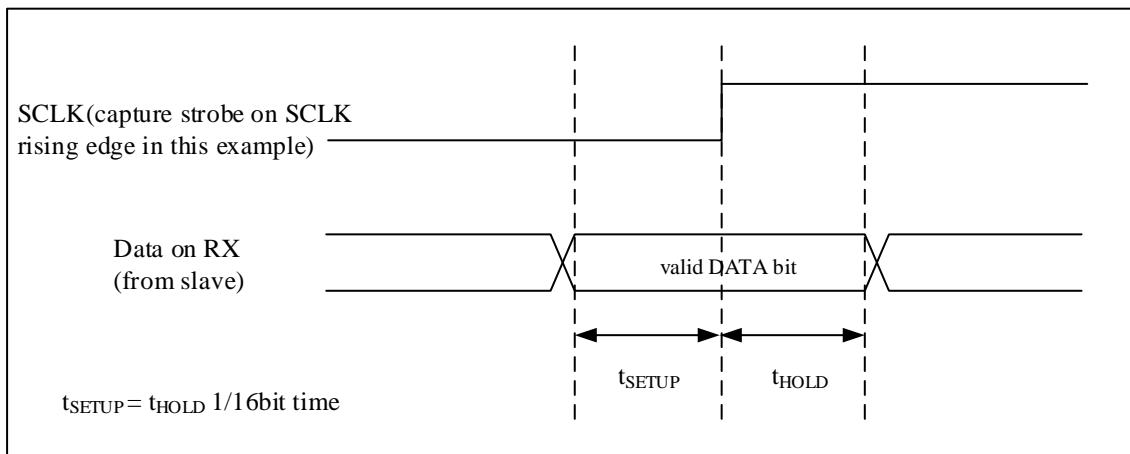


Figure 18-17 RX data sampling / holding time



*Note: the function of CK is different in Smartcard mode, please refer to the Smartcard mode section for details.*

### 18.4.11 Single-wire half-duplex mode

USART supports single-wire half-duplex communication, allowing data to be transmitted in both directions, but only allows data to be transmitted in one direction at the same time. Communication conflicts are managed by software.

Through the USART\_CTRL3.HDMEN bit, you can choose whether to enable half-duplex mode. When using single-wire half-duplex, USART\_CTRL2. CLKEN, USART\_CTRL2. LINMEN, USART\_CTRL3. SCMEN, USART\_CTRL3. IRDAMEN, these bits should be kept clear.

After the half-duplex mode is turned on, the TX pin and the RX pin are interconnected inside the chip, and the Rx pin is no longer used. When there is no data to transmit, TX is always released. Therefore, when not driven by the USART, the TX pin must be configured as a floating input or an open-drain output high.

### 18.4.12 IrDA SIR ENDEC mode

USART supports the IrDA (Infrared Data Association) SIR ENDEC specification.

Through the USART\_CTRL3. IRDAMEN bit, you can choose whether to enable the infrared mode. When using the infrared function, USART\_CTRL2. CLKEN, USART\_CTRL2. STPB[1:0], USART\_CTRL2. LINMEN, USART\_CTRL3. HDMEN, USART\_CTRL3. SCMEN, these bits should be kept clear.

Through the USART\_CTRL3. IRDALP bit, it can be used to select normal mode or low power infrared mode.

#### 18.4.12.1 IrDA normal mode

When USART\_CTRL3.IRDALP=0, select normal infrared mode.

IrDA is a half-duplex communication protocol, so there should be a minimum delay of 10ms between sending and receiving.that uses a inverted return-to-zero modulation scheme (RZI), which uses an infrared light pulse to represent a logic '0', and the pulse width is specified as 3/16 of a bit period in normal mode, as shown in the Figure 18-19.USART only supports up to 115200bps for SIR ENDEC .

The USART sends data to the SIR encoder, and the bit stream output by the USART will be modulated. A modulated stream of pulses is sent from the infrared transmitter and then received by the infrared receiver. The SIR receiver decoder demodulates it and outputs the data to the USART.

The transmit encoder output has opposite polarity to the decoder input. When idle, SIR transmit is low, while SIR

receive is high. The high pulse sent by SIR is '0' and the low level is '1', while SIR reception is the opposite.

If the USART is sending data to the IrDA transmit encoder, then the IrDA receive decoder will ignore any data on the IrDA receive line. If the USART is receiving data sent from the SIR receiver decoder, the data sent by the USART to the IrDA transmitter encoder will not be encoded.

Pulse width is programmable. The IrDA specification requires pulses to be wider than 1.41us. For pulse widths less than 2 cycles, the receiver will filter them out. PSCV is the prescaler value programmed in the USART\_GTP register.

#### 18.4.12.2 IrDA low power mode

When USART\_CTRL3.IRDALP=1, select low power infrared mode.

For the transmitter, when in low power mode, the pulse width is 3 times the low power baud rate, which is a minimum of 1.42MHz. Typically this value is 1.8432MHz ( $1.42 \text{ MHz} < \text{PSC} < 2.12 \text{ MHz}$ ).

For the receiver, the requirement for a valid signal is that the duration of the low level signal must be greater than 2 cycles of the IrDA low power baud rate clock.

Figure 18-18 IrDASIRENDEC-Block diagram

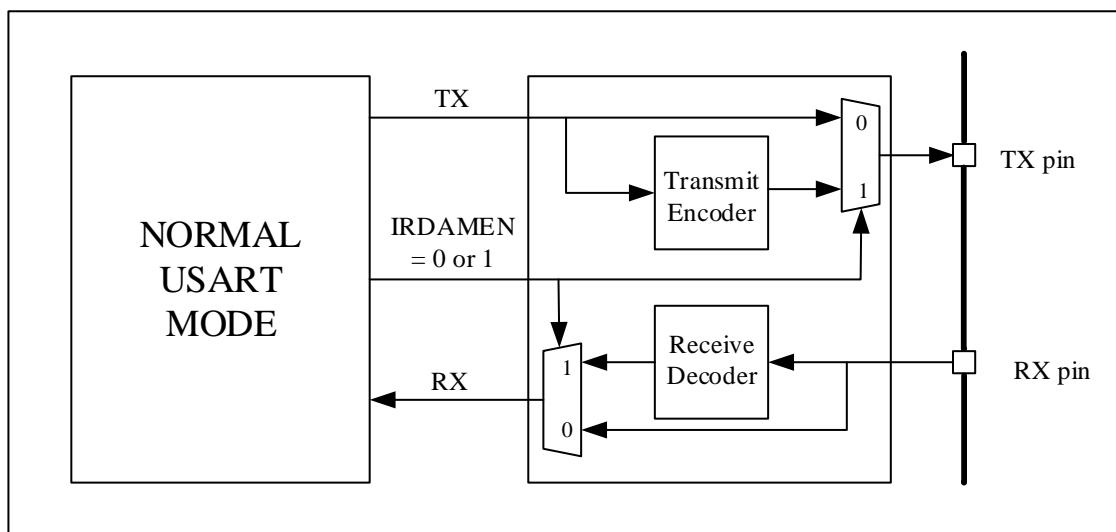
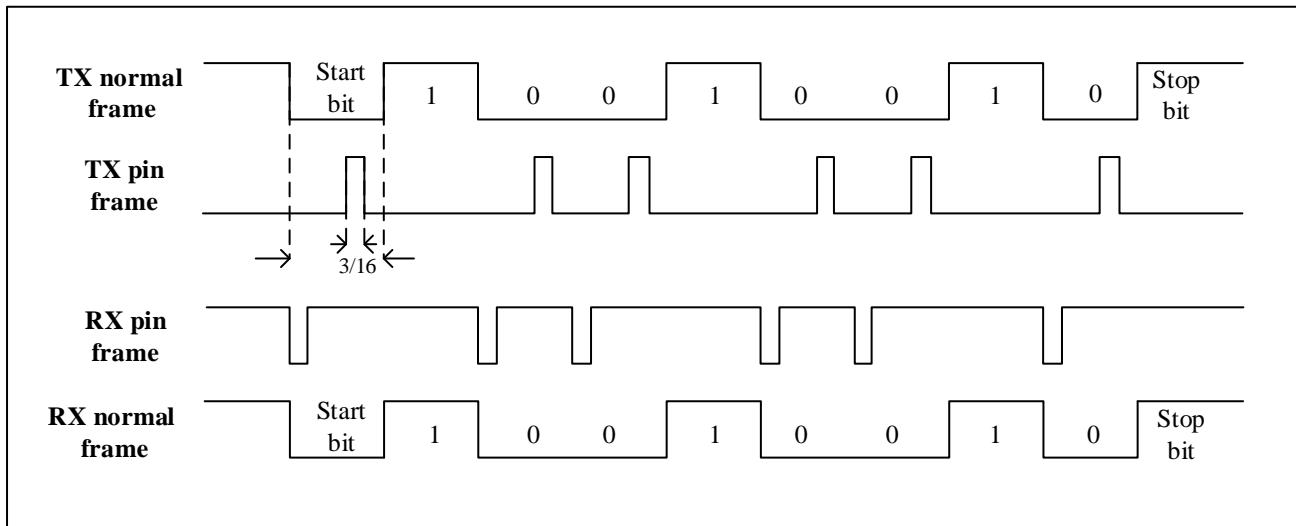


Figure 18-19 IrDA data Modulation (3/16)-normal mode



### 18.4.13 LIN mode

USART supports the ability of a LIN (Local interconnection Network) master to send a synchronization break and the ability of a LIN slave to detect a break. LIN mode can be enabled by configuring the USART\_CTRL2.LINMEN bit.

*Note: When using LIN mode, USART\_CTRL2.STPB[1:0], USART\_CTRL2.CLKEN, USART\_CTRL3.SCMEN, USART\_CTRL3.HDMEN, USART\_CTRL3.IRDAMEN, these bits should be kept clear.*

#### 18.4.13.1 LIN transmitting

When LIN is sent, the length of the data bits sent can only be 8 bits. By setting USART\_CTRL1.SDBRK, a 13-bit '0' will be sent as the break symbol , and insert a stop bit.

#### 18.4.13.2 LIN receiving

Whether the bus is idle or during the transmission of a data frame, as long as the break frame appears, it can be detected. the break symbol detection is independent of the USART receiver.

By configuring the USART\_CTRL2.LINBDL bit, 10-bit or 11-bit break character detection can be selected.

After the receiver detects the start bit, the circuit samples each subsequent bit at the 8th, 9th, and 10th oversampling clock points of each bit. When 10 or 11 consecutive bits are detected as '0' and followed by a delimiter, it means that a LIN break is detected, and USART\_STS.LINBDF is set. Before confirming the break symbol, check the delimiter as it means the RX line has gone back to high. An interrupt is generated if the LIN breaker detection interrupt (USART\_CTRL2.LINBDIEN) is enabled.

If a '1' is sampled before the 10th or 11th sample point, the current detection is canceled and the start bit is searched again.

Figure 18-20 Break detection in LIN mode (11-bit break length-the LINBDL bit is set)

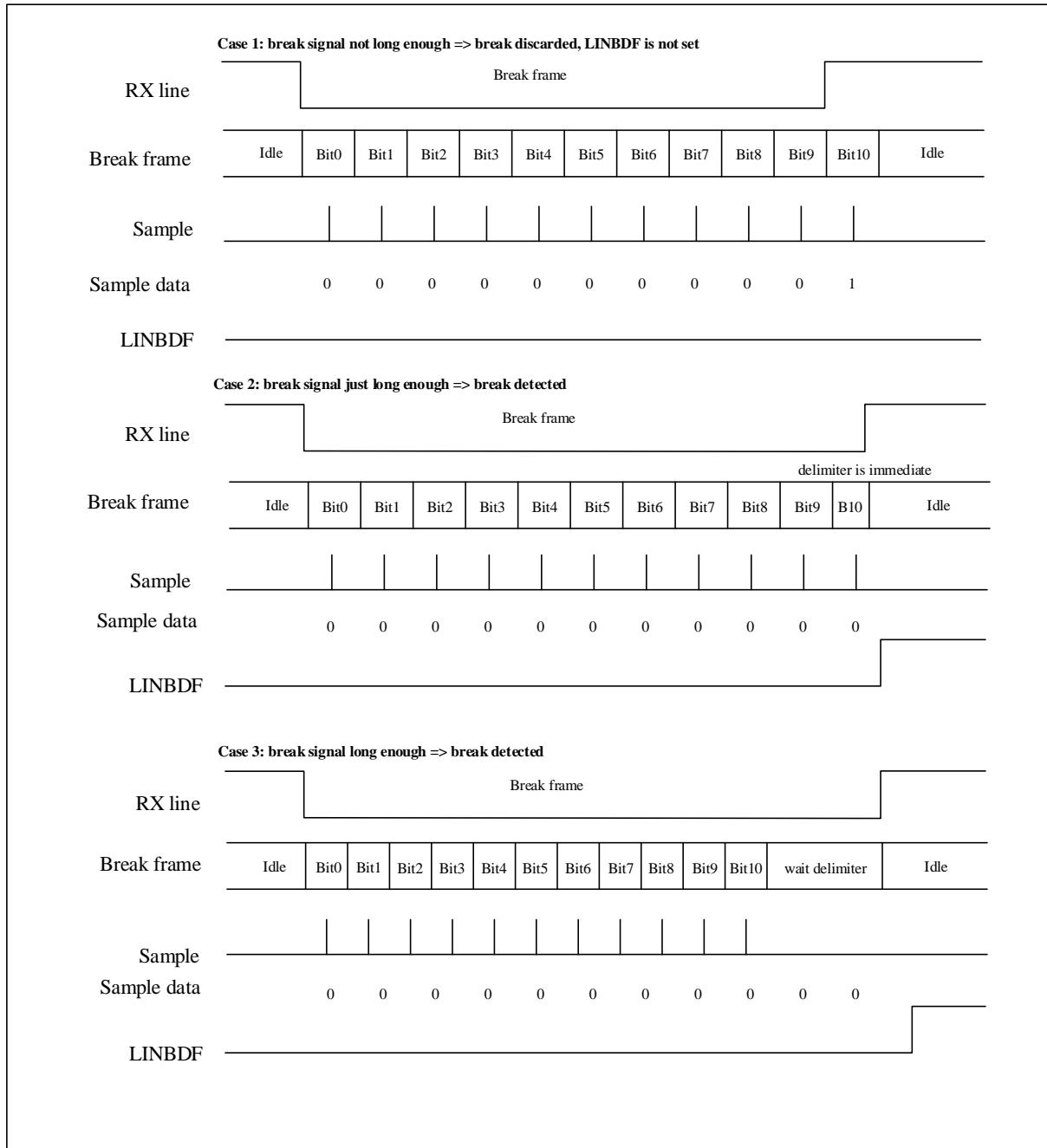
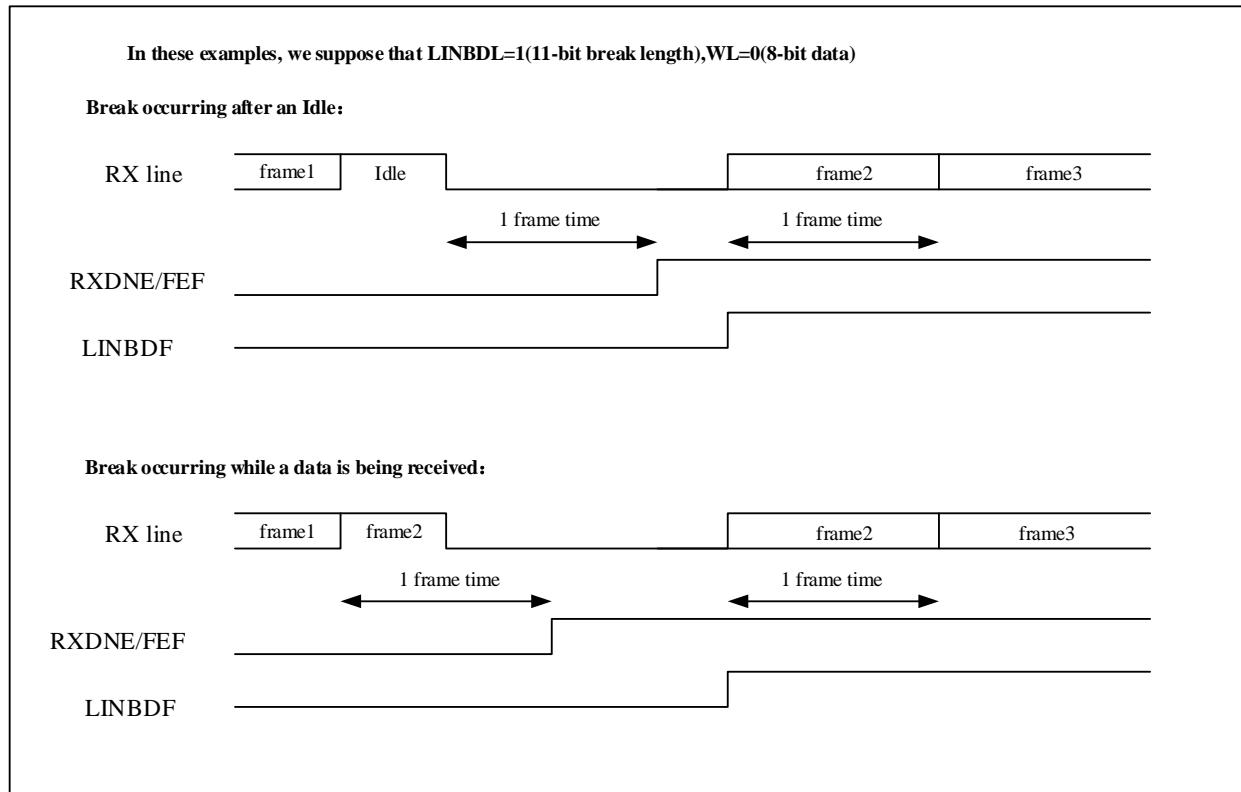


Figure 18-21 Break detection and framing error detection in LIN mode



#### 18.4.14 Smartcard mode (ISO7816)

USART supports smart card protocol. The smart card interface supports the asynchronous smart card protocol defined in the ISO7816-3 standard.

Through the USART\_CTRL3. SCLEN bit, you can choose whether to enable smart card mode. When using smart card mode, USART\_CTRL2. LINMEN, USART\_CTRL3. HDMEN, USART\_CTRL3. IRDAMEN, these bits should be kept clear.

In smart card mode, the USART can provide a clock through the CK pin. The system clock is divided by the prescaler register to provide the clock to the smart card. The CK frequency can be from  $f_{CK}/2$  to  $f_{CK}/62$ , where  $f_{CK}$  is the peripheral input clock.

In smart card mode, 0.5 and 1.5 stop bits can be used when receiving data, and only 1.5 stop bits can be used when sending data. So 1.5 stop bits are recommended as this avoids configuration transitions.

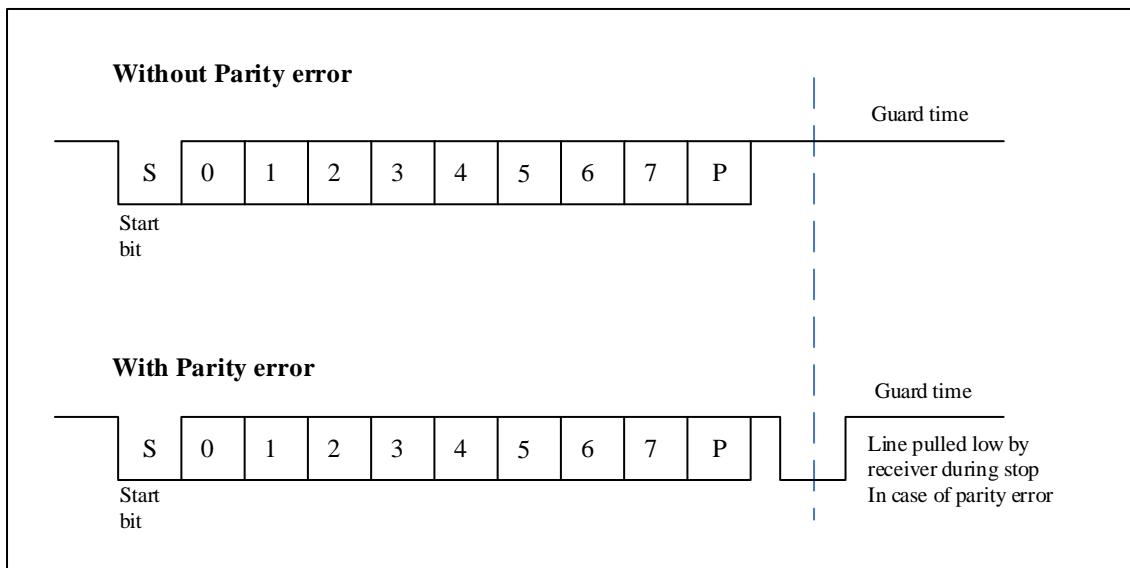
In smart card mode, the data bits should be configured as 8 bits, and the parity bit should be configured.

When a parity error is detected by receiver, the transmit data line is pulled low for one baud clock cycle at the end of the stop bit as NACK signal(If USART\_CTRL3.SCNAACK is set). This NACK signal will generate a framing error on the transmit side (transmit side is configured with 1.5 stop bits).

When the transmitter receives a NACK signal (framing error) from the receiver, it does not detect the NACK as a start bit (according to the ISO protocol, the duration of the received NACK can be 1 or 2 baud clock cycles).

The example given in the following figure illustrates the signal on the data line with and without parity errors.

Figure 18-22 ISO7816-3 Asynchronous Protocol



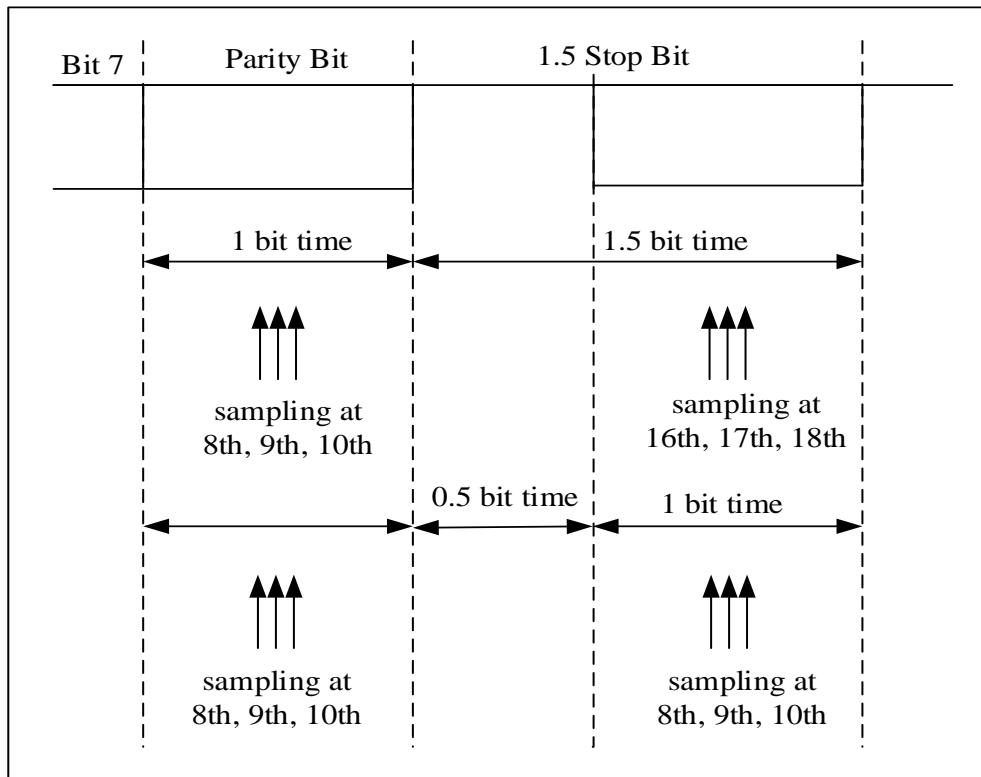
The break frame has no meaning in smart card mode. A 00h data with a framing error will be treated as data instead of a break symbol.

Under normal operation, data will be shifted out of the transmit shift register on the next baud clock. The smart card mode is delayed by a minimum of 1/2 baud clock than normal operation.

In normal operation, USART\_STS.TXC is set when a frame containing data is sent and USART\_STS.TXDE=1. In smart card mode, the transmission completion flag (USART\_STS.TXC) is set high when the guard time counter reaches the value (USART\_GTP.GTV[7:0]). The clearing of the USART\_STS.TXC flag is not affected by the smart card mode.

The following figure details how USART samples NACK signals.

Figure 18-23 Use 1.5 stop bits to detect parity errors



## 18.5 Interrupt request

The various interrupt events of USART are logical OR relations, if the corresponding enable control bit is set, these events can generate their own interrupts, but only one interrupt request can be generated at the same time.

Table 18-7 USART interrupt request

Interrupt function	Interrupt event	Event flag	Enable bit
USART global interrupt	Transmission data register is empty.	TXDE	TXDEIEN
	CTS flag	CTSF	CTSIEN
	Transmission complete	TXC	TXCIEN
	Receive data ready to be read	RXDNE	RXDNEIEN
	Data overrun error detected.	ORERR	
	Idle line detected	IDLEF	IDLEIEN
	Parity error	PEF	PEIEN
	Disconnect flag	LINBDF	LINBDIEN

	Noise, overrun error and framing error in multi-buffer communication	NEF/OREF/FEF	ERRIEN <sup>(1)</sup>
--	--	--------------	-----------------------

(1) This flag bit is used only when DMA is used to receive data(USART\_CTRL3.DMARXEN=1).

## 18.6 Mode support

Table 18-8 USART mode setting<sup>(1)</sup>

Communication mode	USART1	USART2
Asynchronous mode	Y	Y
Hardware flow control mode	Y	Y
DMA communication mode	Y	Y
Multiprocessor	Y	Y
Synchronous mode	Y	Y
Smartcard mode	Y	Y
Single-wire half duplex mode	Y	Y
IrDA infrared mode	Y	Y
LIN	Y	Y

(1) Y = support this mode, N = do not support this mode

## 18.7 USART register

### 18.7.1 USART register overview

Table 18-9 USART register overview

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
000h	USART_STS	Reserved																CTSF	9	LINBDF	8	TXDE	7	TXC	6	RXDNE	5	IDLEF	4	OREF	3	NEF	2	FEF	1	PEF	0		
	Reset Value																	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
004h	USART_DAT	Reserved																DATV[8:0]																					
	Reset Value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
008h	USART_BRCF	Reserved								DIV_Integer[11:0]												DIV_Decimal[3:0]																	
	Reset Value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

### 18.7.2 USART Status register (USART\_STS)

Address offset : 0x00

Reset value : 0x0000 00C0

31															16
Reserved															
15				10	9	8	7	6	5	4	3	2	1	0	
				Reserved		CTSF	LINBDF	TXDE	TXC	RXDNE	IDLEF	OREF	NEF	FEF	PEF
				rc	w0	rc	w0	r	rc	w0	rc	w0	r	r	r

Bit field	Name	Description
31:10	Reserved	Reserved, the reset value must be maintained
9	CTSF	<p>CTS flag</p> <p>If USART_CTRL3.CTSEN bit is set, this bit is set by hardware when the nCTS input changes. If USART_CTRL3.CTSIEN bit is set, an interrupt will be generated.</p> <p>This bit is cleared by software.</p> <p>0:nCTS status line has not changed.</p> <p>1:nCTS status line changes.</p>
8	LINBDF	<p>LIN break detection flag.</p> <p>If USART_CTRL2.LINMEN bit is set, this bit is set by hardware when LIN disconnection is detected. If USART_CTRL2.LINBDIEN bit is set, an interrupt will be generated.</p> <p>This bit is cleared by software.</p> <p>0: LIN break character not detected.</p> <p>1: LIN break character detected.</p>
7	TXDE	The Transmit data register empty.

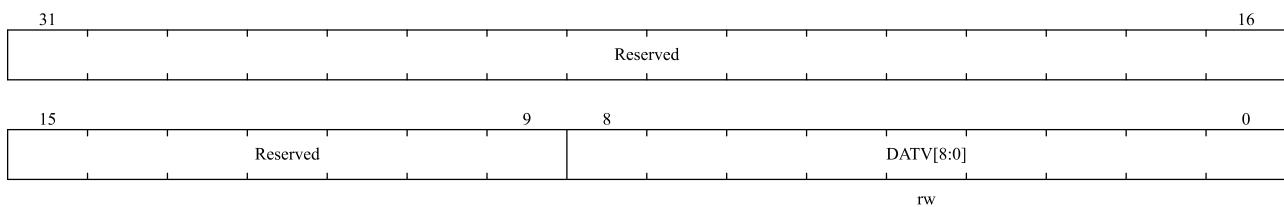
Bit field	Name	Description
		<p>Set to 1 after power-on reset or data to be sent has been sent to the shift register. Setting USART_CTRL1.TXDEIEN will generate an interrupt.</p> <p>This bit is cleared to 0 when the software writes the data to be sent into USART_DAT.</p> <p>0: Send data buffer is not empty. 1: The transmitting data buffer is empty.</p>
6	TXC	<p>Transmission complete.</p> <p>This bit is set to 1 after power-on reset. If USART_STS.TXDE is set, this bit is set when the current data transmission is completed.</p> <p>Setting USART_CTRL1.TXCIEN bit will generate an interrupt.</p> <p>This bit is cleared by software.</p> <p>0: Transmitting did not complete. 1: Send completed.</p>
5	RXDNE	<p>The Read data register not empty.</p> <p>This bit is set when the read data buffer receives data from the shift register. When USART_CTRL1.RXDNEIEN bit is set, an interrupt will be generated.</p> <p>Software can clear this bit by writing 0 to it or reading the USART_DAT register.</p> <p>0: The read data buffer is empty. 1: The read data buffer is not empty.</p>
4	IDLEF	<p>IDLE line detected flag.</p> <p>Within one frame time, the idle state is detected at the RX pin, and this bit is set to 1. When USART_CTRL1.IDLEIEN bit is set, an interrupt will be generated.</p> <p>The software can clear this bit by reading USART_STS first and then reading USART_DAT.</p> <p>0: No idle frame detected. 1: idle frame detected.</p> <p><i>Note: IDLEF bit will not be set high again until USART_STS.RXDNE bit is set (that is, an idle line is detected again).</i></p>
3	OREF	<p>Overrun error</p> <p>With RXDNE set, this bit is set if the USART_DAT register receives data from the shift register. When USART_CTRL3.ERRIEN bit is set, an interrupt will be generated.</p> <p>The software can clear this bit by reading USART_STS first and then reading USART_DAT.</p> <p>0: No overrun error was detected. 1: Overflow error detected.</p>
2	NEF	<p>Noise error flag.</p> <p>When noise is detected in the received frame, this bit is set by hardware. It is cleared by the software sequence (read first USART_STS, read USART_DAT again).</p> <p>0: No noise error detected. 1: Noise error detected.</p> <p><i>Note: this bit will not generate an interrupt because it appears with USART_STS.RXDNE, and the hardware will generate an interrupt when setting the</i></p>

Bit field	Name	Description
		<p><i>USART_STS.RXDNE flag. In the multi-buffer communication mode, if the USART_CTRL3.ERRIEN bit is set, an interrupt will be generated when the NEF flag is set.</i></p>
1	FEF	<p>Framing error. When the data is not synchronized or a large amount of noise is detected, and the stop bit is not received and recognized at the expected time, it will be judged that a framing error has been detected, and this bit will be set to 1. First read USART_STS, then read USART_DAT can cleared this bit.</p> <p>0: No framing errors were detected. 1: A framing error or a Break Character is detected.</p> <p><i>Note: this bit will not generate an interrupt because it appears with USART_STS.RXDNE, and the hardware will generate an interrupt when setting the USART_STS.RXDNE flag. If the currently transmitted data has both framing errors and overload errors, the hardware will continue to transmit the data and only set the USART_STS.OREF flag bit.</i></p> <p><i>In the multi-buffer communication mode, if the USART_CTRL3.ERRIEN bit is set, an interrupt will be generated when the FEF flag is set.</i></p>
0	PEF	<p>Parity error. This bit is set when the parity bit of the received data frame is different from the expected check value. The software can clear this bit by reading USART_STS first and then reading USART_DAT.</p> <p>0: No parity error was detected. 1: Parity error detected.</p>

### 18.7.3 USART Data register (USART\_DAT)

Address offset : 0x04

Reset value : undefined (uncertain value)



Bit field	Name	Description
31:9	Reserved	Reserved, the reset value must be maintained

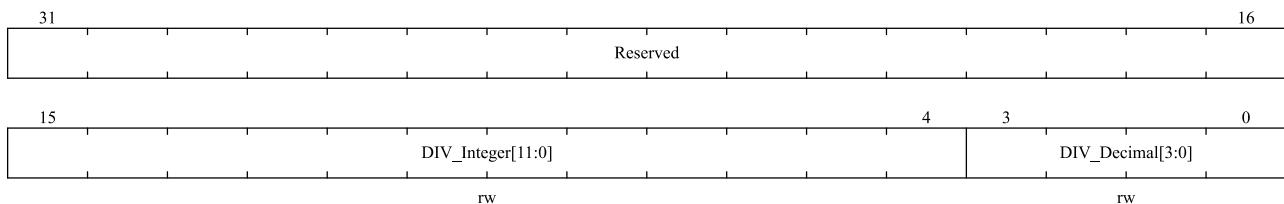
Bit field	Name	Description
8:0	DATV[8:0]	<p>Data value</p> <p>Contains the data sent or received; Software can change the transmitted data by writing these bits, or read the values of these bits to obtain the received data.</p> <p>If parity is enabled, when the transmitted data is written into the register, the highest bit of the data (the 7th or 8th bit depends on USART_CTRL1.WL bit) will be replaced by the parity bit.</p>

#### 18.7.4 USART Baud rate register (USART\_BRCF)

Address offset : 0x08

Reset value : 0x0000 0000

*Note: When USART\_CTRL1.UEN=1, this register cannot be written; The baud counter stops counting if USART\_CTRL1.TXEN or USART\_CTRL1.RXEN are disabled respectively.*

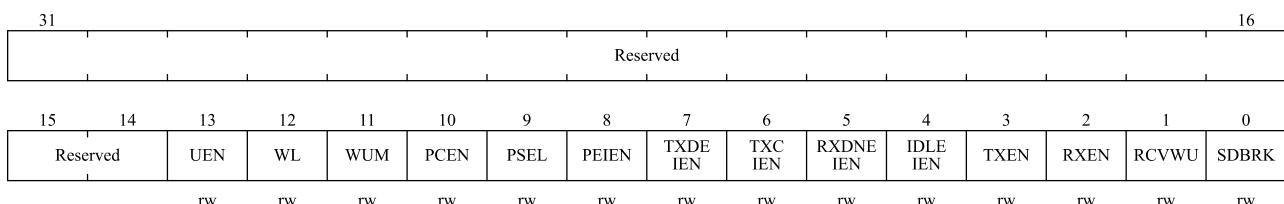


Bit field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained
15:4	DIV_Integer [11:0]	Integer part of baud rate divider.
3:0	DIV_Decimal[3:0]	Fractional part of baud rate divider.

#### 18.7.5 USART control register 1 register (USART\_CTRL1)

Address offset : 0x0C

Reset value : 0x0000 0000



Bit field	Name	Description
31:14	Reserved	Reserved, the reset value must be maintained
13	UEN	<p>USART enable</p> <p>When this bit is cleared, the divider and output of USART stop working after the current byte transmission is completed to reduce power consumption. Software can set or clear</p>

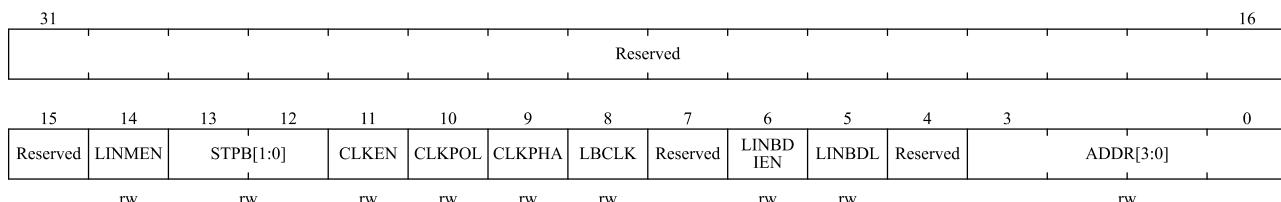
Bit field	Name	Description
		this bit. 0:USART is disabled. 1:USART is enabled.
12	WL	Word length. 0:8 data bits. 1:9 data bits. <i>Note: If data is in transfer, this bit cannot be configured.</i>
11	WUM	Wake up mode from mute mode. 0: Idle frame wake up. 1: Address identifier wake up.
10	PCEN	Parity control enable 0: Parity control is disabled. 1: Parity control is enabled.
9	PSEL	Parity selection. 0: even check. 1: odd check.
8	PEIEN	PE interrupt enable If this bit is set to 1, an interrupt is generated when USART_STS.PEF bit is set. 0: Parity error interrupt is disabled. 1: Parity error interrupt is enabled.
7	TXDEIEN	TXDE interrupt enable If this bit is set to 1, an interrupt is generated when USART_STS.TXDE bit is set. 0: Send buffer empty interrupt is disabled. 1: Send buffer empty interrupt is enabled.
6	TXCIEN	Transmit complete interrupt enable. If this bit is set to 1, an interrupt is generated when USART_STS.TXC is set. 0: Transmission completion interrupt is disabled. 1: Transmission completion interrupt is enabled.
5	RXDNEIEN	RXDNE interrupt enable If this bit is set to 1, an interrupt is generated when USART_STS.RXDNE or USART_STS.OREF is set. 0: Data buffer non-empty interrupt o and overrun error interrupt are disabled. 1: Data buffer non-empty interrupt o and overrun error interrupt are enabled.
4	IDLEIEN	IDLE interrupt enable. If this bit is set to 1, an interrupt is generated when USART_STS.IDLEF is set. 0:IDLE line detection interrupt is disabled. 1: IDLE line detection interrupt is enabled.
3	TXEN	Transmitter enable. 0: The transmitter is disabled. 1: the transmitter is enabled.
2	RXEN	Receiver enable

Bit field	Name	Description
		0: The receiver is disabled. 1: the receiver is enabled.
1	RCVWU	The receiver wakes up  Software can set this bit to 1 to make USART enter mute mode, and clear this bit to 0 to wake up USART.  In idle frame wake-up mode (USART_CTRL1.WUM=0), this bit is cleared by hardware when an idle frame is detected. In address wake-up mode (USART_CTRL1.WUM=1), when an address matching frame is received, this bit is cleared by hardware. Or when an address mismatch frame is received, it is set to 1 by hardware. 0: The receiver is in normal operation mode. 1: The receiver is in mute mode.
0	SDBRK	Send Break Character.  The software transmits a break character by setting this bit to 1. This bit is cleared by hardware during stop bit of the break frame transmission. 0: No break character was sent. 1: Send a break character.

### 18.7.6 USART control register 2 register (USART\_CTRL2)

Address offset : 0x10

Reset value : 0x0000 0000



Bit field	Name	Description
31:15	Reserved	Reserved, the reset value must be maintained
14	LINMEN	LIN mode enable 0:LIN mode is disabled 1:LIN mode enabled
13:12	STPB[1:0]	STOP bits. 00:1 stop bit. 01:0.5 stop bit. 10:2 stop bit. 11:1.5 stop bit.
11	CLKEN	Clock enable 0:CK pin is disabled 1:CK pin enabled

Bit field	Name	Description
10	CLKPOL	<p>Clock polarity.</p> <p>This bit is used to set the polarity of CK pin in synchronous mode.</p> <p>0: CK pin remains low when it is not transmitted to the outside.</p> <p>1: CK pin remains high when it is not sent to the outside.</p>
9	CLKPHA	<p>Clock phase.</p> <p>This bit is used to set the phase of CK pin in synchronous mode.</p> <p>0: Sample the first data at the first clock edge.</p> <p>1: Sample the first data at the second clock edge.</p>
8	LBCLK	<p>The Last bit clock pulse.</p> <p>This bit is used to set whether the clock pulse corresponding to the last transmitted data byte (MSB) is output on CK pin in synchronous mode.</p> <p>0: The clock pulse of the last bit of data is not output from CK.</p> <p>1: The clock pulse of the last bit of data will be output from CK.</p>
7	Reserved	Reserved, the reset value must be maintained
6	LINBDIEN	<p>LIN break detection interrupt enable.</p> <p>If this bit is set to 1, an interrupt will be generated when USART_STS.LINBDF bit is set.</p> <p>0: Disconnect signal detection interrupt is disabled.</p> <p>1: Turn-off signal detection interrupt enabled</p>
5	LINBDL	<p>LIN break detection length.</p> <p>This bit is used to set the length of the break frame.</p> <p>0:10 bit break detection</p> <p>1:11 bit break detection</p> <p><i>Note: LINBDL can be used to control the detection length of Break Characters in LIN mode and other modes, and the detection length is the same as that in LIN mode.</i></p>
4	Reserved	Reserved, the reset value must be maintained
3:0	ADDR[3:0]	<p>USART address.</p> <p>Used in the mute mode of multiprocessor communication, using address identification to wake up a USART device.</p> <p>In address wake-up mode (USART_CTRL1.WUM=1), if the lower four bits of the received data frame are not equal to the ADDR[3:0] value, USART will enter the mute mode; If the lower four bits of the received data frame are equal to the ADDR[3:0] value, USART will be awakened.</p>

*Note: These three bits (USART\_CTRL2.CLKPOL, USART\_CTRL2.CLKPHA, USART\_CTRL2.LBCLK) cannot be overwritten after enabling transmission.*

### 18.7.7 USART control register 3 register (USART\_CTRL3)

Address offset : 0x14

Reset value : 0x0000 0000

31	Reserved												16
15	Reserved	CTS IEN	CTSEN	RTSEN	DMA TXEN	DMA RXEN	SC MEN	SC NACK	HDM EN	IRDA LP	IRDA MEN	ERR IEN	0

rw      rw

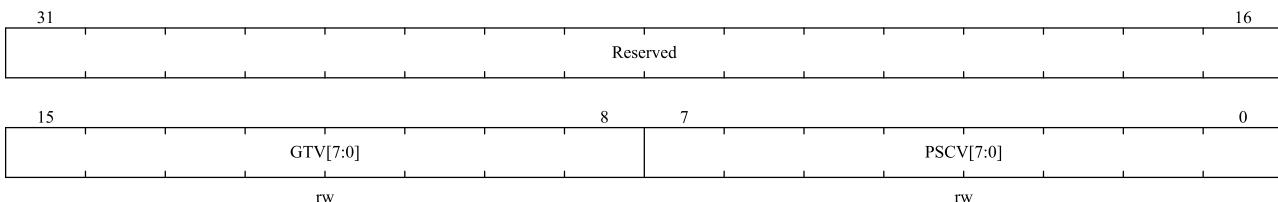
Bit field	Name	Description
31:11	Reserved	Reserved, the reset value must be maintained
10	CTSIEN	<p>CTS interrupt enable.</p> <p>If this bit is set to 1, an interrupt will be generated when USART_STS.CTSF bit is set.</p> <p>0:CTS interrupt is disabled.</p> <p>1:CTS interrupt is enabled.</p>
9	CTSEN	<p>CTS enable.</p> <p>This bit is used to enable the CTS hardware flow control function.</p> <p>0:CTS hardware flow control is disabled.</p> <p>1:CTS hardware flow control is enabled.</p>
8	RTSEN	<p>RTS enable.</p> <p>This bit is used to enable RTS hardware flow control function.</p> <p>0:RTS hardware flow control is disabled.</p> <p>1:RTS hardware flow control is enabled.</p>
7	DMATXEN	<p>DMA transmitter enable.</p> <p>0:DMA transmission mode is disabled.</p> <p>1:DMA transmission mode is enabled.</p>
6	DMARXEN	<p>DMA receiver enable.</p> <p>0:DMA receive mode is disabled.</p> <p>1:DMA receive mode is enabled.</p>
5	SCMEN	<p>Smartcard mode enable.</p> <p>This bit is used to enable Smartcard mode.</p> <p>0: Smartcard mode is disabled.</p> <p>1: Smartcard mode is enabled.</p>
4	SCNACK	<p>Smartcard NACK enable.</p> <p>This bit is used for Smartcard mode to enable transmitting NACK when parity error occurs.</p> <p>0: Do not send NACK when there is a parity error.</p> <p>1: send NACK when there is a parity error.</p>
3	HDMEN	<p>Half-duplex mode enable.</p> <p>This bit is used to enable half-duplex mode.</p> <p>0: Half-duplex mode is disabled.</p> <p>1: Half-duplex mode is enabled.</p>
2	IRDALP	<p>IrDA low-power mode.</p> <p>This bit is used to select the low power consumption mode for IrDA mode.</p>

Bit field	Name	Description
		0: Normal mode. 1: Low power mode.
1	IRDAMEN	IrDA mode enable. 0:IrDA is disabled. 1:IrDA is enabled.
0	ERRIEN	Error interrupt enable. When DMA receive mode (USART_CTRL3.DMARXEN=1) is enabled, an interrupt will be generated when USART_STS.FEF, USART_STS.OREF or USART_STS.NEF bit is set. 0: Error interrupt is disabled. 1: Error interrupt enabled.

