



N32G401F8S7-1 series

32-bit ARM® Cortex®-M4 microcontroller

User manual V1.0

Contents

1 Abbreviations in the text	25
1.1 List of abbreviations for registers	25
1.2 Available peripherals	25
2 Memory and bus architecture.....	26
2.1 System architecture.....	26
2.1.1 Bus architecture	26
2.1.2 Bus address mapping	27
2.1.3 Boot management	31
2.2 Memory system	32
2.2.1 FLASH specification	32
2.2.2 iCache	44
2.2.3 SRAM	46
2.2.4 FLASH register description	46
3 Power control (PWR)	55
3.1 General description.....	55
3.1.1 Power supply.....	55
3.1.2 Power supply supervisor.....	56
3.1.3 NRST	58
3.2 Power modes	58
3.2.1 SLEEP mode.....	60
3.2.2 STOP0 mode.....	61
3.2.3 STOP2 mode.....	62
3.2.4 STANDBY mode	62
3.3 PWR registers	63
3.3.1 PWR register overview	63
3.3.2 Power control register (PWR_CTRL).....	64
3.3.3 Power control status register (PWR_CTRLSTS).....	65
3.3.4 Power control register 2(PWR_CTRL2).....	67
4 Reset and clock control (RCC)	68
4.1 General description.....	68
4.2 Reset Control Unit	68
4.2.1 Power reset.....	68
4.2.2 System reset	68
4.2.3 Backup domain reset.....	70
4.3 Clock control unit	70
4.3.1 Clock Tree Diagram.....	71
4.3.2 HSE clock	71
4.3.3 HSI clock	72

4.3.4	PLL clock.....	73
4.3.5	LSI clock.....	73
4.3.6	System clock (SYSCLK) selection.....	73
4.3.7	Clock security system (CLKSS)	74
4.3.8	RTC clock	74
4.3.9	Watchdog clock.....	74
4.3.10	Clock output(MCO).....	74
4.4	RCC Registers	75
4.4.1	RCC register overview.....	75
4.4.2	Clock Control Register (RCC_CTRL).....	76
4.4.3	Clock Configuration Register (RCC_CFG)	78
4.4.4	Clock Interrupt Register (RCC_CLKINT).....	81
4.4.5	APB2 Peripheral Reset Register (RCC_APB2PRST)	83
4.4.6	APB1 Peripheral Reset Register (RCC_APB1PRST)	84
4.4.7	AHB Peripheral Clock Enable Register (RCC_AHBPCLEN).....	86
4.4.8	APB2 Peripheral Clock Enable Register (RCC_APB2PCLKEN).....	87
4.4.9	APB1 Peripheral Clock Enable Register (RCC_APB1PCLKEN).....	88
4.4.10	Backup Domain Control Register (RCC_BDCTRL).....	89
4.4.11	Clock Control/Status Register (RCC_CTRLSTS)	90
4.4.12	AHB Peripheral Reset Register (RCC_AHBPRST)	92
4.4.13	Clock Configuration Register 2 (RCC_CFG2)	93
4.4.14	Retention Domain Control Register (RCC_RDCTRL)	94
4.4.15	PLL and HSI Configuration Register (RCC_PLLHSIPRE)	95
4.4.16	AHB Peripheral Clock Enable Register 1 (RCC_AHB1CLKEN).....	96
5	GPIO and AFIO.....	97
5.1	Summary.....	97
5.2	Function description	98
5.2.1	I/O mode configuration.....	98
5.2.2	Status after reset.....	103
5.2.3	Atomic bit set and reset.....	103
5.2.4	External Interrupt /wakeup line.....	103
5.2.5	Alternate function	104
5.2.6	I/O configuration of peripherals.....	111
5.2.7	GPIO Locking mechanism.....	112
5.2.8	Precautions for using GPIO	113
5.3	GPIO Registers	113
5.3.1	GPIOA register overview.....	113
5.3.2	GPIOB register overview.....	115
5.3.3	GPIOD register overview	117
5.3.4	GPIO port mode description register (GPIOx_PMODE).....	118
5.3.5	GPIO port type definition (GPIOx_POTYPE)	119
5.3.6	GPIO slew rate configuration register (GPIOx_SR)	120
5.3.7	GPIO port pull-up/pull-down description register (GPIOx_PUPD)	120

5.3.8	GPIO port input data register (GPIOx_PID).....	121
5.3.9	GPIO port output data register (GPIOx_POD)	121
5.3.10	GPIO port bit set/clear register (GPIOx_PBSC).....	122
5.3.11	GPIO port bit clear register (GPIOx_PBC)	123
5.3.12	GPIO port configuration lock register (GPIOx_PLOCK)	123
5.3.13	GPIO Alternate function low register (GPIOx_AFL)	124
5.3.14	GPIO Alternate function high register (GPIOx_AFH)	125
5.3.15	GPIO driver strength configuration register (GPIOx_DS)	126
5.4	AFIO Registers	127
5.4.1	AFIO register overview	127
5.4.2	Alternate function mapping configuration control register (AFIO_RMP_CFG)	128
5.4.3	External interrupt configuration register 1 (AFIO_EXTI_CFG1)	129
5.4.4	External interrupt configuration register 2 (AFIO_EXTI_CFG2)	136
5.4.5	External interrupt configuration register 3 (AFIO_EXTI_CFG3)	143
5.4.6	External interrupt configuration register 4 (AFIO_EXTI_CFG4)	149
5.4.7	5V tolerance configuration register (AFIO_TOL5V_CFG).....	156
5.4.8	Analog filter configuration register 1 (AFIO_EFT_CFG1)	157
5.4.9	Analog filter configuration register 2 (AFIO_EFT_CFG2)	158
5.4.10	Digital glitch filter stage (glitch width) configuration register (AFIO_FILT_CFG).....	159
5.4.11	Digital glitch filter configuration register 1 (AFIO_DIGEFT_CFG1)	159
5.4.12	Digital glitch filter configuration register 2 (AFIO_DIGEFT_CFG2)	160
6	Interrupts and events.....	161
6.1	Nested vectored interrupt controller	161
6.1.1	SysTick calibration value register.....	161
6.1.2	Interrupt and exception vectors.....	161
6.2	External interrupt/event controller (EXTI)	164
6.2.1	Introduction to EXTI.....	164
6.2.2	EXTI main features.....	164
6.2.3	Functional description.....	165
6.2.4	EXTI line mapping	167
6.3	EXTI Registers	168
6.3.1	EXTI register overview.....	168
6.3.2	Interrupt Mask Register(EXTI_IMASK)	168
6.3.3	Event Mask Register(EXTI_EMASK)	169
6.3.4	Rising Edge Trigger Selection Register(EXTI_RT_CFG)	169
6.3.5	Falling Edge Trigger Selection Register(EXTI_FT_CFG)	170
6.3.6	Software Interrupt Event Register(EXTI_SWIE)	170
6.3.7	Interrupt Request Pending Register(EXTI_PEND)	171
6.3.8	RTC Timestamp Selection Register(EXTI_TS_SEL)	171
7	DMA controller	173
7.1	Introduction	173
7.2	Main features	173

7.3 Block diagram	174
7.4 Function description	174
7.4.1 DMA operation	174
7.4.2 Channel priority and arbitration	175
7.4.3 DMA channels and number of transfers.....	175
7.4.4 Programmable data bit width, alignment and endians	175
7.4.5 Peripheral/Memory address incrementation	177
7.4.6 Channel configuration procedure	177
7.4.7 Flow control.....	178
7.4.8 Circular mode	179
7.4.9 Error management.....	179
7.4.10 Interrupt	179
7.4.11 DMA request mapping.....	179
7.5 DMA registers	182
7.5.1 DMA register overview.....	182
7.5.2 DMA interrupt status register (DMA_INTSTS)	183
7.5.3 DMA interrupt flag clear register (DMA_INTCLR).....	184
7.5.4 DMA channel x configuration register (DMA_CHCFGx).....	185
7.5.5 DMA channel x transfer number register (DMA_TXNUMx)	187
7.5.6 DMA channel x peripheral address register (DMA_PADDRx)	187
7.5.7 DMA channel x memory address register (DMA_MADDRx)	188
7.5.8 DMA channel x channel request select register (DMA_CHSELx).....	188
8 CRC calculation unit	191
8.1 CRC introduction.....	191
8.2 CRC main features.....	191
8.2.1 CRC32 module	191
8.2.2 CRC16 module	191
8.3 CRC function description	192
8.3.1 CRC32	192
8.3.2 CRC16	192
8.4 CRC registers.....	193
8.4.1 CRC register overview.....	193
8.4.2 CRC32 data register (CRC_CRC32DAT).....	193
8.4.3 CRC32 independent data register (CRC_CRC32IDAT).....	193
8.4.4 CRC32 control register (CRC_CRC32CTRL).....	194
8.4.5 CRC16 control register (CRC_CRC16CTRL).....	194
8.4.6 CRC16 input data register (CRC_CRC16DAT)	195
8.4.7 CRC cyclic redundancy check code register (CRC_CRC16D)	195
8.4.8 LRC result register (CRC_LRC).....	196
9 Advanced-control timers (TIM1 and TIM8)	197
9.1 TIM1 and TIM8 introduction	197
9.2 Main features of TIM1 and TIM8.....	197

9.3 TIM1 and TIM8 function description	199
9.3.1 Time-base unit.....	199
9.3.2 Counter mode.....	200
9.3.3 Repetition counter	210
9.3.4 Clock selection.....	212
9.3.5 Capture/compare channels	216
9.3.6 Input capture mode	218
9.3.7 PWM input mode	219
9.3.8 Forced output mode	220
9.3.9 Output compare mode.....	221
9.3.10 PWM mode.....	222
9.3.11 One-pulse mode	225
9.3.12 Clearing the OCxREF signal on an external event	226
9.3.13 Complementary outputs with dead-time insertion	227
9.3.14 Break function.....	229
9.3.15 Debug mode.....	231
9.3.16 TIMx and external trigger synchronization	231
9.3.17 Timer synchronization	235
9.3.18 6-step PWM generation	235
9.3.19 Encoder interface mode	236
9.3.20 Interfacing with Hall sensor	239
9.4 TIMx register description(x=1, 8)	241
9.4.1 Register overview	241
9.4.2 Control register 1 (TIMx_CTRL1)	244
9.4.3 Control register 2 (TIMx_CTRL2)	246
9.4.4 Slave mode control register (TIMx_SMCTRL).....	249
9.4.5 DMA/Interrupt enable registers (TIMx_DINTEN).....	251
9.4.6 Status registers (TIMx_STS)	253
9.4.7 Event generation registers (TIMx_EVTGEN)	255
9.4.8 Capture/compare mode register 1 (TIMx_CCMOD1).....	256
9.4.9 Capture/compare mode register 2 (TIMx_CCMOD2).....	259
9.4.10 Capture/compare enable registers (TIMx_CCEN).....	261
9.4.11 Counters (TIMx_CNT).....	264
9.4.12 Prescaler (TIMx_PSC).....	264
9.4.13 Auto-reload register (TIMx_AR).....	265
9.4.14 Repeat count registers (TIMx_REPCNT).....	265
9.4.15 Capture/compare register 1 (TIMx_CCDAT1)	265
9.4.16 Capture/compare register 2 (TIMx_CCDAT2)	266
9.4.17 Capture/compare register 3 (TIMx_CCDAT3)	267
9.4.18 Capture/compare register 4 (TIMx_CCDAT4)	268
9.4.19 Break and Dead-time registers (TIMx_BKDT)	269
9.4.20 DMA Control register (TIMx_DCTRL)	270
9.4.21 DMA transfer buffer register (TIMx_DADDR)	271

9.4.22	Capture/compare mode registers 3(TIMx_CCMOD3)	272
9.4.23	Capture/compare register 5 (TIMx_CCDAT5)	272
9.4.24	Capture/compare register 6 (TIMx_CCDAT6)	273
9.4.25	Capture/compare register 7 (TIMx_CCDAT7)	273
9.4.26	Capture/compare register 8 (TIMx_CCDAT8)	274
9.4.27	Capture/compare register 9 (TIMx_CCDAT9)	274
9.4.28	Break Filter (TIMx_BRKFILT)	274
10	General-purpose timers (TIM2, TIM3, TIM4 and TIM5)	276
10.1	General-purpose timers introduction.....	276
10.2	Main features of General-purpose timers	276
10.3	TIM2、TIM3、TIM4、TIM5 function description.....	280
10.3.1	Time-base unit.....	280
10.3.2	Counter mode.....	281
10.3.3	Clock selection.....	287
10.3.4	Capture/compare channels	291
10.3.5	Input capture mode	294
10.3.6	PWM input mode	295
10.3.7	Forced output mode.....	296
10.3.8	Output compare mode	297
10.3.9	PWM mode.....	298
10.3.10	One-pulse mode	301
10.3.11	Clearing the OCxREF signal on an external event	302
10.3.12	Debug mode.....	303
10.3.13	TIMx and external trigger synchronization.....	303
10.3.14	Timer synchronization	307
10.3.15	Encoder interface mode	311
10.3.16	Interfacing with Hall sensor	314
10.4	TIMx register description(x=2, 3, 4 and 5).....	315
10.4.1	Register overview	315
10.4.2	Control register 1 (TIMx_CTRL1)	316
10.4.3	Control register 2 (TIMx_CTRL2)	318
10.4.4	Slave mode control register (TIMx_SMCTRL).....	320
10.4.5	DMA/Interrupt enable registers (TIMx_DINTEN)	322
10.4.6	Status registers (TIMx_STS)	323
10.4.7	Event generation registers (TIMx_EVTGEN)	325
10.4.8	Capture/compare mode register 1 (TIMx_CCMOD1).....	326
10.4.9	Capture/compare mode register 2 (TIMx_CCMOD2).....	329
10.4.10	Capture/compare enable registers (TIMx_CCEN).....	331
10.4.11	Counters (TIMx_CNT).....	332
10.4.12	Prescaler (TIMx_PSC).....	332
10.4.13	Auto-reload register (TIMx_AR).....	333
10.4.14	Capture/compare register 1 (TIMx_CCDAT1)	333
10.4.15	Capture/compare register 2 (TIMx_CCDAT2)	334

10.4.16	Capture/compare register 3 (TIMx_CCData3)	334
10.4.17	Capture/compare register 4 (TIMx_CCData4)	335
10.4.18	DMA Control register (TIMx_DCTRL)	335
10.4.19	DMA transfer buffer register (TIMx_DADDR)	336
10.4.20	Channel 1 filter register (TIMx_C1FILT)	337
10.4.21	Channel 2 filter register (TIMx_C2FILT)	338
10.4.22	Channel 3 filter register (TIMx_C3FILT)	338
10.4.23	Channel 4 filter register (TIMx_C4FILT)	339
10.4.24	Channel filter Output register (TIMx_FILTO)	340
11	Basic timers (TIM6)	342
11.1	Main features of Basic timers	342
11.2	Basic timers description	342
11.2.1	Time-base unit	342
11.2.2	Counter mode	343
11.2.3	Clock selection	346
11.2.4	Debug mode	346
11.3	TIMx register description(x=6)	346
11.3.1	Register overview	347
11.3.2	Control Register 1 (TIMx_CTRL1)	347
11.3.3	DMA/Interrupt Enable Registers (TIMx_DINTEN)	348
11.3.4	Status Registers (TIMx_STS)	349
11.3.5	Event Generation registers (TIMx_EVTGEN)	349
11.3.6	Counters (TIMx_CNT)	350
11.3.7	Prescaler (TIMx_PSC)	350
11.3.8	Automatic reload register (TIMx_AR)	350
12	Low Power Timer (LPTIM)	351
12.1	Introduction	351
12.2	Main Features	351
12.3	Block diagram	352
12.4	Function description	352
12.4.1	LPTIM clocks and on-off control	352
12.4.2	Prescaler	353
12.4.3	Glitch filter	353
12.4.4	Timer enable	354
12.4.5	Trigger multiplexer	354
12.4.6	Operating mode	355
12.4.7	Waveform generation	357
12.4.8	Register update	358
12.4.9	Counter mode	359
12.4.10	Encoder mode	360
12.4.11	Non-orthogonal encoder mode	361
12.4.12	Timeout function	362

12.4.13 LPTIM interrupts	363
12.5 LPTIM registers.....	364
12.5.1 LPTIM register overview.....	364
12.5.2 LPTIM interrupt and status register (LPTIM_INTSTS)	364
12.5.3 LPTIM interrupt clear register (LPTIM_INTCLR)	365
12.5.4 LPTIM interrupt enable register (LPTIM_INTEN).....	366
12.5.5 LPTIM configuration register (LPTIM_CFG).....	367
12.5.6 LPTIM control register (LPTIM_CTRL)	369
12.5.7 LPTIM compare register (LPTIM_COMP).....	370
12.5.8 LPTIM auto-reload register (LPTIM_ARR).....	371
12.5.9 LPTIM counter register (LPTIM_CNT).....	371
13 Independent watchdog (IWDG)	372
13.1 Introduction	372
13.2 Main features	372
13.3 Function description	373
13.3.1 Register access protection.....	373
13.3.2 Debug mode.....	374
13.3.3 IWDG Freeze.....	374
13.4 User Interface	374
13.4.1 Operate Flow	374
13.4.2 IWDG configuration flow.....	375
13.5 IWDG registers.....	375
13.5.1 IWDG register overview.....	375
13.5.2 IWDG Key Register (IWDG_KEY)	376
13.5.3 IWDG Pre-Scaler Register (IWDG_PREDIV)	376
13.5.4 IWDG Reload Register (IWDG_RELV)	377
13.5.5 IWDG Status Register (IWDG_STS)	377
14 Window watchdog (WWDG)	379
14.1 Introduction	379
14.2 Main features	379
14.3 Function description	379
14.4 Timing for refresh watchdog and interrupt generation	380
14.5 Debug mode.....	381
14.6 User Interface	381
14.6.1 WWDG configuration flow	381
14.7 WWDG registers	382
14.7.1 WWDG register Map.....	382
14.7.2 WWDG control register (WWDG_CTRL)	382
14.7.3 WWDG config register (WWDG_CFG)	383
14.7.4 WWDG status register (WWDG_STS)	383
15 Analog to digital conversion (ADC).....	384

15.1 Introduction	384
15.2 Main features	384
15.3 Function Description.....	385
15.3.1 ADC clock	386
15.3.2 ADC switch control.....	387
15.3.3 Channel selection	387
15.3.4 Internal channel	389
15.3.5 Single conversion mode.....	389
15.3.6 Continuous conversion mode.....	389
15.3.7 Timing diagram.....	389
15.3.8 Analog watchdog	390
15.3.9 Scanning mode	391
15.3.10 Injection channel management.....	391
15.3.11 Discontinuous mode	392
15.4 Calibration	393
15.5 Data aligned	393
15.6 Programmable channel sampling time	395
15.7 Externally triggered conversion.....	395
15.8 DMA requests	396
15.9 Temperature sensor.....	396
15.9.1 Temperature sensor using flow	397
15.10 ADC interrupt	398
15.11 ADC registers.....	398
15.11.1 ADC register overview	399
15.11.2 ADC status register (ADC_STS).....	400
15.11.3 ADC control register 1 (ADC_CTRL1)	401
15.11.4 ADC control register 2 (ADC_CTRL2)	403
15.11.5 ADC sampling time register 1 (ADC_SAMPT1)	405
15.11.6 ADC sampling time register 2 (ADC_SAMPT2)	406
15.11.7 ADC injected channel data offset register x (ADC_JOFFSETx) (x=1...4).....	407
15.11.8 ADC watchdog high threshold register (ADC_WDGHIGH)	407
15.11.9 ADC watchdog low threshold register (ADC_WDGLOW)	407
15.11.10 ADC regular sequence register 1 (ADC_RSEQ1).....	408
15.11.11 ADC regular sequence register 2 (ADC_RSEQ2).....	408
15.11.12 ADC regular sequence register 3 (ADC_RSEQ3).....	409
15.11.13 ADC Injection sequence register (ADC_JSEQ).....	409
15.11.14 ADC injection data register x (ADC_JDATx) (x= 1...4)	410
15.11.15 ADC regulars data register (ADC_DAT)	410
15.11.16 ADC differential mode selection register (ADC_DIFSEL).....	411
15.11.17 ADC calibration factor (ADC_CALFACT).....	411
15.11.18 ADC control register 3 (ADC_CTRL3)	412
15.11.19 ADC sampling time register 3 (ADC_SAMPT3).....	413
16 I²C interface	415

16.1 Introduction	415
16.2 Characteristics	415
16.3 Function description	415
16.3.1 SDA and SCL line control	416
16.3.2 Software communication process	416
16.3.3 Error conditions description.....	425
16.3.4 DMA application	426
16.3.5 Packet error check.....	428
16.3.6 Noise filter	428
16.3.7 SMBus	429
16.4 Debug mode.....	431
16.5 Interrupt request.....	431
16.6 I2C register	432
16.6.1 I2C register overview	432
16.6.2 Control register 1 (I2C_CTRL1)	432
16.6.3 Control register 2 (I2C_CTRL2)	434
16.6.4 Own address register 1 (I2C_OADDR1).....	436
16.6.5 Own address register 2 (I2C_OADDR2).....	436
16.6.6 Data register (I2C_DAT)	437
16.6.7 Status register 1 (I2C_STS1)	437
16.6.8 Status register 2 (I2C_STS2)	440
16.6.9 Clock control register (I2C_CLKCTRL).....	441
16.6.10 Rise time register (I2C_TMRISE)	442
17 Universal synchronous asynchronous receiver transmitter (USART)	444
17.1 Introduction	444
17.2 Main features	444
17.3 Functional block diagram	445
17.4 Function description	445
17.4.1 USART frame format.....	446
17.4.2 Transmitter.....	447
17.4.3 Receiver	450
17.4.4 Generation of fractional baud rate	454
17.4.5 Receiver's tolerance clock deviation	455
17.4.6 Parity control	456
17.4.7 DMA application	457
17.4.8 Hardware flow control	459
17.4.9 Multiprocessor communication	461
17.4.10 Synchronous mode.....	463
17.4.11 Single-wire half-duplex mode	466
17.4.12 IrDA SIR ENDEC mode.....	467
17.4.13 LIN mode.....	469
17.4.14 Smartcard mode (ISO7816)	471
17.5 Interrupt request.....	473

17.6 Mode support.....	474
17.7 USART register	474
17.7.1 USART register overview.....	474
17.7.2 USART Status register (USART_STS).....	476
17.7.3 USART Data register (USART_DAT).....	478
17.7.4 USART Baud rate register (USART_BRCF).....	478
17.7.5 USART control register 1 register (USART_CTRL1).....	479
17.7.6 USART control register 2 register (USART_CTRL2).....	481
17.7.7 USART control register 3 register (USART_CTRL3).....	482
17.7.8 USART guard time and prescaler register (USART_GTP).....	484
18 Serial peripheral interface/Inter-IC Sound (SPI/ I²S)	486
18.1 SPI/ I ² S introduction.....	486
18.2 SPI and I ² S main features	486
18.2.1 SPI features.....	486
18.2.2 I ² S features.....	486
18.3 SPI function description	487
18.3.1 General description	487
18.3.2 SPI work mode	490
18.3.3 Status flag	496
18.3.4 Disabling the SPI	497
18.3.5 SPI communication using DMA.....	498
18.3.6 CRC calculation.....	499
18.3.7 Error flag.....	500
18.3.8 SPI interrupt.....	501
18.4 I ² S function description	502
18.4.1 Supported audio protocols	503
18.4.2 Clock generator.....	509
18.4.3 I ² S Transmission and reception sequence	511
18.4.4 Status flag	513
18.4.5 Error flag.....	514
18.4.6 I ² S interrupt.....	514
18.4.7 DMA function.....	514
18.5 SPI and I ² S register description	515
18.5.1 SPI register overview.....	515
18.5.2 SPI control register 1 (SPI_CTRL1) (not used in I ² S mode)	515
18.5.3 SPI control register 2 (SPI_CTRL2).....	518
18.5.4 SPI status register (SPI_STS)	518
18.5.5 SPI data register (SPI_DAT).....	520
18.5.6 SPI CRC polynomial register (SPI_CRCPOLY) (not used in I ² S mode).....	520
18.5.7 SPI RX CRC register (SPI_CRCRDATA) (not used in I ² S mode).....	520
18.5.8 SPI TX CRC register (SPI_CRCTDATA)	521
18.5.9 SPI_I ² S configuration register (SPI_I2SCFG)	521
18.5.10 SPI_I ² S prescaler register (SPI_I2SPREDIV)	523

19 Real-time clock (RTC).....	524
19.1 Description	524
19.1.1 Specification	525
19.2 RTC function description.....	526
19.2.1 RTC block diagram.....	526
19.2.2 GPIOs of RTC.....	527
19.2.3 RTC register write protection	527
19.2.4 RTC clock and prescaler	527
19.2.5 RTC calendar	528
19.2.6 Calendar initialization and configuration.....	528
19.2.7 Calendar reading	528
19.2.8 Programmable alarms	529
19.2.9 Alarm configuration.....	529
19.2.10 Periodic automatic wakeup.....	530
19.2.11 Wakeup timer configuration	530
19.2.12 Timestamp function	530
19.2.13 Tamper Detection.....	531
19.2.14 Daylight saving time configuration	532
19.2.15 RTC sub-second register shift.....	532
19.2.16 RTC digital clock precision calibration	532
19.2.17 RTC low power mode	533
19.3 RTC Registers.....	534
19.3.1 RTC Register overview.....	534
19.3.2 RTC Calendar Time Register (RTC_TSH)	535
19.3.3 RTC Calendar Date Register (RTC_DATE)	536
19.3.4 RTC Control Register (RTC_CTRL)	536
19.3.5 RTC Initial Status Register (RTC_INITSTS)	538
19.3.6 RTC Prescaler Register (RTC_PRE)	540
19.3.7 RTC Wakeup Timer Register (RTC_WKUPT).....	541
19.3.8 RTC Alarm A Register (RTC_ALARMA).....	541
19.3.9 RTC Alarm B Register (RTC_ALARMB)	542
19.3.10 RTC Write Protection register (RTC_WRP).....	543
19.3.11 RTC Sub-second Register (RTC_SUBS).....	544
19.3.12 RTC Shift Control Register (RTC_SCTRL)	544
19.3.13 RTC Timestamp Time Register (RTC_TST)	545
19.3.14 RTC Timestamp Date Register (RTC_TSD).....	545
19.3.15 RTC Timestamp Sub-second Register (RTC_TSSS)	546
19.3.16 RTC Calibration Register (RTC_CALIB).....	546
19.3.17 RTC Tamper Configuration Register (RTC_TMPCFG)	547
19.3.18 RTC Alarm A sub-second register (RTC_ALRMASS).....	549
19.3.19 RTC Alarm B sub-second register (RTC_ALRMBSS).....	550
19.3.20 RTC Backup registers (RTC_BKP(1~20)).....	550
20 Beeper	552

20.1 Introduction	552
20.2 Function description	552
20.2.1 Main Features	552
20.2.2 Setup frequency for pre-scale ratio of beeper	552
20.2.3 Bypass clock	552
20.2.4 Output Duty cycle	553
20.3 Beeper registers	553
20.3.1 Beeper register overview	553
20.3.2 Beeper Control Register (BEEPER_CTRL)	553
21 Debug support (DBG).....	555
21.1 Overview	555
21.2 SW function	556
21.2.1 Pin allocation	556
21.3 MCU debugging function	556
21.3.1 Low power mode support	556
21.3.2 Peripheral debugging support	557
21.4 DBG registers	557
21.4.1 DBG register overview	557
21.4.2 ID register (DBG_ID)	557
21.4.3 Debug control register (DBG_CTRL)	558
22 Unique device serial number (UID).....	560
22.1 Introduction	560
22.2 UID register	560
22.3 UCID register	560
23 Version history	561
24 Notice	562

List of Table

Table 2-1 List of peripheral register addresses	29
Table 2-2 List of boot mode.....	32
Table 2-3 Flash bus address list	33
Table 2-4 Option byte list	37
Table 2-5 Read protection configuration list	39
Table 2-6 Flash read-write-erase ⁽¹⁾ permission control table	40
Table 2-7 FLASH register overview.....	47
Table 3-1 Power modes.....	58
Table 3-2 Modules running status.....	59
Table 3-3 PWR register overview.....	63
Table 4-1 RCC register overview	75
Table 5-1 I/O port configuration table	98
Table 5-2 I/O List of functional features of the pin	99
Table 5-3 special pins after reset.....	103
Table 5-4 Debug interface signal	105
Table 5-5 Debug port image	105
Table 5-6 ADC external trigger injected conversion alternate function remapping.....	106
Table 5-7 ADC external trigger regular conversion alternate function remapping.....	106
Table 5-8 TIM1 alternate function remapping.....	106
Table 5-9 TIM2 alternate function remapping	106
Table 5-10 TIM3 alternate function remapping	107
Table 5-11 TIM4 alternate function remapping	107
Table 5-12 TIM5 alternate function remapping	107
Table 5-13 TIM8 alternate function remapping	107
Table 5-14 LPTIM alternate function remapping	108
Table 5-15 USART2 alternate function remapping	108
Table 5-16 UART1 alternate function remapping.....	108
Table 5-17 UART4 alternate function remapping.....	108
Table 5-18 I2C1 alternate function remapping	109
Table 5-19 I2C2 alternate function remapping	109

Table 5-20 SPI1/I2S1 alternate function remapping	109
Table 5-21 SPI2/I2S2 alternate function remapping	110
Table 5-22 EVENTOUT alternate function remapping	110
Table 5-23 BEEPER alternate function remapping	110
Table 5-24 SWD alternate function remapping	110
Table 5-25 RTC_REFIN alternate function remapping	110
Table 5-26 MCO alternate function remapping	110
Table 5-27 ADC	111
Table 5-28 TIM1/TIM8	111
Table 5-29 TIM2/3/4/5	111
Table 5-30 LPTIM	111
Table 5-31 USART	111
Table 5-32 UART	111
Table 5-33 I2C	112
Table 5-34 SPI-I2S	112
Table 5-35 SWD	112
Table 5-36 RTC	112
Table 5-37 EVENT_OUT	112
Table 5-38 Other	112
Table 5-39 GPIOA register overview	113
Table 5-40 GPIOB register overview	115
Table 5-41 GPIOD register overview	117
Table 5-42 AFIO register overview	127
Table 6-1 Vector table	161
Table 6-2 EXTI register overview	168
Table 7-1 Programmable data width and endian operation (when PINC = MINC = 1)	175
Table 7-2 Flow control table	178
Table 7-3 DMA interrupt request	179
Table 7-4 DMA request mapping	180
Table 7-5 DMA register overview	182
Table 8-1 CRC register overview	193

Table 9-1 Counting direction versus encoder signals	237
Table 9-2 Register overview	241
Table 9-3 TIMx internal trigger connection.....	251
Table 9-4 Output control bits of complementary OCx and OCxN channels with break function	263
Table 10-1 Counting direction versus encoder signals	312
Table 10-2 Register overview	315
Table 10-3 TIMx internal trigger connection.....	322
Table 10-4 Output control bits of standard OCx channel	332
Table 11-1 Register overview	347
Table 12-1 Pre-scalar division ratios.....	353
Table 12-2 4 trigger inputs corresponding to LPTIM_CFG.TRGSEL[3:0] bits.....	355
Table 12-3 Encoder counting scenarios	360
Table 12-4 Interruption events	363
Table 12-5 LPTIM register overview	364
Table 13-1 IWDG counting maximum and minimum reset time	375
Table 13-2 IWDG register overview.....	375
Table 14-1 Maximum and minimum counting time of WWDG.....	381
Table 14-2 WWDG register overview	382
Table 15-1 ADC pins	385
Table 15-2 Analog watchdog channel selection.....	390
Table 15-3 Right-align data	394
Table 15-4 Left-align data.....	394
Table 15-5 ADC is used for external triggering of regular channels	395
Table 15-6 ADC is used for external triggering of injection channels.....	396
Table 15-7 ADC interrupt	398
Table 15-8 ADC register overview	399
Table 16-1 SDADFW/ SCLDFW recommended configuration	429
Table 16-2 Comparison between SMBus and I2C.....	429
Table 16-3 I ² C interrupt request.....	431
Table 16-4 I2C register overview	432
Table 17-1 Stop bit configuration	448

Table 17-2 Data sampling for noise detection	453
Table 17-3 Error calculation when setting baud rate	455
Table 17-4 when DIV_Decimal = 0. Tolerance of USART receiver	456
Table 17-5 when DIV_Decimal != 0, Tolerance of USART receiver.....	456
Table 17-6 Frame format	456
Table 17-7 USART interrupt request	473
Table 17-8 USART mode setting ⁽¹⁾	474
Table 17-9 USART register overview.....	474
Table 18-1 SPI interrupt request	501
Table 18-2 Use the standard 8MHz HSE clock to get accurate audio frequency.....	511
Table 18-3 I ² S interrupt request.....	514
Table 18-4 SPI register overview.....	515
Table 19-1 RTC feature support.....	525
Table 19-2 RTC register overview	534
Table 20-1 Max and Min frequency supported by beeper and corresponding configure.....	552
Table 20-2 Beeper register overview	553
Table 21-1 Debug port pin	556
Table 21-2 DBG register overview	557

List of Figure

Figure 2-1 Bus architecture	26
Figure 2-2 Bus address map	28
Figure 3-1 Power supply block diagram.....	56
Figure 3-2 Power on reset/power down reset waveform	57
Figure 3-3 PVD threshold diagram.....	57
Figure 4-1 System reset generation	69
Figure 4-2 Clock Tree.....	71
Figure 4-3 HSE clock source	72
Figure 5-1 5V tolerant I/O structure	98
Figure 5-2 Alternate function mode	100
Figure 5-3 Input floating / pull-up / pull-down configuration mode.	101
Figure 5-4 Output mode.....	102
Figure 5-5 Analog input configuration with High Impedance.....	102
Figure 5-6 EXTI global interrupts	104
Figure 6-1 EXTI functional diagram	165
Figure 6-2 External interrupt GPIO mapping	167
Figure 7-1 DMA block diagram.....	174
Figure 8-1 CRC calculation unit block diagram.....	192
Figure 9-1 Block diagram of TIM1	198
Figure 9-2 Block diagram of TIM8	199
Figure 9-3 Counter timing diagram with prescaler division change from 1 to 4	200
Figure 9-4 Timing diagram of up-counting. The internal clock divider factor = 2/N	201
Figure 9-5 Timing diagram of the up-counting, update event when ARPEN=0/1.....	202
Figure 9-6 Timing diagram of the down-counting, internal clock divided factor = 2/N.....	203
Figure 9-7 Timing diagram of the Center-aligned, internal clock divided factor =2/N	204
Figure 9-8 A center-aligned sequence diagram that includes counter overflows and underflows (ARPEN = 1)	205
Figure 9-9 The output waveform corresponding to the asymmetric mode	207
Figure 9-10 CC DATx(x=4,7,8,9), trigger ADC when DIR = 0	208
Figure 9-11 CC DATx(x=4,7,8,9), trigger ADC when DIR = 1	209

Figure 9-12 CCDDATx(x=4,7,8,9), trigger ADC when DIR = 0 or DIR = 1	210
Figure 9-13 Repeat count sequence diagram in down-counting mode	211
Figure 9-14 Repeat count sequence diagram in up-counting mode.....	211
Figure 9-15 Repeat count sequence diagram in center-aligned mode	212
Figure 9-16 Control circuit in normal mode, internal clock divided by 1	213
Figure 9-17 TI2 external clock connection example	213
Figure 9-18 Control circuit in external clock mode 1	214
Figure 9-19 External trigger input block diagram	215
Figure 9-20 Control circuit in external clock mode 2.....	215
Figure 9-21 Capture/compare channel (example: channel 1 input stage).....	216
Figure 9-22 Capture/compare channel 1 main circuit.....	217
Figure 9-23 Output part of channelx (x= 1,2,3,4 for TIM1; x= 1,2,3 for TIM8. Take channel 1 as example)	218
Figure 9-24 Output part of channelx (for TIM8, x= 4).....	218
Figure 9-25 PWM input mode timing.....	220
Figure 9-26 Output compare mode, toggle on OC1	222
Figure 9-27 Center-aligned PWM waveform (AR=8).....	223
Figure 9-28 Edge-aligned PWM waveform (APR=8)	224
Figure 9-29 Example of One-pulse mode.....	225
Figure 9-30 Clearing the OCxREF of TIMx.....	227
Figure 9-31 Complementary output with dead-time insertion.....	228
Figure 9-32 Output behavior in response to a break.....	230
Figure 9-33 Slide filter.....	231
Figure 9-34 Control circuit in reset mode.....	232
Figure 9-35 Control circuit in Trigger mode	233
Figure 9-36 Control circuit in Gated mode.....	233
Figure 9-37 Control circuit in Trigger Mode + External Clock Mode2.....	235
Figure 9-38 6-step PWM generation, COM example (OSSR=1)	236
Figure 9-39 Example of counter operation in encoder interface mode.....	237
Figure 9-40 Encoder interface mode example with IC1FP1 polarity inverted	238
Figure 9-41 Example of Hall sensor interface	240
Figure 10-1 Block diagram of TIM2	277

Figure 10-2 Counter timing diagram with prescaler division change from 1 to 4.....	281
Figure 10-3 Timing diagram of up-counting. The internal clock divider factor = 2/N.....	283
Figure 10-4 Timing diagram of the up-counting, update event when ARPEN=0/1.....	284
Figure 10-5 Timing diagram of the down-counting, internal clock divided factor = 2/N.....	285
Figure 10-6 Timing diagram of the Center-aligned, internal clock divided factor =2/N.....	286
Figure 10-7 A center-aligned sequence diagram that includes counter overflows and underflows (ARPEN = 1)	287
Figure 10-8 Control circuit in normal mode, internal clock divided by 1	288
Figure 10-9 TI2 external clock connection example	289
Figure 10-10 Control circuit in external clock mode 1	290
Figure 10-11 External trigger input block diagram	290
Figure 10-12 Control circuit in external clock mode 2	291
Figure 10-13 Capture/compare channel (example: channel 1 input stage).....	292
Figure 10-14 Capture/compare channel 1 main circuit.....	293
Figure 10-15 Output part of channelx (x=1/2/3/4. Take channel 1 as example)	294
Figure 10-16 Slide filter.....	295
Figure 10-17 PWM input mode timing.....	296
Figure 10-18 Output compare mode, toggle on OC1	298
Figure 10-19 Center-aligned PWM waveform (AR=8).....	299
Figure 10-20 Edge-aligned PWM waveform (APR=8).....	300
Figure 10-21 Example of One-pulse mode.....	301
Figure 10-22 Control circuit in reset mode.....	303
Figure 10-23 Control circuit in reset mode.....	304
Figure 10-24 Control circuit in Trigger mode.....	305
Figure 10-25 Control circuit in Gated mode.....	306
Figure 10-26 Control circuit in Trigger Mode + External Clock Mode2.....	307
Figure 10-27 Block diagram of timer interconnection	308
Figure 10-28 TIM2 gated by OC1REF of TIM1	309
Figure 10-29 TIM2 gated by enable signal of TIM1	310
Figure 10-30 Trigger TIM2 with an update of TIM1.....	310
Figure 10-31 Triggers timers 1 and 2 using the TI1 input of TIM1.....	311
Figure 10-32 Example of counter operation in encoder interface mode.....	313

Figure 10-33 Encoder interface mode example with IC1FP1 polarity inverted	313
Figure 11-1 Block diagram of TIMx (x = 6)	342
Figure 11-2 Counter timing diagram with prescaler division change from 1 to 4	343
Figure 11-3 Timing diagram of up-counting. The internal clock divider factor = 2/N	344
Figure 11-4 Timing diagram of the up-counting, update event when ARPEN=0/1.....	345
Figure 11-5 Control circuit in normal mode, internal clock divided by 1.....	346
Figure 12-1 LPTIM Diagram.....	352
Figure 12-2 Glitch filter timing diagram	354
Figure 12-3 LPTIM output waveform, Continuous counting mode configuration.....	355
Figure 12-4 PTIM output waveform, single counting mode configuration	356
Figure 12-5 LPTIM output waveform, Single counting mode configuration and Set-once mode activated ...	357
Figure 12-6 Waveform generation	358
Figure 12-7 Encoder mode counting sequence.....	361
Figure 12-8 Input waveforms of Input1 and Input2 when the decoder module is working normally	362
Figure 12-9 Input1 and Input2 input waveforms when decoder module is not working	362
Figure 13-1 Functional block diagram of the independent watchdog module.....	373
Figure 14-1 Watchdog block diagram.....	379
Figure 14-2 Refresh window and interrupt timing of WWDG	380
Figure 15-1 Block diagram of a single ADC	385
Figure 15-2 ADC clock.....	387
Figure 15-3 ADC channels and Pin connections	388
Figure 15-4 Timing diagram.....	390
Figure 15-5 Injection conversion delay	392
Figure 15-6 Calibration sequence diagram.....	393
Figure 15-7 Temperature sensor and VREFINT Diagram of the channel.....	397
Figure 16-1 Functional block diagram of I ² C	417
Figure 16-2 I ² C bus protocol.....	417
Figure 16-3 Transfer sequence diagram slave transmitter	420
Figure 16-4 Transfer sequence diagram of slave receiver	421
Figure 16-5 Master transmitter transmission sequence diagram	423
Figure 16-6 Master receiver transmission sequence diagram.....	425

Figure 17-1 USART block diagram	445
Figure 17-2 word length = 8 setting	447
Figure 17-3 word length = 9 setting	447
Figure 17-4 configuration stop bit	448
Figure 17-5 TXC/TXDE changes during transmission	450
Figure 17-6 Start bit detection	451
Figure 17-7 Transmission using DMA	458
Figure 17-8 Reception using DMA	459
Figure 17-9 hardware flow control between two USART	459
Figure 17-10 RTS flow control	460
Figure 17-11 CTS flow controls	461
Figure 17-12 Mute mode using idle line detection	462
Figure 17-13 Mute mode detected using address mark	463
Figure 17-14 USART synchronous transmission example	464
Figure 17-15 USART data clock timing example (WL=0)	465
Figure 17-16 USART data clock timing example (WL=1)	466
Figure 17-17 RX data sampling / holding time	466
Figure 17-18 IrDASIRENDEC-Block diagram	468
Figure 17-19 IrDA data Modulation (3/16)-normal mode	468
Figure 17-20 Break detection in LIN mode (11-bit break length-the LINBDL bit is set)	470
Figure 17-21 Break detection and framing error detection in LIN mode	471
Figure 17-22 ISO7816-3 Asynchronous Protocol	472
Figure 17-23 Use 1.5 stop bits to detect parity errors	473
Figure 18-1 SPI block diagram	487
Figure 18-2 Selective management of hardware/software	488
Figure 18-3 Master and slave applications	489
Figure 18-4 Data clock timing diagram	490
Figure 18-5 Schematic diagram of the change of TE/RNE/BUSY when the host is continuously transmitting in full duplex mode	491
Figure 18-6 Schematic diagram of TE/BUSY change when the host transmits continuously in one-way only mode	492
Figure 18-7 Schematic diagram of RNE change when continuous transmission occurs in receive-only mode	493

(BIDIRMODE=0 and RONLY=1)	492
Figure 18-8 Schematic diagram of the change of TE/RNE/BUSY when the slave is continuously transmitting in full duplex mode.....	494
Figure 18-9 Schematic diagram of TE/BUSY change during continuous transmission in slave unidirectional transmit-only mode.....	494
Figure 18-10 Schematic diagram of TE/BUSY change when BIDIRMODE = 0 and RONLY = 0 are transmitted discontinuously.....	496
Figure 18-11 Transmission using DMA	499
Figure 18-12 Reception using DMA	499
Figure 18-13 I ² S block diagram.....	502
Figure 18-14 I ² S Philips protocol waveform (16/32-bit full precision, CLKPOL = 0)	504
Figure 18-15 I ² S Philips protocol standard waveform (24-bit frame, CLKPOL = 0).....	504
Figure 18-16 I ² S Philips protocol standard waveform (16-bit extended to 32-bit packet frame, CLKPOL = 0)	505
Figure 18-17 The MSB is aligned with 16-bit or 32-bit full precision, CLKPOL = 0.....	506
Figure 18-18 MSB aligns 24-bit data, CLKPOL = 0.....	506
Figure 18-19 MSB-aligned 16-bit data is extended to 32-bit packet frame, CLKPOL = 0	507
Figure 18-20 LSB alignment 16-bit or 32-bit full precision, CLKPOL = 0	507
Figure 18-21 LSB aligns 24-bit data, CLKPOL = 0	508
Figure 18-22 LSB aligned 16-bit data is extended to 32-bit packet frame, CLKPOL = 0.....	508
Figure 18-23 PCM standard waveform (16 bits)	509
Figure 18-24 PCM standard waveform (16-bit extended to 32-bit packet frame).....	509
Figure 18-25 I ² S clock generator structure	510
Figure 18-26 Audio sampling frequency definition.....	510
Figure 19-1 RTC Block Diagram.....	526
Figure 21-1 N32G401F8S7-1 level and Cortex®-M4 level debugging block diagram.....	555

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

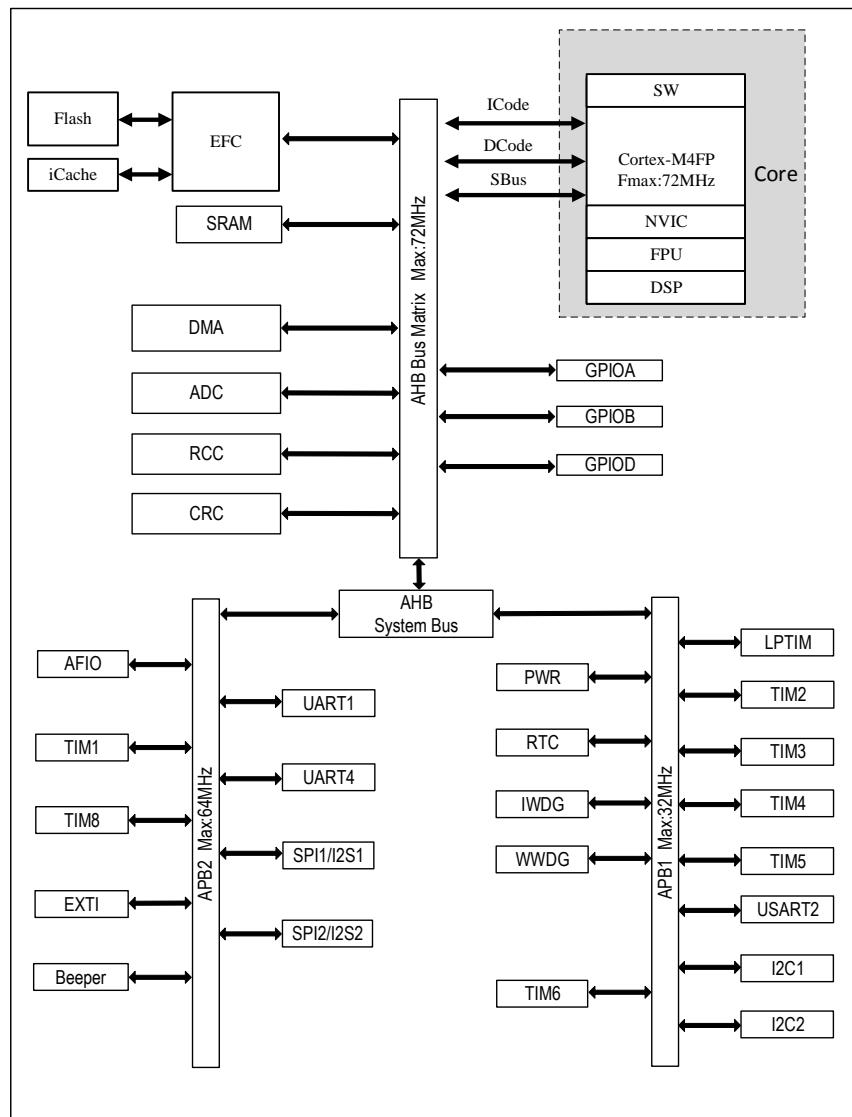
For all models of N32G401F8S7-1 microcontroller series, the existence and number of a peripheral, please refer to the data sheet of the corresponding model.

2 Memory and bus architecture

2.1 System architecture

2.1.1 Bus architecture

Figure 2-1 Bus architecture



- ICode bus: Connect the ICode bus of Cortex™-M4FP core with the flash instruction interface. Instruction prefetching is completed on this bus.
- The DCode bus connects the DCode bus of Cortex™-M4FP core with the data interface of flash memory (constant loading and debugging access).
- SBus bus connects the SBus bus (peripheral bus) of Cortex™-M4FP core to the bus matrix, which coordinates the access between the core and DMA.

- CRC doesn't have DMA hardware handshake, but it has designed matrix interconnection, which supports DMA transmission by software triggering.
- The system consists of two AHB2APB Bridges, i.e. AHB2APB1 and AHB2APB2. APB1 contains 14 APB peripherals and the maximum speed of PCLK1 is 32MHz. APB2 contains 9 APB peripherals with a maximum PCLK2 speed equal to 64MHz.

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. And the address space of SRAM is located in the bit-band Region of SRAM, and atomic accesses can be made through the bit-band Alias to performed read-modify-write operations on the target bits of the bit-band region. The address spaces of all APB and AHB peripherals are located in the bit-band Region of the peripherals. Atomic accesses can be made through the bit-band Alias to performed read-modify-write operations on the target bits of the bit-band region. The specific mapping is as follows:

Figure 2-2 Bus address map

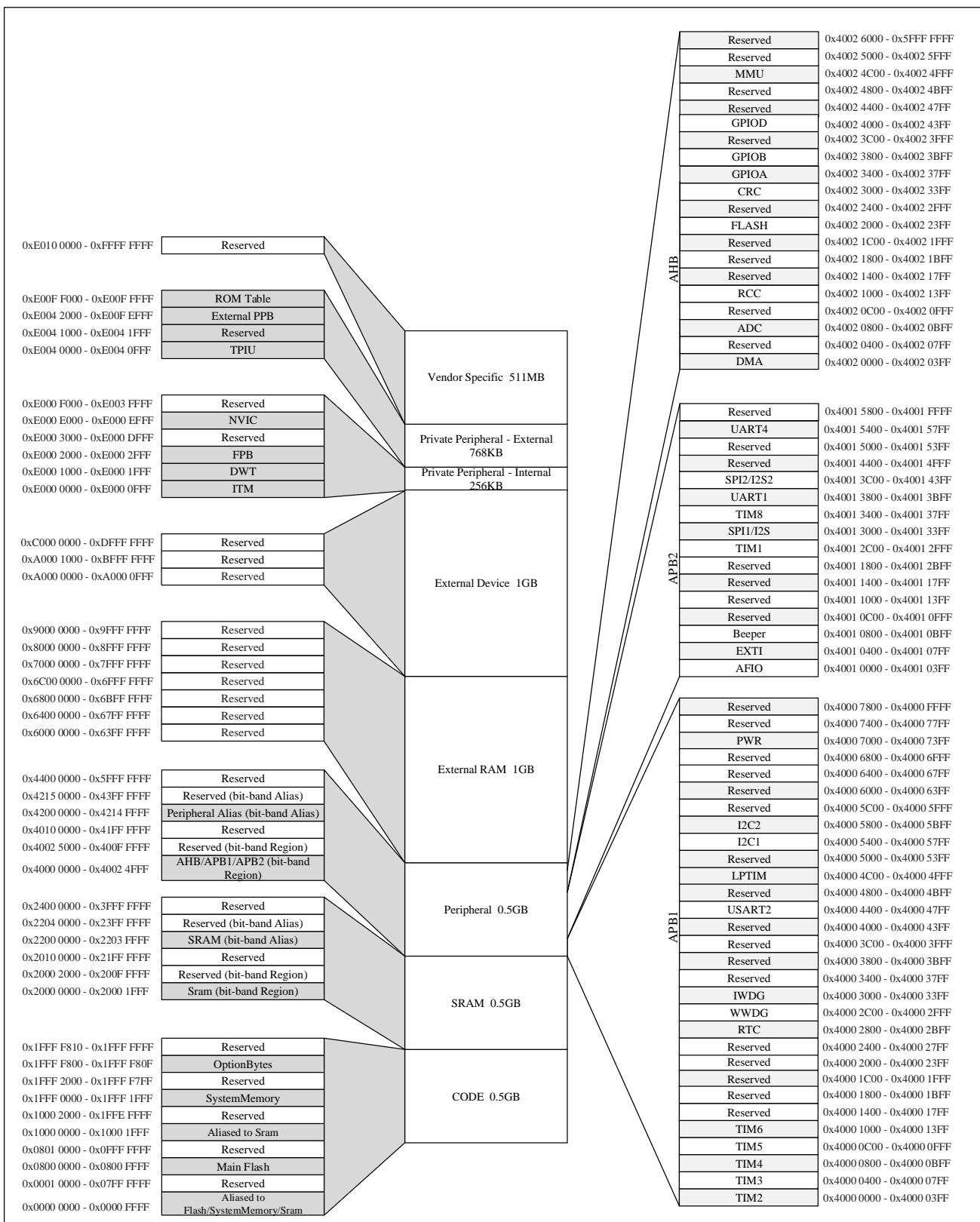


Table 2-1 List of peripheral register addresses

Address range	Peripherals	Bus
0x4002_6000 – 0x5FFF_FFFF	Reserved	AHB
0x4002_5000 – 0x4002_5FFF	Reserved	
0x4002_4C00 – 0x4002_4FFF	MMU	
0x4002_4800 – 0x4002_4BFF	Reserved	
0x4002_4400 – 0x4002_47FF	Reserved	
0x4002_4000 – 0x4002_43FF	GPIOD	
0x4002_3C00 – 0x4002_3FFF	Reserved	
0x4002_3800 – 0x4002_3BFF	GPIOB	
0x4002_3400 – 0x4002_37FF	GPIOA	
0x4002_3000 – 0x4002_33FF	CRC	
0x4002_2400 – 0x4002_2FFF	Reserved	
0x4002_2000 – 0x4002_23FF	FLASH	
0x4002_1C00 – 0x4002_1FFF	Reserved	
0x4002_1800 – 0x4002_1BFF	Reserved	
0x4002_1400 – 0x4002_1FFF	Reserved	
0x4002_1000 – 0x4002_13FF	RCC	
0x4002_0C00 – 0x4002_0FFF	Reserved	
0x4002_0800 – 0x4002_0BFF	ADC	
0x4002_0400 – 0x4002_07FF	Reserved	
0x4002_0000 – 0x4002_03FF	DMA	
0x4001_5800 – 0x4001_FFFF	Reserved	APB2
0x4001_5400 – 0x4001_57FF	UART4	
0x4001_5000 – 0x4001_53FF	Reserved	
0x4001_4400 – 0x4001_4FFF	Reserved	
0x4001_3C00 – 0x4001_43FF	SPI2/I2S2	
0x4001_3800 – 0x4001_3BFF	UART1	
0x4001_3400 – 0x4001_37FF	TIM8	
0x4001_3000 – 0x4001_33FF	SPI1/2S	
0x4001_2C00 – 0x4001_2FFF	TIM1	
0x4001_1800 – 0x4001_2BFF	Reserved	
0x4001_1400 – 0x4001_17FF	Reserved	
0x4001_1000 – 0x4001_13FF	Reserved	
0x4001_0C00 – 0x4001_0FFF	Reserved	
0x4001_0800 – 0x4001_0BFF	Beeper	
0x4001_0400 – 0x4001_07FF	EXTI	
0x4001_0000 – 0x4001_03FF	AFIO	
0x4000_7800 – 0x4000_FFFF	Reserved	APB1

Address range	Peripherals	Bus
0x4000_7400 – 0x4000_77FF	Reserved	
0x4000_7000 – 0x4000_73FF	PWR	
0x4000_6800 – 0x4000_6FFF	Reserved	
0x4000_6400 – 0x4000_67FF	Reserved	
0x4000_6000 – 0x4000_63FF	Reserved	
0x4000_5C00 – 0x4000_5FFF	Reserved	
0x4000_5800 – 0x4000_5BFF	I2C2	
0x4000_5400 – 0x4000_57FF	I2C1	
0x4000_5000 – 0x4000_53FF	Reserved	
0x4000_4C00 – 0x4000_4FFF	LPTIM	
0x4000_4800 – 0x4000_4BFF	Reserved	
0x4000_4400 – 0x4000_47FF	USART2	
0x4000_4000 – 0x4000_43FF	Reserved	
0x4000_3C00 – 0x4000_3FFF	Reserved	
0x4000_3800 – 0x4000_3BFF	Reserved	
0x4000_3400 – 0x4000_37FF	Reserved	
0x4000_3000 – 0x4000_33FF	IWDG	
0x4000_2C00 – 0x4000_2FFF	WWDG	
0x4000_2800 – 0x4000_2BFF	RTC	
0x4000_2400 – 0x4000_27FF	Reserved	
0x4000_2000 – 0x4000_23FF	Reserved	
0x4000_1C00 – 0x4000_1FFF	Reserved	
0x4000_1800 – 0x4000_1BFF	Reserved	
0x4000_1400 – 0x4000_17FF	Reserved	
0x4000_1000 – 0x4000_13FF	TIM6	
0x4000_0C00 – 0x4000_0FFF	TIM5	
0x4000_0800 – 0x4000_0BFF	TIM4	
0x4000_0400 – 0x4000_07FF	TIM3	
0x4000_0000 – 0x4000_03FF	TIM2	

2.1.2.1 Bit banding

Cortex™-M4FP memory image includes two bit-band areas. These two bit-band areas map each word in the alias memory area to a bit in the bit-band memory area. When writing a word in the alias area, it is equivalent to performing a read-modify-write operation on the target bits of the bit segment area.

Both the peripheral registers and SRAM are mapped into a bit-band area, which allows a single bit-band area write and read operation to be performed.

The following mapping formula shows how each byte in the alias area corresponds to the corresponding bit in the bit

band area:

`bitband_byte_addr = bitband_base + (byte_offset × 32) + (bit_number × 4)`

In which:

`bitband_byte_addr` is the address of the byte in the alias memory area, which is mapped to a certain target bit;

`bitband_base` is the starting address of alias area;

`byte_offset` is the serial number of the byte containing the target bit in the bit-band;

`bit_number` is the position of the target bit (0-7).

For example:

The following example shows how to map bit 4 in bytes with SRAM address 0x20000400 in alias area:

$0x22008010 = 0x22000000 + (0x400 \times 32) + (4 \times 4)$.

Writing to address 0x22008010 has the same effect as reading-modify-write to bit 4 of address 0x20000400 bytes in SRAM.

Reading 0x22008010 address returns the value of bit 4 (0x01 or 0x00) of address 0x20000400 bytes in SRAM. Please refer to “Cortex™-M4 Technical Reference Manual” for more information about bit-banding.

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 (PD0) pin and the user option byte BOOT configuration. After a system reset or exit from standby mode, the value of the BOOT pin will be re-latched and the option byte boot configuration (USER2) will be re-read. 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™-M4 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 Flash, the primary Flash memory:
 - ◆ 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 (ICode/DCode/DMA) ;
- 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 (ICode/DCode/DMA) ;
- 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

(ICode/DCode/SBus/DMA) ;

2.1.3.2 Boot configuration

In addition, SRAM can also be accessed through virtual address segment 0x1000_0000, which makes the CPU jump to SRAM to run programs through ICode/DCode after starting from Main Flash or System Memory (note that programs are not started from SRAM and do not belong to startup mode). In addition to the BOOT pin configuration boot program, there are two ways to run the program in SRAM:

- Jump directly to the physical address segment 0x2000_0000 of SRAM to run the program. At this time, the program will be run through SBus.
- Jump to the virtual address segment 0x1000_0000 of SRAM, and internally remap to the physical address segment 0x2000_0000 to run the program. At this time, the program will run efficiently through ICode/DCode.

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	0x1FFF_0000	0x2000 0000
X	1	X	0		0x0800_0000		0x1000 0000
1	X	1	1	System Memory Start	0x0000_0000	0x2000 0000	0x2000 0000
1	0	X	0		0x08000000	0x1FFF_0000	0x1000 0000
0	X	1	1	SRAM start			0x0000_0000
0	0	X	0		0x08000000	0x1FFF_0000	0x2000 0000
							0x1000 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 UART1. The UART1 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 storage area and an information area, which are described separately below:

- The maximum main memory area is 64KB, also known as main flash memory, which contains 32 Page for

storing and running user programs and storing data.

- The information area is 12KB, including 6 Page, and consists of system storage area (8KB), system configuration area (2KB) and option byte area (2KB).
 - ◆ The System Memory area is 8KB, which contains 4 Page, also known as System Memory, and is used to store and run the BOOT program.
 - ◆ The system configuration area is 2KB, including 1 Page.
 - ◆ The Option Byte area is 2KB, containing 1 Page, also known as Option Byte, and the effective space is 16B, 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
Main memory area	Page 0	0x0800_0000 – 0x0800_07FF	2KB
	Page 1	0x0800_0800 – 0x0800_0FFF	2KB
	Page 2	0x0800_1000 – 0x0800_17FF	2KB
	:	:	:
	Page 31	0x0800_F800 – 0x0800_FFFF	2KB
Information area	System memory area	0x1FFF_0000 – 0x1FFF_1FFF	8KB
	System configuration area	0x1FFF_F000 – 0x1FFF_F7FF	2KB
	Option byte area	0x1FFF_F800 – 0x1FFF_F80F	16B
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
	FLASH_OB2	0x4002_2018 – 0x4002_201B	4B
	FLASH_OB	0x4002_201C – 0x4002_201F	4B
	FLASH_WRP	0x4002_2020 – 0x4002_2023	4B
	Reserved	0x4002_2024 – 0x4002_202F	12B
	FLASH_CAH	0x4002_2030 – 0x4002_2033	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 UART1 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

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 AHB interface, and the waiting time must not be less than 25ns. For example, when $HCLK \leq 32MHz$, the minimum number of waiting periods is 0; When $32MHz < HCLK \leq 64MHz$, the minimum number of waiting periods is 1; When $64MHz < HCLK$, 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 8 bytes (2 bytes for write protection, 2 bytes for read protection, 2 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 Instruction prefetching

The instruction prefetch function of Flash module supports the prefetch Buffer of 16B. 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.

2.2.1.6 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 standby/stop0/stop2 mode, and bus address space is allocated for read-write access. They consist of byte with 8 options: 2 byte for write protection, 2 bytes for read protection, 2 byte for configuration option, 2 bytes defined by user, These 8 bytes need to be written through the bus. The option byte block also contains the complement codes corresponding to these 8 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
0x1FFF_F800	nUSER	USER	nRDP1	RDP1
0x1FFF_F804	nData1	Data1	nData0	Data0
0x1FFF_F808	nWRP1	WRP1	nWRP0	WRP0
0x1FFF_F80C	nUSER2	USER2	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
 - ◆ USER[7]: Reserved
 - ◆ USER[6]: IWDGSLEEPFRZ, read through FLASH_OB[8]
 - 0: iwdg freeze in sleep
 - 1: default no freeze
 - ◆ USER[5]: IWDGSTDBYFRZ, read through FLASH_OB[7]
 - 0: iwdg freeze in standby
 - 1: default no freeze
 - ◆ USER[4]: IWDGSTOP2FRZ, read through FLASH_OB[6]
 - 0: iwdg freeze in stop2
 - 1: default no freeze
 - ◆ USER[3]: IWDGSTOP0FRZ, read through FLASH_OB[5]
 - 0: iwdg freeze in stop0
 - 1: default no freeze
 - ◆ USER[2]: nRST_STDBY configuration options, read through FLASH_OB [4]
 - 0: Reset when entering standby mode
 - 1: No reset occurs when entering standby mode
 - ◆ USER[1]: nRST_STOP, read through FLASH_OB[3]
 - 0: Reset occurs when entering stop0/stop2 mode
 - 1: No reset occurs when entering stop0/stop2 mode
 - ◆ USER[0]: WDG_SW configuration options, read through 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 of pages 0-15, bit [0] corresponds to Page0 / 1,.., bit [7] corresponds to page14 / 15;
 - ◆ WRP1: write protection on pages 16-31, bit [0] corresponds to Page16 / 17,.., bit [7] corresponds to Page30 / 31;
- Read protection L2 level option byte: RDP2
 - ◆ Add protection function on the basis of L1, see 2.2.1.8 detailed description of read protection;
 - ◆ Whether RDP2 is turned on or not can be determined by FLASH_OB [31] query;
- **User Configuration 2: USER2**
 - ◆ USER2 [7:3]: Reserved
 - ◆ USER2[2] : nSWBOOT0, read out through FLASH_OB2[26], default is 1
 - ◆ USER2[1]: nBOOT1, read out through FLASH_OB2[23], default is 1
 - ◆ USER2[0]: nBOOT0, read out through FLASH_OB2[27], default is 1

2.2.1.7 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 ~ 31, every 2 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 (8KB) in the system information area stores the boot program and cannot be changed.

The system configuration block (2KB) in the system information area stores the basic information of the chip and cannot be changed.

The option byte block (2KB) 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.

2.2.1.8 Read protection

The user code in flash can be protected from illegal reading by setting read protection. Read protection is mainly aimed at protecting the access operation of main memory area and option byte block after chip sealing operation. 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 pages can be programmed by the code executed in the main flash memory (realizing IAP or data storage and other functions);
- ◆ 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 SWD and then execute it are still valid, or they can be started from the built-in SRAM through 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 to unlock the option byte area in FLASH_OPTKEY;
 - 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;

- ◆ The following access operations to the flash memory will be prohibited:
 - Access the main flash memory by executing code from the built-in SRAM start(including using DMA);
 - Access the main flash memory by SWV (serial line observer), SWD (serial line debugging) and boundary scanning;
- L2 level: Except that SRAM boot disable, 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.

Table 2-6 Flash read-write-erase⁽¹⁾ permission control table

protect grade	Boot mode	Main Flash				Modify the protection level
	Executing user Access area	SWD	Main Flash	System Memory	SRAM	
L0 level	Flash main storage area	Read-Write- Erase	Read-Write- Erase	Read-Write-Erase	Read-Write- Erase	Allow to change to L1 or L2
	The Flash 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 storage area	prohibit	prohibit	Read-Write-Erase	prohibit	
	System configuration area	Only id is readable.	Id only readable	Id readable, Partition information readable	Id only readable	
	SRAM (All)	read and write	read and write	read and write	read and write	
L1 level	Flash main storage area	prohibit	Read-Write- Erase	Read-Write-Erase	Read-Write- Erase	Allow L0 or L2 instead.
	Flash memory mass erase	allow	allow	allow	allow	When it is changed to L0, the main storage

	Flash option byte area	Read-Write-Erase	Read-Write-Erase	Read-Write-Erase	Read-Write-Erase	area will be automatically erased.
	Flash system storage area	prohibit	prohibit	Read-Write-Erase	prohibit	
	System configuration area	prohibit	Id only readable	Id readable, Partition information readable	Id only readable	
	SRAM (All)	read and write	read and write	read and write	read and write	
L2 level	Flash main storage area	SWD Interface is disabled.	Read-Write-Erase	Read-Write-Erase	Read-Write-Erase	Modifications are not allowed.
	Flash memory mass erase		allow	allow	allow	
	Flash option byte area		read only	read only	read only	
	Flash system storage area		prohibit	Read-Write-Erase	prohibit	
	System configuration area		Id only readable	Id readable, Partition information readable	Id only readable	
	SRAM (All)		read and write	read and write	read and write	
protect grade	Boot mode	SRAM				Modify the protection level
	Executing user Access area	SWD	Main Flash	System Memory	SRAM	
L0 level	Flash main storage area	Read-Write-Erase	Read-Write-Erase	Read-Write-Erase	Read-Write-Erase	Allow to change to L1 or L2

	Flash memory mass erase	allow	allow	allow	allow	
	Flash option byte area	Read-Write- Erase	Read-Write- Erase	Read-Write-Erase	Read-Write- Erase	
	Flash system storage area	prohibit	prohibit	Read-Write-Erase	prohibit	
	System configuration area	Id only readable	Id only readable	Id readable, Partition information readable	Id only readable	
	SRAM (All)	read and write	read and write	read and write	read and write	
L1 level	Flash main storage area	prohibit	Read-Write- Erase	Read-Write-Erase	prohibit	Allow L0 or L2 instead. When it is changed to L0, the main storage area will be automatically erased.
	Flash memory mass erase	allow	allow	allow	allow	
	Flash option byte area	Read-Write- Erase	Read-Write- Erase	Read-Write-Erase	Read-Write- Erase	
	Flash system storage area	prohibit	prohibit	prohibit	prohibit	
	System configuration area	prohibit	Id only readable	Id readable Partition information readable	prohibit	
	SRAM (All)	read and write	read and write	read and write	read and write	
L2 level	Flash main storage area	L2 protection level, unable to boot from SRAM				Modifications are not allowed. SWD is prohibited.
	Flash memory mass erase					

	Flash option byte area					
	Flash system storage area					
	System configuration area					
	SRAM (All)					
protect grade	Boot mode	System Memory				Modify the protection level
	Executing user Access area	SWD	Main Flash	System Memory	SRAM	
L0 level	Flash main storage area	Read-Write-Erase	Read-Write-Erase	Read-Write-Erase	Read-Write-Erase	Allow to change to L1 or L2
	Flash memory mass erase	allow	allow	allow	allow	
	Flash option byte area	Read-Write-Erase	Read-Write-Erase	Read-Write-Erase	Read-Write-Erase	
	Flash system storage area	prohibit	prohibit	Read-Write-Erase	prohibit	
	System configuration area	Id only readable	Id only readable	Id readable, Partition information can be read and written	Id only readable	
	SRAM (All)	read and write	read and write	read and write	read and write	
L1 level	Flash main storage area	prohibit	Read-Write-Erase	Read-Write-Erase	Read-Write-Erase	Allow changed to L0 or L2 instead.
	Flash memory mass erase	prohibit	allow	allow	allow	When it is changed to L0, the main storage area will be automatically erased.
	Flash option byte area	Read-Write-	Read-Write-	Read-Write-Erase	Read-Write-	

		Erase	Erase		Erase	
	Flash system storage area	prohibit	prohibit	Read-Write-Erase	prohibit	
	System configuration area	prohibit	Id only readable	Id readable, Partition information can be read and written.	Id only readable	
	SRAM (All)	read and write	read and write	read and write	read and write	
L2 level	Flash main storage area	SWD Interface is disabled.	Read-Write-Erase	Read-Write-Erase	Read-Write-Erase	Modification is not allowed.
	Flash memory mass erase		allow	allow	allow	
	Flash option byte area		read only	read only	read only	
	Flash system storage area		prohibit	Read-Write-Erase	prohibit	
	System configuration area		Id only readable	Id readable, Partition information can be read and written.	Id only readable	
	SRAM (All)		read and write	read and write	read and write	

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

2.2.2 iCache

In order to achieve higher system performance, an instruction buffer needs to be added between the high-speed CPU and the low-speed Flash to improve the instruction execution efficiency. Because of the existence of the instruction buffer, the CPU will be able to work at a higher frequency. When the instruction requested by the CPU is in the instruction buffer, the CPU can obtain the instruction without delay and realize zero waiting for execution. When the current instruction sequence, instruction prefetch sequence and instruction buffer all miss, Flash will be re-read and the Cache will be backfilled and updated. Accordingly, it is equivalent to storing only the jump header of the program in the Cache.

The main features of the instruction buffer are as follows:

- 1KB iCache.
- Support connection mode: 4WAY.

2.2.2.1 Software interface

- Enable
 - ◆ Provide configuration for software to enable/disable iCache. There is no limit to the switching conditions (see the FLASH_AC.ICAHEN bit).
- Reset
 - ◆ Provide software to clear the iCache interface, which must be initiated when iCache is closed. Reset and switching cannot be switched at the same time. First, turn off FLASH_AC.ICAHEN, then write 1 to FLASH_AC.ICAHRST, and then turn on FLASH_AC.ICAHEN.
- Lock
 - ◆ Cache locking mechanism is supported, and the software configuration puts the program into its designated way. When all the ways are locked, the new data will not be written into the cache. After the software resets the cache, the lock state is automatically cleared.
- Additional remarks
 - ◆ Selection of Cache replacement algorithm is not supported.
 - ◆ When using iCache, there is no WB/WT selection when CPU writes operation.

2.2.2.2 Register description

FLASH_AC.ICAHEN and FLASH_AC.ICAHRST are the iCache enable switch and iCache data clear switch respectively.

FLASH_CAHR.LOCKSTR and FLASH_CAHR.LOCKSTOP are the start latch and stop latch of iCache corresponding mode lock, respectively. After iCache is reset, the FLASH_CAHR register automatically returns to the reset value. See for detailed usage method of 2.2.2.3.3 iCache locking.

2.2.2.3 Operating process

2.2.2.3.1 iCache enable and disable

Users can turn on and switch off iCache at any time. If the user program needs to jump between the main memory area and other memory areas, the iCache must be closed and the data of the iCache must be cleared, otherwise, the instruction acquisition error will occur.

2.2.2.3.2 iCache data refresh

The iCache is designed as instruction cache. When the instruction is updated by application software or the instruction jumps between the main memory area and other memory areas, the software must set the FLASH_AC.ICAHRST bit to 1 to clear the data in the instruction cache.

Note: FLASH_AC.ICAHRST bit is a write-only bit, and it returns to 0 when read.

2.2.2.3.3 iCache locking

The software controls the FLASH_CAHR register to lock some repeatedly used codes in iCache to improve the

efficiency of code execution. iCache module has four latch channels, and the size of each channel is 1/4 of the whole cache. When using a single channel, you must ensure that the amount of code to latch is less than the size of each channel. Otherwise need to use more channels to latch the code. The latch function can be used according to the following control flow:

1. Set FLASH_CAH.R.LOCKSTRT[0] to 1;
2. Execute function 1 that needs to be locked in channel 0 (the code amount of function 1 should be less than the size of a single channel);
3. Set FLASH_CAH.R.LOCKSTOP[0] to 1 after the function 1 is executed;
4. Then set FLASH_CAH.R.LOCKSTRT[1] to 1;
5. Execute function 2 that needs to be locked in channel 1 (the code amount of function 2 should be less than the size of a single channel);
6. After the function 2 is executed, set FLASH_CAH.R.LOCKSTOP[1] to 1;

Attention: 1. when the channel is latched, the register operation must follow a fixed process -First set FLASH_CAH.R.LOCKSTRT then set FLASH_CAH.R.LOCKSTOP;

2. The order of channel latch must be 0~3, otherwise it will reduce the execution efficiency.

2.2.3 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 (supports access of SBus, ICode and DCode), and can run programs at full speed in SRAM. The maximum address range of SRAM is 0x2000 0000~0x2000 1FFF.

In Standby modes, SRAM data optional retention; other working mode (Run/Sleep/Stop0/Stop2) data retention normally.

The main features are as follows:

- The maximum capacity is 8KB in total.
- Support byte/half-word/word reading and writing
- I/D/S/DMA can be accessed.
- I/D BUS can run programs at full speed from Remap to SRAM.

2.2.4 FLASH register description

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

2.2.4.1 FLASH register overview

Table 2-7 FLASH register overview

2.2.4.2 FLASH control and status register

See for abbreviations in register descriptions 1.1 section.

2.2.4.2.1 The FLASH access control register (FLASH_AC)

Address offset: 0x00

Reset value: 0x0000 0030

31	Reserved	16
15	Reserved	0
8	ICAHEN	rw
7	ICAHRST	w
6	PRFTBFS	r
5	PRFTBFE	rw
4	Reserved	rw
3	LATENCY	rw
2:0		

Bit field	name	describe
31:8	Reserved	Reserved, the reset value must be maintained.
7	ICAHEN	iCache enable 0: turn off iCache; 1: enable iCache.
6	ICAHRST	iCache reset 0: writing '0' is invalid; 1: write '1' to reset.
5	PRFTBFS	Prefetch buffer status This bit indicates the status of the prefetch buffer 0: The prefetch buffer is closed; 1: The prefetch buffer is open.
4	PRFTBFE	Prefetch buffer enable 0: Close the prefetch buffer; 1: Enable prefetch buffer.
3	Reserved	Reserved, the reset value must be maintained.
2:0	LATENCY	time delay These bits represent the ratio of HCLK period to flash memory access time. 000: zero period delay, when $0 < \text{HCLK} \leq 32\text{MHz}$ 001: one cycle delay, when $32\text{MHz} < \text{HCLK} \leq 64\text{MHz}$ 010: two cycle delay, when $64\text{MHz} < \text{HCLK} \leq 72\text{MHz}$ Other values: reserved

2.2.4.2.2 The FLASH key register (FLASH_KEY)

Address offset: 0x04

Reset value: 0x0000 0000

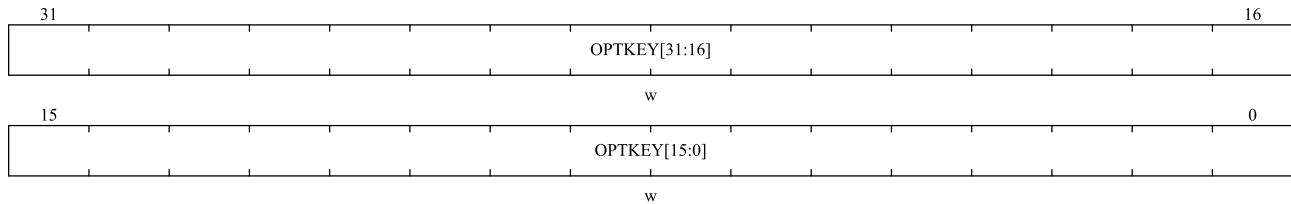
31	FKEY[31:16]	16
15	w	0
15	FKEY[15:0]	0
15	w	0

Bit field	name	describe
31:0	FKEY	Used to unlock the FLASH_CTRL.LOCK bit.

2.2.4.2.3 The FLASH_OPTKEY register (FLASH_OPTKEY)

Address offset: 0x08

Reset value: 0x0000 0000

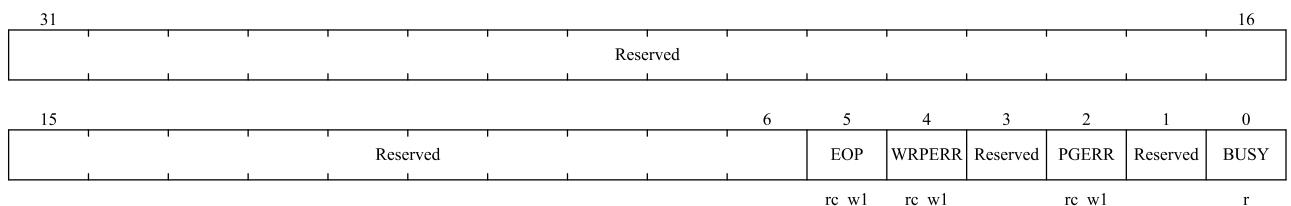


Bit field	name	describe
31:0	OPTKEY	Used to unlock the FLASH_CTRL.OPTWE bit.

2.2.4.2.4 The FLASH status register (FLASH_STS)

Address offset: 0x0c

Reset value: 0x0000 0000

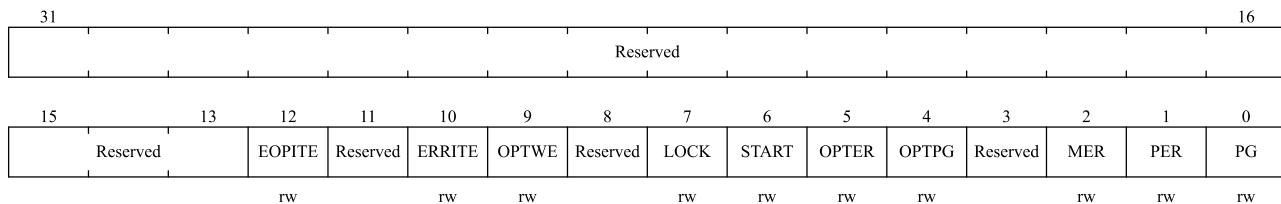


Bit field	name	describe
31:6	Reserved	Reserved, the reset value must be maintained.
5	EOP	End of operation When the flash operation (programming/erasing) is completed, the hardware sets this bit to '1', and writing '1' can clear this bit status. <i>Note: Every successful programming or erasing will set the EOP state.</i>
4	WRPERR	Write protection error When trying to program a write-protected flash address, the hardware sets this bit to '1', and writing '1' can clear this bit.
3	Reserved	Reserved, the reset value must be maintained.
2	PGERR	Programming error When trying to program an address whose content is not '0xFFFF_FFFF', the hardware sets this bit to '1', and writing '1' can clear this state. <i>Note: Before programming, the FLASH_CTRL.START bit must be cleared.</i>
1	Reserved	Reserved, the reset value must be maintained.
0	BUSY	Busy This bit indicates that a flash operation is in progress. At the beginning of flash operation, this bit is set to '1'; This bit is cleared to '0' when the operation ends or an error occurs.

2.2.4.2.5 The FLASH control register (FLASH_CTRL)

Address offset: 0x10

Reset value: 0x0000 0080



Bit field	name	describe
31:13	Reserved	Reserved, the reset value must be maintained.
12	EOPITE	Allow operation completion interrupt. This bit allows an interrupt to be generated when the FLASH_STS.EOP bit becomes '1'. 0: interrupt generation is prohibited; 1: interrupt generation is allowed.
11	Reserved	Reserved, the reset value must be maintained.
10	ERRITE	Error status interrupt allowed This bit allows an interrupt to be generated when a Flash error occurs (when FLASH_STS.PGERR/ FLASH_STS.WRPERR is set to '1'). 0: interrupt generation is prohibited; 1: interrupt generation is allowed.
9	OPTWE	Allow write option byte When this bit is '1', the option byte is allowed to be programmed. When the correct key sequence is written in the FLASH_OPTKEY register, this bit is set to '1'. Software can clear this bit.
8	Reserved	Reserved, the reset value must be maintained.
7	LOCK	Lock You can only write '1'. When this bit is '1', Flash and FLASH_CTRL are locked. After detecting the correct unlocking sequence, hardware clears this bit to '0'. After an unsuccessful unlocking operation, this bit cannot be changed until the next system reset.
6	START	Start When this bit is '1', an erase operation will be triggered. This bit can only be set to '1' by software and cleared to '0' when FLASH_STS.BUSY becomes '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;

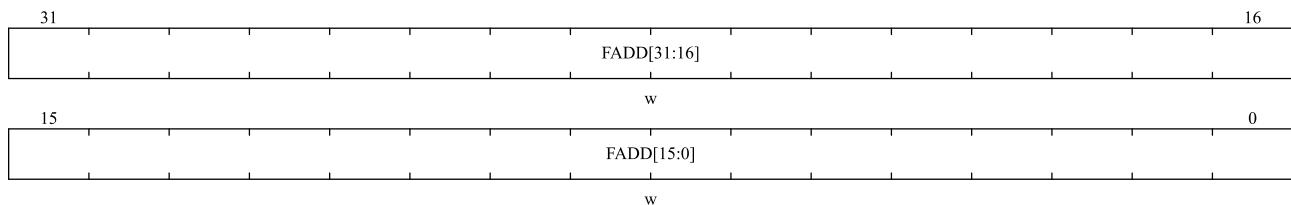
Bit field	name	describe
		1: Enable option bytes program mode.
3	Reserved	Reserved, the reset value must be maintained.
2	MER	Mass erase. 0: disable mass erase mode; 1: enable mass erase mode.
1	PER	Page erase. 0: disable page erase mode; 1: enable page erase mode
0	PG	Program. 0: disable program mode; 1: enable program mode.

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

2.2.4.2.6 The FLASH address register (FLASH_ADD)

Address offset: 0x14

Reset value: 0x0000 0000

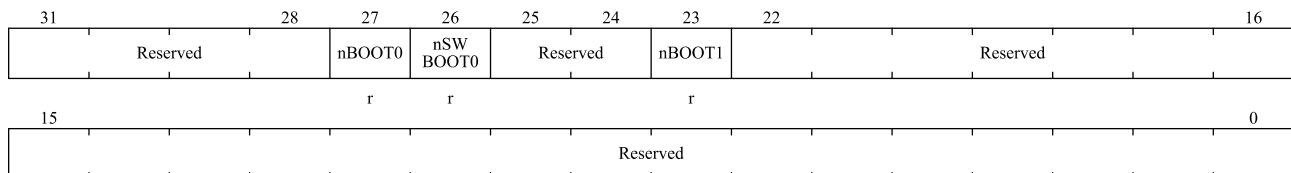


Bit field	name	describe
31:0	FADD	Flash address Select the address to be programmed when programming, and select the page to be erased when page erasing. <i>Note: When the FLASH_STS.BUSY bit is '1', this register cannot be written.</i>

2.2.4.2.7 The FLASH Option byte register 2 (FLASH_OB2)

Address offset: 0x18

Reset value: 0x0c800000



Bit field	name	describe
31:28	Reserved	Reserved, the reset value must be maintained.
27	nBOOT0	nBOOT0

Bit field	name	describe
		<i>Note: This bit is read-only.</i>
26	nSWBOOT0	nSWBOOT0 <i>Note: This bit is read-only.</i>
25:24	Reserved	Reserved, the reset value must be maintained.
23	nBOOT1	nBOOT1 <i>Note: This bit is read-only.</i>
22:0	Reserved	Reserved, the reset value must be maintained.

Note: For the specific combined functions of nBOOT0, nSWBOOT0, and nBOOT1, see chapter 2.1.3 boot Management.

2.2.4.2.8 Option byte register (FLASH_OB)

Address offset: 0x1C

Reset value: 0x03FF FFFF

31	30	26	25	18	17	16					
RDPRT2	Reserved		Data1		Data0						
r 15		10	9	8	7	6					
		r	r	r	r	r					
	Data0	Reserved	IWDGSL EEPFRZ	IWDGST DBYFRZ	IWDGST OP2FRZ	IWDGST OP0FRZ	nRST STDBY	nRST STOP	WDG_SW	RDPRT1	OBERR
			r	r	r	r	r	r	r	r	r

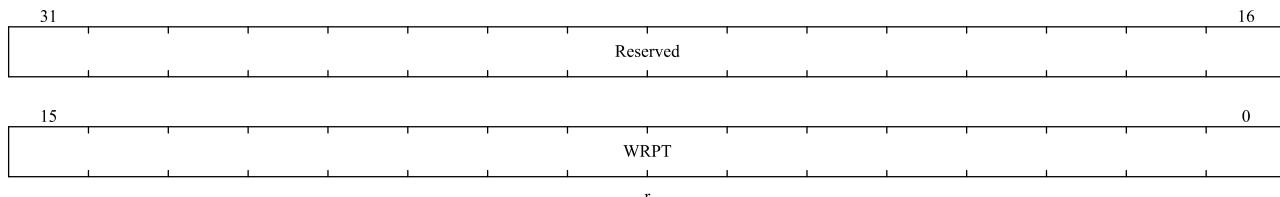
Bit field	name	describe
31	RDPRT2	Read protection L2 level protection 0: Read protection L2 level is not enabled; 1: Read protection L2 level is enabled. <i>Note: This bit is read-only.</i>
30:26	Reserved	Reserved, the reset value must be maintained.
25:18	Data1[7:0]	Data1 <i>Note: This bit is read-only.</i>
17:10	Data0[7:0]	Data0 <i>Note: This bit is read-only.</i>
9	Reserved	Not used (if these bits are written in the Flash option byte, they will be read in this register with no effect on the device.)
8	IWDGSLEEPFRZ	IWDG to freeze in sleep mode 0: iwdg freeze in sleep 1: default no freeze <i>Note: This bit is read-only.</i>
7	IWDGSTDBYFRZ	IWDG to freeze in standby mode 0: iwdg freeze in standby 1: default no freeze <i>Note: This bit is read-only.</i>
6	IWDGSTOP2FRZ	IWDG to freeze in stop2 mode

Bit field	name	describe
		0: iwdg freeze in stop2 1: default no freeze <i>Note: This bit is read-only.</i>
5	IWDGSTOP0FRZ	Set IWDG to freeze in stop mode 0: iwdg freeze in stop0 1: default no freeze <i>Note: This bit is read-only.</i>
4	nRST_STDBY	Enter Standby mode reset configuration. 0: Reset immediately after entering Standby mode; 1: No reset occurs after entering Standby mode. <i>Note: This bit is read-only.</i>
3	nRST_STOP	Enter STOP0/STOP2 mode reset configuration. 0: Reset occurs immediately after entering STOP0/STOP2 mode; 1: No reset occurs after entering the STOP0/STOP2 mode. <i>Note: This bit is read-only.</i>
2	WDG_SW	Set watchdog 0: hardware watchdog; 1: Software watchdog. <i>Note: This bit is read-only.</i>
1	RDPRT1	Read protection L1 level protection 0: Read protection L1 level is not enabled; 1: read protection L1 level is enabled. <i>Note: This bit is read-only.</i>
0	OBERR	Option byte error When this bit is '1', it means that the option byte does not match its complement. <i>Note: This bit is read-only.</i>

2.2.4.2.9 Write protection register (FLASH_WRP)

Address offset: 0x20

Reset value: 0x0000 FFFF



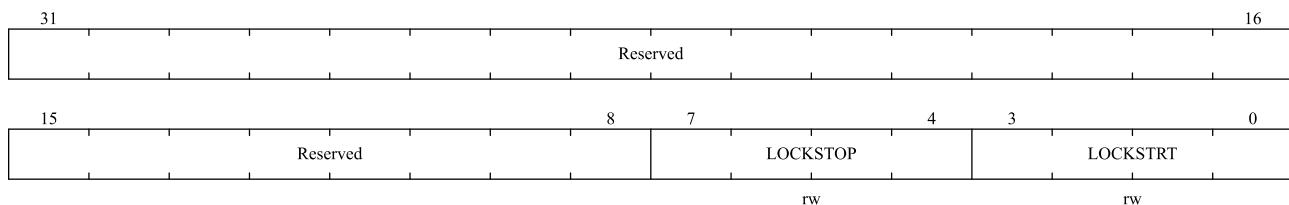
Bit field	name	describe
31:16	Reserved	Reserved, the reset value must be maintained.
15:0	WRPT	Write protect This register contains the write protection option byte loaded by option byte area. 0: write protection takes effect;

Bit field	name	describe
		<p>1: Write protection is invalid.</p> <p><i>Note: These bits are read-only.</i></p>

2.2.4.2.10 CAHR register (FLASH_CAHR)

Address offset: 0x30

Reset value: 0x0000 0000



Bit field	name	describe
31:8	Reserved	Reserved, the reset value must be maintained.
7:4	LOCKSTOP[3:0]	iCache lock stop (see for detailed operation instructions 2.2.2.3.3 iCache locking Chapter). 0: disable 1: enable
3:0	LOCKSTRT[3:0]	iCache lock start. 0: disable 1: enable

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, SLEEP, STOP0, STOP2, and STANDBY mode. PWR controls LDOs, Clock sources, Resets and Flash/SRAMs/GPIO status in different power modes.

3.1.1 Power supply

MCU requires 2.4V to 3.6V operating voltage supply (VDD). VDD are external power supplies.

- ✧ Independent A/D converter power supply. To improve conversion accuracy, the ADC uses a separate power supply to filter and shield interference from burrs on the printed circuit board.
 - The ADC power pin is VDDA.
 - VSSA and VSS are connect in chip.
 - There are no VREF+ and VREF- pins, which are connected to the ADC power supply (VDDA) and ground (VSSA) inside the chip.
- ✧ Voltage regulator
 - According to the power management requirements, the power regulator has MR (The main regulator) and BKR (The backup regulator).
 - The voltage regulator (MR, BKR) is always enabled after reset.
 - The voltage regulator operates in several different modes, depending on the application:
 - RUN mode: The voltage regulator provides V_{DDD} power (core, memory and peripherals) in normal power mode.
 - STOP0 mode: The voltage regulator provides V_{DDDRET} power in low power mode to hold the contents of registers and SRAM.
 - STOP2 mode: Turn off MR and switch to BKR power supply to save register and SRAM contents.

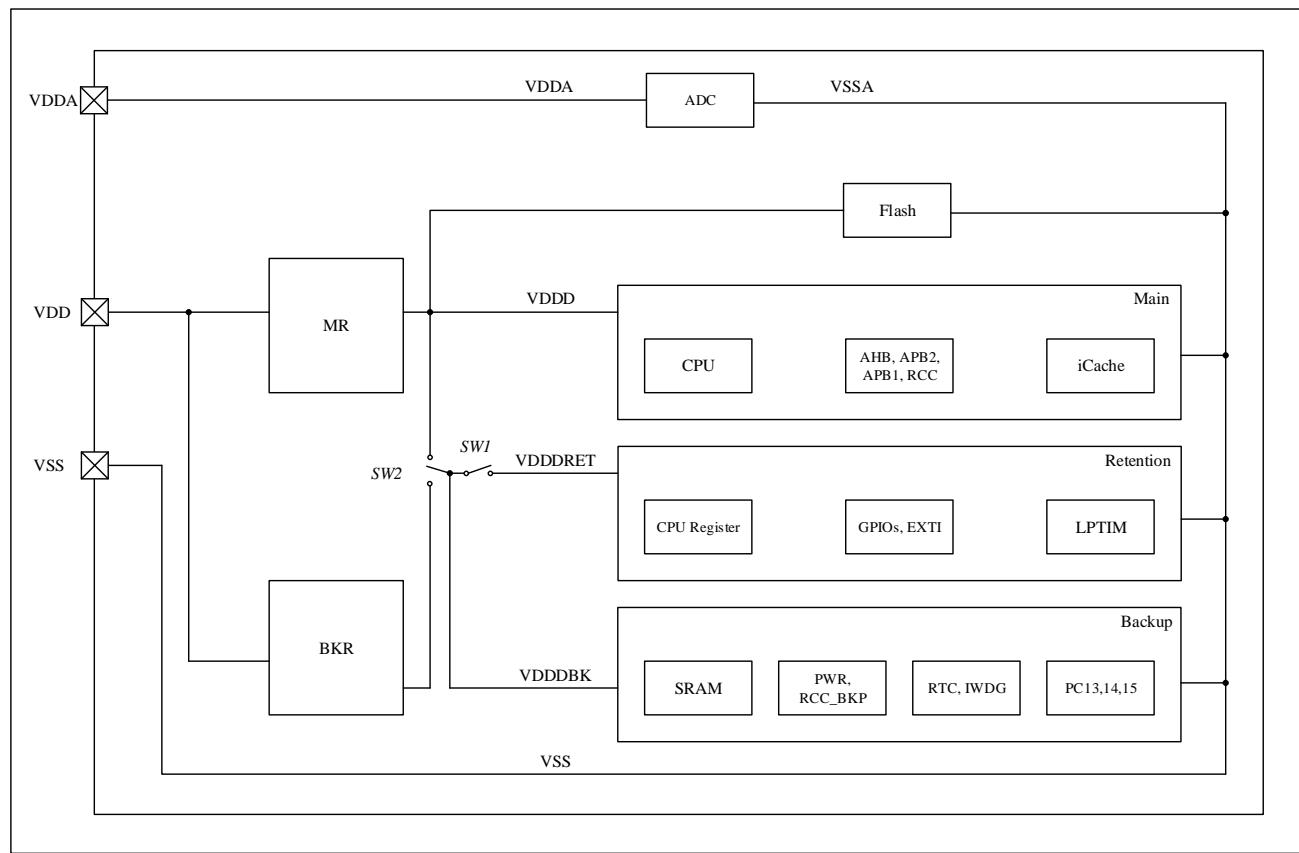
- STANDBY mode: The MR regulator stops power supply and the BKR can be turned on or off as required.
- MR

It is Main Regulator for VDDD supply source. It is mainly used in MCU's RUN mode, SLEEP mode and STOP0 mode. In STOP0 mode, MR in low-power mode is optional, and in normal mode in the other two modes.

- BKR

This Backup Regulator is used in STOP2 and STANDBY modes.

Figure 3-1 Power supply block diagram



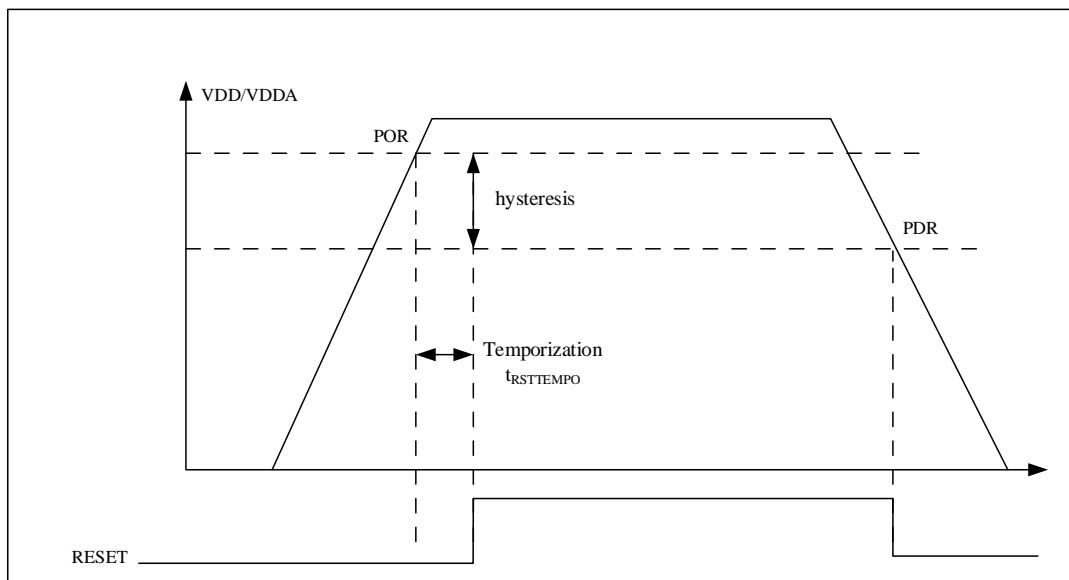
Note: VSS and VSSA are connect in chip.

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 V_{POR}/V_{PDR} , 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

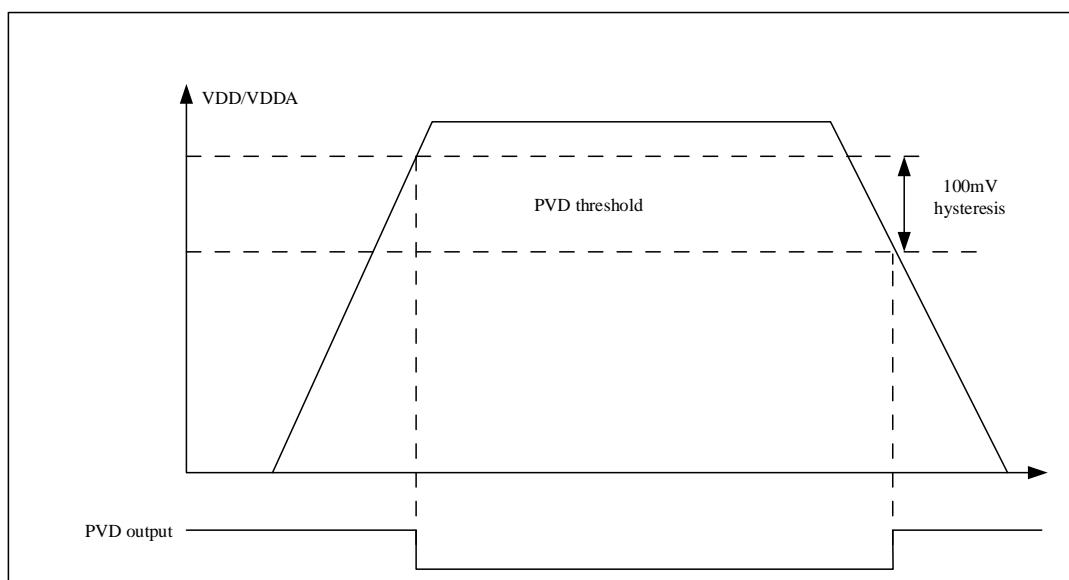


3.1.2.2 Programmable voltage detector (PVD)

PVD monitors the power supply by comparing the VDD voltage with the relevant bits in the power control register (PWR_CTRL). The PWR_CTRL.MSB and PWR_CTRL.PLS select the threshold of the monitoring voltage. Enable PVD by setting the PWR_CTRL.PVDE.

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 the 16th line of the external interrupt 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 the external interrupt 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. The whole circuit generating System reset is in VDDDBK/VDDA domain.

3.2 Power modes

Overall MCU has five power modes: RUN, SLEEP, STOP0, STOP2 and STANDBY. Different mode has different performance and power consumption. A summary of MCU power modes is shown below.

Table 3-1 Power modes

ITEMs MODES	CONDITION	ENTRY	EXIT
RUN	CPU running, peripherals configurable.	Power up, system reset, or wakeup from other power modes.	Enter SLEEP, STOP0, STOP2, STANDBY.
SLEEP	CPU Sleep, All peripherals configurable, Regulator ON, All digital peripherals powered. Interrupts & Events can wakeup CPU.	WFI CPU returns from ISR WFE	Any interrupts wakeup event.
STOP0	CPU SLEEPDEEP, Peripherals Clock gated. MR in LP mode optional. HSE/HSI/PLL disabled. LSI configurable. RTC optional. SRAM/All register retention. All IO retention. After waking up, HSI is enabled.	WFI/WFE: 1) SCB_SCR.SLEEPDEEP = 1, no pending interrupts/events. 2) PWR_CTRL.LPS = 0/1, PWR_CTRL.PDS = 0	Any interrupts wakeup event through EXTI, NRST, IWDG.
STOP2	CPU SLEEPDEEP, CPU register Retention. MR OFF, LSI configurable, HSE/HSI/PLL disabled, all IO retention, SRAM 8KB retention. Modules below are optional running RTC with LSI LPTIMER, PVD. After waking up, HSI is enabled.	WFI/WFE: 1) SCB_SCR.SLEEPDEEP = 1, no pending interrupt/event. 2) PWR_CTRL.LPS = 0, PWR_CTRL.PDS = 0 and PWR_CTRL2.STOP2S = 1	NRST, IWDG, PVD, RTC timer or timestamp or tamper event, LPTIMER, all GPIO can be configured as EXTI.
STANDBY	MR OFF, LSI ON, all other clocks OFF, all IO retention, but reset to default state after wakeup, except NRST, PA0_WKUP. Optional:	WFI/WFE: 1) SCB_SCR.SLEEPDEEP = 1, no pending interrupt/event.	NRST wakeup 1 WKUP_IO RTC timer or alarm or timestamp or tamper or IWDG wakeup. Reset after wakeup, all needs to be

ITEMs MODES	CONDITION	ENTRY	EXIT
	8KB SRAM Retention, RTC work with LSI, IWDG.	2) PWR_CTRL.LPS = 0, PWR_CTRL.PDS = 1	reconfigured.

Note:

1. STOP0 mode, after wakeup, codes can resume and continue from stopped location. No any initialization is needed.
2. STOP2 mode, after wakeup, if stack and global variables are inside SRAM. Codes can resume and continue from stopped location. Peripherals and PWR need to be reinitialized.

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

Table 3-2 Modules running status

Modules	Run	Sleep	Stop 0		Stop 2		Standby	
			-	Wakeup capability	-	Wakeup capability	-	Wakeup capability
CPU ⁽¹⁾	Y	-	-	-	-	-	-	-
Flash memory	O	O	-	-	-	-	-	-
SRAM	Y	Y ⁽²⁾	Y	-	Y	-	O	-
Backup Registers	Y	Y	Y	-	Y	-	Y	-
POR(VDDD)	Y	Y	Y	Y	-	-	-	-
POR(VDDA)	Y	Y	Y	Y	Y	Y	Y	Y
POR(VDBBK)	Y	Y	Y	Y	Y	Y	Y	Y
PVD	O	O	O	O	O	O	-	-
DMA	O	O	-	-	-	-	-	-
HSI	O	O	(3)	-	(3)	-	-	-
HSE	O	O	-	-	-	-	-	-
LSI	O	O	O	-	O	-	O	-
Clock Security System (CSS)	O	O	-	-	-	-	-	-
RTC / Auto wakeup	O	O	O	O	O	O	O	O
Number of RTC Tamper pins	1 ⁽⁴⁾	1 ⁽⁴⁾	1 ⁽⁴⁾	O ⁽⁴⁾	1 ⁽⁴⁾	O ⁽⁴⁾	1 ⁽⁴⁾	O ⁽⁴⁾

USARTx (x = 2)	O	O	-	-	-	-	-	-
UARTx(x=1, 4)	O	O	-	-	-	-	-	-
I2Cx (x=1, 2)	O	O	-	-	-	-	-	-
SPIx (x=1, 2)	O	O	-	-	-	-	-	-
ADCx (x=1)	O	O	-	-	-	-	-	-
COMPx(x=1, 2, 3)	O	O	O	O	-	-	-	-
Temperature sensor	O	O	-	-	-	-	-	-
TIMx(x=1~6, 8)	O	O	-	-	-	-	-	-
LPTIM	O	O	O	O	O	O	-	-
IWDG	O	O	O	O	O	O	O	O
WWDG	O	O	-	-	-	-	-	-
SysTick timer	O	O	-	-	-	-	-	-
CRC	O	O	-	-	-	-	-	-
GPIOs	O	O	O	O	O	O	1 pins ⁽⁵⁾	1 pins ⁽⁵⁾

Note:

1. Y: Yes(Enable), O: Optional(Disabled by default, Enabled by software), -: Not available.
2. SRAM clock can be turned on or off.
3. Some peripherals that can be awakened from Stop mode can ask HSI to be enabled. In this case, HSI is enabled by the peripheral and only supplied to the peripheral.
4. One pin support the Tamper function, namely PA0(Tamper)
5. The pins that can wake up from the STANDBY are PA0(WKUP), and NRST.

3.2.1 SLEEP mode

The CPU stops and all peripherals including peripherals around the Cortex®-M4 core (such as NVIC, SysTick, etc.) can run and wake up the CPU when an interrupt or event occurs.

3.2.1.1 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.

3.2.1.2 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.

3.2.2 STOP0 mode

STOP0 mode is based on the Cortex®-M4 Deep-Sleep, the external clock gating, with the corresponding power management. All clocks working on V_{DDD} stop, PLL, HSI, HSE disable, LSI run. RTC can be kept running, and some peripherals with wake capability can enable HSI when wake condition is detected. 8KB SRAM and all register retention. All I/O status must be consistent with the running status.

3.2.2.1 Enter STOP0 mode

When entering STOP0 mode, the main difference is to set SCB_SCR.SLEEPDEEP = 1 and PWR_CTRL.PDS = 0. Another difference is that MR can be in normal mode or low power mode, depending on PWR_CTRL.LPS. When PWR_CTRL.LPS = 1, the MR is in low power mode. When PWR_CTRL.LPS = 0, the MR is in normal mode.

If a FLASH operation is in progress, the time to enter STOP0 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 STOP0 mode will be delayed until the APB access is completed.

In STOP0 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 turned on by RCC_BDCTRL.RTCEN.
- Internal RC oscillator (LSI RC) optional: It can be turned on by RCC_CTRLSTS.LSIEN.

ADC should be disabled when entering STOP0 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.

3.2.2.2 Exit STOP0 mode

When an interrupt or wake-up event wakes up STOP0 mode, the HSI RC oscillator is selected as the system clock. If the voltage regulator is in low power mode, it takes extra start-up time to wakes up from STOP0 mode. Keeping MR in normal mode when entering STOP0 mode can reduce the startup time but with high power consumption.

3.2.3 STOP2 mode

STOP2 mode is based on the Cortex®-M4 Deep-Sleep, the external clock gating, with the corresponding power supply gating. All clocks working on V_{DDDD} stop, PLL, HSI, HSE disable, LSI run. RTC can be kept running, and some peripherals with wake capability can enable HSI when wake condition is detected. CPU register and 80-byte backup register retention. 8KB SRAM retention. All I/O status must be consistent with the running status. After waking up from STOP2 mode, the code continues to Run from hibernation and needs to be reconfigured using the peripheral interface and PWR.

3.2.3.1 Enter STOP2 mode

To enter STOP2 mode, register bits should be configured as: SCB_SCR.SLEEPDEEP = 1, PWR_CTRL2.STOP2S = 1, PWR_CTRL.PDS = 0, PWR_CTRL.LPS = 0.

In STOP2 mode, if FLASH is being operated, the time to enter STOP2 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 STOP2 mode will be delayed until the APB access is completed.

In STOP2 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 turned on by RCC_BDCTRL.RTCEN.
- Internal RC oscillator (LSI RC) optional: It can be turned on by RCC_CTRLSTS.LSIEN.

ADC should be disabled when entering STOP2 mode to avoid unnecessary power consumption.

Note: If data (global variables, stacks, etc.) needs to be kept in STOP2, they should be stored in SRAM.

3.2.3.2 Exit STOP2 mode

HSI RC oscillator is set as the system clock when STOP2 mode is waken up by an interrupt or wake-up event. The codes will resume execution after STOP2 wakeup from where it stopped. Peripheral initialization is needed after wakeup. PWR and other modules' clock should be enabled in RCC if need access.

3.2.4 STANDBY mode

STANDBY mode is based on Cortex®-M4 Deep-Sleep, with the voltage regulator turned off, PLL, HSI, HSE turned off, LSI running. SRAM retention, RTC and IWDG are optional. All IO retention, but reset to default state after wakeup, except NRST, PA0_WKUP.

Before entering the STANDBY mode, if pins PA13 and PA14 are used as non-debug pins ,and configured as input mode, it is necessary to add strong pull-down resistance on pins PA13 and PA14. The pull-down resistance is

recommended to be within $10K\Omega$.

3.2.4.1 Enter STANDBY mode

STANDBY mode is entered by WFI/WFE with settings SCB_SCR.SLEEPDEEP = 1 and PWR_CTRL.PDS = 1.

If FLASH is being operated, the time to enter STANDBY 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 STANDBY mode will be delayed until the APB access is completed.

In STANDBY 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 turned on by RCC_BDCTRL.RTCEN.
- Internal RC oscillator (LSI RC) optional: It can be turned on by RCC_CTRLSTS.LSIEN.
- SRAM data retention can be enabled by PWR_CTRL2.SRSTBRET.

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

3.2.4.2 Exit STANDBY mode

MCU exits STANDBY mode when external reset (NRST pin), IWDG reset, rising/falling edge of WKUP pin or rising edge of RTC alarm event occurs. Except the power control status register (PWR_CTRLSTS), all registers will be reset after waking up from the STANDBY state.

After waking up from STANDBY mode, code execution is same as power on (boot pin is detected, reset vector initialization, etc.). PWR_CTRLSTS.STBYF flag indicates that MCU exits from STANDBY mode.

3.3 PWR registers

3.3.1 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		
000h	PWR_CTRL1	Reserved																		Reserved															
		Reset Value																		0	CLRRTCWKUP	13	12	11	10	9	8	7	6	5	4	3	2	1	0
004h	PWR_CTRLSTS	Reserved																		Reserved															
		Reset Value																		0	WKUPPS	13	12	11	10	9	8	7	6	5	4	3	2	1	0
008h	PWR_CTRL2	Reserved																		Reserved															
		Reset Value																		1	IWDGRSTEN	13	12	11	10	9	8	7	6	5	4	3	2	1	0

3.3.2 Power control register (PWR_CTRL)

Address offset: 0x00

Reset value: 0x0000 0000 (reset by wakeup from STANDBY mode)

31	Reserved															16											
15	14	13	12	11	10	9	8	7	5	4	3	2	1	0													
Reserved	CLRRTCWKUP	Reserved	CLR WKUP	Reserved	MSB	DBKP		PLS[2:0]		PVDEN	CLR STBF	Reserved	PDS	LPS	rc_w1	rc_w1	rw										

Bit field	Name	Description																																		
31:14	Reserved	Reserved, the reset value must be maintained.																																		
13	CLRRTCWKUP	Clear RTC wakeup flag. This bit is always read as 0. 0: No effect 1: Clear the wakeup flag after 2 System clock cycles.																																		
12	Reserved	Reserved, the reset value must be maintained.																																		
11	CLRWKUP	Clear PA0 wakeup flag. This bit is always read as 0. 0: No effect 1: Clear the wakeup flag after 2 System clock cycles.																																		
10	Reserved	Reserved, the reset value must be maintained.																																		
9	MSB	MSB of 4bits PVD threshold. PVD threshold is controlled below: When the MSB bit is 0, PVD threshold is as: <table border="1"> <tr> <td>[MSB, PWR_CTRL.PLS] (4 bits)</td> <td>Voltage</td> </tr> <tr> <td>0000</td> <td>2.18v</td> </tr> <tr> <td>0001</td> <td>2.28v</td> </tr> <tr> <td>0010</td> <td>2.38v</td> </tr> <tr> <td>0011</td> <td>2.48v</td> </tr> <tr> <td>0100</td> <td>2.58v</td> </tr> <tr> <td>0101</td> <td>2.68v</td> </tr> <tr> <td>0110</td> <td>2.78v</td> </tr> <tr> <td>0111</td> <td>2.88v</td> </tr> </table> When the MSB bit is 1, PVD threshold is as: <table border="1"> <tr> <td>[MSB, PWR_CTRL.PLS](4 bits)</td> <td>Voltage</td> </tr> <tr> <td>1000</td> <td>1.78v</td> </tr> <tr> <td>1001</td> <td>1.88v</td> </tr> <tr> <td>1010</td> <td>1.98v</td> </tr> <tr> <td>1011</td> <td>2.08v</td> </tr> <tr> <td>1100</td> <td>3.28v</td> </tr> <tr> <td>1101</td> <td>3.38v</td> </tr> <tr> <td>1110</td> <td>3.48v</td> </tr> </table>	[MSB, PWR_CTRL.PLS] (4 bits)	Voltage	0000	2.18v	0001	2.28v	0010	2.38v	0011	2.48v	0100	2.58v	0101	2.68v	0110	2.78v	0111	2.88v	[MSB, PWR_CTRL.PLS](4 bits)	Voltage	1000	1.78v	1001	1.88v	1010	1.98v	1011	2.08v	1100	3.28v	1101	3.38v	1110	3.48v
[MSB, PWR_CTRL.PLS] (4 bits)	Voltage																																			
0000	2.18v																																			
0001	2.28v																																			
0010	2.38v																																			
0011	2.48v																																			
0100	2.58v																																			
0101	2.68v																																			
0110	2.78v																																			
0111	2.88v																																			
[MSB, PWR_CTRL.PLS](4 bits)	Voltage																																			
1000	1.78v																																			
1001	1.88v																																			
1010	1.98v																																			
1011	2.08v																																			
1100	3.28v																																			
1101	3.38v																																			
1110	3.48v																																			

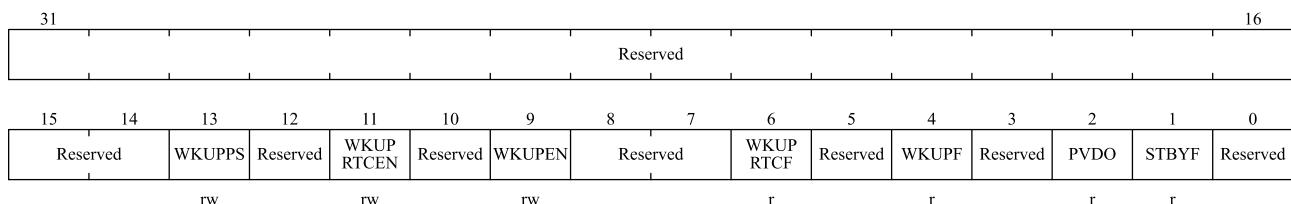
Bit field	Name	Description		
		1111	3.58v	
8	DBKP	Disable backup domain write protection. In reset state, the RTC and backup registers are protected against unauthorized access. This bit must be set to enable write protection to these registers. 0: Access to RTC and Backup registers disabled 1: Access to RTC and Backup registers enabled Note: If the HSE divided by 128 is used as the RTC clock, this bit must remain set to 1.		
7:5	PLS[2:0]	PVD level selection. When PVD threshold needs to be set, please configure these bits together with PWR_CTRL.MSB. <i>Note: refer to the electrical characteristics section of the data book for details.</i>		
4	PVDEN	Enable of Power voltage detector. Software control. 0: PVD disabled 1: PVD enabled		
3	CLRSTBF	Clear STANDBY flag. This bit is always read as 0. 0: No effect. 1: Clear the STANDBY.STBYF Flag .		
2	Reserved	Reserved, the reset value must be maintained.		
1	PDS	Power down deep-sleep. Software will set and clear this bit and config together with PWR_CTRL.LPS. 0: Enter stop mode when the CPU enters deep-sleep. The regulator status depends on PWR_CTRL.LPS. 1: Enter STANDBY mode when the CPU enters deep-sleep mode.		
0	LPS	Low-power deep-sleep. Software will set and clear this bit and config together with PWR_CTRL.PDS. 0: Voltage regulator on during stop mode. 1: Voltage regulator in low-power mode during stop mode.		

3.3.3 Power control status register (PWR_CTRLSTS)

Address offset: 0x04

Reset value: 0x0000 0000 (not reset by wakeup from STANDBY mode)

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



Bit field	Name	Description
31:14	Reserved	Reserved, the reset value must be maintained.
13	WKUPPS	<p>Wakeup polarity selection for PA0.</p> <p>To wakeup STANDBY mode by using rising edge or falling edge. Make sure disable wakeup enable before changing polarity value.</p> <p>0: Rising edge</p> <p>1: Falling edge</p>
12	Reserved	Reserved, the reset value must be maintained.
11	WKUPRTCEN	<p>RTC internal wakeup enable</p> <p>0: Wakeup is disabled</p> <p>1: Wakeup is enabled</p>
10	Reserved	Reserved, the reset value must be maintained.
9	WKUPEN	<p>Enable PA0_WKUP pin</p> <p>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 STANDBY mode.</p> <p>1: WKUP pin is used for wakeup from STANDBY mode.</p> <p>Note: This bit is reset also by a system reset.</p>
8:7	Reserved	Reserved, the reset value must be maintained.
6	WKUPRTCFC	<p>Wakeup flag of RTC</p> <p>Hardware will set this bit. It can be cleared by hardware, by a system reset or by setting PWR_CTRL.CLRRTCWKUP.</p> <p>0: No wakeup event occurred</p> <p>1: A wakeup event was received from the RTC</p>
5	Reserved	Reserved, the reset value must be maintained.
4	WKUPF	<p>Wakeup flag of PA0_WKUP</p> <p>Hardware will set this bit. It can be cleared by hardware, by a system reset or by setting PWR_CTRL.CLRWKUP.</p> <p>0: No wakeup event occurred</p> <p>1: A wakeup event was received from the WKUP pin</p> <p>Note: An additional wakeup event is detected if the WKUP pin is enabled (by setting PWR_CTRLSTS.WKUPEN) when the WKUP pin level is already high.</p>
3	Reserved	Reserved, the reset value must be maintained.
2	PVDO	<p>PVD output.</p> <p>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.MSB and PWR_CTRL.PLS[2:0].</p> <p>1: VDD/VDDA is lower than the PVD threshold selected with PWR_CTRL.MSB and PWR_CTRL.PLS[2:0].</p> <p>Note: The PVD is stopped by STANDBY mode. For this reason, this bit is equal to 0 after STANDBY or reset until PWR_CTRL.PVDEN is set.</p>
1	STBYF	<p>STANDBY mode flag.</p> <p>Hardware will set this bit. It is cleared only by a POR/PDR (power on reset/power</p>

Bit field	Name	Description
		down reset) or by setting PWR_CTRL.CLRSTBF. 0: Device has not been in STANDBY mode 1: Device has been in STANDBY mode
0	Reserved	Reserved, the reset value must be maintained.

3.3.4 Power control register 2(PWR_CTRL2)

Address offset: 0x08

Reset value: 0x0000 04E4 (reset by wakeup from STANDBY mode)

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	Reserved	Reserved, the reset value must be maintained.
8:4	LSITRIM[4:0]	LSI trimming value
3	Reserved	Reserved, the reset value must be maintained.
2	SRSTBRET	SRAM STANDBY mode retention. 0: SRAM is not in retention in STANDBY mode. 1: SRAM is in retention in STANDBY mode.
1	Reserved	Reserved, the reset value must be maintained.
0	STOP2S	STOP2 sleep mode control. 0: No-effect 1: Chip is set to be in STOP2 mode.

4 Reset and clock control (RCC)

4.1 General description

The clock and reset control modules in this system are responsible for the following functions:

- CGU: The CGU sub module is placed in the VDDD power domain and controls the System Clock selection, Clock enable control for peripherals and Clock frequency division functions.
- RCU: The RCU sub module is placed in the VDDD power domain and it is responsible for the System Reset control and generation and soft reset generation for peripherals.
- RCC_BKP: The RCC_BKP sub module is placed in the VDDDBK power domain and it is responsible for Backup domain reset and clock control and clock source selection for RTC clock.
- RCC_RET: The RCC_RET sub module is placed in the VDDDRET power domain and it is responsible for Retention domain reset control and clock source selection for LPTIMER clock.

4.2 Reset Control Unit

N32G401F8S7-1 supports the following three types of reset:

- Power Reset
- System Reset
- Backup domain Reset

4.2.1 Power reset

A Power reset occurs in the following circumstances:

- Power-on/ Power-down reset (POR/PDR reset).
- When exiting standby/stop2 mode.

A power reset sets all registers to their reset values except the registers in the backup domain(see Figure 3-1).

4.2.2 System reset

Except the reset flags in the Control/Status Register (RCC_CTRLSTS)and the registers in the backup domain (see Figure 3-1), a system reset sets all registers to their reset values.

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
- Power reset
- MMU protection reset
- Backup domain EMC reset

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

4.2.2.1 Software reset

A software reset can be generated by setting the SYSRESETREQ bit in Cortex™-M4 Application Interrupt and Reset Control Register. Refer to Cortex™-M4 technical reference manual for further information.

4.2.2.2 Low-power management reset

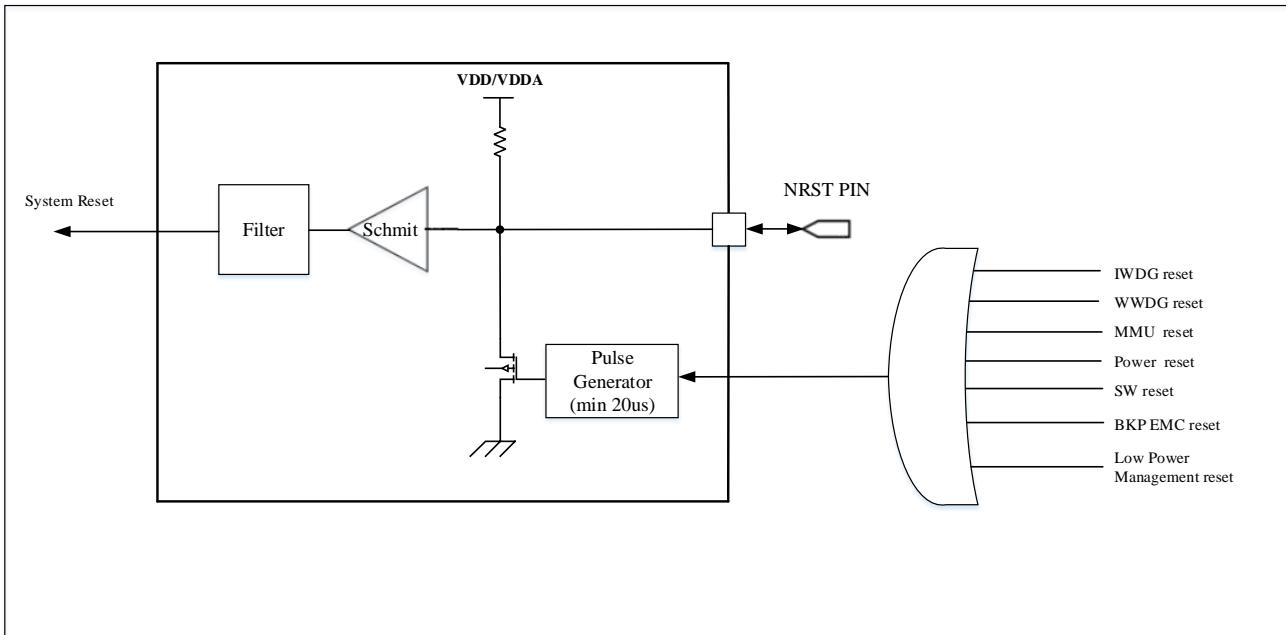
Low-power management reset can be generated by using the following methods:

- Low-power management reset generated when entering Standby mode: This reset is enabled by resetting the nRST_STDBY bit in User Option Bytes. In this case, whenever a Standby mode entry sequence is successfully executed, the system is reset instead of entering Standby mode.
- Low-power management reset generated when entering STOP0/STOP2 modes: This reset is enabled by resetting the nRST_STOP bit in User Option Bytes. In this case, whenever a STOP0/STOP2 mode entry sequence is successfully executed, the system is reset instead of entering STOP0/STOP2 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 20 μ s 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



4.2.3 Backup domain reset

The backup domain has two dedicated resets that only affect the backup domain (see Figure 3-1).

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

- Software reset: The backup domain reset can be generated by setting the RCC_BDCTRL.BDSFTRST bit.
- Power reset: The backup domain reset is generated when VDDDBK is powered up at initial chip power on stage.

4.3 Clock control unit

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

- HSI oscillator clock
- HSE oscillator clock
- PLL clock

The devices have the following one secondary clock sources:

- LSI: 40 kHz low-speed internal RC which drives independent watchdog (IWDG) can be selected by software to drive RTC. RTC can be used for Auto-wakeup from Sleep/Stop0/Stop2/Standby mode.

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

Several prescalers can be used to configure the frequencies of the AHB, the high-speed APB (APB2), and the low-speed APB (APB1) domains. The maximum frequencies of the AHB, APB2, and APB domains are 72MHz, 64MHz, and 32MHz respectively.

Note: If APB2 and APB1 want to run at the maximum frequency, then need to reduce the system clock to 64MHz.

RCC provides the Cortex System Timer (SysTick) external clock with the AHB clock (HCLK) divided by 8. This clock or Cortex clock(HCLK) can be selected to drive the SysTick by programming the SysTick Control and Status Register. The ADC clock is generated by dividing the AHB clock or PLL clock.

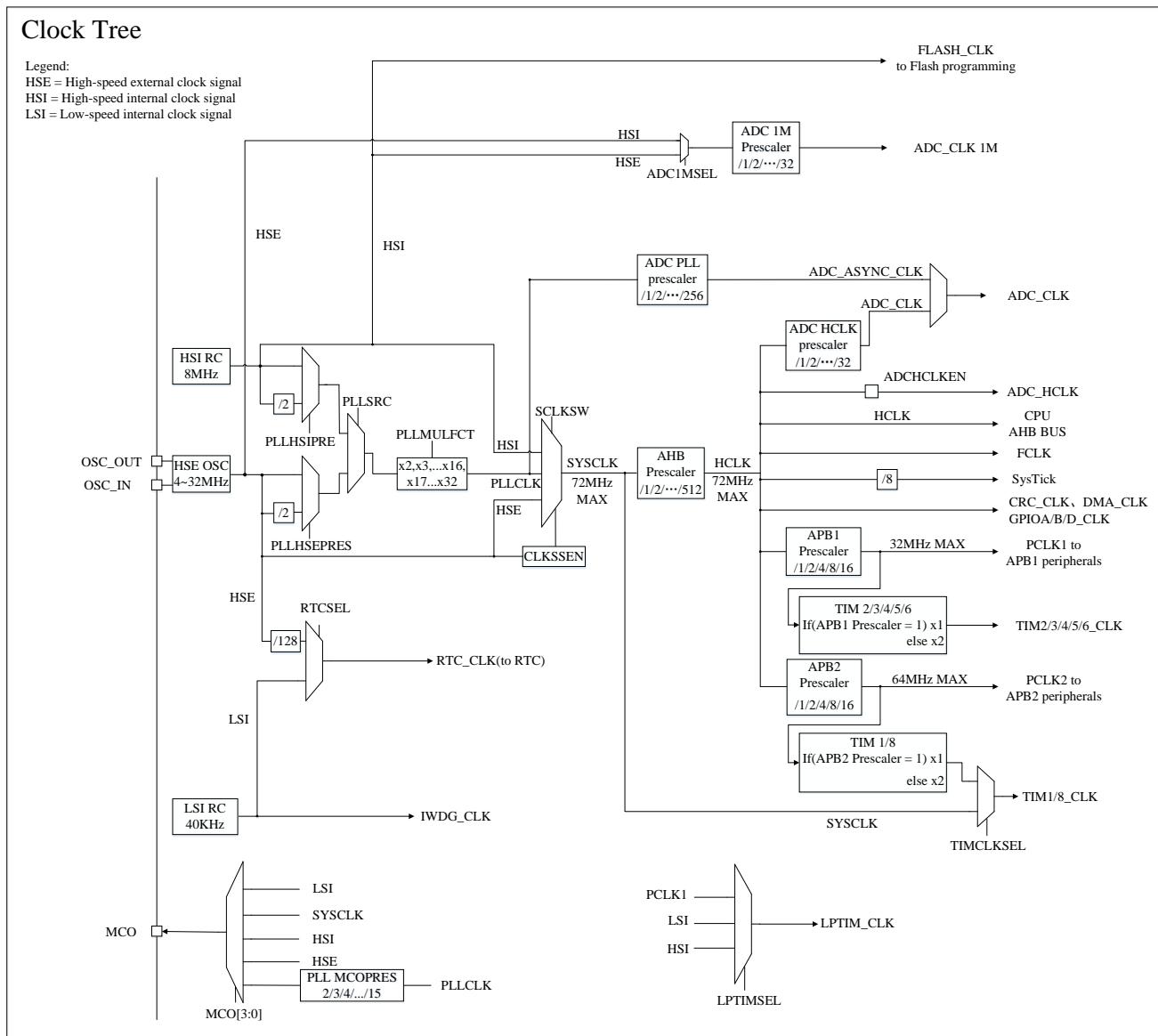
The clock frequencies of timers are automatically set by hardware. There are two scenarios:

- If the APB prescaler is 1, the timer clock frequencies are set to the same frequency as that of the APB domain to which the timers are connected.
- Otherwise, they are set to twice the frequency of the APB domain to which the timers are connected.

