



武汉芯源半导体有限公司

WUHAN XINYUAN SEMICONDUCTOR CO., LTD

CW32L010 Reference Manual

ARM® Cortex®-M0+ Low-power 32-bit MCU

Rev 1.0

www.whxy.com



Statement

Important notice

This reference manual is aimed at application developers and provides complete information on how to use the CW32L010 microcontroller memory and peripherals. This operation applies only to CW32L010 devices.

Disclaimer

Wuhan Xinyuan Semiconductor Co., Ltd. reserves the right to make changes, corrections, enhancements, and modifications to the product and this document at any time without notice. Buyers should obtain the latest product information on the official website.

Copyright statement

The copyright of this manual belongs to Wuhan Xinyuan Semiconductor Co., LTD., and we reserve the right of final interpretation and modification of this manual and statement.

Technical support

If you have any comments or suggestions in the process of using, please feel free to contact us.

Websit : www.whxy.com

Address : 5 WuDaYuan 3rd Road, Donghu High-tech Development Zone Wuhan Hubei, China

Postcode : 430070



Contents

Statement.....	1
1 Documentation conventions.....	22
1.1 Typographical conventions	22
1.2 Register protocol	23
2 System and memory overview.....	24
2.1 System architecture	24
2.2 Memory organization	26
2.2.1 Overview	26
2.2.2 Memory map and register boundary addresses	27
2.3 On-chip SRAM memory	28
2.4 On-chip FLASH memory	29
2.5 One-time programmable OTP memory	30
2.6 System boot configuration	31
2.7 Precautions.....	32
3 Power Control (PWR) and Power Consumption	33
3.1 Overview.....	33
3.2 Power supply supervisor	34
3.3 Operating mode	35
3.3.1 Entering Sleep mode or DeepSleep mode.....	36
3.3.2 Exiting Sleep mode or DeepSleep mode.....	37
3.3.3 Operating Modes and Reset Sources	39
3.4 Low-power consumption applied	40
3.5 Cortex®-M0+ kernel system control register (SCB->SCR)	41
4 Reset and clock control (RCC)	42
4.1 System reset	42
4.1.1 Power on reset/Brown-out reset (POR/BOR).....	42
4.1.2 Pin input reset (NRST)	43
4.1.3 IWDG reset.....	43
4.1.4 LVD low voltage detection reset.....	43
4.1.5 Kernel SYSRESETREQ reset.....	43
4.1.6 Kernel LOCKUP fault reset	43
4.2 Peripheral reset	44
4.3 Clock and Control.....	45
4.3.1 Overview.....	45
4.3.2 System Clock and Operating Mode.....	47



4.3.3	HSE clock.....	48
4.3.4	HSIOSC clock	50
4.3.5	LSE clock.....	51
4.3.6	LSI clock.....	53
4.3.7	SysClk system clock	53
4.3.8	On-chip peripheral clock control.....	54
4.4	Clock Start, Calibration, and Status Detection	55
4.4.1	Clock start.....	55
4.4.2	Clock calibration.....	56
4.4.3	Clock state detection	57
4.4.3.1	Clock stability detection	57
4.4.3.2	Clock start-up failure detection	58
4.4.3.3	In-flight failure detection of clocks.....	59
4.4.4	Clock verification and output.....	60
4.5	SysClk system clock switch.....	61
4.5.1	Standard Clock Switching Process	62
4.5.2	HSI clock switching process between different frequencies.....	63
4.5.3	Example of switching from other clocks to LSE	63
4.5.4	Example of switching from other clocks to HSE.....	64
4.5.5	Example of switching from other clocks to LSI.....	64
4.5.6	Example of switching from other clocks to HSI.....	65
4.6	List of registers	66
4.7	Register description	67
4.7.1	SYSCTRL_CR0 System control register 0.....	67
4.7.2	SYSCTRL_CR1 System Control Register 1	68
4.7.3	SYSCTRL_CR2 SYSCTRL_CR2 System Control Register 2	70
4.7.4	SYSCTRL_HSI Internal high frequency clock control register	72
4.7.5	SYSCTRL_LSI Internal low frequency clock control register.....	73
4.7.6	SYSCTRL_HSE External high frequency crystal control register	74
4.7.7	SYSCTRL_LSE External low frequency crystal control register.....	76
4.7.8	SYSCTRL_IER System interrupt enable control register	77
4.7.9	SYSCTRL_ISR System Interrupt Flag Register	78
4.7.10	SYSCTRL_ICR System interrupt flag clear register	80
4.7.11	SYSCTRL_AHBEN AHB Peripheral Clock Enable Control Register	81
4.7.12	SYSCTRL_APBEN1 APB Peripheral Clock Enable Control Register 1.....	82
4.7.13	SYSCTRL_APBEN2 APB Peripheral Clock Enable Control Register 2.....	83
4.7.14	SYSCTRL_AHBRSR AHB Peripheral Reset Control Register.....	84
4.7.15	SYSCTRL_APBRSR1 APB Peripheral Reset Control Register 1	85
4.7.16	SYSCTRL_APBRSR2 APB Peripheral Reset Control Register 2	86



4.7.17	SYSCTRL_RESETFLAG System Reset Flag Register	87
4.7.18	SYSCTRL_DEBUG Debug Status Timer Control Register.....	88
4.7.19	SYSCTRL_MCO System Clock Output Control Register	89
5	Interrupt.....	90
5.1	Overview.....	90
5.2	Main features	90
5.3	Interrupt priority	90
5.4	Interrupt vector table.....	91
5.5	Interrupt related registers.....	93
5.5.1	NVIC interrupt enable and disable enable	93
5.5.2	NVIC interrupt pending and clear pending	93
5.5.3	NVIC interrupt priority	93
5.5.4	NVIC interrupt mask.....	94
5.5.5	Peripheral interrupt enable	94
5.6	List of registers	95
5.7	Register description	96
5.7.1	NVIC_ISER Interrupt Enable Setting Register.....	96
5.7.2	NVIC_ICER Interrupt Enable Clear Register	96
5.7.3	NVIC_ISPR Interrupt Pending Setup Register	96
5.7.4	NVIC_ICPR Interrupt Pending Clear Register	97
5.7.5	NVIC_IPR0 Interrupt Priority Control Register 0	97
5.7.6	NVIC_IPR1 Interrupt Priority Control Register 1	98
5.7.7	NVIC_IPR2 Interrupt Priority Control Register 2	98
5.7.8	NVIC_IPR3 Interrupt Priority Control Register 3	99
5.7.9	NVIC_IPR4 Interrupt Priority Control Register 4	99
5.7.10	NVIC_IPR5 Interrupt Priority Control Register 5	100
5.7.11	NVIC_IPR6 Interrupt Priority Control Register 6	100
5.7.12	NVIC_IPR7 Interrupt Priority Control Register 7	101
6	RAM Memory	102
6.1	Overview.....	102
6.2	Main features	102
6.3	RAM Memory operation	103
6.3.1	Read operation	103
6.3.2	Write operation.....	103
6.4	Parity check function	104
6.5	List of registers	105
6.6	Register description	106
6.6.1	RAM_ADDR Parity Error Address Register	106



6.6.2	RAM_IER Interrupt Enable Control Register	106
6.6.3	RAM_ISR Interrupt Flag Register	106
6.6.4	RAM_ICR Interrupt Flag Clear Register	106
7	FLASH Memory	107
7.1	Overview	107
7.2	Main features	107
7.3	FLASH Memory organization	107
7.4	FLASH memory read wait cycle configuration	107
7.5	FLASH Memory operation	108
7.5.1	Page Erase	109
7.5.2	Write operation	110
7.5.3	Read operation	112
7.6	FLASH Memory protection	113
7.6.1	Erase and write protection	113
7.6.2	Erase and write PC page protection	114
7.6.3	Read protection	115
7.6.4	Security Runtime Library Protection	116
7.7	FLASH memory programming	117
7.8	Precautions	118
7.9	List of registers	119
7.10	Register description	120
7.10.1	FLASH_CR1 Control register 1	120
7.10.2	FLASH_CR2 Control register 2	120
7.10.3	FLASH_PAGELOCK Erase lock register	121
7.10.4	FLASH_IER Interrupt Enable Register	122
7.10.5	FLASH_ISR Interrupt Flag Register	123
7.10.6	FLASH_ICR Interrupt Flag Clear Register	124
7.10.7	FLASH_SDKCFR Security Runtime Library Area Register	124
8	General-purpose input/output (GPIO)	125
8.1	Overview	125
8.2	Main features	125
8.3	Function description	126
8.3.1	Functional block diagram	126
8.3.2	Digital output	127
8.3.3	Digital input	128
8.3.4	Analogue function	129
8.3.5	Multiplexing function	129
8.3.6	Interrupt Function	131



8.3.7	Other functions.....	132
8.4	Programming example	133
8.4.1	Digital output programming example	133
8.4.2	Digital input programming example.....	133
8.4.3	Analog functional programming example	133
8.4.4	Multiplexed function programming example	133
8.4.5	Interrupt function programming example	134
8.5	List of registers	135
8.6	Register description	136
8.6.1	GPIOx_DIR GPIO I/O direction register (x =A, B)	136
8.6.2	GPIOx_OPENDRAIN GPIO output mode register (x =A, B)	136
8.6.3	GPIOx_PUR GPIO Pull-up resistor register (x =A, B)	137
8.6.4	GPIOx_AFRH GPIO Multiplexing function register high segment (x =A, B)	137
8.6.5	GPIOx_AFRL GPIO Multiplexing function register low segment (x =A, B)	137
8.6.6	GPIOx_ANALOG GPIO Analog Digital Configuration Register (x =A, B)	138
8.6.7	GPIOx_RISEIE GPIO Rising edge interrupt enable register (x =A, B)	138
8.6.8	GPIOx_FALLIE GPIO Falling edge interrupt enable register (x =A, B)	138
8.6.9	GPIOx_ISR GPIO Interrupt flag register (x =A, B)	139
8.6.10	GPIOx_ICR GPIO Interrupt flag clear register (x =A, B)	139
8.6.11	GPIOx_FILTER GPIO Interrupt digital filter configuration register (x =A, B)	139
8.6.12	GPIOx_IDR GPIO input data register (x =A, B)	140
8.6.13	GPIOx_ODR GPIO output data register (x =A, B)	140
8.6.14	GPIOx_BRR GPIO port bit set clear register (x =A, B)	140
8.6.15	GPIOx_BSRR GPIO port bit-setting clear register (x =A, B)	141
8.6.16	GPIOx_TOG GPIO Port Bit Flip Register (x =A, B)	141
9	Cyclic redundancy check (CRC)	142
9.1	Overview.....	142
9.2	Main features	142
9.3	Functional description	143
9.3.1	Algorithmic modes.....	143
9.3.2	Input data bit width	145
9.4	Programming examples.....	146
9.4.1	CRC16_CCITT algorithm mode	146
9.5	List of registers	147
9.6	Register descriptions.....	148
9.6.1	CRC_CR control register	148
9.6.2	CRC_DR data register.....	148
9.6.3	CRC_RESULT result register	148



10	Real time clock (RTC)	149
10.1	Overview	149
10.2	Main features	149
10.3	Functional description	150
10.3.1	Functional block diagram	150
10.3.2	Clock and prescaler	151
10.3.3	Real Time Clock and Calendar	152
10.3.4	RTC initialization settings	153
10.3.5	Register lock function	154
10.3.6	Register access operations	154
10.3.7	RTCOUNT output	155
10.3.8	1Hz signal output	155
10.3.9	Clock error compensation	156
10.3.10	Alarm clock A and alarm clock B	157
10.3.11	Period interrupt function	158
10.3.12	Automatic wake-up function	159
10.3.13	Timestamp function	160
10.3.14	RTC Interrupts	160
10.4	List of registers	161
10.5	Register descriptions	162
10.5.1	RTC_KEY key value register	162
10.5.2	RTC_CR0 control register 0	163
10.5.3	RTC_CR1 control register 1	164
10.5.4	RTC_CR2 control register 2	165
10.5.5	RTC_PSC clock prescaler register	166
10.5.6	RTC_DATE date register	166
10.5.7	RTC_TIME time register	167
10.5.8	RTC_SSCNT subsecond count register	167
10.5.9	RTC_ALARMA alarm clock A control register	168
10.5.10	RTC_ALARMB alarm clock B control register	169
10.5.11	RTC_TAMPDATE timestamp date register	169
10.5.12	RTC_TAMPTIME timestamp time register	170
10.5.13	RTC_IER interrupt enable register	170
10.5.14	RTC_ISR interrupt flag register	171
10.5.15	RTC_ICR interrupt flag clear register	172
10.5.16	RTC_AWTARR wake-up timer reload value register	172
10.5.17	RTC_AWTCNT wake-up timer count value register	173
10.5.18	RTC_COMPCFR1 wake-up clock error compensation register 1	173
11	Basic timer (BTIM)	174



11.1	Overview.....	174
11.2	Main features	174
11.3	Functional description	175
11.3.1	Functional block diagram	175
11.3.1.1	Counting unit.....	176
11.3.1.2	Update Event (UEV).....	178
11.3.1.3	Trigger input channel	179
11.3.1.4	Trigger output channel.....	179
11.3.1.5	Reset input channel.....	180
11.3.1.6	Flip output unit.....	180
11.3.2	Operating modes.....	181
11.3.2.1	Internal count mode.....	182
11.3.2.2	External count mode	183
11.3.2.3	Trigger start mode.....	184
11.3.2.4	Gated count mode.....	185
11.3.3	UIF bit remap	186
11.4	BTIM interrupt	187
11.5	Trigger ADC	188
11.6	Debugging support	189
11.7	Programming examples.....	190
11.7.1	Internal count mode programming example	190
11.7.2	External count mode programming example.....	190
11.7.3	Trigger start mode programming example	191
11.7.4	Gated count mode programming example	192
11.7.5	Counter reset programming example.....	193
11.8	List of registers	194
11.9	Register descriptions.....	195
11.9.1	BTIMx_CR1 control register 1	195
11.9.2	BTIMx_CR2 control register 2	196
11.9.3	BTIMx_SMCR slave mode control register.....	197
11.9.4	BTIMx_IER interrupt enable register.....	199
11.9.5	BTIMx_ISR interrupt flag register	199
11.9.6	BTIMx_ICR interrupt flag clear register	199
11.9.7	BTIMx_EGR event generation register.....	200
11.9.8	BTIMx_CNT count register	200
11.9.9	BTIMx_PSC prescaler register.....	201
11.9.10	BTIMx_ARR auto reload register.....	201
12	Low power timer (LPTIM)	202



12.1	Overview	202
12.2	Main features	202
12.3	Function description	203
12.3.1	Functional block diagram	203
12.3.1.1	Clock source	204
12.3.1.2	Prescaler	204
12.3.1.3	Filter unit	205
12.3.2	Start method	206
12.3.3	Trigger source	207
12.3.4	Operating mode	208
12.3.5	Timeout function	210
12.3.6	Waveform output control	210
12.3.7	Register update	211
12.3.8	Count function	211
12.3.9	Encoding mode	212
12.4	Debug support	214
12.5	Programming example	215
12.5.1	External pulse counting routine	215
12.6	List of registers	216
12.7	Register descriptions	217
12.7.1	PTIM_CFGR Configuration register	217
12.7.2	LPTIM_CR Control register	220
12.7.3	LPTIM_ARR Auto-reload register	220
12.7.4	LPTIM_CNT Counter register	221
12.7.5	LPTIM_CMP Compare register	221
12.7.6	LPTIM_IER Interrupt enable register	222
12.7.7	LPTIM_ISR Interrupt and status register	223
12.7.8	LPTIM_ICR Interrupt clear register	224
13	General-purpose timer (GTIM)	225
13.1	Overview	225
13.2	Main features	225
13.3	Functional description	226
13.3.1	Functional block diagram	226
13.3.1.1	Clock source	227
13.3.1.2	Prescaler	227
13.3.1.3	Counter and count mode	228
13.3.1.4	Reload register	235
13.3.1.5	Update event UEV	235



13.3.1.6	Single pulse mode	236
13.3.1.7	External trigger input channel.....	237
13.3.1.8	Input capture channel	237
13.3.1.9	Output compare channel	238
13.3.2	Operating modes.....	239
13.3.2.1	Internal clock mode.....	239
13.3.2.2	External clock mode	240
13.3.2.3	Reset mode	242
13.3.2.4	Gated mode	243
13.3.2.5	Trigger mode.....	244
13.3.2.6	Combined reset + trigger mode.....	244
13.3.2.7	Combined gated + reset mode	245
13.3.2.8	Quadrature encoder mode.....	245
13.3.2.9	Clock + direction encoder mode	248
13.3.2.10	Directional clock encoder mode	249
13.3.3	Input capture function	251
13.3.3.1	PWM input mode.....	252
13.3.3.2	Input capture source	253
13.3.4	Output compare function	254
13.3.4.1	Match output	256
13.3.4.2	Forced output	257
13.3.4.3	PWM output	258
13.3.4.4	Asymmetric PWM output.....	260
13.3.4.5	Combined PWM output.....	262
13.3.4.6	Retriggerable single pulse mode.....	264
13.3.4.7	Count direction output	265
13.3.4.8	Reference signal clear function.....	265
13.3.5	UIF bit remap	266
13.3.6	Timer cascade ITR.....	266
13.3.7	On-chip peripheral interconnect ETR	268
13.4	GTIM interrupt	269
13.5	Trigger ADC	270
13.6	Debugging support	271
13.7	Programming examples.....	272
13.7.1	External clock mode 1 programming example.....	272
13.7.2	External clock mode 2 programming example.....	272
13.7.3	Reset mode programming example.....	273
13.7.4	Gated mode programming example.....	273
13.7.5	Trigger mode programming example.....	274



13.7.6	Quadrature encoder mode programming example	274
13.7.7	Input capture programming example.....	275
13.7.7.1	Basic input capture mode	275
13.7.7.2	PWM input mode.....	276
13.7.8	Output compare programming example	277
13.8	List of registers	278
13.9	Register descriptions.....	279
13.9.1	GTIM_CR1 Controls register 1.....	279
13.9.2	GTIM_CR2 Control register 2.....	281
13.9.3	GTIM_SMCR Slave mode control register	283
13.9.4	GTIM_IER Interrupt enable register	287
13.9.5	GTIM_ISR Interrupt flag register	288
13.9.6	GTIM_ICR Interrupt flag clear register	290
13.9.7	GTIM_EGR Event generation register	291
13.9.8	GTIM_CCMR1CAP Capture mode register 1	292
13.9.9	GTIM_CCMR1CMP Compare mode register 1	293
13.9.10	GTIM_CCMR2CAP Capture mode register 2	297
13.9.11	GTIM_CCMR2CMP Compare mode register 2	298
13.9.12	GTIM_CCER Capture/compare enable register.....	299
13.9.13	GTIM_CNT Count register.....	300
13.9.14	GTIM_PSC Prescaler register	301
13.9.15	GTIM_ARR Auto-reload register.....	301
13.9.16	GTIM_CCR1 Capture/compare register 1.....	302
13.9.17	GTIM_CCR2 Capture/compare register 2.....	302
13.9.18	GTIM_CCR3 Capture/compare register 3.....	303
13.9.19	GTIM_CCR4 Capture/compare register 4.....	303
13.9.20	GTIM_ECR Encoder control register	304
13.9.21	GTIM_TISEL TI input selection register	305
13.9.22	GTIM_AF1 Multiplexing function option register 1.....	306
13.9.23	GTIM_AF2 Multiplexing function option register 2.....	306
14	Advanced-control timer (ATIM)	307
14.1	Overview.....	307
14.2	Main features	307
14.3	Functional description	308
14.3.1	Functional block diagram.....	308
14.3.1.1	Clock source.....	309
14.3.1.2	Prescaler	309
14.3.1.3	Counter and count mode	310
14.3.1.4	Reload register.....	317



14.3.1.5	Repeat counter	318
14.3.1.6	Update event UEV	319
14.3.1.7	Single pulse mode	320
14.3.1.8	External trigger input channel.....	321
14.3.1.9	Input capture channel.....	321
14.3.1.10	Output compare channel	322
14.3.2	Operating modes.....	323
14.3.2.1	Internal clock mode.....	323
14.3.2.2	External clock mode	324
14.3.2.3	Reset mode	326
14.3.2.4	Gated mode	327
14.3.2.5	Trigger mode.....	328
14.3.2.6	Combined reset + trigger mode.....	328
14.3.2.7	Combined gated + reset mode	329
14.3.2.8	Quadrature encoder mode.....	329
14.3.2.9	Clock + direction encoder mode	332
14.3.2.10	Directional clock encoder mode	333
14.3.3	Input capture function	335
14.3.3.1	PWM input mode.....	336
14.3.3.2	Input capture source	337
14.3.4	Output compare function	338
14.3.4.1	Match output	339
14.3.4.2	Forced output.....	340
14.3.4.3	PWM output	341
14.3.4.4	Asymmetric PWM output.....	343
14.3.4.5	Combined PWM output.....	345
14.3.4.6	Phase shift PWM mode.....	347
14.3.4.7	Complementary output and dead-time insertion.....	349
14.3.4.8	Brake function	351
14.3.4.9	Retriggerable single pulse mode.....	355
14.3.4.10	Count direction output	355
14.3.4.11	Reference signal clear function.....	356
14.3.5	UIF bit remap	357
14.3.6	Timer cascade ITR.....	357
14.3.7	On-chip peripheral interconnect ETR	358
14.4	ATIM interrupt.....	359
14.5	Trigger ADC	360
14.6	Debugging support	361
14.7	Programming examples.....	362



14.7.1	External clock mode 1 programming example.....	362
14.7.2	External clock mode 2 programming example.....	362
14.7.3	Reset mode programming example.....	363
14.7.4	Gated mode programming example.....	363
14.7.5	Trigger mode programming example.....	364
14.7.6	Quadrature encoder mode programming example	364
14.7.7	Input capture programming example.....	365
	14.7.7.1 Basic input capture mode	365
	14.7.7.2 PWM input mode.....	366
14.7.8	Output compare programming example	367
14.8	List of registers	368
14.9	Register descriptions.....	370
14.9.1	ATIM_CR1 Control register 1	370
14.9.2	ATIM_CR2 Control register 2	372
14.9.3	ATIM_SMCR Slave mode control register	375
14.9.4	ATIM_IER Interrupt enable register.....	379
14.9.5	ATIM_ISR Interrupt flag register	381
14.9.6	ATIM_ICR Interrupt flag clear register.....	384
14.9.7	ATIM_EGR Event generation register.....	386
14.9.8	ATIM_CCMR1CAP Capture mode register 1.....	387
14.9.9	ATIM_CCMR1CMP Compare mode register 1.....	389
14.9.10	ATIM_CCMR2CAP Capture mode register 2.....	392
14.9.11	ATIM_CCMR2CMP Compare mode register 2.....	393
14.9.12	ATIM_CCMR3CAP Capture mode register 3.....	394
14.9.13	ATIM_CCMR3CMP Compare mode register 3.....	395
14.9.14	ATIM_CCER Capture/compare enable register	396
14.9.15	ATIM_CNT Count register	399
14.9.16	ATIM_PSC Prescaler register.....	399
14.9.17	ATIM_ARR Auto reload register.....	399
14.9.18	ATIM_RCR Repeat count register	400
14.9.19	ATIM_CCR1 Capture/compare register 1	400
14.9.20	ATIM_CCR2 Capture/compare register 2	401
14.9.21	ATIM_CCR3 Capture/compare register 3	401
14.9.22	ATIM_CCR4 Capture/compare register 4	401
14.9.23	ATIM_CCR5 Capture/compare register 5	402
14.9.24	ATIM_CCR6 Capture/compare register 6	403
14.9.25	ATIM_BDTR Brake and deadband register	403
14.9.26	ATIM_DTR2 Dead time register 2.....	408
14.9.27	ATIM_ECR Encoder control register.....	409



14.9.28	ATIM_TISEL1 TI input selection register 1	410
14.9.29	ATIM_TISEL2 TI input selection register 2	410
14.9.30	ATIM_AF1 Multiplexing function option register 1	411
14.9.31	ATIM_AF2 Multiplexing function option register 2	413
15	Independent watchdog timer (IWDT)	415
15.1	Overview	415
15.2	Main Features	415
15.3	Functional description	416
15.3.1	Functional block diagram	416
15.3.2	Operating modes	416
15.3.3	Window options	417
15.3.4	Register lock function	417
15.3.5	Start refresh and stop	417
15.3.6	Status register	418
15.3.7	Timing duration setting	419
15.4	Programming examples	420
15.4.1	Configure IWDT as independent watchdog	420
15.4.2	Configure IWDT as window watchdog	420
15.4.3	Refresh IWDT (feed dog operation)	420
15.5	List of registers	421
15.6	Register descriptions	422
15.6.1	IWDT_KR Key-value register	422
15.6.2	IWDT_CR Control register	423
15.6.3	IWDT_ARR Reload register	423
15.6.4	IWDT_CNT Count value register	424
15.6.5	IWDT_WINR Window register	424
15.6.6	IWDT_SR Status register	425
16	Universal asynchronous receiver/transmitter (UART)	426
16.1	Overview	426
16.2	Main features	427
16.3	Functional description	428
16.3.1	Functional block diagram	428
16.3.2	Synchronous mode	430
16.3.2.1	Baud rate setting	430
16.3.2.2	Data transmission and reception	431
16.3.3	Asynchronous mode	432
16.3.3.1	Data frame format	432
16.3.3.2	Fractional baud rate generator	433



16.3.3.3	Transmit control	437
16.3.3.4	Receive control	438
16.3.3.5	Single-wire half-duplex mode.....	440
16.3.3.6	Multi-machine communication	440
16.3.3.7	Hardware flow control.....	442
16.3.3.8	RS485 driver enable	444
16.3.3.9	LoopBack mode	445
16.3.3.10	Input signal source/level shift.....	445
16.3.4	LIN Mode.....	446
16.3.4.1	LIN transmit.....	446
16.3.4.2	LIN receive.....	447
16.3.5	Timer operating mode	448
16.3.5.1	Waiting for timeout detection mode	448
16.3.5.2	Receive idle detection mode.....	448
16.3.5.3	Auto baud rate detection mode.....	449
16.3.5.4	General purpose timer mode.....	449
16.4	Low-power mode	450
16.5	UART interrupt.....	451
16.6	Programming examples.....	452
16.6.1	Asynchronous full-duplex programming examples	452
16.6.1.1	Transmit data in query mode.....	452
16.6.1.2	Receive data in query mode.....	453
16.6.1.3	Transmit data in interrupt mode	454
16.6.1.4	Receive data in interrupt mode	455
16.6.2	Synchronous half-duplex programming examples	456
16.6.2.1	Transmit data in query mode.....	456
16.6.2.2	Receive data in query mode.....	456
16.6.3	Single-wire half-duplex programming examples.....	457
16.6.3.1	Transmit data in query mode.....	457
16.6.3.2	Receive data in query mode.....	458
16.7	List of registers	459
16.8	Register descriptions.....	460
16.8.1	UARTx_CR1 Control register 1	460
16.8.2	UARTx_CR2 Control register 2	462
16.8.3	UARTx_CR3 Control register 3	464
16.8.4	UARTx_BRRI Baud rate counter integer part register	464
16.8.5	UARTx_BRRF Baud rate counter fractional register	465
16.8.6	UARTx_TIMARR Timer reload register	465
16.8.7	UARTx_TIMCNT Timer count register.....	465



16.8.8	UARTx_TDR Transmit data register.....	466
16.8.9	UARTx_RDR Receive data register.....	466
16.8.10	UARTx_IER Interrupt enable register.....	467
16.8.11	UARTx_ISR Interrupt flag register.....	468
16.8.12	UARTx_ICR Interrupt flag clear register.....	470
16.8.13	UARTx_ADDR Slave address register.....	471
16.8.14	UARTx_MASK Slave address mask register	471
16.8.15	UARTx_RXMATCH Receive data match register	471
17	Serial peripheral interface (SPI)	472
17.1	Overview.....	472
17.2	Main features	472
17.3	Functional description.....	473
17.3.1	Functional block diagram.....	473
17.3.2	Communication timing and data format	475
17.3.3	Slave select pin CS	477
17.3.4	Full-duplex mode	478
17.3.5	Single-wire half-duplex mode.....	480
17.3.6	Simplex mode.....	482
17.3.6.1	Master single transmitting/slave single receiving.....	483
17.3.6.2	Master single receiving/slave single transmitting.....	484
17.3.7	Multi-machine communication	485
17.3.8	Status flags	487
17.3.9	Error flags	488
17.4	SPI interrupts.....	489
17.5	Programming examples.....	490
17.5.1	Full duplex/master mode.....	490
17.5.1.1	Transmit and receive in query mode	490
17.5.1.2	Transmit and receive in interrupt mode.....	491
17.5.2	Single-wire half-duplex/master mode.....	492
17.5.2.1	Transmit in query mode.....	492
17.5.2.2	Receive in query mode.....	493
17.5.3	Simplex mode/master mode	494
17.5.3.1	Transmit in query mode.....	494
17.5.3.2	Receive in query mode.....	495
17.6	List of registers	496
17.7	Register descriptions.....	497
17.7.1	SPI_CR1 Control register 1.....	497
17.7.2	SPI_CR2 Control register 2.....	499



17.7.3	SPI_CR3 Control register 3	499
17.7.4	SPI_SSI Slave select register	500
17.7.5	SPI_IER Interrupt enable register	501
17.7.6	SPI_ISR Interrupt flag register	502
17.7.7	SPI_ICR Interrupt flag clear register	503
17.7.8	SPI_DR Data register	503
18	I2C interface	504
18.1	Overview	504
18.2	Main features	504
18.3	Protocol description	505
18.3.1	Protocol frame format	505
18.3.2	Transmission response	507
18.3.3	Conflict detection and arbitration	507
18.4	Functional description	509
18.4.1	Functional block diagram	509
18.4.2	Serial clock generator	510
18.4.3	Input filter	511
18.4.4	Address comparator	511
18.4.5	Arbitration and synchronization logic	512
18.4.5.1	SCL synchronization	512
18.4.5.2	SDA arbitration	512
18.4.6	Response control	513
18.4.7	Input signal source/level shift	514
18.4.8	SMBUS timeout detection	515
18.4.9	I2C interrupts	515
18.4.10	Operating modes	516
18.4.10.1	Master transmit mode	516
18.4.10.2	Master receive mode	519
18.4.10.3	Slave receive mode	521
18.4.10.4	Slave transmit mode	523
18.4.10.5	Broadcast receive mode	525
18.4.11	Multi-master communication	527
18.4.12	I2C status codes	527
18.5	Programming examples	532
18.5.1	Master transmits example	532
18.5.2	Master receives example	533
18.5.3	Slave receives example	534
18.5.4	Slave transmits example	534
18.6	List of registers	535



18.7	Register descriptions.....	536
18.7.1	I2C_BRREN Baud rate counter enable register.....	536
18.7.2	I2C_BRR Baud rate counter configuration register.....	536
18.7.3	I2C_CR Control register	537
18.7.4	I2C_DR Data register	539
18.7.5	I2C_STAT status register	539
18.7.6	I2C_ADDR0 Slave address 0 register.....	539
18.7.7	I2C_ADDR1 Slave address 1 register.....	540
18.7.8	I2C_ADDR2 Slave address 2 register.....	540
18.7.9	I2C_MATCH Slave address match register.....	541
19	Infrared modulation transmitter (IR).....	542
19.1	Overview.....	542
19.2	Main features	542
19.3	Functional description	543
19.3.1	Infrared modulation mode	544
19.3.2	Infrared modulation initialization configuration.....	546
19.3.3	Infrared reception.....	546
19.4	IRMOD_CR Infrared modulation control register.....	547
20	Analog-to-digital converter (ADC)	548
20.1	Overview.....	548
20.2	Main features	548
20.3	Functional block diagram	549
20.4	Conversion timing, conversion speed, conversion accuracy, and conversion results	550
20.4.1	Conversion timing.....	550
20.4.2	Conversion speed.....	552
20.4.3	Conversion results.....	552
20.5	Operating modes.....	553
20.5.1	Sequence continuous conversion mode (CONT=1)	554
20.5.2	Sequence scan conversion mode (CONT=0)	556
20.6	External trigger source	558
20.7	Analog watchdog.....	559
20.8	BGR module.....	560
20.9	Temperature sensor	561
20.10	ADC interrupts	562
20.11	List of registers	563
20.12	Register descriptions.....	564



20.12.1	ADC_CR Control register	564
20.12.2	ADC_START Startup register	565
20.12.3	ADC_SAMPLE Sample cycle configuration register	565
20.12.4	ADC_SQRCFR Sequence conversion channel configuration register	566
20.12.5	ADC_AWDTR Analogue watchdog threshold register	566
20.12.6	ADC_AWDCR Analogue watchdog configuration register	567
20.12.7	ADC_TRIGGER External trigger register	568
20.12.8	ADC_IER Interrupt enable register	570
20.12.9	ADC_ISR Interrupt flag register	570
20.12.10	ADC_ICR Interrupt flag clear register	571
20.12.11	ADC_RESULT0 Sequence conversion result register 0	571
20.12.12	ADC_RESULT1 Sequence conversion result register 1	571
20.12.13	ADC_RESULT2 Sequence conversion result register 2	572
20.12.14	ADC_RESULT3 Sequence conversion result register 3	572
20.12.15	ADC_RESULT4 Sequence conversion result register 4	572
20.12.16	ADC_RESULT5 Sequence conversion result register 5	572
20.12.17	ADC_RESULT6 Sequence conversion result register 6	572
20.12.18	ADC_RESULT7 Sequence conversion result register 7	573
21	Analog voltage comparator (VC)	574
21.1	Overview	574
21.2	Main features	574
21.3	Functional description	575
21.3.1	Functional block diagram	575
21.3.2	I/O pins	577
21.3.3	Latency/Response time	577
21.3.4	Polarity selection	578
21.3.5	Digital filter	578
21.3.6	Hysteresis function	579
21.3.7	Window compare function	579
21.3.8	BLANK window function	580
21.4	VC interrupts	581
21.5	Programming examples	581
21.6	List of registers	582
21.7	Register descriptions	583
21.7.1	VC_DIV Resistor divider control register	583
21.7.2	VCx_CR0 Control register 0	584
21.7.3	VCx_CR1 Control register 1	585
21.7.4	VCx_SR Status register	587



22	Low voltage detector (LVD).....	588
22.1	Overview.....	588
22.2	Main features	588
22.3	Functional description	589
22.3.1	Functional block diagram.....	589
22.3.2	Hysteresis function.....	590
22.3.3	Digital filter	591
22.4	LVD interrupts	592
22.5	Programming examples.....	593
22.5.1	Brown-out reset programming example.....	593
22.5.2	Interrupt programming example.....	593
22.6	List of registers	594
22.7	Register descriptions.....	595
22.7.1	LVD_CR0 Control register 0	595
22.7.2	LVD_CR1 Control register 1	596
22.7.3	LVD_SR Status register.....	597
23	Debug interface (DBG)	598
23.1	Overview.....	598
23.2	Serial wire debug port SWD	598
23.3	Debug communication protocol	600
23.3.1	Transmission protocol format.....	600
23.3.2	SW-DP state machine	601
23.3.3	Read and write access for SW-DP and SW-AP	602
23.3.4	SW-DP register	603
23.3.5	SW-AP register.....	604
23.4	Kernel debugging	605
23.5	Breakpoint unit BPU	605
23.6	Data observation points and tracking DWT	605
23.7	Debug components DBG.....	606
23.8	Cautions.....	607
24	Digital signature	608
24.1	Overview.....	608
24.2	Product unique identification (UID) register (80bit).....	608
24.3	Product model register	608
24.4	FLASH capacity register	608
24.5	SRAM capacity register.....	609



24.6 Pin number register..... 609

25 Revision history.....610



1 Documentation conventions

1.1 Typographical conventions

The typographical conventions used in this document are:

<i>Italic</i>	Indicates a special term or citation, or prompts a note.
Bold	Indicates a component or signal name, such as for title descriptions without chapter numbers, titles for diagrams, etc.
English uppercase	Used for terms that have a specific technical meaning and are included in the glossary, such as register or bit field names.

[Color text](#) Indicates a link. Including:

- URL external link, such as www.whxy.com
- Cross-references, internal chapters of this paper refer to each other
- Internal links, figures, tables or appendices, glossary entries



1.2 Register protocol

Symbol example	Describe
0x53CA	A numeric string prefixed with '0x' represents a hexadecimal number, and the value is composed of 0~9 or A~F.
REGNAME[n]	A specific bit of REGNAME, REGNAME can be a register or register field, for example, CR1[2] refers to bit 2 of the CR1 register (field).
REGNAME[m:n]	A specific bit of REGNAME, REGNAME can be a register or a register field, for example, CR1[7:5] refers to the 7th to 5th bits of the CR1 register (field).
RW	Read and write, software can read and write these bit fields.
RO	Read-only, software can only read these bit fields.
WO	Write-only, software can only write to this bit field, and invalid data will be returned when reading this bit field.
RW0	Software can read and write this bit field, only 0 can be written to this bit field, and writing 1 has no effect on this bit field.
RW1	Software can read and write this bit field, only 1 can be written to this bit field, and writing 0 has no effect on this bit field.
R0W1	When the software reads this bit field as 0, only 1 can be written to this bit field, and writing 0 has no effect on this bit field.
R1W0	When the software reads this bit field as 1, only 0 can be written to this bit field, and writing 1 has no effect on this bit field.
RFU	Reserved bit field, please keep the default value.
Word	The data length of a word is 32 bits.
Half Word	The data length of a half word is 16 bits.
Byte	The data length of one byte is 8bit.



2 System and memory overview

2.1 System architecture

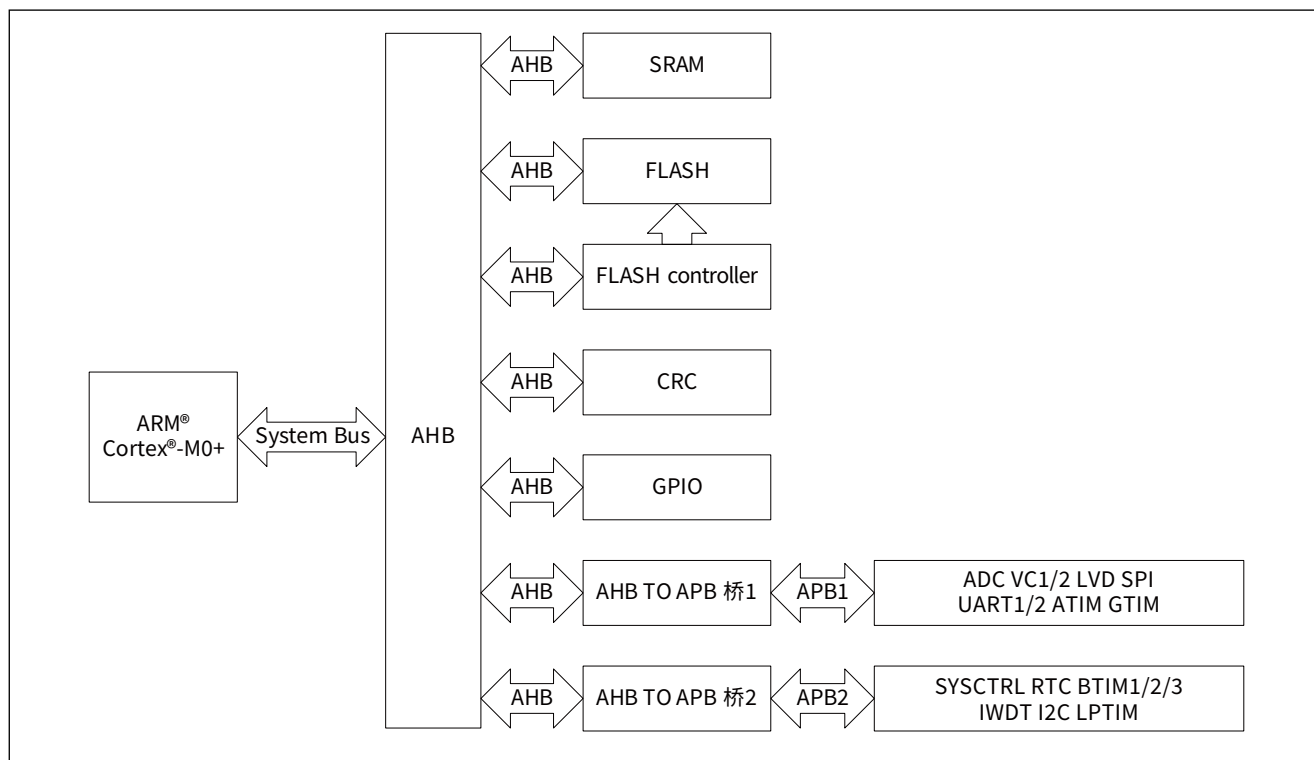
The consists of CW32L010 microcontroller system contains:

- 1 master device:
 - ARM® Cortex®-M0+ core
- Multiple slave device:
 - On-chip SRAM
 - On-chip FLASH
 - FLASH controller
 - CRC redundancy calculation unit
 - GPIO port
 - AHB to APB1 conversion bridge and all devices on the APB1 bus
 - AHB to APB2 conversion bridge and all devices on the APB2 bus



The master and slave devices are connected through AHB bus matrix, and the system architecture is shown as follows:

Figure 2-1 System architecture



- **System bus**
Realize the connection of peripheral bus and bus matrix of M0+ microprocessor.
- **AHB TO APB Bridge 1/2**
Provides a fully synchronized connection from the AHB bus to the APB1/APB2 bus, that is, the bridge between the AHB and APB buses.

2.2 Memory organization

2.2.1 Overview

The CW32L010 core is a 32-bit ARM® Cortex®-M0+ microprocessor with a maximum addressing space of 4GB. The built-in program memory, data memory, peripherals and port registers of the chip are uniformly addressed in the same 4GB linear address space.

Bytes in memory are organized in little-endian format. The lowest byte data of a word storage space is the least significant bit of the word, and the highest byte data is the most significant bit. E.g:

Store 0x1122 3344 in the memory space at address 0x2000 0000, the actual result is:

0x20000000 bytes store 0x44,

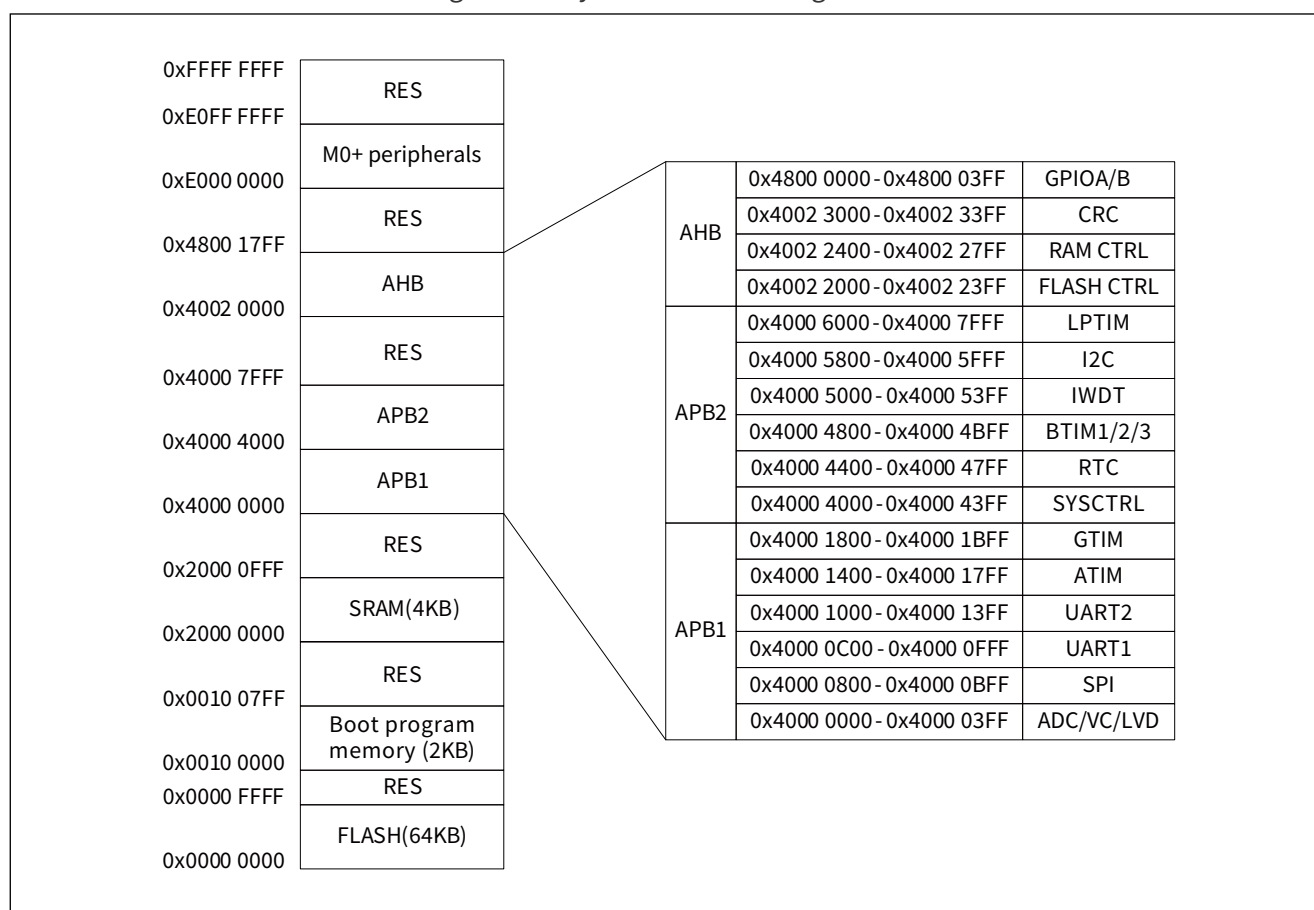
0x20000001 bytes store 0x33,

0x20000002 bytes store 0x22,

0x20000003 bytes store 0x11.

The system address allocation is shown in the figure below, RES is the reserved area.

Figure 2-2 System address assignment



2.2.2 Memory map and register boundary addresses

The detailed starting address space allocation of on-chip memory and peripherals is shown in the following table:

Table 2-1 Memory and Peripheral Address Allocation

Device or Bus	Boundary address	Size	Peripherals
Main FLASH memory	0x0000 0000 - 0x0000 FFFF	64KB	Main FLASH
Boot program memory	0x0010 0000 - 0x0010 07FF	2KB	BootLoader
OTP memory	0x0010 0760 - 0x0010 0775	22B	OTP
SRAM memory	0x2000 0000 - 0x2000 0FFF	4KB	SRAM
APB1 Peripheral	0x4000 0000 - 0x4000 03FF	1KB	ADC/VC/LVD
	0x4000 0800 - 0x4000 0BFF	1KB	SPI
	0x4000 0C00 - 0x4000 0FFF	1KB	UART1
	0x4000 1000 - 0x4000 13FF	1KB	UART2
	0x4000 1400 - 0x4000 17FF	1KB	ATIM
	0x4000 1800 - 0x4000 1BFF	1KB	GTIM
APB2 Peripheral	0x4000 4000 - 0x4000 43FF	1KB	SYSCTRL
	0x4000 4400 - 0x4000 47FF	1KB	RTC
	0x4000 4800 - 0x4000 4BFF	1KB	BTIM123
	0x4000 5000 - 0x4000 53FF	1KB	IWDT
	0x4000 5800 - 0x4000 5FFF	1KB	I2C
	0x4000 6000 - 0x4000 7FFF	1KB	LPTIM
AHB Peripheral	0x4002 2000 - 0x4002 23FF	1KB	FLASH CTRL
	0x4002 2400 - 0x4002 27FF	1KB	RAM CTRL
	0x4002 3000 - 0x4002 33FF	1KB	CRC
	0x4800 0000 - 0x4800 03FF	1KB	GPIOA/B
M0+ Peripheral	0xE000 0000 - 0xE00F FFFF	1MB	M0+ Core Peripheral



2.3 On-chip SRAM memory

CW32L010 integrates 4KB on-chip SRAM, and the starting address is 0x2000 0000. SRAM supports access in 3 widths of byte (8bit), half word (16bit) or full word (32bit), and can be accessed by CPU at the maximum system clock frequency with zero latency.

- Parity check

The SRAM supports the parity check function, which is enabled by default after the chip is powered on, and cannot be configured by the user.

The data bus of SRAM is 36bit, including 32bit of data bits and 4bit of parity bits (each 8bit of data bits is equipped with 1bit of parity bits), which is used to increase the robustness of the memory.

The parity bits are computed and stored when the CPU writes to the SRAM, and they are automatically checked when reading. After the system detects that the parity check fails, it will generate the corresponding interrupt flag.

For a detailed introduction to SRAM, see chapter [6 RAM Memory](#).



2.4 On-chip FLASH memory

The on-chip FLASH memory consists of two physical areas: the main FLASH memory and the boot program memory.

- Main FLASH memory, a total of 64KB, the address space is 0x0000 0000 - 0x0000 FFFF. This area is mainly used to store application code and user data, which is programmable by the user.
- Boot program memory, a total of 2KB, the address space is 0x0010 0000 - 0x0010 07FF. This area is mainly used to store the BootLoader startup program, which has been programmed when the chip leaves the factory and cannot be changed by the user.

The FLASH controller implements various operations on FLASH (erase, write, read).

FLASH supports access with 3 widths of byte (8bit), half word (16bit) or full word (32bit), and the highest frequency of access is 24MHz. If the HCLK clock frequency of the system configuration is higher than 24MHz, a reasonable response waiting time must be configured through the WAIT bit field of the FLASH control register FLASH_CR2 to ensure that the FLASH is correctly accessed.

For a detailed introduction to FLASH, see chapter [7 FLASH Memory](#).



2.5 One-time programmable OTP memory

There is 22B memory space inside the chip as the OTP area, and the address space is 0x0010 0760 – 0x0010 0775. This memory area can be read and written in the same way as the FLASH area, but it can only be written once, and then it can only be read, not erased or written. This area is mainly used to store unchangeable information, such as user-defined product UID, algorithm key, etc., which are written when the device leaves the factory.



2.6 System boot configuration

The device supports the following 2 different boot modes:

- Boot from main FLASH memory, run user program;
- Boot from the boot program memory, and fix the BootLoader program of the running chip. At this time, the user can use the ISP communication protocol to perform FLASH programming through the UART1 interface (PA07/PA08).

The steps for CW32L010 to enter ISP mode are as follows:

Step 1: Set the chip in RESET state;

Step 2: Provide a 50kHz square wave to UART1_RXD (SWDIO) of the chip;

Step 3: Release the chip's RESET state and delay for 5ms;

Step 4: The chip enters ISP mode.

In ISP mode, user CODE code can be written, and the MCU needs to be reset to execute the user application after completing the download.

After the system boot is complete, the CPU obtains the address of the top of the stack from the 0x0000 0000 address of the memory and starts executing the code from the address indicated by 0x0000 0004 of the memory.

The BootLoader program is located in the bootloader memory area and is programmed by the device provider at the time of manufacture. The user can program the FLASH via UART1 (pin PA07/PA08) using the ISP communication protocol.



2.7 Precautions

CW32L010 needs to pay attention to the following matters when using:

- A few registers such as FLASH, SRAM and GPIOx_ODR, CRC_DR support 8bit/16bit/32bit access mode, and other peripherals only support 32bit access mode. 8bit/16bit access to an address that does not support 8bit/16bit access mode will generate a HardFault hardware error exception.
- The RES areas in the system address allocation in [Figure 2-2 System address assignment](#) are reserved address spaces without corresponding physical devices. If these reserved address spaces are accessed, a HardFault hardware error exception will occur.
- After the chip is reset, all peripheral clocks except SYSTICK and SRAM are turned off. Before using these peripherals, the user must pass the peripheral clock enable control register (AHB peripheral clock enable control register SYSCtrl_AHBEN, APB peripheral clock Enable control register SYSCtrl_APBEN1/2) to open the configuration clock and working clock of the corresponding peripheral. Failure to turn on the peripheral configuration clock will result in an access failure: writing data fails, and no HardFault hardware error exception will be generated, and the result of reading data is not available. If the operating clock of the peripheral is not turned on, the main function of the peripheral is invalid.



3 Power Control (PWR) and Power Consumption

3.1 Overview

The CW32L010 needs one set of power supply when working: VDD, VSS.

CW32L010 embeds a 1.5V low dropout LDO voltage regulator to supply power to the digital power domain inside the chip.

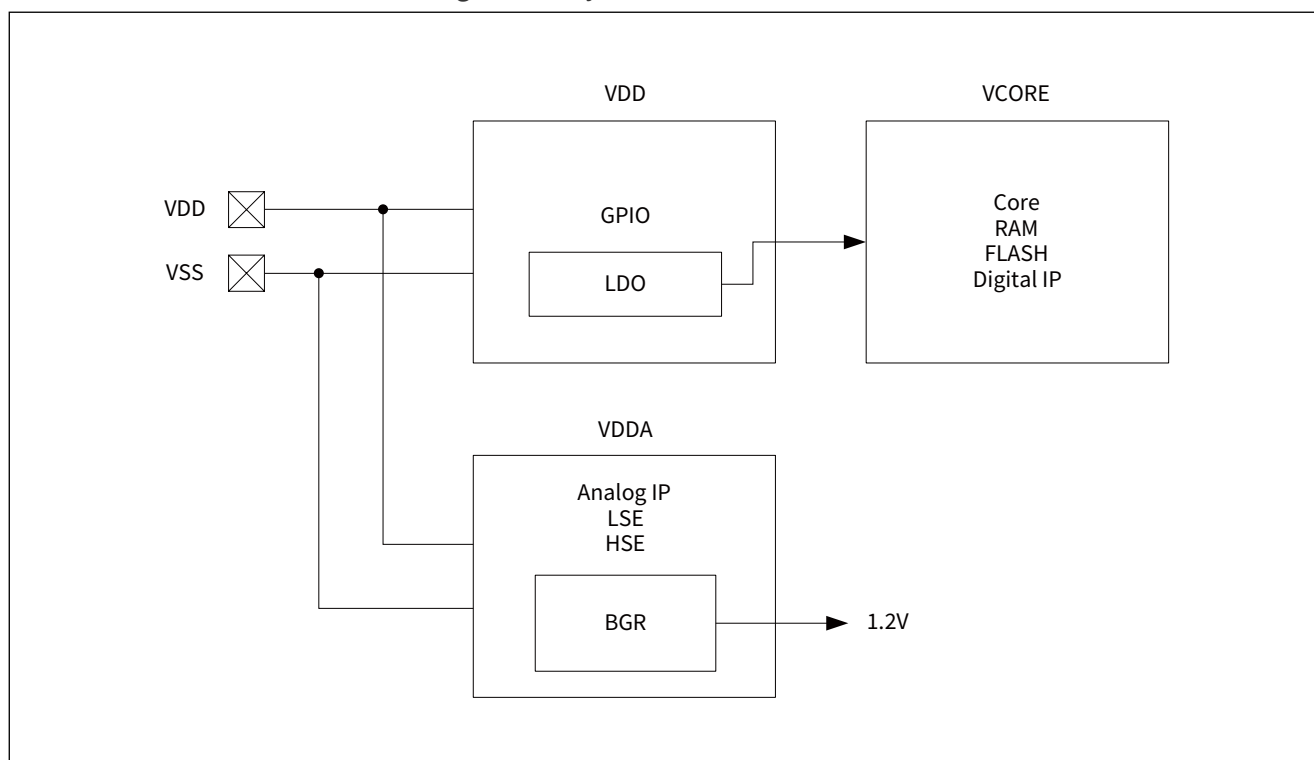
The chip has a built-in reference voltage generator (BGR) circuit, which can provide reference voltage for other analog modules.

Caution:

The difference between the supply voltage and the reference voltage must be higher than 0.3V. Please refer to the relevant section of the datasheet for detailed parameters of the voltage reference.

The division and allocation of power domains within the chip are shown in the following figure:

Figure 3-1 System Power Distribution



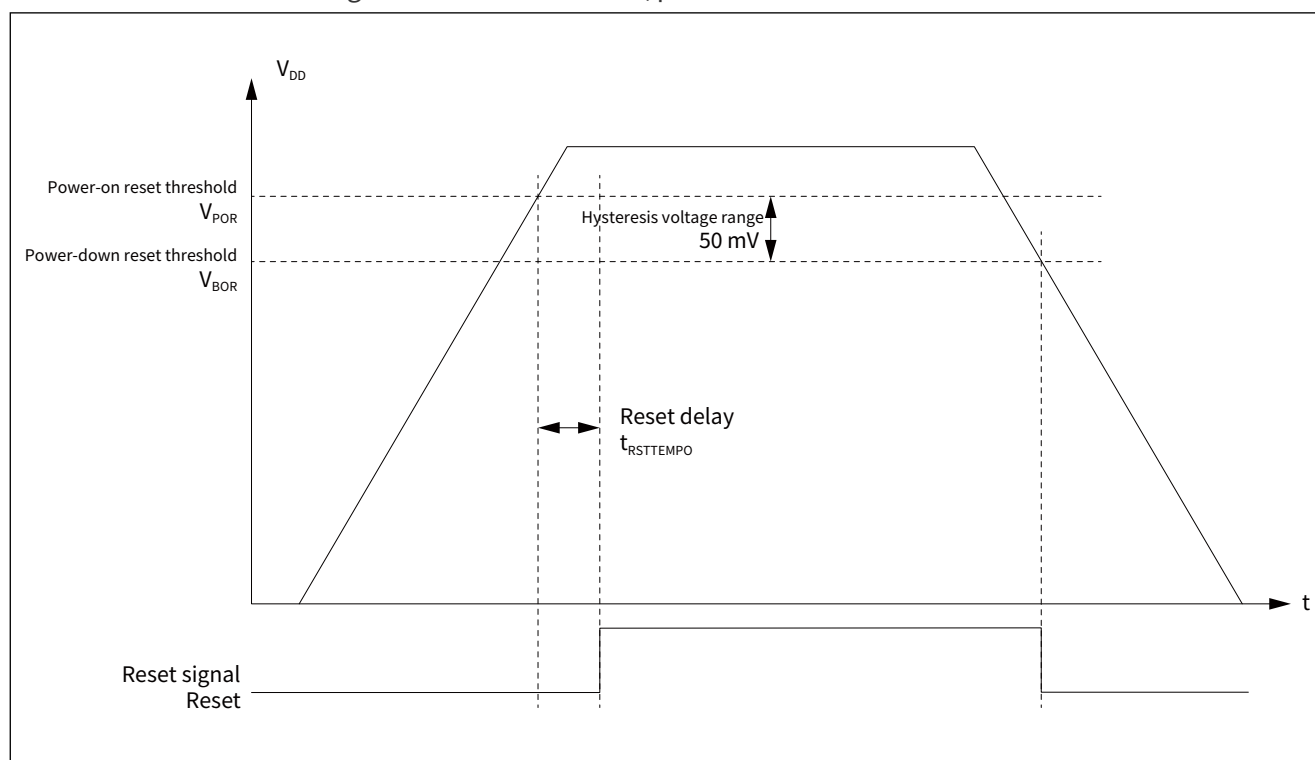
3.2 Power supply supervisor

Power on reset (POR)/ Brown-out reset (BOR)

CW32L010 integrates power-on reset (POR) and Brown-out reset (BOR) power monitoring circuits, and is always in working state after power-on. POR/BOR monitors the VDD power supply voltages. When the monitored power supply voltage is lower than the reset threshold (V_{BOR}), the system will enter the reset state. Users do not need to add an external hardware reset circuit.

The reset signal waveforms of the power-on and power-down phases are shown in the following figure:

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



For more details on the power on / power down reset threshold, refer to the electrical characteristics section in the datasheet.

3.3 Operating mode

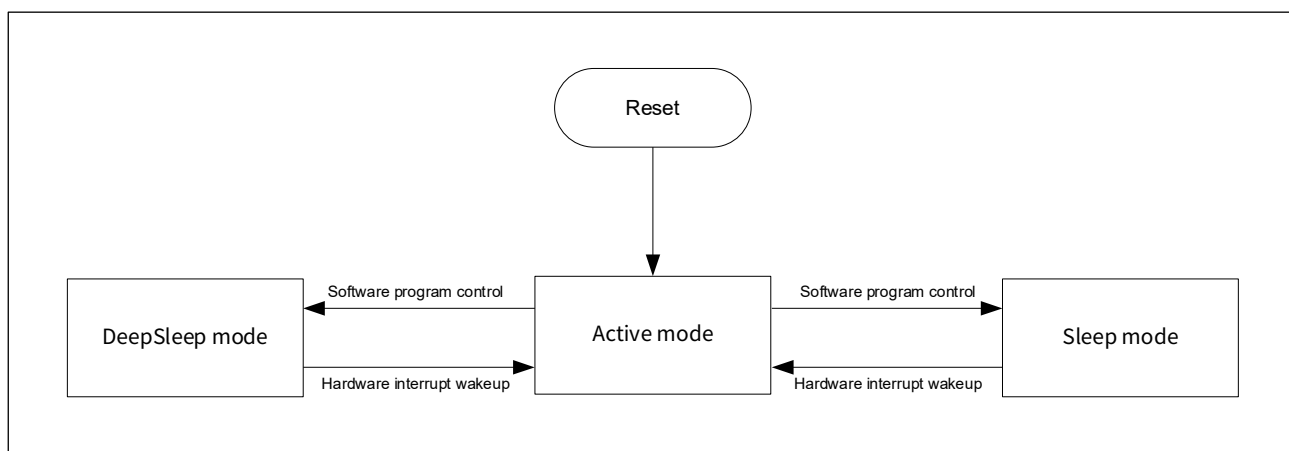
CW32L010 supports three operating modes, and the integrated power management module of the built-in power supply automatically completes the unified management. The three working modes are:

- Active mode
- Sleep mode
- DeepSleep mode

After the power is turned on, the system automatically enters the active mode. Users can enter two low-power operating states of sleep or deep sleep through software programs; In the low-power operating state, the wake-up mechanism can be triggered by a hardware interrupt to return the system to the active mode.

The conversion mechanism of the three operating modes is shown in the following figure:

Figure 3-3 The conversion mechanism of the operating modes



In the active mode, the CPU runs normally, and all module users can use it normally. In Sleep mode, the CPU stops running, all peripherals are unaffected, and all I/O pins remain in the same state. In DeepSleep mode, the CPU stops running, the high-speed clock (HSE, HSIO SC) is automatically turned off, and the low-speed clock (LSE, LSI) remains in the original state.

The CPU and clock states in different operating modes are shown in the following table:

Table 3-1 Operating mode and CPU and Clock States

Operating mode	CPU state	High-speed clock	Low-speed clock
Active mode	Run	ON/OFF	ON/OFF
Sleep mode	Stop	ON/OFF	ON/OFF
DeepSleep mode	Stop	OFF	ON

3.3.1 Entering Sleep mode or DeepSleep mode

Using the M0+ core's ARM waits for interrupt-specific instruction, WFI (Wait for Interrupt), in conjunction with the SLEEPONEXIT and SLEEPDEEP bit fields of the M0+ core's system control register (SCR, System Control Register), Sleep mode or DeepSleep mode can be entered or exited immediately (interrupt service routine).

- Entering immediately
Executing the WFI instruction, the MCU will immediately enter Sleep mode (when SLEEPDEEP is 0) or DeepSleep mode (when SLEEPDEEP is 1)
- Entering on exit
Setting the SLEEPONEXIT bit to 1, after exiting the lowest priority interrupt service routine, the MCU will enter Sleep mode (when SLEEPDEEP is 0) or DeepSleep mode (when SLEEPDEEP is 1), needn't execute the WFI instruction.

Table 3-2 Entering Sleep mode or DeepSleep mode

Entry method	SLEEPONEXIT bit	Condition	SLEEPDEEP bit	Enter mode
Entering immediately	-	Executing the WFI instruction	0	Sleep mode
			1	DeepSleep mode
Entering on exit	1	After exiting the lowest priority interrupt service routine	0	Sleep mode
			1	DeepSleep mode

Caution :

Before entering DeepSleep mode, if the FLASH is performing an erase/write operation, you must wait for the FLASH_CR1.BUSY flag bit to clear to 0, and make sure FLASH_CR1.MODE is 0 at the same time.

In DeepSleep mode, the system will automatically turn off the high-speed clock. If users need to keep some peripherals running in DeepSleep mode, they must start the corresponding low-speed clock and set the peripheral clock to this low-speed clock before entering DeepSleep mode.



3.3.2 Exiting Sleep mode or DeepSleep mode

In Sleep mode or DeepSleep mode, the interrupt can be used to wake up the CPU and return to active mode. However, it is worth noting that if the user executes the WFI command in the interrupt service routine to enter Sleep (including DeepSleep), an interrupt with a higher priority than this interrupt is required to wake up the CPU, so we strongly recommend that the user is ready to enter sleep, all interrupt service routines should be processed and all interrupt requests and interrupt flags should be cleared.

In different operating modes, the types of interrupts that the CPU can respond to are shown in the following table:

Table 3-3 Operating Modes and Interrupt Sources

Interrupt Number	Interrupt Sources	Active mode	Sleep mode	DeepSleep mode
0	IWDT	Y	Y	Y
1	LVD	Y	Y	Y
2	RTC	Y	Y	Y
3	FLASH	Y	Y	N
	RAM	Y	Y	N
4	RCC	Y	Y	Y
5	GPIOA	Y	Y	Y
6	GPIOB	Y	Y	Y
12	ADC	Y	Y	N
13	ATIM	Y	Y	N
14	VC1	Y	Y	Y
15	VC2	Y	Y	Y
16	GTIM	Y	Y	N
19	LPTIM	Y	Y	Y
20	BTIM1	Y	Y	N
21	BTIM2	Y	Y	N
22	BTIM3	Y	Y	N
23	I2C	Y	Y	N
25	SPI	Y	Y	N
27	UART1	Y	Y	Y
28	UART2	Y	Y	Y
31	FAULT	Y	Y	Y

Exiting Sleep mode using an interrupt, the user must enable the enable bit for this interrupt before entering Sleep (including DeepSleep mode).

After the interrupt wakes up and exits Sleep mode, the CPU will immediately enter the interrupt service routine



for this interrupt. If the user does not set this interrupt service routine, and when entering sleep immediately: the CPU will continue to execute the next statement of the WFI instruction that entered sleep; when entering sleep when exiting: continue to execute the next statement of the last entered interrupt service routine. In general, based on system reliability considerations, it is strongly recommended that the user set the service routine for this interrupt, and clear the interrupt request and interrupt flag in the interrupt service routine.

When the interrupt wakes up to exit the DeepSleep mode, the CPU operation state is the same as the exit from the Sleep mode.

In DeepSleep mode, the system will automatically turn off the high-speed clock. When exiting DeepSleep, CW32L010 adds an additional system clock option for users. Users can choose to continue to use the clock used when entering DeepSleep, or select HSI 4MHz as the system clock. Configure the WAKEUPCLK bit field of the system control register SYSCTRL_CR2 to 1, then the internal high-speed clock HSI 4MHz is automatically used as the system clock after the interrupt wakes up and exits the DeepSleep mode, it can speed up the system wake-up.



3.3.3 Operating Modes and Reset Sources

Even in Sleep mode or DeepSleep mode, the CPU can respond to some reset sources. The hardware reset or software reset that the CPU can respond to in different working modes is shown in the following table:

Table 3-4 Operating Modes and Reset Sources

Reset Sources	Active mode	Sleep mode	DeepSleep mode
Power-on reset/Brown-out reset (POR/BOR)	Y	Y	Y
Pin input reset (NRST)	Y	Y	Y
LVD reset	Y	Y	Y
IWDT reset	Y	Y	Y
WWDT reset	Y	Y	N
Kernel LOCKUP fault reset	Y	N	N
Kernel SYSRESETREQ reset	Y	N	N



3.4 Low-power consumption applied

In Sleep mode, the CPU stops running, and all peripherals keep running, including ARM®Cortex®-M0+ core peripherals, such as NVIC, SysTick and other peripherals. Sleep mode consumes less power than active mode.

In DeepSleep mode, the CPU stops running, the high-speed clock is turned off, the low-speed clock remains unchanged, some peripherals can be configured to continue to run, and the NVIC interrupt processing still works. DeepSleep mode consumes much less power than Sleep mode.

Users can reduce system running power consumption by:

Decrease system clock frequency

- Use low-frequency high-speed clock HSI, HSE or low-speed clock LSI, LSE.
- Reduce the frequency of SYSCLK, HCLK, PCLK by programming the prescaler registers:
 - Set the SYSCLK bit field of the SYSTCTRL_CR0 register to select the appropriate clock source.
 - Set the HCLKPRS bit field of the SYSTCTRL_CR0 register to reduce the HCLK frequency.
 - Set the PCLKPRS bit field of the SYSTCTRL_CR0 register to reduce the PCLK frequency.

Turn off clocks and peripherals which not in use during sleep

- AHB bus clock HCLK and APB bus clock PCLK, can be turned off as needed.
- Turn off clocks to peripherals not related to wakeup:
 - AHB peripheral clock enable control register, SYSTCTRL_AHBEN.
 - APB peripheral clock enable control register 1, SYSTCTRL_AHBEN1.
 - APB peripheral clock enable control register 2, SYSTCTRL_AHBEN2.



3.5 Cortex®-M0+ kernel system control register (SCB->SCR)

Address: 0xE000 ED10 Reset value: 0x0000 0000

Bit filed	Name	Permission	Function description
31:5	RFU	-	Reserved bits, please keep the default value
4	SEVONPEND	RW	When set to 1, each new suspension of an interrupt generates an event, which can be used to wake up the processor if WFE sleep is used.
3	RFU	-	Reserved bits, please keep the default value
2	SLEEPDEEP	RW	When set to 1, enter DeepSleep mode when WFI or SLEEPONEXIT is 1 and all interrupt service routines are exited. When set to 0, enter Sleep mode when WFI or SLEEPONEXIT is 1 and all interrupt service routines are exited.
1	SLEEPONEXIT	RW	When set to 1, the processor automatically enters Sleep mode (or DeepSleep mode) when all interrupt service routines are exited. When set to 0, the hibernation function on exit is disabled.
0	RFU	-	Reserved bits, please keep the default value



4 Reset and clock control (RCC)

4.1 System reset

CW32L010 supports the following 6 system resets:

- Power on reset (POR)/ Brown-out reset (BOR)
- Pin input reset (NRST)
- IWDT reset
- LVD reset
- Kernel SYSRESETREQ reset
- Kernel LOCKUP fault reset

The reset range of each system reset is shown in the following table:

Table 4-1 Reset method and range

Reset method	Reset range
Power on reset (POR)/ Brown-out reset (BOR)	Entire MCU
Pin input reset (NRST)	Entire MCU (Except RTC)
IWDT reset	M0+ core/peripheral (Except RAM controller/RTC)
LVD reset	M0+ core/peripheral (Except LVD controlle section/RTC)
Kernel SYSRESETREQ reset	M0+ core (Except SWD debug logic) / Peripheral (Except RAM controller/LVD/RTC)
Kernel LOCKUP fault reset	M0+ core/peripheral (Except RAM controller/LVD/RTC)

After the system reset occurs, the CPU runs again, most of the registers are reset to default values, and the program starts to execute from the reset interrupt entry address of the interrupt vector table.

The user can query the reset source of this system reset through the system reset flag register SYSCTRL_RESETFLAG. The reset flag is set by hardware and cleared by software. It is recommended that users clear the relevant flag bits of this register to 0 after reading the flag bits to avoid confusion after the next reset.

4.1.1 Power on reset/Brown-out reset (POR/BOR)

The CW32L010 integrates special POR and BOR circuits to monitor the power supply voltage, and keeps the chip in the reset state when the power supply voltage is lower than the safe range to prevent the chip from malfunctioning during power-on/power-off. In order to ensure the stable operation of the system, the user must keep the power supply voltage within a safe range.

For the specific functions of POR and BOR, please refer to chapter [3 Power Control \(PWR\) and Power Consumption](#).



4.1.2 Pin input reset (NRST)

CW32L010 has a special reset input pin, inputting a low-level signal of a certain width will cause the system to reset. A dedicated anti-shake circuit is designed inside the chip, and low-level pulse signals shorter than 20μs will be shielded.

The reset input pin has a built-in pull-up resistor. If the user needs to connect an external RC circuit, the influence of the internal pull-up resistor must be considered. For the circuit characteristics of the built-in pull-up resistor, please refer to the relevant section of the datasheet.

4.1.3 IWDG reset

The CW32L010 integrates an independent watchdog (IWDG). When the IWDG or WWDG meets the reset condition, a reset signal will be generated to cause a system reset.

For detailed functions of IWDG reset, please refer to chapter [15 Independent watchdog timer \(IWDG\)](#).

4.1.4 LVD low voltage detection reset

The CW32L010 integrates a low voltage detector (LVD), which can continuously compare the selected monitoring voltage with the set LVD threshold voltage. When the comparison result meets the trigger condition, an LVD reset signal will be generated to cause a system reset.

Compared with the POR/BOR reset method, the LVD low voltage detection reset function is more powerful. The reset threshold voltage, monitoring voltage source, pulse filter width and hysteresis time can be set through the relevant registers.

See chapter [22 Low voltage detector \(LVD\)](#) for LVD details.

4.1.5 Kernel SYSRESETREQ reset

The core SYSRESETREQ reset is a software reset, which is implemented by setting the SYSRESETREQ bit field of the Application Interrupt and Reset Control Status Register (AIRCR) of the ARM® Cortex®-M0+. Setting this bit to 1 by the application program will generate a core SYSRESETREQ reset, thereby implementing a software reset.

4.1.6 Kernel LOCKUP fault reset

When the CPU encounters a serious exception (such as the read instruction is invalid, the bit width does not match the target address when accessing FLASH), it will stop the PC pointer at the current address and lock it, and generate the kernel LOCKUP fault reset signal.

After the chip is powered on, the LOCKUP reset function is disabled by default, and the user needs to manually enable it. By setting the LOCKUP bit field of the system control register SYSCTRL_CR2 to 1, the LOCKUP reset function can be enabled.



4.2 Peripheral reset

Users can use software to reset various peripherals inside CW32L010 individually. Peripheral reset can restore the registers, state machines and various control logic of each peripheral to the default state after power-on reset. The user can perform the independent reset operation of each peripheral module by setting the three registers: `SYSCTRL_AHBRST`, `SYSCTRL_APB1RST1`, and `SYSCTRL_APB1RST2`.

For power-on reset/brown-out reset (POR/BOR), the internal peripheral modules are already in the default state after reset, and the module can be initialized and configured directly, and there is no need to independently reset each module.

For other types of system resets, some peripherals may retain the working state before the reset. If the user needs to restore the peripherals to the default state of power-on, they should reset the peripherals through the corresponding reset registers before using them. The RTC module is a typical peripheral that may need to retain the current state after reset. For specific judgment logic examples, please refer to the relevant routines.



4.3 Clock and Control

4.3.1 Overview

The CW32L010 has a built-in multi-channel clock generation circuit, which generates various clocks of different frequencies through the frequency divider and multiplexer for use by the CPU and various peripherals. The schematic diagram of the system clock tree structure is shown in the [Figure 4-1 System internal clock tree](#).

The internal clock SysClk of the system is divided to provide the advanced high-performance bus clock HCLK for the CPU core, and the HCLK clock is divided to provide the advanced peripheral clock PCLK for digital and analog peripherals.

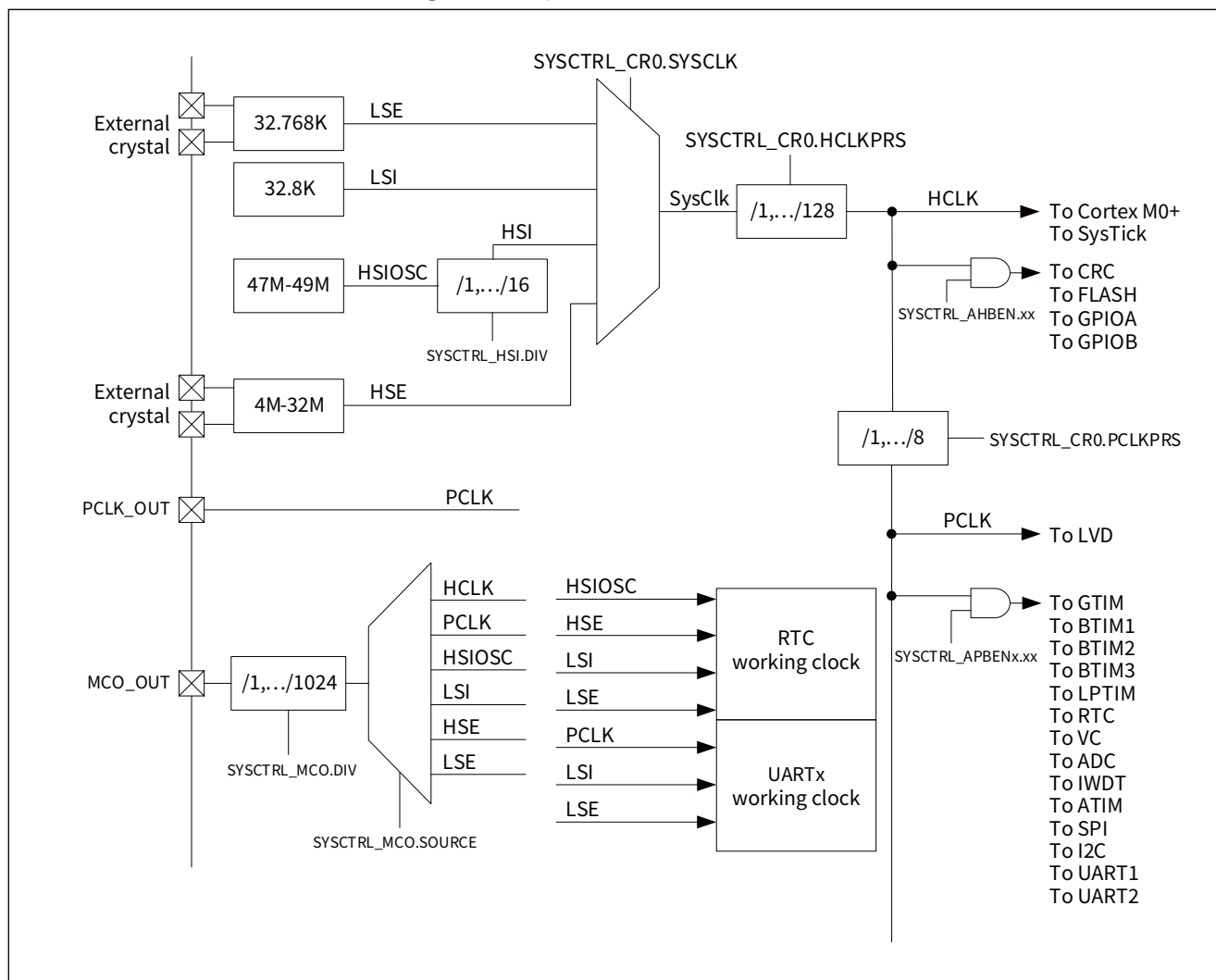
System clock SYSCLK has 4 clock sources:

- External high speed oscillator clock (HSE)
- Low-speed external oscillator clock (LSE)
- The HSI clock is generated by dividing the internal high-speed RC oscillator clock (HSIOSC)
- Low-speed internal RC oscillator clock (LSI)

The system internal clock tree is shown in the following figure:



Figure 4-1 System internal clock tree



The high-speed internal RC oscillator clock HSIOSC is divided by the frequency divider to generate the HSI clock. The frequency division coefficient is set through the DIV bit field of the built-in high-frequency clock control register SYSCTRL_HSI, which supports any frequency division from 1 to 16.

The system clock SysClk can be selected from 4 clock sources: HSE, LSE, HSI, LSI, which can be selected by the SYSCLK bit field of the system control register SYSCTRL_CR0.

Advanced high-performance bus clock HCLK is generated by the frequency division of the system clock SysClk, and is used as the configuration clock and working clock of the M0+ core, SysTick, FLASH, CRC, GPIO and other modules. The frequency division factor is set by the HCLKPRS bit field of the system control register SYSCTRL_CR0, and the effective frequency division factor is 2^n ($n = 0 \sim 7$).

Advanced peripheral clock PCLK, which is generated by frequency division of HCLK, is used as the configuration clock and working clock of peripherals such as timer, SPI, and I2C. The frequency division factor is set by the PCLKPRS bit field of the system control register SYSCTRL_CR0, and the effective frequency division factor is 2^n ($n = 0 \sim 3$).

4.3.2 System Clock and Operating Mode

CW32L010 supports three operating modes: Active mode, Sleep mode and DeepSleep mode.

In the Active mode, the CPU runs normally, and all module users can use it normally.

In the Sleep mode, the CPU stops running, and the clock oscillators and peripherals remain unchanged.

In DeepSleep mode, the CPU stops running, and the high-speed clock HSE, HSIOSC oscillator are automatically turned off to save power; The oscillators of the low-speed clocks LSE, LSI remain unchanged; Whether the system clock SysClk and the HCLK and PCLK clocks are valid depends on the clock source status of the SysClk system clock.

After waking up from DeepSleep mode, if the WAKEUPCLK bit field of the system control register SYSCTRL_CR2 is configured to 1, the system will automatically use the HSI 4MHz as the clock source of the system clock, and the original system clock source remains enabled. If SYSCTRL_CR2.WAKEUPCLK is 0, the system will wait for the system clock source used before the system enters DeepSleep to stabilize before starting to run. Since the HSI starts up quickly and can quickly respond to user needs, it is recommended to switch the clock source of the system clock to the HSI clock or configure SYSCTRL_CR2.WAKEUPCLK to 1 before entering DeepSleep mode.



4.3.3 HSE clock

The high-speed external clock signal (HSE) supports two operating modes:

- For crystal/ceramic resonators, set the MODE bit field of the external high-frequency crystal control register SYSCTRL_HSE to 0, and configure the OSC_IN and OSC_OUT pins as analog functions.
- For external clock input, set the MODE bit field of the external high-frequency crystal control register SYSCTRL_HSE to 1, and configure the OSC_IN pin as a digital input function.

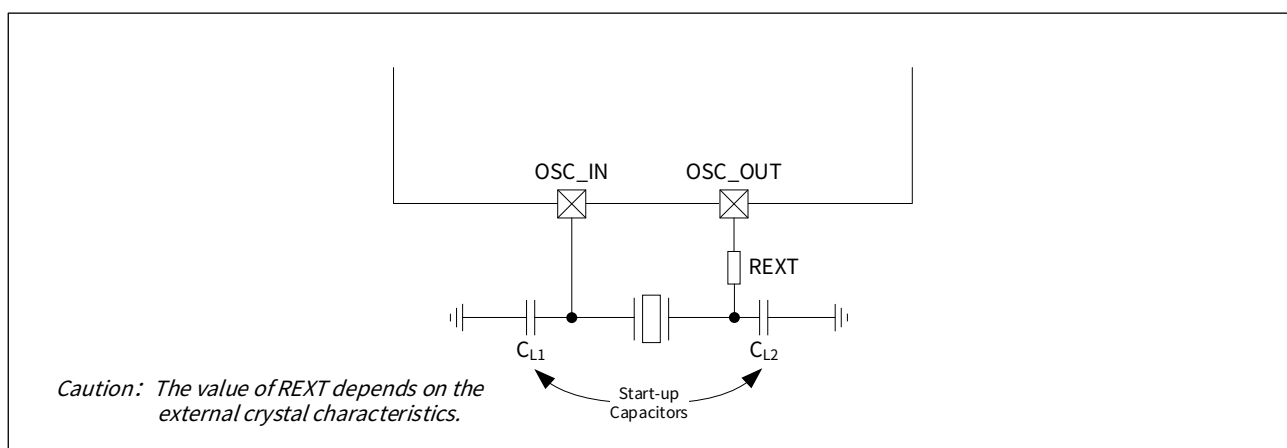
Table 4-2 HSE operating mode configuration

Work mode	SYSCTRL_HSE.MODE	OSC_IN	OSC_OUT
Crystal/Ceramic resonators	0	Analog	Analog
External clock input	1	Digital input	Can be used as general purpose GPIO

Crystal/Ceramic resonators clock mode

The external quartz crystal/ceramic resonator is connected between the OSC_IN and OSC_OUT pins, and cooperates with the start-up capacitor to generate a stable 4~32MHz clock signal. The circuit connection is shown in the following figure:

Figure 4-2 HSE external quartz crystal/ceramic resonator connection method



The crystal/ceramic resonator and start-up capacitor must be placed as close as possible to the chip oscillator pins to minimize oscillator output distortion and start-up time.

The value of the start-up capacitor must be adjusted according to the selected crystal/ceramic resonator. The calculation method is:

$$C_L = (C_{L1} * C_{L2}) / (C_{L1} + C_{L2}) + C_0$$

Among them,

C_L is the nominal load capacitance of the crystal;

C_0 is the stray capacitance value, which is related to the design of the circuit board. Generally, the value can be 4pF ~ 6pF;

C_{L1} and C_{L2} are two external start-up capacitors, usually C_{L1} and C_{L2} are same.

If the user uses a crystal resonator with a nominal load capacitance value of 12.5pF, the common value of the two external starting capacitors for the crystal is 15pF.

Caution: Since there are many crystal manufacturers, the crystal characteristics of each manufacturer are different. If the output clock signal of the HSE oscillator is not ideal, you can contact the crystal supplier for coordination.

Before HSE clock is stable, users can adjust driving capability by setting PDRIVER bit field of external high-frequency crystal control register SYSCTRL_HSE. Eight driving capabilities can be selected. The stronger the driving capability, the greater the power consumption. After HSE clock is stable, users can adjust driving capability by setting DRIVER bit field of external high-frequency crystal control register SYSCTRL_HSE. Eight driving capabilities can be selected. The stronger the driving capability, the greater the power consumption. Users can balance clock reliability, start-up time, and low power consumption according to their needs.

External clock input mode

In the external clock input mode, the external clock is input from the OSN_IN pin, and the OSC_OUT pin can be used as a general-purpose GPIO. The input clock signal can be square wave, sine wave or triangular wave, the duty cycle must be between 40%~60%, and the frequency is between 1~32MHz.

When an external oscillator is used, the PA05 pin (which needs to be configured for HEXEN multiplexed output function) outputs an enable signal that outputs different levels in chip run mode and DeepSleep mode, respectively. The output polarity is configured through the HEXENPOL bit field of the SYSCTRL_HSE register.

In the two working modes of HSE, the HSE clock oscillator is turned off by default after power-on, and is enabled by setting the HSEEN bit field of the system control register SYSCTRL_CR1 to 1.