### 18.7.8 USART guard time and prescaler register (USART\_GTP)

Address offset : 0x18

Reset value : 0x0000 0000



Bit field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained
15:8	GTV[7:0]	Guard time value in Smartcard mode. This bit field specifies the guard time in baud clock. In Smartcard mode, this function is required. The setting time of USART_STS.TXC flag is delayed by GTV[7:0] baud clock cycles.
7:0	PSCV[7:0]	Prescaler value. In IrDA low power consumption mode: these bits are used to set the frequency division coefficient for dividing the peripheral clock (PCLK1/PCLK2) to generate low power consumption frequency. 00000000: reserved-do not write this value. 00000001: divide the source clock by 1. ... 11111111: divide the source clock by 255. In IrDA normal mode: PSCV can only be set to 00000001. In Smartcard mode: PSCV[4:0] is used to set the frequency division of Smartcard clock generated by

Bit field	Name	Description
		<p>peripheral clock (PCLK1/ PCLK2).</p> <p>Coefficient. The actual frequency division coefficient of is twice the set value of PSCV[4:0].</p> <p>0000: reserved-do not write this value.</p> <p>0001: Divide the source clock by 2.</p> <p>0010: Divide the source clock by 4.</p> <p>...</p> <p>1111: Divide the source clock by 62.</p> <p>In Smartcard mode, PSCV[7:5] is reserved.</p>

## 19 Low power universal asynchronous receiver transmitter (LPUART)

### 19.1 Introduction

Low power universal asynchronous receiver transmitter (LPUART) is a low power, full duplex, asynchronous serial communication interface. The LPUART can be clock provided by HSI, HSE, LSI, LSE, SYSCLK and PCLK1. When 32.768khz LSE is selected as the clock source, the LPUART can work in STOP low-power mode with a maximum communications up to 9600bps. LPUART supports receiving data wake-up. By configuring wake-up events, the CPU in STOP2 mode can be woken up.

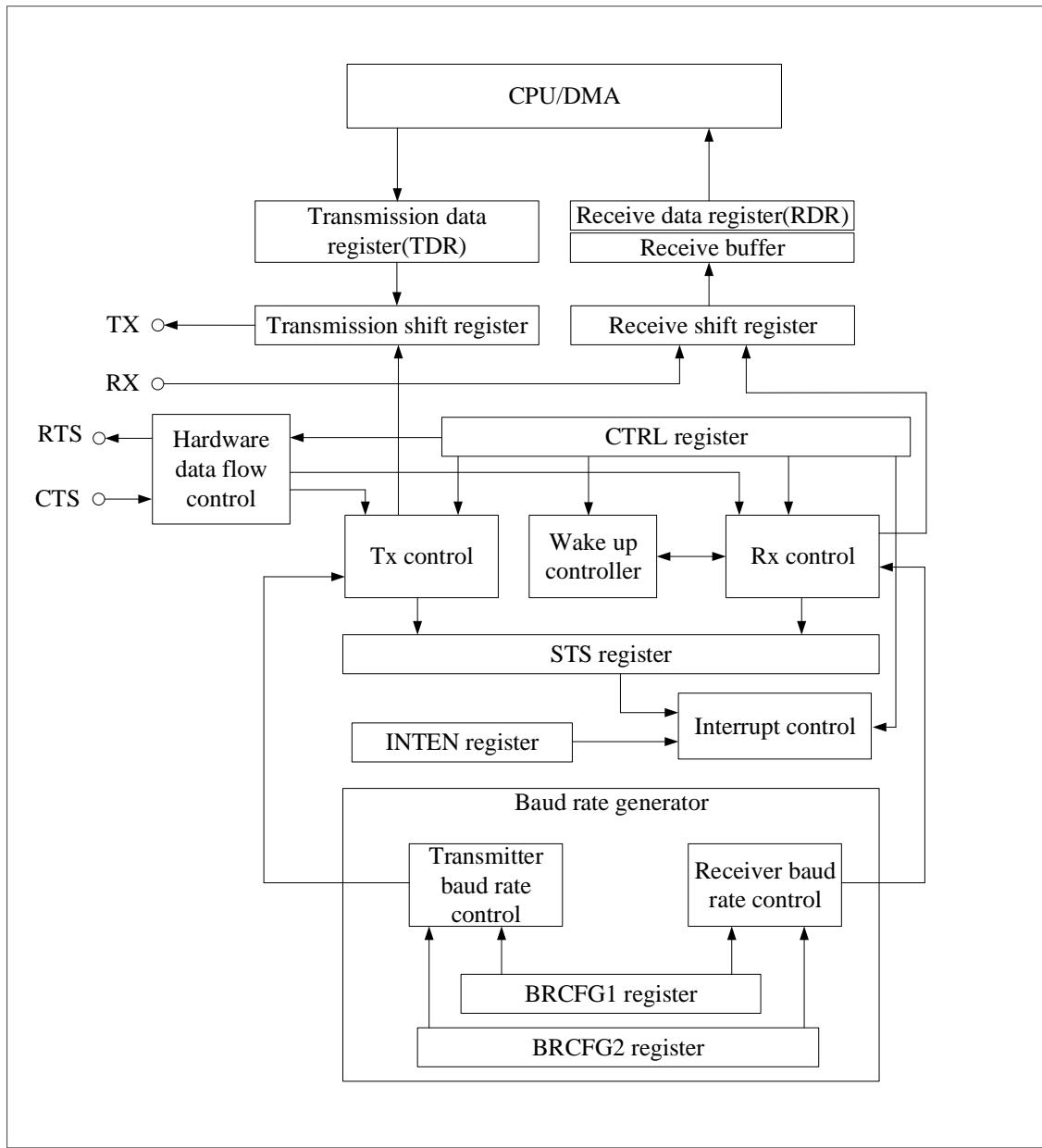
At the same time, when MCU works in RUN mode, LPUART can also be used as a common asynchronous serial port. Users can switch the clock source to HSI, HSE, SYSCLK and PCLK1 to obtain higher communication speed.

### 19.2 Main features

- Full duplex asynchronous communication
- Selectable clock source of HSI, HSE, LSI, LSE, SYSCLK, or PCLK1
- Fractional baud rate generator system: Programmable baud rate shared by sending and receiving up to 1Mbits/s, baud rates from 300bps to 9600bps when using 32.768 kHz clock source (LSE)
- Fixed 8-bit data word length, 1 stop bit and optional 1 parity bit
- Support DMA data transfer
- Support hardware flow control
- Transfer detection flag: Receive buffer full, Receive buffer half full, Receive buffer not empty, Receive buffer overrun, Transmission complete
- Parity control: Odd and even parity selection, Parity can be disable
- Error detection flag: Parity error, Overrun error, Noise error
- 32 byte receive buffer
- Baud rate error correction at low frequencies
- Configurable sampling method of 1 or 3 samples
- Noise detection
- Configurable flow control RTS threshold
- Support STOP mode Configurable source mode
  - ◆ Start bit detection
  - ◆ Receive buffer non-empty detection
  - ◆ A configurable receive byte
  - ◆ A programmable 4-byte frame

## 19.3 Functional block diagram

Figure 19-1 LPUART block diagram



## 19.4 Function description

As shown in Figure 19-1, LPUART bidirectional communication requires at least two pins: receiving data input (RX) and sending data output (TX).

**RX:** Serial data input. When the number of samples is 3, data and noise can be distinguished.

**TX:** Serial data output. When sending is enabled, the pin defaults to be high level.

The following pins are required in hardware flow control mode:

**CTS (Clear To Send):** When transmitter detects that CTS is valid (low level), the next data is sent.

**RTS (Request To Send):** When receiver is ready to receive new data, pull the RTS pin low.

LPUART has the following characteristics:

- Idle status without sending or receiving
- A start bit
- A data word (8 bits) with the least significant bits first
- A stop bit, indicating the end of a data frame
- A status register (LPUART\_STS)
- Data register (LPUART\_DAT)
- Two baud rate configuration registers (LPUART\_BRCFG1 and LPUART\_BRCFG2) using fractional baud rate generators: 16-bit integer and 8-bit decimal representations

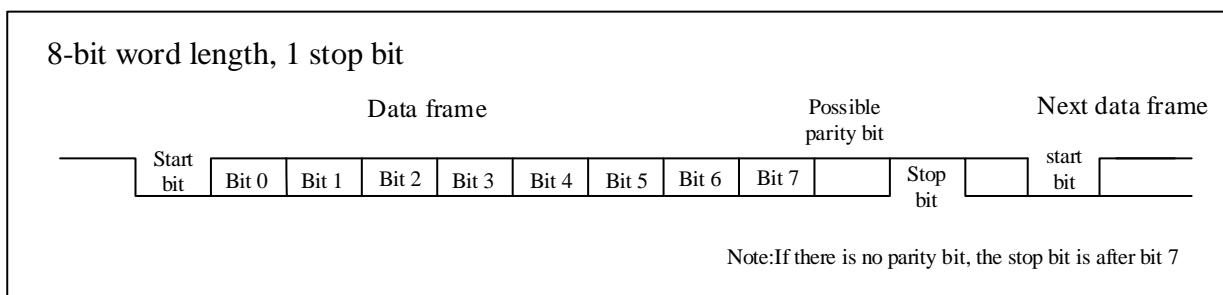
For specific definitions of each bit in the registers above, please refer to Section 19.6 of register Description.

### 19.4.1 LPUART frame format

The LPUART data word length is fixed at 8 bits (see Figure 19-2). During the start bit, TX pin is at a low level and during the stop bit it is at a high level. The parity bit follows the data word when enabled.

Both sending and receiving are driven by two different baud clock generators. When the LPUART\_CTRL.TXEN of transmitter is set, the corresponding baud clock generator generates baud clock. When the start bit is received, the receiver's corresponding baud clock generator generates the clock.

Figure 19-2 frame format



*Note: in this chapter, unless special instruction, setting means that a register is set to state '1', and resetting or clearing means that a register is set to state '0'. Hardware or programs may set or clear a register. Please refer to this chapter for details.*

### 19.4.2 Transmitter

When the Transmit Enable bit (LPUART\_CTRL.TXEN) is set and there is data in the buffer, the transmitter sends 8-bit data words. The data in the shift register is output on the TX pin.

### 19.4.2.1 Transmi process

During an LPUART transmission, the least significant bit of the data is shifted out on TX pin. In this mode, the LPUART\_DAT register contains a buffer between the internal bus and the transmitter shift register (see Figure 19-1).

Each character is preceded by a low level starting bit; and is terminated by a stop bit.

*Note: You cannot reset the LPUART\_CTRL.TXEN bit during data transfer; otherwise the data on the TX pin will be corrupted because the baud rate counter stops counting. The current data being transferred will be lost.*

The LPUART sends data as follows:

1. Configure baud rate, parity check, DMA, flow control, etc.
2. Set the LPUART\_CTRL.TXEN bit to enable data transmission.
3. Write data to the LPUART\_DAT register.
4. Check if the LPUART\_STS.TXC flag is set, it means the transmission is over. If the flag is set, write 1 to the LPUART\_STS.TXC bit to clear the flag.
5. Check the LPUART\_STS.PEF bit to confirm whether the parity is wrong.
6. Otherwise, go to Step 3 and send the next data.

*Note: Be sure to initialize the LPUART module before using the transmitter.*

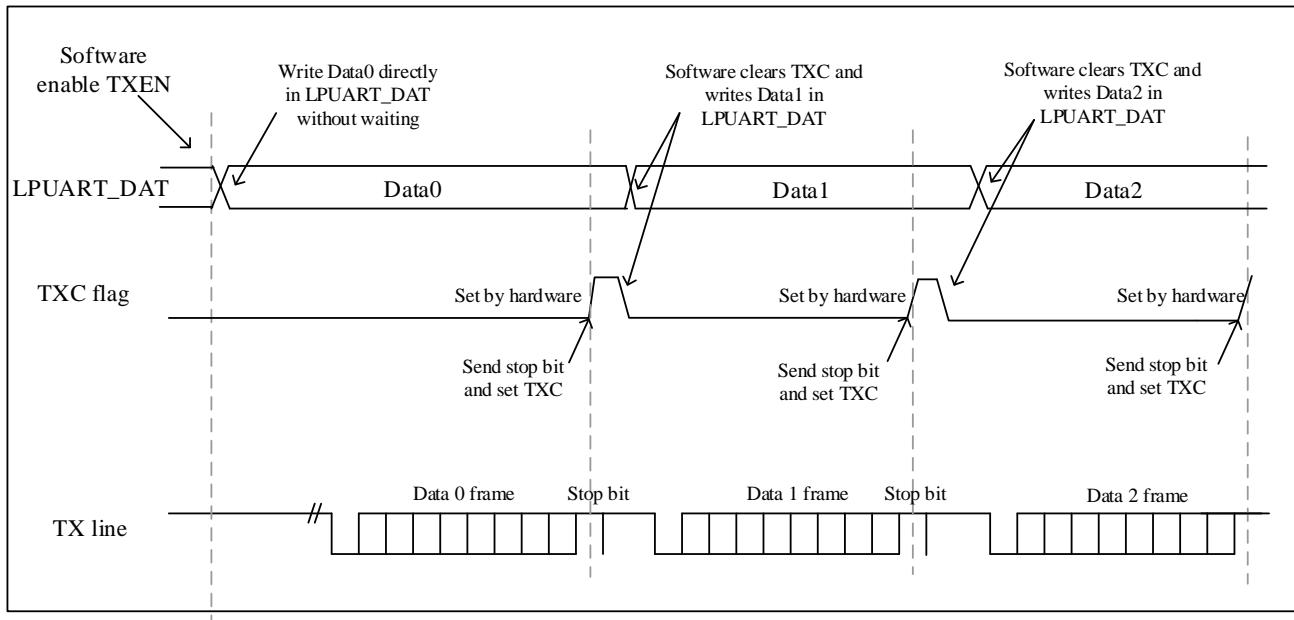
LPUART initialization as follows:

1. Set all flag bits in the LPUART\_STS register to clear the interrupt flag.
2. To enable the interrupt function, configure LPUART\_INTEN.
3. Set LPUART\_CTRL.FLUSH clear the RX buffer.

When send data:

- After configuring the baud rate and setting LPUART\_CTRL.TXEN, the CPU can write directly to the LPUART\_DAT register to send data.
- When a frame transmission is completed (after the stop bit is sent), the LPUART\_STS.TXC bit is set. If the LPUART\_INTEN.TXCIEN bit is set, an interrupt occurs immediately.
- After the last data byte is written to the LPUART\_DAT register, you must wait for LPUART\_STS.TXC=1 before shutting down the LPUART module or setting the microcontroller into low-power mode.

Figure 19-3 TXC changes during transmission



## 19.4.3 Receiver

### 19.4.3.1 Start bit detection

If the LPUART\_CTRL.SMPCNT bit is 0, that is, the number of samples is 3, when there are at least two 0s in the three sample numbers, the start bit is valid. Otherwise it will be invalid.

Sampling values	NF state	Received bit value	Start bit validity
000	0	0	effective
001	1	0	effective
010	1	0	effective
011	1	1	invalid
100	1	0	effective
101	1	1	invalid
110	1	1	invalid
111	0	1	invalid

### 19.4.3.2 Receive process

During LPUART reception, the least significant bits of data are first moved in from the RX pin. In this mode, the LPUART\_DAT register contains a buffer between the internal APB bus and the receive shift register.

The steps for LPUART to receive data are as follows:

1. Configure baud rate, parity check, wake up event/enable, sampling mode, DMA, flow control, etc.
2. Check the interrupt flags of the LPUART\_STS register: buffer is not empty, buffer is half full, buffer is full, buffer overrun;
3. Read the data by reading the LPUART\_DAT register.
4. Return to Step 2 and continue receiving data.

*Note: Please be sure to initialize the LPUART module before using the receiver.*

When receiving a data frame:

- The LPUART\_STS.FIFO\_NE bit is set, and the contents of the shift Register are transferred to the RDR (Receiver Data Register). In other words, the data has been received and can be read (including its associated error flags).
- If the LPUART\_INTEN.FIFO\_NEIEN bit is set, an interrupt is generated.
- Frame errors (parity detection errors), noise or overrun errors are detected during reception, so the error flag will be set.
- In multi-buffer communication mode, the LPUART\_STS.FIFO\_NE flag bit is placed after each byte received and cleared by DMA's read operation on the data register.
- In single buffer mode, the software can clear LPUART\_STS.FIFO\_NE bits by reading the LPUART\_DAT register or by writing 0. The LPUART\_STS.FIFO\_NE bit must be cleared before the end of the next frame of data reception to avoid overrun errors.

#### 19.4.3.3 Overrun error

The LPUART receiving data buffer has a total of 32 bytes. The LPUART\_STS.FIFO\_FU flag will be set after receiving 32 bytes of data. When the buffer data is not read out and causes LPUART\_STS.FIFO\_FU to be not reset in time, if next character is received, an overrun error occurs. This character will be discarded by the hardware. Data can only be transferred from the shift register to the receiving data buffer if the LPUART\_STS.FIFO\_FU bit is cleared. If the next data has been received or the previous DMA request has not been served, the LPUART\_STS.FIFO\_FU flag is still set and an overrun error occurs.

When an overrun error occurs:

- The LPUART\_STS.FIFO\_OV bit is set.
- The receiving data buffer content will not be lost. Reading the LPUART\_DAT register still returns the previous data.
- The contents of the shift register will be overwritten. Any subsequent data received will be lost.
- If the LPUART\_INTEN.FIFO\_OVIE bit is set, an interrupt is generated.
- LPUART\_DAT register read operation, reset LPUART\_STS.FIFO\_OV.

#### 19.4.3.4 Noise error

Noise errors use an over-sampling technique (if the LPUART\_CTRL.SMPCNT bit is 0, that is, the number of samples is 3) to recover data by distinguishing valid input data from noise.

Figure 19-4 Data sampling for noise detection

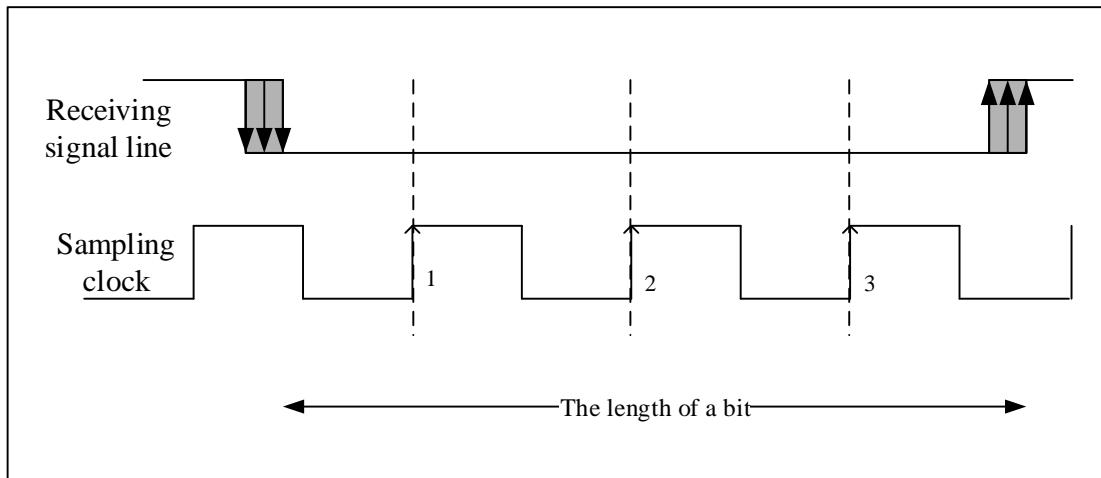


Table 19-1 Data sampling for noise detection

Sampling values	NF state	Received bit value
000	0	0
001	1	0
010	1	0
011	1	1
100	1	0
101	1	1
110	1	1
111	0	1

When noise is detected in a receiving frame, you can do the following:

- If three sample values are inconsistent, set the LPUART\_STS.NF flag immediately.
- The received data is transferred from the shift register to the buffer.
- Software write 1 clears the LPUART\_STS.NF flag bit.

#### 19.4.4 Fractional baud rate generation

Baud rate frequency division coefficient is divided into 16-bit integer part and 8-bit decimal part. The baud rate generator uses the value of the combination of these two parts to determine the baud rate. The fractional baud rate divider will enable the LPUART to generate all standard baud rates.

Baud rate frequency division coefficient (LPUARTDIV) has the following relationship with system clock (PCLK) :

$$\text{TX/RX baud rate} = f_{\text{CLK}} / (\text{LPUARTDIV})$$

Here the  $f_{CLK}$  is the clock for LPUART (the clock source of LPUART can be HSI, HSE, LSI, LSE, SYSCLK, or PCLK1). The value of LPUARTDIV is set in the baud rate configuration registers LPUART\_BRCFG1 and LPUART\_BRCFG2

*Note: After writing LPART\_BRCFG1 and LPUART\_BRCFG2, the baud rate counter is replaced with the new value of the baud rate register. Therefore, do not change the value of the baud rate register during communication.*

#### 19.4.4.1 Configure baud rates through LPUART\_BRCFG1 and LPUART\_BRRCFG2

For example, baud rate = 4800bps, clock frequency = 32768Hz.

LPUARTDIV =  $32768/4800 = 6.82667$ .LPUART\_BRCFG1 = 6 and the value of LPUART\_BRCFG2 is calculated by adding fractions in the table below (the value of LPUART\_BRCFG2 is 0xEFh).

Decimal addition	Carry to the next integer	Bit field	Value
$0.82667 + 0.82667 = 1.65333$	YES	DECIMAL0	1
$1.65333 + 0.82667 = 2.48000$	YES	DECIMAL1	1
$2.48000 + 0.82667 = 3.30667$	YES	DECIMAL2	1
$3.30667 + 0.82667 = 4.13333$	YES	DECIMAL3	1
$4.13333 + 0.82667 = 4.96000$	NO	DECIMAL4	0
$4.96000 + 0.82667 = 5.78667$	YES	DECIMAL5	1
$5.78667 + 0.82667 = 6.61333$	YES	DECIMAL6	1
$6.61333 + 0.82667 = 7.44000$	YES	DECIMAL7	1

When LSE clock (32.768KHz) is used, the values of baud rate configuration registers LPUART\_BRCFG1 and LPUART\_BRCFG2 with different baud rate Settings are as follows:

Baud rate	Divisor	LPUART_BRCFG1	LPUART_BRCFG2
300	109.2267	6Dh	88h
600	54.6133	36h	ADh
1200	27.3067	1Bh	24h
2400	13.6533	0Dh	6Dh
4800	6.8267	06h	EFh
9600	3.4133	03h	4Ah

*Note: The lower the clock frequency of the CPU, the lower the accuracy of a particular baud rate.*

### 19.4.5 Parity control

Reset the LPUART\_CTRL.PCDIS bit, enable parity control (generate a parity bit when sending, parity check when receiving), set or reset the LPUART\_CTRL.PSEL bit selection to use odd or even check. LPUART frame formats are listed in the table below.

Table 19-2 Parity frame format

PCDIS bit	LPUART frame
0	Start bit   8-bit data   parity bit   stop bit
1	Start bit   8 bits data   stop bit

Transfer mode: Parity is enabled by resetting the LPUART\_CTRL.PCDIS bit. If parity fails, the LPUART\_STS.PEF flag is set to '1', and an interrupt occurs if LPUART\_INTEN.PEIE is set.

Odd parity: LPUART\_CTRL.PSEL=1.

Make the number of '1' in one frame data (including parity bit) be an odd number. That is: if Data=11000101, there are 4 '1's, then the parity bit will be '1' (5 '1' in total).

Even parity: LPUART\_CTRL.PSEL=0.

Make the number of '1' in one frame data (including parity bit) be an even number. That is: if Data=11000101, there are 4 '1's, then the parity bit will be '0' (4 '1' in total).

### 19.4.6 DMA application

LPUART can access the transmit data register (TDR) and receive buffer respectively through DMA.

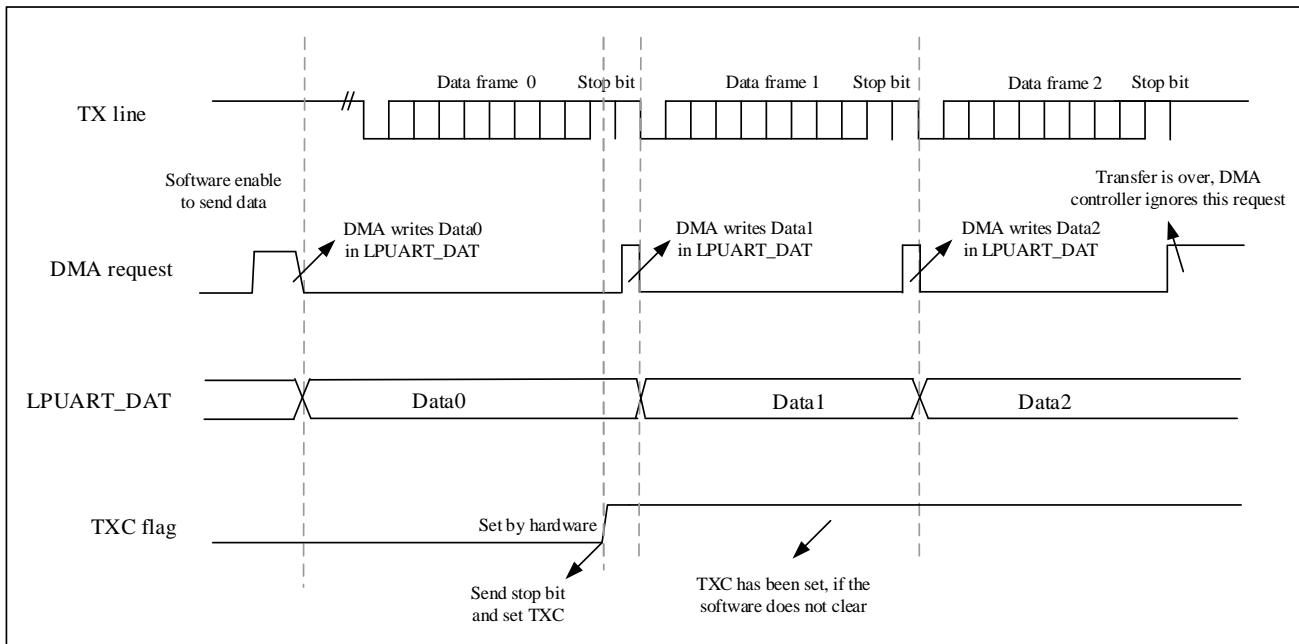
#### 19.4.6.1 DMA transmission

The steps for assigning a DMA channel to the LPUART transmissions are as follows (x indicates the channel number) :

1. Configure the LPUART\_DAT register address as the destination address for DMA transfer, and the memory address as the source address for DMA transfer.
2. Set the total number of bytes to be transmitted.
3. Set the channel priority.
4. Configure to generate DMA interrupts when the transfer is half or all complete.
5. Activate the channel.

Completing a DMA transfer will generate an interrupt on the corresponding DMA channel. In transmission mode, when the DMA has finished the data transfer, the DMA controller sets the DMA\_INTSTS.TXCFx flag. The LPUART\_STS.TXC flag bit is asserted by the hardware to indicate that the transfer is completed. The software needs wait for LPUART\_STS.TXC=1.

Figure 19-5 Sending using DMA



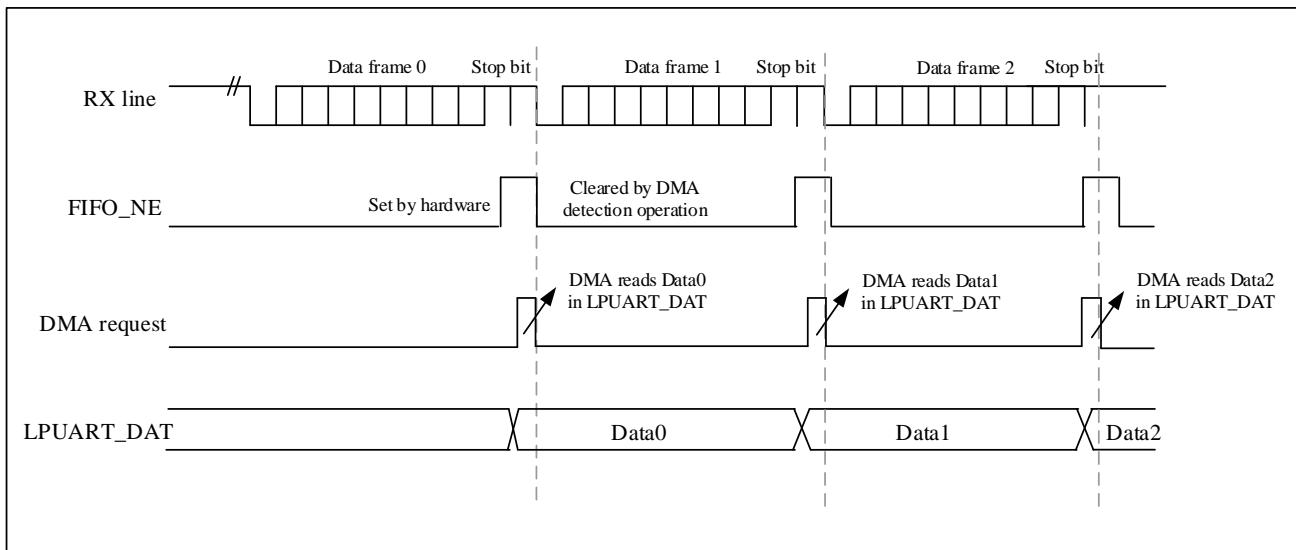
#### 19.4.6.2 DMA reception

The steps for assigning a DMA channel to the LPUART receiving are as follows (x indicates the channel number) :

1. Configure the LPUART\_DAT register address as the source address for transmission and the memory address as the destination address for transmission through the DMA configuration register.
2. Configure the number of DMA bytes to be transferred.
3. Configure the channel priority on the DMA register for data transfer.
4. Configure interrupts to generate DMA interrupts when the transfer is half or all complete.
5. Activate the channel.

When completing the transfer specified by the DMA controller, the DMA controller generates an interrupt on the DMA channel's interrupt vector.

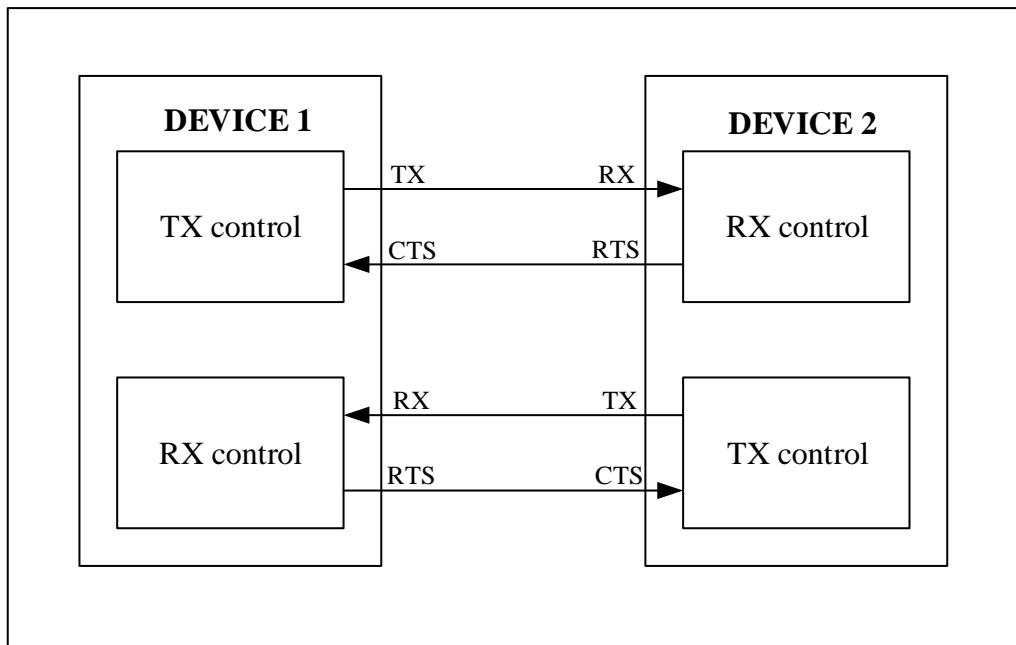
Figure 19-6 Receiving with DMA



#### 19.4.7 Hardware flow control

Hardware flow control functions through CTS input and RTS output. The following figure shows how two devices are connected in this mode.

Figure 19-7 Hardware flow control between two LPUART

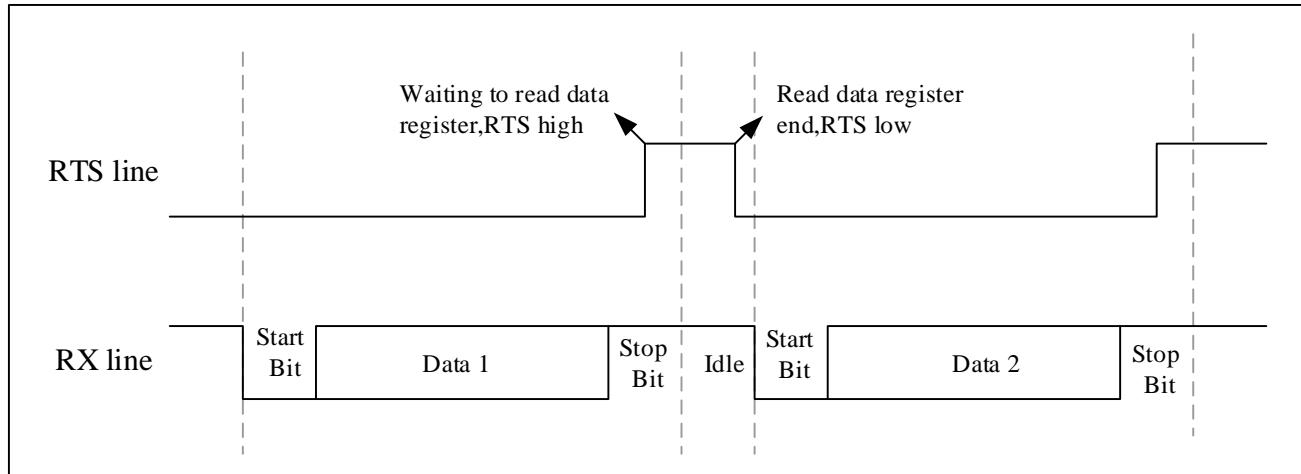


RTS and CTS flow control can be independently enabled by setting LPUART\_CTRL.RTSEN and LPUART\_CTRL.CTSEN.

#### 19.4.7.1 RTS flow control

If RTS flow control is enabled (LPUART\_CTRL.RTSEN=1), the RTS will be driven high (active) when the RTS threshold condition is achieved, otherwise it will be driven low. How is the RTS valid can be selected by the LPUART\_CTRL.RTS\_THSEL[1:0] bits. The RTS threshold can be selected to be effective when the FIFO is half full, 3/4 full, or full. Below is an example of communication with RTS flow control enabled.

Figure 19-8 RTS flow control

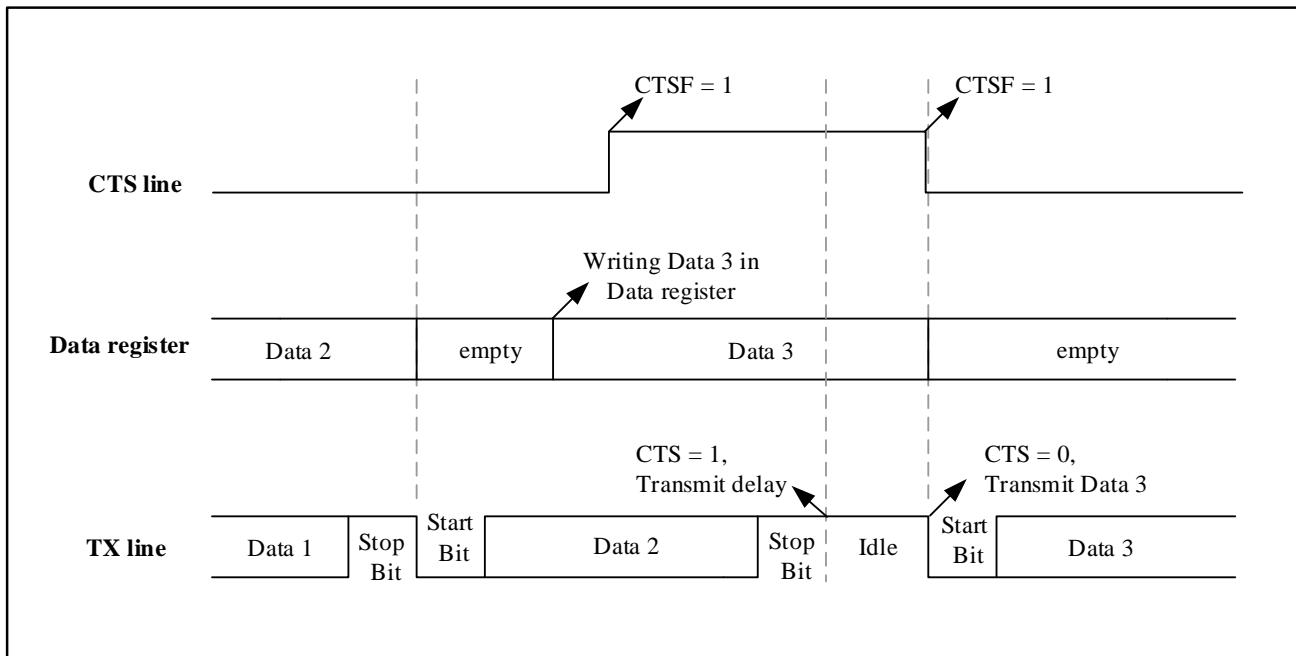


#### 19.4.7.2 CTS flow control

If CTS flow control is enabled (LPUART\_CTRL.CTSEN=1), the sender will check the CTS pin to decide whether or not send data before sending the next frame. If the CTS is pulled low (valid), the sender sends data (assuming that data is ready to be sent). If the CTS is pulled up during transmission, the transmission of the current data frame is stopped after transmission.

If CTS flow control is enabled (LPUART\_CTRL.CTSEN=1), the signal of CTS pin will be changed. See Figure 19-9 for enabling CTS flow control.

Figure 19-9 CTS flow control



#### 19.4.8 Low power wake up

LPUART can work in STOP mode, if the LPUART\_CTRL.WUSTP is set, it can wake up the system on EXTI line 22 when a specific waking up event occurs.

The LPUART waking up event can be handled in the following ways (through the LPUART\_CTRL.WUSEL[1:0]) :

- A waking up event is generated when a start bit is detected
- A waking up event is generated when the receive buffer non-empty flag is set
- A waking up event is generated when data is received and the first byte matches LPUART\_WUDAT[7:0]
- A waking up event is generated when data is received and four bytes match LPUART\_WUDAT[31:0]

When waking up event occurs, the LPUART\_STS.WUF bit will be set.

#### 19.5 Interrupt request

Table 19-3 LPUART interrupt requests

Interrupt function	Interrupt event	Event flag	Enable bit
LPUART global interrupt	Parity check error	PEF	PEIE
	TX complete	TXC	TXCIE
	Receive buffer overrun	FIFO_OV	FIFO_OVIE
	Receive buffer full	FIFO_FU	FIFO_FUIE

	Receive buffer half full	FIFO_HF	FIFO_HFIE
	Receive buffer not empty	FIFO_NE	FIFO_NEIE
	Wake up in STOP mode	WUF	WUFIE

LPUART interrupt events are logical OR. If the corresponding enable control bit is set, these events can generate their own interrupt, but only one interrupt request can be generated at the same time.