FCLK is the free-running clock of Cortex™-M4F. For more details, refer to the ARM Cortex™-M4 technical reference manual.

4.3.1 Clock Tree Diagram

Figure 4-2 Clock Tree



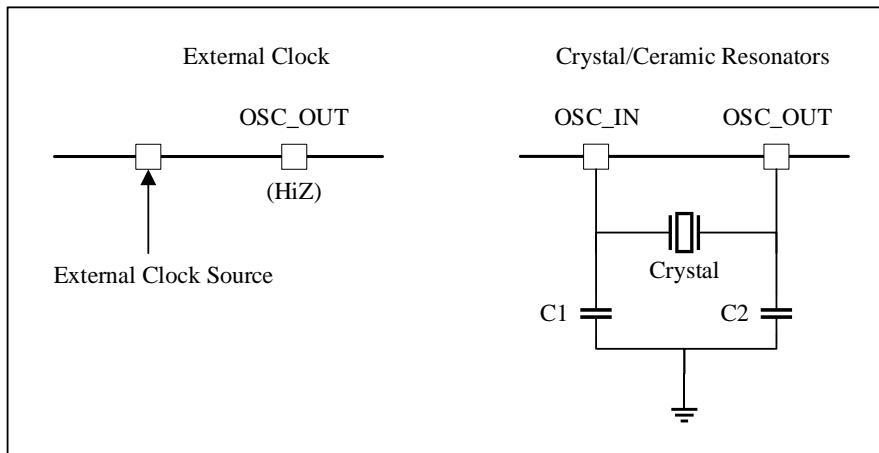
1. The maximum frequency available for the system clock is 72MHz.
2. For more details about the internal and external clock source characteristics, please refer to the "Electrical Characteristics" section in the product datasheet.

4.3.2 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

Figure 4-3 HSE clock source



4.3.2.1 External clock source (HSE bypass)

In this mode, an external clock source must be provided. Its frequency can be up to 32MHz. Users can select this mode by setting the RCC_CTRL.HSEBP and RCC_CTRL.HSEEN bits. The external clock signal must be connected to the OSC_IN pin while the OSC_OUT pin must be left floating (Hi-Z). See Figure 4-3.

4.3.2.2 External crystal/ceramic resonator (HSE crystal)

The 4 to 32MHz 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.

4.3.3 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. 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. After the system reset, the factory calibration value is loaded into the RCC_CTRL.HSICAL[7:0] bits.

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 4.3.7 Clock security system (CLKSS).

4.3.4 PLL clock

The internal PLL can be used to multiply the HSI or the HSE clock frequency. Refer to Figure 4-2 Clock Tree, The PLL configuration (selection of PLL input clock (HSI/HSE and divider) and multiplication factor) must be done before enabling PLL. Once the PLL is enabled, these parameters cannot be changed. The PLL can be configured using control bits in RCC_CTRL and RCC_CFG registers.

An interrupt can be generated when the PLL is ready if enabled in the Clock Interrupt Register (RCC_CLKINT).

4.3.5 LSI clock

The LSI RC oscillator provides the Low Speed Internal (LSI) clock with frequency of around 40 kHz. 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).

4.3.5.1 LSI calibration

The internal low-speed oscillator LSI can be calibrated to compensate for its frequency offset to obtain an RTC time base with acceptable accuracy, and an independent watchdog (IWDG) timeout (when these peripherals are clocked from the LSI).

Calibration can be achieved by measuring the LSI clock frequency using the TIM2's input clock (TIM2_CLK). The measurement is guaranteed by the accuracy of the HSE. The software can obtain the accurate RTC clock base by adjusting the prescaler of the RTC, and obtain the accurate independent watchdog (IWDG) timeout time by calculation.

The LSI calibration steps are as follows:

1. Turn on TIM2 and set channel 3 to input capture mode;
2. Set the TIM2_CTRL1.C3SEL bit to 1, and connect the LSI to channel 3 of TIM2 internally;
3. Measure LSI clock frequency through TIM2 capture/compare 3 events or interrupts;
4. Set the prescaler based on the measurement results and the desired RTC time base and independent watchdog timeout.

4.3.6 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 the Clock Configuration Register (RCC_CFG) and Clock Control Register (RCC_CTRL) indicate which clock is ready and which clock is currently used as the system clock.

4.3.7 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™-M4.

The NMI will be executed continuously until the CLKSSIF interrupt pending bit is cleared. Therefore, it is necessary to clear the 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.

4.3.8 RTC clock

By programming RCC_BDCTRL.RTCSEL[1:0] bits, the RTCCLK clock source can be either the HSE/128 or LSI clocks. This selection cannot be changed unless the backup domain is reset.

Before configuring the RTC clock source, the DBPK bit of the power control register PWR_CTRL must be set to 1 to cancel the write protection.

LSI clock is in the backup domain , HSE clock is not in the backup domain :

- If LSI is selected as RTC clock:
 - ◆ If the V_{DD} supply is switched off, the RTC cannot continue to work
- If the HSE clock divided by 128 is used as the RTC clock:
 - ◆ Only works in run/sleep mode.

4.3.9 Watchdog clock

If the IWDG is started by either hardware option or software access, the LSI oscillator will be forced ON and cannot be disabled. After the LSI oscillator is stabilized, the clock is provided to the IWDG.

4.3.10 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
- PLL clock division
- LSI

The clock selection is controlled by RCC_CFG.MCO[3:0] bits.

4.4 RCC Registers

The RCC registers are accessible through AHB bus. The register description is as follows.

4.4.1 RCC register overview

Table 4-1 RCC 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	RCC_CTRL																																	
004h	RCC_CFG	MCOPRES[3:0]	PLLMULFCT[4]	MCO[3:0]	PLLMULFCT[3:0]	PLLDRDF	PLLLEN	PLLLEN	PLLLEN	PLLLEN	PLLLEN	PLLLEN	PLLLEN	PLLLEN	PLLLEN	PLLLEN	PLLLEN	PLLLEN	PLLLEN	PLLLEN	PLLLEN	PLLLEN	PLLLEN	PLLLEN	PLLLEN	PLLLEN	PLLLEN	PLLLEN	PLLLEN	PLLLEN	PLLLEN			
008h	RCC_CLKINT	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved			
00Ch	RCC_APB2PRST	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved			
010h	RCC_APB1PRST	Reserved	PWRST	Reserved	12C2RST	12C1RST	Reserved	UART2RST	UART4RST	Reserved	PLLHSEPRE																							
014h	RCC_AHBPCLEN	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved			
018h	RCC_APB2PCLEN	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved			
01Ch	RCC_APB1PCLEN	Reserved	PWREN	Reserved	12C2EN	12C1EN	Reserved	SPIEN	ADCEN	Reserved	WWDDGEN	WWDDGEN	WWDDGEN																					

020h	RCC_BDCTRL	Reserved										0	BDSFIRST	Reserved		Reserved											
	Reset Value																										
024h	RCC_CTRLSTS	LPWRRSTF	WWDGRSTF	IWDGRSTF	SFRSTF	PORRSTF	PINRSTF	MMURSTF	RMRSTF	Reserved	BKPEMCF	Reserved															
	Reset Value	0	0	0	0	1	1	0	0	0	0																
028h	RCC_AHBPRST	Reserved										0	ADCRST	Reserved		Reserved											
	Reset Value											0	IOPDRST	Reserved		Reserved											
02Ch	RCC_CFG2	Reserved	TIMCLKSEL	Reserved					ADC1MPRES[4:0]				ADCPLLPRES[4:0]		ADCHPRES[3:0]												
	Reset Value			0						0	0	1	1	1	0	0	0	0	0	0							
034h	RCC_RDCTRL	Reserved										0	LPTMRST	Reserved		Reserved											
	Reset Value											0	LPTMEN	Reserved		Reserved											
040h	RCC_PLLHSIPRE	Reserved																									
	Reset Value											0	PLLHSIPRE	1													
080h	RCC_AHB1CLKEN	Reserved																									
	Reset Value											0	ADCHICKEN	0		PLLHSIPRE											

4.4.2 Clock Control Register (RCC_CTRL)

Address offset: 0x00

Reset value: 0x0000 7783

31	Reserved	26	25	24	23	Reserved	20	19	18	17	16
15			PLL RDF	PLLEN			CLKSSEN	HSEBP	HSERDF	HSEEN	
		r		rw	7		rw	rw	r	rw	
	HSICAL[7:0]					HSITRIM[4:0]		Reserved	HSIRDF	HSIEN	
		r					rw		r	rw	

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

Bit Field	Name	Description
		1: PLL is ready
24	PLLEN	<p>PLL enable</p> <p>Set and cleared by software. When entering the STOP0/STOP2 or standby mode, it is cleared by hardware. This bit cannot be cleared when PLL is used as the system clock.</p> <p>0: Disable PLL</p> <p>1: Enable PLL</p>
23:20	Reserved	Reserved, the reset value must be maintained.
19	CLKSSEN	<p>Clock security system enable</p> <p>Set and cleared by software.</p> <p>0: Disable the clock detector</p> <p>1: Enable the clock detector if the HSE oscillator is ready</p>
18	HSEBP	<p>External high-speed clock bypass enable</p> <p>Set and cleared by software. This bit can only be written when the HSE oscillator is disabled.</p> <p>0: Disable the bypass function of HSE oscillator</p> <p>1: Enable the bypass function of HSE oscillator</p>
17	HSERDF	<p>External high-speed clock ready flag</p> <p>Set by hardware once HSE is ready. This bit takes 6HSE clock cycles to clear after the HSEEN bit is cleared.</p> <p>0: HSE is not ready</p> <p>1: HSE is ready</p>
16	HSEEN	<p>External high-speed clock enable</p> <p>Set and cleared by software. When entering the STOP0/STOP2 or standby mode, it is cleared by hardware. This bit cannot be cleared when HSE is used directly or indirectly as the system clock.</p> <p>0: Disable HSE oscillator</p> <p>1: Enable HSE oscillator</p>
15:8	HSICAL[7:0]	<p>Internal high-speed clock calibration value</p> <p>These bits are automatically initialized at startup.</p>
7:3	HSITRIM[4:0]	<p>Internal high-speed clock correction value</p> <p>Written by software. The values of these bits will be added to the HSICAL[7:0] bits in order to form the final value for calibrating the frequency of the internal HSI RC oscillator. The trimming step is around 10 kHz between two consecutive HSICAL steps, and the default value is 16, which can adjust the HSI to 8 MHz $\pm 1\%$.</p>
2	Reserved	Reserved, the reset value must be maintained.
1	HSIRDF	<p>Internal high-speed clock ready flag</p> <p>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.</p> <p>0: HSI is not ready</p> <p>1: HSI is ready</p>
0	HSIEN	Internal high-speed clock enable

Bit Field	Name	Description
		<p>Set and cleared by software. This bit cannot be cleared when HSI is used as the system clock. When returning from STOP0/STOP2 or standby mode or HSE failure occurs, set by hardware to enable the HSI oscillator. This bit cannot be reset if the HSI is used directly or indirectly as system clock.</p> <p>0: Disable HSI oscillator 1: Enable HSI oscillator</p>

4.4.3 Clock Configuration Register (RCC_CFG)

Address offset: 0x04

Reset value: 0x2000 0000

31	MCOPRES[3:0]	28	27	26	MCO[3:0]	23	22	21	PLLMULFCT[4]	18	17	16
15	rw	14	rw	13	rw	11	10	rw	8	7	4	rw
Reserved	APB2PRES[2:0]		APB1PRES[2:0]		AHBPRES[3:0]				SCLKSTS[1:0]		SCLKSW[1:0]	
	rw		rw		rw				r		rw	

Bit Field	Name	Description
31:28	MCOPRES[3:0]	<p>MCO prescaler</p> <p>Set and cleared by software. Configure PLL clock divider as MCO clock.</p> <p>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</p> <p>Other values: not allowed</p>
27	PLLMULFCT[4]	This bit is combined with bit[21:18] to form a PLL multiplication factor. Please refer to PLLMULFCT[3:0].
26:23	MCO[3:0]	<p>Microcontroller clock output selection</p> <p>Set and cleared by software.</p> <p>0xxx: No clock output</p>

Bit Field	Name	Description
		<p>1000: System clock (SYSCLK) selected 1001: HSI clock selected 1010: HSE clock selected 1011: PLL divided clock selected (MCOPRES defines the division factor) 1100: LSI clock selected Others: reserved</p> <p><i>Note: This clock output may be truncated at startup or during MCO clock source switching. When the system clock is output to the MCO pin, it should be ensured that the output clock frequency does not exceed the maximum frequency of the I/O port (see the data sheet for details of the maximum frequency of the I/O port).</i></p>
22	Reserved	Reserved, the reset value must be maintained.
21:18	PLLMULFCT[3:0]	<p>PLL multiplication factor (including bit 27)</p> <p>Written by software to define PLL multiplication factor. These bits can only be written when the PLL is disabled. The PLL output frequency must not exceed 72MHz.</p> <p>00000: PLL input clock $\times 2$ 00001: PLL input clock $\times 3$ 00010: PLL input clock $\times 4$ 00011: PLL input clock $\times 5$ 00100: PLL input clock $\times 6$ 00101: PLL input clock $\times 7$ 00110: PLL input clock $\times 8$ 00111: PLL input clock $\times 9$ 01000: PLL input clock $\times 10$ 01001: PLL input clock $\times 11$ 01010: PLL input clock $\times 12$ 01011: PLL input clock $\times 13$ 01100: PLL input clock $\times 14$ 01101: PLL input clock $\times 15$ 01110: PLL input clock $\times 16$ 01111: PLL input clock $\times 16$ 10000: PLL input clock $\times 17$ 10001: PLL input clock $\times 18$ 10010: PLL input clock $\times 19$ 10011: PLL input clock $\times 20$ 10100: PLL input clock $\times 21$ 10101: PLL input clock $\times 22$ 10110: PLL input clock $\times 23$ 10111: PLL input clock $\times 24$ 11000: PLL input clock $\times 25$ 11001: PLL input clock $\times 26$ 11010: PLL input clock $\times 27$</p>

Bit Field	Name	Description
		11011: PLL input clock × 28 11100: PLL input clock × 29 11101: PLL input clock × 30 11110: PLL input clock × 31 11111: PLL input clock × 32
17	PLLHSEPRE	HSE prescaler for PLL input Set and cleared by software to divide HSE before PLL entry. This bit can only be written when PLL is disabled. 0: HSE clock not divided 1: HSE divided by 2
16	PLLSRC	PLL clock source Set and cleared by software to select PLL clock source. This bit can only be written when PLL is disabled. 0: HSI clock (divided or not divided) selected as PLL input clock 1: HSE clock (divided or not divided) selected as PLL input clock
15:14	Reserved	Reserved, the reset value must be maintained.
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 64MHz. 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 APB1 clock (PCLK1). Make sure that PCLK1 does not exceed 32MHz. 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). 0xxx: 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

Bit Field	Name	Description
		1110: SYSCLK divided by 256 1111: SYSCLK divided by 512
3:2	SCLKSTS[1:0]	System clock switching status Set and cleared by hardware to indicate which clock source is used as system clock 00: HSI oscillator used as system clock 01: HSE oscillator used as system clock 10: PLL used as system clock 11: Not applicable
1:0	SCLKSW[1:0]	System clock switch Set and cleared by software to select the system clock source. Set by hardware to force HSI selection when exiting from the STOP0/STOP2 or standby mode, or when the HSE oscillator fails (RCC_CTRL.CLKSSEN is enabled). 00: HSI selected as system clock 01: HSE selected as system clock. 10: PLL selected system clock 11: Not allowed

4.4.4 Clock Interrupt Register (RCC_CLKINT)

Address offset: 0x08

Reset value: 0x0000 0000

31	Reserved								24	23	22	21	20	19	18	17	16
									CLKSS ICLR	Reserved	PLL RD ICLR	HSERD ICLR	HSIRD ICLR	Reserved	LSIRD ICLR		
15	13	12	11	10	9	8	w 7	6	5	w 4	w 3	w 2	1	w 0			
Reserved	PLL RD IEN	HSERD IEN	HSIRD IEN	Reserved	LSIRD IEN	CLKS IF	Reserved	PLL RD IF	HSERD IF	HSIRD IF	Reserved	LSIRD IF					
	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	Clock security system interrupt clear Set by the software to clear the CLKSSIF flag. 0: No effect 1: Clear the CLKSSIF flag
22:21	Reserved	Reserved, the reset value must be maintained.
20	PLL RDICLR	PLL ready interrupt clear Set by the software to clear the PLLRDIF flag. 0: No effect 1: Clear the PLLRDIF flag
19	HSERDICLR	HSE ready interrupt clear

Bit Field	Name	Description
		Set by the software to clear the HSERDIF flag. 0: Not used 1: Clear HSERDIF flag
18	HSIRDICLR	HSI ready interrupt clear Set by the software to clear the HSIRDIF flag. 0: Not used 1: Clear the HSIRDIF flag
17	Reserved	Reserved, the reset value must be maintained.
16	LSIRDICLR	LSI ready interrupt clear Set by software to clear the LSIRDIF flag. 0: Not used 1: Clear the LSIRDIF flag
15:13	Reserved	Reserved, the reset value must be maintained.
12	PLL RDIEN	PLL ready interrupt enable Set and cleared by software to enable and disable PLL ready interrupt 0: Disable PLL ready interrupt 1: Enable PLL ready interrupt
11	HSE RDIEN	HSE ready interrupt enable Set and cleared by software to enable and disable HSE ready interrupt. 0: Disable HSE ready interrupt 1: Enable HSE Ready Interrupt
10	HSI RDIEN	HSI ready interrupt enable Set and cleared by software to enable and disable HSI ready interrupt. 0: Disable HSI ready interrupt 1: Enable HSI ready interrupt
9	Reserved	Reserved, the reset value must be maintained.
8	LSI RDIEN	LSI ready interrupt enable Set and cleared by software to enable and disable LSI ready interrupt. 0: Disable LSI ready interrupt 1: Enable LSI ready interrupt
7	CLKSSIF	Clock security system interrupt flag Set by hardware when a failure is detected in the external HSE oscillator. 0: No clock security system interrupt caused by HSE clock failure 1: Clock security system interrupt caused by HSE clock failure
6:5	Reserved	Reserved, the reset value must be maintained.
4	PLL RDIF	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 PLLRDICLR bit. 0: No clock ready interrupt caused by PLL lock 1: Clock ready interrupt caused by PLL lock
3	HSE RDIF	HSE ready interrupt flag Set by hardware when HSERDIEN is set and the HSE clock is ready.

Bit Field	Name	Description
		<p>This bit is cleared by software by setting the HSERDICLR bit.</p> <p>0: No clock ready interrupt caused by HSE oscillator</p> <p>1: Clock ready interrupt caused by HSE oscillator</p>
2	HSIRDIF	<p>HSI ready interrupt flag</p> <p>Set by hardware when HSIRDEN is set and the HSI clock is ready.</p> <p>This bit is cleared by software by setting the HSERDICLR bit.</p> <p>0: No clock ready interrupt caused by HSI oscillator</p> <p>1: Clock ready interrupt caused by HSI oscillator</p>
1	Reserved	Reserved, the reset value must be maintained.
0	LSIRDIF	<p>LSI ready interrupt flag</p> <p>Set by the hardware when LSIRDEN is set and the LSI clock is ready.</p> <p>This bit is cleared by software by setting the LSIRDICLR bit.</p> <p>0: No clock ready interrupt caused by LSI oscillator</p> <p>1: Clock ready interrupt caused by LSI oscillator</p>

4.4.5 APB2 Peripheral Reset Register (RCC_APB2PRST)

Address offset: 0x0c

Reset value: 0x0000 0000

31	Reserved										20	19	18	17	16
15	14	13	12	11	10						rw	rw	2	1	0
Reserved	UART1 RST	TIM8RST	SPI1RST	TIM1RST							Reserved			BEEPRST	AFIORST

Bit Field	Name	Description
31:20	Reserved	Reserved, the reset value must be maintained.
19	SPI2RST	<p>SPI2 reset</p> <p>Set and cleared by software.</p> <p>0: Clear the reset</p> <p>1: Reset SPI2</p>
18	UART4RST	<p>UART4 reset</p> <p>Set and cleared by software.</p> <p>0: Clear the reset</p> <p>1: Reset UART4</p>
17:15	Reserved	Reserved, the reset value must be maintained.
14	UART1RST	<p>UART1 reset</p> <p>Set and cleared by software.</p> <p>0: Clear the reset</p> <p>1: Reset UART1</p>

Bit Field	Name	Description
13	TIM8RST	TIM8 timer reset Set and cleared by software. 0: Clear the reset 1: Reset TIM8 timer
12	SPI1RST	SPI1 reset Set and cleared by software. 0: Clear the reset 1: Reset SPI1
11	TIM1RST	TIM1 timer reset Set and cleared by software. 0: Clear the reset 1: Reset TIM1 timer
10:2	Reserved	Reserved, the reset value must be maintained.
1	BEEPRST	BEEPER reset Set and cleared by software. 0: Clear the reset 1: Reset BEEPER
0	AFIORST	Alternate function IO reset Set and cleared by software. 0: Clear the reset 1: Reset Alternate Function

4.4.6 APB1 Peripheral Reset Register (RCC_APB1PRST)

Address offset: 0x10

Reset value: 0x0000 0000

31	29	28	27	Reserved	23	22	21	20	18	17	16
		PWRRST		Reserved		I2C2RST	I2C1RST		Reserved	USART2 RST	Reserved
15	rw	12	11	10		rw	rw	5	4	3	2
			Reserved	WWDG RST		Reserved		TIM6RST	TIM5RST	TIM4RST	TIM3RST
								rw	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 the reset 1: Reset the power interface
27:23	Reserved	Reserved, the reset value must be maintained.
22	I2C2RST	I2C2 reset

Bit Field	Name	Description
		Set and cleared by software. 0: Clear the reset 1: Reset I2C2
21	I2C1RST	I2C1 reset Set and cleared by software. 0: Clear the reset 1: Reset I2C1
20:18	Reserved	Reserved, the reset value must be maintained.
17	USART2RST	USART2 reset Set and cleared by software. 0: Clear the 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 the reset 1: Reset window watchdog
10:5	Reserved	Reserved, the reset value must be maintained.
4	TIM6RST	TIM6 timer reset Set and cleared by software. 0: Clear the reset 1: Reset TIM6 timer
3	TIM5RST	TIM5 timer reset Set and cleared by software. 0: Clear the reset 1: Reset TIM5 timer
2	TIM4RST	TIM4 timer reset Set and cleared by software. 0: Clear the reset 1: Reset TIM4 timer
1	TIM3RST	TIM3 timer reset Set and cleared by software. 0: Clear the reset 1: Reset TIM3 timer
0	TIM2RST	TIM2 timer reset Set and cleared by software. 0: Clear the reset 1: Reset TIM2 timer

4.4.7 AHB Peripheral Clock Enable Register (RCC_AHBCLKEN)

Address offset: 0x14

Reset value: 0x0000 0014

31	Reserved															16
15	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Reserved	ADCEN	Reserved	IOPDEN	Reserved	IOPBEN	IOPAEN	CRCEN	Reserved	FLITFEN	Reserved	SRAMEN	Reserved	DMAEN			

rw rw

Bit Field	Name	Description
31:13	Reserved	Reserved, the reset value must be maintained.
12	ADCEN	ADC interface clock enable Set and cleared by software. 0: ADC interface clock disabled 1: ADC interface clock enabled
11	Reserved	Reserved, the reset value must be maintained.
10	IOPDEN	IO Port D clock enable Set and cleared by software. 0: IO Port D clock disabled 1: IO Port D clock enabled
9	Reserved	Reserved, the reset value must be maintained.
8	IOPBEN	IO Port B clock enable Set and cleared by software. 0: IO Port B clock disabled 1: IO Port B clock enabled
7	IOPAEN	IO Port A clock enable Set and cleared by software. 0: IO Port A clock disabled 1: IO Port A clock enabled
6	CRCEN	CRC clock enable Set and cleared by software. 0: CRC clock disabled 1: CRC clock enabled
5	Reserved	Reserved, the reset value must be maintained.
4	FLITFEN	FLITF clock enable Set and cleared by software to disable/enable FLITF clock during Sleep mode. 0: FLITF clock disabled during Sleep mode 1: FLITF clock enabled during Sleep mode
3	Reserved	Reserved

Bit Field	Name	Description
2	SRAMEN	SRAM interface clock enable Set and cleared by software to disable/enable SRAM interface clock during Sleep mode. 0: SRAM interface clock disabled during Sleep mode. 1: SRAM interface clock enabled during Sleep mode
1	Reserved	Reserved, the reset value must be maintained.
0	DMAEN	DMA clock enable Set and cleared by software. 0: DMA clock disabled 1: DMA clock enabled

4.4.8 APB2 Peripheral Clock Enable Register (RCC_APB2PCLKEN)

Address offset: 0x18

Reset value: 0x0000 0000

31	Reserved										20	19	18	17	16
											SPI2EN	UART4EN	Reserved		
15	14	13	12	11	10						rw	rw		1	0
Reserved	UART1EN	TIM8EN	SPI1EN	TIM1EN									BEEPEN	AFIOEN	
	rw	rw	rw	rw									rw	rw	

Bit Field	Name	Description
31:20	Reserved	Reserved, the reset value must be maintained.
19	SPI2EN	SPI2 clock enable Set and cleared by software. 0: SPI2 clock disabled 1: SPI2 clock enabled
18	UART4EN	UART4 clock enable Set and cleared by software. 0: UART4 clock disabled 1: UART4 clock enabled
17:15	Reserved	Reserved, the reset value must be maintained.
14	UART1EN	UART1 clock enable Set and cleared by software. 0: UART1 clock disabled 1: UART1 clock enabled
13	TIM8EN	TIM8 Timer clock enable Set and cleared by software. 0: TIM8 timer clock disabled 1: TIM8 timer clock enabled

Bit Field	Name	Description
12	SPI1EN	SPI1 clock enable Set and cleared by software. 0: SPI1 clock disabled 1: SPI1 clock enabled
11	TIM1EN	TIM1 timer clock enable Set and cleared by software. 0: TIM1 timer clock disabled 1: TIM1 timer clock enabled
10:2	Reserved	Reserved, the reset value must be maintained.
1	BEEPEN	BEEPER clock enable Set and cleared by software. 0: BEEPER clock disabled 1: BEEPER clock enabled
0	AFIOEN	Alternate function IO clock enable Set and cleared by software. 0: Alternate Function IO clock disabled 1: Alternate Function IO clock enabled

4.4.9 APB1 Peripheral Clock Enable Register (RCC_APB1PCLKEN)

Address offset: 0x1c

Reset value: 0x0000 0000

31	29	28	27	23	22	21	20	18	17	16
Reserved	PWREN		Reserved	I2C2EN	I2C1EN		Reserved	USART2 EN	Reserved	
15	rw 12	rw 11	rw 10	rw 5	rw 4	rw 3	rw 2	rw 1	rw 0	
Reserved	WWDGEN		Reserved	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN		

Bit Field	Name	Description
31:29	Reserved	Reserved, the reset value must be maintained.
28	PWREN	Power interface clock enable Set and cleared by software. 0: Power interface clock disabled 1: Power interface clock enable
27:23	Reserved	Reserved, the reset value must be maintained.
22	I2C2EN	I2C2 clock enable Set and cleared by software. 0: I2C2 clock disabled 1: I2C2 clock enabled

Bit Field	Name	Description
21	I2C1EN	I2C1 clock enable Set and cleared by software. 0: I2C1 clock disabled 1: I2C1 clock enabled
20:18	Reserved	Reserved, the reset value must be maintained.
17	USART2EN	USART2 clock enable Set and cleared by software. 0: USART2 clock disabled 1: USART2 clock enabled
16:12	Reserved	Reserved, the reset value must be maintained.
11	WWDGEN	Window watchdog clock enable Set and cleared by software. 0: Window watchdog clock disabled 1: Window watchdog clock enabled
10:5	Reserved	Reserved, the reset value must be maintained.
4	TIM6EN	TIM6 timer clock enable Set and cleared by software. 0: TIM6 clock disabled 1: TIM6 clock enabled
3	TIM5EN	TIM5 timer clock enable Set and cleared by software. 0: TIM5 clock disabled 1: TIM5 clock enabled
2	TIM4EN	TIM4 timer clock enable Set and cleared by software. 0: TIM4 clock disabled 1: TIM4 clock enabled
1	TIM3EN	TIM3 timer clock enable Set and cleared by software. 0: TIM3 clock disabled 1: TIM3 clock enabled
0	TIM2EN	TIM2 timer clock enable Set and cleared by software. 0: TIM2 clock disabled 1: TIM2 clock enabled

4.4.10 Backup Domain Control Register (RCC_BDCTRL)

Address offset: 0x20

Reset value: 0x0000 0000

31	Reserved										17	16
											BDSFT RST	
15	14			10	9	8	7					rw 0
RTCEN		Reserved		RTCSEL[1:0]						Reserved		

Bit Field	Name	Description
31:17	Reserved	Reserved, the reset value must be maintained.
16	BDSFTRST	Backup domain software reset Set and cleared by software. 0: Clear the reset 1: Reset the entire backup domain
15	RTCEN	RTC clock enable Set and cleared by software. 0: Disable RTC clock 1: Enable RTC clock
14:10	Reserved	Reserved, the reset value must be maintained.
9:8	RTCSEL[1:0]	RTC clock source selection Set by software to select RTC clock source. Once the RTC clock source is selected, it cannot be changed until the next backup domain is reset. These bits can be reset by setting the BDSFTRST bit. 00: No clock 01: reserved 10: LSI oscillator selected as RTC clock 11: HSE oscillator divided by 128 selected as RTC clock
7:0	Reserved	Reserved, the reset value must be maintained.

Note: the RTCSEL and RTCEN bits in the Backup Domain Control Register (RCC_BDCTRL) are in the backup domain. Therefore, these bits are write-protected after reset, and can only be changed after the PWR_CTRL.DBKP bit is set. These bits can only be cleared by the backup domain reset. Any internal or external reset will not affect these bits.

4.4.11 Clock Control/Status Register (RCC_CTRLSTS)

Address offset: 0x24

Reset value: 0x0c000003

31	30	29	28	27	26	25	24	23	22	21	20	16
LPWR RSTF	WWDG RSTF	IWDG RSTF	SFT RSTF	POR RSTF	PINRSTF	MMU RSTF	RMRSTF	Reserved	BKP EMCF		Reserved	
r 15	r	r	r	r	r	r	rw		r		2	1 0
								Reserved			LSIRD	LSIEN

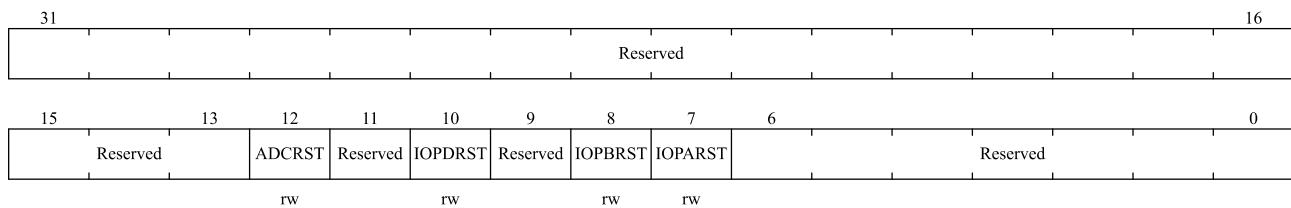
Bit Field	Name	Description
31	LPWRRSTF	<p>Low power reset flag</p> <p>Set by hardware when a low-power management reset occurs.</p> <p>Cleared by software by writing to the RMRSTF bit.</p> <p>0: No low-power management reset occurred</p> <p>1: A low-power management reset occurred</p>
30	WWDGRSTF	<p>Window watchdog reset flag</p> <p>Set by hardware when a window watchdog reset occurs.</p> <p>Cleared by software by writing to the RMRSTF bit.</p> <p>0: No windowed watchdog reset occurred</p> <p>1: Window watchdog reset occurred</p>
29	IWDGRSTF	<p>Independent watchdog reset flag</p> <p>Set by hardware when an independent watchdog reset occurs</p> <p>Cleared by software by writing to the RMRSTF bit.</p> <p>0: No independent watchdog reset occurred</p> <p>1: Independent watchdog reset occurred</p>
28	SFTRSTF	<p>Software reset flag</p> <p>Set by hardware when a software reset occurs.</p> <p>Cleared by software by writing to the RMRSTF bit.</p> <p>0: No software reset occurred</p> <p>1: Software reset occurred</p>
27	PORRSTF	<p>Power-on/power-down reset flag</p> <p>Set by hardware when a power-on/power-down reset occurs</p> <p>Cleared by software by writing to the RMRSTF bit.</p> <p>0: No power on/power off reset occurred</p> <p>1: Power-on/power-off reset occurred</p>
26	PINRSTF	<p>External pin reset flag</p> <p>Set by hardware when a reset from the NRST pin occurs.</p> <p>Cleared by software by writing to the RMRSTF bit.</p> <p>0: No NRST pin reset occurred</p> <p>1: NRST pin reset occurred</p>
25	MMURSTF	<p>MMU reset flag</p> <p>Set by hardware when MMU reset occurs.</p> <p>Cleared by software by writing to the RMRSTF bit.</p> <p>0: No MMU reset occurred</p> <p>1: MMU reset occurred</p>
24	RMRSTF	<p>Clear the reset flag</p> <p>Set by the software to clear the reset flag.</p> <p>0: No effect</p> <p>1: Clear the reset flag</p>
23:22	Reserved	Reserved, the reset value must be maintained.
21	BKPEMCF	<p>Backup domain EMC reset flag</p> <p>Set by hardware when a backup domain EMC reset occurs.</p>

Bit Field	Name	Description
		Cleared by software by writing to the RMRSTF bit. 0: No backup domain EMC reset occurred 1: Backup domain EMC reset occurred
20:2	Reserved	Reserved, the reset value must be maintained.
1	LSIRD	Internal low-speed oscillator ready Set and cleared by hardware to indicate if the internal RC 40 KHz oscillator is ready. After LSIEN is cleared, LSIRD goes low after 3 internal RC 40 KHz oscillator clock cycles. 0: Internal 40KHz RC oscillator clock not ready 1: Internal 40KHz RC oscillator clock ready
0	LSIEN	Internal low-speed oscillator enable Set and cleared by software. 0: Disable the internal RC 40 kHz oscillator 1: Enable the internal RC 40 kHz oscillator

4.4.12 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 interface reset Set and cleared by software. 0: Clear the reset 1: Reset ADC interface
11	Reserved	Reserved, the reset value must be maintained.
10	IOPDRST	IO Port D reset Set and cleared by software. 0: Clear the reset 1: Reset IO Port D Block
9	Reserved	Reserved, the reset value must be maintained.
8	IOPBRST	IO Port B reset Set and cleared by software.

Bit Field	Name	Description
		0: Clear the reset 1: Reset IO Port B Block
7	IOPARST	IO Port A reset Set and cleared by software. 0: Clear the reset 1: Reset IO Port A Block
6:0	Reserved	Reserved, the reset value must be maintained.

4.4.13 Clock Configuration Register 2 (RCC_CFG2)

Address offset: 0x2c

Reset value: 0x0000 3800

31	30	29	28									16	
Reserved	TIMCLK SEL									Reserved			
rw													
15					11	10	9	8			4	3	0
ADC1MPRES[4:0]				ADC1M SEL	Reserved	ADCPLL PRES[4:0]				ADCHPRES[3:0]			
rw				rw	rw				rw			rw	

Bit Field	Name	Description
31:30	Reserved	Reserved, the reset value must be maintained.
29	TIMCLKSEL	<p>TIM1/8 clock source selection</p> <p>Set and cleared by software.</p> <p>0: if APB2 Prescaler is 1,PCLK2 is selected as TIM1/8 clock source ;Otherwise, PCLK2 $\times 2$ is selected as TIM1/8 clock source.</p> <p>1: SYSCLK input clock is selected as TIM1/8 clock source.</p>
28:16	Reserved	Reserved, the reset value must be maintained.
15:11	ADC1MPRES[4:0]	<p>ADC 1M clock prescaler</p> <p>Set and cleared by software to configure the division factor of ADC 1M clock source.</p> <p>00000: ADC 1M clock source not divided</p> <p>00001: ADC 1M clock source divided by 2</p> <p>00010: ADC 1M clock source divided by 3</p> <p>...</p> <p>11110: ADC 1M clock source divided by 31</p> <p>11111: ADC 1M clock source divided by 32</p>
10	ADC1MSEL	<p>ADC 1M clock source selection</p> <p>Set and cleared by software.</p> <p>0: HSI oscillator clock selected as the input clock of ADC 1M</p> <p>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</i></p>

Bit Field	Name	Description
		<i>clock is turned on</i>
9	Reserved	Reserved, the reset value must be maintained.
8:4	ADCPLL PRES[4:0]	<p>ADC PLL prescaler</p> <p>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</p> <p>10000: PLL clock not divided</p> <p>10001: PLL clock divided by 2</p> <p>10010: PLL clock divided by 4</p> <p>10011: PLL clock divided by 6</p> <p>10100: PLL clock divided by 8</p> <p>10101: PLL clock divided by 10</p> <p>10110: PLL clock divided by 12</p> <p>10111: PLL clock divided by 16</p> <p>11000: PLL clock divided by 32</p> <p>11001: PLL clock divided by 64</p> <p>11010: PLL clock divided by 128</p> <p>11011: PLL clock divided by 256</p> <p>Others: PLL clock divided by 256</p>
3:0	ADCHPRES[3:0]	<p>ADC HCLK prescaler</p> <p>Set and cleared by software to configure the division factor from the HCLK clock to the ADC.</p> <p>0000: HCLK clock not divided</p> <p>0001: HCLK clock divided by 2</p> <p>0010: HCLK clock divided by 4</p> <p>0011: HCLK clock divided by 6</p> <p>0100: HCLK clock divided by 8</p> <p>0101: HCLK clock divided by 10</p> <p>0110: HCLK clock divided by 12</p> <p>0111: HCLK clock divided by 16</p> <p>1000: HCLK clock divided by 32</p> <p>Others: HCLK clock divided by 32</p>

4.4.14 Retention Domain Control Register (RCC_RDCTRL)

Address offset: 0x34

Reset value: 0x0000 0000

31	Reserved										16						
15	Reserved	11	LPTIM RST	10	Reserved	9	Reserved	7	LPTIMEN	6	Reserved	5	Reserved	3	2	LPTIMSEL[2:0]	0

Bit Field	Name	Description
31:11	Reserved	Reserved, the reset value must be maintained.
10	LPTIMERRST	LPTIMER reset Set and cleared by software. 0: Clear the reset 1: Reset LPTIMER
9:7	Reserved	Reserved, the reset value must be maintained.
6	LPTIMEN	LPTIMER Clock Enable Set and cleared by software. 0: LPTIMER Clock disabled 1: LPTIMER Clock enabled
5:3	Reserved	Reserved, the reset value must be maintained.
2:0	LPTIMSEL[2:0]	Low-power Timer clock source selection bits. This bits is set and cleared by software 000: APB clock selected as LP Timer clock 001: LSI clock selected as LP Timer clock 010: HSI clock selected as LP Timer clock Others: reserved

4.4.15 PLL and HSI Configuration Register (RCC_PLLHSIPRE)

Address offset: 0x40

Reset value: 0x0000 0003

31	Reserved										16
15	Reserved	1	0	PLLHSI PRE							

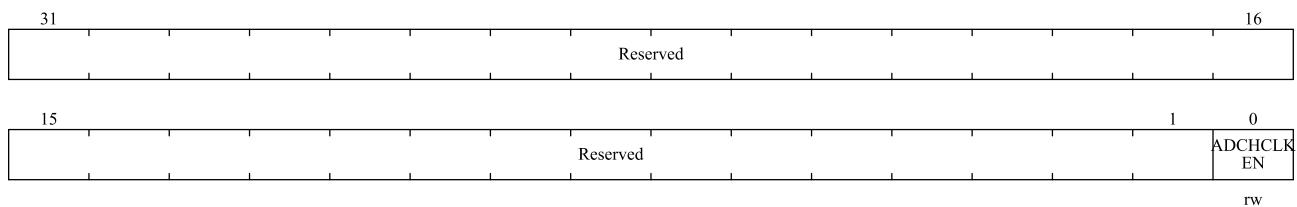
Bit Field	Name	Description
31:1	Reserved	Reserved, the reset value must be maintained.
0	PLLHSIPRE	HSI prescaler for PLL input Set and cleared by software to divide HSI before PLL entry. This bit can be written only when PLL is disabled.

Bit Field	Name	Description
		0: HSI clock not divided 1: HSI clock divided by 2

4.4.16 AHB Peripheral Clock Enable Register 1 (RCC_AHB1CLKEN)

Address offset: 0x80

Reset value: 0x0000 003E



Bit Field	Name	Description
31:1	Reserved	Reserved, the reset value must be maintained.
0	ADCHCLKEN	ADC_HCLK interface clock enable Set and cleared by software. 0: ADC_HCLK interface clock disabled 1: ADC_HCLK interface clock enabled

5 GPIO and AFIO

5.1 Summary

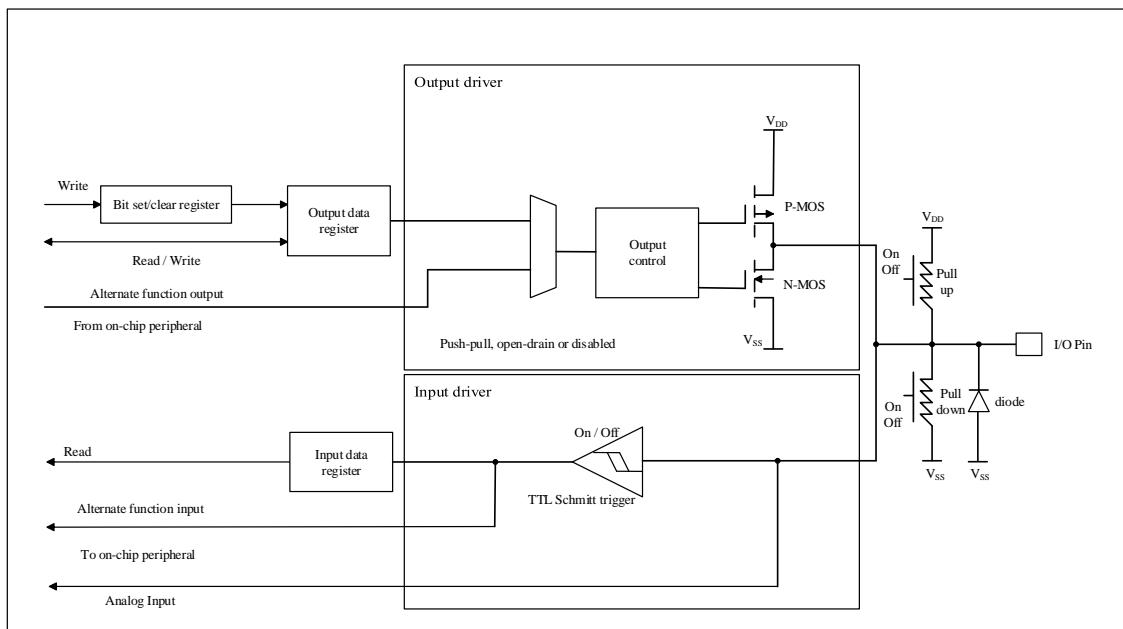
This design supports 16 GPIO, divided into 3 groups (GPIOA/GPIOB/ GPIOD). Each GPIO pin can be configured by software as output (push-pull or open drain), input (with or without pull-up or pull-down) or alternate peripheral function ports (output/input), most GPIO pins are shared with digital or analog alternate peripherals, some I/O 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 (Weak pull-up, typical 50K)
 - ◆ Input pull-down (Weak pull-down, typical 50K)
 - ◆ Analog function
 - ◆ Open drain output
 - ◆ Push-pull output
 - ◆ Push-pull alternate function
 - ◆ Open-drain alternate function
- Individual bit set or bit clear function
- All I/O supports external interrupt function
- Support software remapping I/O alternate function
- Support GPIO lock mechanism, reset the lock state to clear

Each I/O port bit is easily programmed and the I/O port register can be accessed as 32-bit word, half-word or byte since it is an AHB interface. The following figure shows the basic structure of an I/O port. The purpose of the GPIOx_PBSC and GPIOx_PBC registers is to allow atomic read/modify accesses to any of the GPIOx_POD registers. In this way, there is no risk of an IRQ occurring between the read and the modify access.

Figure 5-1 5V tolerant I/O structure



5.2 Function description

5.2.1 I/O mode configuration

The I/O port mode can be configured through the registers GPIOx_PMODE (x=A/B/D), GPIOx_POTYPE (x=A/B/D) and GPIOx_PUPD (x=A/B/D). The I/O configurations in different operation modes are 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
	0	1	Alternate function open-drain + pull-up
	1	0	Alternate function open-drain + pull-down
	1	1	Alternate function open-drain + pull-down

P MODE[1:0]	P OTYPE	P UPD[1:0]		I/O configuration
	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
11	x	0	0	Analog
	x	0	1	Reserved
	x	1	0	
	x	1	1	

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	Alternate function
Output buffer	Disabled	Enabled	Disabled	Automatic configuration from CPU according to peripheral function
Schmitt trigger	Enabled	Enabled	Disabled, Output is forced to 0(ToCore)	Automatic configuration from CPU according to peripheral function
PULL UP/DOWN/FLOATING	Configured	Disabled	Disabled	Automatic matching 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	Disabled	Can be configured,

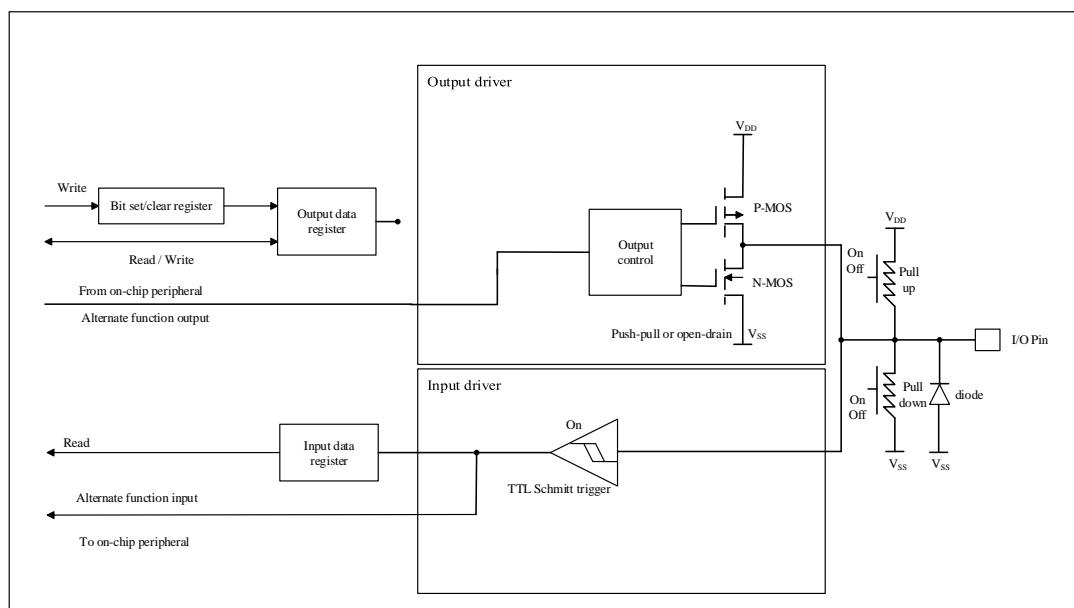
Feature	GPIO Input	GPIO Output	Analog	Alternate function
		output data is "0", GPIO output 1 when output data is "1"		GPIO output 0 when output data is "0", GPIO output 1 when output data is "1"
Input data register (I/O status)	Readable-I/O status	Readable-I/O status	Reads out 0 (SCHMFT OFF)	Readable-I/O status
Output data register(Output value)	Invalid-last written	Readable and written	Invalid-last written value	Invalid-last written value

5.2.1.1 Alternate function output mode

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

- In an open drain or push-pull configuration, the output buffer is turned on.
- Output is driven by the signal coming from the peripheral.
- The schmitt trigger input is activated.
- Weak pull-up and pull-down resistance are enabled/disabled.
- During each AHB clock cycle, the data that appears on the I/O pin is sampled into the input data register.
- In open drain mode, the I/O state can be obtained by reading the input data register.
- In push-pull mode, the read access to the output data register gets the last written value.

Figure 5-2 Alternate function mode

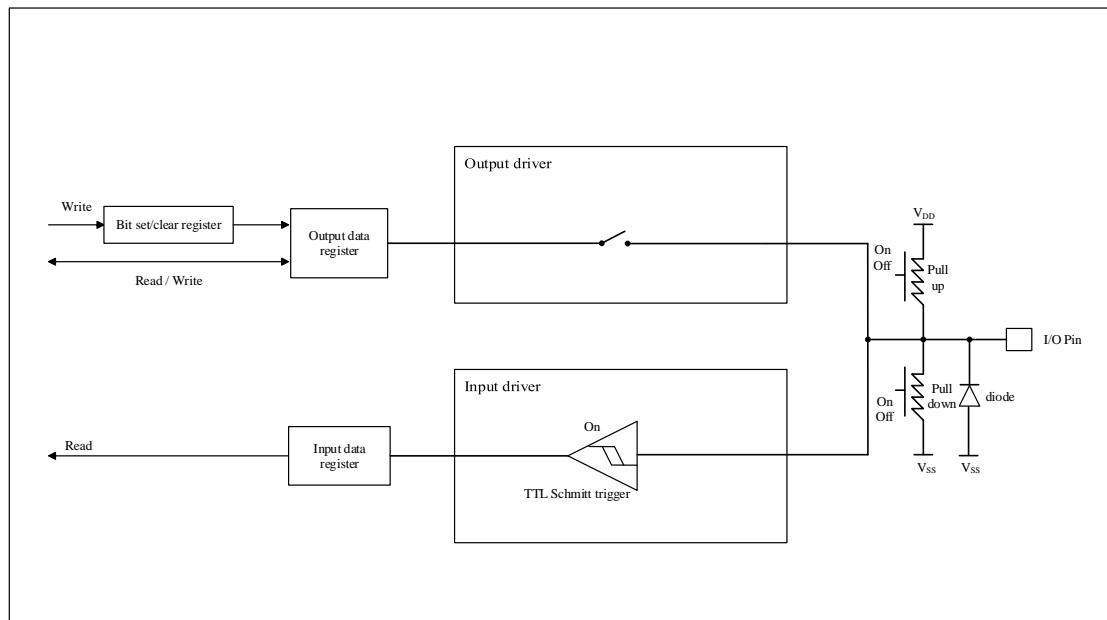


5.2.1.2 Input mode

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

- Output buffer is disabled
- The schmitt trigger input is activated.
- Depending on the input configuration (pull-up, pull-down, or floating), the weak pull-up and pull-down resistance are connected.
- The data that appears on the I/O pin is sampled to the input data register on each AHB clock
- Read access to the input data register can get I/O status

Figure 5-3 Input floating / pull-up / pull-down configuration mode.

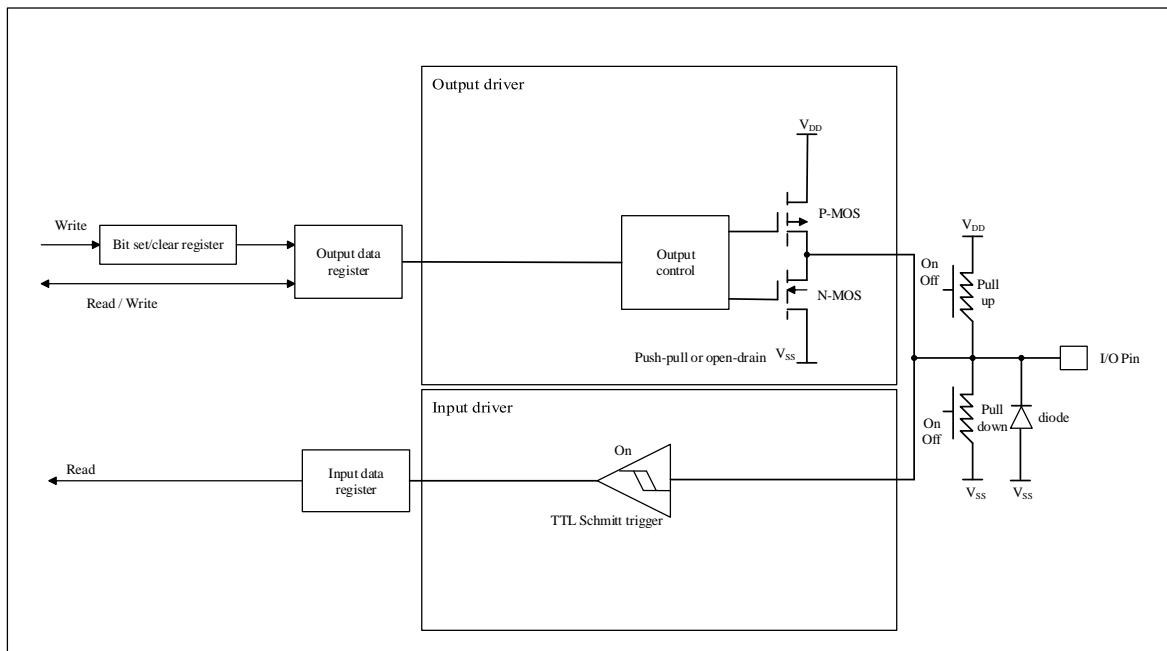


5.2.1.3 Output mode

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

- The output buffer is activated
 - Open drain mode: 0 on the output register activates NMOS, 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 NMOS, while 1 on the output register activates PMOS.
- The schmitt trigger input is activated.
- Weak pull-up and pull-down resistance are enabled/disabled.
- The data that appears on the I/O pin is sampled to the input data register on each AHB clock.
- In open drain mode, the I/O state can be obtained by reading the input data register.
- In push-pull mode, the read access to the output data register gets the last written value.

Figure 5-4 Output mode

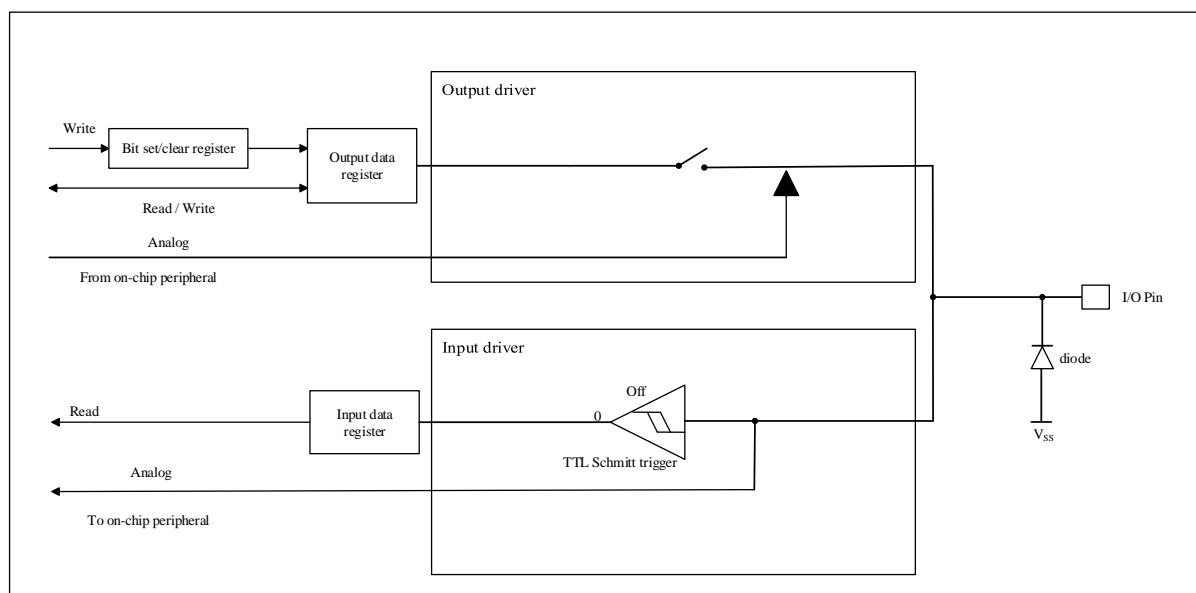


5.2.1.4 Analog mode

When the I/O port is programmed as analog mode:

- The output buffer is disabled.
- The schmitt trigger input is de-activated providing zero consumption for every analog value of the I/O pin. The output of the schmitt trigger is forced to a constant value (0).
- The weak pull-up and pull-down resistors are disabled.
- Read access to the input data register gets the value “0”.

Figure 5-5 Analog input configuration with High Impedance



5.2.2 Status after reset

During and just after reset, the alternate functions are not active, and the I/O ports are configured as analog mode (GPIOx_PMODE.PMODEy [1:0] = 11). However, with the exception of the I/O ports below.

Table 5-3 special pins after reset

TSSOP20	Pin name	Type(1)	Mode	Comments
19	PA13	I/O	Input pull-up	SWDIO
20	PA14	I/O	Input pull-down	SWCLK
1	BOOT0/PD0	I/O	Input pull-down	BOOT0

- BOOT0/PD0 input pull-down by default
- NRST default non-GPIO function, analog pin, no digital control
- After the reset, the default state of the pin associated with the debug system is to enable the SWD and the SWD pin is placed on the input pull-up or pull-down mode
 - PA14: SWDIO input pull-down
 - PA13: SWCLK input pull-up

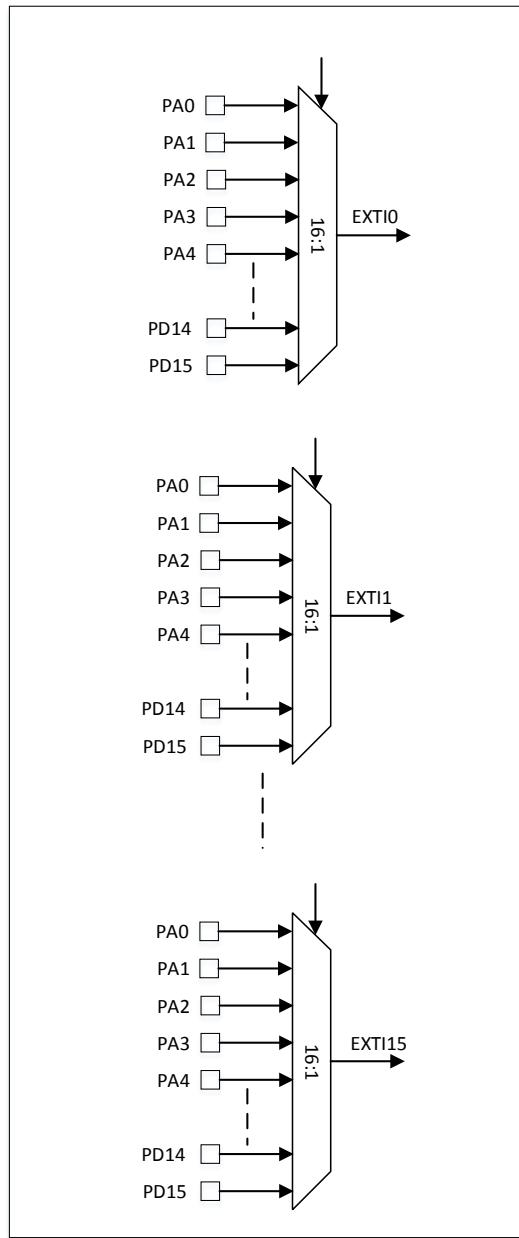
5.2.3 Atomic bit set and reset

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 AHB write operation.

5.2.4 External Interrupt /wakeup line

- All ports have external interrupt capabilities that can be configured in the EXTI module.
- In order to use an external interrupt line, the port must be configured in input mode.
- All ports can be configured for SLEEP/STOP2 and STOP0 mode wake-up function through positive or negative edge triggered interrupts.
- All the I/O ports are connected to 16 external interrupt/event lines as shown Figure 5-6 , and the corresponding control is configured by the register AFIO_EXTI_CFGx.

Figure 5-6 EXTI global interrupts



5.2.5 Alternate function

When I/O ports are configured for alternate function mode, the port bit configuration register (GPIOx_AFL/GPIOx_AFH) and output type register (GPIOx_POTYPE configuration push-pull or open-drain) must be programmed before use, alternate input or output is determined by the peripheral.

5.2.5.1 Clock output MCO

The microcontroller allows the output of clock signals to the external MCO pin (PA9). The corresponding GPIO port register must be configured in alternate function mode. The following fiveclock signals can be selected as MCO clocks, and the clock selection is controlled by the clock configuration register RCC_CFG.MCO[26:23] bits:

- LSI
- SYSCLK
- HSI
- HSE
- PLL divided clock

5.2.5.2 RTC-IOM interface

The RTC and IOM interface supports the following functions:

- Tamper: PA0
- Time-stamp – Any port that is configured as EXTI line can be configured for RTC-Timestamp.

5.2.5.3 HSE pins used as GPIO ports

OSC_IN and OSC_OUT of HSE are mapped to PD14 and PD15 respectively. If HSE is off, the corresponding pin can be used as GPIO. If HSE is on, the corresponding pin goes into analog mode and bypasses the GPIO configuration.

The crystal oscillator is configured as user external clock mode, the pin remains as clock input, and OSC_OUT can still be used as normal GPIO.

5.2.5.4 SWD alternate function

The chip power-on default enables the SWD debug interface, and the interface signal is mapped to the GPIO port, as shown in the following table.

Table 5-4 Debug interface signal

Debug signal	GPIO
SWDIO	PA13
SWCLK	PA14

If you need to use its GPIO function during debugging, you can change the above remapping configuration by setting the alternate remapping and debugging I/O configuration registers (GPIOx_AFL or GPIOx_AFH). See the table below.

Table 5-5 Debug port image

Possible debug ports for	SWD I/O pin allocation	
	PA13/ SWDIO	PA14/ SWCLK
Enable SW-DP.	I/O is not available	I/O is not available
Turn off SW-DP.	I/O available	I/O available

Internal pull-up and pull-down on SWD pins:

It is necessary to ensure that the SWD input pins are not floating since they are directly connected to flip-flops to

control the debug mode features. Special care must be taken with. The SWCLK pin which is directly connected to the clock of some of these flip-flops.

To avoid any uncontrolled I/O levels, the device embeds internal pull-ups and pull-downs on the SWD input pins:

- SWDIO: internal pull-up
- SWCLK: internal pull-down

5.2.5.5 ADC external trigger alternate function remapping

The external trigger source of injection conversion and regular conversion of ADC supports remapping. See alternate remapping and debug I/O configuration register (AFIO_RMP_CFG).

Table 5-6 ADC external trigger injected conversion alternate function remapping

Alternate function	ADC_ETRI = 0	ADC_ETRI = 1
ADC external trigger injected conversion	The ADC external trigger injected conversion is connected to EXTI (0-15).	The ADC external event injected conversion is connected to TIM8_Channel4

Table 5-7 ADC external trigger regular conversion alternate function remapping

Alternate function	ADC_ETRR = 0	ADC_ETRR = 1
ADC external trigger regular conversion	The ADC external trigger regular conversion is connected to EXTI (0-15).	The ADC external event regular conversion is connected to TIM8_TRGO.

5.2.5.6 TIMx alternate function remapping

5.2.5.6.1 TIM1 alternate function remapping

Table 5-8 TIM1 alternate function remapping

Alternate function	Pin	Remap
TIM1_CH1	PA0	AF10
TIM1_CH2	PA9	AF3
TIM1_CH3	PA10	AF3
TIM1_CH4	PA7	AF2
TIM1_BKIN	PA10	AF2
	PA6	AF6
TIM1_CH1N	PA1	AF2
	PA7	AF6
TIM1_CH2N	PA6	AF2
	PD15	AF8
TIM1_CH3N	PB1	AF6
	PA9	AF2
	PD14	AF2

5.2.5.6.2 TIM2 alternate function remapping

Table 5-9 TIM2 alternate function remapping

Alternate function	Pin	Remap

TIM2_ETR	PA0	AF6
TIM2_CH1	PA0	AF3
TIM2_CH2	PA1	AF3
TIM2_CH3	PA2	AF3

5.2.5.6.3 TIM3 alternate function remapping

Table 5-10 TIM3 alternate function remapping

Alternate function	Pin	Remap
TIM3_CH1	PA6	AF3
TIM3_CH2	PA7	AF3
TIM3_CH4	PB1	AF3

5.2.5.6.4 TIM4 alternate function remapping

Table 5-11 TIM4 alternate function remapping

Alternate function	Pin	Remap
TIM4_CH1	PA7	AF9
TIM4_CH2	PB1	AF10
TIM4_CH3	PA4	AF6
TIM4_CH4	PA5	AF6

5.2.5.6.5 TIM5 alternate function remapping

Table 5-12 TIM5 alternate function remapping

Alternate function	Pin	Remap
TIM5_CH1	PA0	AF2
TIM5_CH2	PA1	AF8
TIM5_CH3	PA2	AF7
TIM5_CH4	PA3	AF8

5.2.5.6.6 TIM8 alternate function remapping

Table 5-13 TIM8 alternate function remapping

Alternate function	Pin	Remap
TIM8_ETR	PA0	AF8
TIM8_CH1	PA2	AF9
TIM8_CH2	PA3	AF10
TIM8_CH3	PA4	AF9
	PA6	AF9
TIM8_CH4	PA5	AF9
TIM8_BKIN	PA6	AF7
	PA2	AF10
	PA9	AF10
TIM8_CH1N	PA7	AF7
TIM8_CH3N	PB1	AF1

5.2.5.6.7 LPTIM alternate function remapping

Table 5-14 LPTIM alternate function remapping

Alternate function	Pin	Remap
LPTIM_IN1	PA5	AF12
LPTIM_IN2	PA9	AF12
LPTIM_OUT	PA4	AF12
LPTIM_ETR	PA10	AF12

5.2.5.7 USARTx alternate function remapping

5.2.5.7.1 USART2 alternate function remapping

Table 5-15 USART2 alternate function remapping

Alternate function	Pin	Remap
USART2_CTS	PA0	AF5
USART2_RTS	PA1	AF5
USART2_TX	PA2	AF5
	PA6	AF11
	PD14	AF4
USART2_RX	PA3	AF5
	PA7	AF11
	PD15	AF4
USART2_CK	PA4	AF5
	PA14	AF5

5.2.5.8 UARTx alternate function remapping

5.2.5.8.1 UART1 alternate function remapping

Table 5-16 UART1 alternate function remapping

Alternate function	Pin	Remap
UART1_TX	PA4	AF2
	PA9	AF5
UART1_RX	PA5	AF5
	PA10	AF5

5.2.5.8.2 UART4 alternate function remapping

Table 5-17 UART4 alternate function remapping

Alternate function	Pin	Remap
UART4_TX	PA0	AF11
UART4_RX	PB1	AF7

5.2.5.9 I2C alternate function remapping

5.2.5.9.1 I2C1 alternate function remapping

Table 5-18 I2C1 alternate function remapping

Alternate function	Pin	Remap
I2C1_SCL	PA4	AF8
I2C1_SDA	PA5	AF8
	PA14	AF8

5.2.5.9.2 I2C2 alternate function remapping

Table 5-19 I2C2 alternate function remapping

Alternate function	Pin	Remap
I2C2_SCL	PA3	AF6
	PA9	AF7
	PD15	AF6
I2C2_SDA	PA2	AF6
	PA10	AF7
	PD14	AF6

5.2.5.10 SPI/I2S alternate function remapping

5.2.5.10.1 SPI1/I2S1 alternate function remapping

Table 5-20 SPI1/I2S1 alternate function remapping

Alternate function	Pin	Remap
SPI1_I2S1_NSS_WS	PA1	AF1
	PA4	AF1
SPI1_I2S1_SCK_CK	PA5	AF1
	PA10	AF1
	PB1	AF5
SPI1_I2S1_MISO_MCK	PA0	AF1
	PA6	AF1
SPI1_I2S1_MOSI_SD	PA7	AF1

5.2.5.10.2 SPI2/I2S2 alternate function remapping

Table 5-21 SPI2/I2S2 alternate function remapping

Alternate function	Pin	Remap
SPI2_I2S2_NSS_WS	PA4	AF3
	PA13	AF6
SPI2_I2S2_SCK_CK	PA10	AF6
SPI2_I2S2_MISO_MCK	PA9	AF1
	PA13	AF1
SPI2_I2S2_MOSI_SD	PA14	AF1
	PB1	AF2

5.2.5.11 EVENTOUT alternate function remapping

Table 5-22 EVENTOUT alternate function remapping

Alternate function	Pin	Remap
EVENTOUT	PA0~PA14	AF4

5.2.5.12 BEEPER alternate function remapping

Table 5-23 BEEPER alternate function remapping

Alternate function	Pin	Remap
BEEPER_OUT_P	PA6	AF12
BEEPER_OUT_N	PA7	AF12

5.2.5.13 SWD alternate function remapping

Table 5-24 SWD alternate function remapping

Alternate function	Pin	Remap
SWDIO	PA13	AF0
SWCLK	PA14	AF0

5.2.5.14 RTC_REFIN alternate function remapping

Table 5-25 RTC_REFIN alternate function remapping

Alternate function	Pin	Remap
RTC_REFIN	PA10	AF9

5.2.5.15 MCO alternate function remapping

Table 5-26 MCO alternate function remapping

Alternate function	Pin	Remap
MCO	PA9	AF9

5.2.6 I/O configuration of peripherals

Table 5-27 ADC

ADC pin	GPIO configuration
ADC	Analog mode

Table 5-28 TIM1/TIM8

TIM pin	Configuration	PIN 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
TIM8_ETR	External trigger clock input	Alternate function push-pull

Table 5-29 TIM2/3/4/5

TIM2/3/4/5 pin	Configuration	PIN configuration mode
TIM2/3/4/5_CHx	Input capture channel x	Alternate function push-pull
	Output compare channel x	Alternate function push-pull
TIM2_ETR	External trigger clock input	Alternate function push-pull

Table 5-30 LPTIM

LPTIM pin	Configuration	PIN configuration mode
LPTIM_INx	Digital input	Alternate function push-pull
LPTIM_OUT	Digital input	Alternate function push-pull
LPTIM_ETR	Digital input	Alternate function push-pull

Table 5-31 USART

USART pin	Configuration	GPIO configuration
USARTx_TX	Full duplex transmissions	Alternate function push-pull
	Half duplex synchronous mode	Alternate function push-pull
USARTx_RX	Full duplex transmissions	Alternate function push-pull
	Half duplex synchronous mode - no input	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

Table 5-32 UART

UART pin	Configuration	GPIO configuration
UARTx_TX	Full duplex transmissions	Alternate function push-pull
	Half duplex synchronous mode	Alternate function push-pull
UARTx_RX	Full duplex transmissions	Alternate function push-pull
	Half duplex synchronous mode - no input	Unused, can be used as general I/O.

Table 5-33 I2C

I2C pin	Configuration	GPIO configuration
I2Cx_SCL	SCL-IN Digital Input	Alternate function open-drain
	SCL-OUT Digital output	Alternate function open-drain
I2Cx_SDA	SDA-IN Digital Input	Alternate function open-drain
	SDA-OUT Digital output	Alternate function open-drain

Table 5-34 SPI-I2S

SPI-I2S pin	Configuration	GPIO configuration
SPIx_I2Sx_MISO_MCK	Master mode	Alternate function push-pull
	Slave mode	Alternate function push-pull
SPIx_I2Sx_MOSI_SD	Master mode	Alternate function push-pull
	Slave mode	Alternate function push-pull
SPIx_I2Sx_NSS_WS	Master mode	Alternate function push-pull
	Slave mode	Alternate function push-pull
SPIx_I2Sx_SCK_CK	Master mode	Alternate function push-pull
	Slave mode	Alternate function push-pull

Table 5-35 SWD

SWD pin	Configuration	GPIO configuration
SWDIO	Input Pull-up	Alternate function push-pull + pull up
SWCLK	Input Pull-down	Alternate function push-pull + pull down

Table 5-36 RTC

RTC pin	Configuration	GPIO configuration
RTC_REFIN	Digital input	Alternate function push-pull

Table 5-37 EVENT_OUT

EVENT_OUT pin	Configuration	GPIO configuration
EVENT_OUT	Digital output	Alternate function push-pull

Table 5-38 Other

Other pin	Configuration	GPIO configuration
MCO	Digital output	Alternate function push-pull
EXTI lines	Digital input	Input Floating/Pull-up/Pull-down

5.2.7 GPIO Locking mechanism

- The locking mechanism allows to freeze contents of I/O configuration (GPIOx_PMODE, GPIOx_POTYPE, GPIOx_PUPD, GPIOx_DS, and GPIOx_SR) and alternate function registers (GPIOx_AFL and GPIOx_AFH). When the lock program is executed on one port bit, the configuration of that port bit will no longer be changed until the next reset, referring to the port configuration lock register GPIOx_PLOCK.
- PLOCKK, that is, GPIOx_PLOCK [16], becomes 1 only after the correct sequence w1-> w0-> w1-> r0 (r0 here

is also a must). After that, it becomes 0 only if the system reset is performed. GPIOx_PLOCK.PLOCK[15:0] can only be modified at GPIOx_PLOCK.PLOCKK=0.

- The lock sequence to set GPIOx_PLOCK.PLOCKK bit, w1-> w0-> w1-> r0 will be valid only if the value (1 or 0) in GPIOx_PLOCK.PLOCK [15:0] does not change during this sequence. The GPIOx_PLOCK.PLOCKK bit will not be set if the value in GPIOx_PLOCK.PLOCK [15:0] changes during this sequence.
- As long as GPIOx_PLOCK.PLOCKK=0 and GPIOx_PLOCK.PLOCKx=0 or 1, all configuration and alternate function bits can be modified. When GPIOx_PLOCK.PLOCKK=1 but GPIOx_PLOCK.PLOCK[x]=0, the corresponding configuration and alternate function bits corresponding to GPIOx_PLOCK.PLOCK[x]=0 can be modified.
- Only when GPIOx_PLOCK.PLOCKK=1 and GPIOx_PLOCK.PLOCK[x]=1, the configurations corresponding to GPIOx_PLOCK.PLOCK[x]=1 are locked and can not be modified.
- If the lock sequence operation is wrong, then it must be redone (w1-> w0-> w1-> r0) to initiate the lock operation again.

5.2.8 Precautions for using GPIO

- When VDD and VDDA are not powered on, the voltage applied to GPIO must not exceed 3.6V;
- When the voltage applied to GPIO is 5.5V, VDD and VDDA must not be lower than 2.4V;
- If you do not want the MCU have leakage, you need to ensure that the voltage applied to the GPIO is less than or equal to VDD and VDDA;
- When the voltage applied on GPIO is greater than VDD and VDDA, if you want to reduce the leakage current of MCU, you need to connect a resistor in series with GPIO.

5.3 GPIO Registers

The GPIO registers are accessible through AHB bus. The register description is as follows.

The GPIO port base address is as follows:

GPIOA base address: 0x40023400

GPIOB base address: 0x40023800

GPIOD base address: 0x40024000

5.3.1 GPIOA register overview

Table 5-39 GPIOA 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	GPIOA_PMODE	PMODE15		PMODE14		PMODE13		PMODE12		PMODE11		PMODE10		PMODE9		PMODE8		PMODE7		PMODE6		PMODE5		PMODE4		PMODE3		PMODE2		PMODE1		PMODE0	
	Reset value	1	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

0x1C	GPIOA_PLOCK	Reset value	0x08	0x04	Register	GPIOA_POTYPE	Reset value	31
								30
0x18	GPIOA_PBSC	Reset value	0x0C	0x08	0x04	GPIOA_SR	Reset value	29
								28
0x10	GPIOA_PID	Reset value	0x14	0x0C	0x08	0x04	GPIOA_PUPD	27
								26
0x28	GPIOA_PBC	Reset value	0x18	0x0C	0x08	0x04	GPIOA_POD	25
								24
0x1C	GPIOA_PLOCK	Reset value	0x28	0x18	0x0C	0x08	0x04	23
								22
0x1C	GPIOA_PLOCK	Reset value	0x1C	0x1C	0x1C	0x1C	0x1C	21
								20
0x1C	GPIOA_PLOCK	Reset value	0x1C	0x1C	0x1C	0x1C	0x1C	19
								18
0x1C	GPIOA_PLOCK	Reset value	0x1C	0x1C	0x1C	0x1C	0x1C	17
								16
0x1C	GPIOA_PLOCK	Reset value	0x1C	0x1C	0x1C	0x1C	0x1C	15
								14
0x1C	GPIOA_PLOCK	Reset value	0x1C	0x1C	0x1C	0x1C	0x1C	13
								12
0x1C	GPIOA_PLOCK	Reset value	0x1C	0x1C	0x1C	0x1C	0x1C	11
								10
0x1C	GPIOA_PLOCK	Reset value	0x1C	0x1C	0x1C	0x1C	0x1C	9
								8
0x1C	GPIOA_PLOCK	Reset value	0x1C	0x1C	0x1C	0x1C	0x1C	7
								6
0x1C	GPIOA_PLOCK	Reset value	0x1C	0x1C	0x1C	0x1C	0x1C	5
								4
0x1C	GPIOA_PLOCK	Reset value	0x1C	0x1C	0x1C	0x1C	0x1C	3
								2
0x1C	GPIOA_PLOCK	Reset value	0x1C	0x1C	0x1C	0x1C	0x1C	1
								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	
0x20	GPIOA_AFL	0	0	AFSEL7																														
		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	
0x24	GPIOA_AFH	0	0	AFSEL15																														
		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	
0x2C	GPIOA_DS	0	1	DS15																														
		Reset value	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	

5.3.2 GPIOB register overview

Table 5-40 GPIOB 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	GPIOB_PMODE	1	1	PMODE15	31																													
		Reset value	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	
0x04	GPIOB_POTYPE	1	1	PMD14	30																													
		Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0x08	GPIOB_SR	1	1	PMD13	29																													
		Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0x0C	GPIOB_PUPD	1	1	PUPD12	28																													
		Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

Offset	Register	GPIOB_PID		Reset value	31
		0	0		30
0x10	Reset value	GPIOB_POD		29	29
		0	0		28
0x14	Reset value	GPIOB_PBSC		27	27
		0	0		26
0x18	Reset value	GPIOB_PBC		25	25
		0	0		24
0x28	Reset value	GPIOB_PLOCK		23	23
		0	0		22
0x1C	Reset value	GPIOB_AFL		21	21
		0	0		20
0x20	Reset value	GPIOB_AFH		19	19
		0	0		18
0x24	Reset value	GPIOB_DS		17	17
		0	0		16
0x2C	Reset value	DS15	0	AFSEL7	0
		0	0	AFSEL15	0
0x2D	Reset value	DS14	0	AFSEL0	0
		0	0	AFSEL14	0
0x2E	Reset value	DS13	0	AFSEL6	0
		0	0	AFSEL13	0
0x2F	Reset value	DS12	0	AFSEL5	0
		0	0	AFSEL12	0
0x30	Reset value	DS11	0	AFSEL4	0
		0	0	PLOCKK	0
0x34	Reset value	DS10	0	AFSEL3	0
		0	0	AFSEL11	0
0x35	Reset value	DS9	0	AFSEL2	0
		0	0	AFSEL10	0
0x36	Reset value	DS8	0	AFSEL1	0
		0	0	AFSEL9	0
0x37	Reset value	DS7	0	AFSEL0	0
		0	0	AFSEL8	0
0x38	Reset value	DS6	0	AFSEL1	0
		0	0	AFSEL0	0
0x39	Reset value	DS5	0	AFSEL1	0
		0	0	AFSEL0	0
0x3A	Reset value	DS4	0	AFSEL1	0
		0	0	AFSEL0	0
0x3B	Reset value	DS3	0	AFSEL1	0
		0	0	AFSEL0	0
0x3C	Reset value	DS2	0	AFSEL1	0
		0	0	AFSEL0	0
0x3D	Reset value	DS1	0	AFSEL1	0
		0	0	AFSEL0	0
0x3E	Reset value	DS0	0	AFSEL1	0
		0	0	AFSEL0	0

5.3.3 GPIOD register overview

Table 5-41 GPIOD register overview

Offset	Register	Reset value	GPIOD_PMODE	PMODE15	31
0x00				1	30
0x04	GPIOD_POTYPE			1	29
0x08	GPIOD_SR			1	28
0x0C	GPIOD_PUPD			1	27
0x10	GPIOD_PID			1	26
0x14	GPIOD_POD	Reset value		1	25
0x18	GPIOD_PBSC	Reset value		1	24
	PBC15		PUPD15	0	23
	PBC14		PUPD14	0	22
	PBC13		PUPD13	0	21
	PBC12		PUPD12	0	20
		Reserved	Reserved		19
		Reserved	Reserved		18
					17
					16
	PBC0			1	15
	PBS15	0	POD15	0	14
	PBS14	0	POD14	0	13
	PBS13	0	POD13	0	12
	PBS12	0	POD12	0	11
		Reserved	Reserved		10
		Reserved	Reserved		9
					8
	PBS0	0	POD0	0	7
			PUPD0	1	6
			POT0	0	5
				0	4
					3
					2
					1
					0

5.3.4 GPIO port mode description register (GPIOx_PMODE)

Address offset : 0x00

Reset value(Port A) : 0xABFFFFFF

Reset value(Port B) : 0xFFFFFEBF

Reset value(Port D) : 0xFF000000

Bit field	Name	Description
31:30	PMODEy[1:0]	Mode bits for port x (y = 0...15) 00: Input mode 01: General purpose output mode 10: Alternate function mode 11: Analog function mode
29:28		
27:26		
25:24		
23:22		
21:20		
19:18		<i>Note: For GPIOA, y=0/1/2/3/4/5/6/7/9/10/13/14</i>
17:16		<i>For GPIOB, y=1</i>
15:14		<i>For GPIOD, y=0/14/15</i>
13:12		
11:10		
9:8		
7:6		
5:4		
3:2		
1:0		

5.3.5 GPIO port type definition (GPIOx_POTYPE)

Address offset : 0x04

Reset value (Port A) : 0x00000000

Reset value (Port B) : 0x00000000

Reset value (Port D) : 0x00000000

31																																				16														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	POT15	POT14	POT13	POT12	POT11	POT10	POT9	POT8	POT7	POT6	POT5	POT4	POT3	POT2	POT1	POT0	rw																		

Bit field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained.
15:0	POTy	Output mode bits for port x (y = 0...15) 0: Output push-pull mode 1: Output open-drain mode <i>Note: For GPIOA, y=0/1/2/3/4/5/6/7/9/10/13/14</i> <i>For GPIOB, y=1</i> <i>For GPIOD, y=0/14/15</i>

5.3.6 GPIO slew rate configuration register (GPIOx_SR)

Address offset : 0x08

Reset value (Port A) : 0x0000FFFF

Reset value (Port B) : 0x0000FFFF

Reset value (Port D) : 0x0000F001

Reserved															
31	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

Bit field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained.
15:0	SRy	<p>Slew rate configuration bits y for port GPIOx (y = 0...15):</p> <p>0 : Fast slew rate</p> <p>1 : Slow slew rate</p> <p><i>Note: For GPIOA, y=0/1/2/3/4/5/6/7/9/10/13/14</i></p> <p><i>For GPIOB, y=1</i></p> <p><i>For GPIOD, y=0/14/15</i></p>

5.3.7 GPIO port pull-up/pull-down description register (GPIOx_PUPD)

Address offset : 0x0C

Reset value (Port A) : 0x64000000

Reset value (Port B) : 0x000000100

Reset value (Port D) : 0x000000002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0]	PUPD14[1:0]	PUPD13[1:0]	PUPD12[1:0]	PUPD11[1:0]	PUPD10[1:0]	PUPD9[1:0]	PUPD8[1:0]								
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

PUPD7[1:0] PUPD6[1:0] PUPD5[1:0] PUPD4[1:0] PUPD3[1:0] PUPD2[1:0] PUPD1[1:0] PUPD0[1:0]

rw rw rw rw rw rw rw rw

Bit field	Name	Description
31:30	PUPDy[1:0]	Mode bits for port x (y = 0...15)
29:28		00: No pull-up, pull-down
27:26		01: Pull-up
25:24		10: Pull-down

Bit field	Name	Description
23:22		11: Reserved
21:20		
19:18		<i>Note: For GPIOA, y=0/1/2/3/4/5/6/7/9/10/13/14</i>
17:16		<i>For GPIOB, y=1</i>
15:14		<i>For GPIOD, y=0/14/15</i>
13:12		
11:10		
9:8		
7:6		
5:4		
3:2		
1:0		

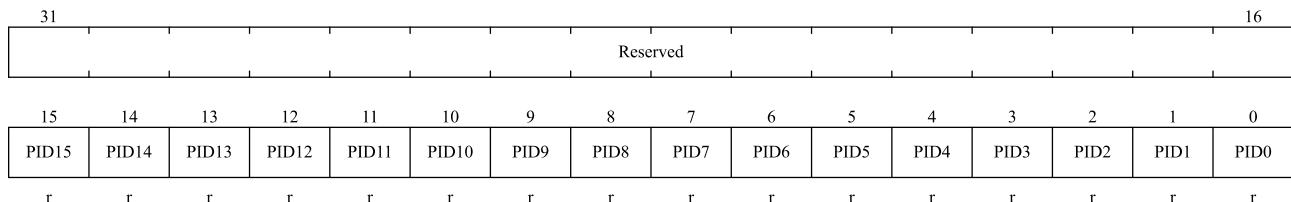
5.3.8 GPIO port input data register (GPIOx_PID)

Address offset : 0x10

Reset value (Port A) : 0x00000000

Reset value (Port B) : 0x00000000

Reset value (Port D) : 0x00000000



Bit field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained.
15:0	PIDy	Port input data (y = 0...15) These bits are read-only. They contain the input value of the corresponding I/O port. <i>Note: For GPIOA, y=0/1/2/3/4/5/6/7/9/10/13/14</i> <i>For GPIOB, y=1</i> <i>For GPIOD, y=0/14/15</i>

5.3.9 GPIO port output data register (GPIOx POD)

Address offset : 0x14

Reset value (Port A) : 0x00000000

Reset value (Port B) : 0x00000000

Reset value (Port D) : 0x00000000

31																16
Reserved																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
POD15	POD14	POD13	POD12	POD11	POD10	POD9	POD8	POD7	POD6	POD5	POD4	POD3	POD2	POD1	POD0	
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	PODy	<p>Port output data (y = 0...15)</p> <p>These bits can be read and written by software. Port output data, the corresponding POD bits can be independently set/cleared by GPIOx_PBSC (x = A...D) register.</p> <p><i>Note: For GPIOA, y=0/1/2/3/4/5/6/7/9/10/13/14</i></p> <p><i>For GPIOB, y=1</i></p> <p><i>For GPIOD, y=0/14/15</i></p>

5.3.10 GPIO port bit set/clear register (GPIOx_PBSC)

Address offset : 0x18

Reset value (Port A) : 0x00000000

Reset value (Port B) : 0x00000000

Reset value (Port D) : 0x00000000

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 (y = 0...15)</p> <p>These bits are write-only and can be accessed in word mode only.</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>
15:0	PBSy	<p>Set bit y of port GPIOx (y = 0...15)</p> <p>These bits are write-only and can be accessed in word mode only.</p> <p>0: Does not affect the corresponding PODy bit</p> <p>1: Set the corresponding PODy bit to 1</p> <p><i>Note: For GPIOA, y=0/1/2/3/4/5/6/7/9/10/13/14</i></p>

Bit field	Name	Description
		<i>For GPIOB, y=1</i> <i>For GPIOD, y=0/14/15</i>

5.3.11 GPIO port bit clear register (GPIOx_PBC)

Address offset : 0x28

Reset value (Port A) : 0x00000000

Reset value (Port B) : 0x00000000

Reset value (Port D) : 0x00000000

31	Reserved															16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
PBC15	PBC14	PBC13	PBC12	PBC11	PBC10	PBC9	PBC8	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0	
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained.
15:0	PBCy	<p>Clear bit y of port GPIOx (y = 0...15)</p> <p>These bits are write-only and can be accessed in word mode only.</p> <p>0: Does not affect the corresponding PODy bit</p> <p>1: Clear the corresponding PODy bit to 0</p> <p><i>Note: For GPIOA, y=0/1/2/3/4/5/6/7/9/10/13/14</i> <i>For GPIOB, y=1</i> <i>For GPIOD, y=0/14/15</i></p>

5.3.12 GPIO port configuration lock register (GPIOx_PLOCK)

Address offset : 0x1C

Reset value (Port A) : 0x00000000

Reset value (Port B) : 0x00000000

Reset value (Port D) : 0x00000000

31	Reserved															17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
PLOCK15	PLOCK14	PLOCK13	PLOCK12	PLOCK11	PLOCK10	PLOCK9	PLOCK8	PLOCK7	PLOCK6	PLOCK5	PLOCK4	PLOCK3	PLOCK2	PLOCK1	PLOCK0		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

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 anytime. It can only be modified using the lock key writing sequence.</p> <p>0: Port configuration lock key not active</p> <p>1: Port configuration lock key active. GPIOx_PLOCK register is locked until an MCU reset occurs.</p> <p>Lock key writing sequence:</p> <p>Write 1 -> write 0 -> write 1 -> read 0 -> read 1</p> <p>The last reading can be omitted, but it can be used to confirm that the lock key has been activated.</p> <p><i>Note: During the lock key writing sequence, the value of PLOCK[15:0] must not change. Any error in the lock sequence will abort the lock.</i></p>
15:0	PLOCKy	<p>Configuration lock bit y of port GPIOx (y = 0...15)</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: For GPIOA, y=0/1/2/3/4/5/6/7/9/10/13/14</i></p> <p><i>For GPIOB, y=1</i></p> <p><i>For GPIOD, y=0/14/15</i></p>

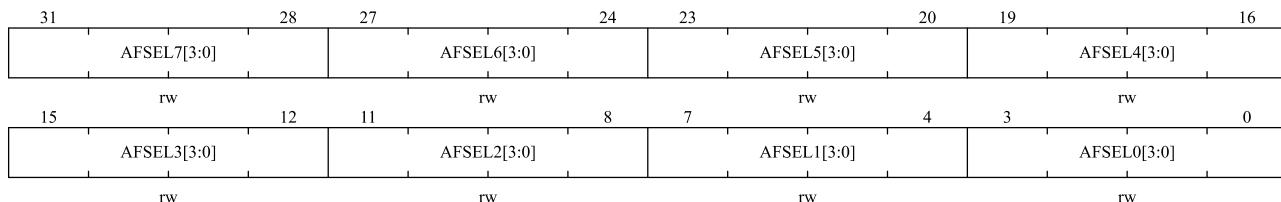
5.3.13 GPIO Alternate function low register (GPIOx_AFL)

Address offset : 0x20

Reset value (Port A) : 0x00000000

Reset value (Port B) : 0x00000000

Reset value (Port D) : 0x00000000



Bit field	Name	Description
31:28	AFSELy[3:0]	Alternate function configuration bits y for port GPIOx (y = 0...7)
27:24		0000: AF0
23:20		0001: AF1
19:16		0010: AF2

Bit field	Name	Description
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
		<p><i>Note: For GPIOA, y=0/1/2/3/4/5/6/7</i></p> <p><i>For GPIOB, y=1</i></p> <p><i>For GPIOD, y=0</i></p>

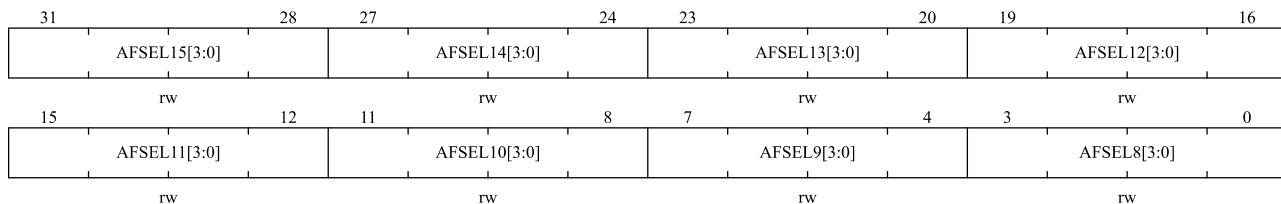
5.3.14 GPIO Alternate function high register (GPIOx_AFH)

Address offset : 0x24

Reset value (Port A) : 0x00000000

Reset value (Port B) : 0x00000000

Reset value (Port D) : 0x00000000



Bit field	Name	Description
31:28	AFSELy[3:0]	Alternate function configuration bits y for port GPIOx (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

Bit field	Name	Description
		<p>1000: AF8 1001: AF9 1010: AF10 1011: AF11 1100: AF12 1101: AF13 1110: AF14 1111: AF15</p> <p><i>Note: For GPIOA, y=9/10/13/14 For GPIOD, y=14/15</i></p>

5.3.15 GPIO driver strength configuration register (GPIOx_DS)

Address offset : 0x2C

Reset value (Port A) : 0x55555555

Reset value (Port B) : 0x55555555

Reset value (Port D) : 0x55000001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DS15[1:0]	DS14[1:0]	DS13[1:0]	DS12[1:0]	DS11[1:0]	DS10[1:0]	DS9[1:0]	DS8[1:0]								
15	rw	14	rw	13	rw	12	rw	11	rw	10	rw	9	rw	8	rw
DS7[1:0]	DS6[1:0]	DS5[1:0]	DS4[1:0]	DS3[1:0]	DS2[1:0]	DS1[1:0]	DS0[1:0]								
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit field	Name	Description
31:30	DSy[1:0]	Port GPIOx drive capability configuration bits y (y = 0...15)
29:28		00 : 2mA driver strength
27:26		10 : 4mA driver strength
25:24		01 : 8mA driver strength
23:22		11 : 12mA driver strength
21:20		
19:18		<i>Note: For GPIOA, y= 0/1/2/3/4/5/6/7/9/10/13/14</i>
17:16		<i>For GPIOB, y=1</i>
15:14		<i>For GPIOD, y=0/14/15</i>
13:12		
11:10		
9:8		
7:6		
5:4		
3:2		

Bit field	Name	Description
1:0		

Note: For Port C, only DS13, DS14 and DS15 bit fields are defined, other fields are reserved. For Port D, only DS0, DS12, DS13, DS14 and DS15 bit fields are defined, other fields are reserved.

5.4 AFIO Registers

The AFIO registers are accessible through APB bus. The register description is as follows.

5.4.1 AFIO register overview

Table 5-42 AFIO 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	4	3	0
	Reset value																												
0x18	AFIO_EFT_CFG1																												
	Reset value																												
0x1C	AFIO_EFT_CFG2	PD15EFTEN	PD14EFTEN																										
	Reset value	0	0																										
0x20	AFIO_FILT_CFG																												
	Reset value																												
0x24	AFIO_DIGEFT_CFG1																												
	Reset value	PD15DIGEFTEN	PD14DIGEFTEN																										
0x28	AFIO_DIGEFT_CFG2	PD0DIGEFTEN																											
	Reset value	0	0																										

5.4.2 Alternate function mapping configuration control register (AFIO_RMP_CFG)

Address offset : 0x00

Reset value : 0x00000000

31	Reserved														16
15	Reserved	rw	SPI1_NSS	rw	SPI2_NSS	rw	ADC_ETRI	rw	ADC_ETRR	rw	EXTI_ETRI[3:0]	rw	EXTI_ETRR[3:0]	rw	0

Bit field	Name	Description
31:12	Reserved	Reserved, the reset value must be maintained.
11	SPI1_NSS	NSS mode of SPI1(when NSS is configured as AFIO push-pull): 0: NSS will be high-z when idle 1: NSS will be 1 when idle
10	SPI2_NSS	NSS mode of SPI2(when NSS is configured as AFIO push-pull): 0: NSS will be high-z when idle 1: NSS will be 1 when idle
9	ADC_ETRI	ADC external trigger injected conversion remapping Set and cleared by software. This bit controls the trigger input connected to ADC external trigger injected conversion. 0: The ADC external trigger injected conversion is connected to EXTI (0-15). 1: The ADC external event injected conversion is connected to TIM8_Channel4
8	ADC_ETRR	ADC external trigger regular conversion remapping. Set and cleared by software. This bit controls the trigger input connected to ADC external trigger regular conversion. 0: The ADC external trigger regular conversion is connected to EXTI (0-15). 1: The ADC external event regular conversion is connected to TIM8_TRGO
7:4	EXTI_ETRI[3:0]	Select interrupt line injection to convert external trigger remapping.
3:0	EXTI_ETRR[3:0]	Select break line rules to convert external trigger remapping.

5.4.3 External interrupt configuration register 1 (AFIO_EXTI_CFG1)

Address offset : 0x04

Reset value : 0x00000000

31	30	29	EXTI3[5:0]	24	23	22	21	EXTI2[5:0]	16
Reserved				rw				rw	
15	14	13	EXTI1[5:0]	8	7	6	5		0
Reserved					rw				
						Reserved		EXTI0[5:0]	
								rw	

Bit field	Name	Description
31:30	Reserved	Reserved, the reset value must be maintained.
29:24	EXTI3[5:0]	EXTI3 configuration These bits are written by software to select the source input for EXTI3 external interrupt. 000000: PA[0] 000001: 1'b0 000010: 1'b0 000011: PD[0] 000100: PA[1] 000101: PB[1]

Bit field	Name	Description
		000110: 1'b0
		000111: 1'b0
		001000: PA[2]
		001001: 1'b0
		001010: 1'b0
		001011: 1'b0
		001100: PA[3]
		001101: 1'b0
		001110: 1'b0
		001111: 1'b0
		010000: PA[4]
		010001: 1'b0
		010010: 1'b0
		010011: 1'b0
		010100: PA[5]
		010101: 1'b0
		010110: 1'b0
		010111: 1'b0
		011000: PA[6]
		011001: 1'b0
		011010: 1'b0
		011011: 1'b0
		011100: PA[7]
		011101: 1'b0
		011110: 1'b0
		011111: 1'b0
		100000: 1'b0
		100001: 1'b0
		100010: 1'b0
		100011: 1'b0
		100100: PA[9]
		100101: 1'b0
		100110: 1'b0
		100111: 1'b0
		101000: PA[10]
		101001: 1'b0
		101010: 1'b0
		101011: 1'b0
		101100: 1'b0
		101101: 1'b0
		101110: 1'b0
		101111: 1'b0

Bit field	Name	Description
		110000: 1'b0 110001: 1'b0 110010: 1'b0 110011: 1'b0 110100: PA[13] 110101: 1'b0 110110: 1'b0 110111: 1'b0 111000: PA[14] 111001: 1'b0 111010: 1'b0 111011: PD[14] 111100: 1'b0 111101: 1'b0 111110: 1'b0 111111: PD[15]
23:22	Reserved	Reserved, the reset value must be maintained.
21:16	EXTI2[5:0]	EXTI2 configuration These bits are written by software to select the source input for EXTI2 external interrupt. 000000: PA[0] 000001: 1'b0 000010: 1'b0 000011: PD[0] 000100: PA[1] 000101: PB[1] 000110: 1'b0 000111: 1'b0 001000: PA[2] 001001: 1'b0 001010: 1'b0 001011: 1'b0 001100: PA[3] 001101: 1'b0 001110: 1'b0 001111: 1'b0 010000: PA[4] 010001: 1'b0 010010: 1'b0 010011: 1'b0 010100: PA[5] 010101: 1'b0

Bit field	Name	Description
		010110: 1'b0 010111: 1'b0 011000: PA[6] 011001: 1'b0 011010: 1'b0 011011: 1'b0 011100: PA[7] 011101: 1'b0 011110: 1'b0 011111: 1'b0 100000: 1'b0 100001: 1'b0 100010: 1'b0 100011: 1'b0 100100: PA[9] 100101: 1'b0 100110: 1'b0 100111: 1'b0 101000: PA[10] 101001: 1'b0 101010: 1'b0 101011: 1'b0 101100: 1'b0 101101: 1'b0 101110: 1'b0 101111: 1'b0 110000: 1'b0 110001: 1'b0 110010: 1'b0 110011: 1'b0 110100: PA[13] 110101: 1'b0 110110: 1'b0 110111: 1'b0 111000: PA[14] 111001: 1'b0 111010: 1'b0 111011: PD[14] 111100: 1'b0 111101: 1'b0 111110: 1'b0 111111: PD[15]

Bit field	Name	Description
15:14	Reserved	Reserved, the reset value must be maintained.
13:8	EXTI1[5:0]	<p>EXTI1 configuration</p> <p>These bits are written by software to select the source input for EXTI1 external interrupt.</p> <p>000000: PA[0] 000001: 1'b0 000010: 1'b0 000011: PD[0] 000100: PA[1] 000101: PB[1] 000110: 1'b0 000111: 1'b0 001000: PA[2] 001001: 1'b0 001010: 1'b0 001011: 1'b0 001100: PA[3] 001101: 1'b0 001110: 1'b0 001111: 1'b0 010000: PA[4] 010001: 1'b0 010010: 1'b0 010011: 1'b0 010100: PA[5] 010101: 1'b0 010110: 1'b0 010111: 1'b0 011000: PA[6] 011001: 1'b0 011010: 1'b0 011011: 1'b0 011100: PA[7] 011101: 1'b0 011110: 1'b0 011111: 1'b0 100000: 1'b0 100001: 1'b0 100010: 1'b0 100011: 1'b0 100100: PA[9] 100101: 1'b0 </p>

Bit field	Name	Description
		100110: 1'b0 100111: 1'b0 101000: PA[10] 101001: 1'b0 101010: 1'b0 101011: 1'b0 101100: 1'b0 101101: 1'b0 101110: 1'b0 101111: 1'b0 110000: 1'b0 110001: 1'b0 110010: 1'b0 110011: 1'b0 110100: PA[13] 110101: 1'b0 110110: 1'b0 110111: 1'b0 111000: PA[14] 111001: 1'b0 111010: 1'b0 111011: PD[14] 111100: 1'b0 111101: 1'b0 111110: 1'b0 111111: PD[15]
7:6	Reserved	Reserved, the reset value must be maintained.
5:0	EXTI0[5:0]	EXTI0 configuration These bits are written by software to select the source input for EXTI0 external interrupt. 000000: PA[0] 000001: 1'b0 000010: 1'b0 000011: PD[0] 000100: PA[1] 000101: PB[1] 000110: 1'b0 000111: 1'b0 001000: PA[2] 001001: 1'b0 001010: 1'b0 001011: 1'b0

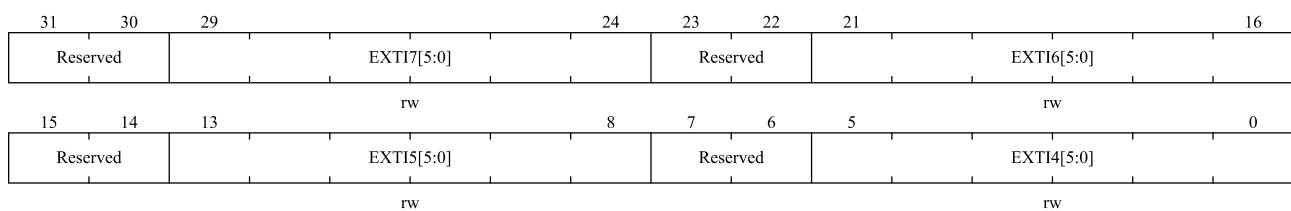
Bit field	Name	Description
		001100: PA[3] 001101: 1'b0 001110: 1'b0 001111: 1'b0 010000: PA[4] 010001: 1'b0 010010: 1'b0 010011: 1'b0 010100: PA[5] 010101: 1'b0 010110: 1'b0 010111: 1'b0 011000: PA[6] 011001: 1'b0 011010: 1'b0 011011: 1'b0 011100: PA[7] 011101: 1'b0 011110: 1'b0 011111: 1'b0 100000: 1'b0 100001: 1'b0 100010: 1'b0 100011: 1'b0 100100: PA[9] 100101: 1'b0 100110: 1'b0 100111: 1'b0 101000: PA[10] 101001: 1'b0 101010: 1'b0 101011: 1'b0 101100: 1'b0 101101: 1'b0 101110: 1'b0 101111: 1'b0 110000: 1'b0 110001: 1'b0 110010: 1'b0 110011: 1'b0 110100: PA[13] 110101: 1'b0

Bit field	Name	Description
		110110: 1'b0 110111: 1'b0 111000: PA[14] 111001: 1'b0 111010: 1'b0 111011: PD[14] 111100: 1'b0 111101: 1'b0 111110: 1'b0 111111: PD[15]

5.4.4 External interrupt configuration register 2 (AFIO_EXTI_CFG2)

Address offset : 0x08

Reset value : 0x00000000



Bit field	Name	Description
31:30	Reserved	Reserved, the reset value must be maintained.
29:24	EXTI7[5:0]	EXTI7 configuration These bits are written by software to select the source input for EXTI7 external interrupt. 000000: PA[0] 000001: 1'b0 000010: 1'b0 000011: PD[0] 000100: PA[1] 000101: PB[1] 000110: 1'b0 000111: 1'b0 001000: PA[2] 001001: 1'b0 001010: 1'b0 001011: 1'b0 001100: PA[3] 001101: 1'b0 001110: 1'b0

Bit field	Name	Description
		001111: 1'b0
		010000: PA[4]
		010001: 1'b0
		010010: 1'b0
		010011: 1'b0
		010100: PA[5]
		010101: 1'b0
		010110: 1'b0
		010111: 1'b0
		011000: PA[6]
		011001: 1'b0
		011010: 1'b0
		011011: 1'b0
		011100: PA[7]
		011101: 1'b0
		011110: 1'b0
		011111: 1'b0
		100000: 1'b0
		100001: 1'b0
		100010: 1'b0
		100011: 1'b0
		100100: PA[9]
		100101: 1'b0
		100110: 1'b0
		100111: 1'b0
		101000: PA[10]
		101001: 1'b0
		101010: 1'b0
		101011: 1'b0
		101100: 1'b0
		101101: 1'b0
		101110: 1'b0
		101111: 1'b0
		110000: 1'b0
		110001: 1'b0
		110010: 1'b0
		110011: 1'b0
		110100: PA[13]
		110101: 1'b0
		110110: 1'b0
		110111: 1'b0
		111000: PA[14]

Bit field	Name	Description
		111001: 1'b0 111010: 1'b0 111011: PD[14] 111100: 1'b0 111101: 1'b0 111110: 1'b0 111111: PD[15]
23:22	Reserved	Reserved, the reset value must be maintained.
21:16	EXTI6[5:0]	EXTI6 configuration These bits are written by software to select the source input for EXTI6 external interrupt. 000000: PA[0] 000001: 1'b0 000010: 1'b0 000011: PD[0] 000100: PA[1] 000101: PB[1] 000110: 1'b0 000111: 1'b0 001000: PA[2] 001001: 1'b0 001010: 1'b0 001011: 1'b0 001100: PA[3] 001101: 1'b0 001110: 1'b0 001111: 1'b0 010000: PA[4] 010001: 1'b0 010010: 1'b0 010011: 1'b0 010100: PA[5] 010101: 1'b0 010110: 1'b0 010111: 1'b0 011000: PA[6] 011001: 1'b0 011010: 1'b0 011011: 1'b0 011100: PA[7] 011101: 1'b0 011110: 1'b0

Bit field	Name	Description
		011111: 1'b0 100000: 1'b0 100001: 1'b0 100010: 1'b0 100011: 1'b0 100100: PA[9] 100101: 1'b0 100110: 1'b0 100111: 1'b0 101000: PA[10] 101001: 1'b0 101010: 1'b0 101011: 1'b0 101100: 1'b0 101101: 1'b0 101110: 1'b0 101111: 1'b0 110000: 1'b0 110001: 1'b0 110010: 1'b0 110011: 1'b0 110100: PA[13] 110101: 1'b0 110110: 1'b0 110111: 1'b0 111000: PA[14] 111001: 1'b0 111010: 1'b0 111011: PD[14] 111100: 1'b0 111101: 1'b0 111110: 1'b0 111111: PD[15]
15:14	Reserved	Reserved, the reset value must be maintained.
13:8	EXTI5[5:0]	EXTI5 configuration These bits are written by software to select the source input for EXTI5 external interrupt. 000000: PA[0] 000001: 1'b0 000010: 1'b0 000011: PD[0] 000100: PA[1]

Bit field	Name	Description
		000101: PB[1]
		000110: 1'b0
		000111: 1'b0
		001000: PA[2]
		001001: 1'b0
		001010: 1'b0
		001011: 1'b0
		001100: PA[3]
		001101: 1'b0
		001110: 1'b0
		001111: 1'b0
		010000: PA[4]
		010001: 1'b0
		010010: 1'b0
		010011: 1'b0
		010100: PA[5]
		010101: 1'b0
		010110: 1'b0
		010111: 1'b0
		011000: PA[6]
		011001: 1'b0
		011010: 1'b0
		011011: 1'b0
		011100: PA[7]
		011101: 1'b0
		011110: 1'b0
		011111: 1'b0
		100000: 1'b0
		100001: 1'b0
		100010: 1'b0
		100011: 1'b0
		100100: PA[9]
		100101: 1'b0
		100110: 1'b0
		100111: 1'b0
		101000: PA[10]
		101001: 1'b0
		101010: 1'b0
		101011: 1'b0
		101100: 1'b0
		101101: 1'b0
		101110: 1'b0

Bit field	Name	Description
		101111: 1'b0 110000: 1'b0 110001: 1'b0 110010: 1'b0 110011: 1'b0 110100: PA[13] 110101: 1'b0 110110: 1'b0 110111: 1'b0 111000: PA[14] 111001: 1'b0 111010: 1'b0 111011: PD[14] 111100: 1'b0 111101: 1'b0 111110: 1'b0 111111: PD[15]
7:6	Reserved	Reserved, the reset value must be maintained.
5:0	EXTI4[5:0]	EXTI4 configuration These bits are written by software to select the source input for EXTI4 external interrupt. 000000: PA[0] 000001: 1'b0 000010: 1'b0 000011: PD[0] 000100: PA[1] 000101: PB[1] 000110: 1'b0 000111: 1'b0 001000: PA[2] 001001: 1'b0 001010: 1'b0 001011: 1'b0 001100: PA[3] 001101: 1'b0 001110: 1'b0 001111: 1'b0 010000: PA[4] 010001: 1'b0 010010: 1'b0 010011: 1'b0 010100: PA[5]

Bit field	Name	Description
		010101: 1'b0
		010110: 1'b0
		010111: 1'b0
		011000: PA[6]
		011001: 1'b0
		011010: 1'b0
		011011: 1'b0
		011100: PA[7]
		011101: 1'b0
		011110: 1'b0
		011111: 1'b0
		100000: 1'b0
		100001: 1'b0
		100010: 1'b0
		100011: 1'b0
		100100: PA[9]
		100101: 1'b0
		100110: 1'b0
		100111: 1'b0
		101000: PA[10]
		101001: 1'b0
		101010: 1'b0
		101011: 1'b0
		101100: 1'b0
		101101: 1'b0
		101110: 1'b0
		101111: 1'b0
		110000: 1'b0
		110001: 1'b0
		110010: 1'b0
		110011: 1'b0
		110100: PA[13]
		110101: 1'b0
		110110: 1'b0
		110111: 1'b0
		111000: PA[14]
		111001: 1'b0
		111010: 1'b0
		111011: PD[14]
		111100: 1'b0
		111101: 1'b0
		111110: 1'b0

Bit field	Name	Description
		111111: PD[15]

5.4.5 External interrupt configuration register 3 (AFIO_EXTI_CFG3)

Address offset : 0x0C

Reset value : 0x00000000

31	30	29	EXTI11[5:0]	24	23	22	21	16
Reserved				Reserved				EXTI10[5:0]
			rw					
15	14	13	EXTI9[5:0]	8	7	6	5	0
Reserved				rw	rw	rw	rw	rw
			EXTI8[5:0]					

Bit field	Name	Description
31:30	Reserved	Reserved, the reset value must be maintained.
29:24	EXTI11[5:0]	<p>EXTI11 configuration</p> <p>These bits are written by software to select the source input for EXTI11 external interrupt.</p> <p>000000: PA[0] 000001: 1'b0 000010: 1'b0 000011: PD[0] 000100: PA[1] 000101: PB[1] 000110: 1'b0 000111: 1'b0 001000: PA[2] 001001: 1'b0 001010: 1'b0 001011: 1'b0 001100: PA[3] 001101: 1'b0 001110: 1'b0 001111: 1'b0 010000: PA[4] 010001: 1'b0 010010: 1'b0 010011: 1'b0 010100: PA[5] 010101: 1'b0 010110: 1'b0 010111: 1'b0</p>

Bit field	Name	Description
		011000: PA[6] 011001: 1'b0 011010: 1'b0 011011: 1'b0 011100: PA[7] 011101: 1'b0 011110: 1'b0 011111: 1'b0 100000: 1'b0 100001: 1'b0 100010: 1'b0 100011: 1'b0 100100: PA[9] 100101: 1'b0 100110: 1'b0 100111: 1'b0 101000: PA[10] 101001: 1'b0 101010: 1'b0 101011: 1'b0 101100: 1'b0 101101: 1'b0 101110: 1'b0 101111: 1'b0 110000: 1'b0 110001: 1'b0 110010: 1'b0 110011: 1'b0 110100: PA[13] 110101: 1'b0 110110: 1'b0 110111: 1'b0 111000: PA[14] 111001: 1'b0 111010: 1'b0 111011: PD[14] 111100: 1'b0 111101: 1'b0 111110: 1'b0 111111: PD[15]
23:22	Reserved	Reserved, the reset value must be maintained.
21:16	EXTI10[5:0]	EXTI10 configuration

Bit field	Name	Description
		<p>These bits are written by software to select the source input for EXTI10 external interrupt.</p> <p>00000: PA[0] 00001: 1'b0 00010: 1'b0 00011: PD[0] 000100: PA[1] 000101: PB[1] 000110: 1'b0 000111: 1'b0 001000: PA[2] 001001: 1'b0 001010: 1'b0 001011: 1'b0 001100: PA[3] 001101: 1'b0 001110: 1'b0 001111: 1'b0 010000: PA[4] 010001: 1'b0 010010: 1'b0 010011: 1'b0 010100: PA[5] 010101: 1'b0 010110: 1'b0 010111: 1'b0 011000: PA[6] 011001: 1'b0 011010: 1'b0 011011: 1'b0 011100: PA[7] 011101: 1'b0 011110: 1'b0 011111: 1'b0 100000: 1'b0 100001: 1'b0 100010: 1'b0 100011: 1'b0 100100: PA[9] 100101: 1'b0 100110: 1'b0 100111: 1'b0 </p>

Bit field	Name	Description
		101000: PA[10] 101001: 1'b0 101010: 1'b0 101011: 1'b0 101100: 1'b0 101101: 1'b0 101110: 1'b0 101111: 1'b0 110000: 1'b0 110001: 1'b0 110010: 1'b0 110011: 1'b0 110100: PA[13] 110101: 1'b0 110110: 1'b0 110111: 1'b0 111000: PA[14] 111001: 1'b0 111010: 1'b0 111011: PD[14] 111100: 1'b0 111101: 1'b0 111110: 1'b0 111111: PD[15]
15:14	Reserved	Reserved, the reset value must be maintained.
13:8	EXTI9[5:0]	EXTI9 configuration These bits are written by software to select the source input for EXTI9 external interrupt. 000000: PA[0] 000001: 1'b0 000010: 1'b0 000011: PD[0] 000100: PA[1] 000101: PB[1] 000110: 1'b0 000111: 1'b0 001000: PA[2] 001001: 1'b0 001010: 1'b0 001011: 1'b0 001100: PA[3] 001101: 1'b0

Bit field	Name	Description
		001110: 1'b0 001111: 1'b0 010000: PA[4] 010001: 1'b0 010010: 1'b0 010011: 1'b0 010100: PA[5] 010101: 1'b0 010110: 1'b0 010111: 1'b0 011000: PA[6] 011001: 1'b0 011010: 1'b0 011011: 1'b0 011100: PA[7] 011101: 1'b0 011110: 1'b0 011111: 1'b0 100000: 1'b0 100001: 1'b0 100010: 1'b0 100011: 1'b0 100100: PA[9] 100101: 1'b0 100110: 1'b0 100111: 1'b0 101000: PA[10] 101001: 1'b0 101010: 1'b0 101011: 1'b0 101100: 1'b0 101101: 1'b0 101110: 1'b0 101111: 1'b0 110000: 1'b0 110001: 1'b0 110010: 1'b0 110011: 1'b0 110100: PA[13] 110101: 1'b0 110110: 1'b0 110111: 1'b0

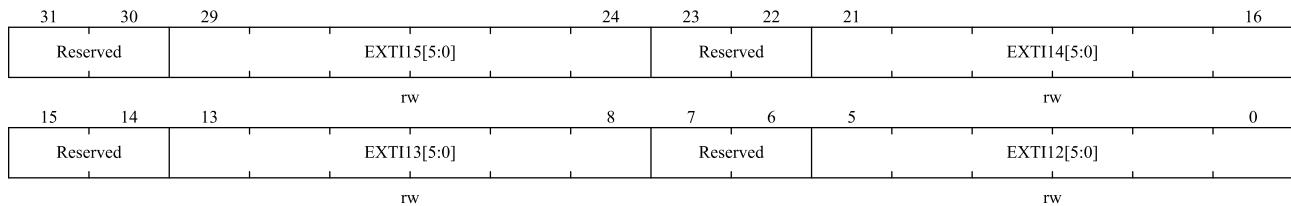
Bit field	Name	Description
		111000: PA[14] 111001: 1'b0 111010: 1'b0 111011: PD[14] 111100: 1'b0 111101: 1'b0 111110: 1'b0 111111: PD[15]
7:6	Reserved	Reserved, the reset value must be maintained.
5:0	EXTI8[5:0]	EXTI8 configuration These bits are written by software to select the source input for EXTI8 external interrupt. 000000: PA[0] 000001: 1'b0 000010: 1'b0 000011: PD[0] 000100: PA[1] 000101: PB[1] 000110: 1'b0 000111: 1'b0 001000: PA[2] 001001: 1'b0 001010: 1'b0 001011: 1'b0 001100: PA[3] 001101: 1'b0 001110: 1'b0 001111: 1'b0 010000: PA[4] 010001: 1'b0 010010: 1'b0 010011: 1'b0 010100: PA[5] 010101: 1'b0 010110: 1'b0 010111: 1'b0 011000: PA[6] 011001: 1'b0 011010: 1'b0 011011: 1'b0 011100: PA[7] 011101: 1'b0

Bit field	Name	Description
		011110: 1'b0
		011111: 1'b0
		100000: 1'b0
		100001: 1'b0
		100010: 1'b0
		100011: 1'b0
		100100: PA[9]
		100101: 1'b0
		100110: 1'b0
		100111: 1'b0
		101000: PA[10]
		101001: 1'b0
		101010: 1'b0
		101011: 1'b0
		101100: 1'b0
		101101: 1'b0
		101110: 1'b0
		101111: 1'b0
		110000: 1'b0
		110001: 1'b0
		110010: 1'b0
		110011: 1'b0
		110100: PA[13]
		110101: 1'b0
		110110: 1'b0
		110111: 1'b0
		111000: PA[14]
		111001: 1'b0
		111010: 1'b0
		111011: PD[14]
		111100: 1'b0
		111101: 1'b0
		111110: 1'b0
		111111: PD[15]

5.4.6 External interrupt configuration register 4 (AFIO_EXTI_CFG4)

Address offset : 0x10

Reset value : 0x00000000



Bit field	Name	Description
31:30	Reserved	Reserved, the reset value must be maintained.
29:24	EXTI15[5:0]	<p>EXTI15 configuration</p> <p>These bits are written by software to select the source input for EXTI15 external interrupt.</p> <p>000000: PA[0] 000001: 1'b0 000010: 1'b0 000011: PD[0] 000100: PA[1] 000101: PB[1] 000110: 1'b0 000111: 1'b0 001000: PA[2] 001001: 1'b0 001010: 1'b0 001011: 1'b0 001100: PA[3] 001101: 1'b0 001110: 1'b0 001111: 1'b0 010000: PA[4] 010001: 1'b0 010010: 1'b0 010011: 1'b0 010100: PA[5] 010101: 1'b0 010110: 1'b0 010111: 1'b0 011000: PA[6] 011001: 1'b0 011010: 1'b0 011011: 1'b0 011100: PA[7] 011101: 1'b0 011110: 1'b0 011111: 1'b0 </p>

Bit field	Name	Description
		100000: 1'b0 100001: 1'b0 100010: 1'b0 100011: 1'b0 100100: PA[9] 100101: 1'b0 100110: 1'b0 100111: 1'b0 101000: PA[10] 101001: 1'b0 101010: 1'b0 101011: 1'b0 101100: 1'b0 101101: 1'b0 101110: 1'b0 101111: 1'b0 110000: 1'b0 110001: 1'b0 110010: 1'b0 110011: 1'b0 110100: PA[13] 110101: 1'b0 110110: 1'b0 110111: 1'b0 111000: PA[14] 111001: 1'b0 111010: 1'b0 111011: PD[14] 111100: 1'b0 111101: 1'b0 111110: 1'b0 111111: PD[15]
23:22	Reserved	Reserved, the reset value must be maintained.
21:16	EXTI14[5:0]	EXTI14 configuration These bits are written by software to select the source input for EXTI14 external interrupt. 000000: PA[0] 000001: 1'b0 000010: 1'b0 000011: PD[0] 000100: PA[1] 000101: PB[1]

Bit field	Name	Description
		000110: 1'b0
		000111: 1'b0
		001000: PA[2]
		001001: 1'b0
		001010: 1'b0
		001011: 1'b0
		001100: PA[3]
		001101: 1'b0
		001110: 1'b0
		001111: 1'b0
		010000: PA[4]
		010001: 1'b0
		010010: 1'b0
		010011: 1'b0
		010100: PA[5]
		010101: 1'b0
		010110: 1'b0
		010111: 1'b0
		011000: PA[6]
		011001: 1'b0
		011010: 1'b0
		011011: 1'b0
		011100: PA[7]
		011101: 1'b0
		011110: 1'b0
		011111: 1'b0
		100000: 1'b0
		100001: 1'b0
		100010: 1'b0
		100011: 1'b0
		100100: PA[9]
		100101: 1'b0
		100110: 1'b0
		100111: 1'b0
		101000: PA[10]
		101001: 1'b0
		101010: 1'b0
		101011: 1'b0
		101100: 1'b0
		101101: 1'b0
		101110: 1'b0
		101111: 1'b0

Bit field	Name	Description
		110000: 1'b0 110001: 1'b0 110010: 1'b0 110011: 1'b0 110100: PA[13] 110101: 1'b0 110110: 1'b0 110111: 1'b0 111000: PA[14] 111001: 1'b0 111010: 1'b0 111011: PD[14] 111100: 1'b0 111101: 1'b0 111110: 1'b0 111111: PD[15]
15:14	Reserved	Reserved, the reset value must be maintained.
13:8	EXTI13[5:0]	EXTI13 configuration These bits are written by software to select the source input for EXTI13 external interrupt. 000000: PA[0] 000001: 1'b0 000010: 1'b0 000011: PD[0] 000100: PA[1] 000101: PB[1] 000110: 1'b0 000111: 1'b0 001000: PA[2] 001001: 1'b0 001010: 1'b0 001011: 1'b0 001100: PA[3] 001101: 1'b0 001110: 1'b0 001111: 1'b0 010000: PA[4] 010001: 1'b0 010010: 1'b0 010011: 1'b0 010100: PA[5] 010101: 1'b0

Bit field	Name	Description
		010110: 1'b0
		010111: 1'b0
		011000: PA[6]
		011001: 1'b0
		011010: 1'b0
		011011: 1'b0
		011100: PA[7]
		011101: 1'b0
		011110: 1'b0
		011111: 1'b0
		100000: 1'b0
		100001: 1'b0
		100010: 1'b0
		100011: 1'b0
		100100: PA[9]
		100101: 1'b0
		100110: 1'b0
		100111: 1'b0
		101000: PA[10]
		101001: 1'b0
		101010: 1'b0
		101011: 1'b0
		101100: 1'b0
		101101: 1'b0
		101110: 1'b0
		101111: 1'b0
		110000: 1'b0
		110001: 1'b0
		110010: 1'b0
		110011: 1'b0
		110100: PA[13]
		110101: 1'b0
		110110: 1'b0
		110111: 1'b0
		111000: PA[14]
		111001: 1'b0
		111010: 1'b0
		111011: PD[14]
		111100: 1'b0
		111101: 1'b0
		111110: 1'b0
		111111: PD[15]

Bit field	Name	Description
7:6	Reserved	Reserved, the reset value must be maintained.
5:0	EXTI12[5:0]	<p>EXTI12 configuration</p> <p>These bits are written by software to select the source input for EXTI12 external interrupt.</p> <p>000000: PA[0] 000001: 1'b0 000010: 1'b0 000011: PD[0] 000100: PA[1] 000101: PB[1] 000110: 1'b0 000111: 1'b0 001000: PA[2] 001001: 1'b0 001010: 1'b0 001011: 1'b0 001100: PA[3] 001101: 1'b0 001110: 1'b0 001111: 1'b0 010000: PA[4] 010001: 1'b0 010010: 1'b0 010011: 1'b0 010100: PA[5] 010101: 1'b0 010110: 1'b0 010111: 1'b0 011000: PA[6] 011001: 1'b0 011010: 1'b0 011011: 1'b0 011100: PA[7] 011101: 1'b0 011110: 1'b0 011111: 1'b0 100000: 1'b0 100001: 1'b0 100010: 1'b0 100011: 1'b0 100100: PA[9] 100101: 1'b0 </p>

Bit field	Name	Description
		100110: 1'b0 100111: 1'b0 101000: PA[10] 101001: 1'b0 101010: 1'b0 101011: 1'b0 101100: 1'b0 101101: 1'b0 101110: 1'b0 101111: 1'b0 110000: 1'b0 110001: 1'b0 110010: 1'b0 110011: 1'b0 110100: PA[13] 110101: 1'b0 110110: 1'b0 110111: 1'b0 111000: PA[14] 111001: 1'b0 111010: 1'b0 111011: PD[14] 111100: 1'b0 111101: 1'b0 111110: 1'b0 111111: PD[15]

5.4.7 5V tolerance configuration register (AFIO_TOL5V_CFG)

Address offset : 0x14

Reset value : 0x00000000

31																16
Reserved																
15	14	13	12		8	7	6	5	4	3	2	1	0			
Reserved	PB1 TOLENN			Reserved		PA7 TOLENN	PA6 TOLENN	PA5 TOLENN	PA4 TOLENN	PA3 TOLENN	PA2 TOLENN	PA1 TOLENN	PA0 TOLENN			
	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bit field	Name	Description
31:14	Reserved	Reserved, the reset value must be maintained.
13	PB1TOLENN	Pin PB1 5V tolerance enable 0: Enable 5V tolerance

Bit field	Name	Description
		1: Disable 5V tolerance
12:8	Reserved	Reserved, the reset value must be maintained.
7	PA7TOLENN	Pin PA7 5V tolerance enable 0: Enable 5V tolerance 1: Disable 5V tolerance
6	PA6TOLENN	Pin PA6 5V tolerance enable 0: Enable 5V tolerance 1: Disable 5V tolerance
5	PA5TOLENN	Pin PA5 5V tolerance enable 0: Enable 5V tolerance 1: Disable 5V tolerance
4	PA4TOLENN	Pin PA4 5V tolerance enable 0: Enable 5V tolerance 1: Disable 5V tolerance
3	PA3TOLENN	Pin PA3 5V tolerance enable 0: Enable 5V tolerance 1: Disable 5V tolerance
2	PA2TOLENN	Pin PA2 5V tolerance enable 0: Enable 5V tolerance 1: Disable 5V tolerance
1	PA1TOLENN	Pin PA1 5V tolerance enable 0: Enable 5V tolerance 1: Disable 5V tolerance
0	PA0TOLENN	Pin PA0 5V tolerance enable 0: Enable 5V tolerance 1: Disable 5V tolerance

5.4.8 Analog filter configuration register 1 (AFIO_EFT_CFG1)

Address offset : 0x18

Reset value : 0x00000000

31	Reserved															18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Reserved	PA14 EFTEN	PA13 EFTEN	Reserved		PA10 EFTEN	PA9 EFTEN	Reserved	PA7 EFTEN	PA6 EFTEN	PA5 EFTEN	PA4 EFTEN	PA3 EFTEN	PA2 EFTEN	PA1 EFTEN	PA0 EFTEN			
	rw	rw			rw	rw		rw										

Bit field	Name	Description
31:18	Reserved	Reserved, the reset value must be maintained.
17	PB1EFTEN	Pin PB1 EFT enable 0: Disable EFT

Bit field	Name	Description
		1: Enable EFT
16:15	Reserved	Reserved, the reset value must be maintained.
14:13	PAyEFTEN	Pin PAy EFT enable (y = 13/14) 0: Disable EFT 1: Enable EFT
12:11	Reserved	Reserved, the reset value must be maintained.
10:9	PAyEFTEN	Pin PAy EFT enable (y = 9/10) 0: Disable EFT 1: Enable EFT
8	Reserved	Reserved, the reset value must be maintained.
7:0	PAyEFTEN	Pin PAy EFT enable (y = 0...7) 0: Disable EFT 1: Enable EFT

5.4.9 Analog filter configuration register 2 (AFIO_EFT_CFG2)

Address offset : 0x1C

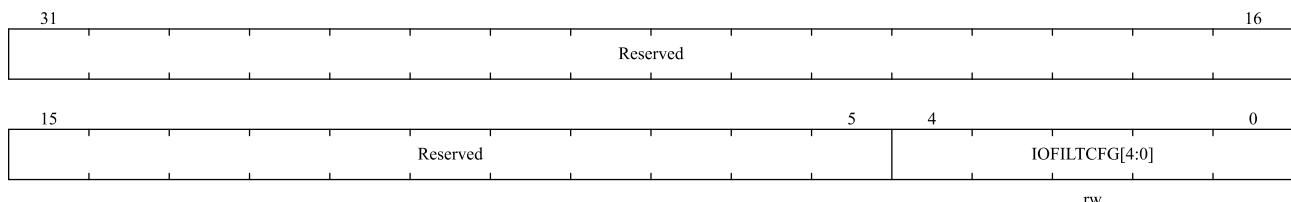
Reset value : 0x00000000

Bit field	Name	Description
31	PD15EFTEN	Pin PD15 EFT enable 0: Disable EFT 1: Enable EFT
30	PD14EFTEN	Pin PD14 EFT enable 0: Disable EFT 1: Enable EFT
29:17	Reserved	Reserved, the reset value must be maintained.
16	PD0EFTEN	Pin PD0 EFT enable 0: Disable EFT 1: Enable EFT
15:0	Reserved	Reserved, the reset value must be maintained.