After the HSE oscillator is started, when the internal clock monitoring module of the chip detects a certain number of HSE clock signals, the HSE clock is considered to be stable. The number of detection clocks can be set through the WAITCYCLE bit field of the external high-frequency crystal control register SYSCTRL_HSE, as shown in the following table:

Table 4-3 HSE Stability Detection Clock Signal Number

SYSCTRL_HSE.WAITCYCLE	Number of HSE detection clocks
00	8192
01	32768
10	131072 (Defaults)
11	262144

The start-up state of the HSE clock can be determined through the STABLE flag of the external high-frequency crystal control register SYSCTRL_HSE. If the STABLE flag is 1, the HSE clock is stable, and if it is 0, the HSE clock is not stable.

Caution:

All parameters of the HSE oscillator should be set before starting, and it is forbidden to modify the relevant parameters after starting.



4.3.4 HSIOSC clock

The HSIOSC clock is generated by the internal RC oscillator, no external circuit is required, the startup speed is fast and the frequency is fixed at 48MHz.

The frequency of the RC oscillator output clock is affected by the chip processing process, operating voltage, ambient temperature and other factors. CW32L010 provides the HSIOSC clock frequency calibration function. Users can calibrate the HSIOSC clock by setting the TRIM bit field value of the built-in high-frequency clock control register SYSCTRL_HSI frequency, see section [4.4.2 Clock calibration](#) for details.

The HSIOSC internal high-speed RC oscillator is turned on by default after the chip is powered on. Users can turn it off by setting the HSIEN bit field of SYSCTRL_CR1 in the system control register to 0. If the user stops and restarts the HSIOSC oscillator, the STABLE flag of the built-in high-frequency clock control register SYSCTRL_HSI can be used to determine whether the HSI clock is stable. If the STABLE flag is 1, it means that the HSIOSC clock is stable, and if it is 0, it means that the HSIOSC clock is not stable yet.

Caution:

All parameters of the HSIOSC oscillator should be set before starting, and it is forbidden to modify the relevant parameters after starting.

The HSIOSC clock outputs the HSI clock after frequency division. The frequency division coefficient is set by the DIV bit field of the built-in high-frequency clock control register SYSCTRL_HSI, which can be set to any frequency division from 1 to 16, and the default after power-on is 12. Therefore, the HSI Clock has a default frequency of 4MHz.



4.3.5 LSE clock

External low-speed clock (LSE) supports two operating modes:

- For crystal/ceramic resonators, set the MODE bit field of the external low-frequency crystal control register SYSCTRL_LSE to 0, and configure the OSC32_IN and OSC32_OUT pins as analog functions.
- For external clock input, set the MODE bit field of the external low-frequency crystal control register SYSCTRL_LSE to 1, and configure the OSC32_IN pin as a digital input function.

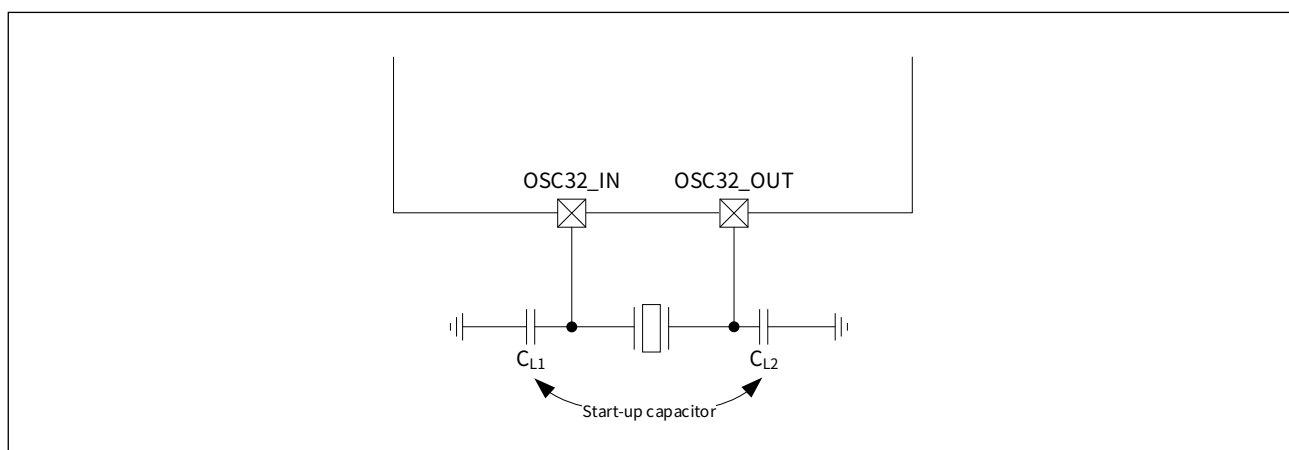
Table 4-4 LSE operating mode configuration

Operating mode	SYSCTRL_LSE.MODE	OSC32_IN	OSC32_OUT
Crystal/Ceramic Resonators	0	Analog	Analog
External clock input	1	Digital input	Can be used as general purpose GPIO

Crystal/Ceramic resonators clock mode

The external low-frequency quartz crystal/ceramic resonator is connected between the two pins of OSC32_IN and OSC32_OUT. With the start-up capacitor, a stable clock signal of up to 1MHz can be generated. LSE is usually used as the operating clock of the system real-time clock RTC. At this time, a quartz crystal/ceramic resonator with a frequency of 32.768kHz should be selected. The external circuit connection is shown in the figure below:

Figure 4-3 LSE external quartz crystal/ceramic resonator connection method



The quartz crystal/ceramic resonator and start-up capacitor must be placed as close as possible to the chip oscillator pins to minimize oscillator output distortion and start-up time.

The value of the start-up capacitor must be adjusted according to the selected crystal/ceramic resonator. The calculation method is:

$$C_L = (C_{L1} \times C_{L2}) / (C_{L1} + C_{L2}) + C_0$$

Among them,

C_L is the nominal load capacitance of the crystal;

C_0 is the stray capacitance value, which is related to the design of the circuit board. Generally, the value can be 4pF~6pF;

C_{L1} and C_{L2} are two external start-up capacitors, usually C_{L1} and C_{L2} are same.

If the user uses a crystal resonator with a nominal load capacitance value of 12.5pF, the common value of the two external starting capacitors for the crystal is 15pF.

Caution: Since there are many crystal manufacturers, the crystal characteristics of each manufacturer are different. If the output clock signal of the LSE oscillator is not ideal, you can contact the crystal supplier for coordination.

Before the LSE clock is stabilised, the user can adjust the drive capability by setting the PDRIVER bit field of the external low frequency crystal control register SYSCTRL_LSE, there are sixteen drive capabilities to choose from, the stronger the drive capability the higher the power consumption. After the LSE clock is stabilised, the user can adjust the drive capability by setting the DRIVER bit field of the external low-frequency crystal control register SYSCTRL_LSE. There are sixteen levels of drive capability to choose from. The stronger the drive capability, the higher the power consumption. Users can balance clock reliability, startup time, and low power consumption as needed.

External clock input mode

In the external clock input mode, the external clock is input from the OSN32_IN pin, and the OSC32_OUT pin can be used as a general-purpose GPIO. The input clock signal can be square wave, sine wave or triangular wave, the duty cycle must be between 45%~55%, and the frequency can be up to 100kHz.

In the two working modes of HSE, the LSE clock oscillator is turned off by default after power-on, and is enabled by setting the LSEEN bit field of the system control register SYSCTRL_CR1 to 1.

After the LSE oscillator is started, when the internal clock monitoring module of the chip detects a certain number of LSE clock signals, the LSE clock is considered to be stable. The number of detection clocks can be set through the WAITCYCLE bit field of the external low-frequency crystal control register SYSCTRL_LSE, as shown in the following table:

Table 4-5 LSE Stability Detection Clock Signal Number

SYSCTRL_LSE.WAITCYCLE	Number of LSE detection clocks
00	256
01	1024
10	4096 (Defaults)
11	16384

The start-up state of the LSE clock can be determined through the STABLE flag of the external low-frequency crystal control register SYSCTRL_LSE. If the STABLE flag is 1, the LSE clock is stable, and if it is 0, the LSE clock is not stable.

Caution: All parameters of the LSE oscillator should be set before starting, and it is forbidden to modify the relevant parameters after starting.

The LSE oscillator has a lock function. After the lock function is turned on, the software is prohibited from clearing the enable bit (SYSCTRL_CR1.LSEEN) of the LSE clock oscillator. That is, once the LSE is turned on, it cannot be turned off by software. The locking function is mainly designed to meet the requirements of some special application scenarios, such as application scenarios sensitive to RTC time loss, which can effectively prevent RTC time loss caused by misoperation. The lock function is disabled by default and is enabled by setting the LSELOCK bit field of the system control register SYSCTRL_CR1 to 1.



4.3.6 LSI clock

The LSI clock is generated by the internal low-speed RC oscillator with a default frequency of 32.8kHz. The internal low-speed RC oscillator requires no external circuitry and is less expensive than the LSE clock, but less accurate than the LSE clock.

The frequency of the RC oscillator output clock is affected by the chip processing process, operating voltage, ambient temperature and other factors. CW32L010 provides the LSI clock frequency calibration function. Users can calibrate the LSI clock frequency by setting the value of the TRIM bit field of the built-in low-frequency clock control register SYSCTRL_LSI. For details, please refer to [4.4.2 Clock calibration](#).

The LSI internal low-speed RC oscillator is disabled by default, and is enabled by setting the LSIEN bit field of the system control register SYSCTRL_CR1 to 1. After the LSI oscillator is started, if the internal clock monitoring module of the chip detects a certain number of LSI clock signals, the LSI clock is considered to be stable. The number of detection clocks can be set by the WAITCYCLE bit field of the built-in low-frequency oscillator control register SYSCTRL_LSI, as shown in the following table:

Table 4-6 LSI Stability Detection Clock Signal Number

SYSCTRL_LSI.WAITCYCLE	Number of LSI detection clocks
00	6 (Defaults)
01	18
10	66
11	258

The STABLE flag of the built-in low-frequency clock control register SYSCTRL_LSI can be used to determine whether the LSI clock is stable. If the STABLE flag is 1, it means that the LSI clock is stable, and if it is 0, it means that the LSI clock is not stable.

Caution:

All parameters of the LSI oscillator should be set before starting, and it is forbidden to modify the relevant parameters after starting.

4.3.7 SysClk system clock

System clock SysClk can choose from 4 clock sources, including HSE, LSE, HSI, and LSI.

After the system power-on reset is completed, HSI is selected as the clock source of SysClk by default, and the default value of the clock frequency is 4MHz.

The user can select the system clock source through the SYSCLK bit field of the system control register SYSCTRL_CR0. After selecting a clock as the system clock source, the user cannot stop the clock by setting the enable bit SYSCTRL_CR1.xxxEN of the clock circuit to 0.



4.3.8 On-chip peripheral clock control

On-chip peripherals generally need the configuration clock and the working clock. The configuration clock is used to respond to the read and write operations of the peripheral registers by the CPU. The working clock is used for the function realization of each peripheral (such as the transmission clock of the UART, the count clock of the timer, etc.).

The configuration clock and working clock of the peripheral must be turned on before using the peripheral, otherwise the peripheral cannot work. By setting the corresponding bits of the AHB peripheral clock enable control register `SYSCTRL_AHBEN` and the APB peripheral clock enable control register `SYSCTRL_APBEN1` and `SYSCTRL_APBEN2` to 1, the configuration clock and working clock of the corresponding peripherals are turned on.

Peripherals such as RTC, UART, and FLASH only open the configuration clock, and the working clock is configured through the clock source selection register of each module.

When the peripherals do not need to be used, the peripherals can be disabled by turning off the configuration clocks and operating clocks of the peripherals, which can effectively reduce the power consumption of the chip.

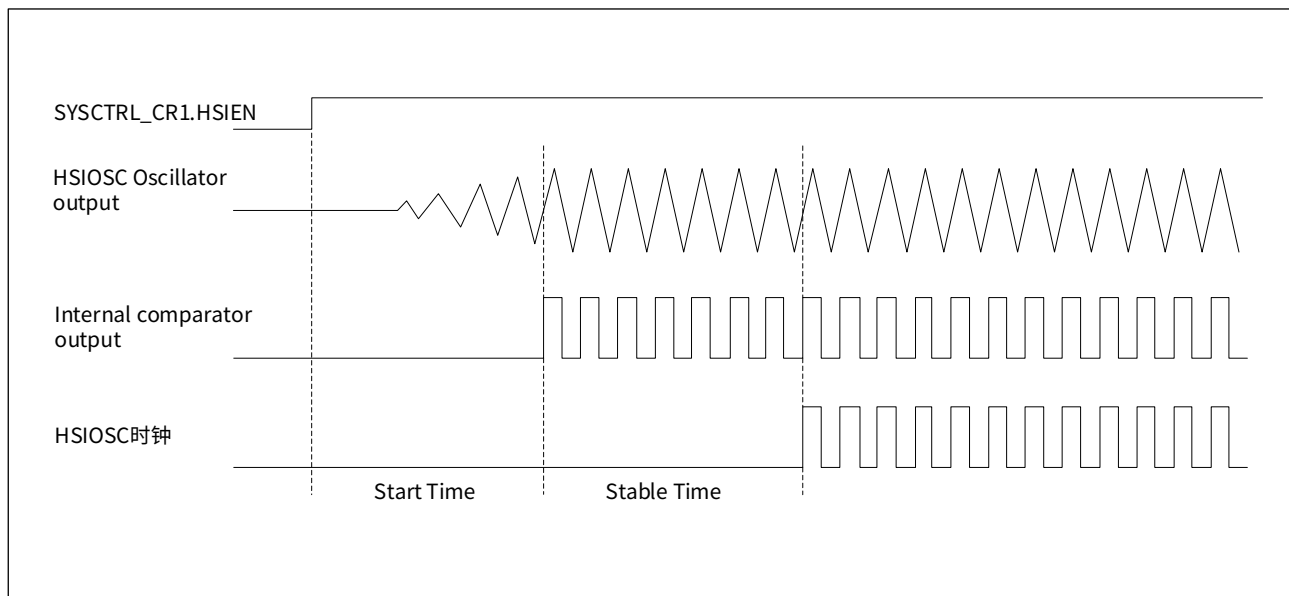


4.4 Clock Start, Calibration, and Status Detection

4.4.1 Clock start

The clock source startup process of CW32L010 is similar. Take HSIOSC as an example to illustrate the stabilization process after clock startup, as shown in the following figure:

Figure 4-4 HSIOSC clock startup process



When SYSCTRL_CR1.HSIEN is set to 1, the HSIOSC clock oscillation circuit starts to work, but the output clock signal amplitude is very small at this time. After the start-up time stage, the amplitude and duty cycle of the output clock signal can meet the needs of the internal sampling circuit and enter the stabilization time stage. In the stable time stage, the internal clock monitoring circuit of the chip counts the clock signals output by the HSIOSC. When the count value reaches the set number, the HSIOSC clock signal is considered to be stable, and the HSIOSC clock stability flag bit SYSCTRL_HSI.STABLE is set to 1.

The clock start-up process of other clock oscillators is similar, but note that the number of detection clocks in the stable phase, the detection time and the number of detection clocks for start-up failure detection are different for each clock oscillator. For details, see the description of each clock oscillator.

4.4.2 Clock calibration

Clock calibration is mainly for HSIOSC clock and LSI clock. Clock frequency calibration is achieved by adjusting the TRIM value of the oscillator.

HSIOSC clock calibration

The safe working range of HSIOSC is 47~49MHz. If it exceeds the safe range, the chip may be abnormal. When the chip leaves the factory, the calibration parameters of the 48MHz frequency point have been pre-adjusted and stored in FLASH. The application program only needs to read and write the calibration value in FLASH to SYSCTRL_HSI.TRIM to obtain a precise 48MHz clock. The storage address of the 48MHz frequency calibration value is: 0x0010 07C0 – 0x0010 07C1. If other frequency clocks are needed, users need to adjust the value of SYSCTRL_HSI.TRIM by themselves.

LSI clock calibration

The LSI output clock frequency range is $32.8\text{kHz} \pm 10\%$. If the LSI output clock frequency is adjusted outside this range, the clock may be abnormal. When the chip leaves the factory, the calibration parameters of the 32.8kHz frequency point are pre-adjusted and stored in the FLASH. The application program only needs to read the calibration value in the FLASH and write it into SYSCTRL_LSI.TRIM to obtain an accurate 32.8kHz clock. 32.8kHz frequency calibration value storage address: 0x0010 07C2 - 0x0010 07C3. If other frequency clocks are needed, users need to adjust the value of SYSCTRL_LSI.TRIM by themselves.



4.4.3 Clock state detection

The chip has the functions of clock stability detection, start-up failure detection, and clock failure detection during clock startup, and supports automatic clock source switching when the currently selected system clock source fails.

4.4.3.1 Clock stability detection

The 4 clock sources, HSE, LSE, HSIOSC, and LSI, all support the clock stability detection function. Users can determine the clock status through the stability flag of the corresponding clock source. The clock stable flag is cleared by hardware when the clock source is turned off, and set 1 by hardware after the clock source is enabled and stable.

Note that the clock stable flag is only for the clock start-up process. During the clock stable operation process, the detection of clock operation failure will not affect the clock stable flag.

Taking the HSIOSC clock source as an example, the HSIOSC clock stability flag and clock stability interrupt flag are described as follows:

- **SYSCTRL_HSI.STABLE**
Cleared by hardware when HSIOSC is turned off, and set by hardware when it detects that the HSIOSC clock is stable.
- **SYSCTRL_ISR.HSISTABLE**
It belongs to the same signal as SYSCTRL_HSI.STABLE. Although this signal is located in the interrupt flag register SYSCTRL_ISR, it will not cause an interrupt request after being set to 1 by hardware.
- **SYSCTRL_ISR.HSIRDY**
When the hardware detects that the clock changes from an unstable state to a stable state (that is, the SYSCTRL_HSI.STABLE flag changes from 0 to 1), it is set to 1, which is the clock stability interrupt flag. Users can clear this flag by setting SYSCTRL_ICR.HSIRDY to 0.



4.4.3.2 Clock start-up failure detection

CW32L010 supports external clock (HSE and LSE) start-up failure detection function. After the external clock is started, the clock detection logic counts the detected clock signals, and if a set number of clock signals are detected within the detection time, the clock starts to vibrate successfully, otherwise the start-up fails.

The number of clocks for HSE start-up failure detection is set through `SYSCTRL_HSE.WAITCYCLE`. The detection time does not need to be set by the user, but is automatically set by the system, as shown in the following table:

Table 4-7 HSE start-up failure detection time and number of detection clocks

<code>SYSCTRL_HSE.WAITCYCLE</code>	Detection time	Number of detection clocks
00	65ms	8192
01	65ms	32768
10	65ms	131072
11	130ms	262144

The number of clocks for LSE start-up failure detection is set through `SYSCTRL_LSE.WAITCYCLE`. The detection time does not need to be set by the user, but is automatically set by the system, as shown in the following table:

Table 4-8 LSE start-up failure detection time and number of detection clocks

<code>SYSCTRL_LSE.WAITCYCLE</code>	Detection time	Number of detection clocks
00	1s	256
01	1s	1024
10	1s	4096
11	2s	16384

The external clock (HSE and LSE) failure detection function is disabled by default, and the corresponding interrupt flag will not be generated if the HSE/LSE fails to start. By setting `SYSCTRL_CR1.HSECCS` to 1 and `SYSCTRL_CR1.LSECCS` to 1 to enable the fault detection function of the external clocks HSE and LSE, respectively.

After enabling the fault detection function of the external clock, if the HSE or LSE fails to start, a start-up failure interrupt flag will be generated (corresponding `SYSCTRL_ISR.HSEFAIL` or `SYSCTRL_ISR.LSEFAIL` is set to 1); if the interrupt is enabled (corresponding `SYSCTRL_IER.HSEFAIL` or `SYSCTRL_IER.LSEFAIL` is set to 1), the CPU will execute the interrupt service routine to handle the failure of clock start-up.

The `LSEEN` can be turned off by software after generating the start-up failure interrupt flag `LSEFAIL`, and then the `LSEFAIL` flag can be cleared by software. If `LSEEN` is not turned off, `LSEFAIL` is generated at intervals as shown in [Table 4-8 LSE start-up failure detection time and number of detection clocks](#).



4.4.3.3 In-flight failure detection of clocks

CW32L010 supports external clock (HSE or LSE) in-flight failure detection function. During the stable operation of the external clock, the clock detection logic continues to count the HSE and LSE clock signals with a certain detection period: If a set number of clock signals are detected during the detection cycle, the operation is normal, otherwise the operation is disabled.

The in-flight failure detection cycle of HSE clock is configured by `SYSCTRL_HSE.DETCNT`, the actual time is `SYSCTRL_HSE.DETCNT/fLSI`, and the number of detected clocks is 0x20000. Considering that clocks have certain deviation, in order to ensure reliable detection function, HSE detection cycle parameter `HSE.DETCNT` shall be configured with certain margin, which is generally configured as $8000/f_{HSE}$ according to HSE operation frequency (where f_{HSE} is HSE clock frequency in MHz).

Example:

If HSE clock frequency is 4MHz, `SYSCTRL_HSE.DETCNT` shall be configured to 2000.

The in-flight failure detection cycle of the LSE clock is not configurable, and is fixed at 256 LSI clock cycles, and the number of detected clocks is 128 LSE.

The external clock LSE failure detection function is disabled by default, and the corresponding interrupt flag will not be generated if the HSE or LSE fails during operation. By setting `SYSCTRL_CR1.HSECCS` to 1 and `SYSCTRL_CR1.LSECCS` to 1, the HSE and LSE clock failure detection function is enabled respectively.

After the failure detection function of the external clock is enabled, if an HSE or LSE failure is detected during operation, a in-flight failure flag will be generated (corresponding `SYSCTRL_ISR.HSEFAULT` or `SYSCTRL_ISR.LSEFAULT` is set to 1), if the interrupt is enabled (corresponding `SYSCTRL_IER.HSEFAULT` or `SYSCTRL_IER.LSEFAULT` is set to 1), the CPU will execute the interrupt service routine for clock in-flight failure process.

After generating the in-flight failure flag `LSEFAULT` it is recommended that software turn off `LSEEN` to clear the `LSESTABLE` state, and then software clears the `LSEFAULT` flag. If `LSEEN` is not turned off, `LSEFAULT` is generated every 256 LSI clock durations (approximately 7.8ms) until LSE returns to normal.

If the current system clock source is HSE, set `SYSCTRL_CR1.CLKCCS` to 0, no action will be taken when HSE fails in-flight; set `SYSCTRL_CR1.CLKCCS` to 1, the system clock source will automatically switch to HSI4MHz clock when HSE fails in-flight.

If the current system clock source is LSE, set `SYSCTRL_CR1.CLKCCS` to 0, no action will be taken when LSE fails in-flight; set `SYSCTRL_CR1.CLKCCS` to 1, system clock source will automatically switch to HSI4MHz clock when LSE fails in-flight.



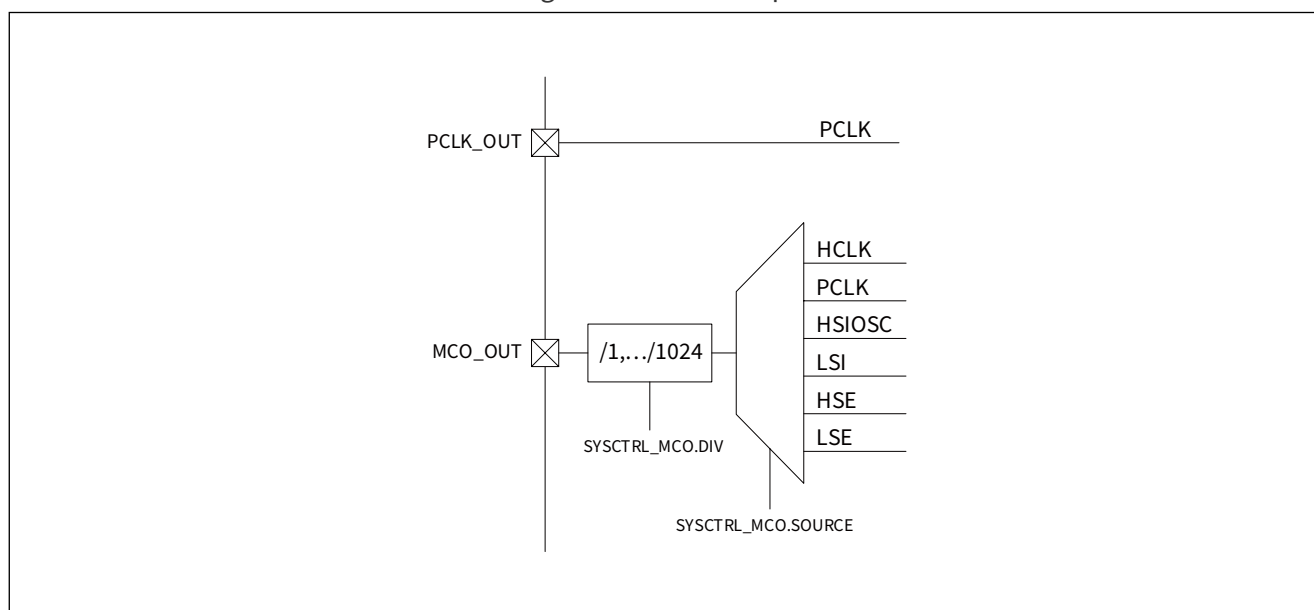
4.4.4 Clock verification and output

CW32L010 supports outputting various internal clock signals to external pins. Users can use this function to measure the current system's CPU operating frequency, system bus frequency, peripheral operating frequency, etc.

- PCLK_OUT pin: Output PCLK clock signal
- MCO_OUT pin: Output HCLK/PCLK/HSIOSC/LSI/HSE/LSE clock signal, before the clock is output to the MCO_OUT pin, it can be divided by a prescaler (effective frequency division factors are 1, 2, 8, 64, 128, 256, 512, 1024), so that the low-bandwidth instrument can accurately measure the signal.

The clock output block diagram is shown in the following figure:

Figure 4-5 Clock output



4.5 SysClk system clock switch

The system clock SysClk can select 4 clock sources, including HSE, LSE, HSI, and LSI. By setting the SYSCLK bit field of the system control register SYSCTRL_CR0, it can be switched between different clock sources.

The main application scenarios include:

- Automatic clock failover
When it is detected that the current system clock source fails, it can quickly and safely switch to other available clock sources to improve product reliability.
- Power management
Switch the clock source of SysClk to low-speed clock in standby state to reduce system power consumption, and switch the clock source of SysClk to high-speed clock in normal use state to quickly respond to user needs.

The clock source safety switching rules are as follows:

- Any two of HSE, LSE, HSI and LSI clock sources can be switched to each other.

The system clock switching operation must be performed in accordance with the clock switching procedures described below, otherwise an exception may occur.

Caution:

When the system clock is switched, the FLASH control register FLASH_CR2.WAIT needs to be configured synchronously to read the waiting period parameters: if the system clock frequency is not greater than 24MHz, the FLASH control register FLASH_CR2.WAIT should be set to 0; if the system clock frequency is greater than 24MHz, the FLASH control register FLASH_CR2.WAIT should be set is 1.



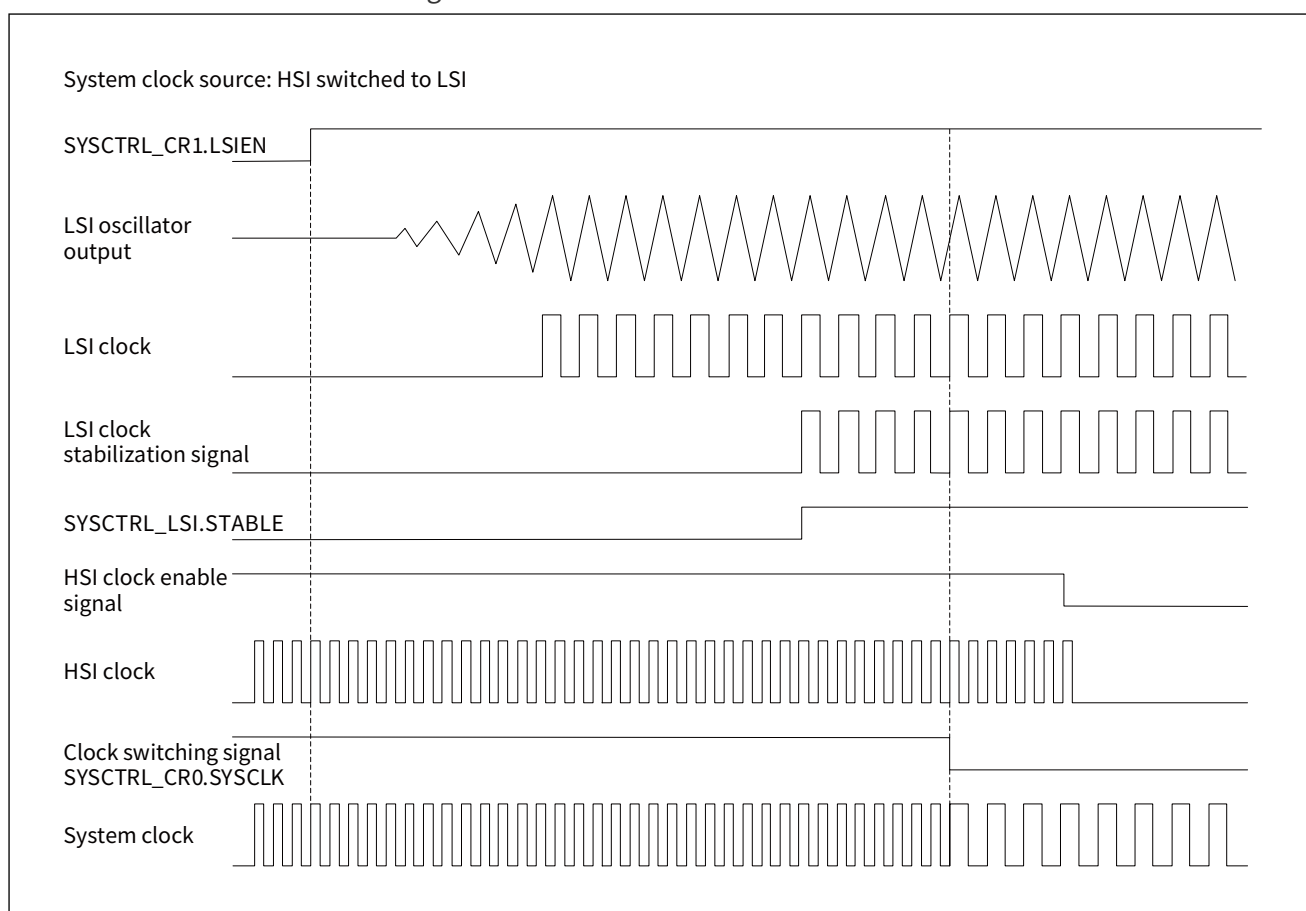
4.5.1 Standard Clock Switching Process

The standard clock switching operation flow is as follows:

- Step 1: If the new clock source is HSE or LSE, you need to configure the corresponding external pin to the appropriate mode: when connecting an external crystal, you need to configure it as an analog function, when connecting an external clock input, you need to set the GPIO as a digital input function and enable the port's input function;
- Step 2: Configure the oscillation parameters of the new clock source, such as drive capability, oscillation amplitude, stabilization period, etc.;
- Step 3: Set the new clock source oscillator enable bit to 1;
- Step 4: According to the higher frequency of the current clock source and the new clock source, configure the FLASH_CR2.WAIT read wait period parameter according to the requirements of the chapter [7 FLASH Memory](#);
- Step 5: Wait for the new clock source to output a stable frequency (the STABLE signal of the new clock source becomes 1);
- Step 6: Configure the system control register SYSCTRL_CR0.SYSCLK, and select the clock source of the system clock SysClk as the new clock source;
- Step 7: According to the frequency of the new clock source, configure the FLASH_CR2.WAIT read waiting period parameter according to the requirements of the chapter [7 FLASH Memory](#);
- Step 8: Turn off clock sources that are no longer in use.

Taking the system clock switching from HSI to LSI as an example, the switching process of the old and new clocks is shown in the following figure:

Figure 4-6 HSI clock switched to LSI clock



4.5.2 HSI clock switching process between different frequencies

Switching between different frequencies of the HSI clock does not require starting a new clock source, but only needs to change the frequency division ratio of the HSI clock prescaler. The switching speed is fast, and there is no need to wait for the clock stabilization time before and after switching. The switching operation process is as follows:

- Step 1: According to the higher frequency of the current clock source and the new clock source, configure the FLASH_CR2.WAIT read wait period parameter according to the requirements of the chapter [7 FLASH Memory](#);
- Step 2: Read the value of the built-in high-frequency clock control register SYSCTRL_HSI register as Value;
- Step 3: According to the required target frequency, set the built-in high-frequency clock control register SYSCTRL_HSI.DIV to the corresponding frequency division coefficient (note that the SYSCTRL_HSI.TRIM field should remain unchanged);
- Step 4: According to the frequency of the new clock source, configure the FLASH_CR2.WAIT read wait period parameter as required in the chapter [7 FLASH Memory](#).

4.5.3 Example of switching from other clocks to LSE

The switching operation process is as follows:

- Step 1: Set PB00 / PB01 pins as analog function ports;
- Step 2: According to the characteristics of the external crystal of the LSE oscillator, configure the oscillation parameters of the new clock source LSE, such as drive capability (set by the external low-frequency crystal control register SYSCTRL_LSE.DRIVER), stable period (set through the external low-frequency crystal control register SYSCTRL_LSE.WAITCYCLE, it is recommended to set to 3), etc.;
- Step 3: Set the system control register SYSCTRL_CR1.LSEEN to 1 to enable the LSE crystal oscillator circuit;

Caution:

The SYSCTRL_CR1 register has the KEY protection feature. The upper 16-bit data of the written data must be 0x5A5A, otherwise it cannot be written.

- Step 4: Cyclic query and wait for the external low-frequency crystal control register SYSCTRL_LSE.STABLE stable flag to become 1, that is, wait for the LSE crystal oscillator circuit to output a stable clock;
- Step 5: Set the system control register SYSCTRL_CR0.SYSCLK to 4, and switch the SysClk clock source to LSE;

Caution:

The SYSCTRL_CR0 register has the KEY protection feature. The upper 16-bit data of the written data must be 0x5A5A, otherwise it cannot be written.

- Step 6: Configure the FLASH_CR2.WAIT read wait period parameter according to the requirements of the chapter [7 FLASH Memory](#);
- Step 7: Set the system control register SYSCTRL_CR1.xxxEN to 0 to turn off the original clock.



4.5.4 Example of switching from other clocks to HSE

The switching operation process is as follows:

Step 1: Set PA00/PA01 pins as analog function ports;

Step 2: According to the characteristics of the external crystal of the HSE oscillator, configure the oscillation parameters of the new clock source HSE, such as drive capability (through the external high-frequency crystal control register `SYSCTRL_HSE.DRIVER` configuration), stabilization period (configured through the external high-frequency crystal control register `SYSCTRL_HSE.WAITCYCLE`, it is recommended to set it to 3), etc.;

Step 3: Set the system control register `SYSCTRL_CR1.HSEEN` to 1 to enable the HSE crystal oscillator circuit;

Caution:

The `SYSCTRL_CR1` register has the KEY protection feature. The upper 16-bit data of the written data must be 0x5A5A, otherwise it cannot be written.

Step 4: According to the higher frequency of the current clock and HSE, configure the `FLASH_CR2.WAIT` read waiting period parameter according to the requirements of the chapter [7 FLASH Memory](#);

Step 5: Cyclic query in a loop and wait for the stable flag of the external high-frequency crystal control register `SYSCTRL_HSE.STABLE` to become 1 to ensure that the crystal oscillator circuit outputs a stable clock;

Step 6: Set the system control register `SYSCTRL_CR0.SYSCLK` to 1, and switch the SysClk clock source to HSE;

Caution:

The `SYSCTRL_CR0` register has the KEY protection feature. The upper 16-bit data of the written data must be 0x5A5A, otherwise it cannot be written.

Step 7: According to the clock frequency of the HSE, configure the `FLASH_CR2.WAIT` read waiting period parameter according to the requirements of the chapter [7 FLASH Memory](#);

Step 8: Set the system control register `SYSCTRL_CR1.xxxEN` to 0 to turn off the original clock.

4.5.5 Example of switching from other clocks to LSI

The switching operation process is as follows:

Step 1: Configure the built-in low-frequency clock control register `SYSCTRL_LSI.TRIM` and the built-in low-frequency clock control register `SYSCTRL_LSI.WAITCYCLE` to appropriate values;

Step 2: Set the system control register `SYSCTRL_CR1.LSIEN` to 1 to enable the LSIRC oscillator circuit;

Caution:

The `SYSCTRL_CR1` register has the KEY protection feature. The upper 16-bit data of the written data must be 0x5A5A, otherwise it cannot be written.

Step 3: Cyclic query and wait for the built-in low-frequency clock control register `SYSCTRL_LSI.STABLE` stable flag to become 1, that is, wait for the LSI to output a stable clock;

Step 4: Set the system control register `SYSCTRL_CR0.SYSCLK` to 3, and switch the SysClk clock source to LSI;

Caution:

The `SYSCTRL_CR0` register has the KEY protection feature. The upper 16-bit data of the written data must be 0x5A5A, otherwise it cannot be written.

Step 5: Configure the `FLASH_CR2.WAIT` read wait period parameter to 0;

Step 6: Set the system control register `SYSCTRL_CR1.xxxEN` to 0 to turn off the original clock.



4.5.6 Example of switching from other clocks to HSI

The switching operation process is as follows:

Step 1: Configure the built-in high-frequency clock control registers `SYSCTRL_HSI.TRIM`, `SYSCTRL_HSI.WAITCYCLE` and `SYSCTRL_HSI.DIV` to appropriate values;

Step 2: Set the system control register `SYSCTRL_CR1.HSIEN` to 1 to enable the `HSIOSC` clock oscillation circuit;

Caution:

The `SYSCTRL_CR1` register has the KEY protection feature. The upper 16-bit data of the written data must be 0x5A5A, otherwise it cannot be written.

Step 3: Cyclic query and wait for the built-in high-frequency clock control register `SYSCTRL_HSI.STABLE` stable flag to become 1, that is, wait for the `HSIOSC` to output a stable clock;

Step 4: According to the higher frequency of the current clock and HSI, configure the `FLASH_CR2.WAIT` read waiting period parameter according to the requirements of the chapter [7 FLASH Memory](#);

Step 5: Set the system control register `SYSCTRL_CR0.SYSCLK` to 0, and switch the `SysClk` clock source to HSI;

Caution:

The `SYSCTRL_CR0` register has the KEY protection feature. The upper 16-bit data of the written data must be 0x5A5A, otherwise it cannot be written.

Step 6: According to the clock frequency of HSI, configure the `FLASH_CR2.WAIT` read waiting period parameter according to the requirements of the chapter [7 FLASH Memory](#);

Step 7: Set the system control register `SYSCTRL_CR1.xxxEN` to 0 to turn off the original clock.



4.6 List of registers

SYSCTRL base address: SYSCTRL_BASE = 0x4000 4000

Table 4-9 List of SYSCTRL registers

Register name	Register address	Register description
SYSCTRL_CR0	SYSCTRL_BASE + 0x00	System Control Register 0
SYSCTRL_CR1	SYSCTRL_BASE + 0x04	System Control Register 1
SYSCTRL_CR2	SYSCTRL_BASE + 0x08	System Control Register 2
SYSCTRL_IER	SYSCTRL_BASE + 0x0C	System Interrupt Enable Control Register
SYSCTRL_ISR	SYSCTRL_BASE + 0x10	System Interrupt Flag Register
SYSCTRL_ICR	SYSCTRL_BASE + 0x14	System Interrupt Flag Clear Register
SYSCTRL_HSI	SYSCTRL_BASE + 0x18	Internal high frequency clock control register
SYSCTRL_HSE	SYSCTRL_BASE + 0x1C	External high frequency crystal control register
SYSCTRL_LSI	SYSCTRL_BASE + 0x20	Internal low frequency clock control register
SYSCTRL_LSE	SYSCTRL_BASE + 0x24	External low frequency crystal control register
SYSCTRL_DEBUG	SYSCTRL_BASE + 0x2C	Debug Status Timer Control Register
SYSCTRL_AHBEN	SYSCTRL_BASE + 0x30	AHB Peripheral Clock Enable Control Register
SYSCTRL_APBEN2	SYSCTRL_BASE + 0x34	APB Peripheral Clock Enable Control Register 2
SYSCTRL_APBEN1	SYSCTRL_BASE + 0x38	APB Peripheral Clock Enable Control Register 1
SYSCTRL_AHBRST	SYSCTRL_BASE + 0x40	AHB Peripheral Reset Control Register
SYSCTRL_APBRS2	SYSCTRL_BASE + 0x44	APB Peripheral Reset Control Register 2
SYSCTRL_APBRS1	SYSCTRL_BASE + 0x48	APB Peripheral Reset Control Register 1
SYSCTRL_RESETFLAG	SYSCTRL_BASE + 0x4C	System Reset Flag Register
SYSCTRL_MCO	SYSCTRL_BASE + 0x70	System Clock Output Control Register



4.7 Register description

See section [1 Documentation conventions](#) for abbreviations used in register descriptions.

4.7.1 SYSCTRL_CR0 System control register 0

Address offset: 0x00 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	KEY	WO	Only when KEY is 0x5A5A, the write operation of this register is valid
15:8	RFU	-	Reserved bits, please keep the default value
7:5	HCLKPRS	RW	Configure the clock source of HCLK 000: Set the clock source of HCLK to SysClk 001: Set the clock source of HCLK to SysClk / 2 010: Set the clock source of HCLK to SysClk / 4 011: Set the clock source of HCLK to SysClk / 8 100: Set the clock source of HCLK to SysClk / 16 101: Set the clock source of HCLK to SysClk / 32 110: Set the clock source of HCLK to SysClk / 64 111: Set the clock source of HCLK to SysClk / 128
4:3	PCLKPRS	RW	Configure the clock source of PCLK 00: Set the clock source of PCLK to HCLK 01: Set the clock source of PCLK to HCLK / 2 10: Set the clock source of PCLK to HCLK / 4 11: Set the clock source of PCLK to HCLK / 8
2:0	SYSCLK	RW	Configure the clock source of SysClk 000: Set the clock source of SysClk to HSI 001: Set the clock source of SysClk to HSE 011: Set the clock source of SysClk to LSI 100: Set the clock source of SysClk to LSE



4.7.2 SYSCTRL_CR1 System Control Register 1

Address offset: 0x04 Reset value: 0x0000 0001

Bit field	Name	Permission	Function description
31:16	KEY	WO	Only when KEY is 0x5A5A, the write operation of this register is valid
15:9	RFU	-	Reserved bits, please keep the default value
8	CLKCCS	RW	Configure the crystal oscillator for safe operation in the event of a failure in operation 0: No action 1: Automatically switch the clock source of SysClk to HSI 4MHz
7	HSECCS	RW	High frequency crystal oscillator clock monitor enable configuration 0: Disable HSE clock monitor 1: Enable HSE clock monitor <i>Caution 1: During operation, if the number of clocks output by the HSE is less than 0x20000 clocks per HSE. DETCNT LSI clock durations, the HSE is judged to be disabled during runtime.</i> <i>Caution 2: During startup, if the condition specified by HSE.WAITCYCLE is not satisfied, the HSE is judged to startup failure.</i> <i>Caution 3: The power consumption of DeepSleep mode will be increased by about 1μA when this function is enabled.</i>
6	LSECCS	RW	Lowfrequency crystal oscillator clock monitor enable configuration 0: Disable LSE clock monitor 1: Enable LSE clock monitor <i>Caution 1: During operation, if the number of clocks output by the LSE is less than 128 within the duration of every 256 LSI clocks (about 7.8ms), the LSE is judged to be disabled during runtime.</i> <i>Caution 2: During startup, if the conditions specified by LSE.WAITCYCLE are not met, it is determined that the LSE has failed to start vibrating.</i> <i>Caution 3: Enable this feature and the power consumption of DeepSleep mode will increase by about 1μA.</i>
5	LSELOCK	RW	External low-speed clock LSE lock control 0: LSEEN can be set and cleared 1: LSEEN only settable <i>Caution 1: The bit field can be reset only by a power-on reset after enable.</i> <i>Caution 2: This function does not work when the LSEFAIL or LSEFAULT flag is generated, i.e., at this time LSEEN can be set or cleared until the above flags are cleared by the user software.</i>



Bit field	Name	Permission	Function description
4	LSEEN	RW	External low-speed clock LSE enable control 0: Disabled 1: Enabled <i>Caution 1: When the system enters DeepSleep, this low-speed clock will not automatically shut down.</i> <i>Caution 2: When LSELOCK=0, LSEEN can be turned off by software, other than that it can only be reset by power-on reset; when LSELOCK=1, only power-on reset can reset LSEEN.</i>
3	LSIEN	RW	Internal low-speed clock LSI enable control 0: Disabled 1: Enabled <i>Caution: When the system enters DeepSleep, this low-speed clock will not automatically shut down.</i>
2	RFU	-	Reserved bits, please keep the default value
1	HSEEN	RW	External high-speed clock HSE enable control 0: Disabled 1: Enabled <i>Caution: When the system enters DeepSleep, this high-speed clock will be automatically turned off.</i>
0	HSIEN	RW	Internal high-speed clock HSIOSC enable control 0: Disabled 1: Enabled <i>Caution: When the system enters DeepSleep, this high-speed clock will be automatically turned off.</i>

Caution:

The LSI clock can be turned on automatically by hardware in the following ways, but the `SYSCTRL_CR1.LSIEN` bit field will not be set. This bit field can only be set by software writing `SYSCTRL_CR1.LSIEN` to 1:

- (a) `SYSCTRL_CR1.HSECCS = 1;`
- (b) `SYSCTRL_CR1.LSECCS = 1;`
- (c) `GPIOx_FILTER.FLTCLK = 0x5 (LSI);`
- (d) `VCx_CR1.FLTCLK = 0x0 (LSI)` and `VCx_CR0.EN = 1;`
- (e) `LVD_CR0.FLTCLK = 0x0 (LSI)` and `LVD_CR0.EN = 1;`
- (f) `IWDT_KR = 0xCCCC (start IWDT).`



4.7.3 SYSCTRL_CR2 SYSCTRL_CR2 System Control Register 2

Address offset: 0x08 Reset value: 0x0000 0C00

Bit field	Name	Permission	Function description
31:16	KEY	WO	Only when KEY is 0x5A5A, the write operation of this register is valid
15	RAMBRKEN	RW	RAM parity error triggers ATIM brake enable control 0: Disabled 1: Enabled
14	LVDBRKEN	RW	LVD output triggers ATIM brake enable control 0: Disabled 1: Enabled
13	DSBRKEN	RW	DeepSleep triggers ATIM brake enable control 0: Disabled 1: Enabled
12	CLLBRKEN	RW	Generate HardFault or Cortex-M0+ LockUp flag to trigger ATIM brake enable control 0: Disabled 1: Enabled
11	LSEBRKEN	RW	LSE in-flight failure triggers ATIM brake enable control 0: Disabled 1: Enabled
10	HSEBRKEN	RW	HSE failure in-flight triggers ATIM brake enable control 0: Disabled 1: Enabled
9:8	RFU	-	Reserved bits, please keep the default value
7	RTCLPM	RW	RTC low power configuration 0: Normal power consumption mode 1: Low power mode, turn on this function to reduce RTC power consumption, software write operation to RTC register can not be synchronized to RTC module
6:4	FLASHWAIT	RW	FLASH instruction fetches cycle configuration 000: 1 HCLK cycle for HCLK≤24MHz 001: 2 HCLK cycle for 24MHz<HCLK≤48MHz Caution: This control bit has the same function as FLASH_CR2[2:0].
3	WAKEUPCLK	RW	When DeepSleep wakes up, the source configuration of the system clock 0: Keep the original system clock source 1: Switch the system clock source to HSI 4MHz, keep the original system clock source enabled



Bit field	Name	Permission	Function description
2	LOCKUP	RW	Cortex-M0+ LockUp Functional configuration 0: Disabled 1: Enabled <i>Caution: When this function is enabled, the CPU will reset the MCU when it reads an invalid instruction.</i>
1	SWDIO	RW	SWD Port function configuration 0: PA08, PA07 are used as SWD ports, GPIO function is not available 1: PA08, PA07 are used as GPIO ports, SWD function is not available
0	RSTIO	RW	NRST port function configuration 0: PB07 is used as NRST port, GPIO function unavailable 1: PB07 is used as GPIO port, NRST function unavailable <i>Caution: As GPIO port, can only be used as input.</i>



4.7.4 SYSCTRL_HSI Internal high frequency clock control register

Address offset: 0x18 Reset value: 0x---- ----

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15	STABLE	RO	HSIOSC Clock Stable Status Bits 0: HSIOSC clock has not stabilized 1: HSIOSC clock has stabilized
14:11	DIV	RW	HSI clock and HSIOSC clock division factor configuration 0000: HSI = HSIOSC / 16 0001: HSI = HSIOSC / 1 0010: HSI = HSIOSC / 2 0011: HSI = HSIOSC / 3 0100: HSI = HSIOSC / 4 0101: HSI = HSIOSC / 5 0110: HSI = HSIOSC / 6 0111: HSI = HSIOSC / 7 1000: HSI = HSIOSC / 8 1001: HSI = HSIOSC / 9 1010: HSI = HSIOSC / 10 (Default) 1011: HSI = HSIOSC / 11 1100: HSI = HSIOSC / 12 1101: HSI = HSIOSC / 13 1110: HSI = HSIOSC / 14 1111: HSI = HSIOSC / 15
10:0	TRIM	RW	Clock frequency adjustment, changing the value of this register bit can adjust the oscillation frequency of the HSIOSC. Every time the TRIM value increases by 1, the oscillation frequency of the HSIOSC increases by about 0.2%, and the total adjustment range is 47 ~ 49MHz. The calibration value of 48MHz has been saved in FLASH, and the accurate frequency can be obtained by reading the calibration value in FLASH and writing it into SYSCTRL_HSI.TRIM. 48.00MHz Calibration value address: 0x0010 07C0 – 0x0010 07C1



4.7.5 SYSCTRL_LSI Internal low frequency clock control register

Address offset: 0x20 Reset value: 0x---- ----

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15	STABLE	RO	LSI Clock Stable Status Bit 0: LSI clock has not stabilized 1: LSI clock has stabilized
14:12	RFU	-	Reserved bits, please keep the default value
11:10	WAITCYCLE	RW	Internal low-speed clock LSI stabilization time selection 00: 6 cycles 01: 18 cycles 10: 66 cycles 11: 258 cycles
9:0	TRIM	RW	Clock frequency adjustment, change the value of this register bit to adjust the oscillation frequency of LSI, every increase of TRIM value by 1 will decrease the oscillation frequency of LSI by about 0.16%, the total adjustment range is 22kHz~46kHz. 1 set of frequency calibration value has been saved in FLASH, read the preset calibration value from FLASH, write it to this bit field to get the accurate LSI frequency. 32.8kHz calibration value address: 0x0010 07C2 - 0x0010 07C3



4.7.6 SYSCTRL_HSE External high frequency crystal control register

Address offset: 0x1C Reset value: 0x0000 0020

Bit field	Name	Permission	Function description
31:25	RFU	-	Reserved bits, please keep the default value
24	DIGFLT	RW	External high-speed clock HSE digital filter control 0: Disable HSE digital filter function. 1: Enable the HSE digital filter function, the HSE frequency should not be greater than 8MHz. Caution 1: It is recommended to enable the filtering function only when the interference signal is extremely strong. Caution 2: When HCLK source is HSE, it is forbidden to modify this control bit.
23:20	PDRIVER	RW	HSE crystal drive capability configuration before HSE clock stabilizes; see DRIVER bit field description.
19	STABLE	RO	HSE clock stable status bit 0: HSE clock has not been stabilized 1: HSE clock has been stabilized
18:8	DETCNT	RW	HSE in-flight failure detection cycle configuration In consideration of clock errors, the DETCNT should generally be set to a value of $8000 / f_{HSE}$ Example: HSE is 4MHz, then DETCNT should be configured to 2000. Caution: DETCNT must be greater than 0.
7	HEXENPOL	RW	Output to external oscillator enable signal polarity configuration 0: Output high level in Active mode, output low level in DeepSleep mode. 1: Output low level in Active mode, output high level in DeepSleep mode. Caution: It is necessary to configure the corresponding pin as HEXEN multiplexed output.
6	MODE	RW	External high-speed clock HSE operating mode configuration 0: Crystal mode, whose output clock is generated by the crystal resonating with the internal circuit 1: Input mode, whose output clock comes from the OSC_IN pin



Bit field	Name	Permission	Function description
5:4	WAITCYCLE	RW	HSE stability detection and startup failure detection clock cycle number selection 00: Wait for 8192 HSE clock signals 01: Wait for 32768 HSE clock signals 10: Wait for 131072 HSE clock signals 11: Wait for 262144 HSE clock signals Caution: When HSECCS is enabled, the startup failure detection circuit will detect the number of HSE output clock signals within a limited time, and the four limit times are 65ms, 65ms, 65ms, and 130ms respectively.
3:0	DRIVER	RW	HSE crystal drive capability configuration 0000: Minimum drive capability ... 0111: Maximum drive capability Caution: You need to select the appropriate drive capability according to the crystal characteristics, load capacitance, and parasitic parameters of the board. The stronger the drive capability, the higher the power consumption; the weaker the drive capability, the lower the power consumption.



4.7.7 SYSCTRL_LSE External low frequency crystal control register

Address offset: 0x24 Reset value: 0x0000 0A2B

Bit field	Name	Permission	Function description
31:19	RFU	-	Reserved bits, please keep the default value
18	STABLE	RO	LSE clock stable status bit 0: LSE clock has not stabilized 1: LSE clock has stabilized
17	PINLOCK	RW	External low-speed clock LSE pin locked to analog state 0: LSE pin rewritable 1: LSE pin locked to analog state
16:12	RFU	-	Reserved bits, please keep the default value
11:8	PDRIVER	RW	Configuration of LSE crystal drive capability before LSE clock stabilizes; see DRIVER bit-field description.
7	RFU	-	Reserved bits, please keep the default value
6	MODE	RW	External low-speed clock LSE operating mode configuration 0: Crystal mode, whose output clock is generated by the crystal resonating with the internal circuit 1: Input mode, whose output clock comes from the OSC32_IN pin
5:4	WAITCYCLE	RW	Selection of the number of clock cycles for LSE stability detection and start-up failure detection 00: Waiting for 256 LSE clock signals 01: Waiting for 1024 LSE clock signals 10: Waiting for 4096 LSE clock signals 11: Waiting for 16384 LSE clock signals <i>Caution: When LSECCS is enabled, the start-up failure detection circuit will detect the number of LSE clock signals within a limited time. The four-stage limited time is 1s, 1s, 1s, and 2s respectively.</i>
3:0	DRIVER	RW	LSE crystal drive capability configuration 0000: Minimum drive capability ... 1111: Maximum drive capability <i>Caution: The appropriate drive capability needs to be selected based on the crystal characteristics, load capacitance, and board parasitics. The stronger the driving ability, the greater the power consumption, and the weaker the driving ability, the smaller the power consumption.</i>

Caution:*Only a power-on reset can reset this register.*

4.7.8 SYSCTRL_IER System interrupt enable control register

Address offset: 0x0C Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	KEY	WO	Only when KEY is 0x5A5A, the write operation of this register is valid
15:9	RFU	-	Reserved bits, please keep the default value
8	HSEFAULT	RW	HSE operation fault interrupt enable control 0: Disable HSE operation fault interrupt 1: Enable HSE operation fault interrupt
7	LSEFAULT	RW	LSE Fail-Run Interrupt Enable Control 0: Disable LSE run-fail interrupt 1: Enable LSE run-fail interrupt
6	HSEFAIL	RW	HSE startup failure interrupt enable control 0: Disable HSE startup failure interrupt 1: Enable HSE startup failure interrupt
5	LSEFAIL	RW	LSE startup failure interrupt enable control 0: Disable LSE start failure interrupt 1: Enable LSE start failure interrupt
4	LSERDY	RW	LSE Stable Interrupt Enable Control 0: Disable LSE stable interrupt 1: Enable LSE stable interrupt
3	LSIRDY	RW	LSI Stable Interrupt Enable Control 0: Disable LSI stable interrupt 1: Enable LSI stable interrupt
2	RFU	-	Reserved bits, please keep the default value
1	HSERDY	RW	HSE stable interrupt enable control 0: Disable HSE stable interrupt 1: Enable HSE stable interrupt
0	HSIRDY	RW	HSIOSC Stable Interrupt Enable Control 0: Disable HSIOSC stable interrupt 1: Enable HSIOSC stable interrupt

Caution 1:

The FAULT interrupt (vector number 47) is executed when HSEFAULT interrupts and LSEFAULT interrupts are generated, and the RCC global interrupt (vector number 20) is executed when the other interrupt is generated.



4.7.9 SYSCTRL_ISR System Interrupt Flag Register

Address offset: 0x10 Reset value: 0x0000 0801

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15	LSESTABLE	RO	LSE clock stable status bit; hardware set to clear 0: LSE clock has not stabilized 1: LSE clock has stabilized
14	LSISTABLE	RO	LSI clock stable status bit; hardware set to clear 0: LSI clock has not stabilized 1: LSI clock has stabilized
13	RFU	-	Reserved bits, please keep the default value
12	HSESTABLE	RO	HSE clock stable status bit; hardware set to clear 0: HSE clock has not stabilized 1: HSE clock has stabilized
11	HSISTABLE	RO	HSIOSC clock stable status bit; hardware set to clear 0: HSIOSC clock has not stabilized 1: HSIOSC clock has stabilized
10:9	RFU	-	Reserved bits, please keep the default value
8	HSEFAULT	RO	HSE clock operation failure interrupt flag bit 0: HSE clock is running normally 1: Oscillation stopped during HSE clock operation
7	LSEFAULT	RO	LSE Clock run failure interrupt flag 0: LSE clock is running fine 1: Oscillation stopped during LSE clock operation <i>Caution: After generating LSEFAULT it is recommended that software turn off LSEEN to clear the LSESTABLE state and then software clear the LSEFAULT flag. If LSEEN is not turned off, LSEFAULT will be generated every 256 LSI clock durations (approximately 7.8ms) until the LSE returns to normal.</i>
6	HSEFAIL	RO	HSE clock start-up failure interrupt flag 0: HSE clock starts up successfully 1: HSE clock start-up failed
5	LSEFAIL	RO	LSE clock start-up failure interrupt flag 0: LSE Clock starts up successfully 1: LSE clock start-up failed <i>Caution: After generating LSEFAIL, LSEEN can be turned off by software and then the LSEFAIL flag can be cleared by software. If LSEEN is not turned off, LSEFAIL is generated at intervals described in the WAITCYCLE bit field of the SYSCTRL_LSE register.</i>



Bit field	Name	Permission	Function description
4	LSERDY	RO	LSE clock stable interrupt flag 0: LSE clock has not stabilized 1: LSE clock has stabilized <i>Caution: Clearing ICR.LSEFAULT, ICR.LSEFAIL, and ICR.LSERDY are able to clear the LSERDY flag.</i>
3	LSIRDY	RO	LSI clock stable interrupt flag 0: LSI clock has not stabilized 1: LSI clock has stabilized
2	RFU	-	Reserved bits, please keep the default value
1	HSE RDY	RO	HSE clock stable interrupt flag 0: HSE clock has not stabilized 1: HSE clock has stabilized <i>Caution: Clearing ICR.HSEFAULT, ICR.HSEFAIL does not clear the HSE RDY flag synchronously.</i>
0	HSIRDY	RO	HSIOSC clock stable interrupt flag 0: HSIO SC clock has not stabilized 1: HSIO SC clock has stabilized

Caution 1:

The FAULT interrupt (vector number 47) is executed when the HSEFAULT interrupt and LSEFAULT interrupt are generated, and the RCC global interrupt (vector number 20) is executed when other interrupts are generated.

Caution 2:

The STABLE signal of the clock is set by the hardware and cleared by the hardware, and the RDY signal is set by the hardware and cleared by the software writing the corresponding ICR register bit field.



4.7.10 SYSCTRL_ICR System interrupt flag clear register

Address offset: 0x14 Reset value: 0x0000 00B9

Bit field	Name	Permission	Function description
31:9	RFU	-	Reserved bits, please keep the default value
8	HSEFAULT	R1W0	HSE clock operation failure interrupt flag clear control W0: Clear the corresponding flag in the ISR register W1: No function
7	LSEFAULT	R1W0	LSE Clock operation failure interrupt flag clear control W0: Clear the corresponding flag in the ISR register W1: No function <i>Caution: Clearing the LSEFAULT flag will simultaneously clear the LSERDY flag.</i>
6	HSEFAIL	R1W0	HSE Clock start-up failure interrupt flag clear control W0: Clear the corresponding flag in the ISR register W1: No function
5	LSEFAIL	R1W0	LSE Clock start-up failure interrupt flag clear control W0: Clear the corresponding flag in the ISR register W1: No function <i>Caution: Clearing the LSEFAULT flag will simultaneously clear the LSERDY flag.</i>
4	LSERDY	R1W0	LSE Clock stable interrupt flag clear control W0: Clear the corresponding flag in the ISR register W1: No function
3	LSIRDY	R1W0	LSI Clock stable interrupt flag clear control W0: Clear the corresponding flag in the ISR register W1: No function
2	RFU	-	Reserved bits, please keep the default value
1	HSERDY	R1W0	HSE Clock stable flag clear control W0: Clear the corresponding flag in the ISR register W1: No function
0	HSIRDY	R1W0	HSIOSC Clock stable interrupt flag clear control W0: Clear the corresponding flag in the ISR register W1: No function



4.7.11 SYSCTRL_AHBEN AHB Peripheral Clock Enable Control Register

Address offset: 0x30 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	KEY	WO	Only when KEY is 0x5A5A, the write operation of this register is valid
15:6	RFU	-	Reserved bits, please keep the default value
5	GPIOB	RW	GPIOB Port configuration clock and working clock enable control 0: Disabled 1: Enabled
4	GPIOA	RW	GPIOA Port configuration clock and working clock enable control 0: Disabled 1: Enabled
3	RFU	-	Reserved bits, please keep the default value
2	CRC	RW	CRC Module configuration clock and working clock enable control 0: Disabled 1: Enabled
1	FLASH	RW	FLASH Module configuration clock enable control 0: Disabled 1: Enabled
0	RFU	-	Reserved bits, please keep the default value



4.7.12 SYSCTRL_APBEN1 APB Peripheral Clock Enable Control Register 1

Address offset: 0x38 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	KEY	WO	Only when KEY is 0x5A5A, the write operation of this register is valid
15:7	RFU	-	Reserved bits, please keep the default value
6	GTIM	RW	GTIM Module configuration clock and working clock enable control 0: Disabled 1: Enabled
5	ATIM	RW	ATIM Module configuration clock and working clock enable control 0: Disabled 1: Enabled
4	UART2	RW	UART2 Module configuration clock enable control 0: Disabled 1: Enabled
3	UART1	RW	UART1 Module configuration clock enable control 0: Disabled 1: Enabled
2	SPI	RW	SPI Module configuration clock and working clock enable control 0: Disabled 1: Enabled
1	VC	RW	VC Module configuration clock enable control 0: Disabled 1: Enabled
0	ADC	RW	ADC Module configuration clock and working clock enable control 0: Disabled 1: Enabled



4.7.13 SYSCTRL_APBEN2 APB Peripheral Clock Enable Control Register 2

Address offset: 0x34 Reset value: 0x0000 0100

Bit field	Name	Permission	Function description
31:16	KEY	WO	Only when KEY is 0x5A5A, the write operation of this register is valid
15:8	RFU	-	Reserved bits, please keep the default value
7	LPTIM	RW	LPTIM Module configuration clock enable control 0: Disabled 1: Enabled
6	I2C	RW	I2C Module configuration clock enable control 0: Disabled 1: Enabled
5	RFU	-	Reserved bits, please keep the default value
4	IWDT	RW	IWDT Module configuration clock enable control 0: Disabled 1: Enabled
3	RFU	-	Reserved bits, please keep the default value
2	BTIM123	RW	BTIM1-3 Module configuration clock and working clock enable control 0: Disabled 1: Enabled
1	RTC	RW	RTC Module configuration clock enable control 0: Disabled 1: Enabled
0	RFU	RW	Reserved bits, please keep the default value



4.7.14 SYSCTRL_AHBRST AHB Peripheral Reset Control Register

Address offset: 0x40 Reset value: 0x0000 007F

Bit field	Name	Permission	Function description
31:6	RFU	-	Reserved bits, please keep the default value
5	GPIOB	RW	GPIOB Module reset control 0: Module is in reset state 1: Module works fine
4	GPIOA	RW	GPIOA Module reset control 0: Module is in reset state 1: Module works fine
3	RFU	-	Reserved bits, please keep the default value
2	CRC	RW	CRC Module reset control 0: Module is in reset state 1: Module works fine
1	FLASH	RW	FLASH Module reset control 0: Module is in reset state 1: Module works fine
0	RFU	-	Reserved bits, please keep the default value



4.7.15 SYSCTRL_APBRS1 APB Peripheral Reset Control Register 1

Address offset: 0x48 Reset value: 0x0000 03FF

Bit field	Name	Permission	Function description
31:7	RFU	-	Reserved bits, please keep the default value
6	GTIM	RW	GTIM Module reset control 0: Module is in reset state 1: Module works fine
5	ATIM	RW	ATIM Module reset control 0: Module is in reset state 1: Module works fine
4	UART2	RW	UART2 Module reset control 0: Module is in reset state 1: Module works fine
3	UART1	RW	UART1 Module reset control 0: Module is in reset state 1: Module works fine
2	SPI	RW	SPI Module reset control 0: Module is in reset state 1: Module works fine
1	VC	RW	VC Module reset control 0: Module is in reset state 1: Module works fine
0	ADC	RW	ADC and OPA Module reset control 0: Module is in reset state 1: Module works fine



4.7.16 SYSCTRL_APBRS2 APB Peripheral Reset Control Register 2

Address offset: 0x44 Reset value: 0x0000 00FF

Bit field	Name	Permission	Function description
31:8	RFU	-	Reserved bits, please keep the default value
7	LPTIM	RW	LPTIM Module reset control 0: Module is in reset state 1: Module works fine
6	I2C	RW	I2C Module reset control 0: Module is in reset state 1: Module works fine
5	WWDT	RW	WWDT Module reset control 0: Module is in reset state 1: Module works fine
4	IWDT	RW	IWDT Module reset control 0: Module is in reset state 1: Module works fine
3	RFU	-	Reserved bits, please keep the default value
2	BTIM123	RW	BTIM Module reset control 0: Module is in reset state 1: Module works fine
1	RTC	RW	RTC Module reset control 0: Module is in reset state 1: Module works fine
0	RFU	-	Reserved bits, please keep the default value



4.7.17 SYSCTRL_RESETFLAG System Reset Flag Register

Address offset: 0x4C Reset value: 0x---- ----

Bit field	Name	Permission	Function description
31:10	RFU	-	Reserved bits, please keep the default value
9	SYSRESETREQ	RW0	Cortex-M0+ CPU SYSRESETREQ soft reset flag 0: No CPU SYSRESETREQ soft reset occurred 1: CPU SYSRESETREQ soft reset has occurred Write 0 to clear, write 1 to invalid
8	LOCKUP	RW0	Cortex-M0+ CPU Lockup reset flag 0: No Lockup reset occurred 1: Lockup reset has occurred Write 0 to clear, write 1 to invalid
7	RFU	-	Reserved bits, please keep the default value
6	RSTB	RW0	NRST pin reset flag 0: No Pin reset occurred 1: Pin reset has occurred Write 0 to clear, write 1 to invalid
5	RFU	-	Reserved bits, please keep the default value
4	IWDT	RW0	IWDT reset flag 0: No IWDT reset occurred 1: IWDT reset has occurred Write 0 to clear, write 1 to invalid
3	LVD	RW0	LVD reset flag 0: No LVD reset occurred 1: LVD reset has occurred Write 0 to clear, write 1 to invalid
2:1	RFU	-	Reserved bits, please keep the default value
0	POR	RW0	POR/BOR reset flag 0: No POR/BOR reset occurred 1: POR/BOR reset has occurred Write 0 to clear, write 1 to invalid



4.7.18 SYSCTRL_DEBUG Debug Status Timer Control Register

Address offset: 0x2C Reset value: 0x0000 07E7

Bit field	Name	Permission	Function description
31:10	RFU	-	Reserved bits, please keep the default value
9	IWDT	RW	In the debugging state, IWDT count function configuration 0: Normal count 1: Pause count
8	RTC	RW	In the debugging state, RTC count function configuration 0: Normal count 1: Pause count
7	RFU	-	Reserved bits, please keep the default value
6	LPTIM	RW	In the debugging state, LPTIM count function configuration 0: Normal count 1: Pause count
5	BTIM123	RW	In the debugging state, BTIM1 / BTIM2 / BTIM3 count function configuration 0: Normal count 1: Pause count
4:2	RFU	-	Reserved bits, please keep the default value
1	GTIM	RW	In the debugging state, GTIM count function configuration 0: Normal count 1: Pause count
0	ATIM	RW	In the debugging state, ATIM count function configuration 0: Normal count 1: Pause count



4.7.19 SYSCTRL_MCO System Clock Output Control Register

Address offset: 0x70 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:7	RFU	-	Reserved bits, please keep the default value
6:4	DIV	RW	MCO output frequency division control 000: 1 Frequency Division 001: 2 Frequency Division 010: 8 Frequency Division 011: 64 Frequency Division 100: 128 Frequency Division 101: 256 Frequency Division 110: 512 Frequency Division 111: 1024 Frequency Division
3:0	SOURCE	RW	MCO output signal source configuration 0000: Zero output 0001: HCLK 0010: PCLK 0011: HSIOOSC 0100: LSI 0101: HSE 0110: LSE



5 Interrupt

5.1 Overview

Nested Vectored Interrupt Controller (NVIC) for the ARM® Cortex®-M0+ core to manage interrupts and exceptions. The NVIC is tightly coupled to the processor core, enabling low-latency exception and interrupt handling.

The processor supports up to 32 interrupt request (IRQ) inputs and supports multiple internal exceptions.

This chapter only introduces the 32 external interrupt requests (IRQ0 ~ IRQ31) of the processor. Please refer to the "ARM® Cortex®-M0+ Technical Reference Manual" and "ARM® v6-M Architecture Reference Manual" for details of the processor's internal exceptions.

5.2 Main features

- 16 inner exceptions
- 32 maskable external interrupts
- 4 programmable priorities
- Low-latency exception and interrupt handling
- Support for interrupt nesting
- Interrupt vector table remapping

5.3 Interrupt priority

The external interrupt can be set to 4 levels of priority, the highest priority is "0", the lowest priority is "3", and the default value is "0".

If a higher priority interrupt occurs while the processor is executing an interrupt handler, the interrupt is preempted. If an interrupt occurs with the same or lower priority as the interrupt being processed, the interrupt will not be preempted, but the state of the new interrupt will change to pending. If multiple pending interrupts have the same priority, the pending interrupt with the smaller interrupt number takes precedence. For example, if both IRQ [0] and IRQ [1] are pending, and both have the same priority, then IRQ [0] is processed first.



5.4 Interrupt vector table

When the ARM® Cortex®-M0+ responds to an interrupt, the processor automatically fetches the start address of the Interrupt Service Routine (ISR) from the interrupt vector table in the memory. The interrupt vector table includes the initial value of the main stack pointer (MSP), the entry address of the service routine for internal exceptions and external interrupts. Each interrupt vector occupies 1 word (4 bytes), and the storage address of the interrupt vector is the vector number multiplied by 4, as shown in the following table:

Table 5-1 Interrupt vector table

Vector number	External interrupt number (IRQ#)	Priority	Interrupt source	Introduction	Address
0	-	-	-	MSP initial value	0x0000 0000
1	-	-3	Reset	Reset vector	0x0000 0004
2	-	-2	NMI	Non-maskable interrupt ²	0x0000 0008
3	-	-1	HardFault	Hardware error exception (fault)	0x0000 000C
4-10	-	-	-	Reserved	0x0000 0010~002B
11	-	Configurable	SVCall	Hypervisors invoked via SWI instructions	0x0000 002C
12-13	-	-	-	Reserved	0x0000 0030~0037
14	-	Configurable	PendSV	Suspendable requests for system services	0x0000 0038
15	-	Configurable	SysTick	System tick timer	0x0000 003C
16	0	Configurable	WDT	Independent watchdog interrupts	0x0000 0040
17	1	Configurable	LVD	LVD global interrupt	0x0000 0044
18	2	Configurable	RTC	RTC global interrupt	0x0000 0048
19	3	Configurable	FLASHRAM ¹	FLASH/RAM global interrupt	0x0000 004C
20	4	Configurable	RCC	RCC global interrupt ³	0x0000 0050
21	5	Configurable	GPIOA	GPIOA global interrupt	0x0000 0054
22	6	Configurable	GPIOB	GPIOB global interrupt	0x0000 0058
23	7	-	-	Reserved	0x0000 005C
24	8	-	-	Reserved	0x0000 0060
25	9	-	-	Reserved	0x0000 0064
26	10	-	-	Reserved	0x0000 0068
27	11	-	-	Reserved	0x0000 006C
28	12	Configurable	ADC	ADC global interrupt	0x0000 0070
29	13	Configurable	ATIM	ATIM global interrupt	0x0000 0074



Vector number	External interrupt number (IRQ#)	Priority	Interrupt source	Introduction	Address
30	14	Configurable	VC1	VC1 global interrupt	0x0000 0078
31	15	Configurable	VC2	VC2 global interrupt	0x0000 007C
32	16	Configurable	GTIM	GTIM global interrupt	0x0000 0080
33	17	-	-	Reserved	0x0000 0084
34	18	-	-	Reserved	0x0000 0088
35	19	Configurable	LPTIM	LPTIM global interrupt	0x0000 008C
36	20	Configurable	BTIM1	BTIM1 global interrupt	0x0000 0090
37	21	Configurable	BTIM2	BTIM2 global interrupt	0x0000 0094
38	22	Configurable	BTIM3	BTIM3 global interrupt	0x0000 0098
39	23	Configurable	I2C	I2C global interrupt	0x0000 009C
40	24	-	-	Reserved	0x0000 00A0
41	25	Configurable	SPI	SPI global interrupt	0x0000 00A4
42	26	-	-	Reserved	0x0000 00A8
43	27	Configurable	UART1	UART1 global interrupt	0x0000 00AC
44	28	Configurable	UART2	UART2 global interrupt	0x0000 00B0
45	29	-	-	Reserved	0x0000 00B4
46	30	-	-	Reserved	0x0000 00B8
47	31	Configurable	FAULT	HSE/LSE operation fail interrupt ⁴	0x0000 00BC

Caution 1:

Since the interrupts of some peripherals multiplex an IRQ interrupt source, the user should first check the interrupt flag bit in the interrupt service routine to determine the peripheral that generates the interrupt.

Caution 2:

NMI is not used in CW32L010.

Caution 3:

HSE, LSE clock signal start-up failure and LSI, LSE, HSIOSC, HSE clock signal stability correspond to RCC global interrupt.

Caution 4:

The failure of the HSE or LSE clock signal during operation corresponds to a FAULT interrupt.



5.5 Interrupt related registers

5.5.1 NVIC interrupt enable and disable enable

The ARM® Cortex-M0+ processor supports up to 32 external interrupt sources, corresponding to the 32 enable bits of the interrupt enable setting register NVIC_ISER and the 32 disable bits of the interrupt enable clear register NVIC_ICER. Setting the enable bit to 1 enables interrupts; Set the disable bit to 1, disable interrupts.

The above NVIC interrupt enable is only for the processor NVIC, whether the interrupt of the peripheral is enabled is also controlled by the interrupt control register of the corresponding peripheral.

5.5.2 NVIC interrupt pending and clear pending

When an interrupt occurs, if the system is processing an interrupt of the same priority or higher, the system will not process the interrupt immediately, but will set the interrupt status to pending and save it in the interrupt pending status register; The pending state will remain valid until the processor enters this interrupt processing, unless the pending state is manually cleared. When the processor begins to enter the interrupt processing, the hardware will automatically clear the corresponding interrupt pending state.

The user can set the status of this interrupt to the pending state by setting the corresponding bit of the interrupt pending setting register NVIC_ISPR. If the system is not processing an interrupt with the same priority or higher priority, the interrupt will be immediately responded and deal with.

The user can set the state of this interrupt to the pending clear state by setting the corresponding bit of the interrupt pending clear register NVIC_ICPR.

5.5.3 NVIC interrupt priority

The interrupt priority control registers NVIC_IPR0~NVIC_IPR7 are used to set the interrupt priority of IRQ0~IRQ31. Each interrupt source uses 8 bits. In CW32L010, only the upper two bits are used, and up to 4 interrupt priorities can be set.

Caution:

The setting of the interrupt priority register of ARM® Cortex-M0+ should be before the interrupt is enabled. The user cannot change the interrupt priority after the interrupt is enabled, which will lead to unpredictable results.



5.5.4 NVIC interrupt mask

In some special occasions, all interrupts need to be disabled, which can be realized by using the interrupt mask register PRIMASK. Only the lowest 1 bit of PRIMASK is valid. If this bit is set to 1, all external interrupts and exceptions except NMI and hardware error exceptions are disabled; after clearing to 0, it is allowed to respond to interrupts and exceptions. This bit defaults to 0 after reset.

ARM® Cortex-M0+ has dedicated ARM instructions for modifying PRIMASK register, CPSIE i and CPSID i, please refer to "ARM® v6-M Architecture Reference Manual" for details.

Assembly instruction example reference:

```
CPSIE i      ; Clear PRIMASK (Enable interrupt)
CPSID i      ; Set up PRIMASK (Disable interrupts)
```

C-language (Calling the CMSIS device driver library) Example reference:

```
void __enable_irq(void);    //Clear PRIMASK
void __disable_irq(void);   //Set up PRIMASK
```

5.5.5 Peripheral interrupt enable

Peripheral modules generally have their own interrupt enable registers. When using interrupts, you must first enable the peripheral interrupt enable, and refer to [Table 5-1 Interrupt vector table](#) to enable the NVIC interrupt for the interrupt source. For the interrupt enable of specific peripherals, please refer to the description of the relevant peripheral modules.



5.6 List of registers

NVIC base address: NVIC_BASE = 0xE000 E000

Table 5-2 List of NVIC registers

Register name	Register address	Register description
NVIC_ISER	NVIC_BASE + 0x100	IRQ0~IRQ31 Interrupt Enable Setting Register
NVIC_ICER	NVIC_BASE + 0x180	IRQ0~IRQ31 Interrupt Enable Clear Register
NVIC_ISPR	NVIC_BASE + 0x200	IRQ0~IRQ31 Interrupt Pending Setup Register
NVIC_ICPR	NVIC_BASE + 0x280	IRQ0~IRQ31 Interrupt Pending Clear Register
NVIC_IPR0	NVIC_BASE + 0x400	IRQ0~IRQ3 Interrupt Priority Control Register 0
NVIC_IPR1	NVIC_BASE + 0x404	IRQ4~IRQ7 Interrupt Priority Control Register 1
NVIC_IPR2	NVIC_BASE + 0x408	IRQ8~IRQ11 Interrupt Priority Control Register 2
NVIC_IPR3	NVIC_BASE + 0x40C	IRQ12~IRQ15 Interrupt Priority Control Register 3
NVIC_IPR4	NVIC_BASE + 0x410	IRQ16~IRQ19 Interrupt Priority Control Register 4
NVIC_IPR5	NVIC_BASE + 0x414	IRQ20~IRQ23 Interrupt Priority Control Register 5
NVIC_IPR6	NVIC_BASE + 0x418	IRQ24~IRQ27 Interrupt Priority Control Register 6
NVIC_IPR7	NVIC_BASE + 0x41C	IRQ28~IRQ31 Interrupt Priority Control Register 7



5.7 Register description

See section [1 Documentation conventions](#) for abbreviations used in register descriptions.

5.7.1 NVIC_ISER Interrupt Enable Setting Register

Address offset: 0x100 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:0	SETIRQ	RW	Set to enable external interrupt IRQ0~IRQ31; write "1" to set, write "0" is invalid. [0]: IRQ0 [1]: IRQ1 [2]: IRQ2 [31]: IRQ31 The read value indicates the current interrupt enable state.

5.7.2 NVIC_ICER Interrupt Enable Clear Register

Address offset: 0x180 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:0	CLRIRQ	RW	Set to disable external interrupt IRQ0~IRQ31; write "1" to set, write "0" is invalid. [0]: IRQ0 [1]: IRQ1 [2]: IRQ2 [31]: IRQ31 The read value indicates the current interrupt enable state.

5.7.3 NVIC_ISPR Interrupt Pending Setup Register

Address offset: 0x200 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:0	SETPEND	RW	Set the pending state of external interrupt IRQ0~IRQ31; write "1" to set, write "0" is invalid. [0]: IRQ0 [1]: IRQ1 [2]: IRQ2 [31]: IRQ31 The read value indicates the current interrupt pending state.



5.7.4 NVIC_ICPR Interrupt Pending Clear Register

Address offset: 0x280 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:0	CLRPEND	RW	Clear the pending state of external interrupt IRQ0~IRQ31; write "1" to clear, write "0" is invalid. [0]: IRQ0 [1]: IRQ1 [2]: IRQ2 [31]: IRQ31 The read value indicates the current interrupt pending state.

5.7.5 NVIC_IPR0 Interrupt Priority Control Register 0

Address offset: 0x400 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:30	IPRIRQ3	RW	The priority of interrupt IRQ3, 00 has the highest priority and 11 has the lowest priority.
29:24	RFU	-	Reserved bits, please keep the default value
23:22	IPRIRQ2	RW	The priority of interrupt IRQ2, 00 has the highest priority and 11 has the lowest priority.
21:16	RFU	-	Reserved bits, please keep the default value
15:14	IPRIRQ1	RW	The priority of interrupt IRQ1, 00 has the highest priority and 11 has the lowest priority.
13:8	RFU	-	Reserved bits, please keep the default value
7:6	IPRIRQ0	RW	The priority of interrupt IRQ0, 00 has the highest priority and 11 has the lowest priority.
5:0	RFU	-	Reserved bits, please keep the default value



5.7.6 NVIC_IPR1 Interrupt Priority Control Register 1

Address offset: 0x404 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:30	IPRIRQ7	RW	The priority of interrupt IRQ7, 00 has the highest priority and 11 has the lowest priority.
29:24	RFU	-	Reserved bits, please keep the default value
23:22	IPRIRQ6	RW	The priority of interrupt IRQ6, 00 has the highest priority and 11 has the lowest priority.
21:16	RFU	-	Reserved bits, please keep the default value
15:14	IPRIRQ5	RW	The priority of interrupt IRQ5, 00 has the highest priority and 11 has the lowest priority.
13:8	RFU	-	Reserved bits, please keep the default value
7:6	IPRIRQ4	RW	The priority of interrupt IRQ4, 00 has the highest priority and 11 has the lowest priority.
5:0	RFU	-	Reserved bits, please keep the default value

5.7.7 NVIC_IPR2 Interrupt Priority Control Register 2

Address offset: 0x408 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:30	IPRIRQ11	RW	The priority of interrupt IRQ11, 00 has the highest priority and 11 has the lowest priority.
29:24	RFU	-	Reserved bits, please keep the default value
23:22	IPRIRQ10	RW	The priority of interrupt IRQ10, 00 has the highest priority and 11 has the lowest priority.
21:16	RFU	-	Reserved bits, please keep the default value
15:14	IPRIRQ9	RW	The priority of interrupt IRQ9, 00 has the highest priority and 11 has the lowest priority.
13:8	RFU	-	Reserved bits, please keep the default value
7:6	IPRIRQ8	RW	The priority of interrupt IRQ8, 00 has the highest priority and 11 has the lowest priority.
5:0	RFU	-	Reserved bits, please keep the default value



5.7.8 NVIC_IPR3 Interrupt Priority Control Register 3

Address offset: 0x40C Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:30	IPRIRQ15	RW	The priority of interrupt IRQ15, 00 has the highest priority and 11 has the lowest priority.
29:24	RFU	-	Reserved bits, please keep the default value
23:22	IPRIRQ14	RW	The priority of interrupt IRQ14, 00 has the highest priority and 11 has the lowest priority.
21:16	RFU	-	Reserved bits, please keep the default value
15:14	IPRIRQ13	RW	The priority of interrupt IRQ13, 00 has the highest priority and 11 has the lowest priority.
13:8	RFU	-	Reserved bits, please keep the default value
7:6	IPRIRQ12	RW	The priority of interrupt IRQ12, 00 has the highest priority and 11 has the lowest priority.
5:0	RFU	-	Reserved bits, please keep the default value

5.7.9 NVIC_IPR4 Interrupt Priority Control Register 4

Address offset: 0x410 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:30	IPRIRQ19	RW	The priority of interrupt IRQ19, 00 has the highest priority and 11 has the lowest priority.
29:24	RFU	-	Reserved bits, please keep the default value
23:22	IPRIRQ18	RW	The priority of interrupt IRQ18, 00 has the highest priority and 11 has the lowest priority.
21:16	RFU	-	Reserved bits, please keep the default value
15:14	IPRIRQ17	RW	The priority of interrupt IRQ17, 00 has the highest priority and 11 has the lowest priority.
13:8	RFU	-	Reserved bits, please keep the default value
7:6	IPRIRQ16	RW	The priority of interrupt IRQ16, 00 has the highest priority and 11 has the lowest priority.
5:0	RFU	-	Reserved bits, please keep the default value



5.7.10 NVIC_IPR5 Interrupt Priority Control Register 5

Address offset: 0x414 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:30	IPRIRQ23	RW	The priority of interrupt IRQ23, 00 has the highest priority and 11 has the lowest priority.
29:24	RFU	-	Reserved bits, please keep the default value
23:22	IPRIRQ22	RW	The priority of interrupt IRQ22, 00 has the highest priority and 11 has the lowest priority.
21:16	RFU	-	Reserved bits, please keep the default value
15:14	IPRIRQ21	RW	The priority of interrupt IRQ21, 00 has the highest priority and 11 has the lowest priority.
13:8	RFU	-	Reserved bits, please keep the default value
7:6	IPRIRQ20	RW	The priority of interrupt IRQ20, 00 has the highest priority and 11 has the lowest priority.
5:0	RFU	-	Reserved bits, please keep the default value

5.7.11 NVIC_IPR6 Interrupt Priority Control Register 6

Address offset: 0x418 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:30	IPRIRQ27	RW	The priority of interrupt IRQ27, 00 has the highest priority and 11 has the lowest priority.
29:24	RFU	-	Reserved bits, please keep the default value
23:22	IPRIRQ26	RW	The priority of interrupt IRQ26, 00 has the highest priority and 11 has the lowest priority.
21:16	RFU	-	Reserved bits, please keep the default value
15:14	IPRIRQ25	RW	The priority of interrupt IRQ25, 00 has the highest priority and 11 has the lowest priority.
13:8	RFU	-	Reserved bits, please keep the default value
7:6	IPRIRQ24	RW	The priority of interrupt IRQ24, 00 has the highest priority and 11 has the lowest priority.
5:0	RFU	-	Reserved bits, please keep the default value



5.7.12 NVIC_IPR7 Interrupt Priority Control Register 7

Address offset: 0x41C Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:30	IPRIRQ31	RW	The priority of interrupt IRQ31, 00 has the highest priority and 11 has the lowest priority.
29:24	RFU	-	Reserved bits, please keep the default value
23:22	IPRIRQ30	RW	The priority of interrupt IRQ30, 00 has the highest priority and 11 has the lowest priority.
21:16	RFU	-	Reserved bits, please keep the default value
15:14	IPRIRQ29	RW	The priority of interrupt IRQ29, 00 has the highest priority and 11 has the lowest priority.
13:8	RFU	-	Reserved bits, please keep the default value
7:6	IPRIRQ28	RW	The priority of interrupt IRQ28, 00 has the highest priority and 11 has the lowest priority.
5:0	RFU	-	Reserved bits, please keep the default value



6 RAM Memory

6.1 Overview

CW32L010 integrates 4KB embedded RAM for users to use, which is used to store various data in the process of program execution. The starting address of the RAM is 0x2000 0000, and the data is stored in the little-endian mode in the RAM, that is, the lowest byte address space stores the least significant byte data of the data.