## 19.6 LPUART registers

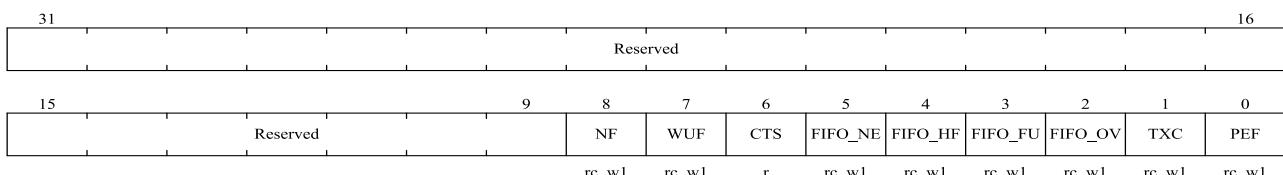
### **19.6.1 LPUART register overview**

Table 19-4 LPUART register overview

### 19.6.2 LPUART status register (LPUART\_STS)

Address offset: 0x00

Reset value: 0x0000 0000

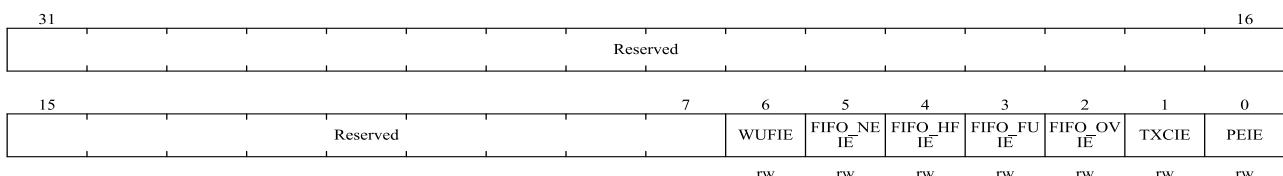


Bit field	Name	Description
31:9	Reserved	Reserved, the reset value must be maintained.
8	NF	Noise detected flag. When noise is detected in the received frame, this bit is set by hardware. This bit is cleared by the software. 0: No noise is detected. 1: Noise is detected.
7	WUF	Wakeup from STOP mode Flag. 0: No wake up event is detected. 1: A wake up event is detected.
6	CTS	CTS signal (hardware flow control) flag. Once the sender requests to send data, it is ready to receive it. 0: CTS line is reset. 1: CTS line is set.
5	FIFO_NE	FIFO non-empty flag. 0: Buffer is empty. 1: Buffer is not empty.RX data is ready to be read
4	FIFO_HF	FIFO half full flag. 0: Buffer is not half full. 1: Buffer is half full.RX data should be read before the buffer is full
3	FIFO_FU	FIFO full flag. 0: Buffer is not full. 1: Buffers is full.RX data should be read out in preparation for receiving new data
2	FIFO_OV	FIFO overrun flag. 0: Buffer did not overrun 1: Buffer overrun.
1	TXC	TX complete flag. 0: TX is disabled or not complete. 1: TX transmission is complete.
0	PEF	Parity check error flag. 0: No parity error detected. 1: Parity error detected

### 19.6.3 LPUART interrupt enable register (LPUART\_INTEN)

Address offset: 0x04

Reset value: 0x0000 0000



Bit field	Name	Description
31:7	Reserved	Reserved, the reset value must be maintained.
6	WUFIE	Wake up interrupt enable 0: Disable wake up interrupt 1: Enable wake up interrupt
5	FIFO_NEIE	Receive buffer not empty interrupt enable 0: Disable buffer non-empty interrupt 1: Enable buffer non-empty interrupt
4	FOFO_HFIE	Receive buffer half-full interrupt enable 0: Disables buffer half-full interrupt 1: Enables buffer half-full interrupt
3	FOFO_FUIE	Receive buffer full interrupt enable 0: Disables buffer full interrupt 1: Enable buffer full interrupt
2	FIFO_OVIE	Receive buffer overrun interrupt enable 0: Disables buffer overrun interrupt 1: Enable buffer overrun interrupt
1	TXCIE	TX complete interrupt enable 0: Disable TX complete interrupt 1: Enable TX complete interrupt
0	PEIE	Parity check error interrupt enable 0: Disable parity error interrupt 1: Enable parity error interrupt

#### 19.6.4 LPUART control register (LPUART\_CTRL)

Address offset: 0x08

Reset value: 0x0000 0200

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	SMPCNT	WUSEL[1:0]	RTSEN	CTSEN	RTS_THSEL[1:0]	WUSTP	DMA_RXEN	DMA_TXEN	LOOPBACK	PCDIS	FLUSH	TXEN	PSEL		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit field	Name	Description
31:15	Reserved	Reserved, the reset value must be maintained.
14	SMPCNT	Specify sampling method 0: 3 sample bits, noise detection is allowed (LPUARTDIV should be large enough, such as greater than 10) 1: 1 sample bits, closed noise detection
13:12	WUSEL[1:0]	Wake up event selection.

Bit field	Name	Description
		00: Start bit detection 01: Non-empty detection of receive buffer 10: A configurable receive byte 11: A programmable 4-byte frame
11	RTSEN	RTS hardware flow control enable 0: Disables RTS hardware flow control 1: Enables RTS hardware flow control
10	CTSEN	CTS hardware flow control enable 0: Disables CTS hardware flow control 1: Enables CTS hardware flow control
9:8	RTS_THSEL[1:0]	RTS threshold selection 00: When FIFO is half full, RTS is effective (pull up) x1: When FIFO is 3/4 full, RTS effective (pull up) 10: When FIFO is full, RTS effective (pull up)
7	WUSTP	LPUART STOP mode wakeup enabled 0: Cannot wake up STOP mode 1: Can wake up the STOP mode
6	DMA_RXEN	DMA RX request enable
5	DMA_TXEN	DMA TX request enable
4	LOOKBACK	Loopback self-test 0: Normal mode 1: Loopback self-test mode
3	PCDIS	Parity control 0: Enables parity bit 1: Disables parity bit
2	FLUSH	Clear receive buffer 0: Disables buffer clear 1: Clear buffer content
1	TXEN	TX enable 0: Disables TX 1: Enables TX
0	PSEL	Odd parity enable 0: Even parity 1: Odd parity

### 19.6.5 LPUART baud rate configuration register 1 (LPUART\_BRCFG1)

Address offset: 0x0C

Reset value: 0x0000 0174

Bit field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained.
15:0	INTEGER[15:0]	<p>Baud rate configuration register 1.</p> <p>The calculation of baud rate configuration register 1 is as follows:</p> <p>If the baud rate is 9600bps and the clock frequency is 32768Hz.</p> $\text{LPUARTDIV} = 32768/9600 = 3.4133$ <p>In this case, the integer part of the LPUARTDIV is 3 and the decimal part is 0.4133. LPUART_BRCFG1 = 3. LPUART_BRCFG2 will be used for baud rate error correction. For the 3-bit sampling method with noise detection characteristics, LPUARTDIV is not large enough at this time, so 1-bit sampling method should be adopted to avoid sampling error.</p>

### 19.6.6 LPUART data register (LPUART\_DAT)

Address offset: 0x10

Reset value: 0x0000 0000

The diagram illustrates a register map with two 32-bit registers. The top register consists of 32 bits, with bit 31 labeled "Reserved" and bits 16 through 0 labeled "DAT[7:0]". The bottom register also consists of 32 bits, with bit 15 labeled "Reserved", bits 8 and 7 labeled "DAT[7:0]", and bit 0 labeled "rw".

<b>Bit field</b>	<b>Name</b>	<b>Description</b>
31:8	Reserved	Reserved, the reset value must be maintained.
7:0	DAT[7:0]	Write to the data register when sending Read the data register when receiving

### 19.6.7 LPUART baud rate configuration register 2 (LPUART\_BRCFG2)

Address offset: 0x14

Reset value: 0x0000 0000

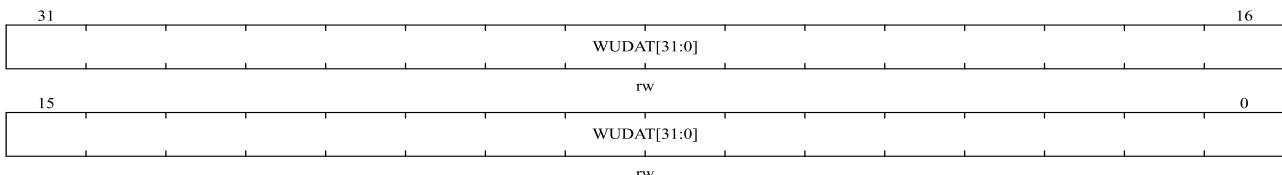
The diagram illustrates a register structure with two 16-bit fields. The top field consists of bits 31 to 16, with the range from 31 to 17 labeled as 'Reserved'. The bottom field consists of bits 15 to 0. It includes a 'Reserved' field from bit 15 to 8, a 'DECIMAL[7:0]' field from bit 7 to 0, and a single bit at bit 0 labeled 'rw'.

Bit field	Name	Description
31:8	Reserved	Reserved, the reset value must be maintained.
7:0	DECIMAL[7:0]	Baud rate configuration register 2 is used for baud rate error correction at low frequencies. For example, If the baud rate is 4800bps and the clock frequency is 32768Hz. $LPUARTDIV = 32768/4800 = 6.8266$ $LPUART\_BRCFG1 = 6$ . In this case, to correct the baud rate error, you should configure register 2 with baud rate. For details on how to configure register 2, refer to the section "Fractional baud rate generation".

### 19.6.8 LPUART wake up data register (LPUART\_WUDAT)

Address offset: 0x18

Reset value: 0x0000 0000



Bit field	Name	Description
31:0	WUDAT[31:0]	When LPUART_CTRL.WUSEL[1:0] = 1x, WUDAT[31:0] is used to check whether the conditions for wake up from STOP mode is matched (byte match or frame match): LPUART_CTRL.WUSEL[1:0] = 10 is used to wake up byte matching. In this case, the first byte is valid LPUART_CTRL.WUSEL[1:0] = 11 is used to wake up frame matching. In this case, all 4 bytes are valid

## 20 Serial peripheral interface/Inter-IC Sound (SPI/ I2S)

### 20.1 SPI introduction

This module is about SPI/I2S. It works in SPI mode by default and users can choose to use I2S by setting the value of registers.

Serial peripheral interface (SPI) is able to work in master or slave mode, support full-duplex and simplex high-speed communication mode, and have hardware CRC calculation and configurable multi-master mode.

On-chip audio interface (I2S) is able to work in master and slave modes in simplex communication, and supports four audio standards: Philips I2S standard, MSB alignment standard, LSB alignment standard and PCM standard.

### 20.2 SPI and I2S main features

#### 20.2.1 SPI features

- Full duplex mode and simplex synchronous mode.
- Support master mode, slave mode and multi-master mode.
- Supports 8-bit or 16-bit data frame format.
- Data bit sequence programmable.
- NSS management by hardware or software.
- Clock polarity and phase programmable.
- Sending and receiving support hardware CRC calculation and check.
- Supports DMA function.

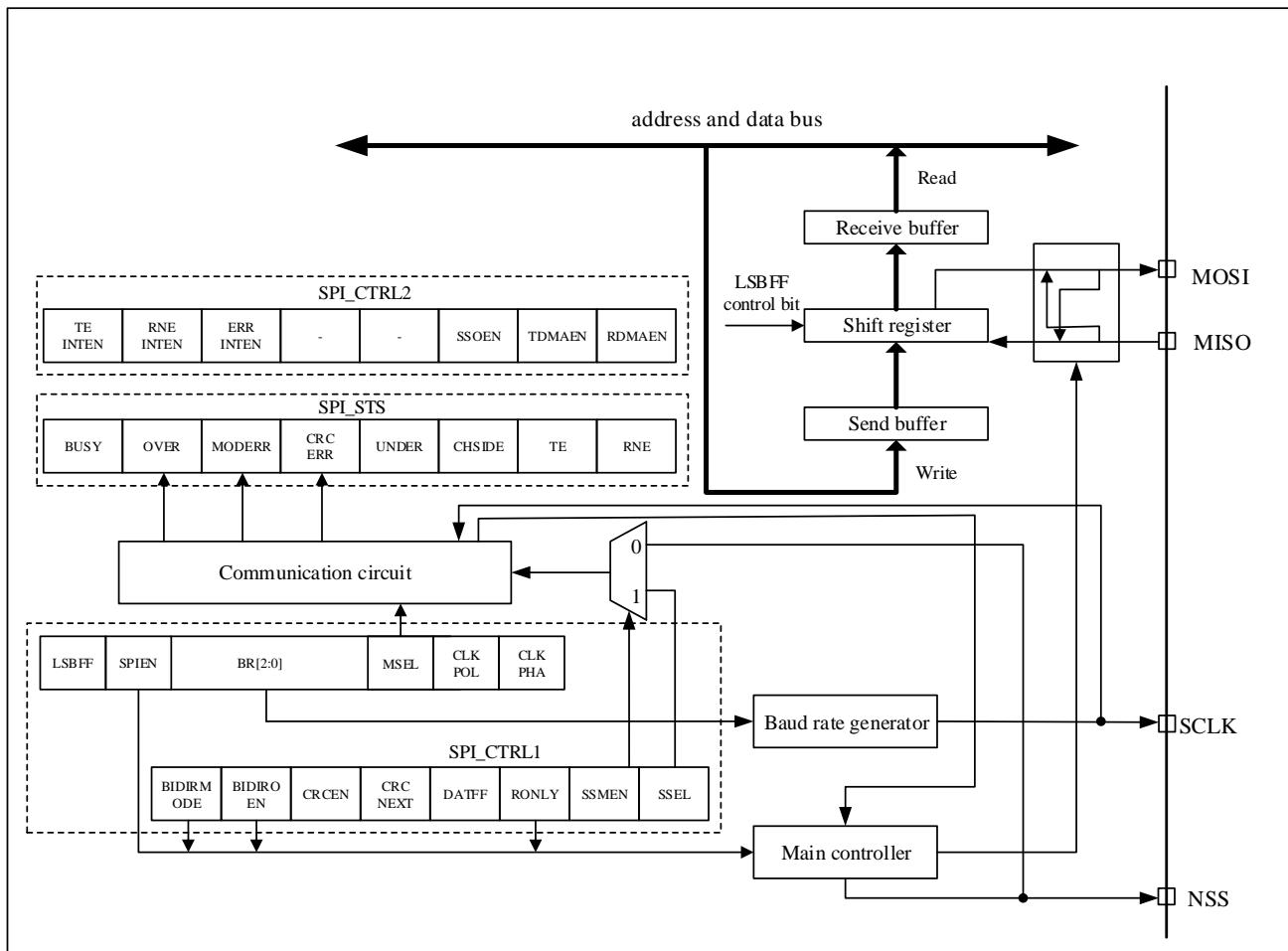
#### 20.2.2 I2S features

- Simplex synchronous mode.
- Supports master mode and slave mode operation.
- Four audio standards are supported: Philips I<sup>2</sup>S standard, MSB alignment standard, LSB alignment standard and PCM standard.
- The audio sampling frequency from 8kHz to 96kHz can be configured.
- Supports 16-bit, 24-bit or 32-bit data length and data frame format (configured according to requirements).
- Steady state clock polarity programmable.
- The data direction is always MSB first.
- Supports DMA function.

## 20.3 SPI function description

### 20.3.1 General description

Figure 20-1 SPI block diagram



To connected external devices, SPI has four pins, which are as follows:

- **SCLK:** serial clock pin. Serial clock signal is output from the SCLK pin of master device and input to SCLK pin of slave device.
- **MISO:** master input/slave output pin. Data is received from the MISO pin of master device and send by the MISO pin of slave device.
- **MOSI:** master output/slave input pin. Data is send by the MOSI pin of master device and received from the MOSI pin of slave device.
- **NSS:** chip select pin. There are two types of NSS pin, internal pin and external pin. If the internal pin detects a high level, SPI works in the master mode. Conversely, SPI works in the slave mode. Users can use a standard I/O pin of the master device to control the NSS pin of the slave device.

### 20.3.1.1 Software NSS mode

The software slave device management is enabled when SPI\_CTRL1.SSMEN = 1

The NSS pin is not used in software NSS mode. In this mode the internal NSS signal level is driven by writing the SPI\_CTRL1.SSEL bit (master mode SPI\_CTRL1.SSEL = 1, slave mode SPI\_CTRL1.SSEL = 0).

### 20.3.1.2 Hardware NSS mode

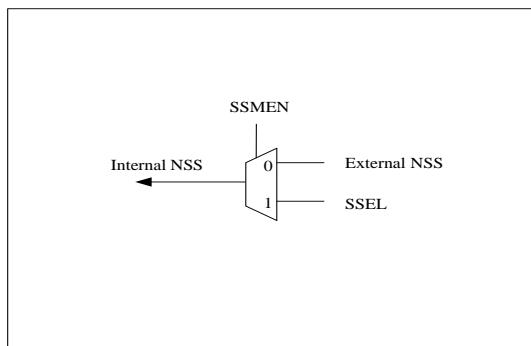
The software slave device management is disabled when SPI\_CTRL1.SSMEN = 0.

**NSS input mode:** The NSS output of the master device is disabled (SPI\_CTRL1.MSEL = 1, SPI\_CTRL2.SSOEN = 0), allowing operation in multi-master mode. The master should connect NSS pin to the high level and the slave should connect NSS pin to the low level during the entire data frame transfer.

**NSS output mode:** NSS output of the master device is enable (SPI\_CTRL1.MSEL = 1, SPI\_CTRL2.SSOEN = 1). SPI as the master device must pull the NSS pin to low level, all device which connected to the master device and set to NSS hardware mode, will detect low level and enter the slave mode automatically. If the master device cannot pull the NSS pin to low level, device will enter the slave mode and generates the master mode failure error.

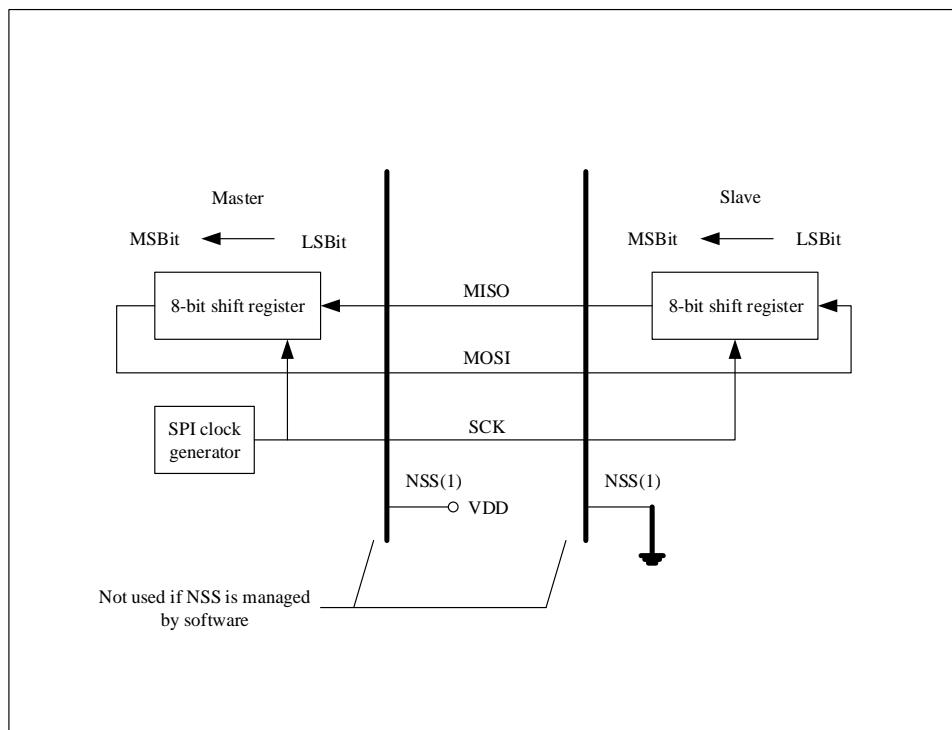
*Note: The choice of software mode or hardware mode depends on whether NSS control is needed in the communication protocol. If not, you can choose the software mode, and release a GPIO pin for other purposes.*

Figure 20-2 Selective management of hardware/software



The following figure is an example of the interconnection of single master and single slave devices.

Figure 20-3 Master and slave applications



*Note: NSS pin is set as input*

SPI is a ring bus structure. The master device outputs a synchronous clock signal through the SCK pin, the MOSI pin of the master device is connected to the MOSI pin of the slave device, and the MISO pin of the master device is connected to the MISO pin of the slave device, so that data can be transferred between devices. Continuous data transfer between master and slave, sending data to slave through MOSI pin and slave sending data to master through MISO pin.

### 20.3.1.3 SPI timing mode

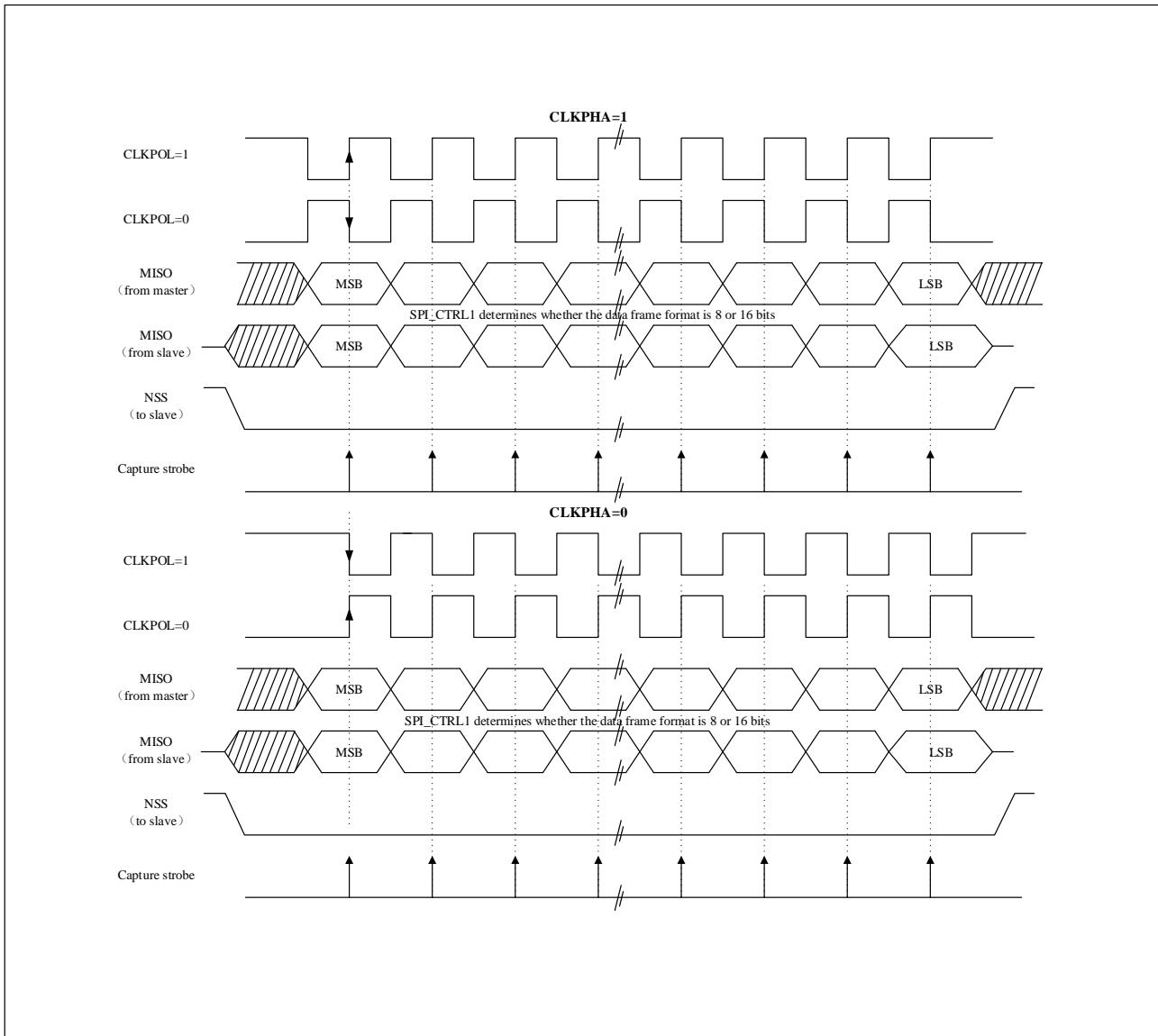
User can select the clock edge of data capture by setting SPI\_CTRL1.CLKPOL bit and SPI\_CTRL1.CLKPHA bit.

- When CLKPOL = 0, CLKPHA = 0, the SCLK pin will keep low in idle state, and the data will be sampled at the first edge, which is rising edge.
- When CLKPOL = 0, CLKPHA = 1, the SCLK pin will keep low in idle state, and the data will be sampled at the second edge, which is falling edge.
- When CLKPOL = 1, CLKPHA = 0, the SCLK pin will keep high in idle state, and the data will be sampled at the first edge, which is falling edge.
- When CLKPOL = 1, CLKPHA = 1, the SCLK pin will keep high in idle state, and the data will be sampled at the second edge, which is rising edge.

Regardless of the timing mode used, the master and slave configuration must be the same.

Figure 20-4 is the combination timing of four CLKPHA and CLKPOL bits transmitted by SPI when the SPI\_CTRL1.LSBFF = 0.

Figure 20-4 Data clock timing diagram



#### 20.3.1.4 Data format

User can selects the data order by setting the SPI\_CTRL1.LSBFF bit. When SPI\_CTRL1.LSBFF = 0, SPI will send the high-order data (MSB) first; When SPI\_CTRL1.LSBFF = 1, SPI will send low-order data (LSB) first.

User can selects the data frame by setting the SPI\_CTRL1.DATFF bit.

### 20.3.2 SPI work mode

#### 20.3.2.1 Master full duplex mode

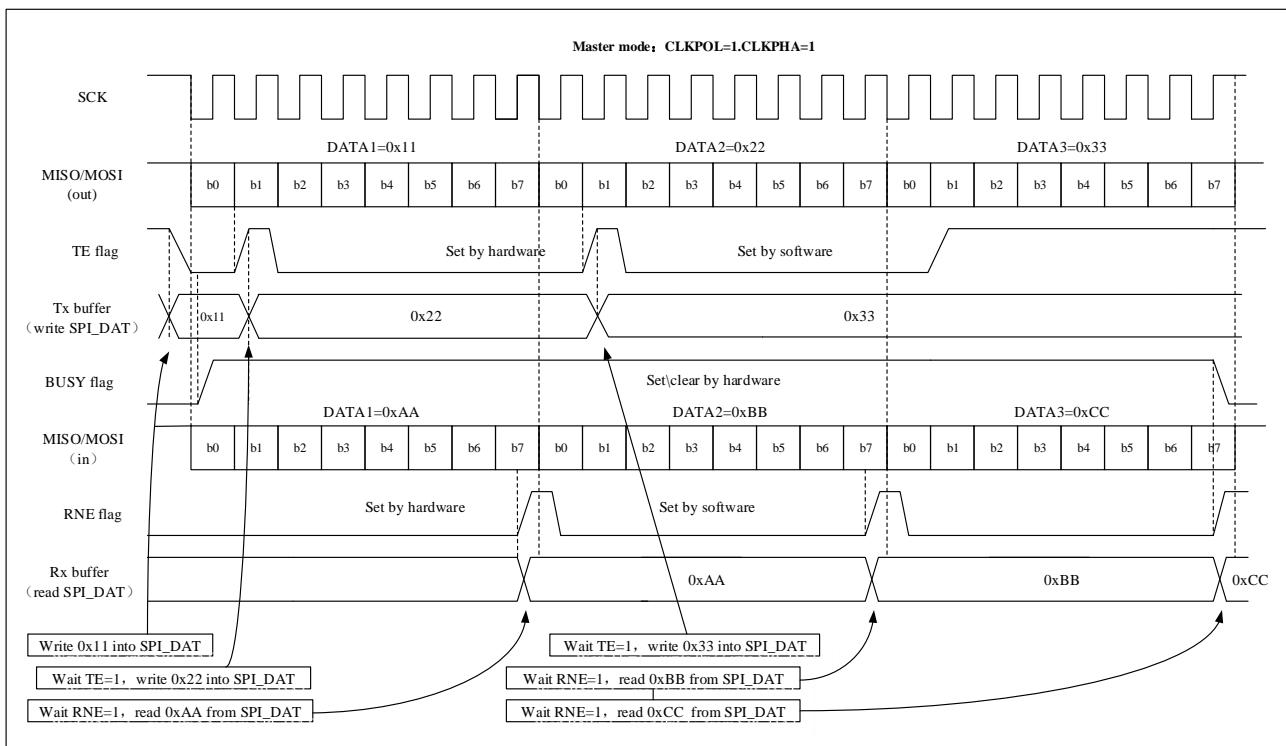
Master full duplex mode (SPI\_CTRL1.MSEL = 1, SPI\_CTRL1.BIDIRMODE = 0, SPI\_CTRL1.RONLY = 0). After the first data is written to the SPI\_DAT register, the transmission will start. When the first bit of the data is sent, the data bytes are loaded from the data register into the shift register in parallel, and then according to the configuration of the SPI\_CTRL1.LSBFF bit, the data bits follow the MSB or LSB order is serially shifted to the MOSI pin. At the

same time, the data received on the MISO pin is serially shifted into the shift register in the same order and then loaded into the SPI\_DAT register in parallel. The software operation process is as follows:

1. Set SPI\_CTRL1.SPIEN = 1, Enable SPI module.
2. Write the first data to be sent into SPI\_DAT register (this operation will clear SPI\_STS.TE bit).
3. Wait for SPI\_STS.TE bit to be set to '1', and write the second data to be sent into SPI\_DAT. Wait for SPI\_STS.RNE bit to be set to '1', read SPI\_DAT to get the first received data, and the SPI\_STS.RNE bit will be cleared by hardware while reading SPI\_DAT. Repeat the above operation, sending subsequent data and receiving n-1 data at the same time;
4. Wait for SPI\_STS.RNE bit to be set to '1' to receive the last data;
5. Wait for SPI\_STS.TE to be set to '1', then wait for SPI\_STS.BUSY bit to be cleared and turn off SPI module.

The process of data sending and data receiving can also be implemented in the interrupt handler generated by the rising edge of the SPI\_STS.RNE or SPI\_STS.TE flag.

Figure 20-5 Schematic diagram of the change of TE/RNE/BUSY when the host is continuously transmitting in full duplex mode



### 20.3.2.2 Master two-wire one-way send-only mode

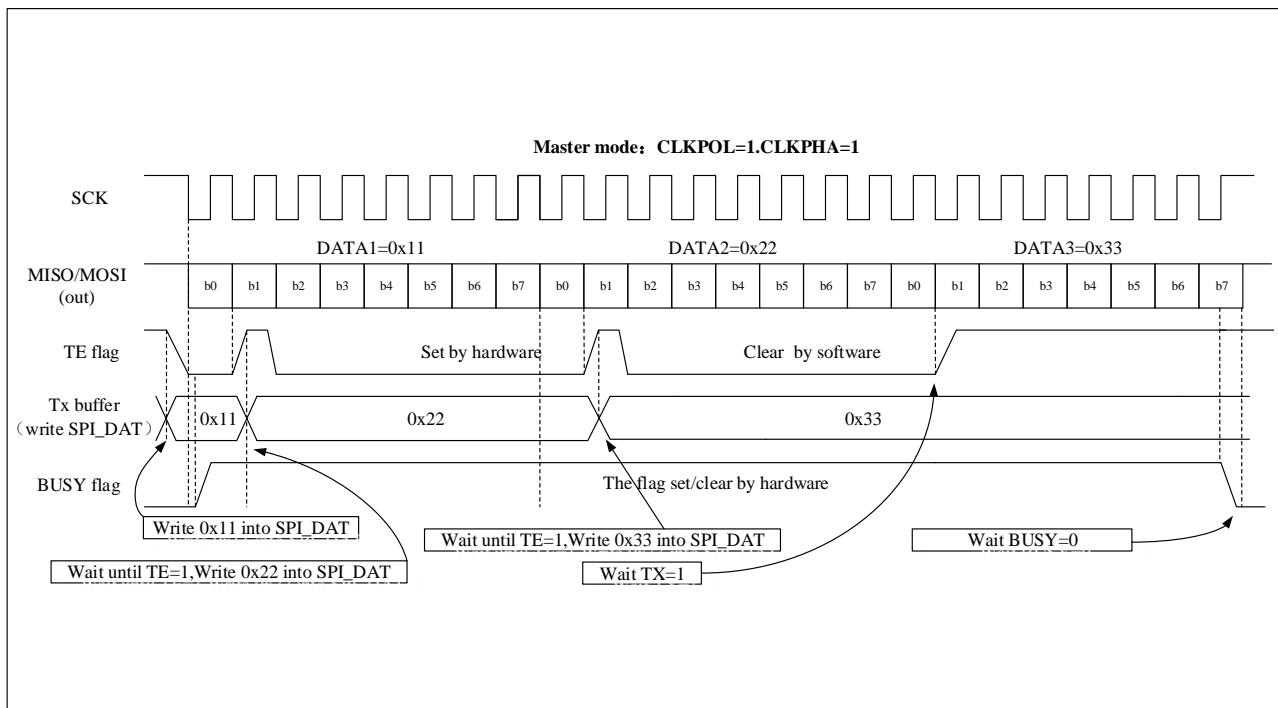
Master two-wire one-way send-only mode (SPI\_CTRL1.MSEL = 1, SPI\_CTRL1.BIDIRMODE = 0, SPI\_CTRL1.RONLY = 0). Master two-wire one-way send-only mode is similar to master full-duplex mode. The difference is that this mode will not read the received data, so the SPI\_STS.OVER bit will be set to '1', and the software will ignore it. The software operation process is as follows:

1. Enable SPI module, set SPI\_CTRL1.SPIEN = 1.

2. Write the first data to be sent into SPI\_DAT register (this operation will clear SPI\_STS.TE bit).
3. Wait for SPI\_STS.TE bit to be set to '1', and write the second data to be sent into SPI\_DAT. Repeat this operation to send subsequent data;
4. After writing the last data to SPI\_DAT, wait for SPI\_STS.TE bit to set '1'; then wait for SPI\_STS.BUSY bit to be cleared to complete the transmission of all data.

The process of data sending can also be implemented in the interrupt handler generated by the rising edge of the TE flag.

Figure 20-6 Schematic diagram of TE/BUSY change when the host transmits continuously in one-way only mode



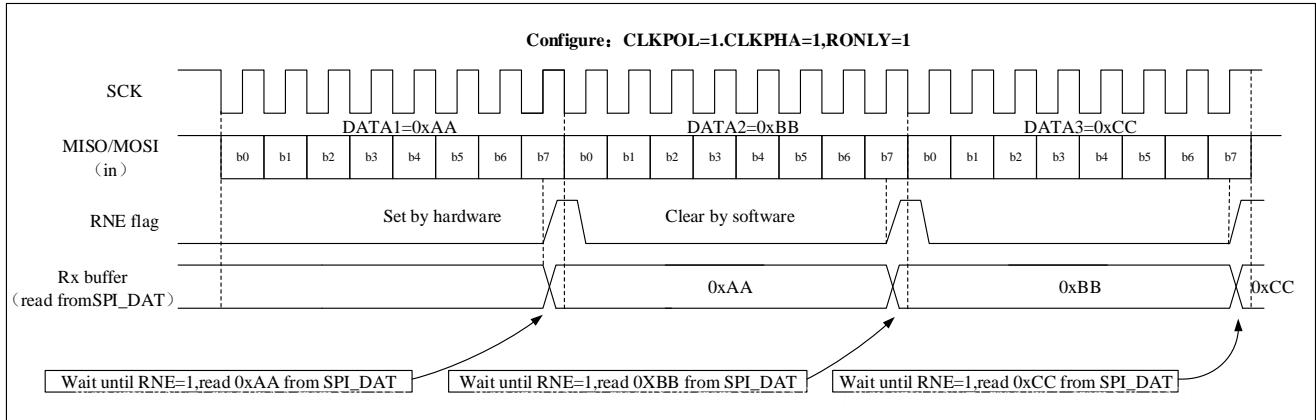
### 20.3.2.3 Master two-wire one-way receive-only mode

Master two-wire one-way receive-only mode (SPI\_CTRL1.MSEL = 1, SPI\_CTRL1.BIDIRMODE = 0, SPI\_CTRL1.RONLY = 1). When SPI\_CTRL1.SPIEN = 1, the receiving process starts. The data bits from the MISO pin are sequentially shifted into the shift register and then loaded into the SPI\_DAT register (receive buffer) in parallel. The software operation process is as follows:

1. Enable the receive-only mode (SPI\_CTRL1.RONLY = 1).
2. Enable SPI module, set SPI\_CTRL1.SPIEN = 1; in master mode, SCLK clock signal is generated immediately, and serial data is continuously received before SPI is turned off (SPI\_CTRL1.SPIEN = 0); in slave mode, serial data is continuously received when the SPI master device pulls low the NSS signal and generates SCLK clock.
3. Wait for SPI\_STS.RNE bit to be set to '1', read the SPI\_DAT register to get the received data, and the SPI\_STS.RNE bit will be cleared by hardware while reading SPI\_DAT register. Repeat this operation to receive all data.

The process of data receiving can also be implemented in the interrupt handler generated by the rising edge of the RNE flag.

Figure 20-7 Schematic diagram of RNE change when continuous transmission occurs in receive-only mode (BIDIRMODE = 0 and RONLY = 1)



#### 20.3.2.4 Master one-wire bidirectional send mode

Master one-wire bidirectional send mode (SPI\_CTRL1.MSEL = 1, SPI\_CTRL1.BIDIRMODE = 1, SPI\_CTRL1.BIDIROEN = 1, SPI\_CTRL1.RONLY = 0). After the data is written to the SPI\_DAT register (send buffer), the transmission process starts. This mode does not receive data. At the same time as the first data bit is sent, the data to be sent is loaded into the shift register in parallel, and then according to the configuration of the SPI\_CTRL1.LSBFF bit, the SPI serially shifts the data bits to the MOSI pin in MSB or LSB order.

The software operation flow of the master one-wire bidirectional send mode is the same as that of the send-only mode.

#### 20.3.2.5 Master one-wire bidirectional receive mode

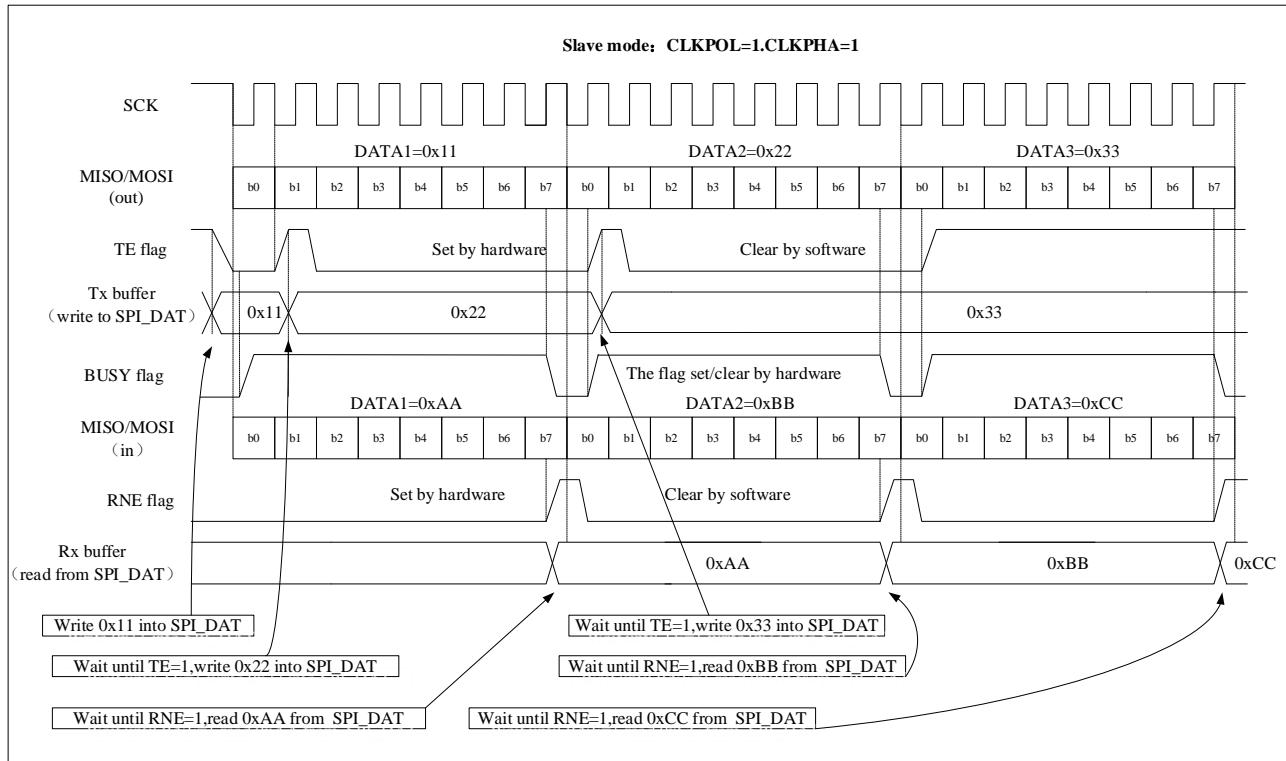
Master one-wire bidirectional receive mode (SPI\_CTRL1.MSEL = 1, SPI\_CTRL1.BIDIRMODE = 1, SPI\_CTRL1.BIDIROEN = 0, SPI\_CTRL1.RONLY = 0). When SPI\_CTRL1.SPIEN = 1, the receiving process starts. There is no data output in this mode, the received data bits are sequentially and serially shifted into the shift register, and then loaded into the SPI\_DAT register (receive buffer) in parallel.

The software operation flow of the master one-wire bidirectional receive mode is the same as that of the receive-only mode.

#### 20.3.2.6 Slave full duplex mode

Slave full duplex mode (SPI\_CTRL1.MSEL = 0, SPI\_CTRL1.BIDIRMODE = 0 and SPI\_CTRL1.RONLY = 0). The data transfer process begins when the slave device receives the first clock edge. Before the master starts data transfer, software must ensure that the data to be send is written to the SPI\_DAT register.