5.4.10 Digital glitch filter stage (glitch width) configuration register (AFIO_FILT_CFG)

Address offset : 0x20

Reset value : 0x00000000

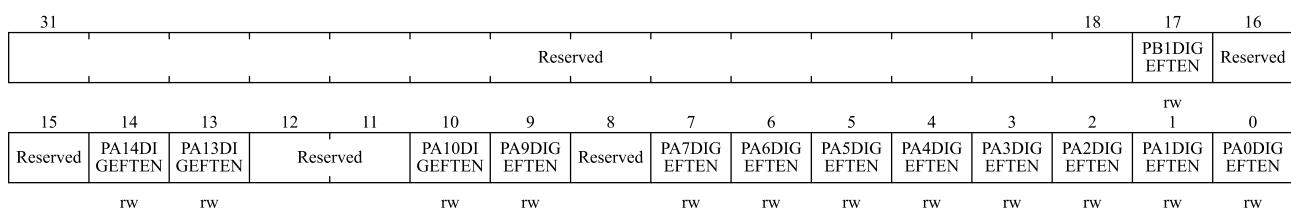


Bit field	Name	Description
31:5	Reserved	Reserved, the reset value must be maintained.
4:0	IOFLITCFG[4:0]	Filter stage control 00000 : Filter bypass Others : counter value indicating minimum pulse width in terms of AHB clock cycles

5.4.11 Digital glitch filter configuration register 1 (AFIO_DIGEFT_CFG1)

Address offset : 0x24

Reset value : 0x00000000



Bit field	Name	Description
31:18	Reserved	Reserved, the reset value must be maintained.
17	PB1DIGEFTEN	Pin PB1 Digital EFT enable 0: Disable EFT 1: Enable EFT
16:15	Reserved	Reserved, the reset value must be maintained.
14:13	PAyDIGEFTEN	Pin PAy Digital EFT enable (y =13/14) 0: Disable EFT 1: Enable EFT
12:11	Reserved	Reserved, the reset value must be maintained.
10:9	PAyDIGEFTEN	Pin PAy Digital EFT enable (y =9/10)

Bit field	Name	Description
		0: Disable EFT 1: Enable EFT
8	Reserved	Reserved, the reset value must be maintained.
7:0	PAyDGFETEN	Pin PAy Digital EFT enable (y = 0...7) 0: Disable EFT 1: Enable EFT

5.4.12 Digital glitch filter configuration register 2 (AFIO_DIGEFT_CFG2)

Address offset : 0x28

Reset value : 0x00000000

Bit field	Name	Description
31	PD15DIGEFTEN	Pin PD15 Digital EFT enable 0: Disable EFT 1: Enable EFT
30	PD14DIGEFTEN	Pin PD14 Digital EFT enable 0: Disable EFT 1: Enable EFT
29:17	Reserved	Reserved, the reset value must be maintained.
16	PD0DIGEFTEN	Pin PD0 Digital EFT enable 0: Disable EFT 1: Enable EFT
15:0	Reserved	Reserved, the reset value must be maintained.

6 Interrupts and events

6.1 Nested vectored interrupt controller

Features

- 48 masked interrupt channels (excluding 16 interrupt lines of Cortex-M4).
- 16 programmable priority levels (4-bit interrupt priority level is used);
- Low-latency exception and interrupt handling;
- Power management control;
- Implementation of system control register;

Nested Vector Interrupt Controller (NVIC) is closely connected with the interface of processor core, which can realize low-latency interrupt handling and efficiently handle late interrupts. The nested vector interrupt controller manages interrupts including kernel exceptions.

6.1.1 SysTick calibration value register

The system tick calibration value is fixed at 16000. When the system tick clock is set to 16MHz (the maximum value of HCLK/8), a 1ms time reference is generated.

6.1.2 Interrupt and exception vectors

Table 6-1 Vector table

Position	Priority	Type of priority	Acronym	Description	Address
-	-	-	-	Reserved	0x0000_0000
-	-3	fixed	Reset	Reset	0x0000_0004
-	-2	fixed	NMI	Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000_0008
-	-1	fixed	HardFault	All class of fault	0x0000_000C
-	0	settable	MemManage	Memory management	0x0000_0010
-	1	settable	BusFault	Pre-fetch fault, memory access fault	0x0000_0014
-	2	settable	UsageFault	Undefined instruction or illegal state	0x0000_0018
-	-	-	-	Reserved	0x0000_001C -

					0x0000_002B
-	3	settable	SVCall	System service call via SWI instruction	0x0000_002C
-	4	settable	Debug Monitor	Debug Monitor	0x0000_0030
-	-	-	-	Reserved	0x0000_0034
-	5	settable	PendSV	Pendable request for system service	0x0000_0038
-	6	settable	SysTick	System tick timer	0x0000_003C
0	7	settable	WWDG	Window watchdog interrupt	0x0000_0040
1	8	settable	PVD	PVD through EXTI Line16 detection interrupt	0x0000_0044
2	9	settable	RTC_TAMPER_STAMP	Tamper, Time-stamp and LESCSS through EXTI Line18 interrupt	0x0000_0048
3	10	settable	RTC_WKUP	RTC Wakeup timer through EXTI line19 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	EXTI Line0 interrupt	0x0000_0058
7	14	settable	EXTI1	EXTI Line1 interrupt	0x0000_005C
8	15	settable	EXTI2	EXTI Line2 interrupt	0x0000_0060
9	16	settable	EXTI3	EXTI Line3 interrupt	0x0000_0064
10	17	settable	EXTI4	EXTI Line4 interrupt	0x0000_0068
11	18	settable	DMA1_Channel1	DMA1 Channel1 global interrupt	0x0000_006C
12	19	settable	DMA1_Channel2	DMA1 Channel2 global interrupt	0x0000_0070
13	20	settable	DMA1_Channel3	DMA1 Channel3 global interrupt	0x0000_0074
14	21	settable	DMA1_Channel4	DMA1 Channel4 global interrupt	0x0000_0078
15	22	settable	DMA1_Channel5	DMA1 Channel5 global interrupt	0x0000_007C
16	23	settable	DMA1_Channel6	DMA1 Channel6 global interrupt	0x0000_0080
17	24	settable	DMA1_Channel7	DMA1 Channel7 global interrupt	0x0000_0084
18	25	settable	DMA1_Channel8	DMA1 Channel8 global interrupt	0x0000_0088

19	26	settable	ADC	ADC global interrupt	0x0000_008C
20	27	settable	MMU	MMU global interrupt	0x0000_0090
21	28	settable	-	Reserved	0x0000_0094
22	29	settable	EXTI9_5	EXTI Line[9:5] interrupts	0x0000_0098
23	30	settable	TIM1_BRK	TIM1 Break interrupt	0x0000_009C
24	31	settable	TIM1_UP	TIM1 Update interrupt	0x0000_00A0
25	32	settable	TIM1_TRG_COM	TIM1 Trigger and Commutation interrupts	0x0000_00A4
26	33	settable	TIM1_CC	TIM1 Capture Compare interrupt	0x0000_00A8
27	34	settable	TIM2	TIM2 global interrupt	0x0000_00AC
28	35	settable	TIM3	TIM3 global interrupt	0x0000_00B0
29	36	settable	TIM4	TIM4 global interrupt	0x0000_00B4
30	37	settable	I2C1_EV	I ² C1 event interrupt	0x0000_00B8
31	38	settable	I2C1_ER	I ² C1 error interrupt	0x0000_00BC
32	39	settable	I2C2_EV	I2C2 event interrupt	0x0000_00C0
33	40	settable	I2C2_ER	I2C2 error interrupt	0x0000_00C4
34	41	settable	SPI1	SPI1 global interrupt	0x0000_00C8
35	42	settable	SPI2	SPI2 global interrupt	0x0000_00CC
36	43	settable	UART1	UART1 global interrupt	0x0000_00D0
37	44	settable	USART2	USART2 global interrupt	0x0000_00D4
38	45	settable	-	Reserved	0x0000_00D8
39	46	settable	EXTI15_10	EXTI Line[15:10] interrupts	0x0000_00DC
40	47	settable	RTCAlarm	RTC alarm through EXTI line 17 interrupt	0x0000_00E0
41	48	settable	LPTIM_WKUP	LPTIM_WKUP through EXTI line 20 interrupt	0x0000_00E4
42	49	settable	TIM8_BRK	TIM8 Break interrupt	0x0000_00E8
43	50	settable	TIM8_UP	TIM8 Update interrupt	0x0000_00EC
44	51	settable	TIM8_TRG_COM	TIM8 Trigger and Commutation interrupts	0x0000_00F0
45	52	settable	TIM8_CC	TIM8 Capture Compare interrupt	0x0000_00F4
46	53	settable	UART4	UART4 global interrupt	0x0000_00F8

47	54	settable	TIM5	TIM5 global interrupt	0x0000_00FC
48	55	settable	TIM6	TIM6 global interrupt	0x0000_0100

6.2 External interrupt/event controller (EXTI)

6.2.1 Introduction to EXTI

The EXTI controller IP captures interrupt/event triggers and translates them into interrupt requests and event pulses. The interrupt requests are sent to the NVIC interrupt controller. The IP has an APB interface through which the following registers can be accessed.

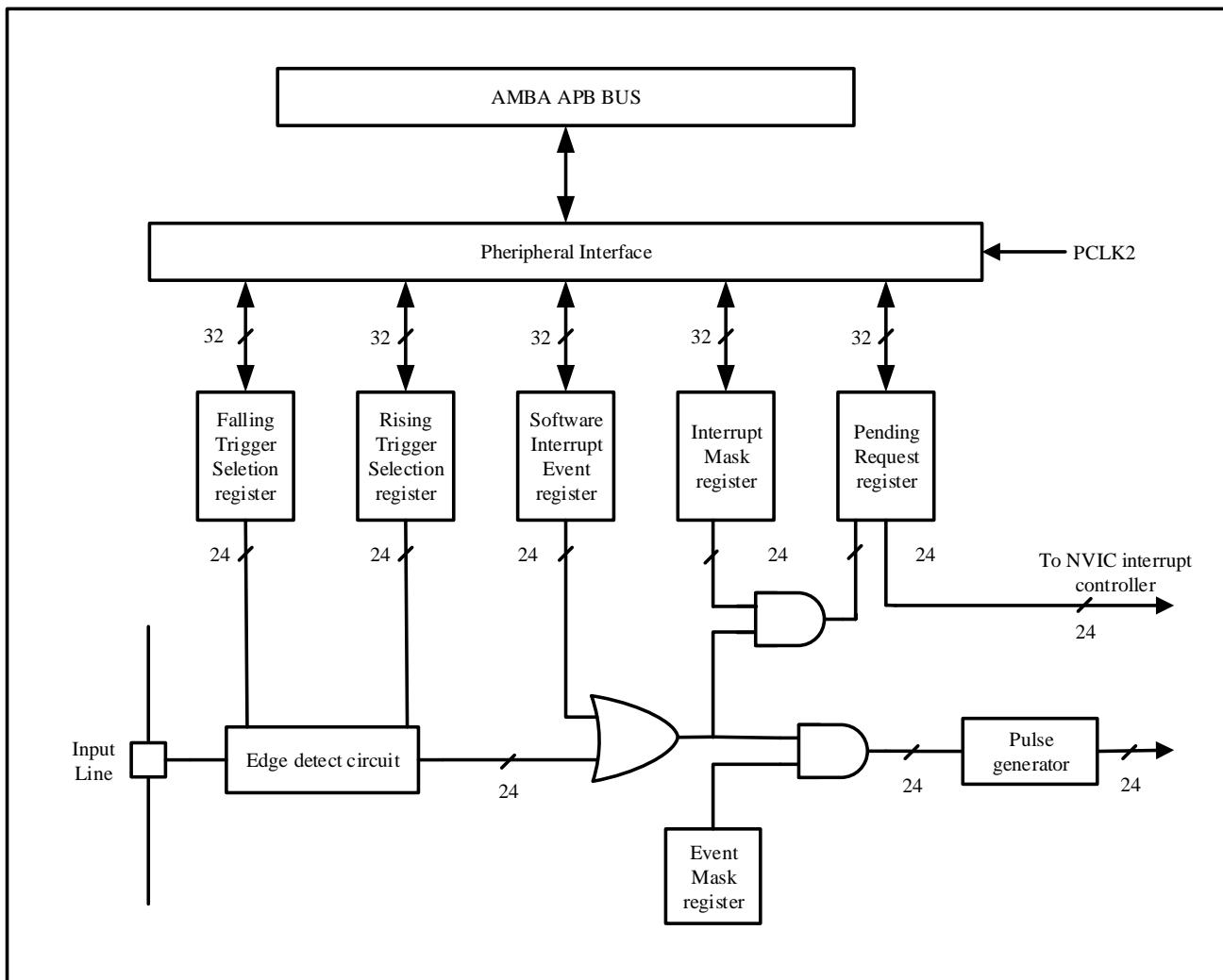
- Rising/Falling edge trigger selection register.
- Dedicated mask bit register for each interrupt/event line.
- Software interrupt/event configuration register.
- Interrupt pending request status register for each interrupt.
- Time Stamp selection register.

6.2.2 EXTI main features

The main features of EXTI controller are as follows:

- Support 24 software interrupt/event requests
- The corresponding interrupt/event of each input line can be independently configured with trigger or mask.
- Each interrupt line has an independent status bit.
- Support pulse or pending input type
- Three types of trigger events are supported: rising edge, falling edge or double edge.
- Wake-up to exit low-power mode

Figure 6-1 EXTI functional diagram



6.2.3 Functional description

EXTI contains 24 interrupt lines, of which 16 are from I/O pins and 8 are from internal modules. To generate an interrupt, the NVIC interrupt channel of the external interrupt controller must be configured to enable the corresponding interrupt line. Select the type of rising edge, falling edge or double edge trigger event through edge trigger configuration registers EXTI_RT_CFG and EXTI_FT_CFG, and write '1' to the corresponding bit of the interrupt mask register EXTI_IMASK to open the interrupt request. When the preset edge trigger polarity is detected on the external interrupt line, an interrupt request will be generated and the corresponding pending bit will be set to '1'. Writing '1' in the corresponding bit of the pending register will clear the interrupt request.

To generate an event, the corresponding event line must be configured and enabled. According to the required polarity of edge detection, set the rising/falling edge trigger configuration register, and write '1' in the corresponding bit of the event mask register to allow the interrupt request. When the preset edge occurs on the event line, an event request pulse will be generated, and the corresponding pending bit will not be set to '1'.

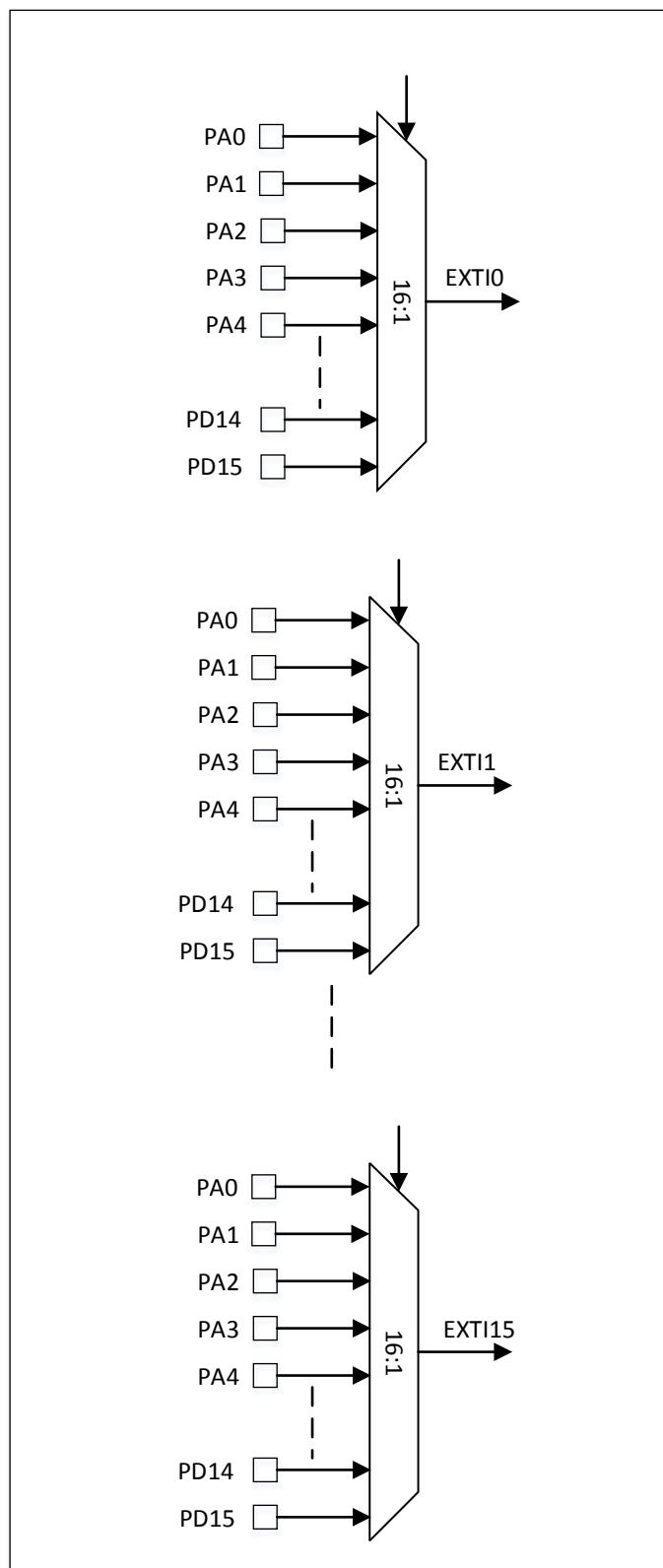
In addition, by writing '1' in the software interrupt/event register, an interrupt/event request can also be generated by

software.

- Hardware interrupt configuration, select and configure 24 lines as interrupt sources as required:
 - ◆ Configure the mask bits of 24 interrupt lines (EXTI_IMASK);
 - ◆ Configure the trigger configuration bits (EXTI_RT_CFG and EXTI_FT_CFG) of the selected disconnection;
 - ◆ The enable and mask bits of the NVIC interrupt channel corresponding to the external interrupt controller enable the requests in 24 interrupt lines to be correctly responded.
- Hardware configuration, select and configure 24 lines as event sources as required:
 - ◆ Configure the mask bits of 24 event lines (EXTI_EMASK);
 - ◆ Configure the trigger configuration bits (EXTI_RT_CFG and EXTI_FT_CFG) of the selected event line.
- Software interrupt/event configuration, select and configure 24 lines as software interrupt/event lines as required:
 - ◆ Configure 24 interrupt/event line mask bits (EXTI_IMASK,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 GPIO mapping



To configure external interrupts/events on the GPIO line through AFIO_EXTI_CFGy, the AFIO clock must be enabled first. The general I/O port is connected to 16 external interrupt/event lines in the way shown above. The other eight EXTI lines are connected as follows:

- EXTI line 16 is connected to PVD output
 - EXTI line 17 is connected to RTC alarm event
 - EXTI line 18 is connected to the RTC timestamp event
 - EXTI line 19 is connected to the RTC wake-up event
 - EXTI line 20 is connected to LPTIM wake-up interrupt

6.3 EXTI Registers

EXTI base address: 0x40010400

6.3.1 EXTI register overview

Table 6-2 EXTI register overview

6.3.2 Interrupt Mask Register(EXTI_IMASK)

Address offset : 0x00

Reset value : 0x00000000

31	Reserved								24	23	22	21	20	19	18	17	16
									IMASK23	IMASK22	IMASK21	IMASK20	IMASK19	IMASK18	IMASK17	IMASK16	
15	14	13	12	11	10	9	8	rw	7	rw	6	rw	5	rw	4	rw	0
IMASK15	IMASK14	IMASK13	IMASK12	IMASK11	IMASK10	IMASK9	IMASK8	IMASK7	IMASK6	IMASK5	IMASK4	IMASK3	IMASK2	IMASK1	IMASK0	rw	rw

Bit field	Name	Description
31:24	Reserved	Reserved, the reset value must be maintained.
23:0	IMASKx	Interrupt mask on line x 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

31	Reserved								24	23	22	21	20	19	18	17	16
									EMASK23	EMASK22	EMASK21	EMASK20	EMASK19	EMASK18	EMASK17	EMASK16	
15	14	13	12	11	10	9	8	rw	7	rw	6	rw	5	rw	4	rw	0

Bit field	Name	Description
31:24	Reserved	Reserved, the reset value must be maintained.
23:0	EMASKx	Event masking on line x 0: Mask 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	RT_CFG21	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	0

Bit field	Name	Description
31:24	Reserved	Reserved, the reset value must be maintained.
23:0	RT_CFGx	The rising edge on line x triggers the configuration bit 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							24	23	22	21	20	19	18	17	16
							FT_CFG23	FT_CFG22	FT_CFG21	FT_CFG20	FT_CFG19	FT_CFG18	FT_CFG17	FT_CFG16	
15	14	13	12	11	10	9	8	rw	rw						
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	Description
31:24	Reserved	Reserved, the reset value must be maintained.
23:0	FT_CFGx	The falling edge on line x triggers the configuration bit. 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 Event Register(EXTI_SWIE)

Address offset : 0x10

Reset value : 0x00000000

31							24	23	22	21	20	19	18	17	16
							SWIE23	SWIE22	SWIE21	SWIE20	SWIE19	SWIE18	SWIE17	SWIE16	
15	14	13	12	11	10	9	8	rw	rw						
SWIE15	SWIE14	SWIE13	SWIE12	SWIE11	SWIE10	SWIE9	SWIE8	SWIE7	SWIE6	SWIE5	SWIE4	SWIE3	SWIE2	SWIE1	SWIE0
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.
23:0	SWIEx	Software interrupt on line x When this bit is '0', writing '1' will set the corresponding pending bit in EXTI_PEND. If this interrupt is allowed in EXTI_IMASK and EXTI_EMASK, an interrupt will be generated at this time. <i>Note: By writing '1' to clear the corresponding bit of EXTI_PEND, this bit can be cleared to '0'.</i>

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
									PEND23	PEND22	PEND21	PEND20	PEND19	PEND18	PEND17	PEND16	
15	14	13	12	11	10	9	8	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	
PEND15	PEND14	PEND13	PEND12	PEND11	PEND10	PEND9	PEND8	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	

Bit field	Name	Description
31:24	Reserved	Reserved, the reset value must be maintained.
23:0	PENDx	<p>Hang bit on line x</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. Write '1' in this bit to clear it, or change the polarity of edge detection to clear this bit.</p>

6.3.8 RTC Timestamp Selection Register(EXTI_TS_SEL)

Address offset : 0x18

Reset value : 0x00000000

31	Reserved								16
15	Reserved								4 3 0 TSSEL[3:0] rw

Bit field	Name	Description
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 8 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.

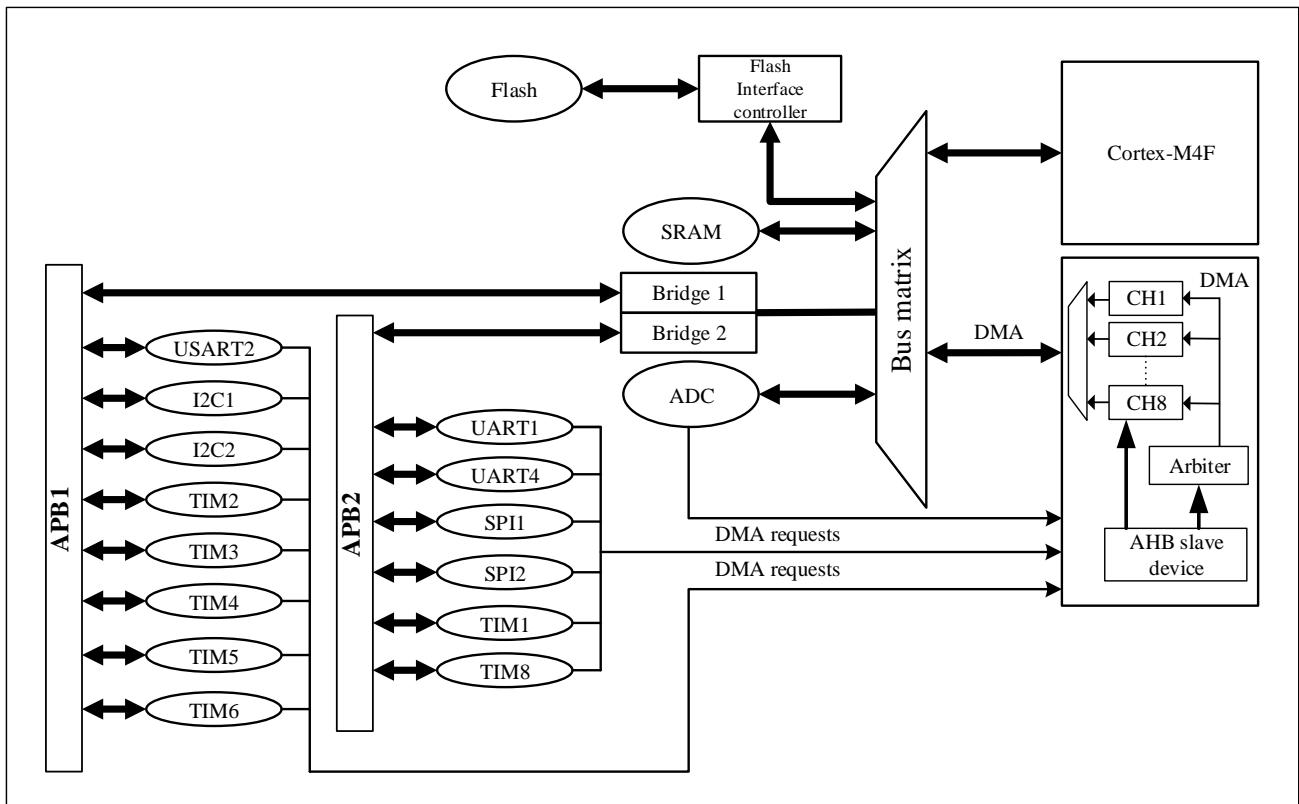
7.2 Main features

DMA main features:

- 8 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 to decide final priority (lower index number channel will has 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 CortexTM-M4F 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 (Only internal source and destination address will be updated, user don't need to know and this won't shows in register).

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 Table 7-1 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 0x1 / B1 0x2 / B2	1: R B0 [7:0] @0x0, W B0 [7:0] @0x0 2: R B1 [7:0] @0x1, W B1 [7:0] @0x1 3: R B2 [7:0] @0x2, W B2 [7:0] @0x2	0x0 / B0 0x1 / B1 0x2 / B2

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

1. Configure interrupt mask bits, 1: enable interrupts, 0 disable interrupts.
2. Configure transfer direction.
3. Configure channel peripheral address and memory address.
4. Configure channel priority, 0: lowest, 3: highest.
5. Configure peripheral and memory address increment.

6. Configure channel transfer block size.
7. If necessary, configure circular mode.
8. 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., 53).
9. Repeat step 1~8 on channel 1~8.
10. 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.

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	APB 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 52 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
sel = 0	ADC
sel = 1	Uart1_tx
sel = 2	Uart1_rx
sel = 3	Usart2_tx
sel = 4	Usart2_rx
sel = 5	Reserved
sel = 6	Reserved
sel = 7	Uart4_tx
sel = 8	Uart4_rx
sel = 9	Spi1_tx
sel = 10	Spi1_rx
sel = 11	Spi2_tx
sel = 12	Spi2_rx
sel = 13	I2c1_tx
sel = 14	I2c1_rx
sel = 15	I2c2_tx
sel = 16	I2c2_rx
sel = 17	Tim1_ch1
sel = 18	Tim1_ch2
sel = 19	Tim1_ch3
sel = 20	Tim1_ch4

sel = 21	Tim1_com
sel = 22	Tim1_up
sel = 23	Tim1_trig
sel = 24	Tim2_ch1
sel = 25	Tim2_ch2
sel = 26	Tim2_ch3
sel = 27	Tim2_ch4
sel = 28	Tim2_up
sel = 29	Tim3_ch1
sel = 30	Tim3_ch3
sel = 31	Tim3_ch4
sel = 32	Tim3_up
sel = 33	Tim3_trig
sel = 34	Tim4_ch1
sel = 35	Tim4_ch2
sel = 36	Tim4_ch3
sel = 37	Tim4_up
sel = 38	Tim5_ch1
sel = 39	Tim5_ch2
sel = 40	Tim5_ch3
sel = 41	Tim5_ch4
sel = 42	Tim5_up
sel = 43	Tim5_trg
sel = 44	Tim6

sel = 45	Tim8_ch1
sel = 46	Tim8_ch2
sel = 47	Tim8_ch3
sel = 48	Tim8_ch4
sel = 49	Tim8_com
sel = 50	Tim8_up
sel = 51	Tim8_trg

7.5 DMA registers

7.5.1 DMA register overview

Table 7-5 DMA register overview

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
ERRF8	HTXF8	TXCF8	GLBF8	ERRF7	HTXF7	TXCF7	GLBF7	ERRF6	HTXF6	TXCF6	GLBF6	ERRF5	HTXF5	TXCF5	GLBF5
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
31/27/23/19/15/11/7/3	ERRFx	<p>Transfer error flag for channel x (x=1...8).</p> <p>Hardware sets this bit when transfer error happen. This bit is cleared by software by writing '1' to DMA_INTCLR.CERRFx bit.</p> <p>0: Transfer error no happened on channel x.</p> <p>1: Transfer error happened on channel x.</p>
30/26/22/18/14/10/6/2	HTXFx	<p>Half transfer flag for channel x (x=1...8).</p> <p>Hardware sets this bit when half transfer is done. This bit is cleared by software by writing '1' to DMA_INTCLR.CHTXFx bit.</p> <p>0: Half transfer not yet done on channel x.</p> <p>1: Half transfer was done on channel x.</p>
29/25/21/17/13/9/5/1	TXCFx	<p>Transfer complete flag for channel x (x=1...8).</p> <p>Hardware sets this bit when transfer is done. This bit is cleared by software by writing '1' to DMA_INTCLR.CTXCFx bit.</p> <p>0: Transfer not yet done on channel x.</p> <p>1: Transfer was done on channel x.</p>
28/24/20/16/12/8/4/0	GLBFx	<p>Global flag for channel x (x=1...8).</p> <p>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.</p> <p>0: No transfer error, half transfer or transfer done event happen on channel x.</p> <p>1: One of transfer error, half transfer or transfer done event happen on channel x.</p>

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
CERRF8	CHTF8	CTXCF8	CGLBF8	CERRF7	CHTF7	CTXCF7	CGLBF7	CERRF6	CHTF6	CTXCF6	CGLBF6	CERRF5	CHTF5	CTXCF5	CGLBF5
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
CERRF4	CHTF4	CTXCF4	CGLBF4	CERRF3	CHTF3	CTXCF3	CGLBF3	CERRF2	CHTF2	CTXCF2	CGLBF2	CERRF1	CHTF1	CTXCF1	CGLBF1
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit field	Name	Description
31/27/23/19/15/11/7/3	CERRFx	<p>Clear transfer error flag for channel x (x=1...8).</p> <p>Software can set this bit to clear ERRF of corresponding channel.</p> <p>0: No action.</p>

Bit field	Name	Description
		1: Reset DMA_INTSTS.ERRF bit of corresponding channel.
30/26/22/18/14/10/6/2	CHTXFx	Clear half transfer flag for channel x (x=1...8). Software can set this bit to clear HTXF of corresponding channel. 0: No action. 1: Reset DMA_INTSTS.HTXF bit of corresponding channel.
29/25/21/17/13/9/5/1	CTXCFx	Clear transfer complete flag for channel x (x=1...8). Software can set this bit to clear TXCF of corresponding channel. 0: No action. 1: Reset DMA_INTSTS.TXCF bit of corresponding channel.
28/24/20/16/12/8/4/0	CGLBFx	Clear global event flag for channel x (x=1...8). 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...8

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

Reset value: 0x0000 0000

31															16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	MEM2MEM	PRIOLVL[1:0]	MSIZE[1:0]	PSIZE[1:0]	MINC	PINC	CIRC	DIR	ERRIE	HTXIE	TXCIE	CHEN			
	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	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 11: Very high
11:10	MSIZE[1:0]	Memory data size. Software can configure data size read/write from/to memory address. 00: 8-bits

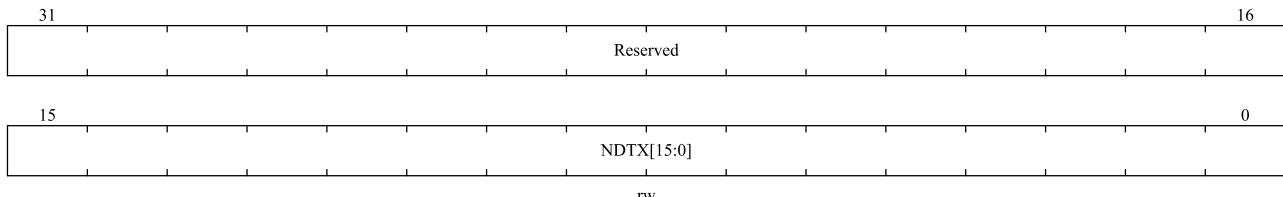
Bit field	Name	Description
		01: 16-bits 10: 32-bits 11: Reserved
9:8	PSIZE[1:0]	Peripheral data size. Software can configure data size read/write from/to peripheral address. 00: 8-bits 01: 16-bits 10: 32-bits 11: Reserved
7	MINC	Memory increment mode. Software can enable/disable memory address increment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

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	<p>Number of data to transfer.</p> <p>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.</p>

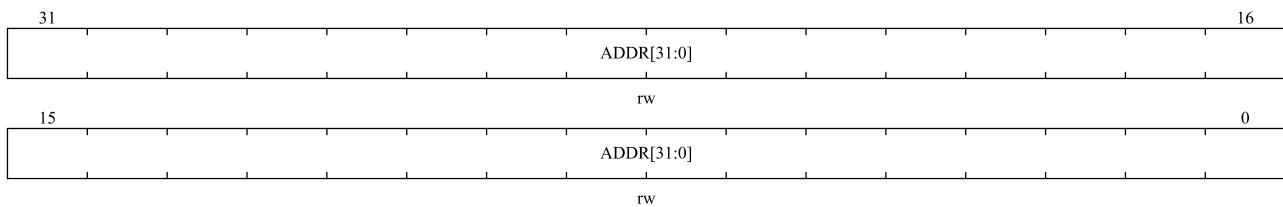
7.5.6 DMA channel x peripheral address register (DMA_PADDRx)

Note: The x is channel number, x = 1...8

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	<p>Peripheral address.</p> <p>Peripheral starting address for DMA to read/write from/to.</p> <p>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.</p>

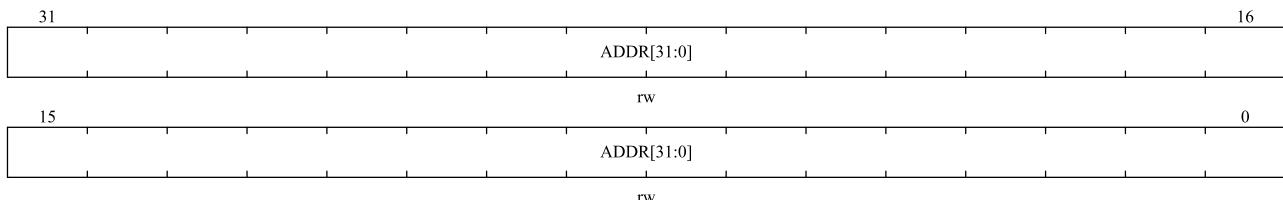
7.5.7 DMA channel x memory address register (DMA_MADDRx)

Note: The x is channel number, x = 1...8

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).



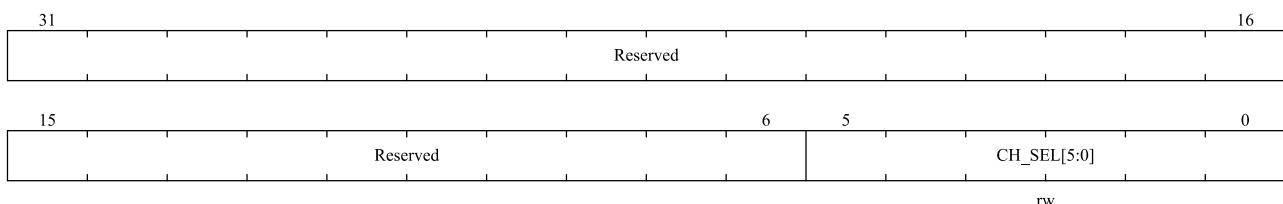
Bit field	Name	Description
31:0	ADDR	ADDR Memory address. Memory starting address for DMA to read/write from/to. Increment of address will be decided by DMA_CHCFGx.MSIZE. With DMA_CHCFGx.MSIZE equal to 01, DMA ignores bit 0 of MADDR and if DMA_CHCFGx.MSIZE equal to 10 DMA will ignore bit [1:0] of MADDR.

7.5.8 DMA channel x channel request select register (DMA_CHSELx)

Note: The x is channel number, x = 1...8

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

Reset value: 0x0000 0000



Bit field	Name	Description
31:6	Reserved	Reserved, the reset value must be maintained.
5:0	CH_SEL[5:0]	DMA channel request selection 0x00: ADC 0x01: UART1_TX 0x02: UART1_RX 0x03: USART2_TX 0x04: USART2_RX 0x05: Reserved

Bit field	Name	Description
		0x06: Reserved 0x07: UART4_TX 0x08: UART4_RX 0x09: SPI1_TX 0x0A: SPI1_RX 0x0B: SPI2_TX 0x0C: SPI2_RX 0x0D: I2C1_TX 0x0E: I2C1_RX 0x0F: I2C2_TX 0x10: I2C2_RX 0x11: TIM1_CH1 0x12: TIM1_CH2 0x13: TIM1_CH3 0x14: TIM1_CH4 0x15: TIM1_COM 0x16: TIM1_UP 0x17: TIM1_TRIG 0x18: TIM2_CH1 0x19: TIM2_CH2 0x1A: TIM2_CH3 0x1B: TIM2_CH4 0x1C: TIM2_UP 0x1D: TIM3_CH1 0x1E: TIM3_CH3 0x1F: TIM3_CH4 0x20: TIM3_UP 0x21: TIM3_TRIG 0x22: TIM4_CH1 0x23: TIM4_CH2 0x24: TIM4_CH3 0x25: TIM4_UP 0x26: TIM5_CH1 0x27: TIM5_CH2 0x28: TIM5_CH3 0x29: TIM5_CH4 0x2A: TIM5_UP 0x2B: TIM5_TRIG 0x2C: TIM6 0x2D: TIM8_CH1 0x2E: TIM8_CH2 0x2F: TIM8_CH3

Bit field	Name	Description
		0x30: TIM8_CH4 0x31: TIM8_COM 0x32: TIM8_UP 0x33: TIM8_TRIG

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

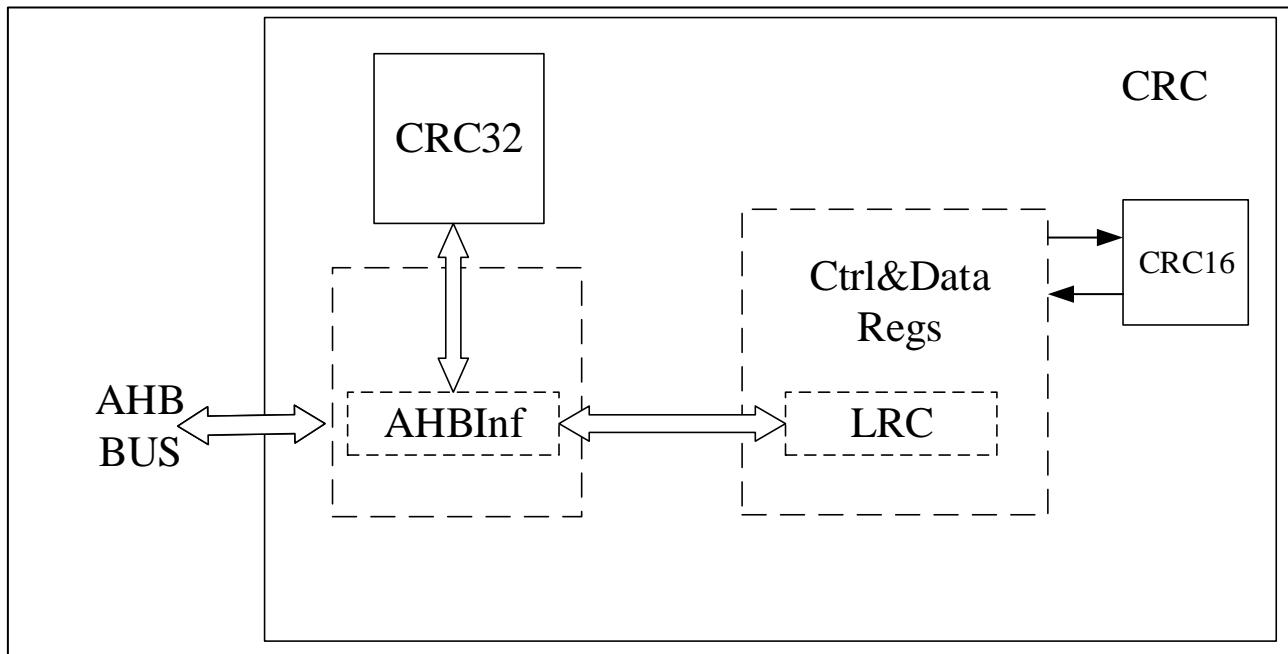
- CRC32($X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^8+X^7+X^5+X^4+X^2+X+1$)
- 32 bits of data to be checked and 32 bits of output check code.
- CRC calculation time: 1 AHB clock cycle (HCLK)
- General-purpose 8-bit register (can be used to store temporary data)

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 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).

BUS writing operation is to be hold during CRC calculation, thus back-back writing or writing-reading is supported.

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.ENDHL 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.

LCR 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.

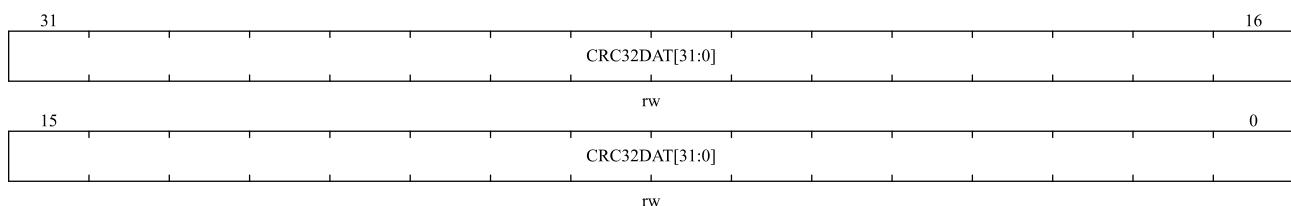
Table 8-1 CRC register overview

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																																	0							
00Ch	CRC16CTRL																														CLR	ENDHL	Reserved								
	Reset Value																																0	0							
010h	CRC16DAT																																	CRC16DAT[7:0]							
	Reset Value																																	0	0	0	0	0	0	0	0
014h	CRC16D																																	CRC16D[15:0]							
	Reset Value																																	0	0	0	0	0	0	0	0
018h	LRC																																	LRCDAT[7:0]							
	Reset Value																																	0	0	0	0	0	0	0	0

8.4.2 CRC32 data register (CRC_CRC32DAT)

Address offset: 0x00

Reset value: 0xFFFF FFFF

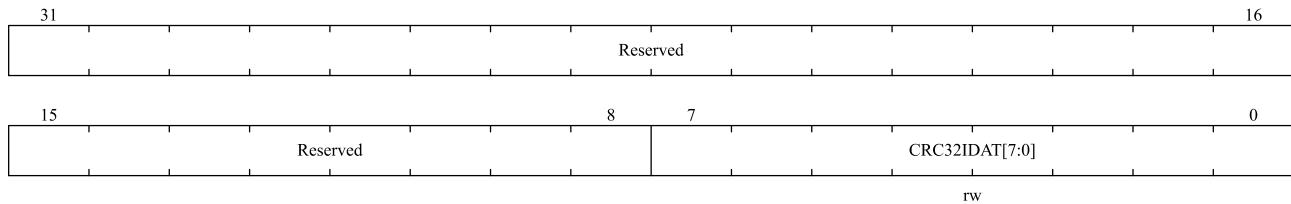


Bit field	Name	Description
31:0	CRC32DAT[31:0]	CRC32 Data register. The written data is the CRC value to be checked. The read data is the CRC calculation result. Only 32-bit operations are supported.

8.4.3 CRC32 independent data register (CRC_CRC32IDAT)

Address offset: 0x04

Reset value: 0x0000 0000



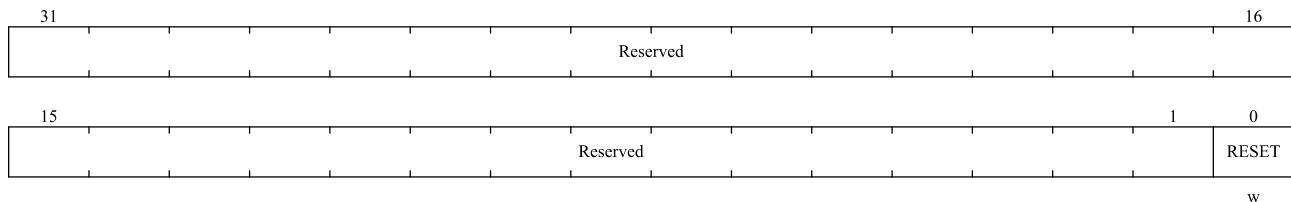
Bit field	Name	Description
31:8	Reserved	Reserved, the reset value must be maintained.
7:0	CRC32IDAT[7:0]	Independent 8-bit data register. General 8 bits data register. It is for temporary stored 1-byte data. CRC_ CRC32CTRL.RESET reset signal will not impact this register.

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

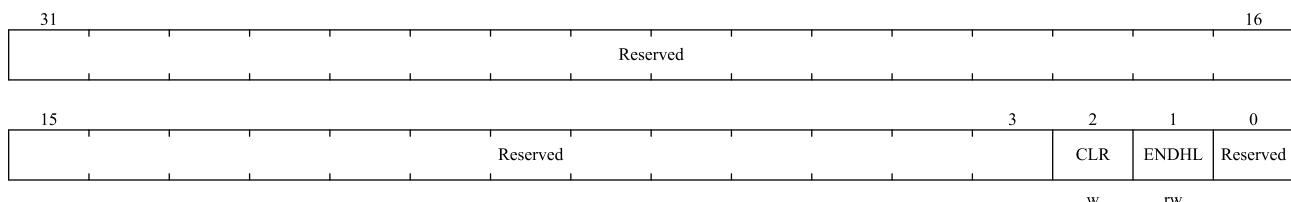


Bit field	Name	Description
31:1	Reserved	Reserved, the reset value must be maintained.
0	RESET	RESET signal. It can reset CRC module and set data register to be 0xFFFF_FFFF. This reset can only write 1, and hardware will clear to 0 automatically.

8.4.5 CRC16 control register (CRC_CRC16CTRL)

Address offset: 0x0C

Reset value: 0x0000 0000



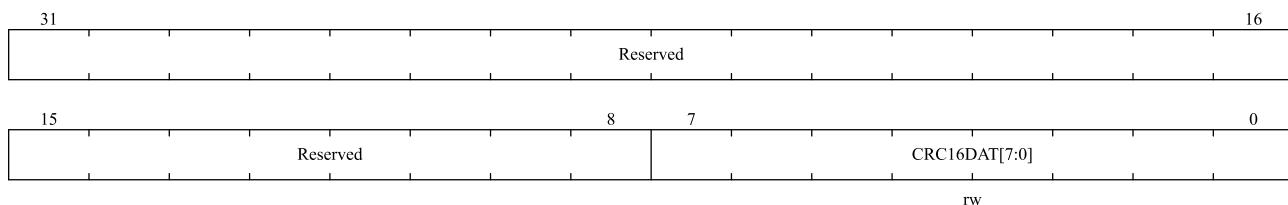
Bit field	Name	Description
31:3	Reserved	Reserved, the reset value must be maintained.
2	CLR	Clear CRC16 results. 0: Not clear. 1: Clear to default value 0x0000. Set this bit to 1 will only maintain 1 clock cycle, hardware will clear automatically. (Software read always 0).
1	ENDHL	Data to be verified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



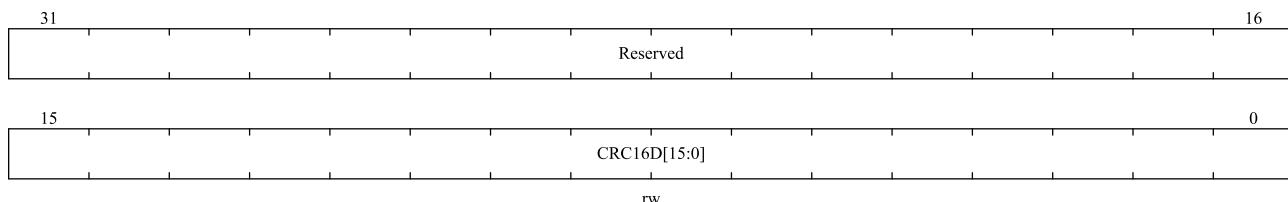
Bit field	Name	Description
31:8	Reserved	Reserved, the reset value must be maintained.
7:0	CRC16DAT[7:0]	Data to be verified.

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.

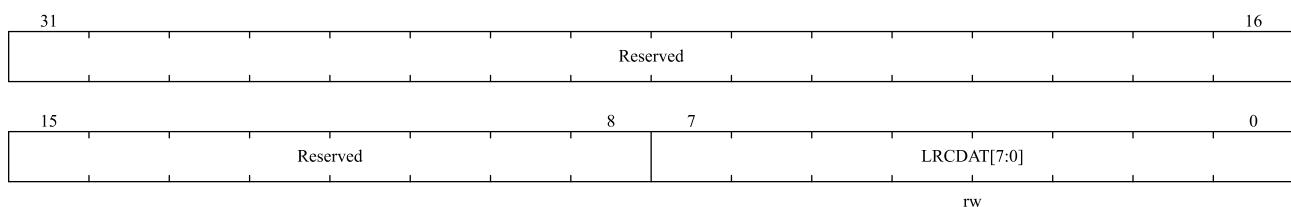
Bit field	Name	Description
15:0	CRC16D[15:0]	16-bit value of cyclic redundancy result data. Every time the software writes the CRC16DAT register, the 16-bit calculated data from CRC16 is updated in this register.

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 CRC_CRC16DAT will be XOR with CRC_LCR register value. The result will be stored in CRC_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

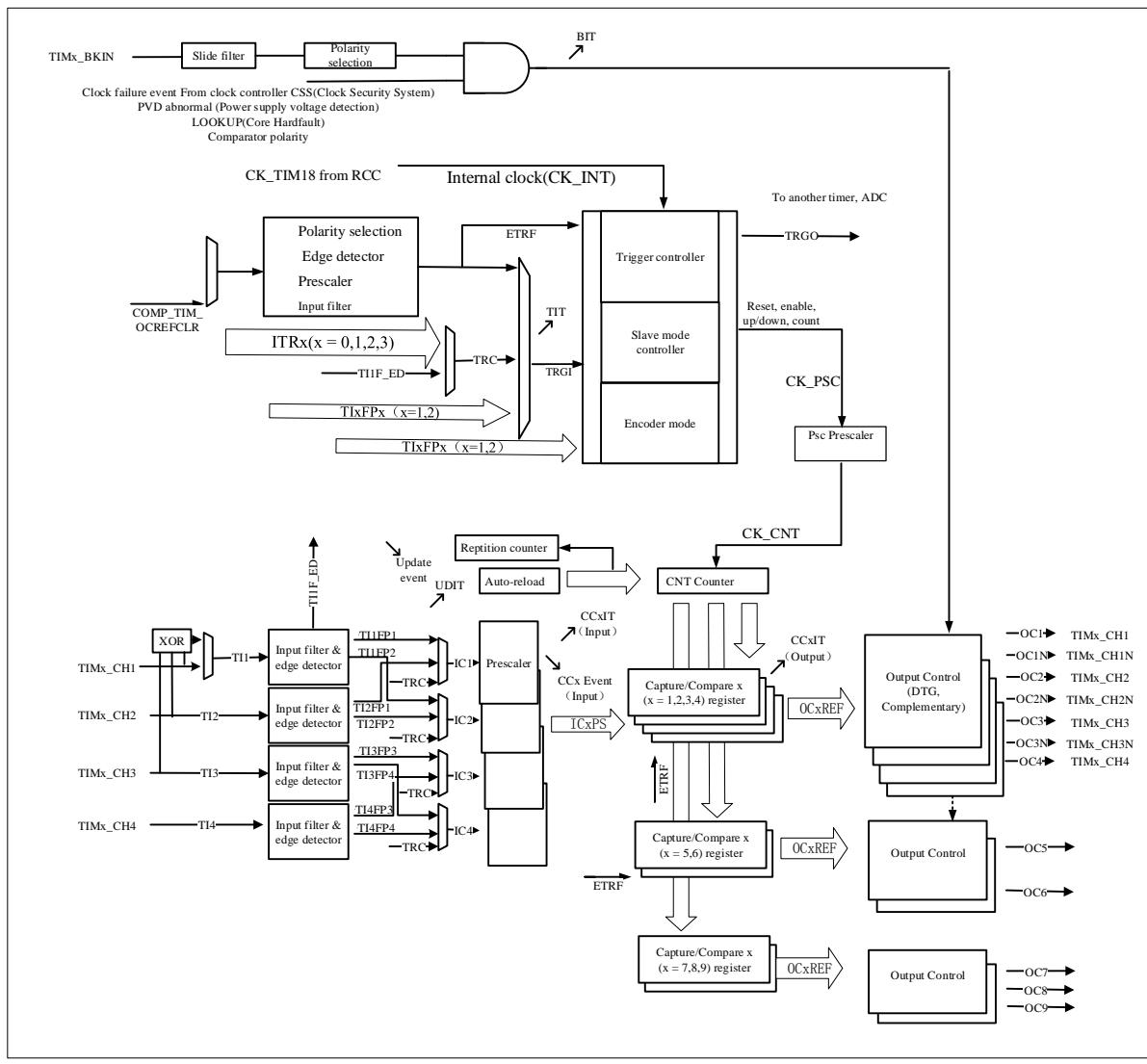
The advanced control timers (TIM1 and TIM8) is mainly used in the following purposes: counting the input signal, measuring the pulse width of the input signal and generating the output waveform, etc.

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

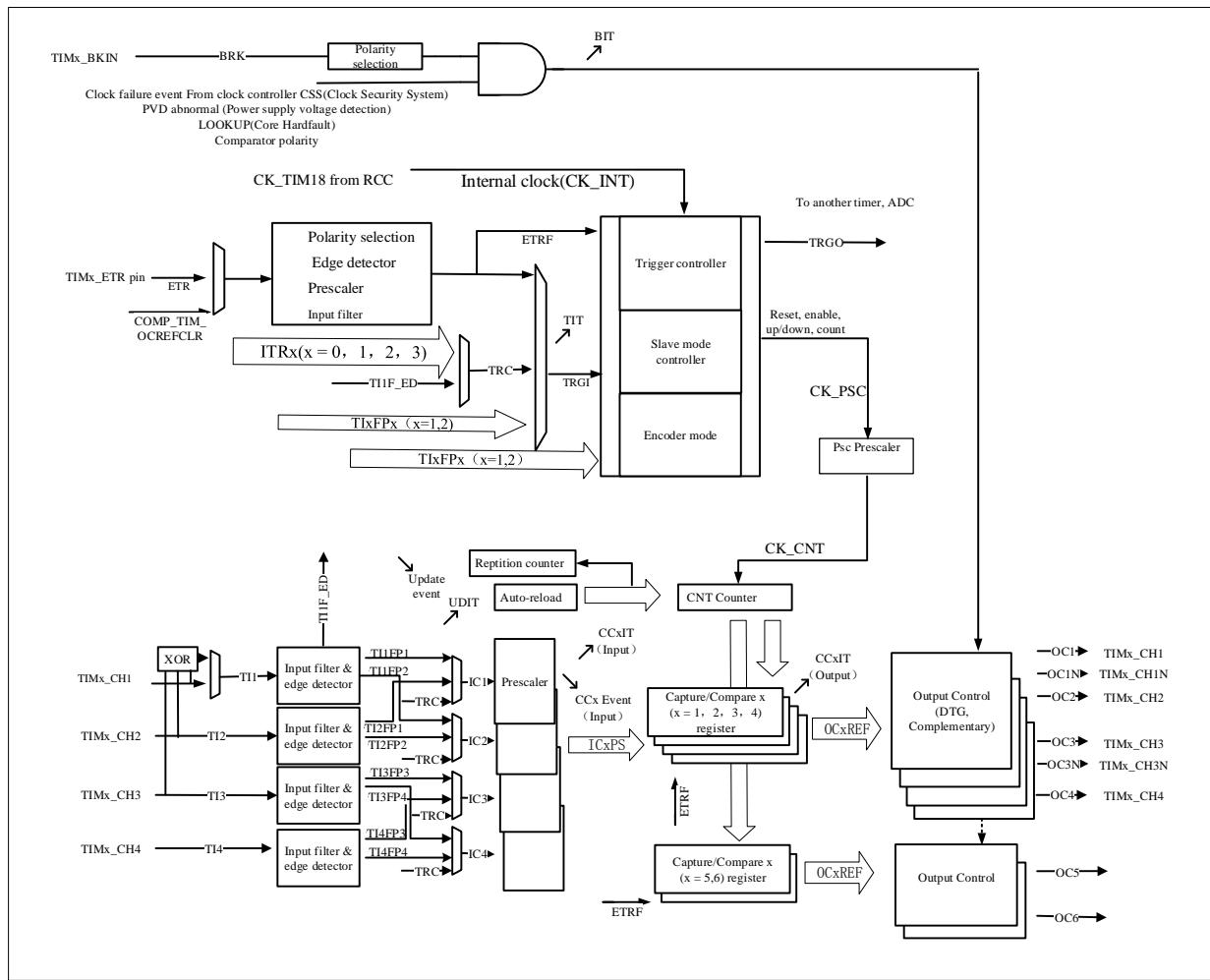
- 16-bit auto-reload counters. (It can realize up-counting, down-counting, up/down counting). TIM1 support center-aligned asymmetric mode.
- 16-bit programmable prescaler. (The frequency division factor can be configured with any value between 1 and 65536)
- Programmable Repetition Counter
- TIM1 up to 9 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 can be linked together internally for timer synchronization or chaining
- For TIM1, the pulse signal output by channel 4/7/8/9 is configured as the rising and falling edges to trigger the ADC.
- 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



The capture channel 1 input can come from IOM

Figure 9-2 Block diagram of TIM8



The capture channel 1 input can come from IOM

9.3 TIM1 and TIM8 function description

9.3.1 Time-base unit

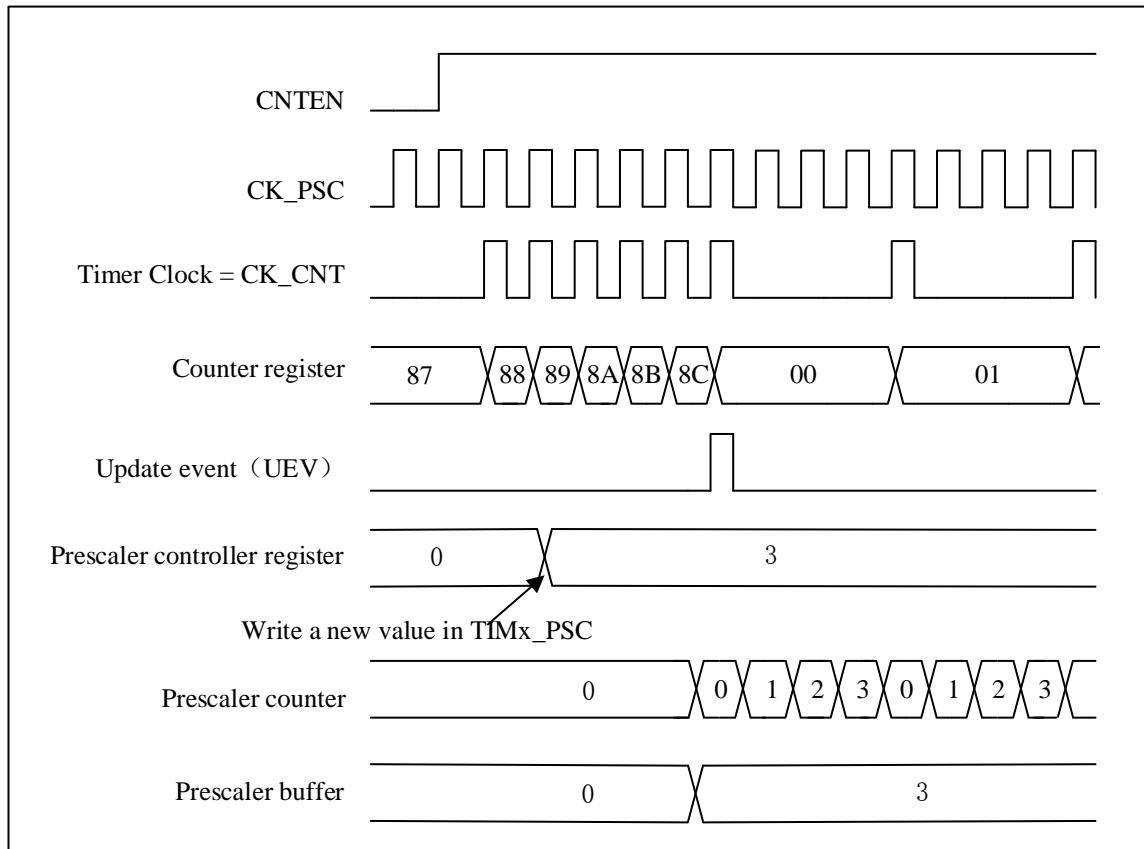
The advanced-control's time-base unit mainly includes: prescaler, counter, auto-reload and repetition counter. When the time base unit is working, the software can read and write the corresponding registers (TIMx_PSC, TIMx_CNT, TIMx_AR and TIMx_REPCNT) at any time.

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 9-3 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.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 9-4 Timing diagram of up-counting. The internal clock divider factor = 2/N

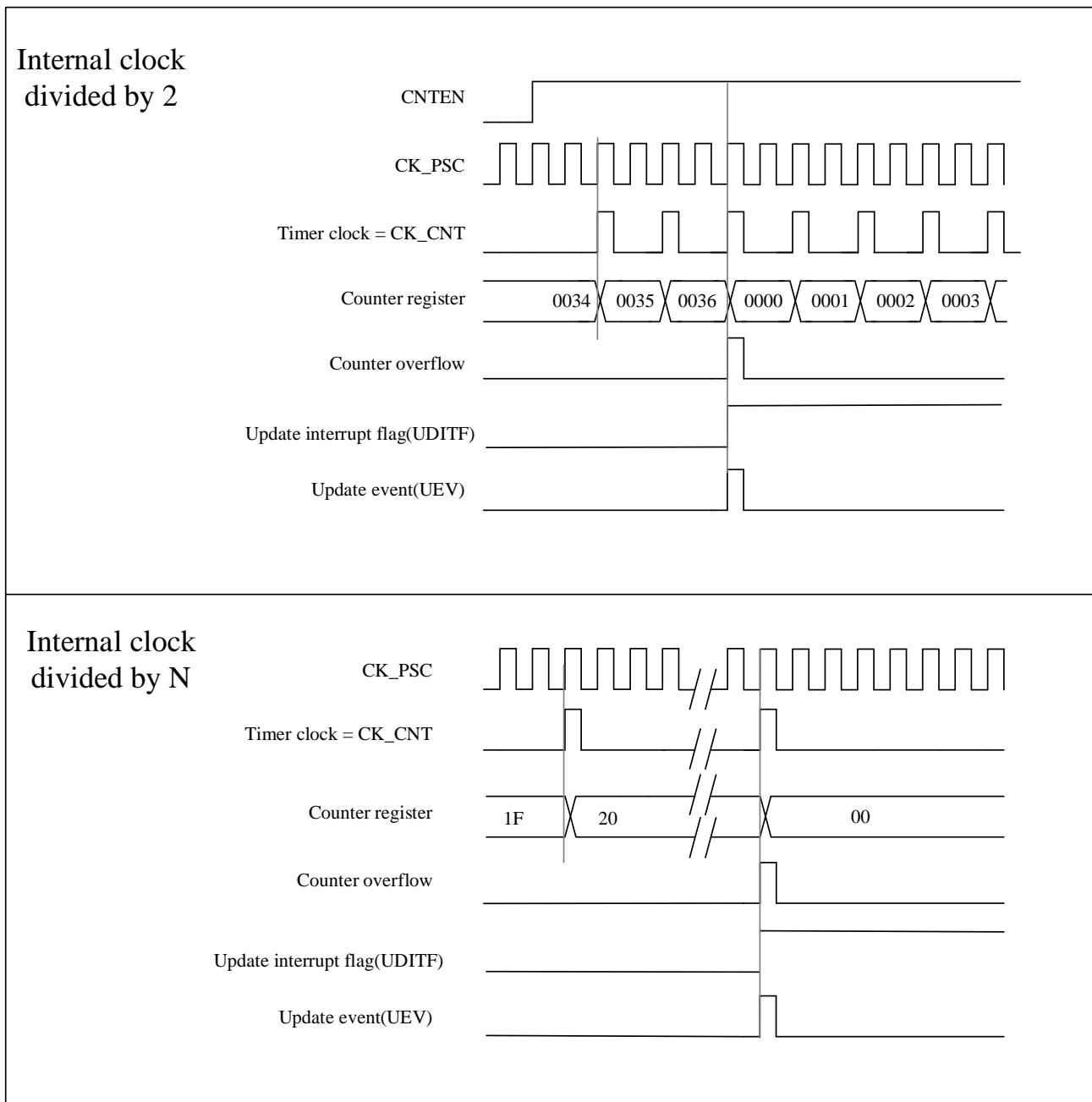
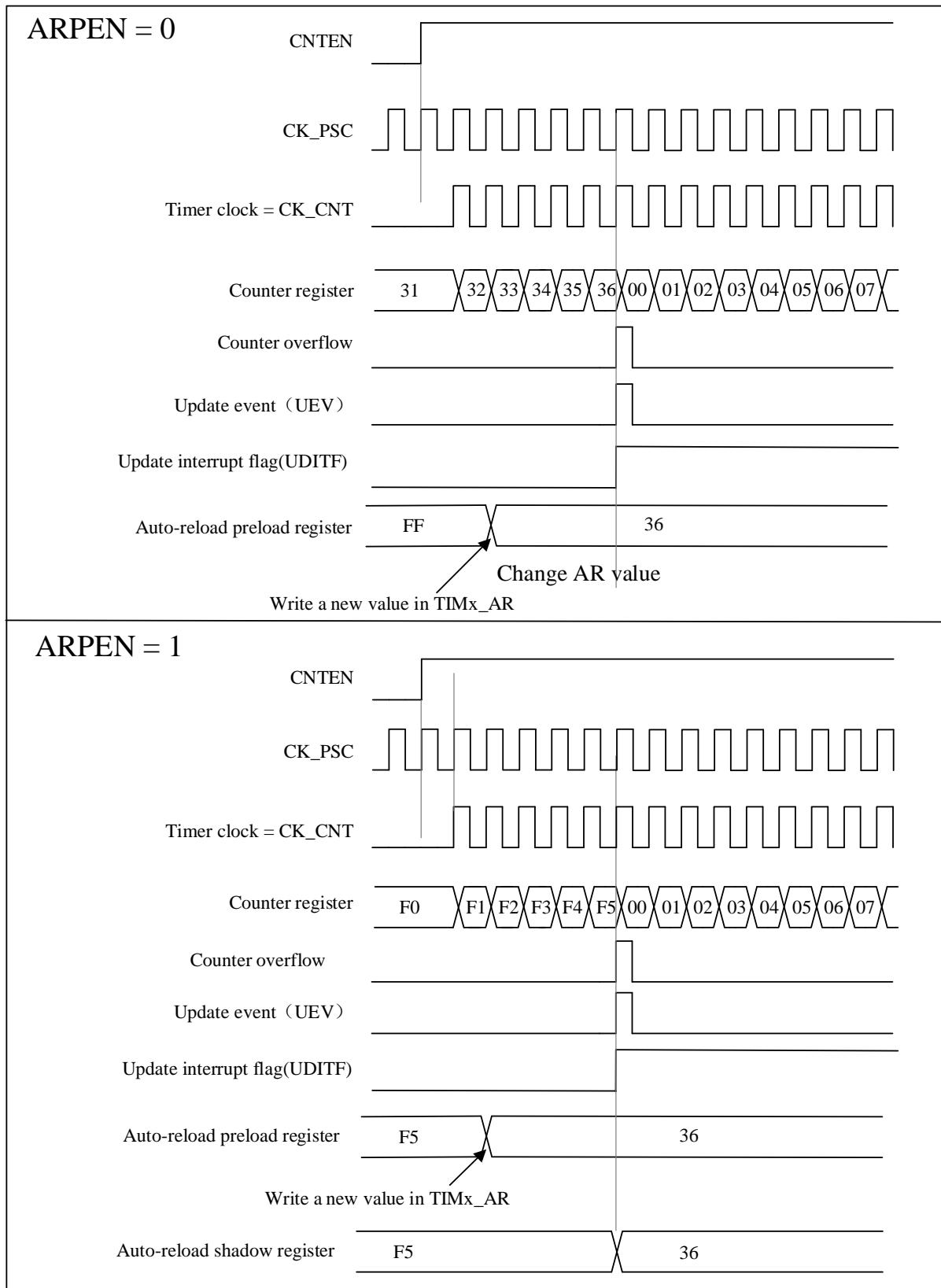


Figure 9-5 Timing diagram of the up-counting, update event when ARPEN=0/1



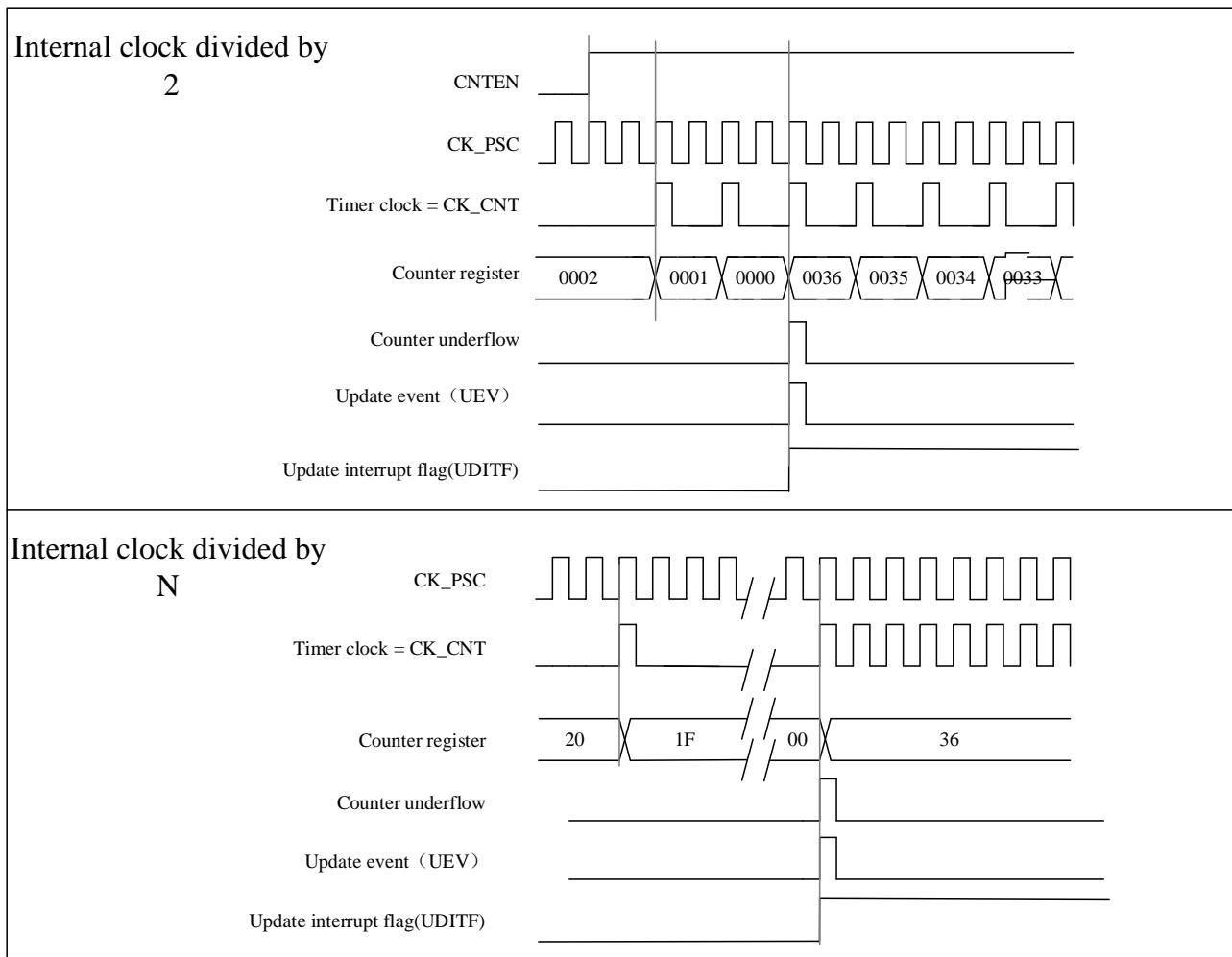
9.3.2.2 Down-counting mode

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 9-6 Timing diagram of the down-counting, internal clock divided factor = 2/N



9.3.2.3 Center-aligned mode

9.3.2.3.1 Center-aligned symmetric mode

In center-aligned symmetric 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 9-7 Timing diagram of the Center-aligned, internal clock divided factor =2/N

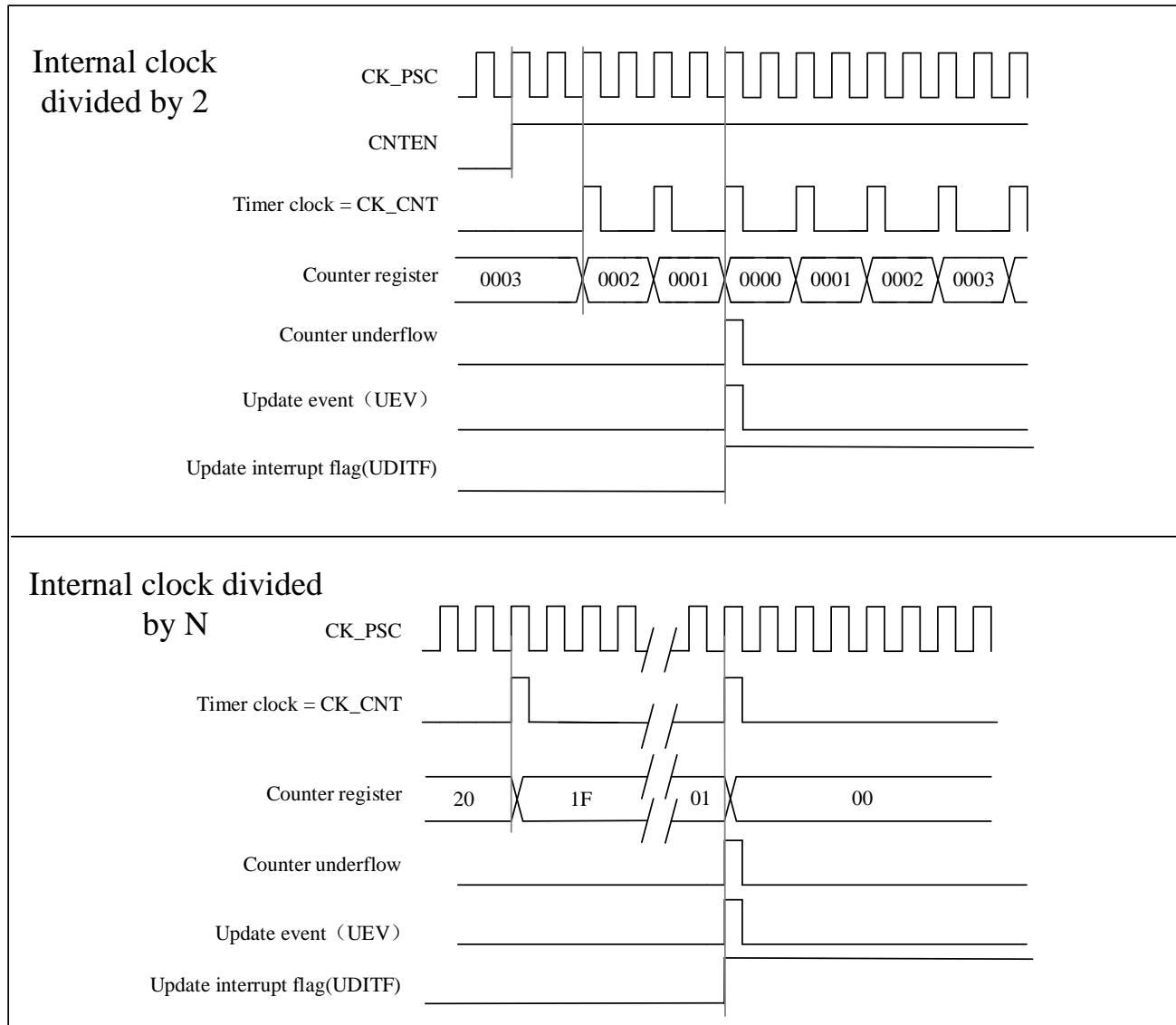
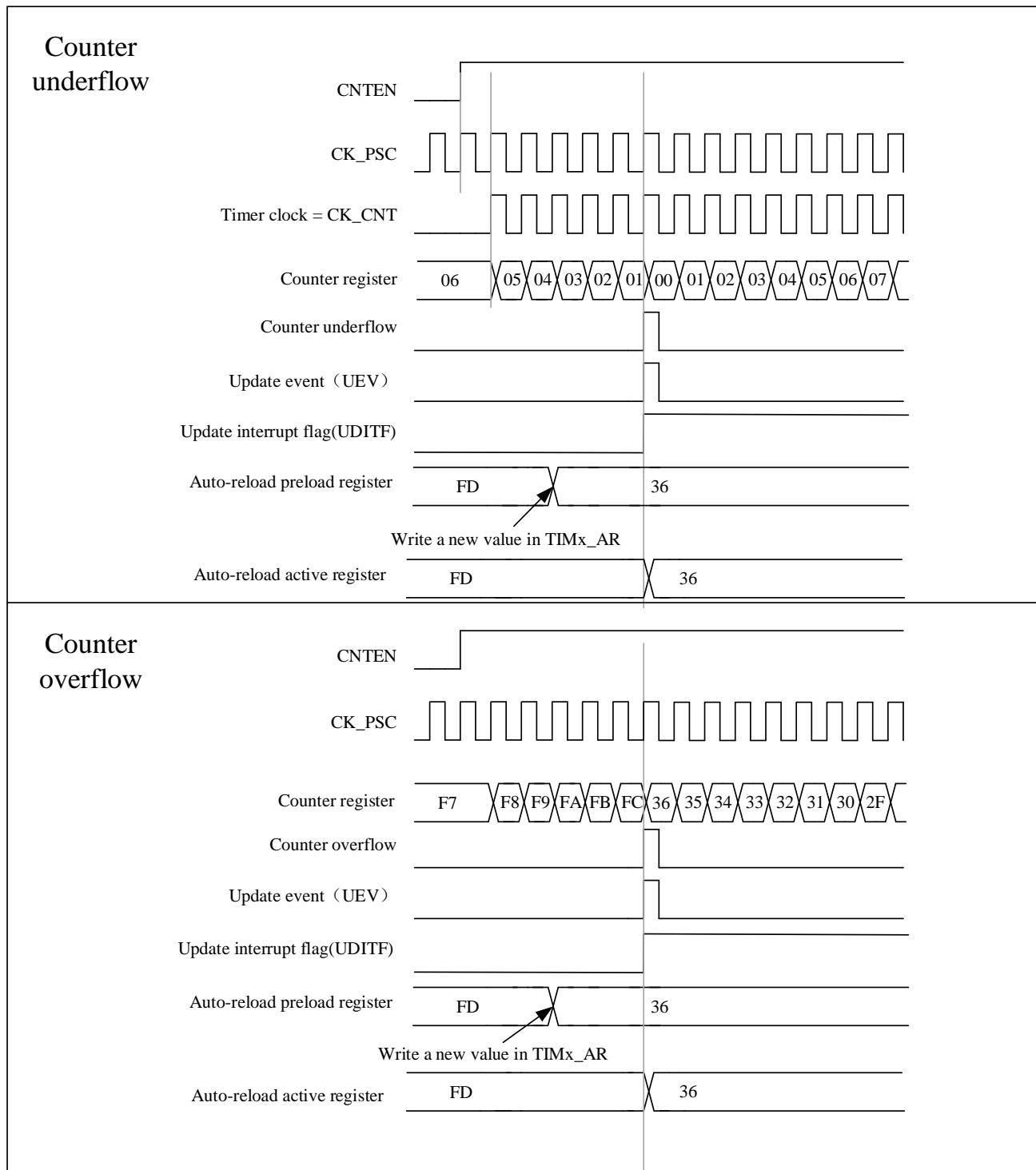


Figure 9-8 A center-aligned sequence diagram that includes counter overflows and underflows (ARPEN = 1)



9.3.2.3.2 Center-aligned asymmetric mode

Center-aligned asymmetric mode is only for TIM1.

In center-aligned asymmetric mode(TIMx_CTRL1.ASYMMETRIC is 1 and TIMx_CTRL1.CAMSEL[1:0] is non-

zero), the counter counts from 0 to the auto-reload value (TIMx_AR) – 1 and generates a counter overflow event, then counts from the auto-reload value down to 1 and generates a counter underflow event and then restarts counting from 0.

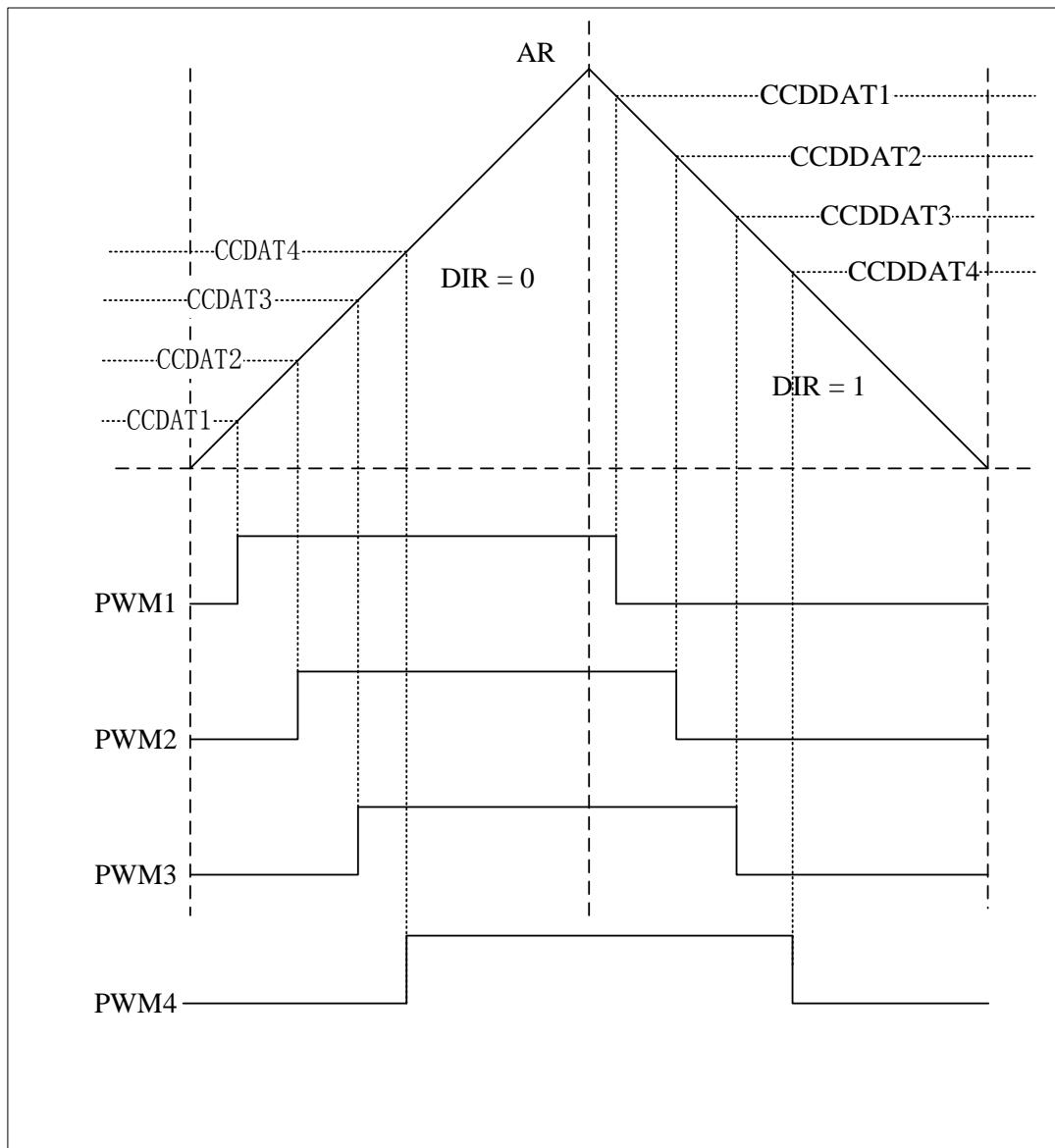
The TIMx_CTRL1.DIR cannot be written in this mode. It is updated by hardware and indicates the current direction of the counter.

When the channel is not 1,2,3,4, the comparison value are compared with CC DATx. When the dead time generator is turned on, note that when DIR = 0, the dead time insertion point is at which the counter value is equal to CC DATx(x=1,2,3,4), and when DIR = 1, the dead time insertion point is at which the counter value is equal to CC D DATx(x=1,2,3,4).

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.

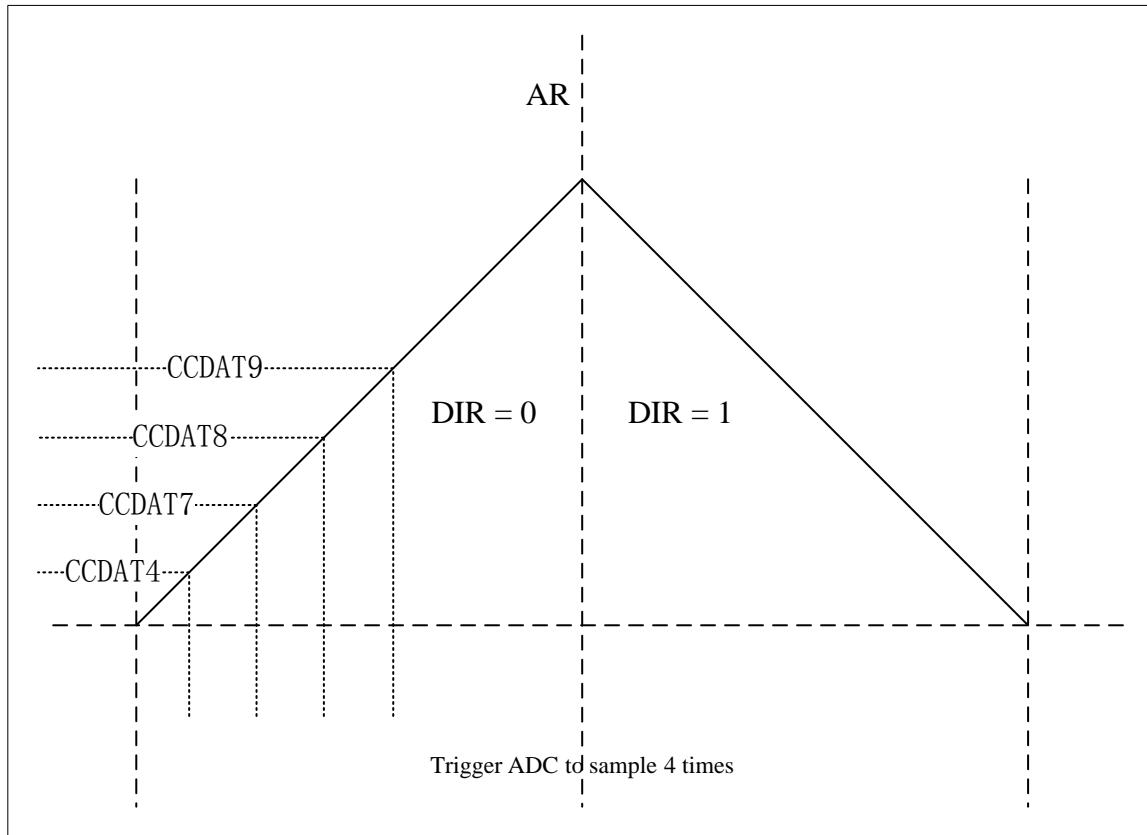
Figure 9-9 The output waveform corresponding to the asymmetric mode



Since the triggering function of CC7/CC8/CC9 three channels is added, and the triggering function of CC4 has been modified, the description of CC4/CC7/CC8/CC9 channels to ADC triggering is now described.

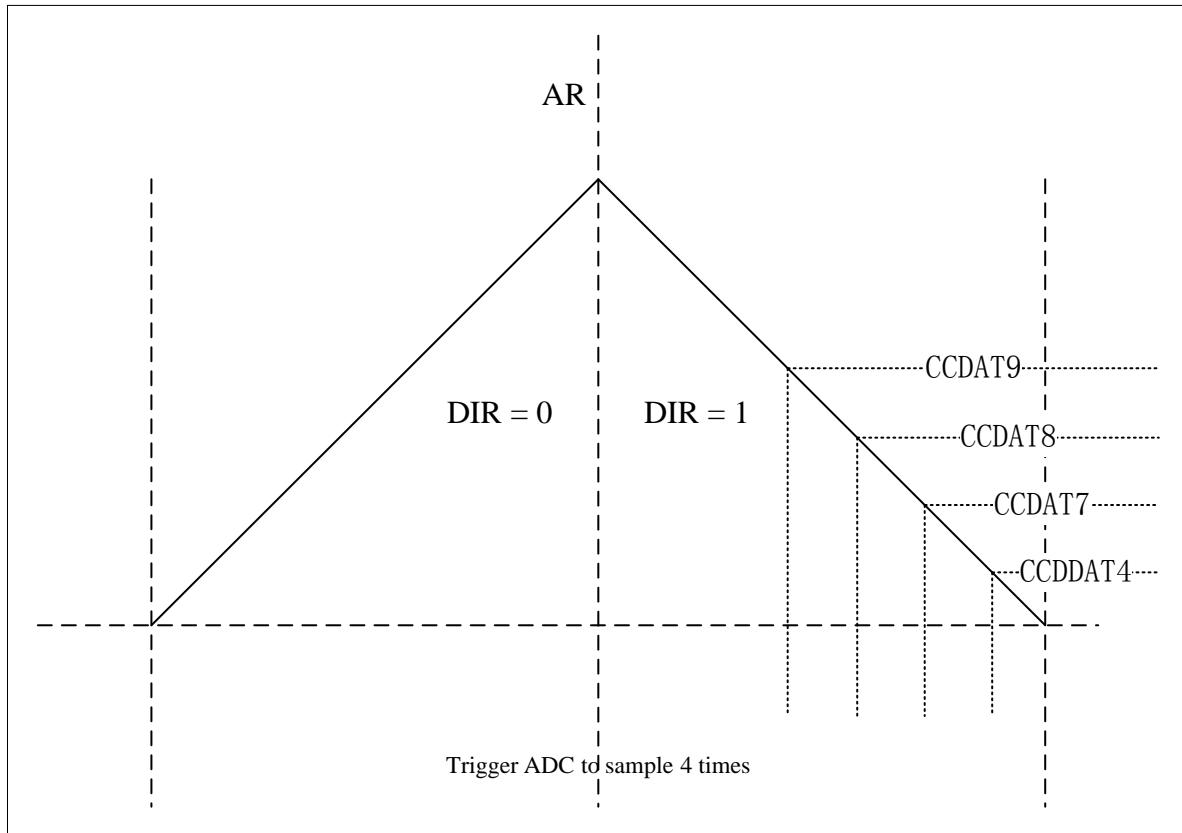
When the timer is working in center-aligned asymmetric mode, each channel(CC4/CC7/CC8/CC9) can individually trigger the ADC only when MMSEL3 = 1. If TIMx_CTRL1.CMODE[1:0]=00 in CCDATx(x=4,7,8,9), the CCDAT value of CCDDATx will only trigger ADC when DIR =0.

Figure 9-10 CCDATx(x=4,7,8,9), trigger ADC when DIR = 0



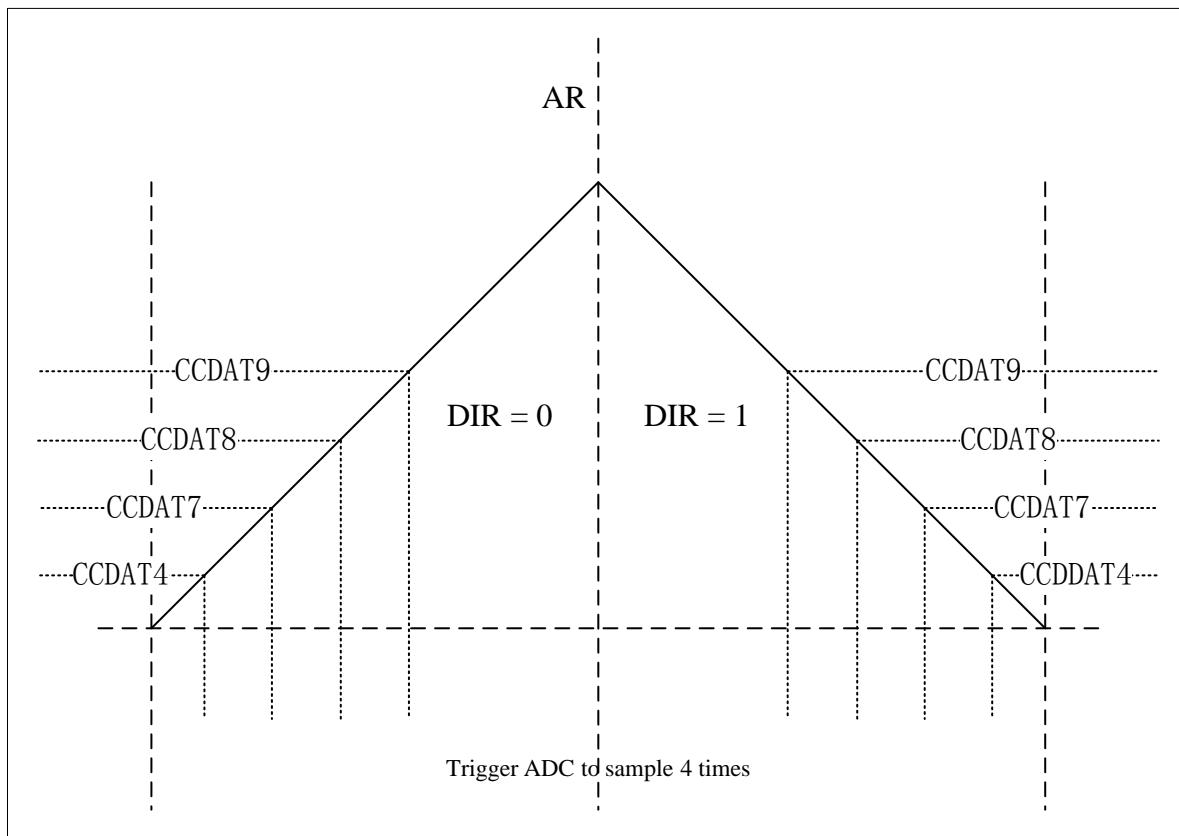
If TIMx_CTRL1.CMODE[1:0]=01 in CCDATx(x=4,7,8,9), the CCDAT/CCDDAT value of CCDATx will only trigger ADC when DIR =1.

Figure 9-11 CCDDATx(x=4,7,8,9), trigger ADC when DIR = 1



If TIMx_CTRL1.CMODE[1:0]=1x in CCDDATx(x=4,7,8,9), the CCDDAT/CCDDAT value of CCDDATx will trigger ADC when DIR =0 or DIR=1.

Figure 9-12 CCDDATx(x=4,7,8,9), trigger ADC when DIR = 0 or DIR = 1



In the preceding figure, channel 4 up counting to CCDDAT4 or down counting to CCDDAT4, channel 7/8/9 up counting or down counting to CCDAT7/8/9, trigger valid.

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.

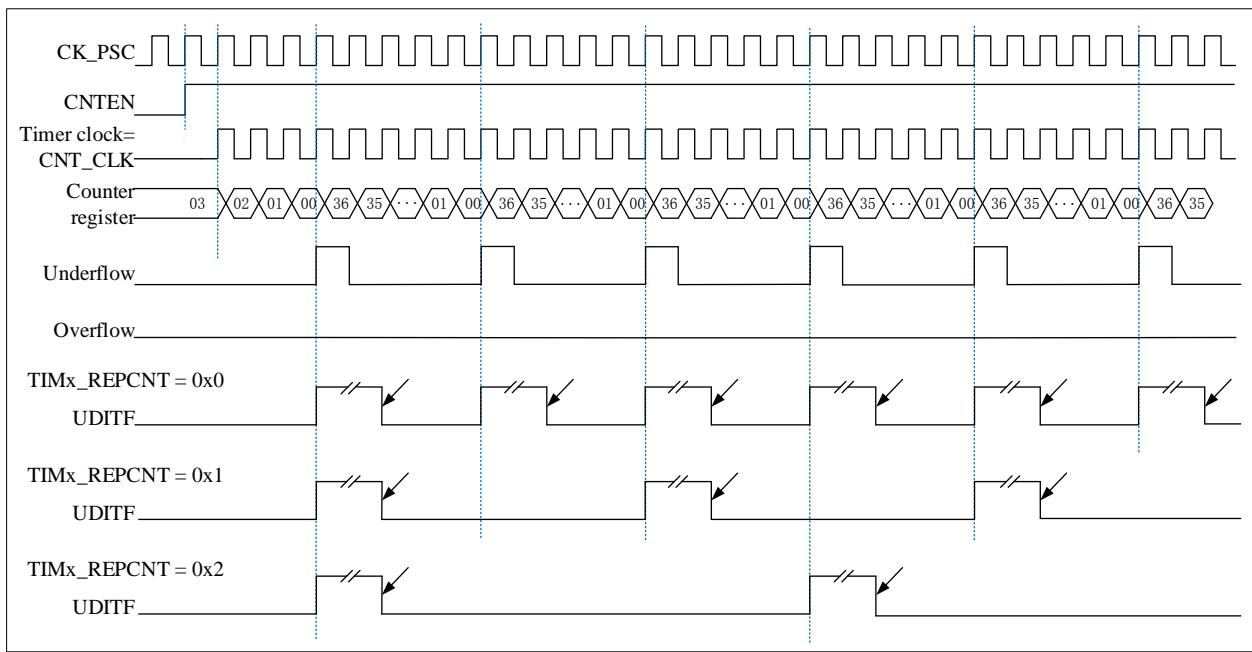
The repetition counter is decremented:

- 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.

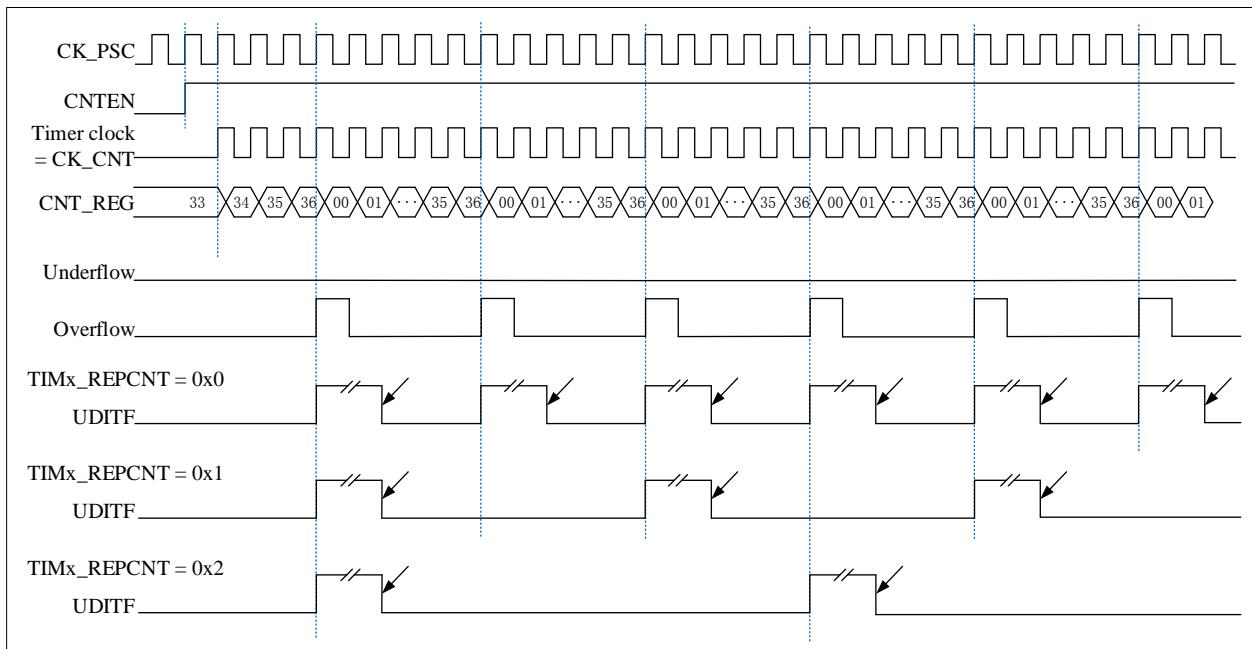
Repetition counters feature automatic reloading. The update event (generated by setting TIMx_EVTGEN.UDGN or hardware through slave mode controller) occurs immediately, regardless of the value of the repeat counter.

Figure 9-13 Repeat count sequence diagram in down-counting mode



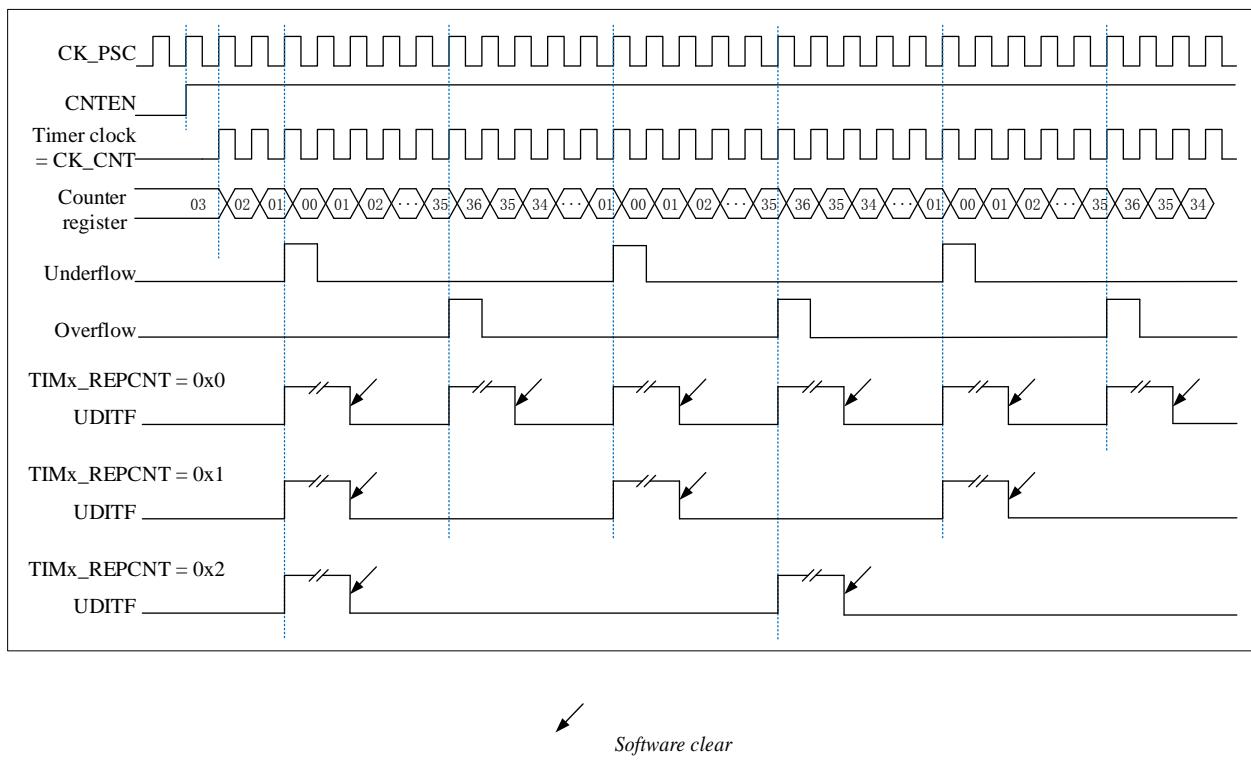
Software clear

Figure 9-14 Repeat count sequence diagram in up-counting mode



Software clear

Figure 9-15 Repeat count sequence diagram in center-aligned mode



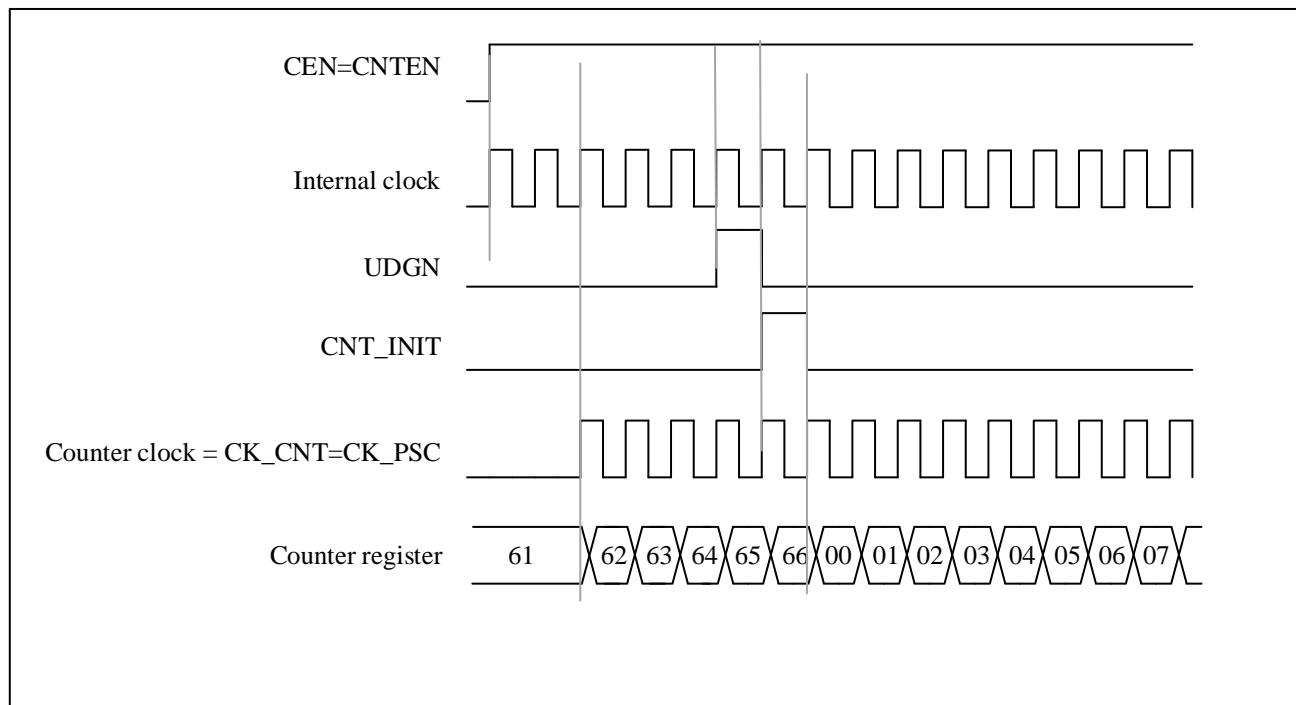
9.3.4 Clock selection

- 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)

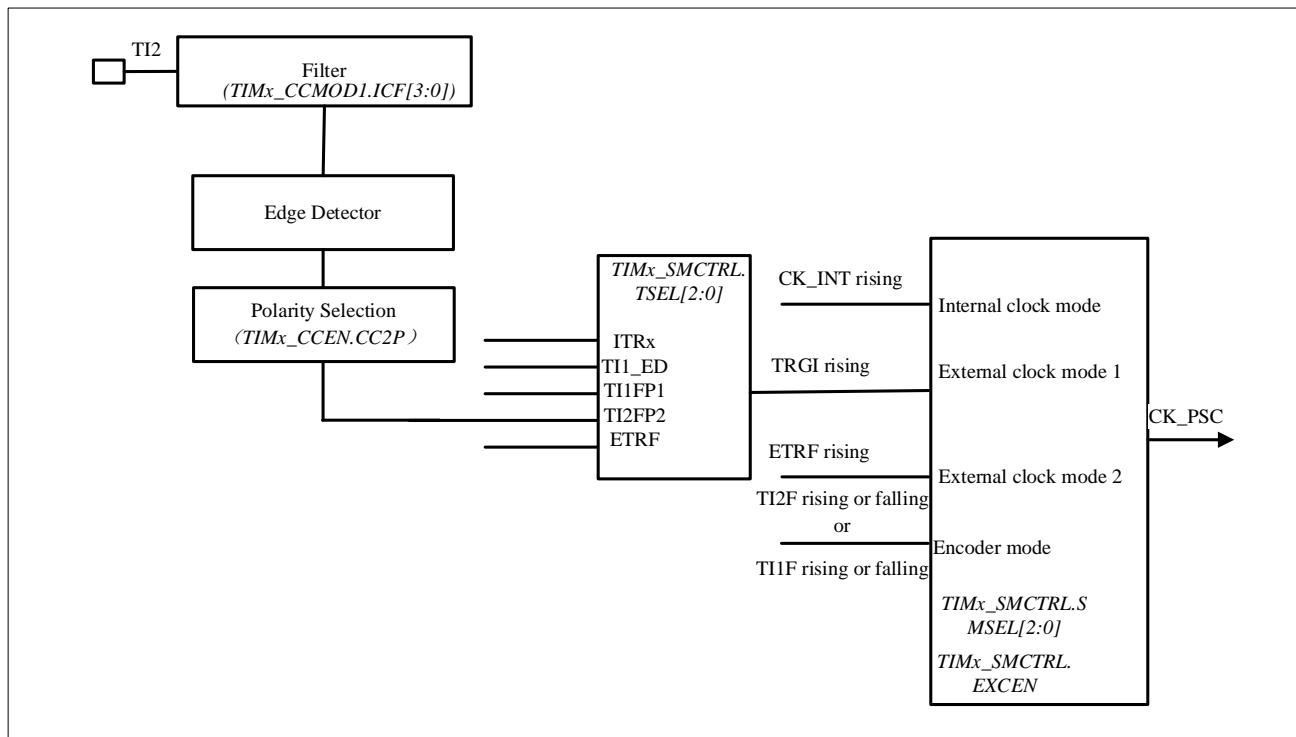
When the TIMx_SMCTRL.SMSEL is equal to “000”, the slave mode controller is disabled. The three control bits (TIMx_CTRL1.CNTEN, TIMx_CTRL1.DIR, TIMx_EVTGEN.UDGN) can only be changed by software (except TIMx_EVTGEN.UDGN, which remains cleared automatically). It is provided that the TIMx_CTRL1.CNTEN bit is written as '1' by soft, the clock source of the prescaler is provided by the internal clock CK_INT.

Figure 9-16 Control circuit in normal mode, internal clock divided by 1



9.3.4.2 External clock source mode 1

Figure 9-17 TI2 external clock connection example



This mode is selected by configuring **TIM_x_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:

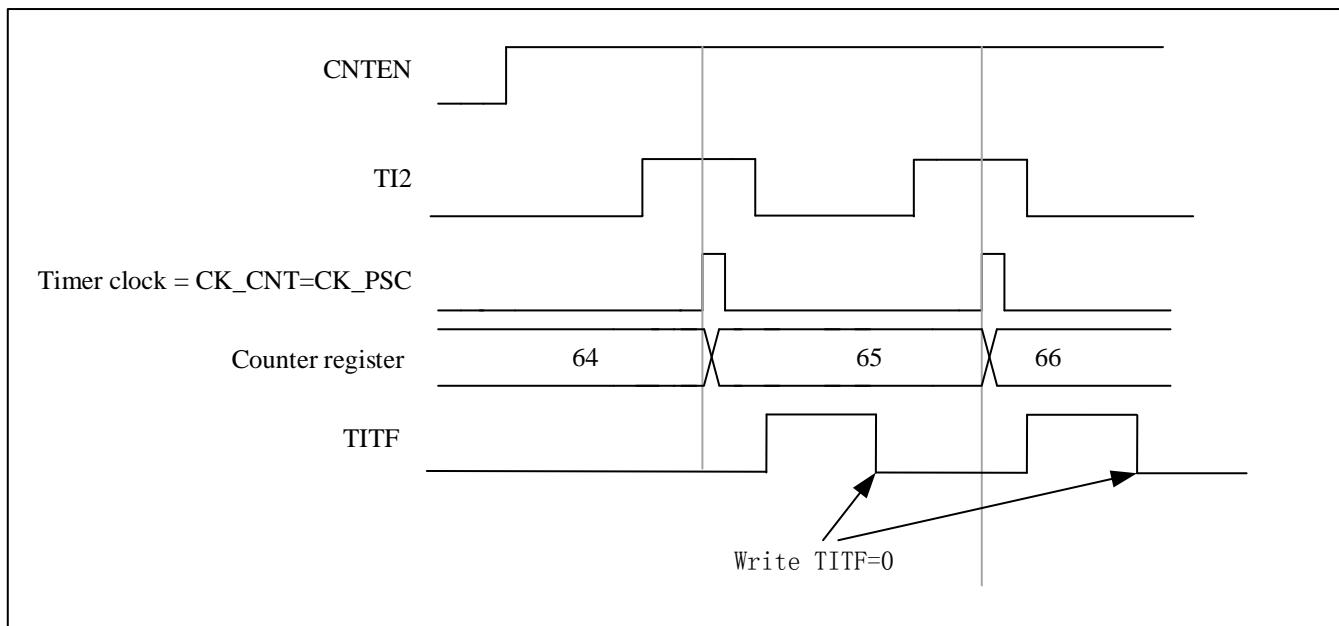
- Configure TIMx_CCMOD1.CC2SEL equal to '01', CC2 channel is configured as input, IC2 is mapped to TI2
- Configure TIMx_CCEN.CC2P equal to '0', select clock rising edge polarity
- To select input filter bandwidth by configuring TIMx_CCMOD1.IC2F[3:0] (if filter is not needed, keep IC2F bit at '0000')
- 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 9-18 Control circuit in external clock mode 1

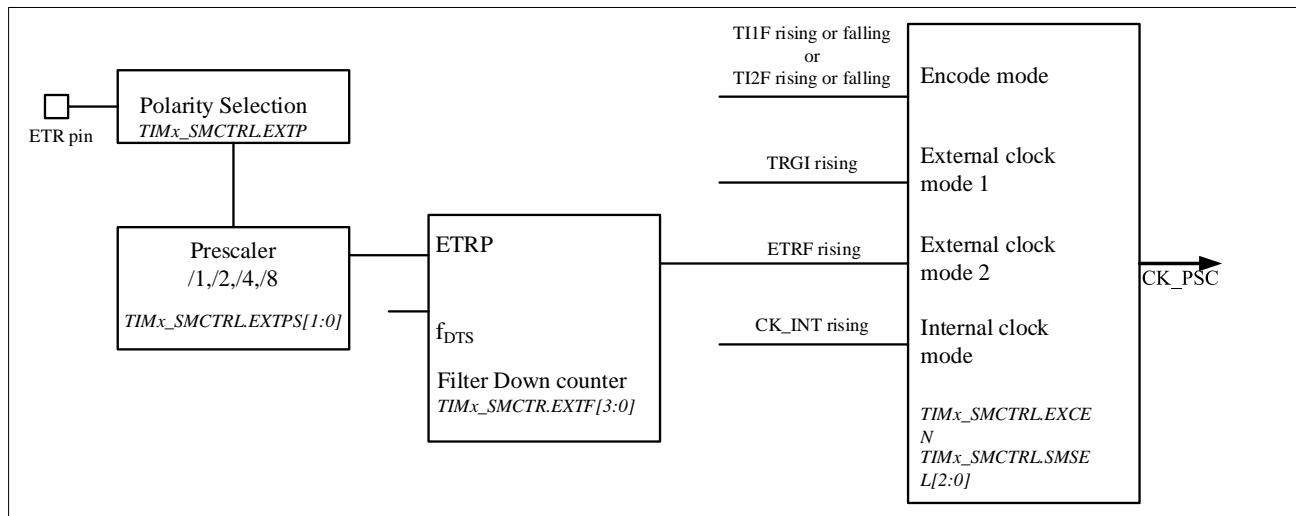


9.3.4.3 External clock source mode 2

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 9-19 External trigger input block diagram

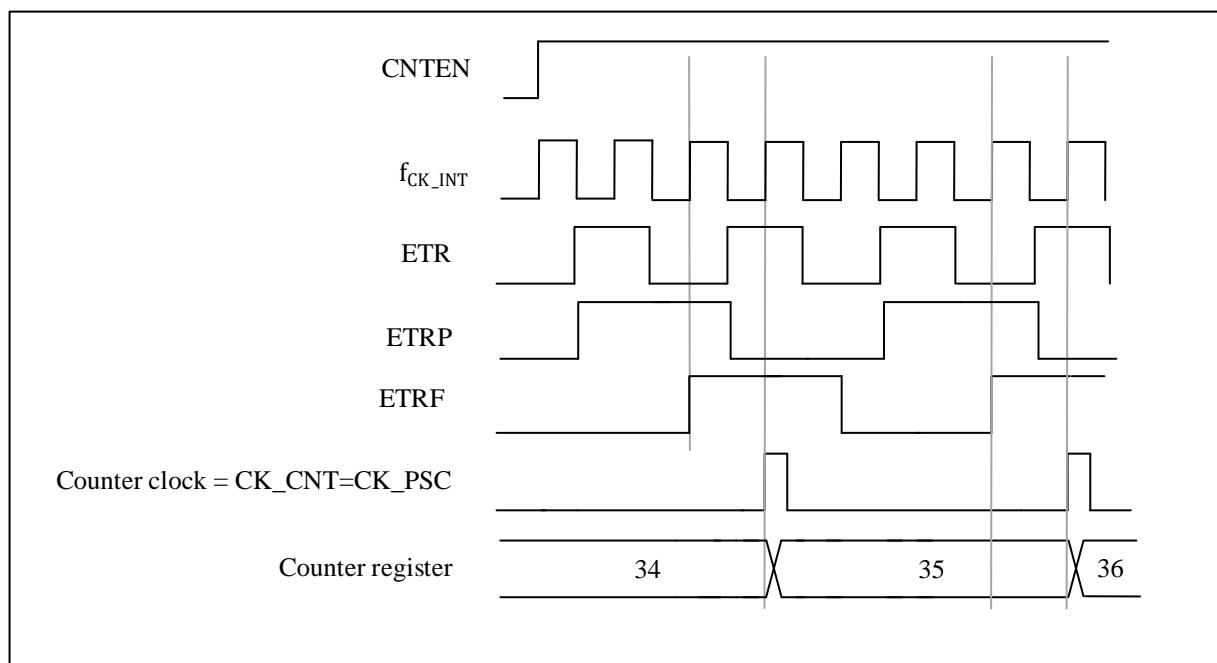


For example, use the following configuration steps to make the up counter count every 2 rising edges on ETR.

- 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 9-20 Control circuit in external clock mode 2

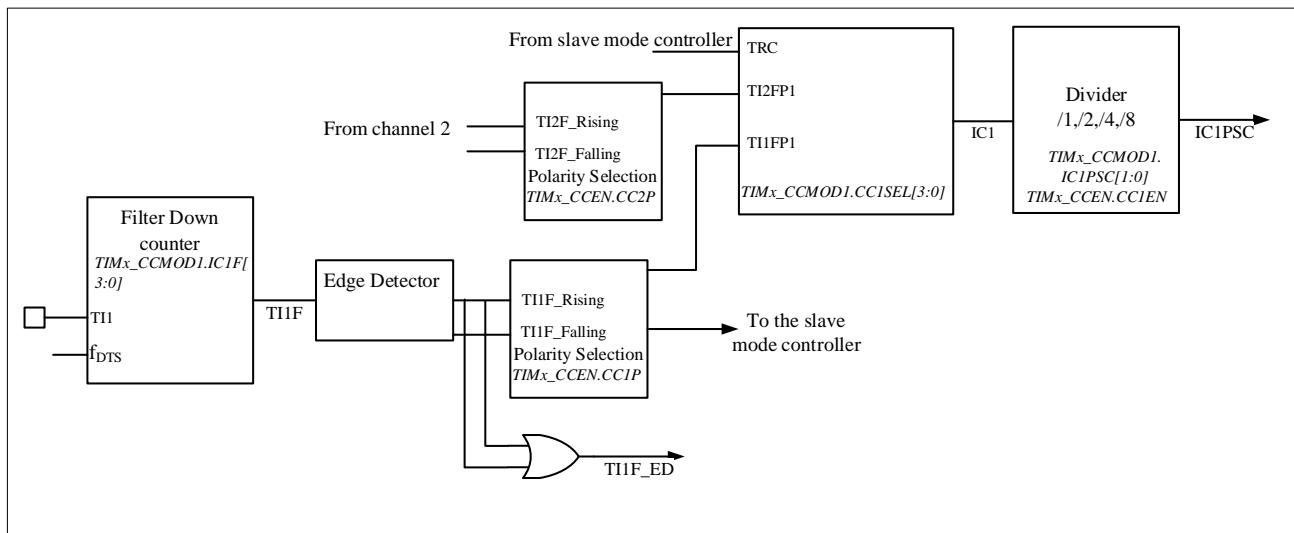


9.3.5 Capture/compare channels

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 TIx is sampled and filtered to generate the signal TIxF. A signal (TIxF_rising or TIxF_falling) is then generated by the edge detector of the polarity select function, the polarity of which is selected by the TIMx_CCEN.CCxF bits. This signal can be used as a trigger input for the slave mode controller. At the same time, the signal ICx is sent to the capture register after frequency division. The following figure shows a block diagram of a capture/compare channel.