6.2 Main features

- Support access in 3 widths of byte (8bit), half word (16bit) or full word (32bit)
- Zero latency, can be accessed by CPU at maximum system clock frequency
- Support parity check function



6.3 RAM Memory operation

The RAM memory operations that the user can perform include: read operations and write operations.

The read and write operations of RAM support three-bit widths of 8bit, 16bit and 32bit. The user program can complete the read and write by directly accessing the absolute address, but it should be noted that the data bit width of read and write must be aligned with the corresponding address boundary, otherwise read and write the operation is invalid and will result in a HardFault hardware error exception.

6.3.1 Read operation

The read operation supports 3 different bit widths, which can be read by direct access to the absolute address, but it should be noted that the read data bit width must be aligned with the corresponding address boundary.

Code example:

8bit read:

```
tempdata = * ( ( uint8_t * ) 0x2000 0001 );
```

16bit read:

```
tempdata = * ( ( uint16_t * ) 0x2000 0002 );
```

32bit read:

```
tempdata = * ( ( uint32_t * ) 0x2000 0004 );
```

6.3.2 Write operation

The write operation supports 3 different bit widths, and data can be written by directly accessing the absolute address, but it should be noted that the written data bit width must be aligned with the corresponding address boundary.

Code example:

8bit write:

```
* ( ( uint8_t * ) 0x2000 0001 ) = 0x12;
```

16bit write:

```
* ( ( uint16_t * ) 0x2000 0002 ) = 0x1234;
```

32bit write:

```
* ( ( uint32_t * ) 0x2000 0004 ) = 0x1234 5678.
```



6.4 Parity check function

CW32L010 supports the parity check function of RAM. After power-on, the parity check function is enabled by default and cannot be configured by the user.

Each byte of RAM data is actually stored in a 9bit physical space, including 8bit data bits and 1bit parity bits.

When the CPU writes to the RAM, the parity check unit of the RAM calculates the check digit and writes it into the corresponding check digit space. When reading RAM data, the data is read together with the check digit. The CPU performs parity check on each byte of the data, and compares the calculated check digit with the read check digit. If they are consistent, it means that the data in the RAM is correct. If it is inconsistent, the parity error flag RAM_ISR.PARITY is set. If the RAM parity error interrupt enables control bit RAM_IER.PARITY is set to 1, the CPU will respond to the interrupt service. The user program can set RAM_ICR.PARITY to 0 to clear the parity error flag.

The user can obtain the RAM address where the parity error occurs by reading the parity error address register RAM_ADDR.



6.5 List of registers

RAM base address: RAM_BASE = 0x4002 2400

Table 6-1 List of RAM registers

Register name	Register address	Register description
RAM_IER	RAM_BASE + 0x00	Interrupt Enable Control Register
RAM_ADDR	RAM_BASE + 0x04	Parity Error Address Register
RAM_ISR	RAM_BASE + 0x08	Interrupt Flag Register
RAM_ICR	RAM_BASE + 0x0C	Interrupt Flag Clear Register



6.6 Register description

See section [1 Documentation conventions](#) for abbreviations used in register descriptions.

6.6.1 RAM_ADDR Parity Error Address Register

Address offset: 0x04 Reset value: 0x2000 0000

Bit field	Name	Permission	Function description
31:0	ADDR	RO	The address of the RAM where the parity error is located

6.6.2 RAM_IER Interrupt Enable Control Register

Address offset: 0x00 Reset value: 0x0000 0001

Bit field	Name	Permission	Function description
31:1	RFU	-	Reserved bits, please keep the default value
0	PARITY	RW	RAM parity error interrupt enable control 0: Disabled 1: Enabled

6.6.3 RAM_ISR Interrupt Flag Register

Address offset: 0x08 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:1	RFU	-	Reserved bits, please keep the default value
0	PARITY	RO	Parity error flag 0: No parity error occurred 1: A parity error has occurred

6.6.4 RAM_ICR Interrupt Flag Clear Register

Address offset: 0x0C Reset value: 0x0000 0001

Bit field	Name	Permission	Function description
31:1	RFU	-	Reserved bits, please keep the default value
0	PARITY	R1W0	Parity Error Flag Clear Control W0: Clear parity error flag W1: No function



7 FLASH Memory

7.1 Overview

CW32L010 integrates 64KB embedded FLASH for users to use, which can be used to store application programs and user data. The chip supports read, erase and write operations of FLASH memory, and supports erase and write protection and read protection; the chip has a built-in high-voltage BOOST circuit required for FLASH programming, and no additional programming voltage is required.

7.2 Main features

- Support access in 3 widths of byte (8bit), half word (16bit) or full word (32bit)
- Supports read, erase and write operations
- Support erase and write protection, read protection
- Supports security runtime library protection

7.3 FLASH Memory organization

CW32L010 integrates 64KB user-accessible FLASH memory, which is managed by paging of 512 bytes per page, with a total of 128 pages. Users can perform full-page erasing and byte-by-byte programming operations on FLASH.

In addition to the user-accessible 64KB FLASH memory, the CW32L010 also integrates a 2KB boot program memory, which solidified with BootLoader and other codes when it leaves the factory, and the user cannot access this memory.

7.4 FLASH memory read wait cycle configuration

The internal FLASH memory of CW32L010 supports the operation clock with the fastest frequency of 24MHz. When the configured HCLK frequency is greater than 24MHz, the number of inserted waiting HCLK cycles needs to be configured through the WAIT bit field of the FLASH control register FLASH_CR2. The configuration rules are shown in the following table:

Table 7-1 FLASH read wait cycle configuration

FLASH_CR2.WAIT	The number of HCLK cycles waiting for the instruction fetch cycle	HCLK Frequency
000	0	HCLK $\leq$ 24MHz
001	1	HCLK $\leq$ 48MHz

Caution:

When configuring the FLASH read wait period, the FLASH configuration clock must be enabled first.



7.5 FLASH Memory operation

User-executable FLASH memory operations include: read operations, erase operations, and write (programming) operations.

The read and write operations of FLASH support 3 widths of 8bit, 16bit and 32bit. The user program can complete the read and write by directly accessing the absolute address, but it should be noted that the data bit width of read and write must be aligned with the corresponding address boundary, otherwise it will cause HardFault hardware error exception.



7.5.1 Page Erase

The minimum unit of the page erase operation of FLASH is 1 page, that is, 512 bytes. After the page erase operation is completed, the data content of all address spaces of the page is 0xFF.

If the page erase operation is performed on the unlocked FLASH page, the operation will fail, and the FLASH_ISR.PAGELOCK flag will be set by hardware. If the FLASH_IER.PAGELOCK is set to 1, the CPU will execute the corresponding interrupt service routine. User can clear the FLASH_ISR.PAGELOCK interrupt flag by setting FLASH_ICR.PAGELOCK to 0.

If the program runs in FLASH, and the page erase operation is performed on the storage space of the page where the PC (program pointer) is located, the operation will fail, and the FLASH_ISR.PC flag will be set by hardware. If FLASH_IER.PC is set to 1, then The CPU will execute the corresponding interrupt service routine. The user can clear the FLASH_ISR.PC interrupt flag by setting FLASH_ICR.PC to 0.

FLASH page erase operation steps are as follows:

Step 1: Set the FLASH operation mode to page erase mode, and write 0x02 to the working mode bit field MODE of the FLASH control register FLASH_CR1;

Caution:

The FLASH_CR1 register has the KEY protection feature. The upper 16bit data of the written data must be 0x5A5A, otherwise it cannot be written.

Step 2: To unlock the erasure protection of the page to be erased, see [Table 7-2 FLASH_PAGELOCK Erase lock protection](#), write 1 to the lock bit field LOCKx corresponding to the erasing lock register FLASH_PAGELOCK;

Caution:

The FLASH_PAGELOCK register has the KEY protection feature. The upper 16bit data of the written data must be 0x5A5A, otherwise it cannot be written.

Step 3: Write any data to any address in the page to be erased, triggering the erase operation of the page.

Caution:

The write data bit width needs to be aligned with the corresponding address boundary.

Code example:

```
*( ( uint8_t * ) 0x0000 0001 ) = 0x55;
```

Step 4: Wait for the erase to complete, and query the FLASH busy flag bit FLASH_ISR.BUSY becomes 0;

Step 5: To erase other pages, repeat steps 2 to 4;

Step 6: After the erase operation is completed, if you want to restore the lock protection of FLASH, you need to write 0 to the lock bit field LOCKx corresponding to the erase/write lock register FLASH_PAGELOCK to activate the lock protection function to protect the FLASH content from being erased by mistake;

Step 7: Set the FLASH operation mode to read mode, and write 0x00 to the working mode bit field MODE of the FLASH control register FLASH_CR1.



7.5.2 Write operation

Based on the characteristics of embedded FLASH, the write operation can only rewrite the bit data in the FLASH memory from '1' to '0', but cannot be rewritten from '0' to '1'. Therefore, before writing data, the page where the corresponding address is located must be erased.

The following three principles must be followed for the FLASH write operation:

- Cannot write to the address whose data bit content is '0'.
- Cannot write to the address in the locked area.
- Cannot write to the address of the page where the PC (program pointer) is located.

When the CW32L010 performs a write operation, it will check whether the contents of the storage space are all '1'. FLASH has three write operation bit widths of 8bit, 16bit and 32bit, and the number of bytes to be checked is 1Byte, 2Byte and 4Byte respectively.

If the internal storage space is not all '1', the write operation will fail, and the FLASH_ISR.PROG flag will be set by hardware. If FLASH_IER.PROG is set to 1, the CPU will execute the corresponding interrupt service routine. User can clear the FLASH_ISR.PROG interrupt flag by setting FLASH_ICR.PROG to 0.

If the storage space of the unlocked FLASH page is written, the operation will fail, and the FLASH_ISR.PAGELOCK flag will be set by hardware. If FLASH_IER.PAGELOCK is set to 1, the CPU will execute the corresponding interrupt service routine. User can clear the FLASH_ISR.PAGELOCK interrupt flag by setting FLASH_ICR.PAGELOCK to 0.

If the program runs in FLASH, and the write operation is performed on the storage space of the page where the PC (program pointer) is located, the operation will fail, and the FLASH_ISR.PC flag will be set by hardware. If FLASH_IER.PC is set to 1, the CPU will Execute the corresponding interrupt service routine. The user can clear the FLASH_ISR.PC interrupt flag by setting FLASH_ICR.PC to 0.



The FLASH write operation steps are as follows:

Step 1: Set the FLASH operation mode to write (programming) mode, and write 0x01 to the working mode bit field MODE of the FLASH control register FLASH_CR1;

Caution:

The FLASH_CR1 register has the KEY protection feature, the high 16bit data of the written data must be 0x5A5A, otherwise it cannot be written.

Step 2: To unlock the erasure protection of the page to be written, see [Table 7-2 FLASH_PAGELOCK Erase lock protection](#), write 1 to the lock bit field LOCKx corresponding to the erasing lock register FLASH_PAGELOCK;

Caution:

The FLASH_PAGELOCK register has the KEY protection feature. The upper 16bit data of the written data must be 0x5A5A, otherwise it cannot be written.

Step 3: Write the data to be written into the target address and trigger the FLASH write operation. Note that the bit width of the written data needs to be aligned with the corresponding address boundary;
8bit width code example:

```
*( ( uint8_t * ) 0x0000 0001 ) = 0x55;
```

16bit width code example:

```
*( ( uint16_t * ) 0x0000 0002 ) = 0x1234;
```

32bit width code example:

```
*( ( uint32_t * ) 0x0000 0004 ) = 0x1234 5678;
```

Step 4: Wait for the programming to be completed, query and wait for the FLASH busy flag bit FLASH_ISR.BUSY to become 0;

Step 5: If you need to write more data to the address range of the currently unlocked area, repeat steps 4 to 5; if the write address exceeds the address range of the current unlocked area, repeat steps 3 to 5;

Step 6: After the write operation is completed, if you need to restore the lock protection of the FLASH, write 0 to the lock bit field LOCKx corresponding to the erase/write lock register FLASH_PAGELOCK to activate the lock protection function to protect the FLASH content from being erased by mistake;

Step 7: Set the FLASH operation mode to read mode, and write 0x00 to the working mode bit field MODE of the FLASH control register FLASH_CR1.



7.5.3 Read operation

CW32L010 supports 3 different bit widths for the FLASH read operation, which can be read by direct access to the absolute address. Note that the read data bit width must be aligned with the corresponding address boundary.

8bit width code example:

```
tempdata = * ( ( uint8_t * ) 0x0000 0001 );
```

16bit width code example:

```
tempdata = * ( ( uint16_t * ) 0x0000 0002 );
```

32bit width code example:

```
tempdata = * ( ( uint32_t * ) 0x0000 0004 );
```



7.6 FLASH Memory protection

FLASH memory has erase and write protection and read protection functions.

The erasure protection includes lock page erasure protection and PC address page erasure protection. Pages in the protected state cannot be erased and written, which can avoid accidental rewriting of FLASH content.

The read protection takes the entire FLASH as the protection object, and does not support single-page protection, which can prevent the user code from being illegally read.

7.6.1 Erase and write protection

CW32L010 realizes the erasing and writing locking of FLASH pages by setting the erasing and writing lock register FLASH_PAGELOCK. Pages in the locked state cannot be page erased or written.

The internal FLASH memory of CW32L010 is divided into 128 pages, and each 8 pages corresponds to a LOCKx lock bit of the FLASH_PAGELOCK register. The LOCKx field has a total of 16 bits, which can realize the lock protection of all 128 pages of FLASH memory. The correspondence between the fields of the erasing lock register FLASH_PAGELOCK and the FLASH lock page is shown in the following table:

Table 7-2 FLASH_PAGELOCK Erase lock protection

FLASH_PAGELOCK bit field	Bit field name	Lock page
15	LOCK15	Page120 – Page127
14	LOCK14	Page112 – Page119
13	LOCK13	Page104- Page111
12	LOCK12	Page96 – Page103
11	LOCK11	Page88 – Page95
10	LOCK10	Page80 – Page87
9	LOCK9	Page72 – Page79
8	LOCK8	Page64- Page71
7	LOCK7	Page56 - Page63
6	LOCK6	Page48 - Page55
5	LOCK5	Page40 - Page47
4	LOCK4	Page32 - Page39
3	LOCK3	Page24 - Page31
2	LOCK2	Page16 - Page23
1	LOCK1	Page8 - Page15
0	LOCK0	Page0 - Page7



When performing page erasing and writing operations on FLASH, you must first unlock the corresponding page by setting the corresponding FLASH_PAGELOCK.LOCKx bit field to 1. An example of the unlock operation is as follows:

Lock Page0-Page7 code example:

```
CW_FLASH->PAGELOCK = 0x5A5A 0000 | ( CW_FLASH->PAGELOCK & 0x0000 FFFE );
```

Unlock Page32-Page39 code example:

```
CW_FLASH->PAGELOCK = 0x5A5A 0000 | ( CW_FLASH->PAGELOCK | 0x0000 0010 );
```

Caution:

The FLASH_PAGELOCK register has the KEY protection feature. The upper 16bit data of the written data must be 0x5A5A, otherwise it cannot be written.

If the page erase or write operation is performed directly on the unlocked FLASH page, the operation will fail and an interrupt flag will be generated. Please refer to section [7.5.1 Page Erase](#) and section [7.5.2 Write operation](#).

7.6.2 Erase and write PC page protection

The FLASH memory of CW32L010 supports the function of erasing and writing PC page protection.

When the user program runs FLASH, if the current program pointer PC is just within the range of the FLASH address page to be erased, the erase operation will fail, and the FLASH_ISR.PC flag will be set by hardware. If FLASH_IER.PC is set to 1, the CPU will execute the corresponding interrupt service routine. The user can clear the FLASH_ISR.PC interrupt flag by setting FLASH_ICR.PC to 0.



7.6.3 Read protection

The CW32L010 supports the FLASH read protection function. After the read protection is set, the FLASH cannot be read by ISP or SWD. Read protection only supports whole-chip FLASH protection, and does not support page-by-page protection.

The read protection is divided into 4 protection levels. The current protection level can be obtained by reading the security bit field SECURITY of the FLASH control register FLASH_CR1. The security bit field is a read-only attribute and cannot be modified. The read protection level can be set by the ISP command, please refer to the ISP programming documentation.

The CPU fetches instructions from FLASH and the program's read operation to FLASH is not affected by the FLASH read protection function.

The read protection function of FLASH is shown in the following table:

Table 7-3 FLASH read protection

Protection level	FLASH_CR1.SECURITY	Function description
Level0	00	Read protection is not set. The FLASH can be read by SWD or ISP.
Level1	01	FLASH content cannot be read by SWD or ISP. The protection level can be reduced to Level 0 through ISP or SWD. After the downgrade, the content in the FLASH is all 0xFF, that is, it is in the empty chip state of the whole chip erasing.
Level2	10	FLASH content cannot be read by SWD or ISP. The protection level can only be lowered by ISP, but after the downgrade, the content in the FLASH is all 0xFF, that is, it is in an empty chip state of whole chip erasing.
Level3	11	FLASH content cannot be read by SWD or ISP. Both ISP and SWD downgrade functions are disabled. At this protection level, the chip can only be programmed once.



7.6.4 Security Runtime Library Protection

The FLASH memory of CW32L010 supports the security runtime library function, which allows the solution provider to store the core algorithm in the security runtime library area for customers to call during secondary development.

After enabling the safe runtime library protection function, the user code can call the function in the safe runtime library area normally, but cannot read out the content of the safe runtime library area by any means (CPU, SWD, ISP), and the safe runtime library protection function can be disabled through the ISP protocol and the data in the safe runtime library storage space can be cleared at the same time only when a correct password is provided.

The method to enable the Security Runtime Vault Protection function is as follows: Write a specific control word at address 0xFFFF0~0xFFFF as shown in the table below:

Table 7-4 Enable security runtime library protection function control word

Address	0xFFFF0	0xFFFF1	0xFFFF2	0xFFFF3	0xFFFF4	0xFFFF5	0xFFFF6	0xFFFF7
Control word	SlibKey0	SlibKey1	SlibKey2	SlibKey3	SlibKey4	SlibKey5	SlibKey6	SlibKey7
Address	0xFFFF8	0xFFFF9	0xFFFFA	0xFFFFB	0xFFFFC	0xFFFFD	0xFFFFE	0xFFFFF
Control word	StartIdx	EndIdx	0x5A	0x5A	0xA5	0xA5	CRCL	CRCH

The meaning of each control word is as follows:

- SlibKey0~SlibKey7 is the password of the security runtime library, which is used to prohibit the protection function of the security runtime library, users should set it according to the actual demand and keep it in a safe place. Do not set the passwords to all FF and all 00.
- StartIdx is the start page number of the security runtime library area, users should set it according to the actual situation, after writing, users can read it out through the START bit field of the FLASH_SDKCFR register.
- EndIdx is the end page number of the security runtime library area, fixed at 0x7F, after writing, the user can read out through the END bit field of the FLASH_SDKCFR register.
- CRCL is the lower part of the result of CRC16_X25 algorithm for 0xFFFF0~0xFFFFD data.
- CRCH is the high byte of the result of the CRC16_X25 algorithm for 0xFFFF0~0xFFFFD data.

There are two ways to disable the security runtime library protection function.

1. Execute chip erase operation through ISP protocol: when the chip receives a chip erase instruction and a correct security runtime library password, it will automatically erase all the data of the chip and disable the security runtime library protection function; when the chip receives a chip erase instruction and an incorrect security runtime library password, no operation is performed, the data in all areas are not erased, and the safe runtime library protection function is still enabled. Refer to the ISP programming documentation for instructions.
2. Perform chip erase operation in RAM: Configure the compiler parameters so that the chip erase operation is performed from RAM. After the erase operation, all the data of the chip is erased and the security runtime library protection function is disabled.



7.7 FLASH memory programming

CW32L010 internal integrated FLASH memory supports FLASH erasing and programming through SWD or ISP. The SWD method is to use the debugging tool to erase and program the FLASH through the SWD interface. The ISP method is to erase and program the FLASH through the BootLoader code pre-installed before the chip leaves the factory. ISP provides fast and efficient in-circuit erasing and programming methods, please refer to ISP programming documentation.



7.8 Precautions

In order to operate FLASH correctly and improve the access efficiency and service life of FLASH, users need to pay attention to the following matters when programming applications:

- Address Alignment Requirements

The address boundary is aligned, that is, the address when accessing FLASH with a 16-bit width must be an even address, and when using a 32-bit width, the address must be an address that is a multiple of 4.

Code example for correct address alignment:

8bit read:

```
tempdata = * ( ( uint8_t * ) 0x0000 0001 );
```

16bit read:

```
tempdata = * ( ( uint16_t * ) 0x0000 0002 );
```

32bit read:

```
tempdata = * ( ( uint32_t * ) 0x0000 0004 );
```

Code example with wrong address alignment:

16bit read:

```
tempdata = * ( ( uint16_t * ) 0x0000 0001 );
```

32bit read:

```
tempdata = * ( ( uint32_t * ) 0x0000 0003 );
```

- Operation complete flag query

When the CPU fetches instructions from FLASH and runs, if the page erase/write operation to FLASH is performed, the CPU will automatically stop the next instruction access, and the hardware will automatically wait for the erase and write operation to complete (the FLASH_ISR.BUSY status bit becomes 0), so the user program does not need to loop through the query operation completion flag to determine whether the operation is completed.

When the CPU fetches instructions from RAM and runs, if a page erase/write operation to FLASH is performed, the CPU will access the next instruction while performing the operation. In order to ensure the correct execution of the next instruction of the program, The user program must query the FLASH_ISR.BUSY flag cyclically after the FLASH erase and write operations, until the FLASH_ISR.BUSY flag becomes 0, and then the subsequent tasks can be performed.

Before entering DeepSleep mode, if the FLASH is performing an erase/write operation, you must wait for the FLASH_ISR.BUSY flag bit to clear to 0, and make sure FLASH_CR1.MODE is 0 at the same time.

- Service life

Based on the characteristics of embedded FLASH, the number of operations and storage time of FLASH are limited. Users should try to avoid frequent erasing and writing operations on the FLASH memory of a certain page or address in the application program to ensure reliable data storage. Please refer to the datasheet for specific life data.

- Data storage mode

The application of CW32L010 stipulates that data is stored in FLASH in little-endian mode, that is, the least significant byte data of the data is stored in the lowest byte address space.



7.9 List of registers

FLASH base address: FLASH_BASE = 0x4002 2000

Table 7-5 List of FLASH registers

Register name	Register address	Register description
FLASH_CR1	FLASH_BASE + 0x00	Control register 1
FLASH_CR2	FLASH_BASE + 0x04	Control register 2
FLASH_PAGELOCK	FLASH_BASE + 0x08	Erase lock register
FLASH_IER	FLASH_BASE + 0x20	Interrupt Enable Register
FLASH_ISR	FLASH_BASE + 0x24	Interrupt Flag Register
FLASH_ICR	FLASH_BASE + 0x28	Interrupt Flag Clear Register
FLASH_SDKCFR	FLASH_BASE + 0x70	Security Runtime Library Area Register



7.10 Register description

See section [1 Documentation conventions](#) for abbreviations used in register descriptions.

7.10.1 FLASH_CR1 Control register 1

Address offset: 0x00 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	KEY	WO	Only when KEY is 0x5A5A, the write operation of this register is valid
15:7	RFU	-	Reserved bits, please keep the default value
6:5	SECURITY	RO	Current protection level 00: Level0, ISP can read and write, SWD can read and write 01: Level1, ISP downgradable, SWD downgradable; Data cannot be read 10: Level2, ISP downgradable, SWD has no function; Data cannot be read 11: Level3, ISP has no function, SWD has no function; Data cannot be read
4:2	RFU	-	Reserved bits, please keep the default value
1:0	MODE	RW	Operation Mode Configuration 00: Read mode, Read 01: Write mode, Program 10: Page erase mode, PageErase 11: Chip erase mode, ChipErase

7.10.2 FLASH_CR2 Control register 2

Address offset: 0x04 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	KEY	WO	Only when KEY is 0x5A5A, the write operation of this register is valid
15:3	RFU	-	Reserved bits, please keep the default value
2:0	WAIT	RW	FLASH instruction fetches cycle configuration 000: 1 HCLK cycle for HCLK ≤ 24MHz 001: 2 HCLK cycle for 24MHz < HCLK ≤ 48MHz Caution: This control bit has the same function as SYSCTRL_CR2[6:4].



7.10.3 FLASH_PAGELOCK Erase lock register

Address offset: 0x08 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	KEY	WO	Only when KEY is 0x5A5A, the write operation of this register is valid
15	LOCK15	RW	Page120 – Page127 erase lock configuration 0: Locked, non-rewritable 1: Open, rewritable
14	LOCK14	RW	Page112 – Page119 erase lock configuration 0: Locked, non-rewritable 1: Open, rewritable
13	LOCK13	RW	Page104 – Page111 erase lock configuration 0: Locked, non-rewritable 1: Open, rewritable
12	LOCK12	RW	Page96 – Page103 erase lock configuration 0: Locked, non-rewritable 1: Open, rewritable
11	LOCK11	RW	Page88 – Page95 erase lock configuration 0: Locked, non-rewritable 1: Open, rewritable
10	LOCK10	RW	Page80 – Page87 erase lock configuration 0: Locked, non-rewritable 1: Open, rewritable
9	LOCK9	RW	Page72 – Page79 erase lock configuration 0: Locked, non-rewritable 1: Open, rewritable
8	LOCK8	RW	Page64 – Page71 erase lock configuration 0: Locked, non-rewritable 1: Open, rewritable
7	LOCK7	RW	Page56 – Page63 erase lock configuration 0: Locked, non-rewritable 1: Open, rewritable
6	LOCK6	RW	Page48 – Page55 erase lock configuration 0: Locked, non-rewritable 1: Open, rewritable
5	LOCK5	RW	Page40 – Page47 erase lock configuration 0: Locked, non-rewritable 1: Open, rewritable
4	LOCK4	RW	Page32 – Page39 erase lock configuration 0: Locked, non-rewritable 1: Open, rewritable



Bit field	Name	Permission	Function description
3	LOCK3	RW	Page24 – Page31 erase lock configuration 0: Locked, non-rewritable 1: Open, rewritable
2	LOCK2	RW	Page16 – Page23 erase lock configuration 0: Locked, non-rewritable 1: Open, rewritable
1	LOCK1	RW	Page8 – Page15 erase lock configuration 0: Locked, non-rewritable 1: Open, rewritable
0	LOCK0	RW	Page0 – Page7 erase lock configuration 0: Locked, non-rewritable 1: Open, rewritable

7.10.4 FLASH_IER Interrupt Enable Register

Address offset: 0x20 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:5	RFU	-	Reserved bits, please keep the default value
4	PROG	RW	Programming Error Interrupt Enable Control 0: Disabled 1: Enabled
3:2	RFU	-	Reserved bits, please keep the default value
1	PAGELOCK	RW	Erase and write PAGELOCK locked page interrupt enable control 0: Disabled 1: Enabled
0	PC	RW	Erase and write the page where the PC is located, interrupt enable control 0: Disabled 1: Enabled



7.10.5 FLASH_ISR Interrupt Flag Register

Address offset: 0x24 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:6	RFU	-	Reserved bits, please keep the default value
5	BUSY	RO	Erase and write status flag 0: Erase and write operation completed 1: Erase and write operation not completed
4	PROG	RO	Programming error flag 0: The content of each byte of the current address to be written is all 0xFF 1: The content of each byte of the current address to be written is not all 0xFF
3:2	RFU	-	Reserved bits, please keep the default value
1	PAGELOCK	RO	Erase and write PAGELOCK locked page interrupt flag 0: The current erase address is outside the page locked by PAGELOCK 1: The current erase address is within the page locked by PAGELOCK
0	PC	RO	Erase and write the interrupt flag of the page where the PC pointer is located 0: The current erase address is outside the page where the PC pointer is located 1: The current erase address is within the page where the PC pointer is located



7.10.6 FLASH_ICR Interrupt Flag Clear Register

Address offset: 0x28 Reset value: 0x0000 001F

Bit field	Name	Permission	Function description
31:5	RFU	-	Reserved bits, please keep the default value
4	PROG	WO	Programming error flag clear W0: Clear programming error flag W1: No function
3:2	RFU	-	Reserved bits, please keep the default value
1	PAGELOCK	R1W0	Erase and write PAGELOCK locked page interrupt flag clear W0: Clear the page interrupt flag locked by erasing PAGELOCK W1: No function
0	PC	R1W0	Erase and write the page where the PC pointer is located to clear the interrupt flag W0: Clear and erase the interrupt flag of the page where the PC pointer is located W1: No function

7.10.7 FLASH_SDKCFR Security Runtime Library Area Register

Address offset: 0x70 Reset value: 0x0000 003F

Bit field	Name	Permission	Function description
31:15	RFU	-	Reserved bits, please keep the default value
14:8	END	RO	The Flash Page where the end address of the Security Runtime Library area is indexed from 0 to 127.
7	RFU	-	Reserved bits, please keep the default value
6:0	START	RO	The Flash Page where the start address of the Security Runtime Library area is indexed from 0 to 127.



8 General-purpose input/output (GPIO)

8.1 Overview

The GPIO controller realizes the connection between various digital and analog circuits inside the chip and the physical pins.

GPIO can be configured as digital input and output and analog functions, supports peripheral function multiplexing, supports 2 interrupt sources of rising edge and falling edge, can wake up the MCU return to active mode through external interrupt in DeepSleep mode.

8.2 Main features

- All registers are read and written through the AHB bus interface
- With digital input and output and analog function
- Digital input and output support common GPIO and function multiplexing
- The analog function can be used as the input signal of ADC, VC, LVD
- Support a variety of internal clock signal output
- Digital input supports internal pull-up and high-impedance modes
- Digital outputs support push-pull and open-drain modes
- The digital output supports atomic bit operations of bit setting, bit clearing, and bit flipping
- Interrupt function supports rising edge, falling edge trigger mode
- Interrupts with digital filtering, selectable from 6 clock sources
- Supports wake-up of MCU via external interrupt in DeepSleep mode

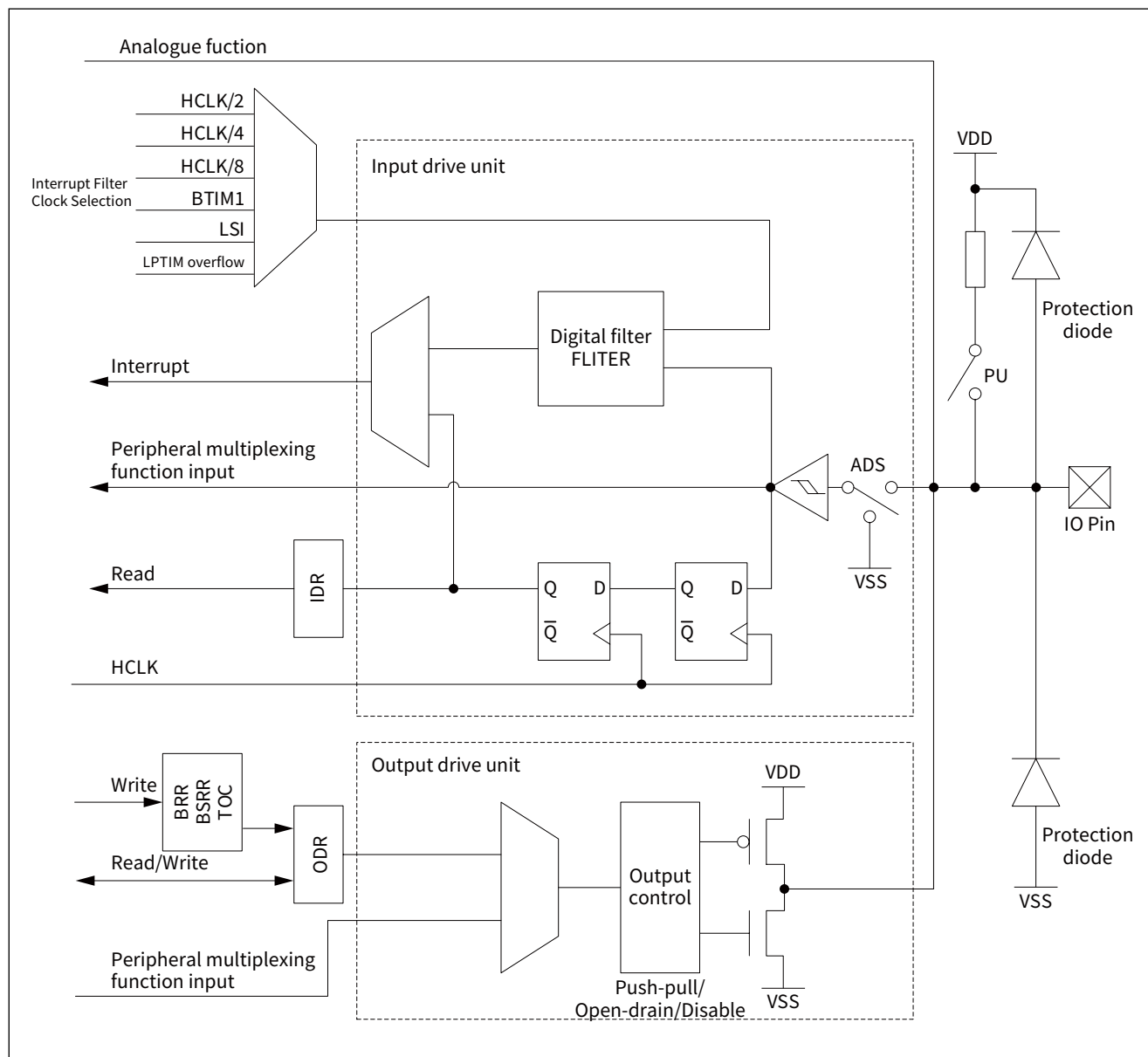


8.3 Function description

8.3.1 Functional block diagram

The functional block diagram of the GPIO controller is shown in the following figure:

Figure 8-1 GPIO functional block diagram



8.3.2 Digital output

Clear the analog-digital configuration register GPIOx_ANALOG[y] (x is the GPIO port number, x=A, B; y is the pin number, y=0~8; the same below), and configure the corresponding GPIO port as Digital function; clear the input and output direction register GPIOx_DIR[y] to configure the GPIO port as output mode. Digital output signal sources can be:

- Output data register GPIOx_ODR
- On-chip digital peripherals

Configure the output mode through the output mode register GPIOx_OPENDRAIN, which can select push-pull output or open-drain output.



8.3.3 Digital input

Clear the analog-digital configuration register `GPIOx_ANALOG[y]` to configure the GPIO port as a digital function; set the input and output direction register `GPIOx_DIR[y]` to configure the corresponding GPIO port as input mode. Digital input signals are configurable:

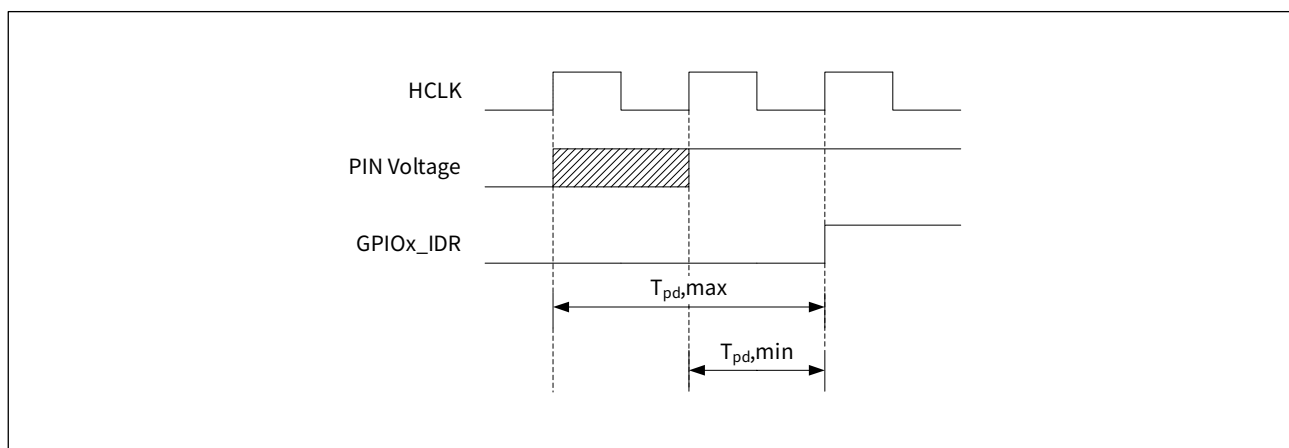
- Reach the input data register `GPIOx_IDR`
- Reach on-chip digital peripherals
- Trigger interrupt

In this mode, the digital input signal is directed to the internal digital input circuit through the ADS switch.

After the level state is confirmed by the Schmitt trigger, it can be directly sent to the input of the digital peripheral pointed to by the on-chip multiplexing function, or through an HCLK-based synchronizer, present on the input data register `GPIOx_IDR[y]`.

The bits of the `GPIOx_IDR` register and the previous latch form a synchronizer, which can avoid the signal instability caused by the pin level jumping during the system clock change, but it will cause a certain read delay. The synchronization timing of the read port pins is shown in the following figure:

Figure 8-2 Read Port Pin Synchronization Timing



In the clock cycle after the rising edge of the system clock, the pin level signal will be latched in the internal register, as shown in the shaded part in the figure, after the next rising edge of the system clock, the stable pin level signal is read, and then a on the rising edge of the system clock, the data is latched into the `GPIOx_IDR` register. The signal delay T_{pd} is 1~2 system clocks.

The built-in hardware filter circuit can also be enabled if this input signal is considered for triggering an interrupt. The filter circuit is implemented based on a double D flip-flop synchronizer, which has 6 clock sources, some of which are specific to the low-power mode. For example, a key debounce operation without software intervention can be easily implemented. For specific clock source options and edge trigger options, see section [8.3.6 Interrupt Function](#).

In this mode, the internal pull-up function can be individually selected to be turned on or off through the pull-up resistor register `GPIOx_PUR`.

8.3.4 Analogue function

For a GPIO port configured with analog function, you can open the GPIO analog signal channel by setting the analog digital configuration register GPIOx_ANALOG[y] to 1. When the GPIO analog signal channel is turned on, the digital function of the port is turned off, the internal pull-up is disconnected, the internal digital input signal is shorted to VSS through the ADS switch, and the internal digital output function is disabled.

8.3.5 Multiplexing function

The multiplexing function of the input and output ports can be realized through the multiplexing function registers (GPIOx_AFRH and GPIOx_AFRL). Each 4-bit bit field in the multiplexing function register corresponds to the multiplexing function selection of a GPIO port, and up to 16 input and output signal targets can logically be selected. The specific definition of GPIO multiplexing function is shown in the following table:

Table 8-1 GPIO Multiplexing function settings

GPIOx_AFRL[4×y]	Multiplexing function	GPIOx_AFRH[4×(z-8)]	Multiplexing function
0000	GPIO	0000	GPIO
0001	AF1	0001	AF1
0010	AF2	0010	AF2
0011	AF3	0011	AF3
0100	AF4	0100	AF4
0101	AF5	0101	AF5
0110	AF6	0110	AF6
0111	AF7	0111	AF7

Caution:

y, z are pin numbers: y=0~7 corresponding to GPIOx_AFRL; z=8~15 corresponding to GPIOx_AFRH.

The specific peripheral functions corresponding to each AF are shown in the following table:

Table 8-2 GPIO multiplexing function assignment table

Pin name	Multiplexing function							
	AF0	AF1	AF2	AF3	AF4	AF5	AF6	AF7
PA00	GPIO	UART1_RXD	I2C_SDA	SPI_SCK	BTIM1_TOGP	BTIM3_ETR	ATIM_BK2	ATIM_CH4
PA01	GPIO	UART1_TXD	I2C_SCL	SPI_CS	BTIM1_TOGN	GTIM_CH3	ATIM_BK	ATIM_CH4N
PA02	GPIO	PCLK_OUT		SPI_CS	VC1_OUT	IR_OUT	ATIM_ETR	ATIM_CH1N
PA03	GPIO	UART2_TXD	LPTIM_CH1	SPI_MISO	BTIM1_ETR	IR_OUT	GTIM_CH4	ATIM_CH3
PA04	GPIO	UART2_RXD	LPTIM_CH2	SPI_MOSI	MCO_OUT	VC2_OUT	GTIM_CH3	ATIM_CH1N
PA05	GPIO	UART1_RXD	LPTIM_OUT	SPI_SCK	I2C_SDA	HEXEN	GTIM_CH2	ATIM_CH2N
PA06	GPIO	UART1_TXD	LPTIM_ETR	SPI_MOSI	I2C_SCL	BTIM2_ETR	GTIM_CH1	ATIM_CH3N
PA07	GPIO	UART1_RXD	I2C_SDA	SPI_MISO	LPTIM_OUT	BTIM2_TOGP	ATIM_BK	ATIM_CH6



Pin name	Multiplexing function							
	AF0	AF1	AF2	AF3	AF4	AF5	AF6	AF7
PA08	GPIO	UART1_TXD	I2C_SCL	MCO_OUT	VC1_OUT	BTIM2_TOGN	ATIM_BK2	ATIM_CH6N
PB00	GPIO	UART1_RXD	UART2_CTS	SPI_SCK	I2C_SDA	LVD_OUT	BTIM3_TOGP	ATIM_CH1
PB01	GPIO	UART1_TXD	UART2_RTS	SPI_CS	I2C_SCL	ADC_SAM	BTIM3_TOGN	ATIM_CH2
PB02	GPIO	UART2_TXD	UART1_CTS	SPI_MISO	VC2_OUT	LPTIM_CH1	GTIM_ETR	ATIM_CH2
PB03	GPIO	UART2_RXD	UART1_RTS	SPI_MOSI	MCO_OUT	BTIM2_TOGN	GTIM_CH4	ATIM_CH2N
PB04	GPIO	UART2_TXD	RTC_OUT	ADC_SAM	BTIM1_ETR	BTIM2_TOGP	GTIM_CH3	ATIM_CH1
PB05	GPIO	UART2_TXD	RTC_TAMP	SPI_MISO	I2C_SDA	GTIM_ETR	GTIM_CH2	ATIM_CH5
PB06	GPIO	UART2_RXD	RTC_OUT	SPI_MOSI	I2C_SCL	ATIM_ETR	GTIM_CH1	ATIM_CH5N



8.3.6 Interrupt Function

When each GPIO is set to digital input mode, it can be used as an external interrupt signal source, and the interrupt signal source can be set to two types: rising edge, and falling edge. Interrupt triggering methods can be used in combination, but share the same interrupt flag bit.

After the interrupt is triggered, the corresponding bit of the interrupt flag register GPIOx_ISR will be set by hardware, and the program can determine which port generated the interrupt by querying GPIOx_ISR. Through the interrupt flag clear register GPIOx_ICR[y], the corresponding interrupt flag bit can be cleared.

The internal interrupt digital filter can digitally filter the input signal on the pin, providing 6 filter clock options, as shown in the following table:

Table 8-3 Digital Filter Clock Selection

GPIOx_FILTER.FLTCLK	Digital Filter Clock Frequency
000	HCLK / 2
001	HCLK / 4
010	HCLK / 8
011	BTIM1 overflow
101	LSI (About 32.8kHz)
111	LPTIM overflow

Due to the wide range of selected filter clock cycles, users can easily implement flexible input interrupt debounce functions. Changes in input levels that are not maintained for more than one full filter clock cycle will not be communicated through the hardware filter to the internal interrupt trigger circuitry. Changes in input levels that persist for more than two complete filter clock cycles must pass through the hardware filter.

For the edge trigger type, considering the sensitivity to the time of the trigger edge, it is recommended to disable the hardware filter function in the interrupt digital filter configuration register GPIOx_FILTER[y], because the hardware filter can improve signal stability while also inserting certain delay.

When CW32L010 works in Sleep mode or DeepSleep mode, the external interrupt function of GPIO can still be used. When an external interrupt is generated, the chip can be woken up from Sleep mode or DeepSleep mode to Active mode.

Caution:

The same group of GPIOx.PINy shares a hardware filter clock source selection register, so the same group of GPIOs can only filter input signal jitter with the same filter clock.



8.3.7 Other functions

Atomic bit operation

The GPIO controller supports bit-setting, bit-clearing and bit-flipping functions.

Writing 1 to the GPIO bit clear register GPIOx_BSRR[y] or the bit clear register GPIOx_BRR[y] will directly change the state of the corresponding bit of the output data register GPIOx_ODR, thereby indirectly affecting the final output level without affecting the Status of other bits in this register.

Writing 1 to the GPIO bit toggle register GPIOx_TOG[y] will toggle the level state of the output port.

Port reset state

After power-on or reset, SWCLK (PA08) and SWDIO (PA07) default to digital pull-up. Other ports are analog high-resistance input by default, and pull-up is not turned on by default.



8.4 Programming example

When configuring the GPIO port, you must first set SYSCTRL_AHBEN.GPIOx to 1, enable the corresponding GPIO configuration clock and working clock.

8.4.1 Digital output programming example

Step 1: Set GPIOx_ANALOG.PINy to 0 to configure the port as a digital function;

Step 2: Set GPIOx_DIR.PINy to 0, configure the port as output;

Step 3: Configure the GPIOx_OPENDRAIN register to set the port output mode;

Step 4: Configure the GPIOx_ODR register to set the port output level.

8.4.2 Digital input programming example

Step 1: Set GPIOx_ANALOG.PINy to 0, configure the port as a digital function;

Step 2: Set GPIOx_DIR.PINy to 1, configure the port as input;

Step 3: Configure the GPIOx_PUR register and select whether to enable the internal pull-up resistor;

Step 4: Read the GPIOx_IDR register and read the port input level.

8.4.3 Analog functional programming example

Step 1: Set GPIOx_ANALOG.PINy to 1 to configure the port for analog function.

8.4.4 Multiplexed function programming example

Step 1: Configure the port as digital output or digital input according to application requirements. For specific register configuration steps, please refer to section [8.4.1 Digital output programming example](#) and section [8.4.2 Digital input programming example](#);

Step 2: Configure the GPIOx_AFRH or GPIOx_AFRL register to set the port multiplexing function, see [Table 8-2 GPIO multiplexing function assignment table](#).



8.4.5 Interrupt function programming example

- Step 1: Configure the port as a digital input. For the specific register configuration steps, see section [8.4.2 Digital input programming example](#);
- Step 2: Configure GPIOx_FILTER.FLTCLK, and select the port interrupt filter clock;
- Step 3: Set GPIOx_FILTER.PINy to 1 to enable the corresponding port filter clock;
- Step 4: Configure the NVIC controller, see chapter [5 Interrupt](#);
- Step 5: Configure the GPIOx_RISEIE, GPIOx_FALLIE registers according to the application requirements, and select the GPIO interrupt trigger mode;
- Step 6: The port interrupt input signal triggers the GPIO interrupt, execute the interrupt service function.



8.5 List of registers

GPIOA base address: GPIOA_BASE = 0x4800 0000

GPIOB base address: GPIOB_BASE = 0x4800 0100

GPIOC base address: GPIOC_BASE = 0x4800 0200

Table 8-4 List of GPIO registers

Register name	Register address	Register description
GPIOx_DIR	GPIOx_BASE + 0x00	GPIOx I/O direction register
GPIOx_OPENDRAIN	GPIOx_BASE + 0x04	GPIOx output mode register
GPIOx_PUR	GPIOx_BASE + 0x10	GPIOx pull-up Resistor Register
GPIOx_AFRH	GPIOx_BASE + 0x14	GPIOx Multiplexing function register high segment
GPIOx_AFRL	GPIOx_BASE + 0x18	GPIOx Multiplexing function register low segment
GPIOx_ANALOG	GPIOx_BASE + 0x1C	GPIOx Analog Digital Configuration Register
GPIOx_RISEIE	GPIOx_BASE + 0x24	GPIOx Rising edge interrupt enable register
GPIOx_FALLIE	GPIOx_BASE + 0x28	GPIOx Falling edge interrupt enable register
GPIOx_ISR	GPIOx_BASE + 0x34	GPIOx Interrupt flag register
GPIOx_ICR	GPIOx_BASE + 0x38	GPIOx Interrupt flag clear register
GPIOx_FILTER	GPIOx_BASE + 0x40	GPIOx Interrupt digital filter configuration register
GPIOx_IDR	GPIOx_BASE + 0x50	GPIOx input data register
GPIOx_ODR	GPIOx_BASE + 0x54	GPIOx output data register
GPIOx_BRR	GPIOx_BASE + 0x58	GPIOx bit clear register
GPIOx_BSRR	GPIOx_BASE + 0x5C	GPIOx bit-setting clear register
GPIOx_TOG	GPIOx_BASE + 0x60	GPIOx bit flip register



8.6 Register description

See section [1 Documentation conventions](#) for abbreviations used in register descriptions.

8.6.1 GPIOx_DIR GPIO I/O direction register (x =A, B)

Address offset: 0x00 Reset value: 0x0000 01FF(GPIOA)

0x0000 00FF(GPIOB)

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	PINy y=0 ~ 15	RW	Port input and output direction control 0: Configure the port as output 1: Configure the port as input

8.6.2 GPIOx_OPENDRAIN GPIO output mode register (x =A, B)

Address offset: 0x04 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	PINy y=0 ~ 15	RW	Port output mode control bit 0: Push-pull output 1: Open-drain output



8.6.3 GPIOx_PUR GPIO Pull-up resistor register (x =A, B)

Address offset: 0x10 Reset value: 0x0000 000C(GPIOA)

0x0000 0080(GPIOB)

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	PINy y=0 ~ 15	RW	Port pull-up resistor enable control 0: Disable pull-up resistors 1: Enable pull-up resistors

8.6.4 GPIOx_AFRH GPIO Multiplexing function register high segment (x =A, B)

Address offset: 0x14 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:28 27:24 23:20 19:16 15:12 11:8 7:4 3:0	AFRy y = 8 ~ 15	RW	Port digital multiplexing function control 0000: GPIO 0001: AF1 0010: AF2 0011: AF3 0100: AF4 0101: AF5 0110: AF6 0111: AF7

8.6.5 GPIOx_AFRL GPIO Multiplexing function register low segment (x =A, B)

Address offset: 0x18 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:28 27:24 23:20 19:16 15:12 11:8 7:4 3:0	AFRy y = 0 ~ 7	RW	Port digital multiplexing function control 0000: GPIO 0001: AF1 0010: AF2 0011: AF3 0100: AF4 0101: AF5 0110: AF6 0111: AF7



8.6.6 GPIOx_ANALOG GPIO Analog Digital Configuration Register (x =A, B)

Address offset: 0x1C Reset value: 0x0000 01FF(GPIOA)

0x0000 007F(GPIOB)

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	PINy y=0 ~ 15	RW	Port analog/digital function configuration 0: Configure the port as a digital function 1: Configure the port as an analog function

8.6.7 GPIOx_RISEIE GPIO Rising edge interrupt enable register (x =A, B)

Address offset: 0x24 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	PINy y=0 ~ 15	RW	Port rising edge interrupt enable control 0: Disable the rising edge interrupt of the corresponding port 1: Enable the rising edge interrupt of the corresponding port

8.6.8 GPIOx_FALLIE GPIO Falling edge interrupt enable register (x =A, B)

Address offset: 0x28 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	PINy y=0 ~ 15	RW	Port falling edge interrupt enable control 0: Disable the falling edge interrupt of the corresponding port 1: Enable the falling edge interrupt of the corresponding port



8.6.9 GPIOx_ISR GPIO Interrupt flag register (x =A, B)

Address offset: 0x34 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	PINy y=0 ~ 15	RO	Port interrupt status flag 0: Enabled interrupt not detected 1: Enabled interrupt has been detected

8.6.10 GPIOx_ICR GPIO Interrupt flag clear register (x =A, B)

Address offset: 0x38 Reset value: 0x0000 01FF(GPIOA)
0x0000 00FF(GPIOB)

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	PINy y=0 ~ 15	R1W0	Port interrupt status flag clear W0: Clear the corresponding interrupt flag bit W1: No function

8.6.11 GPIOx_FILTER GPIO Interrupt digital filter configuration register (x =A, B)

Address offset: 0x40 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:19	RFU	-	Reserved bits, please keep the default value
18:16	FLTCLK	RW	Port interrupt filter clock selection 000: HCLK / 2 001: HCLK / 4 010: HCLK / 8 011: BTIM1 overflow 101: LSI (About 32.8KHz) 111: LPTIM overflow
15:0	PINy y=0 ~ 15	RW	Port filter clock enable control 0: Disable the corresponding port to filter the clock 1: Enable the corresponding port to filter the clock <i>Caution: Filter out pulses with a width less than the FLTCLK clock width.</i>

Caution 1:

If you need the hardware filter function to work properly in DeepSleep mode, set the filter clock to the overflow signal of LSI or LPTIM.



8.6.12 GPIOx_IDR GPIO input data register (x =A, B)

Address offset: 0x50 Reset value: 0x0000 0180(GPIOA)

0x0000 0080(GPIOB)

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	PINy y=0 ~ 15	RO	Read port input level status 0: The port is low level 1: The port is high level

8.6.13 GPIOx_ODR GPIO output data register (x =A, B)

Address offset: 0x54 Reset value: 0x---- ----

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	PINy y=0 ~ 15	RW	Set the port output level 0: Set port output low level 1: Set port output high level

8.6.14 GPIOx_BRR GPIO port bit set clear register (x =A, B)

Address offset: 0x58 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	BRRy y=0 ~ 15	R0W1	Port bit clear control 0: Does not affect the corresponding bits of the GPIOx_ODR register 1: Set the corresponding bit of the GPIOx_ODR register to 0



8.6.15 GPIOx_BSRR GPIO port bit-setting clear register (x =A, B)

Address offset: 0x5C Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	BRRy y=0 ~ 15	R0W1	Port bit-setting clear control 0: Does not affect the corresponding bits of the GPIOx_ODR register 1: Set the corresponding bit of the GPIOx_ODR register to 0
15:0	BSSy y=0 ~ 15	R0W1	Port bit-setting control 0: Does not affect the corresponding bits of the GPIOx_ODR register 1: Set the corresponding bit of the GPIOx_ODR register to 1 <i>Caution: When BRRy and BSSy are both set 1, BSSy has higher priority.</i>

8.6.16 GPIOx_TOG GPIO Port Bit Flip Register (x =A, B)

Address offset: 0x60 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	PINy y=0 ~ 15	R0W1	Port Bit Flip Control 0: Does not affect the corresponding bits of the GPIOx_ODR register 1: Invert the corresponding bit of the GPIOx_ODR register



9 Cyclic redundancy check (CRC)

9.1 Overview

Cyclic Redundancy Check (CRC) is mainly used to verify the correctness and integrity of data transmission or data storage. The CW32L010 integrates a CRC calculation unit, which supports the use of multiple CRC algorithms to perform CRC calculation on the input data.

9.2 Main features

- 8bit input data bit width
- Two kinds of polynomials
CRC-16 polynomial 1: $x^{16} + x^{15} + x^2 + 1$
CRC-16 polynomial 2: $x^{16} + x^{12} + x^5 + 1$
- Eight kinds of commonly used algorithms
Combination based on polynomial, initial value, result XOR value, input/output inversion



9.3 Functional description

The CRC unit obtains the CRC calculation result by performing a 'division' operation on the input data (or the inversion of the input data) and the selected polynomial value, and then inverting or not inverting the obtained remainder, as well as XOR processing.

Before the CRC unit is used, it is necessary to set SYSCTRL_AHBEN.CRC to 1, and turn on the configuration clock and working clock of the CRC unit, which are generally set during system initialization.

9.3.1 Algorithmic modes

The CRC unit of CW32L010 supports multiple algorithm modes. Different CRC algorithms have different parameters such as polynomial, initial value, input data inversion, output data inversion, and result XOR value, as shown in the following table:

Table 9-1 CRC algorithm modes

Algorithm name	Polynomial value	Initial value	Input inversion	Output inversion	Result XOR value
CRC16_IBM	0x8005	0x0000	True	True	0x0000
CRC16_MAXIM	0x8005	0x0000	True	True	0xFFFF
CRC16_USB	0x8005	0xFFFF	True	True	0xFFFF
CRC16_MODBUS	0x8005	0xFFFF	True	True	0x0000
CRC16_CCITT	0x1021	0x0000	True	True	0x0000
CRC16_CCITT_False	0x1021	0xFFFF	False	False	0x0000
CRC16_X25	0x1021	0xFFFF	True	True	0xFFFF
CRC16_XMODEM	0x1021	0x0000	False	False	0x0000



The meaning of each parameter is as follows:

- Polynomial value
The polynomial is the description of the code group, such as CRC-16 polynomial 2: $x^{16} + x^{12} + x^5 + 1$, the corresponding code group is 1 0001 0000 0010 0001. Because the highest bit of the polynomial code group is fixed to 1, and the position of the highest bit is known, the code group after removing the highest bit 1 is generally called a polynomial value. For example, the value of CRC-16 polynomial 2 is 0001 0000 0010 0001, that is 0x1021.
- Initial value
The initial value of the CRC register before calculating the CRC check value.
- Input data inversion
That is, before the calculation starts, the high and low order bits of the data that need to calculate the CRC check value are reversed, such as the data bit 1011, which is 1101 after the inversion.
- Output data inversion
After the CRC calculation is completed, before the XOR value is performed with the result, the high and low order bits of the calculated value are reversed. If the calculation result is 1011, it will be 1101 after inversion.
- Result XOR value
After the CRC calculation is completed, the obtained CRC calculation value is XORed with the result XOR value to obtain the final CRC check value.



9.3.2 Input data bit width

The CRC calculation unit supports 8bit input data bit width. When inputting data, input the low byte first and then the high byte.

For example, the user needs to calculate the CRC check value of the group of data 0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, and the writing sequence is 0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77

Code example:

```
CW_CRC -> DR = 0x00 ;  
CW_CRC -> DR = 0x11 ;  
CW_CRC -> DR = 0x22 ;  
CW_CRC -> DR = 0x33 ;  
CW_CRC -> DR = 0x44 ;  
CW_CRC -> DR = 0x55 ;  
CW_CRC -> DR = 0x66 ;  
CW_CRC -> DR = 0x77 ;
```



9.4 Programming examples

9.4.1 CRC16_CCITT algorithm mode

Step 1: Set CRC_CR.MODE to 0x04, select the CRC16_CCITT algorithm mode, and the initial value of the CRC register is automatically configured by hardware to 0x0000;

Step 2: Write the raw data to be encoded into the data register CRC_DR in turn, and the input data bit width can be selected from 8bit, 16bit, and 32bit. See the code example in section [9.3.2 Input data bit width](#);

Step 3: Read CRC_RESULT to get the CRC check value.

Code example:

```
tempdata = CW_CRC -> RESULT ;
```



9.5 List of registers

CRC base address: CRC_BASE = 0x4002 3000

Table 9-2 List of CRC registers

Register name	Register address	Register description
CRC_CR	CRC_BASE + 0x00	Control register
CRC_DR	CRC_BASE + 0x08	Data register
CRC_RESULT	CRC_BASE + 0x0C	Result register



9.6 Register descriptions

See section [1 Documentation conventions](#) for abbreviations used in register descriptions.

9.6.1 CRC_CR control register

Address offset: 0x00 Reset value: 0x0000 0004

Bit field	Name	Permission	Function description
31:4	RFU	-	Reserved bits, please keep the default value
3:0	MODE	RW	CRC algorithm mode configuration 0000: CRC16_IBM 0001: CRC16_MAXIM 0010: CRC16_USB 0011: CRC16_MODBUS 0100: CRC16_CCITT 0101: CRC16_CCITT_False 0110: CRC16_X25 0111: CRC16_XMODEM

9.6.2 CRC_DR data register

Address offset: 0x08 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:8	RFU	-	Reserved bits, please keep the default value
7:0	DR	RW	Data write register, bit width 8bit

9.6.3 CRC_RESULT result register

Address offset: 0x0C Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	RESULT	RO	CRC calculation result



10 Real time clock (RTC)

10.1 Overview

The Real time clock (RTC) is a dedicated counter/timer that provides calendar information including hours, minutes, seconds, subseconds, day, month, year, and week.

RTC has two independent alarm clocks, the time and date can be set in combination, and the alarm clock interrupt can be generated and output through the pin; it supports the time stamp function, which can be triggered by the pin, record the current date and time, and generate a time stamp interrupt at the same time; Support periodic interrupt; support automatic wake-up function, which can generate interrupts and output through pins; support 1Hz square wave and RTCOUT output functions; support internal clock calibration compensation.

The CW32L010 has a built-in independently calibrated RC clock source with a frequency of 32KHz to provide the drive clock for the RTC. The RTC can run in DeepSleep mode and is suitable for applications requiring low power consumption.

10.2 Main features

- Calendar function (BCD code format)
 - Subseconds, seconds, minutes, hours (support 12/24 time system)
 - Week, day, month and year, support automatic leap year correction
- Interrupt function
 - Alarm clock A, B interrupt, can be output through pin
 - Cycle interruption, wide range: 0.5s to 1 month
 - Automatic wake-up interrupt, range: 61μs ~ 145h, can be output through pins
 - Timestamp interruption
- Calibration compensation function
 - Minimum clock error compensation step size, 0.119ppm
 - 1Hz square wave output pin
- Register lock protection to prevent accidental modification
- Operates in Active mode, Sleep mode, and DeepSleep mode

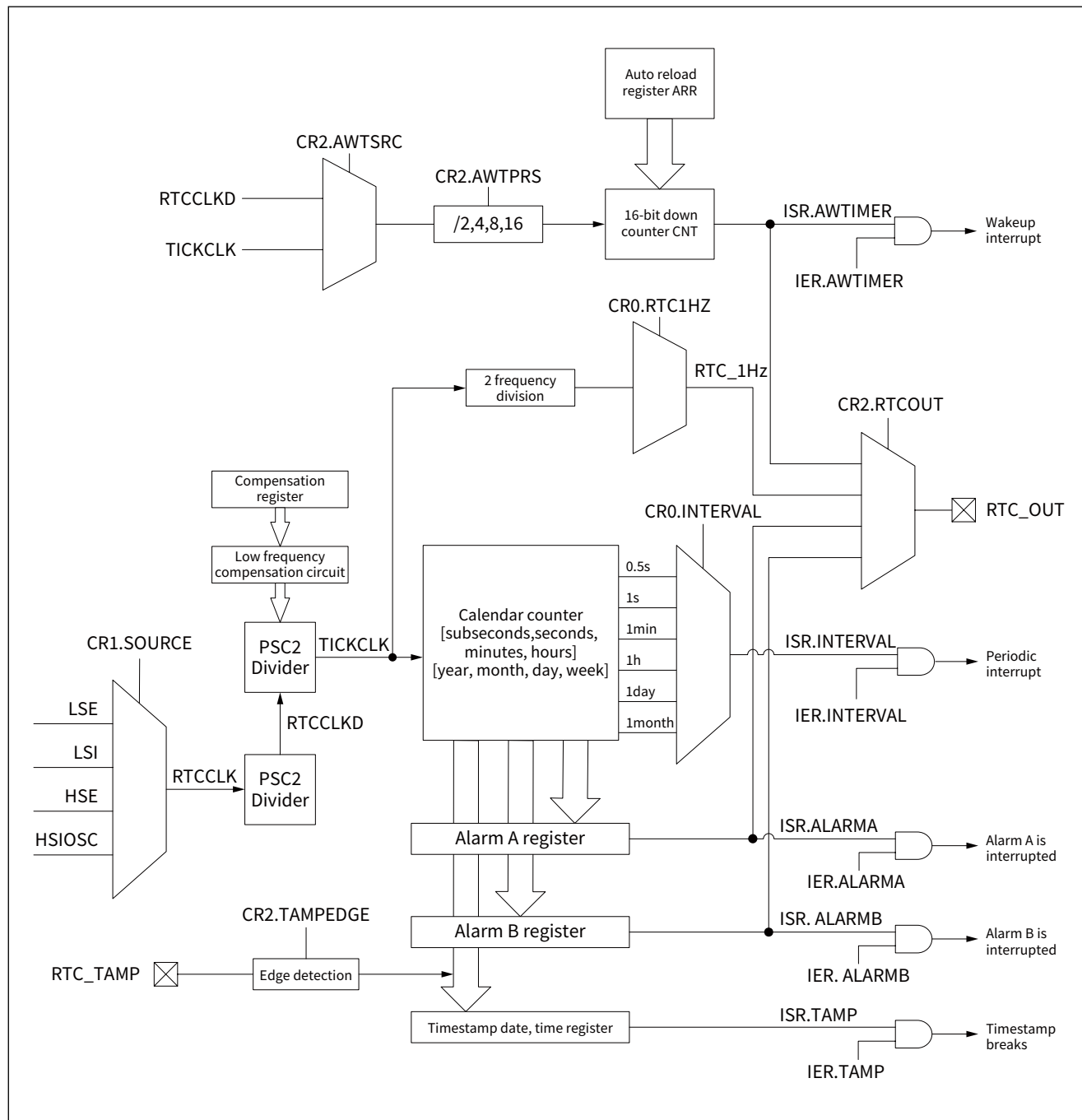


10.3 Functional description

10.3.1 Functional block diagram

The functional block diagram of the real-time clock (RTC) is shown in the following figure:

Figure 10-1 RTC functional block diagram



10.3.2 Clock and prescaler

The real-time clock (RTC) consists mainly of a dedicated high-precision RTC timer with clock source RTCCLK selectable from external low-speed clock LSE, internal low-speed clock LSI, external high-speed clock HSE and internal high-speed clock HSIOSC, selected through the SOURCE bit field of control register RTC_CR1.

The prescaler is divided into 2 programmable prescalers: PSC1 and PSC2.

The clock after dividing by PSC1 is RTCCLKD, which can be used as the clock source of RTC auto wake-up timer, and the clock frequency after maximum dividing by PSC1 should be less than or equal to 1MHz. the calculation formula is as follows:

$$f_{\text{RTCCLKD}} = f_{\text{RTCCLK}} / (\text{PSC1} + 1)$$

The clock divided by PSC2 is TICKCLK, which is used to update the calendar and also as a clock source for the RTC auto wake-up timer. The calculation formula is as follows:

$$f_{\text{TICKCLK}} = f_{\text{RTCCLKD}} / (\text{PSC2} + 1)$$

The user needs to configure PSC1 and PSC2 appropriately so that $f_{\text{TICKCLK}}=2\text{Hz}$.

The default value of PSC1 is 0 and the default value of PSC2 is 0x3FFF, so when the RTCCLK clock source is LSE, i.e. the clock frequency is 32768Hz, the timing clock is 1s.



10.3.3 Real Time Clock and Calendar

The time register RTC_TIME and the date register RTC_DATE respectively record the current time and date value in BCD code format, and will automatically check the validity when writing them, any illegal time or date value will not be written, such as the 32 days, 24 hours, 61 seconds, 13 months, etc.

In the date register RTC_DATE, the YEAR bit field represents the year, the valid value is 0 to 99; the MONTH bit field represents the month, the valid value is 1 to 12; the DAY bit field represents the day, the valid value is 1 to 31; the WEEK bit field represents the week, the valid value is 0 ~6, where 0 means Sunday and 1 to 6 means Monday to Saturday.

In the time register RTC_TIME, the SECOND bit field represents seconds, with a valid value of 0 to 59; the MINUTE bit field represents minutes, with a valid value of 0 to 59; the HOUR bit field represents hours, with a valid value of 1 to 12 or 0 to 23, and 12 hours can be selected. system or 24-hour format. The H24 bit field of the control register RTC_CR0 selects 12 or 24 hours:

- When H24 is '1', select 24-hour clock
- When H24 is '0', 12 o'clock is selected, and the highest bit of the HOUR bit field represents AM/PM:
 - '0' means AM
 - '1' means PM

The HOUR bit field value is detailed in the following table:

Table 10-1 HOUR bit field values in 12/24 time system

12 time system (RTC_CR0.H24=0)				24 time system (RTC_CR0.H24=1)			
Time	HOUR bit field	Time	HOUR bit field	Time	HOUR bit field	Time	HOUR bit field
AM12 o'clock	0x12	PM12 o'clock	0x32	00 o'clock	0x00	12 o'clock	0x12
AM1 o'clock	0x01	PM1 o'clock	0x21	01 o'clock	0x01	13 o'clock	0x13
AM2 o'clock	0x02	PM2 o'clock	0x22	02 o'clock	0x02	14 o'clock	0x14
AM3 o'clock	0x03	PM3 o'clock	0x23	03 o'clock	0x03	15 o'clock	0x15
AM4 o'clock	0x04	PM4 o'clock	0x24	04 o'clock	0x04	16 o'clock	0x16
AM5 o'clock	0x05	PM5 o'clock	0x25	05 o'clock	0x05	17 o'clock	0x17
AM6 o'clock	0x06	PM6 o'clock	0x26	06 o'clock	0x06	18 o'clock	0x18
AM7 o'clock	0x07	PM7 o'clock	0x27	07 o'clock	0x07	19 o'clock	0x19
AM8 o'clock	0x08	PM8 o'clock	0x28	08 o'clock	0x08	20 o'clock	0x20
AM9 o'clock	0x09	PM9 o'clock	0x29	09 o'clock	0x09	21 o'clock	0x21
AM10 o'clock	0x10	PM10 o'clock	0x30	10 o'clock	0x10	22 o'clock	0x22
AM11 o'clock	0x11	PM11 o'clock	0x31	11 o'clock	0x11	23 o'clock	0x23



10.3.4 RTC initialization settings

Step 1: Write 0xCA and 0x53 to the RTC_KEY register sequentially to unlock the RTC register;

Step 2: Configure RTC_CR0.H24 bit field, select 12/24 time system;

Step 3: Configure the RTC_CR1.SOURCE bit field and select the RTC clock source;

Caution:

If you select LSE or LSI, you need to enable and start LSE or LSI first, and wait for the clock to stabilize.

Step 4: Set the correct time and date values, write the time and date registers;

Step 5: Configure the required periodic interrupt unit, automatic wake-up unit, alarm clock A, and alarm clock B;

Step 6: Configure the RTC_IER register, set the periodic interrupt, automatic wake-up interrupt, alarm A, alarm B interrupt;

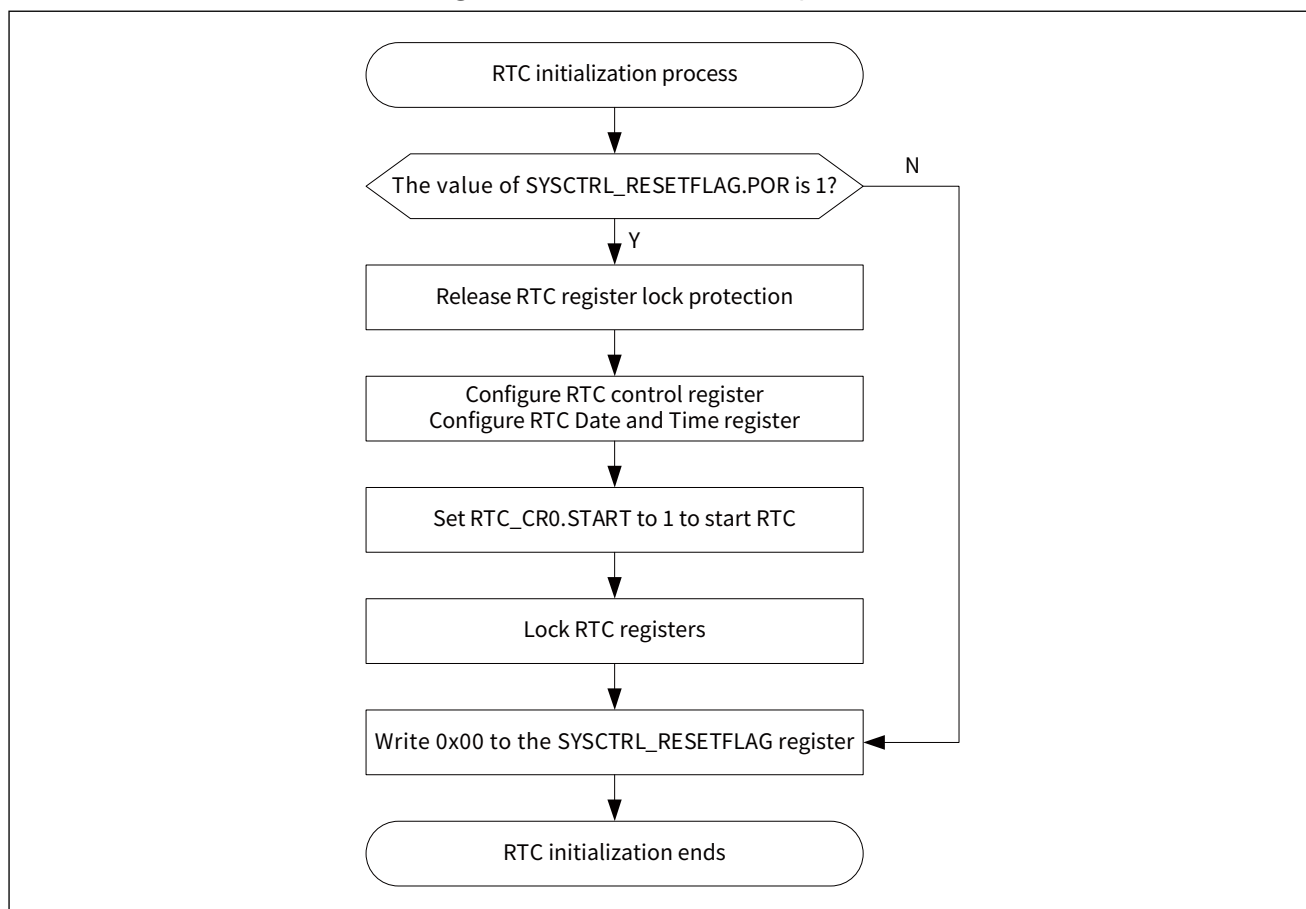
Step 7: Configure the time stamp function and configure the time stamp interruption;

Step 8: The RTC_CR0.START bit field is set to 1 to start the RTC timer;

Step 9: Write 0xCA, 0x-- to the RTC_KEY register sequentially to start the RTC register lock protection.

In user applications, when initializing the RTC, the reset state (SYSCTRL_RESETFLAG) of the MCU should be checked first. If it is a power-on reset (SYSCTRL_RESETFLAG.POR=1), the RTC registers will be reset to their default values. It is necessary to reconfigure the RTC control registers, RTC date and time registers, and restart the RTC; other reset conditions will not reset the RTC registers, the RTC continues to time normally.

Figure 10-2 RTC initialization process



10.3.5 Register lock function

After the CW32L010 is powered on and reset, the RTC registers except RTC_KEY and RTC_ICR are in lock protection state and cannot be modified. Write 0xCA and 0x53 keywords to the key value register RTC_KEY in sequence to release the RTC register lock protection; write 0xCA and 0x-- to the RTC_KEY register in sequence, the RTC register will be locked.

10.3.6 Register access operations

When the RTC is not running, after unlocking the registers, all registers can be read and written directly without going through the ACCESS access mechanism.

When the RTC is running, after unlocking the registers, all registers except timing related registers (RTC_DATE, RTC_TIME, RTC_AWTARR) can be read and written directly without going through the ACCESS access mechanism.

When the RTC is running, since the running clock of the RTC counter may be out of sync with the clock of the RTC control register, in order to ensure that the contents of the RTC_DATE, RTC_TIME and RTC_AWTARR registers can be accessed correctly without affecting the operation of the RTC, it should be ensured that the RTC timing-related registers are accessed in such a way that their timing accuracy is not affected by the access operations by the cooperative use of the WINDOW bit field and the ACCESS bit field of the control register RTC_CR1.

To access the RTC timing-related registers when the RTC is already running, the following operating specifications must be followed:

Step 1: The query waits for RTC_CR1.WAIT to be 0, indicating that loading is completed and can be read or written;

Step 2: Write 0xCA and 0x53 to the RTC_KEY register sequentially to unlock the RTC register;

Step 3: Write 1 to ACCESS to set up the access flag;

Step 4: The query waits for the WINDOW flag to be set to 1;

Step 5: Execute normal RTC register access operation;

Caution:

This step can implement multiple RTC register access instructions, but the total execution time cannot exceed 1s.

Step 6: Write 0 to ACCESS to clear the access flag;

Step 7: Write 0xCA, 0x-- to the RTC_KEY register sequentially to enable the write protection of the RTC register.

Caution:

If the user needs to read the RTC time and date information more quickly, he can choose to read the time or date register twice in a row. As long as the content of the two reads is the same, the read data can be considered valid.

When the RTC is running, since the count clock of the RTC module is not synchronized with the read/write clocks of the registers, reading the sub-second count value register RTC_SSCNT requires following specific operational steps to ensure that the correct sub-second value is read. The following is an example of reading a sub-second moment:

Step 1: Read the sub-second count value from the RTC_SSCNT register and store it into the variable SS1;

Step 2: Read the sub-second count value from the RTC_SSCNT register again and deposit it into the variable SS2;

Step 3: If SS1 and SS2 are not equal, jump to step 1;

Step 4: If SS1[19:0] is 0, skip to step 1.



10.3.7 RTCOUT output

By setting the RTCOUT bit field of the control register RTC_CR2, the RTC_OUT output source can be selected, as shown in the following table:

Table 10-2 RTC_OUT output configuration

RTC_CR2.RTCOUT	RTC_OUT output
00	RTC_1Hz signal
01	Alarm clock A match flag
10	Alarm clock B match flag
11	Wake-up timer overflow flag

10.3.8 1Hz signal output

Configure the RTC_1Hz output signal via the RTC1HZ bit field of control register RTC_CR0 as shown in the following table:

Table 10-3 RTC_1Hz output configuration

	RTC_CR0.RTC1HZ	RTC_1Hz output
Disable error compensation (RTC_COMPEN.EN=0)	01 or 11	The 1Hz signal obtained by 2-division frequency signal of TICKCLK without clock compensation.
Enable error compensation (RTC_COMPEN.EN=1)	01	The 1Hz signal obtained by 2-division frequency signal of TICKCLK with clock compensation.

When RTC_COMPCFR1.EN is set to 0 and RTC_CR0.RTC1HZ is 0x1, the signal output from RTC_1Hz is the unlocked compensated 2-division frequency signal of TICKCLK, which is identical to the RTC's second counting clock. The duration of each cycle of this clock signal is identical.

When RTC_COMPCFR1.EN is set to 1 and RTC_CR0.RTC1HZ is 0x1, the signal output from RTC_1Hz is the clock-compensated 2-division frequency signal of TICKCLK, which is identical to the RTC's second counting clock. The number of RTCCLKD clocks contained in each cycle of this clock signal is two integers in the neighbourhood of the value of the expression $(PSC2+1) \times 2 \pm COMP/N$, see [10.3.9 Clock error compensation](#).



10.3.9 Clock error compensation

The RTC has a built-in clock compensation circuit. When the RTCCLK accuracy cannot meet the requirements, the RTC clock can be compensated to achieve higher precision RTC timing.

The RTC clock error compensation configuration register, RTC_COMPCFR1, has three bit fields for implementing error compensation:

- SIGN, compensation direction
 - When the TICKCLK clock frequency is greater than 2Hz, SIGN is set to 0, increasing the count value and compensating downwards.
 - When the TICKCLK clock frequency is less than 2Hz, SIGN is set to 1, decreasing the count value and compensating upwards.
- PERIOD, compensation period, 32, 128, 256 seconds can be selected as a compensation period
 - Set PERIOD to 00, select 32 seconds as one compensation period, which can achieve fast calibration but with lower accuracy, the compensation step is 0.950ppm and the compensation accuracy is 0.475ppm;
 - Set PERIOD to 01, select 128 seconds as one compensation period, the compensation step is 0.238ppm and the compensation accuracy is 0.119ppm;
 - Set PERIOD to 10, select 256 seconds as one compensation period, the adjustment process is the smoothest, the calibration accuracy is the highest, the compensation step is 0.119ppm and the compensation accuracy is 0.060ppm.
- COMP, compensation value, the clock can be compensated range of $\pm 488.0\text{ppm}$
 - When the compensation period is 32 seconds, the valid range of COMP is 1 ~ 511;
 - When the compensation period is 128 seconds, the valid range of COMP is 1 ~ 2047;
 - When the compensation period is 256 seconds, the valid range of COMP is 1 ~ 4095.

The operation steps of clock calibration are as follows:

Step 1: Turn off the error compensation function and set RTC_COMPCFR1.EN to 0;

Step 2: According to [Table 8-2 GPIO multiplexing function assignment table](#), configure a GPIO port as RTC_OUT output, and set RTC_CR2.RTCOUT to 0 to output uncompensated 1Hz signal;

Step 3: According to section [10.3.4 RTC initialization settings](#), configure LSE as the RTC clock source;

Step 4: The RTC_CR0.START bit field is set to 1 to start the RTC;

Step 5: Use a high-precision frequency meter to measure the 1Hz signal output by the RTC_OUT pin, record the actual measurement frequency f_0 ;

The calculation method of the compensation value COMP:

Assuming the measured value $f_0 = 0.9999888\text{Hz}$, PSC2=0x3FFF, the actual frequency of the RTCCLKD clock f_z :

$$f_z = (\text{PSC2}+1) \times 2 \times 0.9999888 \approx 32767.63\text{Hz}$$

$$\text{Frequency error } \Delta f = (\text{PSC2}+1) \times 2 - f_z = 32768 - 32767.63 = 0.37$$

Select a compensation period of 256 seconds for the highest calibration accuracy and calculate the compensation value:

$$\text{COMP} = \Delta f \times \text{PERIOD} = 0.37 \times 256 = 94.72 \approx 95 = 0x5F$$

Step 6: According to section [10.3.6 Register access operations](#), configure the RTC_COMPCFR1 register. In this example, write 1 to the SIGN bit field, write 10 to the PERIOD bit field, and the COMP bit field is 0x5F;

Step 7: Write RTC_COMPCFR1.EN to 1 to start clock error compensation.

Caution:

Steps 6 and 7 can also be combined, directly operate by halfword, and write 0xE05F to RTC_COMPCFR1.EN.



10.3.10 Alarm clock A and alarm clock B

RTC supports 2 independent alarm clocks (alarm clock A and alarm clock B), which can generate alarm events at any time of the week, and generate alarm interrupts, and output the alarm matching events through the external RTC_OUT pin. Set the ALARMAEN and ALARMBEN bit fields of the control register RTC_CR2 to 1 to enable the alarm clock A and the alarm clock B respectively.

By setting the hour, minute, second match control bits HOUREN, MINUTEEN, SECONDEN and the hour, minute, second count value HOUR, MINUTE, SECOND of the alarm clock A and B control registers (RTC_ALARM_A and RTC_ALARM_B), the alarm clock can be set at 'xx time xx Minutes xx seconds', or 'xx minutes xx seconds' or 'xx hours xx minutes' or 'xx hours' and other combinations to generate alarm events; the alarm week enable control bit WEEKMASK can select any day of the week to generate an alarm event, bit0 represents Sunday, and bits 1 to 6 represent Monday to Saturday.

Using 12 or 24 time system, the setting value of the alarm control register RTC_ALARMx (x=A,B) may be different. The example is as follows:

Table 10-4 Examples of timed alarm clocks

Timing requirement	Time system	RTC_ALARMx value	Alarm clock time
Alarm clock at 6:30 a.m. every morning	12/24 time	0x7F06 3000	06: 30: 00
Alarm clock at 7:00 every weekday	12/24 time	0x3E07 0000	07: 00: 00
Occurs every hour at 50 minutes	12/24 time	0x7F80 5000	xx: 50: 00
Occurs every 10 seconds per minute	12/24 time	0x7F80 8010	xx: xx: 10
Every Monday, it occurs every five seconds per minute at 12 noon.	12 time	0x0232 8005	12PM: xx: 05
Every Monday, it occurs every five seconds per minute at 12 noon.	24 time	0x0212 8005	12: xx: 05
Every Sunday, it occurs per second at 12: 30 p.m.	12 time	0x0112 3080	12AM: 30: xx
Every Sunday, it occurs per second at 12: 30 p.m.	24 time	0x0100 3080	00: 30: xx

When an alarm event occurs, the corresponding alarm match flag bit RTC_ISR.ALARMx will be set to 1 by hardware. If the alarm interrupt enable bit RTC_IER.ALARMx is set to 1, an interrupt request will be generated.

The match flag of alarm A or alarm B can be output through RTC_OUT pin, please refer to section [10.3.7 RTCOUT output](#).