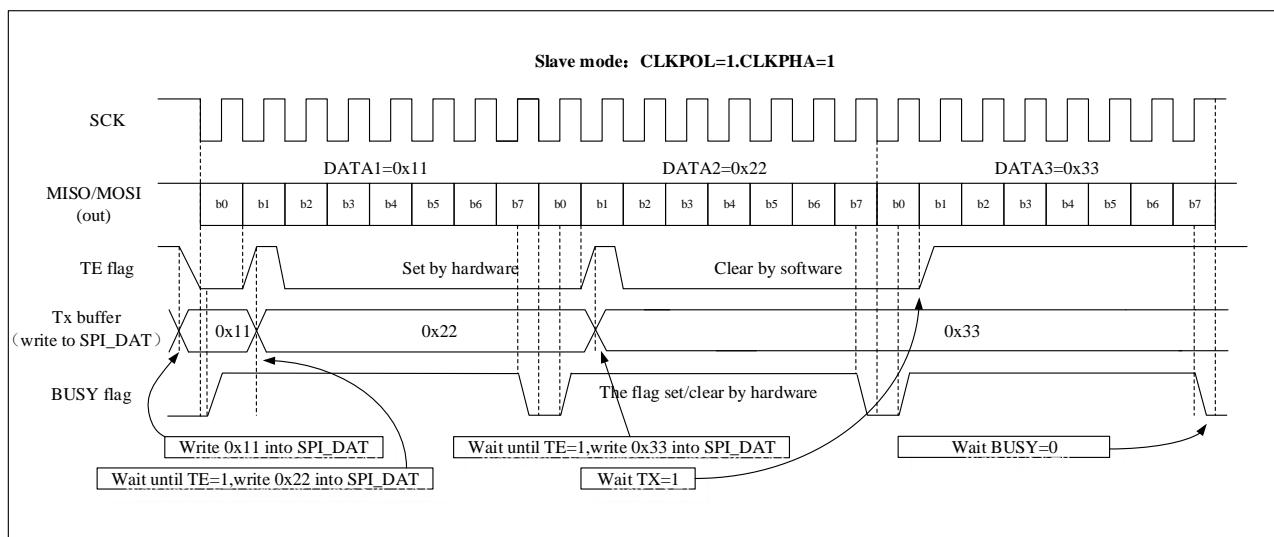
Figure 20-8 Schematic diagram of the change of TE/RNE/BUSY when the slave is continuously transmitting in full duplex mode



### 20.3.2.7 Slave two-wire one-way send-only mode

Slave two-wire one-way send-only mode (SPI\_CTRL1.MSEL = 0, SPI\_CTRL1.BIDIRMODE = 0 and SPI\_CTRL1.RONLY = 0).

Figure 20-9 Schematic diagram of TE/BUSY change during continuous transmission in slave unidirectional transmit-only mode



### 20.3.2.8 Slave two-wire one-way receive-only mode

Slave two-wire one-way receive-only mode (SPI\_CTRL1.MSEL = 0, SPI\_CTRL1.BIDIRMODE = 0 and SPI\_CTRL1.RONLY = 1). The data receiving process begins when the slave device receives the clock signal and the

first data bit from the MOSI pin. The received data bits are sequentially and consecutively shifted serially into an shift register and then loaded into the SPI\_DAT register (receive buffer) in parallel.

### 20.3.2.9 Slave one-wire bidirectional send mode

Slave one-wire bidirectional send mode (SPI\_CTRL1.MSEL = 0, SPI\_CTRL1.BIDIRMODE = 1 and SPI\_CTRL1.BIDIROEN = 1). When the slave device receives the first edge of the clock signal, the sending process starts. No data is received in this mode, and the software must ensure that the data to be sent has been written in the SPI\_DAT register before the SPI master device starts data transmission.

### 20.3.2.10 Slave one-wire bidirectional receive mode

Slave one-wire bidirectional receive mode (SPI\_CTRL1.MSEL = 0, SPI\_CTRL1.BIDIRMODE = 1 and SPI\_CTRL1.BIDIROEN = 0). Data receiving begins when the slave device receives the first clock edge and a data bit from the MOSI pin. There is no data output in this mode, the received data bits are sequentially and consecutively shifted serially into an shift register, and then loaded into the SPI\_DAT register (receive buffer) in parallel.

*Note: The software operation process of the slave can refer to the master.*

### 20.3.2.11 SPI initialization process

1. The baud rate of serial clock is defined by the SPI\_CTRL1.BR[2:0] bits (this step is ignored if it is working in slave mode).
2. Select SPI\_CTRL1.CLKPOL bit and SPI\_CTRL1.CLKPHA bit to define the phase relationship between data transmission and serial clock (see Figure 20-4).
3. Set SPI\_CTRL1.DATFF bit to define 8-bit or 16-bit data frame format.
4. Configure the SPI\_CTRL1.LSBFF bit to define the frame format.
5. Configure the NSS mode as described above for the NSS function.
6. Run mode is configured by SPI\_CTRL1.MSEL bit, SPI\_CTRL1.BIDIRMODE bit, SPI\_CTRL1.BIDIROEN bit and SPI\_CTRL1.RONLY bit.
7. Set the SPI\_CTRL1.SPIEN = 1 to enable SPI.

### 20.3.2.12 Basic send and receive process

When SPI sends a data frame, it first loads the data frame from the data buffer into the shift register, and then starts to send the loaded data. When the data is transferred from the send buffer to the shift register, the send buffer empty flag is set (SPI\_STS.TE = 1), and the next data can be loaded into the send buffer; if the TEINTEN bit is set (SPI\_CTRL2.TEINTEN = 1), an interrupt will be generated; writing data to the SPI\_DAT register will clear the SPI\_STS.TE bit.

At the last edge of the sampling clock, when the data is transferred from the shift register to the receive buffer, the receive buffer non-empty flag is set (SPI\_STS.RNE = 1), at this time the data is ready and can be read from the SPI\_DAT register; if the receive buffer non-empty interrupt is enabled (SPI\_CTRL2.RNEINTEN = 1), an interrupt will be generated; the SPI\_STS.RNE bit can be cleared by reading the SPI\_DAT register data.

In master mode, the sending process starts when data is written to the send buffer. If the next data has been written into the SPI\_DAT register before the current data frame sending is completed, the continuous sending function can

be achieved.

In slave mode, the NSS pin is low, and when the first clock edge arrives, the transmission process begins. In order to avoid accidental data transmission, software must write data to the transmit buffer before data transmission (it is recommended to enable the SPI module before the host sends the clock).

In some configurations, when the last data is sent, the BUSY flag (SPI\_STS.BUSY) can be used to wait for the end of the data sending.

### 20.3.2.13 Continuous and discontinuous transmission

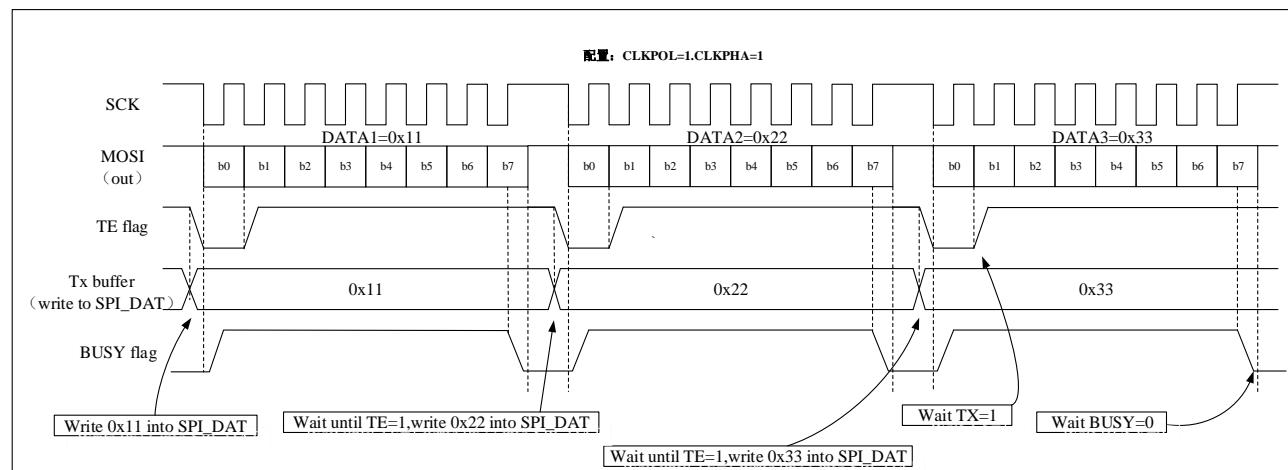
When sending data in master mode, if the software is fast enough to detect each TE (SPI\_STS.TE) rising edge (or TE interrupt), and the data is written to the SPI\_DAT register immediately before the end of the ongoing transmission. At this time, the SPI clock remains continuous between the transmission of data items, and the SPI\_STS.BUSY bit will not be cleared, continuous communication can be achieved.

If the software is not fast enough, it will result in discontinuous communication; in this case, the SPI\_STS.BUSY bit is cleared between the transmission of each data items (see figure below).

In master receive-only mode (SPI\_CTRL1.RONLY = 1), communication is always continuous and the BUSY flag (SPI\_STS.BUSY) is always high.

In slave mode, the continuity of communication is determined by the SPI master device. However, even if the communication is continuous, the BUSY flag (SPI\_STS.BUSY) will be low for at least one SPI clock cycle between each data item (see Figure 20-4).

Figure 20-10 Schematic diagram of TE/BUSY change when BIDIRMODE = 0 and RONLY = 0 are transmitted discontinuously.



### 20.3.3 Status flag

The SPI\_STS register has 3 flag bits to monitor the status of the SPI:

#### 20.3.3.1 Send buffer empty flag bit (TE)

When the send buffer is empty, the TE flag (SPI\_STS.TE) is set to 1, which means that new data can be written into the SPI\_DAT register. When the send buffer is not empty, the hardware will clear this flag to 0.

### 20.3.3.2 Receive buffer non-empty flag bit (RNE)

When the receive buffer is not empty, the RNE flag (SPI\_STS.RNE) is set to 1, so the user knows that there is data in the receive buffer. After reading the SPI\_DAT register, the hardware will set this flag to 0.

### 20.3.3.3 BUSY flag bit (BUSY)

When the transmission starts, the hardware sets the BUSY flag (SPI\_STS.BUSY) to 1, and after the transmission ends, the hardware sets the BUSY flag to 0.

Only when the device is in the master one-wire bidirectional receive mode, the BUSY flag (SPI\_STS.BUSY) will be set to 0 when the communication is in progress.

The BUSY flag (SPI\_STS.BUSY) will be cleared to 0 in the following cases:

- End of transmission (except for continuous communication in master mode);
- Turn off the SPI module (SPI\_CTRL1.SPIEN = 0);
- The master mode error occurs (SPI\_STS.MODERR = 1)

When the communication is discontinuous: the BUSY flag (SPI\_STS.BUSY) is cleared to '0' between the transmission of each data item.

When communication is continuous: in master mode, the BUSY flag (SPI\_STS.BUSY) remains high during the entire transfer process; In slave mode, the BUSY flag (SPI\_STS.BUSY) will be low for 1 SPI clock cycle between each data item transfer. So do not use the BUSY flag to handle the sending and receiving of each data item.

## 20.3.4 Turn off the SPI

In order to turn off the SPI module, different operation modes require different operation steps.

### 20.3.4.1 Master or slave full duplex mode

1. Wait for the RNE flag (SPI\_STS.RNE) to be set to 1 and the last byte to be received;
2. Wait for the TE flag (SPI\_STS.TE) to be set to 1;
3. Wait for the BUSY flag (SPI\_STS.BUSY) to be cleared to 0;
4. Turn off the SPI module (SPI\_CTRL1.SPIEN = 0).

### 20.3.4.2 One-way send-only mode or bidirectional send mode for master or slave

1. After writing the last byte to the SPI\_DAT register, wait for the TE flag (SPI\_STS.TE) to be set to 1;
2. Wait for the BUSY flag (SPI\_STS.BUSY) to be cleared to 0;
3. Turn off the SPI module (SPI\_CTRL1.SPIEN = 0).

### 20.3.4.3 One-way receive-only mode or bidirectional receive mode for master

1. Wait for the penultimate RNE (SPI\_STS.RNE) to be set to 1;
2. Before closing the SPI module (SPI\_CTRL1.SPIEN = 0), wait for 1 SPI clock cycle (using software delay);
3. Wait for the last RNE (SPI\_STS.RNE) to be set before entering shutdown mode (or turning off the SPI module

clock).

#### 20.3.4.4 One-way receive-only mode or bidirectional receive mode for slave

1. The SPI module can be turned off at any time (SPI\_CTRL1.SPIEN = 0), and after the current transfer is over, the SPI module will be turned off;
2. If you want to enter the shutdown mode, you must wait for the BUSY flag (SPI\_STS.BUSY) to be set to 0 before entering the shutdown mode (or turn off the SPI module clock).

#### 20.3.5 SPI communication using DMA

Users can choose DMA for SPI data transfer, the application program can be released, and the system efficiency can be greatly improved.

When the send buffer DMA is enabled (SPI\_CTRL2.TDMAEN = 1), each time the TE flag (SPI\_STS.TE) bit is 1, a DMA request will be generated, and the DMA will automatically write the data to the SPI\_DAT register, which will clear the TE flag (SPI\_STS.TE) bit. When the receive buffer DMA is enabled (SPI\_CTRL2.RDMAEN = 1), each time the RNE flag (SPI\_STS.RNE) bit is set to 1, a DMA request will be generated, and the DMA will automatically read the SPI\_DAT register, which will clear the RNE flag (SPI\_STS.RNE) bit.

When the SPI is only used for sending data, only the send DMA channel of the SPI needs to be enabled (SPI\_CTRL2.TDMAEN = 1).

When the SPI is only used for receiving data, only the receive DMA channel of the SPI needs to be enabled (SPI\_CTRL2.RDMAEN = 1).

In send mode, after DMA has sent all the data to be sent (DMA\_INTSTS.TXCF = 1), BUSY flag (SPI\_STS.BUSY) can monitor to confirm whether SPI communication is over, which can avoid destroying the transmission of the last data when the SPI is turned off or enters the shutdown mode. Therefore, the software needs to wait for the TE flag (SPI\_STS.TE) bit to be set to 1, and wait for the BUSY flag (SPI\_STS.BUSY) bit to be set to 0.

Figure 20-11 Transmission using DMA

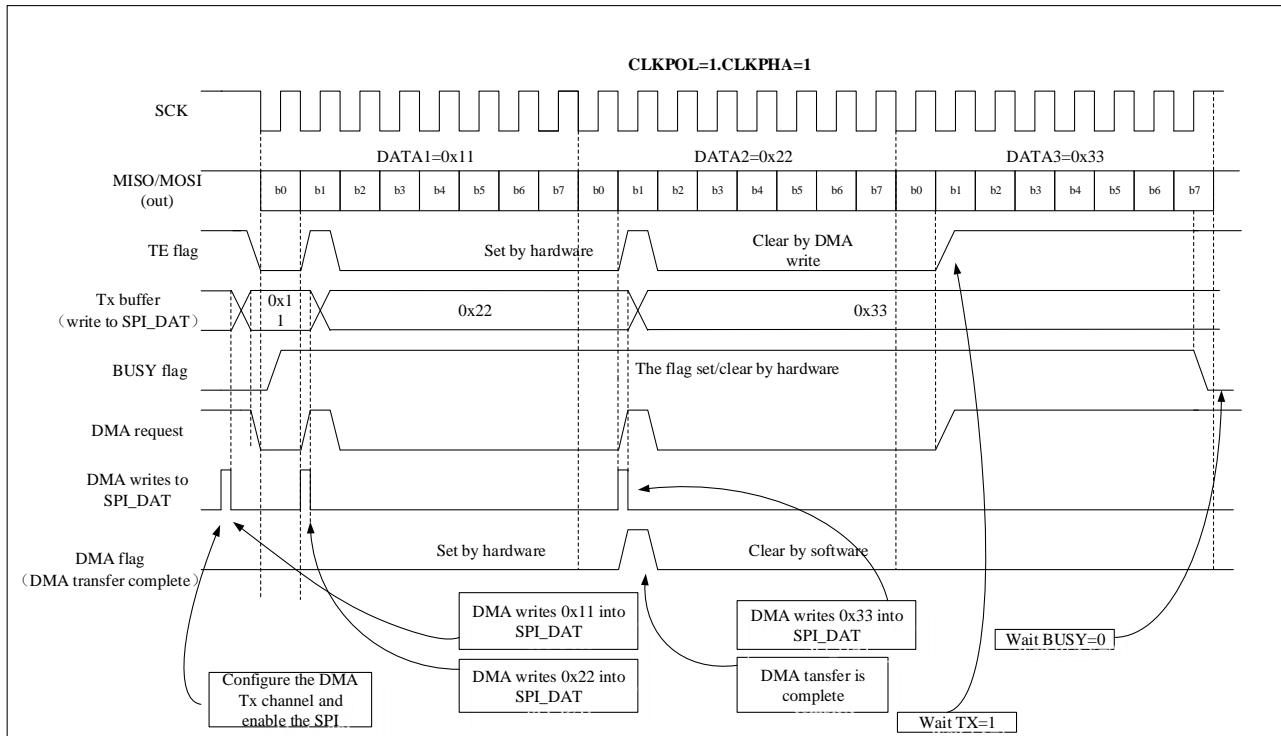
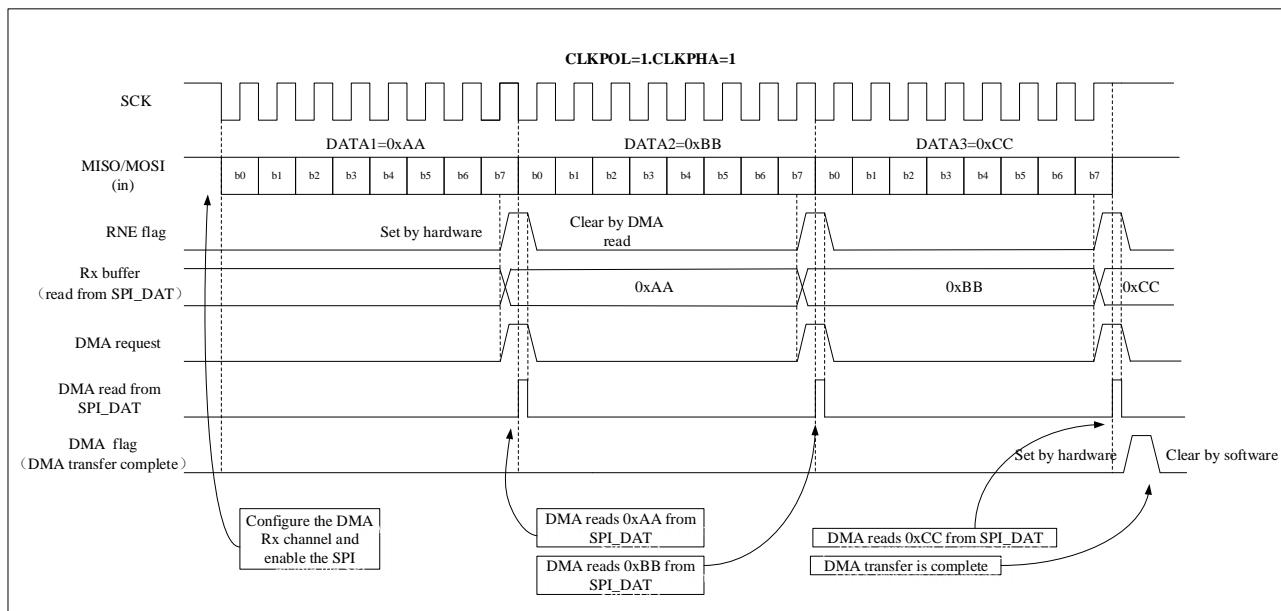


Figure 20-12 Reception using DMA



### 20.3.6 CRC calculation

SPI contains two independent CRC calculators for data sending and data receiving to ensure the correctness of data transmission. According to the sending and receiving data frame format, CRC adopts different calculation methods,

the 8-bit data frame format adopts CRC8, and the 16-bit data frame format adopts CRC16. The polynomial used in the SPI CRC calculation is set by the SPI\_CRCPOLY register, and the user enables the CRC calculation by setting the SPI\_CTRL1.CRCEN = 1.

In send mode, after the last data is written into the send buffer, set the SPI\_CTRL1.CRCNEXT = 1, which indicates that the hardware will start sending the CRC value (SPI\_CRCTDAT value) after sending the data. When the CRC is sent, the CRC calculation will stop.

In receive mode, after the penultimate data frame is received, set the SPI\_CTRL1.CRCNEXT = 1. The received CRC and SPI\_CRCRDAT values are compared, if they are different, the SPI\_STS.CRCERR bit is set to 1. If the SPI\_CTRL2.ERRINTEN bit is set to 1, an interrupt will be generated.

In order to keep the synchronization of the next CRC calculation result of the master-slave device, the user should clear the CRC value of the master-slave device. Setting the SPI\_CTRL1.CRCEN bit resets the SPI\_CRCRDAT and SPI\_CRCTDAT registers. Take the following steps in order: SPI\_CTRL1.SPIEN = 0; SPI\_CTRL1.CRCEN = 0; SPI\_CTRL1.CRCEN = 1; SPI\_CTRL1.SPIEN = 1.

Most importantly, when the SPI is configured in slave mode and CRC is enabled, as long as there is a clock pulse on SCLK pin, the CRC calculation will still be performed even if the NSS pin is high. This situation is common when the master device communicates with multiple slave devices alternately, so it is necessary to avoid CRC misoperation.

When the SPI hardware CRC check is enabled (SPI\_CTRL1.CRCEN = 1) and the DMA is enabled, the hardware automatically completes the sending and receiving of CRC bytes when the communication ends.

## 20.3.7 Error flag

### 20.3.7.1 Master mode failure error (MODERR)

The following two conditions will cause the master mode failure error:

- NSS pin hardware management mode, the master device NSS pin is pulled low;
- NSS pin software management mode, the SPI\_CTRL1.SSEL bit is set to 0.

When a master mode failure error occurs, the SPI\_STS.MODERR bit is set to 1. An interrupt is generated if the user enables the corresponding interrupt(SPI\_CTRL2.ERRINTEN=1). The SPI\_CTRL1.SPIEN bit and SPI\_CTRL1.MSEL bit will be write protected and both are cleared by hardware. SPI is turned off and forced into slave mode

Software performs a read or write operation to the SPI\_STS register, and then writes to the SPI\_CTRL1 register to clear the SPI\_STS.MODERR bit (in multi-master mode, the master's NSS pin must be pulled high first).

Normally, the SPI\_STS.MODERR bit of the slave cannot be set to 1. However, in a multi-master configuration, the slave's SPI\_STS.MODERR bit may be set to 1. In this case, the SPI\_STS.MODERR bit indicates that there is a multi-master collision. The interrupt routine can perform a reset or return to the default state to recover from an error state.

### 20.3.7.2 Overflow error (OVER)

When the SPI\_STS.RNE bit is set to 1, but there is still data sent into the receive buffer, an overflow error will occur. At this time, the overflow flag SPI\_STS.OVER bit is set to 1. An interrupt is generated if the user enables the corresponding interrupt (SPI\_CTRL2.ERRINTEN = 1). All received data is lost, and the SPI\_DAT register retains

only previously unread data.

Read the SPI\_DAT register and the SPI\_STS register in turn to clear the SPI\_STS.OVER bit

### 20.3.7.3 CRC error (CRCERR)

The CRC error flag is used to check the validity of the received data. A CRC error occurs when the received CRC value does not match the SPI\_CRCRDAT value. At this time, the SPI\_STS.CRCERR flag bit is set to '1', and an interrupt will be generated if the user enables the corresponding interrupt (SPI\_CTRL2.ERRINTEN = 1).

### 20.3.8 SPI interrupt

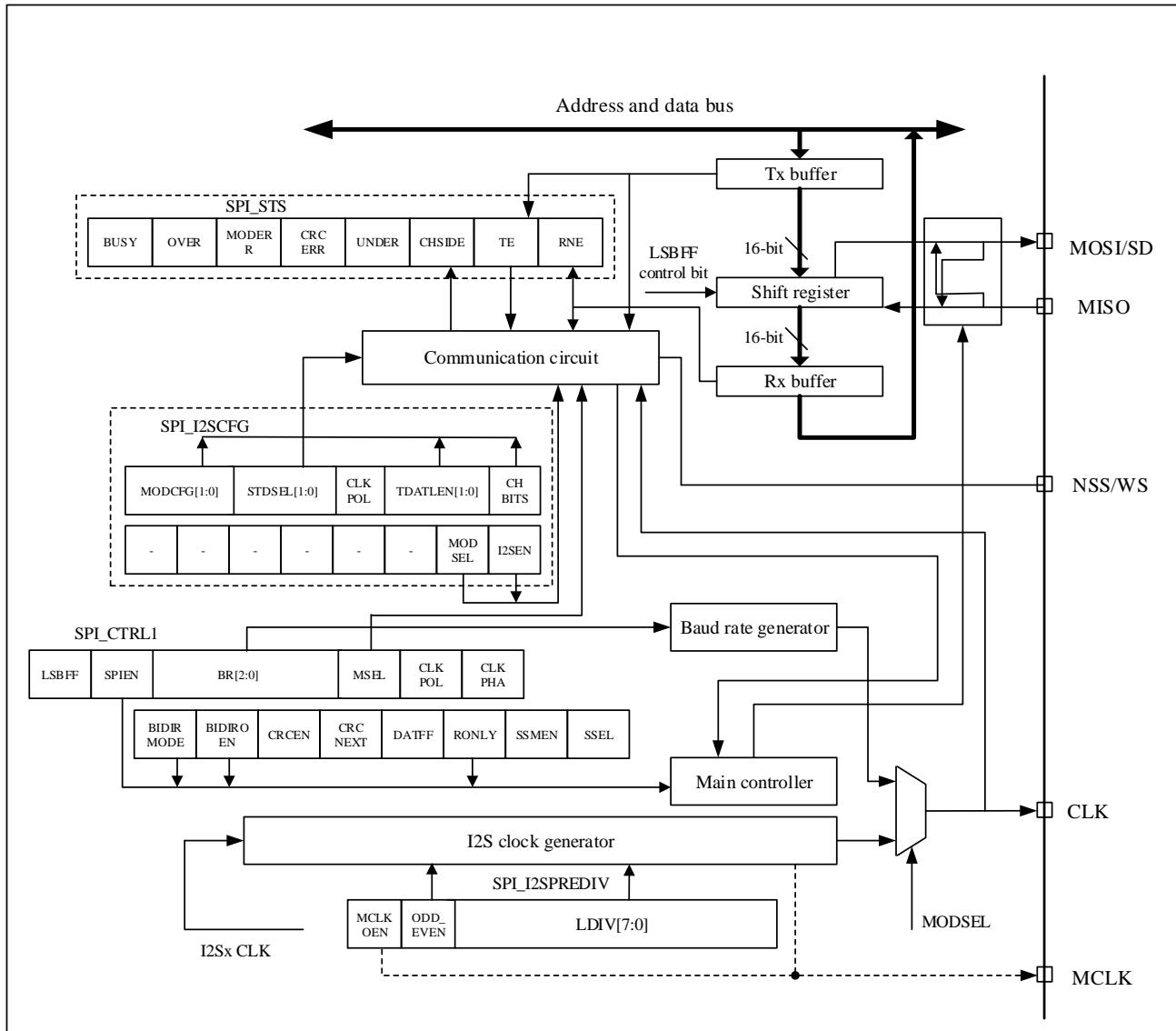
Table 20-1 SPI interrupt request

Interrupt event	Event flag bit	Enable control bit
Send buffer empty flag	TE	TEINTEN
Receive buffer non empty flag	RNE	RNEINTEN
Master mode failure event	MODERR	ERRINTEN
Overflow error	OVER	
CRC error flag	CRCERR	

## 20.4 I<sup>2</sup>S function description

The block diagram of I<sup>2</sup>S is shown in the figure below:

Figure 20-13 I<sup>2</sup>S block diagram



The I<sup>2</sup>S interface uses the same pins, flags and interrupts as the SPI interface. Setting the SPI\_I2SCFG.MODSEL = 1 selects the I<sup>2</sup>S audio interface.

I<sup>2</sup>S has a total of 4 pins, 3 of which are shared with SPI:

- CLK: Serial clock (shared with SCLK pin), CLK generates a pulse every time 1-bit audio data is sent.
- SD: Serial data (shared with MOSI pin), used for data send and receive;
- WS: Channel selection (shared with NSS pin), used as data control signal output in master mode, and used as input in slave mode;

- MCLK: master clock (independent mapping, optional), output  $256 \times F_s$  clock signal to ensure better synchronization between systems.

*Note:  $F_s$  is the sampling frequency of audio signal*

In master mode, I2S uses its own clock generator to generate clock signals for communication, and this clock generator is also the clock source of the master clock output (SPI\_I2SPREDIV.MCLKOEN = 1, the master clock output is enabled).

### 20.4.1 Supported audio protocols

Four audio standards can be selected by setting the SPI\_I2SCFG.STDSEL[1:0] bits:

- I<sup>2</sup>S Philips standard
- MSB alignment standard
- LSB alignment standard
- PCM standard

The audio data of the left channel and the right channel are usually time-division multiplexed, and the left channel always sends data before the right channel. By checking the SPI\_STS.CHSIDE bit, the user can distinguish which channel the received data belongs to. However, in the PCM audio standard, the CHSIDE bit has no meaning.

By setting the SPI\_I2SCFG.TDATLEN bits, the user can set the length of the data to be transmitted, and set the data bit width of the channel by setting the SPI\_I2SCFG.CHBITS bits. There are 4 data formats for sending data as follows:

- 16-bit data is packed into 16-bit data frame
- 16-bit data is packed into a 32-bit data frame (the first 16 bits are meaningful data, and the last 16 bits are set to 0 by hardware)
- 24-bit data is packed into 32-bit data frame (the first 24-bit data is meaningful data, and the latter 8-bit data is set to 0 by hardware)
- 32-bit data is packed into 32-bit data frame

I2S uses the same SPI\_DAT register as SPI to send and receive 16-bit wide data. If I2S needs to send or receive 24-bit or 32-bit wide data, the CPU needs to read or write the SPI\_DAT register twice. On the other hand, when I2S sends or receives 16-bit wide data, the CPU only needs to read or write the SPI\_DAT register once.

Regardless of which data format and communication standard is used, I2S always sends the data high-order bit (MSB) first.

#### 20.4.1.1 I2S Philips standard

Using the I2S Philips standard, the device that sends data changes the data on the falling edge of the clock, and the device that receives data samples the data on the rising edge of the clock. The WS signal should be valid one clock before the first data bit (MSB) is sent and will change on the falling edge of the clock signal.

Figure 20-14 I<sup>2</sup>S Philips protocol waveform (16/32-bit full precision, CLKPOL = 0)

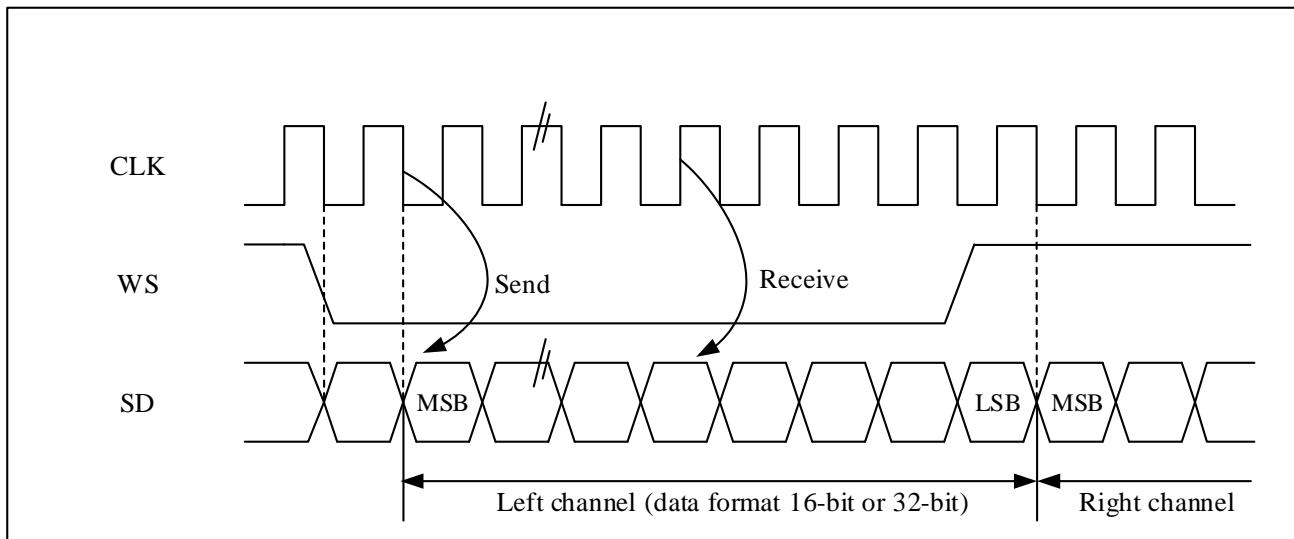
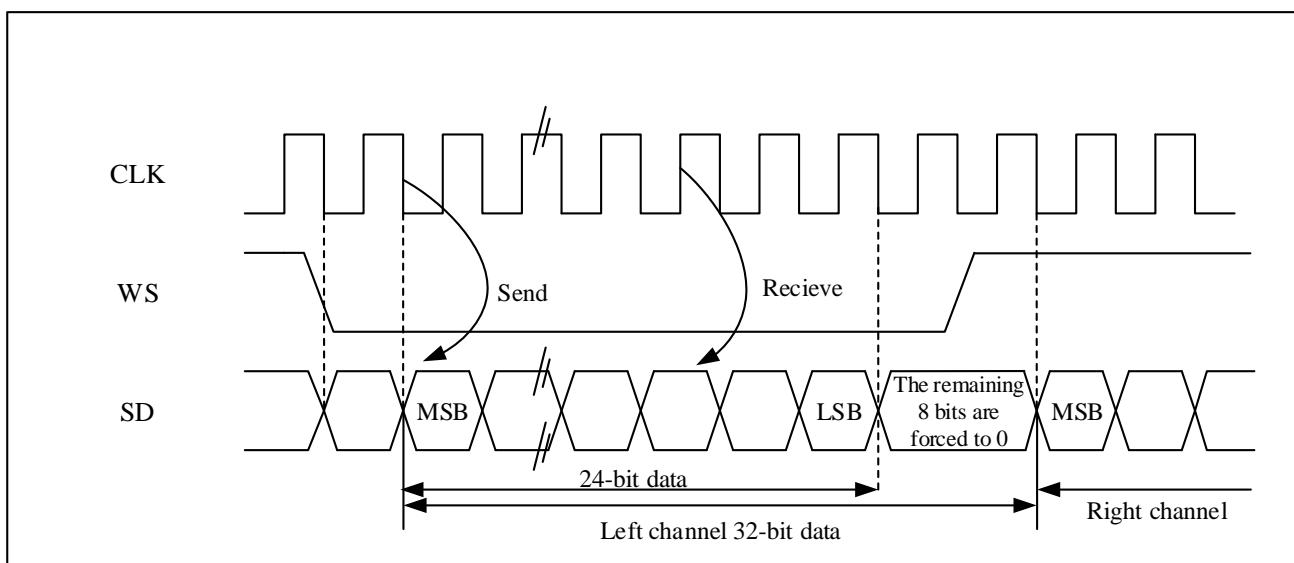
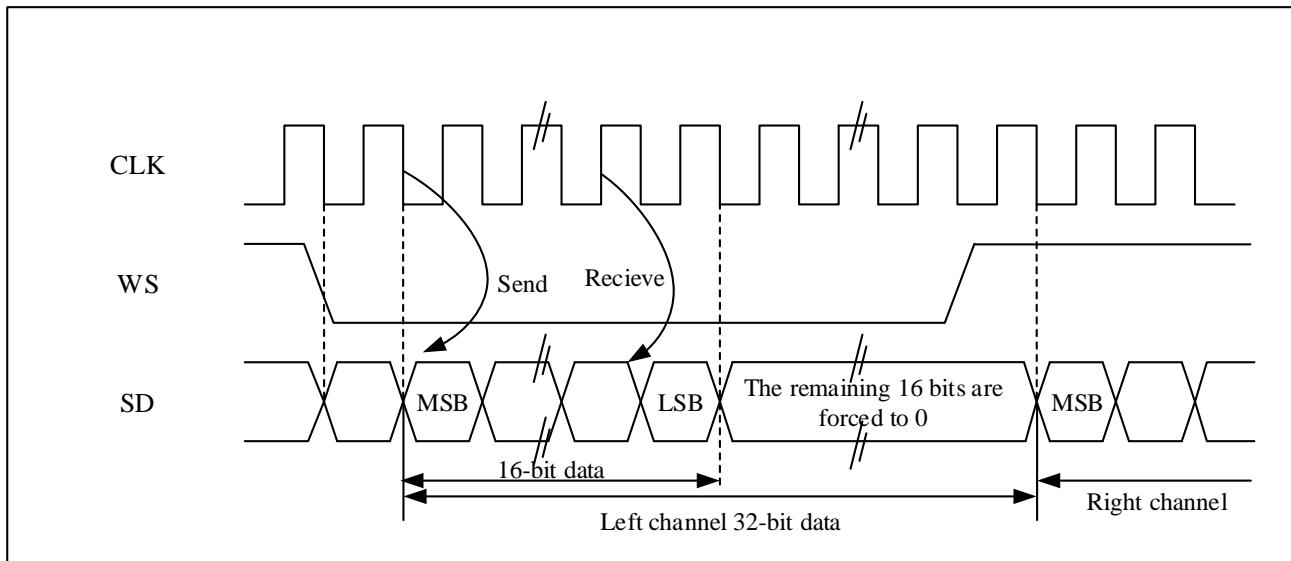


Figure 20-15 I<sup>2</sup>S Philips protocol standard waveform (24-bit frame, CLKPOL = 0)



If the 24-bit data needs to be packaged into 32-bit data frame format, the CPU needs to read or write the SPI\_DAT register twice during each frame of data transmission. For example, if the user sends 24-bit data 0x95AA66, the CPU will first write 0x95AA into the SPI\_DAT register, and then write 0x66XX into the SPI\_DAT register (only the upper 8-bit data is valid, the lower 8-bit data is meaningless and can be any value); if the user receives 24-bit data 0x95AA66, the CPU will first read the SPI\_DAT register to get 0x95AA, and then read the SPI\_DAT register to get 0x6600 (only the upper 8-bit data is valid, and the lower 8-bit data is always 0).

Figure 20-16 I<sup>2</sup>S Philips protocol standard waveform (16-bit extended to 32-bit packet frame, CLKPOL = 0)



If 16-bit data needs to be packed into 32-bit data frame format, the CPU only needs to read or write the SPI\_DAT register once for each frame of data transmission. The lower 16 bits of data for expansion to 32 bits are always set to 0x0000. For example, if the user sends or receives 16-bit data 0x89C1 (extended to 32-bit data is 0x89C10000). In the process of sending data, the upper 16-bit half word (0x89C1) needs to be written into the SPI\_DAT register; the user can write new data until the SPI\_STS.TE bit is set. An interrupt is generated if the user enables the corresponding interrupt. The sending is performed by hardware, even if the last 16 bits (0x0000) are not sent, the hardware will set the TE (SPI\_STS.TE) bit to 1 and the corresponding interrupt will be generated. In the process of receiving data, the RNE flag (SPI\_STS.RNE) will be set to 1 after each time the device receives the upper 16-bit halfword (0x89C1). An interrupt is generated if the user enables the corresponding interrupt. In this way, there is more time between 2 reads and writes, which can prevent underflow or overflow from happening.

#### 20.4.1.2 MSB alignment standard

In the MSB alignment standard, the device sending the data will change the data on the falling edge of the clock, and the device receiving the data will sample the data on the rising edge of the clock. The WS signal and the first data bit (MSB) are generated simultaneously.

The standard data receiving and sending processing mode is the same as I<sup>2</sup>S Philips standard.

Figure 20-17 The MSB is aligned with 16-bit or 32-bit full precision, CLKPOL = 0.