Figure 9-21 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 9-22 Capture/compare channel 1 main circuit

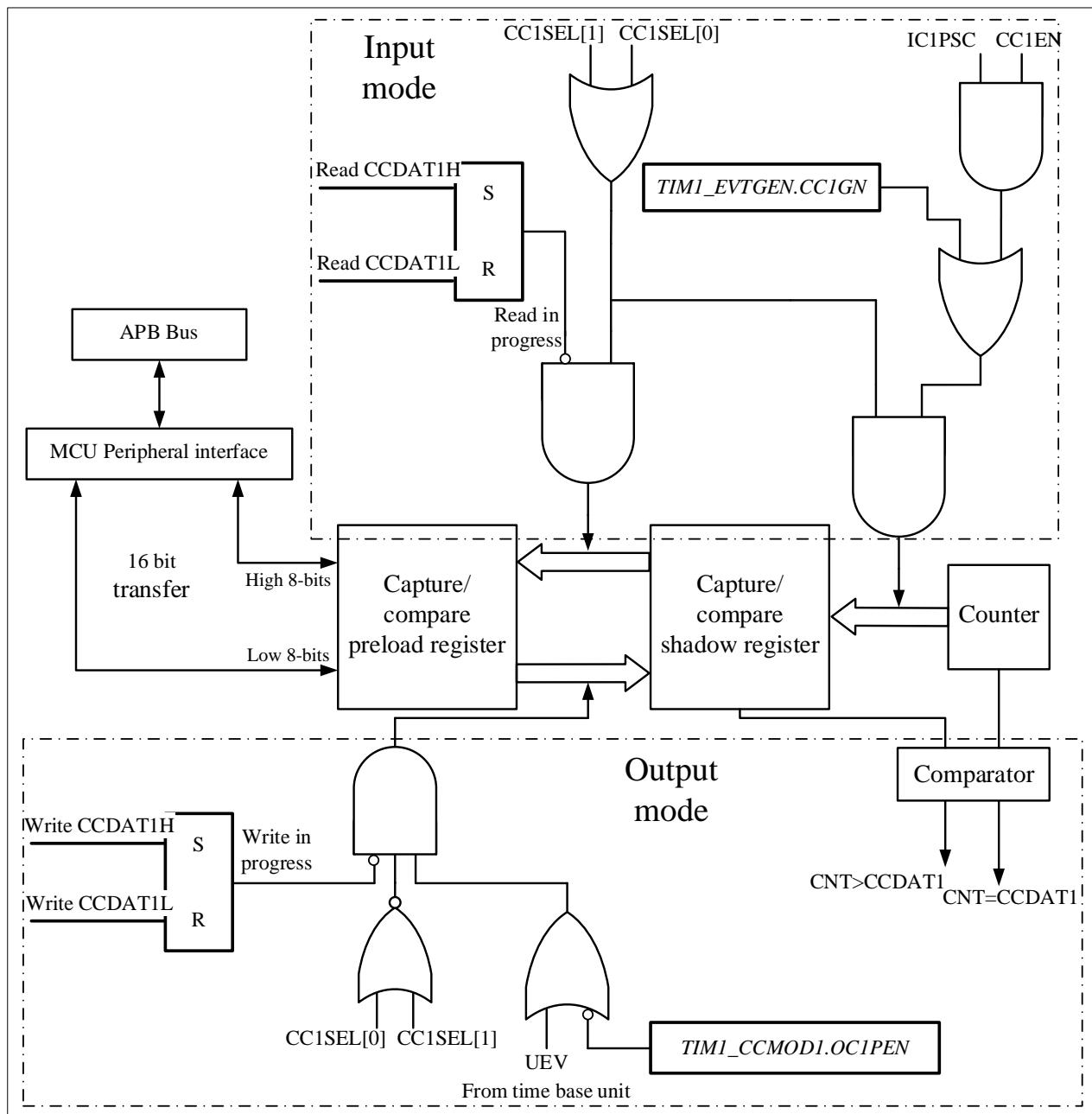


Figure 9-23 Output part of channelx (x=1,2,3,4 for TIM1; x=1,2,3 for TIM8. Take channel 1 as example)

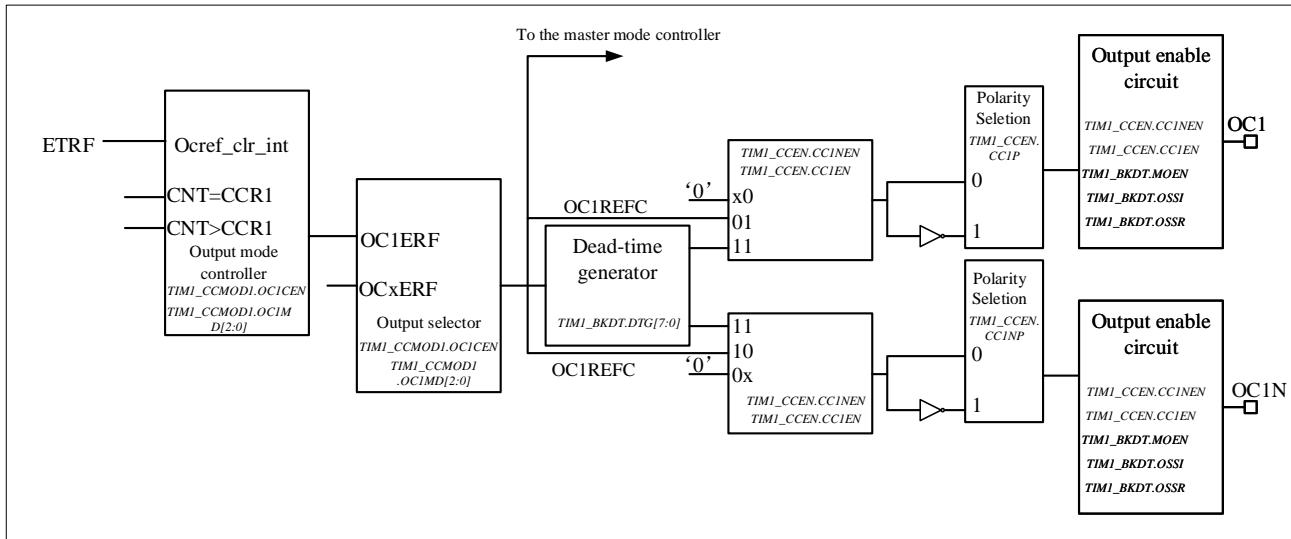
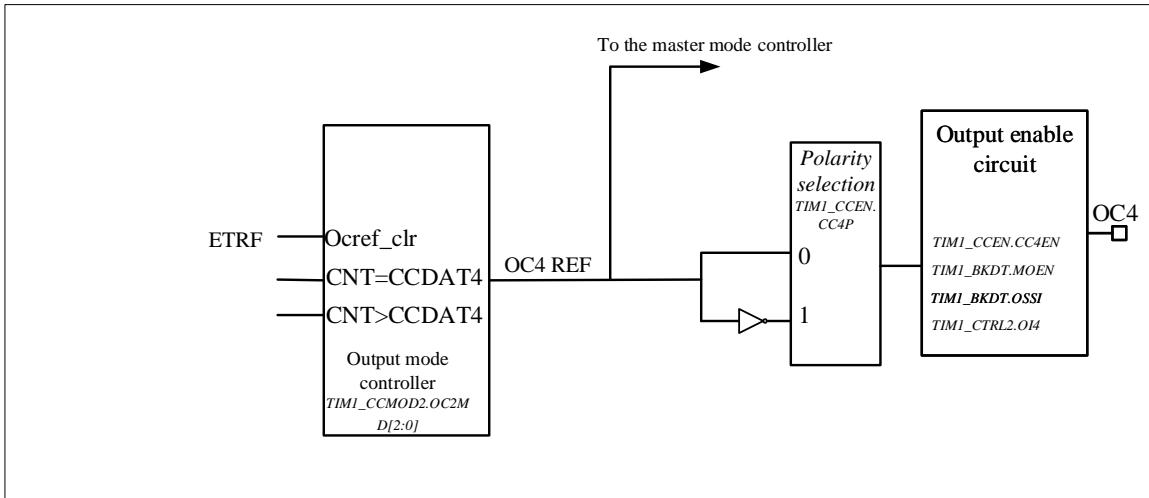


Figure 9-24 Output part of channelx (for TIM8, x=4)



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 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:

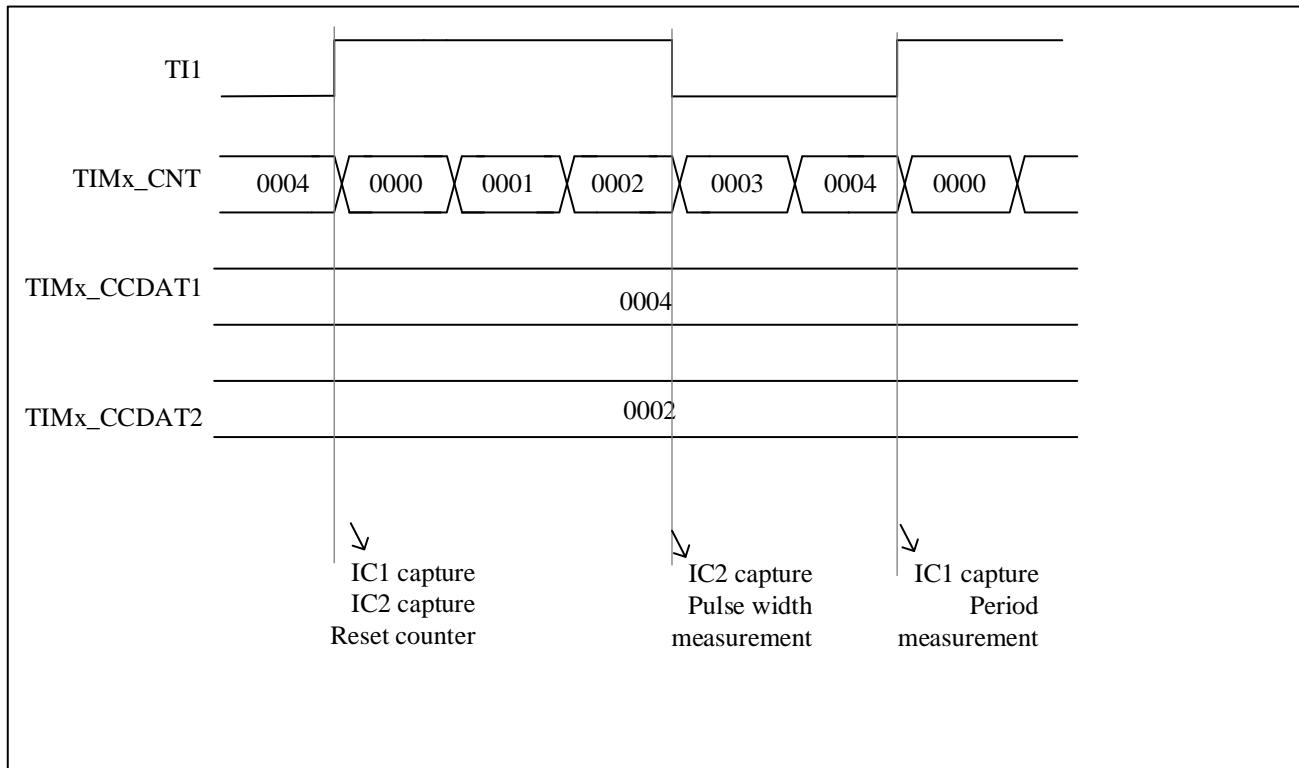
- Two ICx signals are mapped to the same TIx input.
- The two ICx signals are active on edges of opposite polarity.
- Select one of two TIxFP signals as trigger input.
- The slave mode controller is configured in reset mode.

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 9-25 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.CCxITE.
- 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.

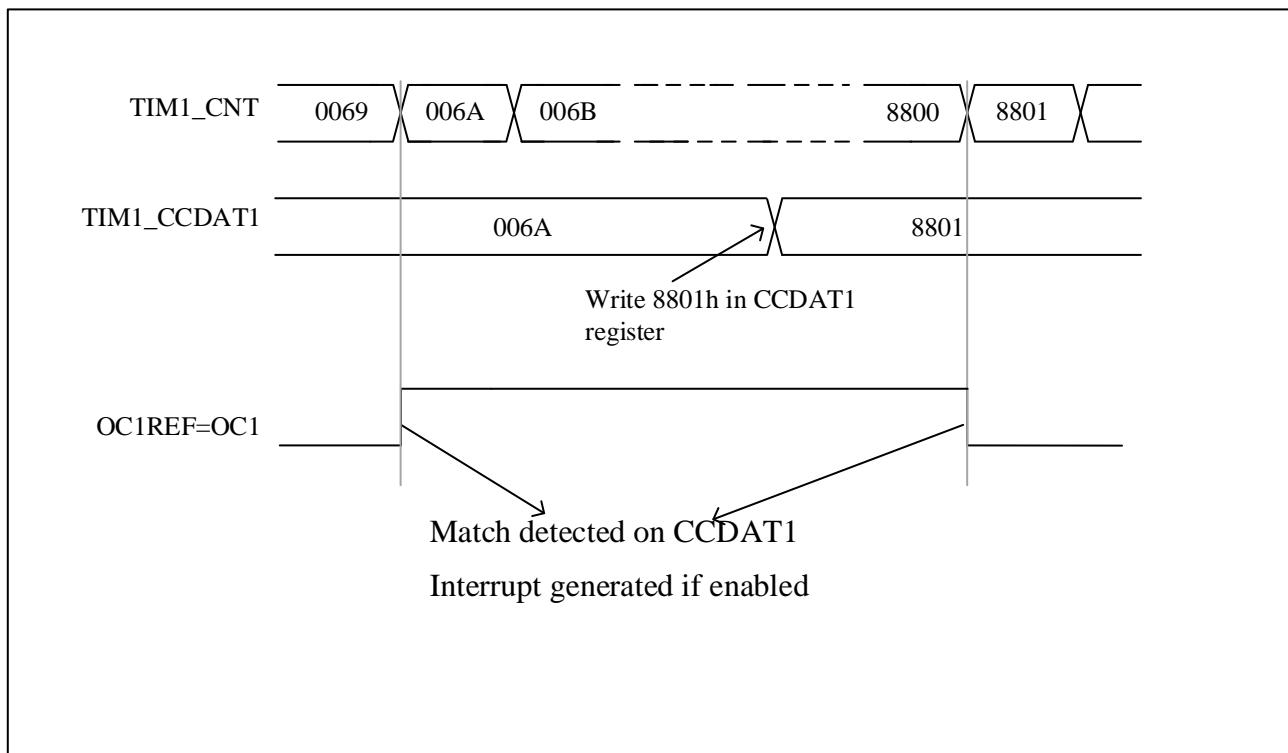
Here are the configuration steps for output compare mode:

- 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 9-26 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_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. 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_CC DATx are always compared with each other when the TIM is under PWM mode.

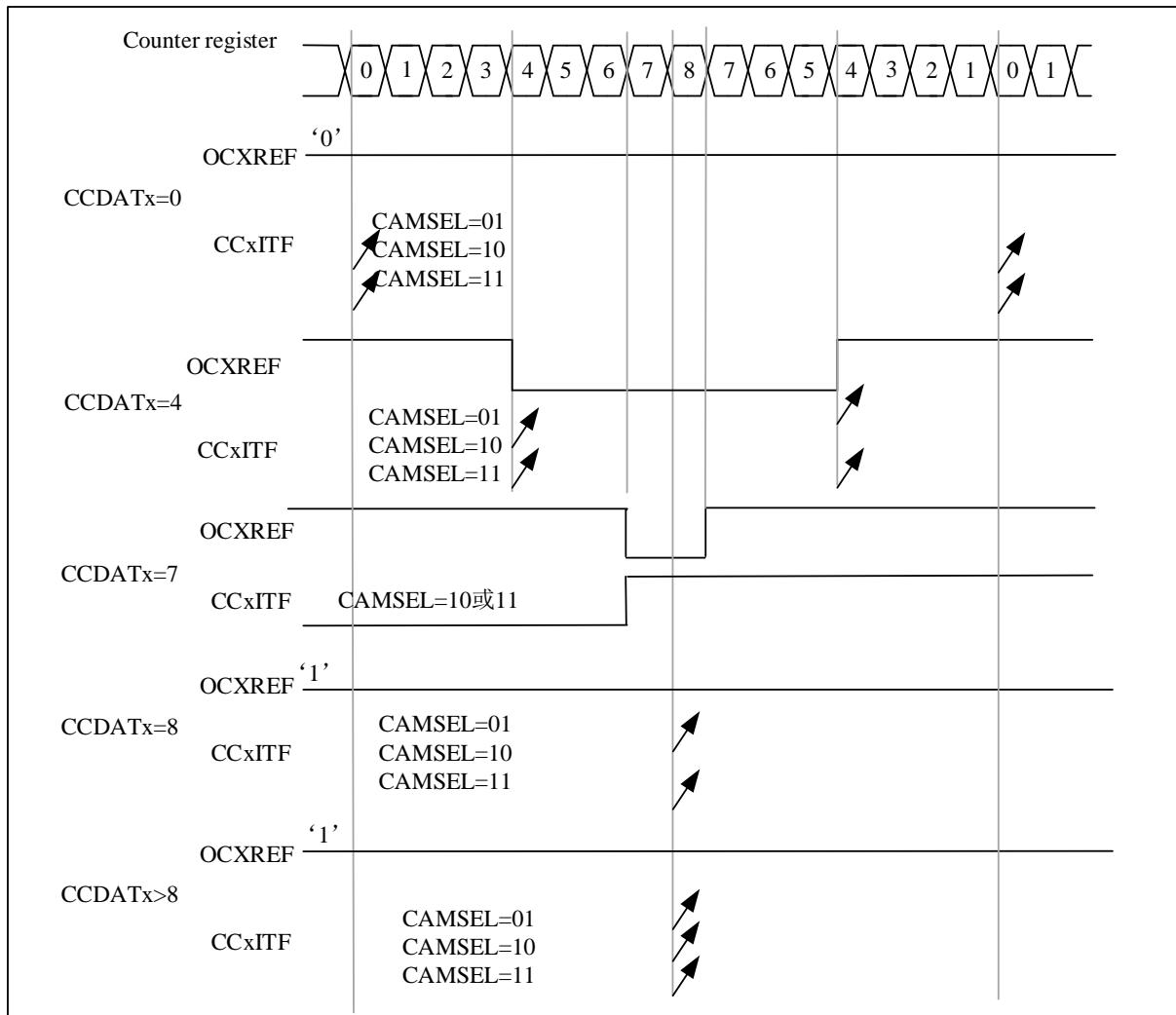
Only if an update event occurs, the preload register will transfer to the shadow register. Therefore user must reset all the registers by setting TIMx_EVTGEN. UDGN before the counter starts counting..

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 9-27 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 center-aligned asymmetric mode

About PWM center-aligned asymmetric mode, see 9.3.2.3.2.

9.3.10.3 PWM edge-aligned mode

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

- **Up-counting**

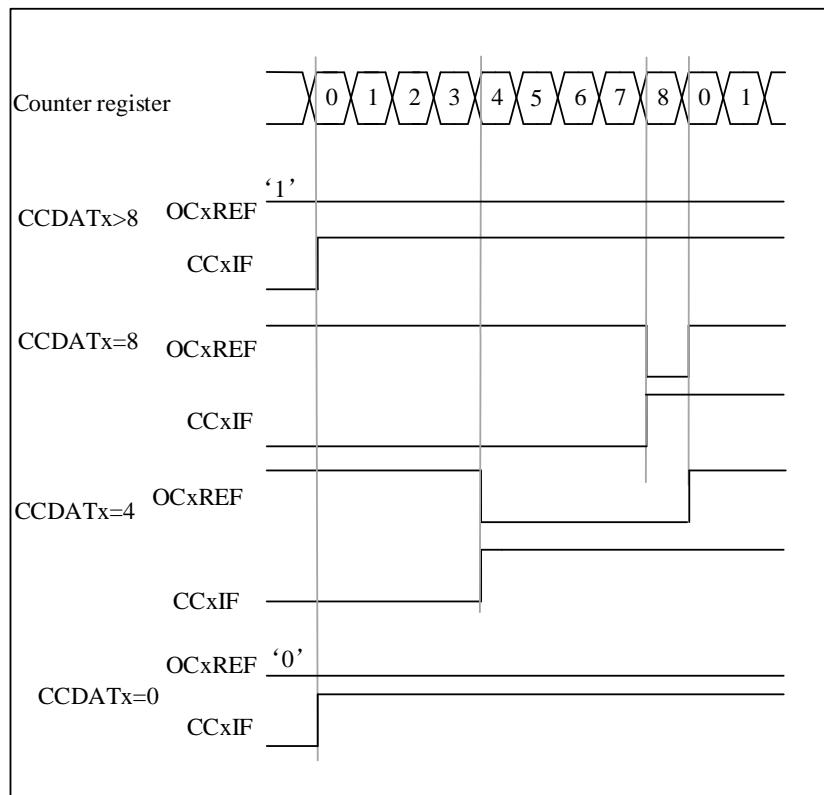
User can set `TIMx_CTRL1.DIR=0` to make counter counts up.

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 9-28 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

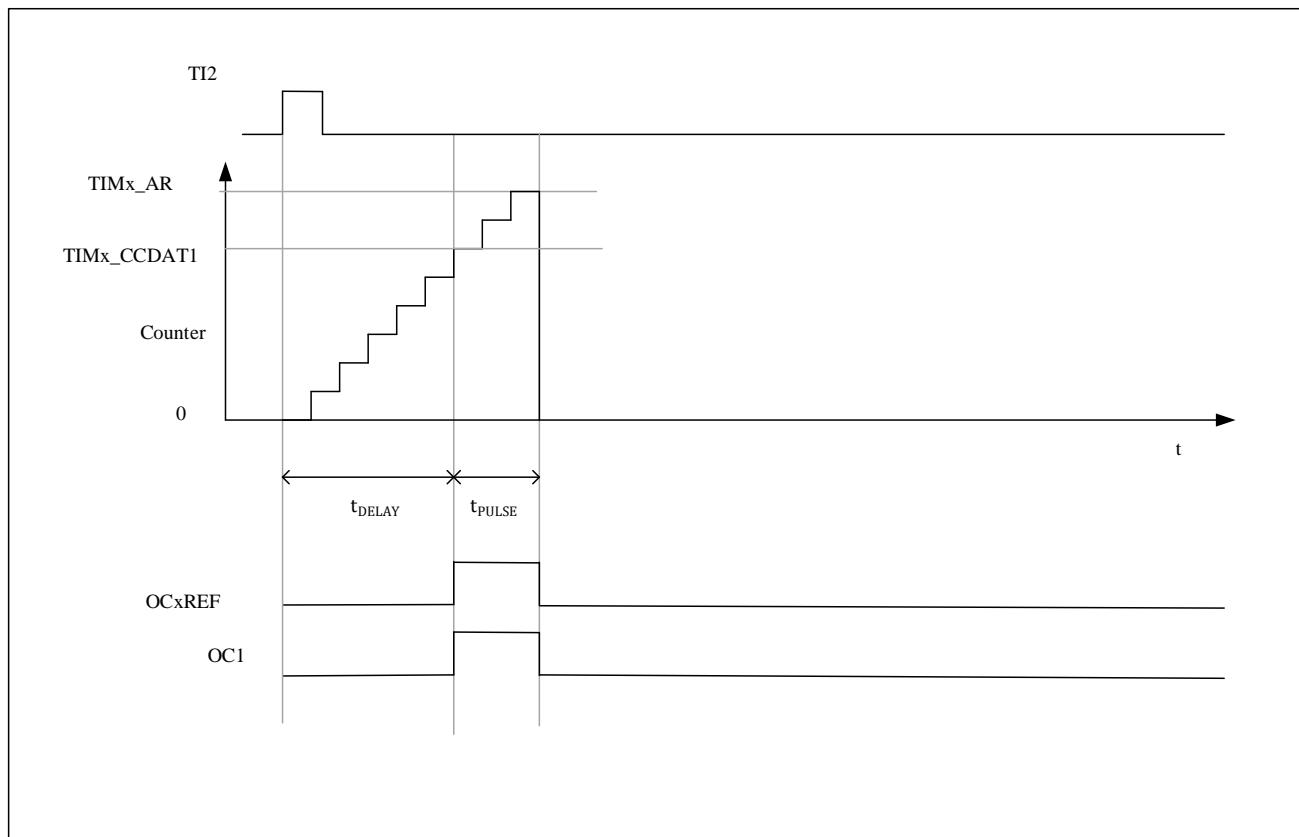
compare value in TIMx_CCDATx is greater than the auto-reload value, the OCxREF will remain 1.

Note: If the nth PWM cycle CCDATx shadow register $\geq \text{AR}$ 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 $\text{OCxREF} = '0'$, 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 9-29 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 $\text{TIMx_CNT} < \text{TIMx_CCDAT1} \leq \text{TIMx_AR}$;
2. TI2FP2 is mapped to TI2, $\text{TIMx_CCMOD1.CC2SEL} = '01'$; TI2FP2 is configured for rising edge detection, $\text{TIMx_CCEN.CC2P} = '0'$;
3. TI2FP2 acts as the trigger (TRGI) of the slave mode controller and starts the counter, $\text{TIMx_SMCTRL.TSEL} =$

‘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

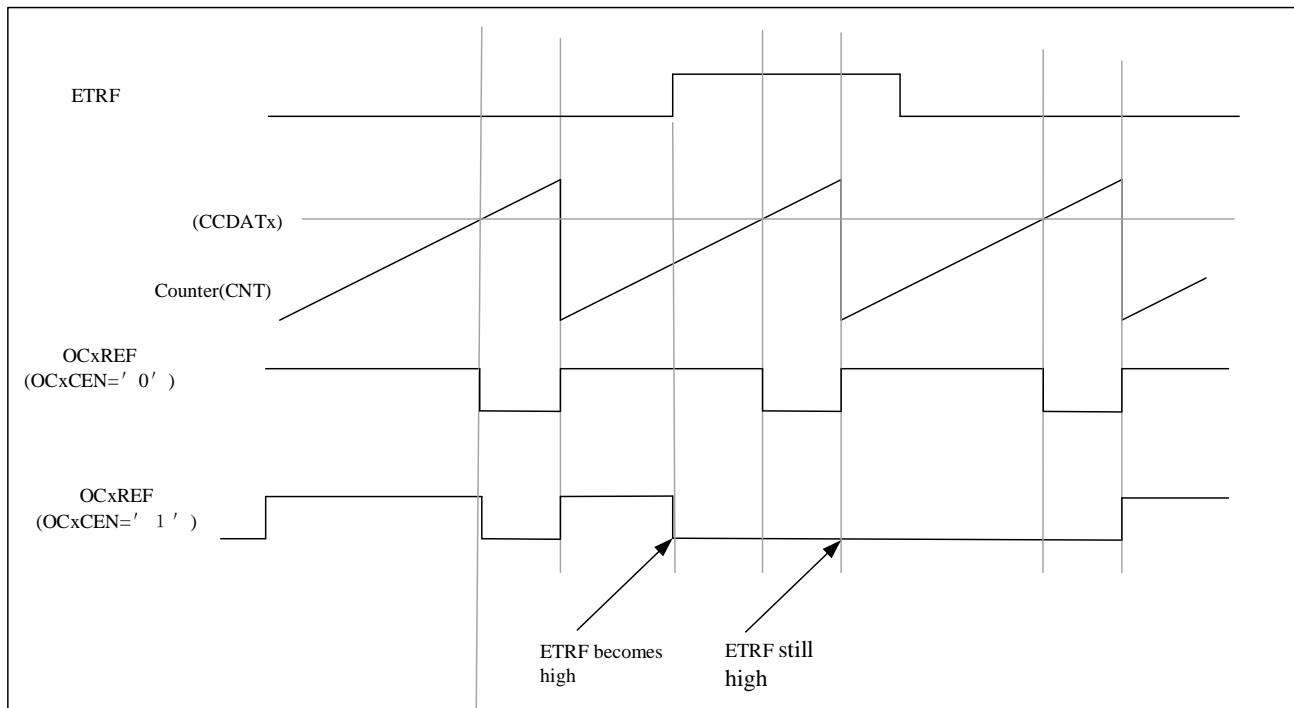
If user set TIMx_CCMODx.OCxCEN=1, high level of ETRF input can be used to driven the OCxREF signal to low, and the OCxREF signal will remains low, until the next UEV happens. Only output compare and PWM modes can use this function. This cannot be used when it is in forced mode.

Here is an example for it. To control the current, user can connect the ETR signal to external IO, 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 9-30 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.

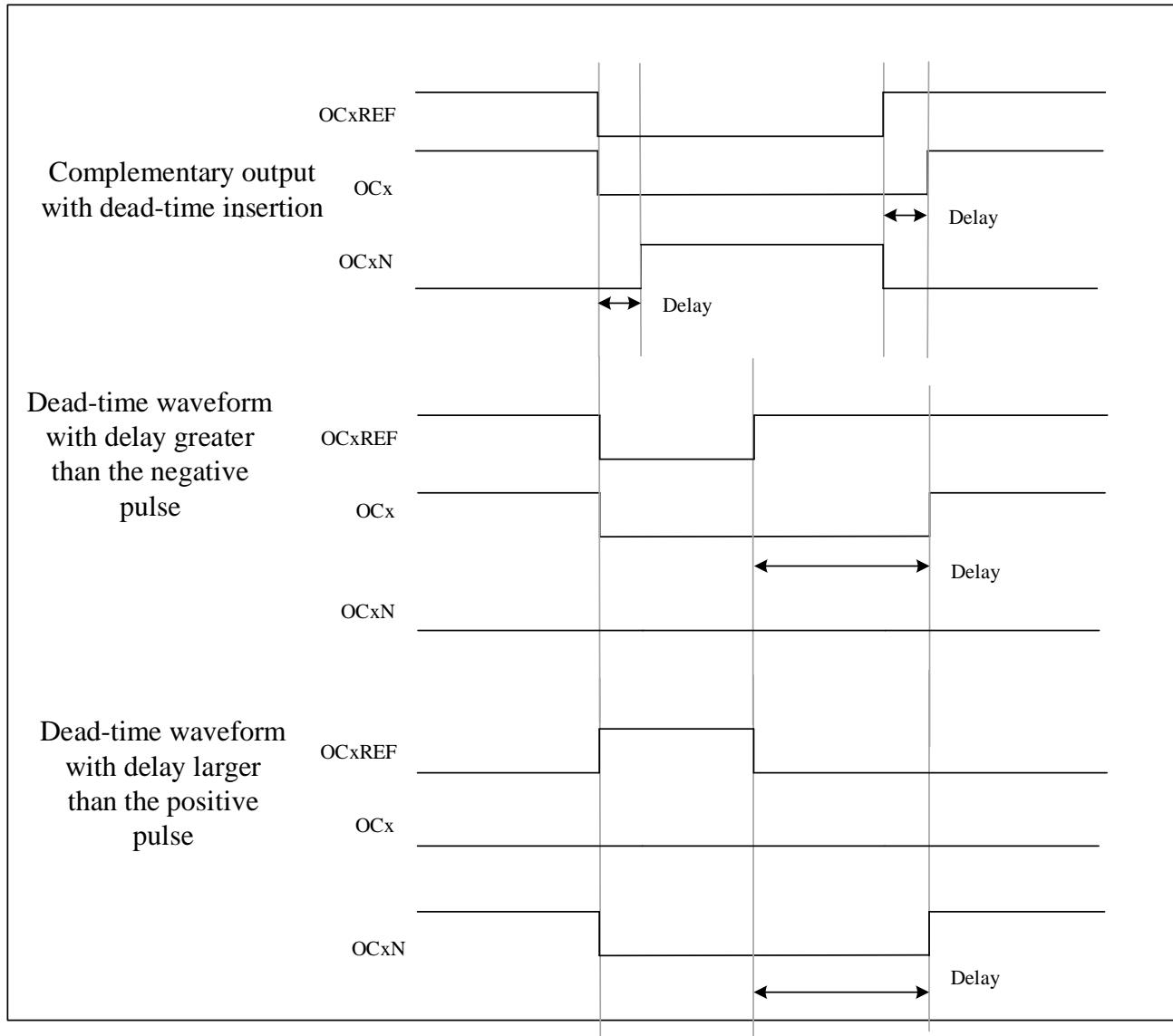
Reference waveform OCxREF can generates 2 outputs OCx and OCxN. And if OCx and OCxN are active high, the OCx ouput signal is the same as the reference signal and the OCxN output signal is the opposite of the reference signal. However, OCx output signal will be delayed relative to the reference rising 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 TIMx_CCEN.CCxP=0, TIMx_CCEN.CCxNP=0, TIMx_BKDT.MOEN=1, TIMx_CCEN.CCxEN=1,

TIMx_CCEN.CCxNEN=1.

Figure 9-31 Complementary output with dead-time insertion



User can set 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 TIMx_CCEN.CCxEN and 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 TIMx_CCEN.CCxEN=0 and TIMx_CCEN.CCxNEN=1, it will not complemented, and OCxN will become active when OCxREF is high. On the other hand, if user set 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, $(CCxP \wedge OIx) \wedge (CCxNP \wedge OIxN) = 0$.

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.
- 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 is 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 $(CCxP \wedge OIx) \wedge (CCxNP \wedge OIxN) = 0$, 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 APB clock cycles).

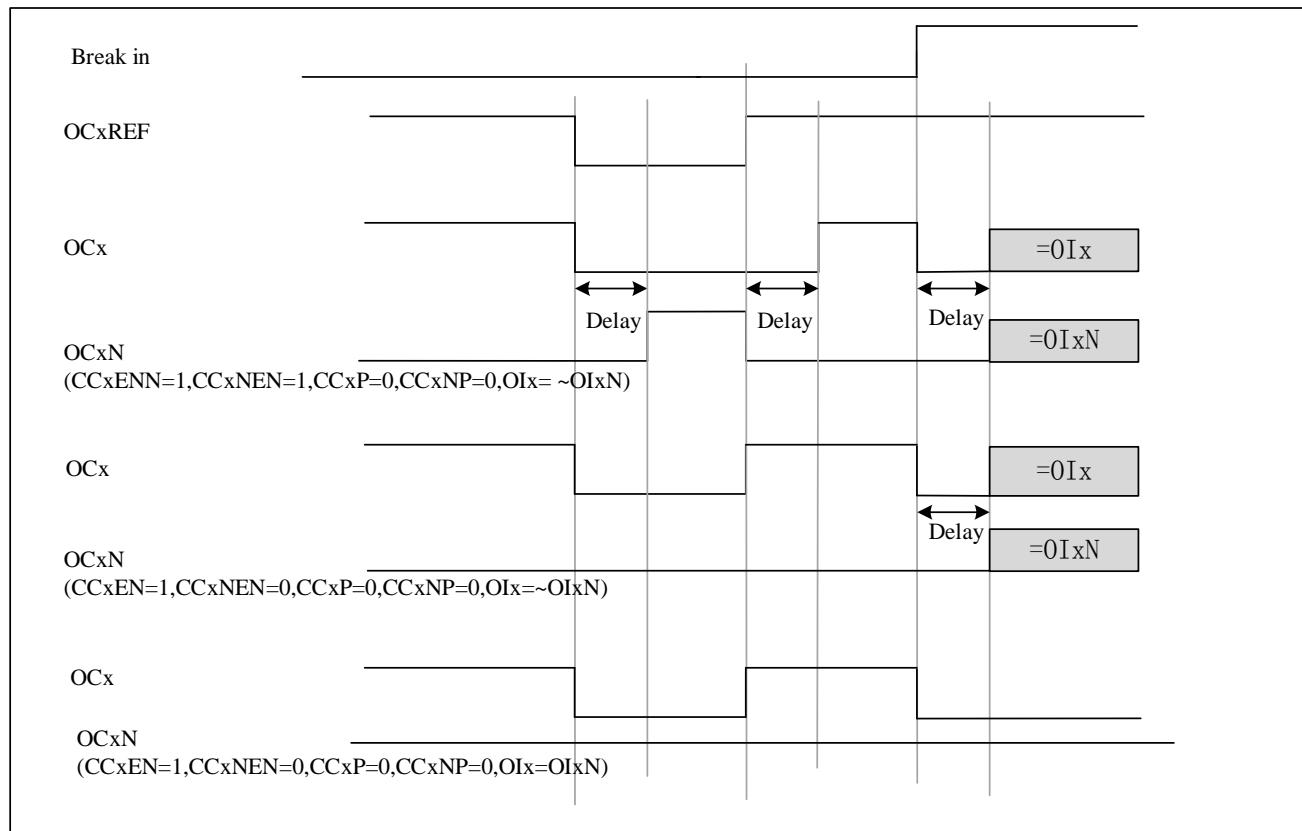
cycles of ck_{tim}).

- ◆ 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

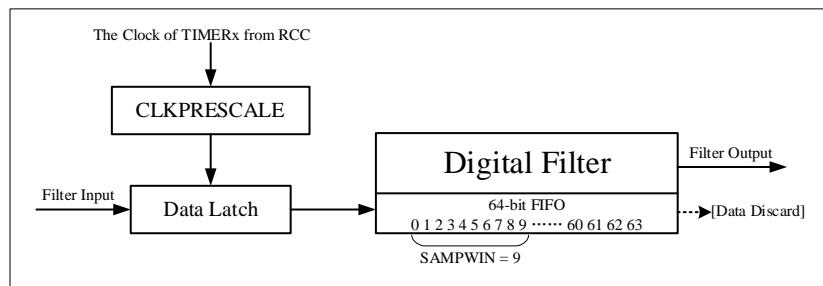
Figure 9-32 Output behavior in response to a break



9.3.14.1 Break filter

Register TIM1_BRKFILT description are as follow:

Figure 9-33 Slide filter



- The digital filter samples break signal at the clock of TIM1 from RCC, accumulating samples in a 64-bits FIFO. Only sampled data within window size defined in TIM1_BRKFILT.WSIZE [5:0] with maximum size 64.
- The filter outputs the majority value inside sample window which is defined by the threshold value in TIM1_BRKFILT.THRESH [5:0] with maximum threshold of 63. This value should be equal or more than half of window size. If neither logic 1 nor logic 0 counts inside sampling window is more than threshold, digital filter maintain previous output value.
- TIM1_BRKFILT.PSC register determines sample rate of corresponding digital filter. Filter FIFO capture one sample value from input at every sample clock.
- If digital filter is off, filter input will bypass to output like a wire.

9.3.15 Debug mode

When the microcontroller is in debug mode (the Cortex-M4 core halted), depending on the DBG_TIMx_STOP configuration in the DBG module, the TIMx counter can either continue to work normally or stop. For more details, see 21.4.3.

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_CC DATx, and generates the update event UEV (TIMx_CTRL1.UPRS=0).

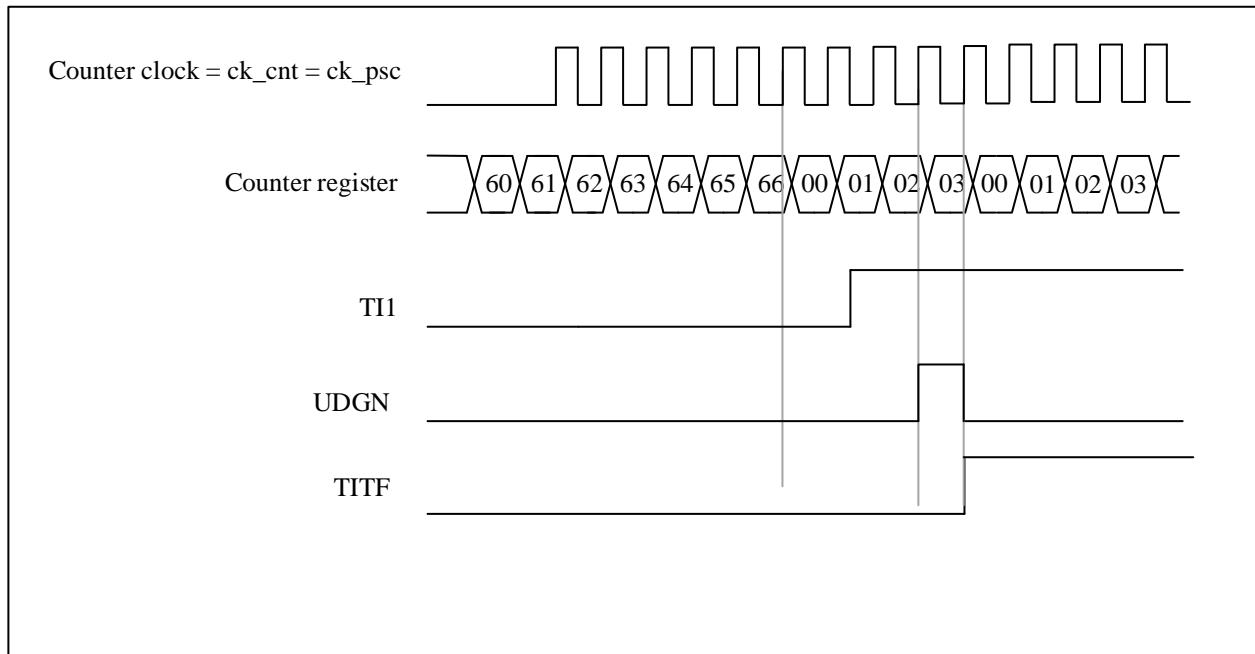
The following is an example of a reset mode:

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. Setting TIMx_CTRL1.CNTEN = 1 to start counter;

After starting the timer, when TI1 detects a rising edge, the counter resets and restarts counting, and the trigger flag is set (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 9-34 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.

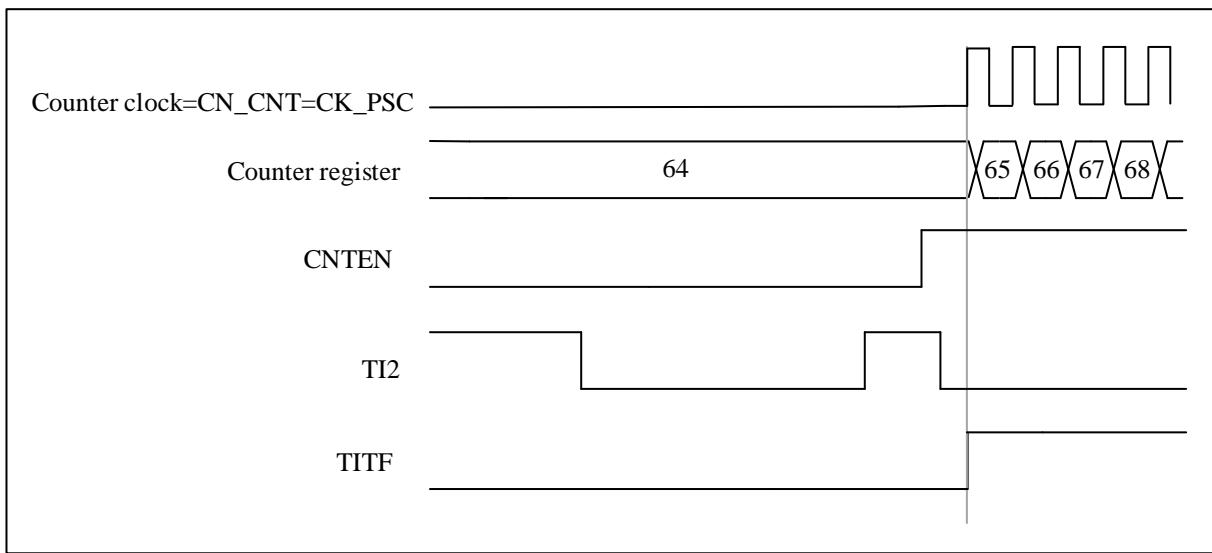
The following is an example of a trigger pattern:

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

When TI2 detects a rising edge, the counter starts counting, and the trigger flag is set (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 9-35 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.

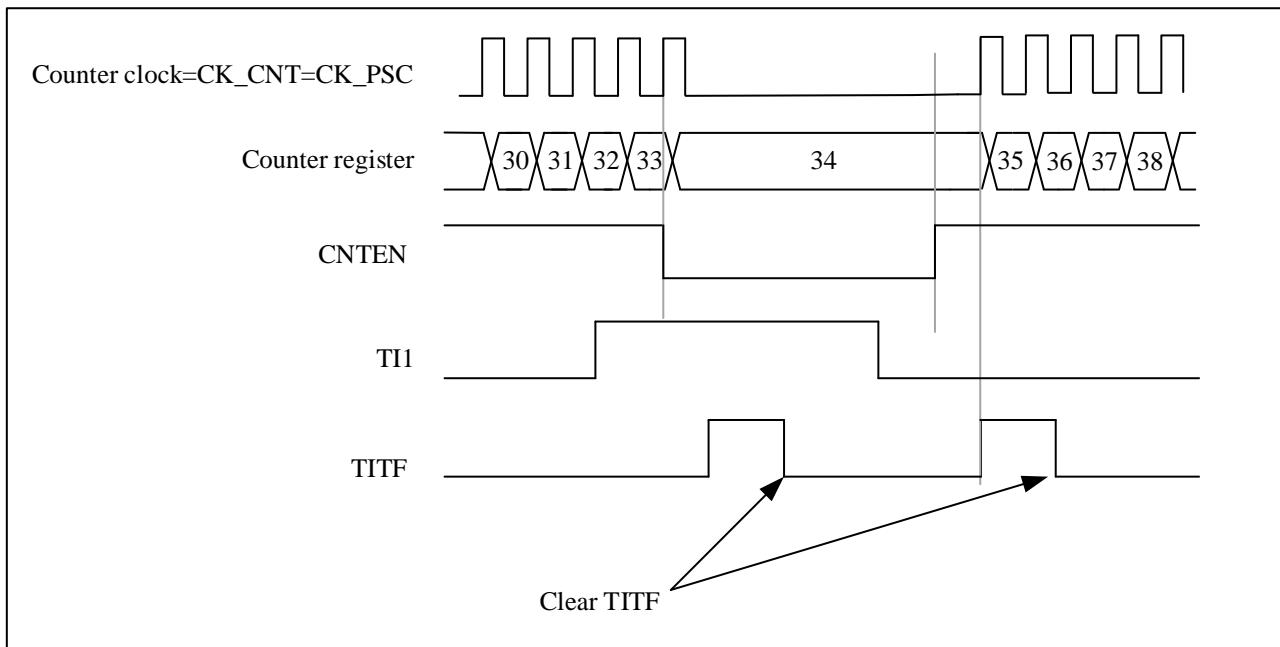
The following is an example of a gated pattern:

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. Setting TIMx_CTRL1.CNTEN = 1 to start counter

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 9-36 Control circuit in Gated mode



9.3.16.4 Slave mode: Trigger Mode + External Clock Mode 2

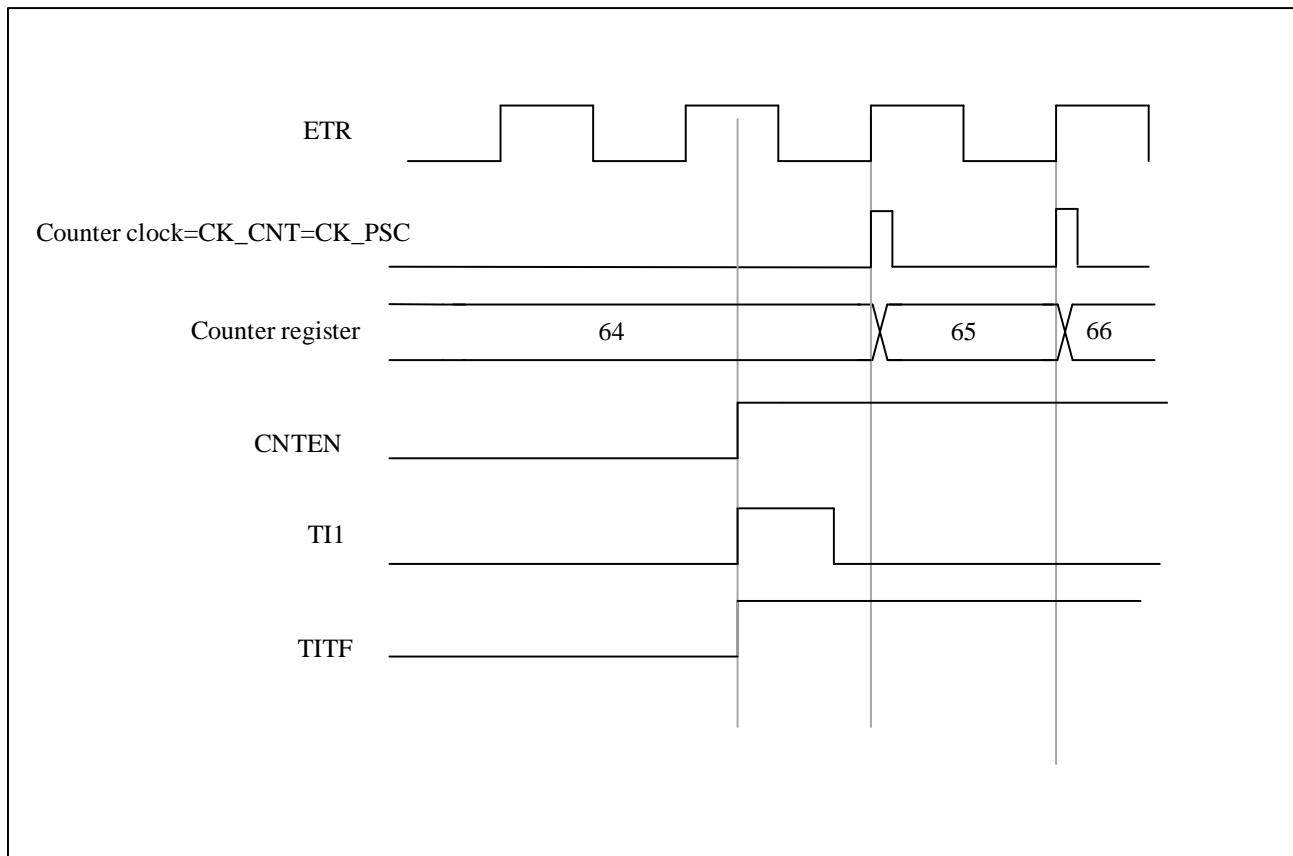
In reset mode, trigger mode and gate control mode, the counter clock can be selected as external clock mode 2, and the ETR signal is used as the external clock source input. At this time, the trigger selection needs to select non-ETRF (TIMx_SMCTRL.TSEL=111).

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 9-37 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.

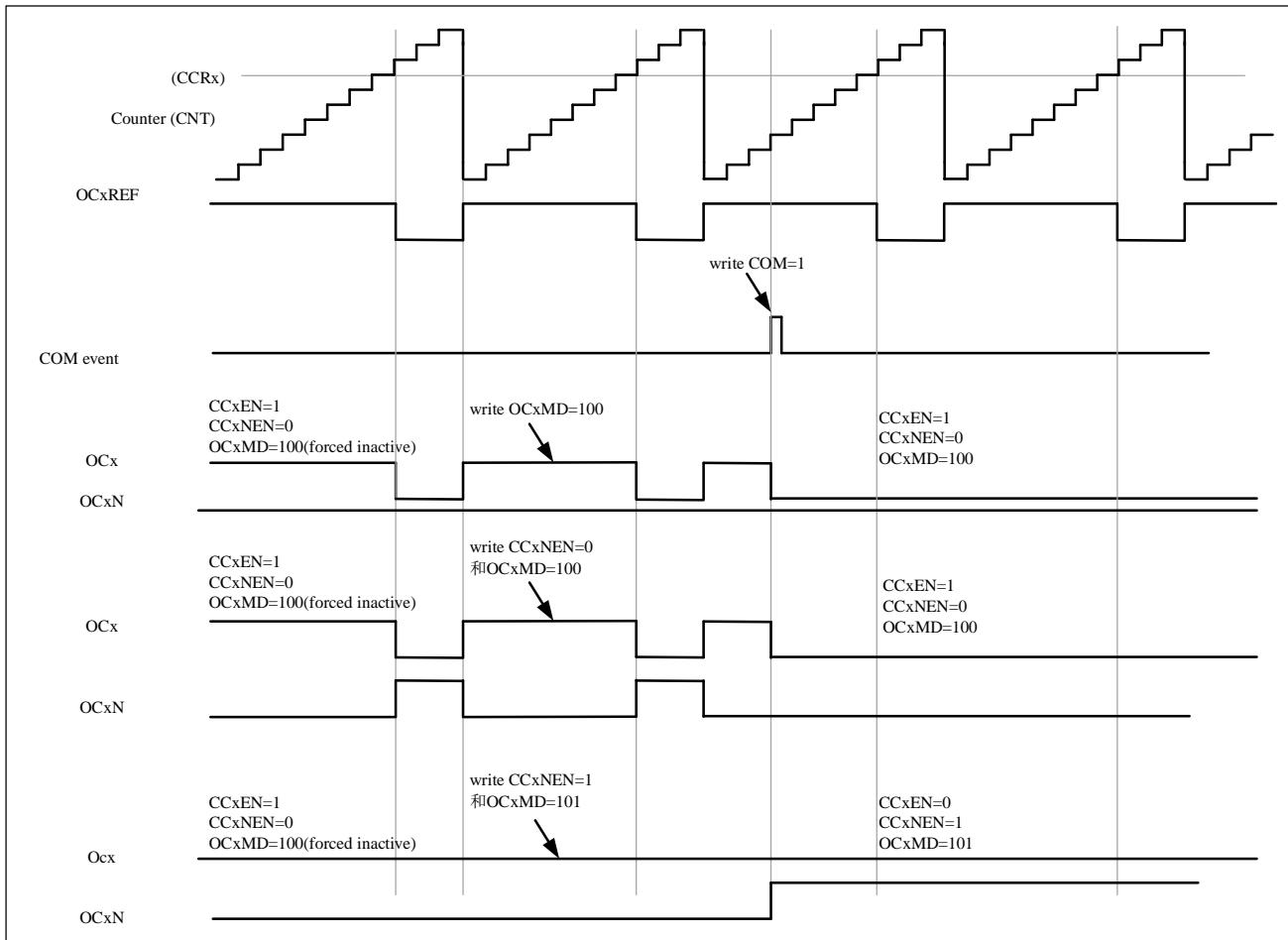
COM commutation event generation method:

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 9-38 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.

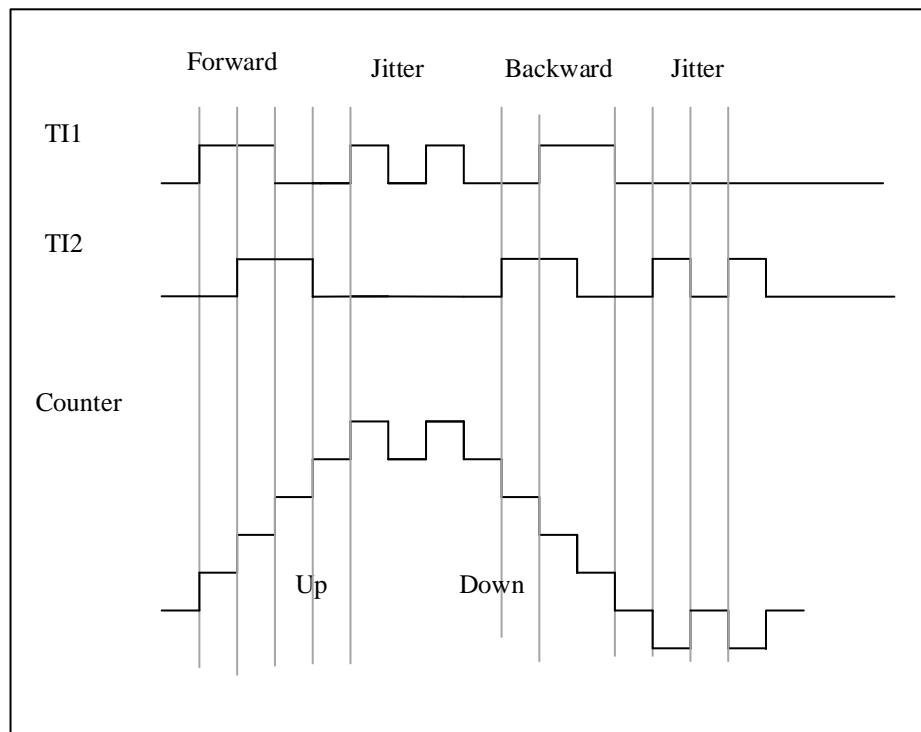
The relationship between the counting direction and the encoder signal is shown in **Table 9-1 Counting direction versus encoder signals:**

Table 9-1 Counting direction versus encoder signals

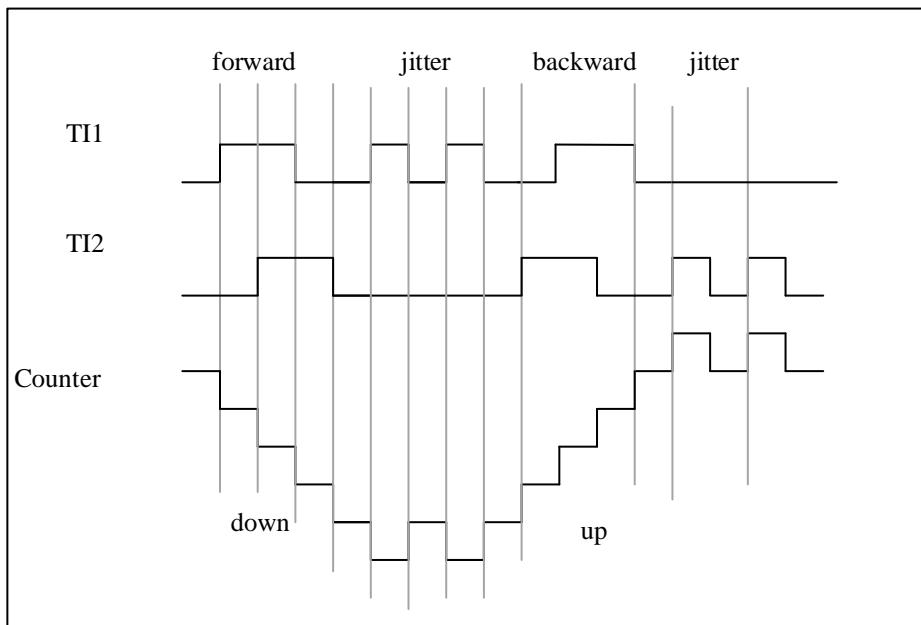
Active edge	Level on opposite signals (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI2FP2 signal	
		Rising	Falling	Rising	Falling
Counting only at TI1	High	Counting down	Counting up	Don't count	Don't count
	Low	Counting up	Counting down	Don't count	Don't count
Counting only at TI2	High	Don't count	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 9-39 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 9-40 Encoder interface mode example with IC1FP1 polarity inverted

9.3.20 Interfacing with Hall sensor

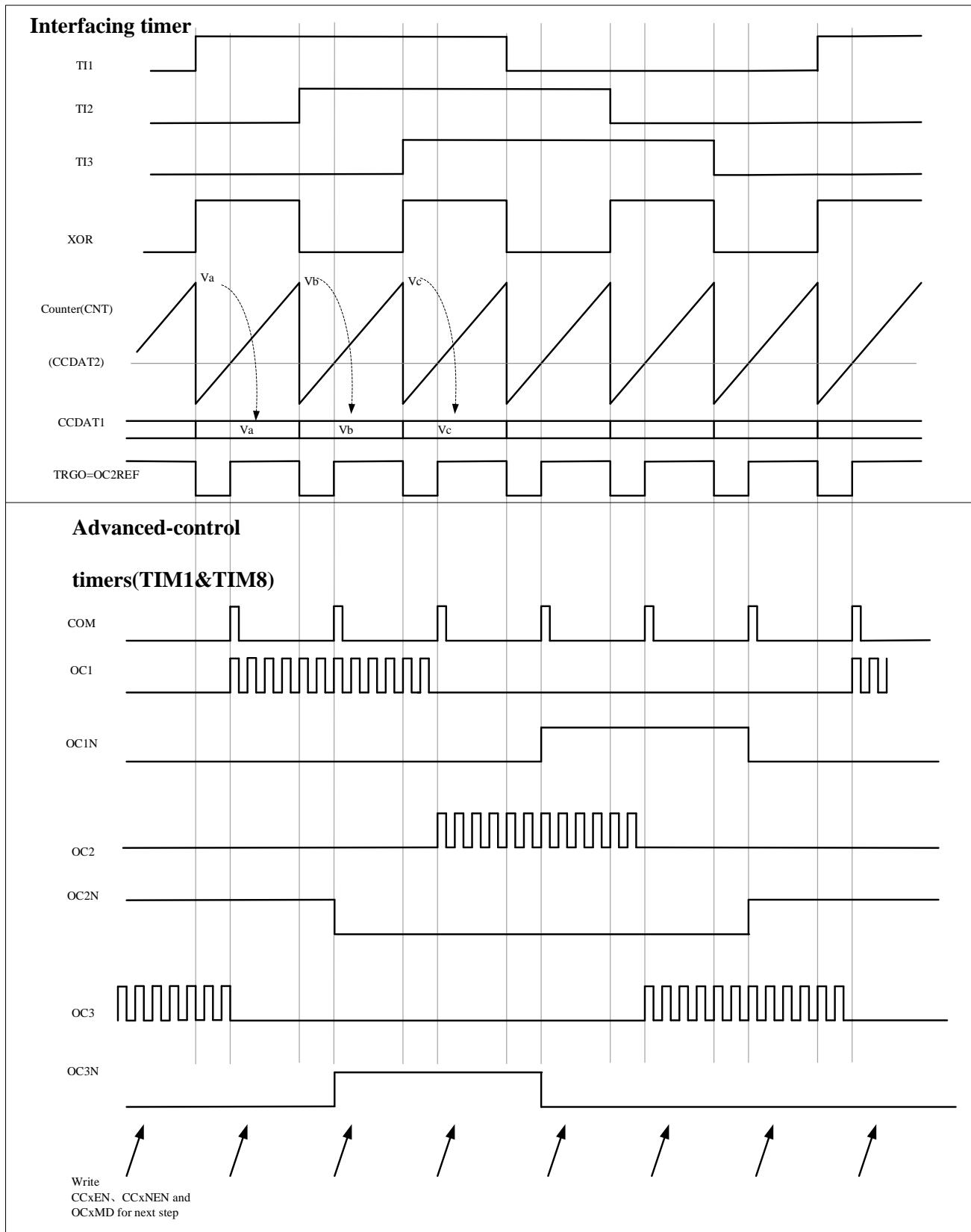
Connect the Hall sensor to the three input pins (CC1, CC2 and CC3) of the timer, and then select the XOR function to pass the inputs of TIMx_CH1, TIMx_CH2 and TIMx_CH3 through the XOR gate as the output of TI1 to channel 1 for capture signal.

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 9-41 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).

When configuring the registers, please refer to the block diagram of the corresponding TIM, the channels and functions of different TIM will be different.

9.4.1 Register overview

Table 9-2 Register overview

Offset	Register	31
0x00	TIMx_CTRL1	30
0x04	Reset value	29
0x08	TIMx_CTRL2	28
0x0C	Reset value	27
0x10	TIMx_SMCTRL	26
0x14	Reset value	25
0x18	TIMx_EVTGEN	24
0x1C	Reset value	23
0x20	TIMx_STS	22
0x24	Reset value	21
0x28	TIMx_DINTEN	20
0x2C	Reset value	19
0x30	Reserved	18
0x34	Reserved	17
0x38	Reserved	16
0x3C	Reserved	15
0x40	TDEN	14
0x44	EXCEN	13
0x48	COMDEN	12
0x4C	EXTPS[1:0]	11
0x50	CC4DEN	10
0x54	CC3DEN	9
0x58	CC2DEN	8
0x5C	CC1DEN	7
0x60	UDEN	6
0x64	MSMD	5
0x68	TI1SEL	4
0x6C	ARPEN	3
0x70	Reserved	2
0x74	MMSEL[2:0]	1
0x78	TISEL[2:0]	0
0x7C	CC2HEN	0
0x80	CC1HEN	0
0x84	UIEN	0
0x88	Reserved	0
0x8C	CCUSEL	0
0x90	Reserved	0
0x94	CCPCTL	0
0x98	CNTEN	0
0xA0	UPRS	2
0xA4	UPDIS	1
0xA8	CNTEN	0

		Reset value																
0x18		TIMx_CCMOD1 Input capture mode																
0x1C		TIMx_CCMOD2 Output comparison mode																
0x1C		TIMx_CCMOD2 Input capture mode																
0x20		TIMx_CCEN																
0x24		TIMx_CNT																
0x28		TIMx_PSC																
0x2C		TIMx_AR																
0x30		TIMx_REPCNT																
0x34		TIMx_CCDDAT1																
		Reset value	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	CCDDAT1[15:0]	Reserved													

0x38	TIMx_CCDAT2	CCDDAT2[15:0]															
	Reset value	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0															
0x3C	TIMx_CCDAT3	CCDDAT3[15:0]															
	Reset value	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0															
0x40	TIMx_CCDAT4	CCDDAT4[15:0]															
	Reset value	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0															
0x44	TIMx_BKDT	Reserved															
	Reset value	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0															
0x48	TIMx_DCTRL	Reserved															
	Reset value	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0															
0x4C	TIMx_DADDR	Reserved															
	Reset value	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0															
0x54	TIMx_CCMOD3	Reserved															
	Reset value	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0															
0x58	TIMx_CCDAT5	Reserved															
	Reset value	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0															
0x5C	TIMx_CCDAT6	Reserved															
	Reset value	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0															

0x60	TIM1_CC DAT7	Reserved												CC DAT7[15:0]												
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0
0x64	TIM1_CC DAT8	Reserved												CC DAT8[15:0]												
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0
0x68	TIM1_CC DAT9	Reserved												CC DAT9[15:0]												
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0
0x6c	TIMx_BRKFILT	Reserved	THRESH[5:0]				Reserved	WSIZE[5:0]				FILTEN	PSC[15:0]													
	Reset value	0	0	0	0	0	0	0	0	0	0	0	b	0	0	0	0	0	0	0	0	0	0	0	0	

9.4.2 Control register 1 (TIMx_CTRL1)

Offset address: 0x00

Reset value: 0x0000 0000

31	Reserved												22	21	20	19	18	17	16
15	14	12	11	10	9	8	7	6	5	4	3	2	rw	rw	rw	rw	ASYMME	PBKPN	LBKPN
CLRSEL	Reserved		CISEL	IOM BKPN	CLKD[1:0]	ARPEN	CAMSEL[1:0]	DIR	ONEPM	UPRS	UPDIS	CNTEN	rw	rw	rw	rw	TRI		
rw		rw	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
21:20	CMODE	<p>In center-aligned asymmetric mode, channel 4/7/8/9 trigger mode, only when TIMx_CTRL2.MMSEL3 = 1, the TRGO output will be valid.</p> <p>00: Up-counting to CC DAT4/7/8/9, trigger valid</p> <p>01: Down-counting, channel 4 count to CC DDAT4, channel 7/8/9 count to CC DAT7/8/9, trigger valid</p> <p>1x: Channel 4 up-counting to CC DAT4 or down-counting to CC DDAT4, channel 7/8/9 up-counting or down-counting to CC DAT7/8/9, trigger valid</p> <p>In center-aligned symmetry mode, channel 4/7/8/9 trigger mode, only when TIMx_CTRL2.MMSEL3 = 1, the TRGO output will be valid.</p> <p>00: Up-counting to CC DAT4/7/8/9, trigger valid</p>

Bit field	Name	Description
		01: Down-counting to CC DAT4/7/8/9, trigger valid 1x: Up-counting or down-counting to CC DAT4/7/8/9, trigger valid <i>NOTE: This function is only for TIM1</i>
19	Reserved	Reserved, the reset value must be maintained
18	ASYMMETRIC	Asymmetric mode enable in center-aligned 0: Disabled 1: Enabled (valid when TIMx_CTRL1.CAMSEL[1:0] is non-zero, each channel will compare to CC DATx when counting up, and compare to CC D DATx when counting down) <i>NOTE: This function is only for TIM1</i>
17	PBK PEN	PVD as BKP enable 0: Disable 1: Enable
16	LBK PEN	LockUp as BKP enable 0: Disable 1: Enable
15	CLRSEL	OCxREF clear selection 0: Select the external OCxREF clear from ETR 1: Reserved
14:12	Reserved	Reserved, the reset value must be maintained
11	C1SEL	Channel 1 selection 0: Select external CH1 signal from IOM 1: Reserved
10	IOMBK PEN	Enabling IOM as BKP 0: Enable. Select external break (from IOM) signal 1: Reserved
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.

Bit field	Name	Description
		<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"> – Counter overflow/underflow – The TIMx_EVTGEN.UDGN bit is set – Update generation from the slave mode controller <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"> – Counter overflow/underflow – The TIMx_EVTGEN.UDGN bit is set – Update generation from the slave mode controller <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> <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								25	24	23	22	21	20	19	18	17	16
15	TRIG4	OI4	OI3N	OI3	OI2N	OI2	OI1N	OI1	rw	8	rw	7	rw	6	rw	4	rw	3
14	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	0

Bit field	Name	Description
31:25	Reserved	Reserved, the reset value must be maintained
24	MMSEL3	Bit3 of the master mode selection. <i>NOTE: This function is only for TIM1</i>
23	TRIG9	OC9REF trig ADC enable 0: Trigger disable 1: Trigger enable <i>NOTE: This function is only for TIM1</i>
22	Reserved	Reserved, the reset value must be maintained
21	TRIG8	OC8REF trig ADC enable 0: Trigger disable 1: Trigger enable <i>NOTE: This function is only for TIM1</i>
20	Reserved	Reserved, the reset value must be maintained
19	TRIG7	OC7REF trig ADC enable 0: Trigger disable 1: Trigger enable <i>NOTE: This function is only for TIM1</i>
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	TRIG4	OC4REF trig ADC enable 0: Trigger disable 1: Trigger enable <i>NOTE: This function is only for TIM1</i>
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. <i>NOTE: This function is only for TIM1</i>
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

Bit field	Name	Description
8	OI1	<p>Output Idle state 1</p> <p>0: When TIMx_BKDT.MOEN = 0, if OC1N is implemented, after dead-time OC1 = 0</p> <p>1: When TIMx_BKDT.MOEN = 0, if OC1N is implemented, after dead-time OC1 = 1</p>
7	TI1SEL	<p>TI1 selection</p> <p>0: TIMx_CH1 pin connected to TI1 input.</p> <p>1: TIMx_CH1, TIMx_CH2, and TIMx_CH3 pins are XOR connected to the TI1 input.</p>
6:4	MMSEL[2:0]	<p>Master Mode Selection</p> <p>These 4 bits (TIMx_CTRL2.MMSEL3 and 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:</p> <p>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.</p> <p>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.</p> <p>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).</p> <p>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.</p> <p>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.</p> <p>100: Compare - OC1REF signal is used as the trigger output (TRGO).</p> <p>101: Compare - OC2REF signal is used as the trigger output (TRGO).</p> <p>110: Compare - OC3REF signal is used as the trigger output (TRGO).</p> <p>111: Compare - OC4REF signal is used as the trigger output (TRGO).</p> <p>1xxx: Compare-If the counter is center-aligned mode: The corresponding edge signal of OC4REF/OC7REF/OC8REF/OC9REF phase or hind used as the trigger output (TRGO), up counting and down counting are configurable, refer specifically to the TIMx_CTRL1.CMODE. If the counter is edge alignment mode: The OC4REF signal is used as the trigger output (TRGO).</p>
3	CCDSEL	<p>Capture/compare DMA selection</p> <p>0: When a CCx event occurs, a DMA request for CCx is sent.</p> <p>1: When an update event occurs, a DMA request for CCx is sent.</p>
2	CCUSEL	<p>Capture/compare control update selection</p> <p>0: If TIMx_CTRL2.CCPCTL = 1, they can only be updated by setting CCUDGN bit</p> <p>1: If TIMx_CTRL2.CCPCTL = 1, they can be updated by setting CCUDGN bit or a rising edge on TRGI.</p>

Bit field	Name	Description
		<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 (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	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 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

Bit field	Name	Description
		11: ETRP frequency divided by 8
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_{DTS} 0001: $f_{SAMPLING} = f_{CK_INT}$, $N = 2$ 0010: $f_{SAMPLING} = f_{CK_INT}$, $N = 4$ 0011: $f_{SAMPLING} = f_{CK_INT}$, $N = 8$ 0100: $f_{SAMPLING} = f_{DTS}/2$, $N = 6$ 0101: $f_{SAMPLING} = f_{DTS}/2$, $N = 8$ 0110: $f_{SAMPLING} = f_{DTS}/4$, $N = 6$ 0111: $f_{SAMPLING} = f_{DTS}/4$, $N = 8$ 1000: $f_{SAMPLING} = f_{DTS}/8$, $N = 6$ 1001: $f_{SAMPLING} = f_{DTS}/8$, $N = 8$ 1010: $f_{SAMPLING} = f_{DTS}/16$, $N = 5$ 1011: $f_{SAMPLING} = f_{DTS}/16$, $N = 6$ 1100: $f_{SAMPLING} = f_{DTS}/16$, $N = 8$ 1101: $f_{SAMPLING} = f_{DTS}/32$, $N = 5$ 1110: $f_{SAMPLING} = f_{DTS}/32$, $N = 6$ 1111: $f_{SAMPLING} = f_{DTS}/32$, $N = 8$</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>
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

Bit field	Name	Description
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)
TIM1	TIM5	TIM2	TIM3	TIM4
TIM8	TIM1	TIM2	TIM4	TIM5

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

Bit field	Name	Description
15	Reserved	Reserved, the reset value must be maintained

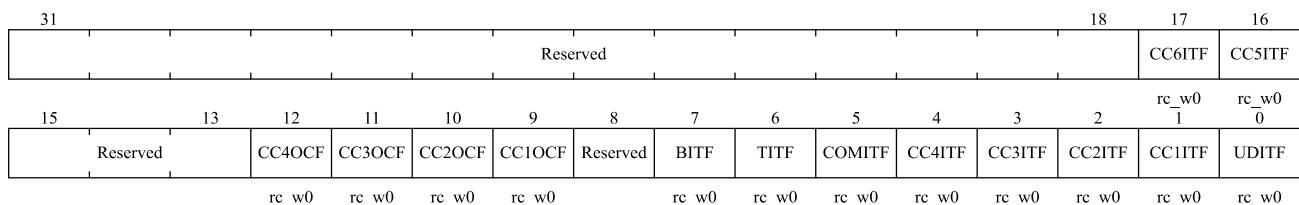
Bit field	Name	Description
14	TDEN	Trigger DMA request enable 0: Disable trigger DMA request 1: Enable trigger DMA request
13	COMDEN	COM DMA request enable 0: Disable COM DMA request 1: Enable COM DMA request
12	CC4DEN	Capture/Compare 4 DMA request enable 0: Disable capture/compare 4 DMA request 1: Enable 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
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

Bit field	Name	Description
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.
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	Reserved	Reserved, the reset value must be maintained
7	BITF	Break interrupt flag This bit is set by hardware once the brake input is active. This bit is cleared by software when the brake input becomes inactive. 0: No break event occurred 1: An active level has been detected
6	TITF	Trigger interrupt flag

Bit field	Name	Description
		<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 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 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>When the corresponding channel of CC1 is in output mode:</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. 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>When the corresponding channel of CC1 is in input mode:</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. 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"> – When TIMx_CTRL1.UPDIS = 0, and repeat counter value overflow or underflow (An update event is generated when the repeat counter equals 0). – When TIMx_CTRL1.UPRS = 0, TIMx_CTRL1.UPDIS = 0, and set the TIMx_EVTGEN.UDGN bit by software to reinitialize the CNT. – When TIMx_CTRL1.UPRS = 0, TIMx_CTRL1.UPDIS = 0, and the counter CNT is reinitialized by the trigger event. (See TIMx_SMCTRL Register description) <p>This bit is cleared by software.</p>

Bit field	Name	Description
		0: No update event occurred 1: Update interrupt occurred

9.4.7 Event generation registers (TIMx_EVTGEN)

Offset address: 0x14

Reset values: 0x0000

15							8	7	6	5	4	3	2	1	0
								BGN	TGN	CCUDGN	CC4GN	CC3GN	CC2GN	CC1GN	UDGN
							w	w	w	w	w	w	w	w	w

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 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 1: Generated a trigger event</p>
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>

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 TIMx_STS.CC1ITF is already pulled high, pull TIMx_STS.CC1OCF high.</p> <p>0: No action</p> <p>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</p> <p>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	rw	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> <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 < TIMx_CCDAT1, OC1REF signal of channel 1 is high, otherwise it is low. In down-counting mode, if TIMx_CNT > 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 < TIMx_CCDAT1, OC1REF signal of channel 1 is low, otherwise it is high. In down-counting mode, if TIMx_CNT > 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]	12	IC2PSC[1:0]	11	CC2SEL[1:0]	10		9	IC1F[3:0]	8		7		4	IC1PSC[1:0]	3		2		1		0
	rw		rw		rw				rw					rw		rw				rw		rw

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_{DTS} frequency</p> <p>0001: f_{SAMPLING} = f_{CLOCK}, N = 2</p> <p>0010: f_{SAMPLING} = f_{CLOCK}, N = 4</p> <p>0011: f_{SAMPLING} = f_{CLOCK}, N = 8</p>

Bit field	Name	Description
		0100: $f_{\text{SAMPLING}} = f_{\text{DTS}}/2$, $N = 6$ 0101: $f_{\text{SAMPLING}} = f_{\text{DTS}}/2$, $N = 8$ 0110: $f_{\text{SAMPLING}} = f_{\text{DTS}}/4$, $N = 6$ 0111: $f_{\text{SAMPLING}} = f_{\text{DTS}}/4$, $N = 8$ 1000: $f_{\text{SAMPLING}} = f_{\text{DTS}}/8$, $N = 6$ 1001: $f_{\text{SAMPLING}} = f_{\text{DTS}}/8$, $N = 8$ 1010: $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$, $N = 5$ 1011: $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$, $N = 6$ 1100: $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$, $N = 8$ 1101: $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, $N = 5$ 1110: $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, $N = 6$ 1111: $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, $N = 8$
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>

9.4.9 Capture/compare mode register 2 (TIMx_CCMOD2)

Offset address: 0x1C

Reset value: 0x0000 0000

See the description of the CCMOD1 register above

Output comparison mode:

31	Reserved				27	26	25	Reserved				22	21	20	Reserved				17	16
					OC9PEN							OC8PEN								
15	14	12	11	rw	10	9	8	rw	7	6	rw	4	3	2	1	rw	0			
OC4CEN	OC4MD[2:0]		OC4PEN	OC4FEN	CC4SEL[1:0]	OC3CEN	OC3MD[2:0]		OC3PEN	OC3FEN	CC3SEL[1:0]									

Bit field	Name	Description
31:27	Reserved	Reserved, the reset value must be maintained
26	OC9PEN	Output compare 9 preload enable <i>NOTE: This function is only for TIM1</i>
25:22	Reserved	Reserved, the reset value must be maintained
21	OC8PEN	Output compare 8 preload enable <i>NOTE: This function is only for TIM1</i>
20:17	Reserved	Reserved, the reset value must be maintained
16	OC7PEN	Output compare 7 preload enable <i>NOTE: This function is only for TIM1</i>
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]	Capture/Compare 4 selection These bits are used to select the input/output and input mapping of the channel 00: CC4 channel is configured as output 01: CC4 channel is configured as input, IC4 is mapped on TI4 10: CC4 channel is configured as input, IC4 is mapped on TI3 11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is only active when the internal trigger input is selected by TIMx_SMCTRL.TSEL. <i>Note: CC4SEL is writable only when the channel is off (TIMx_CCEN.CC4EN = 0).</i>
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]	Capture/Compare 3 selection These bits are used to select the input/output and input mapping of the channel 00: CC3 channel is configured as output 01: CC3 channel is configured as input, IC3 is mapped to TI3 10: CC3 channel is configured as input, IC3 is mapped on TI4 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. <i>Note: CC3SEL is writable only when the channel is off (TIMx_CCEN.CC3EN = 0).</i>

Input capture mode:

15	12	11	10	9	8	7	4	3	2	1	0
IC4F[3:0]	IC4PSC[1:0]	CC4SEL[1:0]			IC3F[3:0]		IC3PSC[1:0]		CC3SEL[1:0]		

Bit field	Name	Description
15:12	IC4F[3:0]	Input Capture 4 filter

Bit field	Name	Description
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	
													CC6P	CC6EN	Reserved	CC5P	CC5EN			
15	14	13	12	11	10	9	8	7	6	rw	rw	rw	5	rw	4	3	2	rw	rw	
Reserved	CC4P	CC4EN	CC3NP	CC3NEN	CC3P	CC3EN	CC2NP	CC2NEN	CC2P	CC2EN	CC1NP	CC1NEN	CC1P	CC1EN			1	0	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	CC6P	<p>Capture/Compare 6 output polarity</p> <p>See TIMx_CCEN.CC1P description.</p>
20	CC6EN	<p>Capture/Compare 6 output enable</p> <p>See TIMx_CCEN.CC1EN description.</p>
19: 18	Reserved	Reserved, the reset value must be maintained
17	CC5P	<p>Capture/Compare 5 output polarity</p> <p>See TIMx_CCEN.CC1P description.</p>

Bit field	Name	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. <i>NOTE: This function is only for TIM1</i>
6	CC2NEN	Capture/Compare 2 complementary output enable See TIMx_CCEN.CC1NEN description. <i>NOTE: This function is only for TIM1</i>
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.
1	CC1P	Capture/Compare 1 output polarity When the corresponding channel of CC1 is in output mode: 0: OC1 active high 1: OC1 active low 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

Bit field	Name	Description
		<p>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.</p> <p>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</p> <p>1: Enable capture</p>

Table 9-4 Output control bits of complementary OCx and OCxN channels with break function

Control bits					Output state ¹⁾	
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 state) OCx=CCxP, OCx_EN=1	OCxREF + polarity, OCxN= OCxREF xor CCxNP, OCxN_EN=1
		1	1	0	OCxREF + polarity, OCx= OCxREF xor CCxP, OCx_EN=1	Off-state (Output enabled with inactive state) OCxN=CCxNP, OCxN_EN=1
		1	1	1	OCxREF + polarity + dead-time,	Complementary to OCxREF + polarity + dead-

Control bits						Output state ¹⁾			
MOEN	OSSI	OSSR	CCxEN	CCxNEN	OCx Output state	OCxN Output state			
					OCx_EN=1	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					

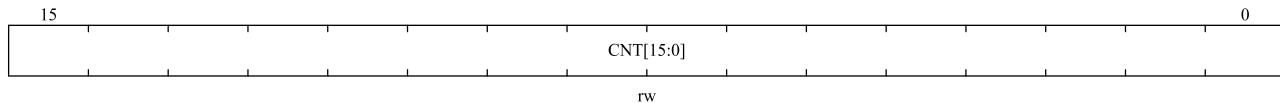
1. If both outputs of a channel are not used (CCxEN = CCxNEN = 0), OIx, OIxN, CCxP and CCxNP must all be cleared.

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

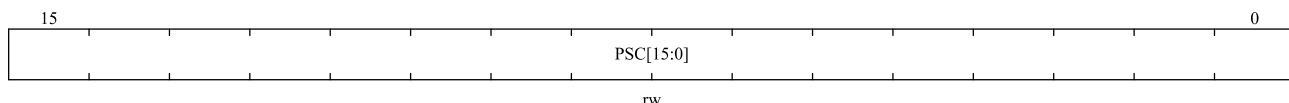


Bit field	Name	Description
15:0	CNT[15:0]	Counter value

9.4.12 Prescaler (TIMx_PSC)

Offset address: 0x28

Reset value: 0x0000

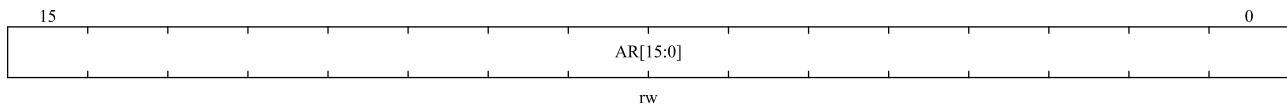


Bit field	Name	Description
15:0	PSC[15:0]	Prescaler value Counter clock fCK_CNT = fCK_PSC/ (PSC [15:0] +1). Each time an update event occurs, the PSC value is loaded into the active prescaler register.

9.4.13 Auto-reload register (TIMx_AR)

Offset address: 0x2C

Reset values: 0xFFFF

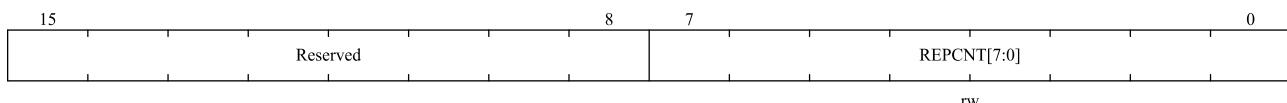


Bit field	Name	Description
15:0	AR[15:0]	<p>Auto-reload value</p> <p>These bits define the value that will be loaded into the actual auto-reload register.</p> <p>See Section 9.3.1 for more details.</p> <p>When the TIMx_AR.AR [15:0] value is null, the counter does not work.</p>

9.4.14 Repeat count registers (TIMx_REPCNT)

Offset address: 0x30

Reset value: 0x0000

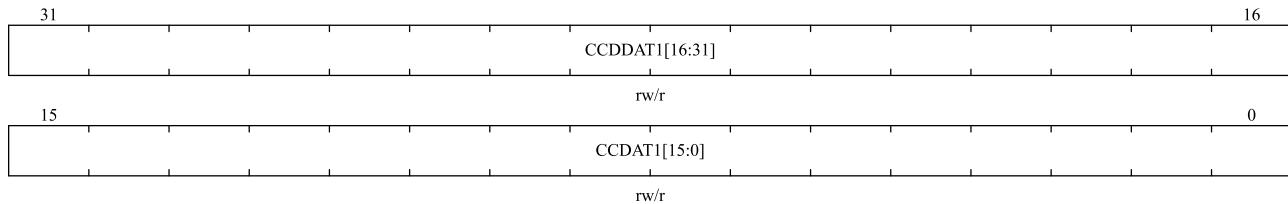


Bit field	Name	Description
15:8	Reserved	Reserved, the reset value must be maintained
7:0	REPCNT[7:0]	<p>Repetition counter value</p> <p>Repetition counter is used to generate the update event or update the timer registers only after a given number (N+1) cycles of the counter, where N is the value of 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_CCDAT1)

Offset address: 0x34

Reset value: 0x0000 0000

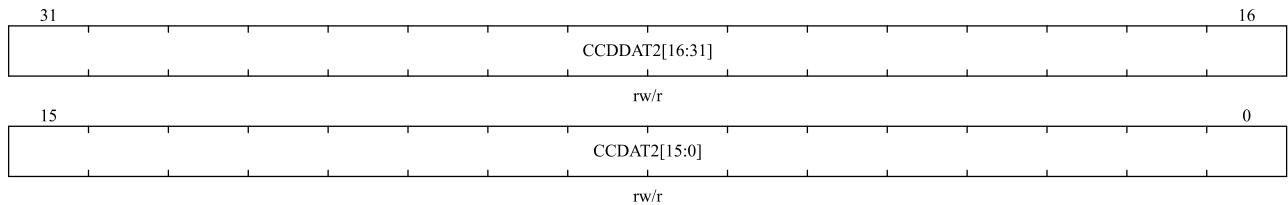


Bit field	Name	Description
31:16	CCDDAT1[15:0]	<p>Capture/Compare 1 value, special for center-aligned asymmetric mode. Only for TIM1.</p> <ul style="list-style-type: none"> CC1 channel can only be configured as output: <p>CCDDAT1 contains the value to be compared to the counter TIMx_CNT (only when TIMx_CTRL1.DIR = 1 and in asymmetric mode), 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>
15:0	CCDAT1[15:0]	<p>Capture/Compare 1 value</p> <ul style="list-style-type: none"> CC1 channel is configured as output: <p>CCDAT1 contains the value to be compared to the counter TIMx_CNT (except when TIMx_CTRL1.DIR = 1 and in asymmetric mode), 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"> CC1 channel is configured as input: <p>CCDAT1 contains the counter value transferred by the last input capture 1 event (IC1).</p> <p>When configured as input mode, register CCDAT1 and CCDDAT1 are only readable.</p> <p>When configured as output mode, register CCDAT1 and CCDDAT1 are readable and writable.</p>

9.4.16 Capture/compare register 2 (TIMx_CCDAT2)

Offset address: 0x38

Reset value: 0x0000 0000

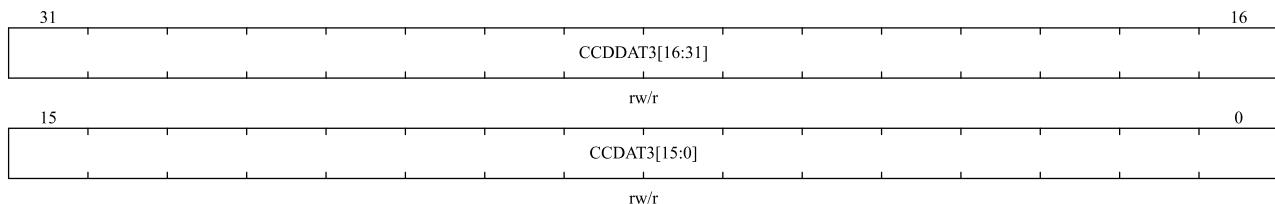


Bit field	Name	Description
31:16	CCDDAT2[15:0]	<p>Capture/Compare 2 values, special for center-aligned asymmetric mode. Only for TIM1.</p> <ul style="list-style-type: none"> ■ CC2 channel can only be configured as output: <p>CCDDAT2 contains the value to be compared to the counter TIMx_CNT (only when TIMx_CTRL1.DIR = 1 and in asymmetric mode), 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>
15:0	CCDAT2[15:0]	<p>Capture/Compare 2 values</p> <ul style="list-style-type: none"> ■ CC2 channel is configured as output: <p>CCDAT2 contains the value to be compared to the counter TIMx_CNT (except when TIMx_CTRL1.DIR = 1 and in asymmetric mode), 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"> ■ CC2 channel is configured as input: <p>CCDAT2 contains the counter value transferred by the last input capture 2 event (IC2).</p> <p>When configured as input mode, registers CCDDAT2 and CCDDAT2 are only readable.</p> <p>When configured as output mode, registers CCDDAT2 and CCDDAT2 are readable and writable.</p>

9.4.17 Capture/compare register 3 (TIMx_CC DAT3)

Offset address: 0x3C

Reset value: 0x0000 0000



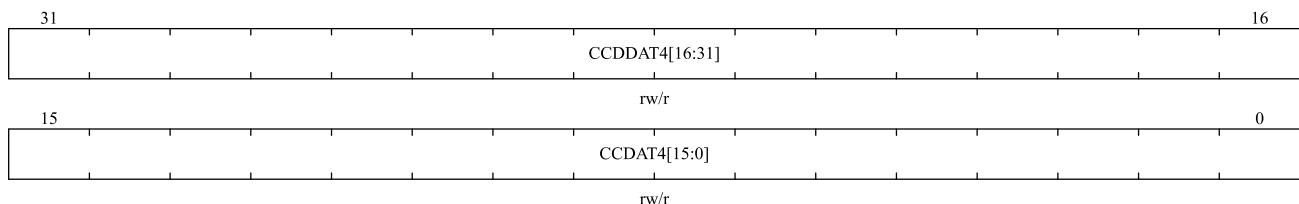
Bit field	Name	Description
31:16	CCDDAT3[15:0]	<p>Capture/Compare 3 value, special for center-aligned asymmetric mode. Only for TIM1.</p> <ul style="list-style-type: none"> ■ CC3 channel can only be configured as output: <p>CCDDAT3 contains the value to be compared to the counter TIMx_CNT (only when TIMx_CTRL1.DIR = 1 and in asymmetric mode), signaling on the OC3 output.</p> <p>If the preload feature is not selected in TIMx_CCMOD1.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>

Bit field	Name	Description
15:0	CCDAT3[15:0]	<p>Capture/Compare 3 value</p> <ul style="list-style-type: none"> ■ CC3 channel is configured as output: CCDAT3 contains the value to be compared to the counter TIMx_CNT (except when TIMx_CTRL1.DIR = 1 and in asymmetric mode), signaling on the OC3 output. 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. ■ CC3 channel is configured as input: CCDAT3 contains the counter value transferred by the last input capture 3 event (IC3). When configured as input mode, register CCDAT3 and CCDDAT3 are only readable. When configured as output mode, register CCDAT3 and CCDDAT3 are readable and writable.

9.4.18 Capture/compare register 4 (TIMx_CC DAT4)

Offset address: 0x40

Reset value: 0x0000 0000



Bit field	Name	Description
31:16	CCDDAT4[15:0]	<p>Capture/Compare 4 value, special for center-aligned asymmetric mode. Only for TIM1.</p> <ul style="list-style-type: none"> ■ CC4 channel can only be configured as output: CCDDAT4 contains the value to be compared to the counter TIMx_CNT (only when TIMx_CTRL1.DIR = 1 and in asymmetric mode), signaling on the OC4 output. 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.
15:0	CCDAT4[15:0]	<p>Capture/Compare 4 value</p> <ul style="list-style-type: none"> ■ CC4 channel is configured as output: CCDAT4 contains the value to be compared to the counter TIMx_CNT (except when TIMx_CTRL1.DIR = 1 and in asymmetric mode), signaling on the OC4 output. 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. ■ CC4 channel is configured as input: CCDAT4 contains the counter value transferred by the last input capture 4 event (IC4). When configured as input mode, register CCDAT4 and CCDDAT4 are only readable. When configured as output mode, register CCDAT4 and CCDDAT4 are readable and writable.

9.4.19 Break and Dead-time registers (TIMx_BKDT)

Offset address: 0x44

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	0
MOEN	AOEN	BKP	BKEN	OSSR	OSSI	LCKCFG[1:0]			DTGN[7:0]

rw rw 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	<p>Main Output enable</p> <p>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.</p> <p>0: OC and OCN outputs are disabled or forced to idle state.</p> <p>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).</p>
14	AOEN	<p>Automatic output enable</p> <p>0: Only software can set TIMx_BKDT.MOEN;</p> <p>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.</p>
13	BKP	<p>Break input polarity</p> <p>0: Low level of the brake input is valid</p> <p>1: High level of the brake input is valid</p> <p><i>Note: Any write to this bit requires an APB clock delay to take effect.</i></p>
12	BKEN	<p>Break enable</p> <p>0: Disable brake input (BRK and CCS clock failure events)</p> <p>1: Enable brake input (BRK and CCS clock failure events)</p> <p><i>Note: Any write to this bit requires an APB clock delay to take effect.</i></p>
11	OSSR	<p>Off-state Selection for Run Mode</p> <p>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.</p> <p>0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal = 0)</p> <p>1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxEN = 1 or CCxNEN = 1. Then, OC/OCN enable output signal = 1</p> <p>For more details, See Section 9.4.10, capture/compare enablement registers (TIMx_CCEN).</p>

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, OC/OCN outputs are disabled (OC/OCN enable output signal = 0)</p> <p>1: When inactive, OC/OCN outputs are forced with their with their idle level as soon as CCxEN = 1 or CCxNEN = 1. Then, OC/OCN 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"> – No write protected. <p>01:</p> <ul style="list-style-type: none"> – LOCK Level 1 <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"> – LOCK Level 2 <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"> – LOCK Level 3 <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:</p> $\text{dead time} = \text{DTGN}[7:0] \times (\text{tDTS})$ <p>DTGN[7:5] = 10x:</p> $\text{dead time} = (64 + \text{DTGN}[5:0]) \times (2 \times \text{tDTS})$ <p>DTGN[7:5] = 110:</p> $\text{dead time} = (32 + \text{DTGN}[4:0]) \times (8 \times \text{tDTS})$ <p>DTGN [then] = 111:</p> $\text{dead time} = (32 + \text{DTGN [4:0]}) \times (16 \times \text{tDTS})$ <p>tDTS value see TIMx_CTRL1.CLKD [1:0].</p>

9.4.20 DMA Control register (TIMx_DCTRL)

Offset address: 0x48

Reset value: 0x0000

15	13	12	8	7	5	4	0
Reserved	DBLEN[4:0]		Reserved		DBADDR[4:0]		
		RW				RW	

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</p> <p>00001: 2 times transfers</p> <p>00010: 3 times transfers</p> <p>...</p> <p>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

Diagram of a 16-bit register labeled BURST[15:0]. The register has a width of 16 bits, ranging from bit 15 on the left to bit 0 on the right. The label "BURST[15:0]" is centered above the register. Below the register, the letters "rw" are centered, indicating it is a readable and writable register.

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 \times 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</p>

Bit field	Name	Description
		<p>(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	14	12	11	10	9	8	7	6	4	3	2	1	0
OC6CEN	OC6MD[2:0]	OC6PEN	OC6FEN	Reserved		OC5CEN	OC5MD[2:0]	OC5PEN	OC5FEN				
rw	rw	rw	rw			rw	rw	rw	rw	rw			

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_CCDAT5)

Offset address: 0x58

Reset value: 0x0000

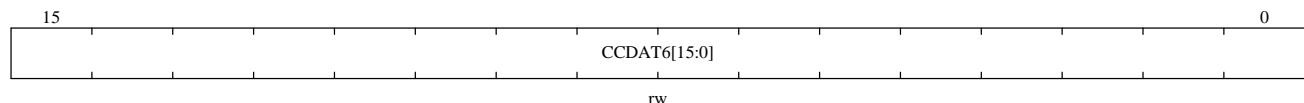
15	CCDAT5[15:0]	0
rw		

Bit field	Name	Description
15:0	CCDAT5[15:0]	<p>Capture/Compare 5 value</p> <ul style="list-style-type: none"> ■ CC5 channel can only configured as output: <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>

9.4.24 Capture/compare register 6 (TIMx_CCDAT6)

Offset address: 0x5C

Reset value: 0x0000

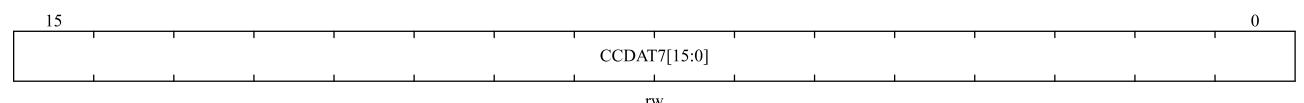


Bit field	Name	Description
15:0	CCDAT6[15:0]	<p>Capture/Compare 6 value</p> <ul style="list-style-type: none"> ■ CC6 channel can only configured as output: <p>CCDAT6 contains the value to be compared to the counter TIMx_CNT, signaling on the OC6 output.</p> <p>If the preload feature is not selected in 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>

9.4.25 Capture/compare register 7 (TIMx_CCDAT7)

Offset address: 0x60

Reset value: 0x0000

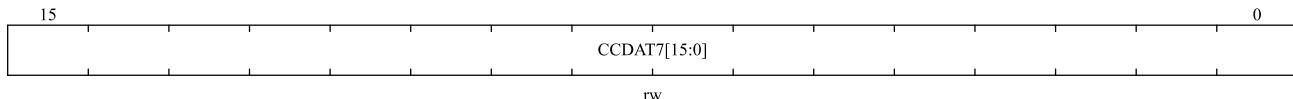


Bit field	Name	Description
15:0	CCDAT7[15:0]	<p>Capture/Compare 7 value</p> <ul style="list-style-type: none"> ■ CC7 channel can only configured as output: <p>CCDAT7 contains the value to be compared to the counter TIMx_CNT, signaling on the OC7 output.</p> <p>If the preload feature is not selected in TIMx_CCMOD2.OC7PEN 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><i>NOTE: This function is only for TIM1</i></p>

9.4.26 Capture/compare register 8 (TIMx_CC DAT8)

Offset address: 0x64

Reset value: 0x0000

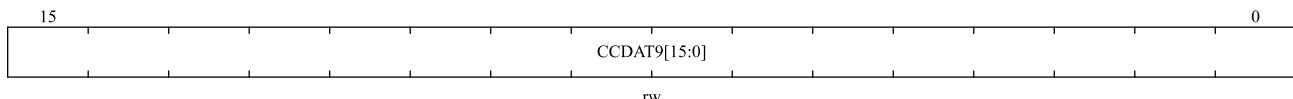


Bit field	Name	Description
15:0	CC DAT8[15:0]	<p>Capture/Compare 8 value</p> <ul style="list-style-type: none"> ■ CC8 channel can only be configured as output: <p>CC DAT8 contains the value to be compared to the counter TIMx_CNT, signaling on the OC8 output.</p> <p>If the preload feature is not selected in TIMx_CCMOD2.OC8PEN 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><i>NOTE: This function is only for TIM1</i></p>

9.4.27 Capture/compare register 9 (TIMx_CC DAT9)

Offset address: 0x68

Reset value: 0x0000

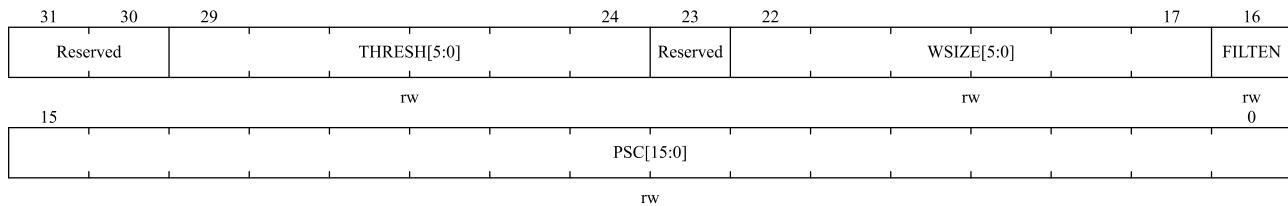


Bit field	Name	Description
15:0	CC DAT9[15:0]	<p>Capture/Compare 9 value</p> <ul style="list-style-type: none"> ■ CC9 channel can only be configured as output: <p>CC DAT9 contains the value to be compared to the counter TIMx_CNT, signaling on the OC9 output.</p> <p>If the preload feature is not selected in TIMx_CCMOD2.OC9PEN 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><i>NOTE: This function is only for TIM1</i></p>

9.4.28 Break Filter (TIMx_BRKFILT)

Offset address: 0x6C

Reset value: 0x0000 0000



Bit field	Name	Description
31:30	Reserved	Reserved, the reset value must be maintained
29:24	THRESH[5:0]	<p>Threshold number of sample logic level to be valid, maximum 63:</p> <p>Threshold value for a valid logic level. Within sample window if number of logic high is more than or equal to threshold value, next logic level will be logic high. Same rule applies to logic low. If both number of 1's and 0's inside window are smaller than threshold, filter output stays unchanged. Threshold value should set to more than or equal to half of Window value.</p> <p>Recommend threshold range is:</p> <p>Minimum: 1 pre-scale clock cycle more than ceiling value of max glitch size (in pre-scale clock cycle) and need to larger than half of window size.</p> <p>for example, if glitch size is $3.2 \times (\text{pre-scale clock period})$, threshold should be $\lceil 3.2 \rceil = 4 + 1 = 5$</p> <p>Maximum: floor value of minimum size of valid signal (in pre-scale clock cycle) and need to be smaller than window size.</p> <p>For example, if minimum message size is $3.2 \times (\text{pre-scale clock period})$, threshold should be floor $(3.2) = 3$.</p>
23	Reserved	Reserved, the reset value must be maintained
22:17	WSIZE[5:0]	<p>Window size value for logic level check, maximum 63:</p> <p>Window size decides how many sampled values will take into consideration for getting next logic level. Build-in FIFO is 64 bits with maximum index 63 which can only set window size to be 63.</p>
16	FILTEN	<p>Filter enable:</p> <p>0: Filter disable.</p> <p>1: Enable filter function.</p>
15:0	PSC[15:0]	<p>Prescaler register value for configure filter sample clock:</p> <p>For this filter, it supports 65535 scale (16 bits).</p> <p>Clock prescaler scaling system clock to sample clock. Sample clock decides distance between two sample point. Only value at sample point will take into consideration for valid logic level calculation.</p>

10 General-purpose timers (TIM2, TIM3, TIM4 and TIM5)

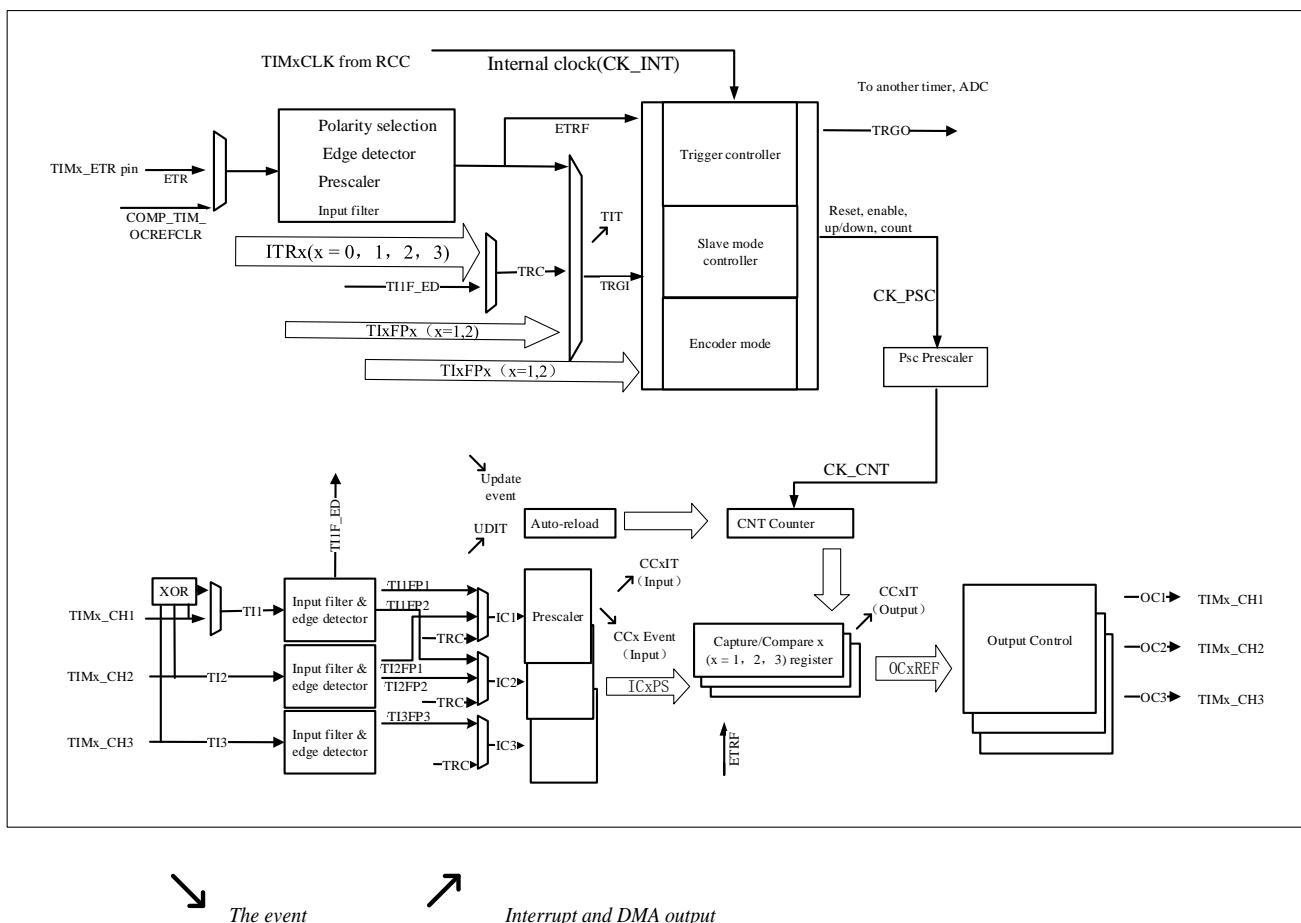
10.1 General-purpose timers introduction

The general-purpose timers (TIM2, TIM3, TIM4 and TIM5) is mainly used in the following purposes: 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)
- TIM2 and TIM3 up to 3 channels, TIM4 and TIM5 up to 4 channels.
- Channel's working modes: 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
- Timer can be controlled by external signal
- Timers can be linked together 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 TIM2



For TIM2, the ETR input is from IOM only.

For TIM2, The capture channel 1 input can come from IOM.

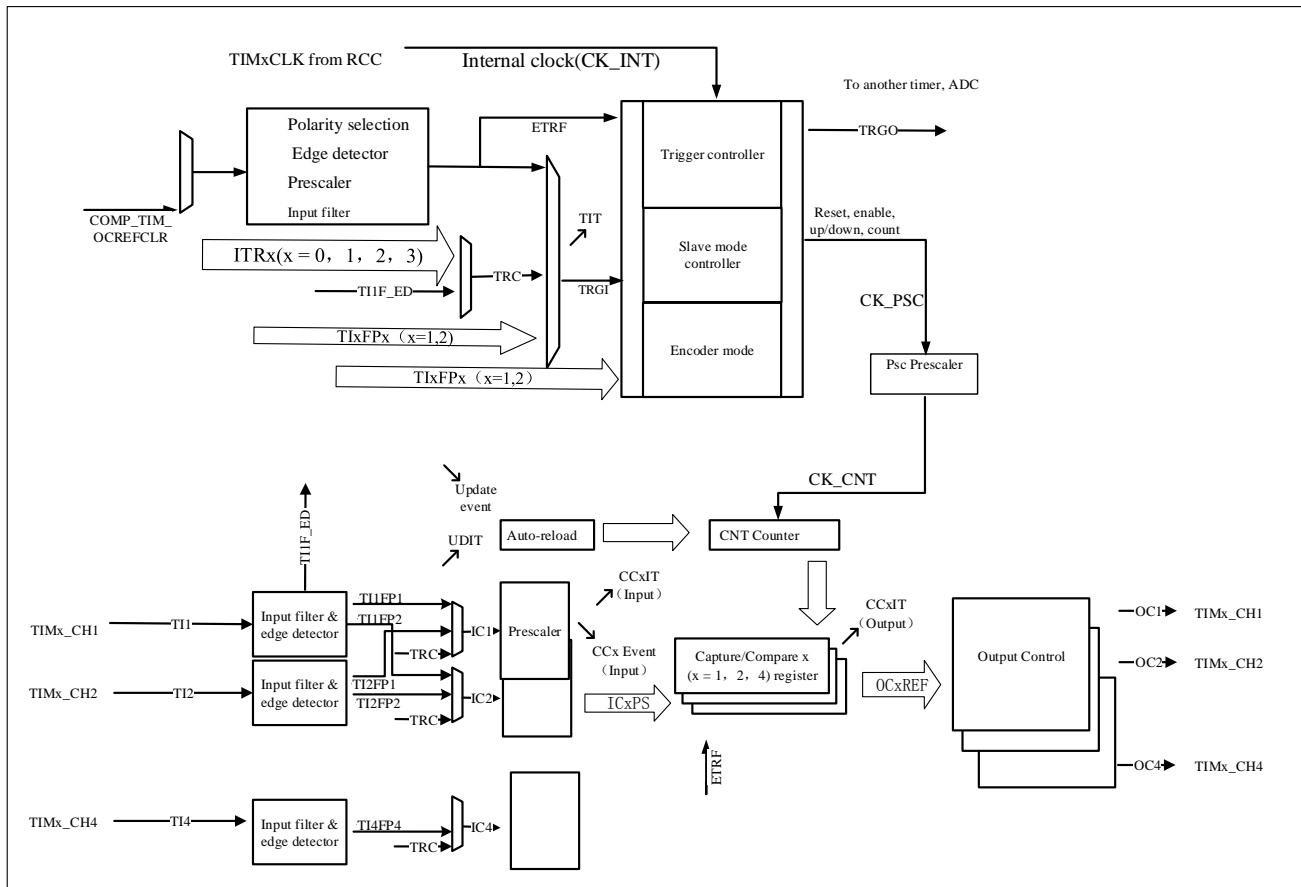
For TIM2, capture channel 2 comes from IOM.

For TIM2, capture channel 3 comes from IOM or LSI.

There is a sliding filter in front of TIM2_CH1, TIM2_CH2 and TIM2_CH3. Sliding filter is only for IOM input, not for LSI input.

General-purpose timers description

图 10-1 Block diagram of TIM3



 The event

 Interrupt and DMA output

For TIM3, ETR input is not support.

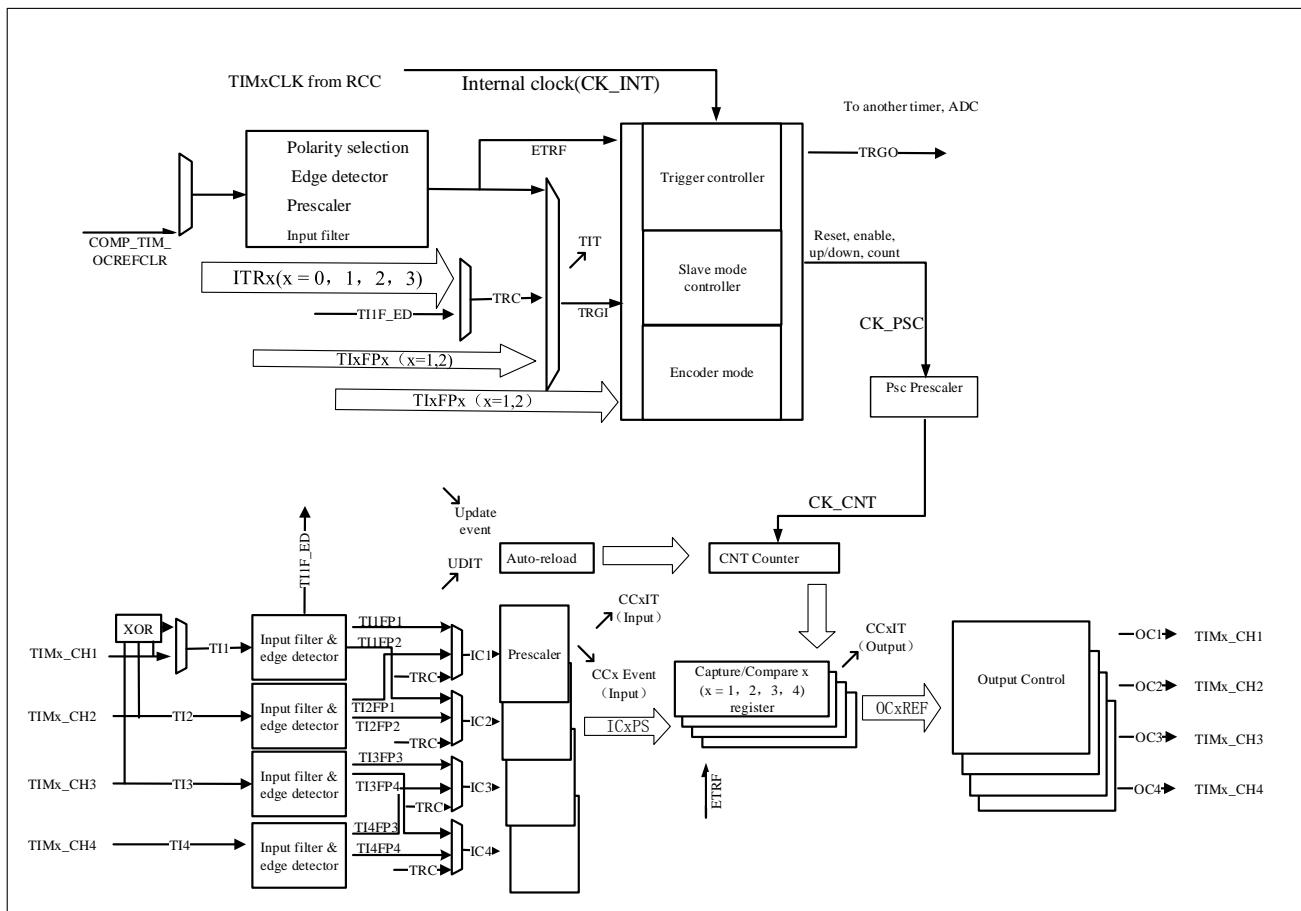
For TIM3, The capture channel 1 input can come from IOM.

For TIM3, capture channel 2 comes from IOM only.

For TIM3, capture channel 4 comes from IOM only.

There is a sliding filter in front of TIM3_CH1, TIM3_CH2 and TIM3_CH4. Sliding filter is only for IOM input.

图 10-2 Block diagram of TIM4



For TIM4, ETR input is not support.

For TIM4, The capture channel 1 input can come from IOM.

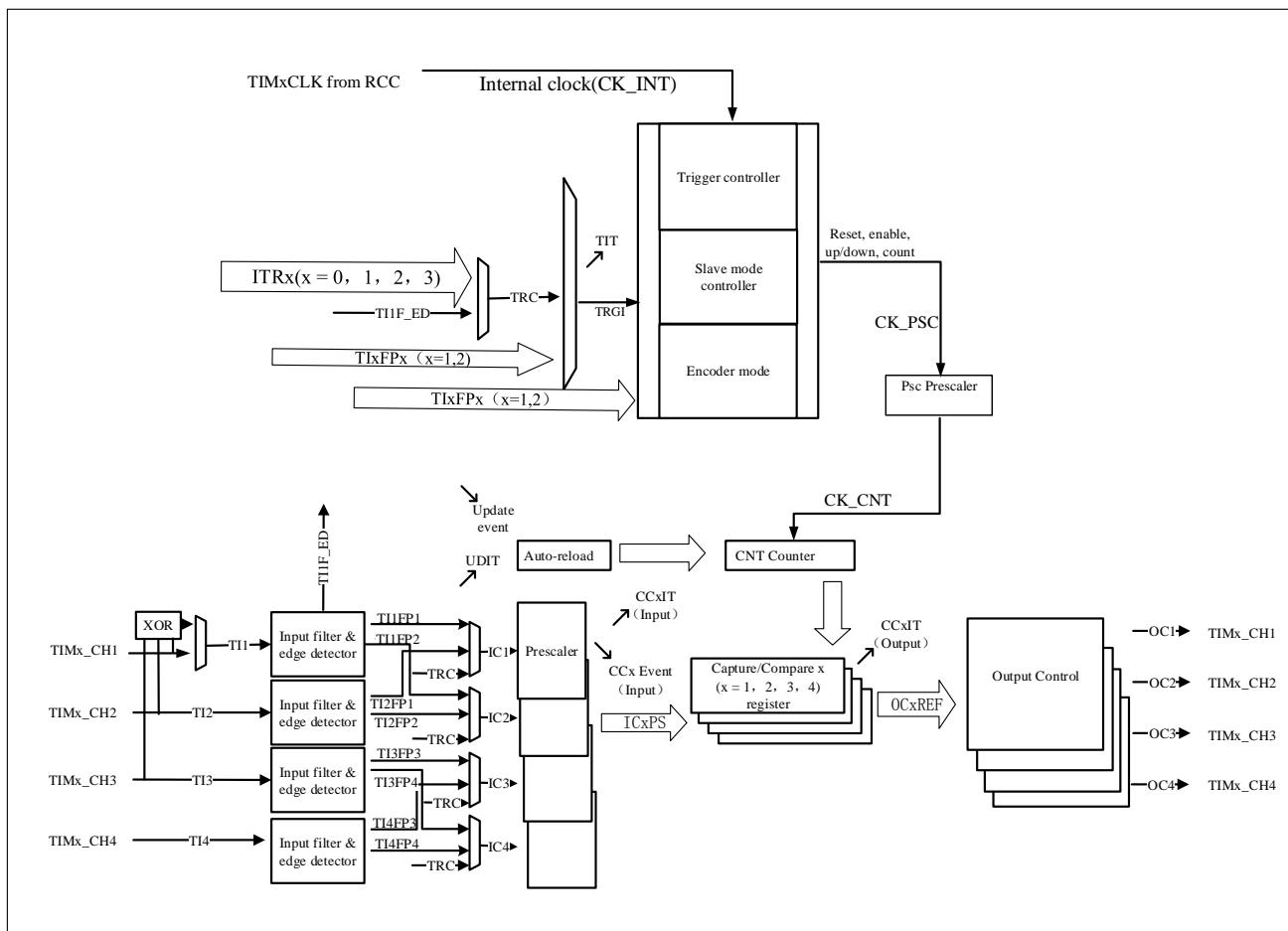
For TIM4, capture channel 2 comes from IOM only.

For TIM4, capture channel 3 comes from IOM only.

For TIM4, capture channel 4 comes from IOM only.

There is a sliding filter in front of TIM4_CH1, TIM4_CH2, TIM4_CH3 and TIM4_CH4. Sliding filter is only for IOM input.

图 10-3 Block diagram of TIM5



 The event

 Interrupt and DMA output

For TIM5, ETR input is not support.

For TIM5, The capture channel 1 input can come from IOM.

For TIM5, capture channel 2 comes from IOM only.

For TIM5, capture channel 3 comes from IOM only.

For TIM5, capture channel 4 comes from IOM only.

There is a sliding filter in front of TIM5_CH1, TIM5_CH2, TIM5_CH3 and TIM5_CH4. Sliding filter is only for IOM input.

10.3 TIM2、TIM3、TIM4、TIM5 function 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.

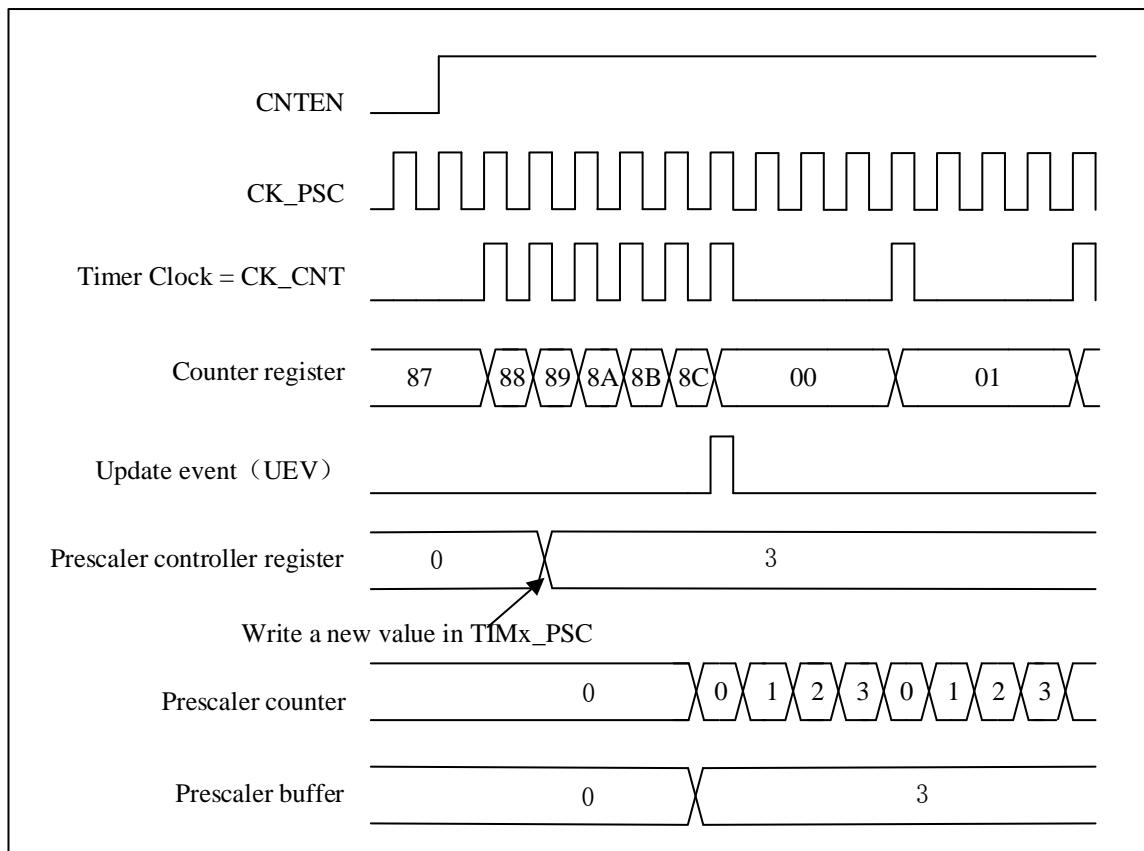
Depending on the setting of the auto-reload preload enable bit (TIMx_CTRL1.AR PEN), the value of the preload

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 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.ARREN = 1.
- The prescaler shadow register is reloaded with the preload value(TIMx_PSC).

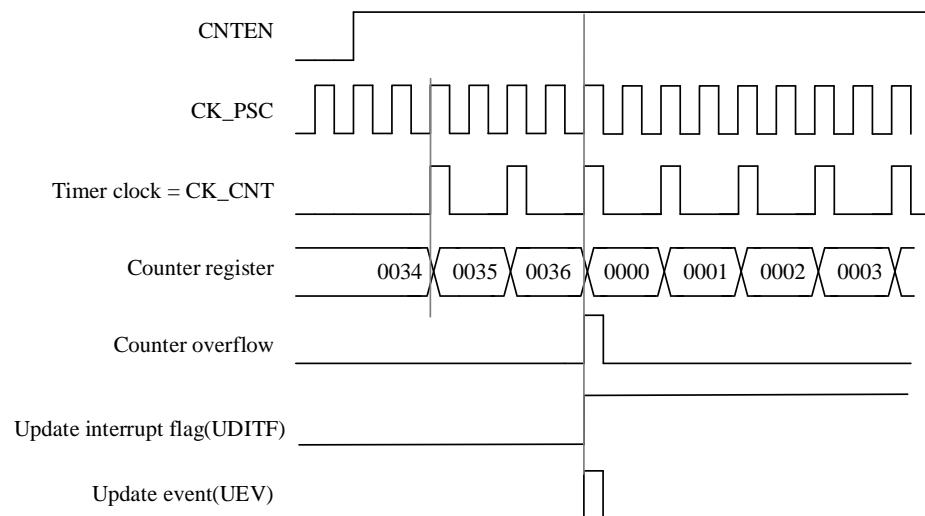
To avoid updating the shadow registers when new values are written to the preload registers, you can disable the update by setting TIMx_CTRL1.UPDIS=1.