10.3.11 Period interrupt function

The RTC has a built-in period interrupt module, which can generate fixed-period interrupt signals. The timing period is configured by the INTERVAL bit field of the control register RTC_CR0, as shown in the following table:

Table 10-5 Timing period configuration

RTC_CR0. INTERVAL	Period of Period Interrupt
000	No period interrupt is generated
001	0.5 seconds, generated every 1 second and 0.5 seconds
010	1 second, generated when the seconds counter changes
011	1 minute, generated when the minute counter changes
100	1 hour, generated when the hour counter changes
101	1 day, generated when the day counter changes
11x	1 month, generated when the month counter changes

Through the INTERVAL bit field of the RTC interrupt enable register RTC_IER, it is possible to set whether the period interrupt generates an interrupt request.



10.3.12 Automatic wake-up function

The automatic wake-up timer is a 16-bit programmable automatic reload subtraction counter, with a selectable clock source, and the timing range is: 61μs ~ 145h. When the counter overflows, it can generate an automatic wake-up interrupt and output the overflow flag through the RTC_OUT pin.

Set the AWTEN bit field of the control register RTC_CR2 to 1 to enable the automatic wake-up function, which is specially designed for low-power applications and can work in all working modes of the MCU.

The AWTSRC and AWTPRS bit fields of RTC control register RTC_CR2 are used to select the count clock source for the automatic wake-up timer. When AWTSRC is set to 0, RTCCLKD is selected as the count clock source; when AWTSRC is set to 1, TICKCLK is selected as the count clock source, at which time the RTC must be enabled. The count clock source can be divided by AWTPRS bit field, and the selectable division coefficients are 2, 4, 8, and 16.

The count period of the automatic wake-up timer is determined by the count clock source and the reload register RTC_AWTARR. The formula for calculating the timing duration is:

$$\text{Auto wake-up timer period} = (\text{RTC\_AWTARR} + 1) / \text{wake-up timer clock frequency}$$

$$\text{Shortest timing: } (0 + 1) / 16384\text{Hz} = 61\mu\text{s}$$

$$\text{Longest timing: } (65535 + 1) / 0.125\text{Hz} = 524288\text{s} = 8738\text{min} \approx 145.63\text{h}$$

Through the AWTIMER bit field of the RTC interrupt enable register RTC_IER, you can select whether to generate an interrupt request when the automatic wake-up timer overflows. The overflow flag of the automatic wake-up timer can be output through the RTC_OUT pin, see section [10.3.7 RTCOUT output](#).

The user can read the current count value of the wake-up timer at any time, and reading the RTC_AWTCNT register should be read multiple times until the same value is read twice.



10.3.13 Timestamp function

The RTC supports the timestamp function, that is, triggered by the RTC_TAMP pin, the current time and date are saved to the timestamp date register RTC_TAMPDATE and the timestamp time register RTC_TAMPTIM, and a timestamp interrupt can be generated at the same time.

The TAMPEDGE bit field of the control register RTC_CR2 is used to select whether the signal triggering the timestamp is valid on the rising edge or the falling edge, and the TAMPEN bit field of the RTC_CR2 register is used to enable the timestamp function.

Users can flexibly select the trigger pin RTC_TAMP, and need to configure this pin as a digital input and multiplexing function, please refer to the pin definition in the datasheet for more details.

When a timestamp event occurs, the timestamp event flag bit RTC_ISR.TAMP will be set to 1. If the timestamp interrupt enable bit RTC_IER.TAMP is set to 1, an interrupt request will be generated.

If after the first timestamp event occurs, the RTC_ISR.TAMP flag is not cleared by software, and the second timestamp event occurs, the timestamp overflow flag RTC_ISR.TAMPOV will be set to 1. If the timestamp overflow interrupt is set, the When the enable bit RTC_IER.TAMPOV is 1, an interrupt request will be generated.

10.3.14 RTC Interrupts

RTC supports 6 interrupt sources: wake-up interrupt, alarm A interrupt, alarm B interrupt, period interrupt, timestamp interrupt and timestamp overflow interrupt.

The opening of the interrupt is controlled by the interrupt enable register RTC_IER. Write '1' to the corresponding bit field to enable the interrupt, and write '0' to disable the interrupt; when an interrupt occurs, the hardware will automatically set the corresponding bit of the interrupt flag register RTC_ISR; Write '0' to the corresponding bit of the register RTC_ICR to clear the corresponding interrupt flag.

In the user application, after the RTC interrupt function is enabled, the same interrupt service routine will be called from the RTC interrupt entry (RTC_IRQHandler) when all RTC interrupt events occur. The specific RTC interrupt source.

After the interrupt event processing is completed, before exiting the interrupt service routine, you must write '0' to the corresponding bit of the RTC interrupt flag clear register RTC_ICR to clear the interrupt flag and avoid repeatedly entering the interrupt service routine.



10.4 List of registers

RTC base address: RTC_BASE = 0x4000 4400

Table 10-6 List of RTC registers

Register name	Register address	Register description
RTC_KEY	RTC_BASE + 0x00	Key value register
RTC_CR0	RTC_BASE + 0x04	Control register 0
RTC_CR1	RTC_BASE + 0x08	Control register 1
RTC_CR2	RTC_BASE + 0x0C	Control register 2
RTC_DATE	RTC_BASE + 0x14	Date register
RTC_TIME	RTC_BASE + 0x18	Time register
RTC_ALARMA	RTC_BASE + 0x1C	Alarm clock A control register
RTC_ALARMB	RTC_BASE + 0x20	Alarm clock B control register
RTC_TAMPDATE	RTC_BASE + 0x24	Timestamp date register
RTC_TAMPTIME	RTC_BASE + 0x28	Timestamp time register
RTC_IER	RTC_BASE + 0x30	Interrupt enable register
RTC_ISR	RTC_BASE + 0x34	Interrupt flag register
RTC_ICR	RTC_BASE + 0x38	Interrupt flag clear register
RTC_AWTARR	RTC_BASE + 0x2C	Wake-up timer reload value register
RTC_AWTCNT	RTC_BASE + 0x3C	Wake-up timer count value register
RTC_PSC	RTC_BASE + 0x40	Clock prescaler register
RTC_SSCNT	RTC_BASE + 0x44	Subsecond count register
RTC_COMPCFR1	RTC_BASE + 0x10	Clock error compensation register 1



10.5 Register descriptions

See section [1 Documentation conventions](#) for abbreviations used in register descriptions.

10.5.1 RTC_KEY key value register

Address offset: 0x00 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:8	RFU	-	Reserved bits, please keep the default value
7:0	KEY	WO	Key value register Write "0x00CA" and "0x0053" to this register in turn to modify the register of the RTC module. Write "0x00CA" and "0x----" to this register in turn, and the registers of the RTC module cannot be modified except for KEY and ICR. <i>Caution: "0x----" is any value except "0x0053".</i>



10.5.2 RTC_CR0 control register 0

Address offset: 0x04 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:8	RFU	-	Reserved bits, please keep the default value
7	START	RW	RTC timer operation control 0: Stop RTC timer 1: Start RTC timer
6:5	RTC1HZ	RW	RTC_1Hz output signal configuration 01: 1Hz signal compensated by RTCCLK clock Others: Reserved
4	RFU	-	Reserved bits, please keep the default value
3	H24	RW	Timing system setting 0: 12 time system 1: 24 time system
2:0	INTERVAL	RW	Timing period setting 000: No timing period is generated 001: 0.5 second, generated when the second timer changes and the 0.5 second moment 010: one second, generated when the second timer changes 011: one minute, generated when the minute timer changes 100: one hour, generated when the hour timer changes 101: one day, generated when the day timer changes 11x: January, generated when the month timer changes



10.5.3 RTC_CR1 control register 1

Address offset: 0x08 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:11	RFU	-	Reserved bits, please keep the default value
10:8	SOURCE	RW	Timing clock RTCCLK source configuration 000: External low speed clock LSE 001: External high speed clock HSE 010: Internal low-speed clock LSI 011: Internal high speed clock HSIOSC
7:3	RFU	-	Reserved bits, please keep the default value
2	WAIT	RO	RTC timing related register 1 loading status 0: Load completed, can read and write 1: Load in progress, cannot read or write <i>Caution: Including SECOND, MINUTE, HOUR, WEEK, DAY, MONTH, YEAR, AWTARR.</i>
1	WINDOW	RO	RTC timing related register 1 access window status 0: Non-read and write window, it is not recommended to read and write registers 1: Read and write window, you can read and write registers
0	ACCESS	RW	RTC timing related register 1 access control 0: Normal counting mode 1: Write/read mode <i>Caution: After the user writes 1 to the ACCESS bit and inquires that the WINDOW flag bit is 1, the user needs to complete the write/read operation of the related registers within 1 second, and then clear the ACCESS bit to 0. If the time for ACCESS bit to 1 is more than 1 second, it may result in a timing error.</i>

Caution 1:

RTC timing related registers are the RTC_DATE, RTC_TIME and RTC_AWTARR registers.



10.5.4 RTC_CR2 control register 2

Address offset: 0x0C Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:11	RFU	-	Reserved bits, please keep the default value
10	ALARMBEN	RW	Alarm clock B enable control 0: Disabled 1: Enabled
9	ALARMAEN	RW	Alarm clock A enable control 0: Disabled 1: Enabled
8	RFU	-	Reserved bits, please keep the default value
7	AWTEN	RW	Wake-up timer enable control 0: Disabled 1: Enabled
6	TAMPEN	RW	Timestamp enable control 0: Disabled 1: Enabled
5:4	RTCOUT	RW	RTC_OUT pin output signal configuration 00: RTC_1Hz signal, see RTC_CR0.RTC1HZ for details 01: Alarm clock A match flag 10: Alarm clock B match flag 11: Wake-up timer overflow flag
3	TAMPEDGE	RW	Timestamp signal edge configuration 0: Record the current time on the rising edge of the RTC_TAMP pin 1: Record the current time on the falling edge of the RTC_TAMP pin
2	AWTSRC	RW	Wake-up timer count clock source configuration 0: RTCCLKD 1: TICKCLK (need to enable RTC)
1:0	AWTPRS	RW	Wake-up timer count clock prescaler configuration 00: DIV2 01: DIV4 10: DIV8 11: DIV16



10.5.5 RTC_PSC clock prescaler register

Address offset: 0x40 Reset value: 0x0000 3FFF

Bit field	Name	Permission	Function description
31:28	RFU	-	Reserved bits, please keep the default value
27:20	PSC1	RW	Timing clock prescaler 1 $f_{\text{RTCCCLK}} = f_{\text{RTCCCLK}} / (\text{PSC1} + 1)$ Caution: The clock frequency after the maximum division of PSC1 should be less than or equal to 1MHz.
19:0	PSC2	RW	Timing clock prescaler 2 $f_{\text{TICKCLK}} = f_{\text{RTCCCLK}} / (\text{PSC2} + 1)$

Caution:User should configure PSC1 and PSC2 appropriately so that $f_{\text{TICKCLK}} = 2\text{Hz}$.

10.5.6 RTC_DATE date register

Address offset: 0x14 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:27	RFU	-	Reserved bits, please keep the default value
26:24	WEEK	RW	Week count value BCD code format, valid value range is 0 ~ 6 0 represents Sunday, 1 ~ 6 represents Monday to Saturday
23:16	YEAR	RW	Year count value BCD code format, valid value range is 00 ~ 99
15:13	RFU	-	Reserved bits, please keep the default value
12:8	MONTH	RW	Month count value BCD code format, valid value range is 1 ~ 12
7:6	RFU	-	Reserved bits, please keep the default value
5:0	DAY	RW	Day count value BCD code format, the valid value range is 1 ~ 28, 1 ~ 29, 1 ~ 30, 1 ~ 31



10.5.7 RTC_TIME time register

Address offset: 0x18 Reset value: 0x0001 0000

Bit field	Name	Permission	Function description
31:22	RFU	-	Reserved bits, please keep the default value
21:16	HOUR	RW	Hour count value BCD code format 24 time system, the valid value range of hour count is 0 ~ 23 12 time system, the valid value range of hour count is 1 ~ 12(AM), 21 ~ 32(PM), bit21 is 0 for AM, 1 for PM
15	RFU	-	Reserved bits, please keep the default value
14:8	MINUTE	RW	Minute count value BCD code format, valid value range is 0 ~ 59
7	RFU	-	Reserved bits, please keep the default value
6:0	SECOND	RW	Second count value BCD code format, valid value range is 0 ~ 59

Caution:*Writing the RTC_TIME register will clear the current value of the RTC_SSCNT register.*

10.5.8 RTC_SSCNT subsecond count register

Address offset: 0x44 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:21	RFU	-	Reserved bits, please keep the default value
20	SSCNT1	RO	Subsecond count 0.5 second flag 0: Current count is less than 0.5 seconds 1: Current count is greater than 0.5 seconds
19:0	SSCNT0	RO	Subsecond count less than 0.5 seconds Subsecond count = (PSC2+1) × SSCNT1 + SSCNT0



10.5.9 RTC_ALARMMA alarm clock A control register

Address offset: 0x1C Reset value: 0x0212 0000

Bit field	Name	Permission	Function description
31	RFU	-	Reserved bits, please keep the default value
30:24	WEEKMASK	RW	Alarm clock week enable control bit0 represents Sunday, bit1 ~ bit6 represent Monday to Saturday respectively If the corresponding bit is 1, the alarm clock is enabled
23	HOUREN	RW	Alarm clock hour bit mask control 0: The alarm clock needs to check the hour count value 1: The alarm clock has nothing to do with the hour count value
22	RFU	-	Reserved bits, please keep the default value
21:16	HOUR	RW	Alarm clock hour count value, BCD format
15	MINUTEEN	RW	Alarm clock minute bit mask control 0: The alarm clock needs to check the minute count value 1: The alarm clock has nothing to do with the minute count value
14:8	MINUTE	RW	Alarm clock minute count value, BCD format
7	SECONDEN	RW	Alarm clock second bit mask control 0: The alarm clock needs to check the second count value 1: The alarm has nothing to do with the second count value
6:0	SECOND	RW	Alarm clock second count value, BCD format



10.5.10 RTC_ALARMB alarm clock B control register

Address offset: 0x20 Reset value: 0x0412 0000

Bit field	Name	Permission	Function description
31	RFU	-	Reserved bits, please keep the default value
30:24	WEEKMASK	RW	Alarm clock week enable control bit0 represents Sunday, bit1 ~ bit6 represent Monday to Saturday respectively If the corresponding bit is 1, the alarm clock is enabled
23	HOUREN	RW	Alarm clock hour bit mask control 0: The alarm has nothing to do with the hour count value 1: The alarm clock needs to check the hour count value
22	RFU	-	Reserved bits, please keep the default value
21:16	HOUR	RW	Alarm hour count value in BCD format
15	MINUTEEN	RW	Alarm clock minute bit mask control 0: The alarm has nothing to do with the minute count value 1: The alarm clock needs to check the minute count value
14:8	MINUTE	RW	Alarm clock minute count value, BCD format
7	SECONDEN	RW	Alarm clock second bit mask control 0: The alarm has nothing to do with the second count value 1: The alarm clock needs to check the second count value
6:0	SECOND	RW	Alarm clock second count value, BCD format

10.5.11 RTC_TAMPDATE timestamp date register

Address offset: 0x24 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:13	WEEK	RO	Week count value when time stamp signal occurs, BCD format
12:8	MONTH	RO	Month count value when timestamp signal occurs, BCD format
7:6	RFU	-	Reserved bits, please keep the default value
5:0	DAY	RO	Day count value when timestamp signal occurs, BCD format



10.5.12 RTC_TAMPTIME timestamp time register

Address offset: 0x28 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:22	RFU	-	Reserved bits, please keep the default value
21:16	HOUR	RO	Hour count value when timestamp signal occurs, BCD format
15	RFU	-	Reserved bits, please keep the default value
14:8	MINUTE	RO	The minute count value when timestamp signal occurs, BCD format
7	RFU	-	Reserved bits, please keep the default value
6:0	SECOND	RO	The second count value when timestamp signal occurs, BCD format

10.5.13 RTC_IER interrupt enable register

Address offset: 0x30 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:7	RFU	-	Reserved bits, please keep the default value
6	INTERVAL	RW	Timing period interrupt enable control 0: Disabled 1: Enabled
5	RFU	-	Reserved bits, please keep the default value
4	TAMPOV	RW	Timestamp overflow interrupt enable control 0: Disabled 1: Enabled
3	TAMP	RW	Timestamp interrupt enable control 0: Disabled 1: Enabled
2	AWTIMER	RW	Wake up timer overflow interrupt enable control 0: Disabled 1: Enabled
1	ALARMB	RW	Alarm clock B interrupt enable control 0: Disabled 1: Enabled
0	ALARMA	RW	Alarm clock A interrupt enable control 0: Disabled 1: Enabled



10.5.14 RTC_ISR interrupt flag register

Address offset: 0x34 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:7	RFU	-	Reserved bits, please keep the default value
6	INTERVAL	RO	Periodic timing completion flag 0: No periodic timing completion event occurred 1: The periodic timing completion event has occurred
5	RFU	-	Reserved bits, please keep the default value
4	TAMPOV	RO	Timestamp overflow flag 0: No timestamp overflow event occurred 1: The timestamp overflow event has occurred
3	TAMP	RO	Timestamp event flag 0: No timestamp event occurred 1: Timestamp event has occurred
2	AWTIMER	RO	Wake-up timer timing completion flag 0: No wake-up timer timing completion event occurred 1: The wake-up timer timing completion event has occurred
1	ALARMB	RO	Alarm clock B match flag 0: No alarm clock B match event occurred. 1: The alarm clock B match event has occurred.
0	ALARMA	RO	Alarm clock A match flag 0: No alarm clock A match event occurred. 1: Alarm clock A match event has occurred.



10.5.15 RTC_ICR interrupt flag clear register

Address offset: 0x38 Reset value: 0x0000 007F

Bit field	Name	Permission	Function description
31:7	RFU	-	Reserved bits, please keep the default value
6	INTERVAL	R1W0	Periodic timing completion flag cleared W0: Clear the periodic timing completion flag W1: No function
5	RFU	-	Reserved bits, please keep the default value
4	TAMPOV	R1W0	Timestamp overflow flag cleared W0: Clear timestamp overflow flag W1: No function
3	TAMP	R1W0	Timestamp event flag cleared W0: Clear timestamp event flag W1: No function
2	AWTIMER	R1W0	Wake-up timer timing completion flag cleared W0: Clear wake-up timer timing completion flag W1: No function
1	ALARMB	R1W0	Alarm clock B match flag cleared W0: Clear alarm clock B match flag W1: No function
0	ALARMA	R1W0	Alarm clock A match flag cleared W0: Clear alarm clock A match flag W1: No function

10.5.16 RTC_AWTARR wake-up timer reload value register

Address offset: 0x2C Reset value: 0x0000 FFFF

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	ARR	RW	Wake-up timer reload value



10.5.17 RTC_AWTCNT wake-up timer count value register

Address offset: 0x3C Reset value: 0x0000 FFFF

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	ARR	RW	Wake-up timer count value

Caution:*Reading data from this register should be read multiple times until the same value is read twice.*

10.5.18 RTC_COMPCFR1 wake-up clock error compensation register 1

Address offset: 0x10 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15	EN	RW	Clock error compensation enable control 0: Disable clock error compensation 1: Enable clock error compensation
14	SIGN	RW	Clock compensation direction 0: Increase count value (TICKCLK clock frequency greater than 2Hz) 1: Decrease count value (TICKCLK clock frequency less than 2Hz)
13:12	PERIOD	RW	Compensation period selection 00: 32 seconds, compensation value is COMP[8:0], compensation step 0.950ppm 01: 128 seconds, compensation value is COMP[10:0], compensation step 0.238ppm 10: 256 seconds, compensation value is COMP[11:0], compensation step 0.119ppm Caution: Compensation accuracy is half of the compensation step.
11:0	COMP	RW	Compensation Value Compensable range $\pm 488.0\text{ppm}$



11 Basic timer (BTIM)

11.1 Overview

CW32L010 integrates three basic timers (BTIM), each BTIM is completely independent and has the same function, each contains a 16-bit automatic reload counter and is driven by a programmable prescaler. BTIM supports four working modes: internal counting mode, external counting mode, triggered start mode and gated counting mode, and supports the interrupt generation on the occurrence of update events and trigger events. Counter reset can be controlled by the reset input signal in different working modes.

11.2 Main features

- 16-bit auto-reload up counter
- Programmable prescaler supports 1, 2, 3, 4, ..., 65536 frequency division
- Supports single counting mode and continuous counting mode
- Trigger input signal (TRGI) controls the timer for multiple slave modes
- Trigger output signal (TRGO) for synchronisation between peripherals
- Reset signal (RSTI) controls counter reset
- Generates interrupts when update and trigger events occur

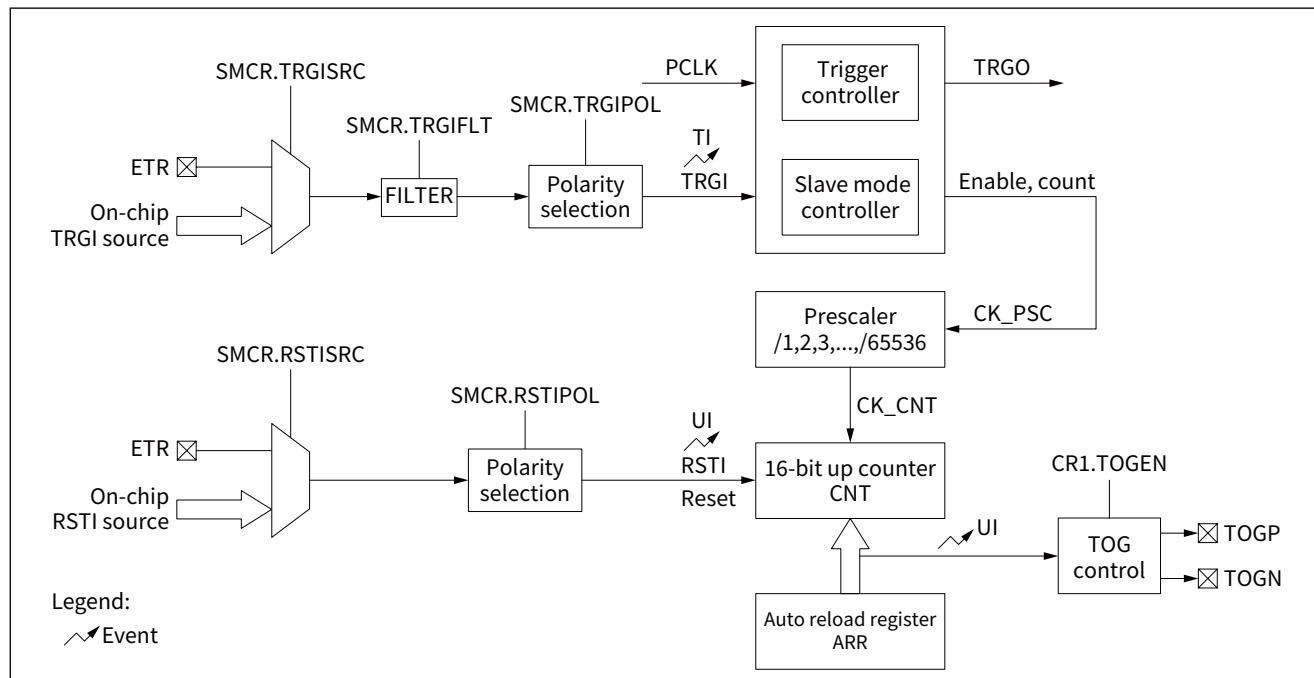


11.3 Functional description

11.3.1 Functional block diagram

The functional block diagram of BTIM is shown in the following figure:

Figure 11-1 BTIM functional block diagram



11.3.1.1 Counting unit

The core components of the counting unit is a 16bit up-counter CNT and a 16bit auto-reload register ARR, and the counter clock can be divided by a prescaler. The counter register, the auto-reload register and the prescaler register can be read and written by software, and read and write operations can be performed even when the counter is running.

The prescaler divides the CK_PSC clock to get the count clock CK_CNT to drive the counter count. The divider coefficient is set through the BTIMx_PSC register, which supports 1,2,3,4,...,65536 divider.

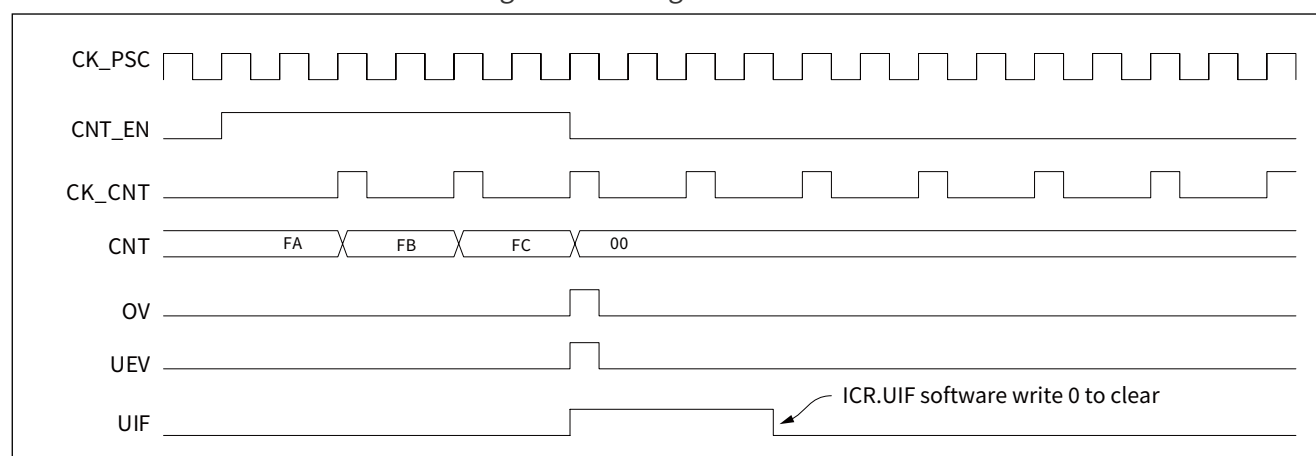
The counter can operate in either single count or continuous count mode, selected by the ONESHOT bit field of control register BTIMx_CR1. When the EN bit field of the BTIMx_CR1 register is set to 1, the counter starts counting up, noting that the actual counter enable signal, CNT_EN, is set to 1 one clock cycle after EN is set to 1.

Single count mode

Set BTIMx_CR1.ONESHOT to 1 to enable the timer work in single count mode. Set BTIMx_CR1.EN to 1 to enable BTIMx, and the counter CNT counts up under the drive of the count clock CK_CNT. When the count value reaches the reload value ARR, the overflow signal OV and the update event UEV are generated (the OV signal and UEV signal will be cleared automatically), the counter update interrupt flag bit BTIMx_ISR.UIF is set by the hardware, and at the same time, the counter stops counting, and BTIMx_CR1.EN is cleared by the hardware automatically.

The following figure shows an example of single count mode, where BTIMx_PSC = 0x01 and BTIMx_ARR = 0xFC.

Figure 11-2 Single count mode

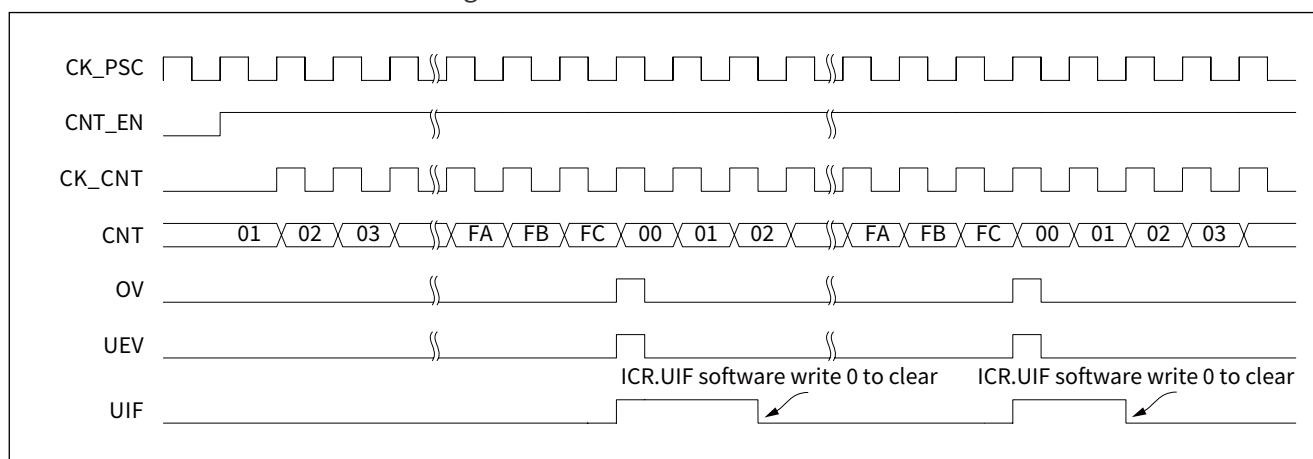


Continuous count mode

Set BTIMx_CR1.ONESHOT to 0 to make the timer work in continuous counting mode. Set BTIMx_CR.EN to 1 to enable BTIMx, and the counter CNT counts up under the drive of the count clock CK_CNT. When the count value reaches the reload value ARR, the counter will start to count up again from 0, and at the same time, the overflow signal OV and the update event UEV will be generated (the OV signal and the UEV signal will be cleared automatically), and the counter update interrupt flag bit, BTIMx_ISR.UIF, will be set by the hardware.

The following figure shows an example of continuous count mode, where BTIMx_PSC=0x00, BTIMx_ARR=0xFC:

Figure 11-3 Continuous count mode



11.3.1.2 Update Event (UEV)

Allow update event to be disabled or enabled by the UDIS bit field of the control register BTIMx_CR1.

Enable UEV

Set the UDIS bit field to 0 to enable UEV. At this time, update request sources can be selected according to the URS bit field, as shown in the following table:

Table 11-1 Update source settings

BTIMx_CR1.URS	Update source
0	The counter overflows. UG set. Update events generated from the slave mode controller.
1	The counter overflows

When an update event occurs, the following actions occur:

1. Reinitialize the counter, i.e. the counter is cleared to 0;
2. The prescaler counter is cleared to zero, but the prescaler ratio is not affected;
3. The event update interrupt flag bit BTIMx_ISR.UIF is set by hardware.

When an update event UEV occurs, the event update interrupt flag BTIMx_ISR.UIF is set by the hardware, and if the interrupt is allowed (BTIMx_IER.UIE is set to 1), an update interrupt request is generated, and BTIMx_ICR.UIF is set to 0 to clear the flag.

Disable UEV

Set the UDIS bit field to 1 to disable UEV and no more update events will be generated.

If the UG bit field is set to 1, or a hardware reset is received from the slave mode controller, the counter and the counter of the prescaler are reinitialized.



11.3.1.3 Trigger input channel

The trigger input signal (TRGI) can be used as the count clock of the counter, as well as the trigger start signal and gate signal of the counter. For details, see [11.3.2 Operating modes](#). For details on the source of the TRGI signal, see the TRGISRC bit field description of the [11.9.3 BTIMx_SMCR slave mode control register](#). The TRGI signal can be filtered and controlled, and the effective polarity of the TRGI signal can be set.

Filter

The filter is a digital filter that samples the trigger input signal at a certain frequency, and the signal is valid when it is continuously sampled to N same levels, otherwise the signal is invalid, so as to filter out the high-frequency clutter signal.

The sampling clock of the filter unit is PCLK or a division of PCLK, and the division ratio of PCLK and the number of sampling points N can be selected from the TRGIFLT bit field of the mode control register BTIMx_SMCR.

Polarity selection

The TRGIPOL bit field of the slave mode control register BTIMx_SMCR is used to select the effective polarity of the trigger input signal. When TRGIPOL is set to 0, the TRGI signal is not reversed and the high level or rising edge is valid. When TRGIPOL is set to 1, the TRGI signal is reversed and the low level or falling edge is valid.

11.3.1.4 Trigger output channel

Through the MMS bit field of the control register BTIMx_CR2, you can select the trigger output signal (TRGO) to be sent to other peripherals for synchronization in BTIM master mode, as shown in the following table:

Table 11-2 BTIM trigger output (TRGO)

BTIMx_CR2.MMS	TRGO signal
000	Reset signal
001	Counter enable signal
010	Update event
011	Overflow signal



11.3.1.5 Reset input channel

BTIM supports to reset the counter by the valid edge of the reset input signal (RSTI) in different modes of operation, while generating update events.

Refer to the RSTISRC bit field description of the slave mode control register BTIMx_SMCR for the exact source of the RSTI signal. The reset signal of the BTIMx_ETR pin can be filtered and controlled by the ETRFLT bit field of the slave mode control register BTIMx_SMCR, the valid polarity of the RSTI signal is set by the RSTIPOL bit field of the BTIMx_SMCR register. When RSTIPOL is set to 0, the RSTI signal is not inverted and the rising edge is valid; When RSTIPOL is set to 1, the RSTI signal is inverted and the falling edge is valid.

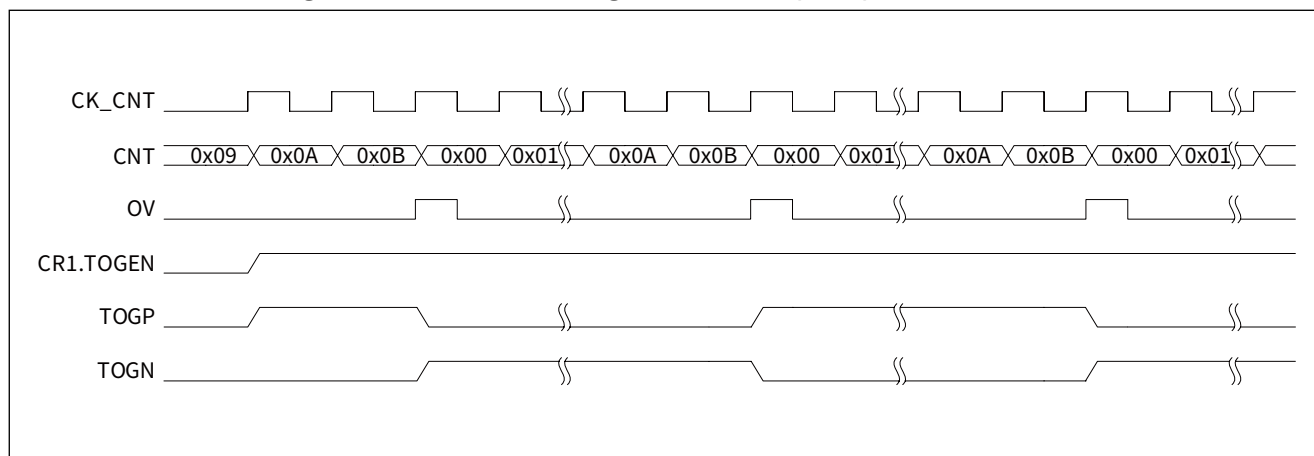
11.3.1.6 Flip output unit

The flip output unit can control the external BTIMx_TOGP and BTIMx_TOGN pins to output the flip signal through the ARR overflow signal OV.

- When BTIMx_CR1.TOGEN is set to 0, both BTIMx_TOGP and BTIMx_TOGN pins output low level.
- When BTIMx_CR1.TOGEN is set to 1, the BTIMx_TOGP and BTIMx_TOGN pins output signals with opposite levels (BTIMx_TOGP default level is high); when the counter ARR overflows, the output levels of BTIMx_TOGP and BTIMx_TOGN pins will flip.

The following figure shows the schematic diagram of BTIMx_TOGP and BTIMx_TOGN pins level flip output in continuous counting mode:

Figure 11-4 Schematic diagram of level flip output (ARR=0x0B)



11.3.2 Operating modes

BTIM supports 4 operating modes: internal count mode, external count mode, trigger start mode, and gated count mode. Configured through the SMS bit field of the basic control register BTIMx_SMCR.

Table 11-3 BTIM operating modes

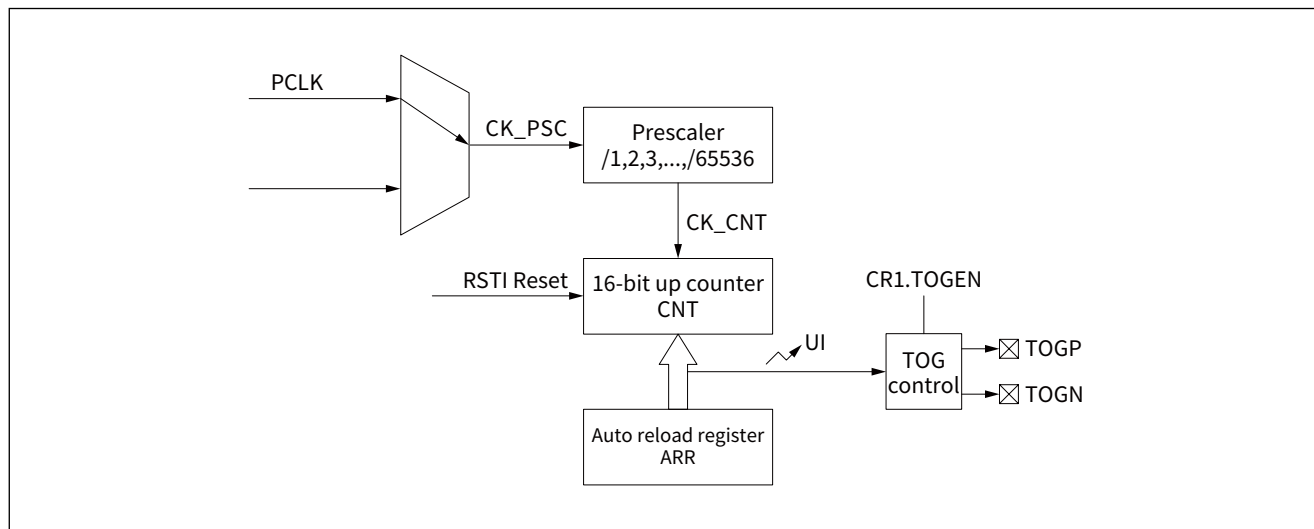
BTIMx_SMCR.SMS	Operating mode	Description
000	Disable slave mode, use internal clock	Clock source is PCLK
001	Trigger start mode	Clock source is PCLK, TRGI signal triggers the counter to start
010	Gated count mode	Clock source is PCLK, TRGI signal as gate signal
011	External count mode	Clock source is TRGI signal



11.3.2.1 Internal count mode

When the SMS bit field of the slave mode control register BTIMx_SMCR is 0x0, the timer slave mode is disabled and the prescaler clock is provided directly by the internal clock PCLK. Setting BTIMx_CR1.EN to 1 will enable the counter to start counting. The internal count mode block diagram is shown below:

Figure 11-5 Internal count model block diagram



In practice, the frequency of the system clock PCLK is known, through the reasonable set of prescaler coefficients PSC can be counted for a fixed duration clock, with the setup of the reload value ARR and the use of the counter event update interrupt flag bit, you can accurately obtain a particular duration, so as to achieve the purpose of timing.

Timing time T calculation formula:

$$T = ((PSC+1)/PCLK) \times (ARR+1)$$

Where, PCLK is the counter clock source, PSC is the prescaler coefficient, and ARR is the reload value.

Example:

When the frequency of the counter clock source PCLK is 24MHz, the timing is required to be 100ms. If the prescaler coefficient PSC is set to 0xFF, calculate:

$$T = 100\text{ms} = ((255+1)/24\text{MHz}) \times (ARR+1)$$

then

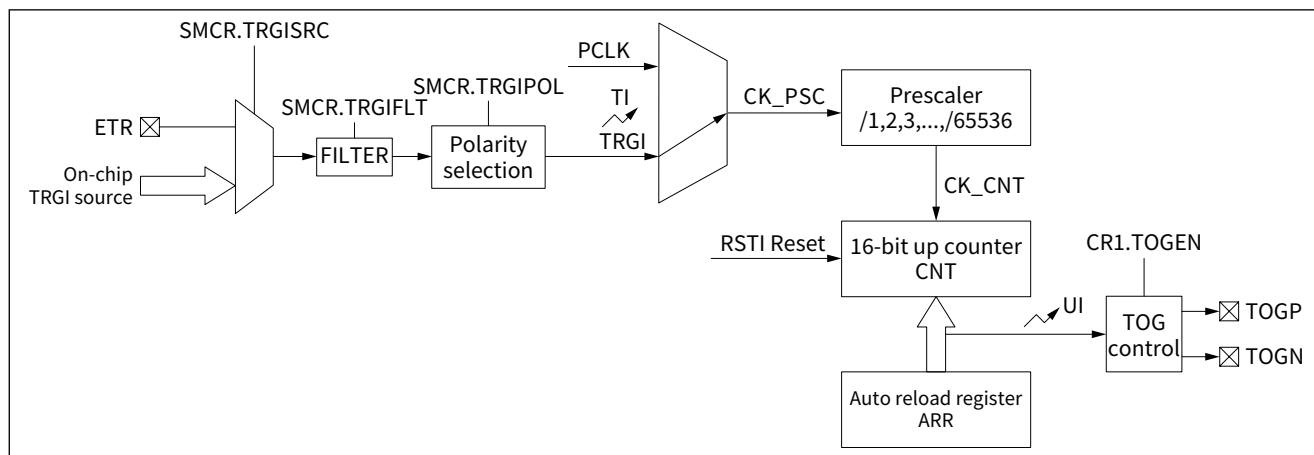
$$ARR = 9374 \text{ (0x249E)}$$

That is, you need to set the reload value ARR to 0x249E.

11.3.2.2 External count mode

When the SMS bit field of the slave mode control register BTIMx_SMCR is 0x3, the counter clock is provided by the rising edge of the selected trigger signal (TRGI). There are various sources of the TRGI signal, which are specifically selected through the TRGISRC bit field of the BTIMx_SMCR register, and can be filtered for control and polarity selection. The external counter mode block diagram is shown below:

Figure 11-6 External counter mode block diagram



Set BTIMx_CR1.EN to 1 to enable the counter, which accumulates under the drive of the CK_CNT clock signal. When TRGI has a valid edge, the BTIMx_ISR.TIF flag is set to 1. Writing 0 to BTIMx_ICR.TIF clears the flag.

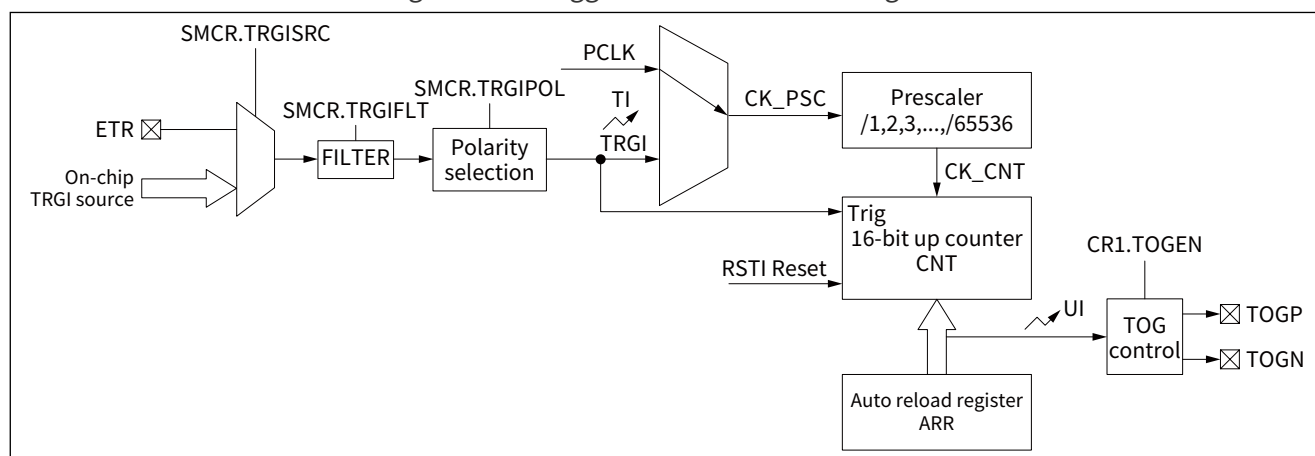
11.3.2.3 Trigger start mode

When the SMS bit field of the slave mode control register BTIMx_SMCR is 0x1, the BTIM is configured to trigger start mode. In this mode, when BTIMx_CR1.EN is set to 1 or a rising edge of the trigger signal TRGI occurs, the counter CNT will be enabled to count the CK_CNT signal of the internal clock PCLK divided by the prescaler.

The source of the TRGI signal is controlled by the TRGISRC bit field of the slave mode control register BTIMx_SMCR, and the TRGI signal filtering can be set by the TRGIFLT bit field, and the effective polarity of the TRGI signal can be set by the TRGIPOL bit field.

The trigger start mode block diagram is shown below:

Figure 11-7 Trigger start mode block diagram

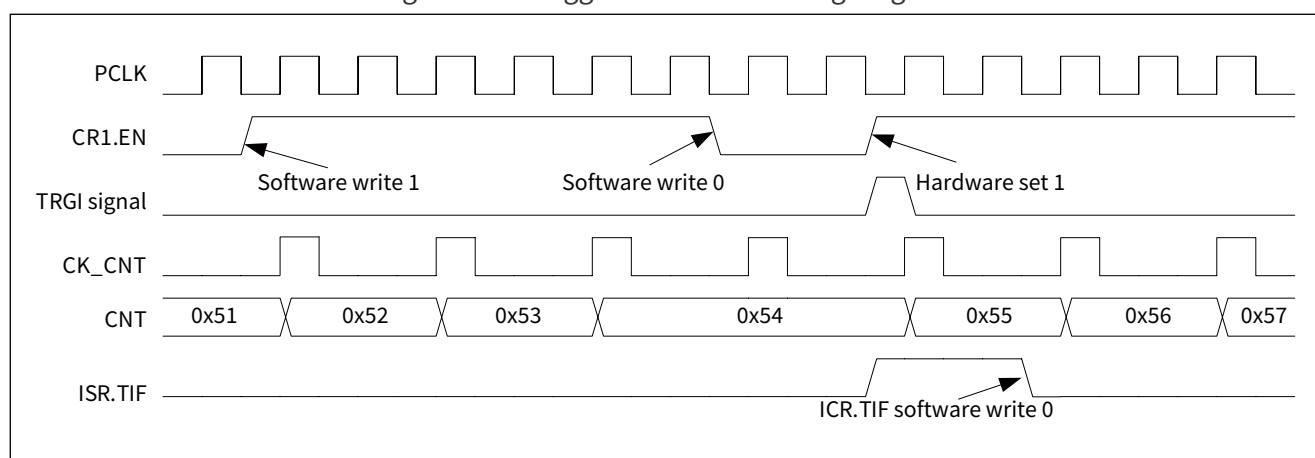


When a valid trigger signal is detected, the following effects occur:

1. BTIMx_CR1.EN is set by hardware;
2. The trigger interrupt flag bit BTIMx_ISR.TIF is set to 1, interrupt can be generated;
3. The counter starts and starts counting.

After the counter is started, the counter counts upwards from the initial value and generates an overflow when the count value reaches the reload value ARR. When the enabled control bit BTIMx_CR1.EN is set to 0 at any time, the counter immediately stops counting; When the enabled control bit BTIMx_CR1.EN is set to 1 again or the trigger signal is valid, the counter continues to count according to the previous setting. The trigger start mode timing diagram is shown below, where BTIMx_PSC=0x01:

Figure 11-8 Trigger start mode timing diagram



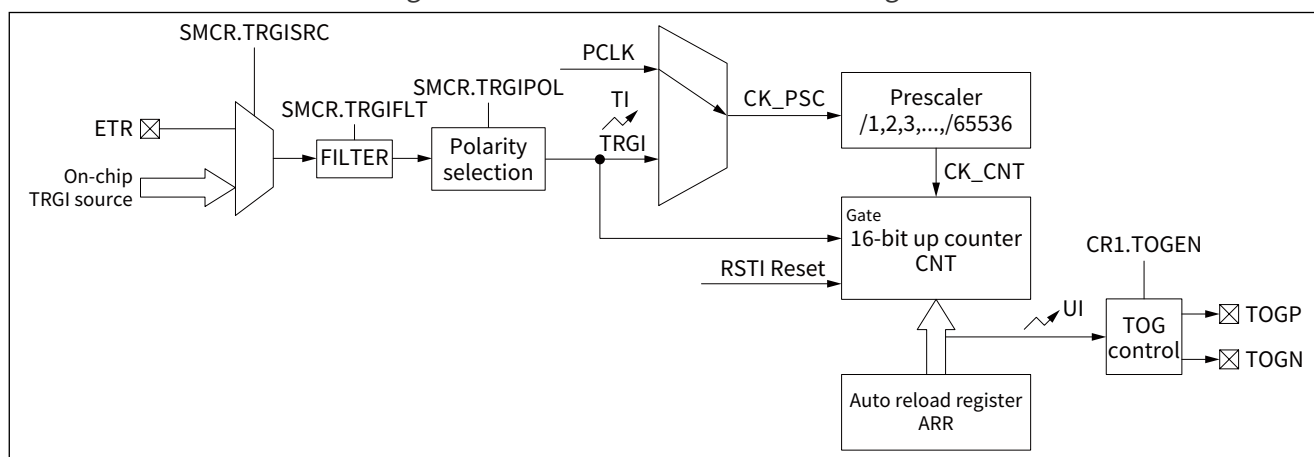
11.3.2.4 Gated count mode

When the SMS bit field of the slave mode control register BTIMx_SMCR is set to 0x2, the BTIM is configured in gated counter mode. In this mode, when the trigger input (TRGI) is high and BTIMx_CR1.EN is 1, the counter will be activated to count the CK_CNT signal after the internal clock PCLK is divided by the prescaler; and when the trigger input (TRGI) is low or BTIMx_CR1.EN is 0, the counter will stop counting immediately.

The source of the TRGI signal is controlled by the TRGISRC bit field of the slave mode control register BTIMx_SMCR, and the TRGI signal filter can be set via the TRGIFLT bit field, and the effective polarity of the TRGI signal can be set via the TRGIPOL bit field.

The gated count mode block diagram is shown below:

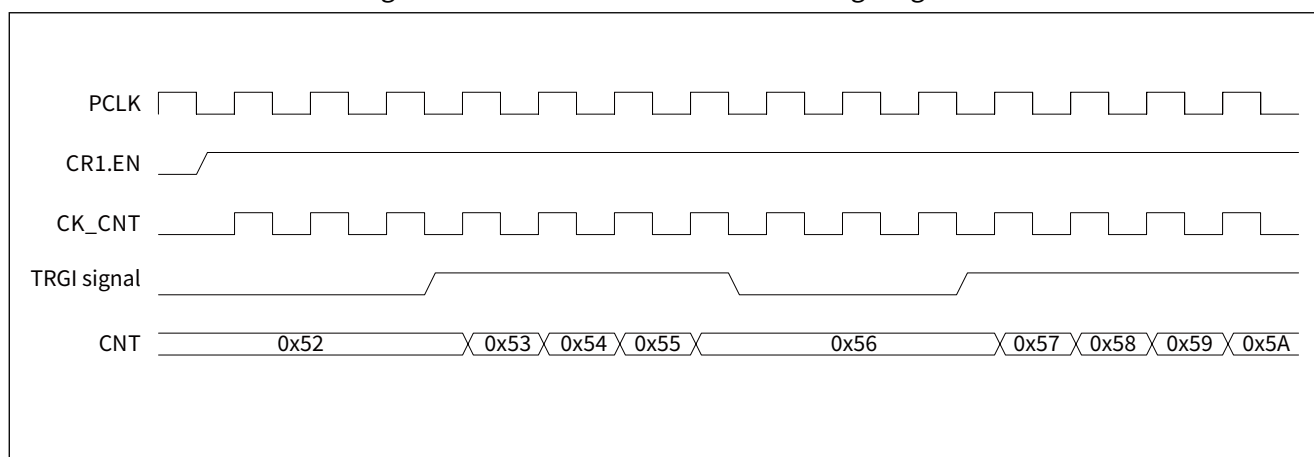
Figure 11-9 Gated count mode block diagram



After the counter is started, the counter starts to count up from the initial value. When the count value reaches the reload value ARR, an overflow occurs. When the running control bit BTIMx_CR1.EN is set to 0 at any time or the gate control signal is invalid at any time, the counter immediately stops counting. When the running control bit BTIMx_CR1.EN is set to 1 again and the gate control signal is valid, the counter continues to count according to the previous setting.

The gated count mode timing diagram is shown in the following figure, where BTIMx_PSC=0x00:

Figure 11-10 Gated counter mode timing diagram



11.3.3 UIF bit remap

Set the UIFREMAP bit in the BTIMx_CR1 register to 1 to enable the UIF state bit remapping function, which can force the update interrupt flag BTIMx_ISR.UIF to continuously copy to the UIFCPY bit field of the BTIMx_CNT register. This allows the counter value and the potential flip-flop condition issued by the UIFCPY flag to be read automatically. In specific cases, this simplifies calculations by avoiding competing conditions when processing is shared between background tasks (counter reads) and interrupts (update interrupts).

There is no delay between UIF and UIFCPY flag enable.

When BTIMx_CR1.UIFREMAP is 0, the UIFCPY bit field is reserved and read as 0.



11.4 BTIM interrupt

BTIM supports 2 interrupt sources. When a BTIM interrupt event occurs, the interrupt flag bit will be set by hardware, and an interrupt request will be generated if the corresponding interrupt enable control bit is set.

In the user BTIM interrupt service procedure, the relevant BTIM interrupt flag bit should be queried for corresponding processing, and the interrupt flag bit should be cleared before exiting the interrupt service procedure to avoid entering the interrupt service procedure repeatedly.

The flag bit, interrupt enable bit, interrupt flag clear bit or clear method of each BTIM interrupt source are shown in the following table:

Table 11-4 BTIM interrupt control

Interrupt event	Interrupt flag bit	Interrupt enable bit	Interrupt flag clear
Update interrupt	ISR.UIF	IER.UIE	Write 0 to ICR.UIF
Trigger interrupt	ISR.TIF	IER.TIE	Write 0 to ICR.TIF



11.5 Trigger ADC

The trigger output signal TRGO of the BTIM can be used to trigger the ADC startup, and the trigger output TRGO has a variety of possible events, which are selected by the MMS bit field of the control register BTIMx_CR2. At the same time, the ADC peripheral needs to configure its external trigger start register to select the corresponding trigger source.

It should be noted that the ADC clock must be enabled before an event can be received from the main timer and that the ADC clock must not be changed in real time when the trigger signal is received from the timer.



11.6 Debugging support

BTIM supports to stop or continue counting in debug mode, which is set by the BTIM123 bit field of the debug status timer control register SYSCTRL_DEBUG.

- Set SYSCTRL_DEBUG.BTIM123 to 1, then suspend the counter counting of BTIM1, BTIM2 and BTIM3 in the debug state;
- Set SYSCTRL_DEBUG.BTIM123 to 0, then the counters of BTIM1, BTIM2 and BTIM3 continue to count in the debug state.



11.7 Programming examples

11.7.1 Internal count mode programming example

Step 1: Set `SYSCTRL_APBEN2.BTIM123` to 1, and enable the configuration clock and working clock of BTIM1, BTIM2, and BTIM3;

Caution:

BTIM1, BTIM2 and BTIM3 share the `SYSCTRL_APBEN2.BTIM123` bit.

Step 2: Set `BTIMx_SMCR.SMS` to 0x00 to make BTIMx work in internal count mode;

Step 3: Configure `BTIMx_CR1.ONESHOT` to select single or continuous count mode. If it is configured as 0, it is continuous count mode, and if it is configured as 1, it is single count mode;

Step 4: Configure `BTIMx_PSC`, set the prescaler division ratio;

Step 5: Configure `BTIMx_ARR`, set BTIMx count overflow time;

Step 6: Set `BTIMx_CNT` to 0x0000 for counting;

Step 7: Set `BTIMx_IER.UIE` to 1 and configure the corresponding NVIC to enable the update interrupt;

Step 8: Set `BTIMx_CR1.EN` to 1, start BTIMx;

Step 9: When the counter overflows, the `BTIMx_ISR.UIF` flag bit is 1, enter the interrupt service routine and set `BTIMx_ICR.UIF` to 0 to clear this interrupt flag.

11.7.2 External count mode programming example

Configure the counter to count on the rising edge of the `BTIMx_ETR` pin. The operation steps are as follows:

Step 1: Set `SYSCTRL_APBEN2.BTIM123` to 1, and enable the configuration clock and working clock of BTIM1, BTIM2, and BTIM3;

Caution:

BTIM1, BTIM2 and BTIM3 share the `SYSCTRL_APBEN2.BTIM123` bit.

Step 2: Configure the GPIO corresponding to the `BTIMx_ETR` pin to multiplexed input mode. For details about register configuration, see [8 General-purpose input/output \(GPIO\)](#);

Step 3: Set `BTIMx_SMCR.TRGISRC` to 0xF, and select the source of TRGI signal as `BTIMx_ETR` pin;

Step 4: Set `BTIMx_SMCR.TRGIFLT` to configure the TRGI input signal filter bandwidth;

Step 5: Set `BTIMx_SMCR.TRGIPOL` to 0 and select the rising edge to be valid;

Step 6: Set `BTIMx_SMCR.SMS` to 0x3 to make BTIMx work in external count mode;

Step 7: Configure `BTIMx_CR1.ONESHOT` to select single or continuous count mode. If configured to 0, it is continuous count mode, and if configured to 1, it is single count mode;

Step 8: Configure `BTIMx_PSC` to set the prescaler division ratio;

Step 9: Configure `BTIMx_ARR` to set the BTIMx count overflow time;

Step 10: Set `BTIMx_IER.UIE` to 1 and configure the corresponding NVIC to enable the update interrupt;

Step 11: Set `BTIMx_CR1.EN` to 1, start BTIMx;

Step 12: When the counter overflows, the `BTIMx_ISR.UIF` flag bit is set to 1 to enter the interrupt service routine, and setting `BTIMx_ICR.UIF` to 0 clears this interrupt flag.



11.7.3 Trigger start mode programming example

The BTIMx_ETR input triggers the start of the counter when a rising edge appears. The operation steps are as follows:

Step 1: Set SYSCTRL_APBEN2.BTIM123 to 1, and enable the configuration clock and working clock of BTIM1, BTIM2, and BTIM3;

Caution:

BTIM1, BTIM2 and BTIM3 share the SYSCTRL_APBEN2.BTIM123 bit.

Step 2: Configure the GPIO corresponding to the BTIMx_ETR pin to multiplexed input mode. For details about register configuration, see [8 General-purpose input/output \(GPIO\)](#);

Step 3: Set BTIMx_SMCR.TRGISRC to 0xF, and select the source of TRGI signal as BTIMx_ETR pin;

Step 4: Set BTIMx_SMCR.TRGIFLT to configure the TRGI input signal filter bandwidth;

Step 5: Set BTIMx_SMCR.TRGIPOL to 0 and select the rising edge to be valid;

Step 6: Set BTIMx_SMCR.SMS to 0x1 to make BTIMx work in trigger start mode;

Step 7: Configure BTIMx_CR1.ONESHOT to select single or continuous count mode. If configured to 0, it is continuous count mode, and if configured to 1, it is single count mode;

Step 8: Configure BTIMx_PSC to set the prescaler division ratio;

Step 9: Configure BTIMx_ARR to set the BTIMx count overflow time;

Step 10: When a rising edge occurs on the BTIMx_ETR input, the counter starts counting and the TIF flag is set to 1.



11.7.4 Gated count mode programming example

The signal input from the external BTIM_ETR pin is used as the gated control signal to control the count of the counter. The operation steps are as follows:

Step 1: Set SYSCTRL_APBEN2.BTIM123 to 1, and enable the configuration clock and working clock of BTIM1, BTIM2, and BTIM3;

Caution:

BTIM1, BTIM2 and BTIM3 share the SYSCTRL_APBEN2.BTIM123 bit.

Step 2: Configure the GPIO corresponding to the BTIMx_ETR pin to multiplexed input mode. For details about register configuration, see [8 General-purpose input/output \(GPIO\)](#);

Step 3: Set BTIMx_SMCR.TRGISRC to 0xF, and select the source of TRGI signal as BTIMx_ETR pin;

Step 4: Set BTIMx_SMCR.TRGIFLT to configure the TRGI input signal filter bandwidth;

Step 5: Set BTIMx_SMCR.TRGIPOL to 0 and select high level to be valid;

Step 6: Set BTIMx_SMCR.SMS to 0x2 to make BTIMx work in gated count mode;

Step 7: Configure BTIMx_CR1.ONESHOT to select single or continuous count mode. If configured to 0, it is continuous count mode, and if configured to 1, it is single count mode;

Step 8: Configure BTIMx_PSC to set the prescaler division ratio;

Step 9: Configure BTIMx_ARR to set the BTIMx count overflow time;

Step 10: Set BTIMx_CR1.EN to 1 to enable counter;

Step 11: The counter starts counting when the BTIMx_ETR input is a high level and stops counting when the BTIMx_ETR input is a low level.



11.7.5 Counter reset programming example

The counter is reset when a rising edge occurs on the RSTI input. The operation steps are as follows:

Step 1: Configure the BTIM operating mode according to the programming example;

Step 2: Configure BTIMx_SMCR.RSTISRC to select the source of the counter reset signal RSTI;

Step 3: Setting BTIMx_SMCR.RSTIPOL to 0 to select rising edge validity;

Step 4: When a rising edge occurs on the RSTI input, the counter is cleared to 0 and starts counting from 0 again, while the UIF flag is set to 1.



11.8 List of registers

BTIM1 base address: BTIM1_BASE = 0x4000 4800

BTIM2 base address: BTIM2_BASE = 0x4000 4840

BTIM3 base address: BTIM3_BASE = 0x4000 4880

Table 11-5 List of BTIM registers

Register name	Register address	Register description
BTIMx_CR1	BTIMx_BASE + 0x00	Control register 1
BTIMx_CR2	BTIMx_BASE + 0x04	Control register 2
BTIMx_SMCR	BTIMx_BASE + 0x08	Slave mode control register
BTIMx_IER	BTIMx_BASE + 0x0C	Interrupt enable register
BTIMx_ISR	BTIMx_BASE + 0x10	Interrupt flag register
BTIMx_ICR	BTIMx_BASE + 0x18	Interrupt flag clear register
BTIMx_EGR	BTIMx_BASE + 0x14	Event generation register
BTIMx_CNT	BTIMx_BASE + 0x24	Count register
BTIMx_PSC	BTIMx_BASE + 0x28	Prescaler register
BTIMx_ARR	BTIMx_BASE + 0x2C	Auto-reload register



11.9 Register descriptions

See section [1 Documentation conventions](#) for abbreviations used in register descriptions.

11.9.1 BTIMx_CR1 control register 1

Address offset: 0x00 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15	TOGEN	RW	TOG pin output waveform enable control 0: TOGP and TOGN both output low level 1: TOGP and TOGN output opposite level
14:12	RFU	-	Reserved bits, please keep the default value
11	UIFREMAP	RW	UIF status bit remapping enable control 0: Remapping is disabled and BTIMx_CNT[31] remains 0 1: Remapping is enabled. BTIMx_CNT[31] is equivalent to BTIMx_ISR.UIF
10:4	RFU	-	Reserved bits, please keep the default value
3	ONESHOT	RW	Single/Continuous count mode control 0: Continuous count mode 1: Single count mode
2	URS	RW	Update the request source configuration 0: When UEV is enabled, all of the following events generate update interrupt - Counter overflow - Set the UG bit field to 1 - Update events generated via the slave mode controller 1: When UEV is enabled, only overflows on the counter generate update interrupt
1	UDIS	RW	Update disable control 0: Enable UEV Update events can be generated by the following events: - Overflow on the counter - Setting the UG bit field to 1 - Update events generated via the slave mode controller 1: Disable UEV No update event is generated. However, if the UG bit field is set to 1 or a hardware reset is received from the slave mode controller, the counters and prescalers are reinitialised.
0	EN	RW	Timer operation control 0: Timer stopped 1: Timer running



11.9.2 BTIMx_CR2 control register 2

Address offset: 0x04 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:7	RFU	-	Reserved bits, please keep the default value
6:4	MMS	RW	Master mode output signal (TRGO) configuration 000: Reset signal, EGR.UG is used as trigger output (TRGO). If the reset is generated by the trigger input, the signal on TRGO will be delayed compared to the actual reset. 001: Enable signal, CR1.EN used as trigger output (TRGO). The trigger output can be used to start multiple timers at the same time, or to control enabling the slave timer over a period of time. The counter enable signal is a combination of the EN control bit and the logic and operation of the trigger input in gated mode. There is a delay in TRGO when the counter enable signal is controlled by a trigger input, except when master/slave mode is selected (see the description for SMCR.MSM). 010: Update signal, select update event as trigger output (TRGO). For example, the master timer can be used as a prescaler for the slave timer. 011: Overflow signal, output a pulse whenever BTIM overflows.
3:0	RFU	-	Reserved bits, please keep the default value



11.9.3 BTIMx_SMCR slave mode control register

Address offset: 0x08 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:18	RFU	-	Reserved bits, please keep the default value
17	TRGIPOL	RW	Trigger input signal (TRGI) polarity selection 0: Positive phase, high level or rising edge is valid 1: Reversed phase, low level or falling edge is valid
16	RSTIPOL	RW	Count reset signal (RSTI) polarity selection 0: Positive phase, rising edge is valid 1: Reversed phase, falling edge is valid
15	RFU	-	Reserved bits, please keep the default value
14:12	TRGIFLT	RW	TRGI signal digital filter circuit sampling clock and sampling points configuration 000: No filter 001: $F_{\text{sample}} = \text{PCLK}$, $N=2$ 010: $F_{\text{sample}} = \text{PCLK}$, $N=4$ 011: $F_{\text{sample}} = \text{PCLK}$, $N=6$ 100: $F_{\text{sample}} = \text{PCLK}/4$, $N=4$ 101: $F_{\text{sample}} = \text{PCLK}/4$, $N=6$ 110: $F_{\text{sample}} = \text{PCLK}/8$, $N=4$ 111: $F_{\text{sample}} = \text{PCLK}/8$, $N=6$
11	MSM	RW	Master/slave mode configuration 0: Nothing is done 1: The trigger input event (TRGI) action of the current timer is delayed to perfectly synchronize the current timer with its slave timer (via TRGO).
10:7	TRGISRC	RW	Trigger input signal (TRGI) source configuration 0000: Reserved 0111: ATIM_Trgo 0001: LVD_OUT signal 1000: BTIM1_Trgo 0010: VCx_OUT signal 1001: BTIM2_Trgo 0011: SPI_NCS signal 1010: BTIM3_Trgo 0100: I2C_SCL signal 1011: GTIM_Trgo 0101: UARTx_RXD signal 1100: Reserved 0110: RTC_OUT signal 1111: BTIMx_ETR signal <i>Caution 1: Do not select the Trgo output by BTIMx itself.</i> <i>Caution 2: When set to 0010, the BTIM1 trigger input signal source is VC1_OUT; BTIM2 trigger input signal source is VC2_OUT; In this case, the BTIM3 trigger input signal source is invalid.</i>



Bit field	Name	Permission	Function description														
6:3	RSTISRC	RW	Count reset signal (RSTI) source configuration <table><tr><td>0000: No reset signal</td><td>0111: ATIM_Trgo</td></tr><tr><td>0001: I2C_SCL/LVD_OUT signal</td><td>1000: BTIM1_Trgo</td></tr><tr><td>0010: VCx_OUT signal</td><td>1001: BTIM2_Trgo</td></tr><tr><td>0011: SPI_NCS signal</td><td>1010: BTIM3_Trgo</td></tr><tr><td>0100: No reset signal</td><td>1011: GTIM_Trgo</td></tr><tr><td>0101: UARTx_RXD signal</td><td>1100: No reset signal</td></tr><tr><td>0110: RTC_OUT signal</td><td>1111: BTIMx_ETR signal</td></tr></table> Caution 1: Reset the counter at the valid edge of the RSTI signal and generate an update event. Caution 2: Do not select the Trgo output by BTIMx itself. Caution 3: When set to 0010, the BTIM1 reset signal source is VC1_OUT; BTIM2 reset signal source is VC2_OUT; The BTIM3 reset signal source is invalid. Caution 4: When set to 0001, the BTIM1 reset signal source is I2C_SCL and the BTIM2/3 reset signal source is LVD_OUT.	0000: No reset signal	0111: ATIM_Trgo	0001: I2C_SCL/LVD_OUT signal	1000: BTIM1_Trgo	0010: VCx_OUT signal	1001: BTIM2_Trgo	0011: SPI_NCS signal	1010: BTIM3_Trgo	0100: No reset signal	1011: GTIM_Trgo	0101: UARTx_RXD signal	1100: No reset signal	0110: RTC_OUT signal	1111: BTIMx_ETR signal
0000: No reset signal	0111: ATIM_Trgo																
0001: I2C_SCL/LVD_OUT signal	1000: BTIM1_Trgo																
0010: VCx_OUT signal	1001: BTIM2_Trgo																
0011: SPI_NCS signal	1010: BTIM3_Trgo																
0100: No reset signal	1011: GTIM_Trgo																
0101: UARTx_RXD signal	1100: No reset signal																
0110: RTC_OUT signal	1111: BTIMx_ETR signal																
2:0	SMS	RW	Slave mode function configuration <table><tr><td>000: Disable slave mode</td></tr><tr><td>001: Trigger start mode: enable counter on rising edge of TRGI</td></tr><tr><td>010: Gated count mode: count PCLK at TRGI high level</td></tr><tr><td>011: External counting mode: counter incremented by 1 on rising edge of TRGI</td></tr></table>	000: Disable slave mode	001: Trigger start mode: enable counter on rising edge of TRGI	010: Gated count mode: count PCLK at TRGI high level	011: External counting mode: counter incremented by 1 on rising edge of TRGI										
000: Disable slave mode																	
001: Trigger start mode: enable counter on rising edge of TRGI																	
010: Gated count mode: count PCLK at TRGI high level																	
011: External counting mode: counter incremented by 1 on rising edge of TRGI																	



11.9.4 BTIMx_IER interrupt enable register

Address offset: 0x0C Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:7	RFU	-	Reserved bits, please keep the default value
6	TIE	RW	Trigger interrupt enable control 0: Disabled 1: Enabled
5:1	RFU	-	Reserved bits, please keep the default value
0	UIE	RW	Update interrupt enable control 0: Disabled 1: Enabled

11.9.5 BTIMx_ISR interrupt flag register

Address offset: 0x10 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:7	RFU	-	Reserved bits, please keep the default value
6	TIF	RO	Trigger interrupt flag 0: No trigger event has occurred 1: Trigger event has occurred
5:1	RFU	-	Reserved bits, please keep the default value
0	UIF	RO	Update interrupt flag 0: No update event has occurred 1: An update event has occurred

11.9.6 BTIMx_ICR interrupt flag clear register

Address offset: 0x18 Reset value: 0x0000 0041

Bit field	Name	Permission	Function description
31:7	RFU	-	Reserved bits, please keep the default value
6	TIF	R1W0	Trigger flag clear W0: Clear trigger flag W1: No function
5:1	RFU	-	Reserved bits, please keep the default value
0	UIF	R1W0	Update flag clear W0: Counter update flag cleared W1: No function



11.9.7 BTIMx_EGR event generation register

Address offset: 0x14 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:7	RFU	-	Reserved bits, please keep the default value
6	TG	WO	This bit is set to 1 by software to generate an event and is automatically cleared by hardware 0: No function 1: Software generated trigger event
5:1	RFU	-	Reserved bits, please keep the default value
0	UG	WO	This bit can be set to 1 by software to generate an event and automatically cleared by hardware 0: No function 1: Re-initialises the counter and generates a register update event. Note that the counter for the prescaler will also be cleared to zero, but the prescaler ratio is not affected.

11.9.8 BTIMx_CNT count register

Address offset: 0x24 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31	UIFCPY	RO	According to the value of the UIFREMAP bit field in the BTIMx_CR1 register, this bit field indicates a different meaning: - UIFREMAP = 0, this bit field is reserved and read as 0 - UIFREMAP = 1, this bit field indicates a read-only copy of the UIF bit of the BTIMx_ISR register
30:16	RFU	-	Reserved bits, please keep the default value
15:0	CNT	RW	Timer value



11.9.9 BTIMx_PSC prescaler register

Address offset: 0x28 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	PSC	RW	Value of the prescaler Counter clock frequency CK_CNT is equal to $f_{CK\_PSC} / (PSC[15:0] + 1)$

11.9.10 BTIMx_ARR auto reload register

Address offset: 0x2C Reset value: 0x0000 FFFF

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	ARR	RW	Timer reloads value



12 Low power timer (LPTIM)

12.1 Overview

The CW32L010 integrates a 16-bit low-power timer (LPTIM), which can realize the function of timing or counting external pulses with very low power consumption. By selecting an appropriate clock source and trigger signal, the function of waking up the system when it is in low-power sleep mode can be realized. There is a compare register inside the LPTIM, which can realize the compare output and PWM output, and can control the polarity of the output waveform. In addition, LPTIM can also be connected with a quadrature encoder to automatically count up and down.

12.2 Main features

- 16bit auto-reload counter
- Programmable prescaler supports 1, 2, 4, 8, 16, 32, 64, 128 division
- Working clock optional:
 - Internal clock: PCLK, LSI, LSE
 - External clock: LPTIM_CH1
- Support single mode and continuous mode
- Rich software and hardware trigger sources
- Clockless external pulse counting
- Input and output polarity selection
- Configurable input port filters
- Output configurable as single pulse, single set or PWM wave
- Quadrature code count function

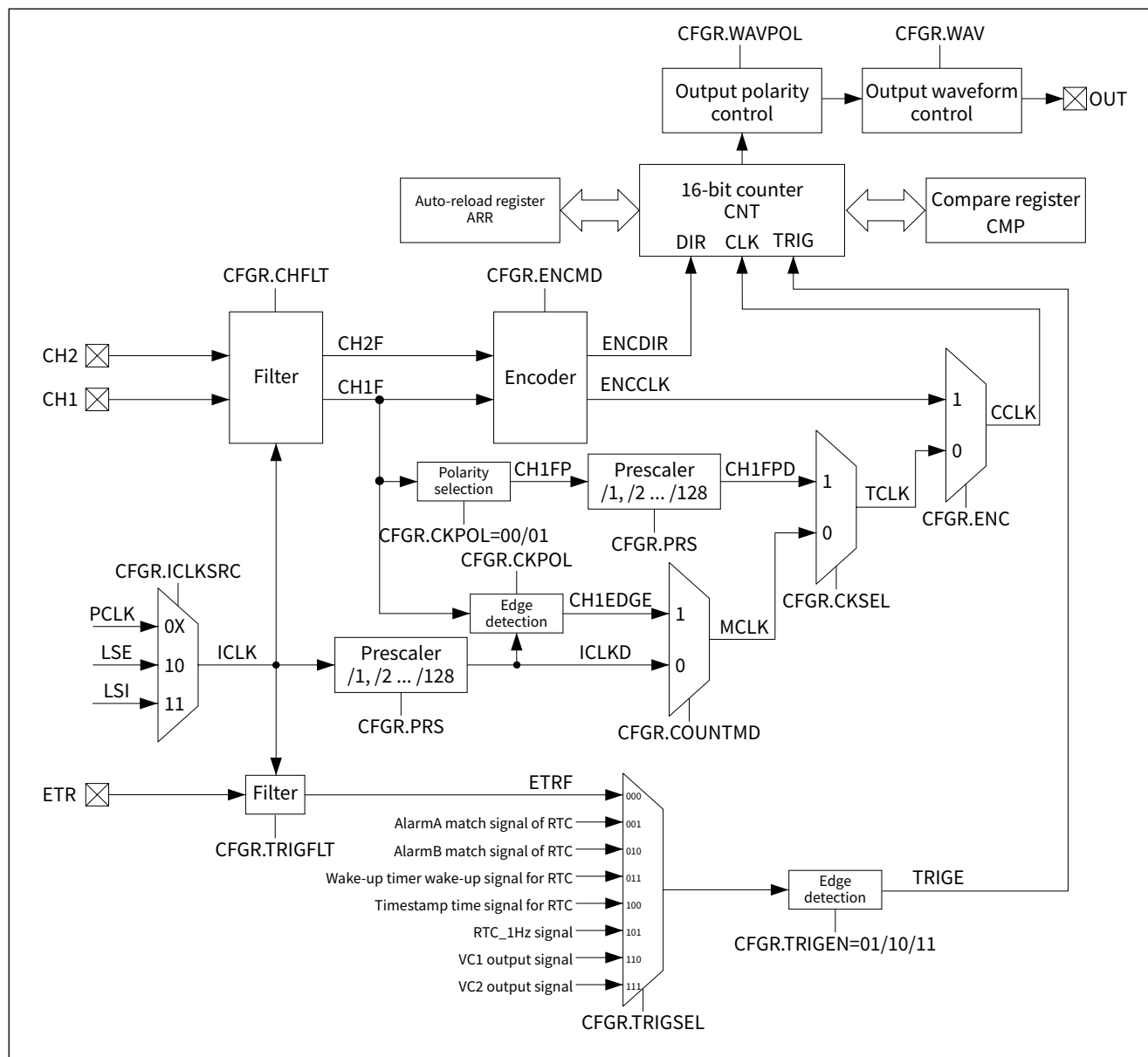


12.3 Function description

12.3.1 Functional block diagram

The functional block diagram of LPTIM is shown in the following figure:

Figure 12-1 LPTIM functional block diagram



12.3.1.1 Clock source

The counting clock source of LPTIM can be selected from PCLK, LSE, LSI, or an external clock input on the LPTIM_CH1 pin, which is configured through the ICLKSRC, COUNTMD and CKSEL bit fields of the configuration register LPTIM_CFGR.

When the external clock input on the LPTIM_CH1 pin is selected as the counting clock source, the LPTIM can work in the following two configuration modes:

- The first configuration mode (CKSEL=0, COUNTMD=1): The external clock input by CH1 is used as the count clock of LPTIM, and the internal clock module also needs to provide clock to LPTIM for filtering and edge detection;
- The second configuration mode (CKSEL=1): The external clock input by CH1 is used as the only clock of the LPTIM. This configuration can be used to implement the timeout function or pulse counter function when all the internal oscillators are turned off in the low power mode.

When configured to use an external clock source, the CKPOL bit field is used to set the valid edge of the external clock signal. If configured to be active on both upper and lower edges, the internal clock signal should also be provided (first configuration). In this case, the frequency of the internal clock signal should be at least four times the frequency of the external clock signal.

12.3.1.2 Prescaler

The prescaler divides the frequency of the counter clock source by a multiple of 2, and the maximum frequency division factor is 128. The prescaler coefficient is configured by the PRS bit field of the configuration register LPTIM_CFGR. The frequency division coefficient configuration is shown in the following table:

Table 12-1 Frequency division coefficient configuration table

PRS	Frequency division factor
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128



12.3.1.3 Filter unit

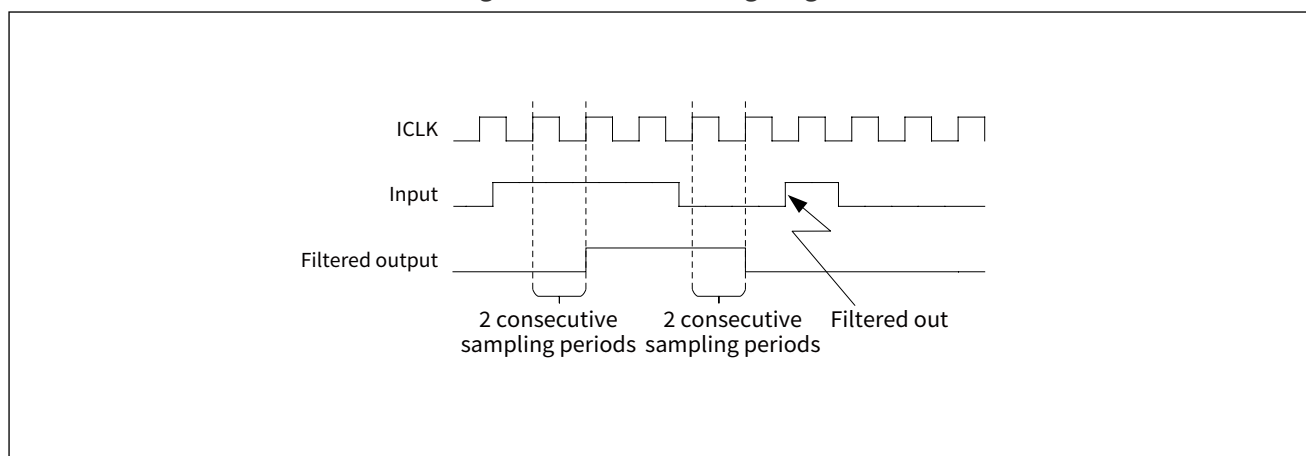
LPTIM inputs, whether external (mapped to GPIO) or internal (mapped on-chip to other on-chip peripherals), can be filtered through a digital filter to prevent glitches and noise perturbations from causing false LPTIM counts or triggering.

In order for the filter to work properly, an internal clock source should first be provided to the LPTIM. This is a necessary condition to ensure the normal operation of the filter. Digital filters can be divided into two groups:

- The first group of digital filters is used for the external input of LPTIM (CH1, CH2), and the sensitivity of the digital filters is controlled by the CHFLT bit;
- The first group of digital filters is used for the external input of LPTIM (CH1, CH2), and the sensitivity of the digital filters is controlled by the CHFLT bit;

The sensitivity of the filter can be changed by changing the number of active level holds that the filter continuously samples the input signal. For example, when the filter is set to an active level for 2 sampling periods, the filtered output waveform is shown in the following figure:

Figure 12-2 Filter timing diagram



Caution:

If no internal clock signal is provided, the digital filter must be disabled by setting the CHFLT and TRGFLT bit fields to '00'. In this case, an external analog filter can be used to shield the LPTIM external input from disturbances.

12.3.2 Start method

The EN bit field of the control register LPTIM_CR is used to enable/disable the LPTIM core. After setting the EN bit to 1, a delay of two count clock cycles is required before the LPTIM actually starts counting.

The LPTIM counter can be started by software or by a trigger event, set by the TRIGEN bit field of the configuration register LPTIM_CFGR.

When setting LPTIM_CFGR.TRIGEN to 00, it is configured for software startup. After starting LPTIM, setting the CNTSTART or SNGSTART bit field to 1 will start the LPTIM counter to count.

When LPTIM_CFGR.TRIGEN is set to 01/10/11, it is configured to trigger the event to start. After starting LPTIM and configuring the operating mode (single or continuous mode), once a valid edge of the trigger input is detected, the LPTIM counter is started to count. For specific trigger input signals, please refer to section [12.3.3 Trigger source](#).

Caution 1:

The LPTIM_CFGR and LPTIM_IER registers can only be modified when the EN bit field is 0 and LPTIM is disabled.

Caution 2:

LPTIM must be enabled (EN=1) before setting the CNTSTART or SNGSTART bit fields. When LPTIM is disabled, any writes to these bits will be discarded by hardware.

Caution 3:

When configured for software startup, after setting the CNTSTART or SNGSTART bit field, the counter does not actually start until after a delay of 3 core clock cycles.

Caution 4:

Since the trigger signal and the timing clock of the LPTIM are not synchronized, after the trigger detects the trigger signal, the timer needs to delay two timing clock cycles before it starts running.



12.3.3 Trigger source

The LPTIM counter can be started by software or after a valid edge of the trigger input is detected. LPTIM supports 8 trigger sources, which are set by the TRIGSEL bit field of the configuration register LPTIM_CFGR, as shown in the following table:

Table 12-2 LPTIM trigger input source table

TRIGSEL bit field	Trigger source
000	LPTIM_ETR pin signal
001	ALARMA match signal for RTC
010	ALARMB match signal for RTC
011	Wake-up timer wake-up signal for RTC
100	Timestamp event signal for RTC
101	RTC_1Hz signal
110	VC1 output signal
111	VC2 output signal

The TRIGEN bit field is used to set the trigger enable and polarity, as shown in the following table:

Table 12-3 LPTIM trigger enable and polarity

TRIGEN	Trigger Mode and Polarity
00	Software start, software writes CNTSTART or SNGSTART to start the counter
01	Hardware trigger, start the counter on the rising edge of the trigger signal
10	Hardware trigger, start the counter on the falling edge of the trigger signal
11	Start the counter on the rising or falling edge of the trigger signal

Caution: If a new trigger event occurs when the timer has already started, it will be ignored (unless the timeout function is enabled).



12.3.4 Operating mode

LPTIM supports two modes of operation:

- Single mode: the counter is started by software or triggered by an event, and stops when the count value reaches the reload value ARR;
- Continuous mode: The counter is started by software or by a trigger event, the counter counts continuously and stops until the timer is disabled.

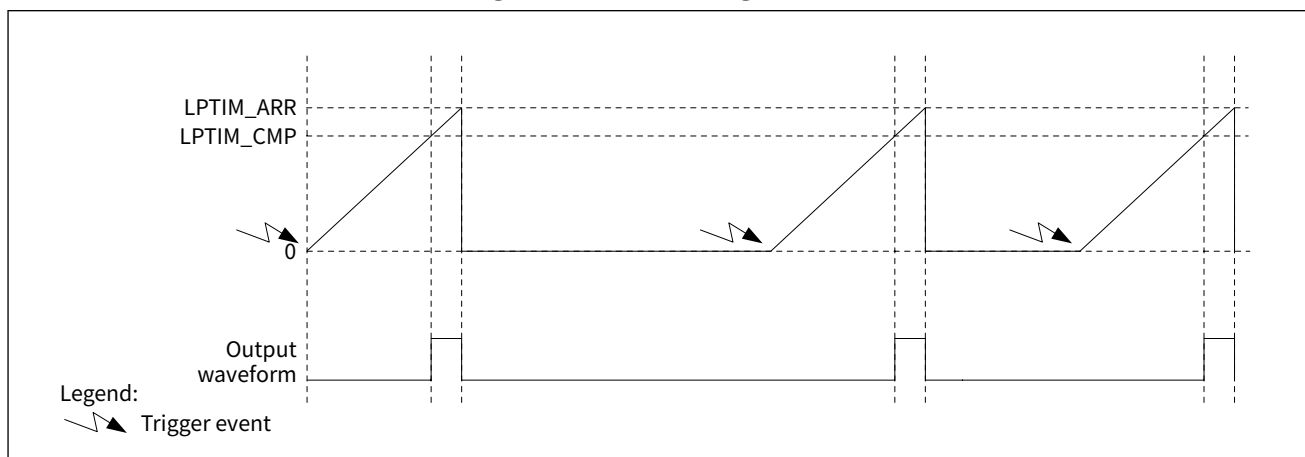
Single mode

In the single-counting mode, the counter can be started by software or triggered by an event, and the counter CNT is driven by the TCLK clock to count up. When the count value is equal to the reload value ARR, the automatic reload match flag LPTIM_ISR.ARRM is set by hardware, and the counter stops counting at the same time.