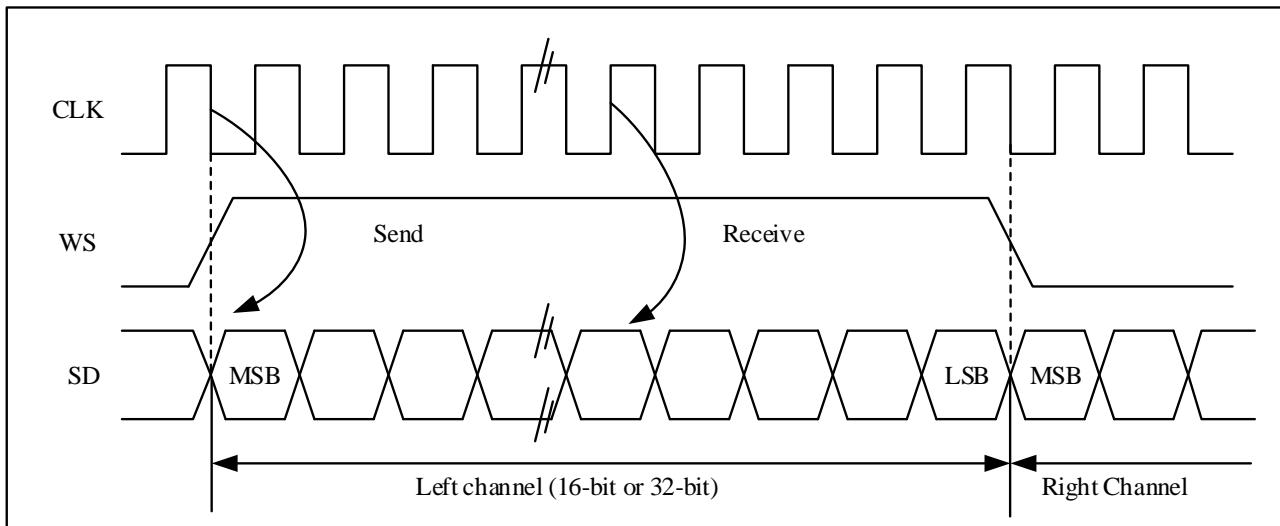


Figure 20-18 MSB aligns 24-bit data, CLKPOL = 0

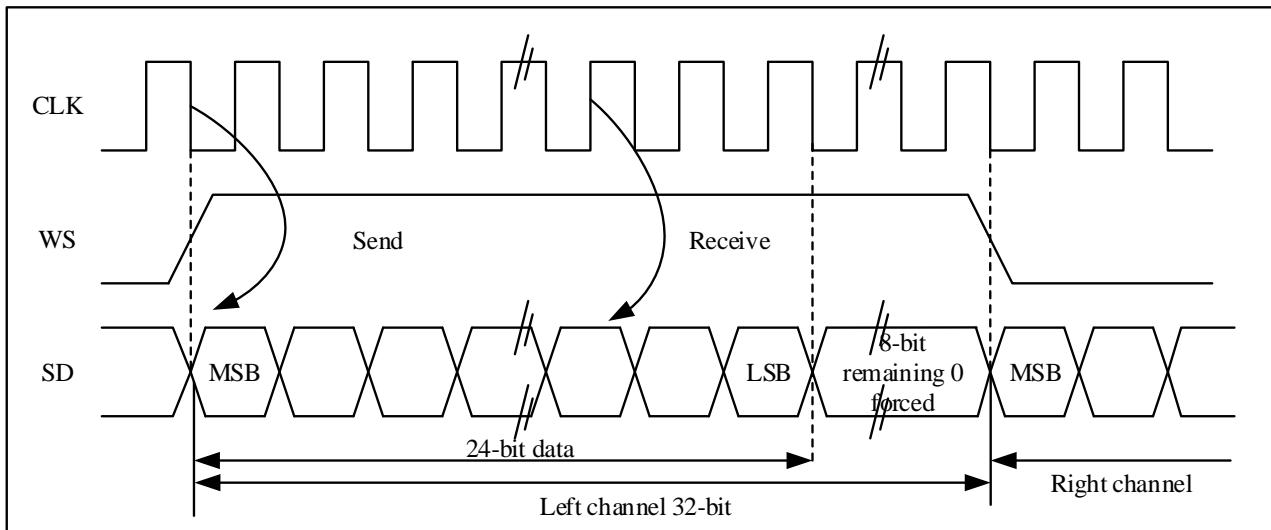
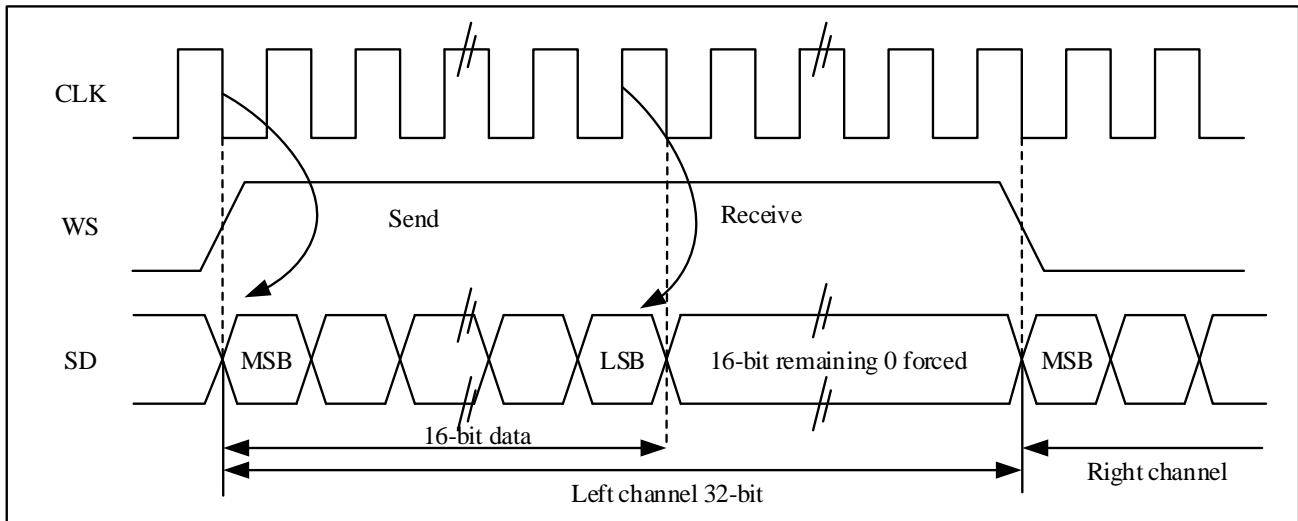


Figure 20-19 MSB-aligned 16-bit data is extended to 32-bit packet frame, CLKPOL = 0



#### 20.4.1.3 LSB alignment standard

In 16-bit or 32-bit full-precision frame format, LSB alignment standard is the same as MSB alignment standard.

Figure 20-20 LSB alignment 16-bit or 32-bit full precision, CLKPOL = 0

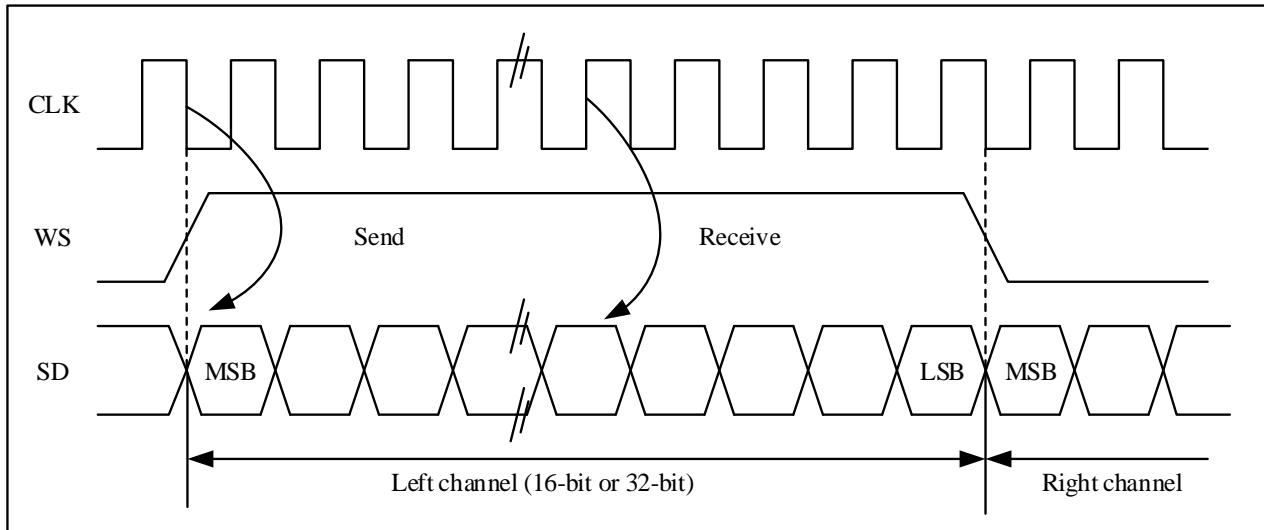
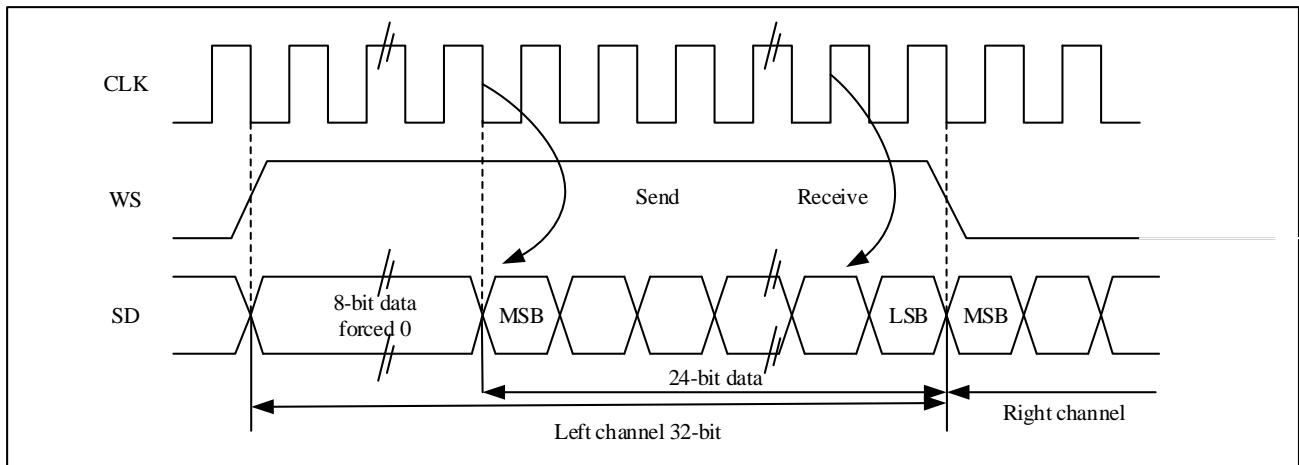
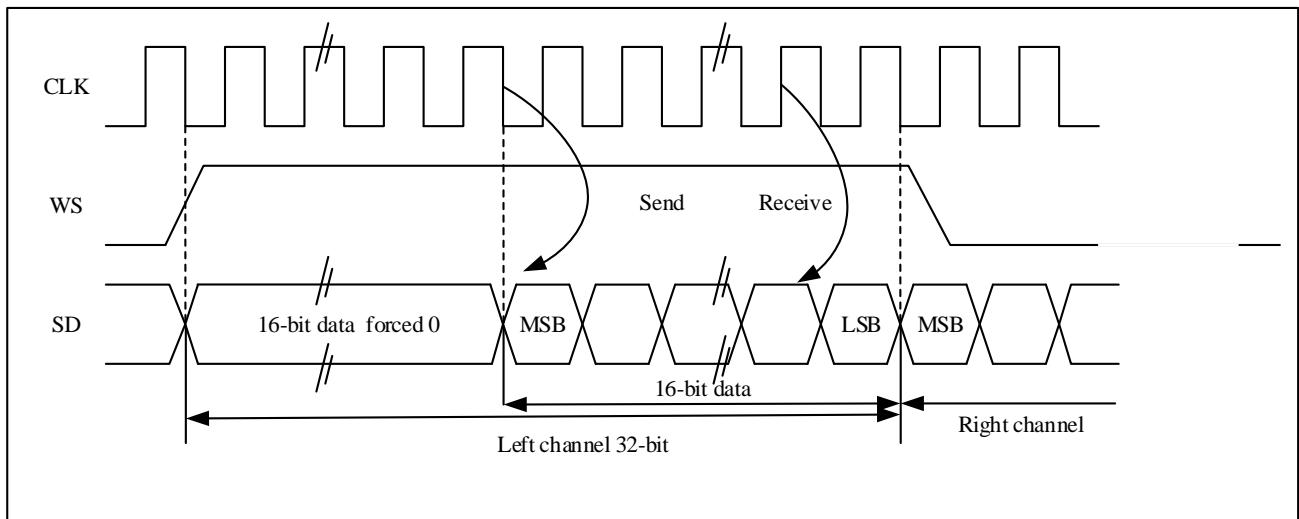


Figure 20-21 LSB aligns 24-bit data, CLKPOL = 0



If the 24-bit data needs to be packed into the 32-bit data frame format, the CPU needs to read or write the SPI\_DAT register twice during each frame of data transmission. For example, if the user sends 24-bit data 0x95AA66, the CPU will first write 0xXX95 (only the lower 8-bit data is valid, the upper 8-bit data is meaningless and can be any value) into the SPI\_DAT register, and then write 0xAA66 into the SPI\_DAT register. If the user receives 24-bit data 0x95AA66, the CPU will first read the SPI\_DAT register to get 0x00095 (only the lower 8 bits are valid, the upper 8 bits are always 0), and then read the SPI\_DAT register to get 0xAA66.

Figure 20-22 LSB aligned 16-bit data is extended to 32-bit packet frame, CLKPOL = 0



If the 16-bit data needs to be packaged into a 32-bit data frame format, the CPU only needs to read or write the SPI\_DAT register once for each frame of data transmission. The upper 16 bits of extended to 32 bits data are set to 0x0000 by hardware, if the user sends or receives 16-bit data 0x89C1 (extended to 32-bit data is 0x000089C1). In the process of sending data, the upper 16-bit halfword (0x0000) needs to be written to the SPI\_DAT register first; once the valid data starts to be send, the next TE (SPI\_STS.TE) event will be generated. In the process of receiving data, once the device receives valid data, the RNE (SPI\_STS.RNE) event will be generated. In this way, there is more time between 2 reads and writes, which can prevent underflow or overflow from happening.

#### 20.4.1.4 PCM standard

In the PCM standard, there are two frame structures, short frame and long frame. The user can select the frame structure by setting the SPI\_I2SCFG.PCMFSYNC bits. The WS signal indicates frame synchronization information. The WS signal for synchronizing long frames is 13 bits effective; the WS signal length for synchronizing short frames is 1 bit.

The standard data receiving and sending processing mode is the same as I<sup>2</sup>S Philips standard.

Figure 20-23 PCM standard waveform (16 bits)

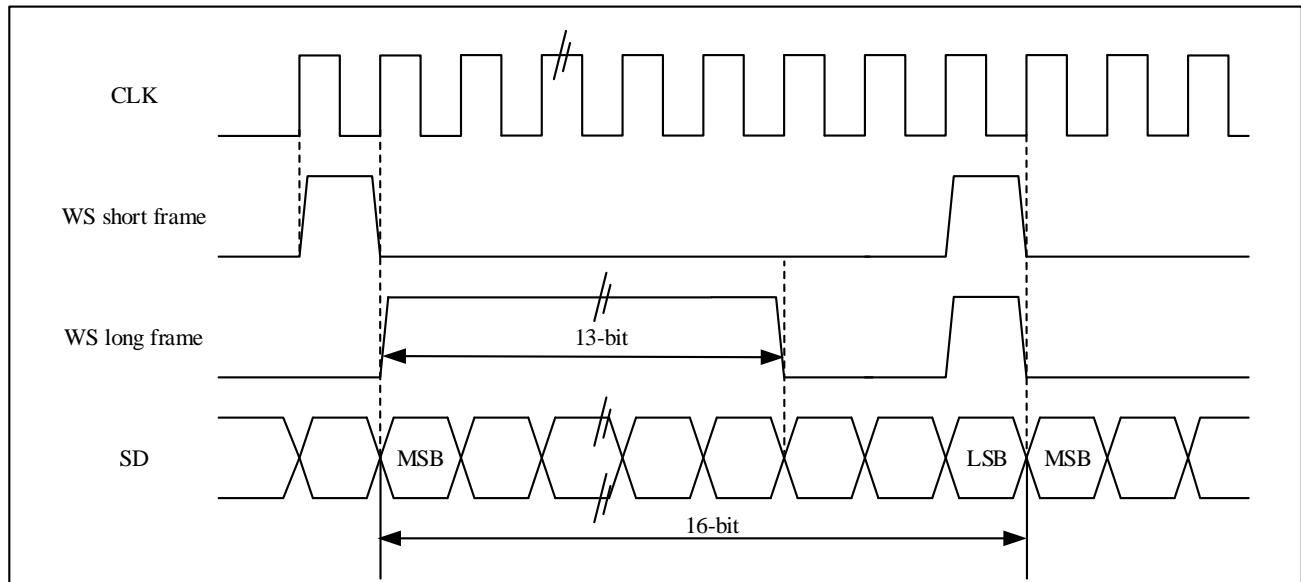
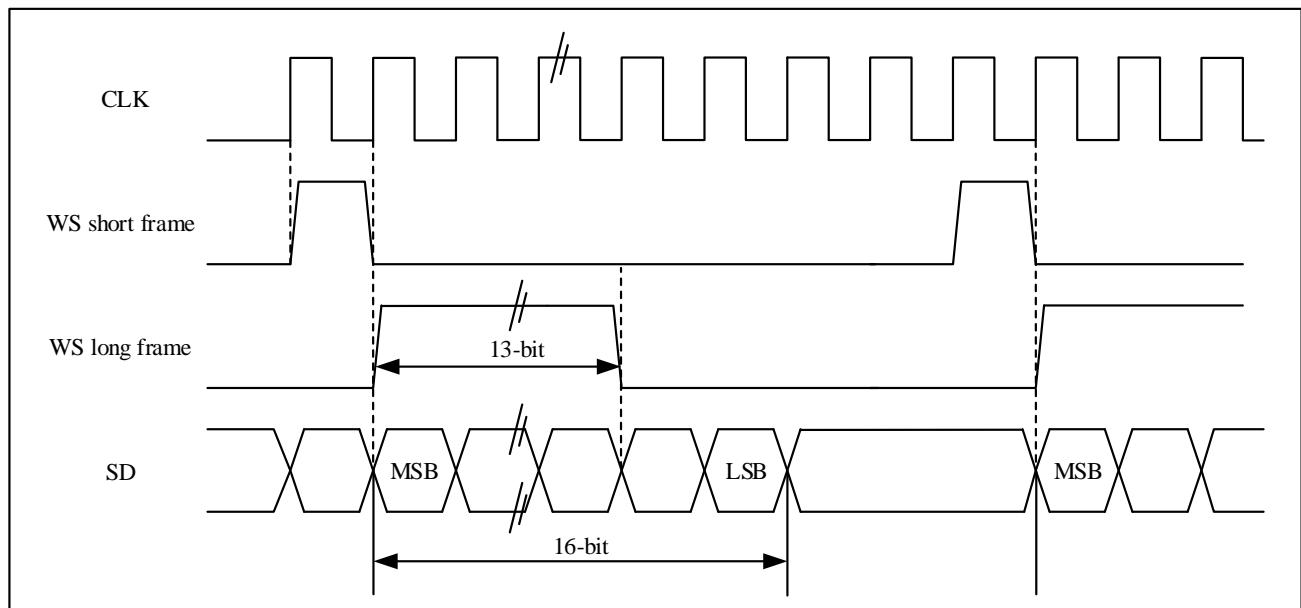


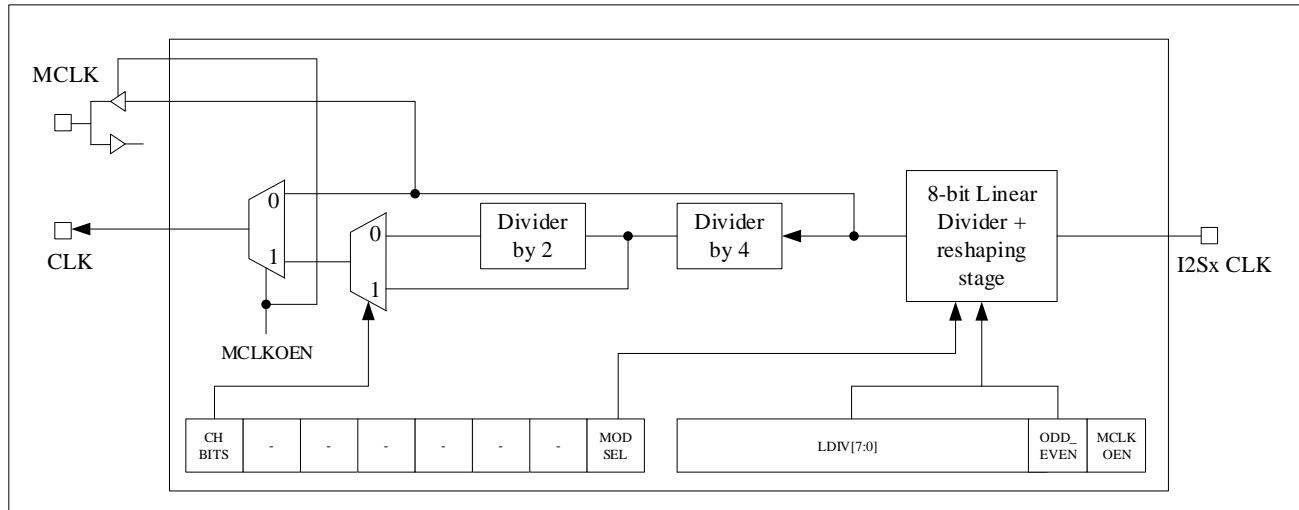
Figure 20-24 PCM standard waveform (16-bit extended to 32-bit packet frame)



## 20.4.2 Clock generator

In the master mode, the linear divider needs to be set correctly in order to obtain the desired audio frequency.

Figure 20-25 I<sup>2</sup>S clock generator structure



*Note: The clock source of I<sup>2</sup>Sx CLK is HSI, HSE or PLL system clock that drives AHB clock.*

The bit rate of I<sup>2</sup>S determines the data flow on the I<sup>2</sup>S data line and the frequency of the I<sup>2</sup>S clock signal.

$$\text{I}^2\text{S bit rate} = \text{number of bits per channel} \times \text{number of channels} \times \text{audio sampling frequency}$$

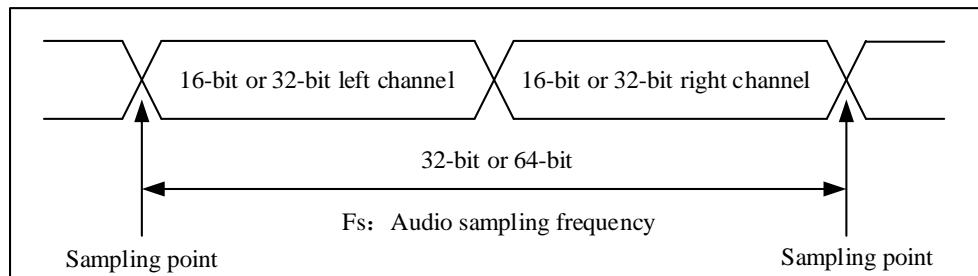
For a signal with left and right channels and 16-bit audio, the I<sup>2</sup>S bit rate is calculated as:

$$\text{I}^2\text{S bit rate} = 16 \times 2 \times F_s$$

If the packet length is 32 bits, there are:

$$\text{I}^2\text{S bit rate} = 32 \times 2 \times F_s$$

Figure 20-26 Audio sampling frequency definition



The sampling signal frequency of the audio can be set by setting the SPI\_I2SPREDIV.ODD\_EVEN bit and the SPI\_I2SPREDIV.LDIV[7:0] bits. Audio can be sampled at 96kHz, 48kHz, 44.1kHz, 32kHz, 22.05kHz, 16kHz, 11.025kHz, or 8kHz (or any value within this range). Set the linear divider according to the following formula:

$$\text{When } MCLKOEN = 1 \text{ and } CHBITS= 0, F_s = I^2Sx \text{ CLK} / [(16 \times 2) \times ((2 \times LDIV) + ODD\_EVEN) \times 8]$$

$$\text{When } MCLKOEN = 1 \text{ and } CHBITS = 1, F_s = I^2Sx \text{ CLK} / [(32 \times 2) \times ((2 \times LDIV) + ODD\_EVEN) \times 4]$$

When MCLKOEN = 0 and CHBITS = 0,  $F_S = I^2Sx \text{ CLK} / [(16 \times 2) \times ((2 \times LDIV) + ODD\_EVEN)]$

When MCLKOEN = 0 and CHBITS = 1,  $F_S = I^2Sx \text{ CLK} / [(32 \times 2) \times ((2 \times LDIV) + ODD\_EVEN)]$

The exact audio frequency can be obtained by referring to the clock configuration in the table below.

Table 20-2 Use the standard 8MHz HSE clock to get accurate audio frequency.

SYSCLK MHz	I <sup>2</sup> S_LDIV		I <sup>2</sup> S_ODD_EVEN		MCLK	Target Fs(Hz)	Real Fs(Hz)		Error	
	16 bits	32 bits	16 bits	32 bits			16 bits	32 bits	16 bits	32 bits
48	8	4	0	0	without	96000	93750	93750	2.34%	2.34%
48	15	8	1	0	without	48000	48387.1	46875	0.81%	2.34%
48	17	8	0	1	without	44100	44117.65	44117.65	0.04%	0.04%
48	23	11	1	1	without	32000	31914.89	32608.7	0.27%	17.00%
48	34	17	0	0	without	22050	22058.82	22058.82	0.04%	0.04%
48	47	23	0	1	without	16000	15957.45	15957.45	0.27%	0.27%
48	68	34	0	0	without	11025	11029.41	11029.41	0.04%	0.04%
48	94	47	0	0	without	8000	7978.72	7978.72	0.27%	0.27%
48	1	1	0	0	yes	96000	93750	93750	2.34%	2.34%
48	2	2	0	0	yes	48000	46875	46875	2.34%	2.34%
48	2	2	0	0	yes	44100	46875	46875	6.29%	6.29%
48	3	3	0	0	yes	32000	31250	31250	2.34%	2.34%
48	4	4	1	1	yes	22050	20833.33	20833.33	5.51%	5.51%
48	6	6	0	0	yes	16000	15625	15625	2.34%	2.34%
48	8	8	1	1	yes	11025	11029.41	11029.41	0.04%	0.04%
48	11	11	1	1	yes	8000	8152.17	8152.17	1.90%	1.90%

### 20.4.3 I<sup>2</sup>S send and receive sequence

#### 20.4.3.1 I<sup>2</sup>S initialization process

1. The user can set the SPI\_I2SPREDIV.LDIV [7:0] bits and SPI\_I2SPREDIV.ODD\_EVEN bit to configure the related prescaler and serial clock baud rate;
2. If the user needs the master device to provide the main clock MCLK to the external DAC/ADC audio device, set the SPI\_I2SPREDIV.MCLKOEN = 1. (Calculate LDIV and ODD\_EVEN according to different clock outputs, see section 20.4.2).
3. The user can set the SPI\_I2SCFG.CLKPOL bit to define the polarity of the communication clock when idle; the user can set the SPI\_I2SCFG.MODSEL = 1 to configure the device to be in I<sup>2</sup>S mode, and set SPI\_I2SCFG.MODCFG[1:0] bits to select the I<sup>2</sup>S master-slave mode and transmission direction (send or receive); set SPI\_I2SCFG.STDSEL[1:0] bits to select the corresponding I<sup>2</sup>S standard (under the PCM standard, set the SPI\_I2SCFG.PCMFSYNC bit to select the PCM frame synchronization mode); set SPI\_I2SCFG.TDATLEN [1: 0] bits to select length of data to be transmitted, and select the number of data bits of per channel by set the SPI\_I2SCFG.CHBITS bit;
4. When user needs to enable interrupt or DMA, the configuration operation is the same as SPI;

5. Finally, set the SPI\_I2SCFG.I2SEN = 1 to start I2S communication.

#### 20.4.3.2 Master mode sending process

When I2S works in master mode, the CLK pin outputs the serial clock, the WS pin generates the channel selection signal, and set the SPI\_I2SPR.MCLKOEN bit to select whether to output the master clock (MCLK).

The sending process begins when data is written to the send buffer. When the data of the current channel is moved from the send buffer to the shift register in parallel, the flag bit TE (SPI\_STS.TE) is set to '1'. At this time, the data of the other channel should be written into SPI\_DAT. The channel corresponding to the current data to be transmitted is confirmed by the flag bit CHSIDE (SPI\_STS.CHSIDE). The value of CHSIDE (SPI\_STS.CHSIDE) is updated when TE (SPI\_STS.TE) is set to '1'. A complete data frame includes left and right channels, and only part of the data frame cannot be transmitted. When the flag bit TE (SPI\_STS.TE) is set to '1', if the SPI\_CTRL2.TEINTEN = 1, an interrupt will be generated.

The operation of writing data depends on the selected I2S standard. See chapter 0 for details.

When the user wants to turn off the I2S function, wait for the TE flag (SPI\_STS.TE) bit to be 1 and the BUSY flag (SPI\_STS.BUSY) bit to be 0, and then clear the SPI\_I2SCFG.I2SEN bit to 0.

#### 20.4.3.3 Slave mode sending process

The sending process of the slave mode is similar to that of the master mode, the difference is as follows:

When I2S works in slave mode, there is no need to configure the clock, and the CLK pin and WS pin are connected to the corresponding pins of the master device. The sending process begins when an external master sends a clock signal, and when a WS signal requires data transfer. Only when the slave device is enabled and the data has been written to the I2S data register, the external master device can start communication.

When the first clock edge representing the next data transfer arrives, the new data has not been written into the SPI\_DAT register, an underflow occurs, and the SPI\_STS.UNDER flag bit is set to 1. If the SPI\_CTRL2.ERRINTEN bit is set to 1, an interrupt is generated to indicate that an error has occurred.

The SPI\_STS.CHSIDE flag indicates which channel the currently transmitted data corresponds to. Compared with the master mode sending process, in the slave mode, SPI\_STS.CHSIDE depends on the WS signal of the external master I2S device (WS signal is 1 means the left channel)

#### 20.4.3.4 Master mode receiving process

Audio is always received in 16-bit packets. According to the configured data and channel length, the received audio data will need to be transferred to the receive buffer once or twice.

When the data is transferred from the shift register to the receive buffer, the SPI\_STS.RNE flag bit is set to 1, at this time, the data is ready and can be read from the SPI\_DAT register. If the SPI\_CTRL2.RNEINTEN bit is set to 1, an interrupt will be generated. Reading the SPI\_DAT register to clear the SPI\_STS.RNE flag. If the previously received data is not read, new data is received again, an overflow occurs, and the SPI\_STS.OVER flag is set to 1. If the SPI\_CTRL2.ERRINTEN bit is set to 1, an interrupt is generated to indicate that an error has occurred.

The channel corresponding to the currently transmitted data can be confirmed by the SPI\_STS.CHSIDE bit. When the SPI\_STS.RNE flag bit is set to 1, the SPI\_STS.CHSIDE value is updated.

The operation of reading data depends on the selected I2S standard. See section 0 for details.

When I<sup>2</sup>S function is turned off, different audio standards, data length and channel length adopt different operation steps:

- Data length is 16 bits, channel length is 32 bits (SPI\_I2SCFG.TDATTLEN = 00, SPI\_I2SCFG.CHBITS = 1), LSB alignment standard (SPI\_I2SCFG.STDSEL = 10).
  1. Wait for the penultimate RNE flag (SPI\_STS.RNE) bit to be set to' 1'.
  2. Software delay, waiting for 17 I<sup>2</sup>S clock cycles.
  3. Turn off I<sup>2</sup>S (SPI\_I2SCFG.I2SEN = 0).
- The data length is 16 bits, the channel length is 32 bits (SPI\_I2SCFG.TDATLEN = 00 and SPI\_I2SCFG.CHBITS = 1), the MSB alignment standard (SPI\_I2SCFG.STDSEL = 01), I<sup>2</sup>S Philips standard (SPI\_I2SCFG.STDSEL = 00) or PCM standard (SPI\_I2SCFG.STDSEL = 11)
  1. Wait for the last RNE flag (SPI\_STS.RNE) bit to be set to' 1'.
  2. Software delay, waiting for 1 I<sup>2</sup>S clock cycle.
  3. Turn off I<sup>2</sup>S (SPI\_I2SCFG.I2SEN = 0).
- Other combinations of SPI\_I2SCFG.TDATLEN and SPI\_I2SCFG.CHBITS and any audio mode selected by SPI\_I2SCFG.STDSEL:
  1. Wait for the penultimate RNE flag (SPI\_STS.RNE) bit to be set to' 1'.
  2. Software delay, waiting for 1 I<sup>2</sup>S clock cycle.
  3. Turn off I<sup>2</sup>S (SPI\_I2SCFG.I2SEN = 0).

#### 20.4.3.5 Slave mode receiving process

The receiving process of the slave mode is similar to that of the master mode, with the following differences:

The CHSIDE flag (SPI\_STS.CHSIDE) indicates which channel corresponds to the currently transmitted data. Compared with the master mode receiving process, in the slave mode, SPI\_STS.CHSIDE depends on the WS signal of the external master device. When the I<sup>2</sup>S function is turned off, clear the SPI\_I2SCFG.I2SEN bit to 0 when the SPI\_STS.RNE flag is 1.

#### 20.4.4 Status flag

There are the following 4 flag bits in the SPI\_STS register for monitoring the status of the I<sup>2</sup>S bus.

##### 20.4.4.1 TX buffer empty flag (TE)

When the send buffer is empty, this flag is set to 1, indicating that new data can be written into the SPI\_DAT register. When the send buffer is not empty, this flag is cleared to 0.

##### 20.4.4.2 RX buffer not empty flag (RNE)

When the receive buffer is not empty, this flag is set to 1, indicating that valid data has been received into the receive buffer. When reading the SPI\_DAT register, this flag is set to 0.

#### 20.4.4.3 BUSY flag (BUSY)

When the transfer starts, the BUSY flag (SPI\_STS.BUSY) is set to 1, and when the transfer ends, the BUSY flag (SPI\_STS.BUSY) is set to 0 by hardware (software operation is invalid).

In master receiving mode (SPI\_I2SCFG.MODCFG = 11), the BUSY flag (SPI\_STS.BUSY) is set to 0 during receiving. When the I2S module is turned off or the transmission is completed, this flag is set to 0.

In the slave continuous communication mode, between each data item transmission, the BUSY flag (SPI\_STS.BUSY) goes low in 1 I<sup>2</sup>S clock cycle. Therefore, do not use the BUSY flag (SPI\_STS.BUSY) to handle the sending and receiving of each data item.

#### 20.4.4.4 Channel flag (CHSIDE)

The CHSIDE (SPI\_STS.CHSIDE) bit is used to indicate the channel where the data currently sent and received is located. Under the PCM standard, this flag has no meaning.

In send mode, the flag is updated when the TE flag (SPI\_STS.TE) is set; in receive mode, the flag is updated when the RNE flag (SPI\_STS.RNE) is set. In the process of sending and receiving, if an overflow (SPI\_STS.OVER) or underflow (SPI\_STS.UNDER) error occurs, this flag is meaningless, and the I2S needs to be turned off and then turned on again.

### 20.4.5 Error flag

The SPI\_STS register has 2 error flag bits.

#### 20.4.5.1 Overflow flag (OVER)

When the RNE flag (SPI\_STS.RNE) is set to 1, but there is still data sent to the receive buffer, an overflow error will occur. At this time, the OVER flag (SPI\_STS.OVER) is set to 1. An interrupt will be generated if the user enables the corresponding interrupt. All data received after this time will be lost, and the SPI\_DAT register only retains the previously unread data.

Reading the SPI\_DAT register and the SPI\_STS register in turn to clear the SPI\_STS.OVER bit.

#### 20.4.5.2 Underflow flag (UNDER)

In slave send mode, when the first clock edge of sending data arrives, if the send buffer is still empty, the UNDER flag (SPI\_STS.UNDER) is set to 1. An interrupt will be generated if the user enables the corresponding interrupt.

Reading the SPI\_STS register to clears the SPI\_STS.UNDER bit.

### 20.4.6 I2S interrupt

The following table lists all I<sup>2</sup>S interrupts.

Table 20-3 I<sup>2</sup>S interrupt request

Interrupt event	Event flag bit	Enable control bit
Send buffer empty flag	TE	TEINTEN
Receive buffer non empty flag	RNE	RNEINTEN

Interrupt event	Event flag bit	Enable control bit
Underflow flag bit	UNDER	ERRINTEN
Overflow flag bit	OVER	

#### 20.4.7 DMA function

Working in I2S mode, it does not need data transmission protection function, so it does not need to support CRC, other DMA functions are the same as SPI mode.

## 20.5 SPI and I2S register description

### 20.5.1 SPI register overview

Table 20-4 SPI register overview

### 20.5.2 SPI control register 1 (SPI\_CTRL1) (not used in I2S mode)

Address: 0x00

Reset value: 0x0000

15 BIDIR MODE	14 BIDIR OEN	13 CRCEN	12 CRC NEXT	11 DATFF	10 RONLY	9 SSMEN	8 SSEL	7 LSBFF	6 SPIEN	5 BR[2:0]	3	2 MSEL	1 CLKPOL	0 CLKPHA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit field	Name	Description
15	BIDIRMODE	<p>Bidirectional data mode enable</p> <p>0: Select the "two-wire one-way" mode.</p> <p>1: Select the "one-wire bidirectional" mode.</p> <p><i>Note: Not used in I<sup>2</sup>S mode.</i></p>
14	BIDIROEN	<p>Output enable in bidirectional mode</p> <p>0: Output disable (receive-only mode).</p> <p>1: Output enabled (send-only mode).</p> <p>In master mode, the "one-wire" data line is the MOSI pin, and in slave mode, the "one-wire" data line is the MISO pin.</p> <p><i>Note: Not used in I<sup>2</sup>S mode.</i></p>
13	CRCEN	<p>Hardware CRC check enable</p> <p>0: Disable CRC calculation.</p> <p>1: Enable CRC calculation.</p> <p><i>Note: This bit can only be written when SPI is disabled (SPI_CTRL1.SPIEN = 0), otherwise an error will occur.</i></p> <p>This bit can only be used in full duplex mode.</p> <p><i>Note: Not used in I<sup>2</sup>S mode.</i></p>
12	CRCNEXT	<p>Send CRC next</p> <p>0: The next sent value comes from the send buffer.</p> <p>1: The next send value comes from the CRC register.</p> <p><i>Note: This bit should be set immediately after the last data is written in SPI_DAT register.</i></p> <p><i>Note: Not used in I<sup>2</sup>S mode.</i></p>
11	DATFF	<p>Data frame format</p> <p>0: 8-bit data frame format is used for sending/receiving.</p> <p>1: 16-bit data frame format is used for sending/receiving.</p> <p><i>Note: This bit can only be written when SPI is disabled (SPI_CTRL1.SPIEN = 0), otherwise an error will occur.</i></p> <p><i>Note: Not used in I<sup>2</sup>S mode.</i></p>
10	RONLY	<p>Only receive mode</p> <p>This bit, together with the SPI_CTRL1.BIDIRMODE bit, determines the transfer direction in two-wire one-way mode. In the application scenario of multiple slave devices, this bit is only set to 1 by the accessed slave device, and only the accessed slave device can output, so as to avoid data line conflicts.</p> <p>0: Full duplex (sending mode and receiving mode).</p> <p>1: Disable output (receive-only mode).</p> <p><i>Note: Not used in I<sup>2</sup>S mode.</i></p>

Bit field	Name	Description
9	SSMEN	<p>Software slave device management</p> <p>When the SPI_CTRL1.SSMEN bit is set to 1, the NSS pin level is determined by the value of the SPI_CTRL1.SSEL bit.</p> <p>0: Disable software slave device management.</p> <p>1: Enable software slave device management.</p> <p><i>Note: Not used in I<sup>2</sup>S mode.</i></p>
8	SSEL	<p>Internal slave device selection</p> <p>This bit only has meaning when the SPI_CTRL1.SSMEN bit is set. It determines the NSS level, and I/O operations on the NSS pin have no effect.</p> <p><i>Note: Not used in I<sup>2</sup>S mode.</i></p>
7	LSBFF	<p>Frame format</p> <p>0: Send MSB first.</p> <p>1: Send LSB first.</p> <p><i>Note: This bit cannot be changed during communication.</i></p> <p><i>Note: Not used in I<sup>2</sup>S mode.</i></p>
6	SPIEN	<p>SPI enable</p> <p>0: Disable SPI device.</p> <p>1: Enable the SPI device.</p> <p><i>Note: Not used in I<sup>2</sup>S mode.</i></p> <p><i>Note: When turning off the SPI device, please follow paragraph 0 section's procedure operation.</i></p>
5:3	BR[2:0]	<p>Baud rate control</p> <p>000: fPCLK/2</p> <p>001: fPCLK/4</p> <p>010: fPCLK/8</p> <p>011: fPCLK/16</p> <p>100: fPCLK/32</p> <p>101: fPCLK/64</p> <p>110: fPCLK/128</p> <p>111: fPCLK/256</p> <p><i>Note: This bit cannot be changed during communication.</i></p> <p><i>Note: Not used in I<sup>2</sup>S mode.</i></p>
2	MSEL	<p>Master device selection</p> <p>0: Configure as the slave device.</p> <p>1: Configure as the master device.</p> <p><i>Note: This bit cannot be changed during communication.</i></p> <p><i>Note: Not used in I<sup>2</sup>S mode.</i></p>
1	CLKPOL	<p>Clock polarity</p> <p>0: In idle state, SCLK remains low.</p> <p>1: In idle state, SCLK remains high.</p> <p><i>Note: This bit cannot be changed during communication.</i></p>

Bit field	Name	Description
		<i>Note: Not used in I<sup>2</sup>S mode.</i>
0	CLKPHA	<p>Clock phase            0: Data sampling starts from the first clock edge.            1: Data sampling starts at the second clock edge.</p> <p><i>Note: This bit cannot be modified while communication is in progress.</i></p> <p><i>Note: Not used in I<sup>2</sup>S mode.</i></p>

### 20.5.3 SPI control register 2 (SPI\_CTRL2)

Address: 0x04

Reset value: 0x0000

15	Reserved				8	7	6	5	4	3	2	1	0
						rw	rw	rw		rw	rw	rw	rw

Bit field	Name	Description
15:8	Reserved	Reserved, the reset value must be maintained
7	TEINTEN	<p>Send buffer empty interrupt enable            0: Disable TE interrupt.            1: Enable TE interrupt, and interrupt request is generated when TE flag (SPI_STS.TE) is set to '1'.</p>
6	RNEINTEN	<p>Receive buffer non-empty interrupt enable            0: Disable RNE interrupt.            1: Enable RNE interrupt, and generate interrupt request when RNE flag (SPI_STS.RNE) is set to '1'.</p>
5	ERRINTEN	<p>Error interrupt enable            When an error (SPI_STS.CRCERR, SPI_STS.OVER, SPI_STS.UNDER, SPI_STS.MODERR) is generated, this bit controls whether an interrupt is generated            0: Disable error interrupt.            1: Enable error interrupt.</p>
4:3	Reserved	Reserved, the reset value must be maintained
2	SSOEN	<p>NSS output enable            0: Disable NSS output in master mode, the device can work in multi-master mode.            1: When the device is turned on, enable NSS output in the master mode, the device cannot work in the multi-master device mode.</p> <p><i>Note: Not used in I<sup>2</sup>S mode.</i></p>
1	TDMAEN	<p>Send buffer DMA enable            When this bit is set, a DMA request is issued as soon as the TE flag (SPI_STS.TE) is set            0: Disable send buffer DMA.            1: Enable send buffer DMA.</p>

Bit field	Name	Description
0	RDMAEN	<p>Receive buffer DMA enable</p> <p>When this bit is set, a DMA request is issued as soon as the RNE flag (SPI_STS.RNE) is set</p> <p>0: Disable receive buffer DMA.</p> <p>1: Enable receive buffer DMA.</p>

## 20.5.4 SPI status register (SPI\_STS)

Address: 0x08

Reset value: 0x0002

15	Reserved	8	7	6	5	4	3	2	1	0
			r	r	r	rc_w0	r	r	r	r

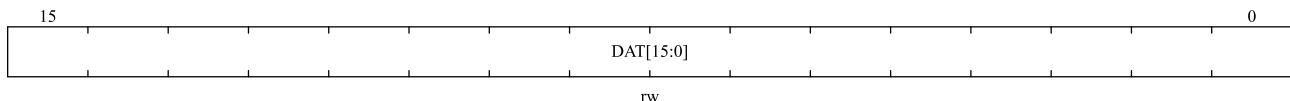
Bit field	Name	Description
15:8	Reserved	Reserved, the reset value must be maintained
7	BUSY	<p>Busy flag</p> <p>0: SPI is not busy.</p> <p>1: SPI is busy communicating or the send buffer is not empty.</p> <p>This bit is set or reset by hardware.</p> <p><i>Note: Use of this flag requires special attention, see section 20.3.3 and section 20.4.4.3 for details..</i></p>
6	OVER	<p>Overflow flag</p> <p>0: No overflow error.</p> <p>1: An overflow error occurred.</p> <p><i>Note: This bit is set by hardware and cleared according to the sequence of software operations. For more information about software sequences, refer to 20.3.7.2 for details.</i></p>
5	MODERR	<p>Mode error</p> <p>0: No mode error.</p> <p>1: A mode error occurred.</p> <p><i>Note: This bit is set by hardware and cleared according to the sequence of software operations. For more information about software sequences, refer to 20.3.7 for details.</i></p> <p><i>Note: Not used in I<sup>S</sup> mode.</i></p>
4	CRCERR	<p>CRC error flag</p> <p>0: The received CRC value matches the value the SPI_CRCRDAT register value.</p> <p>1: The received CRC value does not match the SPI_CRCRDAT register value.</p> <p><i>Note: this bit is set by hardware and cleared by software by writing 0.</i></p> <p><i>Note: Not used in I<sup>S</sup> mode.</i></p>
3	UNDER	<p>Underflow flag</p> <p>0: No underflow occurred.</p> <p>1: Underflow occurred.</p> <p><i>Note: This bit is set by hardware and cleared according to the sequence of software operations.</i></p>

Bit field	Name	Description
		<p><i>For more information about software sequences, refer to 0 for details.</i></p> <p><i>Note: not used in SPI mode.</i></p>
2	CHSIDE	<p>Channel</p> <p>0: The left channel needs to be sent or received; 1: The right channel needs to be sent or received.</p> <p><i>Note: not used in SPI mode. No meaning in PCM mode.</i></p>
1	TE	<p>The send buffer is empty</p> <p>0: The send buffer is not empty. 1: The send buffer is empty.</p>
0	RNE	<p>Receive buffer is not empty</p> <p>0: The receive buffer is empty. 1: The receive buffer is not empty.</p>

## 20.5.5 SPI data register (SPI\_DAT)

Address: 0x0C

Reset value: 0x0000



Bit field	Name	Description
15:0	DAT[15:0]	<p>Data register</p> <p>Data to be sent or received</p> <p>The data register corresponds to two buffers: one for write (send buffer); The other is for read (receive buffer). Write operation writes data to send buffer; The read operation will return the data in the receive buffer.</p> <p>Note on SPI mode: According to the selection of the data frame format by the SPI_CTRL1.DATFF bit, the data sending and receiving can be 8-bit or 16-bit. To ensure correct operation, the data frame format needs to be determined before enabling the SPI.</p> <p>For 8-bit data, the buffer is 8-bit, and only SPI_DAT[7:0] is used when sending and receiving. When receiving, SPI_DAT[15:8] is forced to 0.</p> <p>For 16-bit data, the buffer is 16-bit, and the entire data register is used when sending and receiving, that is, SPI_DAT[15:0].</p>

## 20.5.6 SPI CRC polynomial register (SPI\_CRCPOLY) (not used in I<sup>2</sup>S mode)

Address: 0x10

Reset value: 0x0007

Diagram of register CRCPOLY[15:0]. The register is shown as a horizontal bar divided into 16 segments, indexed from 15 on the left to 0 on the right. The center segment is labeled "CRCPOLY[15:0]" above the bar.