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

Internal clock divided by 2



Internal clock divided by N

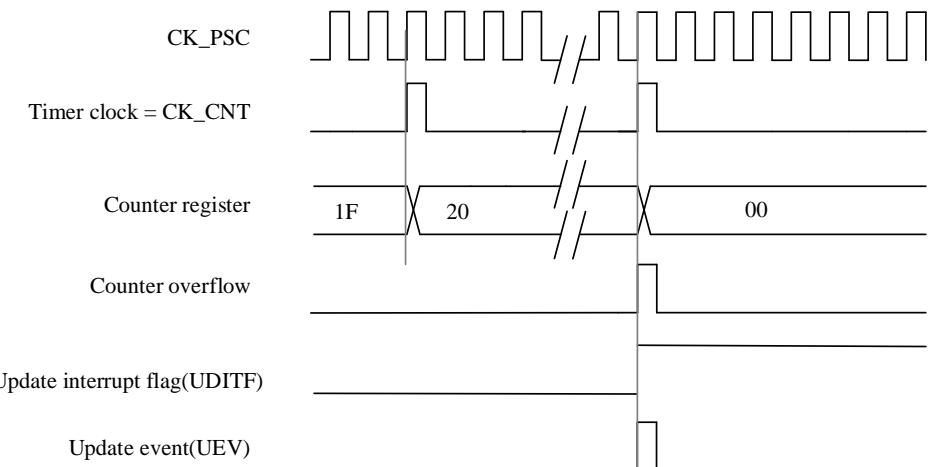
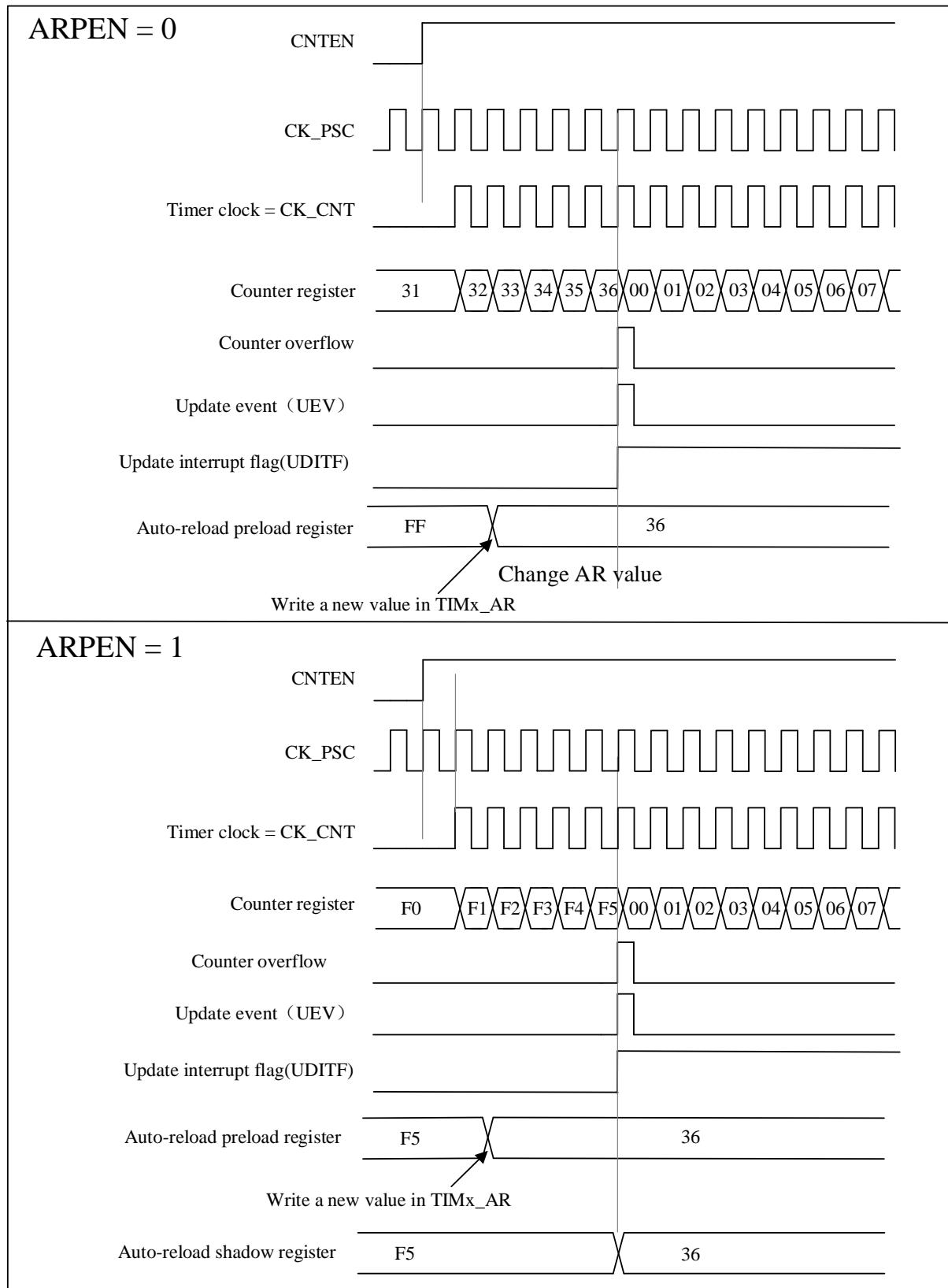


Figure 10-4 Timing diagram of the up-counting, update event when ARPEN=0/1



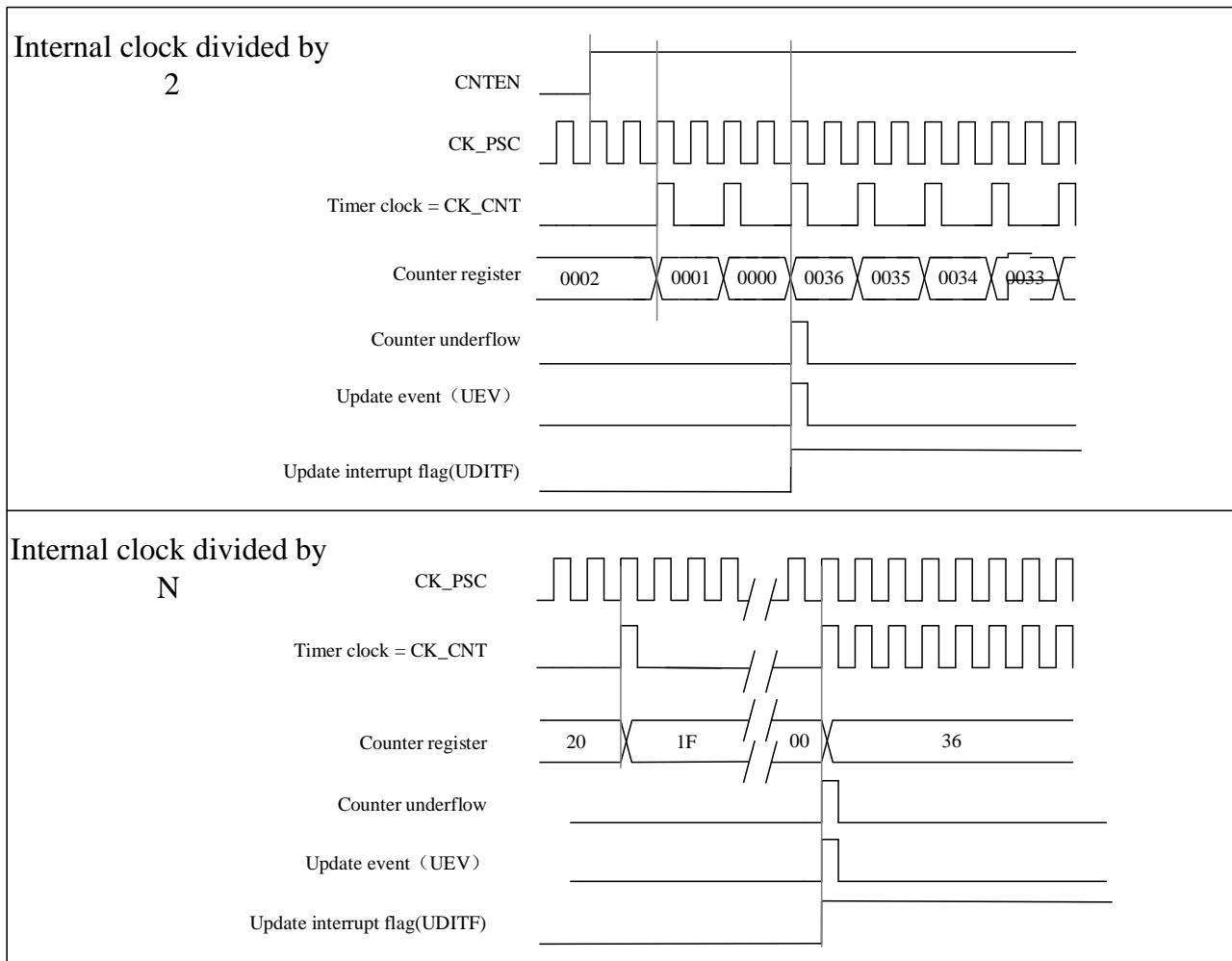
10.3.2.2 Down-counting mode

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

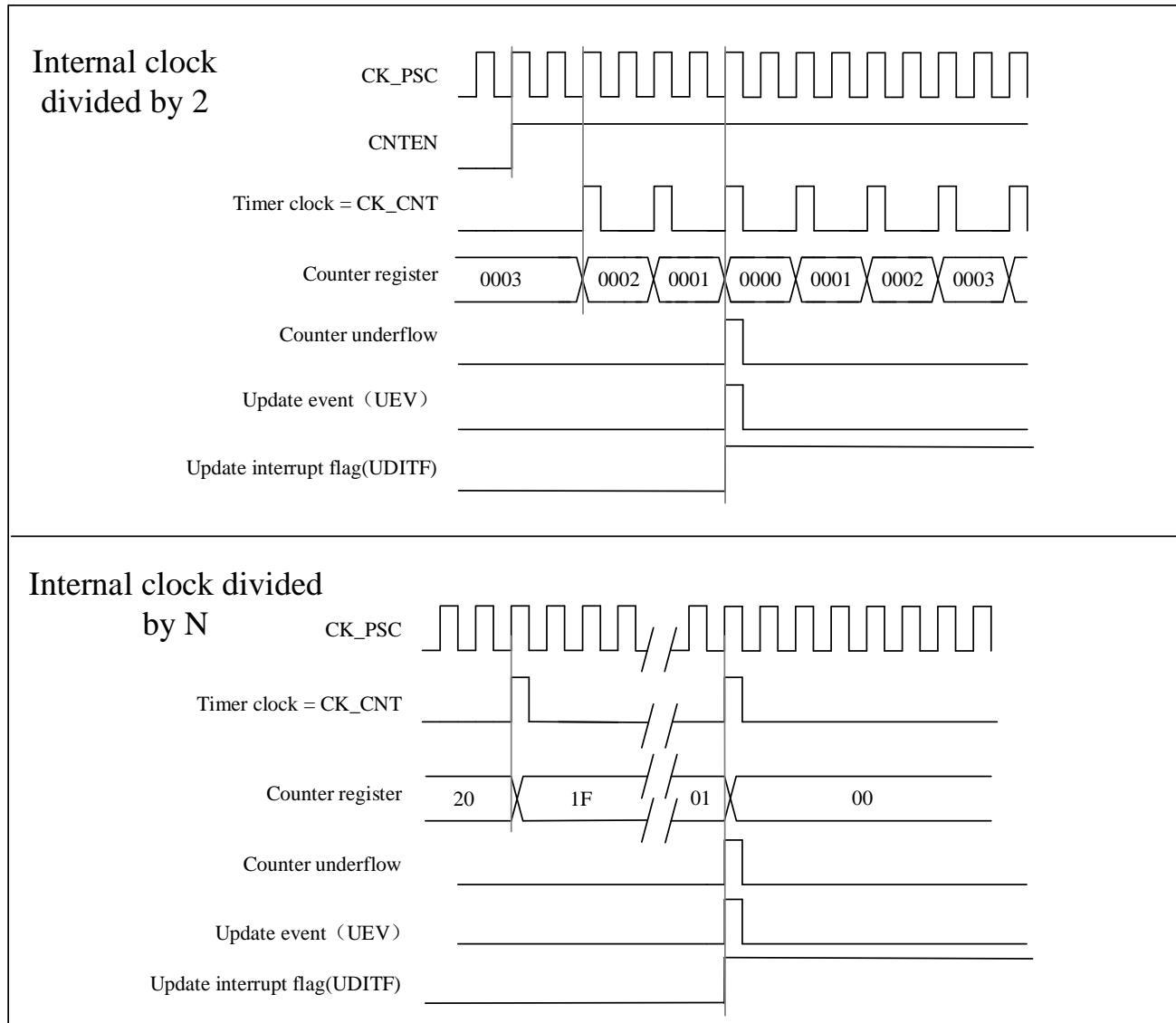
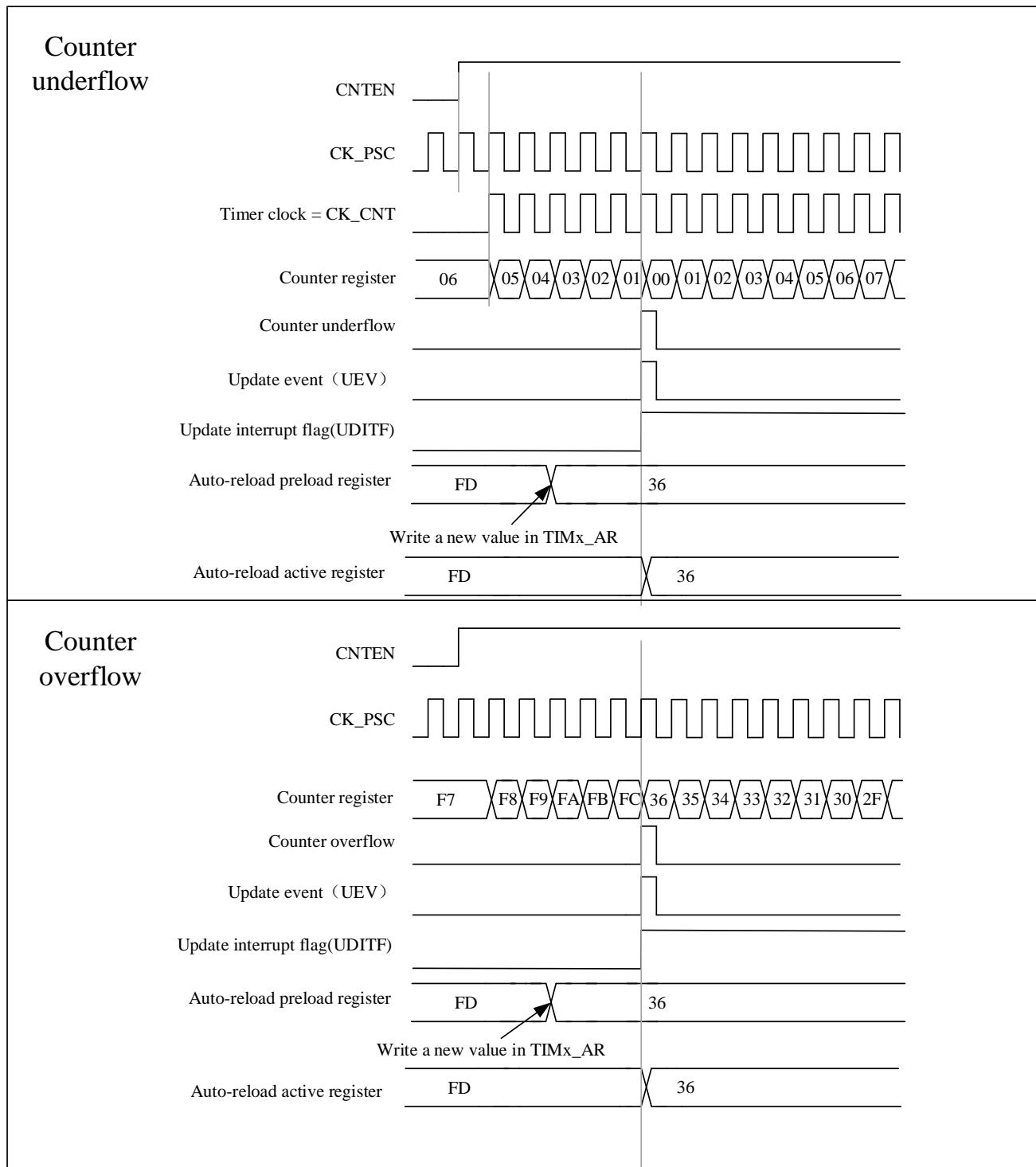


Figure 10-7 A center-aligned sequence diagram that includes counter overflows and underflows (ARPEN = 1)



10.3.3 Clock selection

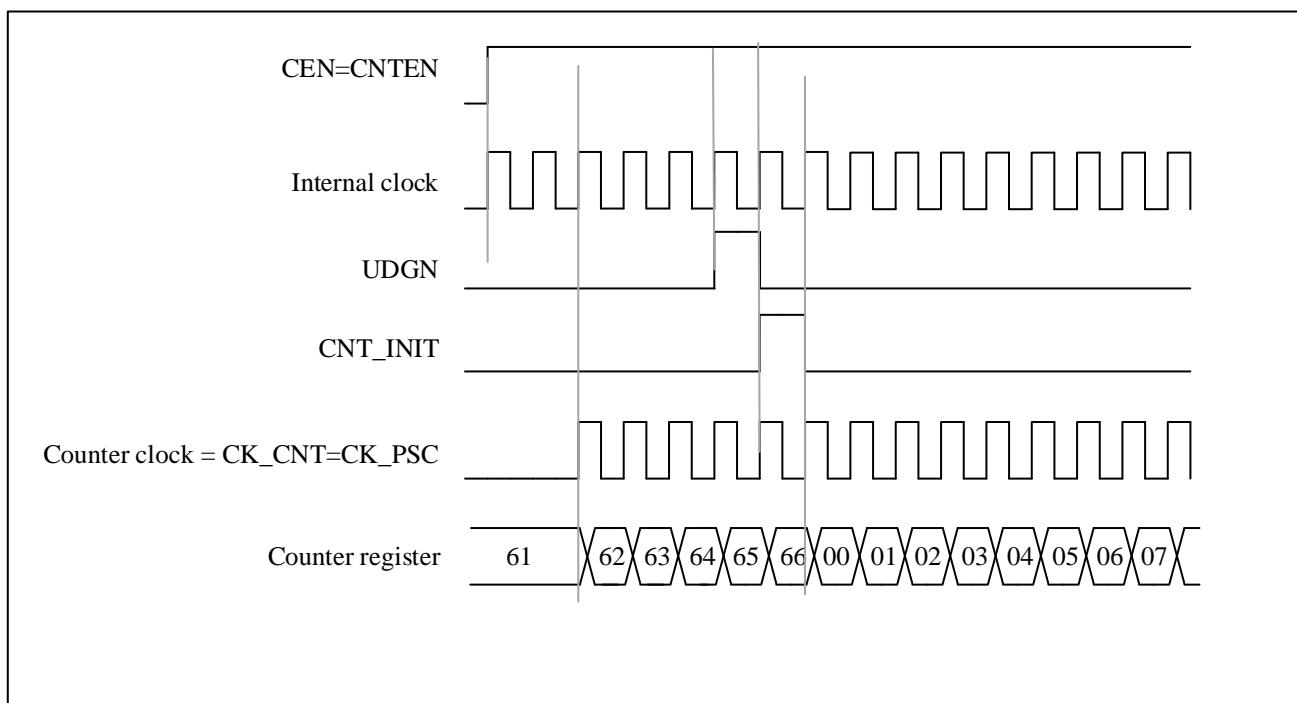
- The internal clock of timers : CK_INT

- 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)

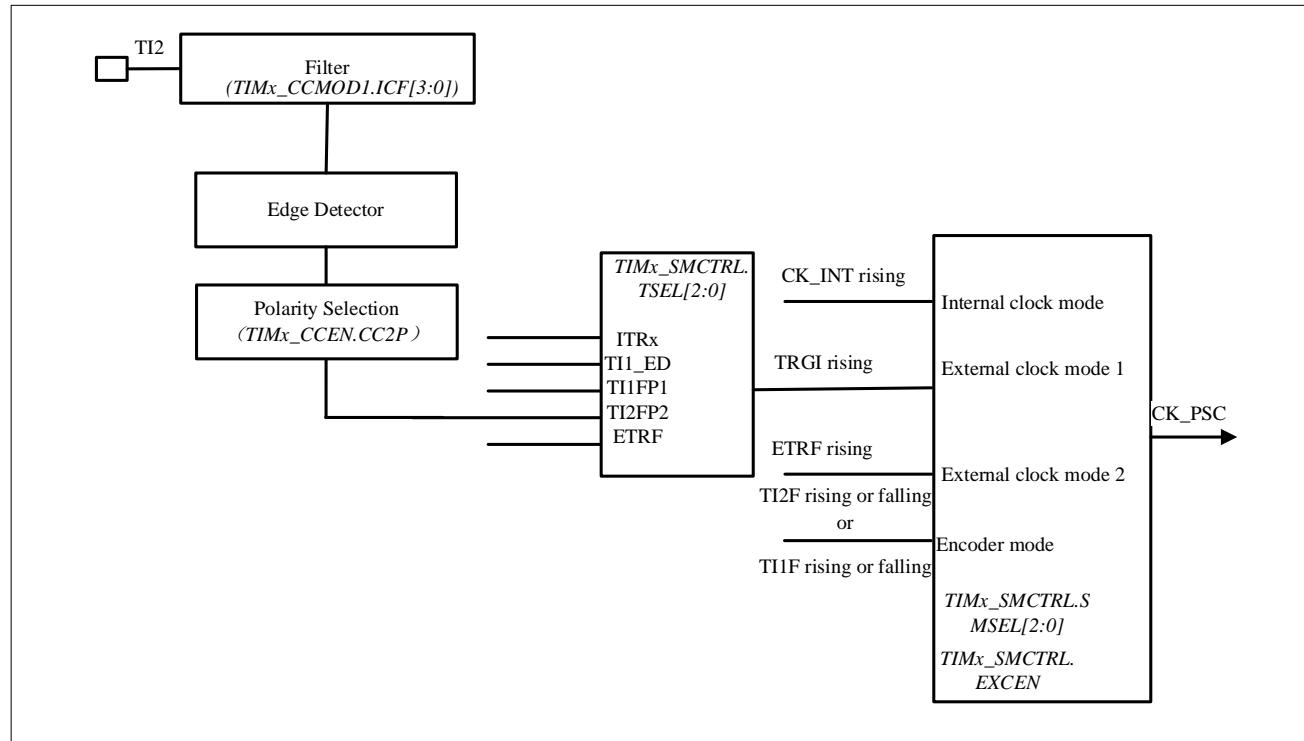
When the `TIMx_SMCTRL.SMSEL` is equal to “000”, the slave mode controller is disabled. The three control bits (`TIMx_CTRL1.CNTEN`、`TIMx_CTRL1.DIR`、`TIMx_EVTGEN.UDGN`) can only be changed by software (except `TIMx_EVTGEN.UDGN`, which remains cleared automatically). It is provided that the `TIMx_CTRL1.CNTEN` 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 T12 input, the configuration steps are as follows:

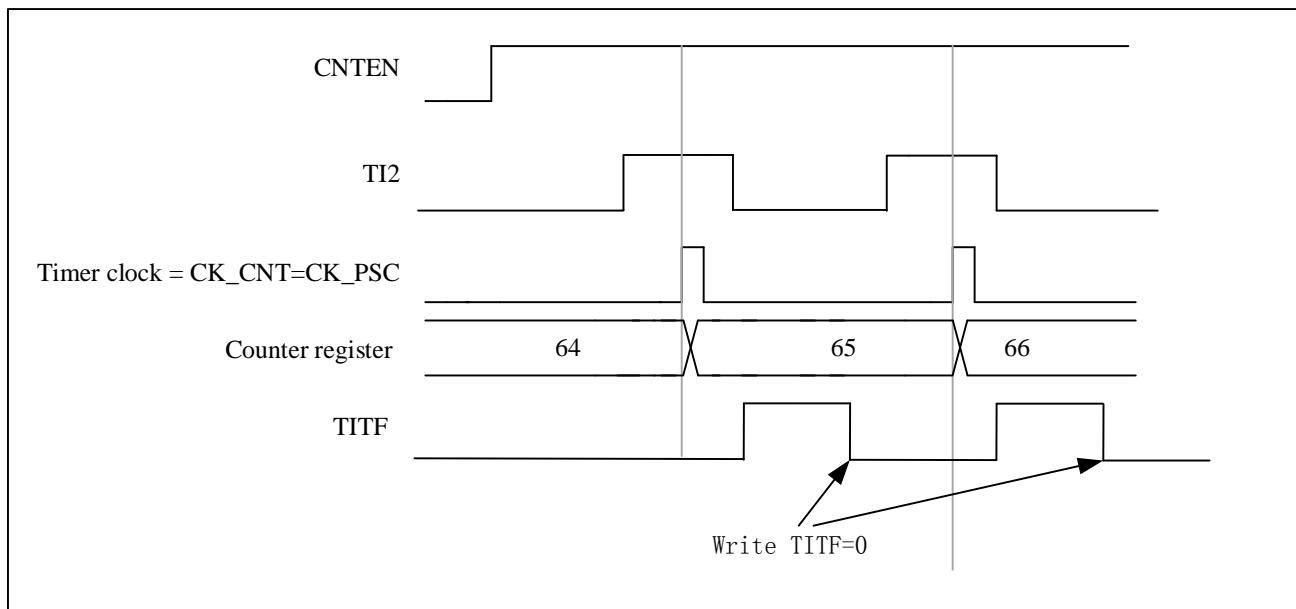
- Configure `TIMx_CCMOD1.CC2SEL` equal to '01', CC2 channel is configured as input, IC2 is mapped to TI2
- Configure `TIMx_CCEN.CC2P` equal to '0', select clock rising edge polarity
- To select input filter bandwidth by configuring `TIMx_CCMOD1.IC2F[3:0]` (if filter is not needed, keep IC2F bit at '0000')
- 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

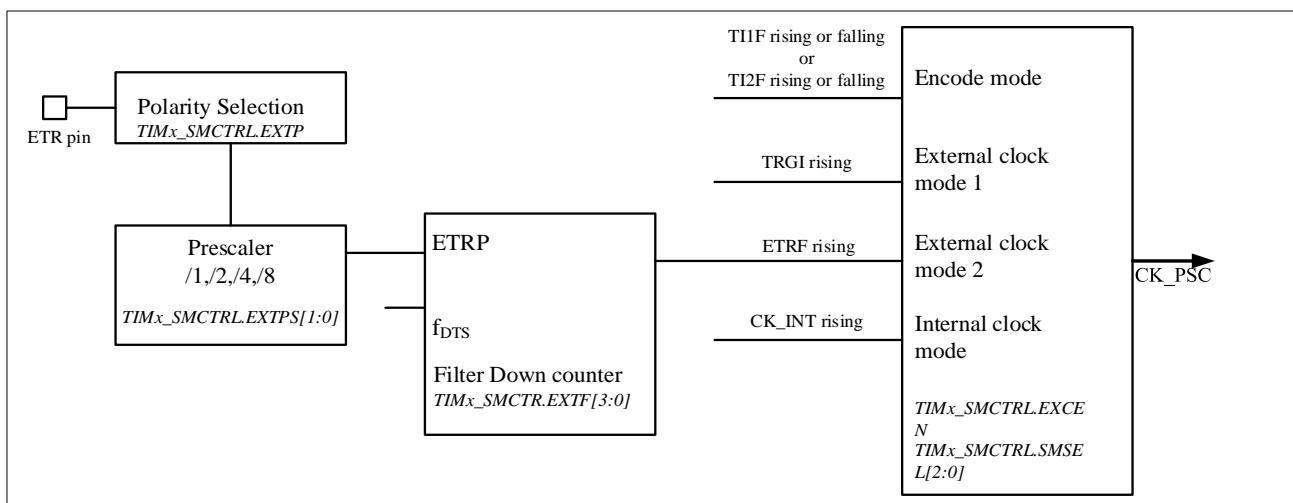


10.3.3.3 External clock source mode 2

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



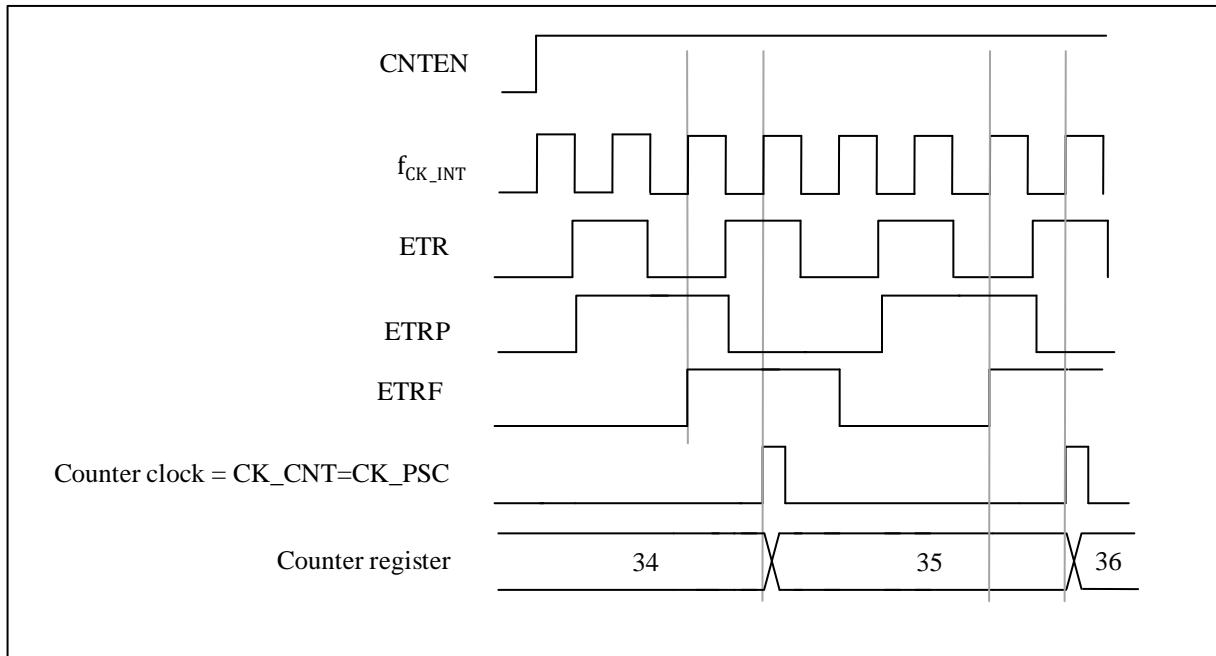
For example, use the following configuration steps to make the up counter count every 2 rising edges on ETR.

- 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

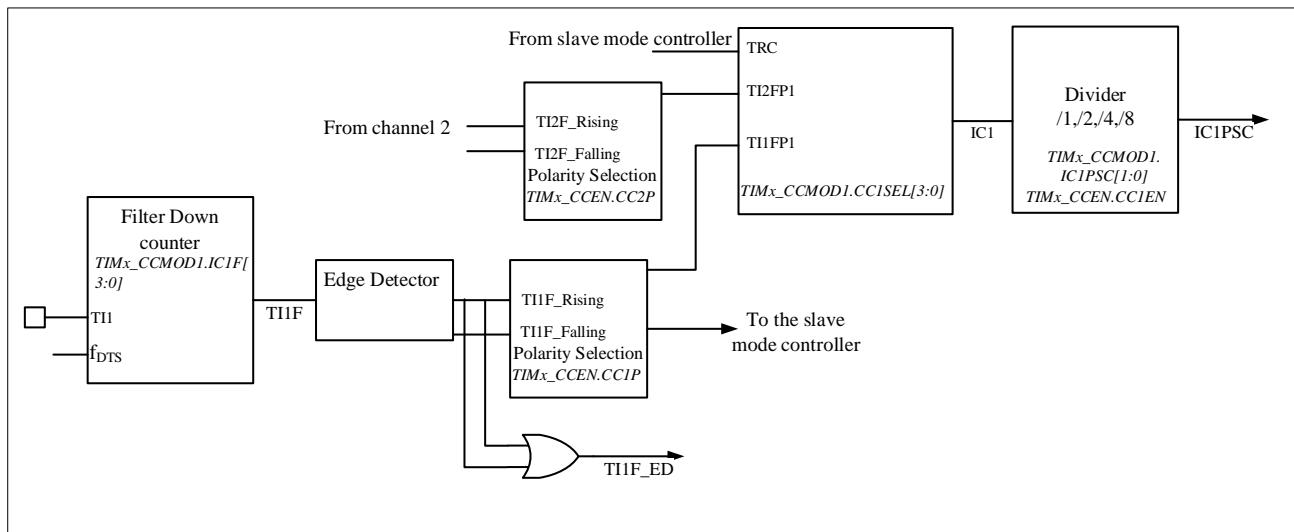


10.3.4 Capture/compare channels

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 TIx is sampled and filtered to generate the signal TIxF. A signal (TIxF_rising or TIxF_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 ICx 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

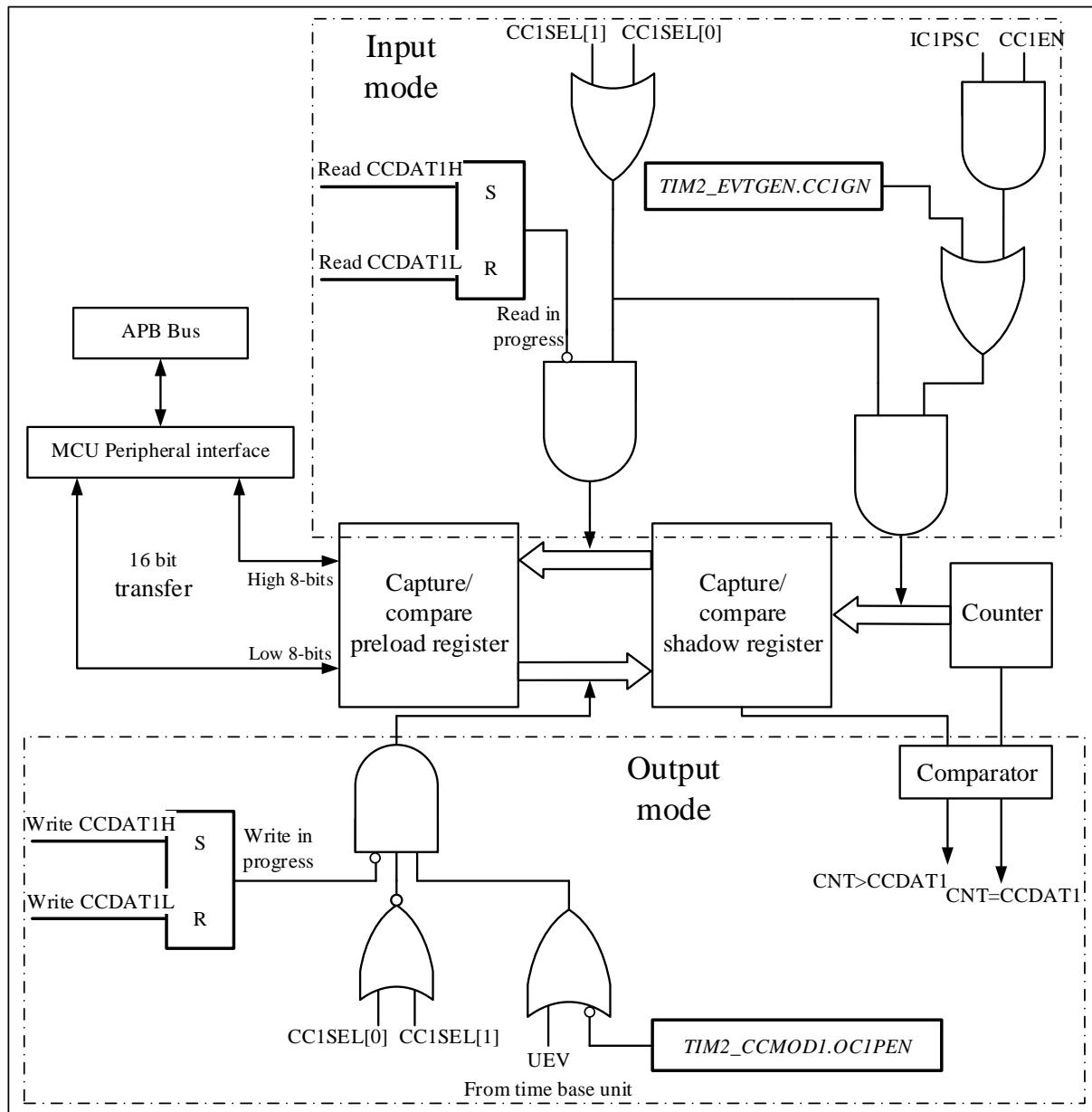
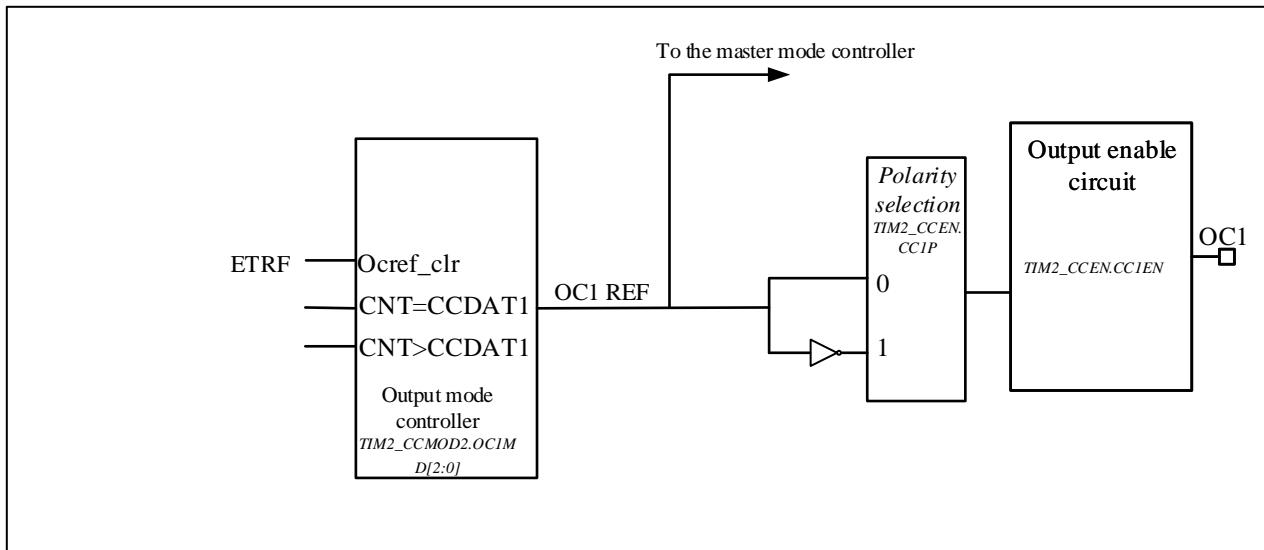


Figure 10-15 Output part of channelx (x=1/2/3/4. Take channel 1 as example)



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 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'.

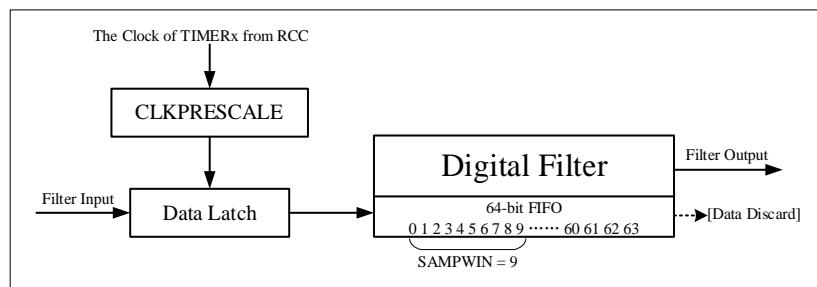
- By configuring TIMx_CCEN .CC1P=0, select the rising edge as the valid transition polarity on the TI1 channel.
- Configure the input prescaler. In this example, configure TIMx_CCMOD1.IC1PSC= '00' to disable the prescaler because we want to capture every valid transition.
- Enable capture by configuring 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

10.3.5.1 Channel input filter

Register TIMx_CxFILT(x = 1, 2, 3, 4) description are as follow:

Figure 10-16 Slide filter



- The digital filter samples channel input signal at the clock of TIMx(x = 2, 3, 4, 5) from RCC, accumulating samples in a 64-bits FIFO. Only sampled data within window size defined in TIMx_CxFILT.WSIZE [5:0] with maximum size 64.
- The filter outputs the majority value inside sample window which is defined by the threshold value in TIMx_CxFILT.THRESH [5:0] with maximum threshold of 63. This value should be equal or more than half of window size. If neither logic 1 nor logic 0 counts inside sampling window is more than threshold, digital filter maintain previous output value.
- TIMx_CxFILT.PSC register determines sample rate of corresponding digital filter. Filter FIFO capture one sample value from input at every sample clock.
- If digital filter is off, filter input will bypass to output like a wire.

10.3.6 PWM input mode

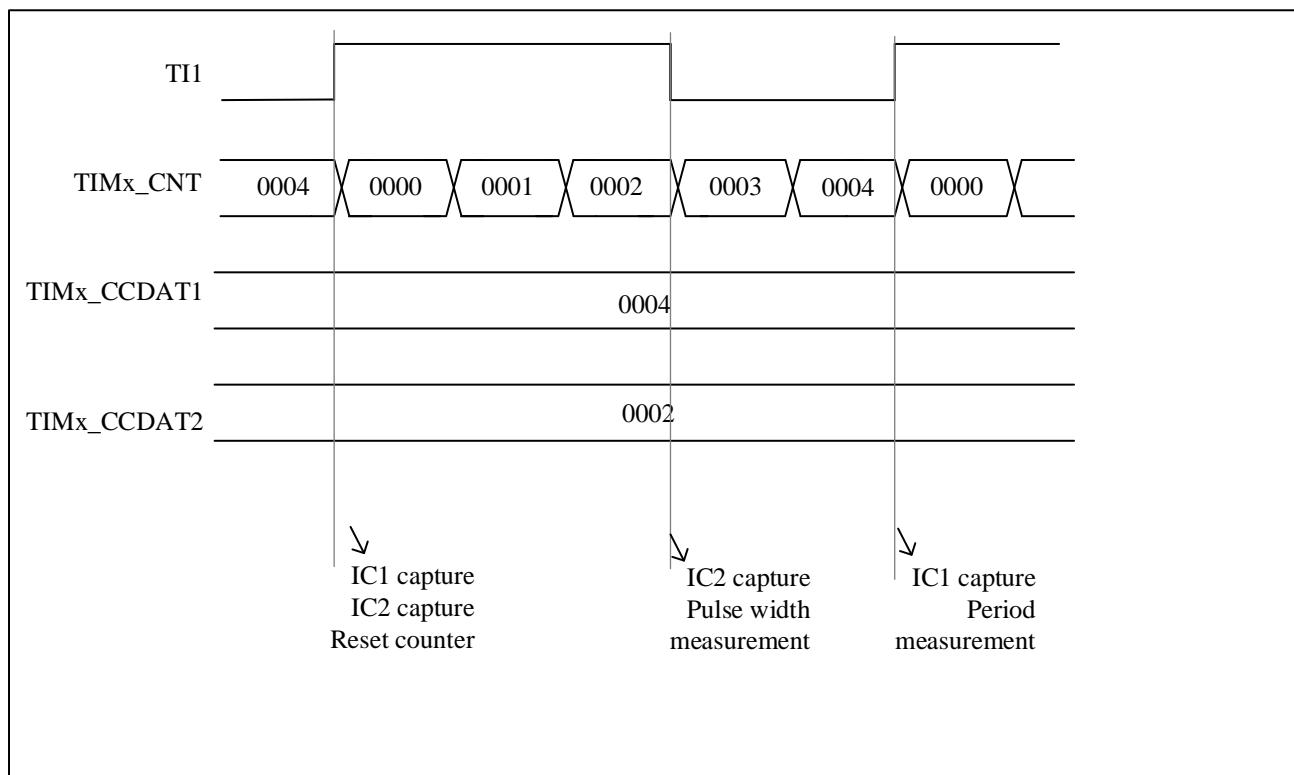
There are some differences between PWM input mode and normal input capture mode, including:

- 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 10-17 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_CCDATx 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_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.

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 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.

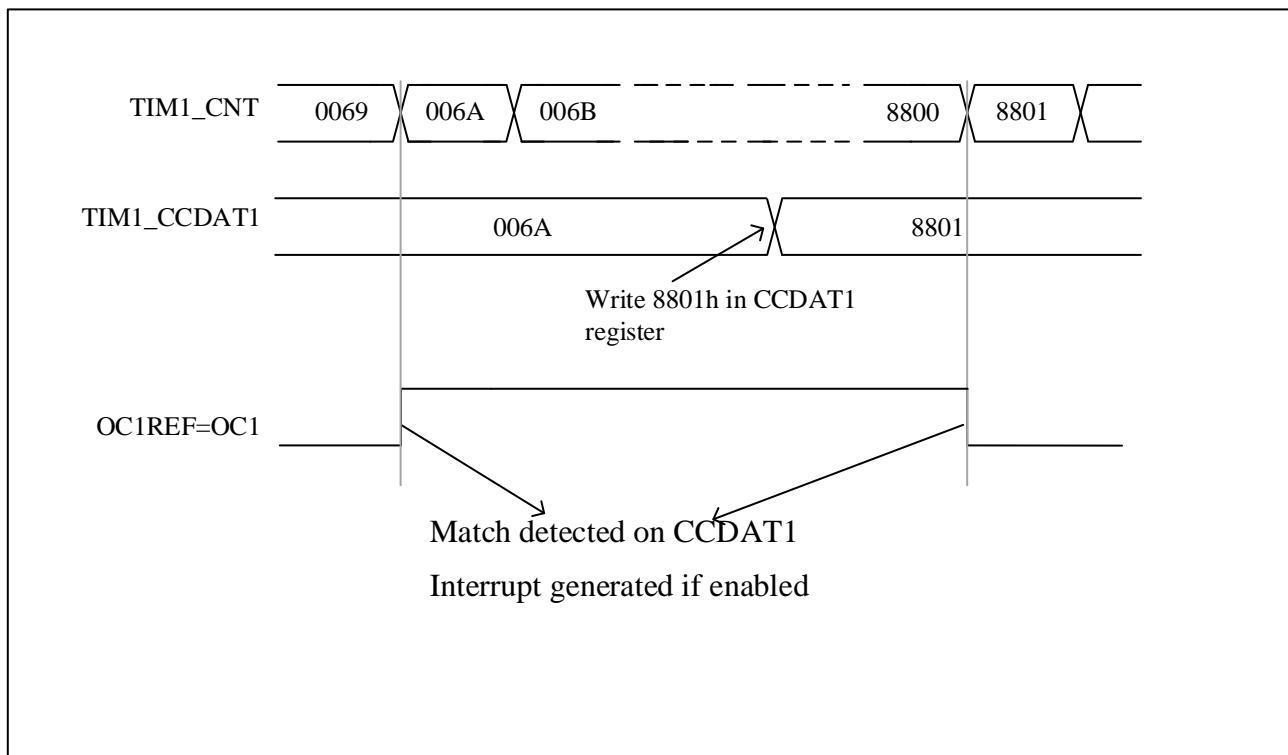
Here are the configuration steps for output compare mode:

- 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 10-18 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.

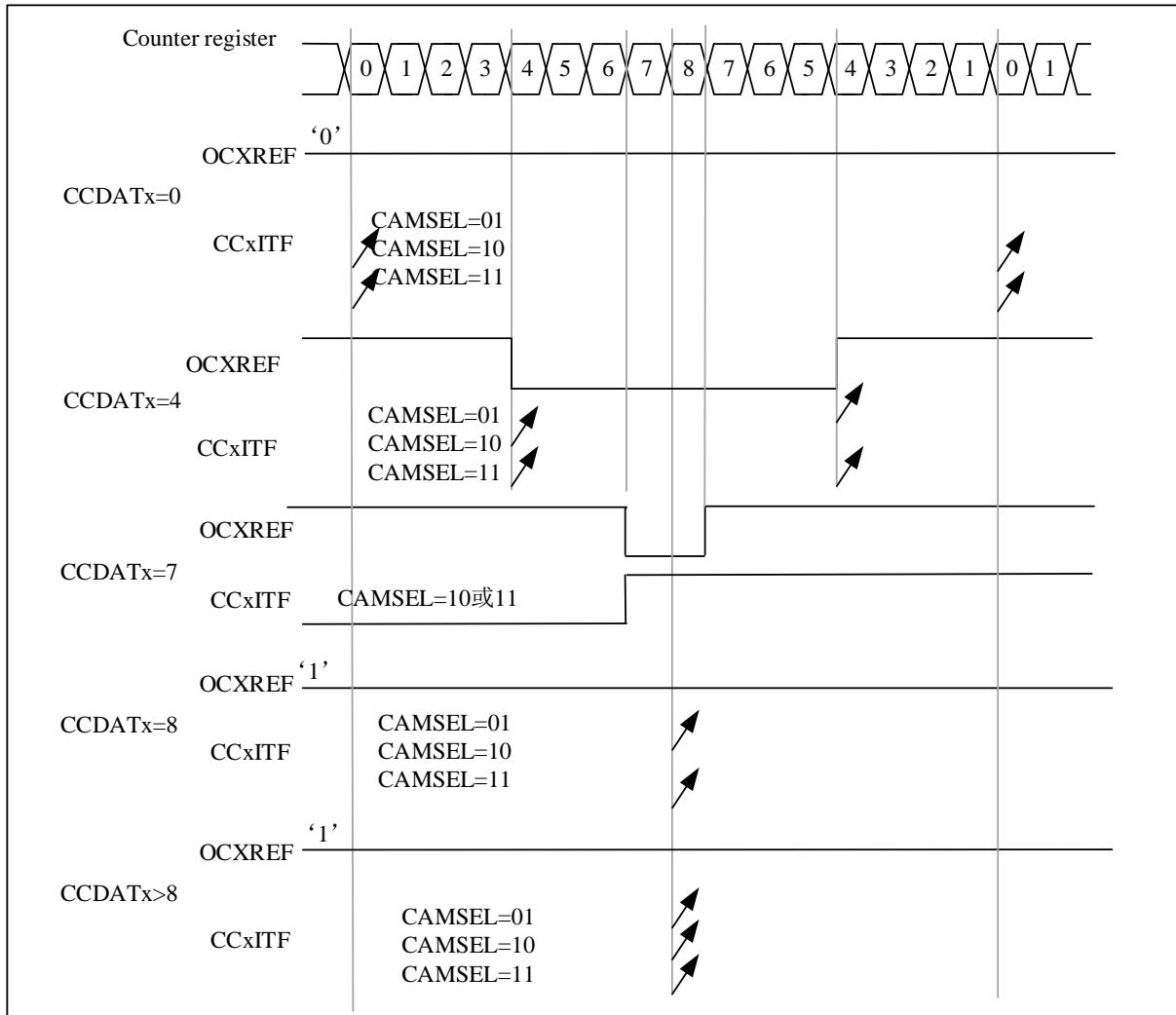
10.3.9.1 PWM center-aligned mode

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 10-19 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**

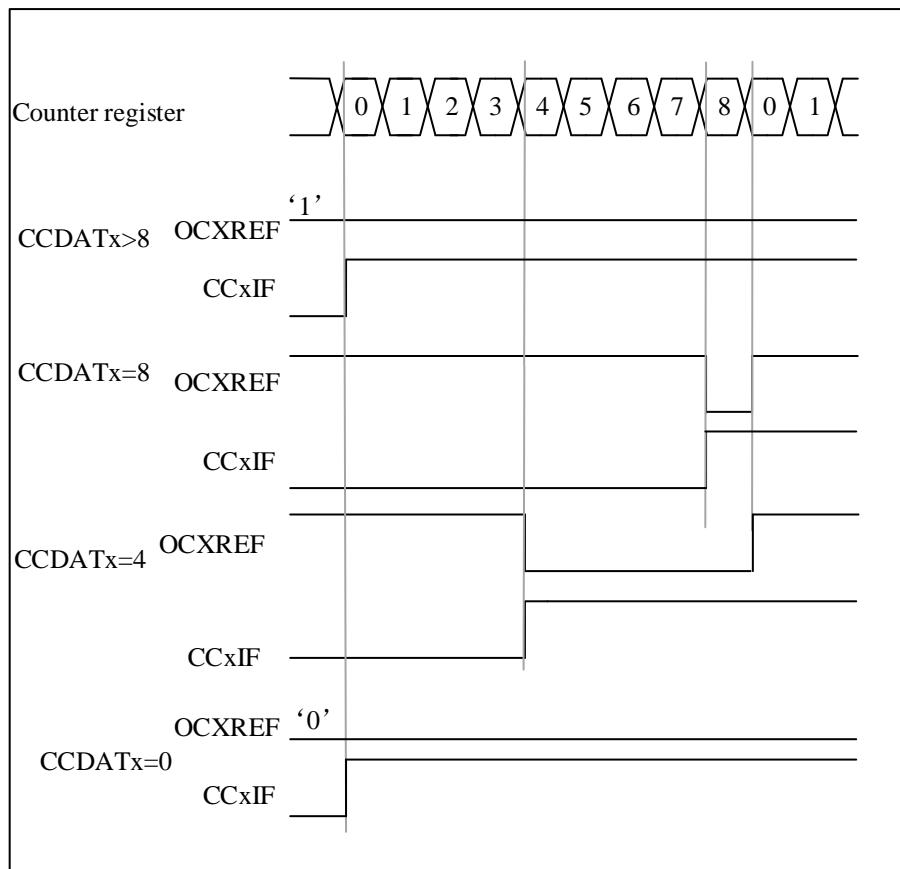
User can set `TIMx_CTRL1.DIR=0` to make counter counts up.

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

Figure 10-20 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.

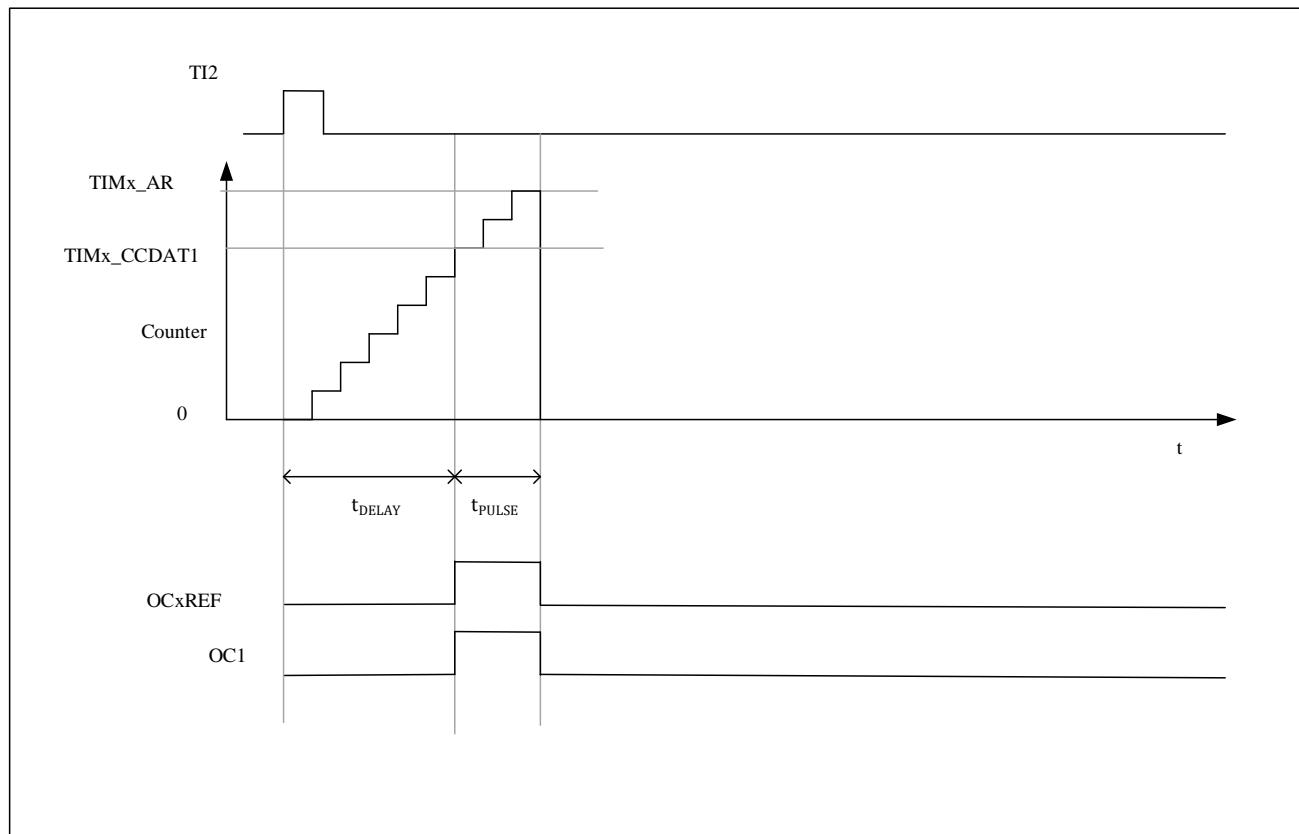
When $\text{TIMx_CNT} > \text{TIMx_CCDATx}$, the reference PWM signal `OCxREF` is low. Otherwise it will be high. If the compare value in `TIMx_CCDATx` is greater than the auto-reload value, the `OCxREF` will remains 1.

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 10-21 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

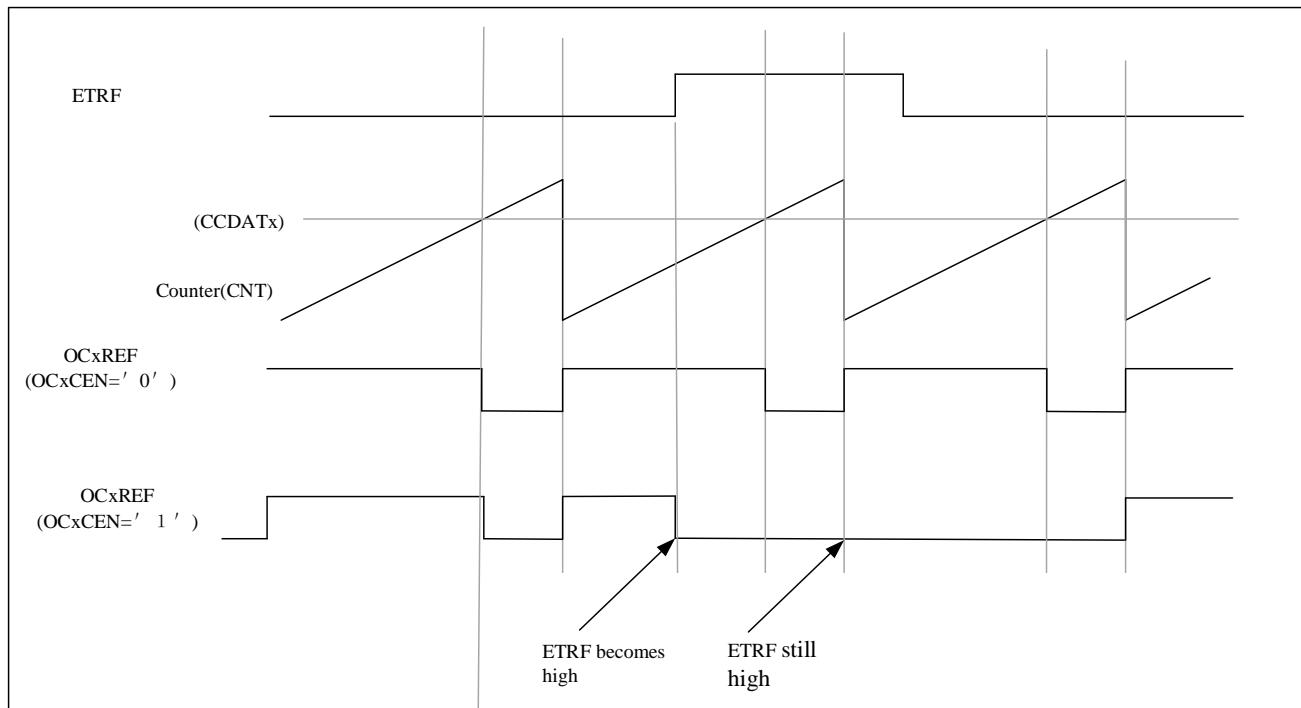
If user set `TIMx_CCMODx.OCxCEN=1`, high level of ETRF input can be used to driven the OCxREF signal to low, and the OCxREF signal will remains low, until the next UEV happens. Only output compare and PWM modes can use this function. This cannot be used when it is in forced mode.

Here is an example for it. To control the current, user can connect the ETR signal to external IO, 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 10-22 Control circuit in reset mode



10.3.12 Debug mode

When the microcontroller is in debug mode (the Cortex-M4 core halted), depending on the `DBG_TIMx_STOP` configuration in the DBG module, the TIMx counter can either continue to work normally or stop. For more details, see 21.4.3.

10.3.13 TIMx and external trigger synchronization

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

10.3.13.1 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:

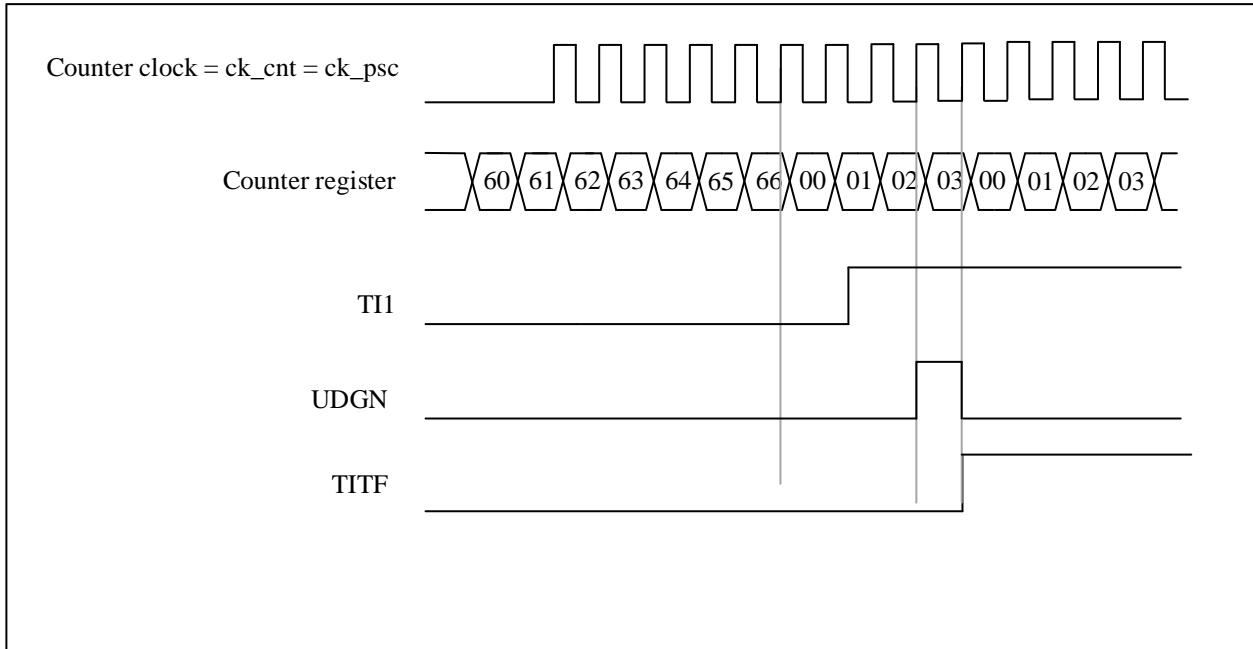
1. Channel 1 is configured as input to detect the rising edge of `TI1` (`TIMx_CCMOD1.CC1SEL=01`, `TIMx_CCEN.CC1P=0`);
2. The slave mode is selected as reset mode (`TIMx_SMCTRL.SMSEL=100`), and the trigger input is selected as `TI1` (`TIMx_SMCTRL.TSEL=101`);
3. Setting `TIMx_CTRL1.CNTEN = 1` to start counter

After starting the timer, when `TI1` detects a rising edge, the counter resets and restarts counting, and the trigger flag

is set (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 10-23 Control circuit in reset mode



10.3.13.2 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.

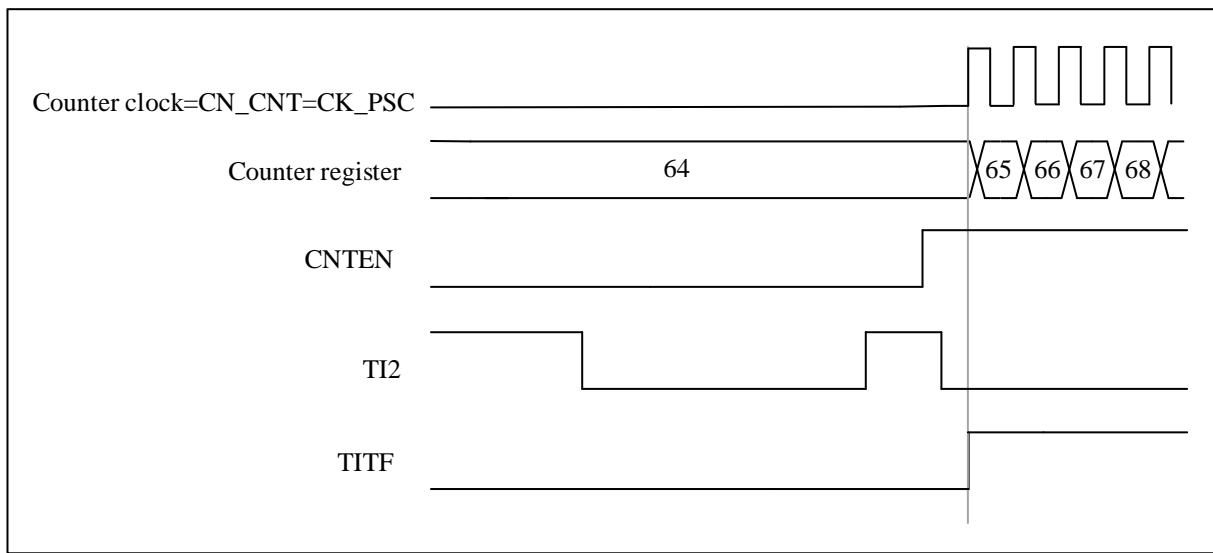
The following is an example of a trigger pattern:

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

When TI2 detects a rising edge, the counter starts counting, and the trigger flag is set (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 TI1 input.

Figure 10-24 Control circuit in Trigger mode



10.3.13.3 Slave mode: Gated mode

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

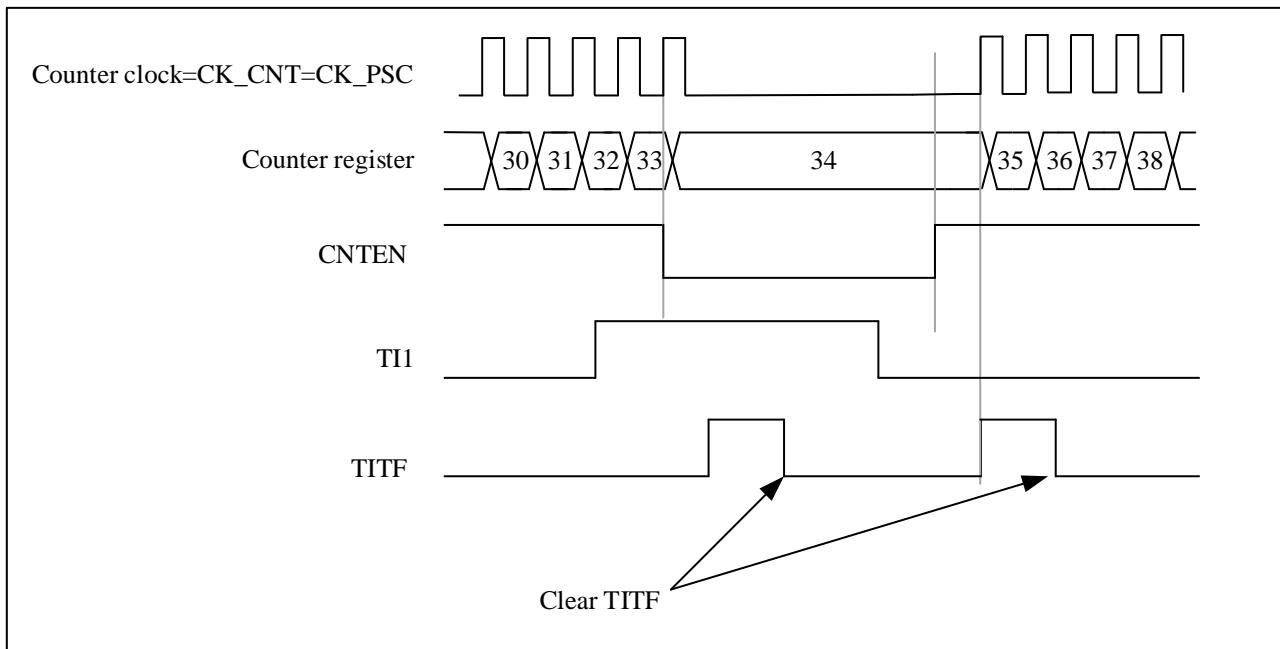
The following is an example of a gated pattern:

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 trigger mode (TIMx_SMCTRL.SMSEL=101), and select TI1 as the trigger input (TIMx_SMCTRL.TSEL=101);
3. Setting TIMx_CTRL1.CNTEN = 1 to start counter

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 10-25 Control circuit in Gated mode



10.3.13.4 Slave mode: Trigger Mode + External Clock Mode 2

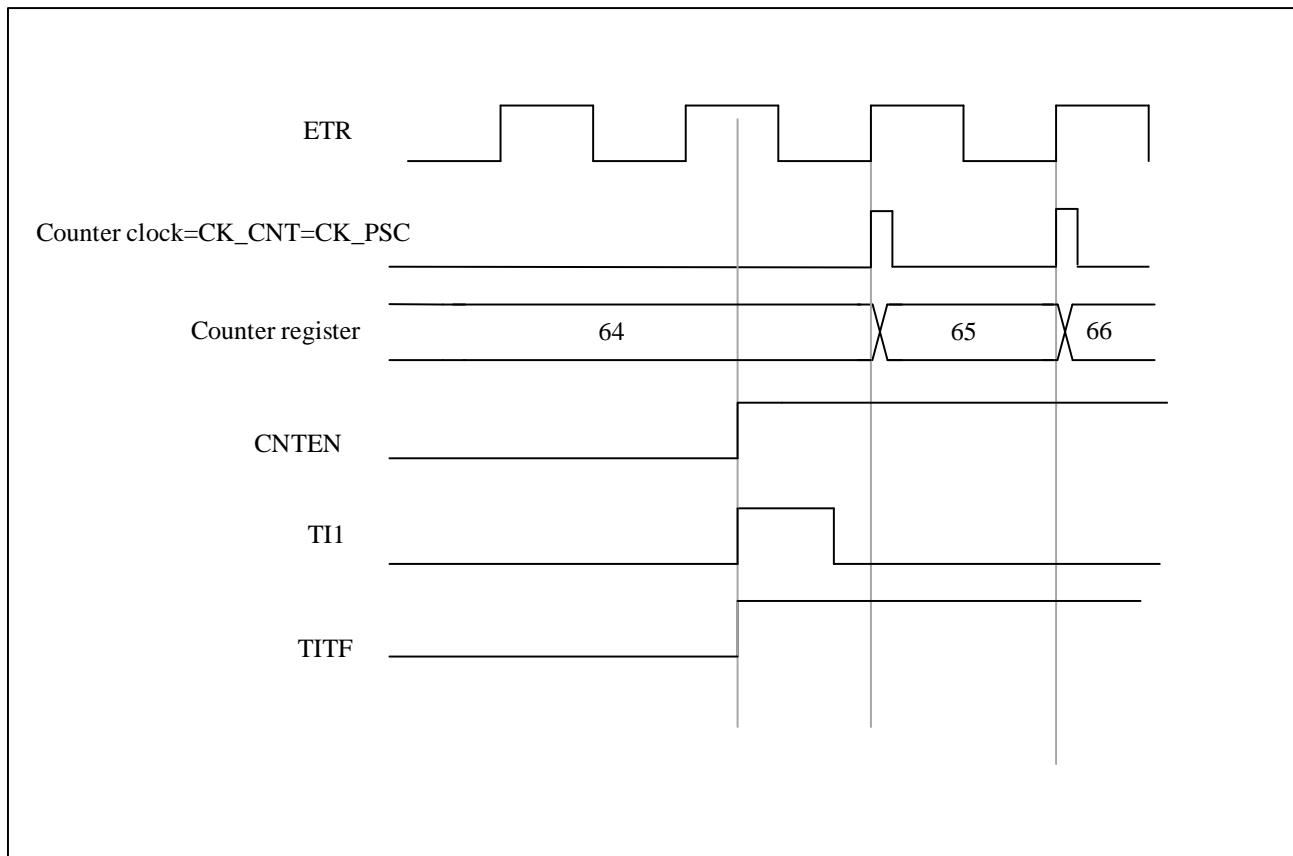
In reset mode, trigger mode and gate control mode, the counter clock can be selected as external clock mode 2, and the ETR signal is used as the external clock source input. At this time, the trigger selection needs to select non-ETRF (TIMx_SMCTRL.TSEL=111).

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 10-26 Control circuit in Trigger Mode + External Clock Mode2

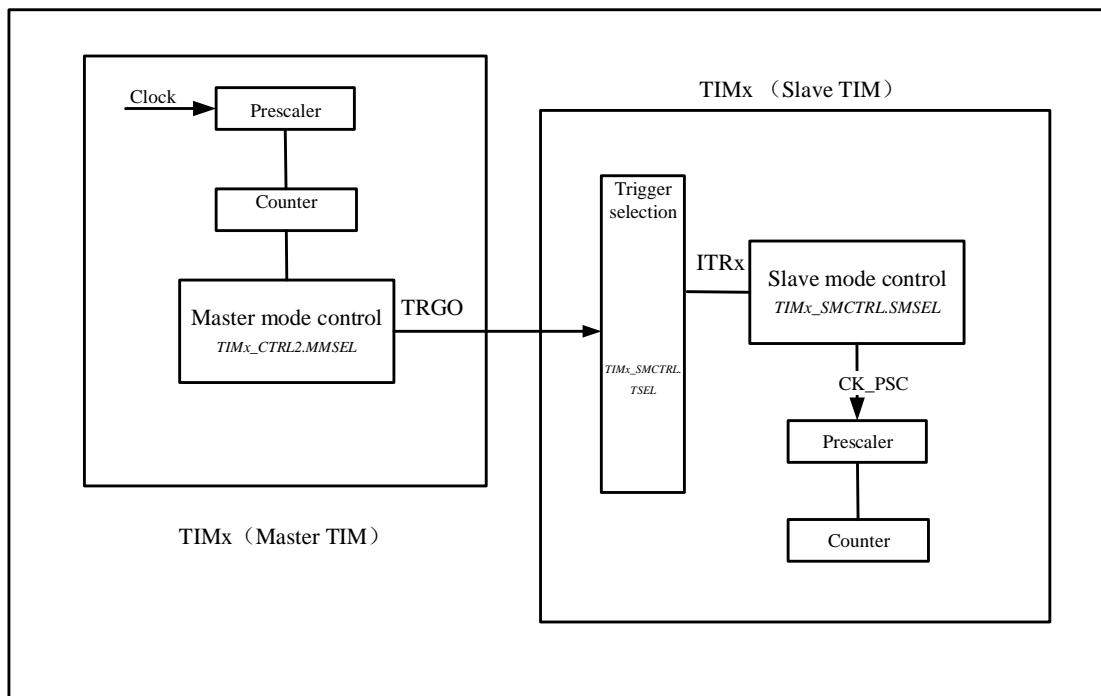


10.3.14 Timer synchronization

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 10-27 Block diagram of timer interconnection



10.3.14.1 Master timer as a prescaler for another timer

Timer 1 as a prescaler for Timer 2. TIM1 is master, TIM2 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 `TIM2_SMCTRL.TSEL='000'`, connect the `TRGO` of TIM1 to TIM2.
- Configure `TIM2_SMCTRL.SMSEL='111'`, the slave mode controller will be configured in external clock mode 1.
- Start TIM2 by setting `TIM2_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 timer2.

10.3.14.2 Master timer to enable another timer

In this example, TIM2 is enabled by the output compare of TIM1. TIM2 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 ($f_{CK_CNT} = f_{CK_INT}/3$).

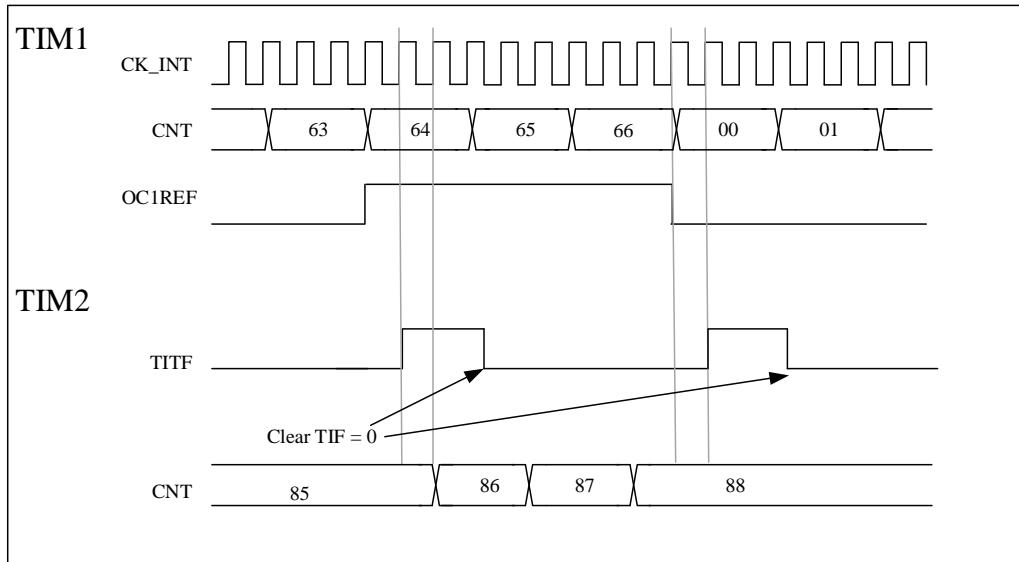
The configuration steps are shown as below.

- 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 `TIM2_SMCTRL.TSEL = '000'` to connect TIM1 trigger output to TIM2.
- Setting `TIM2_SMCTRL.SMSEL = '101'` to set TIM2 to gated mode.
- Setting `TIM2_CTRL1.CNTEN = '1'` to start TIM2.
- Setting `TIM1_CTRL1.CNTEN = '1'` to start TIM1.

Note: The TIM2 clock is not synchronized with the TIM1 clock, this mode only affects the TIM2 counter enable signal.

Figure 10-28 TIM2 gated by OC1REF of TIM1



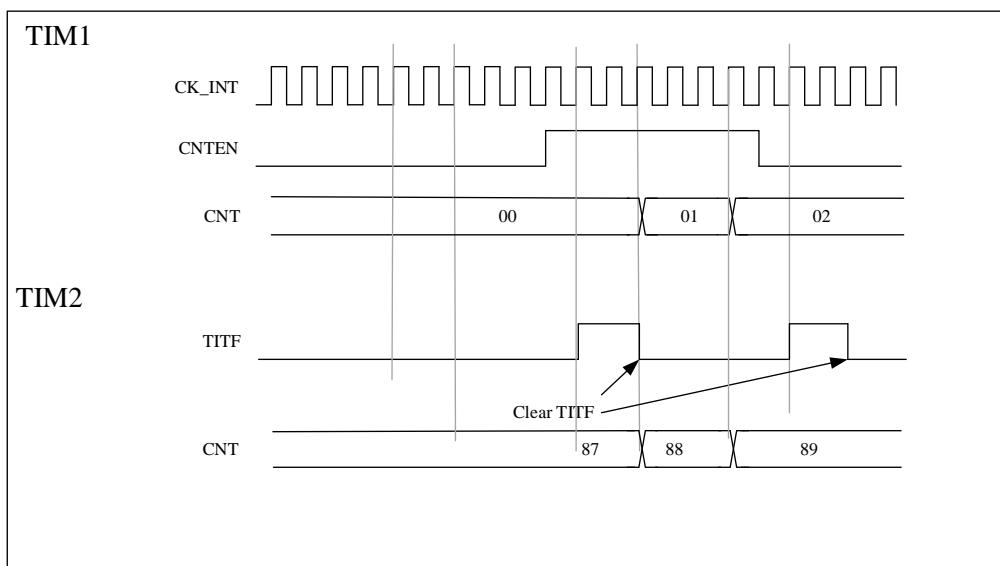
In the next example, Gated TIM2 with enable signal of TIM1, refer to Figure 10-29.

TIM2 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 `TIM2_SMCTRL.TSEL = '000'` to configure TIM2 to get the trigger input from TIM1
- Setting `TIM2_SMCTRL.SMSEL = '101'` to configure TIM2 in gated mode.
- Setting `TIM2_CTRL1.CNTEN = '1'` to start TIM2.
- Setting `TIM1_CTRL1.CNTEN = '1'` to start TIM1.
- Setting `TIM1_CTRL1.CNTEN = '0'` to stop TIM1.

Figure 10-29 TIM2 gated by enable signal of TIM1



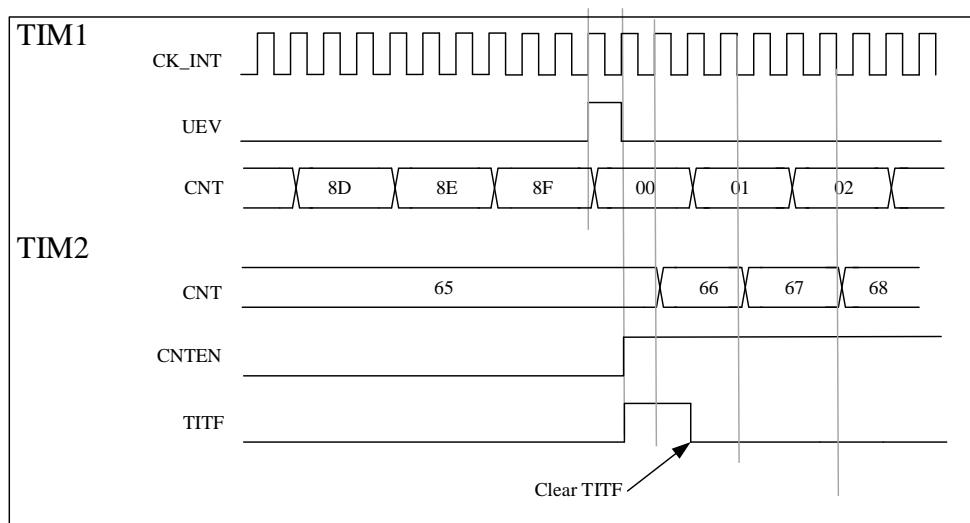
10.3.14.3 Master timer to start another timer

In this example, we can use update event as trigger source. TIM1 is master, TIM2 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 TIM2_SMCTRL.TSEL='000' to connect TIM1 trigger output to TIM2.
- Setting TIM2_SMCTRL.SMSEL='110' to set TIM2 to trigger mode.
- Setting TIM1_CTRL1.CNTEN=1 to start TIM1.

Figure 10-30 Trigger TIM2 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 TIM2 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 TIM2, TIM1 is the master.

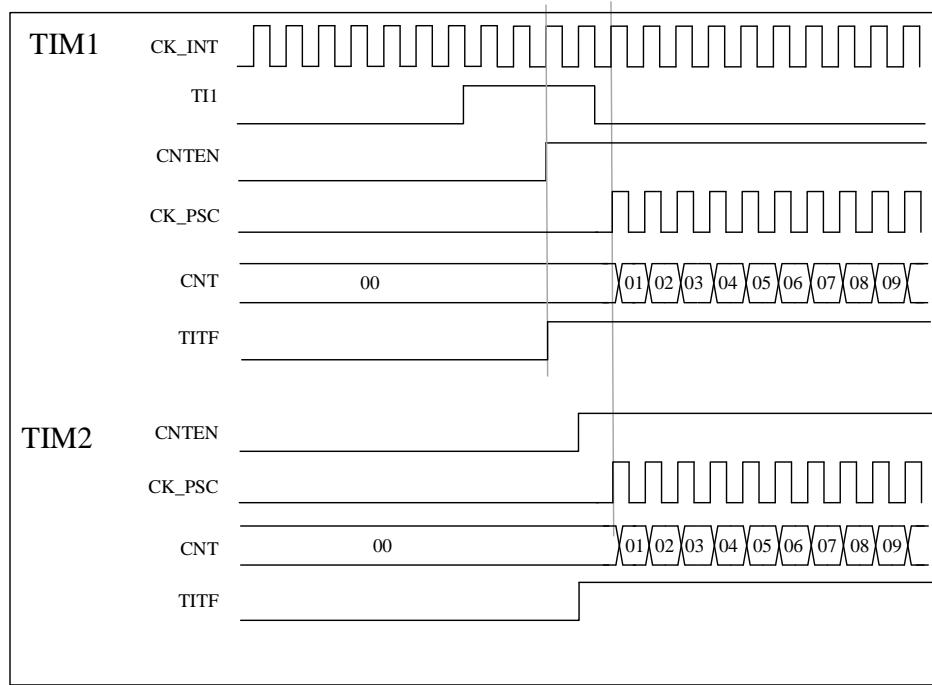
The configuration steps are shown as below:

- 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 TIM2_SMCTRL.TSEL = '000' to connect TIM1 trigger output to TIM2.
- Setting TIM2_SMCTRL.SMSEL = '110' to configure TIM2 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 10-31 Triggers timers 1 and 2 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:

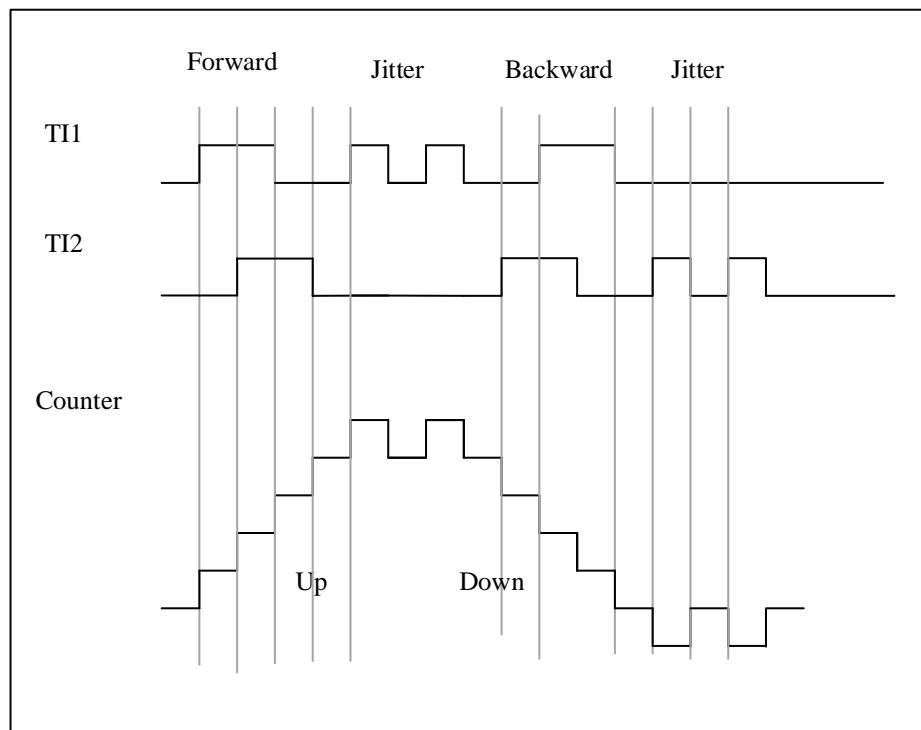
Table 10-1 Counting direction versus encoder signals

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:

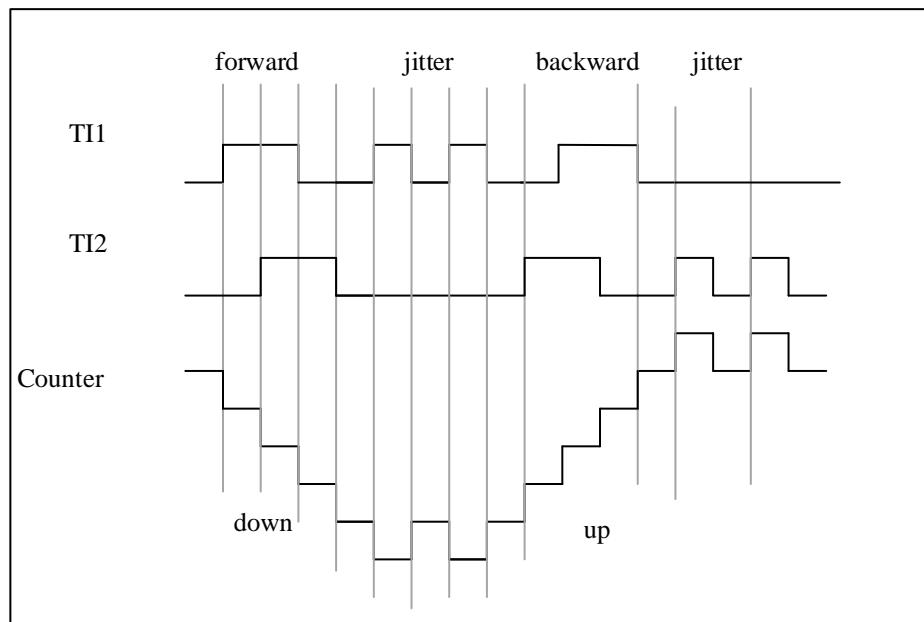
1. IC1FP1 is mapped to TI1 (TIMx_CCMOD1.CC1SEL = '01'), IC1FP1 is not inverted (TIMx_CCEN.CC1P = '0');
2. IC1FP2 is mapped to TI2 (TIMx_CCMOD2.CC2SEL = '01'), IC2FP2 is not inverted (TIMx_CCEN.CC2P = '0');
3. The input is valid on both rising and falling edges (TIMx_SMCTRL.SMSEL = '011');
4. Enable counter TIMx_CTRL1.CNTEN = '1';

Figure 10-32 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 10-33 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=2, 3, 4 and 5)

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).

When configuring the registers, please refer to the block diagram of the corresponding TIM, the channels and functions of different TIM will be different.

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
000h	TIMx_CTRL1															CLRSEL	15
		Reset Value														CASEL	14
004h	TIMx_CTRL2															C3SEL	13
		Reset Value														C2SEL	12
008h	TIMx_SMCTRL															C1SEL	11
		Reset Value														Reserved	10
00Ch	TIMx_DINTEN															0	0
		Reset Value														CLKD1[1:0]	8
010h	TIMx_STS															ETRSEL	0
		Reset Value														ARPEN	7
014h	TIMx_EVTGEN															0	0
		Reset Value														CAMSEL[1:0]	6
018h	TIMx_CCMOD1 Output comparison mode															MMSSEL[2:0]	5
		Reset Value														DIR	4
01Ch	TIMx_CCMOD1 Input capture mode															CCDSEL	0
		Reset Value														ONEPM	3
01Ch	TIMx_CCMOD2 Output comparison mode															UPRS	2
		Reset Value														UPDIS	1
01Ch	TIMx_CCMOD2 Input capture mode															CNTEN	0

10.4.2 Control register 1 (TIMx_CTRL1)

Offset address: 0x00

Reset value: 0x0000 0000

31	Reserved															16
15	CLRSEL	C4SEL	C3SEL	C2SEL	C1SEL	Reserved	CLKD[1:0]	ARPEN	CAMSEL[1:0]	DIR	ONEPM	UPRS	UPDIS	CNTEN		0
	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	CLRSEL	OCxREF clear selection 0: Select the external OCxREF clear from ETR 1: Reserved
14	C4SEL	Channel 4 Selection 0: Select external CH4 (from IOM) signal 1: Reserved <i>Note: This is not for TIM2. For TIM3, TIM4 and TIM5, it is always from IOM.</i>
13	C3SEL	Channel 3 Selection 0: Select external CH3 (from IOM) signal 1: Select internal CH3 (from LSI) signal <i>Note: This is only for TIM2. For TIM4 and TIM5, it is always from IOM.</i>
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: Reserved
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
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.

Bit field	Name	Description
		<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"> – Counter overflow/underflow – The TIMx_EVTGEN.UDGN bit is set – Update generation from the slave mode controller <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"> – Counter overflow/underflow – The TIMx_EVTGEN.UDGN bit is set – Update generation from the slave mode controller <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> <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>

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

Bit field	Name	Description
15:9	Reserved	Reserved, the reset value must be maintained
8	ETRSEL	<p>External Triggered Selection storage (ETR Selection)</p> <p>0: Select external ETR (from IOM) signal; 1: Reserved</p> <p><i>Note: Not for TIM3, TIM4 and TIM5, that is, the ETR of TIM5 can not connect to external input.</i></p>
7	TI1SEL	<p>TI1 selection</p> <p>0: TIMx_CH1 pin connected to TI1 input.</p> <p>1: TIMx_CH1, TIMx_CH2, and TIMx_CH3 pins are XOR connected to the TI1 input.</p> <p><i>Note: Not for TIM3</i></p>
6:4	MMSEL[2:0]	<p>Master Mode Selection</p> <p>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:</p> <p>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.</p> <p>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.</p> <p>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).</p> <p>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.</p> <p>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.</p> <p>100: Compare - OC1REF signal is used as the trigger output (TRGO).</p> <p>101: Compare - OC2REF signal is used as the trigger output (TRGO).</p> <p>110: Compare - OC3REF signal is used as the trigger output (TRGO).</p> <p>111: Compare - OC4REF signal is used as the trigger output (TRGO).</p>
3	CCDSEL	<p>Capture/compare DMA selection</p> <p>0: When a CCx event occurs, a DMA request for CCx is sent.</p> <p>1: When an update event occurs, a DMA request for CCx is sent.</p>
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	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

Bit field	Name	Description
15	EXTP	<p>External trigger polarity</p> <p>This bit is used to select whether the trigger operation is to use ETR or the inversion of ETR.</p> <p>0: ETR active at high level or rising edge.</p> <p>1: ETR active at low level or falling edge.</p>
14	EXCEN	<p>External clock enable</p> <p>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.</p> <p>0: External clock mode 2 disable.</p> <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 (TIMx_SMCTRL.TSEL ≠ '111').</i></p> <p><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></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.</p> <p>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_{DTS}</p> <p>0001: f_{SAMPLING} = f_{CLOCK}, N = 2</p> <p>0010: f_{SAMPLING} = f_{CLOCK}, N = 4</p> <p>0011: f_{SAMPLING} = f_{CLOCK}, N = 8</p> <p>0100: f_{SAMPLING} = f_{DTS}/2, N = 6</p>

Bit field	Name	Description
		0101: $f_{\text{SAMPLING}} = f_{\text{DTS}}/2$, $N = 8$ 0110: $f_{\text{SAMPLING}} = f_{\text{DTS}}/4$, $N = 6$ 0111: $f_{\text{SAMPLING}} = f_{\text{DTS}}/4$, $N = 8$ 1000: $f_{\text{SAMPLING}} = f_{\text{DTS}}/8$, $N = 6$ 1001: $f_{\text{SAMPLING}} = f_{\text{DTS}}/8$, $N = 8$ 1010: $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$, $N = 5$ 1011: $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$, $N = 6$ 1100: $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$, $N = 8$ 1101: $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, $N = 5$ 1110: $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, $N = 6$ 1111: $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, $N = 8$
7	MSMD	Master/ Slave mode 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.
6:4	TSEL[2:0]	Trigger selection These 3 bits are used to select the trigger input of the synchronous counter. 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) For more details on ITRx, see Table 10-3 below. <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>
3	Reserved	Reserved, the reset value must be maintained
2:0	SMSEL[2:0]	Slave mode selection 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) 000: Disable slave mode. If TIMx_CTRL1.CNTEN = 1, the prescaler is driven directly by the internal clock. 001: Encoder mode 1. According to the level of TI2FP2, the counter up-counting or down-counting on the edge of TI1FP1. 010: Encoder mode 2. According to the level of TI1FP1, the counter up-counting or down-counting on the edge of TI2FP2. 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. 100: Reset mode. On the rising edge of the selected trigger input (TRGI), the counter is reinitialized and the shadow register is updated. 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.

Bit field	Name	Description
		<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 TIIF_ED is selected as the trigger input (TIMx_SMCTRL.TSEL=100). This is because TIIF_ED outputs a pulse for each TIIF 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)
TIM2	TIM1	TIM8	TIM3	TIM4
TIM3	TIM1	TIM2	TIM5	TIM4
TIM4	TIM1	TIM2	TIM3	TIM8
TIM5	TIM2	TIM3	TIM4	TIM8

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
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	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	Reserved	Reserved, the reset value must be maintained
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

Bit field	Name	Description
		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	Reserved	13	CC4OCF	CC3OCF	CC2OCF	CC1OCF	8	Reserved	7	TITF	6	Reserved	5	CC4ITF	4	CC3ITF	3	CC2ITF	2	CC1ITF	1	UDITF	0
			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.

Bit field	Name	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 When the corresponding channel of CC1 is in output mode: 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. When the corresponding channel of CC1 is in input mode: This bit is set by hardware when the capture event occurs. This bit is cleared by software or by reading TIMx_CCDAT1. 0: No input capture occurred. 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.
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, overflow or underflow (An update event is generated).

Bit field	Name	Description
		<ul style="list-style-type: none"> When TIMx_CTRL1.UPRS = 0, TIMx_CTRL1.UPDIS = 0, and set the TIMx_EVTGEN.UDGN bit by software to reinitialize the CNT. When TIMx_CTRL1.UPRS = 0, TIMx_CTRL1.UPDIS = 0, and the counter CNT is reinitialized by the trigger event. (See TIMx_SMCTRL Register description) <p>This bit is cleared by software.</p> <p>0: No update event occurred 1: Update interrupt occurred</p>

10.4.7 Event generation registers (TIMx_EVTGEN)

Offset address: 0x14

Reset values: 0x0000

15	Reserved	7	6	5	4	3	2	1	0
		w		w	w	w	w	w	w

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 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>
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>

Bit field	Name	Description
		0: No action 1: Generated a CC1 capture/compare event
0	UDGN	Update generation 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. 0: No action 1: Generated an update event

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	rw	rw	rw	rw		rw	rw	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
9:8	CC2SEL[1:0]	Capture/compare 2 selection These bits are used to select the input/output and input mapping of the channel 00: CC2 channel is configured as output 01: CC2 channel is configured as input, IC2 is mapped on TI2 10: CC2 channel is configured as input, IC2 is mapped on TI1 11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is only active when the internal trigger input is selected by TIMx_SMCTRL.TSEL. <i>Note: CC2SEL is writable only when the channel is off (TIMx_CCEN.CC2EN = 0).</i>
7	OC1CEN	Output Compare 1 clear enable 0: OC1REF is not affected by ETRF input level 1: OC1REF is cleared immediately when the ETRF input level is detected as high

Bit field	Name	Description
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 < TIMx_CCDAT1, OC1REF signal of channel 1 is high, otherwise it is low. In down-counting mode, if TIMx_CNT > 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 < TIMx_CCDAT1, OC1REF signal of channel 1 is low, otherwise it is high. In down-counting mode, if TIMx_CNT > 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>
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>

Bit field	Name	Description
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]	12	IC2PSC[1:0]	11	CC2SEL[1:0]	10		9	8	7	IC1F[3:0]	4	3	2	1	0
		rw		rw		rw				rw		rw		rw		rw

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_{DTS} frequency</p> <p>0001: f_{SAMPLING} = f_{CLOCK}, N = 2</p> <p>0010: f_{SAMPLING} = f_{CLOCK}, N = 4</p> <p>0011: f_{SAMPLING} = f_{CLOCK}, N = 8</p> <p>0100: f_{SAMPLING} = f_{DTS}/2, N = 6</p> <p>0101: f_{SAMPLING} = f_{DTS}/2, N = 8</p> <p>0110: f_{SAMPLING} = f_{DTS}/4, N = 6</p> <p>0111: f_{SAMPLING} = f_{DTS}/4, N = 8</p> <p>1000: f_{SAMPLING} = f_{DTS}/8, N = 6</p> <p>1001: f_{SAMPLING} = f_{DTS}/8, N = 8</p> <p>1010: f_{SAMPLING} = f_{DTS}/16, N = 5</p> <p>1011: f_{SAMPLING} = f_{DTS}/16, N = 6</p> <p>1100: f_{SAMPLING} = f_{DTS}/16, N = 8</p>

Bit field	Name	Description
		1101: $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, $N = 5$ 1110: $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, $N = 6$ 1111: $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, $N = 8$
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 $\text{TIMx_CCEN.CC1EN} = 0$, 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		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]	Capture/Compare 4 selection These bits are used to select the input/output and input mapping of the channel 00: CC4 channel is configured as output 01: CC4 channel is configured as input, IC4 is mapped on TI4 10: CC4 channel is configured as input, IC4 is mapped on TI3 11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is only active when the internal trigger input is selected by TIMx_SMCTRL.TSEL .

Bit field	Name	Description
		<i>Note: CC4SEL is writable only when the channel is off (TIMx_CCEN.CC4EN = 0).</i>
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	12	11	10	9	8	7	4	3	2	1	0
IC4F[3:0]		IC4PSC[1:0]		CC4SEL[1:0]			IC3F[3:0]		IC3PSC[1:0]		CC3SEL[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>

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	CC1P	CC1EN

rw rw 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: 0: OC1 active high 1: OC1 active low 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. 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. <i>Note: If TIMx_BKDT.LCKCFG = 3 or 2, these bits cannot be modified.</i>

Bit field	Name	Description
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.</p> <p>1: Enable - Enable output OC1 signal.</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</p> <p>1: Enable capture</p>

Table 10-4 Output control bits of standard OCx channel

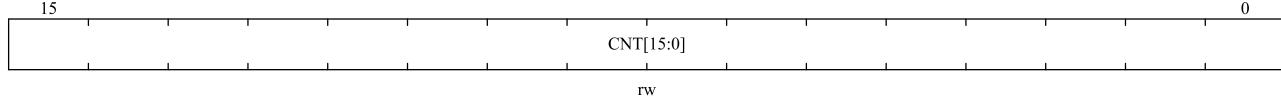
CCxEN	OCx output status
0	Disable output (OCx=0)
1	OCx = OCxREF + polarity

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

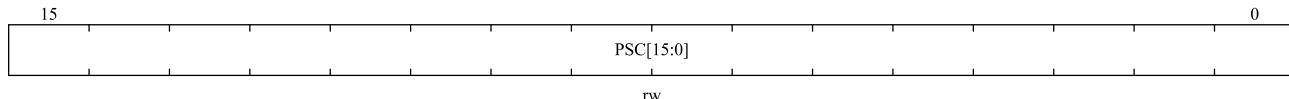


Bit field	Name	Description
15:0	CNT[15:0]	Counter value

10.4.12 Prescaler (TIMx_PSC)

Offset address: 0x28

Reset value: 0x0000

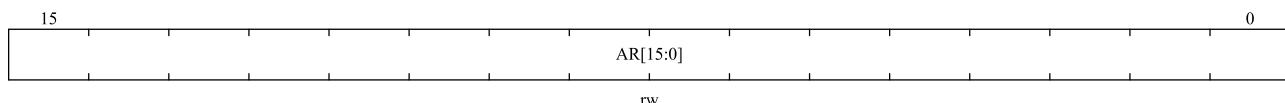


Bit field	Name	Description
15:0	PSC[15:0]	<p>Prescaler value</p> <p>Counter clock $f_{CK_CNT} = f_{CK_PSC} / (PSC [15:0] + 1)$.</p> <p>Each time an update event occurs, the PSC value is loaded into the active prescaler register.</p>

10.4.13 Auto-reload register (TIMx_AR)

Offset address: 0x2C

Reset values: 0xFFFF

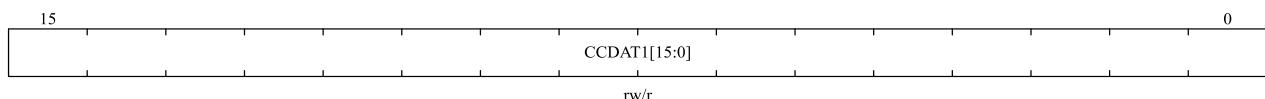


Bit field	Name	Description
15:0	AR[15:0]	<p>Auto-reload value</p> <p>These bits define the value that will be loaded into the actual auto-reload register.</p> <p>See Section 10.3 for more details.</p> <p>When the TIMx_AR.AR [15:0] value is null, the counter does not work.</p>

10.4.14 Capture/compare register 1 (TIMx_CCDAT1)

Offset address: 0x34

Reset value: 0x0000

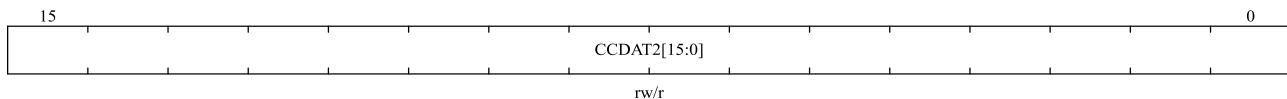


Bit field	Name	Description
15:0	CCDAT1[15:0]	<p>Capture/Compare 1 value</p> <ul style="list-style-type: none"> CC1 channel is configured as output: CCDAT1 contains the value to be compared to the counter TIMx_CNT, signaling on the OC1 output. If the preload feature is not selected in TIMx_CCMOD1.OC1PEN 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. CC1 channel is configured as input: CCDAT1 contains the counter value transferred by the last input capture 1 event (IC1). When configured as input mode, register CCDAT1 is only readable. When configured as output mode, register CCDAT1 is readable and writable.

10.4.15 Capture/compare register 2 (TIMx_CC DAT2)

Offset address: 0x38

Reset value: 0x0000

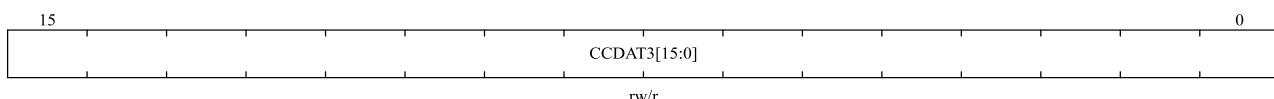


Bit field	Name	Description
15:0	CC DAT2[15:0]	<p>Capture/Compare 2 values</p> <ul style="list-style-type: none"> ■ CC2 channel is configured as output: CC DAT2 contains the value to be compared to the counter TIMx_CNT, signaling on the OC2 output. If the preload feature is not selected in TIMx_CCMOD1.OC2PEN 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. ■ CC2 channel is configured as input: CC DAT2 contains the counter value transferred by the last input capture 2 event (IC2). When configured as input mode, register CC DAT2 is only readable. When configured as output mode, register CC DAT2 is readable and writable.

10.4.16 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"> ■ CC3 channel is configured as output: CC DAT3 contains the value to be compared to the counter TIMx_CNT, signaling on the OC3 output. 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. ■ CC3 channel is configured as input: CC DAT3 contains the counter value transferred by the last input capture 3 event (IC3). When configured as input mode, register CC DAT3 is only readable. When configured as output mode, register CC DAT3 is readable and writable.

10.4.17 Capture/compare register 4 (TIMx_CCData4)

Offset address: 0x40

Reset value: 0x0000

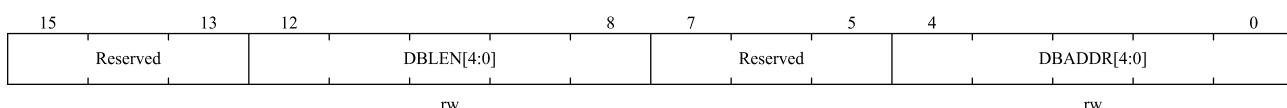


Bit field	Name	Description
15:0	CCData4[15:0]	<p>Capture/Compare 4 value</p> <ul style="list-style-type: none"> ■ CC4 channel is configured as output: CCData4 contains the value to be compared to the counter TIMx_CNT, signaling on the OC4 output. If the preload feature is not selected in TIMx_CCMOD2.OC4PEN 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. ■ CC4 channel is configured as input: CCData4 contains the counter value transferred by the last input capture 4 event (IC4). When configured as input mode, register CCData4 is only readable. When configured as output mode, register CCData4 is readable and writable.

10.4.18 DMA Control register (TIMx_DCTRL)

Offset address: 0x48

Reset value: 0x0000



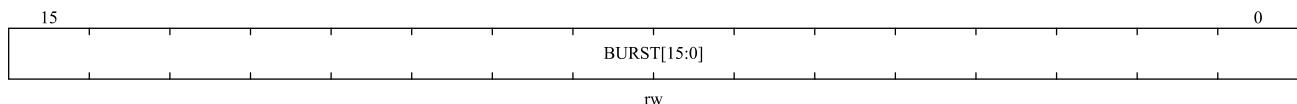
Bit field	Name	Description
15:13	Reserved	Reserved, the reset value must be maintained
12:8	DBLEN[4:0]	<p>DMA Burst Length</p> <p>This bit field defines the number DMA will access (write/read) TIMx_DADDR register.</p> <p>00000: 1 time transfer</p> <p>00001: 2 times transfers</p> <p>00010: 3 times transfers</p> <p>...</p> <p>10001: 18 times transfers</p>
7:5	Reserved	Reserved, the reset value must be maintained.

Bit field	Name	Description
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. 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 \times 4) will be accessed.</p> <p>DMA base address = The address of TIMx_CTRL1 + TIMx_DCTRL. DBADDR * 4; 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</p>

Bit field	Name	Description
		TIMx_CCDAT4 register;

10.4.20 Channel 1 filter register (TIMx_C1FILT)

Offset address: 0x70

Reset value: 0x0000 0000

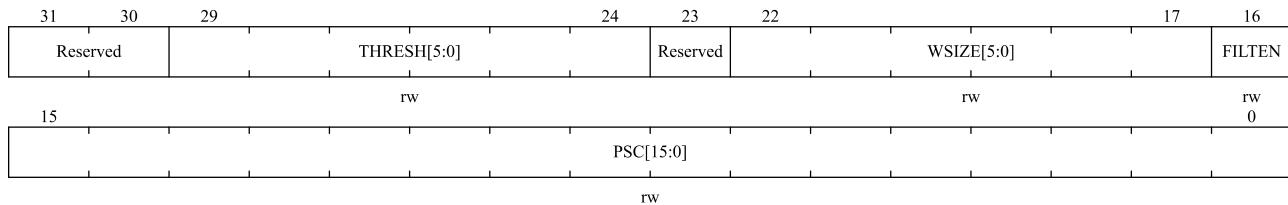
31	30	29	THRESH[5:0]	24	23	22	WSIZE[5:0]	17	16
Reserved				Reserved					FILTEN
15			rw				rw		rw
			PSC[15:0]						0
				rw					

Bit field	Name	Description
31:30	Reserved	Reserved, the reset value must be maintained
29:24	THRESH[5:0]	<p>Threshold number of sample logic level to be valid, maximum 63:</p> <p>Threshold value for a valid logic level. Within sample window if number of logic high is more than or equal to threshold value, next logic level will be logic high. Same rule applies to logic low. If both number of 1's and 0's inside window are smaller than threshold, filter output stays unchanged. Threshold value should set to more than or equal to half of Window value.</p> <p>Recommend threshold range is:</p> <p>Minimum: 1 pre-scale clock cycle more than ceiling value of max glitch size (in pre-scale clock cycle) and need to larger than half of window size.</p> <p>for example, if glitch size is 3.2*(pre-scale clock period), threshold should be $\lceil 3.2 \rceil = 4 + 1 = 5$</p> <p>Maximum: floor value of minimum size of valid signal (in pre-scale clock cycle) and need to be smaller than window size.</p> <p>For example, if minimum message size is 3.2*(pre-scale clock period), threshold should be floor (3.2) = 3.</p>
23	Reserved	Reserved
22:17	WSIZE[5:0]	<p>Window size value for logic level check, maximum 63:</p> <p>Window size decides how many sampled values will take into consideration for getting next logic level. Build-in FIFO is 64 bits with maximum index 63 which can only set window size to be 63.</p>
16	FILTEN	<p>Filter enable:</p> <p>0: Filter disable.</p> <p>1: Enable filter function.</p>
15:0	PSC[15:0]	<p>Prescaler register value for configure filter sample clock:</p> <p>For this filter, it supports 65535 scale (16 bits).</p> <p>Clock prescaler scaling system clock to sample clock. Sample clock decides distance between two sample point. Only value at sample point will take into consideration for valid logic level calculation.</p>

10.4.21 Channel 2 filter register (TIMx_C2FILT)

Offset address: 0x74

Reset value: 0x0000 0000

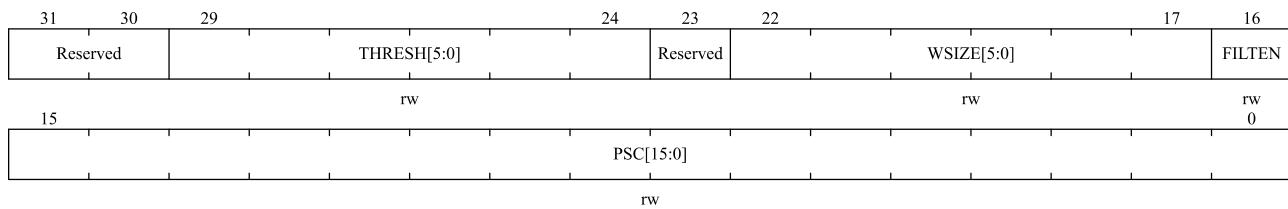


Bit field	Name	Description
31:30	Reserved	Reserved, the reset value must be maintained
29:24	THRESH[5:0]	<p>Threshold number of sample logic level to be valid, maximum 63:</p> <p>Threshold value for a valid logic level. Within sample window if number of logic high is more than or equal to threshold value, next logic level will be logic high. Same rule applies to logic low. If both number of 1's and 0's inside window are smaller than threshold, filter output stays unchanged. Threshold value should set to more than or equal to half of Window value.</p> <p>Recommend threshold range is:</p> <p>Minimum: 1 pre-scale clock cycle more than ceiling value of max glitch size (in pre-scale clock cycle) and need to larger than half of window size.</p> <p>for example, if glitch size is $3.2 \times (\text{pre-scale clock period})$, threshold should be $\lceil 3.2 \rceil = 4 + 1 = 5$</p> <p>Maximum: floor value of minimum size of valid signal (in pre-scale clock cycle) and need to be smaller than window size.</p> <p>For example, if minimum message size is $3.2 \times (\text{pre-scale clock period})$, threshold should be floor $(3.2) = 3$.</p>
23	Reserved	Reserved, the reset value must be maintained
22:17	WSIZE[5:0]	<p>Window size value for logic level check, maximum 63:</p> <p>Window size decides how many sampled values will take into consideration for getting next logic level. Build-in FIFO is 64 bits with maximum index 63 which can only set window size to be 63.</p>
16	FILTEN	<p>Filter enable:</p> <p>0: Filter disable.</p> <p>1: Enable filter function.</p>
15:0	PSC[15:0]	<p>Prescaler register value for configure filter sample clock:</p> <p>For this filter, it supports 65535 scale (16 bits).</p> <p>Clock prescaler scaling system clock to sample clock. Sample clock decides distance between two sample point. Only value at sample point will take into consideration for valid logic level calculation.</p>

10.4.22 Channel 3 filter register (TIMx_C3FILT)

Offset address: 0x78

Reset value: 0x0000 0000

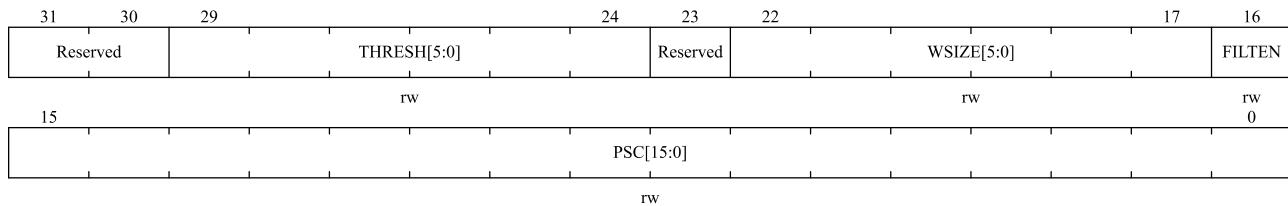


Bit field	Name	Description
31:30	Reserved	Reserved, the reset value must be maintained
29:24	THRESH[5:0]	<p>Threshold number of sample logic level to be valid, maximum 63:</p> <p>Threshold value for a valid logic level. Within sample window if number of logic high is more than or equal to threshold value, next logic level will be logic high. Same rule applies to logic low. If both number of 1's and 0's inside window are smaller than threshold, filter output stays unchanged. Threshold value should set to more than or equal to half of Window value.</p> <p>Recommend threshold range is:</p> <p>Minimum: 1 pre-scale clock cycle more than ceiling value of max glitch size (in pre-scale clock cycle) and need to larger than half of window size.</p> <p>for example, if glitch size is $3.2 * (\text{pre-scale clock period})$, threshold should be $\lceil 3.2 \rceil = 4 + 1 = 5$</p> <p>Maximum: floor value of minimum size of valid signal (in pre-scale clock cycle) and need to be smaller than window size.</p> <p>For example, if minimum message size is $3.2 * (\text{pre-scale clock period})$, threshold should be floor $(3.2) = 3$.</p>
23	Reserved	Reserved, the reset value must be maintained
22:17	WSIZE[5:0]	<p>Window size value for logic level check, maximum 63:</p> <p>Window size decides how many sampled values will take into consideration for getting next logic level. Build-in FIFO is 64 bits with maximum index 63 which can only set window size to be 63.</p>
16	FILTEN	<p>Filter enable:</p> <p>0: Filter disable.</p> <p>1: Enable filter function.</p>
15:0	PSC[15:0]	<p>Prescaler register value for configure filter sample clock:</p> <p>For this filter, it supports 65535 scale (16 bits).</p> <p>Clock prescaler scaling system clock to sample clock. Sample clock decides distance between two sample point. Only value at sample point will take into consideration for valid logic level calculation.</p>

10.4.23 Channel 4 filter register (TIMx_C4FILT)

Offset address: 0x7C

Reset value: 0x0000 0000

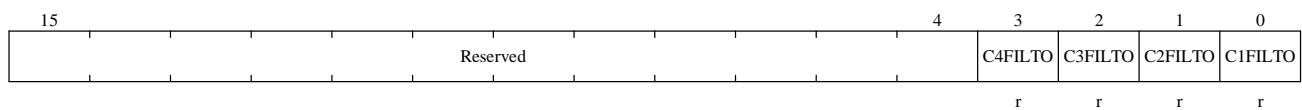
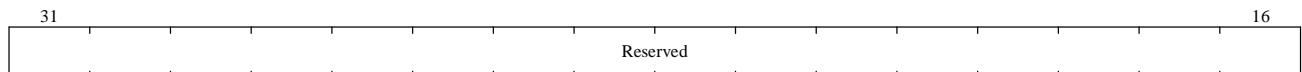


Bit field	Name	Description
31:30	Reserved	Reserved, the reset value must be maintained
29:24	THRESH[5:0]	<p>Threshold number of sample logic level to be valid, maximum 63:</p> <p>Threshold value for a valid logic level. Within sample window if number of logic high is more than or equal to threshold value, next logic level will be logic high. Same rule applies to logic low. If both number of 1's and 0's inside window are smaller than threshold, filter output stays unchanged. Threshold value should set to more than or equal to half of Window value.</p> <p>Recommend threshold range is:</p> <p>Minimum: 1 pre-scale clock cycle more than ceiling value of max glitch size (in pre-scale clock cycle) and need to larger than half of window size.</p> <p>for example, if glitch size is $3.2 \times (\text{pre-scale clock period})$, threshold should be $\lceil 3.2 \rceil = 4 + 1 = 5$</p> <p>Maximum: floor value of minimum size of valid signal (in pre-scale clock cycle) and need to be smaller than window size.</p> <p>For example, if minimum message size is $3.2 \times (\text{pre-scale clock period})$, threshold should be floor $(3.2) = 3$.</p>
23	Reserved	Reserved, the reset value must be maintained
22:17	WSIZE[5:0]	<p>Window size value for logic level check, maximum 63:</p> <p>Window size decides how many sampled values will take into consideration for getting next logic level. Build-in FIFO is 64 bits with maximum index 63 which can only set window size to be 63.</p>
16	FILTEN	<p>Filter enable:</p> <p>0: Filter disable.</p> <p>1: Enable filter function.</p>
15:0	PSC[15:0]	<p>Prescaler register value for configure filter sample clock:</p> <p>For this filter, it supports 65535 scale (16 bits).</p> <p>Clock prescaler scaling system clock to sample clock. Sample clock decides distance between two sample point. Only value at sample point will take into consideration for valid logic level calculation.</p>

10.4.24 Channel filter Output register (TIMx_FILTO)

Offset address: 0x80

Reset value: 0x0000 0000



Bit field	Name	Description
31:4	Reserved	Reserved, the reset value must be maintained
3	C4_FILTO	Channel 4 Filter output, read only
2	C3_FILTO	Channel 3 Filter output, read only
1	C2_FILTO	Channel 2 Filter output, read only
0	C1_FILTO	Channel 1 Filter output, read only

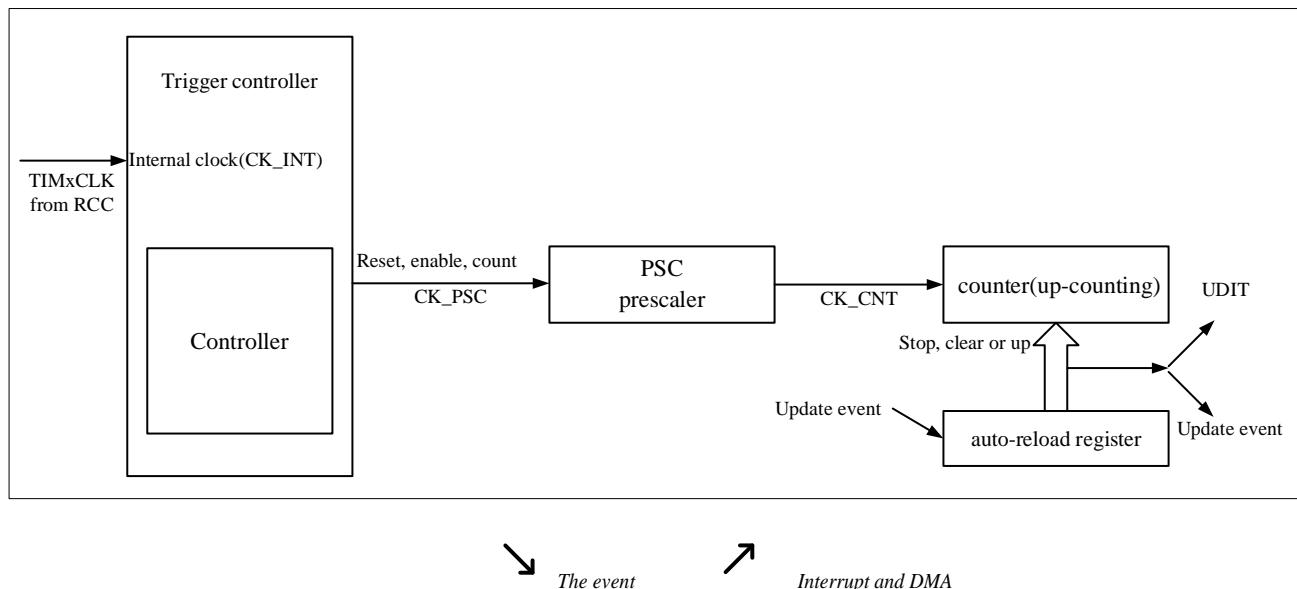
11 Basic timers (TIM6)

The basic timer contains a 16-bit counter.

11.1 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)



11.2 Basic timers description

11.2.1 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.

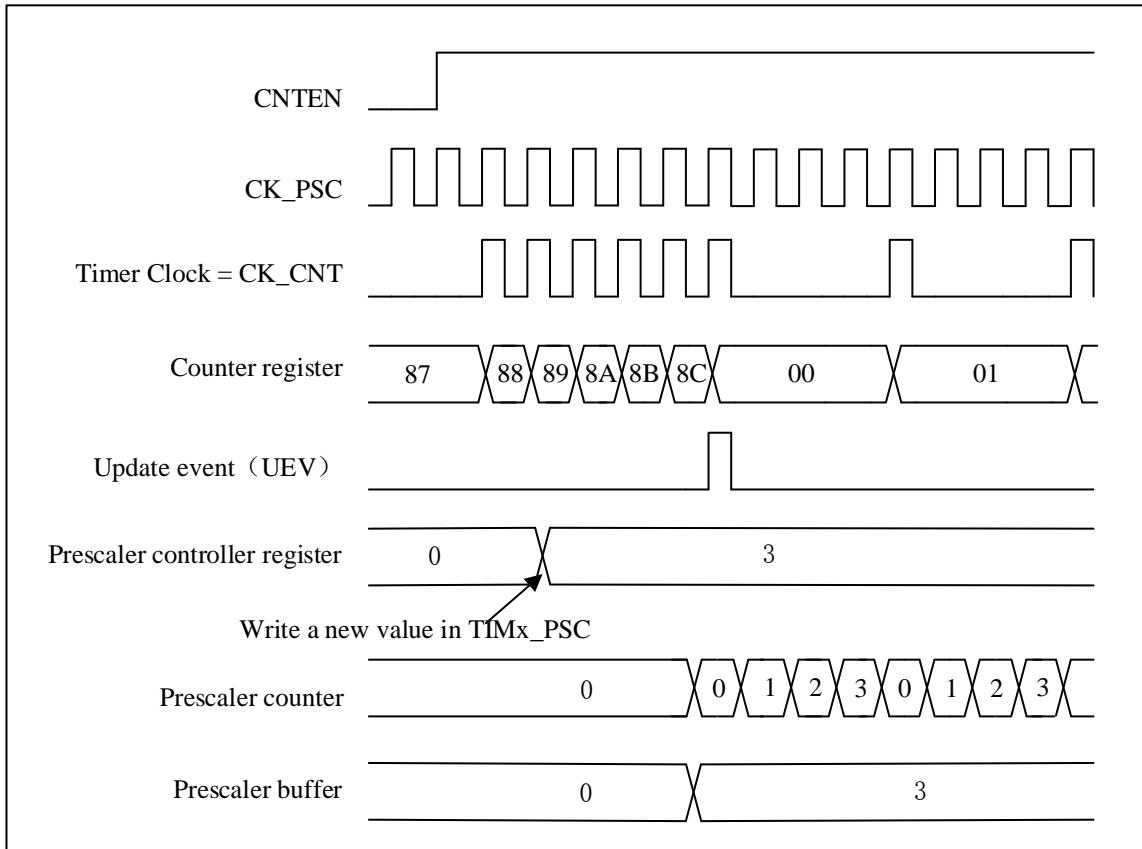
Depending on the setting of the auto-reload preload enable bit (TIMx_CTRL1.ARREN), the value of the preload register is transferred to the shadow register immediately or at each update event UEV. An update event is generated when the counter reaches the overflow condition and it can be generated by software 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.

11.2.1.1 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



11.2.2 Counter mode

11.2.2.1 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.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 11-3 Timing diagram of up-counting. The internal clock divider factor = 2/N

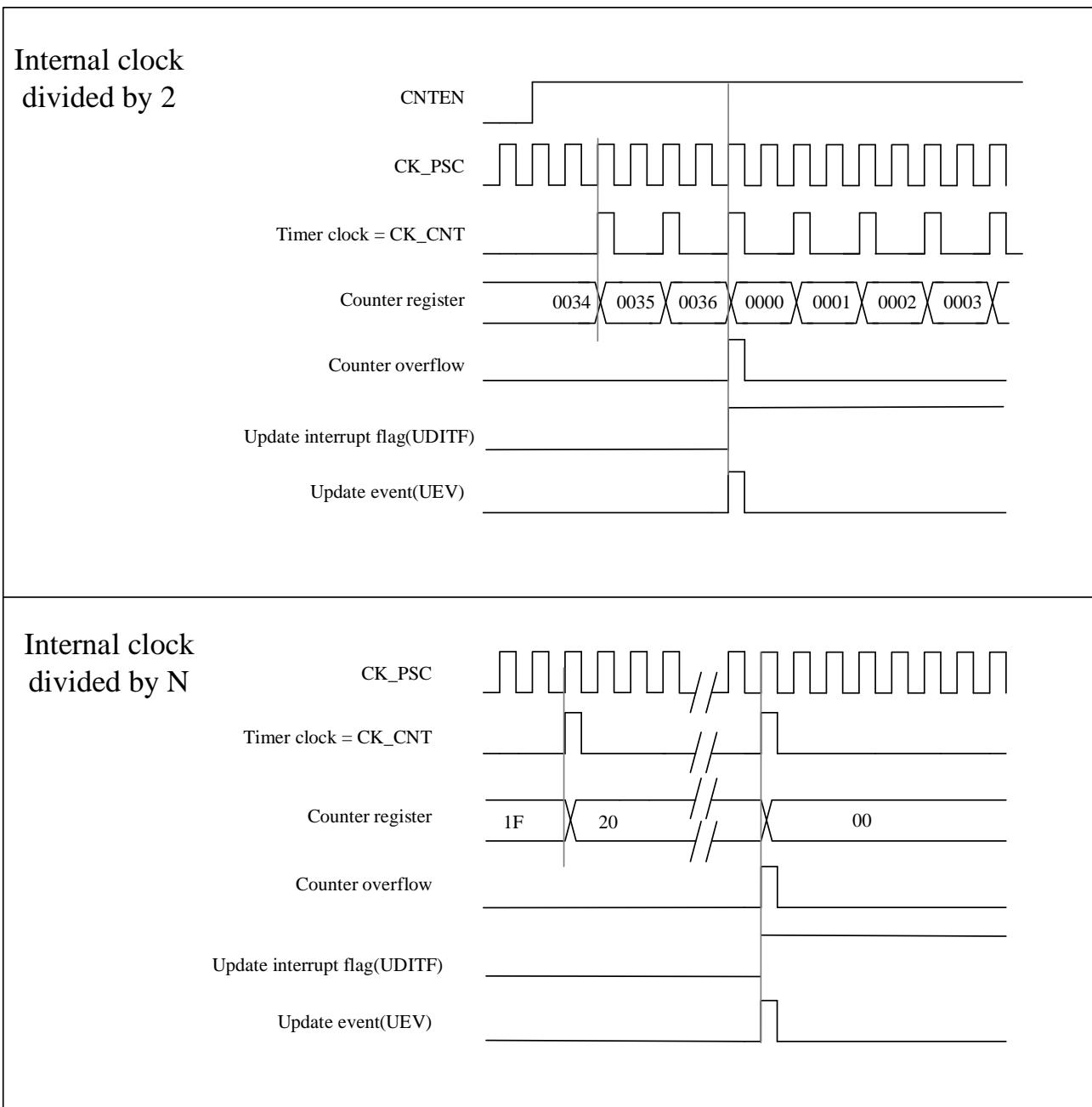
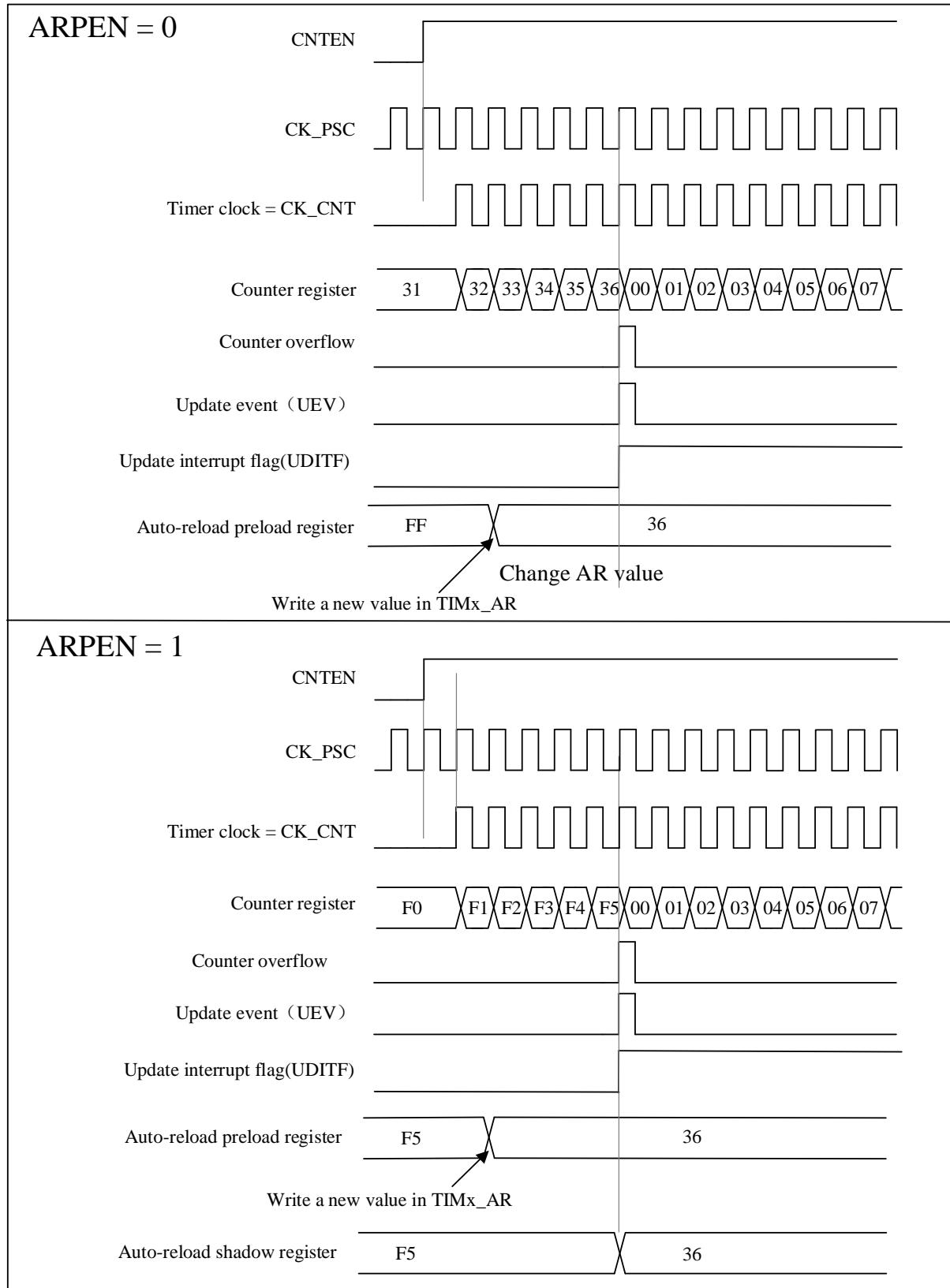


Figure 11-4 Timing diagram of the up-counting, update event when ARPEN=0/1



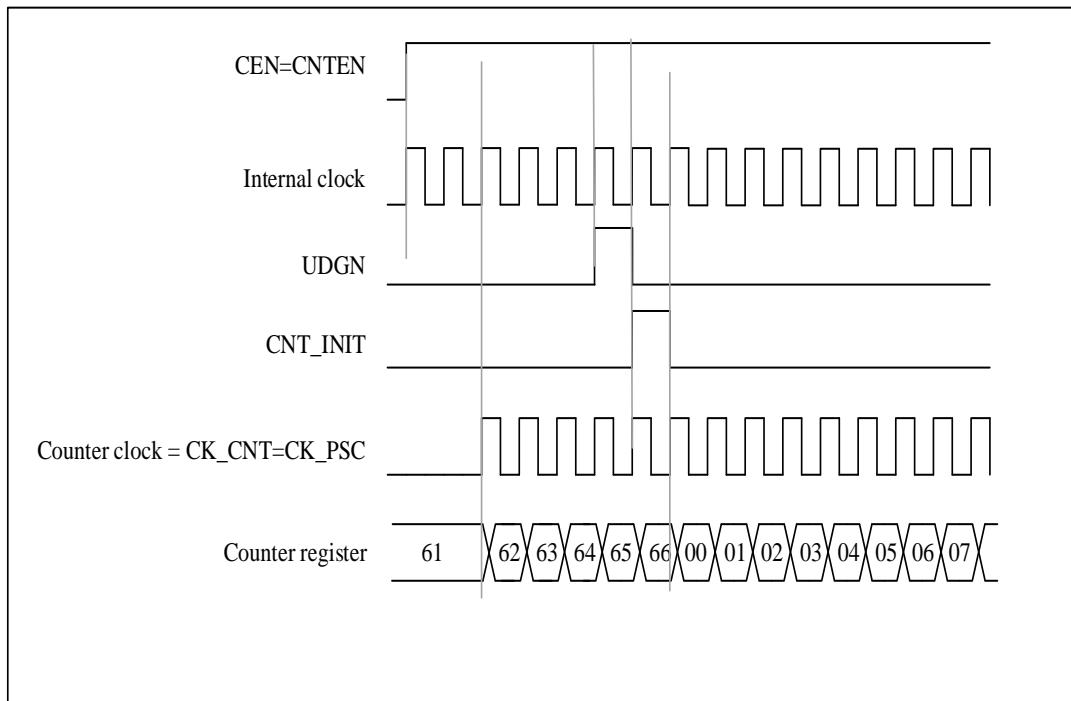
11.2.3 Clock selection

- The internal clock of timers : CK_INT

11.2.3.1 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



11.2.4 Debug mode

When the microcontroller is in debug mode (the Cortex-M4 core halted), depending on the DBG_TIMx_STOP configuration in the DBG module, the TIMx counter can either continue to work normally or stop. For more details, see 21.4.3.