If it is set to software start, after setting LPTIM_CR.EN to 1, set LPTIM_CR.SNGSTART to 1 to start the counter to count.

If set to trigger event start, set LPTIM_CR.EN to 1 and set LPTIM_CR.SNGSTART to 1 to make LPTIM run in single shot mode. After that, a trigger event will start the counter to count. Any trigger events after the counter starts and before the counter reaches the ARR will be discarded. After the counter is stopped, a new trigger event will start the counter again and start counting. As shown below:

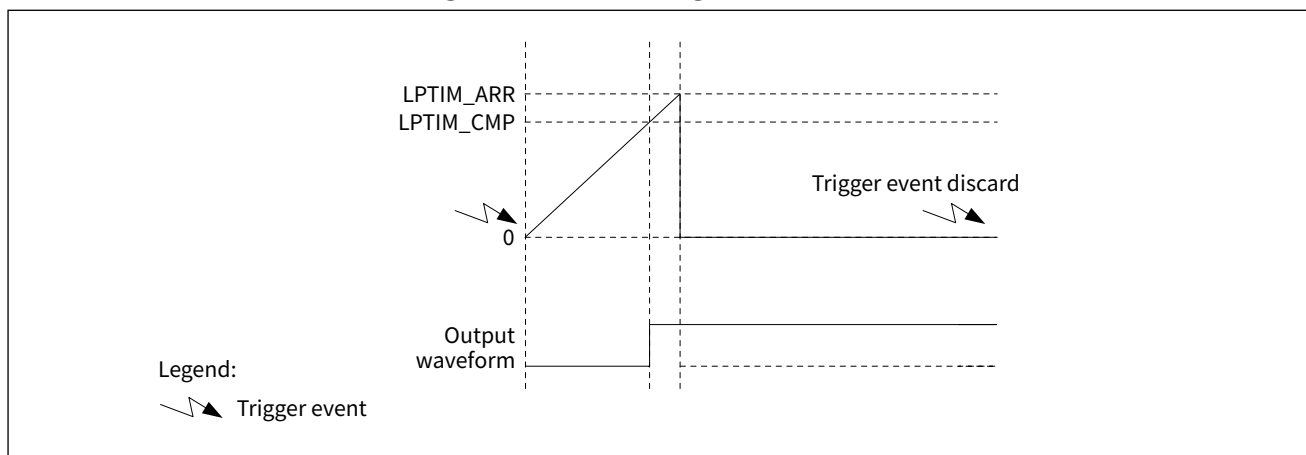
Figure 12-3 LPTIM single mode



Single set output

It should be noted that when the WAVE bit field of the configuration register LPTIM_CFGR is set to 1, the single set mode will be activated. In one-shot mode, the counter is only started once on the first trigger event, and any subsequent trigger events will be discarded, as shown in the following figure:

Figure 12-4 LPTIM single set waveform



If it is set to software start (TRIGEN='00'), after SNGSTART is set to 1, it will start the counter for one count.

Continuous mode

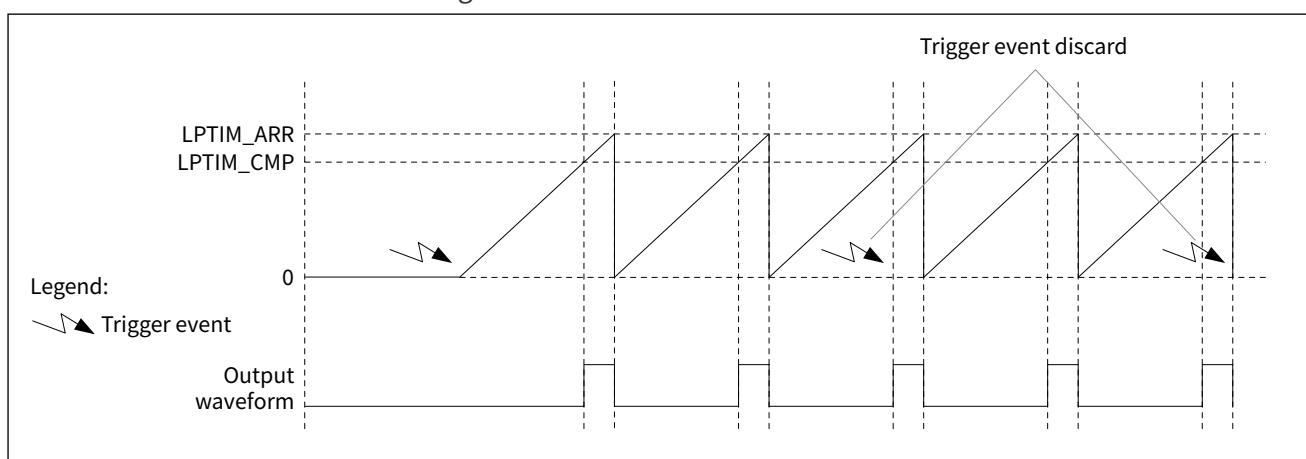
In continuous counting mode, the counter can be started by software or triggered by an event, and the counter CNT counts up under the drive of the TCLK clock. When the count value is equal to the reload value ARR, the automatic reload match flag LPTIM_ISR.ARRM is set by hardware, the counter overflows and starts to count up in the next cycle, and it stops counting until LPTIM_CR.EN is set to 0.

If it is set to software start, after setting LPTIM_CR.EN to 1, set LPTIM_CR.CNTSTART to 1 to start the counter for continuous counting.

If set to trigger event start, set LPTIM_CR.EN to 1 and set LPTIM_CR.CNTSTART to 1 to make LPTIM run in continuous mode. Thereafter, a trigger event will start the counter to count continuously and any subsequent trigger events will be discarded.

The figure below shows an example of counting and output in continuous mode.

Figure 12-5 LPTIM continuous mode



Single mode and continuous mode can be switched between each other during operation.

If previously configured for continuous mode, setting the SNGSTART bit field will cause the LPTIM to switch to one-shot mode. The counter (if activated) will stop as soon as the reload value ARR is reached.

If previously configured for single mode, setting the CNTSTART bit field will cause the LPTIM to switch to continuous mode. The counter (if activated) will restart as soon as the reload value ARR is reached.

12.3.5 Timeout function

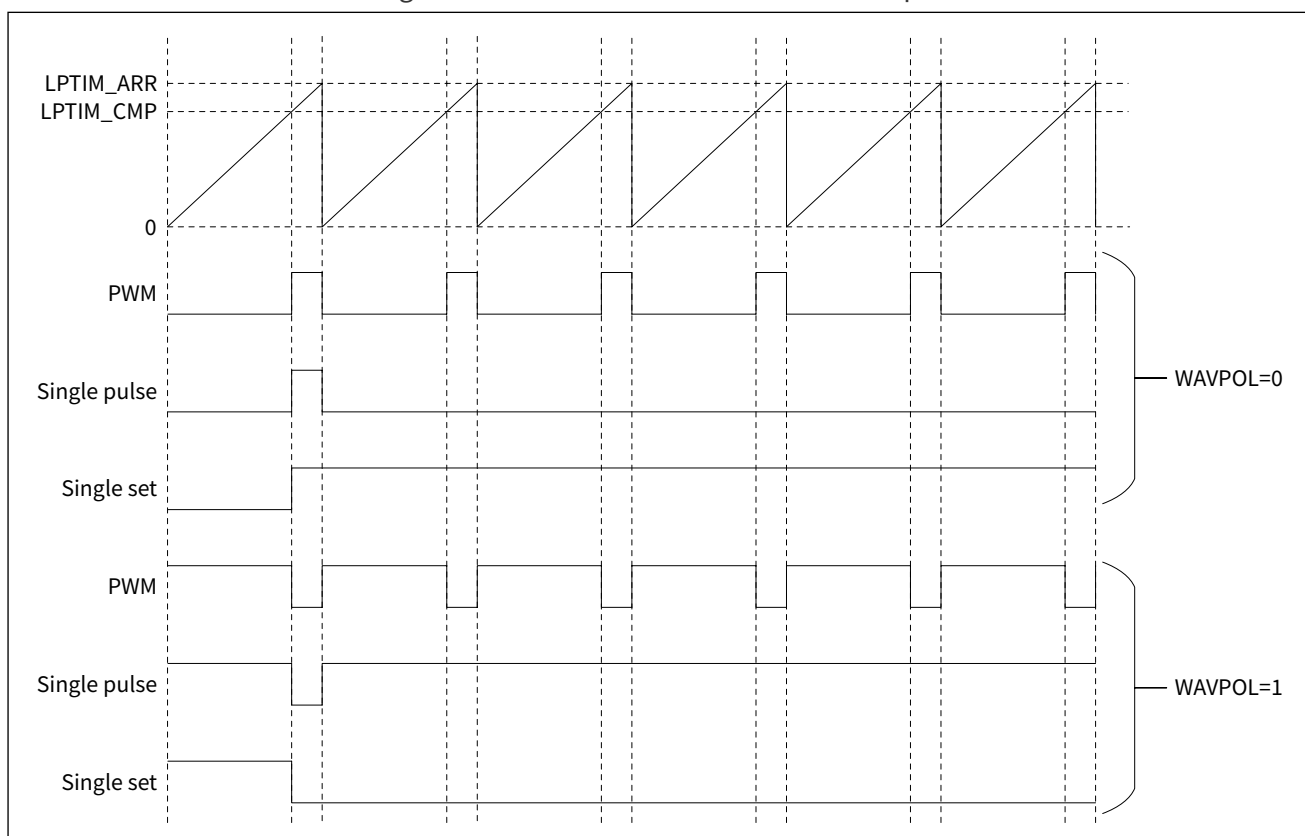
If the TIMOUT bit field is set to 1, when LPTIM is counting, the counter can be reset by a trigger signal, and the counter will start to count again from 0.

This feature can implement a low-power timeout wake-up function, and the timeout threshold is determined by the compare register LPTIM_CMP. When the trigger signal is not generated within the expected time range, that is, after the trigger signal times out, the counter counts to the comparison value CMP, and LPTIM will generate a compare match interrupt to wake up the MCU.

12.3.6 Waveform output control

LPTIM can output PWM wave in continuous mode, single pulse wave or single set waveform can be output in single mode, and the polarity of output waveform can be controlled by WAVPOL bit. When the WAVPOL bit is 0, it is a forward output; when the WAVPOL bit is 1, it is a reverse output. Change the WAVPOL bit field and the polarity change of the output waveform will take effect immediately. When LPTIM is not enabled, modifying the value of WAVPOL will also change the default level polarity of the output. The figure below shows the three waveforms that can be generated on the LPTIM output, in addition, it shows the effect of the WAVPOL bit on the polarity change.

Figure 12-6 Three waveforms of LPTIM output



12.3.7 Register update

The PRELOAD bit field of the configuration register LPTIM_CFGR is used to set the data loading time of the auto-reload register LPTIM_ARR and the compare register LPTIM_CMP. When PRELOAD is set to 0, the LPTIM_ARR and LPTIM_CMP registers will be updated immediately after the write operation; when PRELOAD is set to 1, the value of the write operation is loaded when the count value reaches the previous ARR value (if the timer has already started).

Since the APB interface of LPTIM and the core of LPTIM use different clocks, when writing the LPTIM_ARR register and the LPTIM_CMP register, there is some delay between writing the value to the register and updating the value of the register to the ARR and CMP of the LPTIM. During this delay period, any additional writes to these registers must be avoided. Whether the ARR and CMP are updated can be judged by the ARROK flag and the CMPOK flag in the LPTIM_ISR register.

The ARROK flag and the CMPOK flag in the LPTIM_ISR register are 0, indicating that the value of the corresponding register is being updated. Any writing to the LPTIM_ARR register and the LPTIM_CMP register at this time will lead to unpredictable results. A new write operation to the same register must wait until the ARROK flag and the CMPOK flag in the LPTIM_ISR register are 1, that is, when the register has been updated.

12.3.8 Count function

LPTIM can count the signal input on the external LPTIM_CH1 pin, and can also count the internal clock signal. The count clock source is configured through the CKSEL and COUNTMD bit fields of the configuration register LPTIM_CFGR.

- When CKSEL is 0, LPTIM operation requires an internal clock ICLK.
If COUNTMD is 0, LPTIM counts the pulses of the internal clock source;
If COUNTMD is 1, the LPTIM samples the input signal on the LPTIM_CH1 pin with the internal clock. Therefore, in order not to miss any events, the CH1 signal should not change more frequently than the internal clock supplied to the LPTIM. Also, the internal clock supplied to LPTIM cannot be prescaled (PRS=000). By setting the value of the CKPOL bit field, you can choose to count on the rising edge, falling edge or upper and lower edges of the CH1 signal.
- When CKSEL is 1, LPTIM operation is provided by an external clock source.
In this mode, the value of the COUNTMD bit field has no effect on LPTIM, which does not require an internal clock source (unless the filter is enabled). The external signal input on CH1 of LPTIM is used as the count of LPTIM. For this configuration, the LPTIM counter can count on the rising or falling edge of the clock signal on CH1, but cannot be updated on both the rising and falling edges. Since the signal input to LPTIM's CH1 is also used to clock the LPTIM core logic, after starting LPTIM, the first 3 count clock edges are used for the configuration of the core, and the counter will not start counting until the 4th count clock edge (see [12.5.1 External pulse counting routine](#)).



12.3.9 Encoding mode

LPTIM can process signals from quadrature encoders. The quadrature encoder indicates the position and rotation direction of the rotating element by outputting two pulse signals with a phase difference of 90 degrees. LPTIM in encoder mode is equivalent to connecting an external clock with direction selection, which means that the counter only counts continuously between 0 and the auto-reload value ARR (counting up from 0 to ARR depending on the direction, or counts down from ARR to 0). Therefore, the auto-reload register LPTIM_ARR must be configured before startup.

The two output signals of the encoder generate clock signals through the two external input ports CH1 and CH2 of the LPTIM as the LPTIM counter clock. The phase between these two signals determines the count direction. In encoding mode, the LPTIM must be clocked by the internal clock source, and the frequency of the signals input to CH1 and CH2 must not exceed the 4-frequency division of the LPTIM internal clock frequency, otherwise the LPTIM will not work properly.

Direction changes are indicated by the UP and DOWN flag bits in the LPTIM_ISR register. In addition, if the corresponding enable bit in the LPTIM_IER register is set, an interrupt will be triggered when both direction change events occur.

Before using encoder mode, the LPTIM must first be configured for continuous mode, and then set the ENC bit to 1 to enable encoder mode. After enabling, the LPTIM counter will automatically modify the count value of LPTIM_CNT according to the speed and direction of the encoder. Therefore, the count value of LPTIM_CNT represents the position of the encoder. The counting direction is indicated by the UP and DOWN signs, corresponding to the direction of rotation of the connected sensor.

LPTIM supports 3 encoding counting modes: set LPTIM_CFGR.ENCMD to 0x00, the counter will only count on the edge of CH1; set LPTIM_CFGR.ENCMD to 0x01, the counter will only count on the edge of CH2; set LPTIM_CFGR.ENCMD to 0x02, the counter will count on the edge of both CH1 and CH2.

The relationship between counting direction and encoder signal is shown in the following table:

Table 12-4 Quadrature encoding mode count and direction

Mode	Signal level		CH1		CH2	
	CH1	CH2	Rise	Decline	Rise	Decline
Mode 1	-	High	Count down	Count up	Does not count	Does not count
	-	Low	Count up	Count down	Does not count	Does not count
Mode 2	High	-	Does not count	Does not count	Count up	Count down
	Low	-	Does not count	Does not count	Count down	Count up
Mode 3	High	High	Count down	Count up	Count up	Count down
	Low	Low	Count up	Count down	Count down	Count up

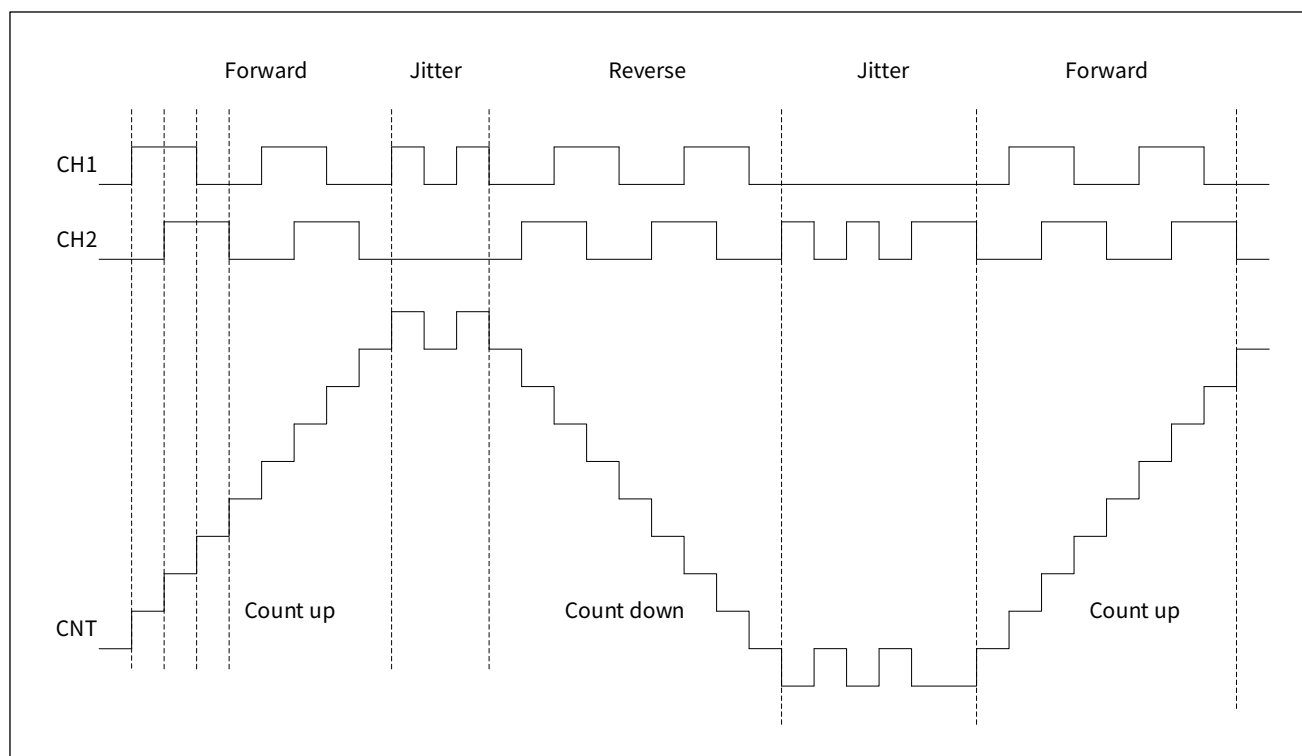
Caution:

In this mode, the LPTIM must be clocked by an internal clock source, so the CKSEL bits must remain at their reset value of 0. In addition, the prescaler division ratio must be equal to its reset value of 1 (PRS bit field must be "000").

The figure below is an example of encoder mode counting, showing the generation and direction control of the count signal.



Figure 12-7 Encoder mode count example



12.4 Debug support

LPTIM supports stop or continue counting in debug mode, which is set by the LPTIM bit field of the debug status timer control register SYSCTRL_DEBUG.

- Set SYSCTRL_DEBUG.LPTIM to 1 to suspend the counter count of LPTIM in debug state;
- Set SYSCTRL_DEBUG.LPTIM to 0, the counter of LPTIM continues to count in debug state.



12.5 Programming example

12.5.1 External pulse counting routine

Step 1: Set SYSCTRL_APBEN2.LPTIM to 1 to enable the configuration clock of LPTIM;

Step 2: Configure the GPIO corresponding to the LPTIM_CH1 pin to the multiplexed input mode. For the specific register configuration, please refer to chapter [8 General-purpose input/output \(GPIO\)](#);

Step 3: Set LPTIM_CFGR.COUNTMD, LPTIM_CFGR.CKSEL to 1, configure LPTIM counting mode, the counting clock is provided by CH1;

Step 4: Set LPTIM_IER.ARRM to 1 to enable ARR auto-reload matching interrupt;

Step 5: Set LPTIM_ARR to 6, the purpose is to trigger the ARR automatic reload matching interrupt after obtaining 10 external count pulses, because the first 3 pulses, CNT does not count;

Step 6: Set LPTIM_CR.EN to 1 to enable LPTIM.



12.6 List of registers

LPTIM base address: LPTIM_BASE = 0x4000 7800

Table 12-5 List of LPTIM registers

Register name	Register address	Register description
LPTIM_ISR	LPTIM_BASE + 0x00	Interrupt and status register
LPTIM_ICR	LPTIM_BASE + 0x04	Interrupt clear register
LPTIM_IER	LPTIM_BASE + 0x08	Interrupt enable register
LPTIM_CFGR	LPTIM_BASE + 0x0C	Configuration register
LPTIM_CR	LPTIM_BASE + 0x10	Control register
LPTIM_CMP	LPTIM_BASE + 0x14	Compare register
LPTIM_ARR	LPTIM_BASE + 0x18	Auto-reload register
LPTIM_CNT	LPTIM_BASE + 0x1C	Counter register



12.7 Register descriptions

See section [1 Documentation conventions](#) for abbreviations used in register descriptions.

12.7.1 PTIM_CFGR Configuration register

Address offset: 0x0C Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:27	RFU	-	Reserved bits, please keep the default value
26:25	ICLKSRC	RW	ICLK source configuration 00: PCLK 01: PCLK 10: LSE 11: LSI
24	ENC	RW	Encoder mode enable 0: Disabled 1: Enabled <i>Caution: Encoding circuits require ICLK to function properly.</i>
23	COUNTMD	RW	Counting mode configuration 0: Timing mode, the divided clock of ICLK drives the counter to count 1: Counting mode, the edge of the CH1 pin drives the counter to count
22	PRELOAD	RW	ARR and CMP register update mode 0: Update registers immediately 1: Update the registers when the LPTIM cycle ends
21	WAVPOL	RW	Polarity configuration of output waveforms 0: PWM positive output, $CNT \geqslant CMP$ output high level 1: PWM reverse output, $CNT < CMP$ output high level
20	WAVE	RW	Waveform control 0: Single start SNGSTRT corresponds to single pulse waveform, continuous start CNTSTRT corresponds to PWM waveform 1: Single set waveform
19	TIMOUT	RW	Trigger timeout enable 0: Trigger events that occur while counting are ignored 1: A trigger event that occurs while counting is in progress will clear the count and restart counting



Bit field	Name	Permission	Function description
18:17	TRIGEN	RW	Trigger enable and polarity 00: Software trigger, start the counter by software writing to the register 01: Hardware trigger, start the counter on the rising edge of the trigger signal 10: Hardware trigger, start the counter on the falling edge of the trigger signal 11: Hardware trigger, start the counter on the rising or falling edge of the trigger signal
16	RFU	-	Reserved bits, please keep the default value
15:13	TRIGSEL	RW	Trigger source selection 000: LPTIM_ETR pin signal 001: ALARMA match signal for RTC 010: ALARMB match signal for RTC 011: Wake-up timer wake-up signal for RTC 100: Timestamp event signal for RTC 101: RTC_1Hz signal 110: VC1 output signal 111: VC2 output signal
12	RFU	-	Reserved bits, please keep the default value
11:9	PRS	RW	ICLK/CH1FP signal frequency division configuration 000: DIV1 001: DIV2 010: DIV4 011: DIV8 100: DIV16 101: DIV32 110: DIV64 111: DIV128 <i>Caution 1: When COUNTMD is 1 and CKSEL is 0, only 000 can be configured.</i> <i>Caution 2: When operating in encoder mode, only 000 can be configured.</i>
8	RFU	-	Reserved bits, please keep the default value
7:6	TRIGFLT	RW	Trigger signal digital filter configuration 00: No filter functions 01: 2 ICLK clock cycles 10: 4 ICLK clock cycles 11: 8 ICLK clock cycles <i>Caution 1: When the ICLK clock is stopped, it can only be configured as no filter function.</i> <i>Caution 2: When the trigger source selects the RTC related signal please do not set the filter time.</i>
5	RFU	-	Reserved bits, please keep the default value



Bit field	Name	Permission	Function description
4:3	CHFLT	RW	CH1 / CH2 channel digital filter configuration 00: No filter functions 01: 2 ICLK clock cycles 10: 4 ICLK clock cycles 11: 8 ICLK clock cycles <i>Caution: Configurable only for no filtering when ICLK is stopped.</i>
2:1	ENCMD	RW	Encoder working mode configuration 00: Code counting mode 1, CH1 signal change edge counting 01: Code counting mode 2, CH2 signal change edge counting 10: Code counting mode 3, CH1/CH2 signal change edge counting 11: Reserved
2:1	CKPOL	RW	CH1 signal count edge configuration 00: The counter increments on the rising edge of the CH1 signal 01: The counter increments on the falling edge of CH1 signal 10: The counter increments when the CH1 signal rises or falls 11: Reserved <i>Caution: When CKSEL is 1, only configurable as 00/01.</i>
0	CKSEL	RW	Counter count clock TCLK source configuration 0: The count clock comes from MCLK and counts PCLK / LSE / LSI / CH1 edges 1: The count clock comes from CH1FPD, and counts the clock input by CH1 <i>Caution: In encoder mode, only 0 can be configured.</i>

Caution:

This register can only be set when LPTIM is disabled.



12.7.2 LPTIM_CR Control register

Address offset: 0x10 Reset value: 0x0000 0018

Bit field	Name	Permission	Function description
31:5	RFU	-	Reserved bits, please keep the default value
4	ARST	RW	Asynchronous reset of counter count value W0: Reset counter count value W1: No function R0: The counter count value is being reset R1: Counter count value reset completed <i>Caution: 0 can be written to reset only when this control bit is read as 1.</i>
3	SRST	RW	Synchronous reset of counter count value W0: Reset counter count value W1: No function R0: The counter count value is being reset R1: Counter count value reset completed <i>Caution: 0 can be written to reset only when this control bit is read as 1.</i>
2	CNTSTART	RW	Continuous Mode Operation Control W0: No function W1: Start continuous mode
1	SNGSTART	RW	One-shot mode operation control W0: No function W1: Start single-pulse mode
0	EN	RW	LPTIM enable control 0: Disable LPTIM 1: Enable LPTIM

12.7.3 LPTIM_ARR Auto-reload register

Address offset: 0x18 Reset value: 0x0000 0001

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	ARR	RW	Auto-reload value



12.7.4 LPTIM_CNT Counter register

Address offset: 0x1C Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	CNT	RO	Counter value <i>Caution: Reading the same value twice in a row is considered read data.</i>

12.7.5 LPTIM_CMP Compare register

Address offset: 0x14 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	CMP	RW	Comparison value



12.7.6 LPTIM_IER Interrupt enable register

Address offset: 0x08 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:7	RFU	-	Reserved bits, please keep the default value
6	DOWN	RW	Counting direction changes to reverse interrupt enable control 0: Disable 1: Enable
5	UP	RW	Counting direction changes to positive interrupt enable control 0: Disable 1: Enable
4	ARROK	RW	Auto-reload register update complete interrupt enable control 0: Disable 1: Enable
3	CMPOK	RW	Compare register update completion interrupt enable control 0: Disable 1: Enable
2	EXTTRIG	RW	Trigger completion interrupt enable control 0: Disable 1: Enable
1	ARRM	RW	Auto-reload match interrupt enable control 0: Disable 1: Enable
0	CMPM	RW	Compare match interrupt enable control 0: Disable 1: Enable

Caution:*This register can only be set when LPTIM is disabled.*

12.7.7 LPTIM_ISR Interrupt and status register

Address offset: 0x00 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:9	RFU	-	Reserved bits, please keep the default value
8:7	DIR	RO	Encoder count direction 00: The encoder stops counting, and the current counting direction is decrement 01: The encoder stops counting, and the current counting direction is incrementing 10: The encoder is counting, the current counting direction is decrement 11: The encoder is counting, and the current counting direction is incrementing
6	DOWN	RO	Encoder counting direction becomes reverse sign 0: The counting direction has not changed 1: Counting direction changes from incrementing to decrementing
5	UP	RO	Encoder counting direction changes to positive sign 0: The counting direction has not changed 1: Counting direction changes from decrementing to incrementing
4	ARROK	RO	Auto-reload register update completion flag 0: Auto-reload registers are being updated 1: Auto-reload registers has been updated
3	CMPOK	RO	Compare register update completion flag 0: Compare register is being updated 1: The compare register has been updated
2	EXTTRIG	RO	Trigger completion flag 0: No trigger event occurred 1: Triggering event has occurred
1	ARRM	RO	Auto-reload match flag 0: The counter count value is not equal to the ARR register value 1: The counter count value is equal to the ARR register value
0	CMPM	RO	Compare match flags 0: The count value of the counter is not equal to the value of the CMP register 1: The count value of the counter is equal to the value of the CMP register



12.7.8 LPTIM_ICR Interrupt clear register

Address offset: 0x04 Reset value: 0x0000 007F

Bit field	Name	Permission	Function description
31:7	RFU	-	Reserved bits, please keep the default value
6	DOWN	R1W0	Counting direction changes to reverse flag clear W0: Clear the count direction to reverse flag W1: No function
5	UP	R1W0	Counting direction changes to positive flag clear W0: Clear the count direction to positive flag W1: No function
4	ARROK	R1W0	Auto-reload register update complete flag cleared W0: Clear auto-reload register update done flag W1: No function
3	CMPOK	R1W0	Compare register update complete flag is cleared W0: Clear compare register update complete flag W1: No function
2	EXTTRIG	R1W0	Trigger completion flag clear W0: Clear trigger completion flag W1: No function
1	ARRM	R1W0	Auto-reload match flag clear W0: Clear auto-reload match flag W1: No function
0	CMPM	R1W0	Compare match flag clear W0: Clear compare match flag W1: No function



13 General-purpose timer (GTIM)

13.1 Overview

The general-purpose timer (GTIM) consists of a 16bit auto-reload counter and is driven by a programmable prescaler. GTIM supports multiple modes of operation such as timing, counting, reset, gating, trigger and encoder, with 4 independent capture/compare channels that measure the pulse width of the input signal (input capture) or generate output waveforms (output compare and PWM).

13.2 Main features

- 16bit increment, decrement and increment/decrement auto reload counter
- Programmable prescaler supports 1,2,3,4,...,65536 frequency division
- Support single count mode and continuous count mode
- 4 independent input capture and output compare channels
- Trigger input signal (TRGI) to control timer for multiple slave modes
- Timer cascade ITR and on-chip peripheral interconnect ETR
- Support incremental (quadrature) encoders and Hall sensor circuits for positioning
- Interrupt generated when various events occur
 - Update event
 - Trigger event
 - Input capture
 - Output compare

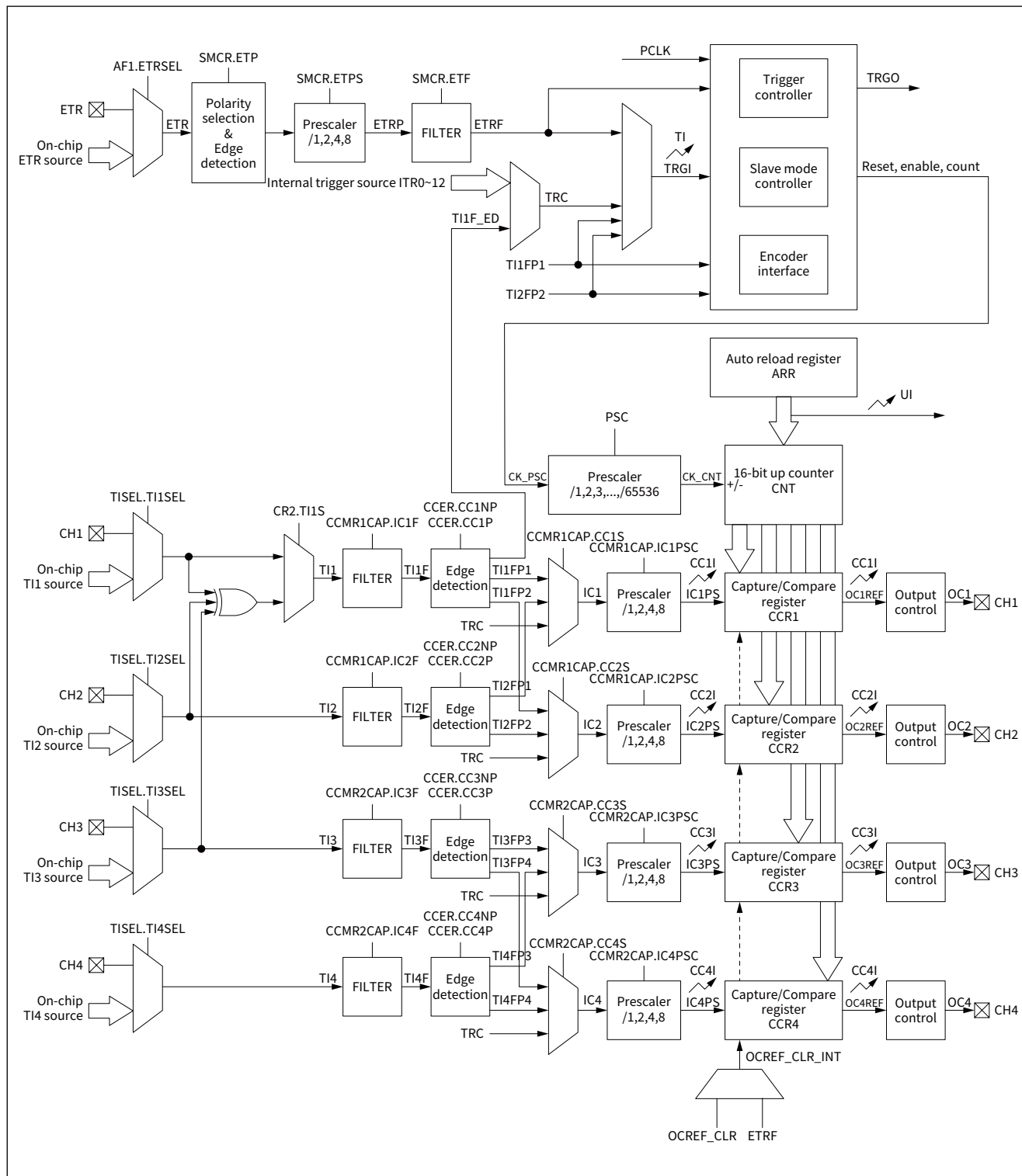


13.3 Functional description

13.3.1 Functional block diagram

The functional block diagram of GTIM is shown in the following figure:

Figure 13-1 GTIM functional block diagram



13.3.1.1 Clock source

The counter's counting clock source can be provided by the internal clock PCLK, the trigger signal TRGI or the external trigger input signal ETRF, which is divided by the prescaler GTIM_PSC to drive the counter for counting. Different operating modes have different clock sources, more details please refer to [13.3.2 Operating modes](#).

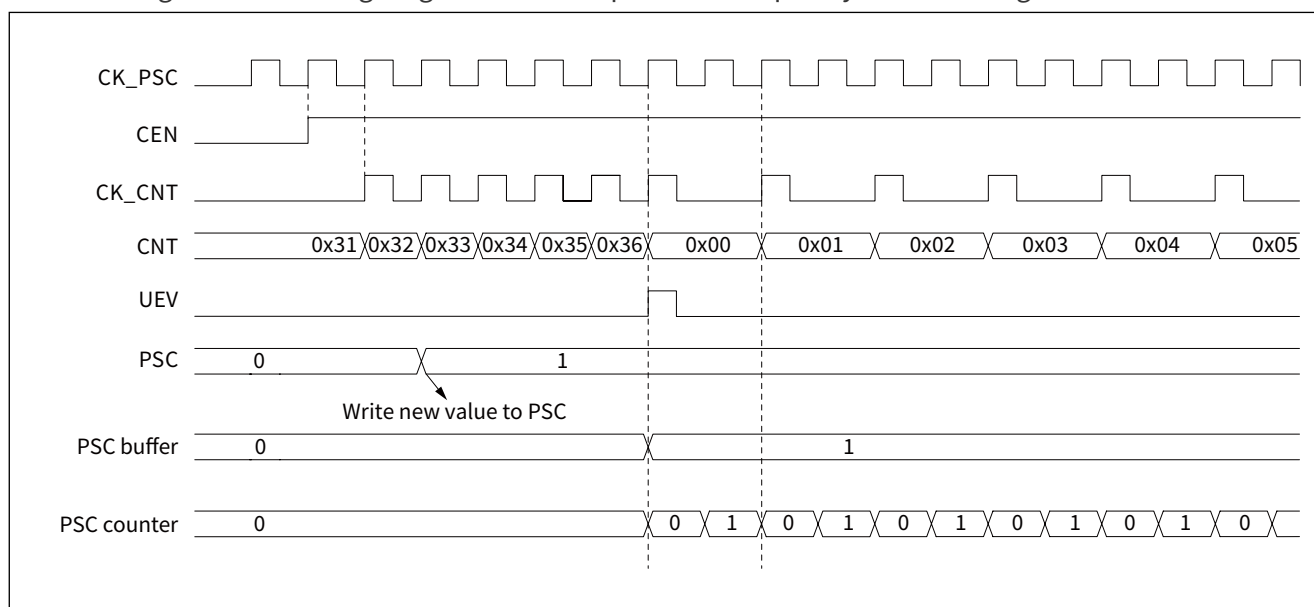
13.3.1.2 Prescaler

The prescaler divides the CK_PSC clock to get the count clock CK_CNT to drive the counter to count. The divider coefficient is set through the GTIM_PSC register, which supports 1,2,3,4,...,65536 divider.

The GTIM_PSC register is buffered and can be modified during runtime, and the new prescaler value will take effect when the next update event occurs.

The following figure gives the timing diagram when the prescaler divider frequency changes from 1 to 2 during operation:

Figure 13-2 Timing diagram when the prescaler frequency divider changes from 1 to 2



13.3.1.3 Counter and count mode

The counter can be set to count up (edge-aligned mode), count down (edge-aligned mode), or count up/down (center-aligned mode). Specifically, it is configured through the CMS and DIR bit fields of the control register GTIM_CR1, as shown in the following table:

Table 13-1 Count mode

GTIM_CR1.CMS	GTIM_CR1.DIR	Count mode
00	0	Count up (edge-aligned mode)
	1	Count down (edge-aligned mode)
01	-	Center-aligned mode 1
10	-	Center-aligned mode 2
11	-	Center-aligned mode 3

When the CEN bit field of the GTIM_CR1 register is set to 1, the counter starts counting in the set mode, noting that the actual counter enable signal, CNT_EN, is set to 1 one clock cycle after CEN is set to 1.

Count up mode

In count up mode, the counter starts counting up from 0 to the reload value ARR and then starts counting up again from 0 while generating a counter overflow event.

When the counter overflows, the overflow signal OV is generated, and the update event UEV is generated (the OV signal and the UEV signal are cleared automatically), and the counter update interrupt flag bit GTIM_ISR.UIF is set by the hardware, and if an interrupt is permitted (by setting GTIM_IER.UIE to 1), an interrupt request is generated, and setting GTIM_ICR.UIF to 0 to clear this flag bit.

The following is an example of counter operation at different clock frequencies, where GTIM_ARR = 0x36:

Figure 13-3 Count up, internal clock frequency division factor is 1

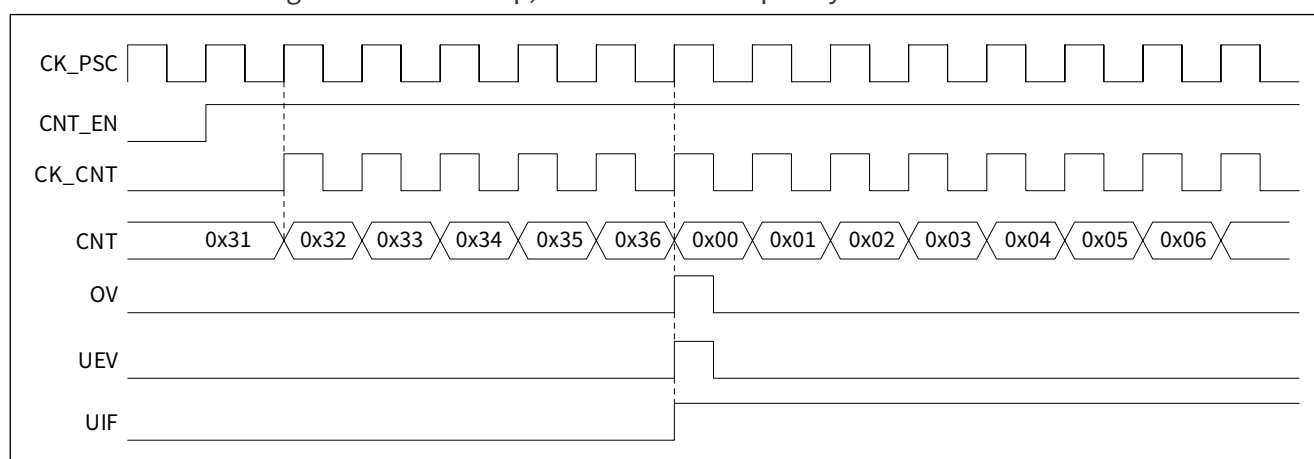


Figure 13-4 Count up, internal clock frequency division factor is 2

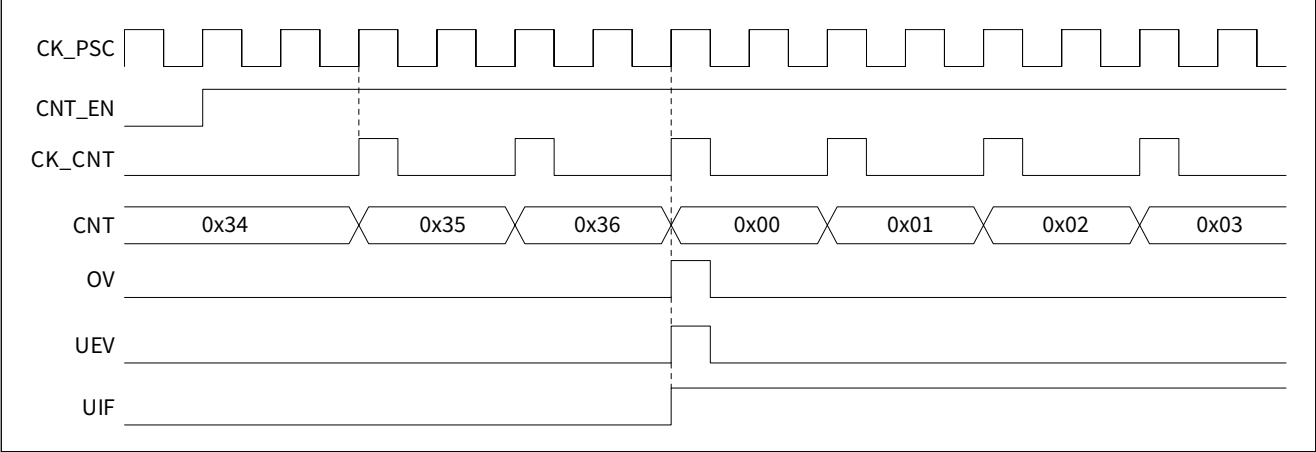


Figure 13-5 Count up, internal clock frequency division factor is 4

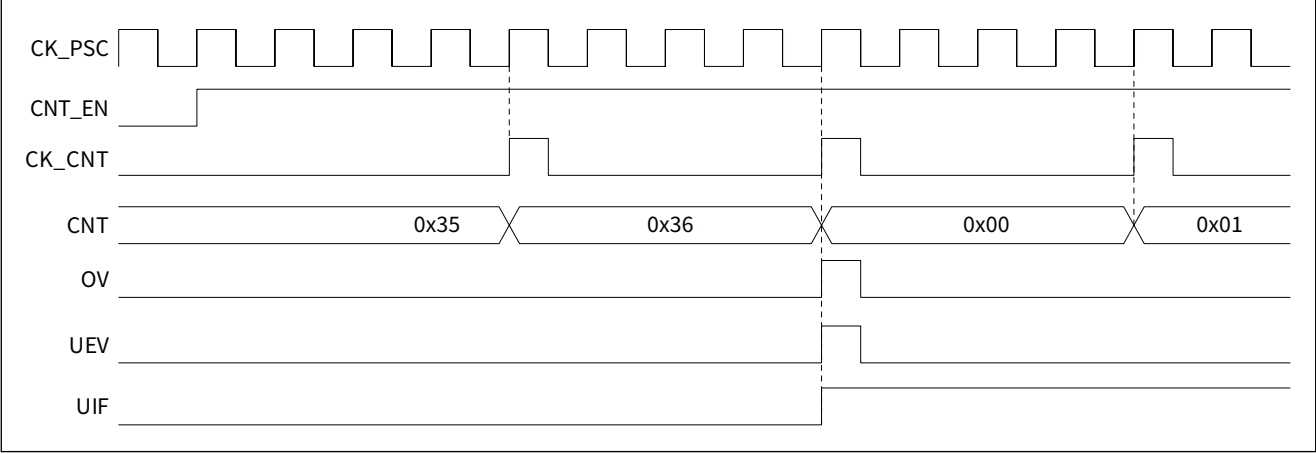


Figure 13-6 Count up, internal clock frequency division factor is N

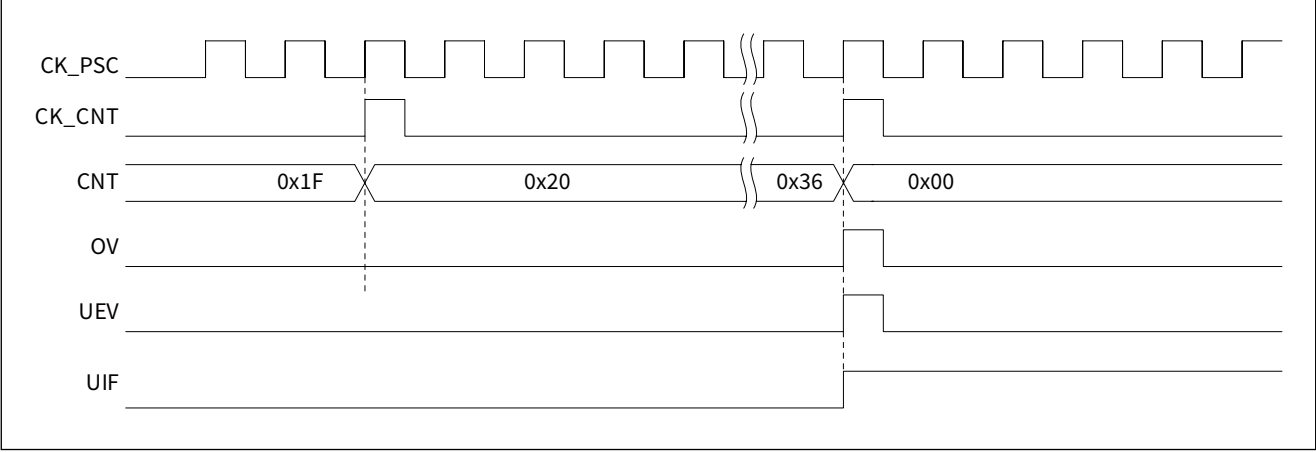


Figure 13-7 Count up, update event when reloading cache is disabled (GTIM_CR1.ARPE=0)

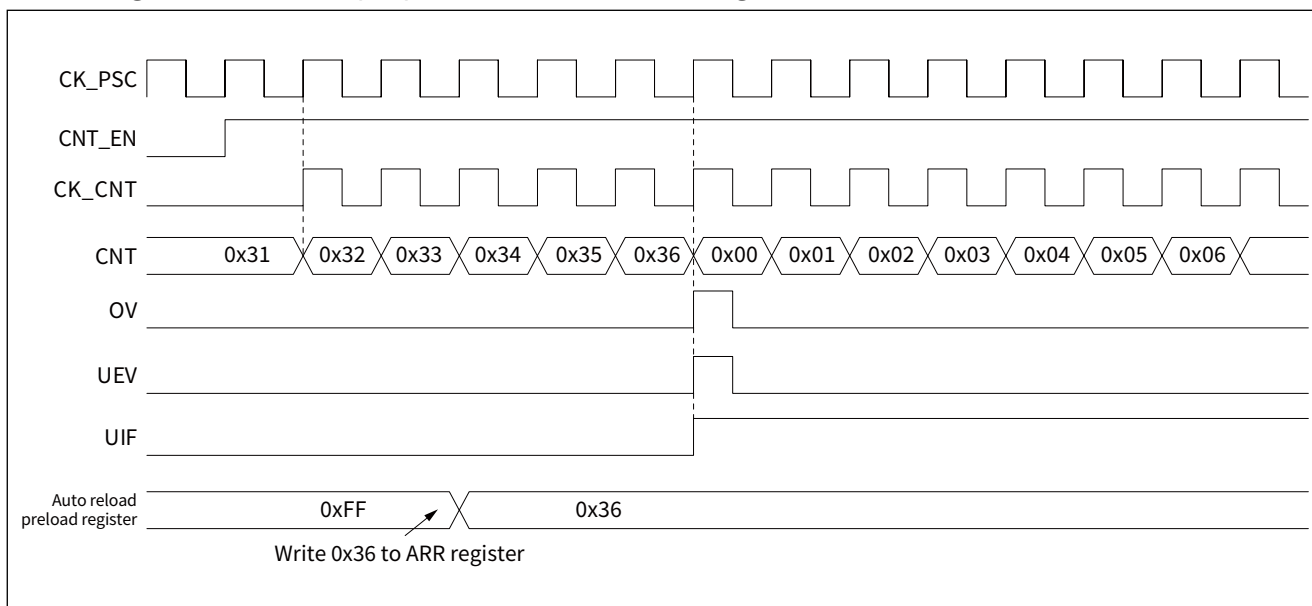
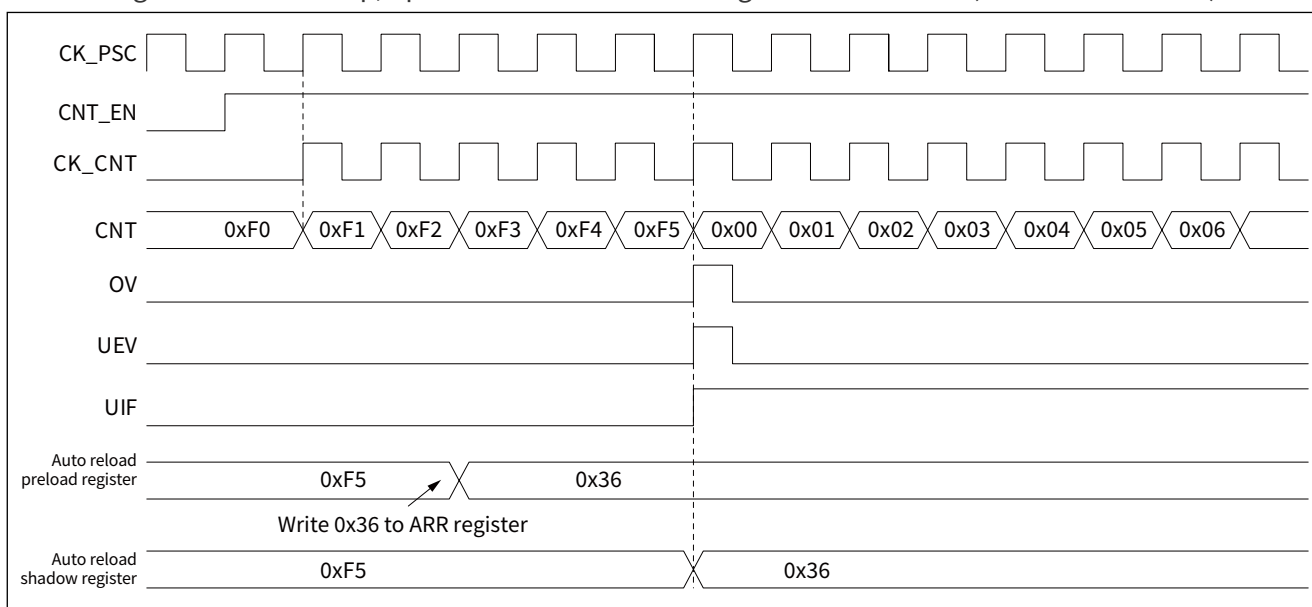


Figure 13-8 Count up, update event when reloading cache is enabled (GTIM_CR1.ARPE=1)



Count down mode

The counter operates in count down mode when the DIR bit of control register GTIM_CR1 is set to 1 in edge-aligned mode.

In count down mode, the hardware automatically loads the ARR value into the counter CNT, the counter starts count down to 0, and then reloads the ARR value to count down while generating a counter underflow event.

When the counter underflows, the underflow signal UND is generated, and the update event UEV is generated (the UND signal and the UEV signal are cleared automatically), and the counter update interrupt flag bit GTIM_ISR.UIF is set by the hardware, and if an interrupt is permitted (by setting GTIM_IER.UIE to 1), an interrupt request is generated, and setting GTIM_ICR.UIF to 0 clears this flag bit.

The following is an example of counter operation at different clock frequencies, where GTIM_ARR = 0x36:

Figure 13-9 Count down, internal clock frequency division factor is 1

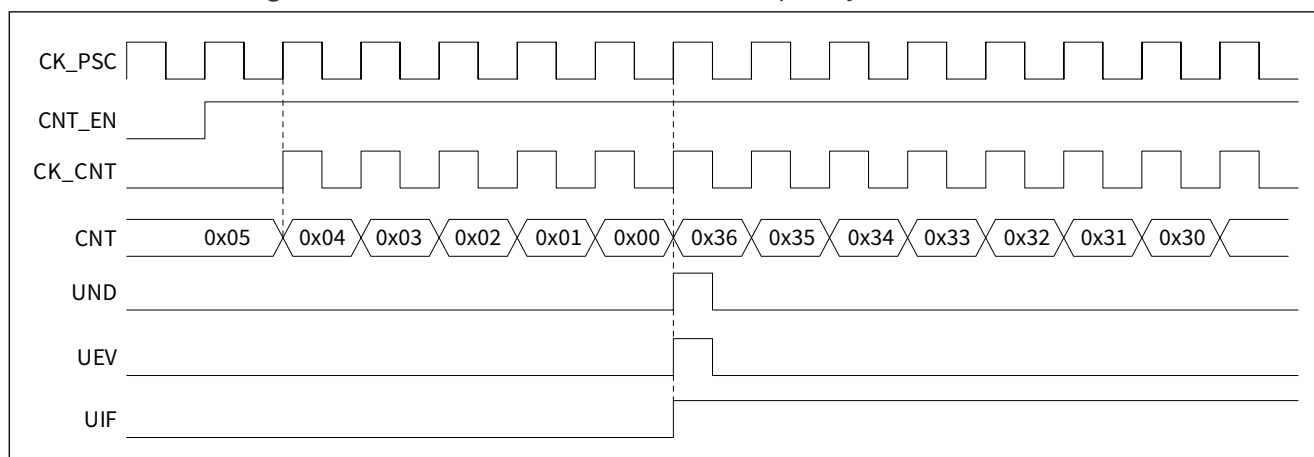


Figure 13-10 Count down, internal clock frequency division factor is 2

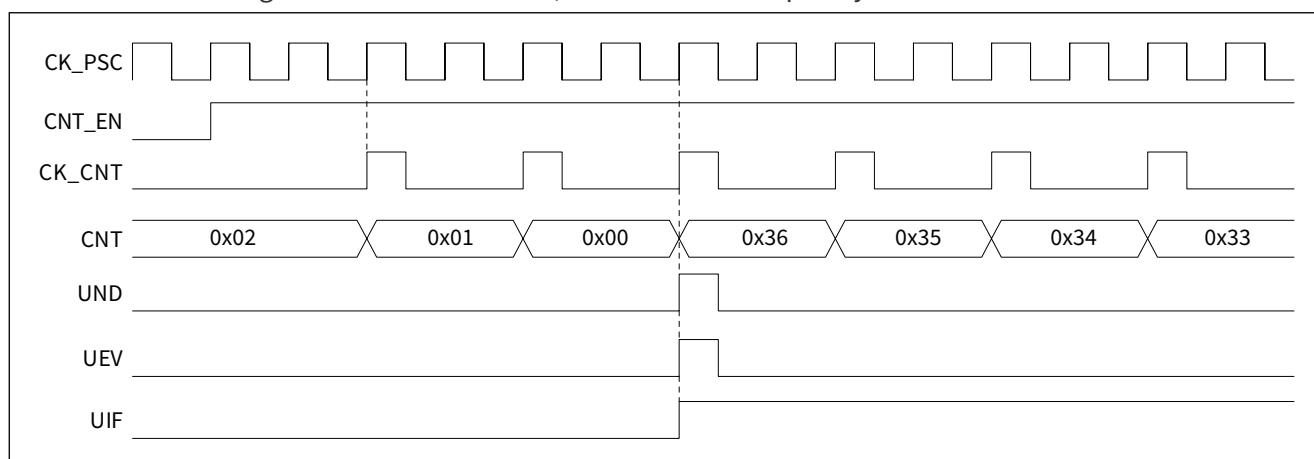


Figure 13-11 Count down, internal clock frequency division factor is 4

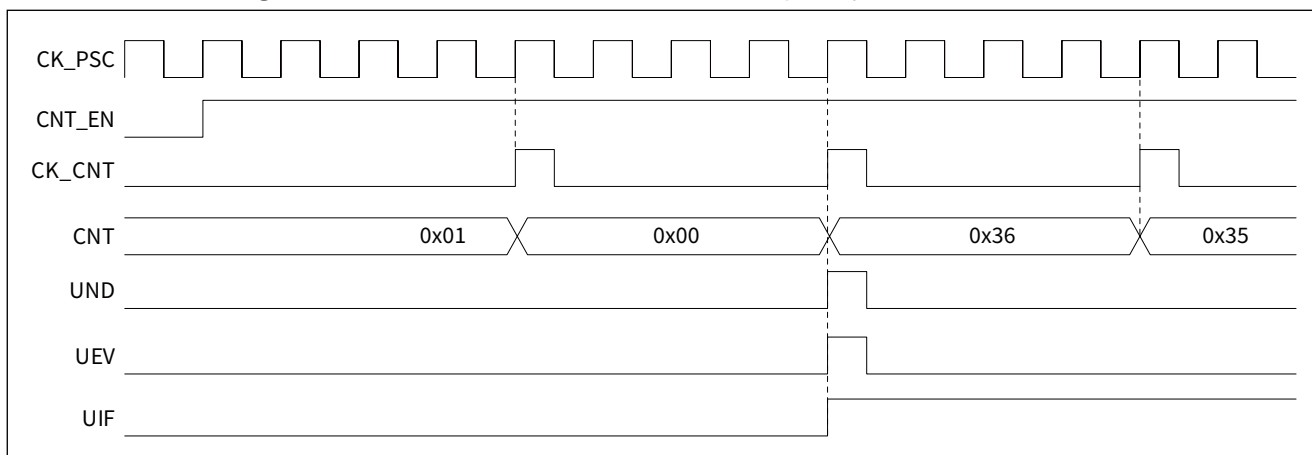
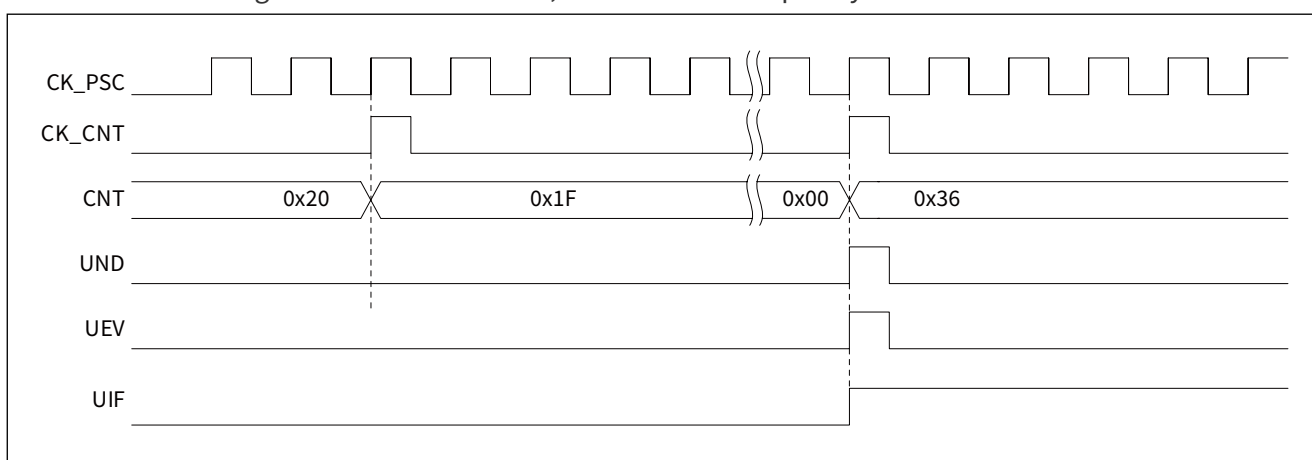


Figure 13-12 Count down, internal clock frequency division factor is N



Center-aligned mode (count up/down)

In center-aligned mode, the counter counting up from 0 to ARR-1, generates a counter overflow event, then starts counting down from the ARR value to 1 and generates a counter underflow event, after which the counting starts counting up again from 0.

The center-aligned modes include center-aligned mode 1, center-aligned mode 2, and center-aligned mode 3. The three modes are identical in counting, except for the timing of the output compare interrupt flag, for details, please refer to the description of the CMS bit field of [13.9.1 GTIM_CR1 Controls register 1](#).

In center-aligned mode, the DIR bit of control register GTIM_CR1 cannot be written by software but can be read, and DIR is updated by hardware and indicates the current count direction. When center-aligned mode is activated, the counter will count up or down based on the current DIR bit field value. It is not possible to modify the DIR and CMS bitfields at the same time through software.

It is not recommended to perform a write operation to the counter while running center-aligned mode, as unexpected results will occur. In particular:

- If a value greater than the auto-reload value ARR ($CNT > ARR$) is written to the counter CNT, the direction will not be updated. That is, if the counter was previously counting up, it continues to count up.
- If a value of 0 or ARR is written to the counter CNT, the count direction is updated, but no update event UEV is generated.

The most secure way to use the center-aligned mode is to generate the update event UEV by software before starting the counter (set GTIM_EGR.UG to 1) and do not perform a write operation to the counter while it is running.

The following are examples of counter operation at different clock frequencies:

Figure 13-13 Center-aligned mode, internal clock frequency division factor is 1, ARR=0x06

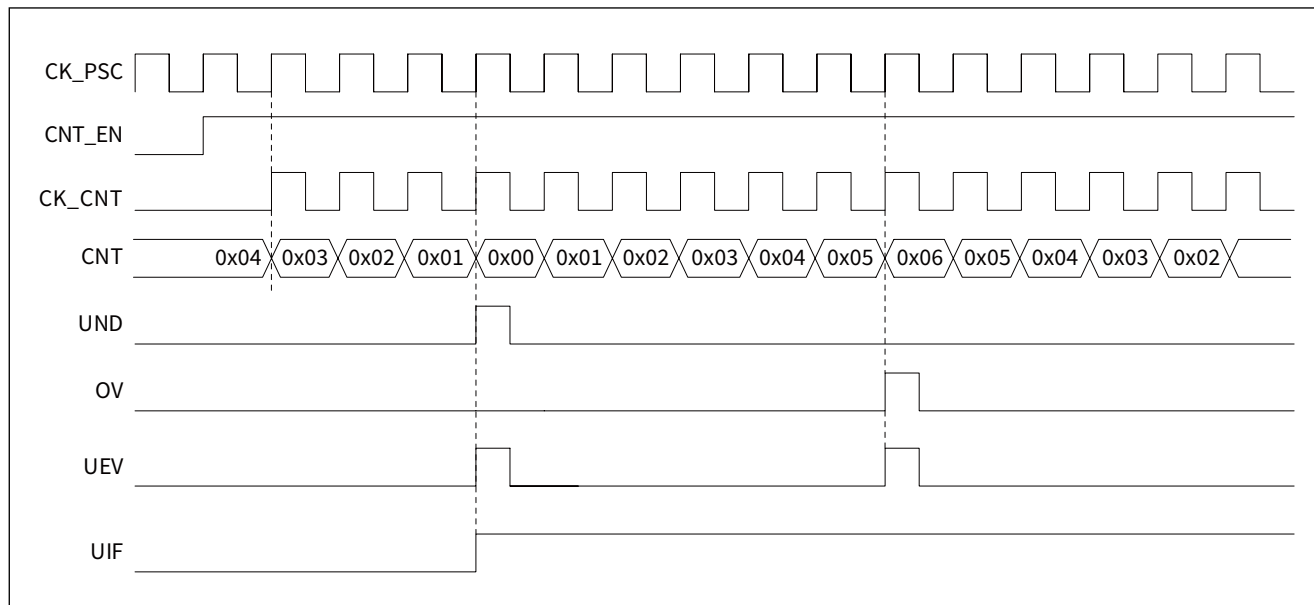


Figure 13-14 Center-aligned mode, internal clock frequency division factor is 2

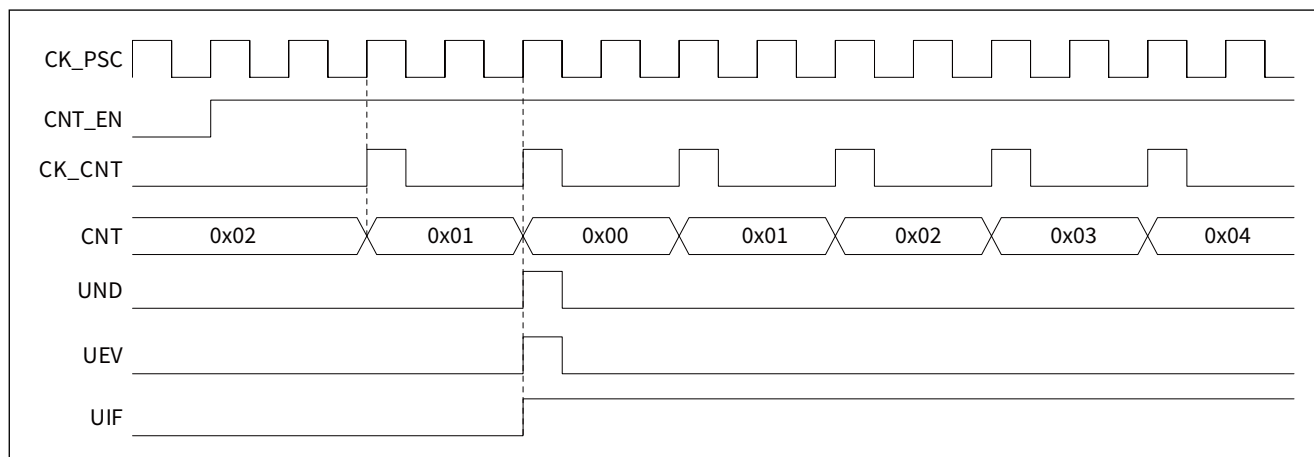


Figure 13-15 Center-aligned mode, internal clock frequency division factor is 4, ARR=0x36

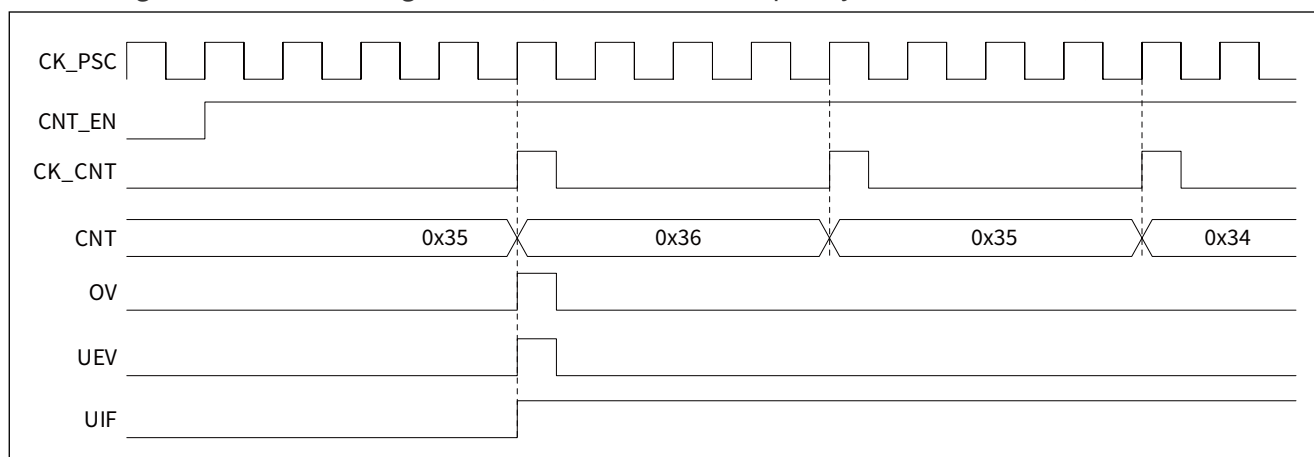


Figure 13-16 Center-aligned mode, internal clock frequency division factor is N

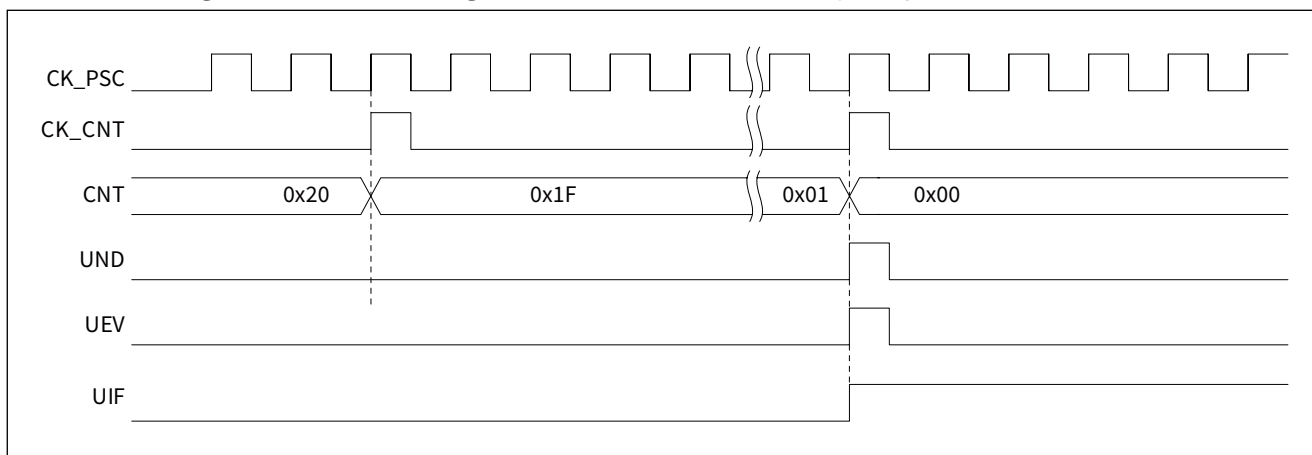


Figure 13-17 Center-aligned mode, update event when reloading cache is disabled (GTIM_CR1.ARPE=0)

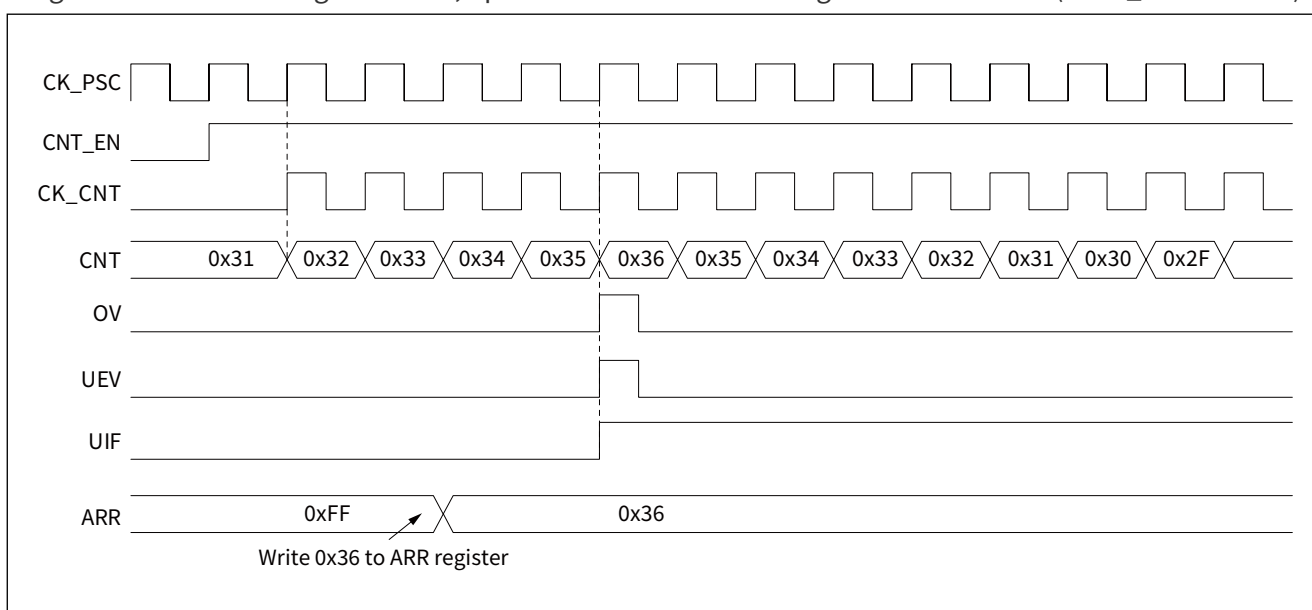
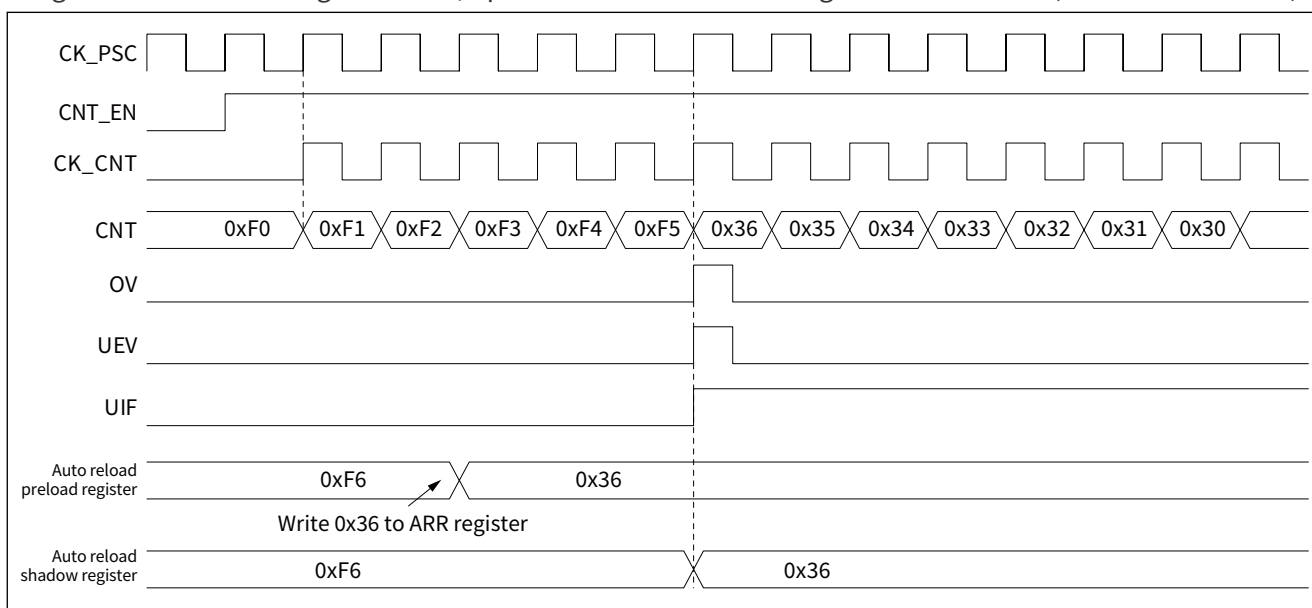


Figure 13-18 Center-aligned mode, update event when reloading cache is enabled (GTIM_CR1.ARPE=1)



13.3.1.4 Reload register

The auto reload register GTIM_ARR has a cache function that is enabled or disabled by controlling the ARPE bit field of register GTIM_CR1.

When the counter is stopped or the cache function is disabled, updating the reload register ARR will immediately update its shadow register. When the timer is running and the cache function is active, modifying the reload register ARR will not update the shadow register immediately, but will only update the value of the reload register ARR into the shadow register when an update event UEV is generated.

13.3.1.5 Update event UEV

Allows updates to be disabled or enabled by the UDIS bit field of the control register GTIM_CR1.

Enable UEV

Set the UDIS bit field to 0 to enable UEV. At this time, update request sources can be selected according to the URS bit field, as shown in the following table:

Table 13-2 Update source settings

GTIM_CR1.URS	Update source
0	The counter overflow, underflow; UG set; Update events generated from the slave mode controller.
1	The counter overflow, underflow.

When an update event occurs, the following actions occur:

1. Reinitialise the counter:
Count down (edge-aligned mode) : Reloads the auto-reload value GTIM_ARR.
Center-aligned mode or count up (edge-aligned mode): the counter is cleared.
2. If the cache register function is used, the corresponding preloaded register is updated to its shadow register.
3. The prescaler counter is cleared and the new prescaler value in GTIM_PSC takes effect.
4. The event update interrupt flag bit GTIM_ISR.UIF is set by hardware.

When an update event UEV occurs, the event update interrupt flag bit GTIM_ISR.UIF is set by hardware. If the interrupt is allowed (set GTIM_IER.UIE to 1), an update interrupt request is generated, and setting GTIM_ICR.UIF to 0 clears the flag bit.

Disable UEV

Set the UDIS bit field to 1 to disable UEV and no more update events are generated.

If the cache register function is used, the value of each corresponding shadow register remains unchanged. However, if the UG bit is set to 1 or a hardware reset is received from the slave mode controller, the counters of the counter and prescaler are reinitialised.



13.3.1.6 Single pulse mode

The counter can work in single count or continuous count mode and is selected by the OPM bit field of the control register GTIM_CR1.

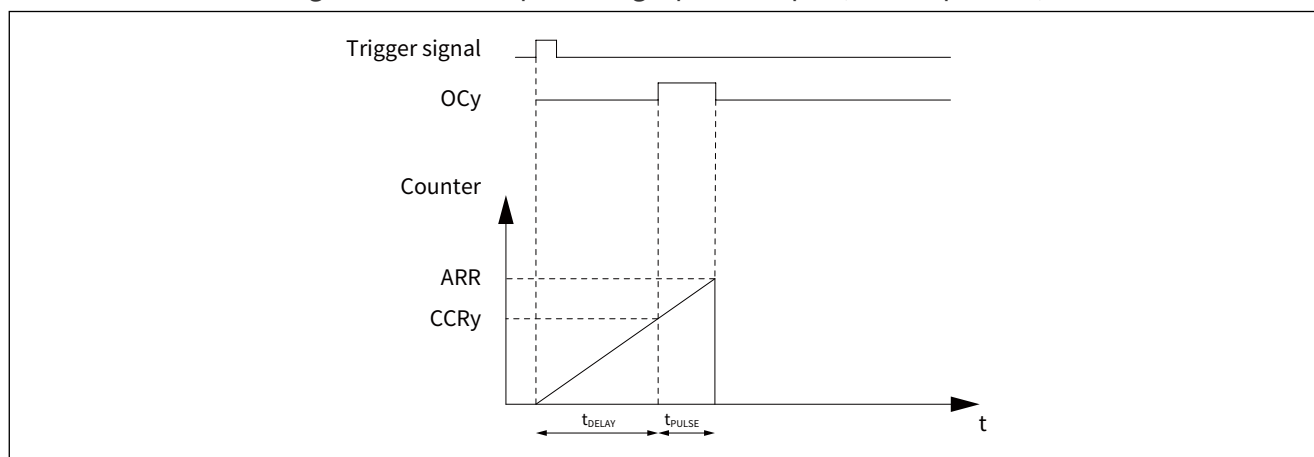
Single count mode

Set GTIM_CR1.OPM to 1 to enable the timer to work in single count mode.

After starting the counter, the counter CNT counts driven by the CK_CNT clock, and the counter stops counting when the next update event occurs while GTIM_CR1.CEN is automatically cleared by hardware.

The counter can be activated by the slave mode controller, together with the output comparison mode of the timer, so that the counter is activated by a trigger signal and generates a single pulse with a programmable pulse width after a programmable delay. As shown in the figure below, the timer starts counting when it detects a valid edge of the trigger signal, and after a delay of t_{DELAY} , it generates a positive pulse of width t_{PULSE} on the compare output port. Where t_{DELAY} is determined by the difference between the initial value of the count register GTIM_CNT and the capture/compare register GTIM_CCRy, and t_{PULSE} is determined by the difference between the auto-reload register GTIM_ARR and the capture/compare register GTIM_CCRy.

Figure 13-19 Example of single pulse output (count up mode)



Sampling the trigger input to activate the OC output requires multiple clock cycles, and the OCyFE bit field of the comparison mode registers (GTIM_CCMR1CMP and GTIM_CCMR2CMP) can be set to 1 to shorten the OC output delay time.

Continuous count mode

Set GTIM_CR1.OPM to 0 so that the timer works in continuous count mode and the counter does not stop counting when an update event occurs.

13.3.1.7 External trigger input channel

The external trigger input signal ETR can be used as a trigger input (TRGI) for the slave mode controller or as a count clock for the counter, see [13.3.2 Operating modes](#). The ETR signal can be input controlled, including polarity selection and edge detection, prescaling and filtering.

Polarity selection and edge detection

The ETP bit field of the slave mode control register GTIM_SMCR is used to select the trigger polarity of the ETR input signal. When ETP is set to 0, ETR is not inverted and is valid at high level or rising edge; when ETP is set to 1, ETR is inverted and is valid at low level or falling edge.

Prescaler

The frequency of external trigger signal ETRP shall not exceed 1/4 of PCLK frequency. when the frequency of ETRP is high, user should divide the external signal by appropriate ETPS prescaler setting to reduce the ETRP frequency, and the dividing coefficient can be set to 1, 2, 4, 8.

Filter

The filter uses digital filtering to sample the input signal at a certain frequency, the signal is valid when it is continuously sampled to N same levels, otherwise the signal is invalid, so as to filter out the high-frequency clutter signal.

The filter of the ETR input signal is configured by the ETF bit field of the slave mode control register GTIM_SMCR, which can set the sampling frequency and the number of sampling points. The sampling frequency f_{SAMPLING} is provided by the clock after f_{DTS} dividing. f_{DTS} is the frequency obtained after f_{PCLK} dividing. The dividing factor is configured by the CKD bit field of the GTIM_CR1 register, which can be set to 1, 2, or 4 divisions.

13.3.1.8 Input capture channel

GTIM supports 4 input capture channels TI1/2/3/4 which can be used as capture commands. Among them, TI1 and TI2 can also be used as trigger input (TRGI) and encoder interface input of slave mode controller, see [13.3.2 Operating modes](#).

Supports input control of Tly signals, including filtering, edge detection and prescaling.

Filter

In the filtering stage, the corresponding Tly input signal is sampled at a certain frequency, and the filtered signal TlyF is generated.

The filter for the Tly input signal is configured by the ICyF bit field of the capture mode registers GTIM_CCMR1CAP and GTIM_CCMR2CAP, which can be set for the sampling frequency and the number of sampling points. The sampling frequency f_{SAMPLING} is provided by the clock after f_{PCLK} or f_{DTS} dividing, f_{DTS} is the frequency obtained after f_{PCLK} dividing, and the dividing factor is configured by the CKD bit field of GTIM_CR1 register, and it can be set to 1, 2, or 4 divisions.

Edge detection

The edge detector with polarity selection generates a TlxFPx signal, for specific valid polarity please see the description of CCyNP and CCyP bit fields in [13.9.12 GTIM_CCER capture/compare enable register](#).

Prescaler

When the Tly signal is used for input capture, the capture channel signal ICy can be prescaled, which is controlled through the ICyPSC bit field of the capture mode registers GTIM_CCMR1CAP and GTIM_CCMR2CAP, and supports 1, 2, 4, and 8 frequency division.

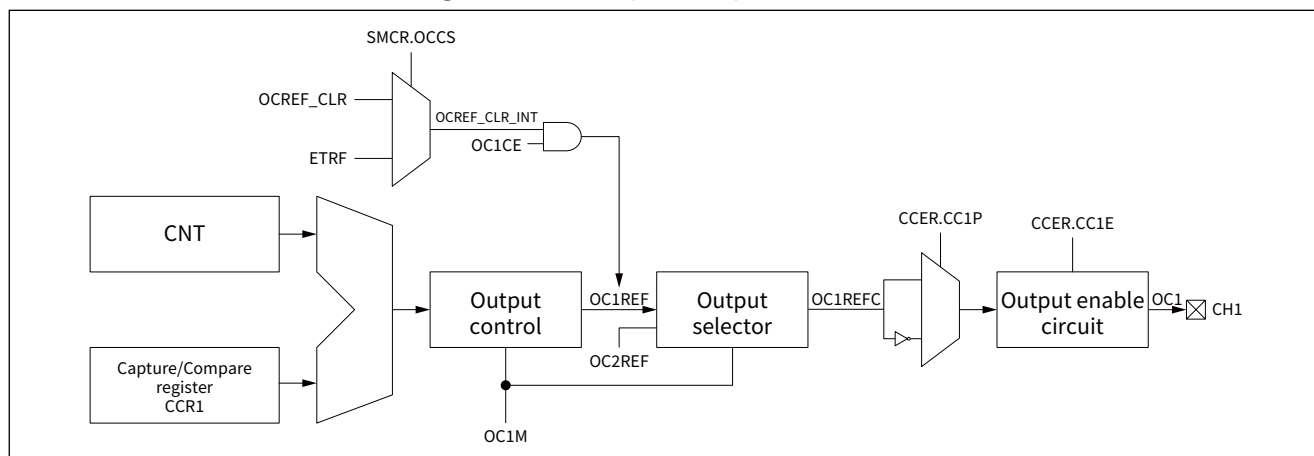


13.3.1.9 Output compare channel

GTIM supports 4 independent output compare channels CH1/2/3/4, each consisting of a capture/compare register (including a shadow register) and an output stage (comparator and output control).