Bit field	Name	Description
15:0	CRCPOLY [15:0]	<p>CRC polynomial register</p> <p>This register contains the polynomial used for the CRC calculation.</p> <p>The reset value is 0x0007, other values can be set according to the application.</p> <p><i>Note: not used in I<sup>2</sup>S mode.</i></p>

#### 20.5.7 SPI RX CRC register (SPI\_CRCRDAT) (not used in I<sup>2</sup>S mode)

Address offset: 0x14

Reset value: 0x0000

The diagram shows the **CRCRDAT[15:0]** register structure. It consists of a horizontal line representing the register, with numerical labels at both ends: 15 on the left and 0 on the right. The center of the register is labeled **CRCRDAT[15:0]**. Below the register line, there is a small vertical tick mark labeled **r**, indicating a read operation.

Bit field	name	describe
15:0	CRCRDAT	<p>Receive CRC register</p> <p>When CRC calculation is enabled, CRCRDAT[15:0] will contain the calculated CRC value of subsequent received bytes. This register is reset when ‘1’ is written to the SPI_CTRL1.CRCEN bit. The CRC calculation uses the polynomial in SPI_CRCPOLY.</p> <p>When the data frame format is set to 8 bits, only the lower 8 bits participate in the calculation and follow the CRC8 standard; when the data frame format is 16 bits, all 16 bits in the register participate in the calculation and follow the CRC16 standard.</p> <p><i>Note: reading this register when the BUSY flag (SPI_STS.BUSY) is '1' may read incorrect values.</i></p> <p><i>Note: not used in I<sup>2</sup>S mode.</i></p>

### 20.5.8 SPI TX CRC register (SPI\_CRCTDAT)

Address offset: 0x18

Reset value: 0x0000

The diagram shows a horizontal register structure for CRCTDAT[15:0]. It consists of 16 vertical lines representing bits, labeled from 15 on the left to 0 on the right. The label "CRCTDAT[15:0]" is centered above the register. Below the register, the letter "r" is positioned, indicating it is a readable register.

Bit field	Name	Description
15:0	CRCTDAT	<p>Send CRC register</p> <p>When CRC calculation is enabled, CRCTDAT[15:0] contains the CRC value calculated by the bytes sent subsequently. This register is reset when ‘1’ is written to the SPI_CTRL1.CRCEN bit.</p>

Bit field	Name	Description
		<p>The CRC calculation uses the polynomial in SPI_CRCPOLY.</p> <p>When the data frame format is set to 8 bits, only the lower 8 bits participate in the calculation and follow the CRC8 standard; when the data frame format is 16 bits, all 16 bits in the register participate in the calculation and follow the CRC16 standard.</p> <p><i>Note: reading this register when the BUSY flag (SPI_STS.BUSY) is '1' may read incorrect values.</i></p> <p><i>Note: not used in I<sup>2</sup>S mode.</i></p>

## 20.5.9 SPI\_I<sup>2</sup>S configuration register (SPI\_I2SCFG)

Address offset: 0x1c

Reset value: 0x0000

15	Reserved	12	MODSEL	I2SEN	MODCFG[1:0]	PCMFSYNC	6	Reserved	5	STDSEL[1:0]	4	CLKPOL	3	TDATLEN[1:0]	2	1	0
				rw	rw	rw				rw		rw		rw	rw	rw	rw

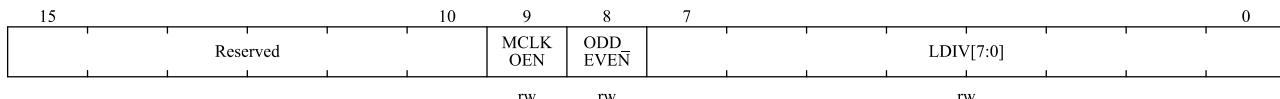
Bit field	Name	Description
15:12	Reserved	Reserved, the reset value must be maintained
11	MODSEL	<p>I<sup>2</sup>S mode selection</p> <p>0: Select SPI mode.</p> <p>1: Select I<sup>2</sup>S mode.</p> <p><i>Note: this bit can only be set when SPI or I<sup>2</sup>S is turned off.</i></p>
10	I <sup>2</sup> SEN	<p>I<sup>2</sup>S enable</p> <p>0: Disable I<sup>2</sup>S.</p> <p>1: Enable I<sup>2</sup>S.</p> <p><i>Note: not used in SPI mode.</i></p>
9:8	MODCFG	<p>I<sup>2</sup>S mode setting</p> <p>00: Slave device sends.</p> <p>01: Slave device receives.</p> <p>10: Master device sends.</p> <p>11: Master device receives.</p> <p><i>Note: This bit can only be set when I<sup>2</sup>S is turned off.</i></p> <p><i>Note: not used in SPI mode.</i></p>
7	PCMFSYNC	<p>PCM frame synchronization</p> <p>0: Short frame synchronization.</p> <p>1: Long frame synchronization.</p> <p><i>Note: This bit is only meaningful when SPI_I2SCFG.STDSEL = 11 (used by the PCM standard).</i></p> <p><i>Note: not used in SPI mode.</i></p>
6	Reserved	Reserved, the reset value must be maintained
5:4	STDSEL	Selection of I <sup>2</sup> S standard
		00: I <sup>2</sup> S Philips standard.

Bit field	Name	Description
		<p>01: High byte alignment standard (left alignment).      10: Low byte alignment standard (right alignment).      11: PCM standard.      See for details of I<sup>2</sup>S standard on section 20.4.1.</p> <p><i>Note: For correct operation, this bit can only be set when I<sup>2</sup>S is turned off.</i></p> <p><i>Note: not used in SPI mode.</i></p>
3	CLKPOL	<p>Static clock polarity      0: I2S clock static state is low level.      1: I2S clock static state is high level.</p> <p><i>Note: For correct operation, this bit can only be set when I<sup>2</sup>S is turned off.</i></p> <p><i>Note: not used in SPI mode.</i></p>
2:1	TDATLEN	<p>Length of data to be transmitted      00: 16-bit data length.      01: 24-bit data length;      10: 32-bit data length;      11: Not allowed.</p> <p><i>Note: For correct operation, this bit can only be set when I<sup>2</sup>S is turned off.</i></p> <p><i>Note: not used in SPI mode.</i></p>
0	CHBITS	<p>Channel length (number of data bits per audio channel)      0: 16 bits wide;      1: 32 bits wide.</p> <p>Writing to this bit is meaningful only when SPI_I2SCFG.TDATLEN = 00, otherwise the channel length is fixed to 32 bits by hardware.</p> <p><i>Note: For correct operation, this bit can only be set when I<sup>2</sup>S is turned off.</i></p> <p><i>Note: not used in SPI mode.</i></p>

### 20.5.10 SPI\_I<sup>2</sup>S prescaler register (SPI\_I2SPREDIV)

Address: 0x20

Reset value: 0x0002



Bit field	Name	Description
15:10	Reserved	Reserved, the reset value must be maintained
9	MCLKOEN	<p>Master clock output enable      0: Disable master clock output.      1: Enable master clock output.</p> <p><i>Note: For correct operation, this bit can only be set when I<sup>2</sup>S is turned off.</i></p> <p><i>Note: not used in SPI mode.</i></p>

Bit field	Name	Description
8	ODD_EVEN	<p>Odd coefficient prescaler</p> <p>0: Actual frequency division factor = LDIV × 2.</p> <p>1: Actual frequency division factor = (LDIV × 2) + 1.</p> <p>See section 20.4.2 for details.</p> <p><i>Note: For correct operation, this bit can only be set when I<sup>2</sup>S is turned off.</i></p> <p><i>Note: not used in SPI mode.</i></p>
7:0	LDIV	<p>I<sup>2</sup>S linear prescaler</p> <p>Disable setting LDIV [7:0] = 0 or LDIV [7:0] = 1</p> <p>See Section 20.4.2 for details.</p> <p><i>Note: For correct operation, this bit can only be set when I<sup>2</sup>S is turned off.</i></p> <p><i>Note: not used in SPI mode.</i></p>

## 21 Real-time clock (RTC)

### 21.1 Description

- The real-time clock (RTC) is an independent BCD timer/counter.
- Daylight saving time compensation supported by software.
- A periodic automatic programmable wakeup timer.
- Two 32-bit registers contain the seconds, minutes, hours, day (day of week), date (day of month), month, and year.
- Independent 32-bit register contain sub-seconds value.
- Two programmable alarms.
- Two 32-bit registers contain two programmable alarms seconds, minutes, hours, day,of week.
- Two 32-bit registers contain two programmable alarms sub-seconds.
- Digital calibration function.
- Reference clock detection: a more precise external source clock (50 or 60 Hz) can be used to improve the calendar precision.
- 2 tamper detection events with configurable filter and internal pull-up.
- Time-Stamp function.
- Multiple Wakeup sources of Interrupt/Event. These include Alarm A, Alarm B, Wakeup Timer, Time-Stamp, Tamper.
- After RTC is enabled by the RCC register and voltage remains in the operating range, RTC will not stop timing in RUN mode, LPRUN mode, SLEEP mode and STOP mode.
- RTC provides a variety of ways to wakeup from SLEEP mode and STOP mode.

### 21.2 Specification

Table 21-1 RTC feature support

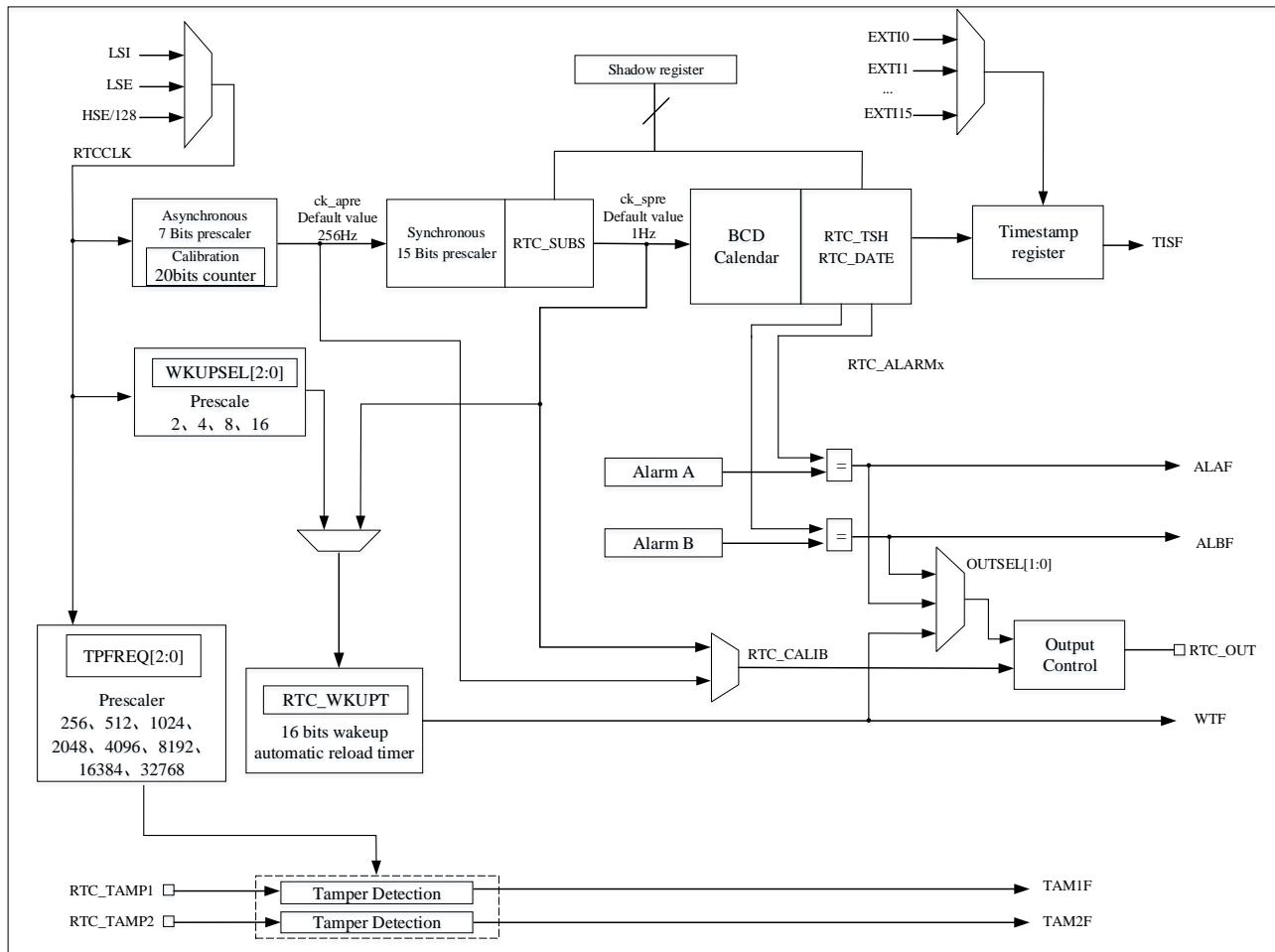
Main function	Description
Clock	RTC clock can be selected from LSI, LSE and HSE, which are 30KHz, 32.768KHz and HSE / 128 respectively
Calendar	Calendar consists of sub second, second, minute, hour (12 or 24 format), day (day of the week), date, month and year. These data are stored in the shadow register of APB module.
Wakeup Timer	Output “RTC_OUT” can be configured to send wakeup events to GPIO. At the same time, it also can be configured as an interrupt/event to wake up the system from SLEEP, STOP modes.

Main function	Description
Alarm	Programmable alarm clock and interrupt function. The alarm can be triggered by any combination of the calendar fields. When the alarm event occurs the alarm flag can be sent to GPIO through “RTC_OUT”, and it also can be used to wake up the CPU or exit from the low power status such as SLEEP, STOP modes.
Tamper	2 Tamper detection logic are a source of system Wakeup should a Tamper event happen on one of the input lines. It is also a source of hardware trigger to LP Timer.
Timestamp	Time-stamp function for GPIO event saving. It is a source to Wakeup system from low power modes. Alternatively a tamper event could be a source of Time-stamp event.
Interrupts/events	Alarm A/Alarm B interrupt Wakeup interrupt Timestamp interrupt Tamper interrupt

## 21.3 RTC function description

### 21.3.1 RTC block diagram

Figure 21-1 RTC block diagram



RTC includes the following functions:

- Alarm A and Alarm B event/interrupt
- Timestamp event/interrupt
- Tamper event/interrupt
- RTC output functions:
  - ◆ 256 Hz or 1Hz clock output (LSE frequency is 32.768 kHz).
  - ◆ Alarm clock output (polarity configurable), Alarm A and Alarm B are optional.
  - ◆ Auto wakeup output (polarity configurable).
- RTC input functions:

- ◆ Timestamp event detection
- ◆ 50 or 60Hz reference clock input
- ◆ Tamper event detection

### 21.3.2 GPIOs of RTC

Timestamp input come from IOM (mapped to PC13) or EXTI module, if EXIT module is needed to start, please refer to the timestamp trigger source selection register (EXTI\_TS\_SEL) for details.

RTC\_OUT (Alarm, Wakeup event or calibration output (256Hz or 1Hz)) is mapped to PC13. Regardless of the PC13 GPIO configuration, the PC13 pin configuration is controlled by the RTC as an output.

PC13 can be used as RTC TAMPER1 tamper detection pin, PA0 can be used as RTC TAMPER2 tamper detection pin, Controlled by the RTC as a pull-up input.

PA10 or PB15 can be used as RTC\_REFCLKIN reference clock input pin.

### 21.3.3 RTC register write protection

After power-up or reset, all RTC registers except RTC\_CTRL, RTC\_TMPCFG, RTC\_INITSTS[13:8] are write-protected. Writes to the RTC registers are enabled by writing a key to the write protection register RTC\_WRP. All write protection RTC registers require the following steps to unlock write protection:

- Write “0xCA” into RTC\_WRP register.
- Write “0x53” into RTC\_WRP register.

After unlocking these registers, it cannot be write protected unless the RTC is soft reset or power cycled. The unlocking mechanism only checks the write operation to the RTC\_WRP register. During or before and after the unlocking process, the write operation to other registers does not affect the unlocking result.

### 21.3.4 RTC clock and prescaler

RTC clock source:

- LSE clock
- LSI clock
- HSE/128 clock

For the purpose of reduction of power consumption, the prescaler is divided into 2 programmable prescalers, they are asynchronous prescaler and synchronous prescaler. If both prescaler are used, it is recommended that the value of the asynchronous divider be as large as possible.

- A 7-bit asynchronous prescaler which is given by RTC\_PRE.DIVA[6:0] bits
- A 15-bit synchronous prescaler which is given by RTC\_PRE.DIVS[14:0] bits

The formula for  $f_{ck\_apre}$  and  $f_{ck\_spre}$  are given below:

$$f_{ck\_apre} = \frac{f_{RTCCLK}}{RTC\_PRE.DIVA[6:0]+1}$$

$$f_{ck\_spre} = \frac{f_{RTCCLK}}{(RTC\_PRE.DIVS[14:0]+1)*(RTC\_PRE.DIVA[6:0]+1)}$$

The ck\_apre clock is used to driven RTC\_SUBS sub-second down counter. When it reaches 0, reload RTC\_SUBS with the value of RTC\_PRE.DIVS[14:0].

### 21.3.5 RTC calendar

There are three shadow registers, they are RTC\_DATE, RTC\_TSH and RTC\_SUBS. The RTC time and date registers can be accessed through the shadow registers. It is also possible to access them directly to avoid the synchronization waiting time. The three shadow registers are as follow:

- RTC\_DATE: set and read date
- RTC\_TSH: set and read time
- RTC\_SUBS: read sub-second

After every two RTCCLK cycles, the current calendar value is copied to the shadow register, and RTC\_INITSTS.RSYF bit is set to 1. This process is not performed in low power (STOP) modes. While exiting these modes, the shadow register updates the values after 2 RTCCLK cycles.

By default, when user try to access the calendar register, it accesses the contents of the shadow register instead. User can access the calendar register directly by setting the RTC\_CTRL.BYPS bit.

When RTC\_CTRL.BYPS=0, calendar values are from shadow registers, when reading RTC\_SUBS, RTC\_TSH or RTC\_DATE register, it is necessary to make ensure the frequency of APB1 clock ( $f_{APB1}$ ) is at least 7 times the frequency of RTC clock ( $f_{RTCCLK}$ ), and APB1 clock frequency lower than RTC clock frequency is not allowed in any case. System reset will reset shadow registers.

*Note: If the sub-second matching interrupt is configured, the first sub-second matching interrupt may not be generated, and the subsequent sub-second matching interrupts are normal, and the first sub-second matching interrupt can be ignored.*

### 21.3.6 Calendar initialization and configuration

The value of prescaler and calendar can be initialized by the following steps:

- Enter initialization mode by setting “1” to RTC\_INITSTS.INITM bit, then wait for RTC\_INITSTS.INITF flag to be set 1.
- Set RTC\_PRE.DIVS[14:0] and RTC\_PRE.DIVA[6:0] value.
- Write the initial calendar values include time and date into the shadow registers (RTC\_TSH and RTC\_DATE) and configure the time format (12 or 24 hours) by the RTC\_CTRL.HFMT bit.
- Exit initialization mode by clearing the RTC\_INITSTS.INITM bit.

The values of calendar counter will automatically loaded from shadow registers after 4 RTCCLK clock cycles, then the calendar counter restarts.

Note: The minimum time before re-entering the initialization mode above should be 1 second apart, as the internal registers use 1Hz clock to update.

*Note: Before entering the RTC initialization mode, ensure that the value of RTC\_SUBS.SS[15:0] is not less than 2, and read RTC\_DATE register once.*

### 21.3.7 Calendar reading

#### 1. Reading calendar value when RTC\_CTRL.BYPS=0

Calendar value is read from shadow registers if RTC\_CTRL.BYPS=0. In order to read RTC calendar registers (RTC\_SUBS, RTC\_TSH and RTC\_DATE) correctly, APB1 clock frequency must be set equal to or greater than 7 times of RTC clock frequency. In any case, APB1 clock frequency must not be less than RTC clock frequency.

If APB1 clock frequency is not equal to or greater than 7 times of RTC clock frequency, refer to the following process to read calendar value.

- Read the data of RTC\_SUBS, RTC\_TSH and RTC\_DATE twice.
- Compare the data read twice, if they are equal, the read data can be considered correct; if they are not equal, read the data for the third time.
- The third time read data can be considered correct.

Shadow registers (RTC\_SUBS, RTC\_TSH and RTC\_DATE) are updated every two RTCCLK cycles. If user want to read calendar value in a short time(less than two RTCCLK cycles), RTC\_INITSTS.RSYF bit must be cleared by software after the first time read.

In some cases, it is necessary to wait until RTC\_INITSTS.RSYF bit is set 1 before read calendar value.

- After waking up from the low power modes (STOP mode), clear RTC\_INITSTS.RSYF bit, then wait RTC\_INITSTS.RSYF bit is set again.
- System reset.
- Calendar complete initialization.
- Calendar complete synchronization.

#### 2. Reading calendar value when RTC\_CTRL.BYPS=1

Reading the calendar value directly from the calendar counter if RTC\_CTRL.BYPS=1. The advantage of this configuration is that read calendar value without delay after wakeup from the low power mode, the disadvantage is that these data of RTC\_SUBS, RTC\_TSH and RTC\_DATE may not be at a time.

To ensure the correctness of read calendar value, it is necessary to read RTC\_SUBS, RTC\_TSH and RTC\_DATE twice, then compare the data read twice, if they are equal, the read data can be considered correct.

*Note: After read RTC\_TSH or RTC\_SUBS register, it needs to read RTC\_DATE register once.*

### 21.3.8 Calibration clock output

When RTC\_CTRL.COEN set to 1, PC13 pin will output calibration clock. If RTC\_CTRL.CALOSEL=0 and RTC\_PRE.DIVA[6:0]=0x7F, the RTC\_CALIB frequency results is  $f_{RTCCLK}/RTC\_PRE.DIVA[6:0]$ . This is equivalent to a calibration output of 256 Hz when the RTCCLK frequency is 32.768 kHz. The rising edge is recommended for there is slight jitter on the falling edge.

When RTC\_CTRL.CALOSEL=1 and "RTC\_PRE.DIVS[14:0]+1" is a non-zero integer multiple of 256, the RTC\_CALIB frequency is given by the formula  $f_{RTCCLK}/(256 * (DIVA+1))$ . This is equivalent to 1Hz calibration output when the RTCCLK frequency is 32.768 kHz and RTC\_PRE.DIVA[6:0] = 0x7F.

*Note: When the RTC\_CALIB or RTC\_ALARM output is selected, the RTC\_OUT pin (PC13) is automatically configured as output.*

### 21.3.9 Programmable alarms

RTC has 2 programmable alarms: Alarm A and Alarm B.

RTC alarm can be enabled or disable by RTC\_CTRL.ALxEN bit. If all the alarm value match the calendar values, the RTC\_INITSTS.ALxF flag will be set. Each calendar field can be selected to trigger alarm interrupt if RTC\_CTRL.ALxIEN bit is enabled.

Alarm output: Alarm A and Alarm B can be mapped to RTC\_ALxRM output when RTC\_CTRL.OUTSEL[1:0] is selected, and output polarity can be configured by RTC\_CTRL.OPOL bit.

Note: If the second field is selected (RTC\_ALARMx.MASK1 bit reset), RTC\_PRE.DIVS[14:0] must be larger than 3 to ensure correct operation.

### 21.3.10 Alarm configuration

Alarm A and Alarm B should be configured in the following below:

- Disable Alarm A/Alarm B by clearing RTC\_CTRL.ALAEN/RTC\_CTRL.ALBEN bit.
- Configure the Alarm x registers (RTC\_ALRMxSS/RTC\_ALARMx)
- Enable Alarm A/Alarm B interrupt by set RTC\_CTRL.ALAIEN/RTC\_CTRL.ALBIEEN bit(this step can be selected as needed )
- Enable Alarm A/Alarm B by setting RTC\_CTRL.ALAEN/ RTC\_CTRL.ALBEN bit.

### 21.3.11 Alarm output

When RTC\_CTRL.OUTSEL[1:0] !=0, RTC\_ALARM alternate function output is enable. There are Alarm A output, Alarm B output and Wakeup output to choose by the value of RTC\_CTRL.OUTSEL[1:0] bits.

RTC\_CTRL.OPOL bit control the polarity of the Alarm A, Alarm B or Wakeup output.

When RTC\_CALIB or RTC\_ALARM output is selected, the RTC\_OUT pin (PC13) is automatically configured as

output.

### 21.3.12 Periodic automatic wakeup

A 16-bit programmable auto-load down counter can generate periodic wakeup flag when reach 0. It is also can be extend the range of wakeup timer to 17 bits. Periodic automatic wakeup can be enabled by setting RTC\_CTRL.WTEN.

There are two wake-up input clock sources can be selected:

- RTC clock (RTCCLK) divided by 2/ 4/8/16.

Assume RTCCLK comes from LSE (32.768KHz), wake-up interrupt period can be configured range from 122us to 32s under the resolution down to 61us.

- Internal clock ck\_spre.

Assume ck\_spre frequency is 1Hz, the available wake-up time range from 2s to 36h, and the resolution is 1 second.

- ◆ When RTC\_CTRL.WKUPSEL [2:0] =10x, the period is range from 2s to 18h.
- ◆ When RTC\_CTRL.WKUPSEL [2:0] = 11x, the period is range from 18h to 36h.

After RTC\_CTRL.WTEN bit is set to 1, the down counter is running and when it reaches 0, RTC\_INITSTS.WTF will be set and the device can exit from low power mode when the periodic wakeup interrupt is enabled by setting the RTC\_CTRL.WTIEN bit.

Periodic wakeup output: periodic wakeup can be mapped to RTC\_ALxRM output when RTC\_CTRL.OUTSEL[1:0] is selected, the RTC\_OUT pin(PC13) is automatically configured as output, and output polarity can be configured by RTC\_CTRL.OPOL bit.

### 21.3.13 Wakeup timer configuration

The wakeup timer automatic reload value should be configured in the following below:

- Disable wakeup timer by clearing RTC\_CTRL.WTEN bit, then wait for RTC\_INITSTS.WTWF flag to be set 1.
- Select wake up timer clock by set RTC\_CTRL.WKUPSEL[2:0] bits.
- Configure the wake-up automatic reload value by set RTC\_WKUPT.WKUPT[15:0] bits.
- Enable Wakeup interrupt by set RTC\_CTRL.WTIEN bit(this step can be selected as needed )
- Enable wakeup timer by setting RTC\_CTRL.WTEN bit

### 21.3.14 Timestamp function

Timestamp can be enabled by setting RTC\_CTRL.TSEN bit to 1. When a timestamp event is detected on the RTC\_TS pin, the calendar values of the event will be stored in the timestamp register (RTC\_TSSS, RTC\_TST, RTC\_TSD), and RTC\_INITSTS.TISF is set to 1. Timestamp event can generate an interrupt if RTC\_CTRL.TSIEN is set to 1. If a new timestamp event is detected when RTC\_INITSTS.TISF has been set to 1 already, the hardware sets RTC\_INITSTS.TISOVF flag to 1, and the timestamp registers (RTC\_TST and RTC\_TSD) will continue to hold the

value of the previous event, which means timestamp registers(RTC\_TST and RTC\_TSD) data will not change when RTC\_INITSTS.TISF=1.

After the timestamp event caused by the synchronization process occurs again, RTC\_INITSTS.TISF is set to 1 in 2 RTC\_CLK cycles. There is no delay in the generation of RTC\_INITSTS.TISOVF. This means that if two timestamp events are very close, this can cause RTC\_INITSTS.TISOVF to be "1" and RTC\_INITSTS.TISF to be "0". Therefore, after detecting that RTC\_INITSTS.TISF is "1", then detect RTC\_INITSTS.TISOVF bit.

Tamper event can trigger timestamp event when RTC\_TMPCFG.TPTS bit is set to 1.

If timestamp events are enabled, the timestamp will capture the calendar read in the timestamp register. When both tamper events and timestamp events are enabled, tamper events can also result in timestamp capture. Timestamp events can be generated on any of the 16 GPIO ports selected by EXTI. The GPIO pins in each port are selected by setting the corresponding EXTI\_TS\_SEL.TSSEL[3:0] bits.

### 21.3.15 Tamper detection

There are 2 tamper detection pin, RTC\_TAMP1 pin is PC13, RTC\_TAMP2 pin is PA0. RTC\_TAMPx pin can be used as tamper event detection function input pin. There are two detection modes, edge detection mode and level detection mode with configurable filtering function.

#### Tamper detection initialization

There are 2 tamper detection pins, each of them can be configured independently. User need to configure tamper detection before enable RTC\_TMPCFG.TPxEN bit. When the tamper event is detected after tamper detection is enable, if RTC\_TMPCFG.TPxINTEN is set to 1, tamper event can generate an interrupt and RTC\_INITSTS.TAMxF bit will be set 1.

When RTC\_INITSTS.TAMxF is set to 1, a new tamper event on the same pin cannot be detected.

#### Timestamp on tamper event

Any tamper event can cause a timestamp event when RTC\_INITSTS.TPTS is set to 1, and RTC\_INITSTS.TISF bit and RTC\_INITSTS.TISOVF bit will be set as a normal timestamp event.

#### Edge detection of tamper input

When RTC\_TMPCFG.TPFLT[1:0] bits set to 0, tamper detection is set to edge detection, and one of rising edge or falling edge is controlled by RTC\_TMPCFG.TPxTRG bit. The RTC\_TAMPx pin will generate a tamper detection event when corresponding edge is detected.

When edge detection is used, in order to ensure that a valid edge occurs after the intrusion event detection is enabled, it is recommended to check the intrusion pin level by software immediately after enabling it; when RTC\_TMPCFG.TPFLT[1:0] = 0 and RTC\_TMPCFG.TPxTRG = 0, if the intrusion is enabled If the intrusion input is high before detection, the intrusion event can be detected by hardware.

#### Filtered level detection of RTC\_TAMPx input

When RTC\_TMPCFG.TPFLT[1:0] bits set to 1/2/3, tamper detection is set to level detection. The value of RTC\_TMPCFG.TPFLT[1:0] determines the number of samples.

The internal pull-up resistance of tamper pin can be precharged before each sampling, and the precharge time is controlled by RTC\_TMPCFG.TPPRCH[1:0] bits. Precharge will be disabled when RTC\_TMPCFG.TPPUDIS set 1.

Using RTC\_TMPCFG.TPFREQ[2:0] to determine the sampling frequency of level detection can optimize the best balance between tamper detection delay and pull-up power consumption.

### 21.3.16 Daylight saving time configuration

Daylight saving time function can be controlled by RTC\_CTRL.SU1H, RTC\_CTRL.AD1H, and RTC\_CTRL.BAKP bits. Calendar will subtract one hour when set RTC\_CTRL.SU1H bit to 1, and add one hour when set RTC\_CTRL.AD1H to 1. RTC\_CTRL.BAKP can be used to record whether this operation was performed.

### 21.3.17 RTC sub-second register shift

When the value of calendar has a sub-second deviation compared to the external precision clock, the shift function can be used to improve the precision of calendar.

Calendar can use RTC\_SCTRL.AD1S and RTC\_SCTRL.SUBF[14:0] bits to control maximum delay or advance 1s. The resolution of the adjustment is  $1/(RTC\_PRE.DIVS[14:0]+1)$  second, it means the higher value of RTC\_PRE.DIVS[14:0], the higher of the resolution. However, to keep the synchronous prescaler output at 1Hz, the higher RTC\_PRE.DIVS[14:0] means the lower RTC\_PRE.DIVA[6:0], then more power consuming.

Note: Before starting a shift operation, user must check RTC\_SUBS.SS[15] bit is 0.

Whenever write RTC\_SCTRL register, the RTC\_INITSTS.SHOPF flag will be set by hardware, which indicate a shift operation is pending. Once this shift operation is complete, the bit is cleared by hardware.

### 21.3.18 RTC digital clock precision calibration

Digital precision calibration is achieved by adjusting the number of RTC clock pulses in the calibration period. Digital precision calibration resolution is 0.954 PPM with the range from -487.1 PPM to +488.5 PPM.

When the input frequency is 32768 Hz, calibration period can be configured as  $2^{20}/2^{19}/2^{18}$  RTCCLK cycles or 32/16/8 seconds. The precision calibration register (RTC\_CALIB) indicates that there has RTC\_CALIB.CM[8:0] RTCCLK clock cycles will be reduced during the specified period.

The value of RTC\_CALIB.CM[8:0] represents the number of RTCCLK pulses to be reduced during specified period. While RTC\_CALIB.CP can be used to increase 488.5 PPM, every  $2^{11}$  RTCCLK cycles will inserts a RTCCLK pulse.

When using RTC\_CALIB.CM[8:0] and RTC\_CALIB.CP in combination, it can increase cycles range from -511 to +512 RTCCLK cycles, and the calibration range from -487.1 ppm to +488.5 ppm, with the resolution is about 0.954 ppm.

The effective calibrated frequency ( $f_{CAL}$ ) can be calculated by using the formula given below:

$$f_{CAL} = f_{RTCCLK} * \left(1 + \frac{RTC\_CALIB.CP * 512 - RTC\_CALIB.CM[8:0]}{2^n + RTC\_CALIB.CM[8:0] - RTC\_CALIB.CP * 512}\right)$$

Note: n=20/19/18

### Calibrated when RTC\_PRE.DIVA[6:0]<3

When the asynchronous prescaler value (RTC\_PRE.DIVA[6:0]) is less than 3, the RTC\_CALIB.CP cannot be programmed to 1, and RTC\_CALIB.CP value will be ignored if it has been set to 1.

When RTC\_PRE.DIVA[6:0]<3, the value of RTC\_PRE.DIVS[14:0] should be decrease. Assume RTCCLK frequency is 32768Hz:

- When RTC\_PRE.DIVA[6:0] = 2, RTC\_PRE.DIVS[14:0] = 8189.
- When RTC\_PRE.DIVA[6:0] = 1, RTC\_PRE.DIVS[14:0] = 16379.
- When RTC\_PRE.DIVA[6:0] = 0, RTC\_PRE.DIVS[14:0] = 32759.

The effective calibrated frequency ( $f_{CAL}$ ) can be calculated by using the formula given below:

$$f_{CAL} = f_{RTCCLK} * \left( 1 + \frac{256 - RTC\_CALIB.CM[8:0]}{2^n + RTC\_CALIB.CM[8:0] - 265} \right)$$

Note: n=20/19/18

### Verify RTC calibration

RTC output 1Hz waveform for measuring and verifying RTC precision.

Up to 2 RTCCLK cycles measurement error may occur when measure the RTC frequency in a limit measurement period. If the measurement period is the same as calibration period, the error can be eliminated.

- The calibration period is 32 seconds (default).

Using an accurate 32-second period to measure the 1Hz calibration output can ensure that the measurement error is within 0.447ppm (0.5 RTCCLK cycles within 32 seconds).

- The calibration period is 16 seconds.

Using an accurate 16-second period to measure the 1Hz calibration output can ensure that the measurement error is within 0.954ppm (0.5 RTCCLK cycles within 16 seconds).

- The calibration period is 8 seconds.

Using an accurate 8-second period to measure the 1Hz calibration output can ensure that the measurement error is within 1.907ppm (0.5 RTCCLK cycles within 8 seconds).

### Dynamic recalibration

When RTC\_INITSTS.INITF=0, RTC\_CALIB register can update by using following steps:

- Wait RTC\_INITSTS.RECPF=0.
- A new value is written to the RTC\_CALIB, then RTC\_INITSTS.RECPF is automatically set to 1.
- The new calibration settings will take effect within 3 ck\_apre cycles after a data write to the RTC\_CALIB.

### 21.3.19 RTC low power mode

The working state of RTC in low power mode.