11.3 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).

11.3.1 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_CTRL1																																	
	Reset Value																																	
004h																																		
008h																																		
00Ch	TIMx_DINTEN																																	
	Reset Value																																	
010h	TIMx_STS																																	
	Reset Value																																	
014h	TIMx_EVTGEN																																	
	Reset Value																																	
018h																																		
01Ch																																		
020h																																		
024h	TIMx_CNT																																	
	Reset Value																																	
028h	TIMx_PSC																																	
	Reset Value																																	
02Ch	TIMx_AR																																	
	Reset Value																																	

11.3.2 Control Register 1 (TIMx_CTRL1)

Offset address: 0x00

Reset value: 0x0000

15	Reserved	8	7	6	5	4	3	2	1	0
			ARPEN		Reserved		ONEPM	UPRS	UPDIS	CNTEN

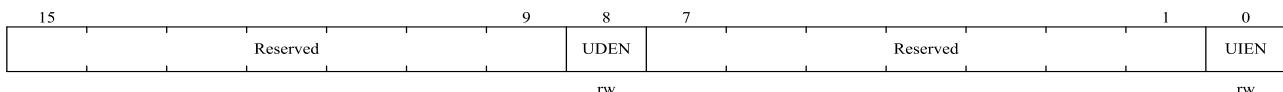
Bit field	Name	Description
15:8	Reserved	Reserved, the reset value must be maintained

Bit field	Name	Description
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

11.3.3 DMA/Interrupt Enable Registers (TIMx_DINTEN)

Offset address: 0x0C

Reset value: 0x0000



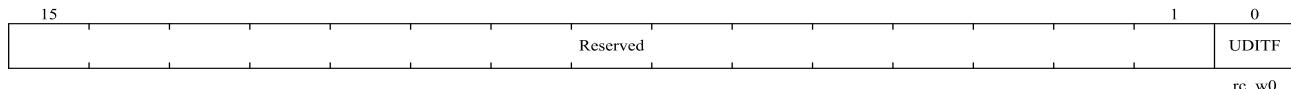
Bit field	Name	Description
15:9	Reserved	Reserved, the reset value must be maintained
8	UDEN	Update DMA Request enable 0: Disable update DMA request 1: Enable update DMA request

Bit field	Name	Description
7:1	Reserved	Reserved, the reset value must be maintained
0	UIEN	Update interrupt enable 0: Disable update interrupt 1: Enables update interrupt

11.3.4 Status Registers (TIMx_STS)

Offset address: 0x10

Reset value: 0x0000

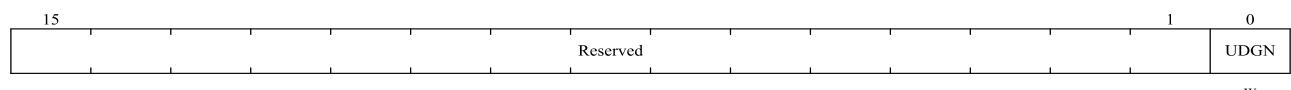


Bit field	Name	Description
15:1	Reserved	Reserved, the reset value must be maintained
0	UDITF	<p>Update interrupt flag</p> <p>This bit is set by hardware when an update event occurs under the following conditions:</p> <ul style="list-style-type: none"> When TIMx_CTRL1.UPDIS = 0, and counter value overflow. When TIMx_CTRL1.UPRS = 0, TIMx_CTRL1.UPDIS = 0, and set the TIMx_EVTGEN.UDGN bit by software to reinitialize the CNT. <p>This bit is cleared by software.</p> <p>0: No update event occurred 1: Update interrupt occurred</p>

11.3.5 Event Generation registers (TIMx_EVTGEN)

Offset address: 0x14

Reset values: 0 x0000

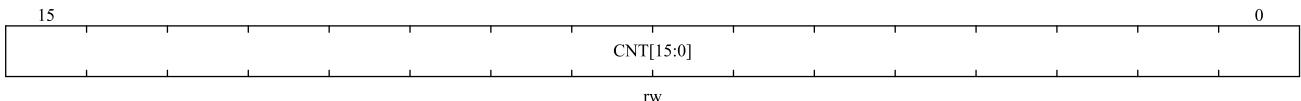


Bit field	Name	Description
15: 1	Reserved	Reserved, the reset value must be maintained.
0	UDGN	<p>UDGN: Update generation</p> <p>Software can set this bit to update configuration register value and hardware will clear it automatically.</p> <p>0: No effect. 1: Timer counter will restart and all shadow register will be updated. It will restart prescaler counter also.</p>

11.3.6 Counters (TIMx_CNT)

Offset address: 0x24

Reset value: 0x0000

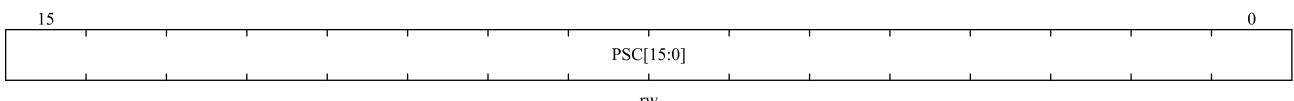


Bit field	Name	Description
15:0	CNT[15:0]	Counter value

11.3.7 Prescaler (TIMx_PSC)

Offset address: 0x28

Reset value: 0x0000

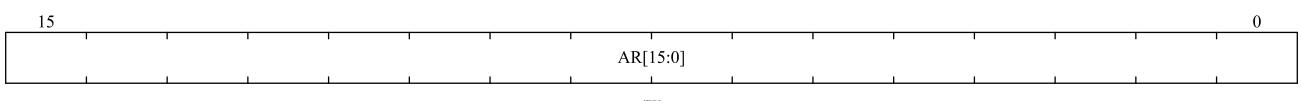


Bit field	Name	Description
15:0	PSC[15:0]	Prescaler value PSC register value will be updated to prescaler register at update event. Counter clock frequency is input clock frequency divide PSC + 1.

11.3.8 Automatic 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 11.2.1 for more details. When the TIMx_AR.AR [15:0] value is null, the counter does not work.

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 Standby 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 upcounter
- Clock prescaler with 3-bit to provide 8 dividing factors (1,2,4,8,16,32,64,128)
- Multiple clock sources

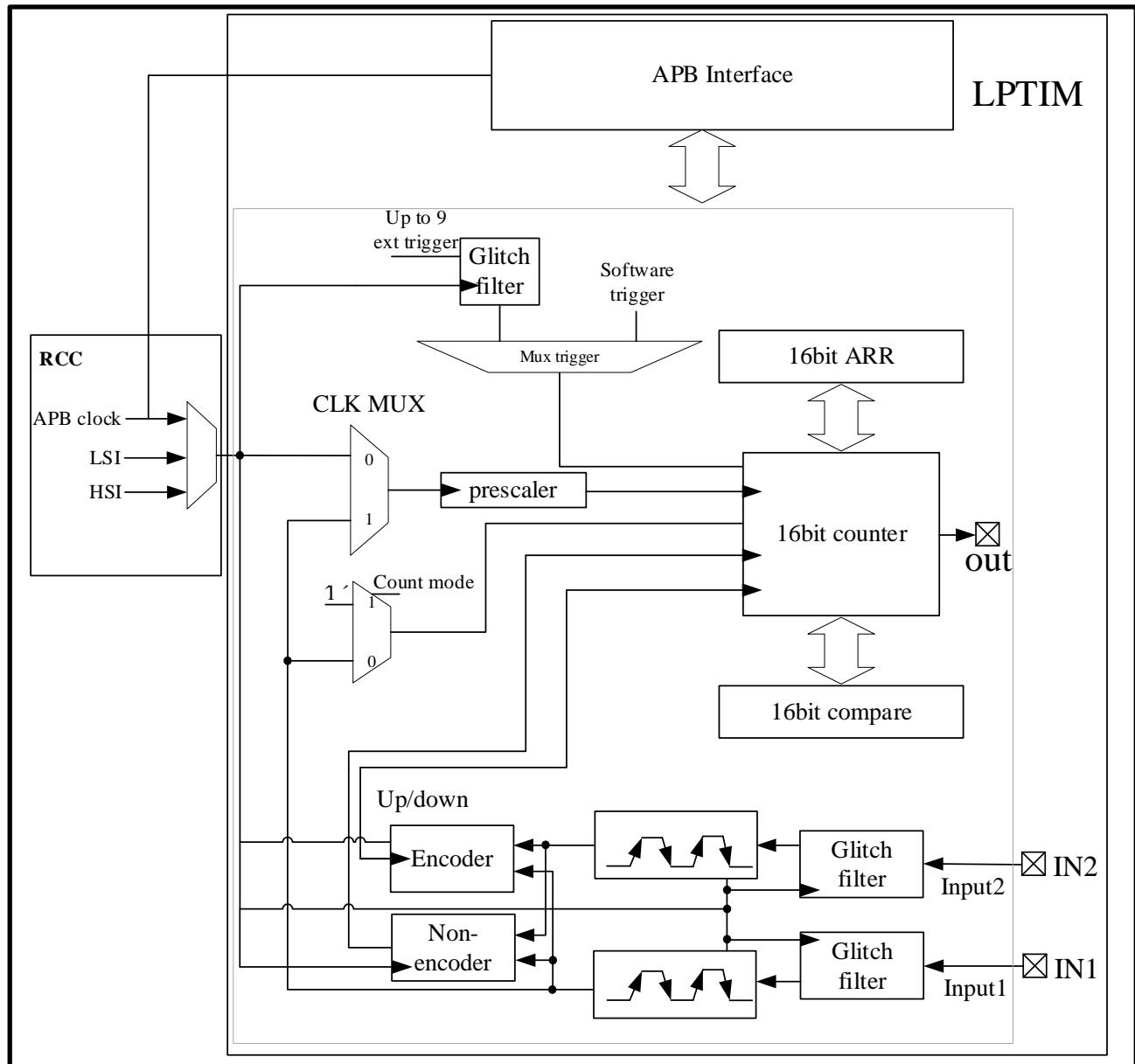
Internal : LSI, HSI or APB1 clock

External : LPTIM Input1 (working with no LP oscillator running used by Pulse Counter application)

- 16-bit auto-reload register
- 16-bit compare register
- Continuous or One-shot counting mode
- Programmable software or hardware input trigger
- Programmable digital filter for filtering glitch
- Configurable output: Pulse, PWM
- Configurable I/O polarity
- Encoder mode
- Pulse counting mode, support single pulse counting, double pulse counting (orthogonal and non-orthogonal)

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 internal clock source is configurable from RCC_RDCTRL.LPTIMSEL[2:0] bits. The external clock source can be selected from 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 external input1. This configuration is suitable for LOW POWER application.

LPTIM_CFG.CLKSEL and LPTIM_CFG.CNTMEN bits are used 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 Prescalar

The LPTIM counter is preceded by a configurable power-of-2 pre-scaler. The prescaler ratio is controlled by the LPTIM_CFG.CLKPRE[2:0] field. The table below lists all the possible division ratios:

Table 12-1 Pre-scalar 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 active edge detection is configured through the LPTIM_CFG.CLKFLT[1:0] bits
- For the internal trigger inputs: The stable active edge detection period is configured through the LPTIM_CFG.RIGFLT[1:0] bits.

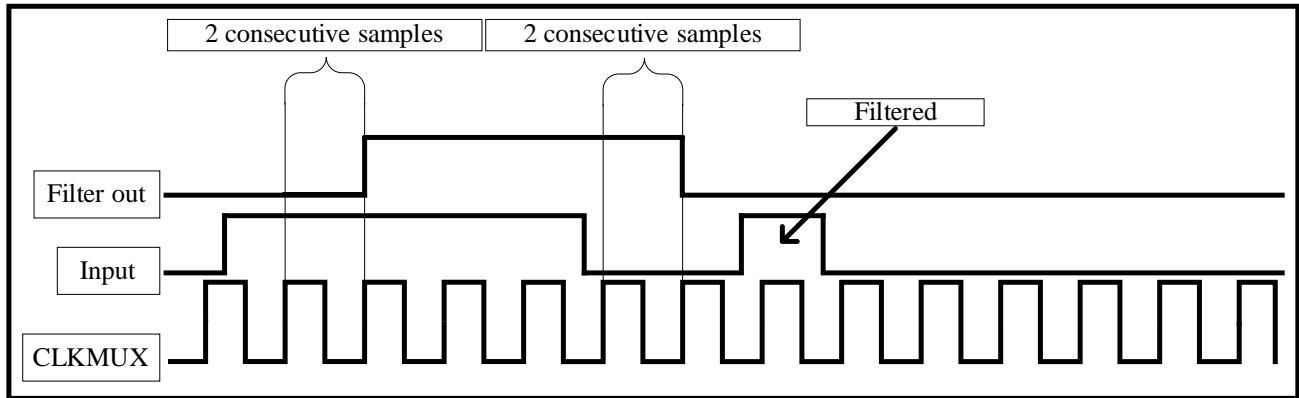
Note: The detection configuration is only applicable for its corresponding inputs.

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.

Figure 12-2 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 GPIO 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 4 trigger inputs.

The trigger source is configured through LPTIM_CFG.TRGEN[1:0] bits. If LPTIM_CFG.TRGEN[1:0] = '00', LPTIM can be triggered by setting the LPTIM_CTRL.TSTCM or LPTIM_CTRL.SNGMST bit. The other values of LPTIM_CFG.TRGEN[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[3:0] is used to select one of the 4 trigger inputs only when LPTIM_CFG.TRGEN[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.

If 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 4 trigger inputs corresponding to LPTIM_CFG.TRGSEL[3:0] bits

Control bits	Corresponding trigger input
0000	PA10
0001	RTC alarm A
0010	RTC alarm B
0011	Reserved
0100	RTC_TAMP

12.4.6 Operating mode

The LPTIM has two operating modes:

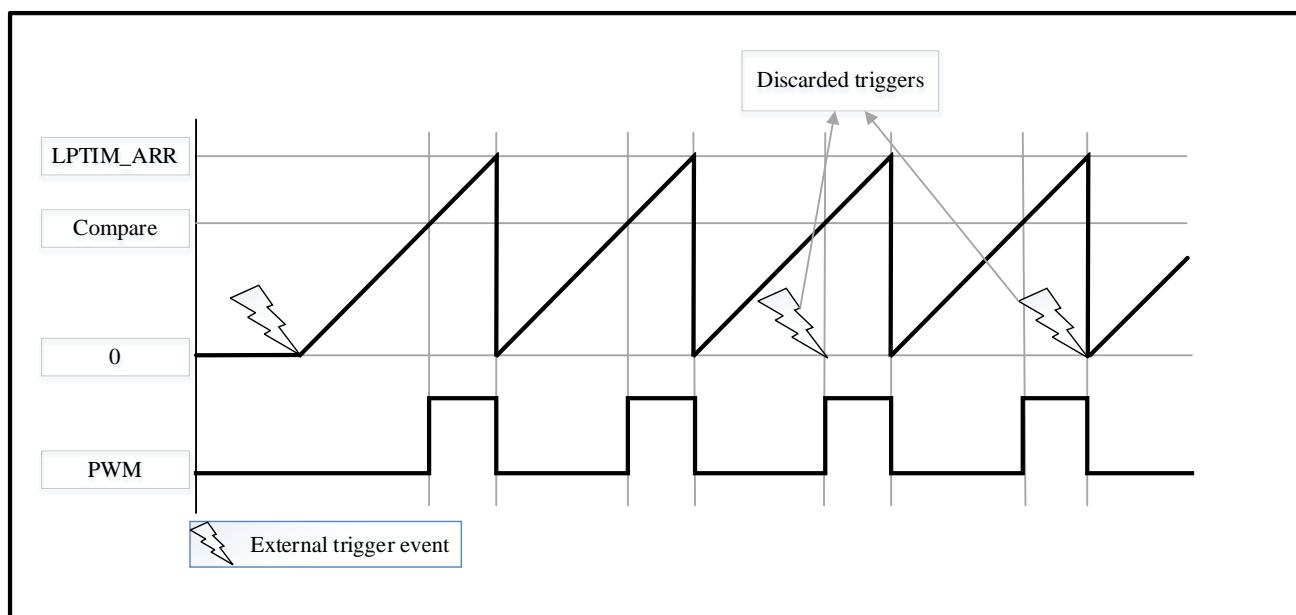
- Continuous mode: A trigger event will start the LPTIM and it will continue running until the user switched off the LPTIM.
- One-shot mode: A trigger event will start the LPTIM and it will stop when the counter value reached LPTIM_ARR.ARRVAL[15:0].

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 LPTIM is enabled (The LPTIM_CTRL.LPTIMEN bit is set to '1').

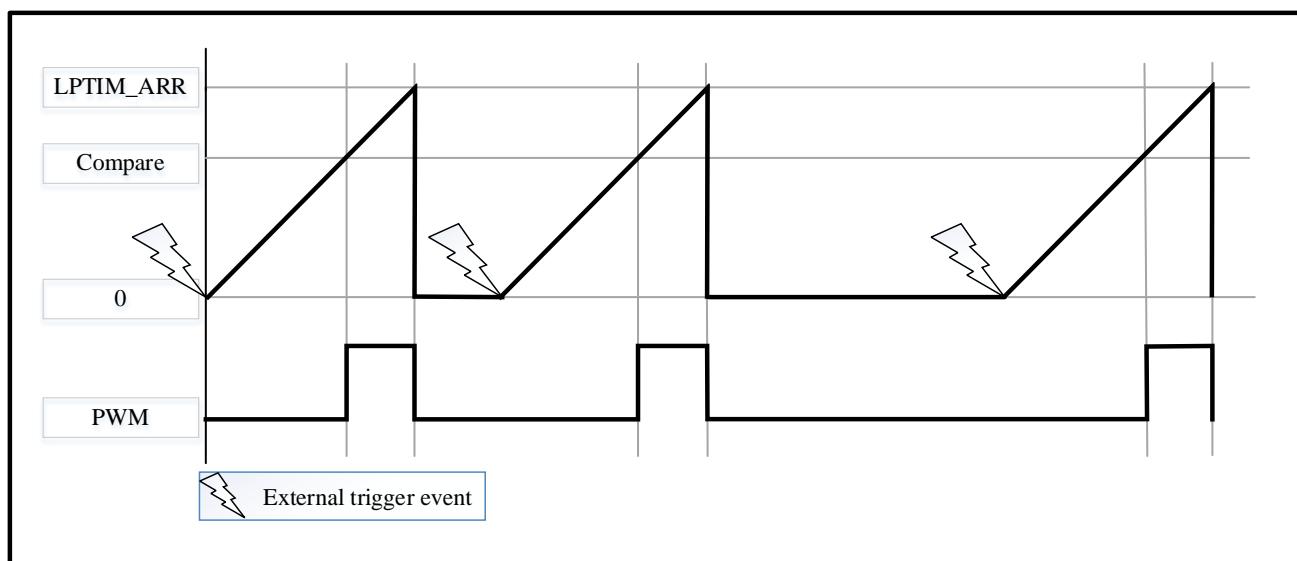
It is possible to switch from one-shot mode to continuous mode. Setting LPTIM_CTRL.SNGMST bit will switch the LPTIM to one-shot counting mode if continuous counting mode was previously selected. The counter stops as soon as it reaches the LPTIM_ARR register value if timer enable. If the one-shot counting 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 new 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.

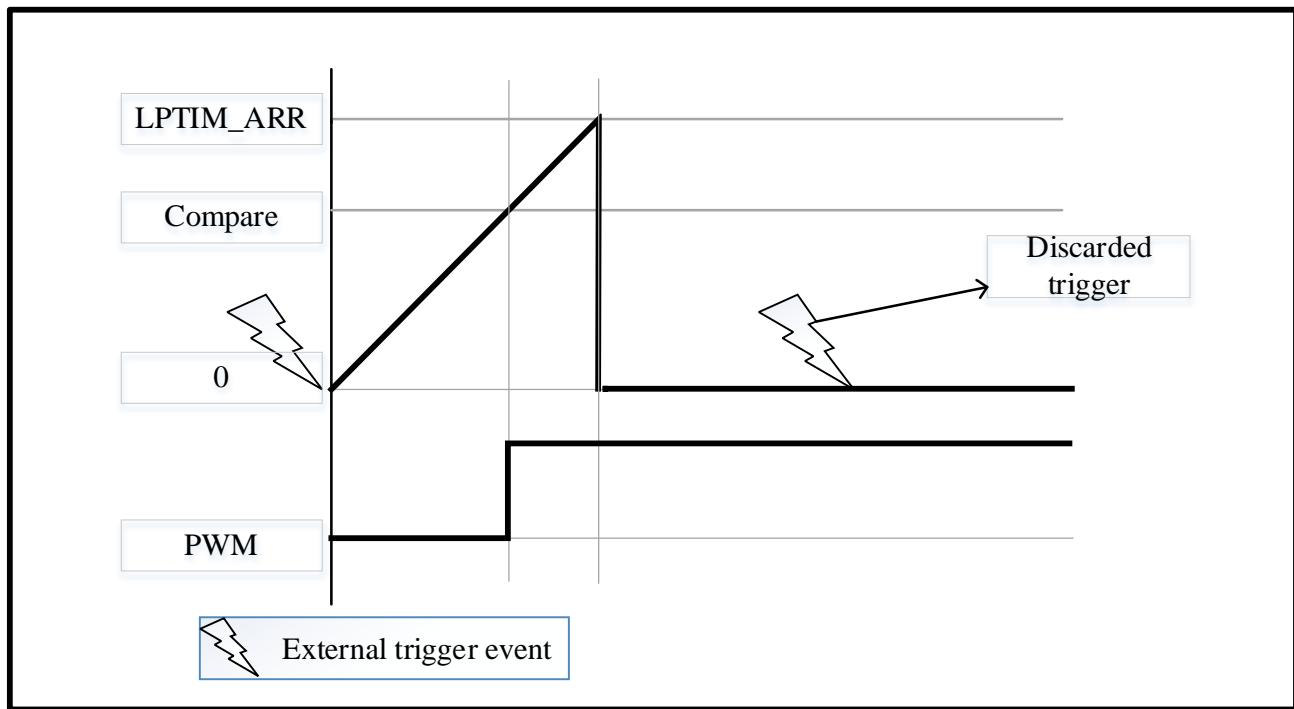
Figure 12-4 PTIM output waveform, single counting mode configuration



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 a 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.

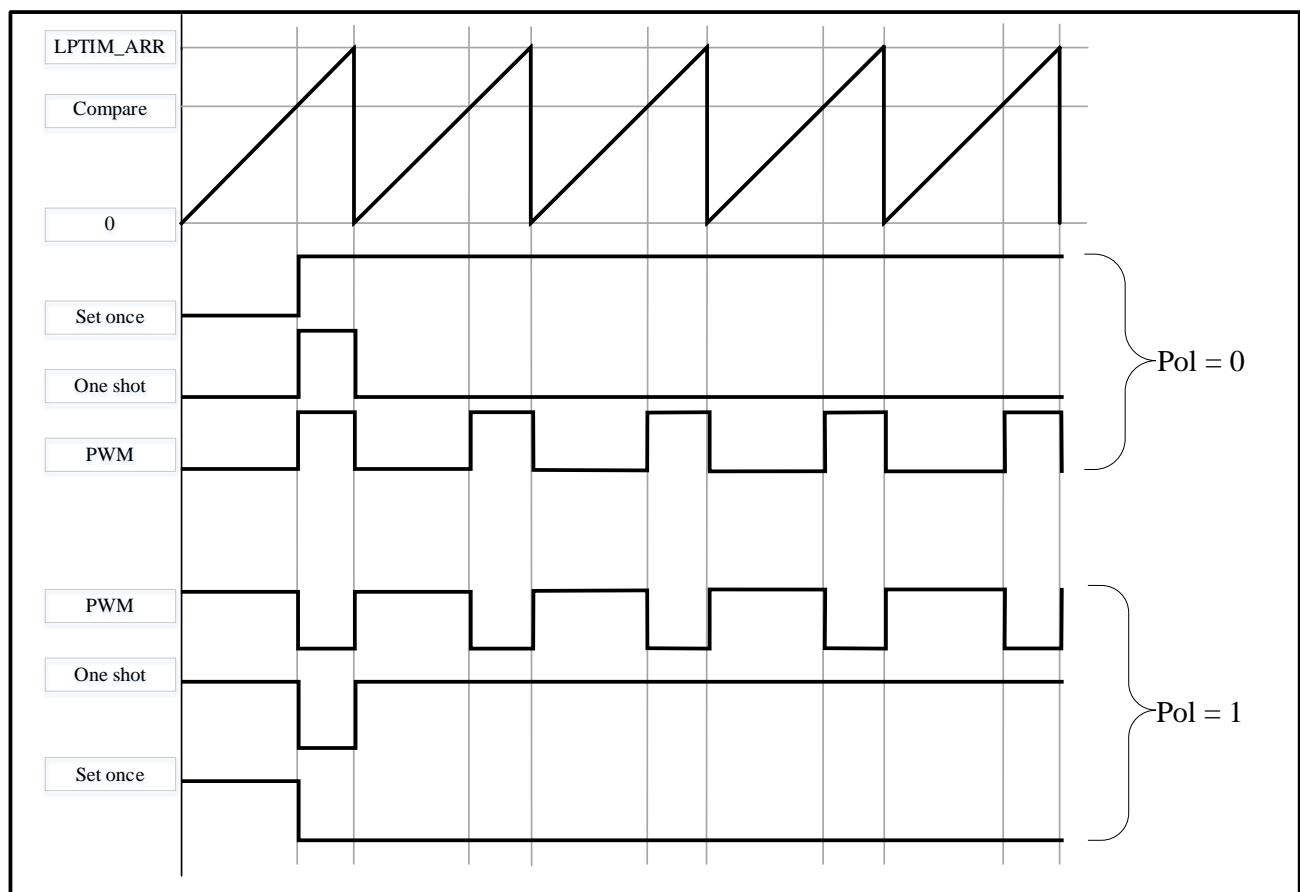
The LPTIM_CFG.WAVEPOL bit controls LPTIM output polarity. The output idle steady level will change immediately after the user configured the polarity, even when the timer is disabled.

Signals with frequencies up to the LPTIM clock frequency divided by 2 can be generated. Only when LPTIM counter counting external 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]='d3. d1,d2 and d3 means decimal 1, 2, 3)

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 immediately after a software write operation without LPTIM enabled. 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 prescaled 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 the two Down and Up flags in the LPTIM_INTSTS register. Also, an interrupt can be generated for both direction change events through setting the LPTIM_INTEN.DOWNIE or LPTIM_INTENUPIE bit.

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 the Up and Down flags, correspond to the rotation direction of the encoder rotor.

Different trigger edges configured using the LPTIM_CFG.CLKPOL[1:0] bits may have different counting scenarios. The following table summarizes the possible combinations, assuming that Input1 and Input2 do not switch at the same time.

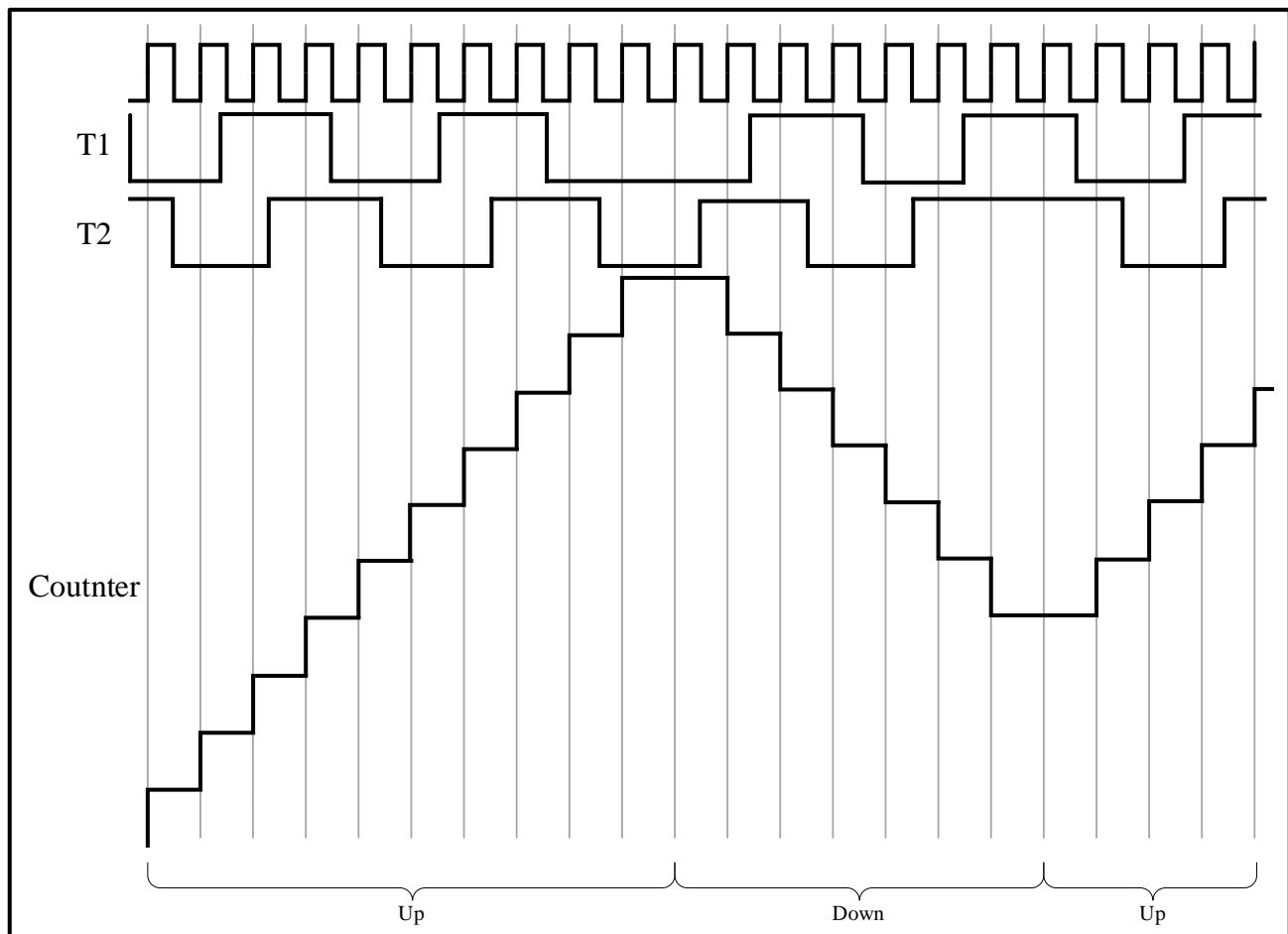
Table 12-3 Encoder counting scenarios

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 trigger 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.CLKPRE[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 LPTIM_ARR.ARRVAL[15:0] or LPTIM_ARR.ARRVAL[15:0] 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.

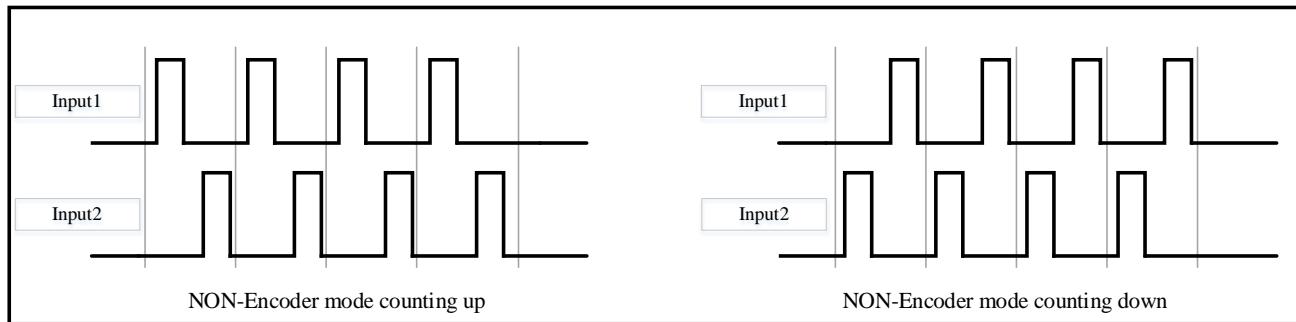
The change of counting direction is updated by the two Down and Up flags in the LPTIM_INTSTS register. Also, an interrupt can be generated for both direction change events through setting the LPTIM_INTEN.DOWNIE or LPTIM_INTEN.UPIE bit.

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 the Up and Down flags, correspond to the rotation direction of the encoder rotor.

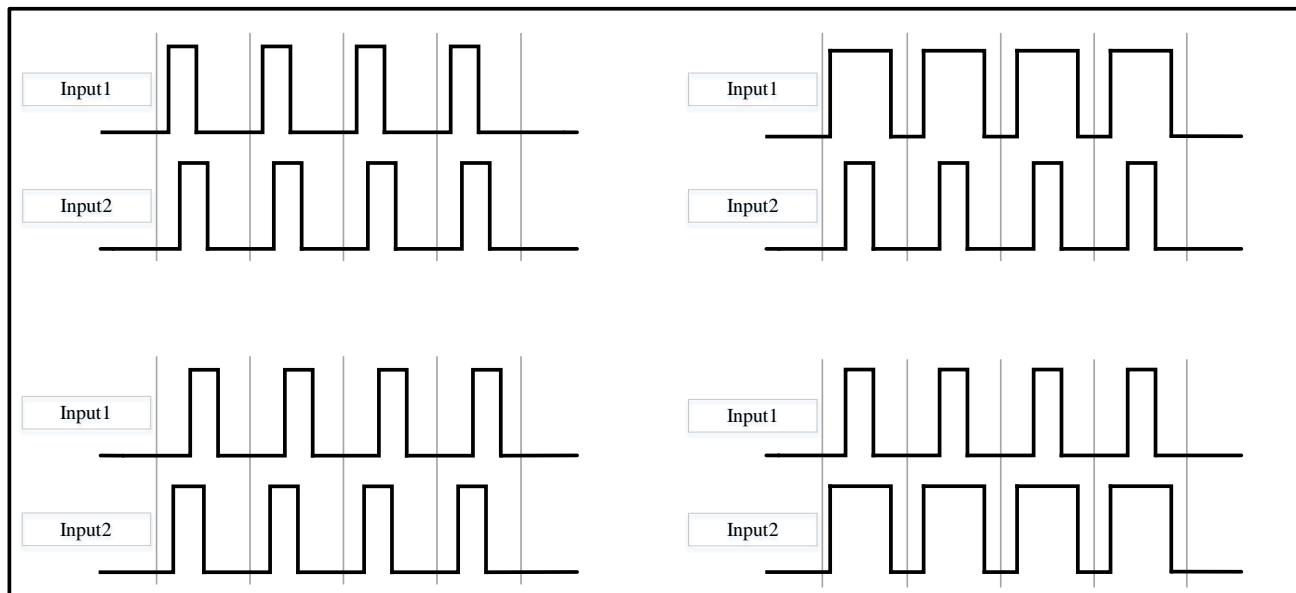
The following two waveforms, the decoder module can work properly, when there is no case that both Input1 and Input2 are high.

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
- Autoreload 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 signals up-counting direction change – DOWN flag signals down-counting direction change.

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	6	5	4	3	2	1	0		
000h	LPTIM_INTSTS																																		
	Reset Value																																		
004h	LPTIM_INTCLR																																		
	Reset Value																																		
008h	LPTIM_INTEN																																		
	Reset Value																																		
00Ch	LPTIM_CFG																																		
	Reset Value																																		
010h	LPTIM_CTRL																																		
	Reset Value																																		
014h	LPTIM_COMP																																		
	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

31	Reserved															16							
15	Reserved															7	6	5	4	3	2	1	0

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 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 CMPUPD 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

31	Reserved										16
15	Reserved	7	6	5	4	3	2	1	0		
			DOWN CF	UP CF	ARRUPD CF	CMPUPD CF	EXTRIG CF	ARRM CF	CMPM CF		
			w	w	w	w	w	w	w		

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	Autoreload 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	Autoreload 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')

31	Reserved								16
15	Reserved				7	6	5	4	3 2 1 0
						DOWN IE	UP IE	ARRUPD IE	CMPUPD IE EXTRIG IE ARRM IE CMPM IE
						rw	rw	rw	rw rw rw rw

Bit Field	Name	Description
31:7	Reserved	Reserved, the reset value must be maintained.
6	DOWNIE	Direction change to down interrupt enable bit. 0: DOWN interrupt disabled 1: DOWN interrupt enabled
5	UPIE	Direction change to up interrupt enable bit. 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 succeeded interrupt enable bit. 0: CMPUPD interrupt disabled 1: CMPUPD interrupt enabled
2	EXTRIGIE	External trigger valid edge interrupt enable bit. 0: EXTRIG interrupt disabled 1: EXTRIG interrupt enabled
1	ARRMIE	Auto reload match interrupt enable bit. 0: ARRM interrupt disabled 1: ARRM interrupt enabled

Bit Field	Name	Description
0	CMPMIE	Compare match interrupt enable bit. 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				NENC	ENC	CNTMEN	RELOAD	WAVE POL	WAVE	TIMOUT EN	TRGEN[1:0]	TRG SEL[3]
15	13	12	11	rw 9	rw 8	rw 7	rw 6	rw 5	rw 4	rw 3	rw 2	rw 1	rw 0
TRGSEL[2:0]	Reserved	CLKPRE[2:0]	Reserved	TRIGFLT[1:0]	Reserved	CLKFLT[1:0]	Reserved	CLKPOL[1:0]	CLKSEL				
rw		rw		rw		rw		rw		rw		rw	rw

Bit Field	Name	Description
31:26	Reserved	Reserved, the reset value must be maintained.
25	NENC	Non-Encoder mode enable 0: Non-Encoder mode disabled 1: Non-Encoder 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
20	WAVE	Waveform shape

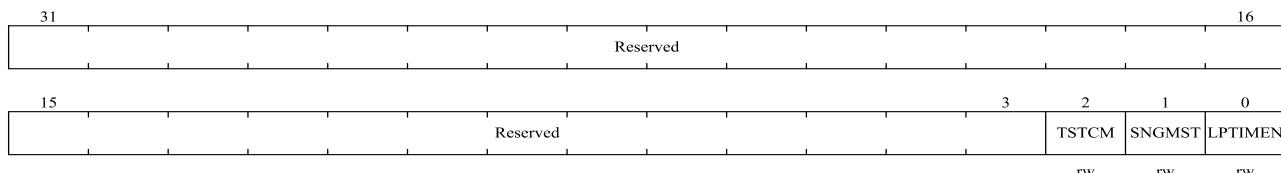
Bit Field	Name	Description
		<p>The WAVE bit controls the output shape</p> <p>0: Deactivate Set-once mode, PWM / One Pulse waveform (depending on LPTIM_CTRL.TSTCM or LPTIM_CTRL.SNGMST bit)</p> <p>1: Activate the Set-once mode</p>
19	TIMOUTEN	<p>Timeout enable</p> <p>0: A trigger event arriving when the 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>10: Falling edge is the active edge</p> <p>11: Both edges are active edges</p>
16:13	TRGSEL[3:0]	<p>Trigger selector</p> <p>The TRGSEL bits select the trigger source that will serve as a trigger event for the LPTIM among the below 4 available sources:</p> <p>0000: PA10</p> <p>0001: RTC alarm A</p> <p>0010: RTC alarm B</p> <p>0011: Reserved</p> <p>0100: RTC_TAMP</p> <p>Other values reserved</p>
12	Reserved	Reserved, the reset value must be maintained.
11:9	CLKPRE[2:0]	<p>Clock division factor bit.</p> <p>000: / 1</p> <p>001: / 2</p> <p>010: / 4</p> <p>011: / 8</p> <p>100: / 16</p> <p>101: / 32</p> <p>110: / 64</p> <p>111: / 128</p>
8	Reserved	Reserved, the reset value must be maintained.
7:6	TRIGFLT[1:0]	<p>Configure the data filter trigger bit.</p> <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>

Bit Field	Name	Description
		01: Trigger active level change must be stable for at least 2 clock periods before it is considered as valid trigger. 10: Trigger active level change must be stable for at least 4 clock periods before it is considered as valid trigger. 11: Trigger active level change must be stable for at least 8 clock periods before it is considered as valid trigger.
5	Reserved	Reserved, the reset value must be maintained.
4:3	CLKFLT[1:0]	Digital filter external clock input configuration 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 00: Any external clock signal level change is considered as a valid transition. 01: External clock signal level change must be stable for at least 2 clock periods before it is considered as valid transition. 10: External clock signal level change must be stable for at least 4 clock periods before it is considered as valid transition. 11: External clock signal level change must be stable for at least 8 clock periods before it is considered as valid transition.
2:1	CLKPOL[1:0]	Clock Polarity If LPTIM is clocked by an external clock source: 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: 00: The rising edge is the active edge used for counting 01: The falling edge is the active edge used for counting 10: Both edges are active edges 11: Not allowed <i>Note: 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.</i> If the LPTIM is configured in Encoder mode (LPTIM_CFG.ENC bit is set): 00: The encoder is active on rising edge 01: The encoder is active on falling edge 10: The encoder is active on double sided edge
0	CLKSEL	Clock selector The CKSEL bit selects which clock source the LPTIM will use: 0: LPTIM is clocked by internal clock source (APB1 clock or any of the embedded oscillators) 1: LPTIM is clocked by an external clock source through the LPTIM external Input1

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</p> <p>This bit is set by software and cleared by hardware.</p> <p>In case of software start (LPTIM_CFG.TRGEN[1:0] = '00'), setting this bit starts the LPTIM in Continuous mode.</p> <p>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.</p> <p>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.</p> <p>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</p> <p>This bit is set by software and cleared by hardware.</p> <p>In case of software start (LPTIM_CFG.TRGEN[1:0] = '00'), setting this bit starts the LPTIM in single pulse mode.</p> <p>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.</p> <p>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.</p> <p>This bit can only be set when the LPTIM is enabled. It will be automatically reset by hardware.</p>
0	LPTIMEN	<p>LPTIM enable</p> <p>The LPTIMEN bit is set and cleared by software.</p> <p>0: LPTIM is disabled</p> <p>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')

Diagram illustrating the register structure for R31. The register is 32 bits wide, with bit 31 at the top and bit 0 at the bottom. Bit 31 is labeled 'Reserved'. Bits 15 to 0 are labeled 'CMPVAL[15:0]'. The bottom bit is labeled 'rw'.

31	Reserved														16	
15	CMPVAL[15:0]														0	
																rw

Bit Field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained.
15:0	CMPVAL[15:0]	Compare value CMPVAL[15:0] 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')

Diagram illustrating a memory-mapped register structure. The top line, labeled "Reserved", spans from address 31 to 16. The bottom line, labeled "ARRVAL[15:0]", spans from address 15 to 0. Both lines have a "rw" label at the bottom center.

Bit Field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained.
15:0	ARRVAL[15:0]	<p>Auto reload value</p> <p>ARRVAL[15:0] is the autoreload value for the LPTIM.</p> <p>This value must be strictly greater than the LPTIM_COMP.CMPVAL[15:0] value.</p>

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]	Counter value When the LPTIM is running with an asynchronous clock, reading the LPTIM_CNT register may return unreliable values. So in this case it is necessary to perform two consecutive read accesses and verify that the two returned values are identical. If identical, the reading is reliable.

13 Independent watchdog (IWDG)

13.1 Introduction

The N32G401F8S7-1 has built-in independent watchdog (IWDG) and window watchdog (WWDG) 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 driven by Low-speed internal clock (LSI clock) running at 40 KHz, which will still run even if the 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.

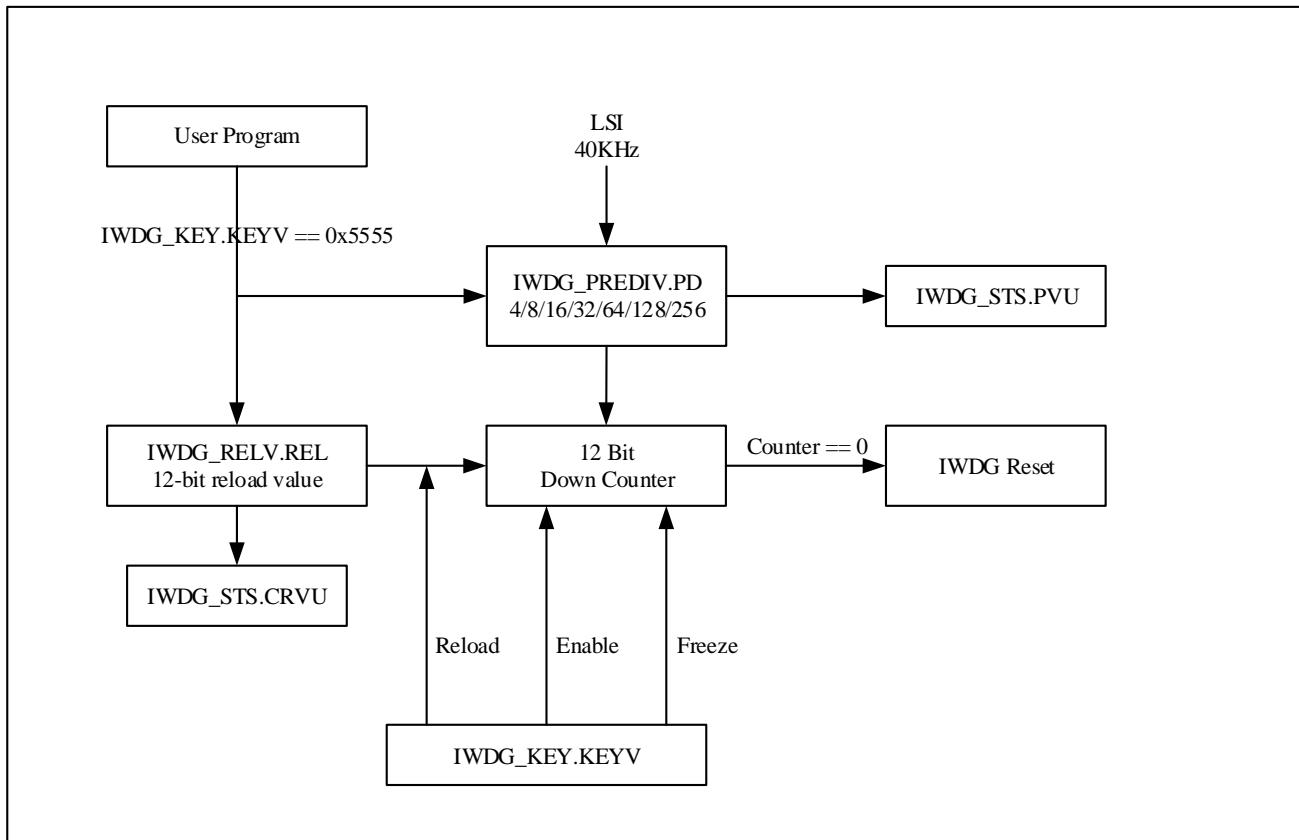
IWDG can be configured through APB1 bus with clock frequency 32 MHz. However, IWDG is working mainly in LSI clock and in Backup power domain (BKP) with VDD supply voltage.

13.2 Main features

- Independent 12-bit down-counter
- RC oscillator provides independent clock source, which can also operate in SLEEP/STOP0/STOP2/STANDBY modes.
- Reset and low-power wake-up can be matched.
- A system reset (if watchdog activated) occurs when the down counter reaches 0x000.

13.3 Function description

Figure 13-1 Functional block diagram of the independent watchdog module



Note: Watchdog function is in V_{DD} power supply area, and it can still work normally in SLEEP/STOP0/STOP2/STANDBY modes.

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 generate 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 prescaler 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 prescaler value and/or reload value is updating. After the prescaler 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-M4 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'. See the chapter on debugging module for details 21.3.2.

13.3.3 IWDG Freeze

Once IWDG enables (no matter from hardware or software), unless generate system reset or configure run-time freeze by writing 0x4567 to IWDG_KEY.KEYV[15:0] bits, IWDG would not stop counting. User can also configure IWDG freeze under certain working mode. IWDG provides freeze option in sleep, stop0, stop2 and standby mode. When IWDG is on, it will force LSI clock to be on.

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). Corresponding definition, width, address, read write access and reset information is shown in following table:

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 prescaler value, it must wait until the IWDG_STS.CRVU bit is reset before changing the reload value, the same as changing the prescaler 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 IWDG counting maximum and minimum reset time

Pre-scale factor	PD[2:0]	Minimum (ms) RL[11:0]=0		Maximum (ms) RL[11:0]=0xFFFF
		RL[11:0]=0xFFFF		
/4	000	0.1		409.6
/8	001	0.2		819.2
/16	010	0.4		1638.4
/32	011	0.8		3276.8
/64	100	1.6		6553.6
/128	101	3.2		13107.2
/256	11x	6.4		26214.4

13.4.2 IWDG configuration flow

Software flow:

1. Write 0x5555 to IWDG_KEY.KEYV[15:0] bits to enable write access of IWDG_PREDIV and IWDG_RELV registers.
2. Check IWDG_STS.PVU bit or IWDG_STS.CRVU bit, if they are 0, continue next step.
3. Configure IWDG_PREDIV.PD[2:0] bits to select pre-scale value.
4. Configure IWDG_RELV.REL[11:0] bits reload value.
5. Writing 0xAAAA to IWDG_KEY.KEYV[15:0] bits to upload counter with reload value.
6. Enable watchdog by software or hardware writing 0xCCCC to IWDG_KEY.KEYV[15:0] bits.

If user wants change pre-scale and reload value, repeat step 1~5. If not, just feed the dog with step 5.

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]																					
	Reset value											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]		

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			
	Reset value																												0	0	0					
0x08	IWDG_RELV	Reserved																								REL[11:0]										
	Reset value	1	1	1	1	1	1	1	1	1	1																									
0x0C	IWDG_STS	Reserved																																	CRVU	EVU
	Reset value																																		0	0

13.5.2 IWDG Key Register (IWDG_KEY)

Address offset : 0x00

Reset value : 0x00000000

31																														16		
Reserved																																
15																														0		
KEYV[15:0]																																
rc_w0																																

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 0x4567: Run time freeze for IWDG, stop IWDG when run 0x89AB: Restore from run time freeze.

13.5.3 IWDG Pre-Scaler Register (IWDG_PREDIV)

Address offset : 0x04

Reset value : 0x00000000

31																														16		
Reserved																																
15																														0		
Reserved																																
PD[2:0]																																
rw																																

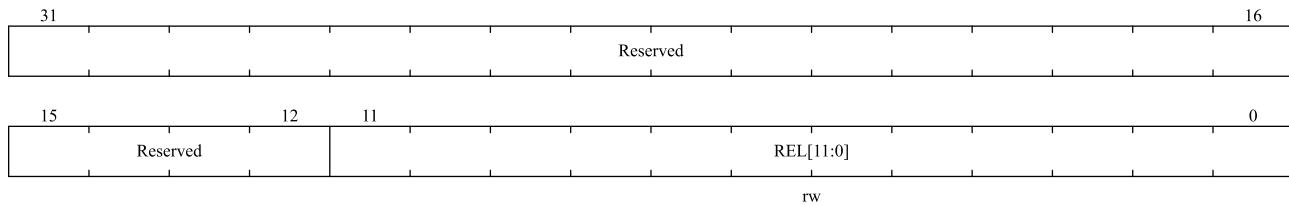
Bit field	Name	Description
31:3	Reserved	Reserved, the reset value must be maintained.

Bit field	Name	Description
2:0	PD[2:0]	<p>Pre-frequency division factor</p> <p>Prescaler 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:</p> <ul style="list-style-type: none"> 000 : divider /4 001 : divider /8 010 : divider /16 011 : divider /32 100 : divider /64 101 : divider /128 11x : divider /256 <p><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></p>

13.5.4 IWDG Reload Register (IWDG_RELV)

Address offset : 0x08

Reset value : 0x00000FFF

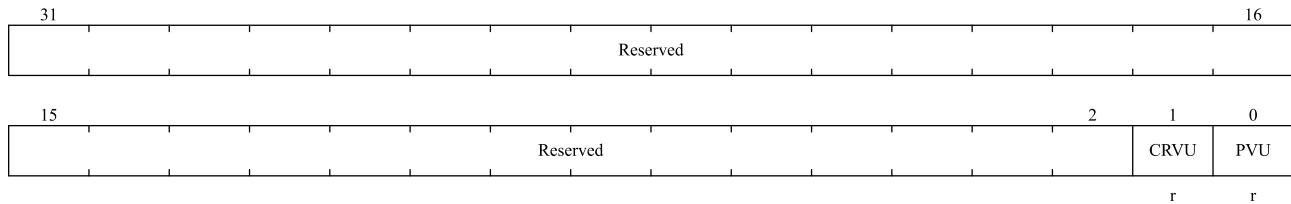


Bit field	Name	Description
31:12	Reserved	Reserved, the reset value must be maintained.
11:0	REL[11:0]	<p>Watchdog counter reload value.</p> <p>With write protection. Defines the reload value of the watchdog counter, which is loaded to the counter every time 0xAAAA is written to IWDG_KEY.KEYV[15:0] bits. The counter then starts to count down from this value. The watchdog timeout period can be calculated from this reloading value and the clock prescaler 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 prescaler value update</p> <p>Prescaler Value Update: this bit indicates an update of prescaler 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 32MHz 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 counter value changes from 0x40 to 0x3F.

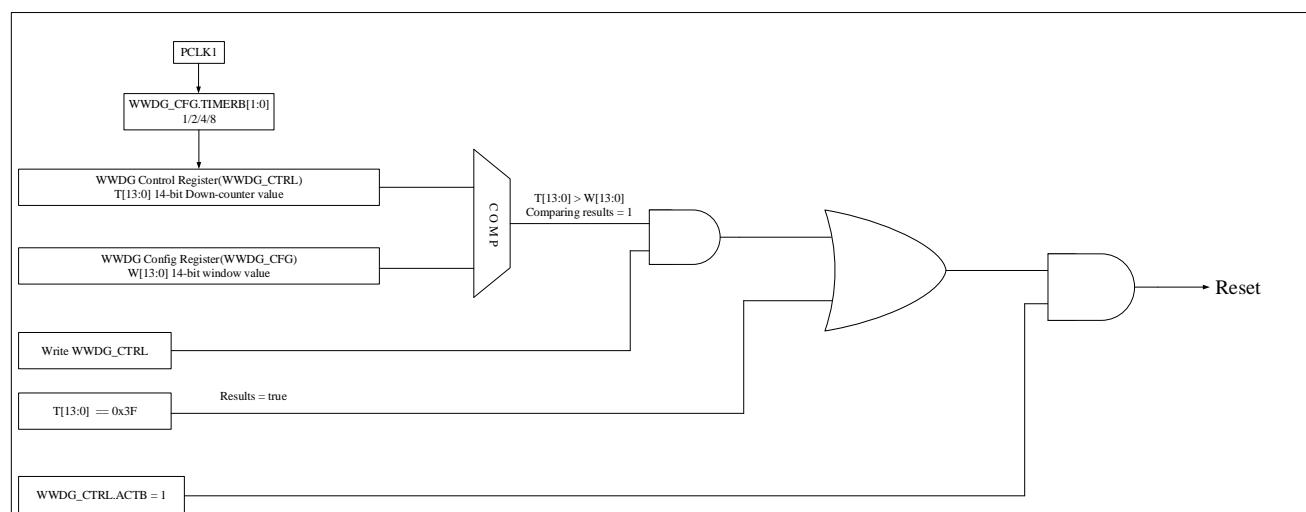
14.2 Main features

- 14-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 14-bit (WWDG_CTRL.T[13:0]) down-counter reaches 0x3F, 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 14-bit down-counter runs independently, and the counter keeps counting down whether WWDG is

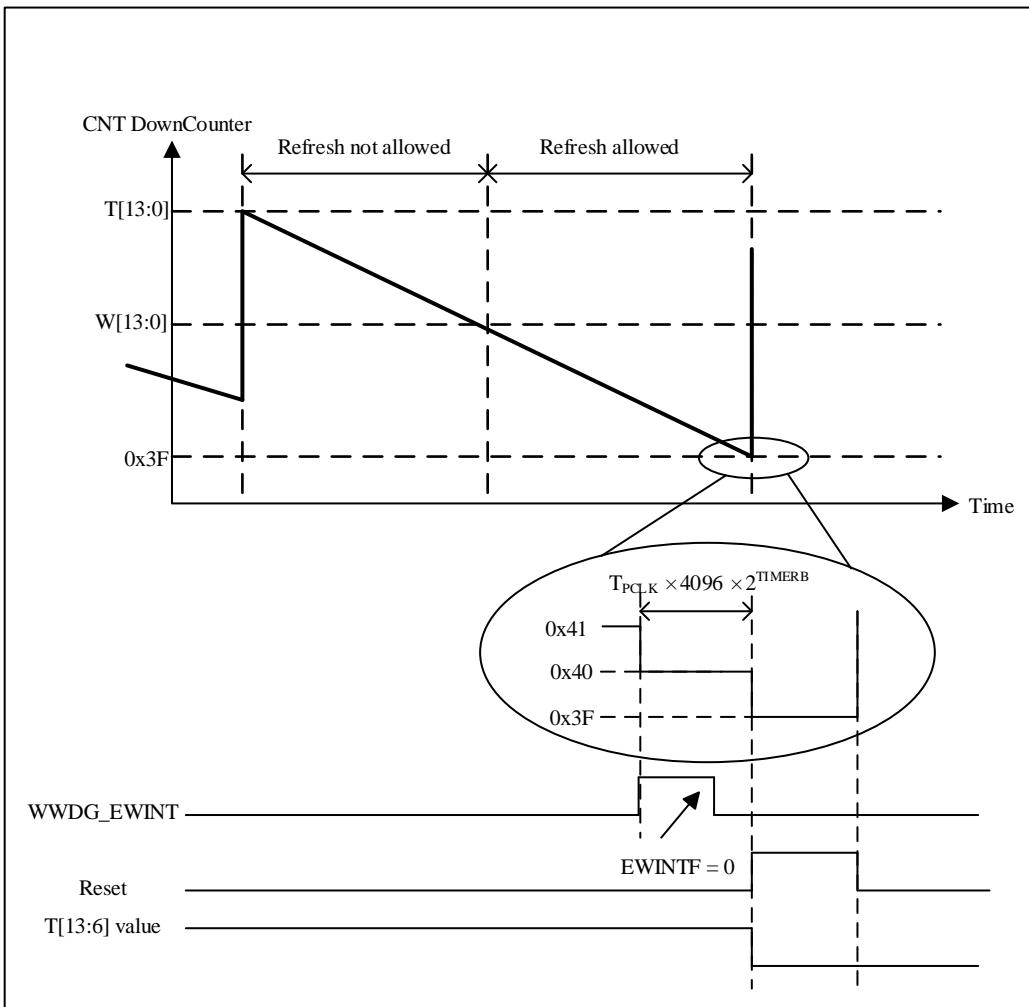
enabled or not. Therefore, need to set any of higher 8 MSB bit (WWDG_CTRL.T [13:6]) to 1, preventing reset right after enable. The prescaler value set by the clock APB1 and WWDG_CFG.TIMERB[1:0] bits determine the decrement speed of the counter. WWDG_CFG.W[13:0] bits set the upper limit of the window.

When the down-counter is refreshed before reaching the window register value or after the counter value changes from 0x40 to 0x3F, 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[13:0] value (maximum value 0x3FFF) and 0x3F, refresh outside this window will generates reset request to MCU. Counter count down from 0x3FFF to 0x3F using scaled

APB1 clock, the maximum counting time and minimum counting time is shown in Table 14-1 (assuming APB1 clock 32 MHz) with calculate equation:

$$T_{WWDG} = T_{PCLK1} \times 4096 \times 2^{TIMERB} \times (T[13:0] - 0X3F + 1)$$

In which:

T_{WWDG} :WWDG timeout

T_{PCLK1} :APB1 clock interval in ms

Minimum-maximum timeout value at PCLK1 = 32MHz

Table 14-1 Maximum and minimum counting time of WWDG

TIMERB	Maximum counting (ms)	Minimum counting (ms)
0	2089	0.128
1	4178	0.256
2	8356	0.512
3	16712	1.024

14.5 Debug mode

In debug mode (Cortex-M4 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'. See the chapter on debugging module for details 21.3.2.

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[15:14] bits to configure pre-scale factor for WWDG.
- 3) Software configure WWDG_CTRL.T [13:0] bits, setting starting value of counter. Need to set any of higher 8 MSB bit (WWDG_CTRL.T [13:6]) to 1, preventing reset right after enable.
- 4) Configure WWDG_CFG.W[13:0] bits to configure upper boundary window value;
- 5) Setting WWDG_CTRL.ACTB[14] bit to enable WWDG;
- 6) Software operates WWDG_STS. EWINTF[0] bit to clear wake-up interrupt flag;
- 7) Configure WWDG_CFG. EWINT[16] bit to enable early wake-up interrupt.

14.7 WWDG registers

14.7.1 WWDG register Map

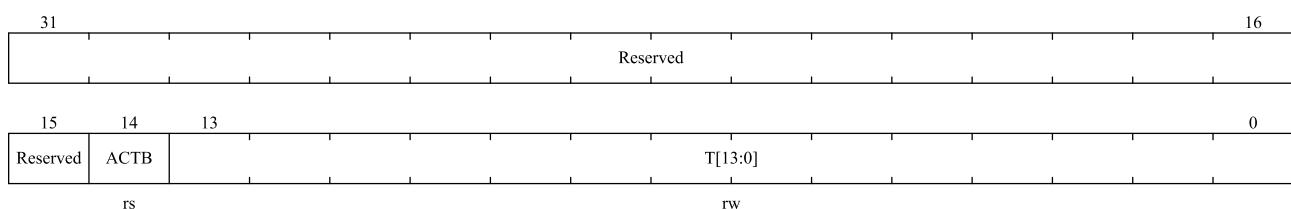
Table 14-2 WWDG register overview

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			
0x00	WWDG_CTRL																																			
	Reset value																		0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	
0x04	WWDG_CFG																																			
	Reset value																		0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
0x08	WWDG_STS																																			
	Reset value																																			

14.7.2 WWDG control register (WWDG_CTRL)

Address offset : 0x00

Reset value : 0x00003FFF

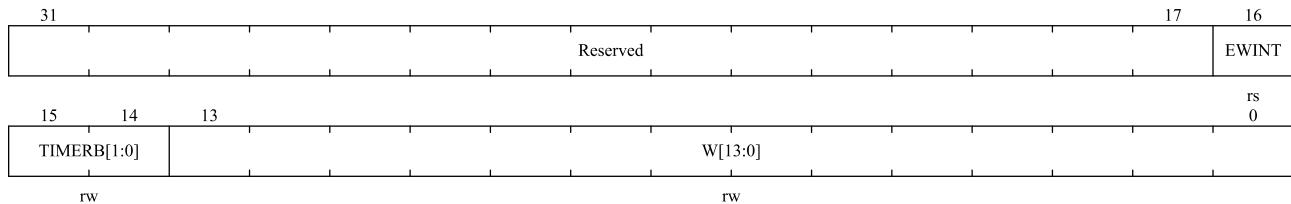


Bit field	Name	Description
31:15	Reserved	Reserved, the reset value must be maintained.
14	ACTB	Activation bit When ACTB=1, the watchdog can generate a reset. This bit is set by software and only cleared by hardware after a reset. When ACTB = 1, the watchdog can generate a reset. 0: Disable watchdog 1: Enable watchdog
13:0	T[13:0]	These bits contain the value of the watchdog counter. It is decremented every $(4096 \times 2^{\text{TIMERB}})$ PCLK1 cycles. A reset is produced when it rolls over from 0x40 to 0x3F.

14.7.3 WWDG config register (WWDG_CFG)

Address offset : 0x04

Reset value : 0x00003FFF

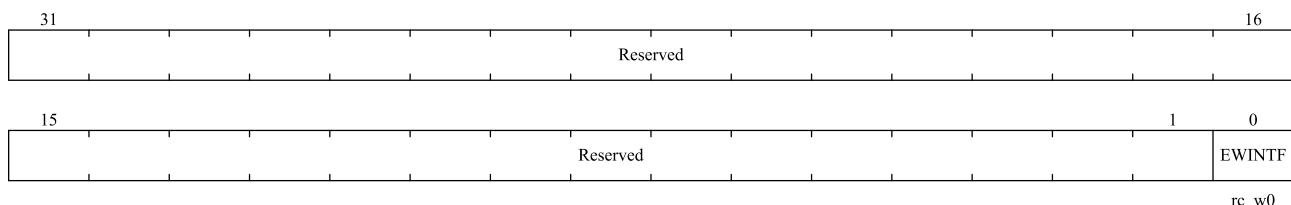


Bit field	Name	Description
31:17	Reserved	Reserved, the reset value must be maintained.
16	EWINT	Early wake-up interrupt When set, an interrupt occurs whenever the counter reaches the value 0x40. This interrupt is only cleared by hardware after a reset.
15:14	TIMERB[1:0]	Timer base. The time base of the prescaler can be modified as follows: 00: CK Counter Clock (PCLK1 div 4096) div 1 01: CK Counter Clock (PCLK1 div 4096) div 2 10: CK Counter Clock (PCLK1 div 4096) div 4 11: CK Counter Clock (PCLK1 div 4096) div 8
13:0	W[13:0]	14-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



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, 12 channels, measuring 9 external and 3 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 72MHz.

15.2 Main features

- Supports 1 ADC, supports single-ended and differential inputs, and can measure up to 9 external and 3 internal sources
- Support 12-bit, 10-bit, 8-bit, 6-bit resolution configurable. The highest sampling rate is 4.2MSPS at 12bit resolution
- 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.
 - ◆ PLL can be configured as a sampling clock source, up to 72MHz, support frequency division 1,2,4,6,8,10,12,16,32,64,128,256
 - ◆ The AHB_CLK can be configured as the sampling clock source, up to 72MHz, and supports frequency division 1,2,4,6,8,10,12,16,32
 - ◆ The timing clock is used for internal timing functions and the frequency must be configured to 1MHz
- Supports timer trigger ADC sampling
- Programmable channel sampling interval
- Support auto scan mode
- Support 2 conversion modes
 - ◆ Single conversion
 - ◆ Continuous conversion
- Support discontinuous mode
- Support self-calibration
- 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 3.6V
- ADC input voltage range: $V_{REF-} \leq V_{IN} \leq V_{REF+}$

15.3 Function Description

Figure 15-1 is a functional block diagram of an ADC.

Figure 15-1 Block diagram of a single ADC

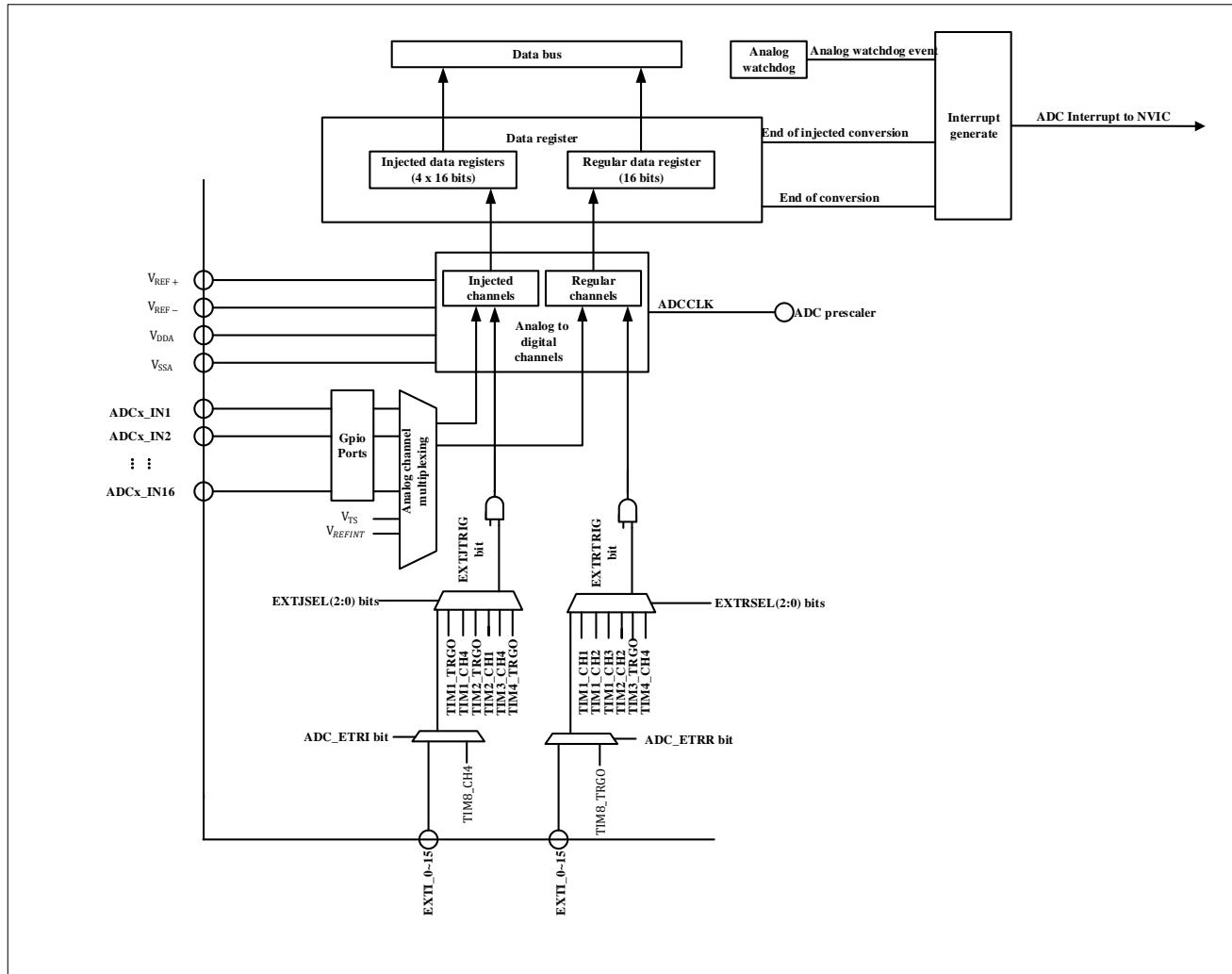


Table 15-1 ADC pins

Name	Signal types	Annotations
$V_{DDA}(1)$	Input, analog power supply	Equivalent to V_{DD} analog power supply and: $2.4V \leq V_{DDA} \leq V_{DD}(3.6V)$
$V_{SSA}(1)$	Input, analog power supply ground	Equivalent to V_{SS} Analog power supply ground, V_{SSA} connect to V_{SS} in

		chip
V_{REF+}	Input, analog reference positive	High/positive reference voltage used by ADC, $2.4V \leq V_{REF+} \leq V_{DDA}$
V_{REF-}	Input, analog reference negative	The low/negative reference voltage used by the ADC, $V_{REF-} = V_{SSA}$
ADCx_IN[16:1]	Analog input signal	9 analog external input channels

1. V_{DDA} and V_{SSA} . They should be separately connected to V_{DD} and V_{SS} .
2. If there is a V_{REF-} Pins (depending on the package), it must be connected to V_{SSA} .
3. If there are no V_{REF+} pins (depending on the package), try to ensure that the voltage values of V_{DDA} and V_{DD} are the same, otherwise the ADC accuracy will be affected.

15.3.1 ADC clock

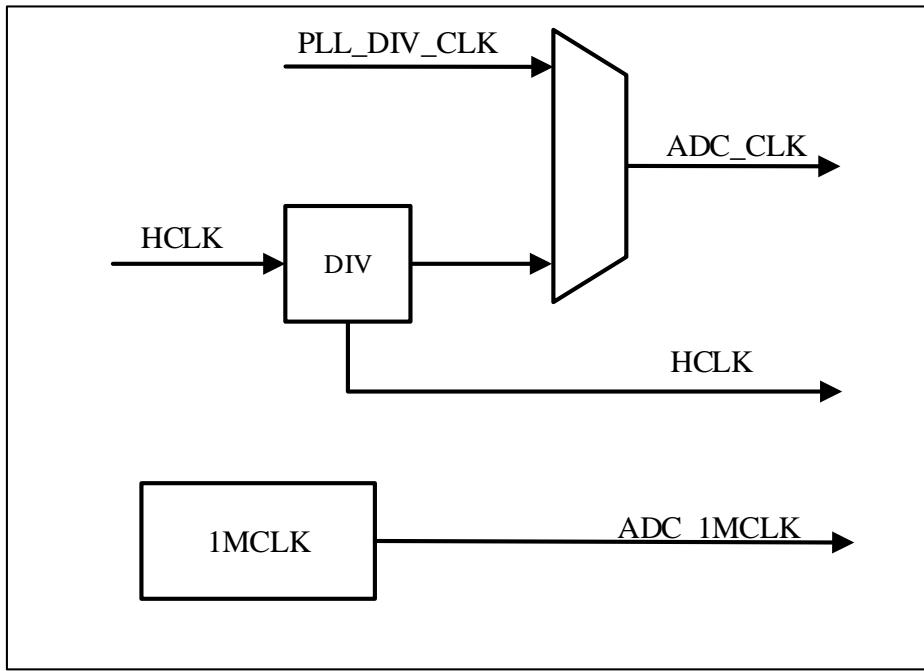
An ADC requires three clocks, HCLK, ADC_CLK and ADC_1MCLK.

- 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 divider are synchronous clock, while PLL divider and system divider are asynchronous clock. 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 72 MHz, support frequency division 1,2,4,6,8,10,12,16,32, 64,128,256
2. The AHB_CLK frequency division can be configured as a working clock up to 72MHz. The AHB_CLK frequency division can be 1,2,4,6,8,10,12,16,32
3. When switching the ADC 1M clock source, you need to ensure that the HSI clock is turned on

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.ON bit and placing the ADC in power-off mode. In this mode, the ADC consumes almost no power (just a few μ A). Power-down can be checked by polling the ADC_CTRL3.PDRDY bit.

When the ADC is disabled, the default mode is power-down. In this mode, as long as the power is on, there is no need to re-calibrate, and the calibration value is automatically maintained in the ADC. To further reduce power consumption, the ADC has a deep sleep mode. When ADC Disable is in deep sleep mode, the calibration value inside the ADC is lost and needs to be recalibrated. Deep sleep saves about 0.2 μ A of power consumption.

15.3.3 Channel selection

Each channel can be configured as a regular sequence and an injection sequence.

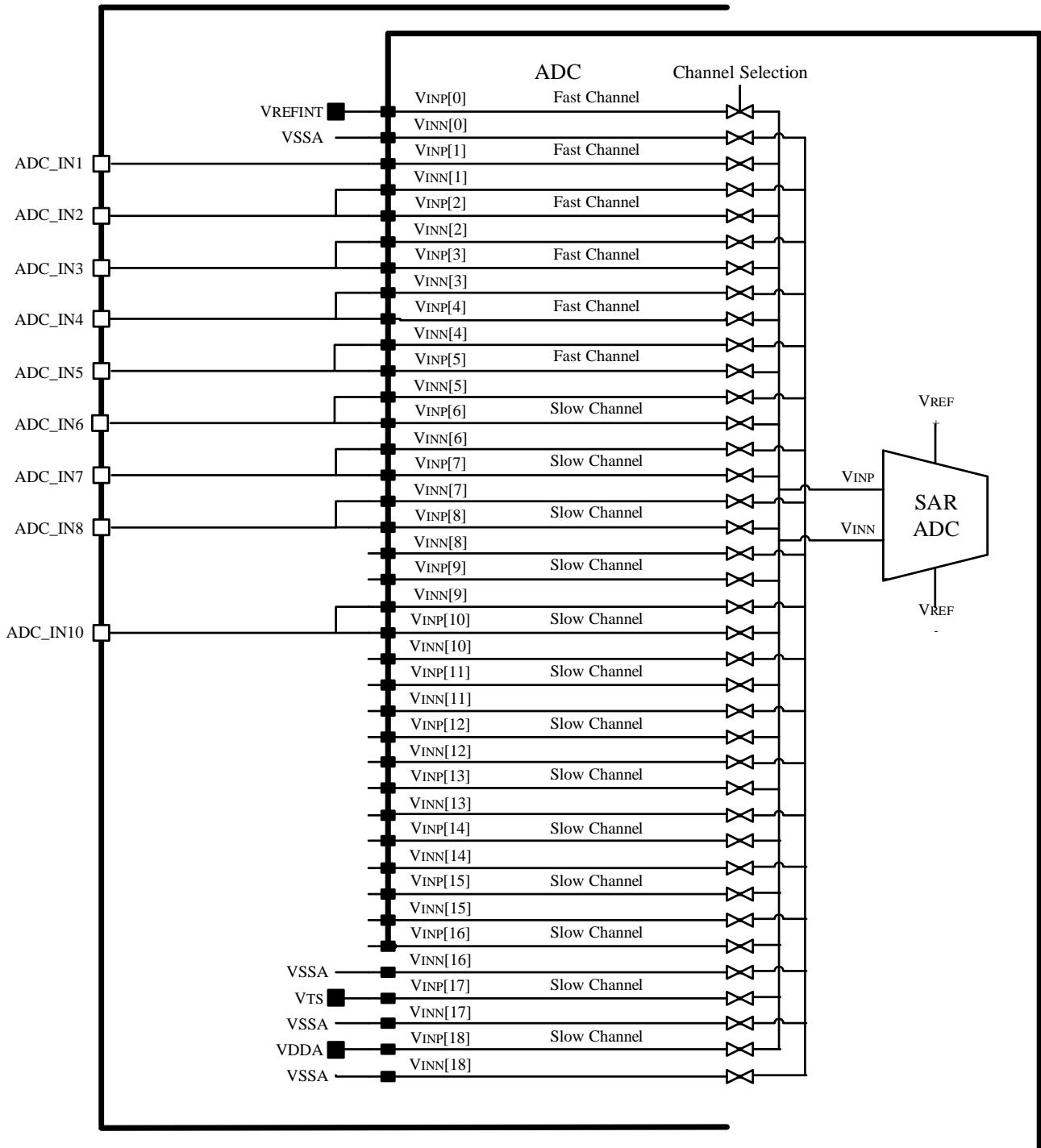
Injection sequence consists of multiple conversions, up to a maximum of 4. The ADC_JSEQ register specifies the injection channel and the conversion order of the injection channel. The ADC_JSEQ.JLEN[21:20] bits specified injection sequence length.

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.

Figure 15-3 ADC channels and Pin connections



15.3.4 Internal channel

- The temperature sensor connects to channel ADC_IN17
- VDDA connects to channel ADC_IN18
- V_{REFINT} connects to ADC_IN0

Internal channels can be converted by injection or regular channels.

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(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(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

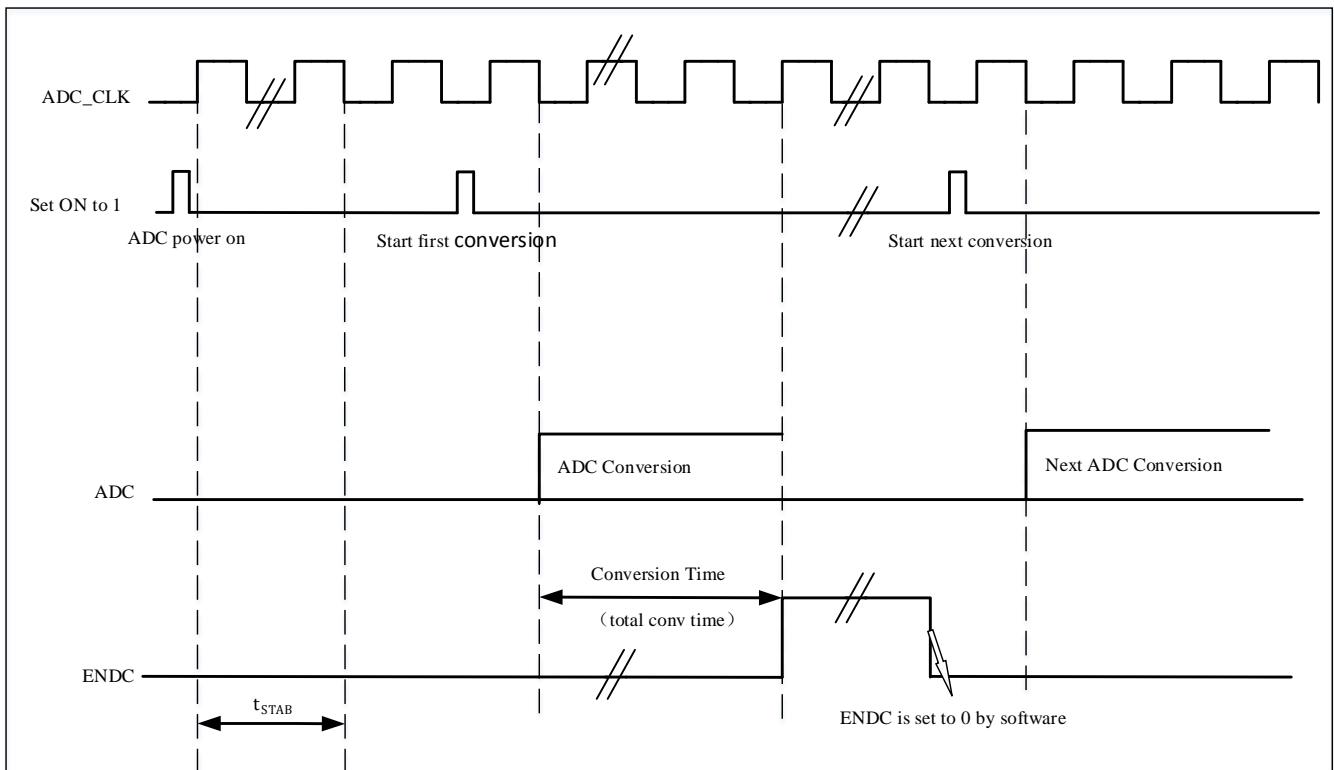
The ADC can enter the continuous conversion mode by configuring ADC_CTRL2.CTU to 1. In this mode, external triggering or setting ADC_CTRL2.ON to 1 can start the ADC to start conversion, and the ADC will continuously convert the selected channel. Continuous mode is only valid for regular channels, not 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 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(tSTAB) 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 15-4 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 conversion value of the ADC is related to the comparison of the thresholds is done before it is done. 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.AWDGSGLEN and ADC_CTRL1.AWDGCH[4:0].

Table 15-2 Analog watchdog channel selection

Scenario that simulates a watchdog	ADC_CTRL1 register control bit		
	AWDGSGLEN a	AWDGERCH a	AWDGEJCH a
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 ⁽¹⁾ Injection channel	1	0	1
A single ⁽¹⁾ Regulars of the channel	1	1	0

A single ⁽¹⁾ Injection or regular channels	1	1	1
---	---	---	---

(1) Select by ADC_CTRL1.AWDGCH[4:0] bit

15.3.9 Scanning 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 regular or Injected 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 regular channels is completed. Injected channel does not support continuous mode. 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 cover 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.

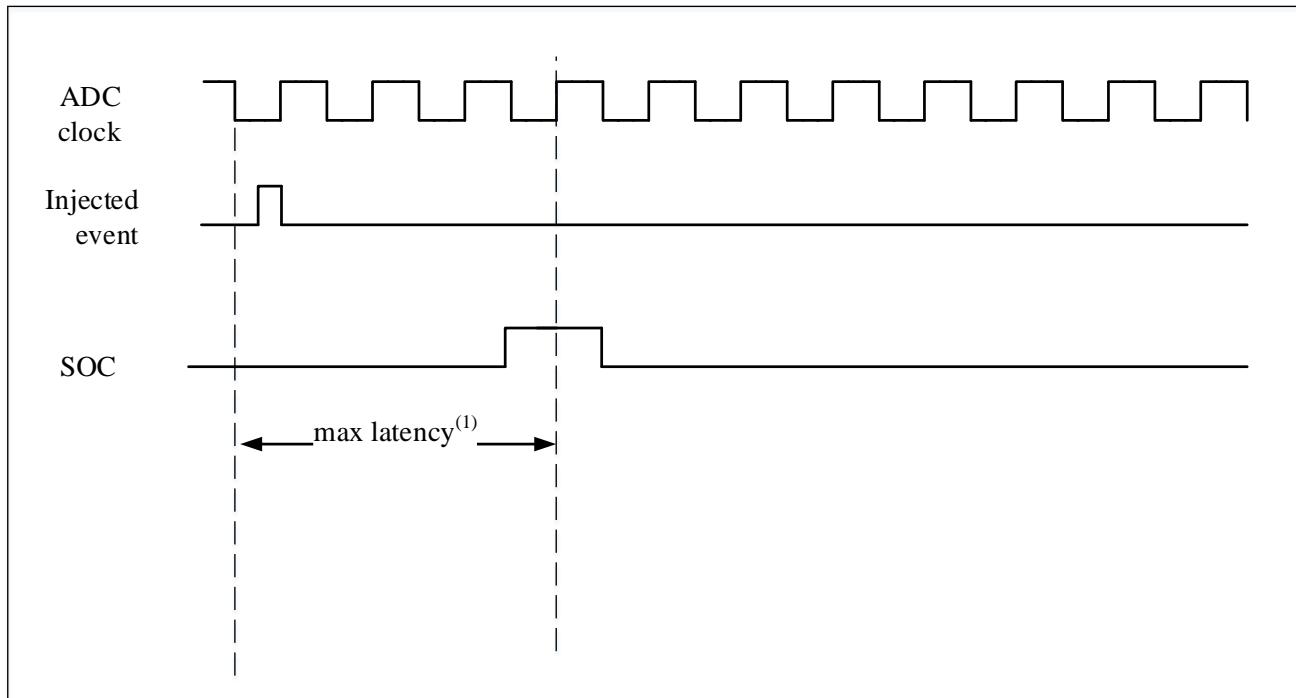
When the ADC clock prescale factor is 2, there is a delay of two ADC clock intervals when the conversion sequence changes from regular to injection or injection to regular. When the ADC clock prescale factor is 4 to 8, there is a delay of one ADC clock intervals when the conversion sequence changes from 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 15-5 Injection conversion delay



(1) 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.

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. Next trigger 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. Next trigger 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.

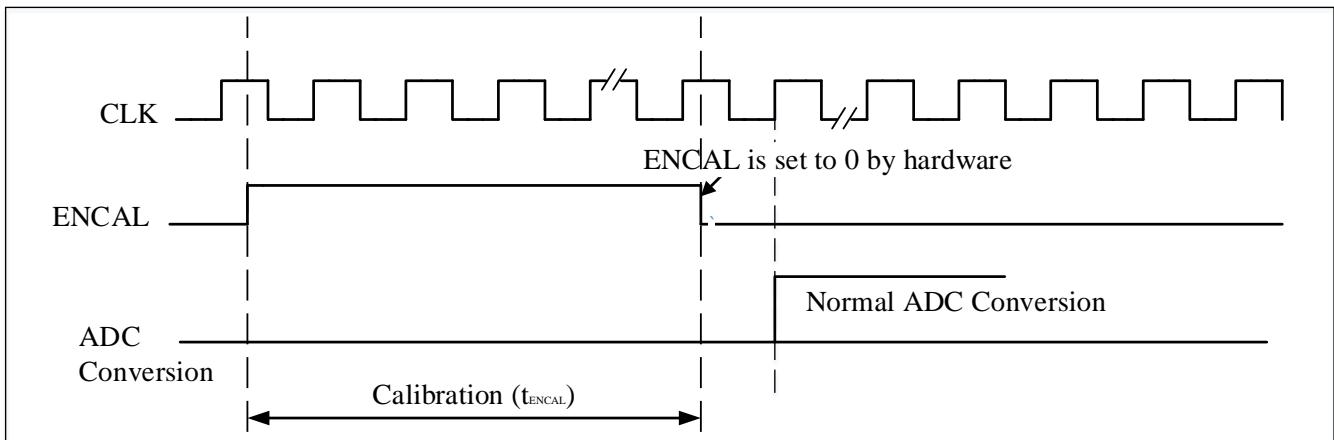
15.4 Calibration

In order to reduce the error, the ADC will have a built-in self-calibration mechanism. Before the A/D conversion, this self-calibration mechanism is used to calculate a calibration factor on each capacitor. Errors due to changes in the internal capacitor bank during conversion are eliminated by this calibration factor. The application program sets the ADC_CTRL2.ENCAL bit to 1 to start self-calibration. During the calibration, the ADC_CTRL2.ENCAL bit remains 1. After the calibration, the ADC_CTRL2.ENCAL bit is cleared by hardware, and then the A/D conversion starts.

There are two points to note when using the self-calibration mechanism:

- It is recommended to perform a calibration after each power-on. If the ADC has been converted and is in continuous conversion mode, the calibration operation cannot be completed.
- The default is single-end calibration, and for differential automatic calibration, you must set ADC_CTRL3.CALDIF to 1. Then write 1 to ADC_CTRL2.ENCAL bit and wait for calibration to complete (ADC_CTRL2.ENCAL bit will clear 0 automatically after calibration)

Figure 15-6 Calibration sequence diagram



15.5 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

(12bit resolution)

SYM	SYM	SYM	SYM	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
-----	-----	-----	-----	-----	-----	----	----	----	----	----	----	----	----	----	----

(10bit resolution)

SYM	SYM	SYM	SYM	SYM	SYM	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
-----	-----	-----	-----	-----	-----	----	----	----	----	----	----	----	----	----	----

(8bit resolution)

SYM	D7	D6	D5	D4	D3	D2	D1	D0							
-----	-----	-----	-----	-----	-----	-----	-----	----	----	----	----	----	----	----	----

(6bit resolution)

SYM	D5	D4	D3	D2	D1	D0									
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	----	----	----	----	----	----

The regular sequence

(12bit resolution)

0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
---	---	---	---	-----	-----	----	----	----	----	----	----	----	----	----	----

(10bit resolution)

0	0	0	0	0	0	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----

(8bit resolution)

0	0	0	0	0	0	0	0	D7	D6	D5	D4	D3	D2	D1	D0
---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----

(6bit resolution)

0	0	0	0	0	0	0	0	0	0	D5	D4	D3	D2	D1	D0
---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----

Table 15-4 Left-aligne data

Injection sequence

(12bit resolution)

SYM	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0
-----	-----	-----	----	----	----	----	----	----	----	----	----	----	---	---	---

(10bit resolution)

SYM	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0	0
-----	----	----	----	----	----	----	----	----	----	----	---	---	---	---	---

(8bit resolution)

SYM	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0	0	0	0
-----	----	----	----	----	----	----	----	----	---	---	---	---	---	---	---

(6bit resolution)

SYM	D5	D4	D3	D2	D1	D0	0	0	0	0	0	0	0	0	0	0
-----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

The regular sequence

(12bit resolution)

D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0	0
-----	-----	----	----	----	----	----	----	----	----	----	----	---	---	---	---	---

(10bit resolution)

D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0	0	0	0
----	----	----	----	----	----	----	----	----	----	---	---	---	---	---	---	---

(8bit resolution)

D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0	0	0	0	0	0
----	----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---

(6bit resolution)

D5	D4	D3	D2	D1	D0	0	0	0	0	0	0	0	0	0	0	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---	---

15.6 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:

$$T_{CONV} = \text{Sampling time} + x \text{ cycles}$$

Note: x at the resolution of 6/8/10/12 bit corresponds to 6.5/8.5/10.5/12.5.

Example:

ADCCLK=72MHz, the sampling time is 4.5 cycles and resolution is 12bit, the total conversion time is "4.5 + 12.5" ADCCLK Cycles, that is:

$$T_{CONV} = 4.5 + 12.5 = 17 \text{ cycle} = 0.2361\mu\text{s}$$

15.7 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_RMP_CFG.ADC_ETRR and AFIO_RMP_CFG.ADC_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

EXTRSEL[2:0]	Trigger source	Type
000	TIM1_CC1 event	Internal signal from the on-chip timer

EXTSEL[2:0]	Trigger source	Type
001	TIM1_CC2 event	
010	TIM1_CC3 event	
011	TIM2_CC2 event	
100	TIM3_TRGO event	
101	TIM4_CC4 event	
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_RMP_CFG.ADC_ETRI and AFIO_RMP_CFG.ADC_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	
001	TIM1_CC4 event	
010	TIM2_TRGO event	
011	TIM2_CC1 event	
100	TIM3_CC4 event	
101	TIM4_TRGO event	
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.

15.8 DMA requests

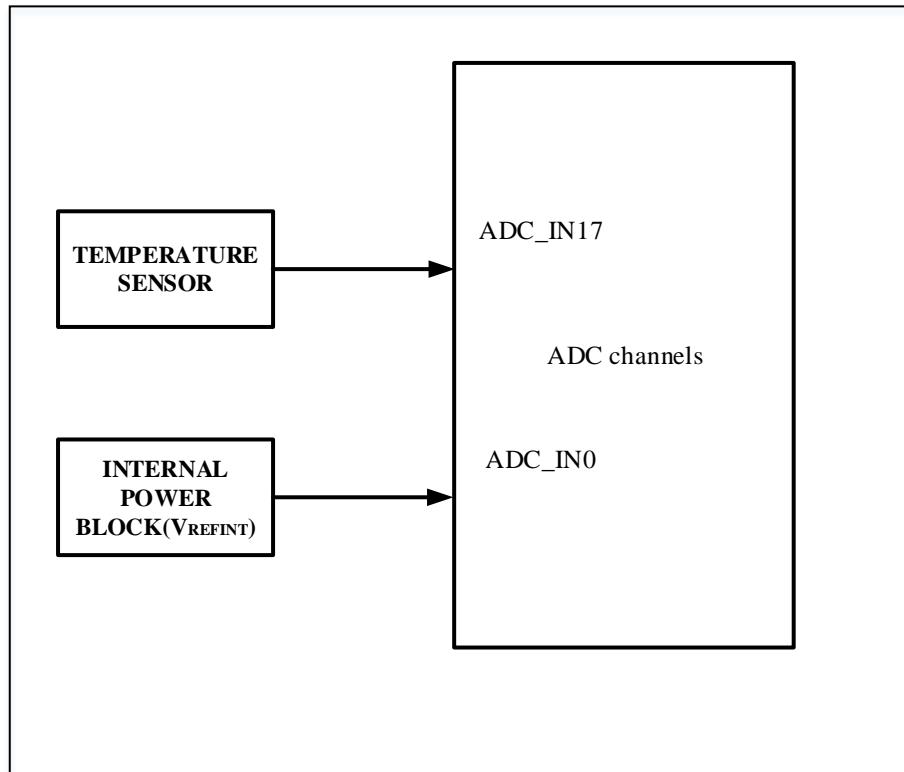
In order to avoid excessive data load when converting multiple regular channels, resulting in errors in the regular channel conversion result stored in the ADC_DAT register, you can set the ADC_CTRL2.ENDMA bit to 1 to use the DMA mode. When the ADC regular channel conversion ends, it will be a DMA request 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.

15.9 Temperature sensor

Set the ADC_CTR2.TEMPEN bit to 1, enable the temperature sensor and VREFINT, 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_IN17 channel. When the temperature sensor is working, the ideal sampling time is 17.1us; when the temperature sensor is not working, the ADC_CTR2.TEMPEN bit can be cleared by software to reduce power consumption. Figure 15-7 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 15-7 Temperature sensor and VREFINT Diagram of the channel



15.9.1 Temperature sensor using flow

- 1) Configure the channel (ADC_IN17) and sampling time of the channel to be 17.1 us
- 2) Set ADC_CTRL2.TEMPEN bit to 1 to enable temperature sensor and V_{REFINT}
- 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:

$$\text{Temperature (}^{\circ}\text{C) = \{ (V_{Temperature} - V_{SENSE}) / Avg_Slope \} + Temperature-T_{offset}}$$

In which:

V_{Temperature} = Temperature corresponds to V_{SENSE}

T_{offset} = 1.25 °C

Temperature is the calibration temperature

Avg_Slope = temperature and V_{SENSE} Average slope of a curve (mV/°C or μV/°C)

Refer to the Electrical characteristics chapter in the datasheet for actual Avg_Slope value.

Note: There is a settling time before the sensor wakes up from the power-off mode to the correct output of VSENSE; 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.

15.10 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 or injection sequence conversion is complete	ENDC	ENDCIEN
The injection sequence conversion is complete	JENDC	JENDCIEN
The simulated watchdog status bit is set	AWDG	AWDGIEN
Any regular channel interruption is enabled	ENDCA	ENDCAIEN
Any injection channel interruption is enabled	JENDCA	JENDCAIEN

15.11 ADC registers

Refer to Section 1. 1 for some abbreviations used in register descriptions.

These peripheral registers must be operated in word (32-bit) mode.

15.11.1 ADC register overview

Table 15-8 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																																	
	Reset Value																																	
014h	ADC_JOFFSET1																																	
	Reset Value																																	
018h	ADC_JOFFSET2																																	
	Reset Value																																	
01Ch	ADC_JOFFSET3																																	
	Reset Value																																	
020h	ADC_JOFFSET4																																	
	Reset Value																																	
024h	ADC_WDGHIGH																																	
	Reset Value																																	
028h	ADC_WDGLOW																																	
	Reset Value																																	
02Ch	ADC_RSEQ1																																	
	Reset Value																																	
030h	ADC_RSEQ2																																	
	Reset Value																																	
034h	ADC_RSEQ3																																	
	Reset Value																																	
038h	ADC_JSEQ																																	
	Reset Value																																	
03Ch	ADC_JDAT1																																	
	Reset Value																																	
040h	ADC_JDAT2																																	
	Reset Value																																	
044h	ADC_IDAT3																																	
	Reset Value																																	
048h	ADC_IDAT4																																	
	Reset Value																																	
04Ch	ADC_DAT																																	
	Reset Value																																	
050h	ADC_DIFSEL																																	
	Reset Value																																	

054h	ADC_CALFACT	Reserved	CALFACTD[6:0]							Reserved							CALFACTS[6:0]							
			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
058h	ADC_CTRL3	Reserved								VABTMEN	DPWMOD	JENDCAEN	ENDCAEN	BPCAL	PDRDY	RDY	CKMOD	CALALD	CALDIF	RES[1:0]				
			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
05Ch	ADC_SAMPT3	Reserved								SAMPSEL								SAMP[2:0]						
			0	0	0	0	0	0	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

31	Reserved	16
15	Reserved	0
7	JENDCA	rc_w0
6	ENDCA	rc_w0
5	STR	rc_w0
4	JSTR	rc_w0
3	JENDC	rc_w0
2	ENDC	rc_w0
1	AWDG	rc_w0

Bit field	Name	Description
31:7	Reserved	Reserved, the reset value must be maintained
6	JENDCA	<p>Any injected channel end of conversion flag</p> <p>This bit is set by hardware at the end of any injection channel conversion and cleared by software.</p> <p>0: Conversion is not complete; 1: Conversion is complete.</p>
5	ENDCA	<p>Any regular channel end of conversion flag</p> <p>This bit is set by hardware at the end of any channel (regular or injection) conversion and cleared by software.</p> <p>0: Conversion is not complete; 1: Conversion is complete.</p>
4	STR	<p>Regular channel start flag</p> <p>This bit is set by hardware at the start of regular channel conversion and cleared by software.</p> <p>0: Regular channel conversion has not started. 1: Regular channel conversion has started.</p>
3	JSTR	<p>Injected channel start flag</p> <p>This bit is set by hardware at the start of the injection channel conversion and cleared by software.</p> <p>0: Injection sequence channel conversion has not started. 1: Injection sequence channel conversion has started.</p>
2	JENDC	<p>Injected channel end of conversion</p> <p>This bit is set by hardware at the end of all injection sequence channel conversions and cleared by software</p>

Bit field	Name	Description
		0: Conversion is not complete. 1: Conversion is complete.
1	ENDC	Regular 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	20	19	16
									AWDG ERCH	AWDG EJCH	Reserved	Reserved	DUSEL[3:0]		
15	13	12	11	10	9	8	rw	7	rw	6	5	4	rw	rw	0
DCTU[2:0]	DJCH	DREGCH	AUTOJC	AWDG SGLEN	SCANMD	JENDC IEN	AWD GIEN	ENDCIEI					AWDGCH[4:0]		
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	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.
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.
21:20	Reserved	Reserved, the reset value must be maintained
19:16	DUSEL[3:0]	Dual mode selection Software uses these bits to select modes of operation. 0000: Independent mode 0001-1111: Reserved
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

Bit field	Name	Description
		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[4:0] 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>
7	JENDCIEN	Interrupt enable for injected channels This bit is set and cleared by the software to disallow or allow interrupts after all injection channel conversions have finished. 0: Disable JENDC interruption. 1: Enable JENDC interruption. An interrupt occurred when hardware set ADC_STS.JENDC bit.
6	AWDGIEN	Analog watchdog interrupt enable This bit is set and cleared by software to disallow or allow interrupt generated by simulated watchdog. In scan mode, if the watchdog detects an out-of-range value, the scan is aborted only when that bit is set. 0: Disable simulated watchdog interruption.

Bit field	Name	Description
		1: Enable simulated watchdog interruption.
5	ENDCIEN	<p>Interrupt enable for regular channels</p> <p>This bit is set and cleared by the software to disallow or allow interrupts to occur after the regular or injected channel conversion ends.</p> <p>0: Disable ENDC interruption.</p> <p>1: Enable ENDC interruption. An interrupt occurred when hardware set ADC_STS.ENDC bit.</p>
4:0	AWDGCH[4:0]	<p>Analog watchdog channel select bits</p> <p>These bits are set and cleared by software to select input channels that simulate watchdog protection.</p> <p>00000: ADC analog input channel 0 00001: ADC analog input channel 1 00010: ADC analog input channel 2 00011: ADC analog input channel 3 00100: ADC analog input channel 4 00101: ADC analog input channel 5 00110: ADC analog input channel 6 00111: ADC analog input channel 7 01000: ADC analog input channel 8 01001: Reserved 01010: ADC analog input channel 10 01011: 01100: 01101: 01110: 01111: 10000: Reserved 10001: ADC analog input channel 17 10010: ADC analog input channel 18 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
									TEMPEN	SWSTR RCH	SWSTR JCH	EXT RTRIG	EXTRSEL[2:0]			Reserved
15	14	12	11	10	9	8	rw	7	rw	rw	rw	rw	3	rw	1	0
EXT JTRIG	EXTSEL[2:0]	ALIG	Reserved	ENDMA					Reserved				ENCAL	CTU	ON	
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	TEMPEN	<p>Temperature sensor and V_{REFINT} Enable</p> <p>This bit is set and cleared by the software to enable or disable the temperature sensor and V_{REFINT} Channel.</p> <p>0: Disables the temperature sensor and V_{REFINT}.</p>

Bit field	Name	Description								
		1: Enable the temperature sensor and V _{REFINT} .								
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: indicates the CC4 event of timer 4</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: indicates the CC2 event of timer 2</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: indicates the CC4 event of timer 4	010: indicates the CC3 event of timer 1	110: EXTI line 0~15/TIM8_TRGO event	011: indicates the CC2 event of timer 2	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: indicates the CC4 event of timer 4									
010: indicates the CC3 event of timer 1	110: EXTI line 0~15/TIM8_TRGO event									
011: indicates the CC2 event of timer 2	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>								
14:12	EXTJSEL[2:0]	<p>External event select for injected sequence</p> <p>These bits select the External event used to trigger the injected sequence conversion.</p> <p>The triggering configuration of ADC is as follows</p> <table> <tbody> <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: indicates the TRGO event of timer 4</td> </tr> <tr> <td>010: indicates the TRGO event of timer 2</td> <td>110: EXTI line 0~15/TIM8_CC4 event</td> </tr> <tr> <td>011: indicates the CC1 event of timer 2</td> <td>111: SWSTRJCH</td> </tr> </tbody> </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: indicates the TRGO event of timer 4	010: indicates the TRGO event of timer 2	110: EXTI line 0~15/TIM8_CC4 event	011: indicates the CC1 event of timer 2	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: indicates the TRGO event of timer 4									
010: indicates the TRGO event of timer 2	110: EXTI line 0~15/TIM8_CC4 event									
011: indicates the CC1 event of timer 2	111: SWSTRJCH									

Bit field	Name	Description
11	ALIG	Data alignment This bit is set and cleared by the software. Refer to Table 15-3 and Table 15-4. 0: Right-aligned. 1: Left-aligned.
10:9	Reserved	Reserved, the reset value must be maintained
8	ENDMA	Direct memory access mode This bit is set and cleared by the software. See the DMA Controller chapter for details. 0: Do not use DMA mode. 1: Use DMA mode.
7:3	Reserved	Reserved, the reset value must be maintained
2	ENCAL	A/D calibration This bit is set by software to start calibration and cleared by hardware at the end of calibration. 0: Calibration completed; 1: Starts calibration.
1	CTU	Continuous conversion This bit is set and cleared by the software. If this bit is set, the conversion continues until the bit is cleared. 0: Single conversion mode. 1: Continuous conversion mode.
0	ON	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. 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 15-4. 0: Close ADC conversion/calibration and enter power-down mode. 1: Start ADC and start conversion. 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.

15.11.5 ADC sampling time register 1 (ADC_SAMPT1)

Address offset: 0x0C

Reset value: 0x0000 0000

31					24	23	21	20	18	17	16
Reserved											
15	14	12	11	9	8	rw	6	5	rw	3	rw
SAMP15_0	SAMP14[2:0]	SAMP13[2:0]	SAMP12[2:0]	SAMP11[2:0]	SAMP10[2:0]						
rw	rw	rw	rw	rw	rw						

Bit field	Name	Description
31:24	Reserved	Reserved, the reset value must be maintained

Bit field	Name	Description																
23:0	SAMPx[2:0]	<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> <p>ADC_SAMPT3.SAMPSEL = 0, the sampling time is set as follows:</p> <table> <tbody> <tr><td>000: 1.5 cycles</td><td>100: 41.5 cycles</td></tr> <tr><td>001: 7.5 cycles</td><td>101: 55.5 cycles</td></tr> <tr><td>010: 13.5 cycles</td><td>110: 71.5 cycles</td></tr> <tr><td>011: 28.5 cycles</td><td>111: 239.5 cycles</td></tr> </tbody> </table> <p>ADC_SAMPT3.SAMPSEL = 1, the sampling time is set as follows:</p> <table> <tbody> <tr><td>000: 1.5 cycles</td><td>100: 19.5 cycles</td></tr> <tr><td>001: 2.5 cycles</td><td>101: 61.5 cycles</td></tr> <tr><td>010: 4.5 cycles</td><td>110: 181.5 cycles</td></tr> <tr><td>011: 7.5 cycles</td><td>111: 601.5 cycles</td></tr> </tbody> </table>	000: 1.5 cycles	100: 41.5 cycles	001: 7.5 cycles	101: 55.5 cycles	010: 13.5 cycles	110: 71.5 cycles	011: 28.5 cycles	111: 239.5 cycles	000: 1.5 cycles	100: 19.5 cycles	001: 2.5 cycles	101: 61.5 cycles	010: 4.5 cycles	110: 181.5 cycles	011: 7.5 cycles	111: 601.5 cycles
000: 1.5 cycles	100: 41.5 cycles																	
001: 7.5 cycles	101: 55.5 cycles																	
010: 13.5 cycles	110: 71.5 cycles																	
011: 28.5 cycles	111: 239.5 cycles																	
000: 1.5 cycles	100: 19.5 cycles																	
001: 2.5 cycles	101: 61.5 cycles																	
010: 4.5 cycles	110: 181.5 cycles																	
011: 7.5 cycles	111: 601.5 cycles																	

15.11.6 ADC sampling time register 2 (ADC_SAMPT2)

Address offset: 0x10

Reset value: 0x0000 0000

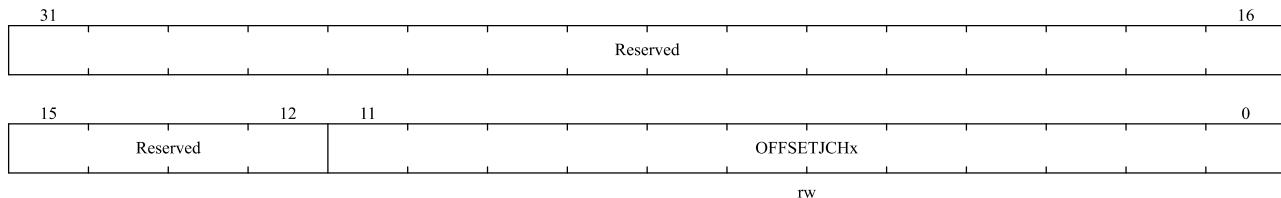
31	30	29	27	26	24	23	21	20	18	17	16
Reserved		SAMP9[2:0]		SAMP8[2:0]		SAMP7[2:0]		SAMP6[2:0]		SAMP5[2:1]	
15	14	rw	12	11	rw	9	8	rw	6	5	rw
SAMP5_0		SAMP4[2:0]		SAMP3[2:0]		SAMP2[2:0]		SAMP1[2:0]		SAMP0[2:0]	
rw		rw		rw		rw		rw		rw	

Bit field	Name	Description																
31:30	Reserved	Reserved, the reset value must be maintained																
29:0	SAMPx[2:0]	<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> <p>ADC_SAMPT3.SAMPSEL = 0, the sampling time is set as follows:</p> <table> <tbody> <tr><td>000: 1.5 cycles</td><td>100: 41.5 cycles</td></tr> <tr><td>001: 7.5 cycles</td><td>101: 55.5 cycles</td></tr> <tr><td>010: 13.5 cycles</td><td>110: 71.5 cycles</td></tr> <tr><td>011: 28.5 cycles</td><td>111: 239.5 cycles</td></tr> </tbody> </table> <p>ADC_SAMPT3.SAMPSEL = 1, the sampling time is set as follows:</p> <table> <tbody> <tr><td>000: 1.5 cycles</td><td>100: 19.5 cycles</td></tr> <tr><td>001: 2.5 cycles</td><td>101: 61.5 cycles</td></tr> <tr><td>010: 4.5 cycles</td><td>110: 181.5 cycles</td></tr> <tr><td>011: 7.5 cycles</td><td>111: 601.5 cycles</td></tr> </tbody> </table>	000: 1.5 cycles	100: 41.5 cycles	001: 7.5 cycles	101: 55.5 cycles	010: 13.5 cycles	110: 71.5 cycles	011: 28.5 cycles	111: 239.5 cycles	000: 1.5 cycles	100: 19.5 cycles	001: 2.5 cycles	101: 61.5 cycles	010: 4.5 cycles	110: 181.5 cycles	011: 7.5 cycles	111: 601.5 cycles
000: 1.5 cycles	100: 41.5 cycles																	
001: 7.5 cycles	101: 55.5 cycles																	
010: 13.5 cycles	110: 71.5 cycles																	
011: 28.5 cycles	111: 239.5 cycles																	
000: 1.5 cycles	100: 19.5 cycles																	
001: 2.5 cycles	101: 61.5 cycles																	
010: 4.5 cycles	110: 181.5 cycles																	
011: 7.5 cycles	111: 601.5 cycles																	

15.11.7 ADC injected channel data offset register x (ADC_JOFFSETx) (x=1...4)

Address offset: 0x14-0x20

Reset value: 0x0000 0000

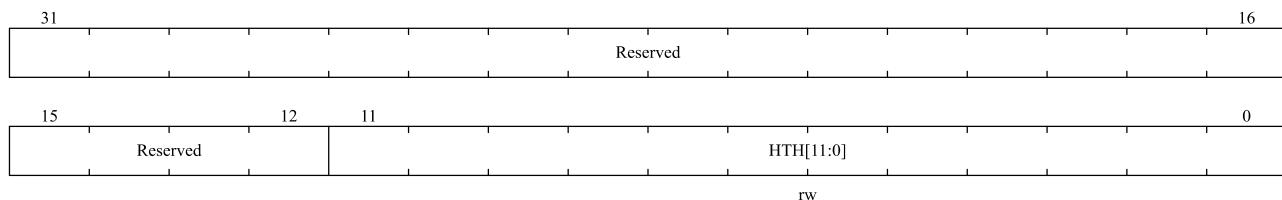


Bit field	Name	Description
31:12	Reserved	Reserved, the reset value must be maintained
11:0	OFFSETJCHx[11:0]	Data offset for injected channel x These bits define the values used to subtract from the original conversion data when the conversion is injected into the channel. The result of the conversion can be read in the ADC_JDATx register.

15.11.8 ADC watchdog high threshold register (ADC_WDGHIGH)

Address offset: 0x24

Reset value: 0x0000 0FFF

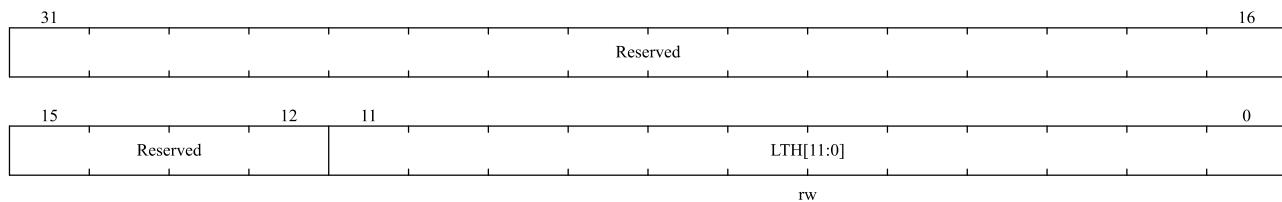


Bit field	Name	Description
31:12	Reserved	Reserved, the reset value must be maintained
11:0	HTH[11:0]	Analog watchdog high threshold These bits define the high thresholds for analog watchdog.

15.11.9 ADC watchdog low threshold register (ADC_WDGLOW)

Address offset: 0x28

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.

15.11.10 ADC regular sequence register 1 (ADC_RSEQ1)

Address offset: 0x2C

Reset value: 0x0000 0000

31	Reserved				24	23	LEN[3:0]		20	19	16	
15	14	10	9	rw	5	4	rw	0	rw	rw	rw	
SEQ16_0	SEQ15[4:0]				SEQ14[4:0]				SEQ13[4:0]			
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
23:20	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
19:15	SEQ16[4:0]	16th conversion in regular sequence These bits are software-defined as the number (0 to 18) of the 16th conversion channel in the conversion sequence.
14:10	SEQ15[4:0]	15th conversion in regular sequence
9:5	SEQ14[4:0]	14th conversion in regular sequence
4:0	SEQ13[4:0]	13th conversion in regular sequence

15.11.11 ADC regular sequence register 2 (ADC_RSEQ2)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	25	24	20	19	16	
Reserved	SEQ12[4:0]				SEQ11[4:0]			
15	14	10	9	rw	5	4	rw	
SEQ10_0	SEQ9[4:0]				SEQ8[4:0]			
rw	rw	rw	rw	rw	rw	rw	rw	

Bit field	Name	Description
31:30	Reserved	Reserved, the reset value must be maintained

Bit field	Name	Description
29:25	SEQ12[4:0]	12th conversion in regular sequence These bits are software-defined as the number (0 to 18) of the 12th conversion channel in the conversion sequence.
24:20	SEQ11[4:0]	11th conversion in regular sequence
19:15	SEQ10[4:0]	10th conversion in regular sequence
14:10	SEQ9[4:0]	9th conversion in regular sequence
9:5	SEQ8[4:0]	8th conversion in regular sequence
4:0	SEQ7[4:0]	7th conversion in regular sequence

15.11.12 ADC regular sequence register 3 (ADC_RSEQ3)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	25	24	20	19	16
Reserved		SEQ6[4:0]		SEQ5[4:0]		SEQ4[4:1]	
15	14		rw	10	9	rw	rw
SEQ4_0		SEQ3[4:0]		SEQ2[4:0]		SEQ1[4:0]	
rw		rw		rw		rw	

Bit field	Name	Description
31:30	Reserved	Reserved, the reset value must be maintained
29:25	SEQ6[4:0]	6th conversion in regular sequence These bits are software-defined as the number (0 to 18) of the 6th transition channel in the conversion sequence.
24:20	SEQ5[4:0]	5th conversion in regular sequence
19:15	SEQ4[4:0]	4th conversion in regular sequence
14:10	SEQ3[4:0]	3rd conversion in regular sequence
9:5	SEQ2[4:0]	2nd conversion in regular sequence
4:0	SEQ1[4:0]	1st conversion in regular sequence

15.11.13 ADC Injection sequence register (ADC_JSEQ)

Address offset: 0x38

Reset value: 0x0000 0000

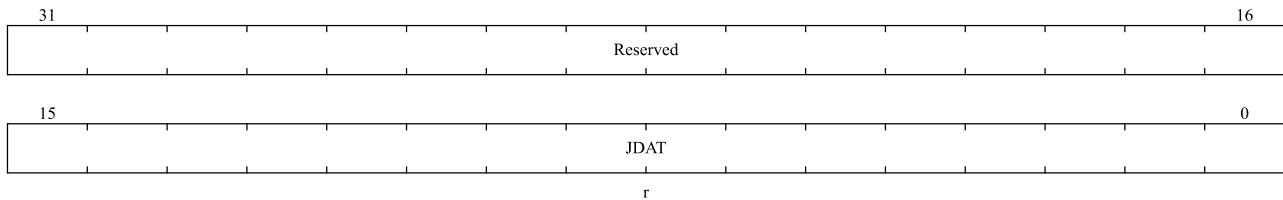
31	22	21	20	19	16
	Reserved		JLEN[1:0]		JSEQ4[4:1]
15	14	10	9	rw	rw
JSEQ4_0		JSEQ3[4:0]		JSEQ2[4:0]	
rw		rw		rw	

Bit field	Name	Description
31:22	Reserved	Reserved, the reset value must be maintained
21:20	JLEN[1:0]	<p>Injected sequence length</p> <p>These bits are software-defined as the number of channels in the injected channel conversion sequence.</p> <p>00: 1 conversion 01: 2 conversions 10: 3 conversions 11: 4 conversions</p>
19:15	JSEQ4[4:0]	<p>This is the 4th conversion in the injected sequence.</p> <p>These bits are software-defined as the number (0 to 18) of the fourth transition channel in the conversion sequence.</p> <p><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 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></p>
14:10	JSEQ3[4:0]	3rd conversion in injected sequence
9:5	JSEQ2[4:0]	2nd conversion in injected sequence
4:0	JSEQ1[4:0]	1st conversion in injected sequence

15.11.14 ADC injection data register x (ADC_JDATx) (x= 1...4)

Address offset: 0x3C - 0x48

Reset value: 0x0000 0000

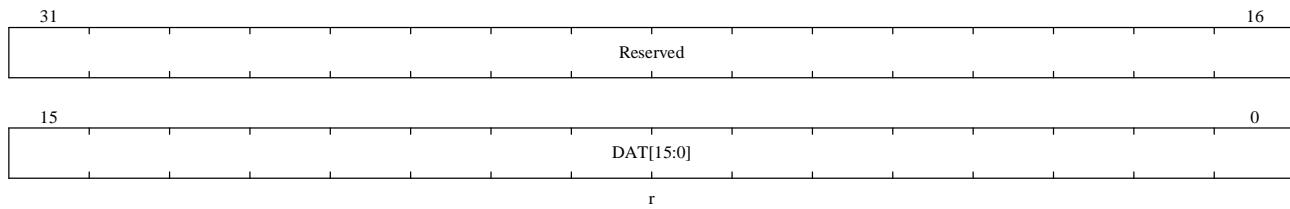


Bit field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained
15:0	JDAT[15:0]	<p>Injected data for conversions</p> <p>These bits are read-only and contain the conversion results of the injected channel. The data is left-aligned or right-aligned</p>

15.11.15 ADC regulars data register (ADC_DAT)

Address offset: 0x4C

Reset value: 0x0000 0000

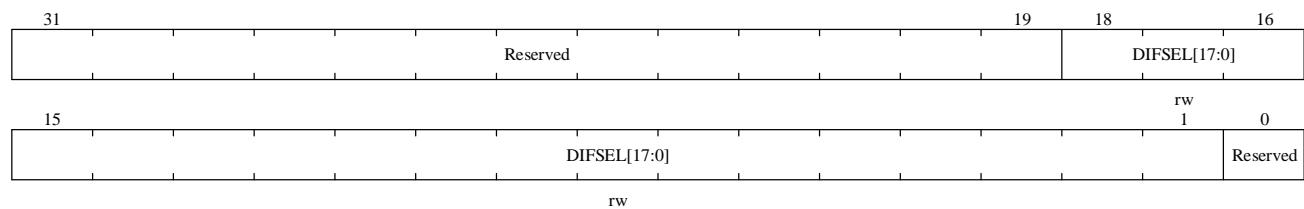


Bit field	Name	Description
32:16	Reserved	Reserved, the reset value must be maintained
15:0	DAT[15:0]	Regular data for conversion These bits are read-only and contain the conversion results of the injected channel. The data is left-aligned or right-aligned as shown in Table 15-3 and Table 15-4.

15.11.16 ADC differential mode selection register (ADC_DIFSEL)

Address offset: 0x50

Reset value: 0x0000 0000

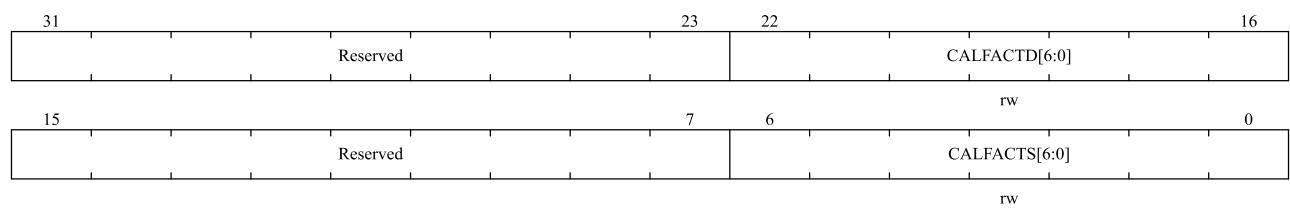


Bit field	Name	Description
31:19	Reserved	Reserved, the reset value must be maintained
18:1	DIFSEL[17:0]	Differential mode for channels 18 to 1 DIFSEL[i] = 0: ADC channel input i+1 is configured in single-ended mode; DIFSEL[i] = 1: ADC channel input i+1 is configured in differential mode
0	Reserved	Reserved, the reset value must be maintained

15.11.17 ADC calibration factor (ADC_CALFACT)

Address offset: 0x54

Reset value: 0x0000 0000



Bit field	Name	Description
31:23	Reserved	Reserved, the reset value must be maintained
22:16	CALFACTD[6:0]	<p>Calibration factors in differential mode</p> <p>This bit can be written by hardware or software</p> <p>After the differential input calibration is complete, the hardware will update it according to the calibration coefficient.</p> <p>Software can write these bits with a new calibration factor. If the new calibration coefficient is different from the current coefficient stored in the analog ADC, the coefficient will be applied after a new differential calibration is initiated.</p> <p><i>Note: software allows write only when ADC_CTRL2.ON=1, ADC_STS.STR =0, ADC_STS.JSTR =0 (ADC does not process conversion or start conversion)</i></p>
15:7	Reserved	Reserved, the reset value must be maintained
6:0	CALFACTS[6:0]	<p>Calibration factors in Single-Ended mode</p> <p>This bit can be written by hardware or software</p> <p>After the single-end input calibration is completed, the hardware will update it according to the calibration coefficient.</p> <p>Software can write these bits with a new calibration factor. If the new calibration coefficient is different from the current coefficient stored in the analog ADC, the coefficient will be applied after a new single-ended calibration is initiated.</p> <p><i>Note: software allows write only when ADC_CTRL2.ON=1, ADC_STS.STR =0, ADC_STS.JSTR =0 (ADC does not process conversion or start conversion)</i></p>

15.11.18 ADC control register 3 (ADC_CTRL3)

Address offset: 0x58

Reset value: 0x0000 0043

31	Reserved														16
15	Reserved	VABT MEN	DPWMOD	JENDC AIEN	ENDC AIEN	BPCAL	PDRDY	RDY	CKMOD	CALALD	CALDIF	RES[1:0]	1	0	

rw rw rw rw rw r r rw rw rw rw rw rw

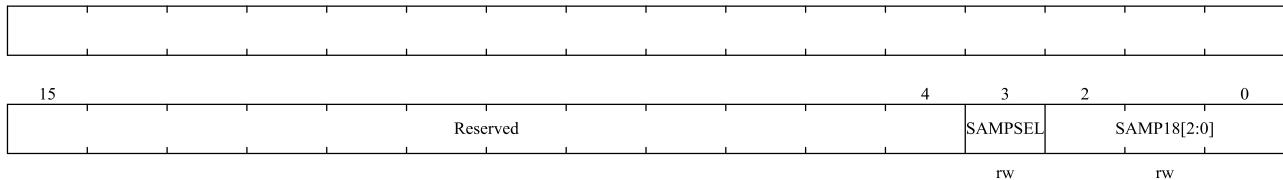
Bit field	Name	Description
31:12	Reserved	Reserved, the reset value must be maintained
11	VBATMEN	<p>Vbat monitor enable</p> <p>0: Disable</p> <p>1: Enable</p>
10	DPWMOD	<p>Deep power mode</p> <p>0: When the ADC is disabled, the ADC enters low power mode</p> <p>1: When the ADC is disabled, the ADC enters deep low power mode</p>
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>

Bit field	Name	Description
		0: ADC_STS.JENDCA interrupt is disabled 1: ADC_STS.JENDCA interrupt is enabled
8	ENDCAIEN	Interrupt enable for any regular channels This bit is set and cleared by the software to enable/disable regular channel conversion to end the interrupt 0: ADC_STS.ENDCA interrupt is disabled 1: ADC_STS.ENDCA interrupt is enabled
7	BPCAL	Bypass calibration 0: Disable 1: Enabled
6	PDRDY	ADC power down ready 0: Not ready 1: Get ready
5	RDY	ADC ready 0: Not ready 1: Get ready
4	CKMOD	Clock mode 0: Select AHB for synchronization clock 1: Select PLL for asynchronous clock
3	CALALD	Calibration auto load 0: Disables automatic loading 1: Enables automatic loading
2	CALDIF	Differential mode for calibration This bit is set and cleared by software to configure the calibrated single-ended or differential input mode 0: Writing ADC_CTRL2.ENCAL bits will start calibration in single-ended input mode 1: Writing ADC_CTRL2.ENCAL bits will start calibration in differential input mode
1:0	RES[1:0]	Data resolution This bit is set and cleared by the software to select the resolution of the conversion 00: 6-bits 01: 8-bits 10: 10-bits 11: 12-bits

15.11.19 ADC sampling time register 3 (ADC_SAMPT3)

Address offset: 0x5C

Reset value: 0x0000 0000



Bit field	Name	Description																
31:4	Reserved	Reserved, the reset value must be maintained																
3	SAMPSEL	<p>Sample Time Selection</p> <p>When SAMPSEL = 0, the value of SAMPx[2:0] is set as follows:</p> <table> <tr> <td>000: 1.5 cycles</td> <td>100: 41.5 cycles</td> </tr> <tr> <td>001: 7.5 cycles</td> <td>101: 55.5 cycles</td> </tr> <tr> <td>010: 13.5 cycles</td> <td>110: 71.5 cycles</td> </tr> <tr> <td>011: 28.5 cycles</td> <td>111: 239.5 cycles</td> </tr> </table> <p>When SAMPSEL = 1, the value of SAMPx[2:0] is set as follows:</p> <table> <tr> <td>000: 1.5 cycles</td> <td>100: 19.5 cycles</td> </tr> <tr> <td>001: 2.5 cycles</td> <td>101: 61.5 cycles</td> </tr> <tr> <td>010: 4.5 cycles</td> <td>110: 181.5 cycles</td> </tr> <tr> <td>011: 7.5 cycles</td> <td>111: 601.5 cycles</td> </tr> </table>	000: 1.5 cycles	100: 41.5 cycles	001: 7.5 cycles	101: 55.5 cycles	010: 13.5 cycles	110: 71.5 cycles	011: 28.5 cycles	111: 239.5 cycles	000: 1.5 cycles	100: 19.5 cycles	001: 2.5 cycles	101: 61.5 cycles	010: 4.5 cycles	110: 181.5 cycles	011: 7.5 cycles	111: 601.5 cycles
000: 1.5 cycles	100: 41.5 cycles																	
001: 7.5 cycles	101: 55.5 cycles																	
010: 13.5 cycles	110: 71.5 cycles																	
011: 28.5 cycles	111: 239.5 cycles																	
000: 1.5 cycles	100: 19.5 cycles																	
001: 2.5 cycles	101: 61.5 cycles																	
010: 4.5 cycles	110: 181.5 cycles																	
011: 7.5 cycles	111: 601.5 cycles																	
2:0	SAMP18[2:0]	<p>Channel Sample Time</p> <p>The channel sampling time definition is consistent with ADC_SAMPT2</p>																

16 I²C interface

16.1 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.