The output control unit is used to control the waveform of the output port when comparing and matching, and can configure the output compare mode, output polarity selection and output enable. Taking compare channel 1 as an example, its block diagram is shown below:

Figure 13-20 Output compare channel 1



13.3.2 Operating modes

The GTIM supports multiple operating modes, which are configured by the SMS bit field of the slave mode control register GTIM_SMCR, as shown in the following table:

Table 13-3 GTIM operating modes

GTIM_SMCR.SMS value	Slave mode selection
0000	Disable slave mode, use internal clock
0111	External clock mode 1
0100	Reset mode
0101	Gated mode
0110	Trigger mode
1000	Combined reset + Trigger mode
1001	Combined gated + Reset mode
1110	Quadrature encoder mode, x1 mode
1111	Quadrature encoder mode, x1 mode
0001	Quadrature encoder mode, x2 mode
0010	Quadrature encoder mode, x2 mode
0011	Quadrature encoder mode, x4 mode
1010	Encoder mode (clock + direction), x2 mode
1011	Encoder mode (clock + direction), x1 mode
1100	Encoding mode (with direction clock), x2 mode
1101	Encoding mode (with direction clock), x1 mode

Caution:

1. The slave mode select bits include GTIM_SMCR[16] and GTIM_SMCR[2:0] to be configured in combination.
2. Slave mode select bits SMS[3:0] support the preload function, please refer to the description of SMSPE and SMSPS bit field in [13.9.3 GTIM_SMCR Slave mode control register](#).

13.3.2.1 Internal clock mode

When the SMS bit field of the slave mode control register GTIM_SMCR is 0x0, the timer slave mode is disabled and the prescaler clock is provided directly by the internal clock PCLK. Setting GTIM_CR1.CEN to 1 will enable the counter to start counting.



13.3.2.2 External clock mode

External clock mode 1

When the SMS bit field of the slave mode control register GTIM_SMCR is 0x7, the counter clock is supplied by the rising edge of the selected trigger signal (TRGI). There are several trigger options for the TRGI signal, which are selected through the TS bit field of the GTIM_SMCR register, as shown in the table below:

Table 13-4 TRGI signal source

GTIM_SMCR.TS bit field value	TRGI signal source
00111	External trigger input (ETRF)
000xx/01xxx/10000	Internal trigger (ITR0~12)
00100	TI1 edge detector (TI1F_ED)
00101	Filtered timer input 1 (TI1FP1)
00110	Filtered timer input 2 (TI2FP2)

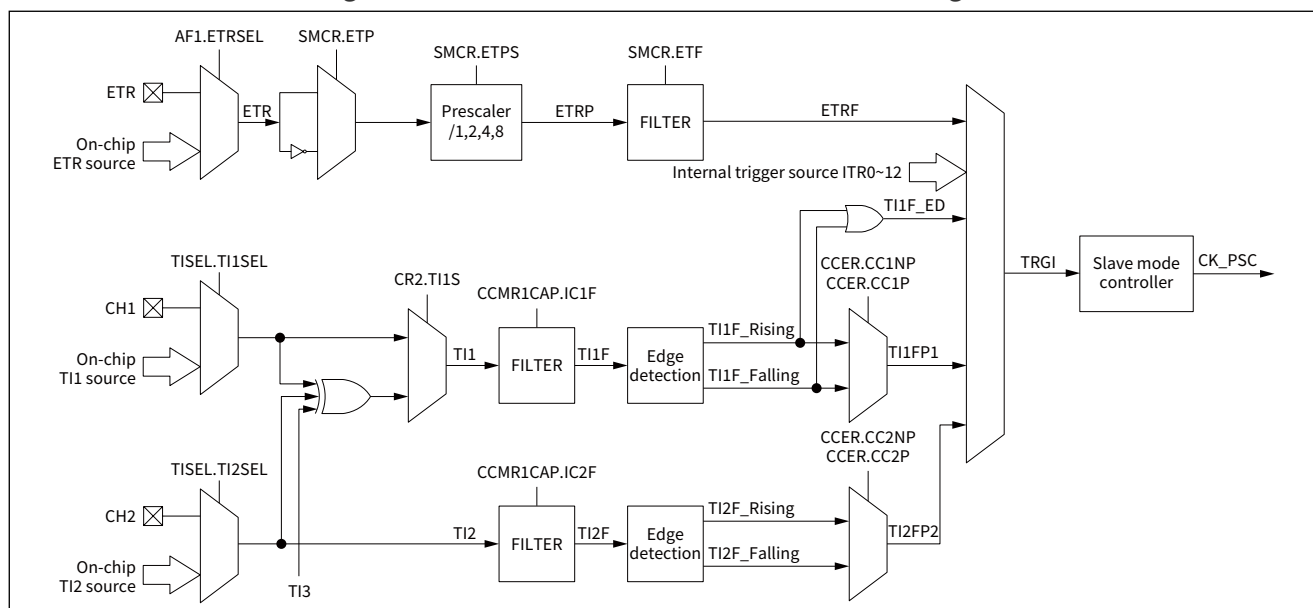
The ETR input signal source can be external GTIM_ETR pin or other peripherals on-chip, please refer to [13.3.7 On-chip peripheral interconnect ETR](#). When ETR is selected as the TRGI signal source, the external trigger polarity can be selected via GTIM_SMCR.ETP, the pre-scaler frequency can be set via GTIM_SMCR.ETPS, and the filtering control can be performed via GTIM_SMCR.ETF.

The internal trigger ITR source is the trigger output signal TRGO of BTIM and ATIM, and TXD/RXD signals of UART, please refer to [13.3.6 Timer cascade ITR](#).

Both TI1 and TI2 have filtering and edge detection functions. TI1 and TI2 are controlled by filtering through the IC1F and IC2F bit fields of the capture mode register GTIM_CCMR1CAP, respectively, and edge detection through the CC1P/CC1NP and CC2P/CC2NP bit fields of the capture/compare enable register GTIM_CCER, respectively.

The external clock mode 1 connection diagram is shown below:

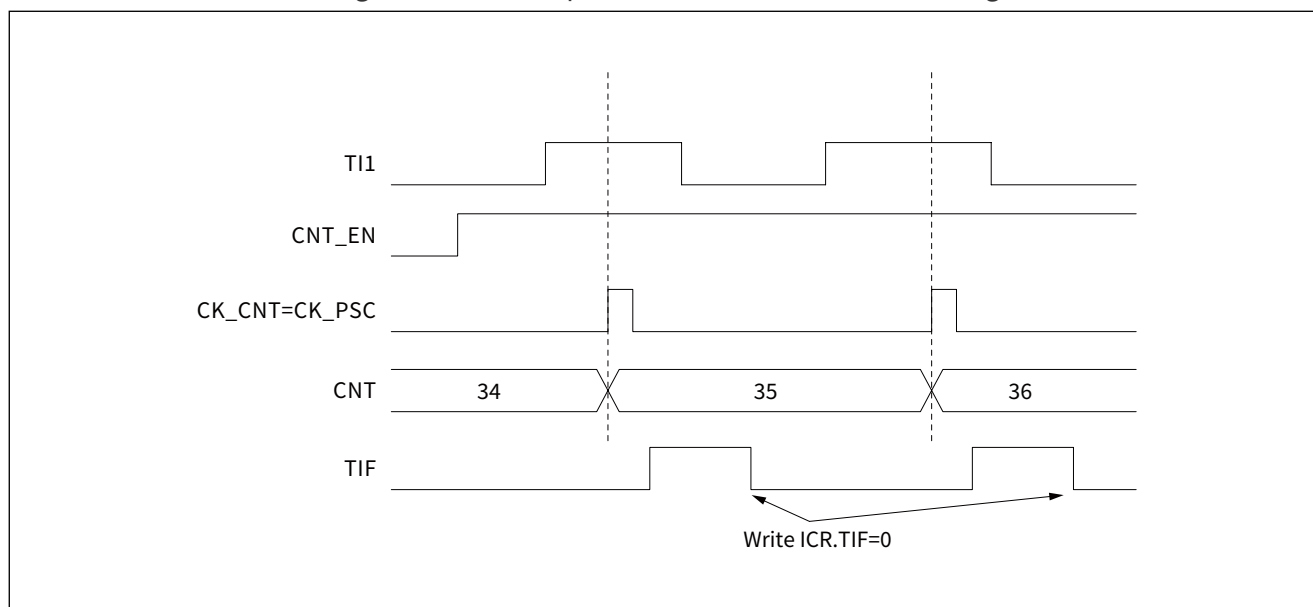
Figure 13-21 External clock mode 1 connection diagram



When a valid edge appears in TRGI, the GTIM_ISR.TIF flag will be set to 1, and the flag can be cleared by writing 0 to GTIM_ICR.TIF.

The following figure shows the external clock mode 1 timing diagram with TI1 rising edge configured. The delay between the rising edge of TI1 and the actual counter clock is due to the resynchronisation circuitry of the TI1 input.

Figure 13-22 Example of external clock mode 1 timing



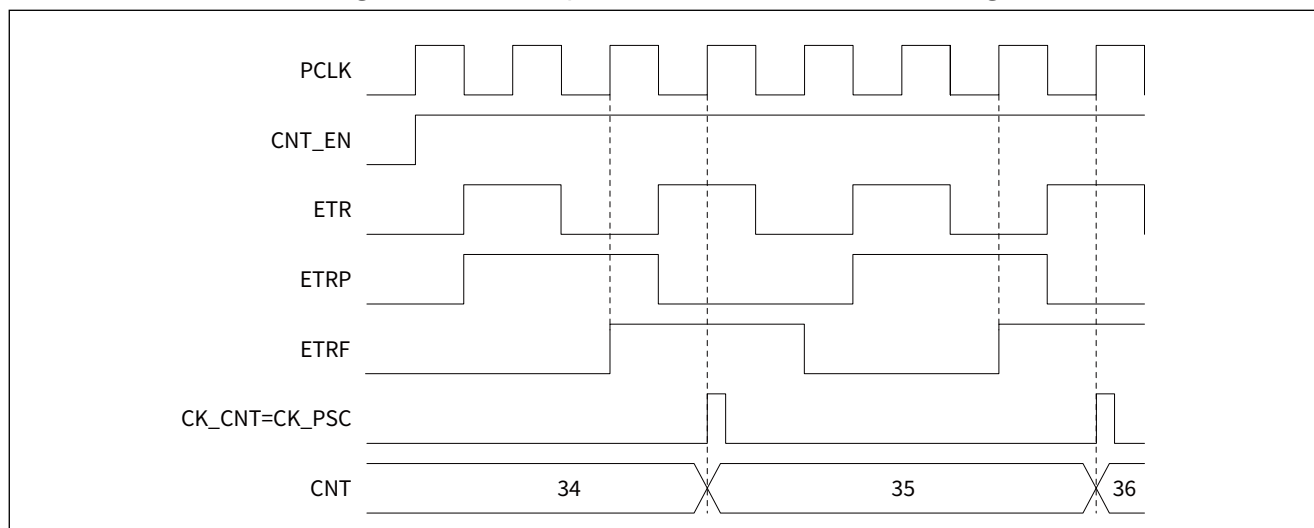
External clock mode 2

Set the ECE bit field of the slave mode control register GTIM_SMCR to 1 to enable external clock mode 2. In this mode, the counter clock is supplied by any valid edge of the ETRF signal, which has the same effect as selecting external clock mode 1 and connecting the TRGI to the ETRF (SMS=0111 and TS=00111), refer to [Figure 13-21 External clock mode 1 connection diagram](#).

External clock mode 2 can be used in conjunction with the following slave modes: reset mode, gated mode, and trigger mode, where the TRGI in slave mode must not be connected to the ETRF (i.e., the TS bit field cannot be set to 0x7). If both external clock mode 1 and external clock mode 2 are enabled, the external clock input is ETRF. External clock mode 2 cannot be used in conjunction with encoder mode.

The following figure shows a timing example of external clock mode 2 when ETR is set to 2 divisions, the delay between the rising edge of ETR and the actual counter clock is caused by the resynchronisation circuitry of the ETRP signal, so the maximum frequency that can be correctly captured by the counter is up to 1/4 of the internal clock PCLK frequency, and the user should divide the external signals appropriately when the ETRP signal becomes faster.

Figure 13-23 Example of external clock mode 2 timing



13.3.2.3 Reset mode

When the SMS bit field of the slave mode control register GTIM_SMCR is 0x4, the GTIM is configured in reset mode and the reset of the counter is controlled by the TRGI signal.

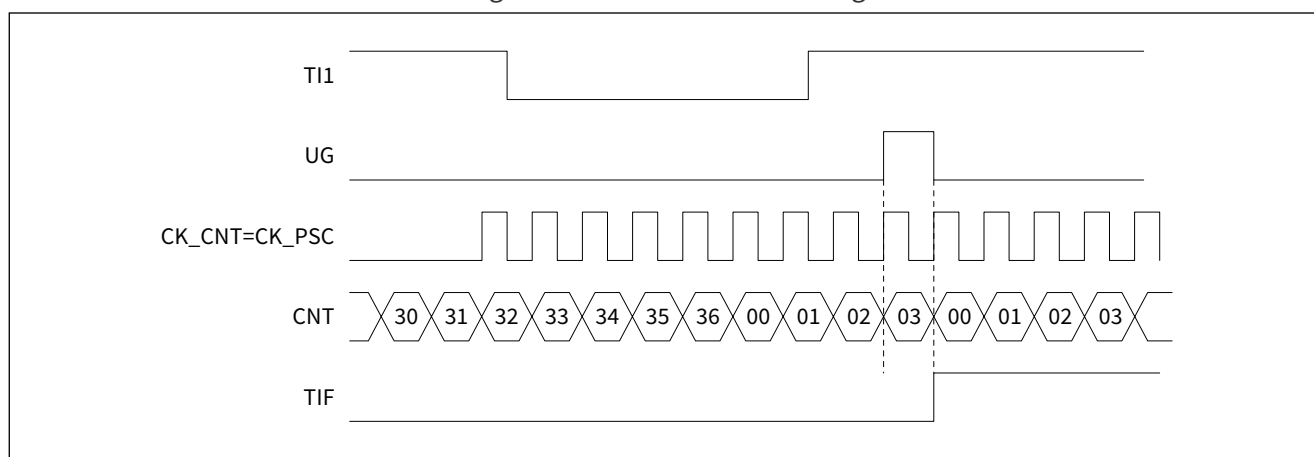
The source of the TRGI signal is controlled by the TS bit field of the slave mode control register GTIM_SMCR. See [Table 13-4 TRGI signal source](#).

When a valid trigger input (TRGI) is detected, the following effects occur:

- Reinitialise the counters and prescaler counters.
- If the URS bit field of control register GTIM_CR1 is 0, an update event UEV is generated, and the event update interrupt flag GTIM_ISR.UIF is set to 1. An interrupt request can be generated.
- The trigger interrupt flag GTIM_ISR.TIF is set to 1, which can generate an interrupt request.

The following figure shows an example of the reset mode timing diagram. Set GTIM_CR1.CEN to 1 to enable the counter, the counter counts normally according to the counting clock CK_CNT, and when a rising edge of TI1 occurs, the counter is cleared to 0 and starts counting from 0 again, while the GTIM_ISR.TIF flag is set to 1. The delay between the rising edge of TI1 and the actual counter reset is caused by the resynchronisation circuitry of the TI1 input.

Figure 13-24 Reset mode timing



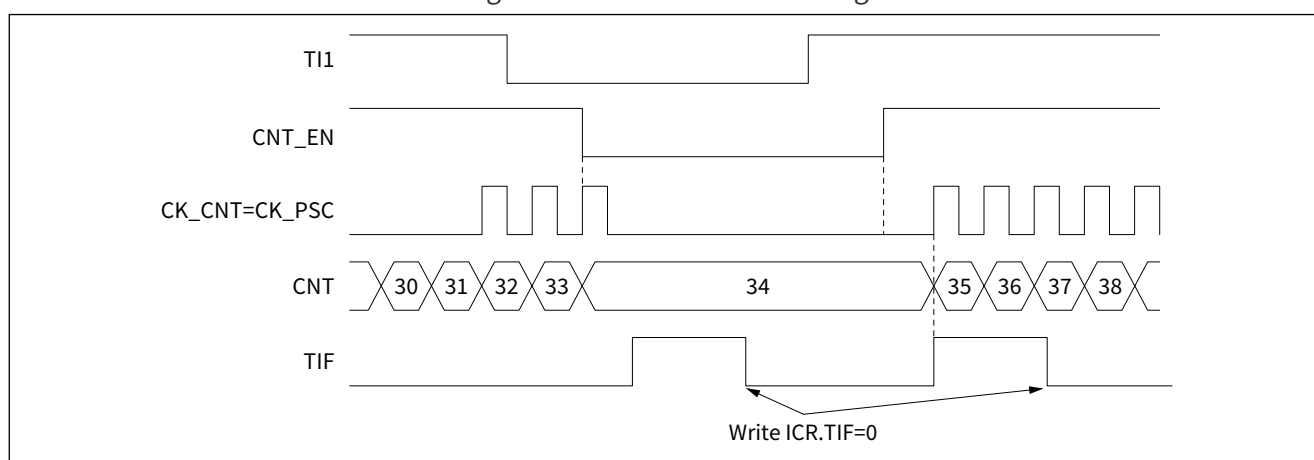
13.3.2.4 Gated mode

When the SMS bit field of the slave mode control register GTIM_SMCR is 0x5, the GTIM is configured for gated mode. In this mode, the counter is started counting when the trigger input (TRGI) is high and GTIM_CR1.CEN is 1. The counter stops counting immediately (but is not reset) when the trigger input (TRGI) is low or GTIM_CR1.CEN is 0. Both the start and stop of the counter are controlled.

The source of TRGI signals is controlled by the TS bit field of the slave mode control register GTIM_SMCR. For details, see [Table 13-4 TRGI signal source](#), however it should be noted that TI1F_ED cannot be selected as a trigger input in gated mode.

The following figure shows an example of gated mode timing diagram. When TI1 is high, the counter starts counting; when TI1 is low, the counter pauses counting. The GTIM_ISR.TIF flag bit is set to 1 when the counter starts and stops. The delay between the edge of TI1 and the actual counter enable signal CNT_EN is caused by the resynchronisation circuitry of the TI1 input.

Figure 13-25 Gated mode timing



13.3.2.5 Trigger mode

When the SMS bit field of the slave mode control register GTIM_SMCR is 0x6, the GTIM is configured for trigger mode. In this mode, when GTIM_CR1.CEN is set to 1 or when the rising edge of the trigger signal TRGI appears, the start counter is triggered (but not reset).

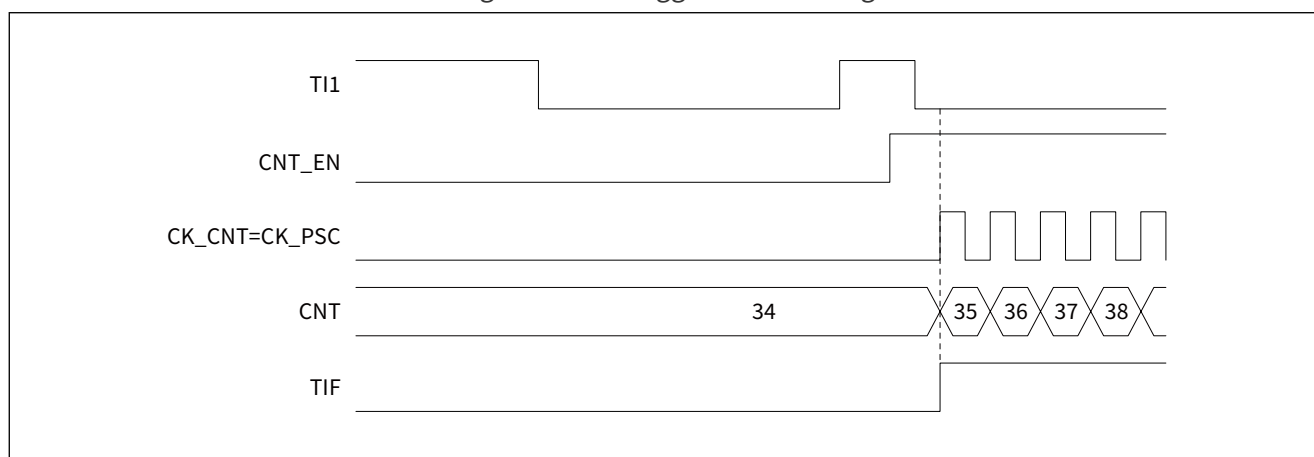
The source of the TRGI signal is controlled by the TS bit field of the slave mode control register GTIM_SMCR, see [Table 13-4 TRGI signal source](#).

When a valid trigger signal is detected, the following effects occur:

- GTIM_CR1.CEN is set by hardware.
- The trigger interrupt flag bit GTIM_ISR.TIF is set to 1, which can generate an interrupt request.
- The counter is activated and starts counting.

The following figure shows an example of a trigger mode timing diagram. When a rising edge of TI1 occurs, the counter starts counting while the GTIM_ISR.TIF flag bit is set to 1. The delay between the rising edge of TI1 and the actual counter startup is caused by the resynchronisation circuitry of the TI1 input.

Figure 13-26 Trigger mode timing



13.3.2.6 Combined reset + trigger mode

When the SMS bit field of the slave mode control register GTIM_SMCR is 0x8, the GTIM is configured for combined reset + trigger mode. In this mode, when a rising edge of the selected trigger input (TRGI) occurs, the counter will be reinitialised and the counter will be started, and the trigger interrupt flag GTIM_ISR.TIF will be set to 1. If the URS bit field of the control register GTIM_CR1 is 0, an update event UEV will be generated, and the event update interrupt flag GTIM_ISR.UIF will be set by hardware.

The source of the TRGI signal is controlled by the TS bit field of the slave mode control register GTIM_SMCR, see [Table 13-4 TRGI signal source](#).

13.3.2.7 Combined gated + reset mode

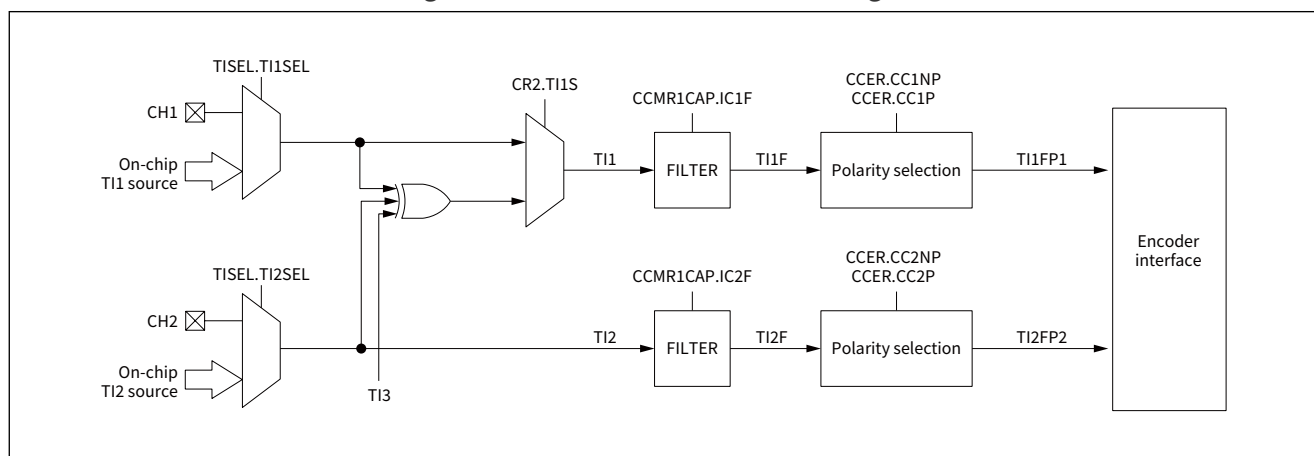
When the SMS bit field of the slave mode control register GTIM_SMCR is 0x9, the GTIM is configured for the combined gating + reset mode. In this mode, the start and stop of the counter are controlled. When the trigger input (TRGI) is high and GTIM_CR1.CEN is 1, the counter is started to count. When the trigger input (TRGI) is low, the counter immediately stops counting and is reset. When the counter is started and stopped, the GTIM_ISR.TIF flag bit is set to 1. When the counter is reset, GTIM_ISR.UIF flag bit is set to 1.

The source of the TRGI signal is controlled by the TS bit field of the slave mode control register GTIM_SMCR, see [Table 13-4 TRGI signal source](#).

13.3.2.8 Quadrature encoder mode

GTIM supports quadrature encoder mode for receiving and decoding quadrature encoder signals. In this mode, it allows a direct connection to an external quadrature encoder via the CH1 and CH2 pins, which enables the counter to count up or down automatically according to the hopping sequence of the input signals, and the counter value always indicates the position of the encoder. The functional block diagram is shown below:

Figure 13-27 Encoder mode block diagram



The CH1 and CH2 input signals have filtering and polarity selection functions, which are filtering controlled through the IC1F and IC2F bit fields of the GTIM_CCMR1CAP register, respectively, and the input polarity is selected through the CC1P and CC2P bit fields of the GTIM_CCER register, and the CC1NP and CC2NP bit fields must be kept clear.

Set GTIM_CR1.CEN to 1 to enable the counter, the counter will be driven by each valid hop of the signals TI1FP1 and TI2FP2 after the input signals of the channel CH1 and CH2 pins have been filtered and polarity selected, and according to the hopping sequence of the two input signals, the counting pulses and the direction signals are generated, refer to [Table 13-5 Relationship between counting direction and encoder signal](#). The encoder's current count direction flag GTIM_CR1.DIR is automatically set and cleared by hardware, and the DIR bit is counted whenever a signal transition occurs on any input (CH1 or CH2), regardless of whether the counter is counting only at the TI1FP1 or TI2FP2 edges, or at both TI1FP1 and TI2FP2.

GTIM supports several quadrature coded counting modes, which are set by the SMS bit field of the slave mode control register GTIM_SMCR. The relationship between the counting direction and the encoder signals in different modes is shown in the table below:

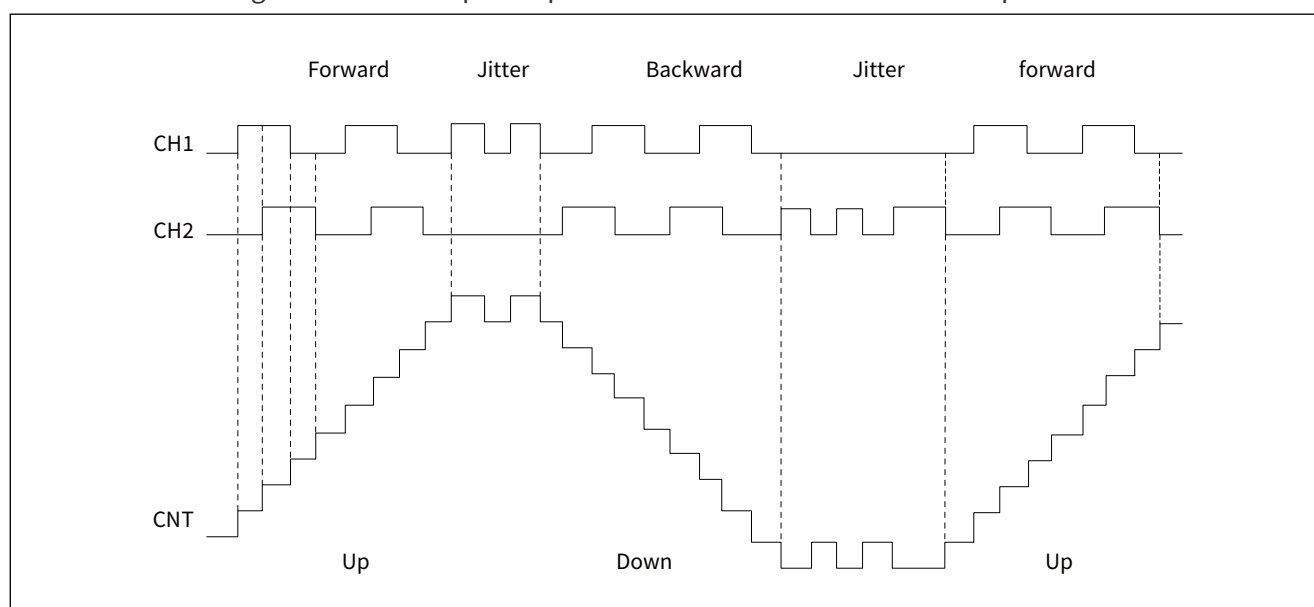
Table 13-5 Relationship between counting direction and encoder signal

Mode	SMS	Signal level		TI1FP1		TI2FP2	
		TI2FP2	TI1FP1	Rising	Falling	Rising	Falling
Counting on the TI1FP1 edge (x1 mode)	1110	High	-	Down count	Up count	No count	No count
		Low	-	No count	No count	No count	No count
Counting on the TI2FP2 edge (x1 mode)	1111	-	High	No count	No count	Up count	Down count
		-	Low	No count	No count	No count	No count
Counting on the TI1FP1 edge (x2 mode)	0001	High	-	Down count	Up count	No count	No count
		Low	-	Up count	Down count	No count	No count
Counting on the TI2FP2 edge (x2 mode)	0010	-	High	No count	No count	Up count	Down count
		-	Low	No count	No count	Down count	Up count
Counting on the TI1FP1 and TI2FP2 edges (x4 mode)	0011	High	High	Down count	Up count	Up count	Down count
		Low	Low	Up count	Down count	Down count	Up count

The third signal from the encoder output represents the mechanical zero point and can be connected to an external trigger input pin to trigger a counter reset.

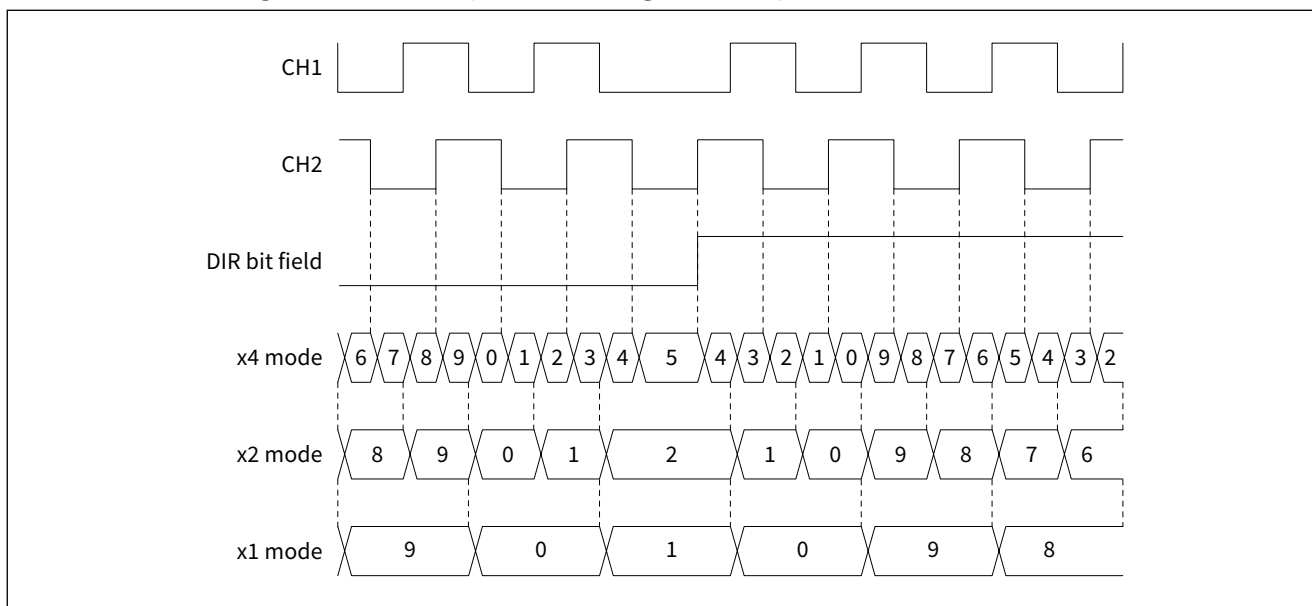
The following figure is an example of quadrature encoded counting mode-x4 mode operation, showing the generation of the counting signal and direction control. It also shows how input jitter is suppressed when a double edge is selected; jitter may be generated when the sensor is positioned close to a transition point.

Figure 13-28 Example of quadrature encoder mode-x4 mode operation



The following figure shows the timer count values during encoder inversion for various quadrature coded counting modes, with neither CH1 nor CH2 inputs inverted in the example.

Figure 13-29 Example of counting for each quadrature encoder mode



The encoder mode provides information about the current position of the sensor. The counter automatically counts up or down based on the pulses output from the encoder, and the current counter value indicates the current position of the encoder. Using another timer configured in capture mode, the rising or falling edge of the encoder signal can be captured and its timestamp recorded. By measuring the time interval between the two events, the corresponding dynamic information such as speed, acceleration and deceleration can be calculated; The encoder output indicating the mechanical zero position can be used for this purpose, and by measuring the time interval from the mechanical zero position to some other position, the change in position and velocity relative to the zero position can be calculated.

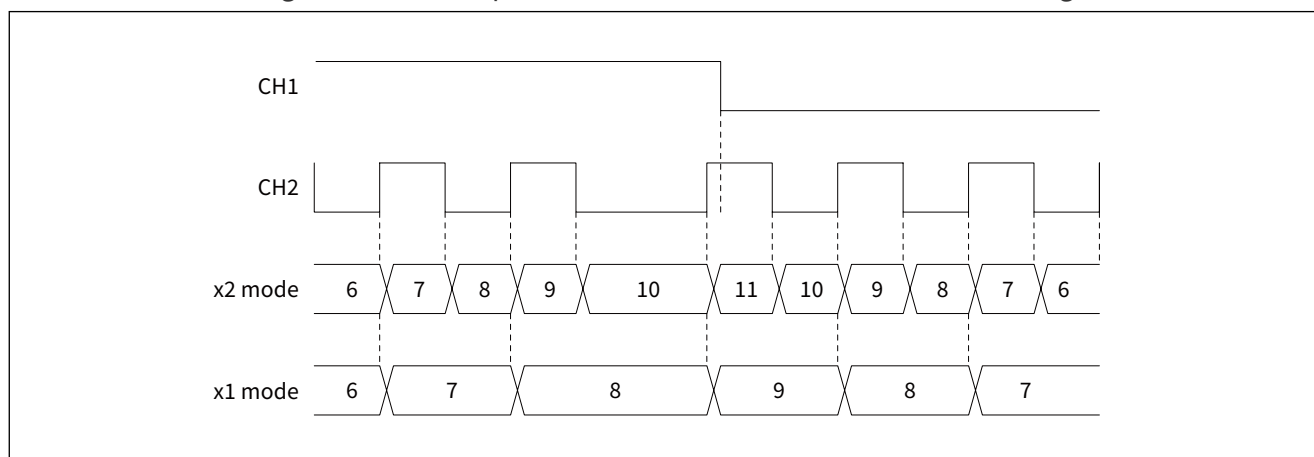
13.3.2.9 Clock + direction encoder mode

The timer also supports clock plus direction encoder mode, which requires configuring the SMS bit field of the slave mode control register GTIM_SMCR to be 0xA or 0xB, which correspond to x2 mode and x1 mode respectively. The functional block diagram can be referred to [Figure 13-27 Encoder mode block diagram](#), which allows filtering and polarity control of the input signal.

Encoders with direction usually have two output signals: a clock signal and a direction signal. Using the clock signal, the rotational speed and position change of the rotary encoder can be calculated, while the direction signal can be used to determine the direction of rotation. In x2 mode, the counter counts at the rising and falling edge of the clock signal; In x1 mode, the counter counts on a single clock edge based on the value of the CC2P bit field, where CC2P 0 corresponds to rising edge sensitivity and CC2P 1 corresponds to falling edge sensitivity. The polarity of the direction signal on the CH1 channel is set by the CC1P bit: CC1P is 0 corresponding to positive polarity (counts up when CH1 is high, counts down when CH1 is low) and CC1P is 1 corresponding to negative polarity (counts up when CH1 is low, counts down when CH1 is high).

The following figure shows an example of counter counting in clock plus direction encoder mode, with neither CH1 nor CH2 inputs inverted in the example:

Figure 13-30 Example of clock + direction encoder mode counting



13.3.2.10 Directional clock encoder mode

The timer also supports directional clock encoder mode, which requires configuring the SMS bit field of the slave mode control register GTIM_SMCR to be 0xC or 0xD, corresponding to x2 mode or x1 mode, respectively. The functional block diagram can be referred to [Figure 13-27 Encoder mode block diagram](#), which allows filtering and polarity control of the input signal.

The directional clock encoder provides an up count clock line and a down count clock line, respectively, depending on the direction of rotation. In x2 mode, the counter counts on the rising and falling edges of either of the two clock lines; the CC1P and CC2P bits encode the clock idle state, with CCyP being 0 corresponding to high level idle and CCyP being 1 corresponding to low level idle. In x1 mode, the counter counts on a single clock edge based on the value of the CC1P and CC2P bit fields, with CCyP being 0 corresponding to falling-edge sensitive and high-level idle, and CCyP being 1 corresponding to rising-edge sensitive and low-level idle.

The following table shows the relationship between the counting direction and the encoder signal in different modes:

Table 13-6 Relationship between counting direction and encoder signal

Mode	SMS	Signal level		TI1FP1		TI2FP2	
		TI2FP2	TI1FP1	Rising	Falling	Rising	Falling
x2 mode CCyP=0	1100	High	High	Down count	Down count	Up count	Up count
		Low	Low	No count	No count	No count	No count
x2 mode CCyP=1		High	High	No count	No count	No count	No count
		Low	Low	Down count	Down count	Up count	Up count
x1 mode CCyP=0	1101	High	High	No count	Down count	No count	Up count
		Low	Low	No count	No count	No count	No count
x1 mode CCyP=1		High	High	No count	No count	No count	No count
		Low	Low	Down count	No count	Up count	No count

The following figure shows an example of counting in directional clock encoder mode:

Figure 13-31 Example of directional clock encoder mode counting (CC1P = CC2P = 0)

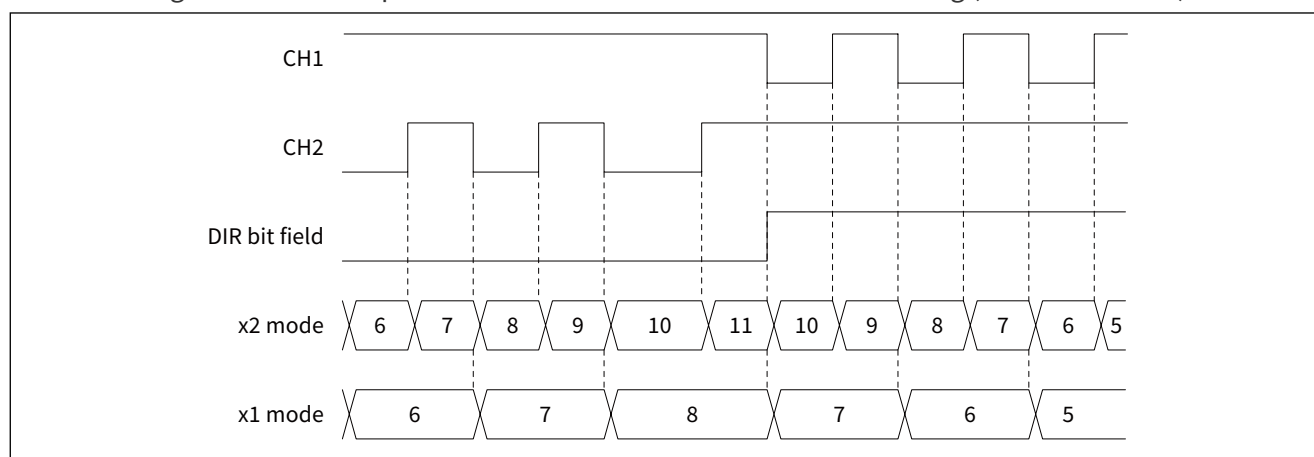
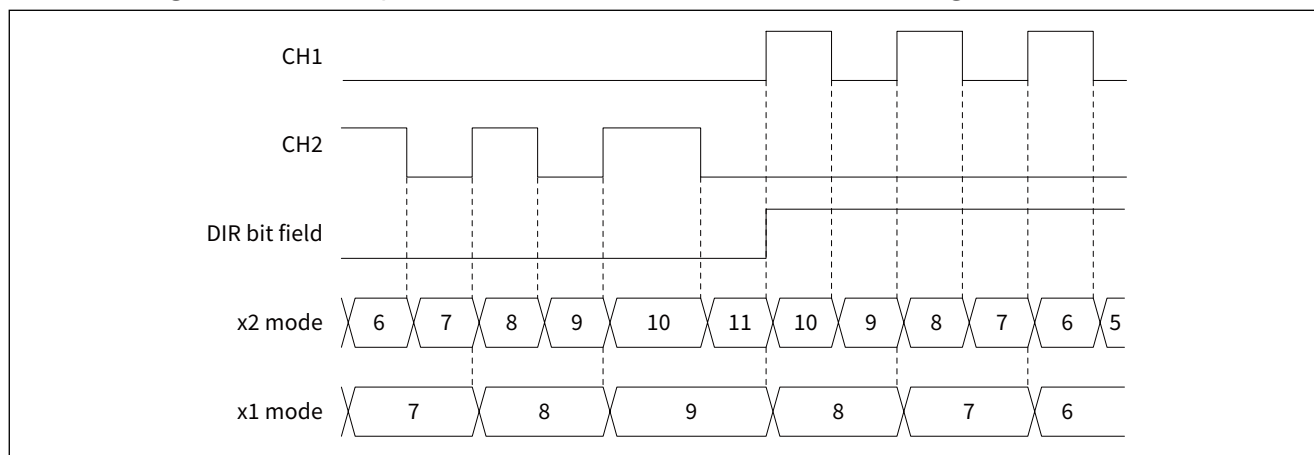


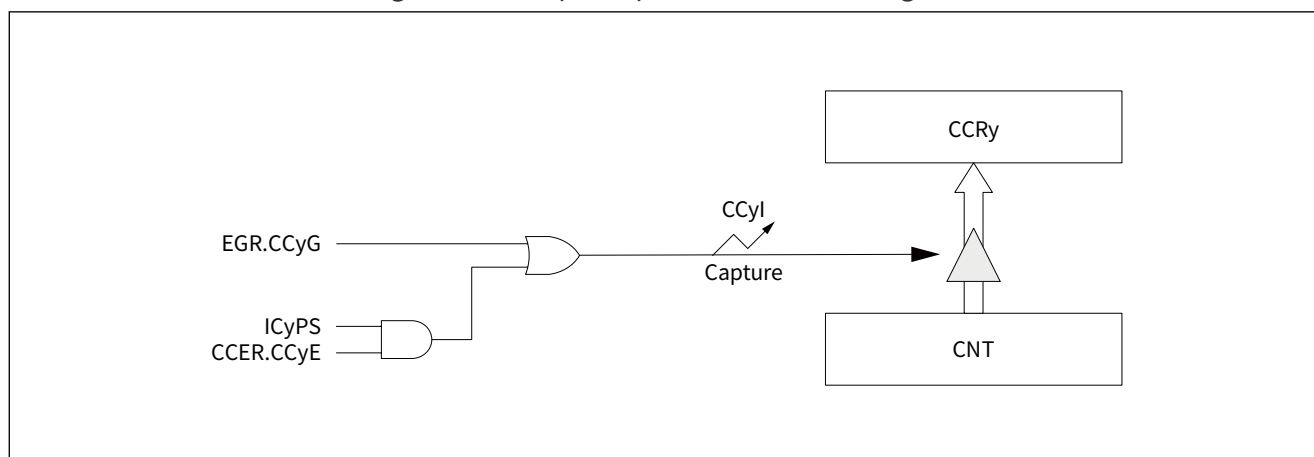
Figure 13-32 Example of directional clock encoder mode counting (CC1P = CC2P = 1)



13.3.3 Input capture function

GTIM supports the input capture function. When the CCyS bit field of the capture mode registers (GTIM_CCMR1CAP and GTIM_CCMR2CAP) is set to a non-zero value, the CCy channel is configured as an input, and at the same time, the input mapping of the corresponding capture channel ICy is specified. Input capture is supported to be triggered by software or hardware, and its function block diagram is shown below:

Figure 13-33 Input capture mode block diagram



Software trigger: When GTIM_EGR.CCyG is set to 1, a capture will be triggered by software immediately, and the value of the current counter CNT is latched into the capture/compare register CCRy of the corresponding channel to complete a capture, and the CCyG bit is cleared by the hardware automatically after the completion of the capture.

Hardware trigger: When a valid edge is detected on the input channel Tly, the value of the current counter CNT is latched into the capture/compare register CCRy of the corresponding channel to complete a capture. The valid edge that triggers the capture is configured through the CCyNP and CCyP bit fields of the capture/compare enable register GTIM_CCER as shown in the table below:

Table 13-7 Input capture mode configuration

GTIM_CCER register		Capture mode configuration
CCyNP bit field value	CCyP bit field value	
0	0	Rising edge capture
0	1	Falling edge capture
1	1	Both rising and falling edge capture

When a capture occurs, the capture/compare interrupt flag GTIM_ISR.CCyIF of the corresponding channel is set to 1 by hardware and:

- If interrupt is enabled (setting GTIM_IER.CCyIE to 1), an interrupt request is generated.
- If the GTIM_ISR.CCyIF flag bit is already high when the capture event occurs, the repeat capture flag bit GTIM_ISR.CCyOF will be set to 1 by hardware.
- Setting GTIM_ICR.CCyIF to 0 or reading capture register CCRy clears the GTIM_ISR.CCyIF flag bit, and setting GTIM_ICR.CCyOF to 0 clears the GTIM_ISR.CCyOF flag bit.

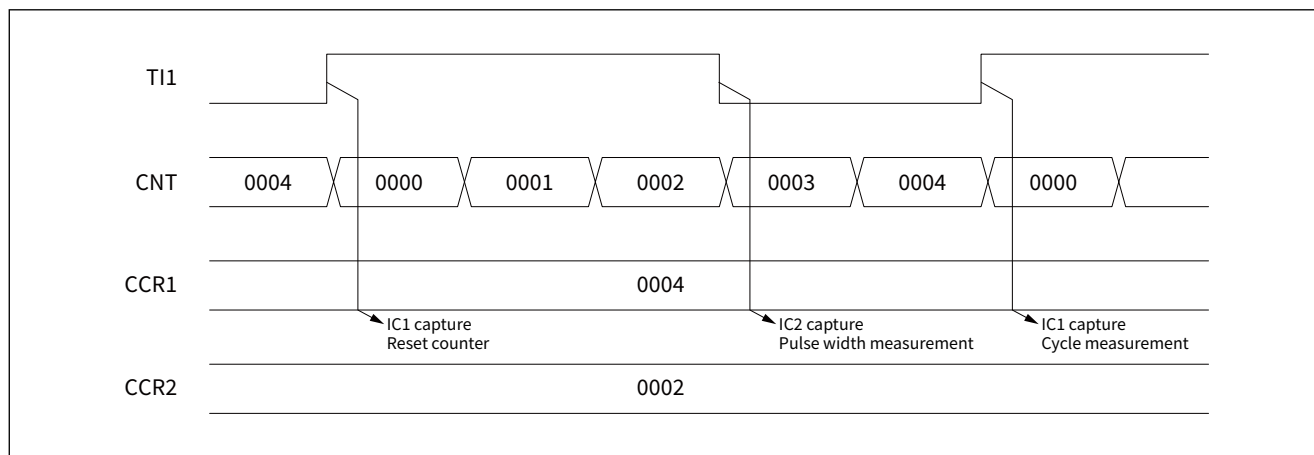
13.3.3.1 PWM input mode

PWM input mode is an application of input capture mode. The input capture function supports the same channel input signal to be captured by two capture registers. Combined with the slave mode reset function, the period and pulse width of the PWM input signal can be easily measured.

The PWM input mode can only be used with the TI1 and TI2 signals because only TI1FP1 and TI2FP2 are connected to the slave mode controller.

The following figure shows an example of the PWM signal timing when it is input from the TI1 channel:

Figure 13-34 PWM input mode timing



13.3.3.2 Input capture source

The GTIM input capture source can be the external GTIM_CHy pin or other peripherals on-chip, configured through the TlySEL bit field of the TI input select register GTIM_TISEL.

When GTIM_TISEL.TlySEL is 0x0, the source of the Tly channel input capture signal is the external GTIM_CHy pin, in which case the corresponding pin needs to be configured as a multiplexing function through the GPIO multiplexing function registers (GPIOx_AFRH and GPIOx_AFRL).

When GTIM_TISEL.TlySEL is 0x1~0xF, the Tly channel input capture signals come from other peripherals on-chip, as shown in the following table:

Table 13-8 General-purpose timer input capture source

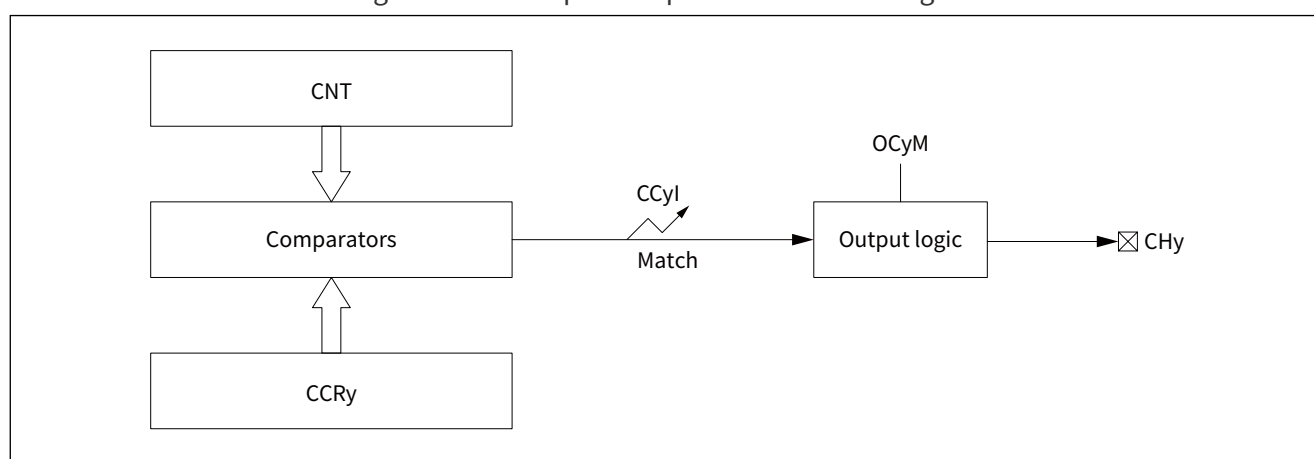
GTIM_TISEL.TlySEL bit field value	Input capture source for the Tly channel
0001	VC1_OUT
0010	VC2_OUT
0011	UART1_RXD
0100	UART2_RXD
0101	MCO_OUT
0110	HSE_FAULT
0111	LSE_FAULT
1000	RTC_OUT
1001	LSI_OUT
1010	BTIM1_Trgo
1011	BTIM2_Trgo
1100	BTIM3_Trgo
1111	ATIM_Trgo



13.3.4 Output compare function

The GTIM supports the output compare function. Setting the CCyS bit field of the compare mode registers (GTIM_CCMR1CMP and GTIM_CCMR2CMP) to 0 configures the CHy channel as an output. In the output compare mode, the value of the current counter CNT is compared with the value of the capture/compare register CCRy of the corresponding channel CHy. When the two match, the compare channel CHy outputs to a settable level state, and at the same time generates a compare interrupt. The following diagram shows the functional block diagram:

Figure 13-35 Output compare mode block diagram



The output action of the CHy pin is set by the OCyM bit field of the compare mode registers (GTIM_CCMR1CMP and GTIM_CCMR2CMP) as shown in the table below:

Table 13-9 Output compare mode configuration

OCyM bit field value	Compare mode configuration
0000	When comparing and matching, OCyREF keeps the original level
0001	When comparing and matching, OCyREF is set to 1
0010	When comparing and matching, OCyREF is set to 0
0011	When comparing and matching, OCyREF flips
0100	Force OCyREF to low level
0101	Force OCyREF to high level
0110	PWM mode 1
0111	PWM mode 2
1000	Retriggerable OPM mode 1
1001	Retriggerable OPM mode 2
1011	Count direction output
1100	Combined PWM mode 1
1101	Combined PWM mode 2
1110	Asymmetric PWM mode 1
1111	Asymmetric PWM mode 2

When a compare match occurs, the capture/compare interrupt flag GTIM_ISR.CCyIF of the corresponding channel is set to 1 by hardware, while:

- If interrupt is enabled (setting GTIM_IER.CCyIE to 1), an interrupt request will be generated.
- Setting GTIM_ICR.CCyIF to 0 clears the GTIM_ISR.CCyIF flag bit.

The capture/compare register GTIM_CCRy has a cache function and the use of the cache function is selected through the OCyPE bit field. When OCyPE is 0, the comparison cache function of channel CHy is disabled, and the GTIM_CCRy register can be updated by software at any time; the updated value takes effect immediately and affects the output waveform; When OCyPE is 1, the comparison cache function of channel CHy is enabled, and updating the GTIM_CCRy register will not take effect immediately, and the value of GTIM_CCRy register will be updated to the valid register only when the update event UEV occurs.



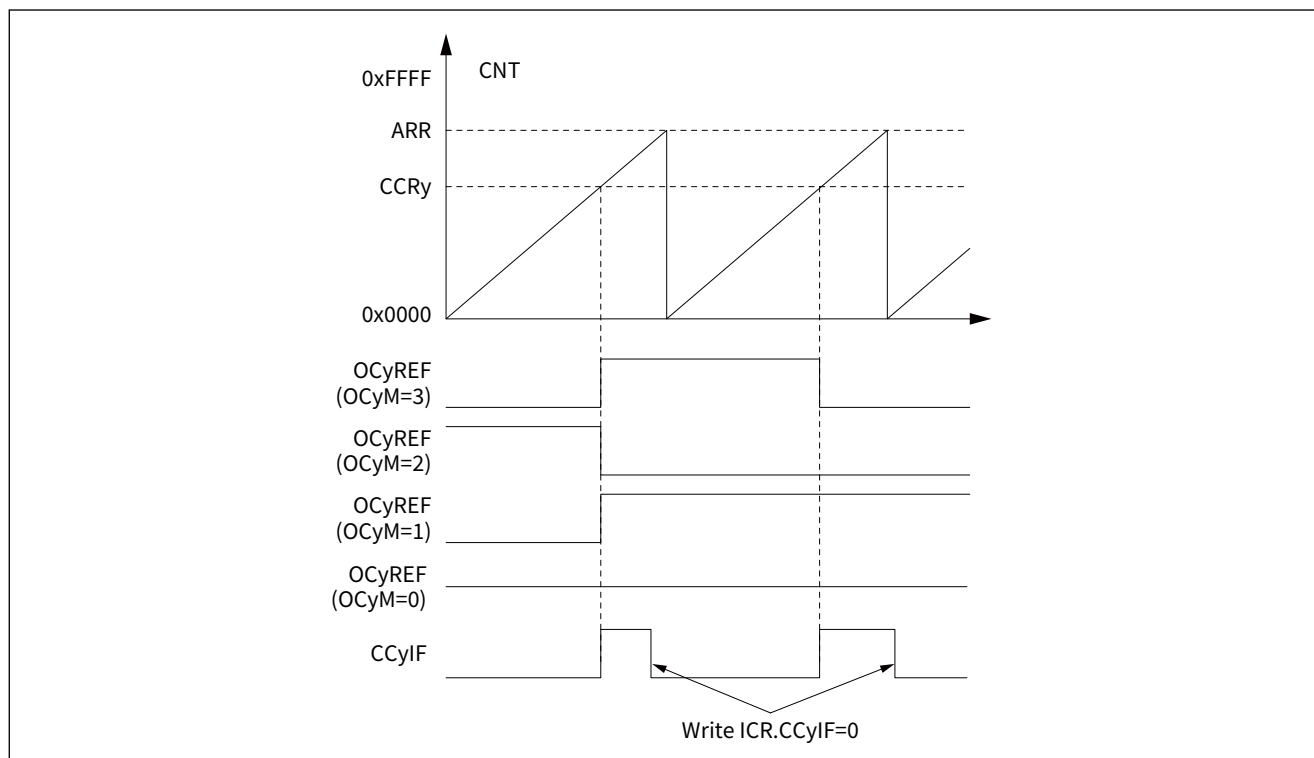
13.3.4.1 Match output

The match output mode is used to control the output waveform or to indicate that a certain time period has passed.

- Set the OCyM bit field to 0x0. When comparing and matching, the reference signal OCyREF keeps its level while the GTIM_ISR.CCyIF flag bit field is set to 1.
- Set the OCyM bit field to 0x1, when comparing and matching, the reference signal OCyREF is set to high level while the GTIM_ISR.CCyIF flag bit field is set to 1.
- Set the OCyM bit field to 0x2, when comparing and matching, the reference signal OCyREF is set to low level while the GTIM_ISR.CCyIF flag bit field is set to 1.
- Set the OCyM bit field to 0x3, when comparing and matching, the reference signal OCyREF is flipped while the GTIM_ISR.CCyIF flag bit field is set to 1.

The following figure shows an example of the timing for each compare-match output mode, with the example being the edge-aligned incremental mode:

Figure 13-36 Compare match output mode



13.3.4.2 Forced output

In forced output mode, the output comparison signal can be forced to a high or low state directly by software without relying on the comparison result of the capture/compare register GTIM_CCRy and the count register GTIM_CNT.

Set the OCyM bit field of the compare mode registers (GTIM_CCMR1CMP and GTIM_CCMR2CMP) to 0x4 to force the reference signal OCyREF to a low level; set the OCyM bit field to 0x5 to force the reference signal OCyREF to a high level.

In forced output mode, the comparison between the GTIM_CCRy register and counter GTIM_CNT is still in progress, and the corresponding flags are set to 1. The corresponding interrupt requests are also generated.



13.3.4.3 PWM output

Pulse width modulation (PWM) mode can generate a signal with a frequency determined by the reload register GTIM_ARR and a duty cycle determined by the capture/comparison register GTIM_CCRy.

Writing 0x6 (PWM mode 1) or 0x7 (PWM mode 2) to the OCyM bit field of the compare mode registers (GTIM_CCMR1CMP and GTIM_CCMR2CMP) can independently set each CHy output channel to generate a PWM all the way through the output states as shown in the table below:

Table 13-10 PWM mode 1/2 output state

Operating mode	Count direction	PWM mode 1	PWM mode 2
Edge-aligned mode Center-aligned mode	Up count	CNT < CCRy, OCyREF is high	CNT < CCRy, OCyREF is low
	Down count	CNT > CCRy, OCyREF is low	CNT > CCRy, OCyREF is high

The following are examples of waveforms for each PWM mode in different count modes, where GTIM_ARR is 0x08:

Figure 13-37 PWM mode 1, edge-aligned mode, count up

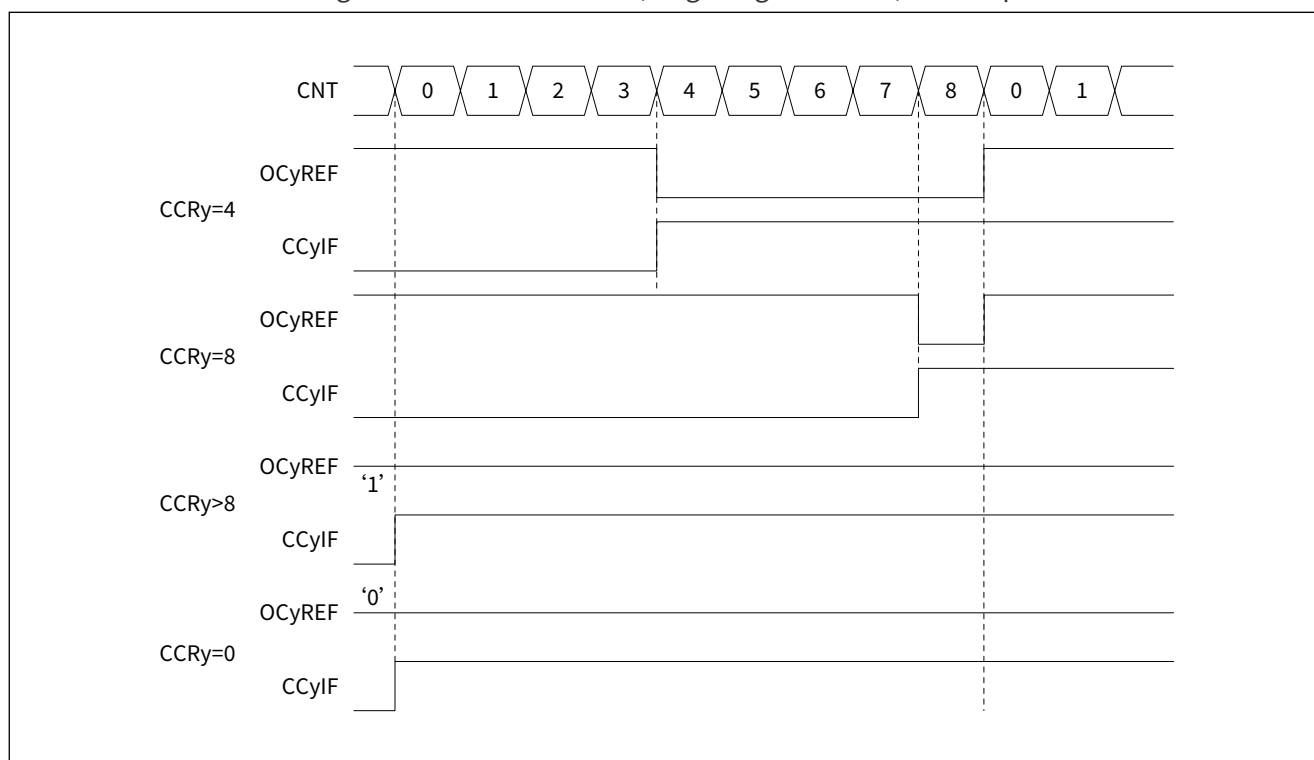


Figure 13-38 PWM mode 1, edge-aligned mode, count down

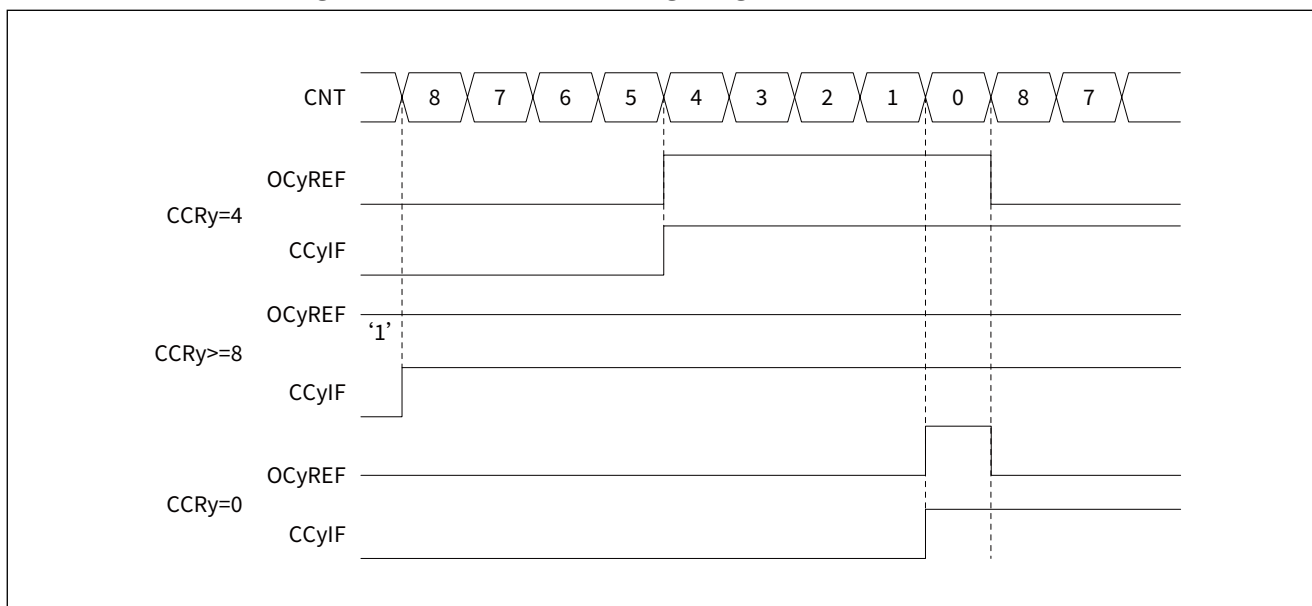
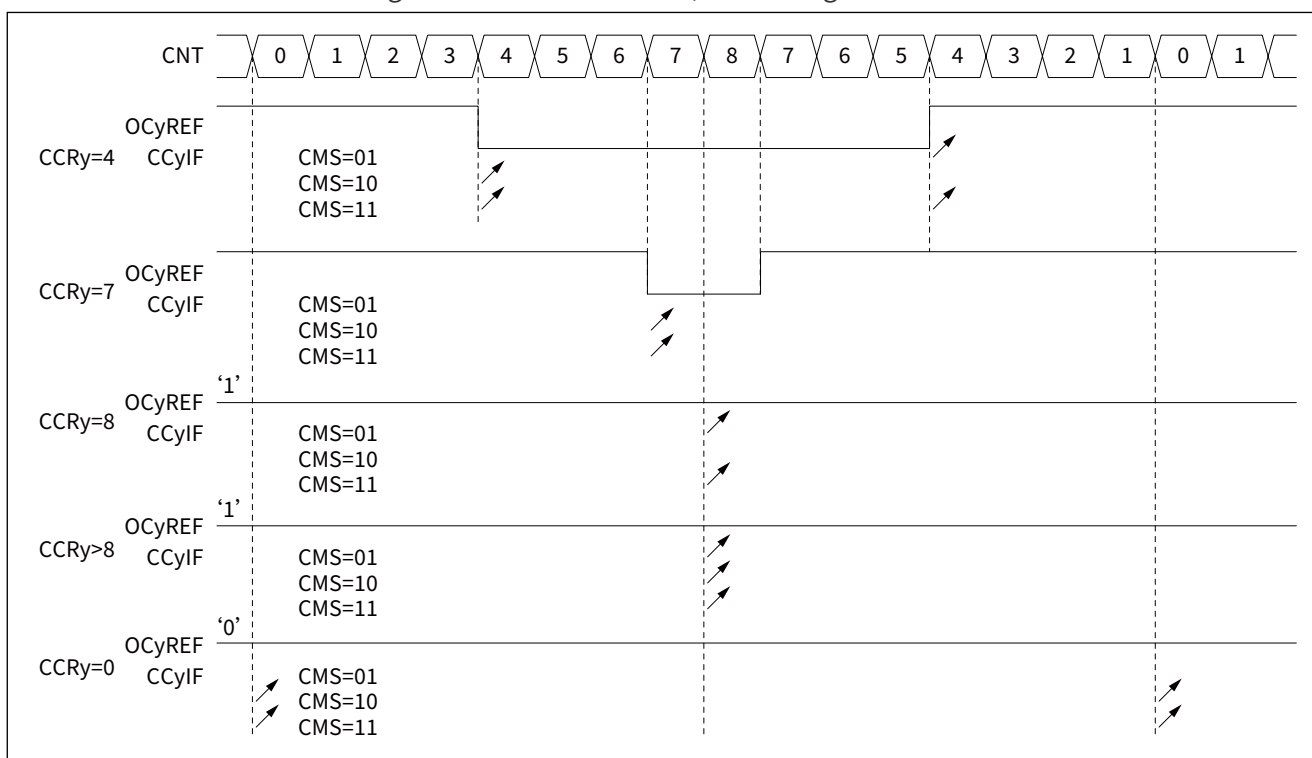


Figure 13-39 PWM mode 1, center-aligned mode



13.3.4.4 Asymmetric PWM output

The asymmetric PWM output is relative to the symmetric PWM output when counting in center-aligned mode. In the asymmetric PWM output mode, two PWM signals with programmable phase shifts are allowed to be generated in centre-aligned mode, with the frequency determined by the reload register GTIM_ARR, and the duty cycle and phase shifts determined by the two capture/comparison registers GTIM_CCRy, where one of the CCRy registers controls the PWM during counting up, and the other CCRy register controls the PWM during counting down, the details are shown below:

- OC1REFC is controlled by CCR1 (count up) and CCR2 (count down).
- OC2REFC is controlled by CCR2 (count up) and CCR1 (count down).
- OC3REFC is controlled by CCR3 (count up) and CCR4 (count down).
- OC4REFC is controlled by CCR4 (count up) and CCR3 (count down).

Writing 0xE (asymmetrical PWM mode 1) or 0xF (asymmetrical PWM mode 2) to the OCyM bit field of the compare mode registers (GTIM_CCMR1CMP and GTIM_CCMR2CMP) can independently set each CHy output channel to generate asymmetrical PWM.

When the channel is set to be used as an asymmetric PWM output channel, the other auxiliary channel can still be used. For example, when channel CH1 is configured to generate an OC1REFC signal in asymmetric PWM mode 1, channel CH2 can still be configured to output an OC2REF signal in PWM mode 2 or to output an OC2REFC signal in asymmetric PWM mode 2.

The following figure shows an example of a waveform for asymmetric PWM mode 1/2, where ARR=8, CCR1=0, CCR2=8, CCR3=3, and CCR4=5:

Figure 13-40 Example of asymmetric PWM mode 1

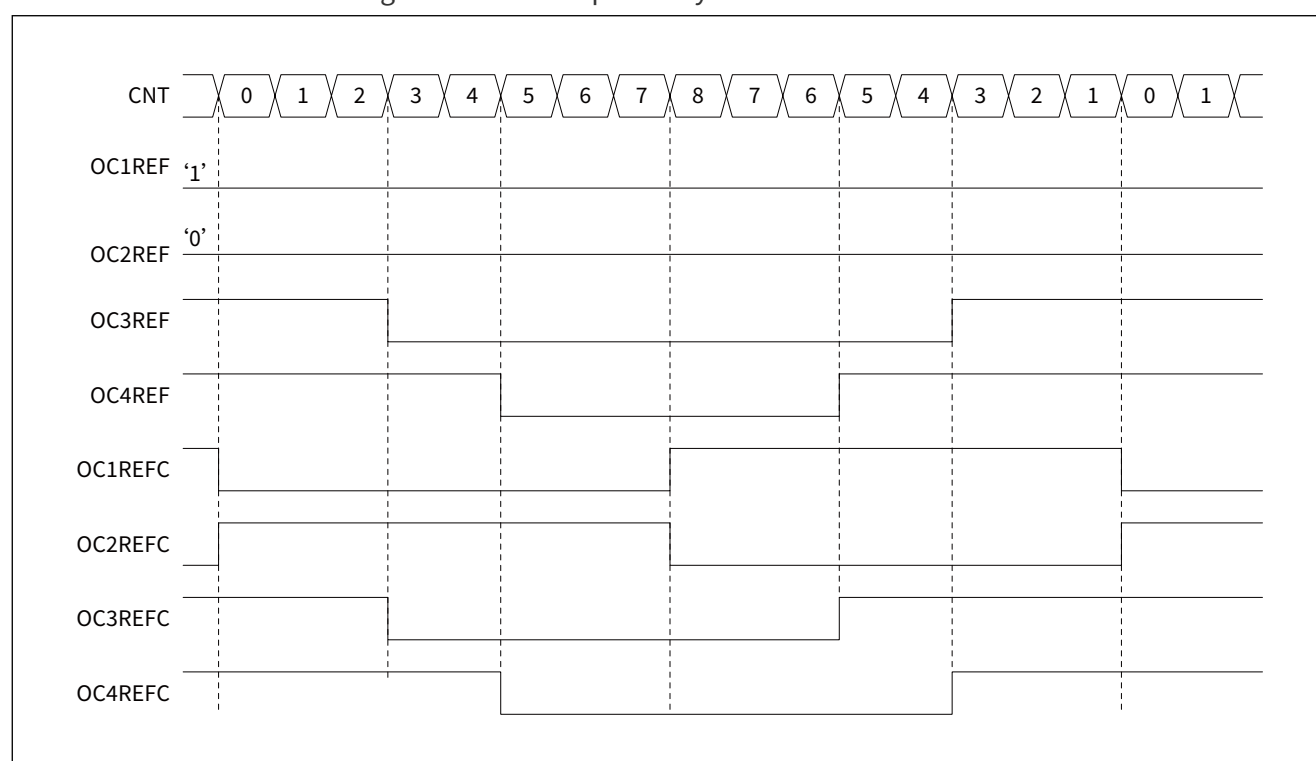
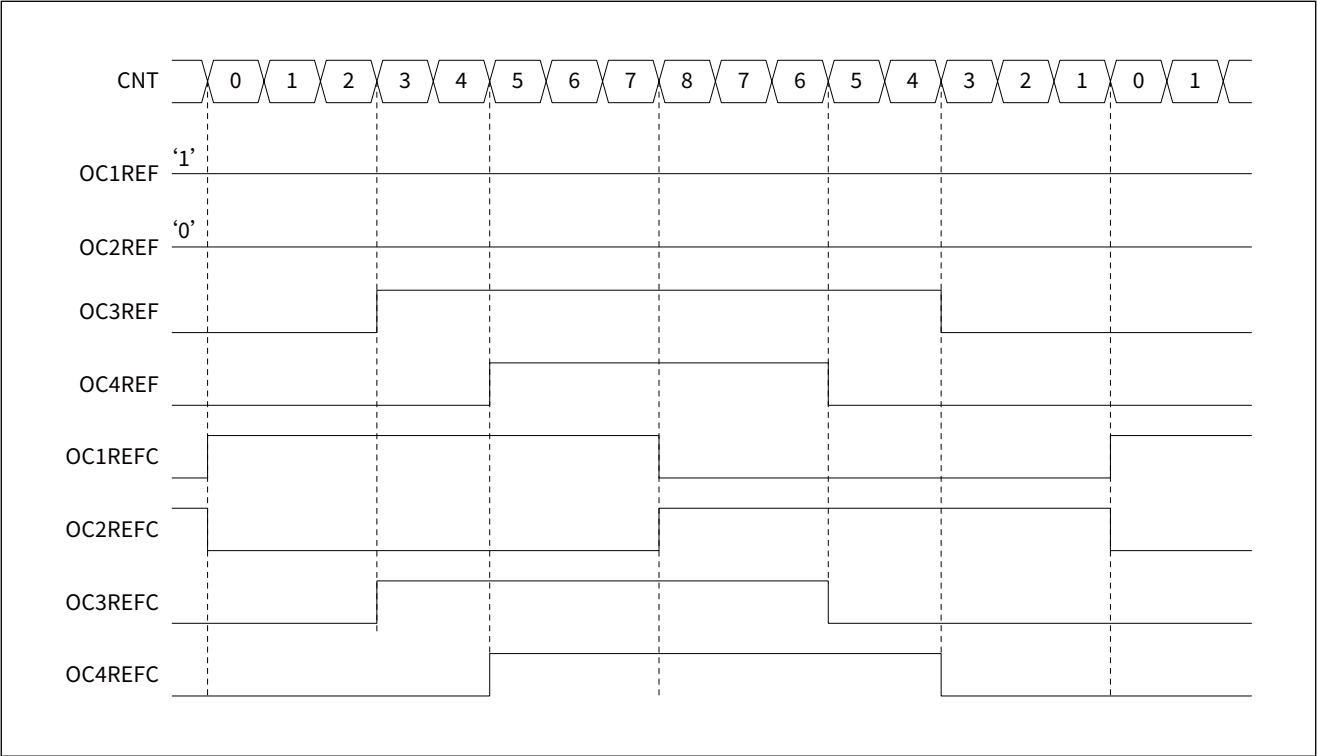


Figure 13-41 Example of asymmetric PWM mode 2



13.3.4.5 Combined PWM output

In combined PWM output mode, two PWM pulses with programmable delay and phase shift are allowed to be generated in either edge-aligned or center-aligned mode, with the frequency determined by the reload register GTIM_ARR, and the duty cycle and delay determined by the two capture/comparison registers GTIM_CCRy.

In this mode, the OCyREFC signal is generated by the logic OR operation or the combination of logic AND operation of the two reference signals OCyREF, and the combined PWM mode can be independently selected through the OCyM bit field of the comparison mode registers (GTIM_CCMR1CMP and GTIM_CCMR2CMP) as shown in the table below:

Table 13-11 Combined PWM mode configuration

OCyM bit field value	Output state
0xC (Combined PWM mode 1)	OC1REFC (or OC2REFC) is generated by the logical OR operation of OC1REF and OC2REF. OC3REFC (or OC4REFC) is generated by the logical OR operation of OC3REF and OC4REF.
0xD (Combined PWM mode 1)	OC1REFC (or OC2REFC) is generated by the logical AND operation of OC1REF and OC2REF. OC3REFC (or OC4REFC) is generated by the logic AND operation of OC3REF and OC4REF.

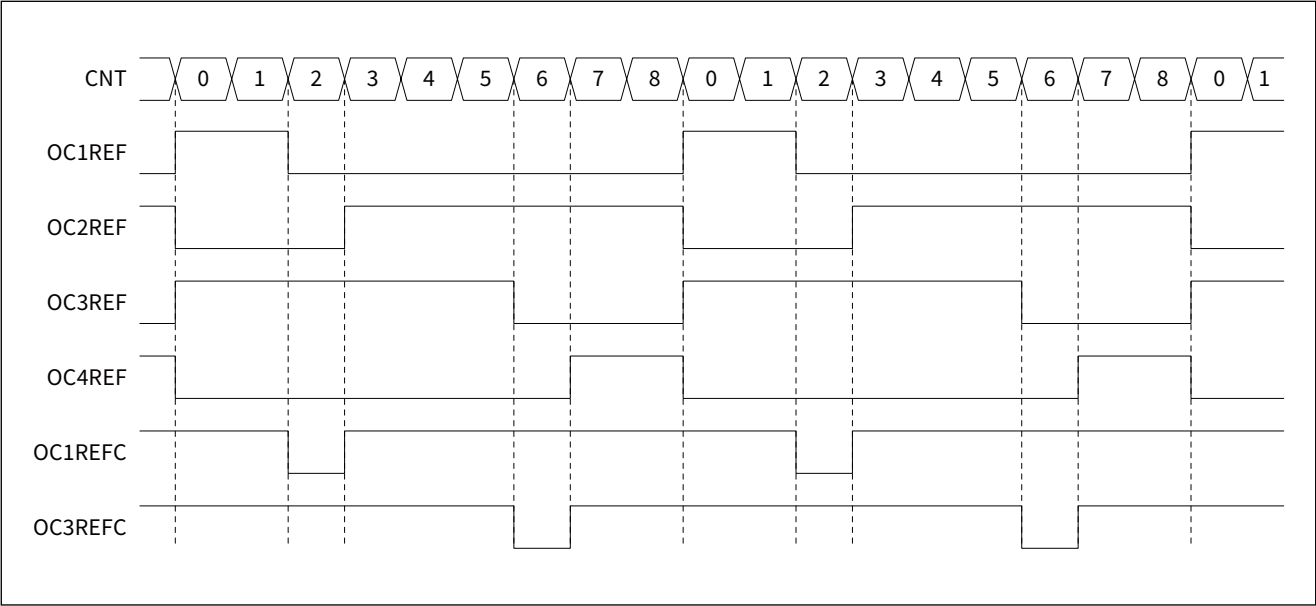
When the set channel is used as a combined PWM output channel, the other auxiliary channel can still be used, but it must be set to the opposite PWM mode. For example, when channel CH1 is configured to generate the OC1REFC signal in combined PWM mode 1, channel CH2 can still be used, and CH2 can be configured to output the OC2REF signal in PWM mode 2.

The following figure shows an example of the signals for the combined PWM mode 1 in the edge-aligned count up mode, which is configured as follows:

- The reload value ARR is 8.
- Channel 1 is configured as combined PWM mode 1, CCR1=2.
- Channel 2 is configured as PWM mode 2, CCR2=3.
- Channel 3 is configured as combined PWM mode 1, CCR3=6.
- Channel 4 is configured as PWM mode 2, CCR4=7.



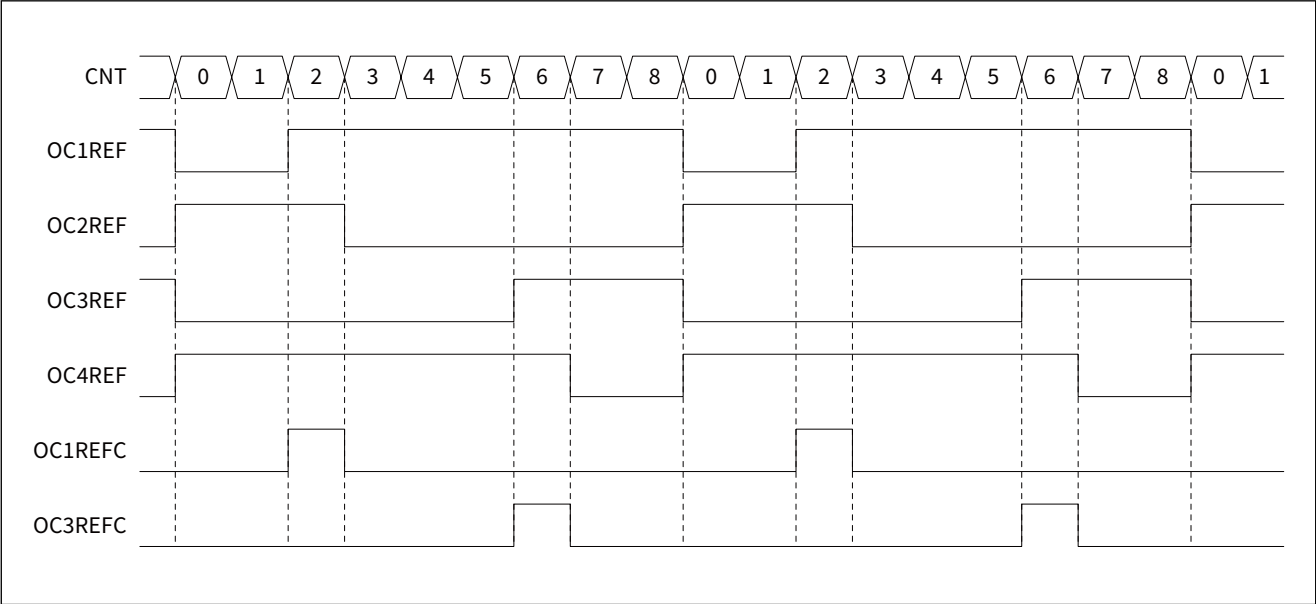
Figure 13-42 Example of combined PWM mode 1



The following figure shows an example of the signals for the combined PWM mode 2 in the edge-aligned count up mode, which is configured as follows:

- The reload value ARR is 8.
- Channel 1 is configured as combined PWM mode 2 , CCR1=2.
- Channel 2 is configured as PWM mode 1 , CCR2=3.
- Channel 3 is configured as combined PWM mode 2, CCR3=6.
- Channel 4 is configured as PWM mode 1, CCR4=7.

Figure 13-43 Example of combined PWM mode 2



13.3.4.6 Retriggerable single pulse mode

The retriggerable single pulse mode, in conjunction with the combined reset+trigger mode (SMS=8), allows the counter to be reset and activated by a trigger signal (TRGI) and generates pulses of programmable length, with the pulse width determined by the ARR.

The differences compared to the single pulse mode are as follows:

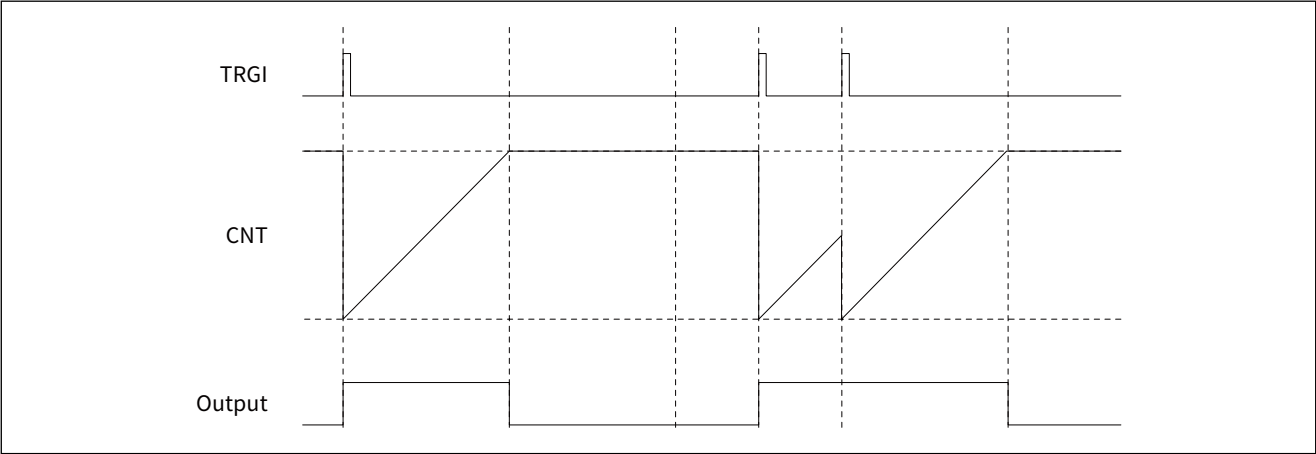
Table 13-12 Retriggerable single pulse mode and single pulse mode

Single pulse mode	Retriggerable single pulse mode
When a trigger occurs, a single pulse with a programmable pulse width is generated after a programmable delay.	When a trigger occurs, the pulse is generated immediately without programmable delay
The new trigger is invalid.	If the pulse generated by the previous trigger is not completed and a new trigger occurs, the pulse will be prolonged

In this mode, the counter can only be set to edge-aligned mode, not center-aligned mode. When configured for counting up mode, the corresponding CCRy must be set to 0. When configured for counting down mode, the corresponding CCRy must be greater than or equal to ARR.

The following figure shows an example of retriggerable single pulse mode 2:

Figure 13-44 Example of retriggerable single pulse mode 2



13.3.4.7 Count direction output

Set the OCyM bit field of the GTIM_CCMRxCMP register to 0xB, which outputs a count direction signal (a copy of the DIR bit in the GTIM_CR1 register) on the corresponding CHy channel.

This function can be used to monitor the counting direction (or direction of rotation) in encoder mode, or to indicate the up/down phase signals in center-aligned PWM mode.