Lower Power Mode		RTC Working State										Exit Low Power Mode					
SLEEP		Normal work										RTC interrupt					
STOP		Normal work when the clock source of RTC is LSE or LSI										Alarm A, Alarm B, Periodic Wakeup, Tamper event and Timestamp event					

## 21.4 RTC Registers

### 21.4.1 RTC register overview

Table 21-2 RTC register overview

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0									
000h	RTC_TSH	Reserved										APM	22	HOU[3:0]			MIT[2:0]			MIU[3:0]			SCU[3:0]			SCT[2:0]			DAU[3:0]													
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									
004h	RTC_DATE	Reserved					YRT[3:0]			YRU[3:0]			WDU[2:0]			MOT			MOU[3:0]			DAT[1:0]			DAU[3:0]			0			0											
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									
008h	RTC_CTRL	Reserved					COEN	OUTSEL[1:0]			OPOL	CALOSEL			BAKP			SUIH			AD1H			TSEN			ALAFN			ALBEN			ALAEF			0						
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									
00Ch	RTC_INITSTS	Reserved										RECPF	TAM2F			TAM1F			TISOVF			TISF			TSEN			WTEN			ALBF			ALBEN			ALAF			0		
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									
010h	RTC_PRE	Reserved					DIVA[6:0]					DIVS[14:0]										0										0										
	Reset Value	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									
014h	RTC_WKUPT	Reserved										WKUPT[15:0]										1										1										
	Reset Value	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									
01Ch	RTC_ALARMA	MASK4	WKDSEL	DTT[1:0]	DTU[3:0]			MASK3	APM	DTU[3:0]	HOT[1:0]	HOU[3:0]			MIT[2:0]			MIU[3:0]			SET[2:0]			SEU[3:0]			INITM			HFMT			0									
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									
020h	RTC_ALARM_B	MASK4	WKDSEL	DTT[1:0]	DTU[3:0]			MASK3	APM	HOT[1:0]	HOU[1:0]	HOU[3:0]			MIT[2:0]			MIU[3:0]			SET[2:0]			SEU[3:0]			MASK1			RSYF			0									
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									
024h	RTC_WRP	Reserved										PKEY[7:0]										0										0										
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									
028h	RTC_SUBS	Reserved										SS[15:0]										0										0										
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									
02Ch	RTC_SCTRL	ADIS	Reserved										SUBF[14:0]										0										0									
	Reset Value	0	0										0										0										0									
030h	RTC_TST	Reserved					APM	HOT[1:0]			HOU[3:0]			MIT[2:0]			MIU[3:0]			SET[2:0]			SEU[3:0]			Reserved			0			0										
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
034h	RTC_TSD	Reserved					YRT[3:0]			YRU[3:0]			WDU[2:0]			MOU[3:0]			Reserved	DAT[1:0]			DAU[3:0]			0	0	0	0	0	0					
	Reset Value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
038h	RTC_TSSS	Reserved										SSE[15:0]																								
	Reset Value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
03Ch	RTC_CALIB	Reserved										CP			CW8			CW16			Reserved			CM[8:0]												
	Reset Value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
040h	RTC_TMPCFG	Reserved					TP2MF	TP2NOE	TP2INTEN	TP1MF	TP1NOE	TP1INTEN	TPUDIS			TPPRCH[1:0]			TPFLTU[1:0]			TPFREQ[2:0]			TPTS			Reserved			TP2TRG		TP1TRG		TPIEN	
	Reset Value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
044h	RTC_ALRMASS	Reserved	MASKSSA[3:0]			Reserved					SSV[14:0]																									
	Reset Value		0	0	0						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
048h	RTC_ALRMBSS	Reserved	MASKSSB[3:0]			Reserved					SSV[14:0]																									
	Reset Value		0	0	0						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

## 21.4.2 RTC Calendar Time Register (RTC\_TSH)

Address offset: 0x00

Reset value: 0x0000 0000

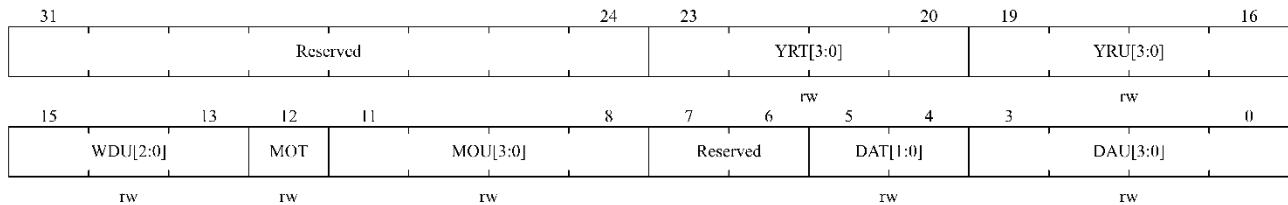
31	Reserved										23	22	21	20	19	18	17	16										
15	14	12	11	8	7	rw	6	rw	4	3	rw																	
Reserved	MIT[2:0]			MIU[3:0]			Reserved	SCT[2:0]			SCU[3:0]																	

Bit field	Name	Description
31:23	Reserved	Reserved, the reset value must be maintained.
22	APM	AM/PM format. 0: AM format or 24-hour format 1: PM format
21:20	HOT[1:0]	Describes the hour tens value in BCD format
19:16	HOU[3:0]	Describes the hour units value in BCD format
15	Reserved	Reserved, the reset value must be maintained.
14:12	MIT [2: 0]	Describes the minute tens value in BCD format
11:8	MIU[3:0]	Describes the minute units value in BCD format
7	Reserved	Reserved, the reset value must be maintained.
6:4	SCT[2:0]	Describes the second tens value in BCD format
3:0	SCU[3:0]	Describes the second units value in BCD format

### 21.4.3 RTC Calendar Date Register (RTC\_DATE)

Address offset: 0x04

Reset value: 0x0000 2101

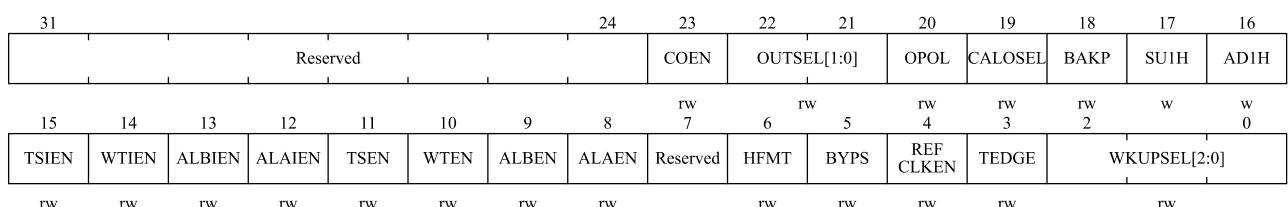


Bit field	Name	Description
31:24	Reserved	Reserved, the reset value must be maintained.
23:20	YRT[3:0]	Describes the year tens value in BCD format
19:16	YRU[3:0]	Describes the year units value in BCD format
15:13	WDU[2:0]	Describes which Week day 000: Forbidden 001: Monday ... 111: Sunday
12	MOT	Describes the month tens value in BCD format
11:8	MOU[3:0]	Describes the month units value in BCD format
7:6	Reserved	Reserved, the reset value must be maintained.
5:4	DAT[1:0]	Describes the date tens value in BCD format
3:0	DAU[3:0]	Describes the date units value in BCD format

### 21.4.4 RTC Control Register (RTC\_CTRL)

Address offset: 0x08

Reset value: 0x0000 0000



Bit field	Name	Description
31:24	Reserved	Reserved, the reset value must be maintained.
23	COEN	Calibration output enable This bit controls RTC_CALIB output

Bit field	Name	Description
		0: Disable calibration output 1: Enable calibration output
22:21	OUTSEL[1:0]	Output selection These bits are used to select the alarm/wakeup output 00: Disable output 01: Enable Alarm A output 10: Enable Alarm B output 11: Enable Wakeup output
20	OPOL	Output polarity bit This bit is used to configure the polarity of output. 0: Outputs high level when the selected output triggers(see OUTSEL[1:0]) 1: Outputs low level when the selected output triggers(see OUTSEL[1:0])
19	CALOSEL	Calibration output selection When RTC_CTRL.COEN=1, RTCCLK = 32.768KHz and prescale at their default value (RTC_PRE.DIVA[6:0]=127 and RTC_PRE.DIVS[14:0]=255). 0: Calibration output is 256 Hz 1: Calibration output is 1 Hz
18	BAKP	Daylight saving time record This bit is written by the user 0: Not record daylight saving time 1: Record daylight saving time
17	SU1H	Subtract 1 hour (winter time change) 1 hour will be subtracted to the calendar time when the current hour value is not 0. This bit is always read as 0. 0: No effect. 1: Subtracts 1 hour to the current time.
16	AD1H	Add 1 hour (summer time change) When this bit is set, 1 hour can be added to the calendar time. This bit is always read as. 0: No effect. 1: Adds 1 hour to the current time.
15	TSIEN	Time-stamp interrupt enable 0: Disable time-stamp interrupt. 1: Enable time-stamp interrupt.
14	WTIEN	Wakeup timer interrupt enable 0: Disable wakeup timer interrupt. 1: Enable wakeup timer interrupt.
13	ALBIEN	Alarm B interrupt enable 0: Disable Alarm B interrupt 1: Enable Alarm B Interrupt
12	ALAIEN	Alarm A interrupt enable 0: Disable Alarm A interrupt

Bit field	Name	Description
		1: Enable Alarm A interrupt
11	TSEN	<p>Timestamp enable 0: Disable timestamp 1: Enable timestamp</p>
10	WTEN	<p>Wakeup timer enable 0: Disable wakeup timer 1: Enable wakeup timer</p>
9	ALBEN	<p>Alarm B enable 0: Disable Alarm B 1: Enable Alarm B</p>
8	ALAEN	<p>Alarm A enable 0: Disable Alarm A 1: Enable Alarm A</p>
7	Reserved	Reserved, the reset value must be maintained.
6	HFMT	<p>Hour format bit 0: 24 hour format 1: Am/PM format</p>
5	BYPS	<p>Bypass values from the shadow registers 0: Calendar values are copied from the shadow registers, which are refreshed every two RTCCLK cycles. 1: Calendar values are copied directly from the calendar counters. <i>Note: If the frequency of the APB1 clock falls below seven times the frequency of RTCCLK, RTC_CTRL.BYPS bit must be set to '1'</i></p>
4	REFCLKEN	<p>RTC_REFIN reference clock detection enable (50 or 60 Hz) 0: Disable RTC_REFIN detection 1: Enable RTC_REFIN detection <i>Note: RTC_PRE.DIVS must be 0x00FF</i></p>
3	TEDGE	<p>Time-stamp event active edge 0: Input rising edge creates a timestamp event 1: Input falling edge creates a timestamp event TSEN need to be reset when TEDGE is changed to avoid unwanted RTC_INITSTS.TISF setting.</p>
2:0	WKUPSEL[2:0]	<p>Wakeup clock selection 000: RTC clock is divided by 16 001: RTC clock is divided by 8 010: RTC clock is divided by 4 011: RTC clock is divided by 2 10x: ck_spre (usually 1Hz) clock is selected 11x: ck_spre (usually 1Hz) clock is selected and <math>2^{16}</math> is added to the RTC_WKUPT.WKUPT counter.</p>

#### 21.4.5 RTC Initial Status Register (RTC\_INITSTS)

Address offset: 0x0C

Reset value: 0x0000 0007

Reserved															RECPF
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	r 0
Reserved	TAM2F	TAM1F	TISOVF	TISF	WTF	ALBF	ALAF	INITM	INITF	RSYF	INITSF	SHOPF	WTWF	ALBWF	ALAWF
rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rw	r	rc_w0	r	r	r	r	r	r

Bit field	Name	Description
31:17	Reserved	Reserved, the reset value must be maintained.
16	RECPF	<p>Recalibration pending flag</p> <p>The RECPF status flag is automatically set to ‘1’ when software writes to the RTC_CALIB register, indicating that the RTC_CALIB register is blocked. After the new calibration settings are processed, this bit returns to ‘0’.</p>
15	Reserved	Reserved, the reset value must be maintained.
14	TAM2F	<p>RTC_TAMP2 detection flag</p> <p>This flag is set to ‘1’ by hardware when a tamper event is detected on the RTC_TAMP2 input pin.</p> <p>This flag can be cleared by software writing 0</p>
13	TAM1F	<p>RTC_TAMP1 detection flag</p> <p>This flag is set to ‘1’ by hardware when a tamper event is detected on the RTC_TAMP1 input pin.</p> <p>This flag can be cleared by software writing 0</p>
12	TISOVF	<p>The time-stamp overflow flag</p> <p>This flag is set to ‘1’ by hardware when a time-stamp event happens when TISF bit is set.</p> <p>This flag can be cleared by software writing 0. It is advised to check and clear TISOVF only after clearing the TISF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TISF bit is being cleared.</p>
11	TISF	<p>Time-stamp flag</p> <p>This flag is set to ‘1’ by hardware when a time-stamp event happens.</p> <p>This flag can be cleared by software writing 0</p>
10	WTF	<p>Wake up timer flag</p> <p>This flag is set by hardware when the value of wakeup auto-reload counter reaches 0.</p> <p>This flag is cleared by software by writing 0.</p> <p>This flag must be cleared by software at least 1.5 RTCCLK periods before WTF is set again.</p>
9	ALBF	Alarm B flag

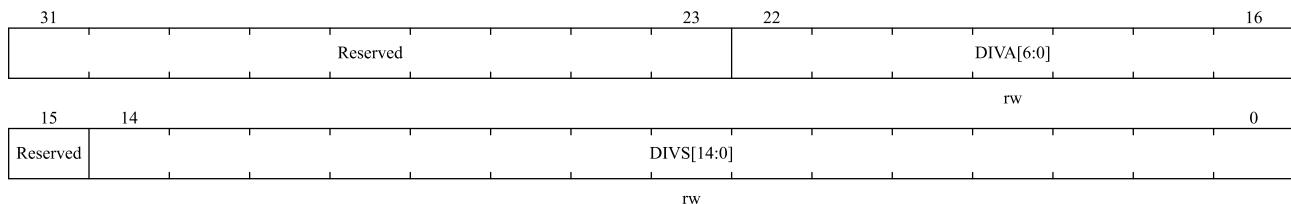
Bit field	Name	Description
		<p>This flag is set to ‘1’ by hardware when the time/date registers value match the Alarm B register values.</p> <p>This flag can be cleared by software writing 0</p>
8	ALAF	<p>Alarm A flag</p> <p>This flag is set to ‘1’ by hardware when the time/date registers value match the Alarm A register values.</p> <p>This flag can be cleared by software writing 0</p>
7	INITM	<p>Enter initialization mode</p> <p>0: Free running mode</p> <p>1: Enter initialization mode and set calendar time value, date value, and prescale value.</p>
6	INITF	<p>Initialization flag</p> <p>RTC is in initialization state when this bit is ‘1’, and calendar time, date and prescale value can be updated.</p> <p>0: Calendar time, date and prescale value can not be updated</p> <p>1: Calendar time, date and prescale value can be updated</p>
5	RSYF	<p>Register synchronization flag</p> <p>This flag is set to ‘1’ by hardware when the calendar value are copied into the shadow registers. This bit is cleared by hardware when in initialization mode, while a shift operation is pending (SHOPF=1), or when in bypass shadow register mode (RTC_CTRL.BYPS=1). This bit can also be cleared by software.</p> <p>It is cleared either by software or by hardware in initialization mode.</p> <p>0: Calendar shadow register not yet synchronized</p> <p>1: Calendar shadow register synchronized</p>
4	INITSF	<p>Initialization status flag</p> <p>This flag is set to ‘1’ by hardware when the calendar year field is different from 0 (which is the RTC domain reset state).</p> <p>0: Calendar has not been initialized</p> <p>1: Calendar has been initialized</p>
3	SHOPF	<p>Shift operation pending flag</p> <p>This flag is set to ‘1’ by hardware as soon as a shift operation is initiated by a write to the RTC_SCTRL register. It is cleared by hardware when the corresponding shift operation has been completed, note that writing to the SHOPF bit has no effect.</p> <p>0: No shift operation is pending</p> <p>1: A shift operation is pending</p>
2	WTWF	<p>Wakeup timer write flag</p> <p>0: Wakeup timer configuration update is not allowed</p> <p>1: Wakeup timer configuration update is allowed</p>
1	ALBWF	<p>Alarm B write flag</p> <p>This flag is set to ‘1’ by hardware when Alarm B values can be changed, after the RTC_CTRL.ALBEN bit has been set to 0.</p>

Bit field	Name	Description
		0: Alarm B update is not allowed 1: Alarm B update is allowed
0	ALAWF	Alarm A write flag. This flag is set to '1' by hardware when Alarm A values can be changed, after the RTC_CTRL.ALAEN bit has been set to 0. 0: Alarm A update is not allowed 1: Alarm A update is allowed

#### 21.4.6 RTC Prescaler Register (RTC\_PRE)

Address offset: 0x10

Reset value: 0x007F 00FF

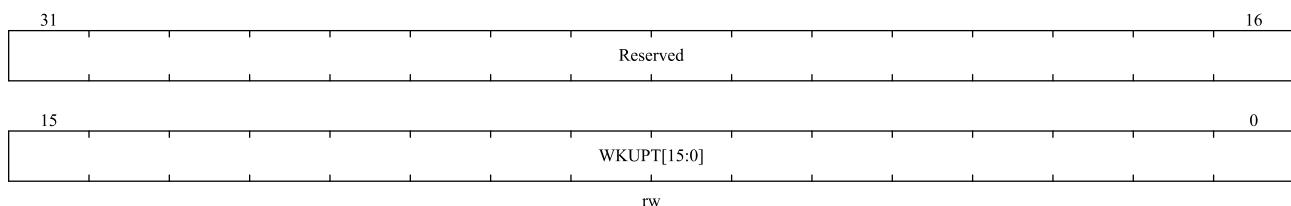


Bit field	Name	Description
31:23	Reserved	Reserved, the reset value must be maintained.
22:16	DIVA[6:0]	Asynchronous prescaler factor $f_{ck\_apre} = \text{RTCCLK}/(\text{DIVA}[6:0]+1)$
15	Reserved	Reserved, the reset value must be maintained.
14:0	DIVS[14:0]	Synchronous prescaler factor $f_{ck\_spre} = f_{ck\_apre}/(\text{DIVS}[14:0]+1)$

#### 21.4.7 RTC Wakeup Timer Register (RTC\_WKUPT)

Address offset: 0x14

Reset value: 0x0000 FFFF



Bit field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained.
15:0	WKUPT[15:0]	Wake up auto-reload value bits

Bit field	Name	Description
		<p>The RTC_INITSTS.WTF flag is set every (WKUPT[15:0] + 1) ck_wut cycles when the RTC_CTRL.WTEN=1. The wakeup timer becomes 17-bits When RTC_CTRL.WKUPSEL[2]=1.</p> <p><i>Note:</i></p> <p><i>This register change (such as the second setting or later Settings) needs to be changed in the wakeup interrupt, otherwise the changed Settings will not take effect immediately, but will take effect after the next wakeup;</i></p> <p><i>In particular, when RTC_CTRL.WKUPSEL[2:0] is set to 010, the modified setting does not take effect immediately, but will take effect after wake up in the next cycle.</i></p>

#### 21.4.8 RTC Alarm A Register (RTC\_ALARMA)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	24	23	22	21	20	19	16
MASK4	WKDSEL	DTT[1:0]		DTU[3:0]	MASK3	APM	HOT[1:0]		HOU[3:0]		
rw 15	rw 14	rw 12	rw 11	rw 8	rw 7	rw 6	rw 4	rw 3	rw 0		
MASK2	MIT[2:0]		MIU[3:0]	MASK1	SET[2:0]			SEU[3:0]			
rw	rw		rw	rw	rw		rw		rw		

Bit field	Name	Description
31	MASK4	Alarm date mask 0: Date/day match 1: Date/day not match
30	WKDSEL	Week day selection 0: DTU[3:0] represents the date units 1: DTU[3:0] represents week day only. DTT[1:0] is not considered
29:28	DTT[1:0]	Describes the date tens value in BCD format
27:24	DTU[3:0]	Describes the date units value in BCD format
23	MASK3	Alarm hours mask 0: Hours match 1: Hours not match
22	APM	AM/PM notation 0: AM or 24 hours format 1: PM format
21:20	HOT[1:0]	Describes the hour tens value in BCD format
19:16	HOU[3:0]	Describes the hour units value in BCD format
15	MASK2	Alarm minutes mask 0: Minutes match 1: Minutes not match

Bit field	Name	Description
14:12	MIT[2:0]	Describes the minute tens value in BCD format
11:8	MIU[3:0]	Describes the minute units value in BCD format
7	MASK1	Alarm seconds mask 0: Seconds match 1: Seconds not match
6:4	SET[2:0]	Describes the second tens value in BCD format
3:0	SEU[3:0]	Describes the second units value in BCD format

### 21.4.9 RTC Alarm B Register (RTC\_ALARMB)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	24	23	22	21	20	19	16
MASK4	WKDSEL	DTT[1:0]		DTU[3:0]	MASK3	APM	HOT[1:0]		HOU[3:0]		
rw 15	rw 14	rw 12	rw 11	rw 8	rw 7	rw 6	rw 4	rw 3	rw 0		
MASK2	MIT[2:0]		MIU[3:0]	MASK1	SET[2:0]		SEU[3:0]				
rw	rw		rw	rw	rw	rw	rw	rw	rw		

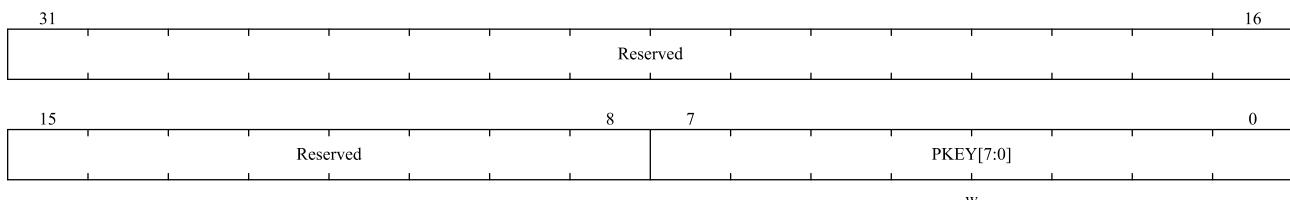
Bit field	Name	Description
31	MASK4	Alarm date mask 0: Date/day match 1: Date/day not match
30	WKDSEL	Week day selection 0: DTU[3:0] represents the date units 1: DTU[3:0] represents week day only. DTT[1:0] is not considered
29:28	DTT[1:0]	Describes the date tens value in BCD format
27:24	DTU[3:0]	Describes the date units value in BCD format
23	MASK3	Alarm hours mask 0: Hours match 1: Hours not match
22	APM	AM/PM notation 0: AM or 24 hours format 1: PM format
21:20	HOT[1:0]	Describes the hour tens value in BCD format
19:16	HOU[3:0]	Describes the hour units value in BCD format
15	MASK2	Alarm minutes mask 0: Minutes match 1: Minutes not match
14:12	MIT[2:0]	Describes the minute tens value in BCD format
11:8	MIU[3:0]	Describes the minute units value in BCD format

Bit field	Name	Description
7	MASK1	Alarm seconds mask 0: Seconds match 1: Seconds not match
6:4	SET[2:0]	Describes the second tens value in BCD format
3:0	SEU[3:0]	Describes the second units value in BCD format

#### 21.4.10 RTC Write Protection register (RTC\_WRP)

Address offset: 0x24

Reset value: 0x0000 0000

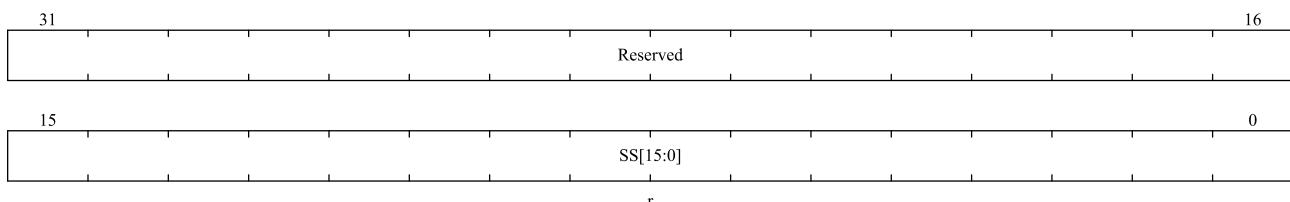


Bit field	Name	Description
31:8	Reserved	Reserved, the reset value must be maintained.
7:0	PKEY[7:0]	Write protection key Reading this byte always returns 0x00. For detail on how to unlock RTC register write protection, see chapter RTC register write protection.

#### 21.4.11 RTC Sub-second Register (RTC\_SUBS)

Address offset: 0x28

Reset value: 0x0000 0000



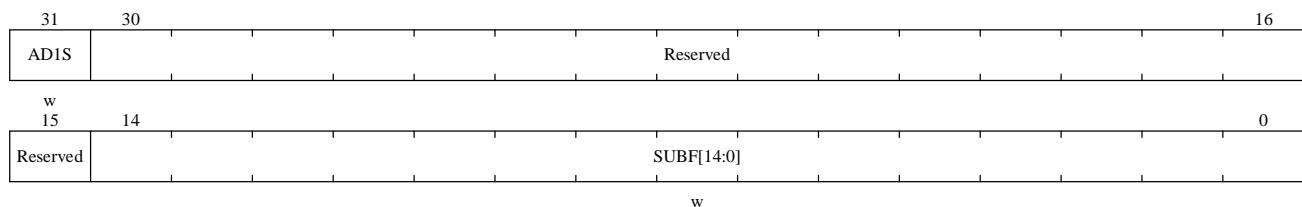
Bit field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained.
15:0	SS[15:0]	Sub-second bit. This value is the value of the synchronous prescaler counter. This sub-second value is calculated by the below formula: Sub-second value = (RTC_PRE.DIVS[14:0]-SS)/( RTC_PRE.DIVS[14:0]+1)

Bit field	Name	Description
		<i>Note: SS[15:0] can be larger than RTC_PRE.DIVS[14:0] only after the shift operation is finished. In this case, the correct time/date is one second slower than the time/date indicated by RTC_TSH/RTC_DATE.</i>

### 21.4.12 RTC Shift Control Register (RTC\_SCTRL)

Address offset: 0x2C

Reset value: 0x0000 0000



Bit field	Name	Description
31	AD1S	Add one second 0: No add one second. 1: Add one second to the clock/calendar This bit can only be written and read as zero. Writing to this bit does not have an impact when RTC_INITSTS.SHOPF=1.
30:15	Reserved	Reserved, the reset value must be maintained.
14:0	SUBF[14:0]	Subtract a fraction of a second These bits can only be written and read as zero.. Writing to this bit does not have an impact when RTC_INITSTS.SHOPF=1. The value which is written to SUBF is added to the synchronous prescaler counter, and the clock will delay: $\text{Delay (seconds)} = (\text{SUBF}[14:0]+1) / (\text{DIVS}[14:0] + 1)$ AD1S bit can be used together with the SUBF[14:0] bits: $\text{Advance (seconds)} = (1 - ((\text{SUBF}[14:0]+1)/(\text{DIVS}[14:0] + 1)))$ <i>Note: RTC_INITSTS.RSYF bit will be cleared when write SUBF[14:0]. When RTC_INITSTS.RSYF=1, the shadow registers have been updated with the shifted time. When AD1S is set to 1, SUBF[14:0] cannot be all 0</i>

### 21.4.13 RTC Timestamp Time Register (RTC\_TST)

Address offset: 0x30

Reset value: 0x0000 0000

31	Reserved								23	22	21	20	19	16
									APM	HOT[1:0]			HOU[3:0]	
15	14	12	11	8	7	r	6	r	4	3	r	0		
Reserved	MIT[2:0]		MIU[3:0]		Reserved		SET[2:0]			SEU[3:0]				
r		r			r		r		r		r			

Bit field	Name	Description
31:23	Reserved	Reserved, the reset value must be maintained.
22	APM	AM/PM notation 0: AM or 24-hour clock 1: PM
21:20	HOT[1:0]	Describes the hour tens value in BCD format
19:16	HOU[3:0]	Describes the hour units value in BCD format
15	Reserved	Reserved. Must be kept at the reset value
14:12	MIT[2:0]	Describes the minute tens value in BCD format
11:8	MIU[3:0]	Describes the minute units value in BCD format
7	Reserved	Reserved, the reset value must be maintained.
6:4	SET[2:0]	Describes the second tens value in BCD format
3:0	SEU[3:0]	Describes the second units value in BCD format

#### 21.4.14 RTC Timestamp Date Register (RTC\_TSD)

Address offset: 0x34

Reset value: 0x0000 0000

31	Reserved								24	23	YRT[3:0]			20	19	16
											YRU[3:0]					
15	13	12	11	8	7	r	6	r	5	4	3	r	0			
WDU[2:0]	MOT		MOU[3:0]		Reserved		DAT[1:0]			DAU[3:0]						
r		r		r			r		r		r					

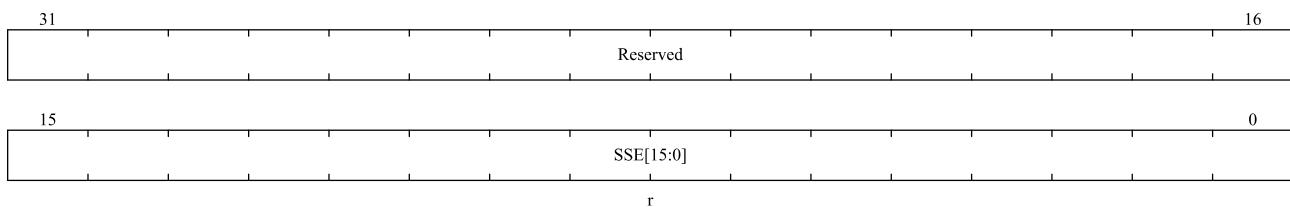
Bit field	Name	Description
31:24	Reserved	Reserved, the reset value must be maintained.
23:20	YRT[3:0]	Describes the year tens value in BCD format
19:16	YRU[3:0]	Describes the year units value in BCD format
15:13	WDU[2:0]	Describes which Week day 000: Forbidden 001: Monday ... 111: Sunday
12	MOT	Describes the month tens value in BCD format

Bit field	Name	Description
11:8	MOU[3:0]	Describes the month units value in BCD format
7:6	Reserved	Reserved, the reset value must be maintained.
5:4	DAT[1:0]	Describes the date tens value in BCD format
3:0	DAU[3:0]	Describes the date units value in BCD format

### 21.4.15 RTC Timestamp Sub-second Register (RTC\_TSSS)

Address offset: 0x38

Reset value: 0x0000 0000

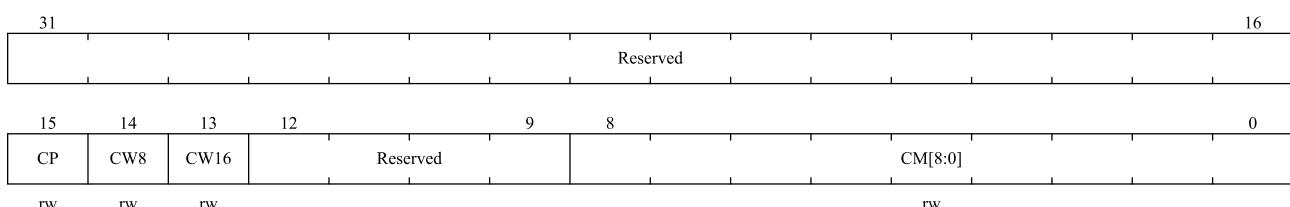


Bit field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained.
15:0	SSE[15:0]	<p>Sub second value</p> <p>SSE[15:0] is the value in the synchronous prescaler counter. The fraction of a second is provided by the formula below:</p> $\text{Second fraction} = (\text{RTC\_PRE.DIVS}[14:0] - \text{SSE}) / (\text{RTC\_PRE.DIVS}[14:0] + 1)$ <p>Note: SSE can be larger than DIVS only after a shift operation. In that case, the correct time/date is one second less than as indicated by RTC_TSH/RTC_DATE.</p>

### 21.4.16 RTC Calibration Register (RTC\_CALIB)

Address offset: 0x3C

Reset value: 0x0000 0000



Bit field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained.
15	CP	<p>Increase frequency of RTC by 488.5 ppm</p> <p>This feature is intended to be used along with CM[8:0]. When RTCCLK frequency is 32768 Hz, the number of RTCCLK pulses added during a 32-second window is ((512</p>

Bit field	Name	Description
		* CP) – CM[8:0]). 0: No add pulse. 1: One RTCCLK pulse is inserted every $2^{11}$ pulses.
14	CW8	Select an 8-second calibration cycle period 0: Not effect. 1: Select an 8-second calibration period. When CW8 is set to '1', the 8-second calibration cycle period is selected. <i>Note: when CW8 = 1, CM[1:0] will always be '00'</i>
13	CW16	To select a 16-second calibration cycle period 0: Not effect. 1: Select a calibration period of 16 seconds. If CW8 = 1, this bit cannot be set to 1. <i>Note: when CW16 = 1, CM[0] will always be '0'</i>
12:9	Reserved	Reserved, the reset value must be maintained.
8:0	CM[8:0]	Negative calibration bits The number of mask pulse out of $2^{20}$ RTCCLK pulses. This effectively decreases the frequency of the calendar with a resolution of 0.9537 ppm.

### 21.4.17 RTC Tamper Configuration Register (RTC\_TMPCFG)

Address offset: 0x40

Reset value: 0x0000 0000

31	Reserved										22	21	20	19	18	17	16
15	14	13	12	11	10	8	7	6	rw	5	4	rw	3	rw	2	1	rw
TPPUDIS	TPPRCH[1:0]	TPFLT[1:0]		TPFREQ[2:0]	TPTS	Reserved	TP2TRG	TP2EN	TPINTEN	TP1TRG	TP1EN						
rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit field	Name	Description
31:22	Reserved	Reserved, the reset value must be maintained.
21	TP2MF	Tamper 2 mask flag 0: Not mask tamper 2 event. 1: Mask tamper 2 event. <i>Note: The Tamper 2 interrupt must not be enabled when TP2MF is set.</i>
20	Reserved	Reserved, the reset value must be maintained.
19	TP2INTEN	Tamper 2 interrupt enable 0: Disable tamper 2 interrupt when TPINTEN = 0. 1: Enabled tamper 2 interrupt
18	TP1MF	Tamper 1 mask flag 0: Not mask tamper 1 event. 1: Mask tamper 1 event.

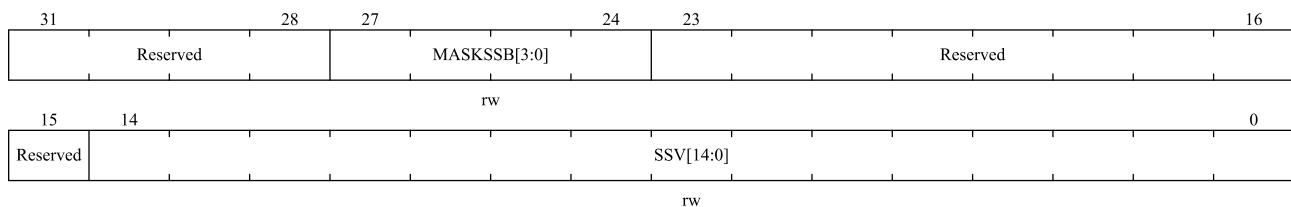
Bit field	Name	Description
<i>Note: The Tamper 1 interrupt must not be enabled when TP1MF is set.</i>		
17	Reserved	Reserved, the reset value must be maintained.
16	TP1INTEN	Tamper 1 interrupt enable 0: Disable tamper 1 interrupt when TPINTEN = 0. 1: Enabled tamper 1 interrupt
15	TPPUDIS	RTC_TAMPx Pull-up disable bit. 0: Enable precharge RTC_TAMPx pins before each sampling. 1: Disable precharge RTC_TAMPx pins
14:13	TPPRCH[1:0]	RTC_TAMPx Precharge duration. These bits determine the the precharge time before each sampling. 0x0: 1 RTCCLK cycles 0x1: 2 RTCCLK cycles 0x2: 4 RTCCLK cycles 0x3: 8 RTCCLK cycles
12:11	TPFLT[1:0]	RTC_TAMPx filter count These bits determine the number of consecutive samples when occur active level. 0x0: Triggers a tamper event at the active level. 0x1: Triggers a tamper event after 2 consecutive samples at the active level. 0x2: Triggers a tamper event after 4 consecutive samples at the active level. 0x3: Triggers a tamper event after 8 consecutive samples at the active level.
10:8	TPFREQ[2:0]	Tamper sampling frequency This bit determines the frequency at the each RTC_TAMPx input is sampled. 0x0: Sampling once every 32768 RTCCLK (1 Hz when RTCCLK = 32.768 KHz). 0x1: Sampling once every 16384 RTCCLK. 0x2: Sampling once every 8192 RTCCLK. 0x3: Sampling once every 4096 RTCCLK. 0x4: Sampling once every 2048 RTCCLK. 0x5: Sampling once every 1024 RTCCLK. 0x6: Sampling once every 512 RTCCLK. 0x7: Sampling once every 256 RTCCLK.
7	TPTS	Tamper event trigger timestamp 0: Disable tamper event trigger timestamp 1: Enable tamper event trigger timestamp TPTS is valid even if TSEN=0 in the RTC_CTRL register.
6:5	Reserved	Reserved, the reset value must be maintained.
4	TP2TRG	Tamper 2 event trigger edge if TPFLT[1:0] != 00, tamper detection is in level mode: 0: low level trigger a tamper detection event. 1: high level trigger a tamper detection event. if TPFLT = 00, tamper detection is in edge mode: 0: Rising edge trigger a tamper detection event.

Bit field	Name	Description
		1: Falling edge trigger a tamper detection event
3	TP2EN	Tamper 2 detection enable 0: Disable tamper detection 1: Enable tamper detection
2	TPINTEN	Tamper event interrupt enable. 0: Disable tamper interrupt 1: Enable tamper interrupt <i>Note: This bit enables the interrupt of all tamper pins events, regardless of TPxINTEN level. If this bit is cleared, each tamper event interrupt can be individually enabled by setting TPxINTEN.</i>
1	TP1TRG	Tamper 1 event trigger edge if TPFLT[1:0] != 00, tamper detection is in level mode: 0: low level trigger a tamper detection event. 1: high level trigger a tamper detection event. if TPFLT = 00, tamper detection is in edge mode: 0: Rising edge trigger a tamper detection event. 1: Falling edge trigger a tamper detection event
0	TP1EN	Tamper 1 detection enable 0: Disable tamper detection 1: Enable tamper detection

#### 21.4.18 RTC Alarm A sub-second register (RTC\_ALRMASS)

Address offset: 0x44

Reset value: 0x0000 0000



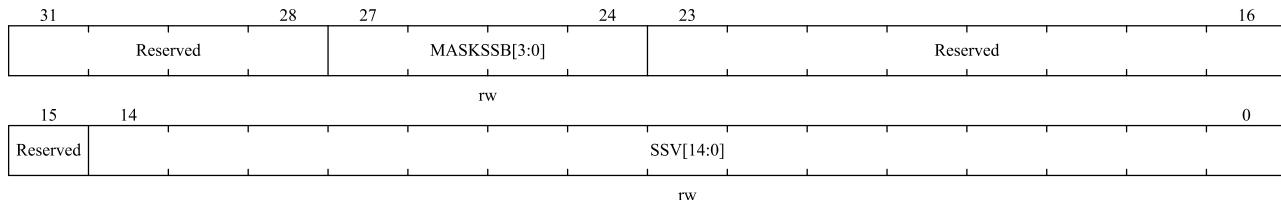
Bit field	Name	Description
31:28	Reserved	Reserved, the reset value must be maintained.
27:24	MASKSSB[3:0]	Mask the most significant bit from this bits. 0x0: No comparison on sub seconds for Alarm. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match). 0x1: Only SSV[0] is compared and other bits are not compared. 0x2: Only SSV[1:0] are compared and other bits are not compared. 0x3: Only SSV[2:0] are compared and other bits are not compared. ...

Bit field	Name	Description
		0xC: Only SSV[11:0] are compared and other bits are not compared. 0xD: Only SSV[12:0] are compared and other bits are not compared. 0xE: Only SSV[13:0] are compared and other bits are not compared. 0xF: SSV[14:0] are compared Synchronization counter RTC_SUBS.SS[15] bit is never compared.
23:15	Reserved	Reserved, the reset value must be maintained.
14:0	SSV[14:0]	Sub seconds value This value is compared with the synchronous prescaler counter RTC_SUBS.SS[14:0], and bit number of compared is controlled by MASKSSB[3:0].

#### 21.4.19 RTC Alarm B sub-second register (RTC\_ALRMBSS)

Address offset: 0x48

Reset value: 0x0000 0000



Bit field	Name	Description
31:28	Reserved	Reserved, the reset value must be maintained.
27:24	MASKSSB[3:0]	Mask the most significant bit from this bits. 0x0: No comparison on sub seconds for Alarm. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match). 0x1: Only SSV[0] is compared and other bits are not compared. 0x2: Only SSV[1:0] are compared and other bits are not compared. 0x3: Only SSV[2:0] are compared and other bits are not compared. ... 0xC: Only SSV[11:0] are compared and other bits are not compared. 0xD: Only SSV[12:0] are compared and other bits are not compared. 0xE: Only SSV[13:0] are compared and other bits are not compared. 0xF: SSV[14:0] are compared. Synchronization counter RTC_SUBS.SS[15] bit is never compared.
23:15	Reserved	Reserved, the reset value must be maintained.
14:0	SSV[14:0]	Sub seconds value This value is compared with the synchronous prescaler counter RTC_SUBS.SS[14:0], and bit number of compared is controlled by MASKSSB[3:0].



## 22 Operational amplifier (OPAMP)

The OPAMP module can be flexibly configured and is suitable for independent op amp PGA and follower mode applications. The input range of OPAMP is 0V to VDDA and the output range is 0.1V to VDDA-0.1V. The output of the OPAMP is internally connected to the input channel of the ADC for analog signal measurement.

### 22.1 OPAMP features

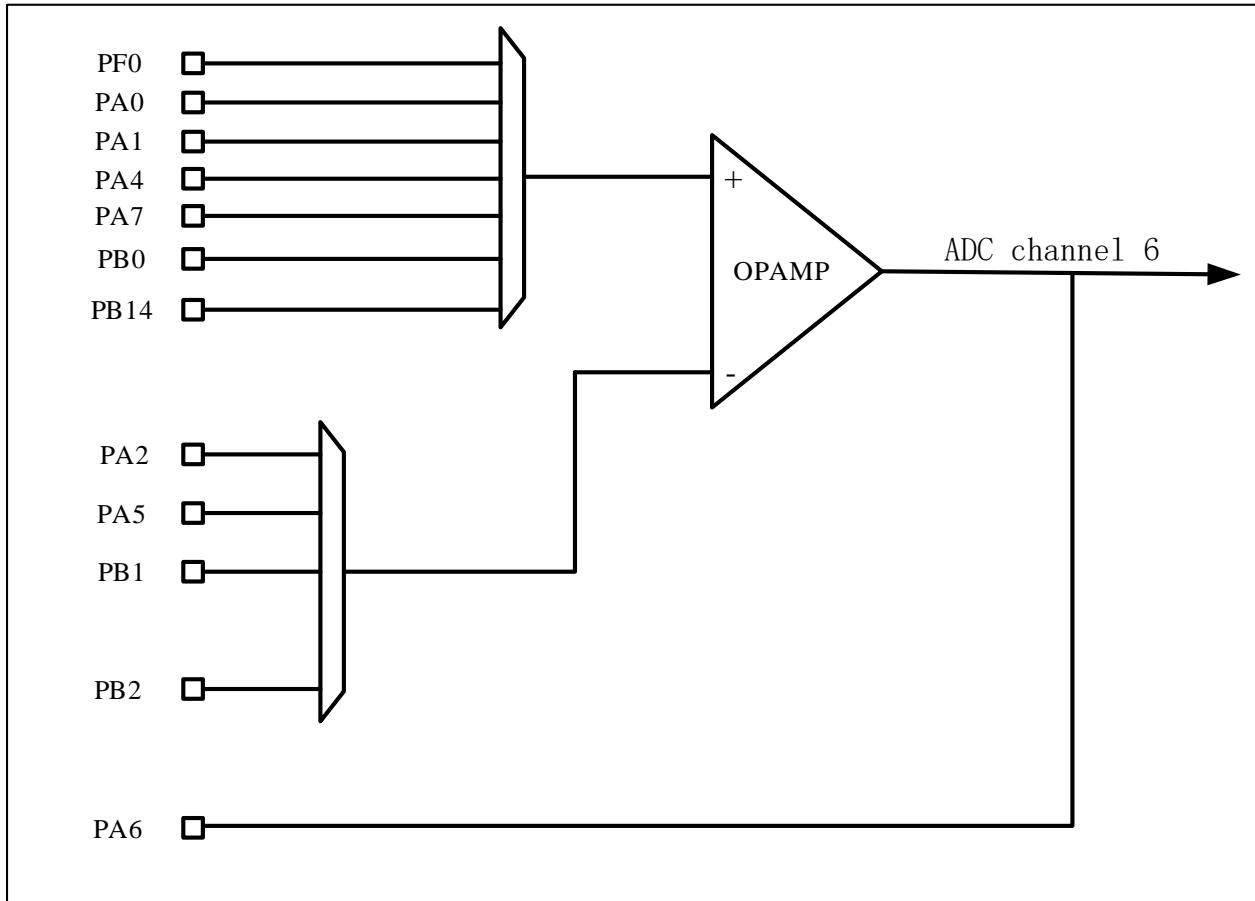
- Support rail-to-rail input, the input range is 0 to VDDA, and the output range is 0.1 to VDDA-0.1 programmable gain
- Can be configured as instrument amplifiers through external resistance connections
- Multiple working modes can be configured:
  - ◆ Independent op amp mode (all ports lead out, magnification can be set through the external resistor)
  - ◆ Voltage follower
  - ◆ Forward/reverse programmable gain amplifier
- Internal PGA GAIN programmable gain settings are 2X, 4X, 8X, 16X, 32X
- Gain-bandwidth product up to 2.5MHz
- Support TIM1\_CC6 automatically switches the input PIN of OPAMP
- Support independent write protection

#### 22.1.1 OPAMP function description

OPAMP can be configured to various different PGA modes through register selection, and can also be configured as the OPAMP function of the user using external components. The output of the OPAMP can be used as the input channel to the ADC. The OPAMP outputs are connected to the analog channel of the ADC as follows.