16.2 Characteristics

- Same interface can have both master function and slave function
- Parallel-bus to I²C protocol converter
- Supports 7-bit/10-bit address mode and broadcast addressing
- As I²C master, it can generate clock, start and stop signal
- As I²C 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, byte transfer successfully interrupt and error event interrupt
- Optional clock extending function
- Support DMA mode
- Optional PEC (packet error check) generation and verification
- Programmable analog and digital noise filters
- 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 I²C functions supported by the product.

16.3 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 turn on or disable interrupt according to their needs.

16.3.1 SDA and SCL line control

I2C 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. The data on I2C 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 I2C 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 sending end bit is started (I2C_STS1.TXDATE = 1, I2C_STS1.BSF = 1), the I2C 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 (I2C_STS1.RXDATNE = 1, I2C_STS1.BSF = 1), the I2C 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 (I2C_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 (I2C_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.

16.3.2 Software communication process

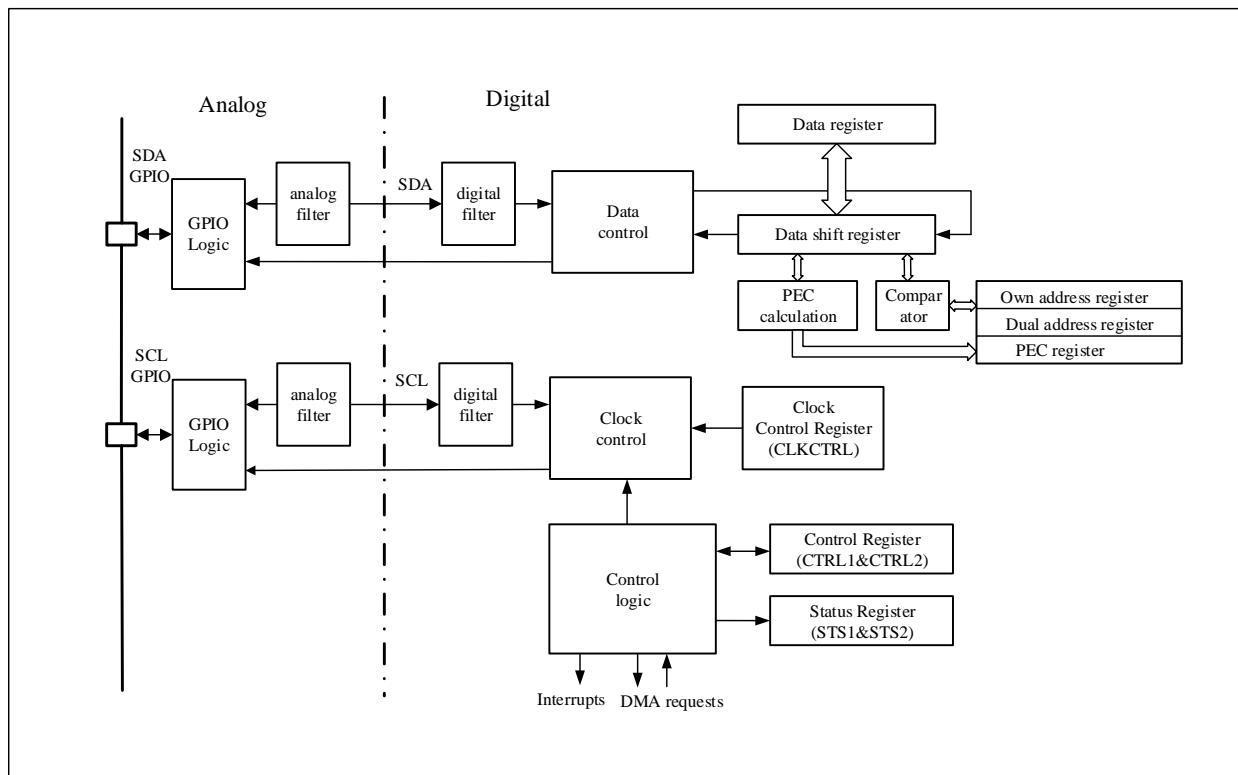
The data transmission of I2C device is divided into master and slave. Master is the device responsible for initializing the transmission of data on the bus and generating clock signal. At this time, any addressed device is a slave. Whether the I2C device is a master or a slave, it can send or receive data. Therefore, the I2C interface supports four operation modes:

- Slave transmitter mode
- Slave receiver mode
- Master transmitter mode
- Master receiver mode

After system reset, I2C works in slave mode by default. The I2C 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 I²C interface is shown in the figure below.

Figure 16-1 Functional block diagram of I²C

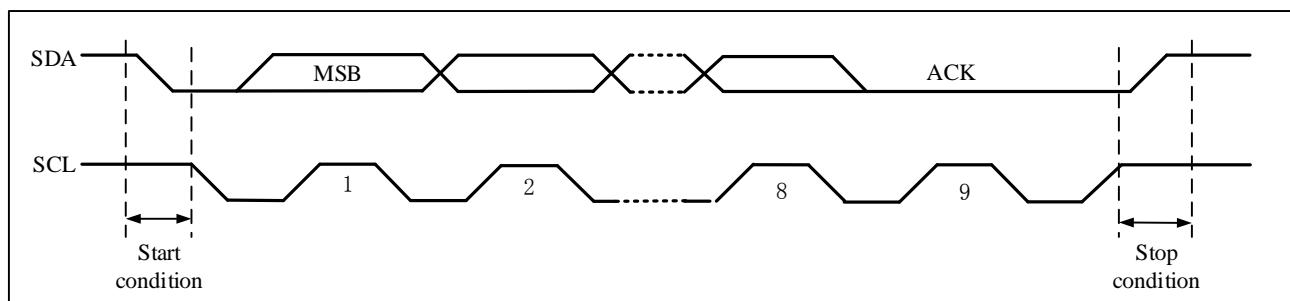


Note: in SMBus mode, SMBALERT signals are not supported .

16.3.2.1 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 16-2 I²C bus protocol



16.3.2.2 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.
- The 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 periods, the clock line is released and goes high, after which there is no difference in the state of the device clock and SCL lines, and all devices will begin counting their high periods, the device that completes the high 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.

16.3.2.3 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 16-2 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).

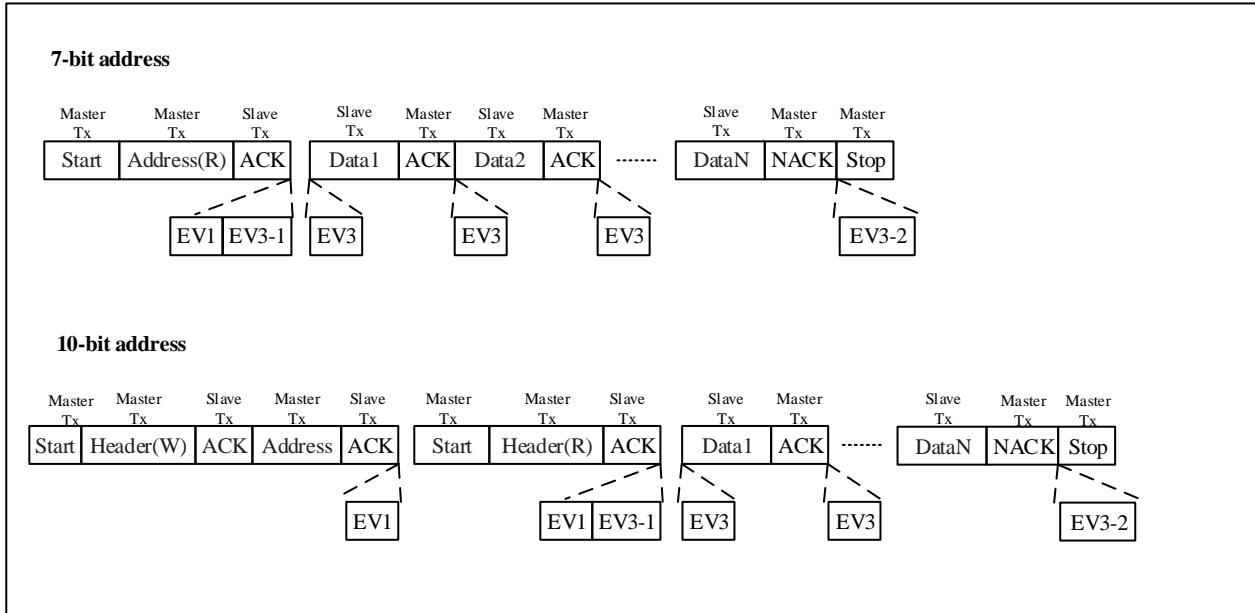
16.3.2.4 I2C slave transmission mode

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 16-3 Transfer sequence diagram slave transmitter



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.

16.3.2.5 I2C slave receiving mode

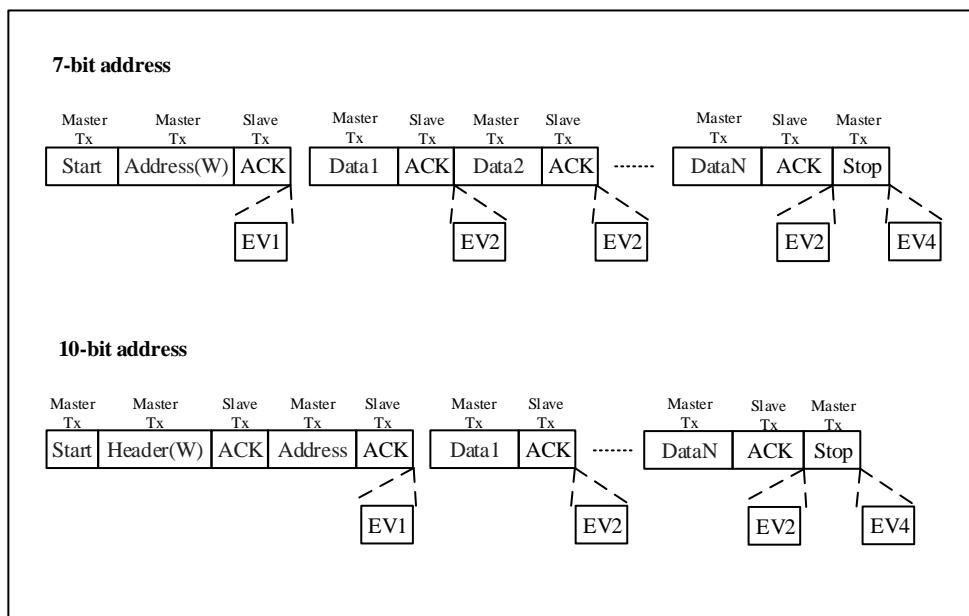
When receiving data in slave mode, the software should operate as follows:

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 16-4 Transfer sequence diagram of slave receiver



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.

16.3.2.6 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 bit is 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 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).

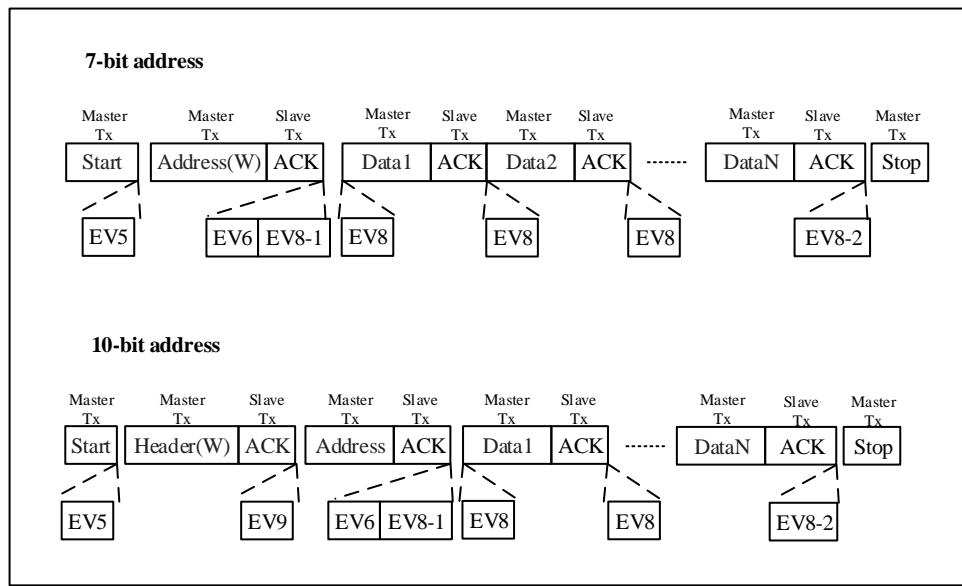
In the 7-bit address mode, only one address byte needs to be sent out. Once the address byte is sent out:

- ◆ 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".

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 16-5 Master transmitter transmission 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.

16.3.2.7 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.

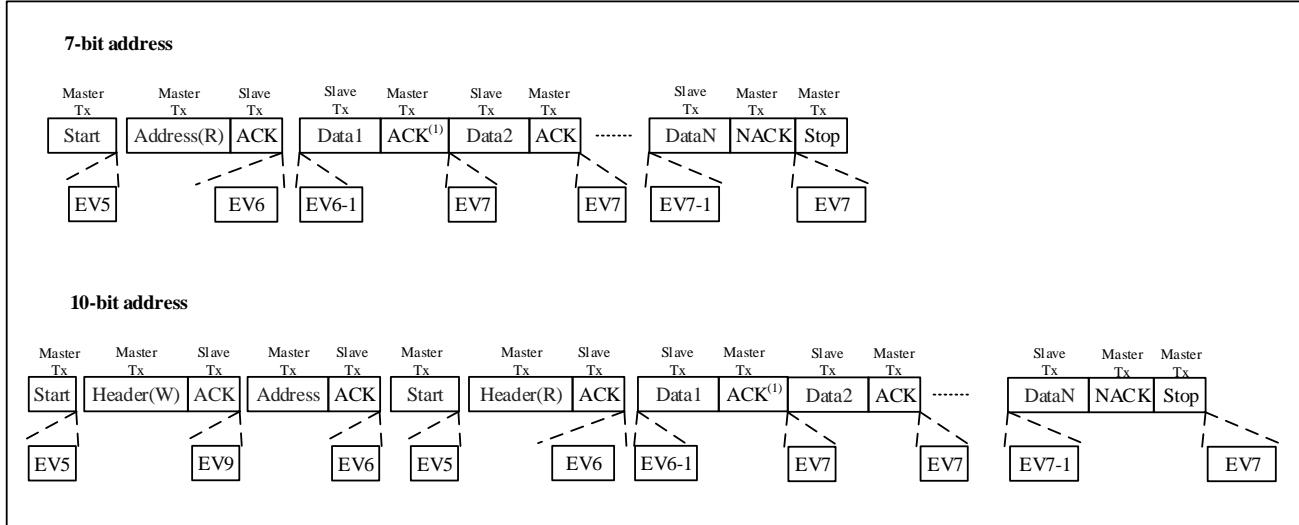
Note: In the receiving mode, the master device sets the lowest bit as '1' when sending the slave address.

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 16-6 Master receiver transmission 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.

16.3.3 Error conditions description

I2C errors mainly include bus error, acknowledge error, arbitration loss, overload\ underload error. These errors may cause communication failure.

16.3.3.1 Acknowledge Failure(ACKFAIL)

The interface have a acknowledge bit is detected that does not match the expectation, it will occurs acknowledge failure 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.

16.3.3.2 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, when the bus does not release by hardware,as the same time it done not affect the current status of sending,The current sending 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 situaiton. It is an wrong start condition, the slave device as a restart and waits for addredss or stop condition. It is an wrong stop condition, the slave device as a normal stop condition and the hardware releases the bus.

16.3.3.3 Arbitration Lost(ARLOST)

The interface have arbitration lost is detected, 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. Hardware release the bus.

16.3.3.4 Overrun/Underrun Error(OVERRUN)

In slave mode, disable clock extend prone to Overrun/Underrun Error(OVERRUN):

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 a 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 a underrun error. In this situation, the previous byte in the I2C_DAT register is sending repeatedly. And User make sure it happen underrun error. The receiver discard repeatedly byte, and transmitter should follow specified time according to I2C bus standard updata I2C_DAT register.

In sending the first byte, it must clear I2C_STS1.ADDRF bit and the I2C_DAT register must be written before the first SCL is raised. If cannot make sure do that, the first byte should be discard by receiver.

16.3.4 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 transfer mode, if in EOT interrupt handler DMA request need to be disable, and I2C_STS1.BSF event first come before stop condition set.

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, in interrupt enable.

16.3.4.1 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 channel for I2C transmission, (x is the channel number) the following step must be operate:

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.

16.3.4.2 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 operate:

1. In DMA_PADDRx register set the address of the I2C_DAT register. In every I2C_STS1.RXDATEN event, data will send from address to storage area.
2. In DMA_MADDRx register set the memory area address. In every I2C_STS1.RXDATEN 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.RXDATEN 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 channel.

7. When DMA transfer data is done, DMA need to send 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 receiving, do not set I2C_CTRL2.BUFINTEN bit.

16.3.5 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 used 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.RXDATNE 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 transfer 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.

16.3.6 Noise filter

I2C interface standard requires that 50ns burr can be filtered on SCL/SDA. So analog filter and digital filter are added in design. By default, analog filter are enable and can be set I2C_TMRSE.SCLAFENN/SDAAFENN to disable. The analog filter can configure I2C_TMRSE.SCLAFE/SDAAF to set filter burrs with the width of 5ns, 15ns, 25ns, 35ns. Digital filter can be set I2C_TMRSE.SCLDFW/SDADFW is a non-zero value to enable. The max width of filter is SCLDEW[3:0]/SDADFW[3:0]*T_{PCLK}. Enabling the digital filter will increase the hold time of SDA and the increment is (SDADFW[3:0]+1)*T_{PCLK}.

Table 16-1 SDADFW/ SCLDFW recommended configuration

APB1 clock	SDADFW/ SCLDFW Max value	
	Standard mode	Fast mode
2MHz<=APB1<=5MHz	0	0
5MHz<APB1<=10MHz	10	0
10MHz<APB1<=20MHz	15	2
20MHz<APB1<=30MHz	15	2
30MHz<APB1<=32MHz	15	3

16.3.7 SMBus

16.3.7.1 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 communicate 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).
- The data format is similar. SMBus data format is similar to 7-bit address format of I2C(See Figure 16-2).
- Both are master-slave communication modes, and the master device provides the clock.
- Both support multi master.

Differences between SMBus and I2C:

Table 16-2 Comparison between SMBus and I2C

SMBus	I ² 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

16.3.7.2 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.

16.3.7.3 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.

16.3.7.4 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.

16.3.7.5 Address resolution protocol

The SMBus resolves address conflict 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).

16.3.7.6 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.

16.3.7.7 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.

16.4 Debug mode

When the microcontroller enters the debug mode (Cortex-M4 core is in the stop state), configure the DBG_CTRL.I2CxSMBUS_TIMEOUT bit in the DBG module, Select SMBUS timeout to continue normal work or stop. See section 21.3.2 for details.

16.5 Interrupt request

All I2C interrupt requests are listed in the following table.

Table 16-3 I²C interrupt request

Interrupt function	Interrupt event	Event flag	Set control bit
I2C event 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
	Send buffer is empty.	TXDATE	
I2C error interrupt	Bus error	BUSERR	ERRINTEN
	Lost arbitration (master)	ARLOST	
	Acknowledge fail	ACKFAIL	
	Overrun/underrun	OVERRUN	
	PEC error	PECERR	
	Timeout /Tlow error	TIMOUT	

Note: 1. STARTBF, ADDR F, ADDR10F, STOPF, BSF, RXDATNE and TXDATE are merged into the event interrupt channel through logical OR.

2. BUSERR, ARLOST, ACKFAIL, OVERRUN, PECERR and TIMEOUT are merged into the error interrupt channel through logical OR.

16.6 I2C register

These peripheral registers can be operated by half word (16 bits) or word (32 bits)

16.6.1 I2C register overview

Table 16-4 I2C 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	I2C_CTRL1	Reserved										Reserved										CLKFREQ[5:0]											
	Reset Value	0	SWRESET	0	DMALAST	0	PIC	12	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	I2C_CTRL2	Reserved										Reserved										ADDR[7:1]											
	Reset Value	0	ADDRMODE	0	DMAEN	0	ACKPOS	11	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	I2C_OADDR1	Reserved										Reserved										DATA[7:0]											
	Reset Value	0	ADDR[9:8]	0	EVINTEN	0	STOPGEN	9	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	I2C_OADDR2	Reserved										Reserved										ADDR2[7:1]											
	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	
010h	I2C_DAT	Reserved										Reserved										TMRISE[5:0]											
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
014h	I2C_STS1	Reserved										Reserved										CLKCTRL[11:0]											
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
018h	I2C_STS2	Reserved										PECVAL[7:0]										TMRISE[5:0]											
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
01Ch	I2C_CLKCTRL	Reserved										Reserved										SMBTYPE[3]											
	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	
020h	I2C_TMRISE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

16.6.2 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	PEC	ACK POS	ACKEN	STOP GEN	START GEN	NO EXTEND	GCEN	PECEN	ARPEN	SMB TYPE	Reserved	SMB MODE	EN	

Bit field	Name	Description
15	SWRESET	<p>Software reset</p> <p>Make sure the I2C bus is idle before resetting this bit.</p> <p>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>
14:13	Reserved	Reserved, the reset value must be maintained.
12	PEC	<p>Packet error checking</p> <p>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.</p> <p>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)</p> <p>It can be set or cleared by software. Or when I2C_CTRL1.EN=0, it will be cleared by hardware.</p> <p>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. 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</p> <p>It can be set or cleared by software. Or when I2C_CTRL1.EN equals to 0, it will be cleared by hardware.</p> <p>0: No acknowledge send; 1: Send an acknowledge after receiving a byte</p>
9	STOPGEN	<p>Stop generation</p> <p>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.,</p> <p>In the master mode: 0: No stop condition generates; 1: Generate a stop condition.</p> <p>In the slave mode: 0: No stop condition generates; 1: Release SCL and SDA lines after the current byte.</p>

Bit field	Name	Description
		<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</p> <p>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.</p> <p>0: No start condition generates; 1: Generate a start conditions.</p>
7	NOEXTEND	<p>Clock extending disable (Slave mode)</p> <p>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</p> <p>0: Enable Clock extending. 1: Disable Clock extending.</p>
6	GCEN	<p>General call enable</p> <p>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</p> <p>0: Disable PEC module; 1: Enable PEC module.</p>
4	ARPEN	<p>ARP enable</p> <p>0: Disable ARP; 1: Enable ARP.</p> <p>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.</p>
3	SMBTYPE	<p>SMBus type</p> <p>0: Device 1: Host</p>
2	Reserved	Reserved, the reset value must be maintained.
1	SMBMODE	<p>SMBus mode</p> <p>0: I2C mode; 1: SMBus mode.</p>
0	EN	<p>I2C Peripheral enable</p> <p>0: Disable I2C module; 1: Enable I2C module</p> <p><i>Note: If this bit is cleared when the 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></p> <p><i>In master mode, this bit must never be cleared until the communication has ended.</i></p>

16.6.3 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	BUFINT EN	EVTINT EN	ERRINT EN	Reserved			CLKFREQ[5:0]	

rw rw rw rw rw rw

Bit field	Name	Description
15:13	Reserved	Reserved, the reset value must be maintained.
12	DMALAST	<p>DMA last transfer</p> <p>0: Next DMA EOT is not the last transfer</p> <p>1: Next DMA EOT is the last transfer</p> <p><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></p>
11	DMAEN	<p>DMA requests enable</p> <p>0: Disable DMA</p> <p>1: Enable DMA</p>
10	BUFINTEN	<p>Buffer interrupt enable</p> <p>0: When I2C_STS1.TXDATE=1 or I2C_STS1.RXDATNE=1, any interrupt is not generated.</p> <p>1: If I2C_CTRL2.EVTINTEN= 1, When I2C_STS1.TXDATE=1 or I2C_STS1.RXDATNE=1, interrupt will be generated.</p>
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> <p>I2C_STS1.STARTBF = 1 (Master)</p> <p>I2C_STS1.ADDRF = 1 (Master/Slave)</p> <p>I2C_STS1.ADD10F = 1 (Master)</p> <p>I2C_STS1.STOPF = 1 (Slave)</p> <p>I2C_STS1.BSF = 1 with no I2C_STS1.TXDATE or I2C_STS1.RXDATNE event</p> <p>I2C_STS1.TXDATE = 1 if I2C_CTRL2.BUFINTEN = 1</p> <p>I2C_STS1.RXDATNE = 1 if I2C_CTRL2.BUFINTEN = 1</p>
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> <p>I2C_STS1.BUSERR = 1;</p> <p>I2C_STS1.ARLOST = 1;</p> <p>I2C_STS1.ACKFAIL = 1;</p> <p>I2C_STS1.OVERRUN = 1;</p> <p>I2C_STS1.PECERR = 1;</p> <p>I2C_STS1.TIMOUT = 1;</p>
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 timming.</p>

Bit field	Name	Description
		000000: Disable 000001: Disable 000010: 2MHz 000011: 3MHz ... 100000: 32MHz 100001~111111: Disable.

16.6.4 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]	
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>

16.6.5 Own address register 2 (I2C_OADDR2)

Address offset: 0x0C

Reset value: 0x0000

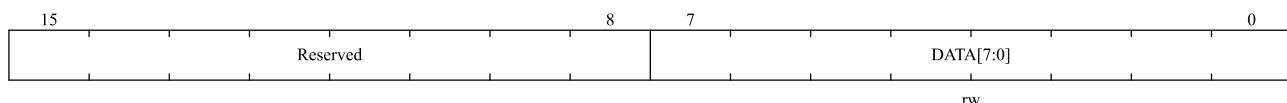
Bit field	Name	Description
15:8	Reserved	Reserved, the reset value must be maintained.

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>

16.6.6 Data register (I2C_DAT)

Address offset: 0x10

Reset value: 0x0000

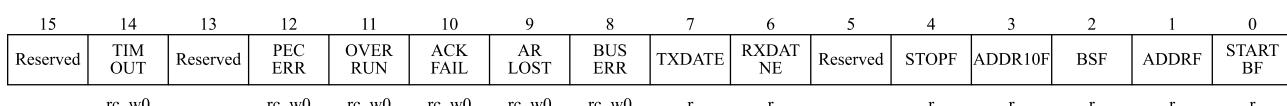


Bit field	Name	Description
15:8	Reserved	Reserved, the reset value must be maintained.
7:0	DATA[7:0]	8-bit data register Send or receive data buffer. <i>Note: In the slave mode, the address will not be copied into the data register;</i> <i>Note: if I2C_STS1.TXDATE =0, data can still be written into the 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>

16.6.7 Status register 1 (I2C_STS1)

Address offset: 0x14

Reset value: 0x0000



Bit field	Name	Description
15	Reserved	Reserved, the reset value must be maintained.
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: ■ SCL has kept low for 25ms (Timeout).

Bit field	Name	Description
		<ul style="list-style-type: none"> ■ Master cumulative clock low extend time more than 10 ms (Tlow:mext). ■ Slave cumulative clock low extend time more than 25 ms (Tlow:sext). <p>Timeout in slave mode: slave device resets the communication and hardware frees the bus.</p> <p>Timeout in master mode: hardware sends the stop condition.</p>
13	Reserved	Reserved, the reset value must be maintained.
12	PECERR	<p>PEC Error in reception</p> <p>Writing '0' to this bit by software can clear it, or it is cleared by hardware when I2C_CTRL1.EN=0.</p> <p>0: No PEC error</p> <p>1: PEC error: receiver will returns NACK Whether the I2C_CTRL1.ACKEN bit is enabled</p>
11	OVERRUN	<p>Overrun/Underrun</p> <p>Writing '0' to this bit by software can clear it, or it is cleared by hardware when I2C_CTRL1.EN=0.</p> <p>0: No Overrun/Underrun</p> <p>1: Overrun/Underrun</p> <p>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.</p>
10	ACKFAIL	<p>Acknowledge failure</p> <p>Writing '0' to this bit by software can clear it, or it is cleared by hardware when I2C_CTRL1.EN=0.</p> <p>0: No acknowledge failed;</p> <p>1: Acknowledge failed.</p>
9	ARLOST	<p>Arbitration lost (master mode)</p> <p>Writing '0' to this bit by software can clear it, or it is cleared by hardware when I2C_CTRL1.EN=0.</p> <p>0: No arbitration lost;</p> <p>1: Arbitration lost.</p> <p>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).</p> <p><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></p>
8	BUSERR	<p>Bus error</p> <p>Writing '0' to this bit by software can clear it, or it is cleared by hardware when I2C_CTRL1.EN=0.</p> <p>0: No start or stop condition error</p> <p>1: Start or stop condition error</p>
7	TXDATE	<p>Data register empty (transmitters)</p> <p>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.</p>

Bit field	Name	Description
		<p>0: Data register is not empty; 1: Data register is empty.</p> <p>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.</p> <p>If a NACK is received, or the next byte to be sent is PEC(I2C_CTRL1.PEC=1), this bit will not be set.</p> <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: The master device 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:</p>

Bit field	Name	Description
		<p>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> <p><i>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.</i></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: The 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 occurs, or the SMBus device default address or SMBus host 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>

16.6.8 Status register 2 (I2C_STS2)

Address offset: 0x18

Reset value: 0x0000

15	PECVAL[7:0]	8	7	6	5	4	3	2	1	0
		r	r	r	r	r	Reserved	TRF	Busy	MS MODE

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: the received address matches the content in OADDR1;</p>

Bit field	Name	Description
		1: the received address matches the content in OADDR2.
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 when I2C_CTRL1.EN=0.</p> <p>0: SMBus host address was not received;</p> <p>1: when I2C_CTRL1.SMBTYPE=1 and I2C_CTRL1.AR PEN=1, the SMBus host address is received.</p>
5	SMBDADDR	<p>SMBus device default address (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: The default address of SMBus device has not been received;</p> <p>1: when I2C_CTRL1.AR PEN=1, the default address of SMBus device is received.</p>
4	GCALLADDR	<p>General call address(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: No general call address was received;</p> <p>1: when I2C_CTRL1.GCEN=1, general call address was received.</p>
3	Reserved	Reserved, the reset value must be maintained.
2	TRF	<p>Transmitter/receiver</p> <p>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.</p> <p>0: Data receiving mode;</p> <p>1: Data transmission mode;</p> <p>At the end of the whole address transmission stage, this bit is set according to the R/W bit of the address byte.</p>
1	BUSY	<p>Bus busy</p> <p>Hardware clears this bit when a stop condition is detected.</p> <p>0: No data communication on the bus;</p> <p>1: Data communication is in progress on the bus.</p> <p>When detecting that SDA or SCL is low level, the hardware sets this bit to '1';</p> <p><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></p>
0	MSMODE	<p>Master/slave mode</p> <p>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.</p> <p>0: In slave mode;</p> <p>1: In master mode.</p> <p>When the interface is in the master mode (I2C_STS1.STARTBF=1), the hardware sets this bit;</p>

16.6.9 Clock control register (I2C_CLKCTRL)

Address offset: 0x1c

Reset value: 0x0000

Note: 1. F_{PCLK1} 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 I²C is turned off (I2C_CTRL1.EN=0)

Bit field	Name	Description
15	FSMODE	I2C master mode selection 0: I2C in standard mode(duty cycle defaults to 1/1); 1: I2C in fast mode(duty cycle can be configured).
14	DUTY	Duty cycle in fast mode 0: $T_{low}/T_{high} = 2$; 1: $T_{low}/T_{high} = 16/9$
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"> ■ If duty cycle = $T_{low}/T_{high} = 1/1$: $CLKCTRL = f_{PCLK1}(\text{Hz})/100000/2$ $T_{low} = CLKCTRL \times T_{PCLK1}$ $T_{high} = CLKCTRL \times T_{PCLK1}$ ■ If duty cycle = $T_{low}/T_{high} = 2/1$: $CLKCTRL = f_{PCLK1}(\text{Hz})/100000/3$ $T_{low} = 2 \times CLKCTRL \times T_{PCLK1}$ $T_{high} = CLKCTRL \times T_{PCLK1}$ ■ If duty cycle = $T_{low}/T_{high} = 16/9$: $CLKCTRL = f_{PCLK1}(\text{Hz})/100000/25$ $T_{low} = 16 \times CLKCTRL \times T_{PCLK1}$ $T_{high} = 9 \times CLKCTRL \times T_{PCLK1}$ <p>For example, if $f_{PCLK1}(\text{Hz}) = 8\text{MHz}$, duty cycle = 1/1, $CLKCTRL = 8000000/100000/2 = 0x28$.</p> <p><i>Note: 1. The minimum setting value is 0x04 in standard mode and 0x01 in fast mode;</i></p> <p><i>2. $T_{high} = T_{r(SCL)} + T_{w(SCLH)}$. See the definitions of these parameters in the data sheet for details.</i></p> <p><i>3. $T_{low} = T_{f(SCL)} + T_{w(SCLL)}$, see the definitions of these parameters in the data sheet for details;</i></p>

16.6.10 Rise time register (I2C_TMRISE)

Address offset: 0x20

Reset value: 0x0002

Note: The I2C_TMRISE register function is only valid in master mode. changed when I2C is disabled

(I2C_CTRL1.EN=0).

31	30	29	28	27	26	25	22	21	18	17	16
SCLAF ENN	SCLAFW[1:0]	SDAAF ENN	SDAAFW[1:0]		SCLDFW[3:0]		SDADFW[3:0]		Reserved		
rw 15	rw	rw	rw		rw		rw	6	5	rw	0
Reserved						TMRISE[5:0]					

Bit field	Name	Description
31	SCLAFENN	SCL analog filter enable. 0: enable 1: disable
30:29	SCLAFW[1:0]	SCL analog filter width selection: 00: 5ns 01: 15ns 10: 25ns 11: 35ns <i>Note: analog filter has a wide range of variance with different PVT</i>
28	SDAAFENN	SDA analog filter enable. 0: enable 1: disable
27:26	SDAAFW[1:0]	SDA analog filter width selection: 00: 5ns 01: 15ns 10: 25ns 11: 35ns <i>Note: analog filter has a wide range of variance with different PVT</i>
25:22	SCLDFW[3:0]	SCL digital filter width selection. 0000: disable SCL digital filter other values: the numbers of pclk cycles
21:18	SDADFW[3:0]	SDA digital filter width selection. 0000: disable SDA digital filter other values: the numbers of pclk cycles
17:6	Reserved	Reserved, the reset value must be maintained.
5:0	TMRISE[5:0]	Maximum rise time in fast/standard mode (master mode). These bits must be set to the maximum SCL rising time given in the I2C bus specification, and incremented step is 1. 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 and TPCLK1=125ns ,09h(1000ns/125 ns + 1) must be written in TMRISE[5:0] . If the result is not an integer, write the integer part to TMRISE[5:0] to ensure the t _{HIGH} parameter. The filter value can be added to TMRISE[5:0] .

17 Universal synchronous asynchronous receiver transmitter (USART)

17.1 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.

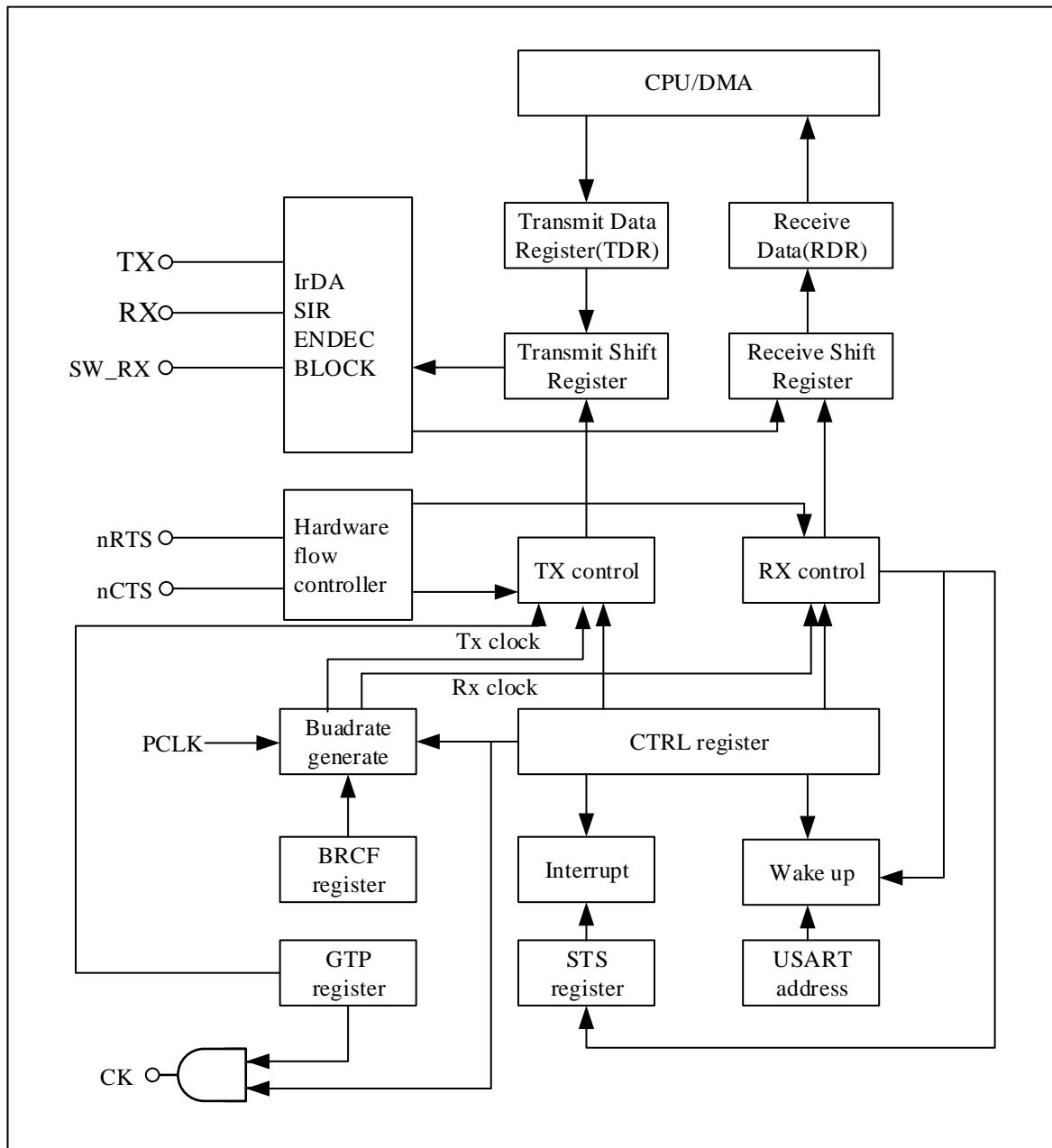
The USART has programmable transmit and receive baud rates and can communicate continuously using DMA. It also supports multiprocessor communication, LIN mode, synchronous mode, single-wire half-duplex communication, smart card asynchronous protocol, IrDA SIR ENDEC function, and hardware flow control function.

17.2 Main features

- Full-duplex operation
- Single-wire half-duplex operation
- Baud rate generator, the highest baud rate can reach 4Mbit/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 Interconnection 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

17.3 Functional block diagram

Figure 17-1 USART block diagram



17.4 Function description

As shown in the Figure 17-1, 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. The commonly used frame structure is: 1 start bit + 8 or 9 data bits (least significant bit first) + 1 parity bit (optional) + 0.5,1.5 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.

17.4.1 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, followed by the start bit of a data frame containing the data .

A break frame means that all bits in one frame period receive '0'. At the end of the break frame, the transmitter inserts 1 or 2 more stop bits ('1') to acknowledge the start bit.

Figure 17-2 word length = 8 setting

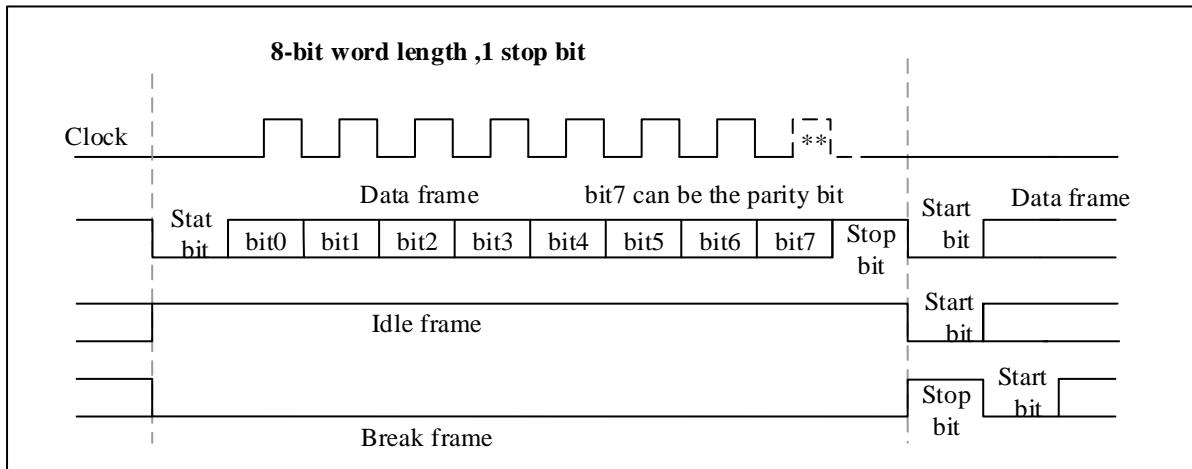
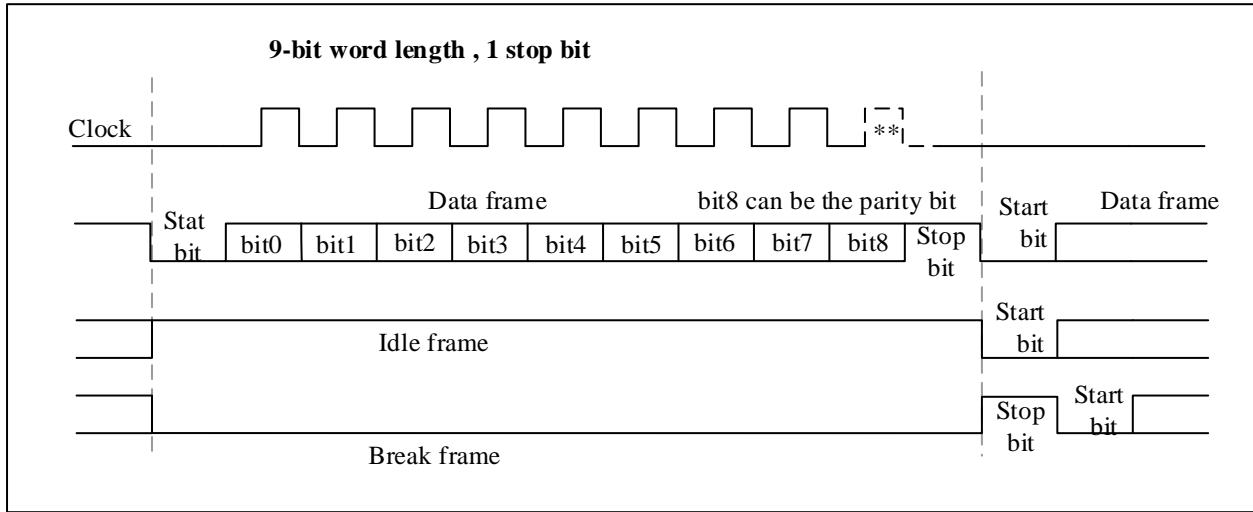


Figure 17-3 word length = 9 setting



17.4.2 Transmitter

After the transmitter is enabled, the data entered into the transmit shift register is sent out through the TX pin.

17.4.2.1 Idle frame

Setting USART_CTRL1.TXEN will cause the USART to transmit an idle frame before the first data frame.

17.4.2.2 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.

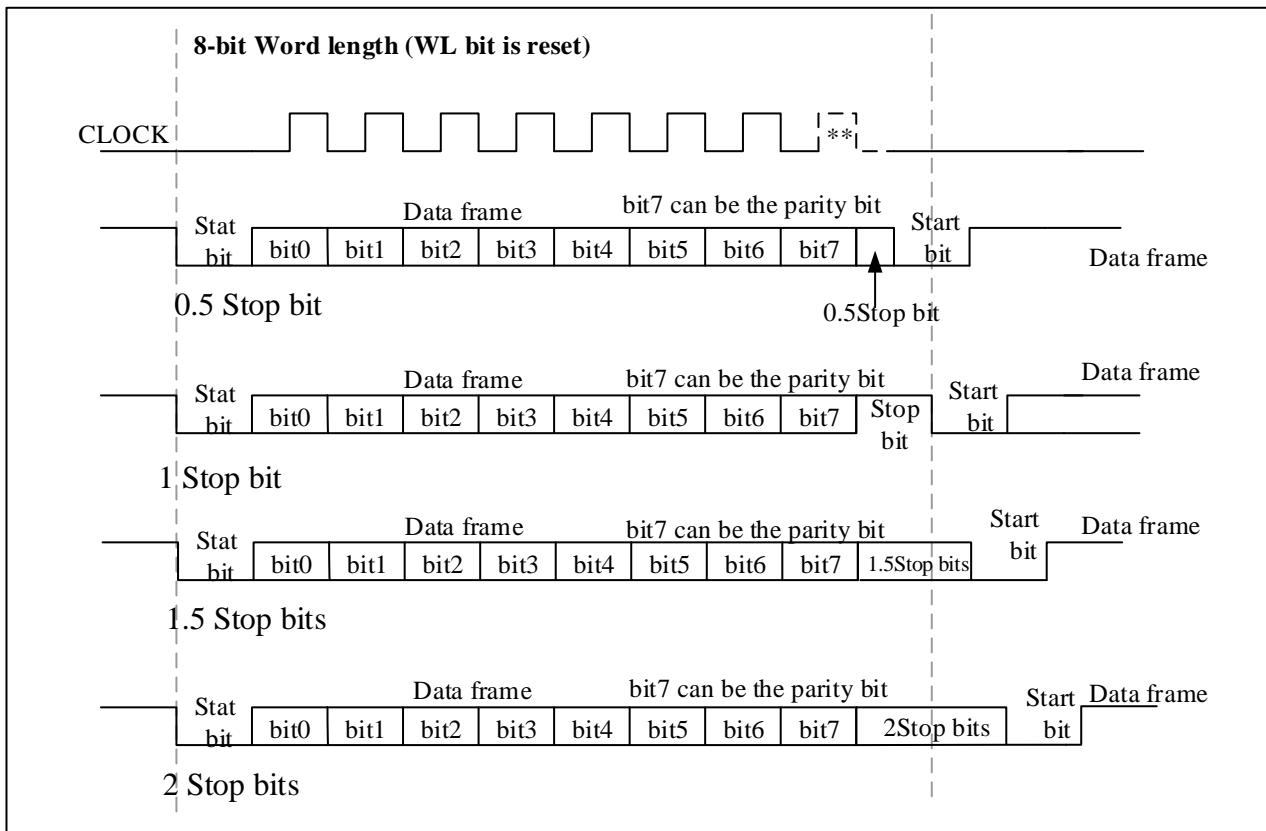
17.4.2.3 Stop bit

The characters are followed by stop bits, the number of which can be configured by setting USART_CTRL2.STPB[1:0].

Table 17-1 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 17-4 configuration stop bit



17.4.2.4 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.

17.4.2.5 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.

17.4.2.6 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). 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:

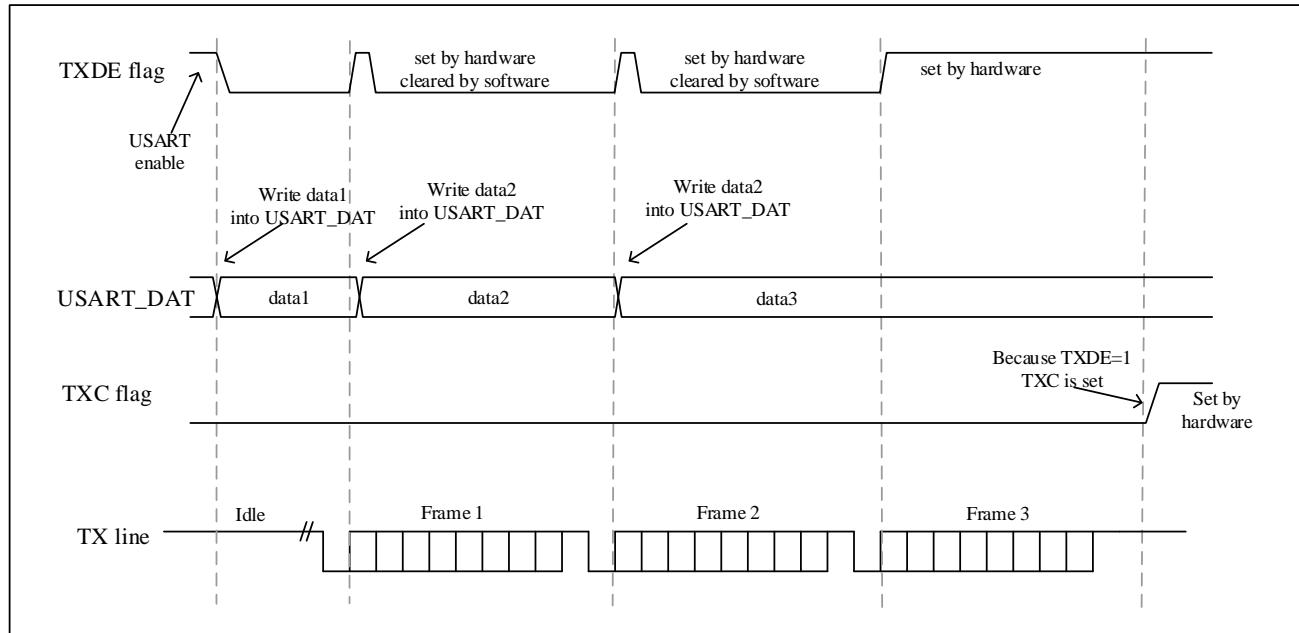
- When the transmit shift register is not sending data and is in an idle state, the data is directly put into the shift register for transmission, and the USART_STS.TXDE bit is set by hardware;
- When the transmit shift register is sending data, the data is stored in the TDR register, and after the current transmission is completed, the data is put into the shift register.

An interrupt is generated if USART_CTRL1.TXDEIEN is '1'.

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 17-5 TXC/TXDE changes during transmission



17.4.3 Receiver

17.4.3.1 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.

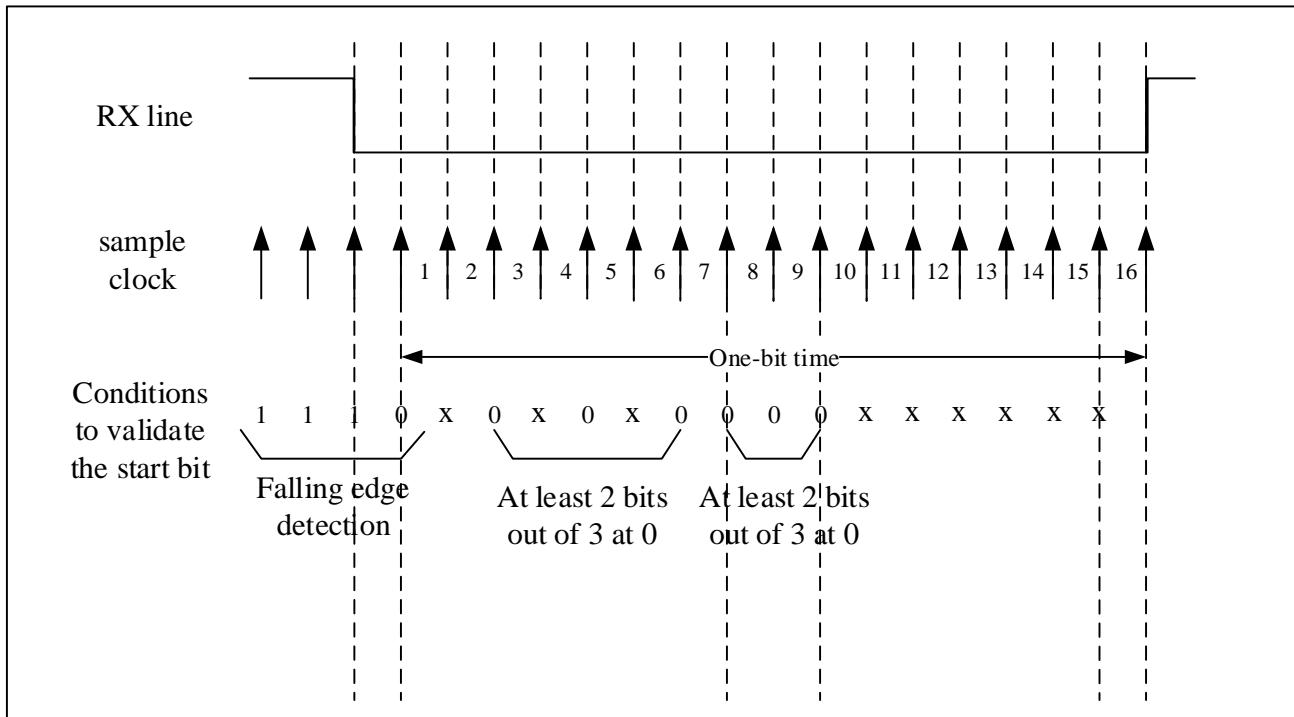
The samples of the 3rd, 5th, and 7th bits have two '0' points, and at the same time, the samples of the 8th, 9th, and 10th bits have three '0' points, then the start bit is confirmed, but it will be set NEF noise flag.

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 17-6 Start bit detection



17.4.3.2 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 17.4.14 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.

17.4.3.3 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.RXDNEIEN 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.

17.4.3.4 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).

17.4.3.5 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).

17.4.3.6 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 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.

17.4.3.7 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 read 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.

17.4.3.8 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 read 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 17-2 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
-----	---	---	-----------

17.4.4 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.

$$\text{TX / RX baud rate} = f_{CK} / (16 * \text{USARTDIV})$$

where f_{CK} is the clock provided to the peripheral:

- PCLK1 is used for USART2, up to 32MHz;
- PCLK2 is used for UART1, UART4, up to 64 MHz.

USARTDIV is an unsigned fixed-point number.

17.4.4.1 USARTDIV and USART_BRCF register configuration

Example 1:

If USARTDIV = 27.75, then:

$$\text{DIV_Decimal} = 16 * 0.75 = 12 = 0x0C$$

$$\text{DIV_Integer} = 27 = 0x1B$$

$$\text{So USART_BRCF} = 0x1BC$$

Example 2:

If USARTDIV = 20.98, then:

$$\text{DIV_Decimal} = 16 * 0.98 = 15.68$$

Nearest integer: $\text{DIV_Decimal} = 16 = 0x10$, out of configurable range, so a carry to integer is required

$$\text{So DIV_Integer} = 20 + 1 = 21 = 0x15$$

$$\text{DIV_Decimal} = 0x0$$

$$\text{So USART_BRCF} = 0x150$$

Example 3:

If USART_BRCF = 0x19B:

$$\text{DIV_Integer} = 0x19 = 25$$

$$\text{DIV_Decimal} = 0x0B = 11$$

So USARTDIV = $25+11/16 = 25.6875$

Table 17-3 Error calculation when setting baud rate

Baud rate		$f_{PCLK}=32MHz$			$f_{PCLK}=64MHz$		
serial number	Kbps	reality	Set value in register	Error(%)	reality	Set value in register	Error(%)
1	2.4	2.400	833.333	0%	2.4	1666.667	0%
2	9.6	9.600	208.333	0.02%	9.6	416.667	0%
3	19.2	19.2	104.167	0.02%	19.2	208.333	0.02%
4	57.6	57.6	34.722	0.05%	57.6	69.444	0.05%
5	115.2	115.384	17.361	0.16%	115.2	34.722	0.05%
6	230.4	230.769	8.681	0.16%	230.769	17.361	0%
7	460.8	461.538	3.6875	0.69%	461.538	8.681	0%
8	921.6	923.076	2.170	1%	923.076	4.340	0.8%
9	1687.5	1687.7	1.185	0%	2250	2.370	0%
10	3375	impossible	impossible	impossible	3375	1.185	0%

Notes: The lower the clock frequency of the CPU, the lower the error for a particular baud rate.

17.4.5 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 17-4 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 17-5 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%

17.4.6 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 17-6 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.PEIE1 is enabled, an interrupt is generated.

17.4.7 DMA application

The USART supports the DMA mode using multi-buffer configuration, which can realize high-speed data communication.

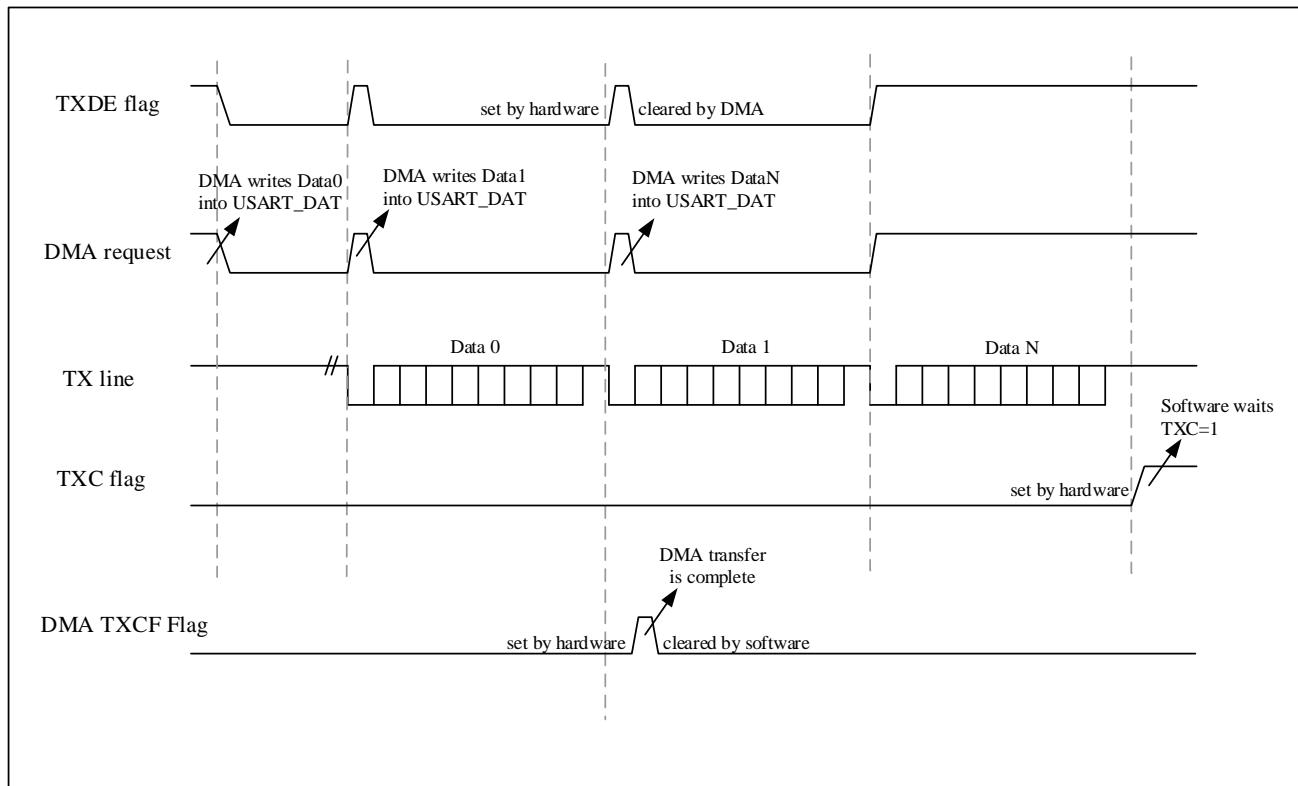
17.4.7.1 Using DMA transmission

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 17-7 Transmission using DMA



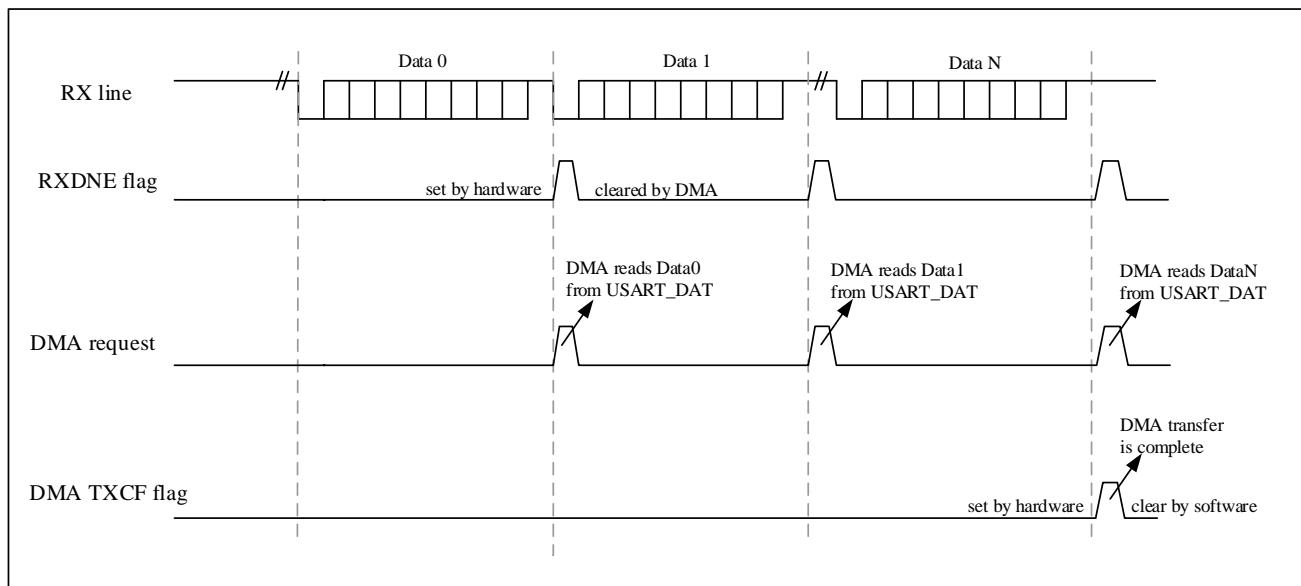
17.4.7.2 Using DMA reception

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 17-8 Reception using DMA

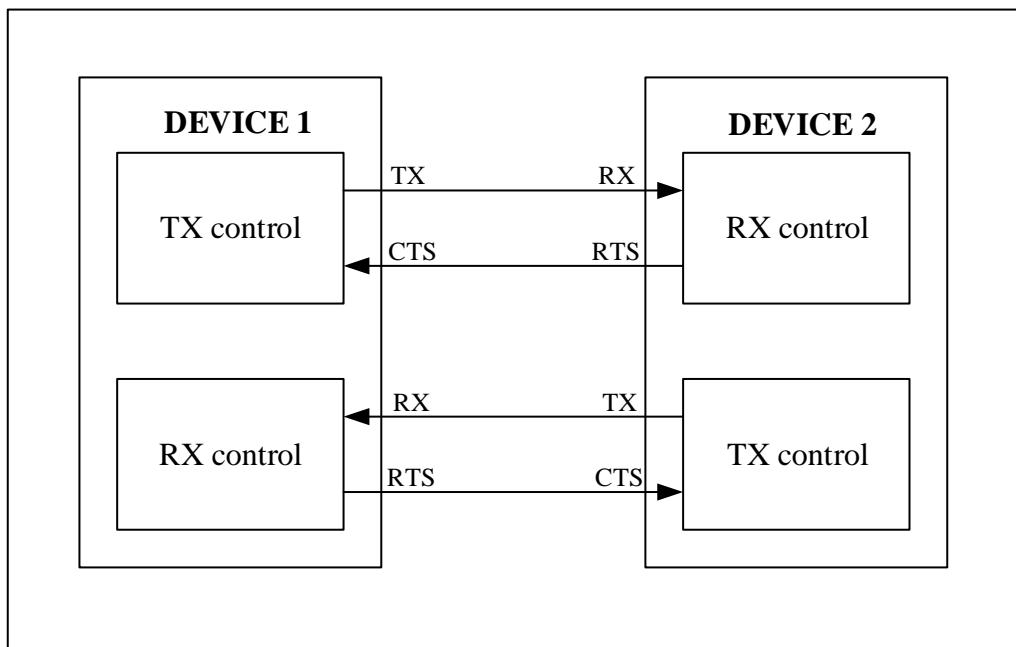


In multi-buffer communication mode, the error flag will be set when there is a frame error, overload or noise error. An interrupt will be generated if the error interrupt is enabled (USART_CTRL3.ERRIEN=1).

17.4.8 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.

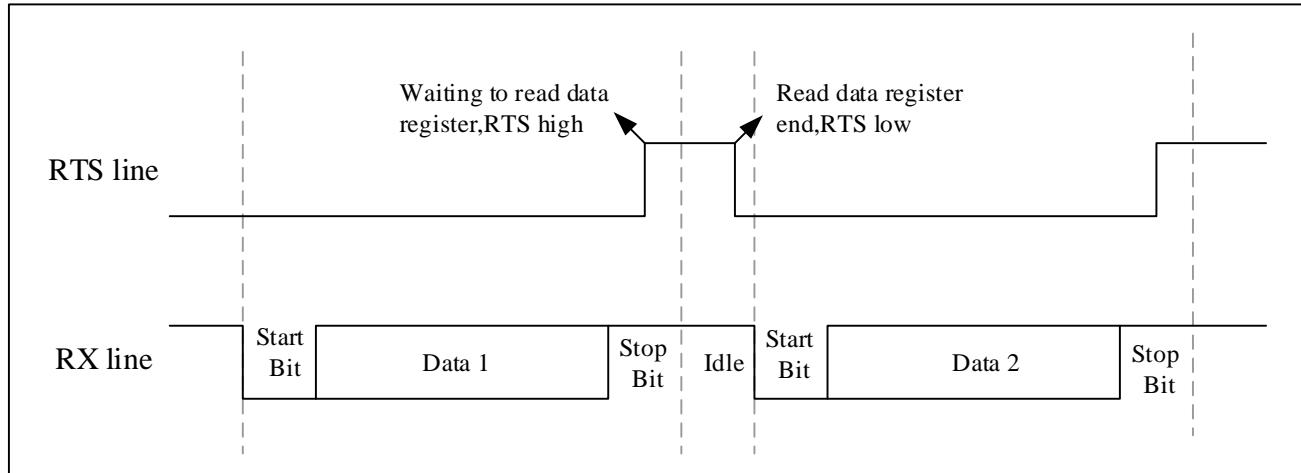
Figure 17-9 hardware flow control between two USART



17.4.8.1 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 17-10 RTS flow control

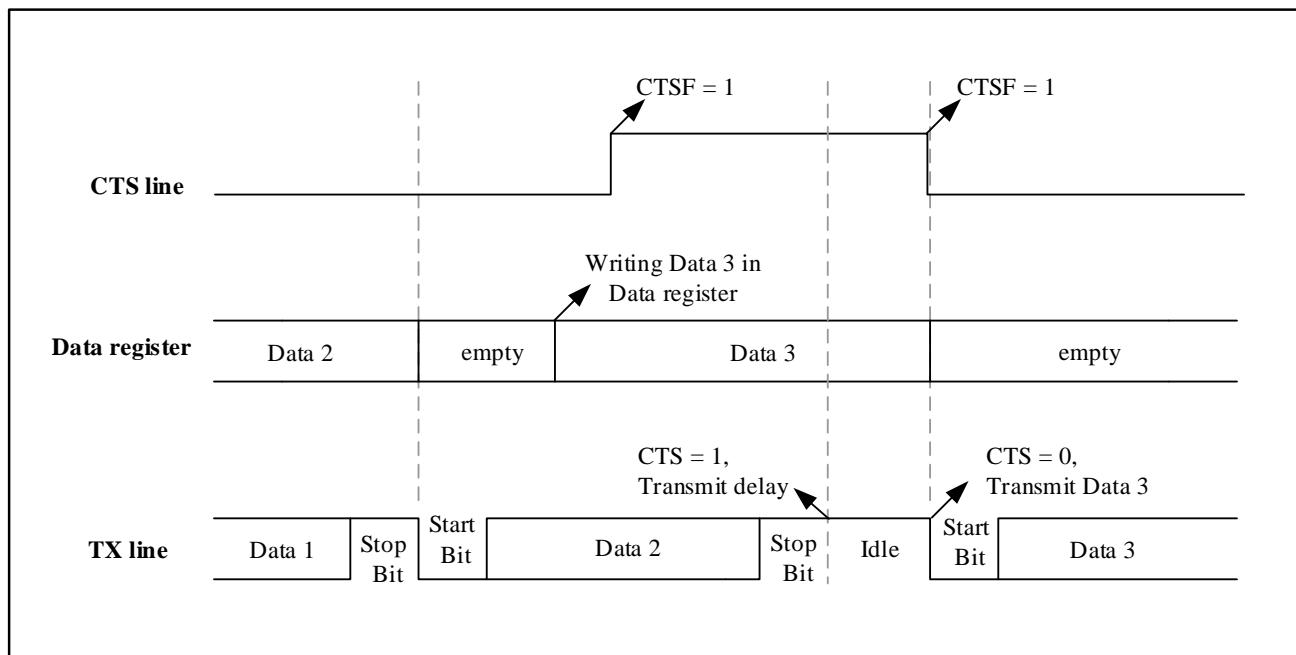


17.4.8.2 CTS flow control

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 17-11 CTS flow controls



17.4.9 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 ANDed 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.

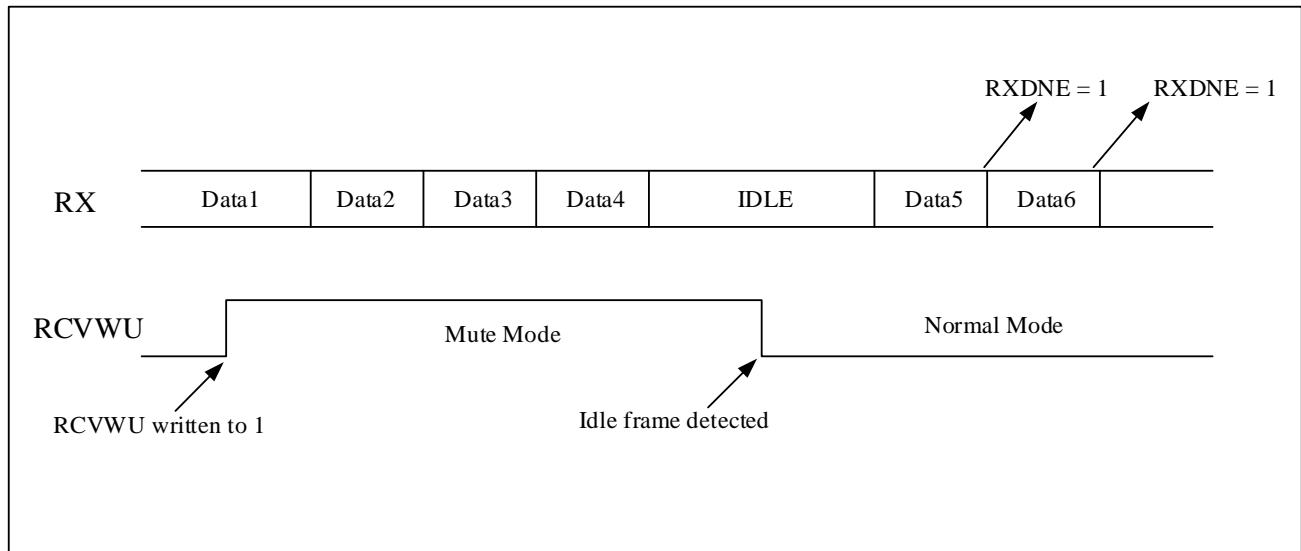
17.4.9.1 Idle line detection

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 17-12 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 17-12 Mute mode using idle line detection



17.4.9.2 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;

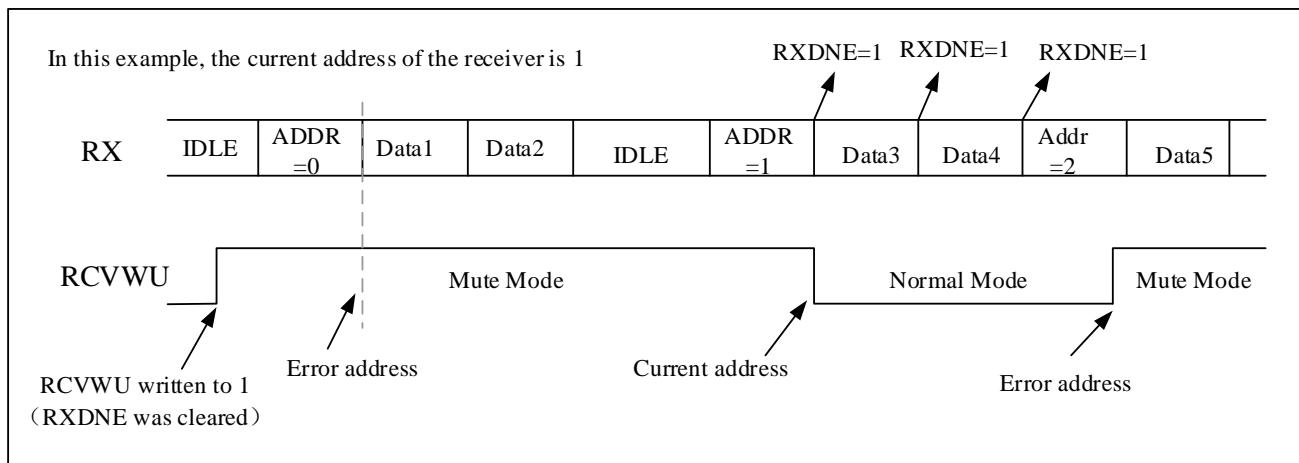
Note: When the receive buffer contains no data (RXDNE=0 in USART_STS), the USART_CTRL1.RCVWU bit can be written to 0 or 1. Otherwise, the write operation is ignored.

- 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 17-13 Mute mode detected using address mark



17.4.10 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.

17.4.10.1 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 external clock is not activated.

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.

17.4.10.2 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.

17.4.10.3 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 17-14 USART synchronous transmission example

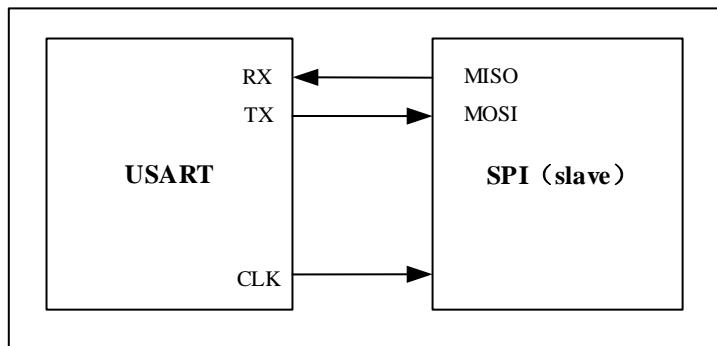


Figure 17-15 USART data clock timing example (WL=0)

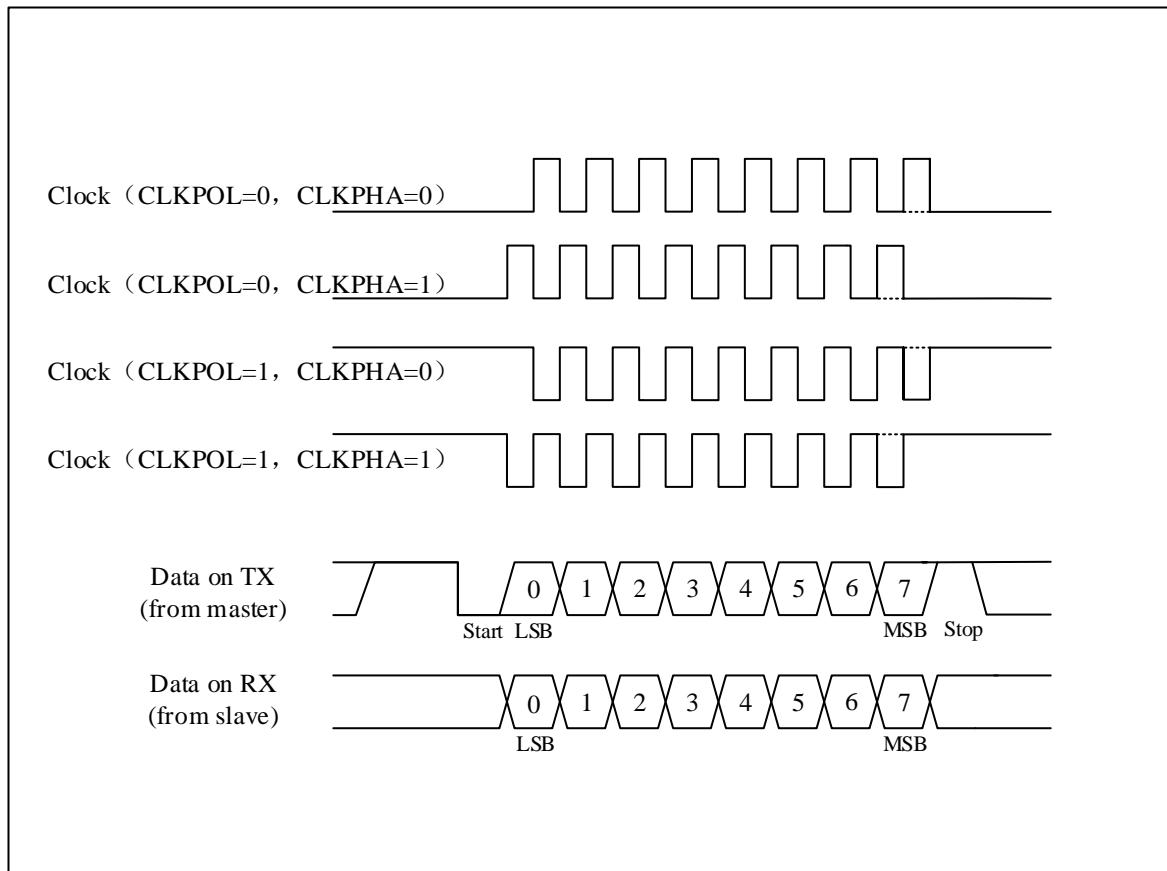


Figure 17-16 USART data clock timing example (WL=1)

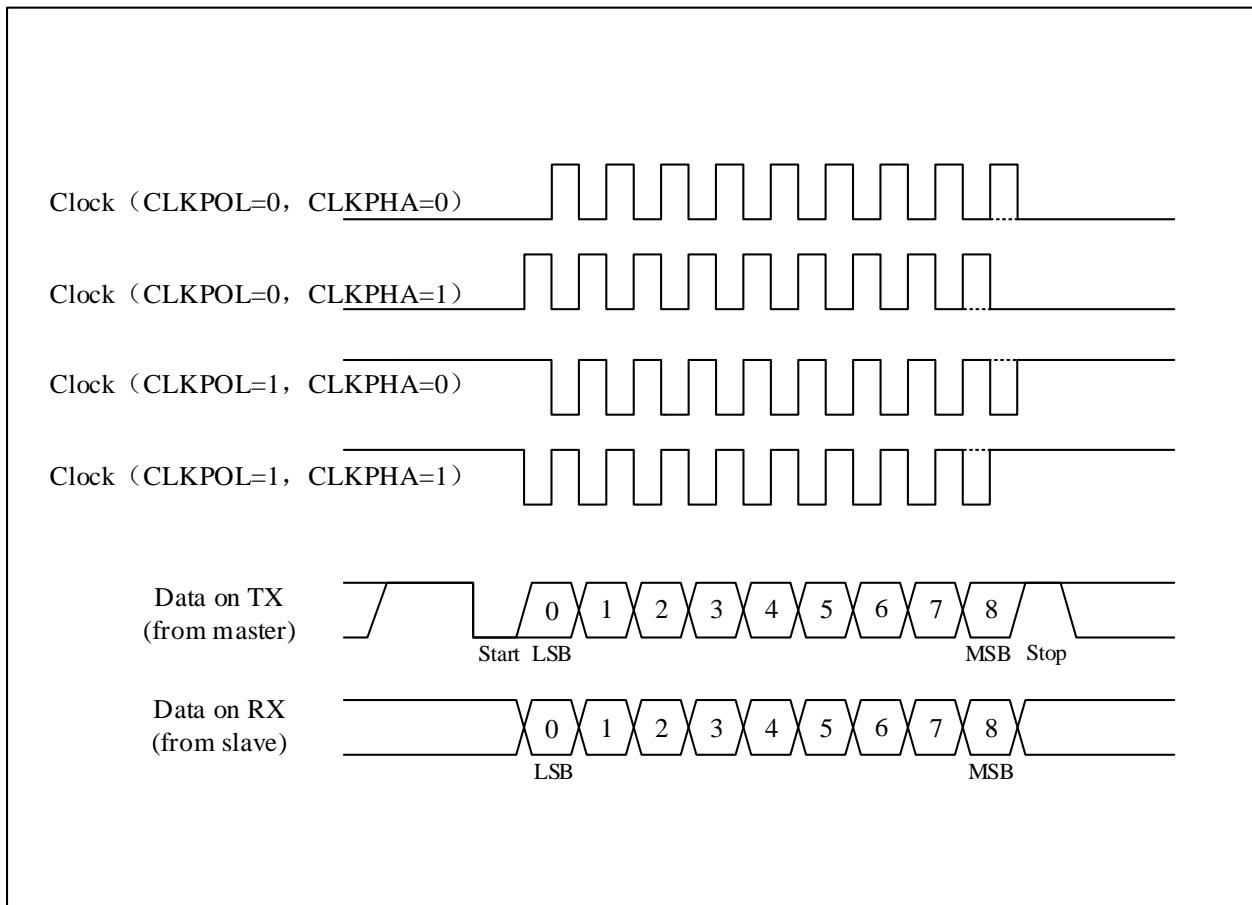
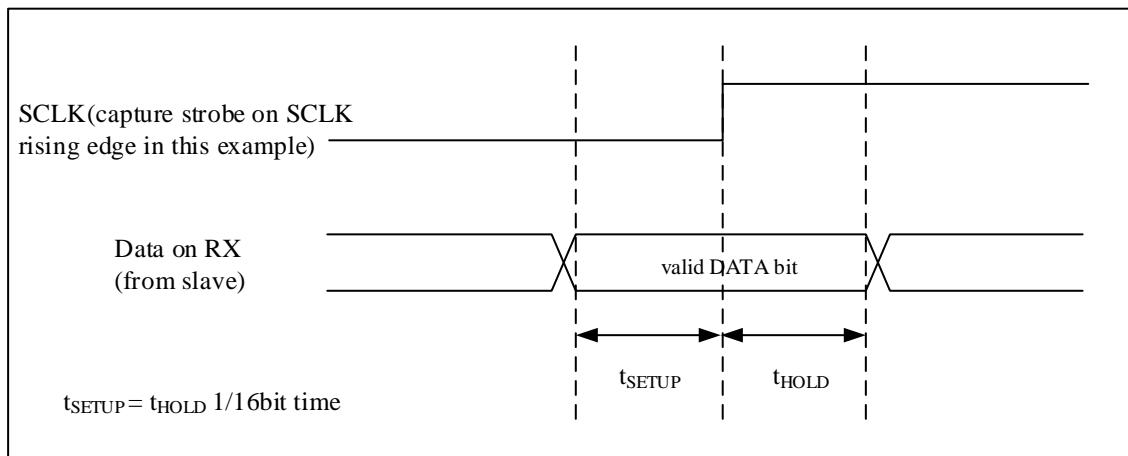


Figure 17-17 RX data sampling / holding time



Note: the function of CK is different in Smartcard mode, please refer to the Smartcard mode section for details.

17.4.11 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.

17.4.12 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.

17.4.12.1 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. It uses an 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 Figure 17-19. 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.

17.4.12.2 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 MHz < PSC < 2.12 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 17-18 IrDASIRENDEC-Block diagram

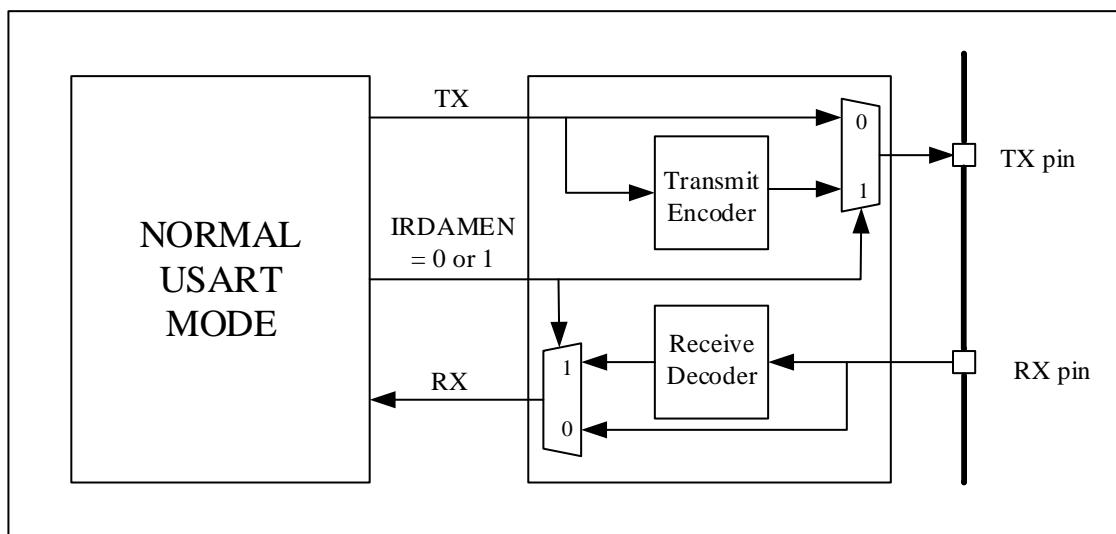
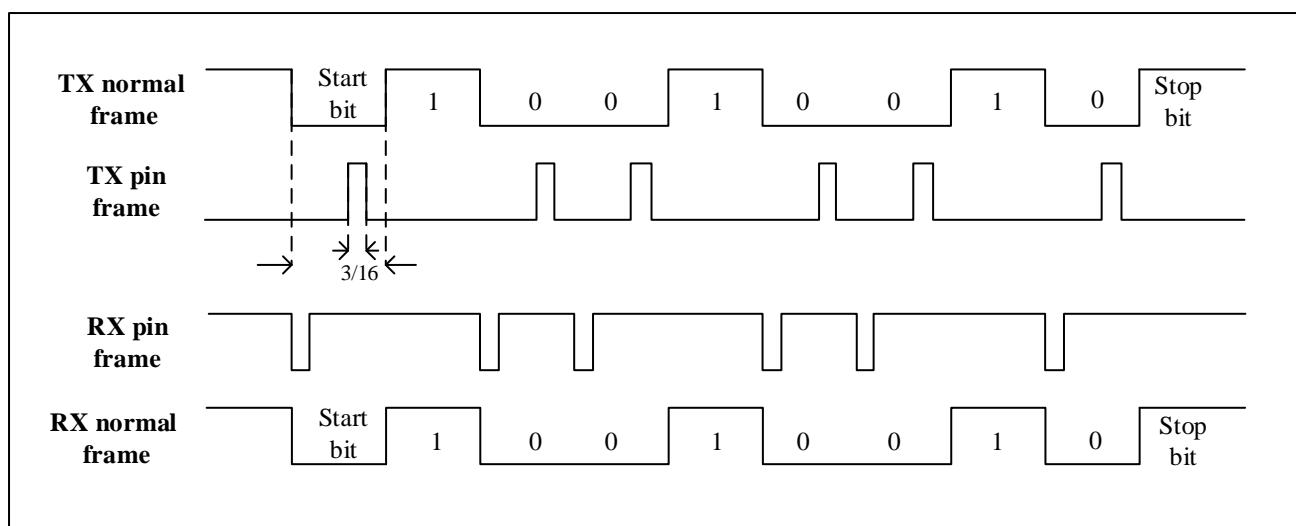


Figure 17-19 IrDA data Modulation (3/16)-normal mode



17.4.13 LIN mode

The 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.

17.4.13.1 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.

17.4.13.2 LIN receiving

Whether the bus is idle or during the transmission of a data frame, as long as the break symbol 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 17-20 Break detection in LIN mode (11-bit break length-the LINBDF bit is set)

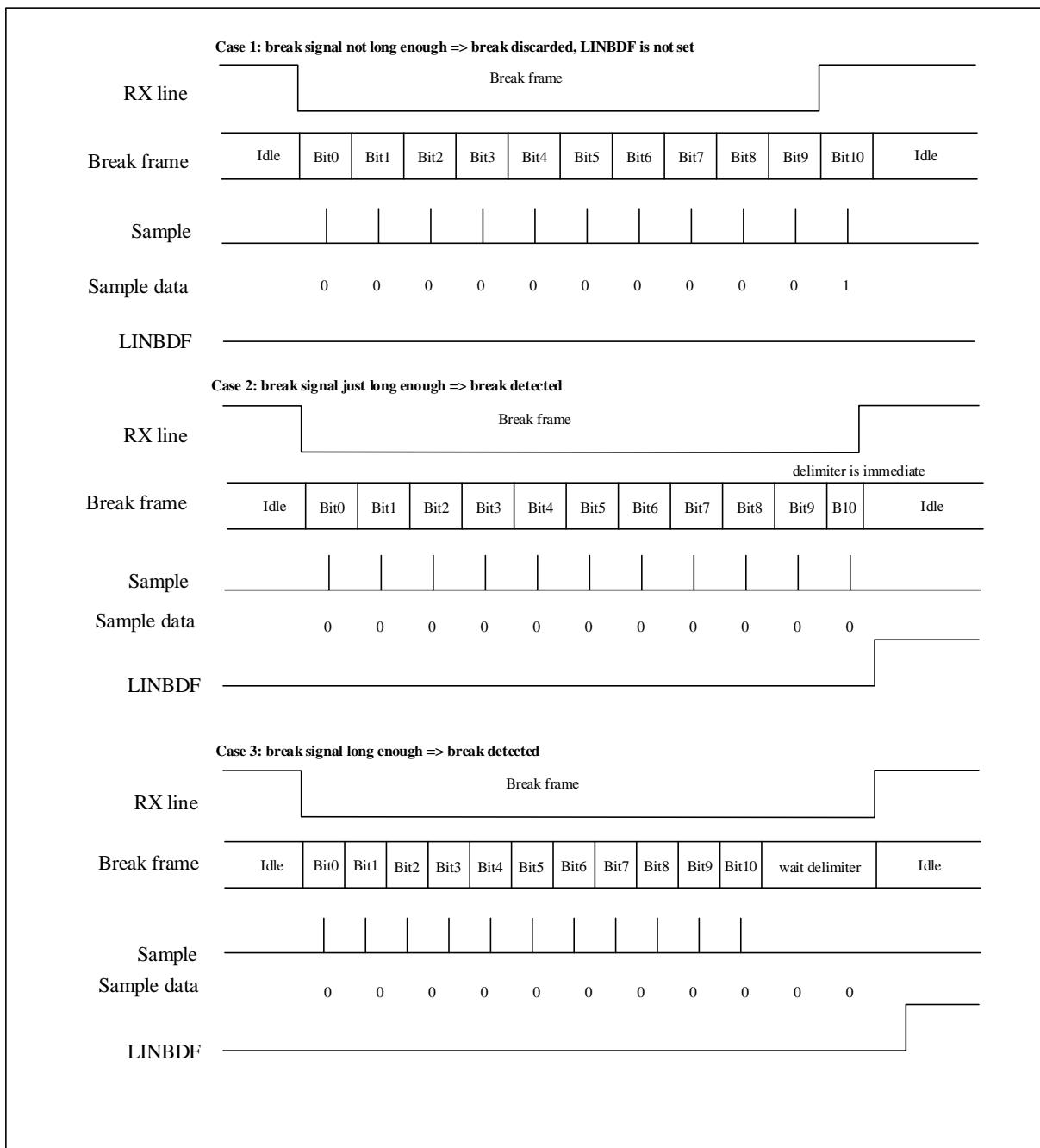
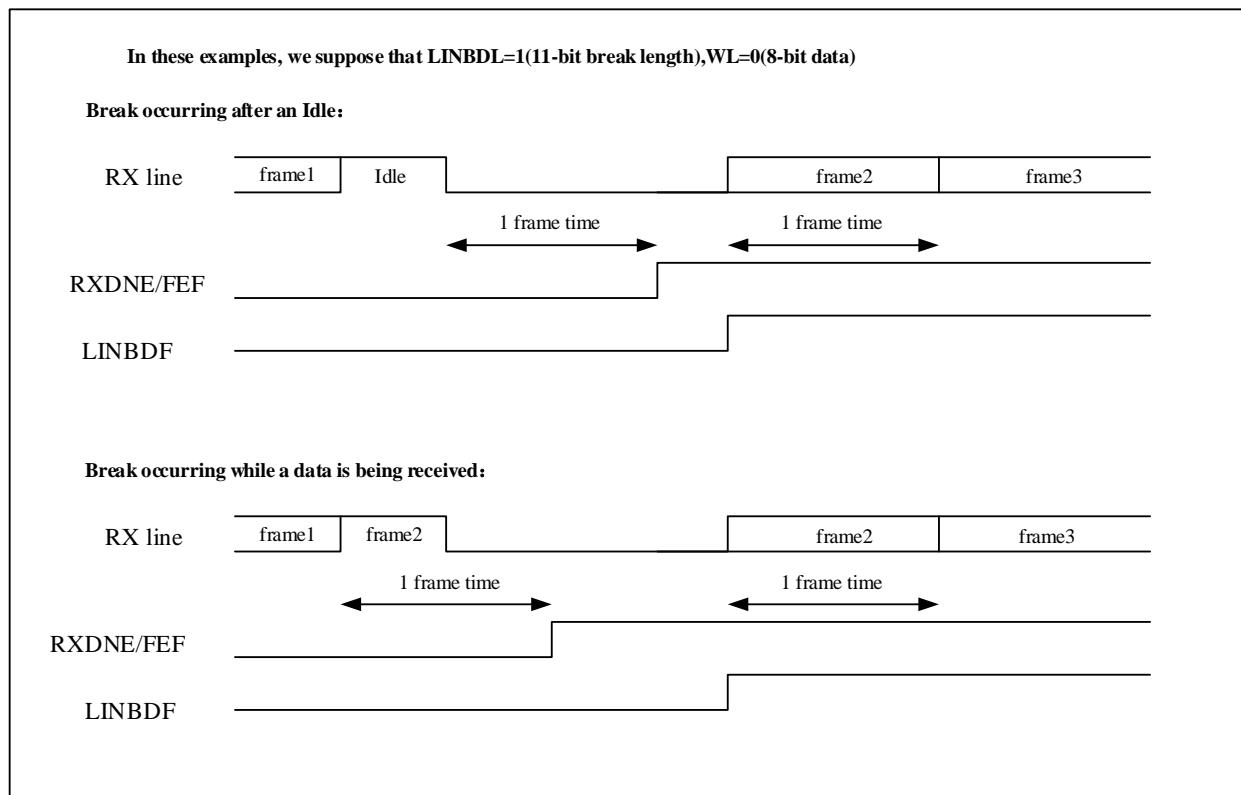


Figure 17-21 Break detection and framing error detection in LIN mode



17.4.14 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.

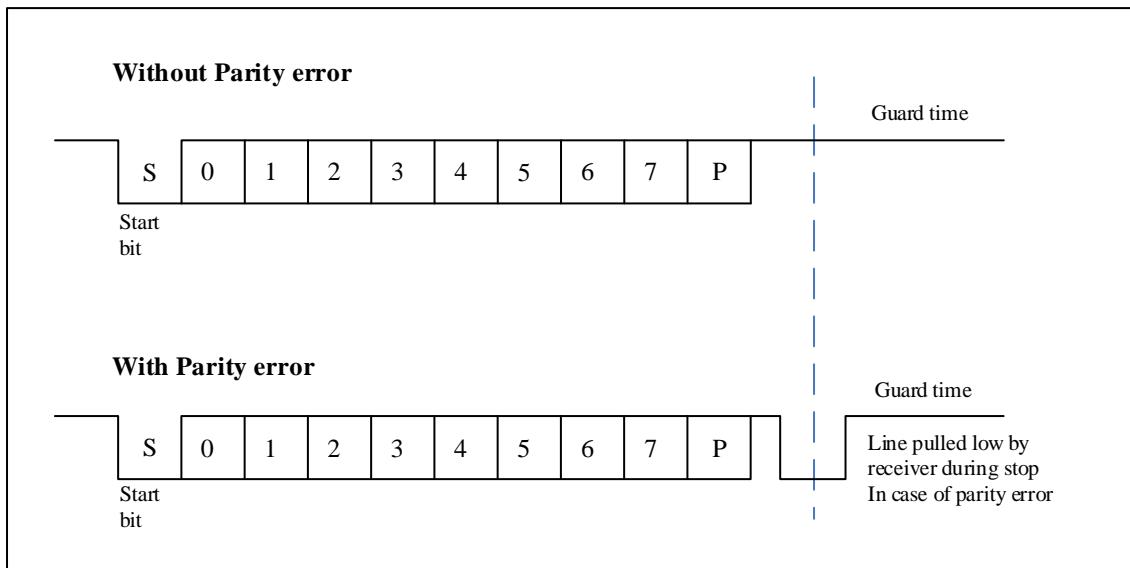
When a parity error is detected by receiver, the transmit data line is pulled low for one baud clock cycle at the end of

the stop bit as NACK signal (If USART_CTRL3.SCNACK 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 17-22 ISO7816-3 Asynchronous Protocol



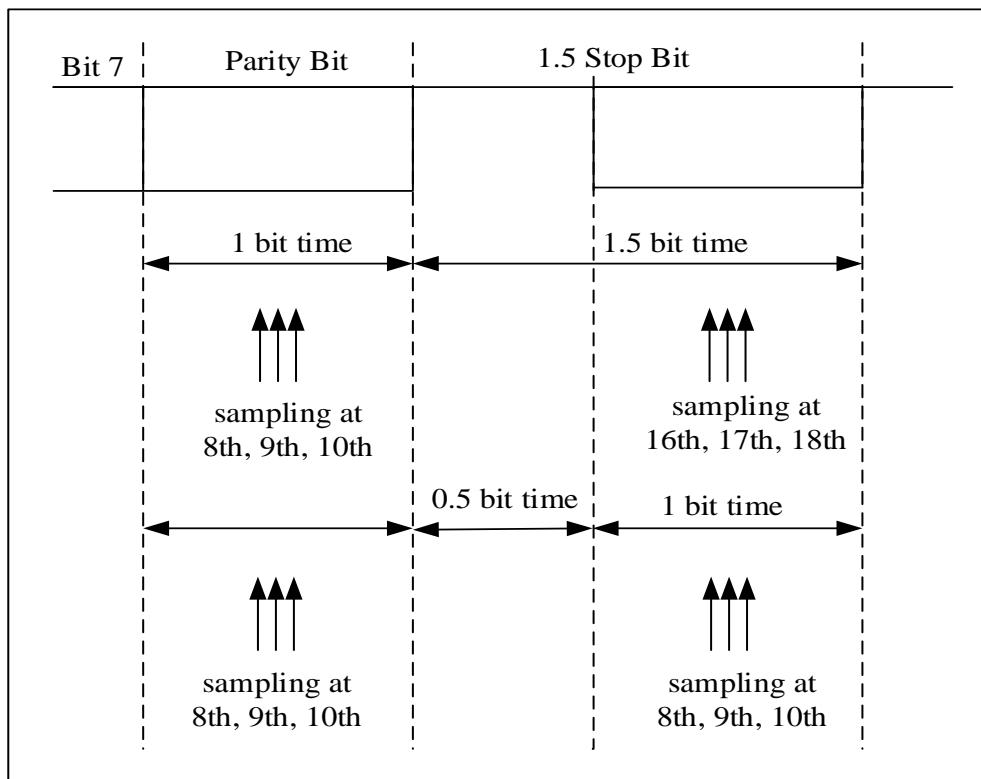
The break frame has no meaning in smart card mode. A 00h data with a framing error will be treated as data instead of a break symbol.

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 17-23 Use 1.5 stop bits to detect parity errors



17.5 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 17-7 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 ⁽¹⁾
--	--	--------------	-----------------------

(1) This flag bit is used only when DMA is used to receive data(USART_CTRL3.DMARXEN=1).

17.6 Mode support

Table 17-8 USART mode setting⁽¹⁾

Communication mode	UART1	USART2	UART4
Asynchronous mode	Y	Y	Y
Hardware flow control mode	N	Y	N
DMA communication mode	Y	Y	Y
Multiprocessor	Y	Y	Y
Synchronous mode	N	Y	N
Smartcard mode	N	Y	N
Single-wire half duplex mode	Y	Y	Y
IrDA infrared mode	Y	Y	Y
LIN	Y	Y	Y

(1) Y = support this mode, N = do not support this mode

17.7 USART register

17.7.1 USART register overview

Table 17-9 USART register overview

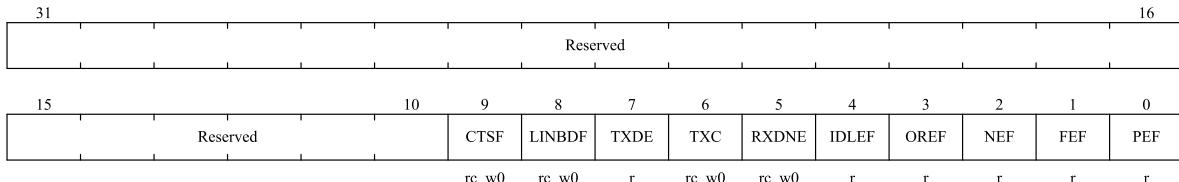
Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	USART_STS	Reserved																		CTSF	9	8	7	6	5	4	3	2	1	0			
	Reset Value																			0	0	1	1	0	0	0	0	0	0	0			
004h	USART_DAT	Reserved																		DATV[8:0]													
	Reset Value																			0	0	0	0	0	0	0	0	0	0	0			
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		

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	
00Ch	USART_CTRL1																																	
	Reset Value																																	
010h	USART_CTRL2																																	
	Reset Value																																	
014h	USART_CTRL3																																	
	Reset Value																																	
018h	USART_GTP																																	
	Reset Value																																	

17.7.2 USART Status register (USART_STS)

Address offset : 0x00

Reset value : 0x0000 00C0



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> <p><i>Note: This bit is invalid for UART1/4.</i></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	<p>The Transmit data register empty.</p> <p>Set to 1 after power-on reset or data to be sent has been sent to the shift register.</p> <p>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.</p> <p>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.</p>

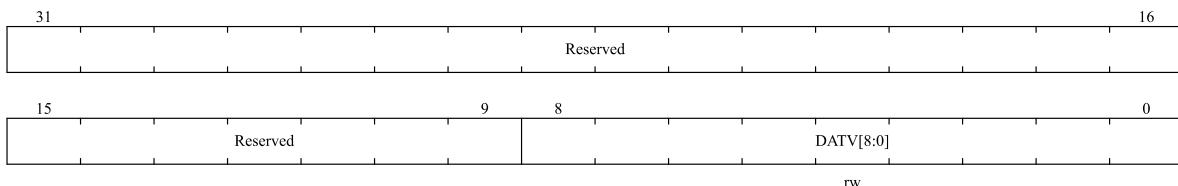
		1: Send completed.
5	RXDNE	<p>The Read data register not empty.</p> <p>This bit is set when the read data buffer receives data from the shift register. When USART_CTRL1.RXDNEIEN bit is set, an interrupt will be generated. Software can clear this bit by writing 0 to it or reading the USART_DAT register.</p> <p>0: The read data buffer is empty.</p> <p>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. The software can clear this bit by reading USART_STS first and then reading USART_DAT.</p> <p>0: No idle frame detected.</p> <p>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.</p> <p>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.</p> <p>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 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.</p> <p>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.</p> <p>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. 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>

		<p><i>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.</p> <p>This bit is set when the parity bit of the received data frame is different from the expected check value.</p> <p>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>

17.7.3 USART Data register (USART_DAT)

Address offset : 0x04

Reset value : undefined (uncertain value)



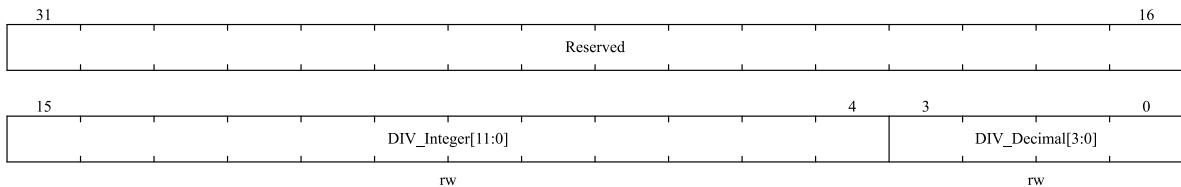
Bit field	Name	Description
31:9	Reserved	Reserved, the reset value must be maintained
8:0	DATV[8:0]	<p>Data value</p> <p>Contains the data sent or received; Software can change the transmitted data by writing these bits, or read the values of these bits to obtain the received data.</p> <p>If parity is 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>

17.7.4 USART Baud rate register (USART_BRCF)

Address offset : 0x08

Reset value : 0x0000 0000

Note: The baud counter stops counting if USART_CTRL1.TXEN or USART_CTRL1.RXEN are disabled respectively.

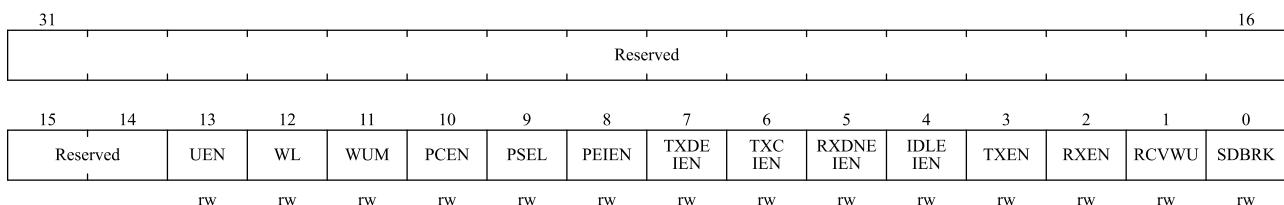


Bit field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained
15:4	DIV_Integer[11:0]	Integer part of baud rate divider.
3:0	DIV_Decimal[3:0]	Fractional part of baud rate divider.

17.7.5 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 this bit.</p> <p>0:USART is disabled.</p> <p>1:USART is enabled.</p>
12	WL	<p>Word length.</p> <p>0:8 data bits.</p> <p>1:9 data bits.</p> <p><i>Note: If data is in transit, this bit cannot be configured.</i></p>
11	WUM	<p>Wake up mode from mute mode.</p> <p>0: Idle frame wake-up.</p> <p>1: the address identifier wakes up.</p>
10	PCEN	<p>Parity control enable</p> <p>0: Verification control is disabled.</p> <p>1: Verification control is enabled.</p>
9	PSEL	<p>Parity selection.</p> <p>0: even check.</p>

Bit field	Name	Description
		1: odd check.
8	PEIEN	<p>PE interrupt enable</p> <p>If this bit is set to 1, an interrupt is generated when USART_STS.PEF bit is set.</p> <p>0: Parity error interrupt is disabled.</p> <p>1: Parity error interrupt is enabled.</p>
7	TXDEIEN	<p>TXDE interrupt enable</p> <p>If this bit is set to 1, an interrupt is generated when USART_STS.TXDE bit is set.</p> <p>0: Send buffer empty interrupt is disabled.</p> <p>1: Send buffer empty interrupt is enabled.</p>
6	TXCIEN	<p>Transmit complete interrupt enable.</p> <p>If this bit is set to 1, an interrupt is generated when USART_STS.TXC is set.</p> <p>0: Transmission completion interrupt is disabled.</p> <p>1: Transmission completion interrupt is enabled.</p>
5	RXDNEIEN	<p>RXDNE interrupt enable</p> <p>If this bit is set to 1, an interrupt is generated when USART_STS.RXDNE or USART_STS.OREF is set.</p> <p>0: Data buffer non-empty interrupt o and overrun error interrupt are disabled.</p> <p>1: Data buffer non-empty interrupt o and overrun error interrupt are enabled.</p>
4	IDLEIEN	<p>IDLE interrupt enable.</p> <p>If this bit is set to 1, an interrupt is generated when USART_STS.IDLEF is set.</p> <p>0: IDLE line detection interrupt is disabled.</p> <p>1: IDLE line detection interrupt is enabled.</p>
3	TXEN	<p>Transmitter enable.</p> <p>0: The transmitter is disabled.</p> <p>1: the transmitter is enabled.</p>
2	RXEN	<p>Receiver enable</p> <p>0: The receiver is disabled.</p> <p>1: the receiver is enabled.</p>
1	RCVWU	<p>The receiver wakes up</p> <p>Software can set this bit to 1 to make USART enter mute mode, and clear this bit to 0 to wake up USART.</p> <p>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.</p> <p>0: The receiver is in normal operation mode.</p> <p>1: The receiver is in mute mode.</p>
0	SDBRK	<p>Send Break Character.</p> <p>The software transmits a break character by setting this bit to 1.</p> <p>This bit is cleared by hardware during stop bit of the break frame transmission.</p> <p>0: No break character was sent.</p> <p>1: Send a break character.</p>

17.7.6 USART control register 2 register (USART_CTRL2)

Address offset : 0x10

Reset value : 0x0000 0000

31	Reserved														16
15	14	13	12	11	10	9	8	7	6	5	4	3	0	ADDR[3:0]	
Reserved	LINMEN	STPB[1:0]	CLKEN	CLKPOL	CLKPHA	LBCLK	Reserved	LINBD IEN	LINBDL	Reserved				rw	

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. <i>Note: For UART1/4, only one stop bit and two stop bits are valid.</i>
11	CLKEN	Clock enable 0:CK pin is disabled 1:CK pin enabled <i>Note: This bit cannot be used for UART1/4.</i>
10	CLKPOL	Clock polarity. This bit is used to set the polarity of CK pin in synchronous mode. 0: CK pin remains low when it is not transmitted to the outside. 1: CK pin remains high when it is not sent to the outside. <i>Note: This bit is invalid for UART1/4.</i>
9	CLKPHA	Clock phase. This bit is used to set the phase of CK pin in synchronous mode. 0: Sample the first data at the first clock edge. 1: Sample the first data at the second clock edge. <i>Note: This bit cannot be used for UART1/4.</i>
8	LBCLK	The Last bit clock pulse. 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. 0: The clock pulse of the last bit of data is not output from CK.

Bit field	Name	Description
		1: The clock pulse of the last bit of data will be output from CK. <i>Note: This bit cannot be used for UART1/4.</i>
7	Reserved	Reserved, the reset value must be maintained
6	LINBDIEN	LIN break detection interrupt enable. If this bit is set to 1, an interrupt will be generated when USART_STS.LBDF bit is set. 0: Disconnect signal detection interrupt is disabled. 1: Turn-off signal detection interrupt enabled
5	LINBDL	LIN break detection length. This bit is used to set the length of the break frame. 0:10 bit break detection 1:11 bit break detection <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>
4	Reserved	Reserved, the reset value must be maintained
3:0	ADDR[3:0]	USART address. Used in the mute mode of multiprocessor communication, using address identification to wake up a USART device. 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.

17.7.7 USART control register 3 register (USART_CTRL3)

Address offset : 0x14

Reset value : 0x0000 0000

31	Reserved														16
15	Reserved	11	10	9	8	7	6	5	4	3	2	1	0		
			rw												

Bit field	Name	Description
31:11	Reserved	Reserved, the reset value must be maintained
10	CTSIEN	CTS interrupt enable. If this bit is set to 1, an interrupt will be generated when USART_STS.CTSF bit is set. 0:CTS interrupt is disabled. 1:CTS interrupt is enabled.

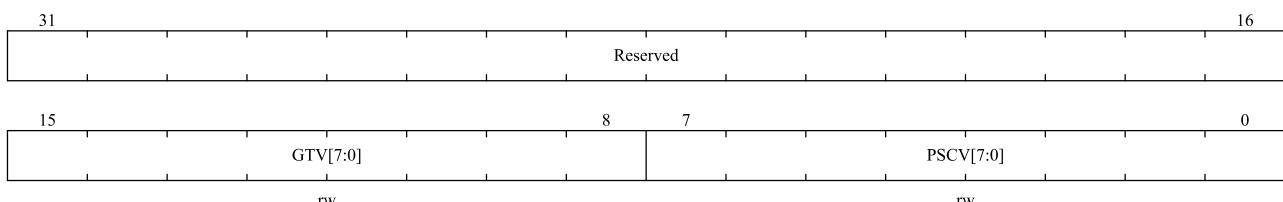
Bit field	Name	Description
		<i>Note: This bit cannot be used for UART1/4</i>
9	CTSEN	<p>CTS enable.</p> <p>This bit is used to enable the CTS hardware flow control function.</p> <p>0:CTS hardware flow control is disabled.</p> <p>1:CTS hardware flow control is enabled.</p> <p><i>Note: This bit cannot be used for UART1/4</i></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> <p><i>Note: This bit cannot be used for UART1/4</i></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> <p><i>Note: This bit cannot be used for UART1/4</i></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> <p><i>Note: This bit cannot be used for UART1/4</i></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> <p>0: Normal mode.</p> <p>1: Low power consumption mode.</p>
1	IRDAMEN	<p>IrDA mode enable.</p> <p>0:IrDA is disabled.</p> <p>1:IrDA is enabled.</p>
0	ERRIEN	<p>Error interrupt enable.</p> <p>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</p>

Bit field	Name	Description
		bit is set. 0: Error interrupt is disabled. 1: Error interrupt enabled.

17.7.8 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]	<p>Guard time value in Smartcard mode.</p> <p>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.</p> <p><i>Note: This bit is invalid for UART1/4.</i></p>
7:0	PSCV[7:0]	<p>Prescaler value.</p> <p>In IrDA low power consumption mode:</p> <p>these bits are used to set the frequency division coefficient for dividing the peripheral clock (PCLK1/PCLK2) to generate low power consumption frequency.</p> <p>00000000: reserved-do not write this value.</p> <p>00000001: divide the source clock by 1.</p> <p>...</p> <p>11111111: divide the source clock by 255.</p> <p>In IrDA normal mode:</p> <p>PSCV can only be set to 00000001.</p> <p>In Smartcard mode:</p> <p>PSCV[4:0] is used to set the frequency division of Smartcard clock generated by 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> <p><i>Note: This bit is invalid for UART1/4.</i></p>
--	--	--

18 Serial peripheral interface/Inter-IC Sound (SPI/ I²S)

18.1 SPI/ I²S 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.

Both of them are synchronous serial interface communication protocols.

18.2 SPI and I²S main features

18.2.1 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 functionality.

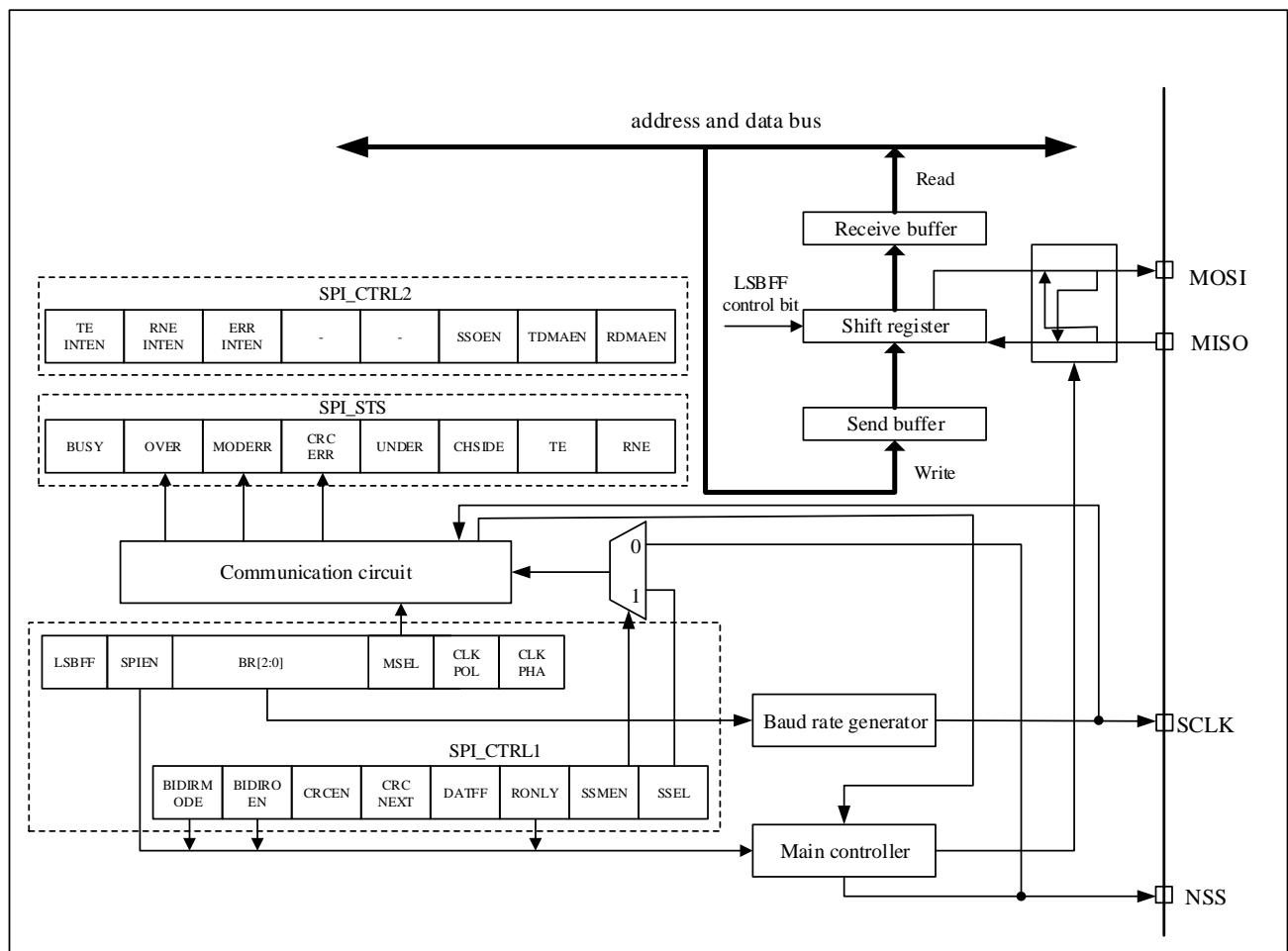
18.2.2 I²S features

- Simplex synchronous mode.
- Supports master mode and slave mode operation.
- Four audio standards are supported: Philips I²S 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 functionality.

18.3 SPI function description

18.3.1 General description

Figure 18-1 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.

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).

Hardware NSS mode

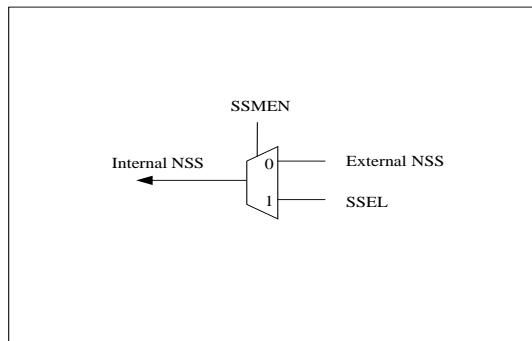
The software slave device management is disabled when SPI_CTRL1.SSMEN=0.

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.

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.

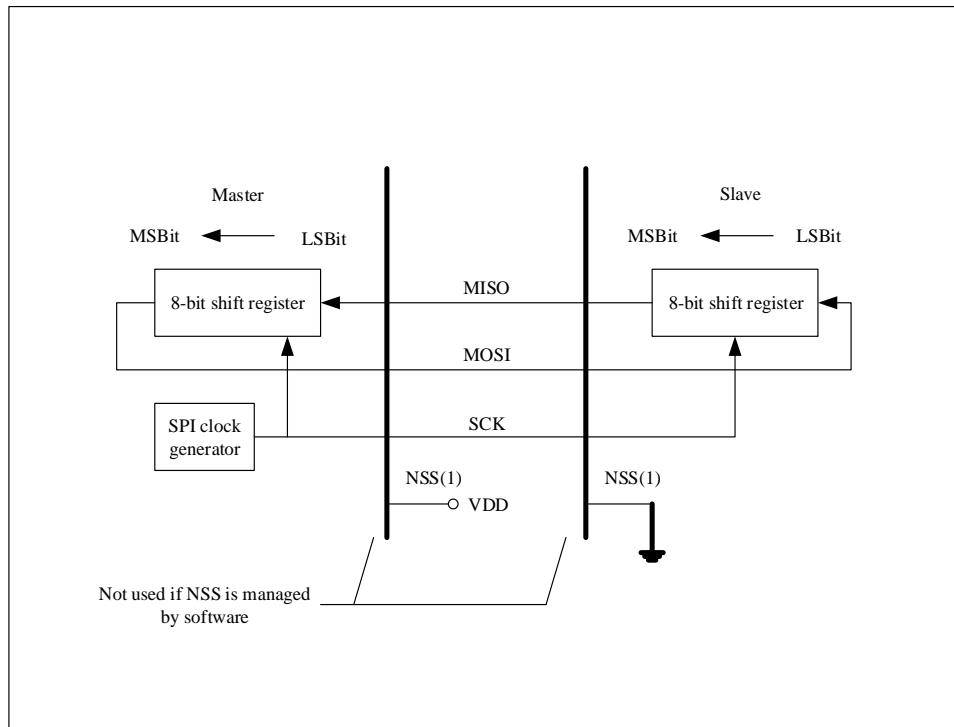
Note: The choice of software mode or hardware mode depends on whether NSS control is needed in the communication protocol. If not, you can choose the software mode, and release a GPIO pin for other purposes.

Figure 18-2 Selective management of hardware/software



The following figure is an example of the interconnection of single master and single slave devices

Figure 18-3 Master and slave applications



Note: NSS pin is set as input

SPI is a ring bus structure, and the master device outputs a synchronous clock signal through SCK pin, the MOSI pin of master device is connected with the MOSI pin of slave device, and the MISO pin of master device is connected with the MISO pin of slave device. Serial transfer of data between master device and slave device, sending data to slave device through MOSI pin and having the lowest bit, while the highest bit of slave device is transmitted to the lowest bit of master device through MISO pin. When the second bit of data is sent, the data of the lowest bit will be shifted to the left by one bit and the new data will be stored in the lowest bit.

SPI timing mode

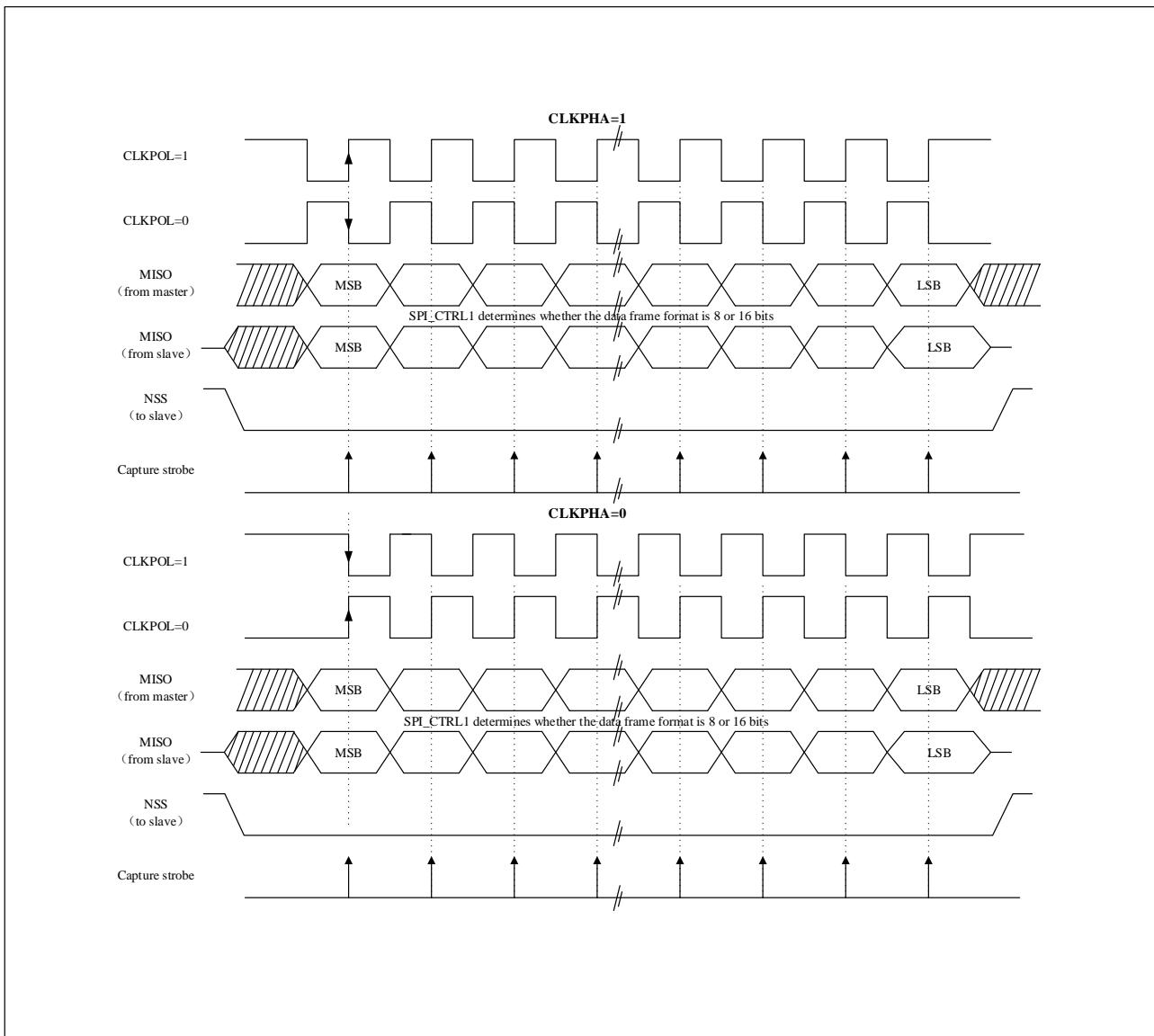
User can selects the clock edge of data capture by setting SPI_CTRL1.CLKPOL bit and SPI_CTRL1.CLKPHA bit.