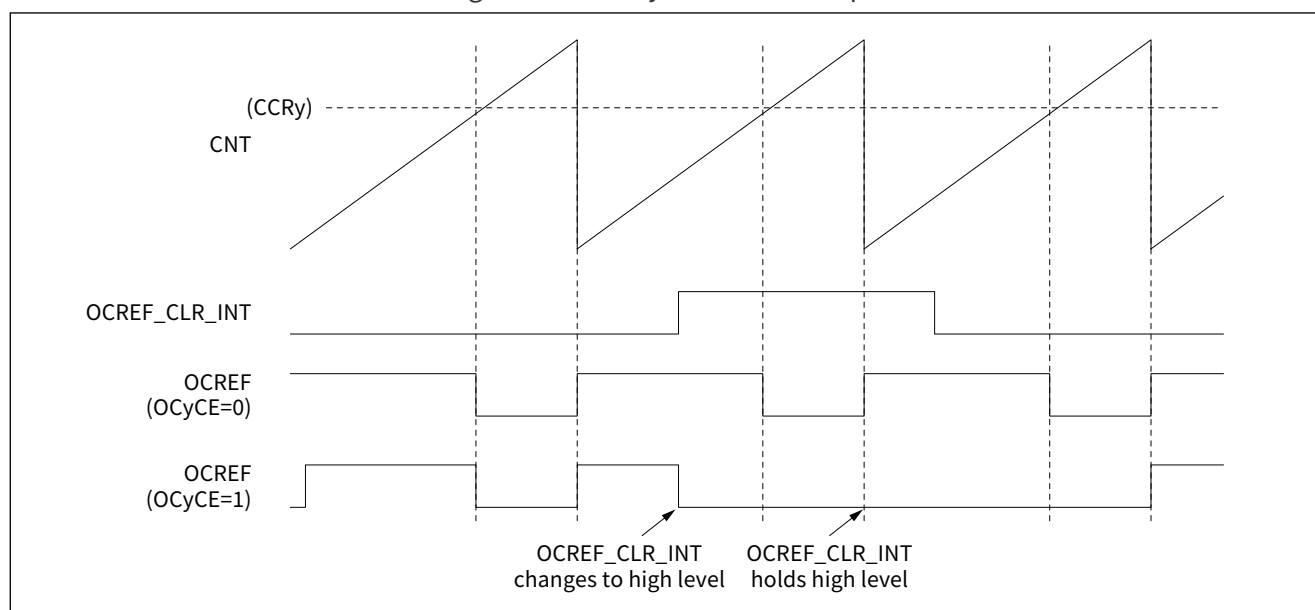
13.3.4.8 Reference signal clear function

Set the OCyCE bit field of the GTIM_CCMRxCMP register to 1, which enables the OCyREF signal clearing function of the corresponding channel, i.e., when a high level is detected on the OCREf_CLR_INT input, the OCyREF signal can be cleared to zero and the OCyREF signal will remain low until the next update event UEV occurs. This function is only available in output compare and PWM modes and is not available in forced output mode.

The clearing source OCREf_CLR_INT can be selected as either OCREf_CLR or ETRF, which is selected by the OCCS bit field of the slave mode control register GTIM_SMCR, and there are also multiple input sources for OCREf_CLR, see the description of the OCRSEL bit field in [13.9.23 GTIM_AF2 multiplexing function option register 2](#).

The following figure shows the OCyREF signal state when the OCREf_CLR_INT input goes high:

Figure 13-45 OCyREF clear example



13.3.5 UIF bit remap

Set the UIFREMAP bit in the GTIM_CR1 register to 1, which enables the UIF status bit remapping function, forcing the update interrupt flag GTIM_ISR.UIF to be copied consecutively into the UIFCPY bit field of the GTIM_CNT register. This allows automatically reading the counter value and the potential flip-flop condition issued by the UIFCPY flag. This simplifies the calculation of angular velocity by avoiding competing conditions when processing is shared between background tasks (counter reads) and interrupts (update interrupts).

There is no delay between UIF and UIFCPY flag enable.

When GTIM_CR1.UIFREMAP is 0, the UIFCPY bit field is reserved and read as 0.

13.3.6 Timer cascade ITR

Timer cascading can be achieved by using the ITR signal, i.e. connecting the trigger output (TRGO) of the master timer to the ITR input of the slave timer, which in turn is connected to the trigger input (TRGI) of the slave timer. The ITR cascade can perform a variety of functions, such as using one timer as a prescaler for another timer, or the master timer performing a reset, enabling, or providing a clock to the slave timer's counter.

The source of the GTIM's cascade input ITR is the trigger output signals (TRGO) of the BTIM and ATIM, and TXD/RXD signals of UART, which can be selected by the TS bit field of the slave mode control register GTIM_SMCR, as configured in the table below:

Table 13-13 ITR signal source

GTIM_SMCR.TS bit field value	ITR signal name	ITR signal source
00000	ITR0	ATIM_Trgo
00001	ITR1	-
00010	ITR2	-
00011	ITR3	-
01000	ITR4	-
01001	ITR5	-
01010	ITR6	UART1_TXD
01011	ITR7	UART2_TXD
01100	ITR8	UART1_RXD
01101	ITR9	UART2_RXD
01110	ITR10	BTIM1_Trgo
01111	ITR11	BTIM2_Trgo
10000	ITR12	BTIM3_Trgo

By controlling the MMS bit field of register GTIM_CR2, it is possible to select the message (TRGO) that will be sent to the slave timer for synchronisation in GTIM master mode, connected to the ITR input of the other timer. The following table shows this:



Table 13-14 GTIM trigger outputs (TRGO)

GTIM_CR2.MMS	TRGO signal
0000	Reset
0001	Counter enable signal CNT_EN
0010	Update event
0011	Channel 1 compare capture pulse
0100	Channel 2 compare capture pulse
0101	Channel 3 compare capture pulse
0110	Channel 4 compare capture pulse
0111	OC1REF signal
1000	OC2REF signal
1001	OC3REF signal
1010	OC4REF signal
1011	OC1REFC signal
1100	OC2REFC signal
1101	OC3REFC signal
1110	OC4REFC signal
1111	Encoder clock output



13.3.7 On-chip peripheral interconnect ETR

The source of the GTIM's ETR signal can be the external GTIM_ETR pin or other peripherals on-chip, which can be selected through the ETRSEL bit field of the multiplexing function option register GTIM_AF1.

When the ETR signal is input from the GTIM_ETR pin, the specific external input port can be referred to [Table 8-2 GPIO multiplexing function assignment table](#), and needs to be multiplexed and configured through the GPIO multiplexing function registers (GPIOx_AFRH and GPIOx_AFRL); when the ETR signal is input from other peripheral devices on-chip, it can realise the on-chip and off-chip peripheral interconnection.

The following table shows the sources of ETR signals:

Table 13-15 ETR signal source

AF1.ETRSEL bit field value	ETR signal source
0000	GTIM_ETR pin
0001	VC1_OUT
0010	VC2_OUT
0101	ADC_AWD
0111	HSE_FAULT
1000	LVD_OUT
1101	UART1_TXD
1110	UART2_TXD
1111	LSE_FAULT



13.4 GTIM interrupt

GTIM supports 10 interrupt sources. When a GTIM interrupt event occurs, the interrupt flag bit will be set by hardware, and an interrupt request will be generated if the corresponding interrupt enable control bit is set.

In the user GTIM interrupt service procedure, the relevant GTIM interrupt flag bit should be queried for corresponding processing, and the interrupt flag bit should be cleared before exiting the interrupt service procedure to avoid entering the interrupt service procedure repeatedly.

The following table shows the flag bit, interrupt enable bit, interrupt flag clear bit or clear method of each GTIM interrupt source:

Table 13-16 GTIM interrupt control

Interrupt event	Interrupt flag bit	Interrupt enable bit	Interrupt flag clear
Update interrupt	ISR.UIF	IER.UIE	Write 0 to ICR.UIF
Capture/compare 1 interrupt	ISR.CC1IF	IER.CC1IE	Write 0 to ICR.CC1IF or read CCR1 register
Capture/compare 2 interrupt	ISR.CC2IF	IER.CC2IE	Write 0 to ICR.CC2IF or read CCR2 register
Capture/compare 3 interrupt	ISR.CC3IF	IER.CC3IE	Write 0 to ICR.CC3IF or read CCR3 register
Capture/compare 4 interrupt	ISR.CC4IF	IER.CC4IE	Write 0 to ICR.CC4IF or read CCR4 register
Trigger interrupt	ISR.TIF	IER.TIE	Write 0 to ICR.TIF
Encoder index interrupt	ISR.IDXF	IER.IDXIE	Write 0 to ICR.IDXF
Encoder direction change interrupt	ISR.DIRF	IER.DIRIE	Write 0 to ICR.DIRF
Encoder index error interrupt	ISR.IERRF	IER.IERRIE	Write 0 to ICR.IERRF
Encoder transition error interrupt	ISR.TERRF	IER.TERRIE	Write 0 to ICR.TERRF



13.5 Trigger ADC

The timer supports a variety of internal signals to trigger the startup of the ADC, such as capturing the compare signal and triggering the output signal TRGO. There are several possible events for triggering the output TRGO, which are selected by the MMS bit field of the control register GTIM_CR2. At the same time, the ADC peripherals need to configure their external trigger start registers to select the corresponding trigger source.

It should be noted that the ADC clock must be enabled before an event can be received from the main timer and that the ADC clock must not be changed in real time when the trigger signal is received from the timer.



13.6 Debugging support

GTIM supports stopping or continuing counting in debug mode, which is set up by the GTIM bit field of the debug state timer control register SYSCTRL_DEBUG:

- Set SYSCTRL_DEBUG.GTIM to 1, then the counter of GTIM is suspended in the debug state.
- Set SYSCTRL_DEBUG.GTIM to 0, then the counter of GTIM continues to count while in the debug state.



13.7 Programming examples

13.7.1 External clock mode 1 programming example

The following example configures the incremental counter to count on the rising edge of the GTIM_CH1 channel, with the following steps:

- Step 1: Set SYSCTRL_AHBEN.GPIOx to 1 and SYSCTRL_APBEN1.GTIM to 1, enable the GPIO clock corresponding to the GTIM_CH1 pin and the GTIM configuration clock and operating clock;
- Step 2: Configure the GPIO corresponding to the GTIM_CH1 pin into the multiplexed input mode, for specific register configurations, please refer to [8 General-purpose input/output \(GPIO\)](#);
- Step 3: Set GTIM_TISEL.TI1SEL to 0 to select the TI1 source as the GTIM_CH1 channel;
- Step 4: Set GTIM_CCMR1CAP.IC1F to configure the input filter bandwidth;
- Step 5: Set GTIM_CCER.CC1NP to 0 and GTIM_CCER.CC1P to 0 to select the rising edge count to be valid;
- Step 6: Set GTIM_SMCR.TS to 0x5 to select the TRGI input as TI1FP1;
- Step 7: Set GTIM_SMCR.SMS to 0x7 to make GTIM to work in external clock mode 1;
- Step 8: Set GTIM_IER.UIE to 1 and configure the corresponding NVIC to enable the update interrupt;
- Step 9: Configure the counter prescaler GTIM_PSC;
- Step 10: Set the expected reload value GTIM_ARR;
- Step 11: Set GTIM_CR1.CEN to 1 to enable the counter;
- Step 12: When the counter overflows, the GTIM_ISR.UIF flag bit is set to 1 to enter the interrupt service routine, and set GTIM_ICR.UIF to 0 to clear this interrupt flag.

13.7.2 External clock mode 2 programming example

The following is a programming example for external clock mode 2 + trigger mode, where the GTIM_CH1 input triggers to start the counter when a rising edge occurs, and the counter counts on each rising edge of the GTIM_ETR input signal, with the following steps:

- Step 1: Set SYSCTRL_AHBEN.GPIOx to 1 and SYSCTRL_APBEN1.GTIM to 1, enable the GPIO clocks corresponding to the GTIM_ETR and GTIM_CH1 pins and the GTIM configuration clock and operating clock;
- Step 2: Configure the GPIOs corresponding to the GTIM_ETR, GTIM_CH1 pins into the multiplexed input mode, for specific register configurations, please refer to [8 General-purpose input/output \(GPIO\)](#);
- Step 3: Set GTIM_AF1.ETRSEL to 0 to select the ETR source as the GTIM_ETR pin;
- Step 4: Set GTIM_SMCR.ETP to 0 to select ETR rising edge valid;
- Step 5: Set GTIM_SMCR.ETPS to 0 to disable the prescaler for ETR;
- Step 6: Set GTIM_SMCR.ETF to 0 to disable ETR input filtering;
- Step 7: Set GTIM_SMCR.ECE to 1 to enable external clock mode 2;
- Step 8: Set GTIM_TISEL.TI1SEL to 0 to select the TI1 source as the GTIM_CH1 channel;
- Step 9: Set GTIM_CCMR1CAP.IC1F to 0 to disable TI1 input filtering;
- Step 10: Set GTIM_CCER.CC1NP to 0 and GTIM_CCER.CC1P to 0 to select detection of TI1 rising edge;
- Step 11: Set GTIM_SMCR.TS to 0x5 to select the TRGI input as TI1FP1;
- Step 12: Set GTIM_SMCR.SMS to 0x6 to make GTIM work in trigger mode;
- Step 13: When enabled on a rising edge of the GTIM_CH1 input, the counter counts on each rising edge of the GTIM_ETR input signal.



13.7.3 Reset mode programming example

In the following example, the counter is reinitialised when a rising edge occurs on the GTIM_CH1 input, with the following steps:

- Step 1: Set SYSCTRL_AHBEN.GPIOx to 1 and SYSCTRL_APBEN1.GTIM to 1. Enable the GPIO clock corresponding to the GTIM_CH1 pin and the GTIM configuration clock and operating clock;
- Step 2: Configure the GPIO corresponding to the GTIM_CH1 pin into the multiplexed input mode, for specific register configurations, please refer to [8 General-purpose input/output \(GPIO\)](#);
- Step 3: Set GTIM_TISEL.TI1SEL to 0 to select the TI1 source as the GTIM_CH1 channel;
- Step 4: Set GTIM_CCMR1CAP.IC1F to configure the input filter bandwidth;
- Step 5: Set GTIM_CCER.CC1NP to 0 and GTIM_CCER.CC1P to 0 to select the detection of the rising edge;
- Step 6: Set GTIM_SMCR.TS to 0x5 to select the TRGI input as TI1FP1;
- Step 7: Set GTIM_SMCR.SMS to 0x4 to make GTIM to work in reset mode;
- Step 8: Set the expected reload value GTIM_ARR;
- Step 9: Set GTIM_CR1.CEN to 1 to enable the counter and start normal counting;
- Step 10: When a rising edge occurs on the GTIM_CH1 input, the counter is cleared to zero and counting starts again from 0.

13.7.4 Gated mode programming example

In the following example, the signal input from the GTIM_CH1 pin is used as the gated signal to control the counter counting, with the following steps:

- Step 1: Set SYSCTRL_AHBEN.GPIOx to 1 and SYSCTRL_APBEN1.GTIM to 1. Enable the GPIO clock corresponding to the GTIM_CH1 pin and the GTIM configuration clock and operating clock;
- Step 2: Configure the GPIO corresponding to the GTIM_CH1 pin into the multiplexed input mode, for specific register configurations, please refer to [8 General-purpose input/output \(GPIO\)](#);
- Step 3: Set GTIM_TISEL.TI1SEL to 0 to select the TI1 source as the GTIM_CH1 channel;
- Step 4: Set GTIM_CCMR1CAP.IC1F to configure the input filter bandwidth;
- Step 5: Set GTIM_CCER.CC1NP to 0 and GTIM_CCER.CC1P to 0 to select detect high level;
- Step 6: Set GTIM_SMCR.TS to 0x5 to select the TRGI input as TI1FP1;
- Step 7: Set GTIM_SMCR.SMS to 0x5 to make GTIM to work in gated mode;
- Step 8: Set the expected reload value GTIM_ARR;
- Step 9: Set GTIM_CR1.CEN to 1 to enable the counter;
- Step 10: When GTIM_CH1 input is high, the counter starts counting; when GTIM_CH1 input is low, the counter stops counting.



13.7.5 Trigger mode programming example

In the following example, the counter is triggered to start when a rising edge occurs on the GTIM_CH1 input, with the following steps:

- Step 1: Set SYSCTRL_AHBEN.GPIOx to 1 and SYSCTRL_APBEN1.GTIM to 1. Enable the GPIO clock corresponding to the GTIM_CH1 pin and the GTIM configuration clock and operating clock;
- Step 2: Configure the GPIO corresponding to the GTIM_CH1 pin into the multiplexed input mode, for specific register configurations, please refer to [8 General-purpose input/output \(GPIO\)](#);
- Step 3: Set GTIM_TISEL.TI1SEL to 0 to select the TI1 source as the GTIM_CH1 channel;
- Step 4: Set GTIM_CCMR1CAP.IC1F to configure the input filter bandwidth;
- Step 5: Set GTIM_CCER.CC1NP to 0 and GTIM_CCER.CC1P to 0 to select the detection of the rising edge;
- Step 6: Set GTIM_SMCR.TS to 0x5 to select the TRGI input as TI1FP1;
- Step 7: Set GTIM_SMCR.SMS to 0x6 to make GTIM to work in trigger mode;
- Step 8: Set the expected reload value GTIM_ARR;
- Step 9: When a rising edge occurs on the GTIM_CH1 input, the counter starts counting and the TIF flag is set to 1.

13.7.6 Quadrature encoder mode programming example

In the following example, configure GTIM to work in quadrature encoder mode-x4 mode, with the following steps:

- Step 1: Set SYSCTRL_AHBEN.GPIOx to 1, SYSCTRL_APBEN1.GTIM to 1, and enable the GPIO clocks corresponding to the GTIM_CH1 and GTIM_CH2 pins as well as the GTIM configuration clock and operating clock;
- Step 2: Configure the GPIO corresponding to the GTIM_CH1 and GTIM_CH2 pins into the multiplexed input mode, for specific register configurations, please refer to [8 General-purpose input/output \(GPIO\)](#);
- Step 3: Set GTIM_TISEL.TI1SEL to 0 to select the TI1 source as GTIM_CH1 channel;
- Step 4: Set GTIM_TISEL.TI2SEL to 0 to select the TI2 source as the GTIM_CH2 channel;
- Step 5: Set GTIM_CCMR1CAP.IC1F and GTIM_CCMR1CAP.IC2F to configure the input filter bandwidth;
- Step 6: Set GTIM_CCER.CC1NP and GTIM_CCER.CC1P to 0, CH1 input is not inverted;
- Step 7: Set GTIM_CCER.CC2NP and GTIM_CCER.CC2P to 0, CH2 input is not inverted;
- Step 8: Set GTIM_SMCR.SMS to 0x3 to make GTIM work in quadrature encoder mode-x4 mode;
- Step 9: Set the appropriate reload value GTIM_ARR;
- Step 10: Set GTIM_CR1.CEN to 1 to enable the counter.



13.7.7 Input capture programming example

13.7.7.1 Basic input capture mode

In the following example, execute an input capture when a rising edge occurs on the GTIM_CH1 input, with the following steps:

Step 1: Set SYSCTRL_AHBEN.GPIOx to 1, SYSCTRL_APBEN1.GTIM to 1, and enable the GPIO clock corresponding to the GTIM_CH1 pin and the GTIM configuration clock and operating clock;

Step 2: Configure the GPIO corresponding to the GTIM_CH1 pin into the multiplexed input mode, for specific register configurations, please refer to [8 General-purpose input/output \(GPIO\)](#);

Step 3: Set GTIM_TISEL.TI1SEL to 0 to select the TI1 source as the GTIM_CH1 channel;

Step 4: Set GTIM_CCMR1CAP.IC1F to configure the TI1 input filter bandwidth;

Step 5: Set GTIM_CCER.CC1NP to 0 and GTIM_CCER.CC1P to 0 to select detection of IC1 rising edge;

Step 6: Set GTIM_CCMR1CAP.CC1S to 1, the CC1 channel is configured as an input and IC1 is mapped to TI1;

Step 7: Set GTIM_CCMR1CAP.IC1PSC to 0 to disable the IC1 prescaler;

Step 8: Set GTIM_CCER.CC1E to 1 to enable input capture 1 mode;

Step 9: Set GTIM_IER.CC1IE to 1 and configure the corresponding NVIC to enable the input capture 1 interrupt;

Step 10: Configure the counter prescaler GTIM_PSC;

Step 11: Set the expected reload value GTIM_ARR;

Step 12: Set GTIM_CR1.CEN to 1 to enable the counter and start normal counting;

Step 13: When a capture occurs, the value of the current counter CNT is latched into the capture/compare register CCR1, completing a capture, while the GTIM_ISR.CC1IF flag bit is set to 1 to enter the interrupt service procedure, and this interrupt flag is cleared by setting GTIM_ICR.CC1IF to 0.



13.7.7.2 PWM input mode

The following example measures the period and duty cycle of the PWM signal input from the GTIM_CH1 pin, with the following steps:

- Step 1: Set SYSCTRL_AHBEN.GPIOx to 1, SYSCTRL_APBEN1.GTIM to 1, and turn on the GPIO clock corresponding to the GTIM_CH1 pin and the GTIM configuration clock and operating clock;
- Step 2: Configure the GPIO corresponding to the GTIM_CH1 pin into the multiplexed input mode, for specific register configurations, please refer to [8 General-purpose input/output \(GPIO\)](#);
- Step 3: Set GTIM_TISEL.TI1SEL to 0 to select the TI1 source as the GTIM_CH1 channel;
- Step 4: Set GTIM_CCMR1CAP.IC1F to configure the TI1 input filter bandwidth;
- Step 5: Set GTIM_CCER.CC1NP to 0 and GTIM_CCER.CC1P to 0 to select detection of IC1 rising edge;
- Step 6: Set GTIM_CCMR1CAP.CC1S to 1, the CC1 channel is configured as an input and IC1 is mapped to TI1;
- Step 7: Set GTIM_CCMR1CAP.IC1PSC to 0 to disable the IC1 prescaler;
- Step 8: Set GTIM_CCER.CC1E to 1 to enable input capture 1 mode;
- Step 9: Set GTIM_IER.CC1IE to 1 and configure the corresponding NVIC to enable input capture 1 interrupt;
- Step 10: Set GTIM_CCER.CC2NP to 0 and GTIM_CCER.CC2P to 1 to select detect IC2 falling edge;
- Step 11: Set GTIM_CCMR1CAP.CC2S to 2, CC2 channel configured as input, IC2 mapped to TI1;
- Step 12: Set GTIM_CCMR1CAP.IC2PSC to 0 to disable the IC2 prescaler;
- Step 13: Set GTIM_CCER.CC2E to 1 to enable input capture 2 mode;
- Step 14: Set GTIM_SMCR.TS to 0x5 to select the TRGI input as TI1FP1;
- Step 15: Set GTIM_SMCR.SMS to 0x4 to enable the GTIM to work in reset mode;
- Step 16: Configure the counter prescaler GTIM_PSC;
- Step 17: Set the expected reload value GTIM_ARR;
- Step 18: Set GTIM_CR1.CEN to 1 to enable the counter and start normal counting.



13.7.8 Output compare programming example

In the following example, channel CH1 outputs the set waveform externally, with the following steps:

- Step 1: Set `SYSCTRL_AHBEN.GPIOx` to 1, `SYSCTRL_APBEN1.GTIM` to 1, enable the GPIO clock corresponding to the `GTIM_CH1` pin and the GTIM configuration clock and working clock;
- Step 2: Configure the GPIO corresponding to the `GTIM_CH1` pin into the multiplexed output mode, for specific register configurations, please refer to [8 General-purpose input/output \(GPIO\)](#);
- Step 3: Set `GTIM_CCER.CC1P` to 0 to select the output polarity;
- Step 4: Set `GTIM_CCMR1CMP.CC1S` to 0 to configure the CC1 channel as an output;
- Step 5: Set `GTIM_CCMR1CMP.OC1M` to select output compare 1 mode;
- Step 6: Set `GTIM_CCER.CC1E` to 1 to output the OC1 signal on the corresponding output pin;
- Step 7: Configure the counter prescaler `GTIM_PSC`;
- Step 8: Set the expected reload value `GTIM_ARR`;
- Step 9: Set the expected comparison value `GTIM_CCR1`;
- Step 10: Set `GTIM_IER.CC1IE` to 1 and configure the corresponding NVIC to enable the compare 1 interrupt;
- Step 11: Set `GTIM_CR1.CEN` to 1 to enable the counter;
- Step 12: When a compare match occurs, the `GTIM_ISR.CC1IF` flag bit is set to 1 to enter the interrupt service procedure, and this interrupt flag is cleared by setting `GTIM_ICR.CC1IF` to 0.



13.8 List of registers

GTIM base address: GTIM_BASE = 0x4000 1800

Table 13-17 List of GTIM registers

Register name	Register address	Register description
GTIM_CR1	GTIM_BASE + 0x00	Control register 1
GTIM_CR2	GTIM_BASE + 0x04	Control register 2
GTIM_SMCR	GTIM_BASE + 0x08	Slave mode control register
GTIM_IER	GTIM_BASE + 0x0C	Interrupt enable register
GTIM_ISR	GTIM_BASE + 0x10	Interrupt flag register
GTIM_ICR	GTIM_BASE + 0x70	Interrupt flag clear register
GTIM_EGR	GTIM_BASE + 0x14	Event generation register
GTIM_CCMR1CAP	GTIM_BASE + 0x18	Capture mode register 1
GTIM_CCMR1CMP	GTIM_BASE + 0x18	Compare mode register 1
GTIM_CCMR2CAP	GTIM_BASE + 0x1C	Capture mode register 2
GTIM_CCMR2CMP	GTIM_BASE + 0x1C	Compare mode register 2
GTIM_CCER	GTIM_BASE + 0x20	Capture/compare enable register
GTIM_CNT	GTIM_BASE + 0x24	Count register
GTIM_PSC	GTIM_BASE + 0x28	Prescaler register
GTIM_ARR	GTIM_BASE + 0x2C	Auto reload register
GTIM_CCR1	GTIM_BASE + 0x34	Capture/compare register 1
GTIM_CCR2	GTIM_BASE + 0x38	Capture/compare register 2
GTIM_CCR3	GTIM_BASE + 0x3C	Capture/compare register 3
GTIM_CCR4	GTIM_BASE + 0x40	Capture/compare register 4
GTIM_ECR	GTIM_BASE + 0x58	Encoder control register
GTIM_TISEL	GTIM_BASE + 0x5C	TI input selection register
GTIM_AF1	GTIM_BASE + 0x60	Multiplexing function option register 1
GTIM_AF2	GTIM_BASE + 0x64	Multiplexing function option register 2



13.9 Register descriptions

See section [1 Documentation conventions](#) for abbreviations used in register descriptions.

13.9.1 GTIM_CR1 Controls register 1

Address offset: 0x00 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:12	RFU	-	Reserved bits, please keep the default value
11	UIFREMAP	RW	UIF status bit remap 0: Remapping is disabled, GTIM_CNT[31] remains 0 1: Remapping is enabled, GTIM_CNT[31] is equivalent to GTIM_ISR.UIF
10	RFU	-	Reserved bits, please keep the default value
9:8	CKD	RW	Configure the PCLK clock to filter clock division ratio 00: $f_{DTS} = f_{PCLK}/1$ 01: $f_{DTS} = f_{PCLK}/2$ 10: $f_{DTS} = f_{PCLK}/4$ Caution: The filter clock is used for ETR filtering and TI filtering.
7	ARPE	RW	Auto reload preload enable 0: Disable ARR register buffering 1: Enable ARR register buffering
6:5	CMS	RW	Center-aligned mode selection 00: Edge aligned mode. The counter counts up or down according to the direction bit (DIR). 01: Center-aligned mode 1. The counter alternates between counting up or down. Only when the counter is counting down, the output compare interrupt flag of the channel configured as an output (CCyS=00 in the GTIM_CCMRx register) is set to 1. 10: Center aligned mode 2. the counter alternates between counting up and counting down. Only when the counter is counting up, the output compare interrupt flag of the channel configured as an output (CCyS=00 in the GTIM_CCMRx register) is set to 1. 11: Center aligned mode 3. The counter alternates between counting up and counting down. When the counter counts up or down, the output compare interrupt flag of the channel configured as an output (CCyS=00 in the GTIM_CCMRx register) is set to 1. Caution 1: As long as the counter is enabled (CEN=1), it must not be switched from edge-aligned mode to center-aligned mode. Caution 2: The initial count direction for centre-aligned mode is determined by the DIR bit field.



Bit field	Name	Permission	Function description
4	DIR	RW/RO	Count direction 0: Counter count up 1: Counter count down <i>Caution: When the timer is configured in center-aligned mode or encoder mode, this bit is read-only.</i>
3	OPM	RW	Single pulse mode 0: Counter does not stop counting when an update event occurs 1: Counter stops counting when the next update event occurs (clears the CEN bit to zero)
2	URS	RW	Update request source This bit is set to 1 and cleared to 0 by software to select the UEV event source. 0: When UEV is enabled, all of the following events generate an update interrupt. Including: - Counter overflow/underflow - Set UG bit to 1 - Update events generated by the slave mode controller 1: When UEV is enabled, only counter overflow/underflow generates an update interrupt.
1	UDIS	RW	Update disabled This bit is set to 1 and cleared to 0 by software to enable/disable UEV event generation. 0: UEV enabled. Update events (UEV) can be generated by one of the following events, which then updates the value of the shadow register. - Counter overflow/underflow - Set UG bit to 1 - Update events generated by the slave mode controller 1: UEV disabled. No update event is generated and the values of the respective shadow registers (ARR, PSC, and CCRy) remain unchanged. However, if the UG bit field is set to 1, or a hardware reset is received from the slave mode controller, the counter and prescaler are reinitialised.
0	CEN	RW	Counter enable 0: Counter disabled 1: Counter enabled <i>Caution 1: Only if the CEN bit field is set to 1 in advance by software can external clock, gated mode and encoder mode be used. Trigger modes can be used by automatically setting the CEN bit field to 1 via hardware.</i> <i>Caution 2: In single pulse mode, the CEN bit is automatically cleared to 0 when an update event occurs.</i>



13.9.2 GTIM_CR2 Control register 2

Address offset: 0x04 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:27	RFU	-	Reserved bits, please keep the default value
26:25	MMSH	RW	Master mode selection It is the highest bit of the MMS and must be used with the MMS.
24:8	RFU	-	Reserved bits, please keep the default value
7	TI1S	RW	TI1 input selection 0: GTIM_CH1 pin is connected to the TI1 input 1: XOR combination of GTIM_CH1, CH2 and CH3 pins are connected to TI1 input
6:4	MMS	RW	Master mode selection These bits select the information that will be sent to the slave timer for synchronisation in master mode (TRGO). Used in conjunction with MMSH, these bits are combined as follows: 0000: Reset - The UG bit in the GTIM_EGR register is used as the trigger output (TRGO). If the reset is generated by a trigger input (slave mode controller configured for reset mode), the signal on TRGO is delayed compared to the actual reset. 0001: Enable - The counter enable signal CNT_EN is used as a trigger output (TRGO). This trigger output can be used to start multiple timers at the same time, or to control the enabling of slave timers over a period of time. The counter enable signal consists of a logical sum operation combining the CEN control bit with the trigger input in gated mode. When the counter enable signal is controlled by a trigger input, there is a delay on the TRGO, except when master/slave mode is selected (please refer to the description of the MSM bit in 13.9.3 GTIM_SMCR Slave mode control register). 0010: Update - Selects the update event as the trigger output (TRGO). For example, the master timer can be used as a prescaler for the slave timer. 0011: Compare capture pulse - Once an input capture or compare match event occurs on channel 1, the trigger output sends a positive pulse (TRGO), even if the CC1IF flag has been set to 1 (or even if it has been high-level). 0100: Compare capture pulse - Once an input capture or compare match event occurs on channel 2, the trigger output sends a positive pulse (TRGO), even if the CC2IF flag has been set to 1 (or even if it has been high-level).



Bit field	Name	Permission	Function description
6:4 (continued)	MMS	RW	0101: Compare capture pulse - Once an input capture or compare match event occurs on channel 3, the trigger output sends a positive pulse (TRGO), even if the CC3IF flag has been set to 1 (or even if it has been high-level). 0110: Compare capture pulse - Once an input capture or compare match event occurs on channel 4, the trigger output sends a positive pulse (TRGO), even if the CC4IF flag has been set to 1 (or even if it has been high-level). 0111: OC1_REF - OC1REF signal is used as trigger output (TRGO). 1000: OC2_REF - OC2REF signal is used as trigger output (TRGO). 1001: OC3_REF - OC3REF signal is used as trigger output (TRGO). 1010: OC4_REF - OC4REF signal is used as trigger output (TRGO). 1011: OC1_REFC - OC1REFC signal is used as trigger output (TRGO). 1100: OC2_REFC - OC2REFC signal is used as trigger output (TRGO). 1101: OC3_REFC - OC3REFC signal is used as trigger output (TRGO). 1110: OC4_REFC - OC4REFC signal is used as trigger output (TRGO). 1111: Encoder clock output - The encoder clock signal is used as a trigger output (TRGO). This code applies to the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. No other SMS[3:0] values are permitted, which may result in unexpected behaviour. Caution: The clock of the slave timer or ADC must be enabled before an event can be received from the master timer; and the clock of the slave timer or ADC must not be changed in real time when a trigger signal is received from the master timer.
3:0	RFU	-	Reserved bits, please keep the default value



13.9.3 GTIM_SMCR Slave mode control register

Address offset: 0x08 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:26	RFU	-	Reserved bits, please keep the default value
25	SMSPS	RW	Preload source This bit selects the trigger source for the transfer of the SMS[3:0] bit fields from the preload register to the active registers 0: Transfer is triggered by a timer update event 1: Transfer is triggered by an index event
24	SMSPE	RW	Preload enable This bit selects whether the SMS[3:0] bits are preloaded or not 0: Preload disabled 1: Preload enabled
23:22	RFU	-	Reserved bits, please keep the default value
21:20	TSH	RW	Trigger selection It is the highest bit of the TS, and needs to be used in conjunction with the TS.
19:17	RFU	-	Reserved bits, please keep the default value
16	SMSH	RW	Slave mode selection It is the highest bit of SMS, and needs to be used in conjunction with the SMS.
15	ETP	RW	External trigger polarity This bit selects whether ETR or the inversion of ETR will be used for the trigger operation 0: ETR not inverted, high or rising edge is valid 1: ETR inverted, low level or falling edge is valid
14	ECE	RW	External clock enabled This bit enables external clock mode 2 0: Disable external clock mode 2 1: Enables external clock mode 2, the counter clock is provided by any valid edge of the ETRF signal. <i>Caution 1: Set the ECE bit field to 1 which has the same effect as selecting External Clock Mode 1 and connecting TRGI to ETRF (SMS=111 and TS=00111).</i> <i>Caution 2: External clock mode 2 can be used in conjunction with the following slave modes: reset mode, gated mode and trigger mode. However, in this case TRGI must not be connected to ETRF (TS bit must not be 00111).</i> <i>Caution 3: If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.</i>



Bit field	Name	Permission	Function description
13:12	ETPS	RW	External trigger prescaler The external trigger signal ETRP frequency must not exceed 1/4 of the PCLK frequency. the ETRP frequency can be reduced by enabling the prescaler. This method is useful when a fast external clock is input. 00: Prescaler disabled 01: 2-division ETRP frequency 10: 4-division ETRP frequency 11: 8-division ETRP frequency
11:8	ETF	RW	External trigger filter This bit field defines the sampling frequency of the ETRP signal and the digital filter bandwidth applicable to the ETRP. The digital filter consists of an event counter, where only every N consecutive events are considered as a valid output edge: 0000: No filter, sampled at frequency f_{DTS} 0001: $f_{SAMPLING}=f_{DTS}/1$, N=2 0010: $f_{SAMPLING}=f_{DTS}/1$, N=4 0011: $f_{SAMPLING}=f_{DTS}/1$, 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
7	MSM	RW	Master/ slave mode 0: No action is performed 1: The action of the trigger input event (TRGI) of the current timer is postponed to allow perfect synchronisation (via TRGO) of the current timer with its slave timers. This setting applies when multiple timers are synchronised by a single external event.



Bit field	Name	Permission	Function description
6:4	TS	RW	Trigger input TRGI selection This bit field selects the trigger input that will be used to synchronise the counter. 00000: Internal trigger 0 (ITR0) 00001: Internal trigger 1 (ITR1) 00010: Internal trigger 2 (ITR2) 00011: Internal trigger 3 (ITR3) 00100: TI1 edge detector (TI1F_ED) 00101: Filtered timer input 1 (TI1FP1) 00110: Filtered timer input 2 (TI2FP2) 00111: External trigger input (ETRF) 01000: Internal trigger 4 (ITR4) 01001: Internal trigger 5 (ITR5) 01010: Internal trigger 6 (ITR6) 01011: Internal trigger 7 (ITR7) 01100: Internal trigger 8 (ITR8) 01101: Internal trigger 9 (ITR9) 01110: Internal trigger 10 (ITR10) 01111: Internal trigger 11 (ITR11) 10000: Internal trigger 12 (ITR12) Other values: Reserved Caution 1: These bits can only be changed if they are not used (e.g. SMS=0000) to avoid incorrect edge detection during transition. Caution 2: The other bit fields are in bits 21 and 20 of the same register and need to be used in conjunction with the TS. Caution 3: For specific ITRx sources, please refer to Table 13-13 ITR signal source ..
3	OCCS	RW	OCREF cleared selection This bit is used to select the OCREF clear source. 0: OCREF_CLR_INT connected to OCREF_CLR input. 1: OCREF_CLR_INT connected to ETRF. Caution: OCREF_CLR input source please refer to 13.9.23 GTIM_AF2 multiplexing function option register 2 for the description of the OCRSEL bit field.
2:0	SMS	RW	Slave mode selection When an external signal is selected, the valid edge of the trigger signal (TRGI) is related to the polarity selected on the external input (see the input related control registers). 0000: Slave mode disabled - If CEN = '1', the prescaler clock is supplied directly from the internal clock. 0001: Quadrature encoder mode - x2 mode , the counter counts up/down on the TI1FP1 edge according to the TI2FP2 level.



Bit field	Name	Permission	Function description
2:0 (continued)	SMS	RW	0010: Quadrature encoder mode - x2 mode, the counter counts up/down according to the TI1FP1 level at the TI2FP2 edge. 0011: Quadrature encoder mode - x4 mode, the counter counts on the TI1FP1 and TI2FP2 edges, the direction of the count depends on the level of the other input. 0100: Reset mode - The counter is reinitialised and a register update event is generated on the occurrence of a rising edge of the selected trigger input (TRGI). 0101: Gated mode - Enables the counter clock when the trigger input (TRGI) is high. The counter stops counting (but not resetting) as soon as the trigger input goes low. Counter start and stop are controlled. 0110: Trigger mode - Starts the counter (but does not reset) on a rising edge of the trigger signal (TRGI). Only the start of the counter is controlled. 0111: External clock mode 1 - Counter clock is provided by the rising edge of the selected trigger signal (TRGI). 1000: Combined reset + trigger mode - Reinitialises the counter on the rising edge of the selected trigger input (TRGI), generates a register update event and starts the counter. 1001: Combined gated + reset mode - The counter is enabled and starts counting when the trigger input (TRGI) goes high, the counter stops counting and is reset when the trigger input goes low, in this mode the start and stop of the counter are controlled. 1010: Encoder mode - Clock + direction, x2 mode. 1011: Encoder mode - Clock+direction, x1 mode, TI2FP2's edge polarity is set by CC2P. 1100: Encoder mode - Clock with direction, x2 mode. 1101: Encoder mode - Clock with direction, x1 mode, TI1FP1 and TI2FP2 edge polarity is set by CC1P and CC2P. 1110: Quadrature encoder mode - x1 mode, counts only TI1FP1 edges, whose edge polarity is set by CC1P. 1111: Quadrature encoder mode - x1 mode, counts only TI2FP2 edges, whose polarity is set by CC2P. Caution 1: If <i>TI1F_ED</i> is selected as the trigger input (<i>TS=00100</i>), the gated mode must not be used. In fact, <i>TI1F_ED</i> outputs one pulse for each <i>TI1F</i> transition, while the gated mode checks the level of the trigger signal. Caution 2: The clock of the slave peripheral (timer, ADC, etc.) that receives the <i>TRGO</i> signal must be enabled before the event can be received from the master timer; and the clock frequency must not be changed in real time (prescaler) when the trigger signal is received from the master timer.



13.9.4 GTIM_IER Interrupt enable register

Address offset: 0x0C Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:24	RFU	-	Reserved bits, please keep the default value
23	TERRIE	RW	Encoder transition error interrupt enable control 0: Disable transition error interrupt 1: Enable transition error interrupt
22	IERRIE	RW	Encoder index error interrupt enable control 0: Disable index error interrupt 1: Enable index error interrupt
21	DIRIE	RW	Encoder direction change interrupt enabled 0: Disable direction change interrupt 1: Enable direction change interrupt
20	IDXIE	RW	Encoder index interrupt enable control 0: Disable index interrupt 1: Enable index interrupt
19:7	RFU	-	Reserved bits, please keep the default value
6	TIE	RW	Trigger interrupt enabled 0: Disable trigger interrupt 1: Enable trigger interrupt
5	RFU	-	Reserved bits, please keep the default value
4	CC4IE	RW	Capture/compare 4 interrupt enabled 0: Disable capture/compare 4 interrupt 1: Enable capture/compare 4 interrupt
3	CC3IE	RW	Capture/compare 3 interrupt enabled 0: Disable capture/compare 3 interrupt 1: Enable capture/compare 3 interrupt
2	CC2IE	RW	Capture/compare 2 interrupt enabled 0: Disable capture/compare 2 interrupt 1: Enable capture/compare 2 interrupt
1	CC1IE	RW	Capture/compare 1 interrupt enabled 0: Disable capture/compare 1 interrupt 1: Enable capture/compare 1 interrupt
0	UIE	RW	Update interrupt enabled 0: Disable update interrupt 1: Enable update interrupt



13.9.5 GTIM_ISR Interrupt flag register

Address offset: 0x10 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:24	RFU	-	Reserved bits, please keep the default value
23	TERRF	RW	Encoder transition error interrupt flag 0: No transition error occurred 1: A transition error has occurred
22	IERRF	RW	Encoder index error interrupt flag 0: No index error occurred 1: An indexing error has occurred
21	DIRF	RW	Encoder direction change interrupt flag 0: No encoding direction change has occurred 1: Encoding direction change has occurred
20	IDXF	RW	Encoder index interrupt flag 0: Index flag has not occurred 1: Index flag has occurred
19:13	RFU	-	Reserved bits, please keep the default value
12	CC4OF	RW	Capture/compare 4 repeat capture flag, see CC1OF description
11	CC3OF	RW	Capture/compare 3 repeat capture flag, see CC1OF description
10	CC2OF	RW	Capture/compare 2 repeat capture flag, see CC1OF description
9	CC1OF	RW	Capture/compare 1 repeat capture flag This flag bit is set to 1 by hardware only when the corresponding channel is configured in input capture mode. 0: No repeat capture detected 1: Counter value has been captured in the GTIM_CCR1 register and the CC1IF flag has been set to 1
8:7	RFU	-	Reserved bits, please keep the default value
6	TIF	RW	Trigger interrupt flag In all modes except gated mode, this flag will be set by hardware to 1 when a TRGI trigger event occurs after enabling the slave mode controller (a valid edge is detected on the TRGI input.) When gated mode is selected, this flag will be set to 1 when the counter is started and stopped. This flag needs to be cleared by software. 0: No trigger event occurred 1: Trigger interrupt pending
5	RFU	-	Reserved bits, please keep the default value
4	CC4IF	RW	Capture/compare 4 interrupt flag, see CC1IF description



Bit field	Name	Permission	Function description
3	CC3IF	RW	Capture/compare 3 interrupt flag, see CC1IF description
2	CC2IF	RW	Capture/compare 2 interrupt flag, see CC1IF description
1	CC1IF	RW	Capture/compare 1 interrupt flag This flag is set to 1 by hardware. It is cleared by software (Input capture or output compare modes), or by reading the GTIM_CCR1 register (Input capture mode only). 0: No compare match/input capture event occurs 1: Compare match/input capture event occurs If channel CC1 is configured as an output: This flag is set to 1 when the value of the counter GTIM_CNT matches the value of the GTIM_CCR1 register. When the value of GTIM_CCR1 is greater than the value of GTIM_ARR, the CC1IF bit will become high-level when the counter overflows (in count up mode and center-aligned mode) or underflows (in count down mode) occur. There are 3 possible flag setting options in center-aligned mode, see the CMS bit in the GTIM_CCR1 register for a complete description. If channel CC1 is configured as input: This bit is set to 1 if: the counter value is captured in the GTIM_CCR1 register (according to the edge sensitivity defined by the CC1P and CC1NP bit settings in GTIM_CCER and an edge is detected on IC1).
0	UIF	RW	Update interrupt flag This bit is set to 1 by hardware when an update event occurs, but needs to be cleared by software. 0: No update occurred 1: Update interrupt pending. This bit is set to 1 by hardware when the register is updated under the following conditions: - Overflow or underflow and when UDIS=0 in the GTIM_CR1 register. - URS=0 and UDIS=0 in the GTIM_CR1 register and when the CNT is reinitialised by software using the UG bit in the GTIM_EGR register. - URS=0 and UDIS=0 in the GTIM_CR1 register and when the CNT is reinitialised by a trigger event (see slave mode control register description).



13.9.6 GTIM_ICR Interrupt flag clear register

Address offset: 0x70 Reset value: 0x00F0 1E5F

Bit field	Name	Permission	Function description
31:24	RFU	-	Reserved bits, please keep the default value
23	TERRF	R1W0	Encoder transition error interrupt flag cleared 0: Clear encoder transition error interrupt flag 1: No function
22	IERRF	R1W0	Encoder index error interrupt flag cleared 0: Clear encoder index error interrupt flag 1: No function
21	DIRF	R1W0	Encoder direction change interrupt flag cleared 0: Clear encoder direction change interrupt flag 1: No function
20	IDXF	R1W0	Encoder index interrupt flag cleared 0: Clear encoder index interrupt flag 1: No function
19:13	RFU	-	Reserved bits, please keep the default value
12	CC4OF	R1W0	Capture/compare 4 repeat capture flag cleared, see CC1OF description
11	CC3OF	R1W0	Capture/compare 3 repeat capture flag cleared, see CC1OF description
10	CC2OF	R1W0	Capture/compare 2 repeat capture flag cleared, see CC1OF description
9	CC1OF	R1W0	Capture/compare 1 repeat capture flag cleared 0: Clear capture/compare 1 repeat capture flag 1: No function
8:7	RFU	-	Reserved bits, please keep the default value
6	TIF	R1W0	Trigger interrupt flag cleared 0: Clear trigger interrupt flag 1: No function
5	RFU	-	Reserved bits, please keep the default value
4	CC4IF	R1W0	Capture/compare 4 interrupt flag cleared, see CC1IF description
3	CC3IF	R1W0	Capture/compare 3 interrupt flag cleared, see CC1IF description
2	CC2IF	R1W0	Capture/compare 2 interrupt flag cleared, see CC1IF description
1	CC1IF	R1W0	Capture/compare 1 interrupt flag cleared 0: Clear capture/compare 1 interrupt flag 1: No function
0	UIF	R1W0	Update interrupt flag cleared 0: Clear update interrupt flag 1: No function



13.9.7 GTIM_EGR Event generation register

Address offset: 0x14 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:7	RFU	-	Reserved bits, please keep the default value
6	TG	RW	Trigger generated This bit is set to 1 by software to generate an event and is automatically cleared by hardware. 0: No operation is performed 1: The TIF flag in the GTIM_ISR register is set to 1. When enabled, the associated interrupt event can occur.
5	RFU	-	Reserved bits, please keep the default value
4	CC4G	RW	Capture/compare 4 generation, see CC1G description
3	CC3G	RW	Capture/compare 3 generation, see CC1G description
2	CC2G	RW	Capture/compare 2 generation, see CC1G description
1	CC1G	RW	Capture/compare 1 generated This bit is set to 1 by software to generate an event and is automatically cleared by hardware. 0: No operation is performed 1: Capture/compare events are generated on channel 1 and are classified as follows depending on the configuration of CC1: If channel CC1 is configured as output: - When enabled, the CC1IF flag is set to 1 and the corresponding interrupt request is sent. If channel CC1 is configured as an input: - The current value of the counter will be captured in the GTIM_CCR1 register. When enabled, the CC1IF flag is set to 1 and the corresponding interrupt request is sent. If the CC1IF flag is already high, the CC1OF flag will be set to 1. Caution: CC1E must be enabled, otherwise CCR1 register will capture the current value of the counter, but no interrupt flag will be generated, and no interrupt can be entered.
0	UG	RW	Update generated This bit can be set to 1 by software and automatically cleared by hardware. 0: No operation is performed 1: Reinitialise the counter and generate a register update event. Note that the prescaler counter will also be cleared (but the prescaler ratio is not affected). The counter will be cleared if center-aligned mode or up-count mode is selected; if down-count mode is selected, the counter will use the auto-reload value GTIM_ARR.



13.9.8 GTIM_CCMR1CAP Capture mode register 1

Address offset: 0x18 Reset value: 0x0000 0000

This register is the same register as GTIM_CCMR1CMP and can be used in input capture mode or output compare mode. The channel direction is defined by configuring the corresponding CCyS bit. All other bits of this register function differently in input capture mode and output compare mode.

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:12	IC2F	RW	Input capture 2 filter, see IC1F description
11:10	IC2PSC	RW	Input capture 2 prescaler, see IC1PSC description
9:8	CC2S	RW	Capture/compare 2 selection This bit field defines the channel direction (input/output) and the input mapping used. 00: CC2 channel configured as output 01: CC2 channel configured as input, IC2 mapped to TI2 10: CC2 channel configured as input, IC2 mapped to TI1 11: CC2 channel configured as input, IC2 mapped to TRC, this mode is only valid when the internal trigger input is selected via the TS bit (GTIM_SMCR register) Caution: Data can be written to the CC2S bit only when the channel is closed (CC2E=0 in GTIM_CCER).
7:4	IC1F	RW	Input capture 1 filter This bit field defines the sampling frequency of the TI1 input and the sample length of the digital filter applicable to TI1. The digital filter consists of an event counter, where every N consecutive events are considered to be only one valid output edge. 0000: No filter, sampled at frequency f_{DTS} 0001: $f_{SAMPLING}=f_{PCLK}$, N=2 0010: $f_{SAMPLING}=f_{PCLK}$, N=4 0011: $f_{SAMPLING}=f_{PCLK}$, N=8 0100: $f_{SAMPLING}=f_{DTS}/2$, N=6 0101: $f_{SAMPLING}=f_{DTS}/2$, N=8 0110: $f_{SAMPLING}=f_{DTS}/4$, N=6 0111: $f_{SAMPLING}=f_{DTS}/4$, N=8 1000: $f_{SAMPLING}=f_{DTS}/8$, N=6 1001: $f_{SAMPLING}=f_{DTS}/8$, N=8 1010: $f_{SAMPLING}=f_{DTS}/16$, N=5 1011: $f_{SAMPLING}=f_{DTS}/16$, N=6 1100: $f_{SAMPLING}=f_{DTS}/16$, N=8 1101: $f_{SAMPLING}=f_{DTS}/32$, N=5 1110: $f_{SAMPLING}=f_{DTS}/32$, N=6 1111: $f_{SAMPLING}=f_{DTS}/32$, N=8



Bit field	Name	Permission	Function description
3:2	IC1PSC	RW	Input capture 1 prescaler This bit field defines the prescaler ratio for the CC1 input (IC1). The prescaler is reset as soon as CC1E=0 (GTIM_CCER register). 00: No prescaler, capture is performed every time an edge is detected on the capture input. 01: Capture is performed every 2 events. 10: Capture is performed every 4 events. 11: Capture is performed every 8 events.
1:0	CC1S	RW	Capture/compare 1 selection This bit field defines the channel direction (input/output) and the input mapping used. 00: CC1 channel configured as output 01: CC1 channel configured as input, IC1 mapped to TI1 10: CC1 channel configured as input, IC1 mapped to TI2 11: CC1 channel configured as input, IC1 mapped to TRC, this mode is only valid when internal trigger input is selected via the TS bit (GTIM_SMCR register) Caution: Data can be written to the CC1S bit only when the channel is off (CC1E=0 in GTIM_CCER).

13.9.9 GTIM_CCMR1CMP Compare mode register 1

Address offset: 0x18 Reset value: 0x0000 0000

This register is the same register as GTIM_CCMR1CAP and can be used in input capture mode or output compare mode. The channel direction is defined by configuring the corresponding CCyS bit. All other bits of this register function differently in input capture mode and output compare mode.

Bit field	Name	Permission	Function description
31:25	RFU	-	Reserved bits, please keep the default value
24	OC2MH	RW	Used with OC2M, it is the highest bit of OC2M
23:17	RFU	-	Reserved bits, please keep the default value
16	OC1MH	RW	Used with OC1M, it is the highest bit of OC1M
15	OC2CE	RW	Output compare 2 clear enabled, see OC1CE description
14:12	OC2M	RW	Output compare 2 mode, see OC1M description
11	OC2PE	RW	Output compare 2 preload enabled, see OC1PE description
10	OC2FE	RW	Output compare 2 fast enabled, see OC1FE description



Bit field	Name	Permission	Function description
9:8	CC2S	RW	Capture/compare 2 selection This bit field defines the channel direction (input/output) and the input mapping used. 00: CC2 channel configured as output 01: CC2 channel configured as input, IC2 mapped to TI2 10: CC2 channel configured as input, IC2 mapped to TI1 11: CC2 channel configured as input, IC2 mapped to TRC, this mode is only valid when internal trigger input is selected via the TS bit (GTIM_SMCR register) Caution: Data can be written to the CC2S bit only when the channel is disabled (CC2E=0 in GTIM_CCER).
7	OC1CE	RW	Output compare 1 clear enabled 0: OC1REF is not affected by the OCREF_CLR_INT input 1: OC1REF is immediately cleared when a high level is detected on the OCREF_CLR_INT input
6:4	OC1M	RW	Output compare 1 mode These bits define the behaviour of the output reference signal OC1REF which provides OC1. OC1REF is valid at high level, while the valid level of OC1 depends on the CC1P bit. 0000: Freeze - Comparison of output compare register GTIM_CCR1 with counter GTIM_CNT will not have any effect on the output. (This mode is used for time base generation). 0001: Set channel 1 to output a valid level when matched. The OC1REF signal is forced to high level when counter GTIM_CNT is matched with capture/compare register 1 (GTIM_CCR1). 0010: Set channel 1 to output an invalid level when matched. When counter GTIM_CNT matches capture/compare register 1 (GTIM_CCR1), the OC1REF signal is forced to low level. 0011: Flipped - OC1REF is flipped when GTIM_CNT = GTIM_CCR1. 0100: Forced to invalid level - OC1REF is forced to low level. 0101: Forced to valid level - OC1REF is forced to high level. 0110: PWM mode 1 - In count up mode, channel 1 is valid (OC1REF=1) as long as GTIM_CNT < GTIM_CCR1, otherwise it is invalid (OC1REF=0). In count down mode, channel 1 is invalid (OC1REF=0) as long as GTIM_CNT > GTIM_CCR1, otherwise it is valid (OC1REF=1).



Bit field	Name	Permission	Function description
6:4 (continued)	OC1M	RW	0111: PWM mode 2 - In count up mode, channel 1 is invalid (OC1REF=0) whenever GTIM_CNT < GTIM_CCR1, otherwise it is valid (OC1REF=1). In count down mode, channel 1 is valid (OC1REF=1) as long as GTIM_CNT > GTIM_CCR1, otherwise it is invalid (OC1REF=0). 1000: Retriggerable OPM mode 1 - In count up mode, the channel is in valid state until a trigger event is detected (on the TRGI signal). Then, a comparison is made in PWM mode 1 and the channel will become valid again at the next update. In count down mode, the channel is invalid until a trigger event is detected (on the TRGI signal). Then, a comparison is made in PWM mode 1 and the channel becomes invalid again at the next update. 1001: Retriggerable OPM mode 2 - In count up mode, the channel is invalid until a trigger event is detected (on the TRGI signal). Then, a comparison is made in PWM mode 2 and the channel becomes invalid again at the next update. In count down mode, the channel is valid until a trigger event is detected (on the TRGI signal). Then, the comparison is performed in PWM mode 2 and the channel becomes valid again at the next update. 1010: Reserved 1011: Count direction output, OC1REF signal is the count direction signal for the DIR bit. 1100: Combined PWM mode 1 - OC1REF behaves the same as in PWM mode 1. OC1REFC is the logical OR of OC1REF and OC2REF. 1101: Combined PWM mode 2 - OC1REF behaves the same as in PWM mode 2. OC1REFC is the logical AND of OC1REF and OC2REF. 1110: Asymmetric PWM mode 1 - OC1REF behaves the same as in PWM mode 1. OC1REFC outputs OC1REF when the counter counts up and OC2REF when the counter counts down. 1111: Asymmetric PWM mode 2 - OC1REF behaves the same as in PWM mode 2. OC1REFC outputs OC1REF when the counter counts up and OC2REF when the counter counts down. Caution: In PWM mode, OCyREF level will be changed only when the comparison result is changed or the output comparison mode is switched from 'Freeze' mode to 'PWM' mode.



Bit field	Name	Permission	Function description
3	OC1PE	RW	Output compare 1 preload enabled 0: Disable the preload register related to GTIM_CCR1. Data can be written to GTIM_CCR1 at any time and the new value will be used immediately after writing. 1: Enable the preload register related to GTIM_CCR1. Read/write access to the preloaded registers. The GTIM_CCR1 preloaded value is loaded into the valid registers each time that an update event is generated. Caution: <i>PWM mode without validation of the preload register is only possible in single pulse mode (OPM bit field in the GTIM_CR1 register is set to 1). The behaviour is not guaranteed in other cases.</i>
2	OC1FE	RW	Output compare 1 fast enabled This bit is used to speed up the effect of a trigger input event on the CC output. 0: Even if the trigger is on, CC1 will work properly according to the counter and CCR1 values. When an edge is present on the trigger input, the minimum delay to activate the CC1 output is 5 clock cycles. 1: The presence of a valid edge on the trigger input corresponds to a comparison match on the CC1 output. Subsequently, OC is set to the compare level regardless of the comparison result. The delay time between sampling the trigger input and activating the CC1 output is reduced to 3 clock cycles. OCyFE works only when the channel is configured for PWM1 or PWM2 mode.
1:0	CC1S	RW	Capture/compare 1 selection This bit field defines the channel direction (input/output) and the input used. 00: CC1 channel configured as output 01: CC1 channel configured as input, IC1 mapped to TI1 10: CC1 channel configured as input, IC1 mapped to TI2 11: CC1 channel configured as input, IC1 mapped to TRC, this mode is only valid when the internal trigger input is selected via the TS bit (GTIM_SMCR register) Caution: <i>Data can be written to the CC1S bit only when the channel is off (CC1E=0 in GTIM_CCR).</i>



13.9.10 GTIM_CCMR2CAP Capture mode register 2

Address offset: 0x1C Reset value: 0x0000 0000

This register is the same register as GTIM_CCMR2CMP and can be used in input capture mode or output compare mode. The channel direction is defined by configuring the corresponding CCyS bit. All other bits of this register function differently in input capture mode and output compare mode.

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:12	IC4F	RW	Input capture 4 filter, see IC1F description
11:10	IC4PSC	RW	Input capture 4 prescaler, see IC1PSC description
9:8	CC4S	RW	Capture/compare 4 selection This bit field defines the channel direction (input/output) and the input mapping used. 00: CC4 channel configured as output 01: CC4 channel configured as input, IC4 mapped to TI4 10: CC4 channel configured as input, IC4 mapped to TI3 11: CC4 channel configured as input, IC4 mapped to TRC, this mode is only valid when internal trigger input is selected via the TS bit (GTIM_SMCR register) Caution: Data can be written to the CC4S bit only when the channel is off (CC4E=0 in GTIM_CCER).
7:4	IC3F	RW	Input capture 3 filter, see IC1F description
3:2	IC3PSC	RW	Input capture 3 prescaler, see IC1PSC description
1:0	CC3S	RW	Capture/compare 3 selection This bit field defines the channel direction (input/output) and the input mapping used. 00: CC3 channel configured as output 01: CC3 channel configured as input, IC3 mapped to TI3 10: CC3 channel configured as input, IC3 mapped to TI4 11: CC3 channel configured as input, IC3 mapped to TRC, this mode is only valid when internal trigger input is selected via the TS bit (GTIM_SMCR register) Caution: Data can be written to the CC3S bit only when the channel is off (CC3E=0 in GTIM_CCER).



13.9.11 GTIM_CCMR2CMP Compare mode register 2

Address offset: 0x1C Reset value: 0x0000 0000

This register is the same register as GTIM_CCMR2CAP and can be used in input capture mode or output compare mode. The channel direction is defined by configuring the corresponding CCyS bit. All other bits of this register function differently in input capture mode and output compare mode.

Bit field	Name	Permission	Function description
31:25	RFU	-	Reserved bits, please keep the default value
24	OC4MH	RW	Used with OC4M, it is the highest bit of the OC4M
23:17	RFU	-	Reserved bits, please keep the default value
16	OC3MH	RW	Used with OC3M, it is the highest bit of the OC3M
15	OC4CE	RW	Output compare 4 clear enabled, see OC1CE description
14:12	OC4M	RW	Output compare 4 mode, see OC1M description
11	OC4PE	RW	Output compare 4 preload enabled, see OC1PE description
10	OC4FE	RW	Output compare 4 fast enabled, see OC1FE description
9:8	CC4S	RW	Capture/compare 4 selection This bit field defines the channel direction (input/output) and the input used. 00: CC4 channel configured as output 01: CC4 channel configured as input, IC4 mapped to TI4 10: CC4 channel configured as input, IC4 mapped to TI3 11: CC4 channel configured as input, IC4 mapped to TRC, this mode is only valid when the internal trigger input is selected via the TS bit (GTIM_SMCR register) Caution: Data can be written to the CC4S bit only when the channel is off (CC4E=0 in GTIM_CCER).
7	OC3CE	RW	Output compare 3 clear to enabled, see OC1CE description
6:4	OC3M	RW	Output compare 3 mode, see OC1M description
3	OC3PE	RW	Output compare 3 preload enabled, see OC1PE description
2	OC3FE	RW	Output compare 3 fast enabled, see OC1FE description
1:0	CC3S	RW	Capture/compare 3 selection This bit field defines the channel direction (input/output) and the input used. 00: CC3 channel configured as output 01: CC3 channel configured as input, IC3 mapped to TI3 10: CC3 channel configured as input, IC3 mapped to TI4 11: CC3 channel configured as input, IC3 mapped to TRC. This mode is only valid when the internal trigger input is selected via the TS bit (GTIM_SMCR register) Caution: Data can be written to the CC3S bit only when the channel is off (CC3E=0 in GTIM_CCER).



13.9.12 GTIM_CCER Capture/compare enable register

Address offset: 0x20 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15	CC4NP	RW	Capture/compare 4 complementary output polarity, see CC1NP
14	RFU	-	Reserved bits, please keep the default value
13	CC4P	RW	Capture/compare 4 output polarity, see CC1P description
12	CC4E	RW	Capture/compare 4 output enabled, see CC1E description
11	CC3NP	RW	Capture/compare 3 complementary output polarity, see CC1NP
10	RFU	-	Reserved bits, please keep the default value
9	CC3P	RW	Capture/compare 3 output polarity, see CC1P description
8	CC3E	RW	Capture/compare 3 output enabled, see CC1E description
7	CC2NP	RW	Capture/compare 2 complementary output polarity, see CC1NP description
6	RFU	-	Reserved bits, please keep the default value
5	CC2P	RW	Capture/compare 2 output polarity, see CC1P description
4	CC2E	RW	Capture/compare 2 output enabled, see CC1E description
3	CC1NP	RW	Capture/compare 1 complementary output polarity CC1 channel configured as output: in this case CC1NP must be held clear. CC1 channel configured as input: this bit is used with CC1P to define TI1FP1/TI2FP1 polarity. See CC1P description.
2	RFU	-	Reserved bits, please keep the default value
1	CC1P	RW	Capture/compare 1 output polarity 0: OC1 is valid at high level (output mode) / edge sensitivity selection (input mode) 1: OC1 is valid at low level (output mode) / edge sensitivity selection (input mode) When the CC1 channel is configured as an input, the CC1NP/CC1P bits select the valid polarity of TI1FP1 and TI2FP1 for trigger or capture operations. CC1NP=0, CC1P=0: Non-inverting/rising edge trigger. The circuit is sensitive to the rising edge of TIxFP1 (performing a capture or trigger operation in reset mode, external clock mode, or trigger mode), and TIxFP1 is not inverted (performing a trigger operation in gated mode or encoder mode).



Bit field	Name	Permission	Function description
1 (Continued)	CC1P	RW	- CC1NP=0, CC1P=1: Inverted/falling edge trigger. Circuit is sensitive to TlxFP1 falling edge (performs capture or trigger operation in reset mode, external clock mode, or trigger mode) and TlxFP1 is inverted (performs trigger operation in gated mode or encoder mode). - CC1NP=1, CC1P=1: Non-inverting/dual edge trigger. The circuit is sensitive to both the rising and falling edges of TlxFP1 (performs a capture or trigger operation in reset mode, external clock mode, or trigger mode), and TlxFP1 is not inverted (performs a trigger operation in gated mode). This configuration must not be used in encoder mode. - CC1NP=1, CC1P=0: This configuration is reserved and must not be used.
0	CC1E	RW	Capture/compare 1 output enabled 0: Capture mode disabled/OC1 not active 1: Capture mode enabled / OC1 signal output on corresponding output pin <i>Caution: This bit must be configured to 1 when using the input capture feature.</i>

13.9.13 GTIM_CNT Count register

Address offset: 0x24 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31	UIFCPY	RO	According to the value of the UIFREMAP bit field in the GTIM_CR1 register, this bit field indicates different meanings: UIFREMAP = 0, this bit field is reserved and read as 0. UIFREMAP = 1, this bit field indicates a read-only copy of the UIF bit of the GTIM_ISR register.
30:16	RFU	-	Reserved bits, please keep the default value
15:0	CNT	RW	Count value



13.9.14 GTIM_PSC Prescaler register

Address offset: 0x28 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	PSC	RW	Prescaler value The counter clock frequency CK_CNT is equal to $f_{CK\_PSC} / (PSC[15:0] + 1)$. The PSC contains the value to be loaded into the valid prescaler register each time an update event occurs, including when the counter is cleared by the UG bit in the GTIM_EGR register, or by the trigger controller when configured for 'reset mode'.

13.9.15 GTIM_ARR Auto-reload register

Address offset: 0x2C Reset value: 0x0000 FFFF

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	ARR	RW	Auto reload value ARR is the value to be loaded into the real auto reload register, the counter does not work when the auto reload value is empty.



13.9.16 GTIM_CCR1 Capture/compare register 1

Address offset: 0x34 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	CCR1	RW	Capture/compare 1 value If channel CC1 is configured to output: CCR1 is the preload value for capture/compare register 1. This value takes effect immediately if the preload function is not enabled via the OC1PE bit in the GTIM_CCMR1CMP register; otherwise it takes effect only when an update event occurs (copied to the valid capture/compare register 1). The effective capture/compare register contains the value to be compared with counter GTIM_CNT and signalled on the OC1 output. If channel CC1 is configured as input: CCR1 is the counter value at the time of the last input capture 1 event (IC1). Only the GTIM_CCR1 register can be read, it cannot be programmed.

13.9.17 GTIM_CCR2 Capture/compare register 2

Address offset: 0x38 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	CCR2	RW	Capture/compare 2 value If channel CC2 is configured to output: CCR2 is the preload value for capture/compare register 2. This value takes effect immediately if the preload function is not enabled via the OC2PE bit in the GTIM_CCMR1CMP register; otherwise it takes effect only when an update event occurs (copied to the valid capture/compare register 2). The effective capture/compare register contains the value to be compared with counter GTIM_CNT and signalled on the OC2 output. If channel CC2 is configured as input: CCR2 is the counter value at the time of the last input capture 2 event (IC2). Only the GTIM_CCR2 register can be read, it cannot be programmed.



13.9.18 GTIM_CCR3 Capture/compare register 3

Address offset: 0x3C Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	CCR3	RW	Capture/compare 3 value If channel CC3 is configured to output: CCR3 is the preload value for capture/compare register 3. This value takes effect immediately if the preload function is not enabled via the OC3PE bit in the GTIM_CCMR2CMP register; otherwise it takes effect only when an update event occurs (copied to valid capture/compare register 3). The effective capture/compare register contains the value to be compared with counter GTIM_CNT and signalled on the OC3 output. If channel CC3 is configured as input: CCR3 is the counter value at the time of the last Input Capture 3 event (IC3). Only the GTIM_CCR3 register can be read, it cannot be programmed.

13.9.19 GTIM_CCR4 Capture/compare register 4

Address offset: 0x40 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	CCR4	RW	Capture/compare 4 value If channel CC4 is configured to output: CCR4 is the preload value for capture/compare register 4. This value takes effect immediately if the preload function is not enabled via the OC4PE bit in the GTIM_CCMR2CMP register; otherwise it takes effect only when an update event occurs (copied to the valid capture/compare register 4). The effective capture/compare register contains the value to be compared with counter GTIM_CNT and signalled on the OC4 output. If channel CC4 is configured as input: CCR4 is the counter value at the time of the last input capture 4 event (IC4). Only the GTIM_CCR4 register can be read, it cannot be programmed.



13.9.20 GTIM_ECR Encoder control register

Address offset: 0x58 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:8	RFU	-	Reserved bits, please keep the default value
7:6	IPOS	RW	Coding index position In quadrature encoder mode (SMS[3:0] = 0001, 0010, 0011, 1110, 1111), this bit indicates in which AB input configuration the index event resets the counter. 00: When AB = 00, the index resets the counter 01: When AB = 01, the index resets the counter 10: When AB = 10, the index resets the counter 11: When AB = 11, the index resets the counter In directional clock encoder mode or clock plus direction encoder mode (SMS[3:0] = 1010, 1011, 1100, 1101), these bits indicate at which level the index event resets the counter. In directional clock encoder mode, this applies to both clock inputs. x0: when the clock signal is low, the index resets the counter x1: When the clock signal is high, the index resets the counter
5	FIDX	RW	This bit indicates whether only the first index is considered 0: Index is always active 1: Only the first index resets the counter
4:3	RFU	-	Reserved bits, please keep the default value
2:1	IDIR	RW	Index direction This bit field indicates in which count direction the index resets the counter 00: Reset counter in any counting direction 01: Reset counter only when counting up 10: Reset counter only when counting down 11: Reserved
0	IE	RW	Index enabled 0: Index disabled 1: Index enabled



13.9.21 GTIM_TISEL TI input selection register

Address offset: 0x5C Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:28	RFU	-	Reserved bits, please keep the default value
27:24	TI4SEL	RW	TI4 channel input capture signal source selection, see TI1SEL description
23:20	RFU	-	Reserved bits, please keep the default value
19:16	TI3SEL	RW	TI3 channel input capture signal source selection, see TI1SEL description
15:12	RFU	-	Reserved bits, please keep the default value
11:8	TI2SEL	RW	TI2 channel input capture signal source selection, see TI1SEL description
7:4	RFU	-	Reserved bits, please keep the default value
3:0	TI1SEL	RW	TI1 channel input capture signal source selection 0000: GTIM_CH1 0001: VC1_OUT 0010: VC2_OUT 0011: UART1_RXD 0100: UART2_RXD 0101: MCO_OUT 0110: HSE_FAULT 0111: LSE_FAULT 1000: RTC_OUT 1001: LSI_OUT 1010: BTIM1_Trgo 1011: BTIM2_Trgo 1100: BTIM3_Trgo 1111: ATIM_Trgo



13.9.22 GTIM_AF1 Multiplexing function option register 1

Address offset: 0x60 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:18	RFU	-	Reserved bits, please keep the default value
17:14	ETRSEL	RW	ETR source selection for GTIM 0000: GTIM_ETR 0001: VC1_OUT 0010: VC2_OUT 0101: ADC_AWD 0111: HSE_FAULT 1000: LVD_OUT 1101: UART1_TXD 1110: UART2_TXD 1111: LSE_FAULT
13:0	RFU	-	Reserved bits, please keep the default value

13.9.23 GTIM_AF2 Multiplexing function option register 2

Address offset: 0x64 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:19	RFU	-	Reserved bits, please keep the default value
18:16	OCRSEL	RW	OCREF_CLR source selection 000: VC1_OUT 001: VC2_OUT 100: ADC_AWD 111: ETRF
15:0	RFU	-	Reserved bits, please keep the default value



14 Advanced-control timer (ATIM)

14.1 Overview

The advanced timer (ATIM) consists of a 16bit auto-reload counter driven by a programmable prescaler. ATIM supports a variety of operating modes such as timing, counting, reset, gating, triggering, and encoder with 6 independent capture/compare channels for 6 independent PWM outputs or 6 pairs of complementary PWM outputs with dead time, or capture of 6 inputs. It can be used for basic timing/counting, measuring the pulse width and period of input signals, and generating output waveforms (PWM, single pulse, complementary PWM with dead time insertion, etc.).

14.2 Main features

- 16bit counting up, down and up/down auto-reload counter.
- Programmable prescaler supports 1, 2, 3, 4, ..., 65536 divider frequencies.
- Support single count mode and continuous count mode.
- 6 independent input capture and output compare channels.
- Dead time programmable complementary PWM output.
- Supports double-point comparison phase shift and logic and phase shift.
- Supports 2 brake inputs.
- Trigger input signal (TRGI) control timer for multiple slave modes.
- Timer cascade ITR and on-chip peripheral interconnect ETR.
- Supports incremental (quadrature) encoder and Hall sensor circuits for positioning.
- Generates interrupt requests when multiple events occur:
 - Update event
 - Trigger event
 - Input capture
 - Output compare
 - Reversal event
 - Brake event
 - Encoder event

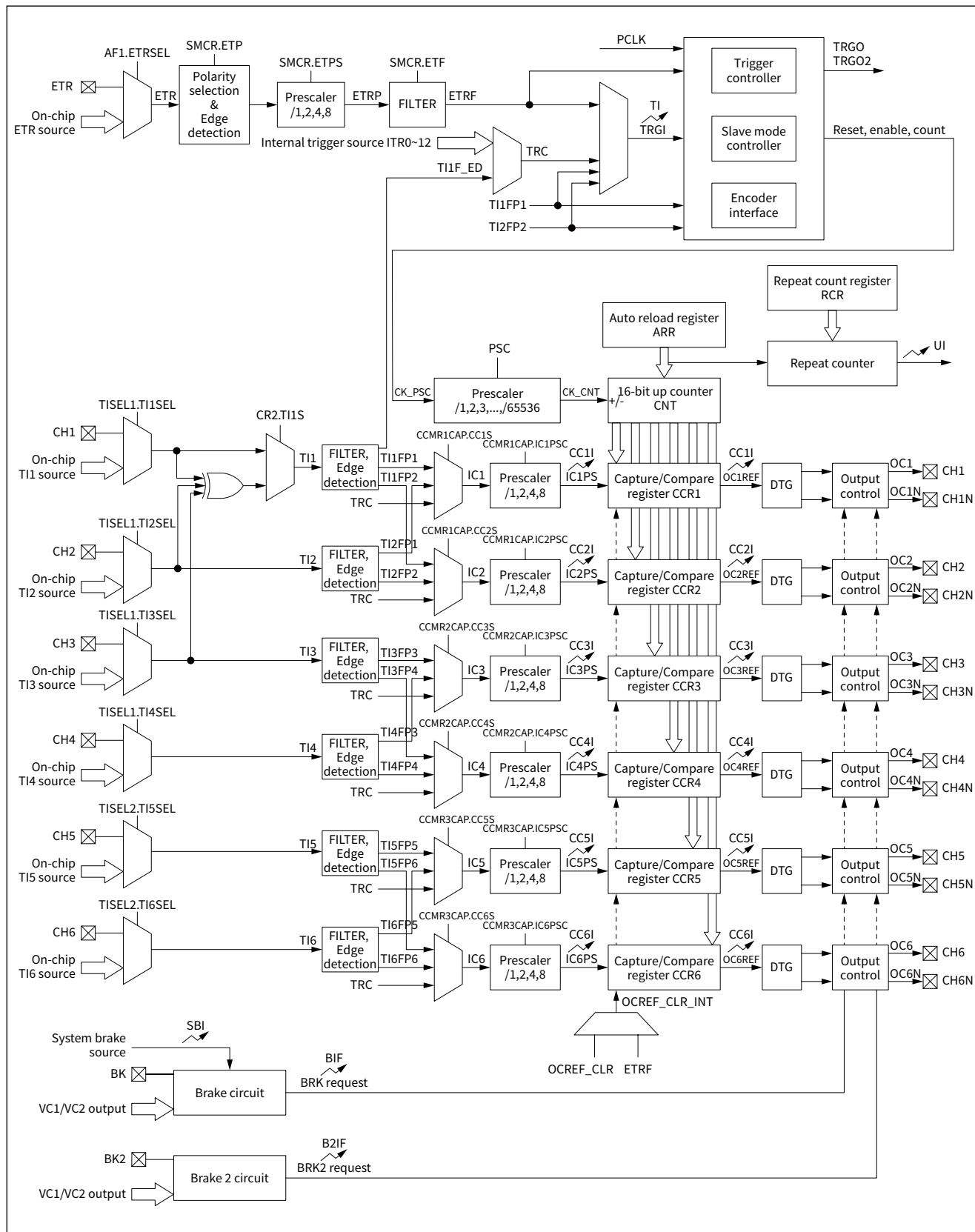


14.3 Functional description

14.3.1 Functional block diagram

The functional block diagram of ATIM is shown in the following figure:

Figure 14-1 ATIM functional block diagram



14.3.1.1 Clock source

The counter's counting clock source can be provided by the internal clock PCLK, the trigger signal TRGI or the external trigger input signal ETRF, which is divided by the prescaler ATIM_PSC to drive the counter for counting. Different clock sources are available in different operating modes, specific please refer to [14.3.2 Operating modes](#).

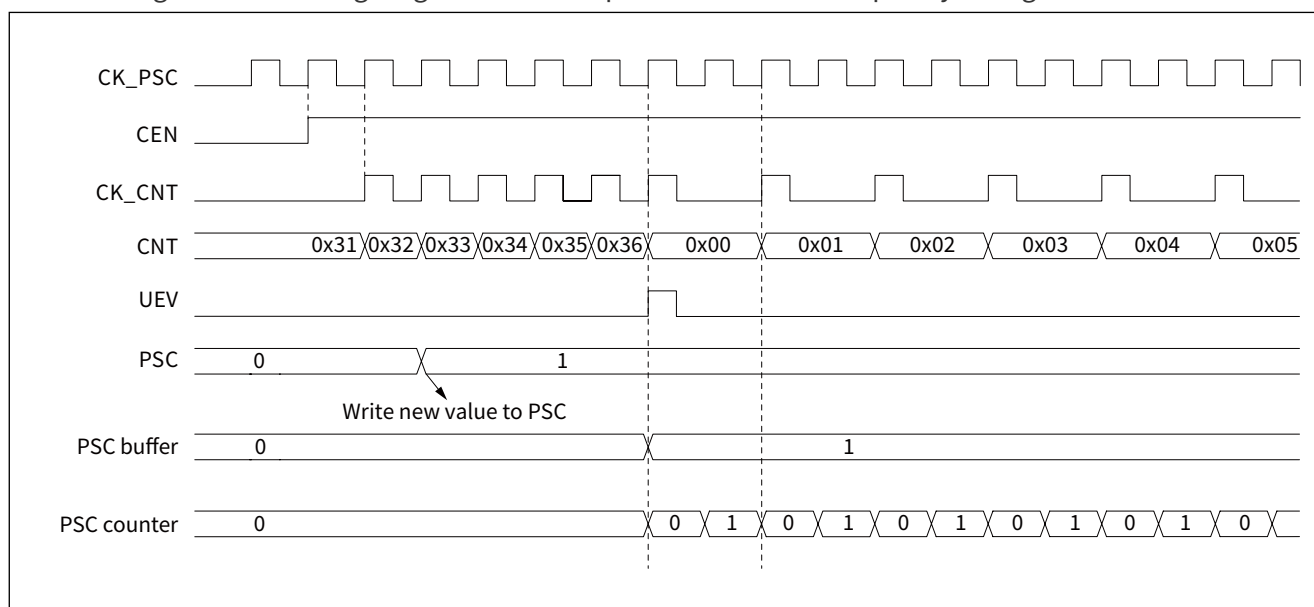
14.3.1.2 Prescaler

The prescaler divides the CK_PSC clock to obtain the count clock CK_CNT to drive the counter count. The divider coefficient is set through the ATIM_PSC register, which supports 1, 2, 3, 4, ..., 65536 divider.

The ATIM_PSC register is buffered and can be modified during runtime, and the new prescaler value will take effect at the next update event.

The following figure shows the timing diagram when the division frequency of the prescaler is changed from 1 to 2 during operation.

Figure 14-2 Timing diagram when the prescaler division frequency changes from 1 to 2



14.3.1.3 Counter and count mode

The counter can be set to count up (edge-aligned mode), count down (edge-aligned mode), or count up/down (center-aligned mode). Specifically, it is configured through the CMS and DIR bit fields of the control register ATIM_CR1, as shown in the following table:

Table 14-1 Counting mode

ATIM_CR1.CMS	ATIM_CR1.DIR	Counting mode
00	0	Count up (edge-aligned mode)
	1	Count down (edge-aligned mode)
01	-	Center-aligned mode 1
10	-	Center-aligned mode 2
11	-	Center-aligned mode 3

When the CEN bit field of the ATIM_CR1 register is set to 1, the counter starts counting in the set mode, noting that the counter will start counting one clock cycle after the CEN bit is set to 1.

Count up mode

In count up mode, the counter starts counting up from 0 to the reload value ARR, and then starts counting up again from 0, generating an overflow event on the counter and generating an overflow signal OV (the OV signal is cleared automatically).

If a repeat counter is used, an update event UEV will be generated when the number of overflows reaches the repeat value plus 1 (REP+1) (the UEV signal will be cleared automatically), otherwise an update event UEV will be generated each time the counter overflows, and the counter update interrupt flag bit, ATIM_ISR.UIF, is set by hardware, so that if an interrupt is permitted (setting ATIM_IER.UIE to 1), an interrupt request will be generated, the flag bit is cleared by setting ATIM_ICR.UIF to 0.