The output of OPAMP is connected to analog input Channel6 of ADC.

Figure 22-1 Block diagram of OPAMP connection diagram

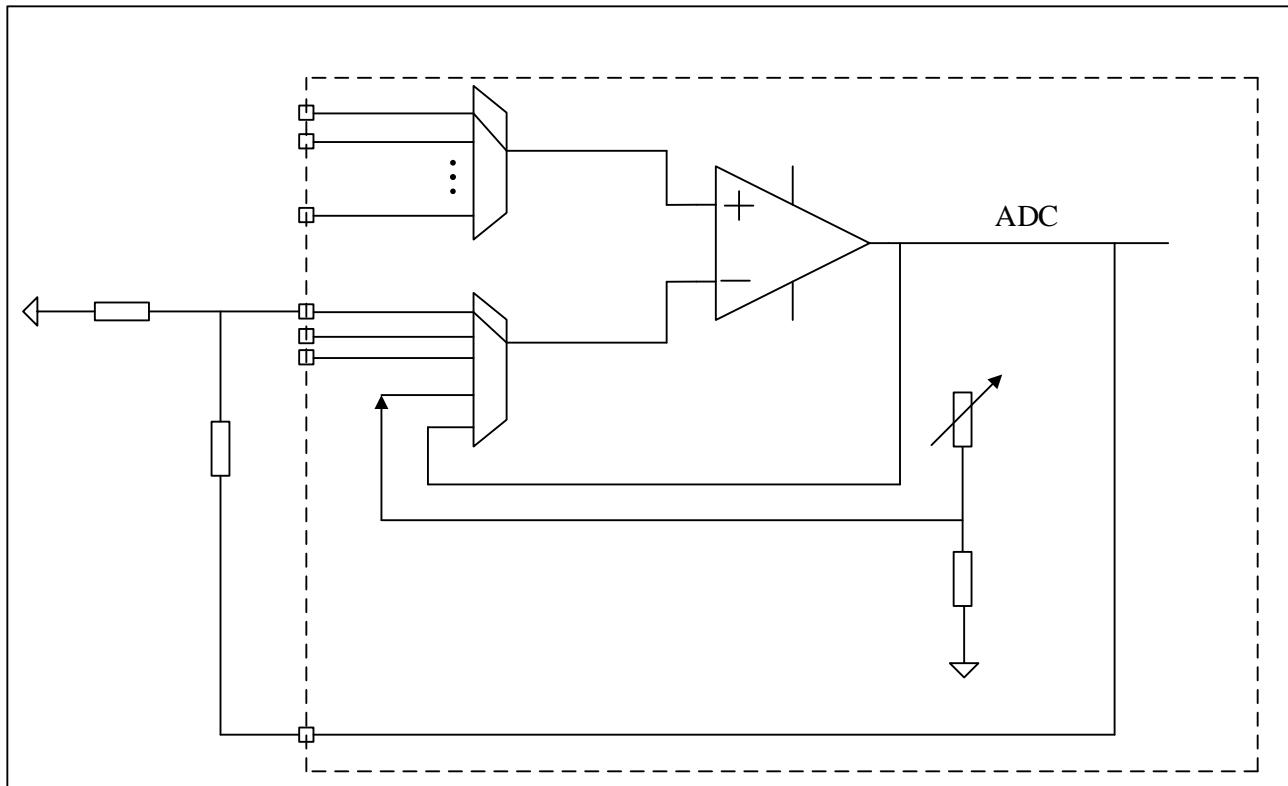


## 22.2 OPAMP working mode

### 22.2.1 OPAMP standalone operational amplifier mode

In external amplification mode, the amplification factor is determined by the connected resistance and capacitance. OPAMAP\_CS.MOD is set to 2'b00 or 2'b01 for op amp function, OPAMAP\_CS.VPSSEL or OPAMAP\_CS.VPSEL selects non-inverted input, and OPAMAP\_CS.VMSSEL or OPAMAP\_CS.VMSEL selects inverted input. An external resistor is used to form a closed-loop amplification system. As shown in the figure below, the non inverted terminal, inverted terminal and output terminal of the OPAMP are all connected to the external port, and the amplification factor is determined by the external resistance-capacitance network.

Figure 22-2 OPAMP Standalone Operational Amplifier Mode



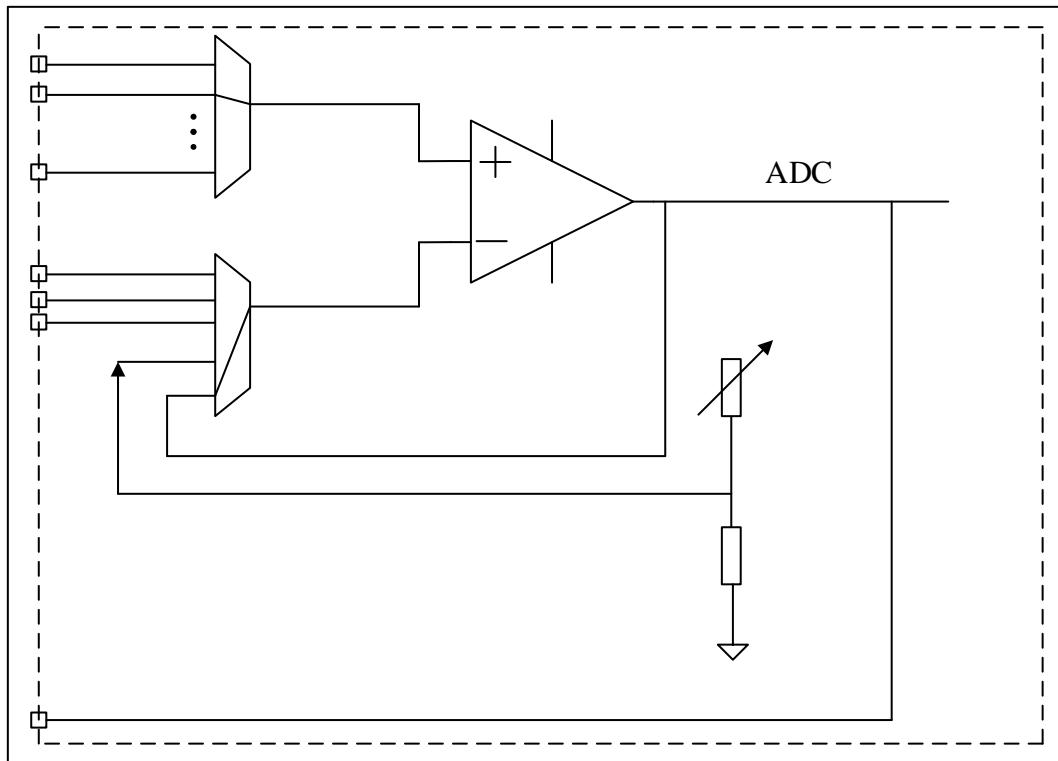
## 22.2.2 OPAMP follow mode

In follow mode, the output voltage is directly following the input voltage. The VMSEL terminal must be configured to connect to the OPAMP output port.

OPAMP\_CS.MOD is set to 2'b11 for internal follow function, OPAMAP\_CS.VPSSEL or OPAMAP\_CS.VPSEL selects non-inverted input, OPAMAP\_CS.VMSSEL or OPAMAP\_CS.VMSEL is connected to the output port from inside the chip.

Unoccupied VM pins can be used as other GPIOs.

Figure 22-3 OPAMP follow mode



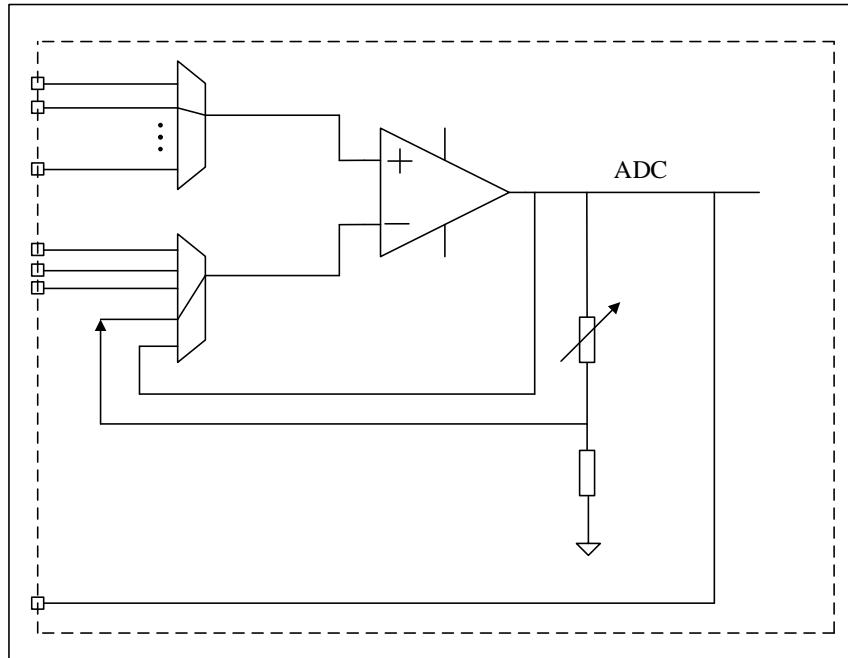
### 22.2.3 OPAMP internal programmable gain (PGA) mode

Internal amplification mode, namely PGA mode, amplifies the input voltage through the built-in resistance feedback network

OPAMP\_CS.MOD is set to 2'b10 for PGA function, supports 2/4/8/16/32 magnification, OPAMP\_CS.VMSSEL or OPAMP\_CS.VMSEL pins must be set to float. OPAMP\_CS.VPSSEL or OPAMP\_CS.VPSEL selects non-inverted input. The non-inverted input can be connected to an external pin, which can be a resistive network. Set OPAMP\_CS.PGAGAN to gain selection. The output of an OPAMP can be a resistive network.

OPAMP's VM input pin can be used as a normal GPIOs.

Figure 22-4 Internal programmable gain mode



#### 22.2.4 OPAMP independent write protection

The write protection of OPAMP can be set independently by configuring the OPAMP\_LOCK register. When the write protection is set, the software will not be able to write to the corresponding OPAMP register, and the write protection function can be canceled only after the chip is reset.

#### 22.2.5 OPAMP TIMER controls the switching mode

In some applications, input switching of OPAMP can be done through TIM1\_CC6. TIM1\_CC6 controls the input switching of the OPAMP.

When TIM1\_CC6 is 1, OPAMP are selected as the ports are configured by VPSSEL or VMSSEL. Otherwise, VPSEL or VMSEL is used.

OPAMP\_CS.TCMEN is set to 1 to enable the automatic switching input function. The process of configuring automatic switching is as follows:

- Enable automatic switching function OPAMP\_CS.TCMEN
- Configure two conversion MUX configuration (VPSEL,VMSEL,VPSSEL,VMSSEL)
- Start OPAMP and TIM

## 22.3 OPAMP registers

### **22.3.1 OPAMP register overview**

Table 22-1 OPAMP register overview

### 22.3.2 OPAMP Control Status Register (OPAMP\_CS)

Address offset: 0x00

Reset value: 0x0000 0000

31	Reserved												20	19	17	16
15	14	13	12	11	9	8	7	6	5	3	rw	2	1	0		
VMSSEL[1:0]	TCMEN	Reserved		VPSEL[2:0]	Reserved	VMSEL[1:0]		PGAGAIN[2:0]	MOD[1:0]	EN						
rw	rw			rw		rw		rw		rw					rw	

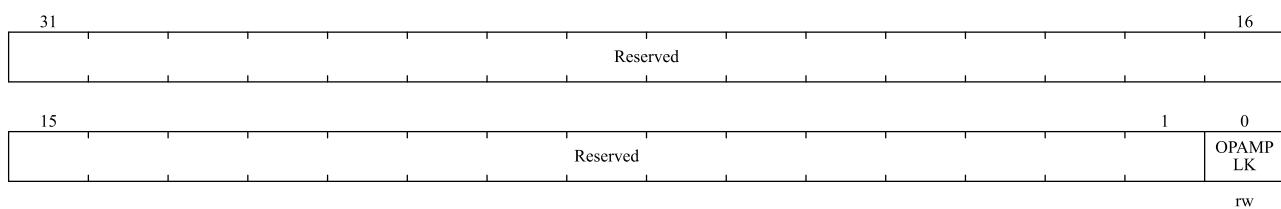
Bit Field	Name	Description		
31:20	Reserve	Reserved□the reset value must be maintained		
19:17	VPSSEL[2:0]	OPAMP Non inverted input secondary selection. Same definition as VPSEL		
16	Reserve	Reserved□the reset value must be maintained		
15:14	VMSSEL[1:0]	OPAMP Inverted input secondary selection Same definition as VMSEL		
13	TCMEN	Timer control automatic switching mode enable  This bit is set or cleared by software and is used to control the automatic switching of primary and secondary inputs (VPSEL,VMSEL and VPSSEL,VMSSEL).  TIM1_CC6 automatically switches to OPAMP.  0: Turn off automatic switching 1: Turn on automatic switching		
12	Reserve	Reserved□the reset value must be maintained		
11:9	VPSEL[2:0]	OPAMP Non Inverted input selection <table border="1" style="margin-left: 20px;"> <tr> <td>Enumerated value</td> <td>OPAMP</td> </tr> </table>	Enumerated value	OPAMP
Enumerated value	OPAMP			

Bit Field	Name	Description	
		000	PF0
		001	PA0
		010	PA1
		011	PA4
		100	PA7
		101	PB0
		110	PB14
8	Reserve	Reserved □ the reset value must be maintained	
7:6	VMSEL[1:0]	OPAMP Inverted input selection	
		Enumerated value	OPAMP
		00	PA2
		01	PA5
		10	PB1
		11	PB2
5:3	PGAGAN[2:0]	OPAMP Programmable amplifier gain value	
		000: Internal PGA gain 2	
		001: Internal PGA gain 4	
		010: Internal PGA gain 8	
		011: Internal PGA gain 16	
		100: Internal PGA gain 32	
		Others: internal PGA gain 2	
2:1	MOD[1:0]	OPAMP PGA mode	
		0x: External zoom mode	
		10: Internal PGA enable	
		11: Internal follow mode	
0	EN	OPAMP Enable	
		0: Disable	
1: Enable			

### 22.3.3 OPAMP Lock Register (OPAMP LOCK)

Address offset: 0x4

Reset value: 0x0000 0000



Bit Field	Name	Description
31:1	Reserve	Reserved□the reset value must be maintained
0	OPAMPLK	<p>OPAMP Lock bit After the reset, this bit can be written only once 0: The OPAMP register is readable and writable 1: The OPAMP register is read-only</p>

## 23 Beeper

### 23.1 Introduction

The Beeper module supports complementary outputs and can generate periodic signals to drive external passive Beepers. Used to generate a prompt tone or an alarm sound.

### 23.2 Function description

Beeper as an independent module, is mounted on the APB1 bus, with a maximum operating frequency of 48MHz. The timbre can be divided into 21 grades, and different timbres can be adjusted by configuring the relevant registers when using. One of the two outputs is usually turned off. If the reverse output is enabled, the two outputs are turned on at the same time and the outputs are complementary. It supports working in LPRUN mode. At this time, the output frequency of the Beeper depends on PCLK1, and is the 4 frequency division value of PCLK1.

### 23.3 Beeper registers

These peripheral registers must be operated in word (32-bit) mode.

#### 23.3.1 Beeper register overview

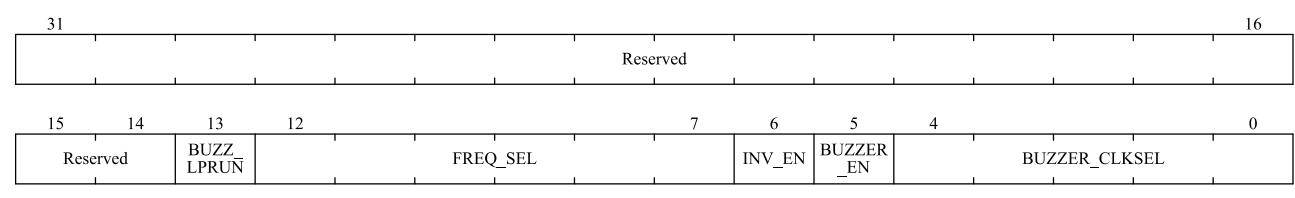
Table 23-1 Beeper register overview

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
000h	BEEPER_CTRL																																	

#### 23.3.2 Beeper control register (BEEPER\_CTRL)

Address offset: 0x00

Reset value: 0x0000 0000



Bit Field	Name	Description
31:14	Reserved	Reserved, the reset value must be maintained.

Bit Field	Name	Description
13	BUZZ_LPRUN	Beeper low power mode selection, if BUZZ_LPRUN and BUZZER_EN are turned on at the same time, Beeper will enter LPRUN mode. At this time, the Beeper output frequency is the 4-divided value of PCLK1. 0: Normal mode. 1: LPRUN mode.
12:7	FREQ_SEL[5:0]	MCU APB frequency selection signal. The corresponding frequency signal is selected according to the APB frequency, so that the output frequency will not be distorted. The value ranges from 1 to 48M, and 0 is also mapped to 48M. 000000: 48M 000001: 1M 000010: 2M 000011: 3M ... 110000: 48M
6	INV_EN	Beeper complementary output enable 0: There is only one output, and the other output is closed. 1: Both outputs are turned on, and the outputs are complementary
5	BUZZER_EN	Beeper enable 0: Beeper disabled 1: Beeper enabled
4:0	BUZZER_CLKSEL[4:0]	Beeper output frequency selection: 00001: L1 (Bass 1) 00010: L2 (Bass 2) 00011: L3 (Bass 3) 00100: L4 (Bass 4) 00101: L5 (Bass 5) 00110: L6 (Bass 6) 00111: L7 (Bass 7)  01000: M1 (Alto 1) 01001: M2 (Alto 2) 01010: M3 (Alto 3) 01011: M4 (Alto 4) 01100: M5 (Alto 5) 01101: M6 (Alto 6) 01110: M7 (Alto 7)  01111: H1 (Treble 1) 10000: H2 (Treble 2) 10001: H3 (Treble 3) 10010: H4 (Treble 4)

Bit Field	Name	Description
		10011: H5 (Treble 5) 10100: H6 (Treble 6) 10101: H7 (Treble 7)  default: ...

## 24 Arithmetic units (HDIV and SQRT)

### 24.1 Introduction to HDIV and SQRT

The divider (HDIV) and square root calculator (SQRT) are mainly used in some scenarios with high requirements for computing energy efficiency, and are used to partially supplement the lack of calculation of the microcontroller. The divider and square root calculator can perform division or square root calculator of unsigned 32-bit integers.

### 24.2 HDIV and SQRT function description

- Only support word operation
- 8 clock cycles to complete an unsigned integer division operation
- 32-bit dividend, 32-bit divisor, output 32-bit quotient and 32-bit remainder
- The divisor is a zero warning flag, and the division operation is ending flag
- 32-bit unsigned square integer, 16-bit squared root output
- 8 clock cycles to complete an unsigned integer square operation
- Checking whether the calculation is complete or not can be determined by setting the interrupt enable or querying the relevant register bit

### 24.3 HDIV registers

#### 24.3.1 HDIV register overview

Table 24-1 HDIV register overview

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	HDIV_CTRLSTS	Reserved																									HDIVEN	HDIVIF	HDIVDF	HDIVBUSY	HDIVSTART	HDIVEN	
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
004h	HDIV_DIVIDEND	DIVIDEND																															
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
008h	HDIV_DIVISOR	DIVISOR																															
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
00Ch	HDIV_QUOTIENT	QUOTIENT																															
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
010h	HDIV_REMAINDER	REMAINDER																															
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

014h	HDIV_DIVBY0	Reserved	DIVBY0
	Reset Value		0

### 24.3.2 HDIV control status register (HDIV\_CTRLSTS)

Offset address: 0x00

Reset value: 0x0000 0000

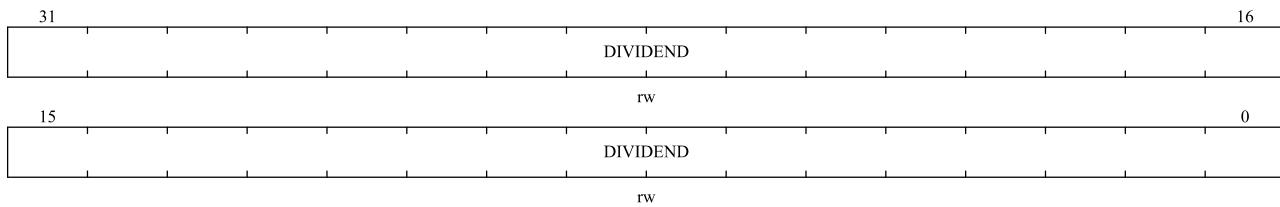
31	Reserved	16																							
15	Reserved	<table border="1"> <tr> <td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td> </tr> <tr> <td></td><td></td><td>HDIVEN</td><td>HDIVIF</td><td>HDIVDF</td><td>HDIV BUSY</td><td>HDIV START</td><td>HDIVEN</td> </tr> <tr> <td></td><td></td><td>rw</td><td>rc_wl</td><td>r</td><td>r</td><td>rw</td><td>rw</td> </tr> </table>	6	5	4	3	2	1	0			HDIVEN	HDIVIF	HDIVDF	HDIV BUSY	HDIV START	HDIVEN			rw	rc_wl	r	r	rw	rw
6	5	4	3	2	1	0																			
		HDIVEN	HDIVIF	HDIVDF	HDIV BUSY	HDIV START	HDIVEN																		
		rw	rc_wl	r	r	rw	rw																		

Bit field	Name	Description
31:6	Reserved	Reserved, the reset value must be maintained
5	HDIVEN	Divider interrupt enable 0: Disable interrupt 1: Enable interrupt
4	HDIVIF	The divider interrupt flag bit, which generates an interrupt signal when the interrupt enable is turned on 0: No interrupt is generated 1: The division calculation is completed and an interrupt is generated Software writes '1' to clear this status bit.
3	HDIVDF	Completion of calculation flag. 0: The division calculation has not ended or has not started 1: The division calculation ends. Readable quotient and remainder
2	HDIVBUSY	Divider busy flag 0: Divider idle 1: Divider is working
1	HDIVSTART	Divider start bit 0: Wait for the division operation to start 1: Start the division calculation Writing this register bit triggers the start of the division operation
0	HDIVEN	Divider module enable 0: Close the divider module 1: Enable divider module

### 24.3.3 HDIV dividend register (HDIV\_DIVIDEND)

Offset address: 0x04

Reset value: 0x0000 0000

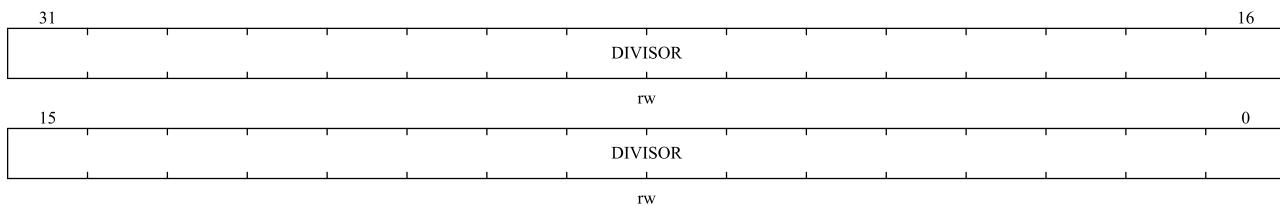


Bit field	Name	Description
31:0	DIVIDEND	32-bit unsigned integer as dividend

#### 24.3.4 HDIV divisor register (HDIV\_DIVISOR)

Offset address: 0x08

Reset value: 0x0000 0000

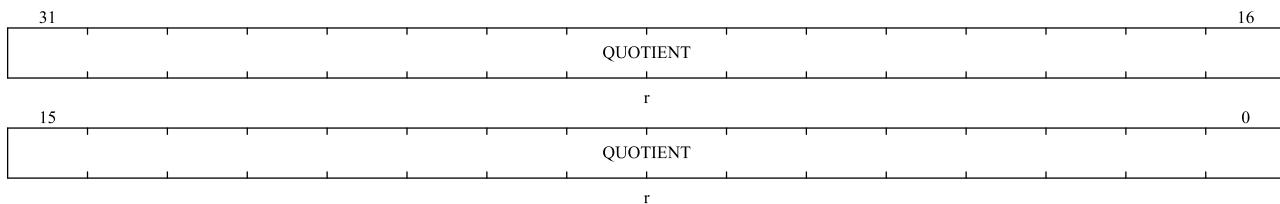


Bit field	Name	Description
31:0	DIVISOR	32-bit unsigned integer as divisor

#### 24.3.5 HDIV quotient register (HDIV\_QUOTIENT)

Offset address: 0x0C

Reset value: 0x0000 0000

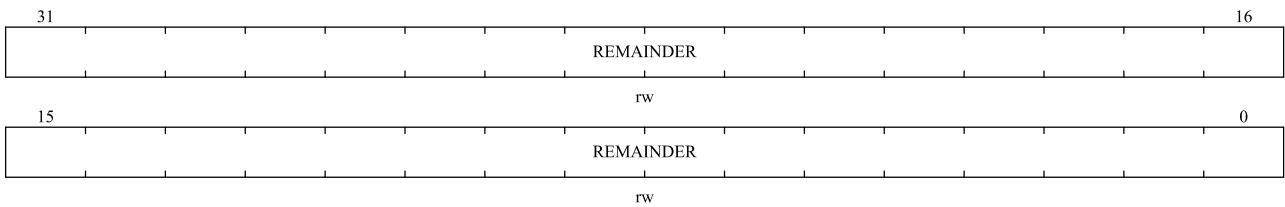


Bit field	Name	Description
31:0	QUOTIENT	Quotient calculated by the divider

#### 24.3.6 HDIV remainder register (HDIV\_REMAINDER)

Offset address: 0x10

Reset value: 0x0000 0000

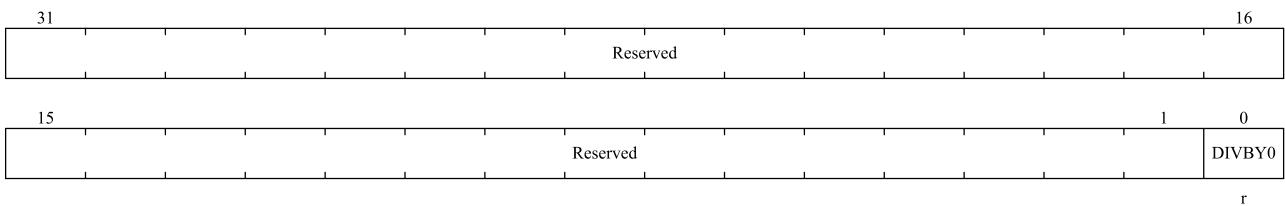


Bit field	Name	Description
31:0	REMAINDER	Remainder calculated by the divider

### 24.3.7 HDIV divide by zero register (HDIV\_DIVBY0)

Offset address: 0x14

Reset value: 0x0000 0000



Bit field	Name	Description
31:1	Reserved	Reserved, the reset value must be maintained
0	DIVBY0	Divisor is 0 flag 0: Divisor is not 0 1: Divisor is 0

## 24.4 SQRT registers

### 24.4.1 SQRT register overview

Table 24-2 SQRT register overview

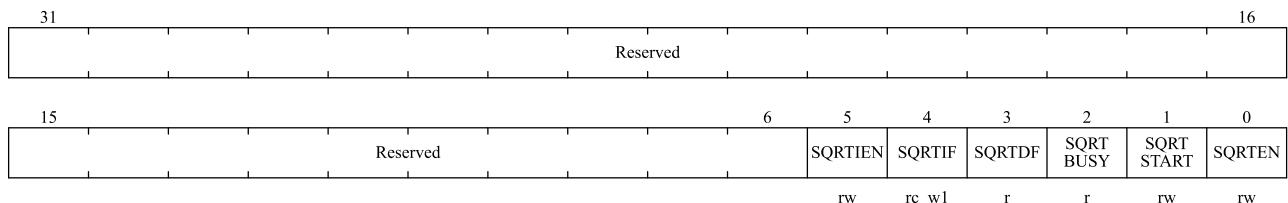
Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	SQRT_CTRLSTS	Reserved																															
	Reset Value																																
004h	SQRT_RADICAND	RADICAND																															
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
008h	SQRT_ROOT	Reserved														ROOT																	

Reset Value 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

#### 24.4.2 SQRT control status register (SQRT\_CTRLSTS)

Offset address: 0x00

Reset value: 0x0000 0000

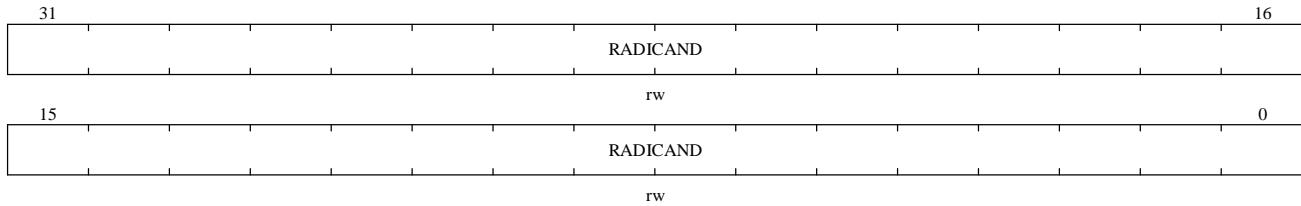


Bit field	Name	Description
31:6	Reserved	Reserved, the reset value must be maintained
5	SQRTIEN	The square root calculator interrupt enable 0□Disable interrupt 1□Enable interrupt
4	SQRTIF	The square root calculator interrupt flag bit, which generates an interrupt signal when the interrupt enable is turned on 0□No interrupt is generated 1□The square root calculation is completed and an interrupt is generated Software writes '1' to clear the status bit
3	SQRTDF	The square root calculator computes the complete flag bit. 0: The calculation has not ended or not started 1: End of calculation
2	SQRTBUSY	The square root calculator busy flag bit 0: The opening calculator is idle 1: The square root calculator is working
1	SQRTSTART	The square root calculation start bit 0: Waiting for the start of square root calculation 1: Configure 1 to start the square root calculation Write this register bit to trigger the start of the square root operation
0	SQRTEN	The square root calculator module enable: 0: Turn off the square root calculator module 1: Enable the square root calculator module

#### 24.4.3 SQRT Radicand register (SQRT RADICAND)

Offset address: 0x04

Reset value: 0x0000 0000

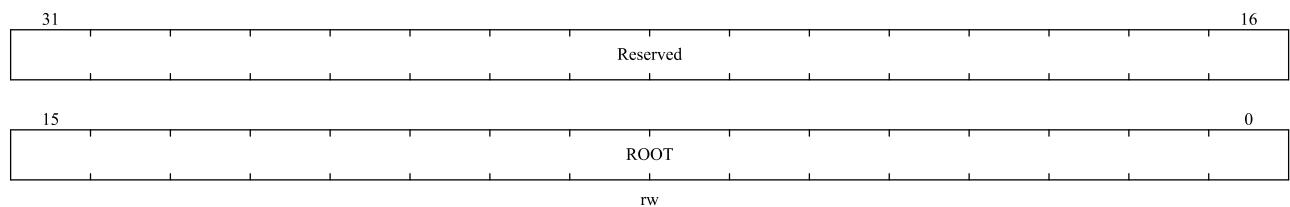


Bit field	Name	Description
31:0	RADICAND	32-bit unsigned integer as radicand

#### 24.4.4 SQRT square root register (SQRT\_ROOT)

Offset address: 0x08

Reset value: 0x0000 0000



Bit field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained
15:0	ROOT	16-bit square root output

## 25 Debug support (DBG)

### 25.1 Overview

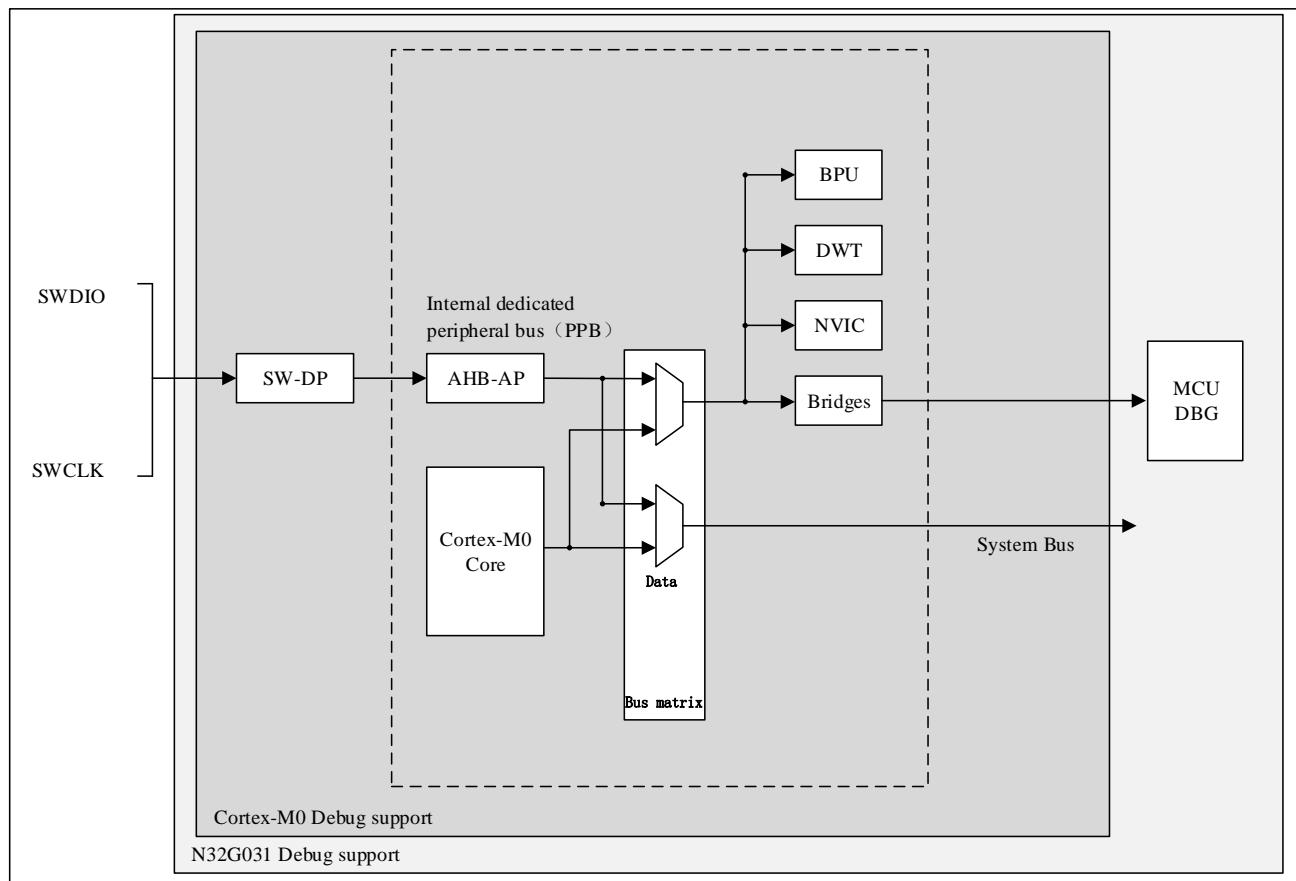
N32G031 uses Cortex®-M0 core, which integrates hardware debugging module. Support instruction breakpoint (stop when instruction fetches value) and data breakpoint (stop when data access). When the core is stopped, the user can view the internal state of the core and the external state of the system. After the user's query operation is completed, the core and peripherals can be restored, and the corresponding program can continue to be executed.

The hardware debugging module of the N32G031 core can be used when it is connected to the debugger (when it is not disabled).

N32G031 supports the following debugging interfaces:

- Serial interface

**Figure 25-1 N32G031 level and Cortex®-M0 level debugging block diagram**



The ARM Cortex®-M0 core hardware debugging module can provide the following debugging functions:

- SW-DP: Serial debugging port
- AHP-AP: AHB access port

- BPU: Breakpoint generation
- DWT: Data trigger

Reference:

- Cortex®-M0 Technical Reference Manual (TRM)
- ARM debug interface V5 structure specification
- ARM CoreSight development tool set (r1p0 version) technical reference manual

## 25.2 SWD function

The debugging tool can call the debugging function through the above-mentioned SWD debugging interface.

### 25.2.1 Pin assignment

SWD (serial debug) interface consists of two pins: SWCLK (clock pin) and SWDIO (data input and output pin) to provide two-pin interface.

The pin assignment of SWD debug interface is shown in the following table:

Table 25-1 Debug port pin

Debug port	Pin assignment
SWDIO	PA13
SWCLK	PA14

## 26 Unique device serial number (UID)

### 26.1 Introduction

MCU series products have two built-in unique device serial numbers with different lengths, namely 96-bit UID (Unique device ID) and 128-bit UCID (Unique Customer ID). These two device serial numbers are stored in the system configuration block of the flash memory, and the information is programmed during manufacture, and any MCU microcontroller is guaranteed to be unique under any circumstances. It can be read by user applications or external devices through CPU or SWD interface and cannot be modified.

UID is 96 bits, which is usually used as serial number or password. When writing flash memory, this unique identifier is combined with software encryption and decryption algorithm to further improve the security of code in flash memory, and it can also be used to activate Secure Bootloader with security function.

UCID is 128 bits and complies with the definition of the Nations Technologies chip serial number. It contains information about chip production and version.

In addition to the above two device serial numbers, there is also a 32-bit DBGMCU\_ID, which contains the chip version number, chip model, and Flash/SRAM capacity information.

### 26.2 UID register

Start address: 0x1FFF\_F4FC, 96 bits in length.

### 26.3 UCID register

Start address: 0x1FFF\_F4D0, 128 bits in length.

### 26.4 DBGMCU\_ID register

Start address: 0x1FFF\_F508, 32 bits in length. For different bytes, the low byte comes first and the high byte follows; for the same byte, the high byte comes first and the low byte follows.

Table 26-1 DBGMCU\_ID bit description

Description	Size	Remark
Chip version number	4bit	The lower 4 bits of the chip version number.
	4bit	The upper 4 bits of the chip version number.
Chip model	4bit	The upper 4 bits of the device model number. The device model consists of 12 high, middle and low bits, representing the model of the chip.
	4bit	The middle 4 bits of the device model number.
	4bit	The lower 4 bits of the device model number.
Flash capacity	4bit	Flash capacity indicator. 16KB as unit, FLASH size = N * 16KB

SRAM capacity	4bit	SRAM capacity indicator. 4KB as unit, SRAM size = N * 4KB
Reserved	4bit	Keep it all 1.

## 27 Version history

Date	Version	Modify
2022.4.28	V2.0	Initial version
2022.9.13	V2.1	<p>Section 7.4.11, modify DMA channel sel 36 to 35</p> <p>Section 9.4.20 and section 10.4.18, modify DMA burst valid address range of TIM, for register after TIMx_DCTRL, DMA burst is invalid</p> <p>Section 3.1.1.1, 3.4.7, 3.2.3, 3.1.1, delete 1.0v output in stop mode.</p> <p>Section 19.2, modify LPUART max rate to 1Mbps</p> <p>Section 21.4.12, when AD1S is set to 1, SUBF[14:0] cannot be all 0</p>
2023.8.1	V2.2.0	<p>Section 26.2 and 26.3, modify the starting address of UID and UCID</p> <p>Section 6.1.1, modify the value from 9000 to 6000, 1.5ms to 1ms.</p> <p>Section 5.2.2, delete “PA0 are multiplexed to OSC_IN”</p> <p>Section 20.4.2, modify clock generator diagram</p> <p>Section 21.3.6, add notes when RTC enters initialization mode</p> <p>Section 21.4.12, modify RTC_SCTRL. SUBF[14:0] register description</p> <p>Section 7.4.6, add notes related to the configuration process</p> <p>Section 21.3.7, add notes when calendar initialization and canlendar reading</p> <p>Section 13.4.1, modify Table 13-1 IWDG counting maximum and minimum reset time</p> <p>Section 17.3.2.6, add notes for configuring the slave address in 7-bit address mode</p>

## 28 Notice

This document is the exclusive property of Nations Technologies Inc. (Hereinafter referred to as NATIONS). This document, and the product of NATIONS described herein (Hereinafter referred to as the Product) are owned by NATIONS under the laws and treaties of the People's Republic of China and other applicable jurisdictions worldwide.

NATIONS does not grant any license under its patents, copyrights, trademarks, or other intellectual property rights. Names and brands of third party may be mentioned or referred thereto (if any) for identification purposes only.

NATIONS reserves the right to make changes, corrections, enhancements, modifications, and improvements to this document at any time without notice. Please contact NATIONS and obtain the latest version of this document before placing orders.

Although NATIONS has attempted to provide accurate and reliable information, NATIONS assumes no responsibility for the accuracy and reliability of this document.

It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. In no event shall NATIONS be liable for any direct, indirect, incidental, special, exemplary, or consequential damages arising in any way out of the use of this document or the Product.

NATIONS Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at user's risk. User shall indemnify NATIONS and hold NATIONS harmless from and against all claims, costs, damages, and other liabilities, arising from or related to any customer's Insecure Usage.

Any express or implied warranty with regard to this document or the Product, including, but not limited to, the warranties of merchantability, fitness for a particular purpose and non-infringement are disclaimed to the fullest extent permitted by law.

Unless otherwise explicitly permitted by NATIONS, anyone may not use, duplicate, modify, transcribe or otherwise distribute this document for any purposes, in whole or in part.