- 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 18-4 is the combination timing of four CLKPHA and CLKPOL bits transmitted by SPI when the SPI_CTRL1.LSBFF=0.

Figure 18-4 Data clock timing diagram



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.

18.3.2 SPI work mode

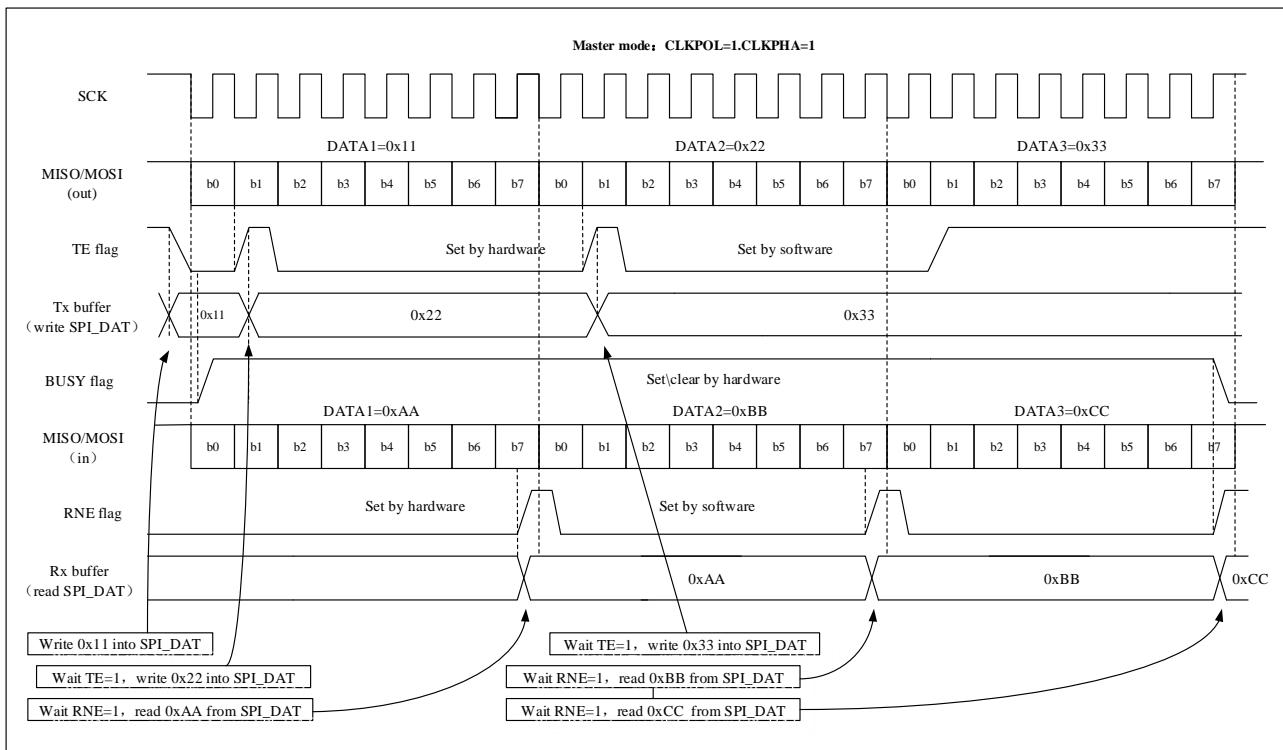
■ Master full duplex mode (SPI_CTRL1.MSEL=1, SPI_CTRL1.BIDIRMODE=0, SPI_CTRL1.RONLY=0)

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. 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. 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 18-5 Schematic diagram of the change of TE/RNE/BUSY when the host is continuously transmitting in full duplex mode



■ **Master two-wire one-way send-only mode (SPI_CTRL1.MSEL=1, SPI_CTRL1.BIDIRMODE=0, SPI_CTRL1.RONLY=0)**

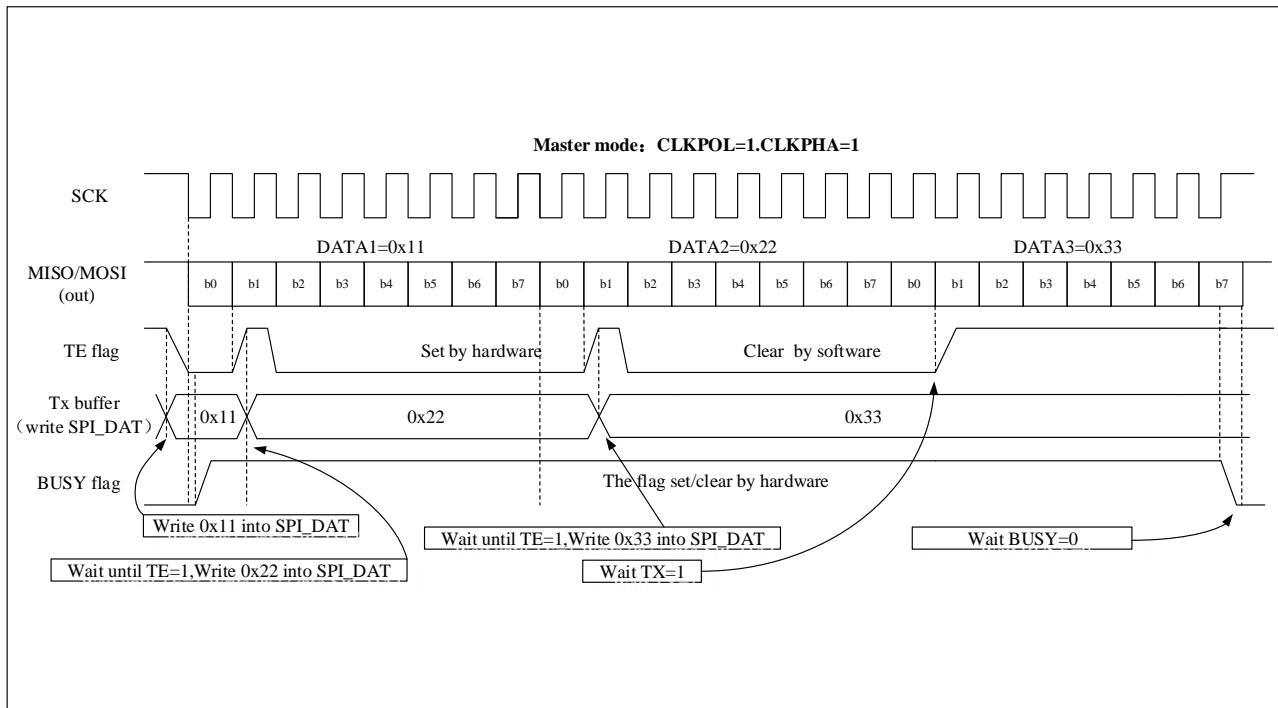
Master two-wire one-way send-only mode is similar to master full-duplex mode. The difference is that this mode will not read the received data, so the SPI_STS.OVER bit will be set to '1', and the software will ignore it. The software operation process is as follows:

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 18-6 Schematic diagram of TE/BUSY change when the host transmits continuously in one-way only mode



■ **Master two-wire one-way receive-only mode (SPI_CTRL1.MSEL=1, SPI_CTRL1.BIDIRMODE=0, SPI_CTRL1.RONLY=1)**

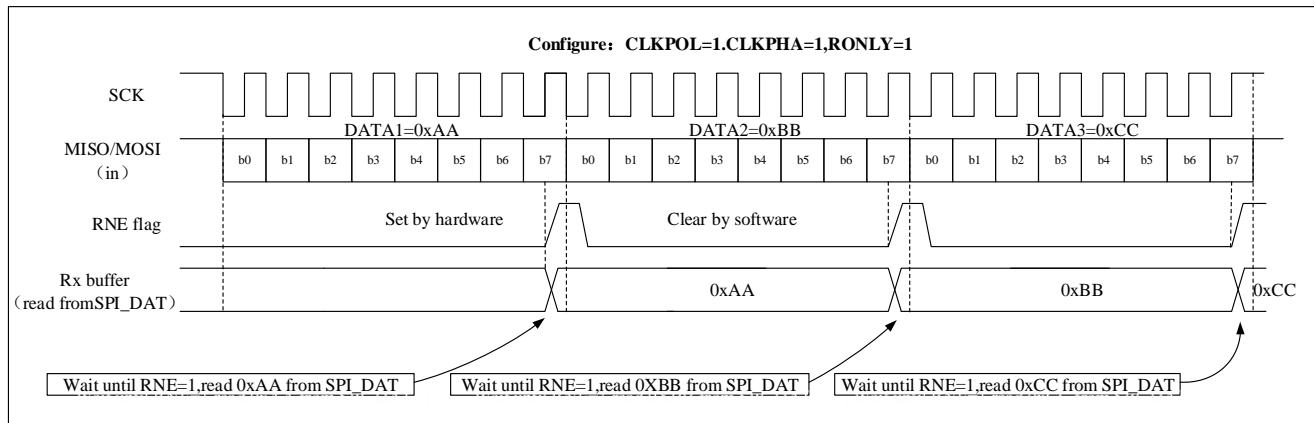
When SPI_CTRL1.SPIEN=1, the receiving process starts. The data bits from the MISO pin are sequentially shifted into the 8-bit shift register and then loaded into the SPI_DAT register(receive buffer) in parallel. The software operation process is as follows:

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(SPI_STS.RNE).

Figure 18-7 Schematic diagram of RNE change when continuous transmission occurs in receive-only mode (BIDIRMODE=0

and RONLY=1)



- **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 send, the data to be sent is loaded into the 8-bit 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.

- **Master one-wire bidirectional receive mode (SPI_CTRL1.MSEL=1, SPI_CTRL1.BIDIRMODE=1, SPI_CTRL1.BIDIROEN=0, SPI_CTRL1.RONLY=0)**

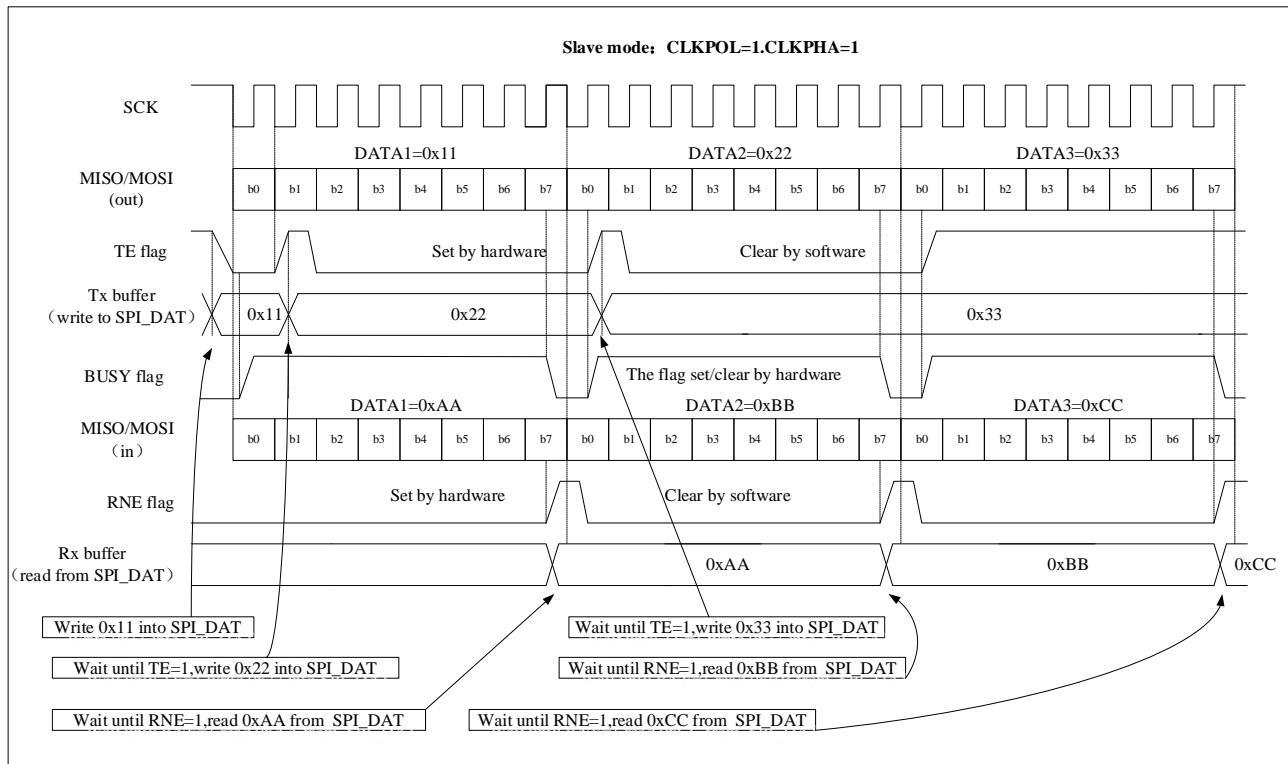
When SPI_CTRL1.SPIEN=1, the receiving process starts. There is no data output in this mode, the received data bits are sequentially and serially shifted into the 8-bit shift register, and then loaded into the SPI_DAT register (receive buffer) in parallel

The software operation flow of the master one-wire bidirectional receive mode is the same as that of the receive-only mode.

- **Slave full duplex mode (SPI_CTRL1.MSEL=0, SPI_CTRL1.BIDIRMODE=0, SPI_CTRL1.RONLY=0)**

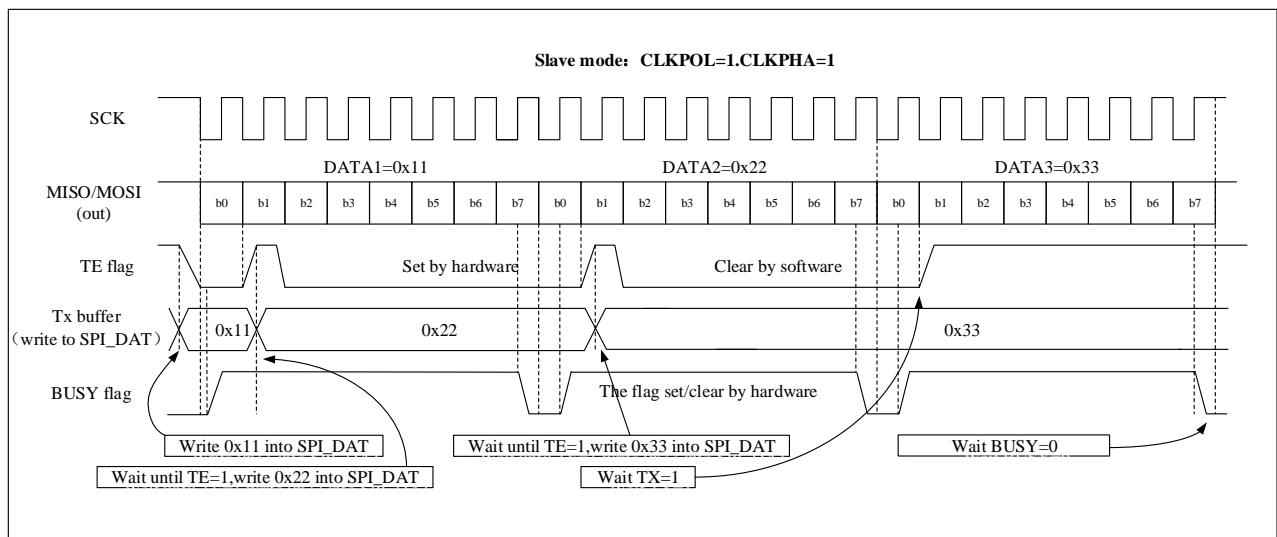
The data transfer process begins when the slave device receives the first clock edge. Before the master starts data transfer, software must ensure that the data to be send is written to the SPI_DAT register.

Figure 18-8 Schematic diagram of the change of TE/RNE/BUSY when the slave is continuously transmitting in full duplex mode



- Slave two-wire one-way send-only mode (SPI_CTRL1.MSEL=0, SPI_CTRL1.BIDIRMODE=0 and SPI_CTRL1.RONLY=0)

Figure 18-9 Schematic diagram of TE/BUSY change during continuous transmission in slave unidirectional transmit-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 8-bit shift register and then loaded into the SPI_DAT register (receive buffer) in parallel.

- **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.

- **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 8-bit 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.

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 18-4).
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.

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 level is low, and the sending process starts when the first edge of the clock signal comes. To avoid accidental data transfers, software must write data to the send buffer before the data sending (it is recommended to enable the SPI slave before the master 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.

Continuous and discontinuous transmission.

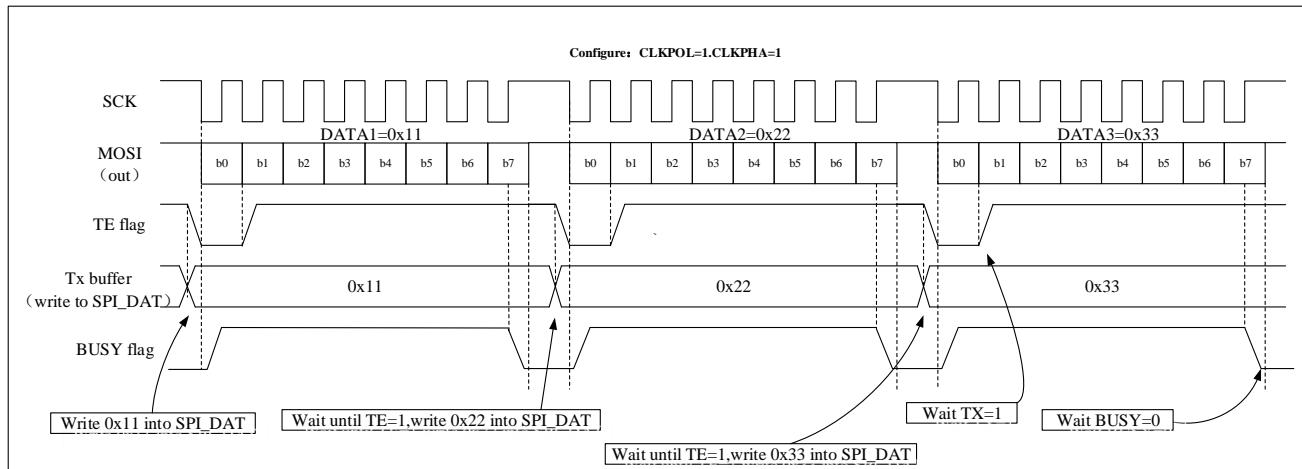
When sending data in master mode, if the software is fast enough to detect each TE(SPI_STS.TE) rising edge (or TE interrupt), and the data is written to the SPI_DAT register immediately before the end of the ongoing transmission. At this time, the SPI clock remains continuous between the transmission 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 18-10 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 18-9).

Figure 18-10 Schematic diagram of TE/BUSY change when BIDIRMODE = 0 and RONLY = 0 are transmitted discontinuously.



18.3.3 Status flag

The SPI_STS register has 3 flag bits to monitor the status of the SPI:

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.

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.

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.

18.3.4 Disabling the SPI

In order to turn off the SPI module, different operation modes require different operation steps:

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).

Two-wire one-way send-only mode or one-wire 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).

Two-wire one-way receive-only mode or one-wire 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).

Two-wire one-way receive-only mode or one-wire 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).

18.3.5 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. At this time, the received data is not read, the OVER flag(SPI_STS.OVER) is set to 1, and the software does not need to process this flag.

When the SPI is only used for receiving data, only the receive DMA channel of the SPI needs to be enabled.

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 18-11 Transmission using DMA

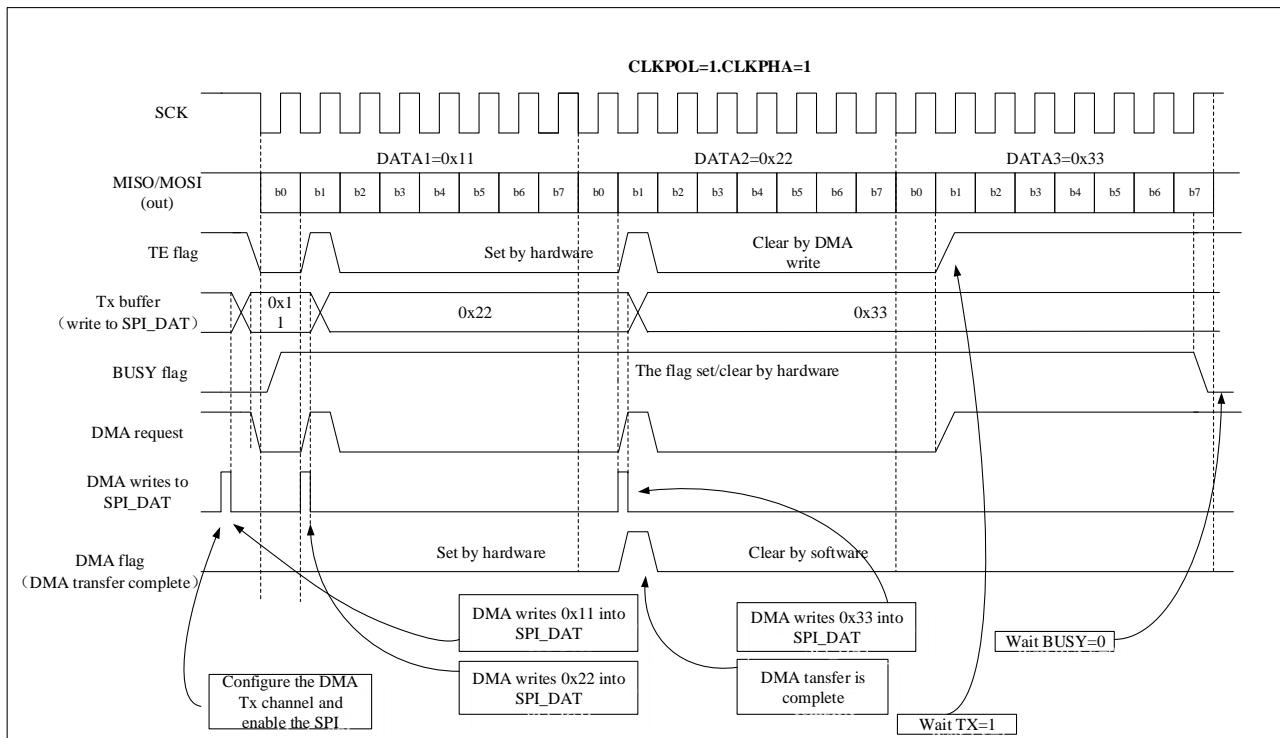
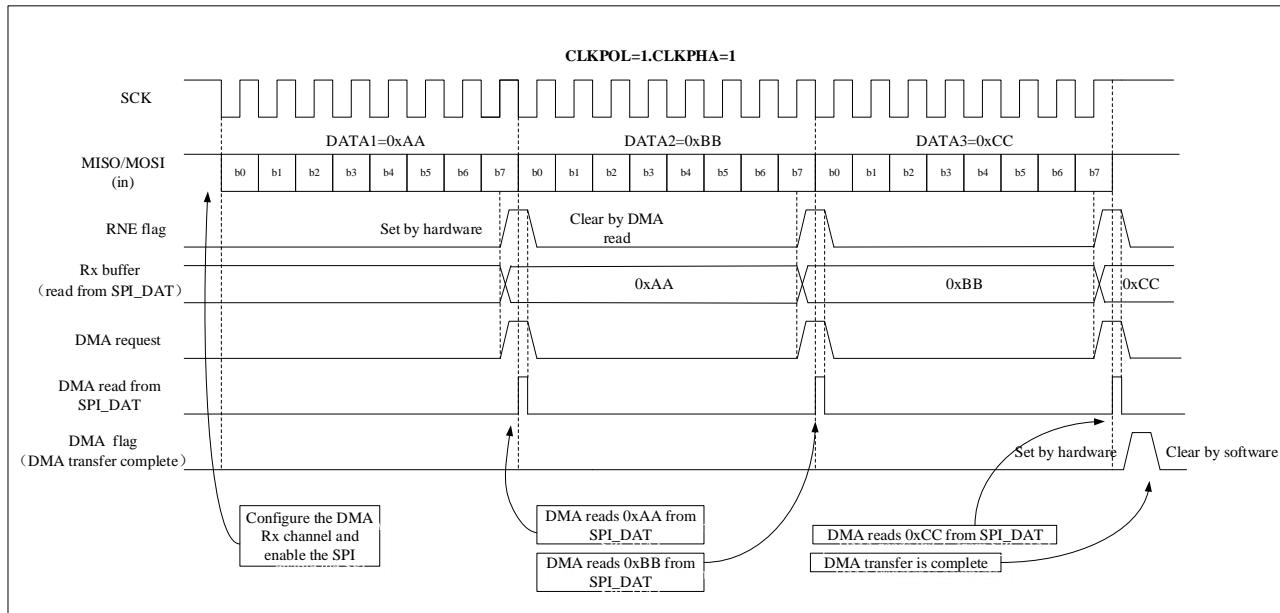


Figure 18-12 Reception using DMA



18.3.6 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.

18.3.7 Error flag

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.

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.

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).

18.3.8 SPI interrupt

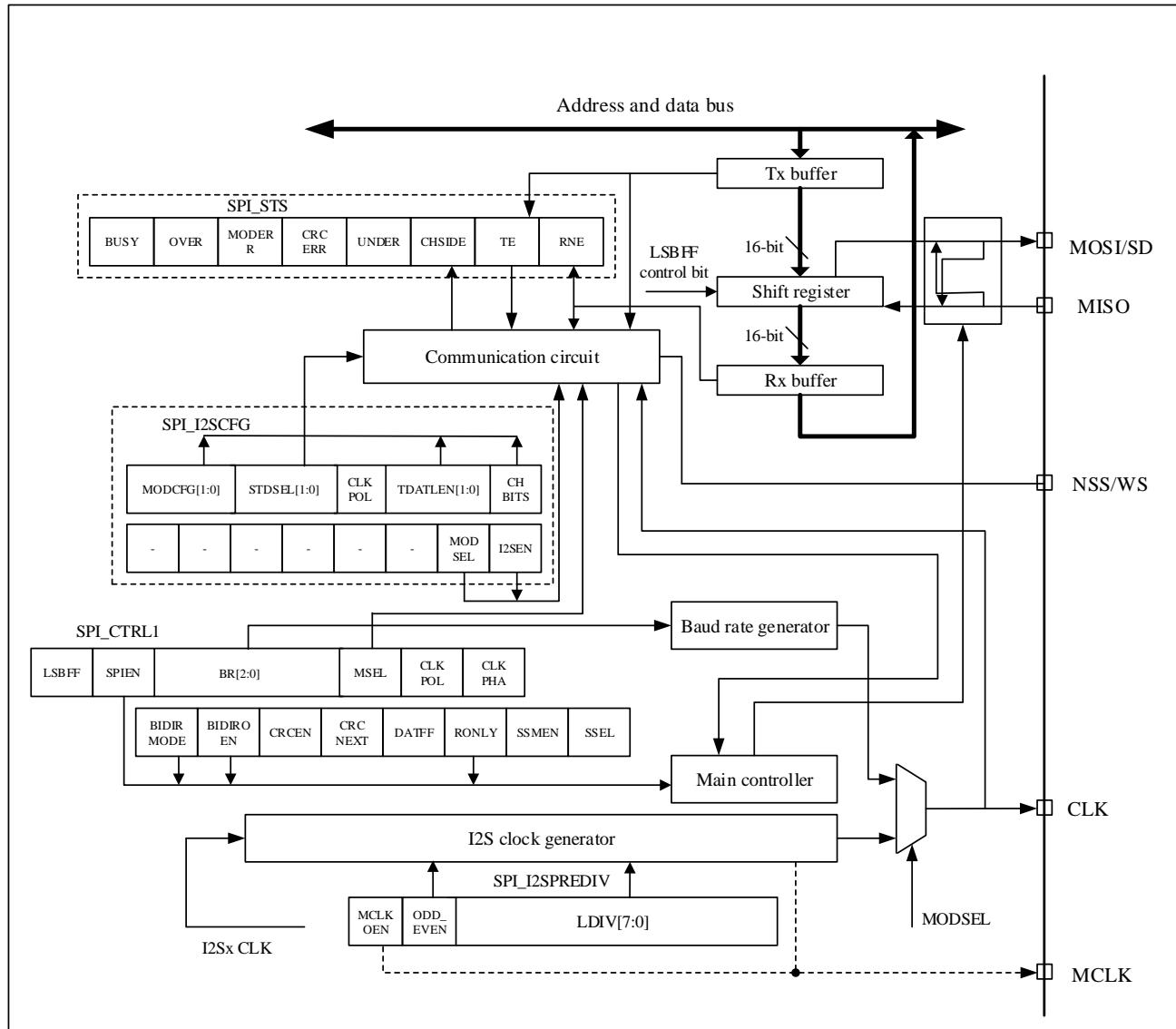
Table 18-1 SPI interrupt request

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	

18.4 I²S function description

The block diagram of I²S is shown in the figure below:

Figure 18-13 I²S block diagram



The I²S interface uses the same pins, flags and interrupts as the SPI interface. Setting the SPI_I2SCFG.MODSEL=1 selects the I²S audio interface.

I²S 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).

18.4.1 Supported audio protocols

Four audio standards can be selected by setting the SPI_I2SCFG.STDSEL[1:0] bits:

- I²S 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.

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 18-14 I²S Philips protocol waveform (16/32-bit full precision, CLKPOL = 0)

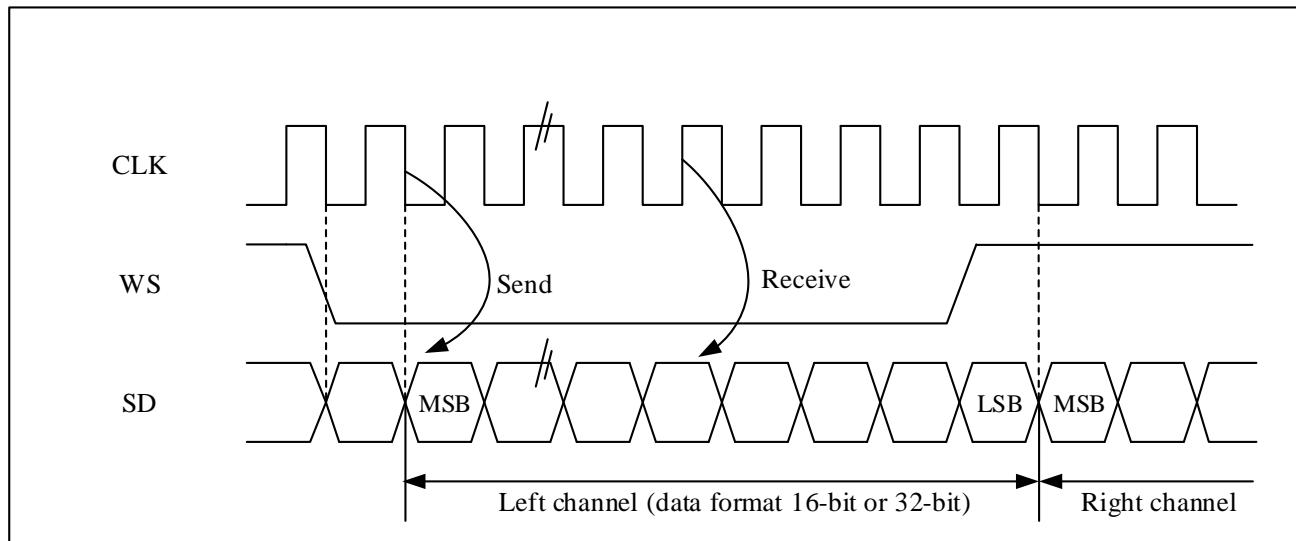
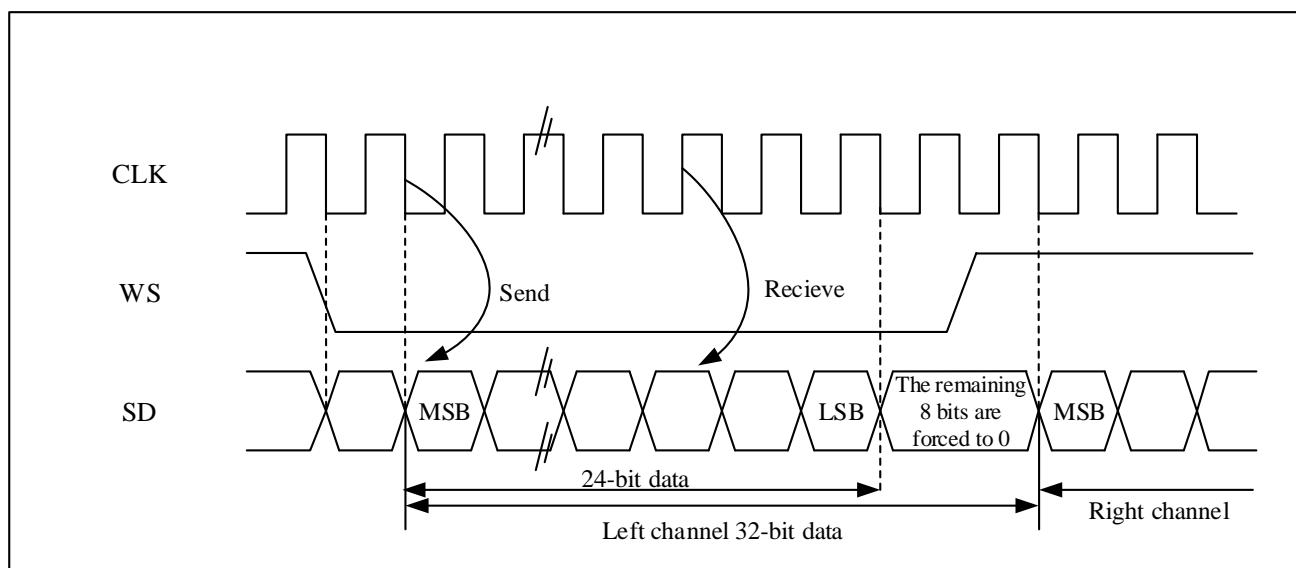
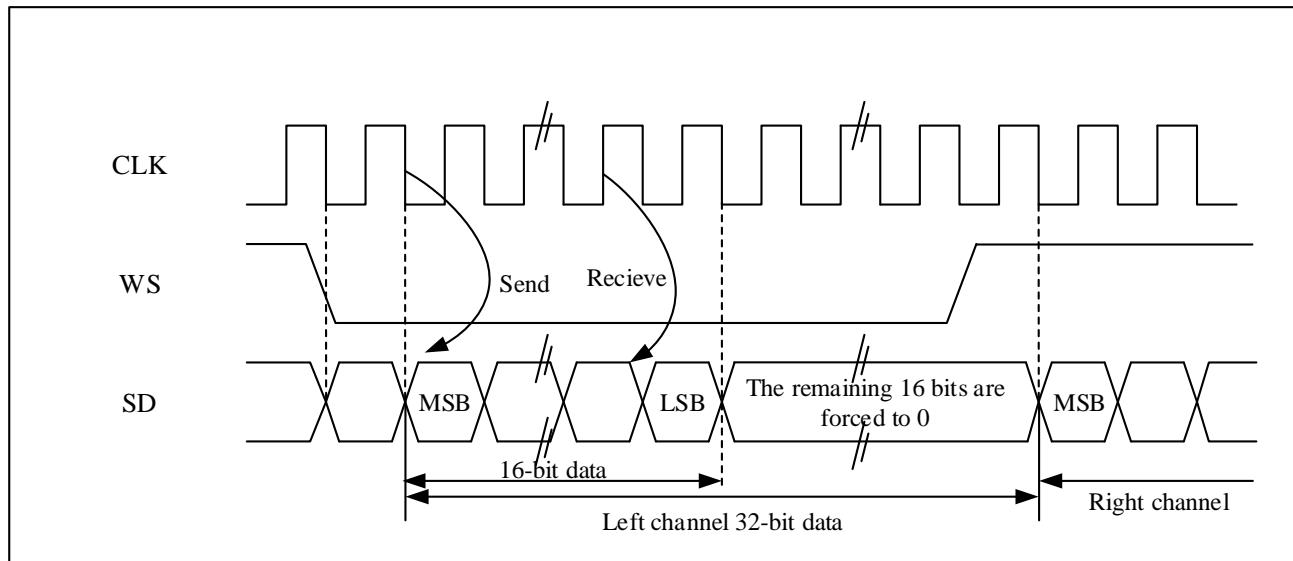


Figure 18-15 I²S Philips protocol standard waveform (24-bit frame, CLKPOL = 0)



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 18-16 I²S 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.

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 I²S Philips standard.

Figure 18-17 The MSB is aligned with 16-bit or 32-bit full precision, CLKPOL = 0.

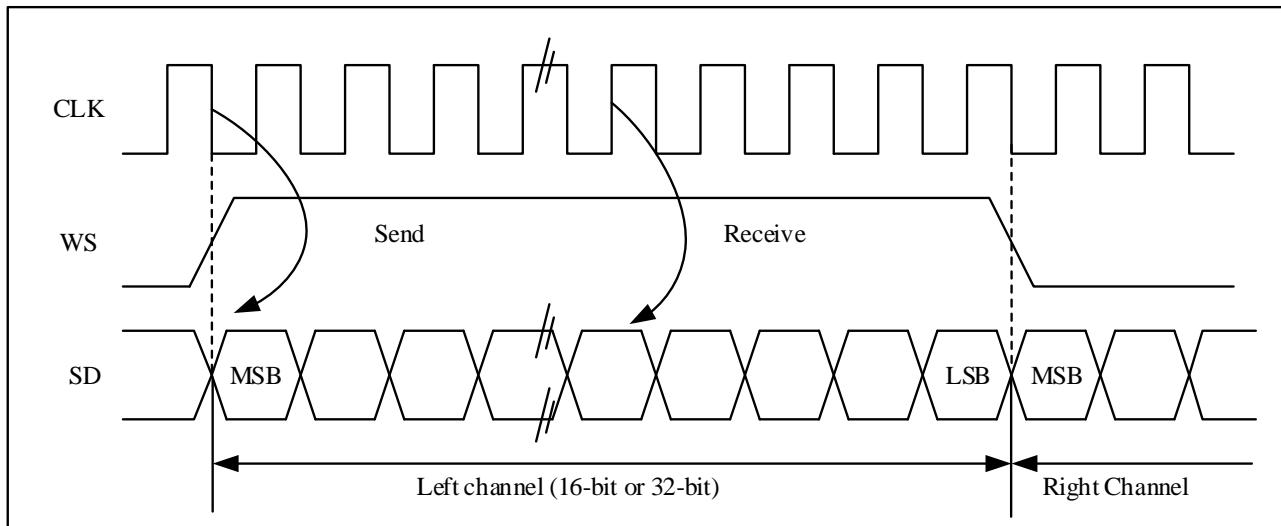


Figure 18-18 MSB aligns 24-bit data, CLKPOL = 0

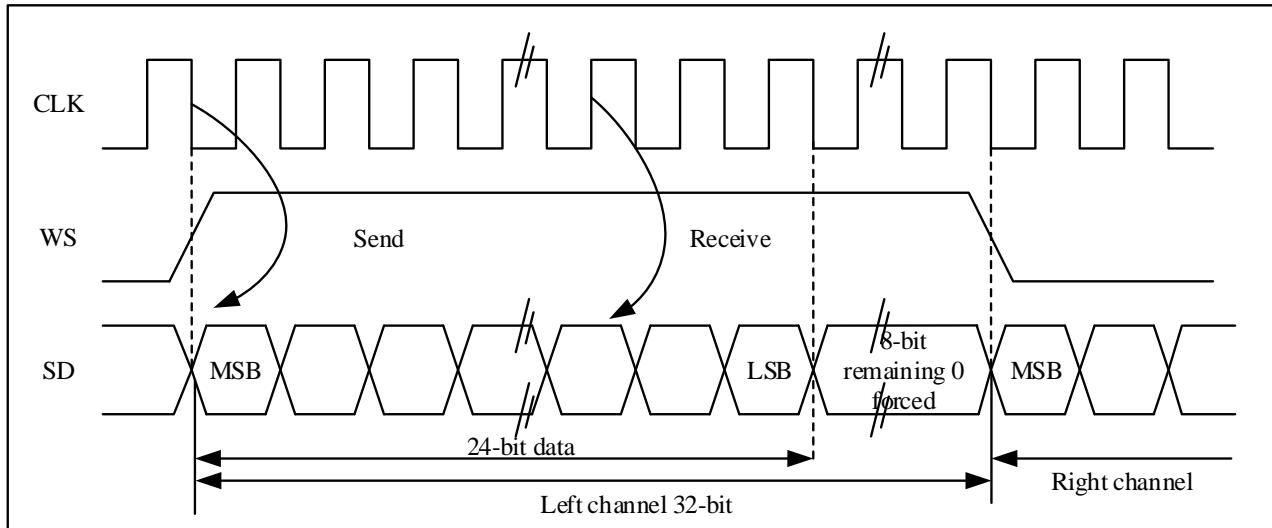
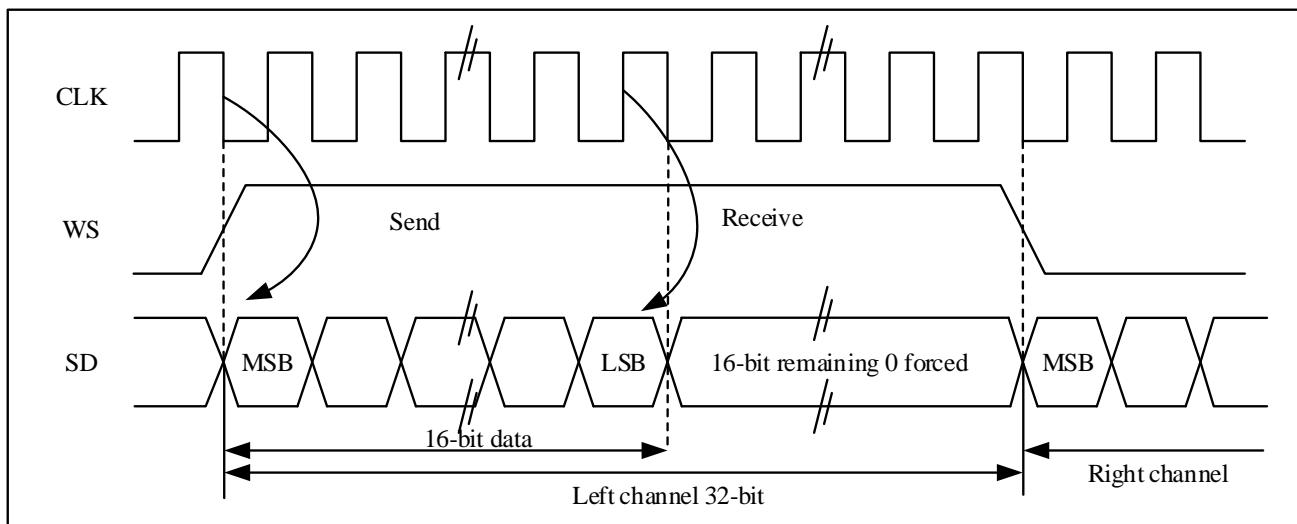


Figure 18-19 MSB-aligned 16-bit data is extended to 32-bit packet frame, CLKPOL = 0



LSB alignment standard

In 16-bit or 32-bit full-precision frame format, LSB alignment standard is the same as MSB alignment standard.

Figure 18-20 LSB alignment 16-bit or 32-bit full precision, CLKPOL = 0

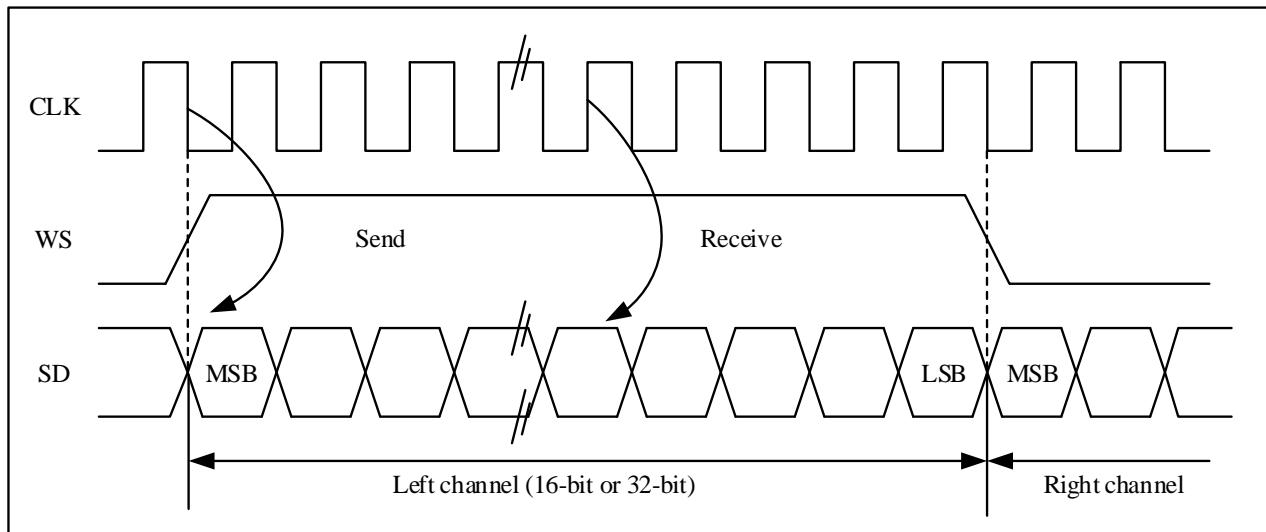
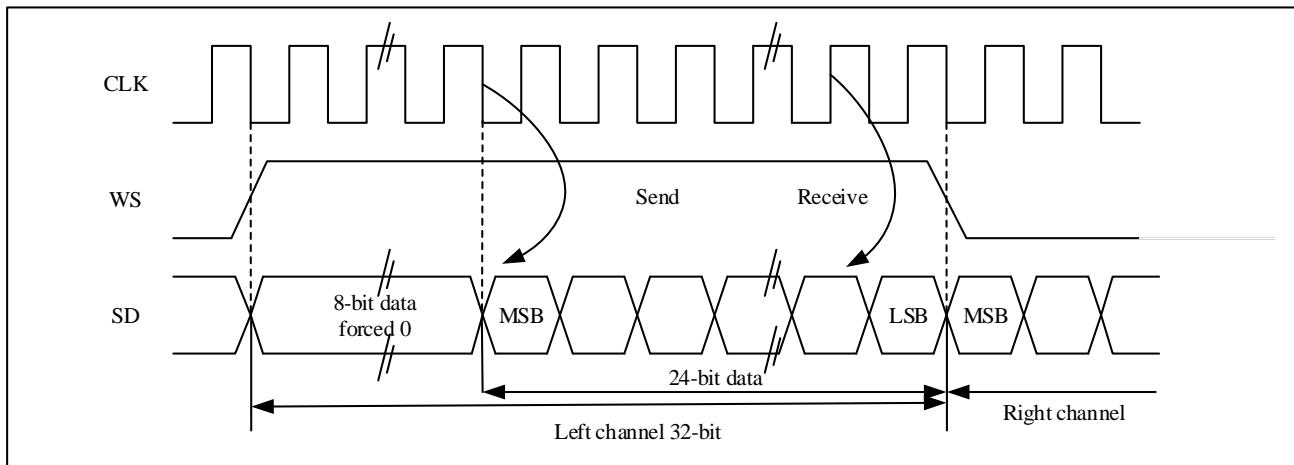
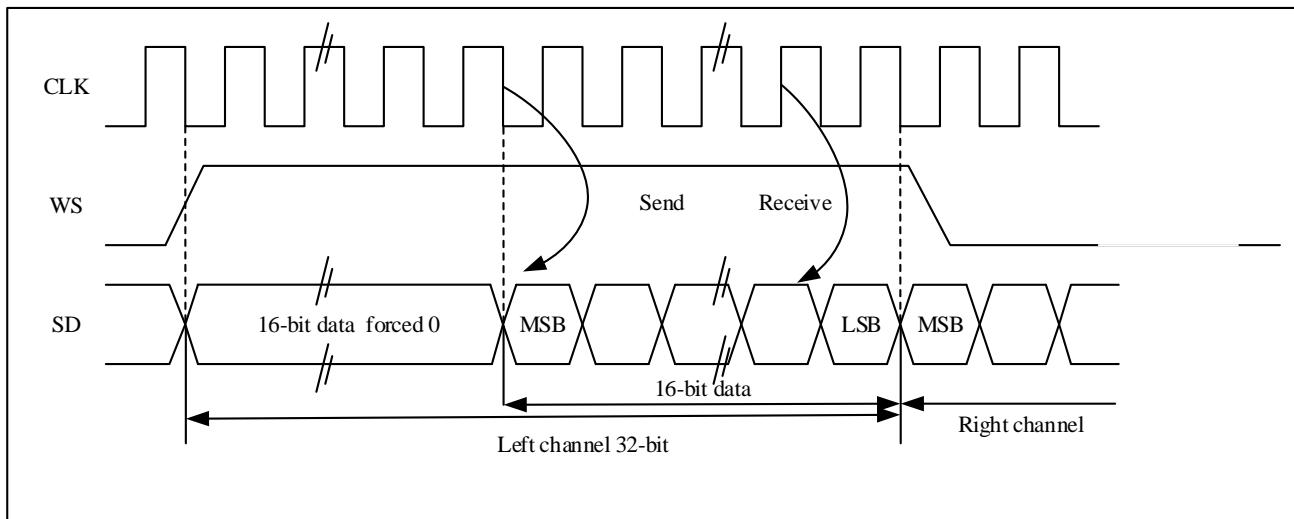


Figure 18-21 LSB aligns 24-bit data, CLKPOL = 0



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 0x0095 (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 18-22 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.

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 I²S Philips standard.

Figure 18-23 PCM standard waveform (16 bits)

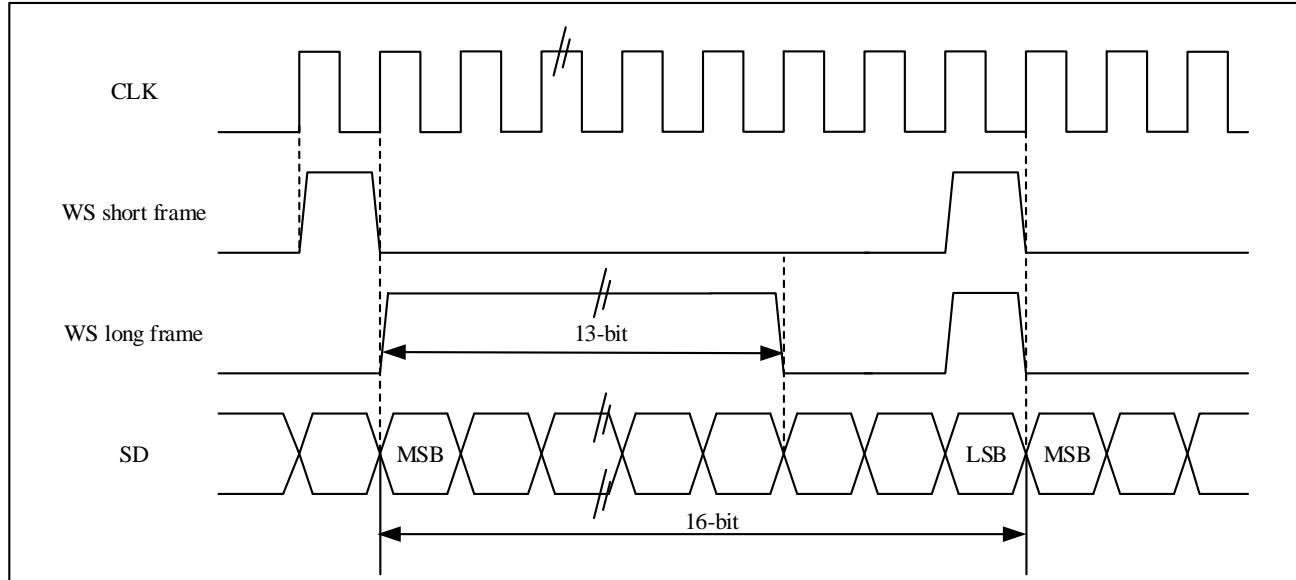
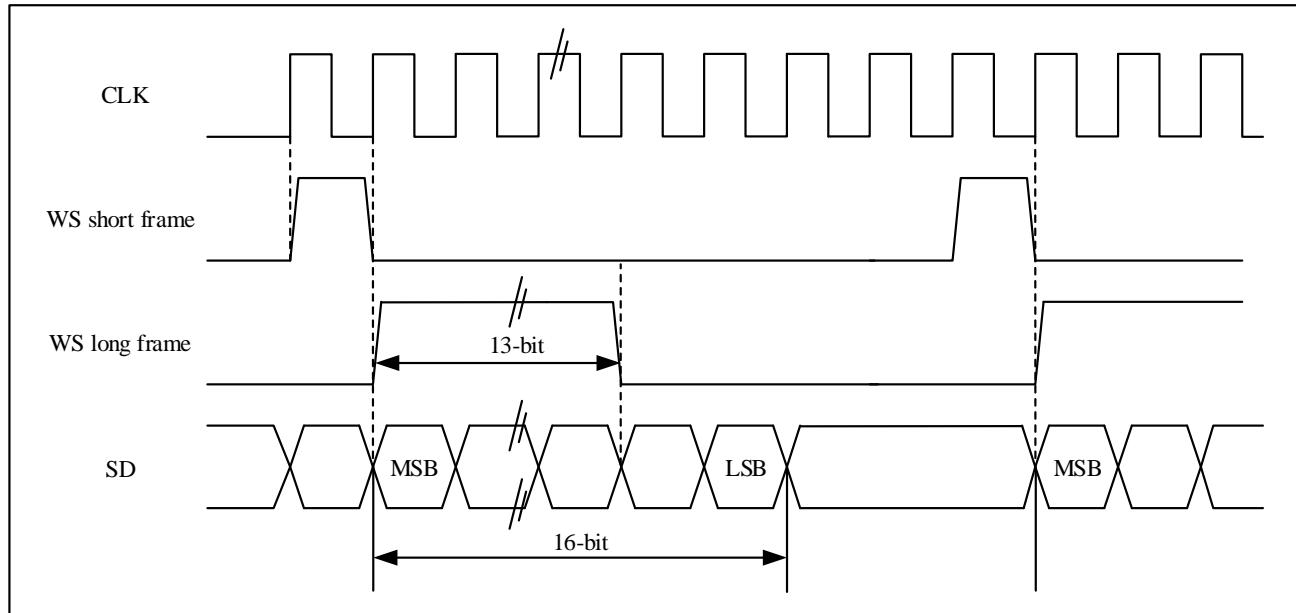


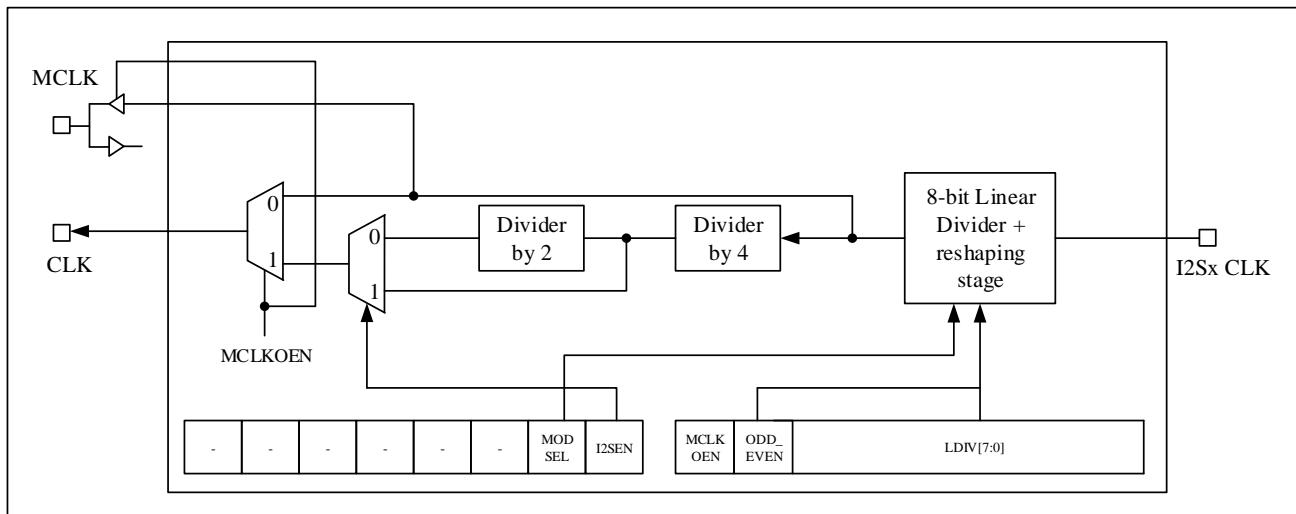
Figure 18-24 PCM standard waveform (16-bit extended to 32-bit packet frame)



18.4.2 Clock generator

In the master mode, the linear divider needs to be set correctly in order to obtain the desired audio frequency.

Figure 18-25 I²S clock generator structure



Note: The clock source of I²Sx CLK is HSI, HSE or PLL system clock that drives AHB clock.

The bit rate of I²S determines the data flow on the I²S data line and the frequency of the I²S clock signal.

$$\text{I}^2\text{S bit rate} = \text{number of bits per channel} \times \text{number of channels} \times \text{audio sampling frequency}$$

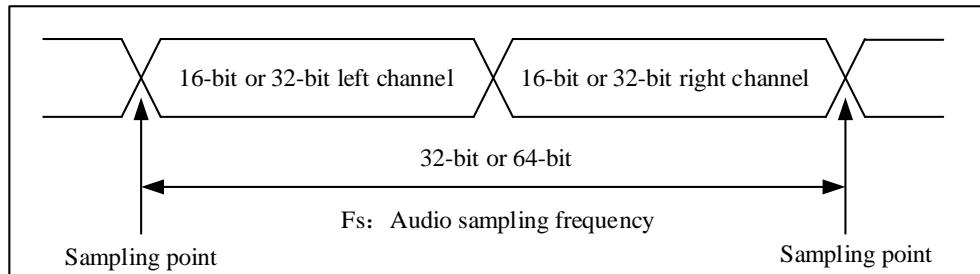
For a signal with left and right channels and 16-bit audio, the I²S bit rate is calculated as:

$$\text{I}^2\text{S bit rate} = 16 \times 2 \times F_s$$

If the packet length is 32 bits, there are:

$$\text{I}^2\text{S bit rate} = 32 \times 2 \times F_s$$

Figure 18-26 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:

$$\text{When } \text{MCLKOEN} = 1 \text{ and } \text{CHBITS} = 0, F_s = I^2\text{Sx CLK} / [(16 \times 2) \times ((2 \times LDIV) + ODD_EVEN) \times 8]$$

$$\text{When } \text{MCLKOEN} = 1 \text{ and } \text{CHBITS} = 1, F_s = I^2\text{Sx CLK} / [(32 \times 2) \times ((2 \times LDIV) + ODD_EVEN) \times 4]$$

$$\text{When } \text{MCLKOEN} = 0 \text{ and } \text{CHBITS} = 0, F_s = I^2\text{Sx CLK} / [(16 \times 2) \times ((2 \times LDIV) + ODD_EVEN)]$$

$$\text{When } \text{MCLKOEN} = 0 \text{ and } \text{CHBITS} = 1, F_s = I^2\text{Sx CLK} / [(32 \times 2) \times ((2 \times LDIV) + ODD_EVEN)]$$

The exact audio frequency can be obtained by referring to the clock configuration in the table below.

Table 18-2 Use the standard 8MHz HSE clock to get accurate audio frequency.

SYSCLK (MHz)	I ² S_LDIV		I ² 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
72	11	6	1	0	without	96000	97826.09	93750	1.90%	2.34%
72	23	11	1	1	without	48000	47872.34	48913.04	0.27%	1.90%
72	25	13	1	0	without	44100	44117.65	43269.23	0.04%	1.88%
72	35	17	0	1	without	32000	32142.86	32142.86	0.44%	0.44%
72	51	25	0	1	without	22050	22058.82	22058.82	0.04%	0.04%
72	70	35	1	0	without	16000	15675.75	16071.43	0.27%	0.45%
72	102	51	0	0	without	11025	11029.41	11029.41	0.04%	0.04%
72	140	70	1	1	without	8000	8007.11	7978.72	0.09%	0.27%
72	2	2	0	0	yes	96000	70312.15	70312.15	26.76%	26.76%
72	3	3	0	0	yes	48000	46875	46875	2.34%	2.34%
72	3	3	0	0	yes	44100	46875	46875	6.29%	6.29%
72	4	4	1	1	yes	32000	31250	31250	2.34%	2.34%
72	6	6	1	1	yes	22050	21634.61	21634.61	1.88%	1.88%
72	9	9	0	0	yes	16000	15625	15625	2.34%	2.34%
72	13	13	0	0	yes	11025	10817.3	10817.3	1.88%	1.88%
72	17	17	1	1	yes	8000	8035.71	8035.71	0.45%	0.45%

18.4.3 I²S Transmission and reception sequence

I²S initialization sequence

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 18.4.2).
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 I2S mode, and set SPI_I2SCFG.MODCFG[1:0] bits to select the I2S master-slave mode and transmission direction (send or receive); set SPI_I2SCFG.STDSEL[1:0] bits to select the corresponding I2S 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.

Sending sequence

Master mode

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_I2SPREDIV.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 18.4.1 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.

Slave mode

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, CHSIDE depends on the WS signal of the external master I2S device (WS signal is 1 means the left channel)

Receiving sequence

Master mode

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 18.4.1 for details.

When I2S 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.TDATLEN = 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 I²S clock cycles.
 3. Turn off I²S (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), I²S 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 I²S clock cycle.
 3. Turn off I²S (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 I²S clock cycle.
 3. Turn off I²S (SPI_I2SCFG.I2SEN=0).

Slave mode

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 I2S function is turned off, clear the SPI_I2SCFG.I2SEN bit to 0 when the SPI_STS.RNE flag is 1.

18.4.4 Status flag

There are the following 4 flag bits in the SPI_STS register for monitoring the status of the I2S bus.

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.

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.

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 I²S clock cycle. Therefore, do not use the BUSY flag(SPI_STS.BUSY) to handle the sending and receiving of each data item.

Channel Side 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.

18.4.5 Error flag

The SPI_STS register has 2 error flag bits.

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.

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 clear the SPI_STS.UNDER bit.

18.4.6 I²S interrupt

The following table lists all I²S interrupts.

Table 18-3 I²S interrupt request

Interrupt event	Event flag bit	Enable control bit
Send buffer empty flag	TE	TEINTEN
Receive buffer non empty flag	RNE	RNEINTEN
Underflow flag bit	UNDER	ERRINTEN
Overflow flag bit	OVER	

18.4.7 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.

18.5 SPI and I²S register description

18.5.1 SPI register overview

Table 18-4 SPI 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	SPI_CTRL1																	BIDIRMODE	15																	
	Reset Value																	0	0	BIDIROEN	14															
004h	SPI_CTRL2																	CRCEN	13																	
	Reset Value																	CRCNEXT	12																	
008h	SPI_STS																	DATAFF	11																	
	Reset Value																	RONLY	10																	
00Ch	SPI_DAT																	SSSMEN	9																	
	Reset Value																	SSEL	8																	
010h	SPI_CRCPOLY																	CRCPOLY[15:0]																		
	Reset Value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
014h	SPI_CRCRDAT																	CRCRDAT[15:0]																		
	Reset Value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
018h	SPI_CRCTDAT																	CRCTDAT[15:0]																		
	Reset Value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
01Ch	SPI_I2SCFG																	MODSEL	1	12SEN	1	MODCFG[1:0]	1	PCMFSYNC	1	0	0	0	0	0	0	0	0	0	0	
	Reset Value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
020h	SPI_I2SPREDIV																	MCLKOEN	1	ODD_EVEN	1	STDSEL[1:0]	1	CLKPOL	1	TDATLEN[1:0]	1	LDIV[7:0]	1	CHBITS	1	CHSIDE	1	MSEL	2	
	Reset Value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

18.5.2 SPI control register 1 (SPI_CTRL1) (not used in I2S mode)

Address: 0x000

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
BIDIR MODE	BIDIR OEN	CRCEN	CRC NEXT	DATFF	RONLY	SSSMEN	SSEL	LSBFF	SPIEN	BR[2:0]	MSEL	CLKPOL	CLKPHA			

Bit field	name	describe
15	BIDIRMODE	Bidirectional data mode enable 0: Select the "two-wire one-way" mode. 1: Select the "one-wire bidirectional" mode.

Bit field	name	describe
		<p><i>Note: Not used in I²S mode.</i></p>
14	BIDIROEN	<p>Output enable in bidirectional mode 0: Output disable (receive-only mode). 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 I²S mode.</i></p>
13	CRCEN	<p>Hardware CRC check enable 0: Disable CRC calculation. 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 I²S mode.</i></p>
12	CRCNEXT	<p>Send CRC next 0: The next sent value comes from the send buffer. 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 I²S mode.</i></p>
11	DATFF	<p>Data frame format 0: 8-bit data frame format is used for sending/receiving. 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 I²S 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). 1: Disable output (receive-only mode).</p> <p><i>Note: Not used in I²S mode.</i></p>
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. 1: Enable software slave device management.</p> <p><i>Note: Not used in I²S mode.</i></p>

Bit field	name	describe
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 I²S 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 I²S 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 I²S mode.</i></p> <p><i>Note: When turning off the SPI device, please follow paragraph 18.3.4 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 I²S 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 I²S 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> <p><i>Note: Not used in I²S mode.</i></p>
0	CLKPHA	<p>Clock phase</p> <p>0: Data is sampled on the first clock edge.</p> <p>1: Data is sampled on the second clock edge.</p> <p><i>Note: This bit cannot be changed during communication.</i></p> <p><i>Note: Not used in I²S mode.</i></p>

18.5.3 SPI control register 2 (SPI_CTRL2)

Address: 0x04

Reset value: 0x0000

15	Reserved				8	7	6	5	4	3	2	1	0
						TE INTEN	RNE INTEN	ERR INTEN		Reserved	SSOEN	TDMAEN	RDMAEN

rw rw rw rw rw rw rw

Bit field	name	describe
15:8	Reserved	Reserved, the reset value must be maintained.
7	TEINTEN	<p>Send buffer empty interrupt enable</p> <p>0: Disable TE interrupt.</p> <p>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</p> <p>0: Disable RNE interrupt.</p> <p>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</p> <p>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</p> <p>0: Disable error interrupt.</p> <p>1: Enable error interrupt.</p>
4:3	Reserved	Reserved, the reset value must be maintained.
2	SSOEN	<p>NSS output enable</p> <p>0: Disable NSS output in master mode, the device can work in multi-master mode.</p> <p>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 I²S mode.</i></p>
1	TDMAEN	<p>Send buffer DMA enable</p> <p>When this bit is set, a DMA request is issued as soon as the TE flag(SPI_STS.TE) is set</p> <p>0: Disable send buffer DMA.</p> <p>1: Enable send buffer DMA.</p>
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>

18.5.4 SPI status register (SPI_STS)

Address: 0x08

Reset value: 0x0002

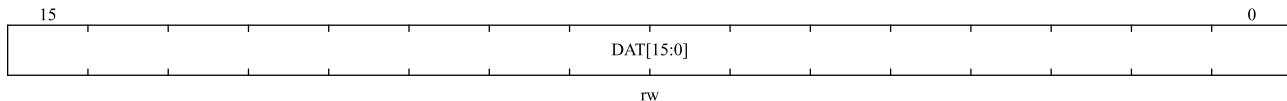
15	Reserved	8	7	6	5	4	3	2	1	0	
				BUSY	OVER	MODERR	CRCERR	UNDER	CHSIDE	TE	RNE
				r	r	r	rc_w0	r	r	r	r

Bit field	name	describe
15:8	Reserved	Reserved, the reset value must be maintained.
7	BUSY	<p>Busy flag 0: SPI is not busy. 1: SPI is busy communicating or the send buffer is not empty. This bit is set or reset by hardware.</p> <p><i>Note: special attention should be paid to the use of this sign, see Section 18.3.3 and Section 18.3.4 for details..</i></p>
6	OVER	<p>Overflow flag 0: No overflow error. 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 18.3.7 for details.</i></p>
5	MODERR	<p>Mode error 0: No mode error. 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 18.3.7 for details.</i></p> <p><i>Note: Not used in I²S mode.</i></p>
4	CRCERR	<p>CRC error flag 0: The received CRC value matches the value the SPI_CRCRDAT register value. 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 I²S mode.</i></p>
3	UNDER	<p>Underflow flag 0: No underflow occurred. 1: Underflow 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 18.4.5 for details.</i></p> <p><i>Note: not used in SPI mode.</i></p>
2	CHSIDE	<p>Channel 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 0: The send buffer is not empty. 1: The send buffer is empty.</p>
0	RNE	<p>Receive buffer is not empty 0: The receive buffer is empty. 1: The receive buffer is not empty.</p>

18.5.5 SPI data register (SPI_DAT)

Address: 0x0C

Reset value: 0x0000

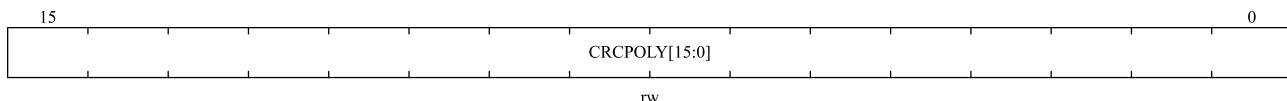


Bit field	name	describe
15:0	DAT[15:0]	<p>Data register</p> <p>Data to be sent or received</p> <p>The data register corresponds to two buffers: one for write (send buffer); The other is for read (receive buffer). Write operation writes data to send buffer; The read operation will return the data in the 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>

18.5.6 SPI CRC polynomial register (SPI_CRCPOLY) (not used in I²S mode)

Address: 0x10

Reset value: 0x0007

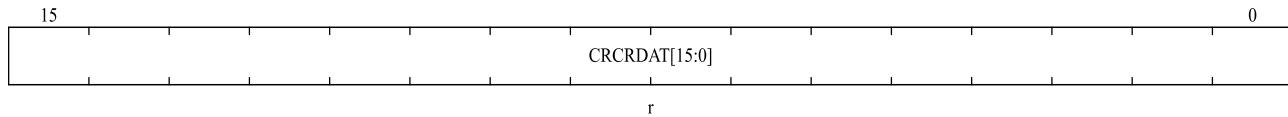


Bit field	name	describe
15:0	CRCPOLY [15:0]	<p>CRC polynomial register</p> <p>This register contains the polynomial used for the CRC calculation.</p> <p>The reset value is 0x0007, other values can be set according to the application.</p> <p><i>Note: not used in I²S mode.</i></p>

18.5.7 SPI RX CRC register (SPI_CRCRDAT) (not used in I²S mode)

Address offset: 0x14

Reset value: 0x0000



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 I²S mode.</i></p>

18.5.8 SPI TX CRC register (SPI_CRCTDAT)

Address offset: 0x18

Reset value: 0x0000

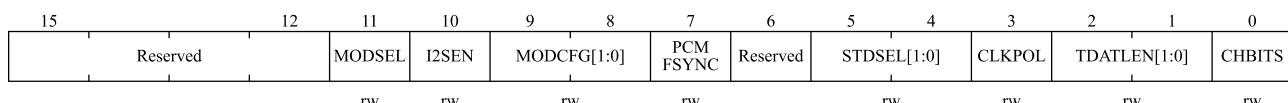


Bit field	Name	Description
15:0	CRCTDAT	<p>Send CRC register</p> <p>When CRC calculation is enabled, CRCTDAT[15:0] contains the CRC value calculated by the bytes sent subsequently. This register is reset when '1' is written to the SPI_CTRL1.CRCEN bit. The CRC calculation uses the polynomial in SPI_CRCPOLY.</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 I²S mode.</i></p>

18.5.9 SPI_I²S configuration register (SPI_I2SCFG)

Address offset: 0x1c

Reset value: 0x0000



Bit field	Name	Description
15:12	Reserved	Reserved, the reset value must be maintained.
11	MODSEL	<p>I²S mode selection</p> <p>0: Select SPI mode.</p> <p>1: Select I²S mode.</p> <p><i>Note: this bit can only be set when SPI or I²S is turned off.</i></p>
10	I ² SEN	<p>I²S enable</p> <p>0: Disable I²S.</p> <p>1: Enable I²S.</p> <p><i>Note: not used in SPI mode.</i></p>
9:8	MODCFG	<p>I²S 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 I²S 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	<p>Selection of I²S standard</p> <p>00: I²S Philips standard.</p> <p>01: High byte alignment standard (left alignment).</p> <p>10: Low byte alignment standard (right alignment).</p> <p>11: PCM standard.</p> <p>See for details of I²S standard on section 18.4.1.</p> <p><i>Note: For correct operation, this bit can only be set when I²S is turned off.</i></p> <p><i>Not used in SPI mode.</i></p>
3	CLKPOL	<p>Static clock polarity</p> <p>0: I2S clock static state is low level.</p> <p>1: I2S clock static state is high level.</p> <p><i>Note: For correct operation, this bit can only be set when I²S 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</p> <p>00: 16-bit data length.</p> <p>01: 24-bit data length;</p> <p>10: 32-bit data length;</p> <p>11: Not allowed.</p> <p><i>Note: For correct operation, this bit can only be set when I²S is turned off.</i></p> <p><i>Note: not used in SPI mode.</i></p>

Bit field	Name	Description
0	CHBITS	<p>Channel length (number of data bits per audio channel)</p> <p>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 I²S is turned off.</i></p> <p><i>Note: not used in SPI mode.</i></p>

18.5.10 SPI_I²S prescaler register (SPI_I2SPREDIV)

Address: 0x20

Reset value: 0x0002

15	Reserved	10	MCLK OEN	9	ODD EVEN	8	7	LDIV[7:0]	0
			rw		rw			rw	

Bit field	Name	Description
15:10	Reserved	Reserved, the reset value must be maintained.
9	MCLKOEN	<p>Master clock output enable</p> <p>0: Disable master clock output. 1: Enable master clock output.</p> <p><i>Note: For correct operation, this bit can only be set when I²S is turned off.</i></p> <p><i>Note: not used in SPI mode.</i></p>
8	ODD _EVEN	<p>Coefficient prescaler</p> <p>0: actual frequency division coefficient = LDIV $\times 2$. 1: actual frequency division coefficient = (LDIV $\times 2$)+1.</p> <p>See Section 18.4.2 for details.</p> <p><i>Note: For correct operation, this bit can only be set when I²S is turned off. Use this bit only in I²S master mode.</i></p> <p><i>Not used in SPI mode.</i></p>
7:0	LDIV	<p>I²S linear prescaler</p> <p>Setting LDIV [7:0] = 0 or LDIV [7:0] = 1 is prohibited.</p> <p>See Section 18.4.2 for details.</p> <p><i>Note: For correct operation, this bit can only be set when I²S is turned off. Use this bit only in I²S master mode.</i></p> <p><i>Not used in SPI mode.</i></p>

19 Real-time clock (RTC)

19.1 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 (day of week), and date (day of month).
- 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.
- One tamper detection events with configurable filter and internal pull-up.
- Time-Stamp function.
- 20 backup registers which can keep data under low power mode.
- 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 any mode (include RUN mode, SLEEP mode, STOP0 mode, STOP2 mode and STANDBY mode).
- RTC provides a variety of ways to wakeup from all low-power modes (RUN mode, SLEEP mode, STOP0 mode, STOP2 mode and STANDBY mode).

19.1.1 Specification

Table 19-1 RTC feature support

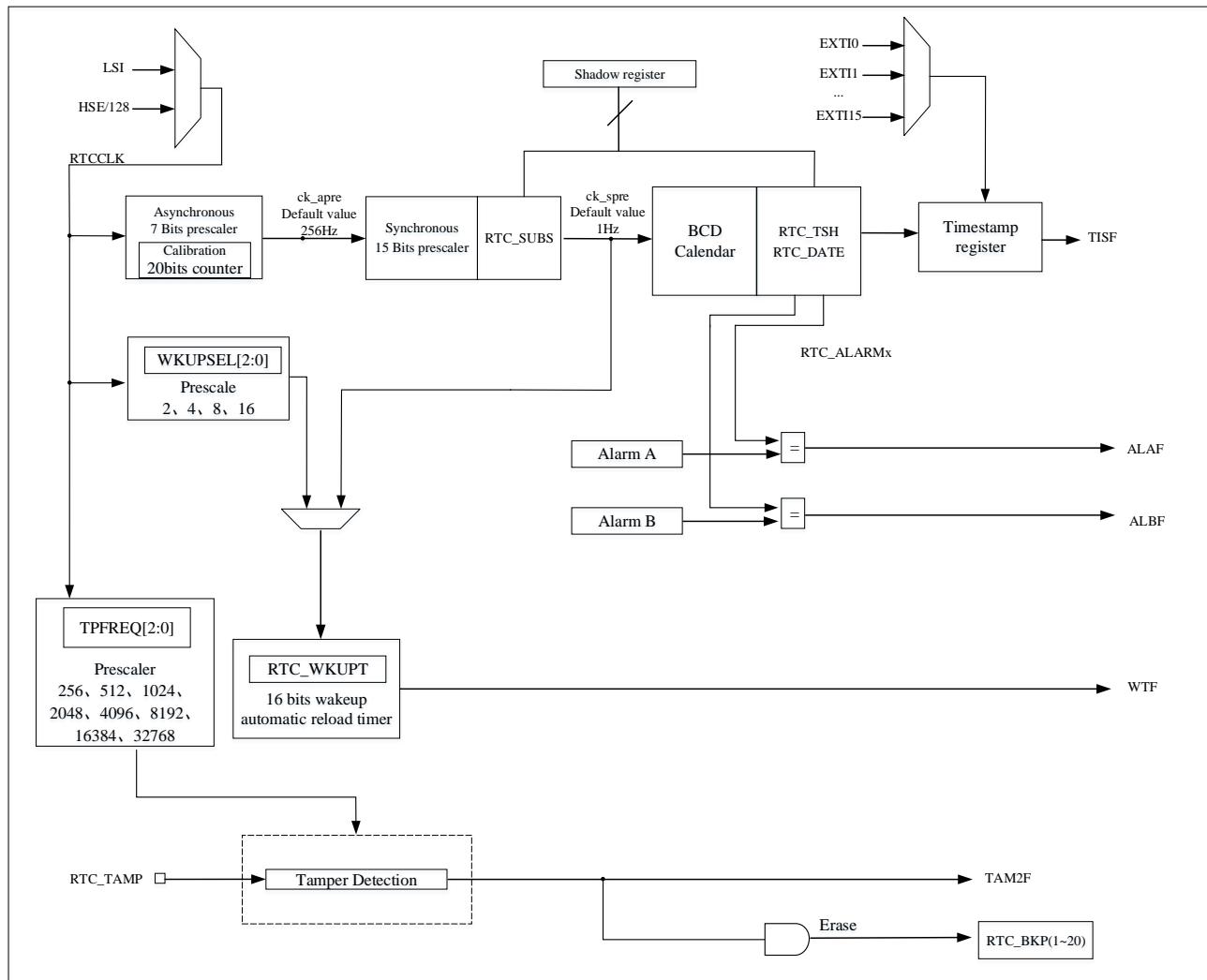
Main function	Description
Clock	RTC clock can be selected from LSI and HSE, which are 40KHz, 32.768KHz and HSE / 128 respectively
Reset	<p>The APB interface is reset by the system. Some register reset from RTC module is synchronized with APB reset.</p> <p>Following Registers to be cleared by System Resets only</p> <ul style="list-style-type: none"> ● RTC_SUBS ● RTC_TSH ● RTC_DATA ● RTC_INITSTS(few bits) <p>RTC core area reset is reset by backup area.</p> <p>Used to Reset the RTC logic as well as few registers the content to be retained during Low-Power modes. These include</p> <ul style="list-style-type: none"> ● RTC_CTRL ● RTC_PRE ● RTC_CALIB ● RTC_SCTRL ● RTC_TSSS, RTC_TST and RTC_TSD ● RTC_TMPCFG ● RTC_WKUPT ● RTC_ALRMASS/RTC_ALRMA ● RTC_ALRMBSS/RTC_ALRMB ● RTC_OPT ● RTC_BKP(1~20)
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	It can be configured as an interrupt/event to wake up the system from SLEEP, STOP0 ,STOP2 and STANDBY modes.
Alarm	Programmable alarm clock and interrupt function. The alarm can be triggered by any combination of the calendar fields, and it also can be used to wake up the CPU or exit from the low power status such as SLEEP, STOP0, STOP2 and STANDBY modes.
Tamper	1 Tamper detection logic are a source of system Wakeup should a Tamper event happen on the input lines. The Tamper event also causes an erase of Back up registers when enabled. 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/event

Main function	Description
	Wakeup interrupt/event Timestamp interrupt/event Tamper interrupt/event
Backup registers	20 backup registers

19.2 RTC function description

19.2.1 RTC block diagram

Figure 19-1 RTC Block Diagram



RTC includes the following functions:

- Alarm A and Alarm B event/interrupt
- Timestamp event/interrupt
- Tamper event/interrupt

- 20 32-bit backup registers
- RTC input functions:
 - ◆ Timestamp event detection
 - ◆ 50 or 60Hz reference clock input
 - ◆ Tamper event detection

19.2.2 GPIOs of RTC

Timestamp input come from EXTI module, please refer to the timestamp trigger source selection register (EXTI_TS_SEL) for details.

PA0 can be used as RTC tamper detection pin.

PA10 can be used as RTC_REFCLKIN reference clock input pin.

19.2.3 RTC register write protection

PWR_CTRL.DBKP bit (see the Power Control section) is cleared in default, so PWR_CTRL.DBKP bit must set to “1” to enable write access to the RTC register. Once the backup domain is reset, all write protection RTC registers are write protected. 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.

19.2.4 RTC clock and prescaler

RTC clock source:

- 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:

$$f_{ck_apre} = \frac{f_{RTCCLK}}{RTC_PRE.DIVA[6:0]+1}$$

$$f_{\text{ck_spre}} = \frac{f_{\text{RTCCLK}}}{(RTC_PRE.DIVS[14:0]+1)*(RTC_PRE.DIVA[6:0]+1)}$$

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].

19.2.5 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 & standby) 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.

19.2.6 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: Before entering the RTC initialization mode, ensure that the value of RTC_SUBS.SS[15:0] is not less than 2.

19.2.7 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 (STANDBY 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;

19.2.8 Programmable alarms

RTC has 2 programmable alarms: Alarm A and Alarm B.

RTC alarm can be enabled or disabled by RTC_CTRL.ALxEN bit. If the alarm value match the calendar values, the RTC_INITSTS.ALxF flag will be set 1. Each calendar field can be selected to trigger alarm interrupt if RTC_CTRL.ALxIEN bit is enabled.

Note: If the seconds field is selected (RTC_ALARMx.MASK1 bit reset), RTC_PRE.DIVS[14:0] must be larger than 3 to ensure correct operation.

19.2.9 Alarm configuration

Alarm A and Alarm B should be configured in the following below:

- Disable Alarm A/Alarm B by clearing RTC_CTRL.ALAE/RTC_CTRL.ALBE 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.ALAEEN/ RTC_CTRL.ALBEN bit.

19.2.10 Periodic automatic wakeup

A 16-bit programmable auto-load down counter can generate periodic wakeup flag if reaches 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 LSI (40KHz), wake-up interrupt period can be configured range from 100us to 26s under the resolution down to 50us.

- 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.

19.2.11 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

19.2.12 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.

19.2.13 Tamper Detection

There are three tamper detection pin, RTC_TAMP pin is PA0. RTC_TAMP 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.

When RTC_TAMP event is detected, RTC_BKP(1~20) registers will be erased if RTC_TMPCFG.TPNOE=0.

Tamper detection initialization

There are three tamper detection pins, each of them can be configured independently. User need to configure tamper detection before enable RTC_TMPCFG.TPEN bit. When the tamper event is detected after tamper detection is enable, if RTC_TMPCFG.TPINEN is set to 1, tamper event can generate an interrupt and RTC_INITSTS.TAMF bit will be set 1.

When RTC_INITSTS.TAMF bit 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_TMPCFG.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.TPTRG bit. The RTC_TAMP pin will generate a tamper detection event when corresponding edge is detected.

Because of RTC_BKP(1~20) can be reset when tamper event is detected, it is necessary to ensure that tamper event detection and writing to RTC_BKP(1~20) will not occur at the same time. It is recommended to start the tamper detection function after writing RTC_BKP(1~20).

Filtered level detection of RTC_TAMP 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.

19.2.14 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 remember this adjustment or not.

19.2.15 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.

19.2.16 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 the 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.

19.2.17 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							
STOP0		Normal work when the clock source of RTC is LSI												Alarm A, Alarm B, Periodic Wakeup, Tamper event and Timestamp event							
STOP2		Normal work when the clock source of RTC is LSI												Alarm A, Alarm B, Periodic Wakeup, Tamper event and Timestamp event							
STANDBY		Normal work when the clock source of RTC is LSI												Alarm A, Alarm B, Periodic Wakeup, Tamper event and Timestamp event							

19.3 RTC Registers

19.3.1 RTC Register overview

Table 19-2 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												HOU[3:0]		MIT[2:0]		MIU[3:0]		SCT[2:0]		SCU[3:0]		DAT[1:0]		DAU[3:0]		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												
004h	RTC_DATE	Reserved				YRT[3:0]				YRU[3:0]				WDU[2:0]		MOT		MOU[3:0]		0		0		0		0		0														
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												
008h	RTC_CTRL	Reserved												BAKP		SUIH		ADIH		TSIEN		WTEN		ALBIEN		ALAIEN		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												
00Ch	RTC_INITSTS	Reserved												RECPF		TAMF		TISOVF		0		TISF		0		TSEN		WTEN		ALBEN		ALAEN		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												
010h	RTC_PRE	Reserved				DIVA[6:0]								DIVS[14:0]															0													
	Reset Value	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												
014h	RTC_WKUPT	Reserved												WKUPT[15:0]															0													
	Reset Value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1												
01Ch	RTC_ALARMA	MASK4	0	WKDSEL	0	DTT[1:0]	DTU[3:0]		MASK3	0	APM	0	HOT[1:0]	HOU[3:0]			MASK2	0	MIT[2:0]		MIU[3:0]		MASK1	0	SET[2:0]		SEU[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	0	0	0	0											
020h	RTC_ALARMIB	MASK4	0	WKDSEL	0	DTT[1:0]	DTU[3:0]		MASK3	0	APM	0	HOT[1:0]	HOU[3:0]			MASK2	0	MIT[2:0]		MIU[3:0]		MASK1	0	SET[2:0]		SEU[3:0]		0		0		0		0		0		0		0	
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0											
024h	RTC_WRP	Reserved												PKEY[7:0]															0		0		0		0		0		0			
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0											
028h	RTC_SUBS	Reserved												SS[15:0]															0		0		0		0		0		0			
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0											

19.3.2 RTC Calendar Time Register (RTC_TSH)

Address offset: 0x00

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	rw	6	rw	4	3			rw	0
Reserved	MIT[2:0]			MIU[3:0]			Reserved	SCT[2:0]			SCU[3:0]			
	rw			rw				rw			rw			

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

Bit field	Name	Description
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

19.3.3 RTC Calendar Date Register (RTC_DATE)

Address offset: 0x04

Reset value: 0x0000 2101

31	Reserved								24	YRT[3:0]				20	YRU[3:0]				19					16
15	WDU[2:0]	MOT	13	12	11	MOU[3:0]	8	7	6	r	5	4	3	r	DAU[3:0]	0	r	r	r	r	r	r	r	r
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	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
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

19.3.4 RTC Control Register (RTC_CTRL)

Address offset: 0x08

Reset value: 0x0000 0000

31	Reserved															19	18	17	16
15	TSIEN	WTIEN	ALBIEN	ALAIEN	TSEN	WTEN	ALBEN	ALAEN	Reserved	HFMT	BYPS	REF CLKEN	TEDGE	WKUPSEL[2:0]	2	rw	w	w	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bit field	Name	Description
31:19	Reserved	Reserved, the reset value must be maintained
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 1: Enable Alarm A interrupt
11	TSEN	Timestamp enable 0: Disable timestamp 1: Enable timestamp
10	WTEN	Wakeup timer enable 0: Disable wakeup timer 1: Enable wakeup timer
9	ALBEN	Alarm B enable 0: Disable Alarm B 1: Enable Alarm B
8	ALAEN	Alarm A enable 0: Disable Alarm A 1: Enable Alarm A
7	Reserved	Reserved, the reset value must be maintained
6	HFMT	Hour format bit 0: 24 hour format 1: Am/PM format

Bit field	Name	Description
5	BYPS	<p>Bypass values from the shadow registers</p> <p>0: Calendar values are copied from the shadow registers, which are refreshed every two RTCCLK cycles.</p> <p>1: Calendar values are copied directly from the calendar counters.</p> <p><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)</p> <p>0: Disable RTC_REFIN detection</p> <p>1: Enable RTC_REFIN detection</p> <p><i>Note: RTC_PRE.DIVS must be 0x00FF</i></p>
3	TEDGE	<p>Time-stamp event active edge</p> <p>0: Input rising edge creates a timestamp event</p> <p>1: Input falling edge creates a timestamp event</p> <p><i>RTC_CTRL.TSEN need to be reset when TEDGE is changed to avoid unwanted RTC_INITSTS.TISF setting.</i></p>
2:0	WKUPSEL[2:0]	<p>Wakeup clock selection</p> <p>000: RTC clock is divided by 16</p> <p>001: RTC clock is divided by 8</p> <p>010: RTC clock is divided by 4</p> <p>011: RTC clock is divided by 2</p> <p>10x: ck_spre (usually 1Hz) clock is selected</p> <p>11x: ck_spre (usually 1Hz) clock is selected and 2^{16} is added to the RTC_WKUPT.WKUPT counter.</p>

19.3.5 RTC Initial Status Register (RTC_INITSTS)

Address offset: 0x0C

Reset value: 0x0000 0007

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

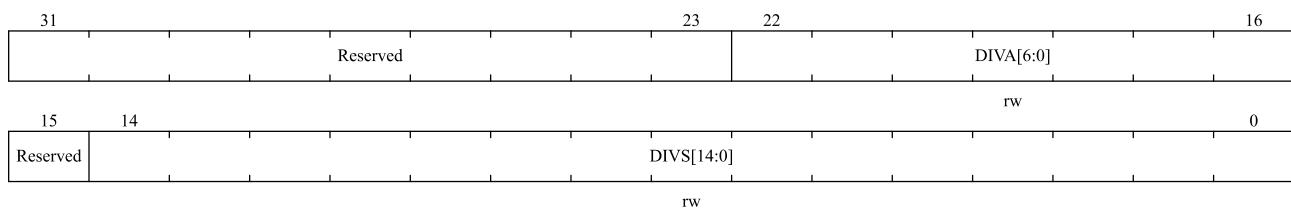
Bit field	Name	Description
14	TAMF	<p>RTC_TAMP2 detection flag</p> <p>This flag is set to '1' by hardware when a tamper event is detected on the RTC_TAMP2 input pin.</p> <p>This flag can be cleared by software writing 0</p>
13	Reserved	Reserved, the reset value must be maintained
12	TISOVF	<p>The time-stamp 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	<p>Alarm B flag</p> <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>

Bit field	Name	Description
		0: Calendar shadow register not yet synchronized 1: Calendar shadow register synchronized
4	INITSF	Initialization status flag This flag is set to '1' by hardware when the calendar year field is different from 0 (which is the RTC domain reset state). 0: Calendar has not been initialized 1: Calendar has been initialized
3	SHOPF	Shift operation pending flag 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. 0: No shift operation is pending 1: A shift operation is pending
2	WTWF	Wakeup timer write flag 0: Wakeup timer configuration update is not allowed 1: Wakeup timer configuration update is allowed
1	ALBWF	Alarm B write flag 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. 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.ALAREN bit has been set to 0. 0: Alarm A update is not allowed 1: Alarm A update is allowed

19.3.6 RTC Prescaler Register (RTC_PRE)

Address offset: 0x10

Reset value: 0x007F 00FF



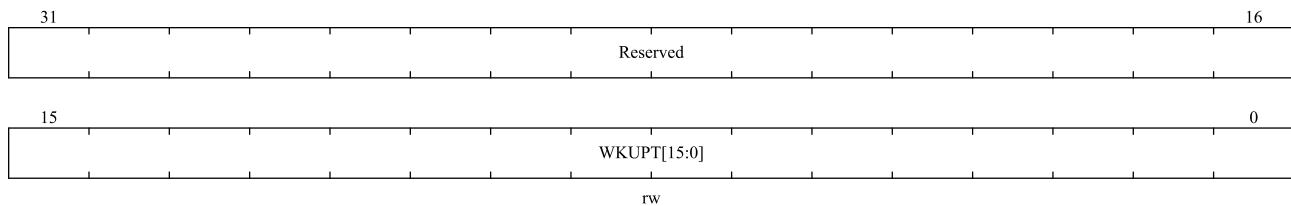
Bit field	Name	Description
31:23	Reserved	Reserved, the reset value must be maintained
22:16	DIVA[6:0]	Asynchronous prescaler factor $f_{ck_apre} = \text{RTCCLK}/(\text{DIVA}[6:0]+1)$

Bit field	Name	Description
15	Reserved	Reserved, the reset value must be maintained
14:0	DIVS[14:0]	Synchronous prescaler factor $f_{ck_spre} = f_{ck_apre}/(\text{DIVS}[14:0]+1)$

19.3.7 RTC Wakeup Timer Register (RTC_WKUPT)

Address offset: 0x14

Reset value: 0x0000 FFFF

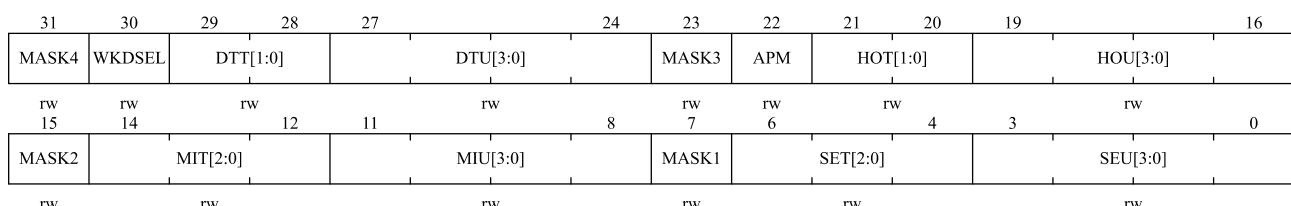


Bit field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained
15:0	WKUPT[15:0]	<p>Wake up auto-reload value bits</p> <p>The RTC_INITSTS.WTF flag is set every (WKUPT[15:0] + 1) ck_wut cycles when the RTC_CTRL.WTEN=1. The wakeup timer becomes 17-bits When RTC_CTRL.WKUPSEL[2]=1. These bits can not be set to 0.</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>

19.3.8 RTC Alarm A Register (RTC_ALARMA)

Address offset: 0x1C

Reset value: 0x0000 0000



Bit field	Name	Description
31	MASK4	Alarm date mask 0: Date/day match

Bit field	Name	Description
		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
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

19.3.9 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]				

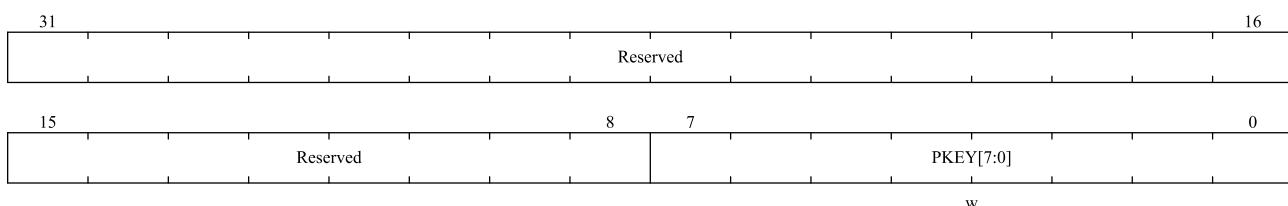
Bit field	Name	Description
31	MASK4	Alarm date mask 0: Date/day match 1: Date/day not match
30	WKDSEL	Week day selection 0: DTU[3:0] represents the date units

Bit field	Name	Description
		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
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

19.3.10 RTC Write Protection register (RTC_WRP)

Address offset: 0x24

Reset value: 0x0000 0000

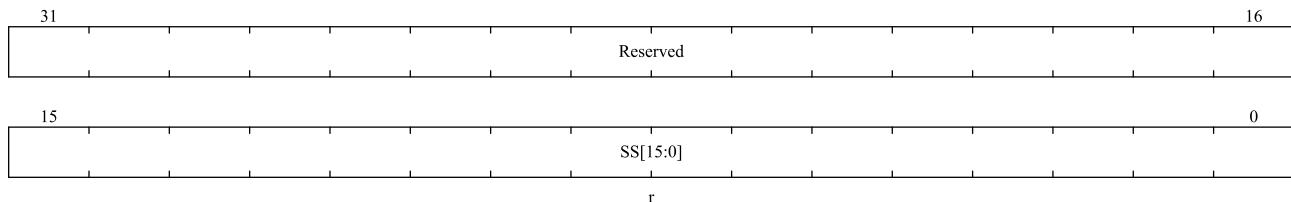


Bit field	Name	Description
31:8	Reserved	Reserved, the reset value must be maintained
7:0	PKEY[7:0]	Write protection key Reading this byte always returns 0x00. For detail on how to unlock RTC register write protection, see chapter RTC write protection.

19.3.11 RTC Sub-second Register (RTC_SUBS)

Address offset: 0x28

Reset value: 0x0000 0000

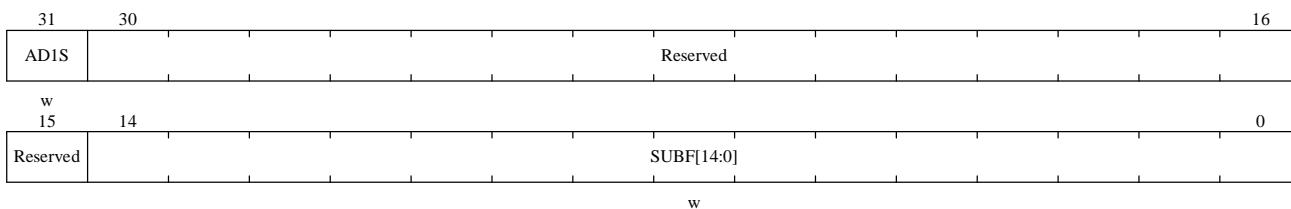


Bit field	Name	Description
31:16	Reserved	Reserved, the reset value must be maintained
15:0	SS[15:0]	<p>Sub-second value.</p> <p>The value is the counter value of synchronous prescaler. This sub-second value is calculated by the below formula:</p> $\text{Sub-second value} = (\text{RTC_PRE.DIVS}[14:0]-\text{SS})/(\text{RTC_PRE.DIVS}[14:0]+1)$ <p><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></p>

19.3.12 RTC Shift Control Register (RTC_SCTRL)

Address offset: 0x2C

Reset value: 0x0000 0000



Bit field	Name	Description
31	AD1S	<p>Add one second</p> <p>0: No add one second.</p> <p>1: Add one second to the clock/calendar</p> <p>This bit can only be written and read as zero. Writing to this bit does not have an impact when RTC_INITSTS.SHOPF=1.</p>
30:15	Reserved	Reserved, the reset value must be maintained
14:0	SUBF[14:0]	<p>Subtract a fraction of a second</p> <p>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[14:0] is added to the synchronous prescaler counter, and the clock will delay:</p>

Bit field	Name	Description
		<p>Delay (seconds) = $(\text{SUBF}[14:0]+1) / (\text{DIVS}[14:0] + 1)$</p> <p>AD1S bit can be used together with the SUBF[14:0]bits:</p> <p>Advance (seconds) = $(1 - ((\text{SUBF}[14:0]+1) / (\text{DIVS}[14:0] + 1)))$.</p> <p><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.</i></p>

19.3.13 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]			
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, 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	SET[2:0]	Describes the second tens value in BCD format
3:0	SEU[3:0]	Describes the second units value in BCD format

19.3.14 RTC Timestamp Date Register (RTC_TSD)

Address offset: 0x34

Reset value: 0x0000 0000

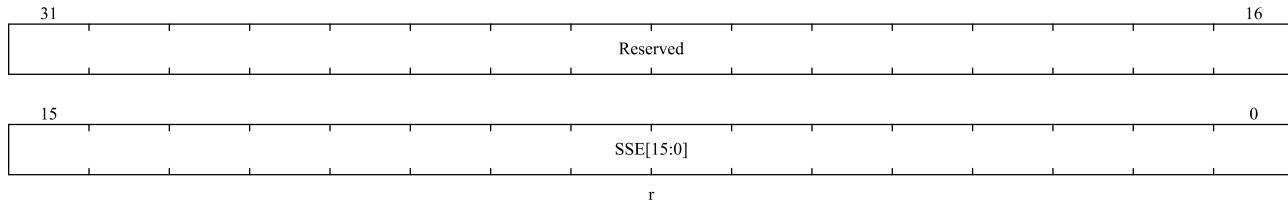
31	Reserved								24	23	YRT[3:0]			20	19	16
										r						
15	13	12	11	8	7	r	6	r	5	4	3	r				0
WDU[2:0]	MOT		MOU[3:0]	Reserved	DAT[1:0]				DAU[3:0]							r
	r		r		r			r		r						r

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
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

19.3.15 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]	Sub second value SSE[15:0] is the value in the synchronous prescaler counter. The fraction of a second is provided by the formula below: Second fraction = (RTC_PRE.DIVS[14:0] – SSE[15:0]) / (RTC_PRE.DIVS[14:0] + 1) <i>Note: SSE[15:0] can be larger than RTC_PRE.DIVS[14:0] only after a shift operation. In that case, the correct time/date is one second less than as indicated by RTC_TSH/RTC_DATE.</i>

19.3.16 RTC Calibration Register (RTC_CALIB)

Address offset: 0x3C

Reset value: 0x0000 0000

31														16
Reserved														
15	14	13	12			9	8							0
CP	CW8	CW16				Reserved								rw

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 $((512 * CP) - CM[8:0])$.</p> <p>0: No add pulse.</p> <p>1: One RTCCLK pulse is inserted every 2^{11} pulses.</p>
14	CW8	<p>Select an 8-second calibration cycle period</p> <p>0: Not effect.</p> <p>1: Select an 8-second calibration period.</p> <p>When CW8 is set to '1', the 8-second calibration cycle period is selected.</p> <p><i>Note: when CW8 = 1, CM[1:0] will always be '00'</i></p>
13	CW16	<p>To select a 16-second calibration cycle period</p> <p>0: Not effect.</p> <p>1: Select a calibration period of 16 seconds. If CW8 = 1, this bit cannot be set to 1.</p> <p><i>Note: when CW16 = 1, CM[0] will always be '0'</i></p>
12:9	Reserved	Reserved, the reset value must be maintained
8:0	CM[8:0]	<p>Negative calibration bits</p> <p>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.</p>

19.3.17 RTC Tamper Configuration Register (RTC_TMPCFG)

Address offset: 0x40

Reset value: 0x0000 0000

31														16
Reserved														
15	14	13	12	11	10	8	7	6	rw	rw	rw	rw	rw	1
TPUDIS	TPPRCH[1:0]	TPFLT[1:0]			TPFREQ[2:0]	TPTS		Reserved	TPTRG	TPEN	TPINTEN			0

Bit field	Name	Description
31:22	Reserved	Reserved, the reset value must be maintained
21	TPMF	<p>Tamper mask flag</p> <p>0: Not mask tamper event.</p>

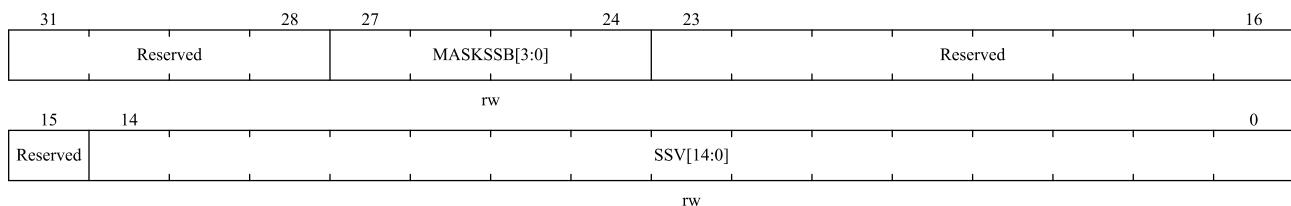
Bit field	Name	Description
		<p>1: Mask tamper event.</p> <p><i>Note: The Tamper interrupt must not be enabled when TPMF is set.</i></p>
20	TPNOE	<p>Tamper no erase</p> <p>0: Backup registers values are erased by Tamper event.</p> <p>1: Backup registers values are not erased by Tamper event.</p>
19	TP2INTEN	<p>Tamper interrupt enable</p> <p>0: Disable tamper interrupt when TP2INTEN = 0.</p> <p>1: Enabled tamper interrupt</p>
18:16	Reserved	Reserved, the reset value must be maintained
15	TPPUDIS	<p>RTC_TAMPx Pull-up disable bit.</p> <p>0: Enable precharge RTC_TAMP pins before each sampling.</p> <p>1: Disable precharge RTC_TAMP pins</p>
14:13	TPPRCH[1:0]	<p>RTC_TAMP Precharge duration.</p> <p>These bits determine the the precharge time before each sampling.</p> <p>0x0: 1 RTCCLK cycles</p> <p>0x1: 2 RTCCLK cycles</p> <p>0x2: 4 RTCCLK cycles</p> <p>0x3: 8 RTCCLK cycles</p>
12:11	TPFLT[1:0]	<p>RTC_TAMP filter count</p> <p>These bits determine the number of consecutive samples when occur active level.</p> <p>0x0: Triggers a tamper event at the active level.</p> <p>0x1: Triggers a tamper event after 2 consecutive samples at the active level.</p> <p>0x2: Triggers a tamper event after 4 consecutive samples at the active level.</p> <p>0x3: Triggers a tamper event after 8 consecutive samples at the active level.</p>
10:8	TPFREQ[2:0]	<p>Tamper sampling frequency</p> <p>This bit determines the frequency at the each RTC_TAMP input is sampled.</p> <p>0x0: Sampling once every 32768 RTCCLK.</p> <p>0x1: Sampling once every 16384 RTCCLK.</p> <p>0x2: Sampling once every 8192 RTCCLK.</p> <p>0x3: Sampling once every 4096 RTCCLK.</p> <p>0x4: Sampling once every 2048 RTCCLK.</p> <p>0x5: Sampling once every 1024 RTCCLK.</p> <p>0x6: Sampling once every 512 RTCCLK.</p> <p>0x7: Sampling once every 256 RTCCLK.</p>
7	TPTS	<p>Tamper event trigger timestamp</p> <p>0: Disable tamper event trigger timestamp</p> <p>1: Enable tamper event trigger timestamp</p> <p>TPTS is valid even if RTC_CTRL.TSEN=0.</p>
6:5	Reserved	Reserved, the reset value must be maintained
4	TPTRG	<p>Tamper event trigger edge</p> <p>if TPFLT[1:0] != 00, tamper detection is in level mode:</p> <p>0: low level trigger a tamper detection event.</p>

Bit field	Name	Description
		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
3	TPEN	Tamper 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 TPINTEN level. If this bit is cleared, each tamper event interrupt can be individually enabled by setting TPINTEN.</i>
1:0	Reserved	Reserved, the reset value must be maintained

19.3.18 RTC Alarm A sub-second register (RTC_ALRMASS)

Address offset: 0x44

Reset value: 0x0000 0000



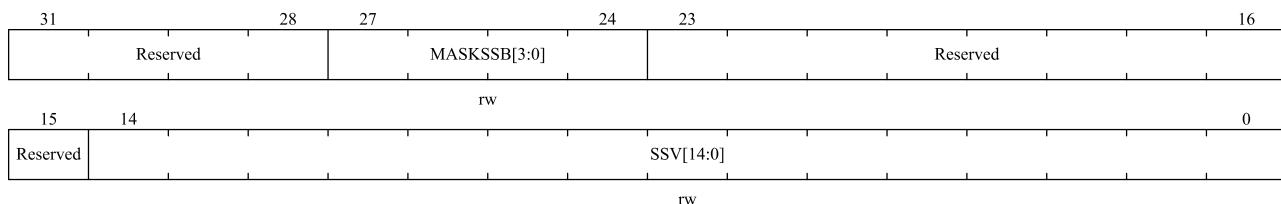
Bit field	Name	Description
31:28	Reserved	Reserved, the reset value must be maintained
27:24	MASKSSB[3:0]	Mask the most significant bit from this bits. 0x0: No comparison on sub seconds for Alarm. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match). 0x1: 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. ... 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

Bit field	Name	Description
		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].

19.3.19 RTC Alarm B sub-second register (RTC_ALRMBSS)

Address offset: 0x48

Reset value: 0x0000 0000

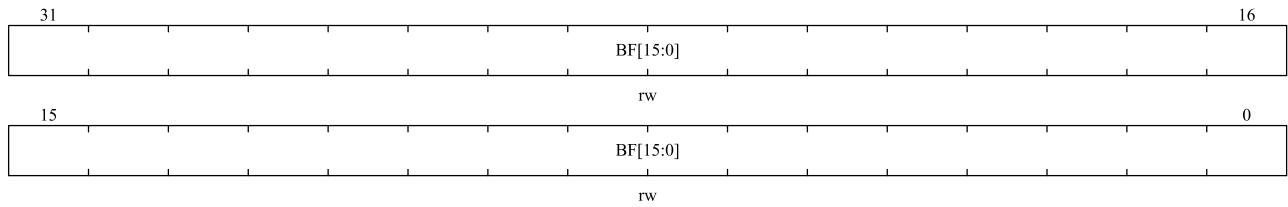


Bit field	Name	Description
31:28	Reserved	Reserved, the reset value must be maintained
27:24	MASKSSB[3:0]	<p>Mask the most significant bit from this bits.</p> <p>0x0: No comparison on sub seconds for Alarm. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).</p> <p>0x1: Only SSV[0] is compared and other bits are not compared.</p> <p>0x2: Only SSV[1:0] are compared and other bits are not compared.</p> <p>0x3: Only SSV[2:0] are compared and other bits are not compared.</p> <p>...</p> <p>0xC: Only SSV[11:0] are compared and other bits are not compared.</p> <p>0xD: Only SSV[12:0] are compared and other bits are not compared.</p> <p>0xE: Only SSV[13:0] are compared and other bits are not compared.</p> <p>0xF: SSV[14:0] are compared</p> <p>Synchronization counter RTC_SUBS.SS[15] bit is never compared.</p>
23:15	Reserved	Reserved, the reset value must be maintained
14:0	SSV[14:0]	<p>Sub seconds value</p> <p>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].</p>

19.3.20 RTC Backup registers (RTC_BKP(1~20))

Address offset: 0x50 to 0x9C

Reset value: 0x0000 0000



Bit field	Name	Description
31:0	BF[31:0]	<p>Backup data</p> <p>These registers can be wrote and read by software.</p> <p>These registers are powered by the BKR when MR is turned off, so when the system is reset, these registers are not reset and the contents of the registers are still valid when the device is operating in a low power mode.</p> <p>If RTC_TMPCFG.TPNOE=0, these registers are reset when tamper x event detection happens.</p>

20 Beeper

20.1 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.

20.2 Function description

As an independent module, the Beeper is mounted on the APB2 bus with a maximum operating frequency of 64MHz. One of the two outputs is usually turned off. If the complementary output is enabled, the two outputs are turned on at the same time and the outputs are complementary.

20.2.1 Main Features

Under normal operation mode, beeper can chose from two main clock sources:

1. Beeper supports clock range from (64MHz – 1MHz) for APB clock and 40KHz LSI clock .
2. Beeper outputs adjustable tone depending on register configuration (39 Hz to 64MHz).

Table 20-1 Max and Min frequency supported by beeper and corresponding configure.

	Frequency	Clk Source	PSC	BEEPDIV	BYPASSEN
Max	64 MHz	APB2 (64MHz)	0	/	1
Min	39 Hz	LSI (40K)	/	1024	0

3. Beeper can also by-pass clock signal to output ports by setting BEEPER_CTRL.BYPASSEN bit.

Beeper 2 output is complementary output for beeper 1 output, setting register bit will enable this output port. For both beeper it only requires configure once.

20.2.2 Setup frequency for pre-scale ratio of beeper

This beeper design supports APB2 clock working frequency between 1 to 64 MHz and need to set BEEPER_CTRL.PSC bits in correct manner in order to prevent distortion.

Pre-scale ratio is design to scale down APB2 clock to 1 MHz pulse signal and used to generate 64 MHz ~ 976.5625 Hz signal. However, to achieve accurate 8 KHz or 4 KHz, we can set APB2 clock to 64 MHz with divide ratio of 16 to get 4 MHz output. Then by setting BEEPER_CTRL.BEEPDIV dumber of 250 we can get 8 KHz signal and with BEEPER_CTRL.BEEPDIV = 500 we can get 4 KHz signal.

20.2.3 Bypass clock

Output clock of first stage after selection can be bypassed by setting BEEPER_CTRL.BYPASSEN bit. Which means PCLK/LSI can pass out to output directly. However for PCLK after pre-scale the duty cycle will not be 50%. The

max and min frequency supported for beeper is as Table 20-1.

After setting those register value, beeper can output with corresponding frequency after two stage of divide. Making beeper can output up to 64 MHz and as low as 39 Hz.

If user wants complementary output beeper 2 to work, just set BEEPER_CTRL.INVEN[2] bit to 1.

20.2.4 Output Duty cycle

There are two stages of frequency divider. First stage support odd frequency division. Output of this stage is waveform without 50% duty cycle. This is only work for PCLK. For LSI clock source, they are with 50% duty cycle. This means with bypass enable, using LSI clock source can output 50% duty cycle waveform. Without bypass, output waveform of beeper (beeper 1 or 2) should with 50% duty cycle.

20.3 Beeper registers

20.3.1 Beeper register overview

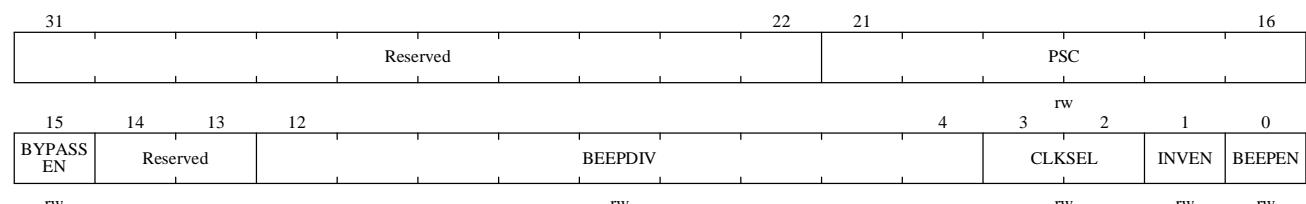
Table 20-2 Beeper register overview

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	Reserved										PSC					BYPASSEN	Reserved		BEEPDIV							CLKSEL	INVEN	BEEPEN				
	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			

20.3.2 Beeper Control Register (BEEPER_CTRL)

Address offset: 0x00

Reset value: 0x0000 0000



Bit field	Name	Description
31:22	Reserved	Reserved, the reset value must be maintained.
21:16	PSC	APB clock prescale factor. 00_0000: By pass value 00_0001: Frequency scale done by 2

Bit field	Name	Description
	 11_1111: Frequency scale done by 64
15	BYPASSEN	Bypass function enable Bypass at second stage with select clock signal 1: Bypass enable 0: Bypass disable
14:13	Reserved	Reserved, the reset value must be maintained.
12:4	BEEPDIV	Beeper divide factor 0_0000_0000: Output frequency value divided by 2 ((BEEPDIV+1)*2) 0_0000_0001: Output frequency value divided by 4 0_0000_0010: Output frequency value divided by 6 1_1111_1111: Output frequency value divided by 1024
3:2	CLKSEL	Clock source selection 0x : Using pre-scale APB clock as next stage clock 10 : Using LSI clock as next stage clock source 11 : Reserved
1	INVEN	Beeper complementary output enable Beeper_out1 and Beeper_out2 are complementary output signal. 0: There is only one output, and the other output is closed. 1: Both outputs are turned on, and the outputs are complementary
0	BEEPEN	Beeper enable 1: Beeper enable 0: Beeper disable

21 Debug support (DBG)

21.1 Overview

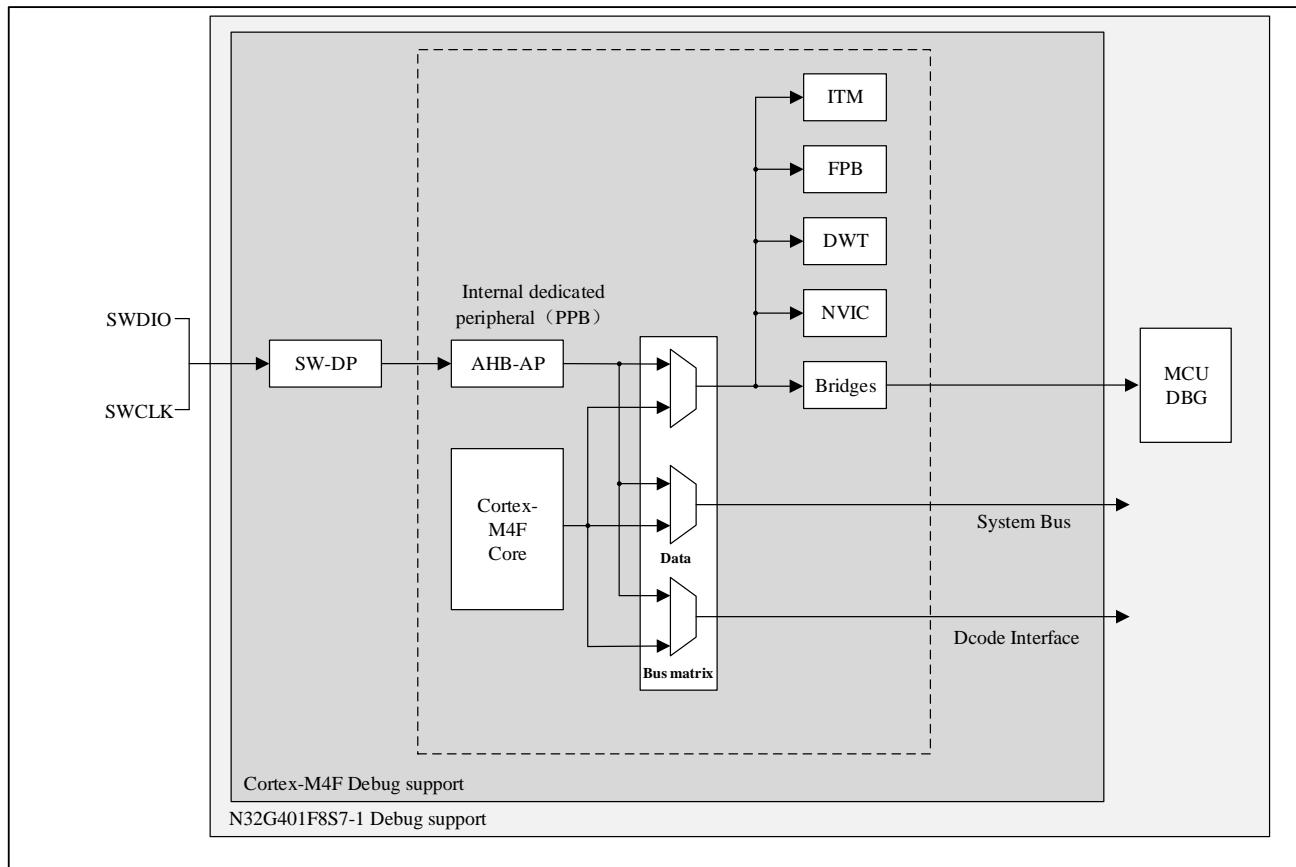
N32G401F8S7-1 uses Cortex®-M4 core, which integrates hardware debugging module. Support instruction breakpoint (stop when instruction fetches value) and data breakpoint (stop when data access). When the kernel is stopped, the user can view the internal state of the kernel and the external state of the system. After the user's query operation is completed, the kernel and peripherals can be restored, and the corresponding program can continue to be executed.

The hardware debugging module of the N32G401F8S7-1 kernel can be used when it is connected to the debugger (when it is not disabled).

N32G401F8S7-1 supports the following debugging interfaces:

- Serial wire

Figure 21-1 N32G401F8S7-1 level and Cortex®-M4 level debugging block diagram



The ARM Cortex™-M4F core hardware debugging module can provide the following debugging functions:

- SW-DP: serial debug port
- AHB-AP: AHB access port
- ITM: execution tracking unit

- FPB: Flash instruction breakpoint
- DWT: data trigger

Reference:

- Cortex™-M4 Technical Reference Manual (TRM)
- ARM debugging interface V5 structure specification
- ARM CoreSight development tool set (r1p0 version) technical reference manual

The system supports low-power mode debugging and debugging of some peripherals. The peripherals supporting debugging include: CAN, I2C interface and TIMER, WWDG and IWDG modules. The user needs to set the corresponding bit of the debug control register (DBG_CTRL) to 1 when debugging with low power consumption or peripherals.

21.2 SW function

The debugging tool can call the debugging function through the SW debugging interface mentioned above.

21.2.1 Pin allocation

SWD (serial debugging) interface includes two pins: SWCLK (clock pin) and SWDIO (data input and output pin).

See the following Table for the pin allocation of SW debugging interface:

Table 21-1 Debug port pin

Debug port	Pin allocation
SWDIO	PA13
SWCLK	PA14

- When SW debugging interface is enabled, SW debugging interface will be used by default after reset.
- When the debugging function is not used, the above two pins can be used as general-purpose GPIO.

21.3 MCU debugging function

21.3.1 Low power mode support

N32G401F8S7-1 can provide a variety of low power consumption modes (refer to chapter 3. Power control for details). When debugging, ensure that the FCLK and HCLK of the kernel are on, and provide the necessary clock for kernel debugging. Users can debug MCU in low power mode according to specific operation (ensuring the output of FCLK or HCLK in low power mode).

If users want to debug MCU in low power mode, they first need the debugger to configure registers related to low power mode:

- SLEEP mode:

Debugger needs to configure the SLEEP bit of DBG_CTRL register (configure HCLK output to have the same

frequency as FCLK).

■ STOP mode:

The debugger needs to configure the STOP bit of the DBG_CTRL register to start the internal RC oscillator to provide the clock for HCLK and FCLK.

21.3.2 Peripheral debugging support

When the corresponding bit of the peripheral control bit in the DBG_CTRL register is set to 1, the corresponding peripheral enters the debugging state after the kernel stops:

- Timer peripheral: the timer counter stops and debugs;
- I2C peripheral: the SMBUS of I2C keeps the state and carries out debugging;
- WWDG/IWDG peripheral: WWDG/IWDG counter clock stops and debugs;

21.4 DBG registers

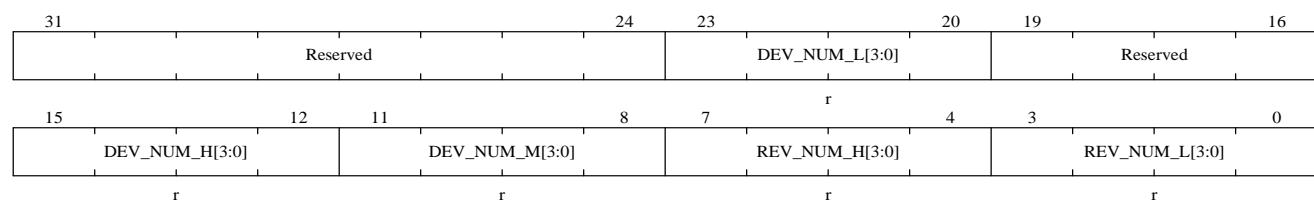
21.4.1 DBG register overview

The DBG register map and reset values are listed below. These peripheral registers must be operated as words (32 bits). The base address of the register is 0xE0042000.

Table 21-2 DBG 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	DBG_ID																																					
	Reset Value																																					
004h	DBG_CTRL																																					
	Reset Value																																					

21.4.2 ID register (DBG_ID)



Address offset: 0x04

Only 32-bit access is supported, and fixed values cannot be modified

Bit field	Name	Description
31:24	Reserved	Reserved, must keep the reset value.
23:20	DEV_NUM_L[3:0]	Lower 4 digits of equipment model. Device model consists of 12 bits, including high, medium and low, representing the model of MCU.
19:16	Reserved	Reserved, must keep the reset value.
15:12	DEV_NUM_H[3:0]	The upper 4 bits of device. See the description of DEV_NUM_L[3:0].
11:8	DEV_NUM_M[3:0]	The middle 4 bits of device. See the description of DEV_NUM_L[3:0].
7:4	REV_NUM_H[3:0]	High 4 bits of MCU version number
3:0	REV_NUM_L[3:0]	Low 4 bits of MCU version number

21.4.3 Debug control register (DBG_CTRL)

Address offset: 0x04

POR reset value: 0x0000 0000 (not reset by system reset)

31	Reserved										20	19	18	17	16
15	14	13	12	11	10	9	8	7			rw	rw	rw	rw	rw
12C1SM BUS TIMEOUT	Reserved	TIM4 _STOP	TIM3 _STOP	TIM2 _STOP	TIM1 _STOP	WWDG _STOP	IWDG _STOP		Reserved		3	2	1	0	

Bit field	Name	Description
31:20	Reserved	Reserved, the reset value must be maintained
19:17	TIMx_STOP	Stop the timer counter when the core stops (x=6,5,8). Set or cleared by software. 0: When the core stops, the clock is still provided to the counter of the related timer, and the timer output works normally; 1: When the core stops, turn off the clock of the counter of the related timer and turn off the output of the timer at the same time.
16:15	I2CxSMBUS_TIMEOUT	Stop the SMBUS timeout mode when the core stops (x=2,1). Set or cleared by software. 0: Same as normal mode operation; 1: Freeze the timeout control of SMBUS.
14	Reserved	Reserved, the reset value must be maintained
13:10	TIMx_STOP	When the kernel enters the debugging state, the counter stops working (x=4,3,2,1). Set or cleared by software. 0: The counter of the selected timer still works normally;

Bit field	Name	Description
		1: The counter of the selected timer stops working.
9	WWDG_STOP	<p>When the kernel enters the debug state, the debug window watchdog stops working.</p> <p>Set or cleared by software.</p> <p>0: The window watchdog counter still works normally; 1: Window watchdog counter stops working.</p>
8	IWDG_STOP	<p>The watchdog stops working when the kernel enters the debugging state.</p> <p>Set or cleared by software.</p> <p>0: Watchdog counter still works normally; 1: Watchdog counter stops working.</p>
7:3	Reserved	Reserved, must keep the reset value.
2	STDBY	<p>Debug standby mode.</p> <p>Set or cleared by software.</p> <p>0:(FCLK off, HCLK off) The whole digital circuit is powered off. From the software point of view, exiting the STANDBY mode is the same as resetting (except that some status bits indicate that the microcontroller has just exited from the STANDBY state).</p> <p>1:(FCLK ON, HCLK ON) The digital circuit part is not powered down, and the FCLK and HCLK clocks are clocked by the internal RLD oscillator. In addition, it is the same as resetting that the microcontroller exits the STANDBY mode by generating a system reset.</p>
1	STOP	<p>Debug stop mode.</p> <p>Set or cleared by software.</p> <p>0:(FCLK OFF, HCLK OFF) In stop mode, the clock controller disables all clocks (including HCLK and FCLK). When exiting from STOP mode, the configuration of the clock is the same as that after reset (the microcontroller is clocked by the 8MHz internal RC oscillator (HSI)). Therefore, the software must reconfigure the clock control system to start PLL, crystal oscillator, etc.</p> <p>1:(FCLK ON, HCLK ON) In stop mode, the FCLK and HCLK clocks are provided by the internal RC oscillator. When exiting the stop mode, the software must reconfigure the clock system to start PLL, crystal oscillator, etc. (the same operation as when this bit is set to 0).</p>
0	SLEEP	<p>Debug sleep mode.</p> <p>Set or cleared by software.</p> <p>0:(FCLK is on, HCLK is off) In sleep mode, FCLK is provided by the previously configured system clock, while HCLK is off. Since sleep mode does not reset the configured clock system, the software does not need to reconfigure the clock system when exiting from sleep mode.</p> <p>1:(FCLK ON, HCLK ON) In sleep mode, both the FCLK and HCLK clocks are provided by the previously configured system clock.</p>

22 Unique device serial number (UID)

22.1 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.

The UCID is 128 bits and complies with the definition of the national technology chip serial number. It contains information about chip production and version.

22.2 UID register

Start address: 0x1FFF_F7F0, 96 bits in length.

22.3 UCID register

Start address: 0x1FFF_F7C0, 128 bits in length.

23 Version history

Date	Version	Remark
2023.1.10	V1.0	Initial release

24 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.