The following is an example of counter operation at different clock frequencies, where ATIM_ARR = 0x36, ATIM_RCR = 0x00:

Figure 14-3 Count up, internal clock frequency division factor is 1

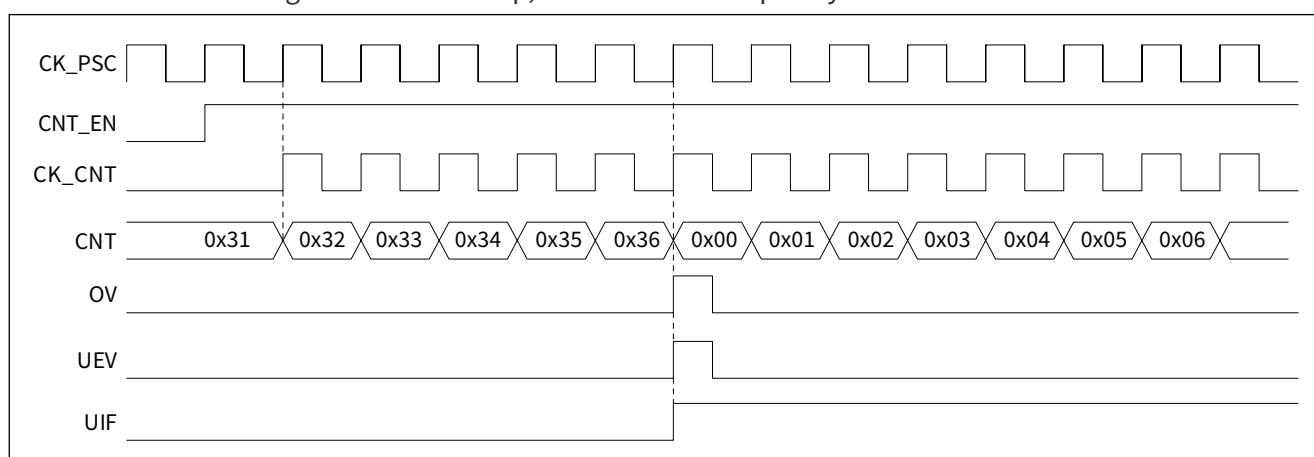


Figure 14-4 Count up, internal clock frequency division factor is 2

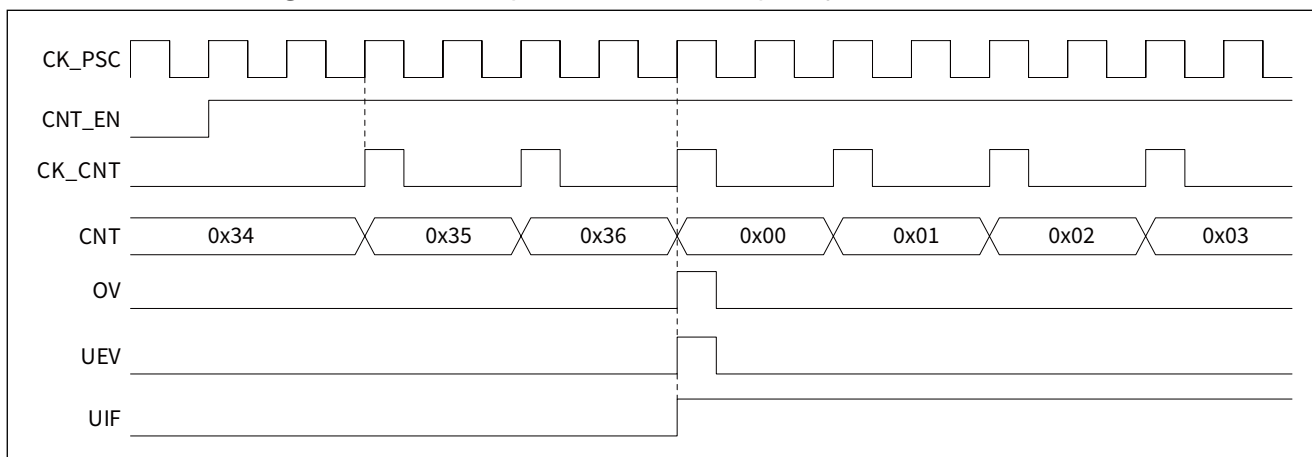


Figure 14-5 Count up, internal clock frequency division factor is 4

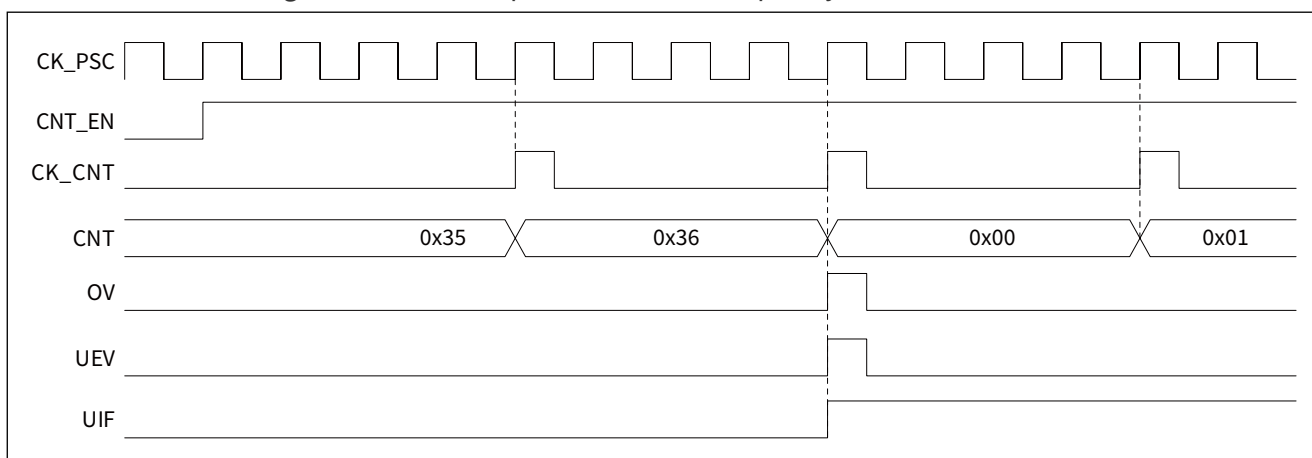


Figure 14-6 Count up, internal clock frequency division factor is N

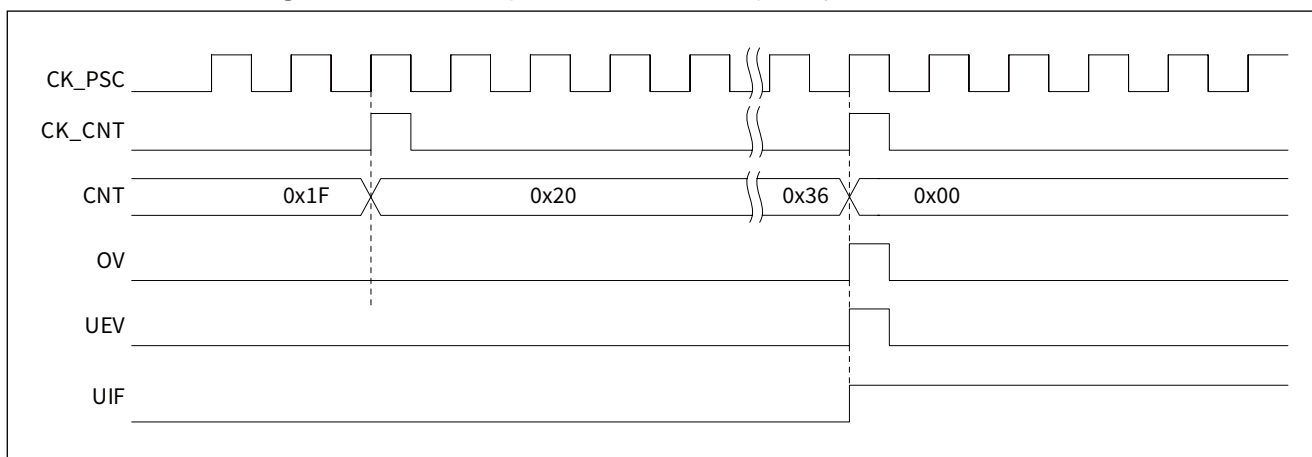


Figure 14-7 Count up, update event when reloading cache is disabled (ATIM_CR1.ARPE=0)

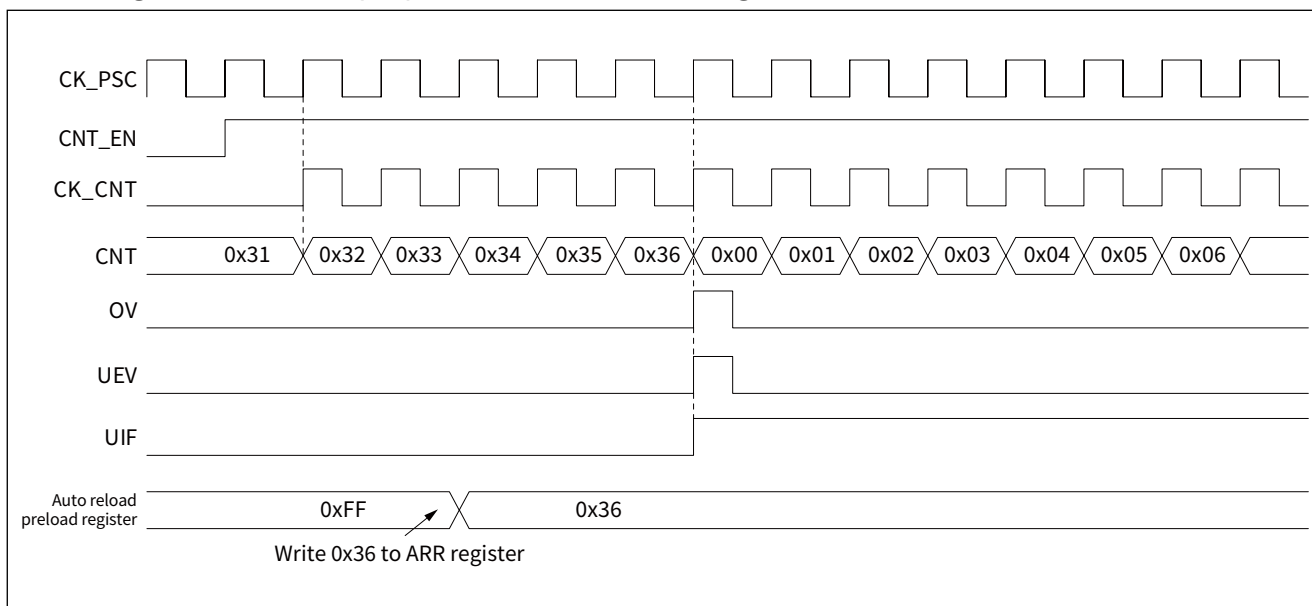
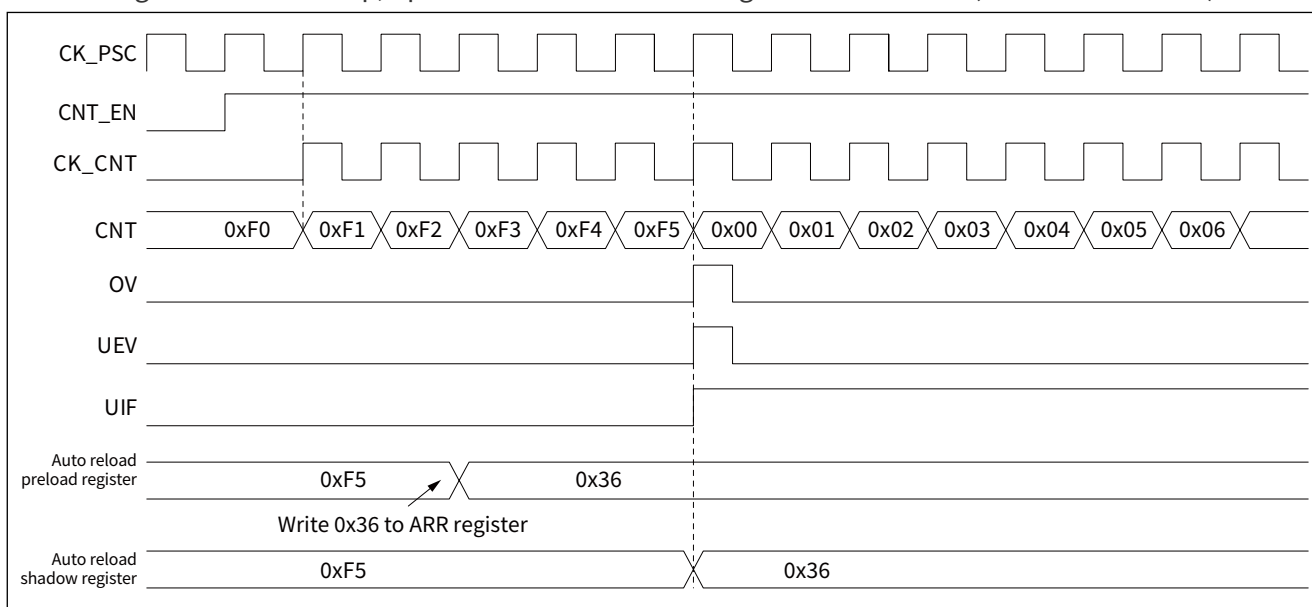


Figure 14-8 Count up, update event when reloading cache is enabled (ATIM_CR1.ARPE=1)



Count down mode

The counter operates in count down mode when the DIR bit of control register ATIM_CR1 is set to 1 in edgealigned mode.

In count down mode, the hardware automatically loads the ARR value into the counter CNT, the counter starts counting down to 0, and then reloads the ARR value to count down while generating a counter underflow event and generating the underflow signal UND (the UND signal is cleared automatically).

If a repeat counter is used, an update event UEV will be generated when the number of overflows reaches the repeat value plus 1 (REP+1) (the UEV signal will be cleared automatically), otherwise an update event UEV will be generated each time the counter underflows, and the counter update interrupt flag bit, ATIM_ISR.UIF, is set by hardware, so that if an interrupt is permitted (setting ATIM_IER.UIE to 1), an interrupt request will be generated, the flag bit is cleared by setting ATIM_ICR.UIF to 0.

The following is an example of counter operation at different clock frequencies, where ATIM_ARR = 0x36, ATIM_RCR = 0x00.



Figure 14-9 Count down, internal clock frequency division factor is 1

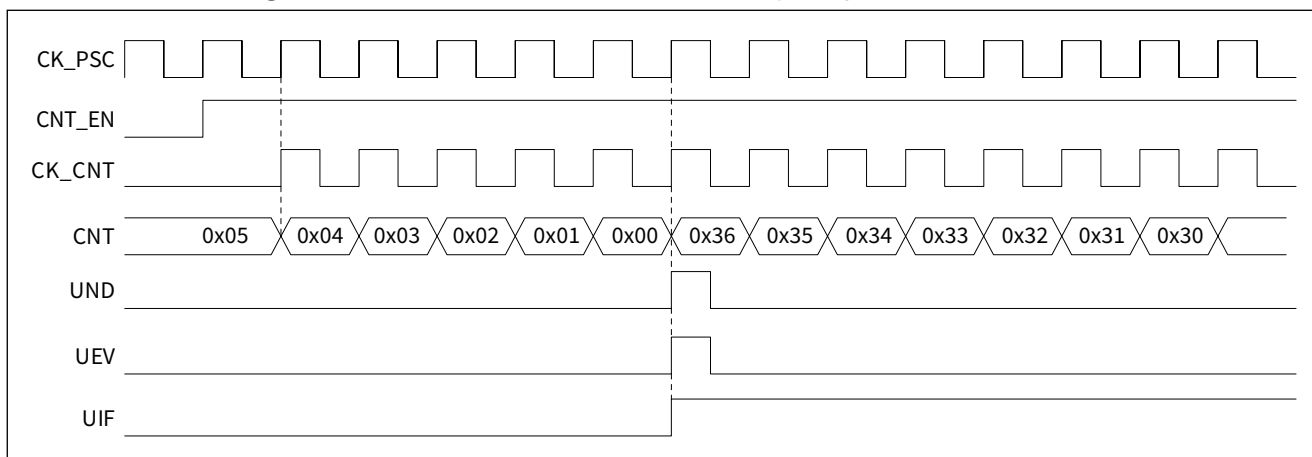


Figure 14-10 Count down, internal clock frequency division factor is 2

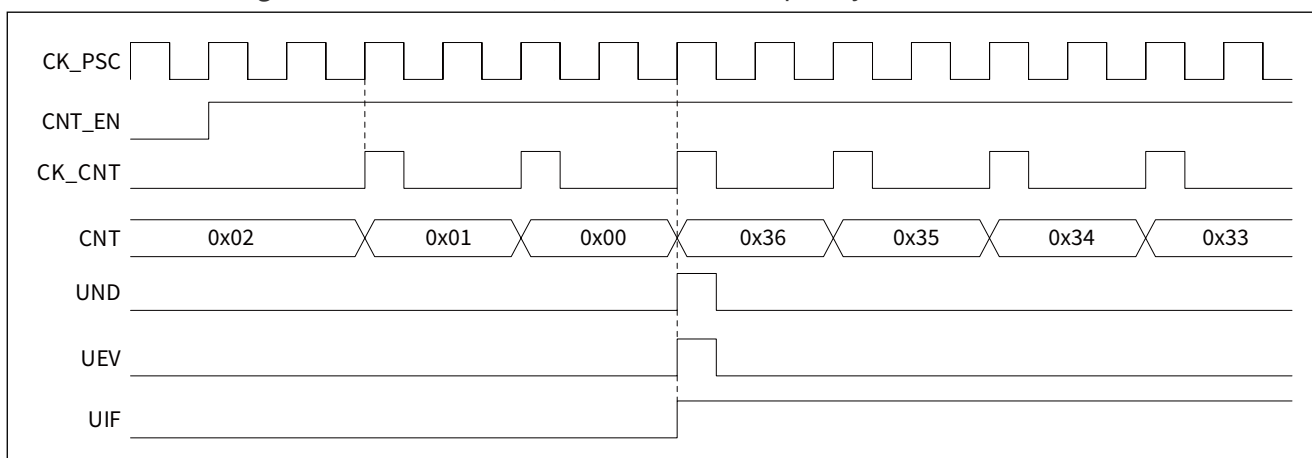


Figure 14-11 Count down, internal clock frequency division factor is 4

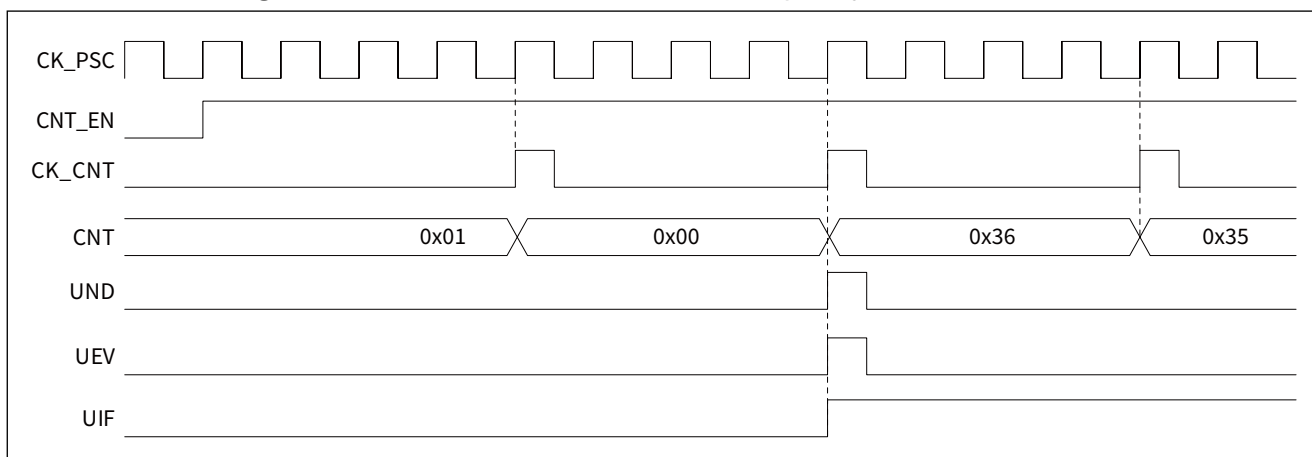
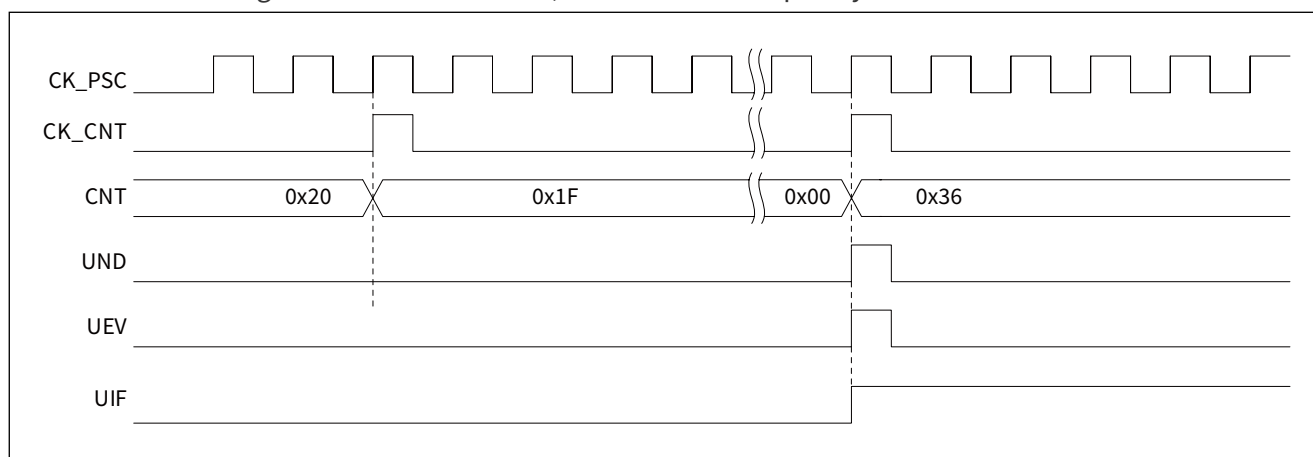


Figure 14-12 Count down, internal clock frequency division factor is N



Center-aligned mode (count up/down)

In center-aligned mode, the counter counting up from 0 to ARR-1, generates a counter overflow event, then starts counting down from the ARR value to 1 and generates a counter underflow event, after which the counting starts counting up again from 0.

The center-aligned modes include center-aligned mode 1, center-aligned mode 2, and center-aligned mode 3. The three modes are identical in counting, except for the timing of the output compare interrupt flag, for details, please refer to the description of the CMS bit field of [14.9.1 ATIM_CR1 controls register 1](#).

In center-aligned mode, the DIR bit of control register ATIM_CR1 cannot be written by software but can be read, and DIR is updated by hardware and indicates the current count direction. When center-aligned mode is activated, the counter will count up or down based on the current DIR bit field value. It is not possible to modify the DIR and CMS bit fields at the same time through software.

It is not recommended to perform a write operation to the counter while running center-aligned mode, as unexpected results will occur. In particular:

- If a value greater than the auto-reload value ARR (CNT>ARR) is written to the counter CNT, the direction will not be updated. That is, if the counter was previously counting up, it continues to count up.
- If a value of 0 or ARR is written to the counter CNT, the count direction is updated, but no update event UEV is generated.

The most secure way to use the center-aligned mode is to generate the update event UEV by software before starting the counter (set ATIM_EGR.UG to 1) and do not perform a write operation to the counter while it is running.

The following are examples of counter operation at different clock frequencies, without the use of a repeat counter:

Figure 14-13 Center-aligned mode, internal clock frequency division factor is 1, ARR=0x06

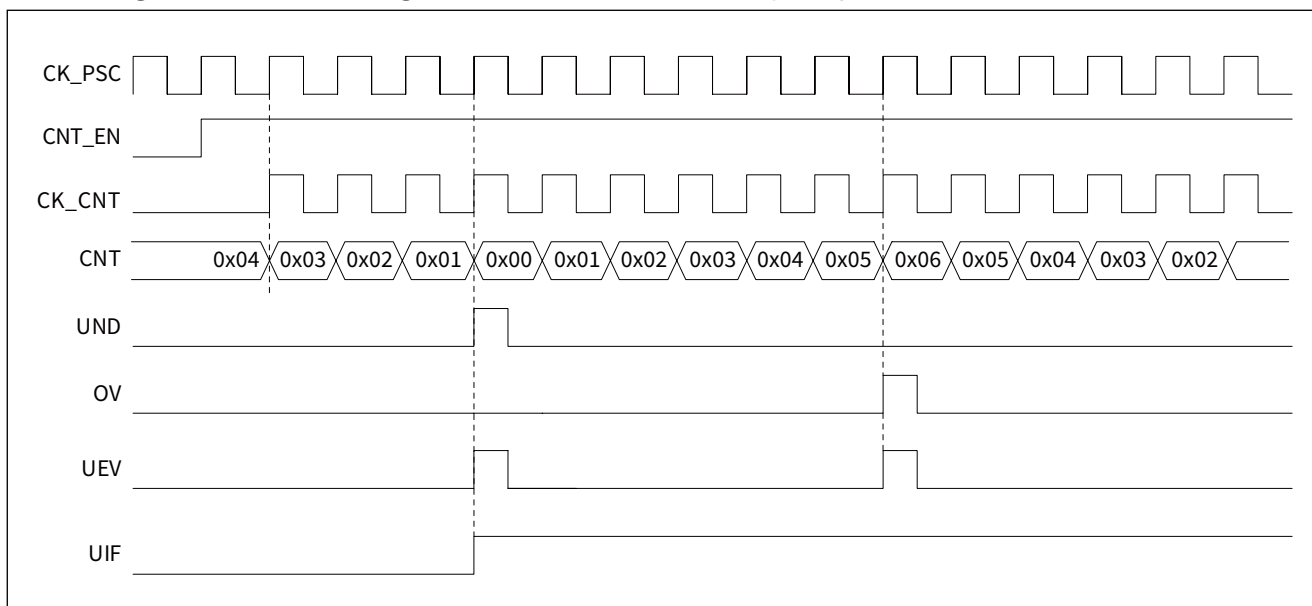


Figure 14-14 Center-aligned mode, internal clock frequency division factor is 2

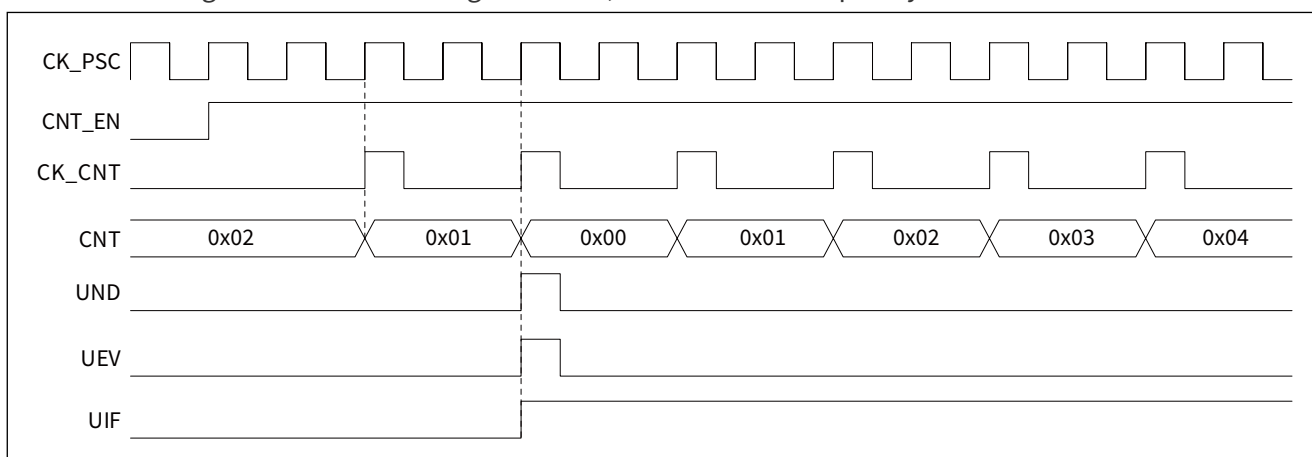


Figure 14-15 Center-aligned mode, internal clock frequency division factor is 4, ARR=0x36

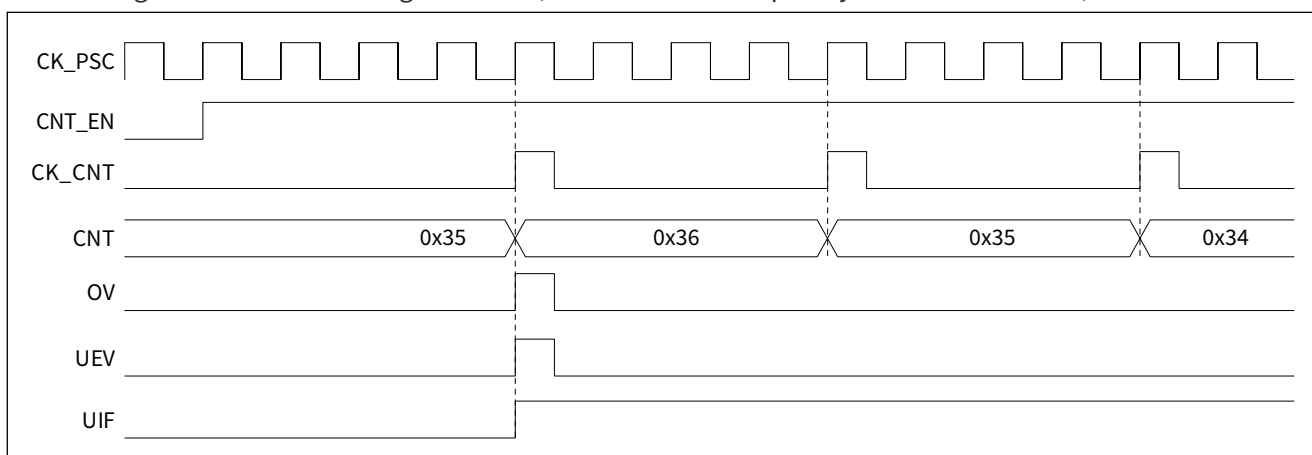


Figure 14-16 Center-aligned mode, internal clock frequency division factor is N

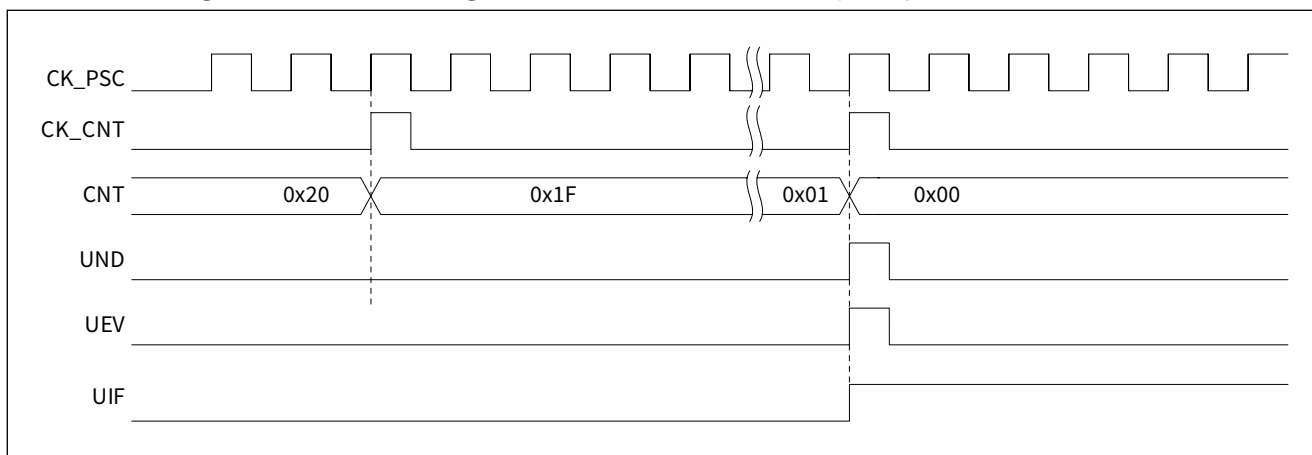


Figure 14-17 Center-aligned mode, update event when reloading cache is disabled (ATIM_CR1.ARPE=0)

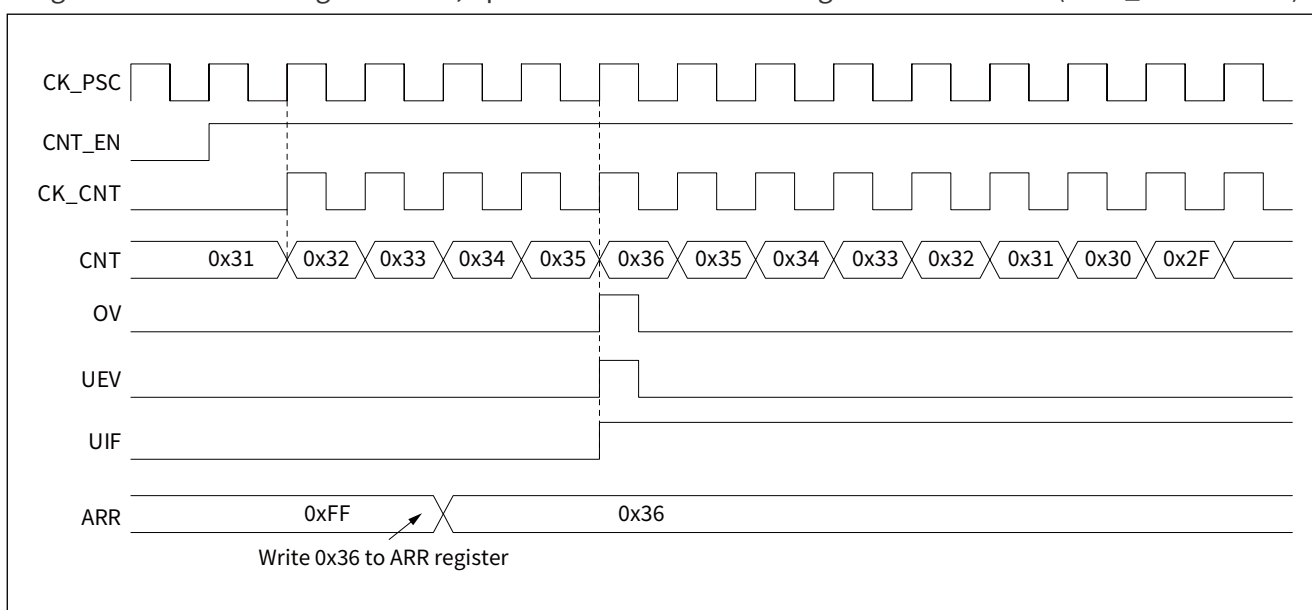
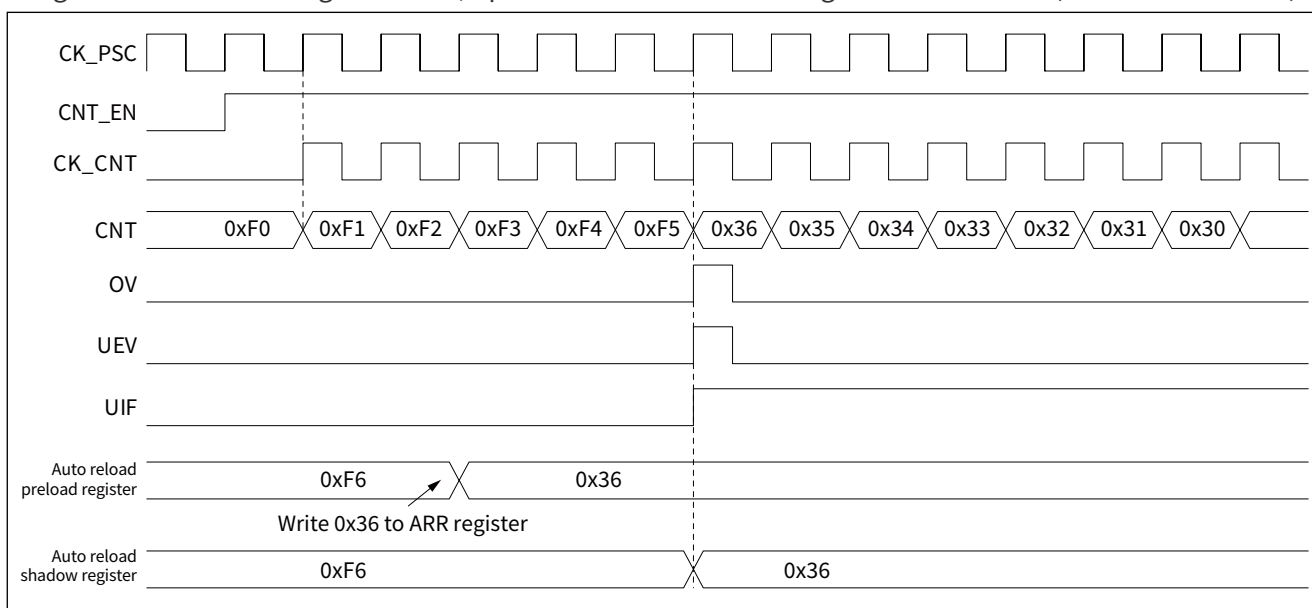


Figure 14-18 Center-aligned mode, update event when reloading cache is enabled (ATIM_CR1.ARPE=1)



14.3.1.4 Reload register

The auto reload register ATIM_ARR has a cache function that is enabled or disabled by controlling the ARPE bit field of register ATIM_CR1.

When the counter is stopped or the cache function is disabled, updating the reload register ARR will immediately update its shadow register. When the timer is running and the cache function is active, modifying the reload register ARR will not update the shadow register immediately, but will only update the value of the reload register ARR into the shadow register when an update event UEV is generated.



14.3.1.5 Repeat counter

Starting ATIM automatically loads the REP value of the repeat counter register ATIM_RCR into the repeat counter and automatically decrements it by 1 when the main counter overflows.

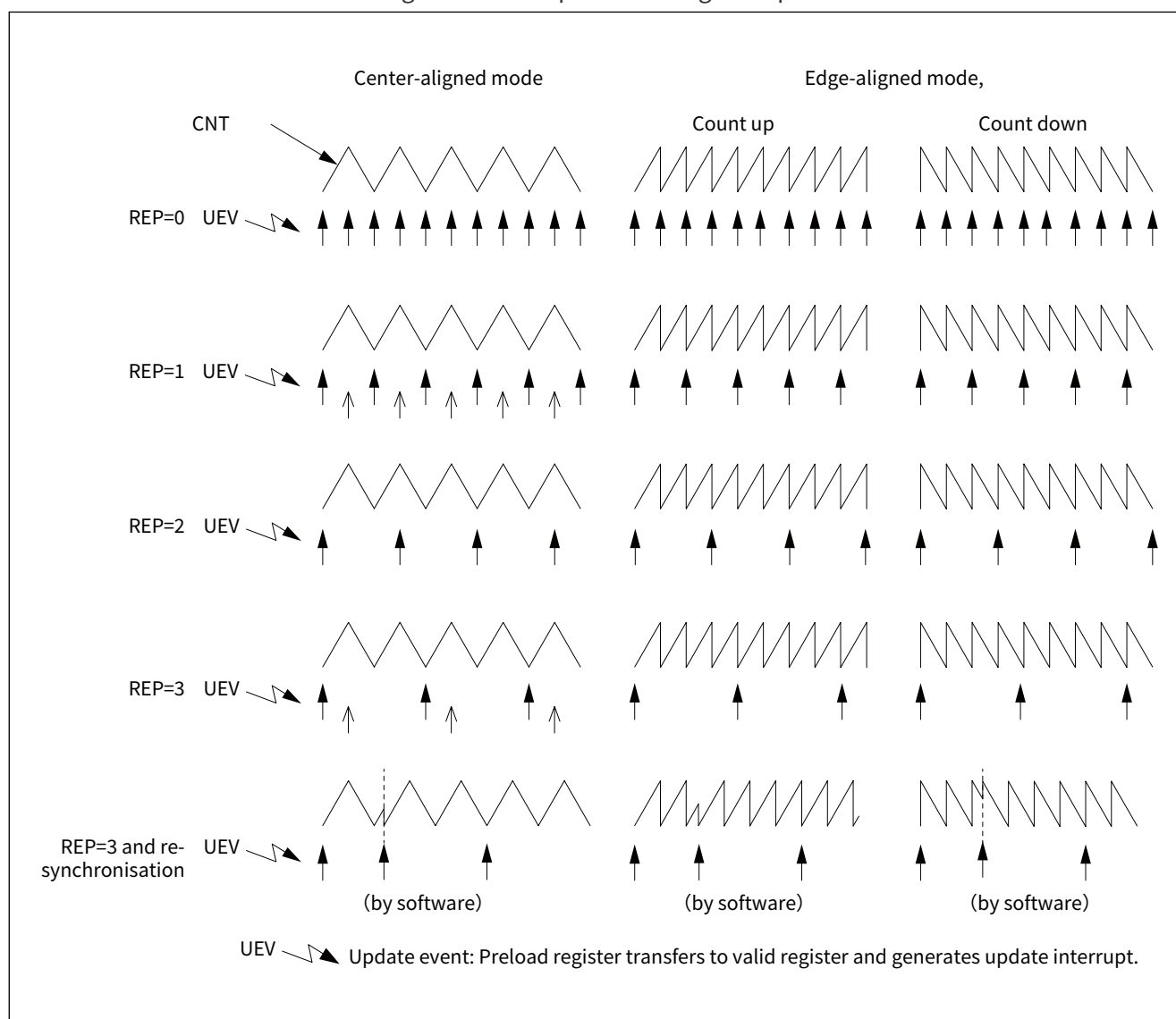
When using the repeat counter function (REP setting for ATIM_RCR is not 0), overflow of the master counter triggers an update event UEV only if the repeat counter is decremented to 0.

When the UG is updated by software or reset by slave mode, the update event UEV is generated immediately, regardless of the current repeat counter value, and, the content of REP in the ATIM_RCR register will be loaded into the repeat counter immediately.

In center-aligned mode, if the repeat value REP is odd, an update event occurs on overflow or underflow depending on when the ATIM_RCR register is written and when the start counter is started: update event UEV occurs on overflow if the RCR register is written before the start counter; update event UEV occurs on underflow if the RCR register is written after the start counter.

The following figure shows an example of generating update event UEV in different counting modes with different repeat count values:

Figure 14-19 Repeat counting example



14.3.1.6 Update event UEV

Allows updates to be disabled or enabled by the UDIS bit field of the control register ATIM_CR1.

Enable UEV

Set the UDIS bit field to 0 to enable UEV. At this time, update request sources can be selected according to the URS bit field, as shown in the following table:

Table 14-2 Update source settings

ATIM_CR1.URS	Update source
0	The counter overflow, underflow; UG set; Update events generated from the slave mode controller.
1	The counter overflow, underflow.

When an update event occurs, the following actions occur:

1. Reinitialise the counter:
Count down (edge-aligned mode) : Reloads the auto-reload value ATIM_ARR;
Center-aligned mode or count up (edge-aligned mode): the counter is cleared.
2. If the repeat counter is enabled, the repeat counter is reloaded with the contents of the ATIM_RCR register.
3. If the cache register function is used, the corresponding preloaded register is updated to its shadow register.
4. The prescaler counter is cleared and the new prescaler value in ATIM_PSC takes effect.
5. The event update interrupt flag bit ATIM_ISR.UIF is set by hardware.

When an update event UEV occurs, the event update interrupt flag bit ATIM_ISR.UIF is set by hardware. If the interrupt is allowed (set ATIM_IER.UIE to 1), an update interrupt request is generated, and setting ATIM_ICR.UIF to 0 clears the flag bit.

Caution:

If a repeat counter is used (ATIM_RCR.REP is not 0), an update event UEV is generated only if the count reaches the repeat count (ATIM_RCR.REP reaches 0).

Disable UEV

Set the UDIS bit field to 1 to disable UEV and no more update events are generated.

If the cache register function is used, the value of each corresponding shadow register remains unchanged. However, if the UG bit is set to 1 or a hardware reset is received from the slave mode controller, the counters of the counter and prescaler are reinitialised.



14.3.1.7 Single pulse mode

The counter can work in single count or continuous count mode and is selected by the OPM bit field of the control register ATIM_CR1.

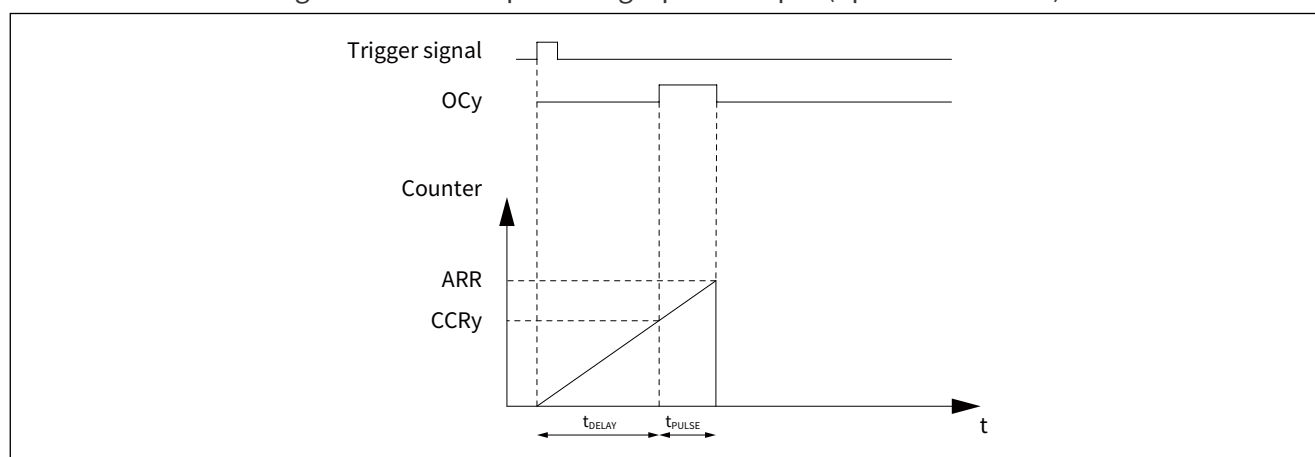
Single count mode

Set ATIM_CR1.OPM to 1 to enable the timer to work in single count mode.

After starting the counter, the counter CNT counts driven by the CK_CNT clock, and the counter stops counting when the next update event occurs while ATIM_CR1.CEN is automatically cleared by hardware.

The counter can be activated by the slave mode controller, together with the output comparison mode of the timer, so that the counter is activated by a trigger signal and generates a single pulse with a programmable pulse width after a programmable delay. As shown in the figure below, the timer starts counting when it detects a valid edge of the trigger signal, and after a delay of t_{DELAY} , it generates a positive pulse of width t_{PULSE} on the compare output port. Where t_{DELAY} is determined by the difference between the initial value of the count register ATIM_CNT and the capture/compare register ATIM_CCRy, and t_{PULSE} is determined by the difference between the auto-reload register ATIM_ARR and the capture/compare register ATIM_CCRy.

Figure 14-20 Example of single pulse output (up-count method)



Sampling the trigger input to activate the OC output requires multiple clock cycles, and the OCyFE bit field of the comparison mode register ATIM_CCMRxCMP can be set to 1 to shorten the OC output delay time.

Continuous count mode

Set ATIM_CR1.OPM to 0 so that the timer works in continuous count mode and the counter does not stop counting when an update event occurs.

14.3.1.8 External trigger input channel

The external trigger input signal ETR can be used as a trigger input (TRGI) for the slave mode controller or as a count clock for the counter, see [14.3.2 Operating modes](#). The ETR signal can be input controlled, including polarity selection and edge detection, prescaling and filtering.

Polarity selection and edge detection

The ETP bit field of the slave mode control register ATIM_SMCR is used to select the trigger polarity of the ETR input signal. When ETP is set to 0, ETR is not inverted and is valid at high level or rising edge; when ETP is set to 1, ETR is inverted and is valid at low level or falling edge.

Prescaler

The frequency of external trigger signal ETRP shall not exceed 1/4 of PCLK frequency. when the frequency of ETRP is high, user should divide the external signal by appropriate ETSP prescaler setting to reduce the ETRP frequency, and the dividing coefficient can be set to 1, 2, 4, 8.

Filter

The filter uses digital filtering to sample the input signal at a certain frequency, the signal is valid when it is continuously sampled to N same levels, otherwise the signal is invalid, so as to filter out the high-frequency clutter signal.

The filter of the ETR input signal is configured by the ETF bit field of the slave mode control register ATIM_SMCR, which can set the sampling frequency and the number of sampling points. The sampling frequency f_{SAMPLING} is provided by the clock after f_{DTS} dividing. f_{DTS} is the frequency obtained after f_{PCLK} dividing. The dividing factor is configured by the CKD bit field of the ATIM_CR1 register, which can be set to 1, 2, or 4 divisions.

14.3.1.9 Input capture channel

ATIM supports 6 input capture channels TI1/2/3/4/5/6 which can be used as capture commands. Among them, TI1 and TI2 can also be used as trigger input (TRGI) and encoder interface input of slave mode controller, see [14.3.2 Operating modes](#).

Supports input control of Tly signals, including filtering, edge detection and prescaling.

Filter

In the filtering stage, the corresponding Tly input signal is sampled at a certain frequency, and the filtered signal TlyF is generated.

The filter for the Tly input signal is configured by the ICyF bit field of the capture mode register ATIM_CCMRxCAP, which can be set for the sampling frequency and the number of sampling points. The sampling frequency f_{SAMPLING} is provided by the clock after f_{PCLK} or f_{DTS} dividing, f_{DTS} is the frequency obtained after f_{PCLK} dividing, and the dividing factor is configured by the CKD bit field of ATIM_CR1 register, and it can be set to 1, 2, or 4 divisions.

Edge detection

The edge detector with polarity selection generates a TlxFPx signal, for specific valid polarity please see the description of CCyNP and CCyP bit fields in [14.9.14 ATIM_CCER capture/compare enable register](#).

Prescaler

When the Tly signal is used for input capture, the capture channel signal ICy can be prescaled, which is controlled through the ICyPSC bit field of the capture mode register ATIM_CCMRxCAP, and supports 1, 2, 4, and 8 frequency division.

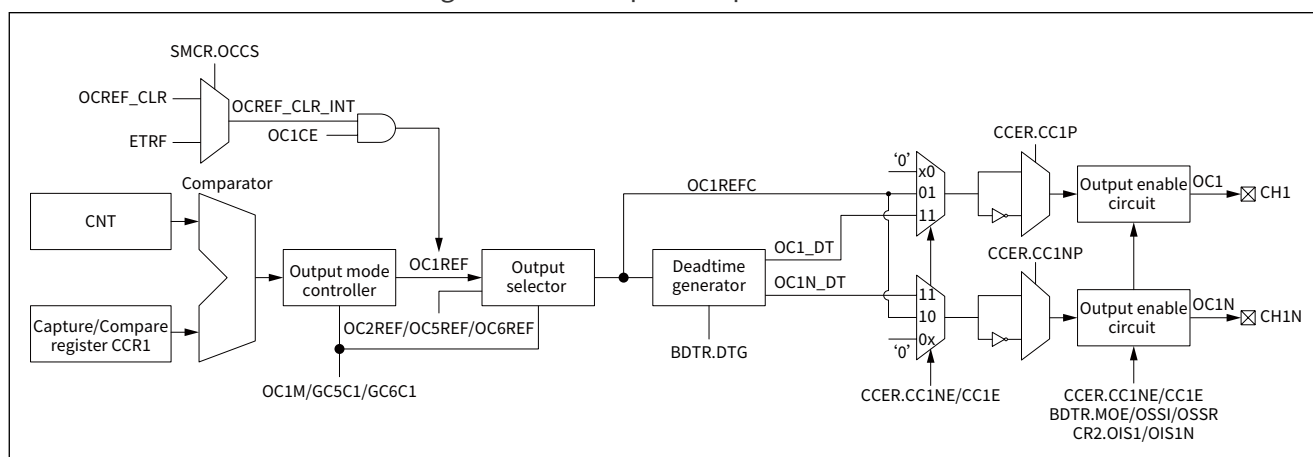


14.3.1.10 Output compare channel

ATIM supports 6 independent output compare channels CH1/2/3/4/5/6 (all channels support complementary output CHyN), each consisting of a capture/compare register (including a shadow register) and an output stage (comparator and output control).

The output control unit is used to control the waveform of the output port when comparing and matching, and can configure the output compare mode, output polarity selection and output enable. Taking compare channel 1 as an example, its block diagram is shown below:

Figure 14-21 Output compare channel 1



14.3.2 Operating modes

The ATIM supports multiple operating modes, which are configured by the SMS bit field of the slave mode control register ATIM_SMCR, as shown in the following table:

Table 14-3 ATIM operating modes

ATIM_SMCR.SMS	Slave mode selection
0000	Disable slave mode, use internal clock
0111	External clock mode 1
0100	Reset mode
0101	Gated mode
0110	Trigger mode
1000	Combined reset + Trigger mode
1001	Combined gated + Reset mode
1110	Quadrature encoder mode, x1 mode
1111	Quadrature encoder mode, x1 mode
0001	Quadrature encoder mode, x2 mode
0010	Quadrature encoder mode, x2 mode
0011	Quadrature encoder mode, x4 mode
1010	Encoder mode (clock + direction), x2 mode
1011	Encoder mode (clock + direction), x1 mode
1100	Encoding mode (with direction clock), x2 mode
1101	Encoding mode (with direction clock), x1 mode

Caution:

1. The slave mode select bits include ATIM_SMCR[16] and ATIM_SMCR[2:0] to be configured in combination.
2. Slave mode select bits SMS[3:0] support the preload function, please refer to the description of SMSPE and SMSPS bit field.

14.3.2.1 Internal clock mode

When the SMS bit field of the slave mode control register ATIM_SMCR is 0x0, the timer slave mode is disabled and the prescaler clock is provided directly by the internal clock PCLK. Setting ATIM_CR1.CEN to 1 will enable the counter to start counting.



14.3.2.2 External clock mode

External clock mode 1

When the SMS bit field of the slave mode control register ATIM_SMCR is 0x7, the counter clock is supplied by the rising edge of the selected trigger signal (TRGI). There are several trigger options for the TRGI signal, which are selected through the TS bit field of the ATIM_SMCR register, as shown in the table below:

Table 14-4 TRGI signal source

ATIM_SMCR.TS bit field value	TRGI signal source
00111	External trigger input (ETRF)
000xx/01xxx/10000	Internal trigger (ITR0~12)
00100	TI1 edge detector (TI1F_ED)
00101	Filtered timer input 1 (TI1FP1)
00110	Filtered timer input 2 (TI2FP2)

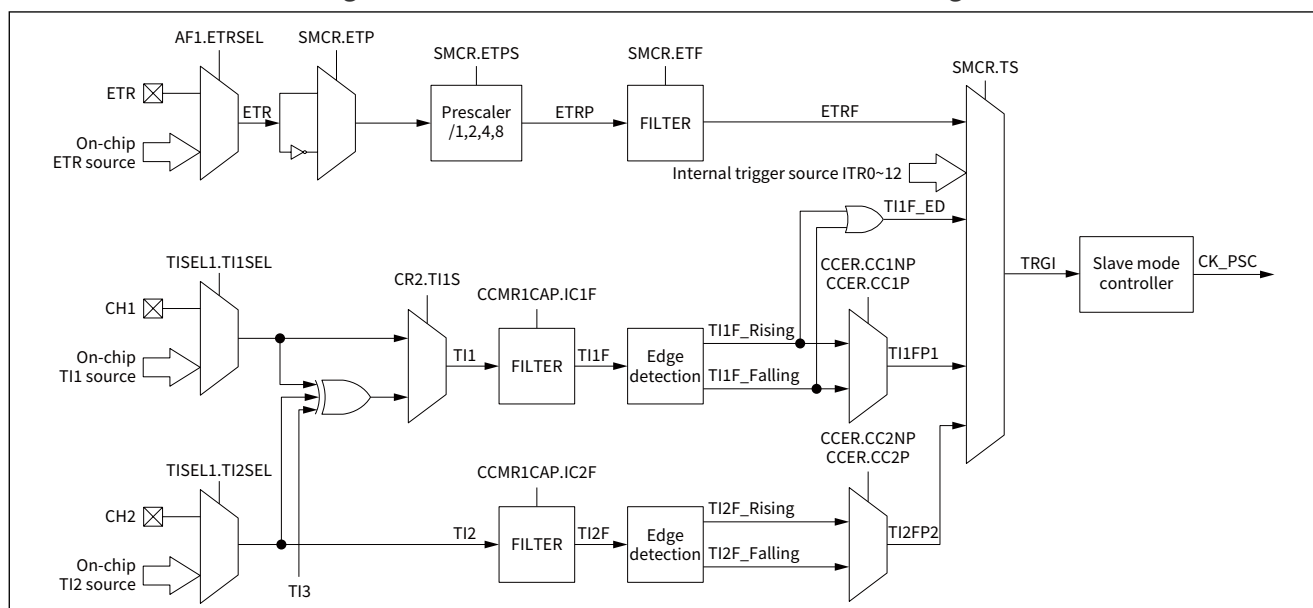
The ETR input signal source can be external ATIM_ETR/GTIM_ETR pins or other peripherals on-chip, please refer to [14.3.7 On-chip peripheral interconnect ETR](#). When ATIM_ETR is selected as the TRGI signal source, the external trigger polarity can be selected via ATIM_SMCR.ETP, the pre-scaler frequency can be set via ATIM_SMCR.ETPS, and the filtering control can be performed via ATIM_SMCR.ETF.

The internal trigger ITR sources are the trigger output signal TRGO of BTIM and GTIM, and the TXD/RXD signals of UART, please refer to [14.3.6 Timer cascade ITR](#).

Both TI1 and TI2 have filtering and edge detection functions. TI1 and TI2 are controlled by filtering through the IC1F and IC2F bit fields of the capture mode register ATIM_CCMR1CAP, respectively, and edge detection through the CC1P/CC1NP and CC2P/CC2NP bit fields of the capture/compare enable register ATIM_CCER, respectively.

The external clock mode 1 connection diagram is shown below:

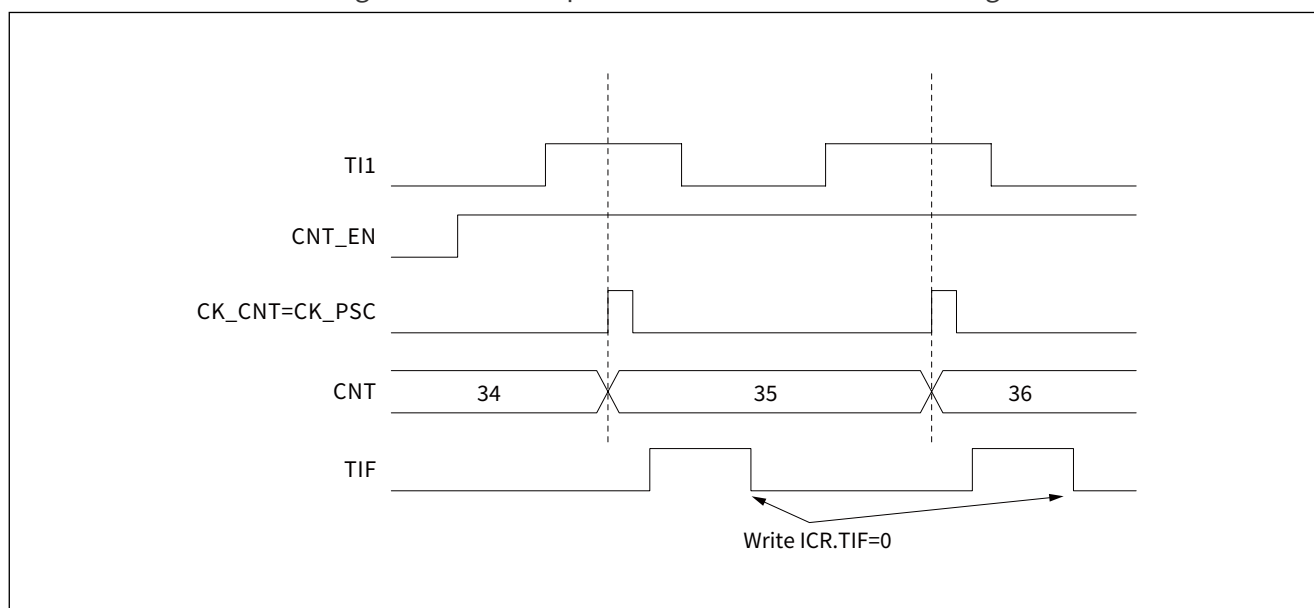
Figure 14-22 External clock mode 1 connection diagram



When a valid edge appears in TRGI, the ATIM_ISR.TIF flag will be set to 1, and the flag can be cleared by writing 0 to ATIM_ICR.TIF.

The following figure shows the external clock mode 1 timing diagram with TI1 rising edge configured. The delay between the rising edge of TI1 and the actual counter clock is due to the resynchronisation circuitry of the TI1 input.

Figure 14-23 Example of external clock mode 1 timing



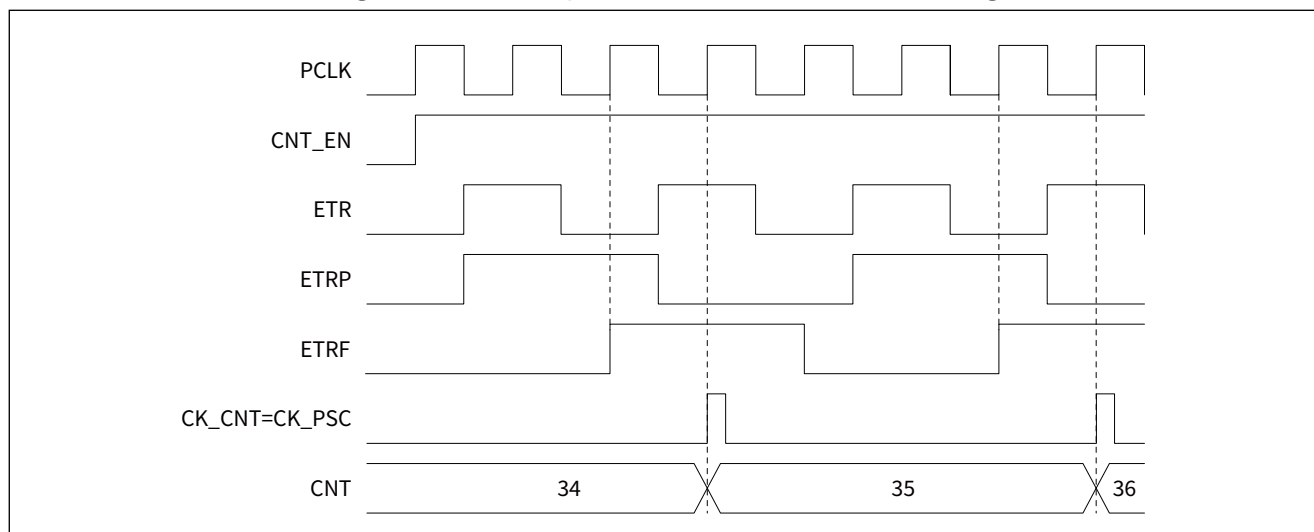
External clock mode 2

Set the ECE bit field of the slave mode control register ATIM_SMCR to 1 to enable external clock mode 2. In this mode, the counter clock is supplied by any valid edge of the ETRF signal, which has the same effect as selecting external clock mode 1 and connecting the TRGI to the ETRF (SMS=0111 and TS=00111), refer to [Figure 14-22 External clock mode 1 connection diagram](#).

External clock mode 2 can be used in conjunction with the following slave modes: reset mode, gated mode, and trigger mode, where the TRGI in slave mode must not be connected to the ETRF (i.e., the TS bit field cannot be set to 0x7). If both external clock mode 1 and external clock mode 2 are enabled, the external clock input is ETRF. External clock mode 2 cannot be used in conjunction with encoder mode.

The following figure shows a timing example of external clock mode 2 when ETR is set to 2 divisions, the delay between the rising edge of ETR and the actual counter clock is caused by the resynchronisation circuitry of the ETRP signal, so the maximum frequency that can be correctly captured by the counter is up to 1/4 of the internal clock PCLK frequency, and the user should divide the external signals appropriately when the ETRP signal becomes faster.

Figure 14-24 Example of external clock mode 2 timing



14.3.2.3 Reset mode

When the SMS bit field of the slave mode control register ATIM_SMCR is 0x4, the ATIM is configured in reset mode and the reset of the counter is controlled by the TRGI signal.

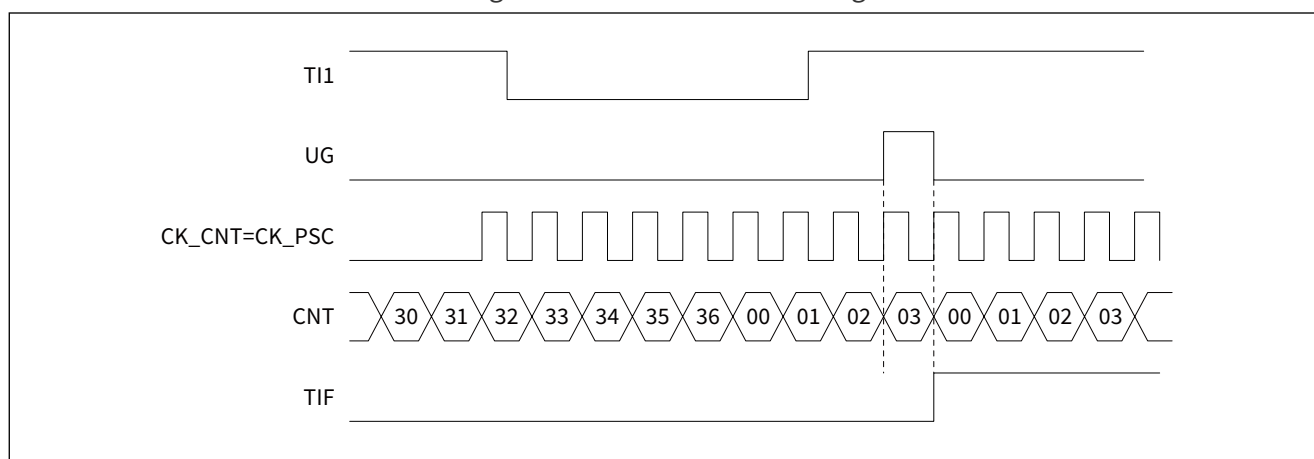
The source of the TRGI signal is controlled by the TS bit field of the slave mode control register ATIM_SMCR. See [Table 14-4 TRGI signal source](#).

When a valid trigger input (TRGI) is detected, the following effects occur:

- Reinitialise the counters and prescaler counters.
- If the URS bit field of control register ATIM_CR1 is 0, an update event UEV is generated, and the event update interrupt flag ATIM_ISR.UIF is set to 1. An interrupt request can be generated.
- The trigger interrupt flag ATIM_ISR.TIF is set to 1, which can generate an interrupt request.

The following figure shows an example of the reset mode timing diagram. Set ATIM_CR1.CEN to 1 to enable the counter, the counter counts normally according to the counting clock CK_CNT, and when a rising edge of TI1 occurs, the counter is cleared to 0 and starts counting from 0 again, while the ATIM_ISR.TIF flag is set to 1. The delay between the rising edge of TI1 and the actual counter reset is caused by the resynchronisation circuitry of the TI1 input.

Figure 14-25 Reset mode timing



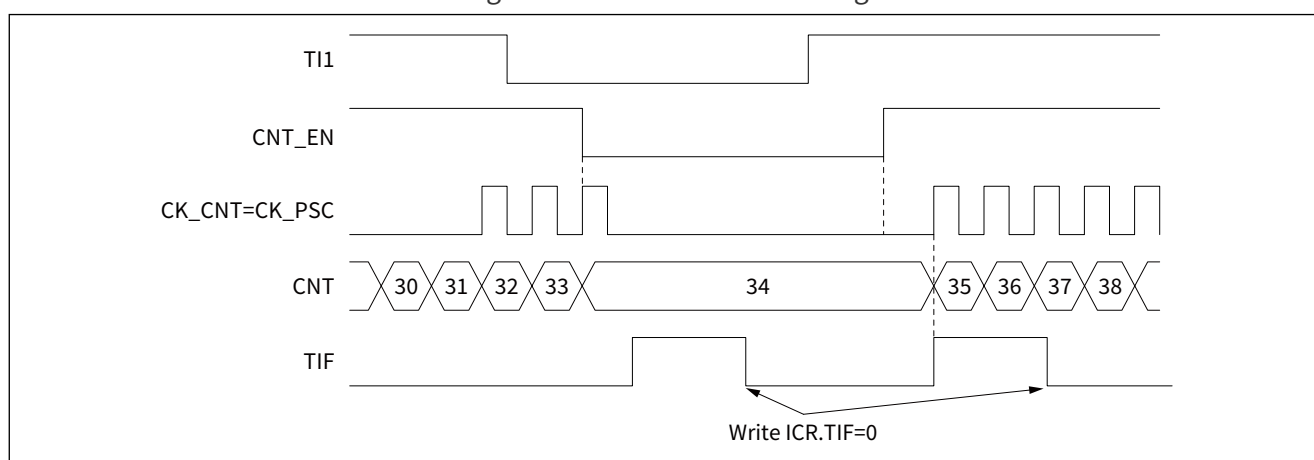
14.3.2.4 Gated mode

When the SMS bit field of the slave mode control register ATIM_SMCR is 0x5, the ATIM is configured for gated mode. In this mode, the counter is started counting when the trigger input (TRGI) is high and ATIM_CR1.CEN is 1. The counter stops counting immediately (but is not reset) when the trigger input (TRGI) is low or ATIM_CR1.CEN is 0. Both the start and stop of the counter are controlled.

The source of TRGI signals is controlled by the TS bit field of the slave mode control register ATIM_SMCR. For details, see [Table 14-4 TRGI signal source](#), however it should be noted that TI1F_ED cannot be selected as a trigger input in gated mode.

The following figure shows an example of gated mode timing diagram. When TI1 is high, the counter starts counting; when TI1 is low, the counter pauses counting. The ATIM_ISR.TIF flag bit is set to 1 when the counter starts and stops. The delay between the edge of TI1 and the actual counter enable signal CNT_EN is caused by the resynchronisation circuitry of the TI1 input.

Figure 14-26 Gated mode timing



14.3.2.5 Trigger mode

When the SMS bit field of the slave mode control register ATIM_SMCR is 0x6, the ATIM is configured for trigger mode. In this mode, when ATIM_CR1.CEN is set to 1 or when the rising edge of the trigger signal TRGI appears, the start counter is triggered (but not reset).

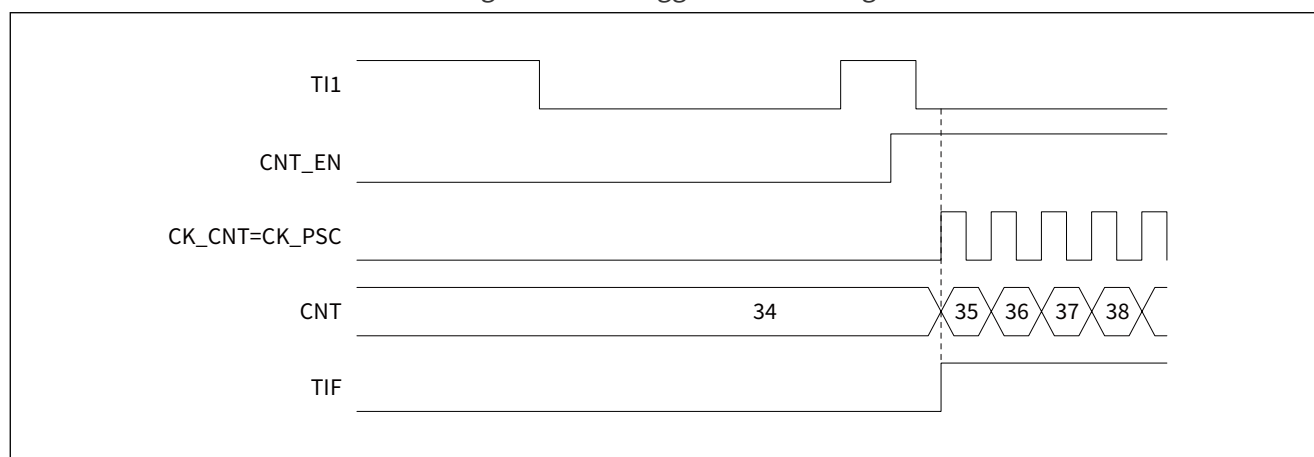
The source of the TRGI signal is controlled by the TS bit field of the slave mode control register ATIM_SMCR, see [Table 14-4 TRGI signal source](#).

When a valid trigger signal is detected, the following effects occur:

- ATIM_CR1.CEN is set by hardware.
- The trigger interrupt flag bit ATIM_ISR.TIF is set to 1, which can generate an interrupt request.
- The counter is activated and starts counting.

The following figure shows an example of a trigger mode timing diagram. When a rising edge of TI1 occurs, the counter starts counting while the ATIM_ISR.TIF flag bit is set to 1. The delay between the rising edge of TI1 and the actual counter startup is caused by the resynchronisation circuitry of the TI1 input.

Figure 14-27 Trigger mode timing



14.3.2.6 Combined reset + trigger mode

When the SMS bit field of the slave mode control register ATIM_SMCR is 0x8, the ATIM is configured for combined reset + trigger mode. In this mode, when a rising edge of the selected trigger input (TRGI) occurs, the counter will be reinitialised and the counter will be started, and the trigger interrupt flag ATIM_ISR.TIF will be set to 1. If the URS bit field of the control register ATIM_CR1 is 0, an update event UEV will be generated, and the event update interrupt flag ATIM_ISR.UIF will be set by hardware.

The source of the TRGI signal is controlled by the TS bit field of the slave mode control register ATIM_SMCR, see [Table 14-4 TRGI signal source](#).

14.3.2.7 Combined gated + reset mode

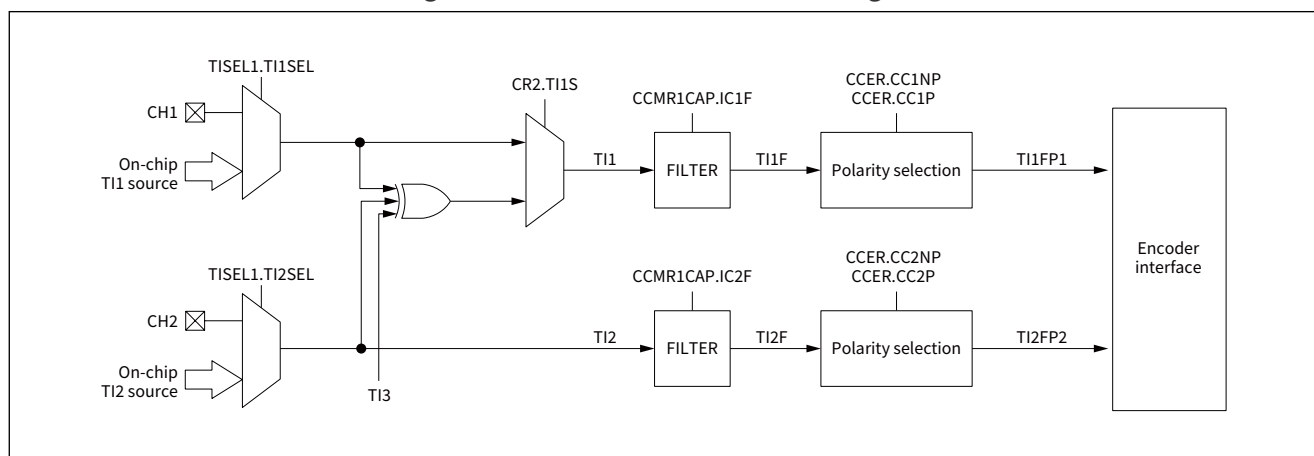
When the SMS bit field of the slave mode control register ATIM_SMCR is 0x9, the ATIM is configured for the combined gating + reset mode. In this mode, the start and stop of the counter are controlled. When the trigger input (TRGI) is high and ATIM_CR1.CEN is 1, the counter is started to count. When the trigger input (TRGI) is low, the counter immediately stops counting and is reset. When the counter is started and stopped, the ATIM_ISR.TIF flag bit is set to 1. When the counter is reset, ATIM_ISR.UIF flag bit is set to 1.

The source of the TRGI signal is controlled by the TS bit field of the slave mode control register ATIM_SMCR, see [Table 14-4 TRGI signal source](#).

14.3.2.8 Quadrature encoder mode

ATIM supports quadrature encoder mode for receiving and decoding quadrature encoder signals. In this mode, it allows a direct connection to an external quadrature encoder via the CH1 and CH2 pins, which enables the counter to count up or down automatically according to the hopping sequence of the input signals, and the counter value always indicates the position of the encoder. The functional block diagram is shown below:

Figure 14-28 Encoder mode block diagram



The CH1 and CH2 input signals have filtering and polarity selection functions, which are filtering controlled through the IC1F and IC2F bit fields of the ATIM_CCMR1CAP register, respectively, and the input polarity is selected through the CC1P and CC2P bit fields of the ATIM_CCER register, and the CC1NP and CC2NP bit fields must be kept clear.

Set ATIM_CR1.CEN to 1 to enable the counter, the counter will be driven by each valid hop of the signals TI1FP1 and TI2FP2 after the input signals of the channel CH1 and CH2 pins have been filtered and polarity selected, and according to the hopping sequence of the two input signals, the counting pulses and the direction signals are generated, refer to [Table 14-5 Relationship between counting direction and encoder signal](#). The encoder's current count direction flag ATIM_CR1.DIR is automatically set and cleared by hardware, and the DIR bit is counted whenever a signal transition occurs on any input (CH1 or CH2), regardless of whether the counter is counting only at the TI1FP1 or TI2FP2 edges, or at both TI1FP1 and TI2FP2.

ATIM supports several quadrature coded counting modes, which are set by the SMS bit field of the slave mode control register ATIM_SMCR. The relationship between the counting direction and the encoder signals in different modes is shown in the table below:

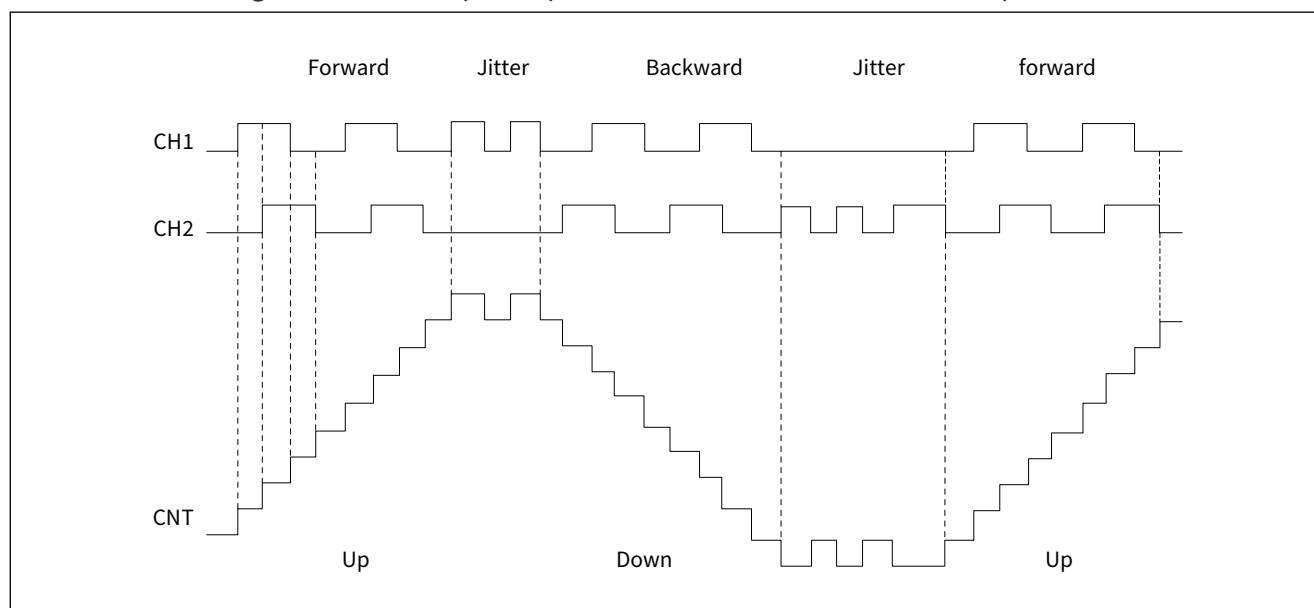
Table 14-5 Relationship between counting direction and encoder signal

Mode	SMS	Signal level		TI1FP1		TI2FP2	
		TI2FP2	TI1FP1	Rising	Falling	Rising	Falling
Counting on the TI1FP1 edge (x1 mode)	1110	High	-	Down count	Up count	No count	No count
		Low	-	No count	No count	No count	No count
Counting on the TI2FP2 edge (x1 mode)	1111	-	High	No count	No count	Up count	Down count
		-	Low	No count	No count	No count	No count
Counting on the TI1FP1 edge (x2 mode)	0001	High	-	Down count	Up count	No count	No count
		Low	-	Up count	Down count	No count	No count
Counting on the TI2FP2 edge (x2 mode)	0010	-	High	No count	No count	Up count	Down count
		-	Low	No count	No count	Down count	Up count
Counting on the TI1FP1 and TI2FP2 edge (x4 mode)	0011	High	High	Down count	Up count	Up count	Down count
		Low	Low	Up count	Down count	Down count	Up count

The third signal from the encoder output represents the mechanical zero point and can be connected to an external trigger input pin to trigger a counter reset.

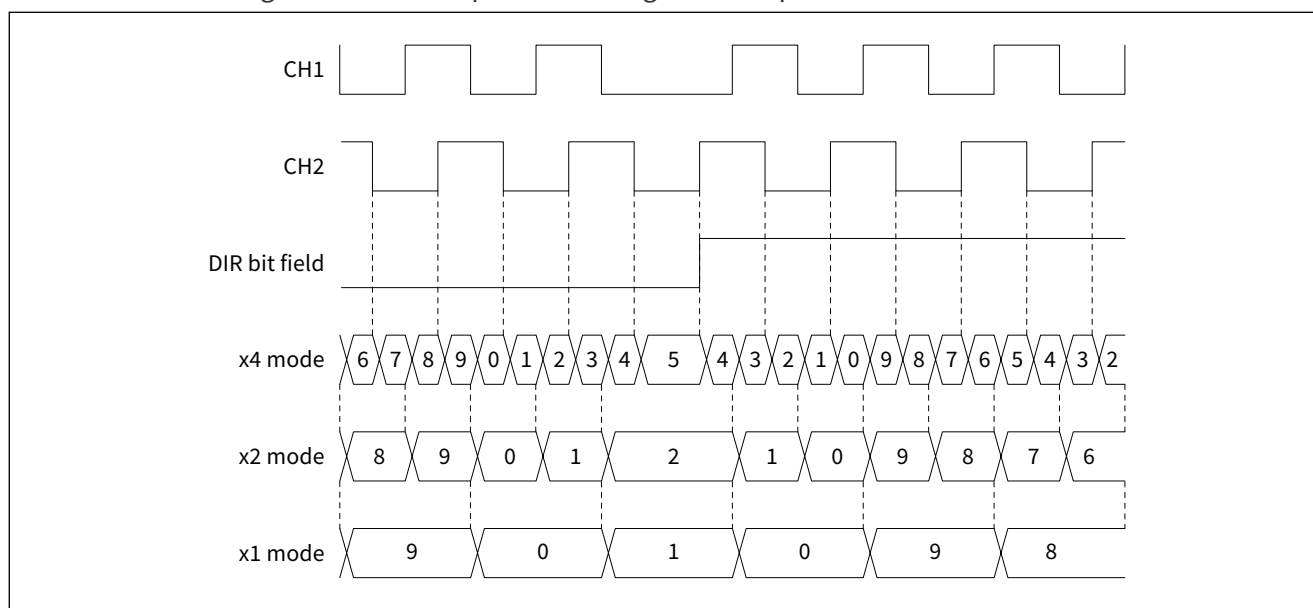
The following figure is an example of quadrature encoded counting mode-x4 mode operation, showing the generation of the counting signal and direction control. It also shows how input jitter is suppressed when a double edge is selected; jitter may be generated when the sensor is positioned close to a transition point.

Figure 14-29 Example of quadrature encoder mode-x4 mode operation



The following figure shows the timer count values during encoder inversion for various quadrature coded counting modes, with neither CH1 nor CH2 inputs inverted in the example.

Figure 14-30 Example of counting for each quadrature encoder mode



The encoder mode provides information about the current position of the sensor. The counter automatically counts up or down based on the pulses output from the encoder, and the current counter value indicates the current position of the encoder. Using another timer configured in capture mode, the rising or falling edge of the encoder signal can be captured and its timestamp recorded. By measuring the time interval between the two events, the corresponding dynamic information such as speed, acceleration and deceleration can be calculated; The encoder output indicating the mechanical zero position can be used for this purpose, and by measuring the time interval from the mechanical zero position to some other position, the change in position and velocity relative to the zero position can be calculated.

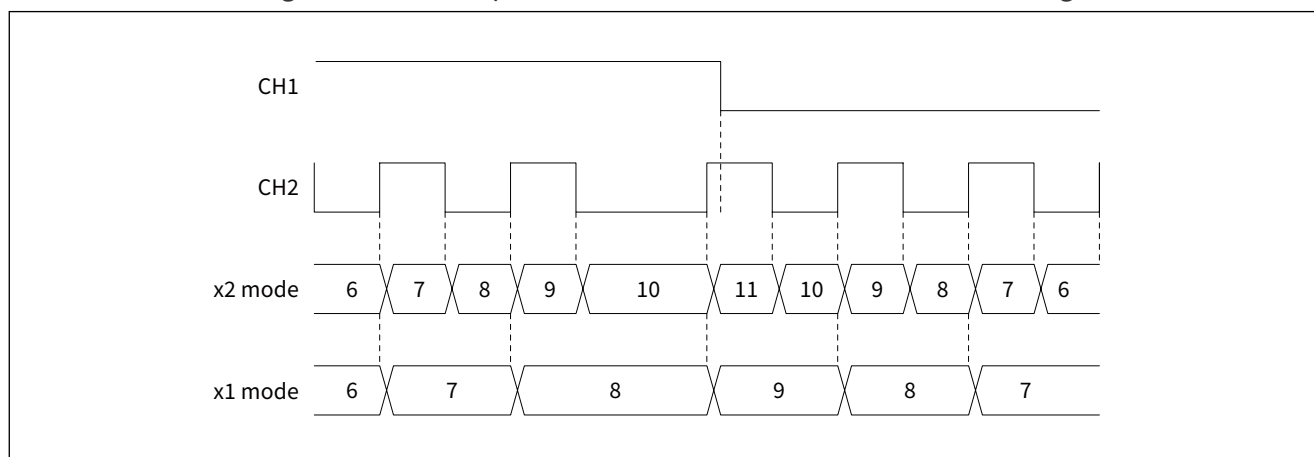
14.3.2.9 Clock + direction encoder mode

The timer also supports clock plus direction encoder mode, which requires configuring the SMS bit field of the slave mode control register ATIM_SMCR to be 0xA or 0xB, which correspond to x2 mode and x1 mode respectively. The functional block diagram can be referred to [Figure 14-28 Encoder mode block diagram](#), which allows filtering and polarity control of the input signal.

Encoders with direction usually have two output signals: a clock signal and a direction signal. Using the clock signal, the rotational speed and position change of the rotary encoder can be calculated, while the direction signal can be used to determine the direction of rotation. In x2 mode, the counter counts at the rising and falling edge of the clock signal; In x1 mode, the counter counts on a single clock edge based on the value of the CC2P bit field, where CC2P 0 corresponds to rising edge sensitivity and CC2P 1 corresponds to falling edge sensitivity. The polarity of the direction signal on the CH1 channel is set by the CC1P bit: CC1P is 0 corresponding to positive polarity (counts up when CH1 is high, counts down when CH1 is low) and CC1P is 1 corresponding to negative polarity (counts up when CH1 is low, counts down when CH1 is high).

The following figure shows an example of counter counting in clock plus direction encoder mode, with neither CH1 nor CH2 inputs inverted in the example:

Figure 14-31 Example of clock + direction encoder mode counting



14.3.2.10 Directional clock encoder mode

The timer also supports directional clock encoder mode, which requires configuring the SMS bit field of the slave mode control register ATIM_SMCR to be 0xC or 0xD, corresponding to x2 mode or x1 mode, respectively. The functional block diagram can be referred to [Figure 14-28 Encoder mode block diagram](#), which allows filtering and polarity control of the input signal.

The directional clock encoder provides an up count clock line and a down count clock line, respectively, depending on the direction of rotation. In x2 mode, the counter counts on the rising and falling edges of either of the two clock lines; the CC1P and CC2P bits encode the clock idle state, with CCyP being 0 corresponding to high level idle and CCyP being 1 corresponding to low level idle. In x1 mode, the counter counts on a single clock edge based on the value of the CC1P and CC2P bit fields, with CCyP being 0 corresponding to falling-edge sensitive and high-level idle, and CCyP being 1 corresponding to rising-edge sensitive and low-level idle.

The following table shows the relationship between the counting direction and the encoder signal in different modes:

Table 14-6 Relationship between counting direction and encoder signal

Mode	SMS	Signal level		TI1FP1		TI2FP2	
		TI2FP2	TI1FP1	Rising	Falling	Rising	Falling
x2 mode CCyP=0	1100	High	High	Down count	Down count	Up count	Up count
		Low	Low	No count	No count	No count	No count
x2 mode CCyP=1		High	High	No count	No count	No count	No count
		Low	Low	Down count	Down count	Up count	Up count
x1 mode CCyP=0	1101	High	High	No count	Down count	No count	Up count
		Low	Low	No count	No count	No count	No count
x1 mode CCyP=1		High	High	No count	No count	No count	No count
		Low	Low	Down count	No count	Up count	No count

The following figure shows an example of counting in directional clock encoder mode:

Figure 14-32 Example of directional clock encoder mode counting (CC1P = CC2P = 0)

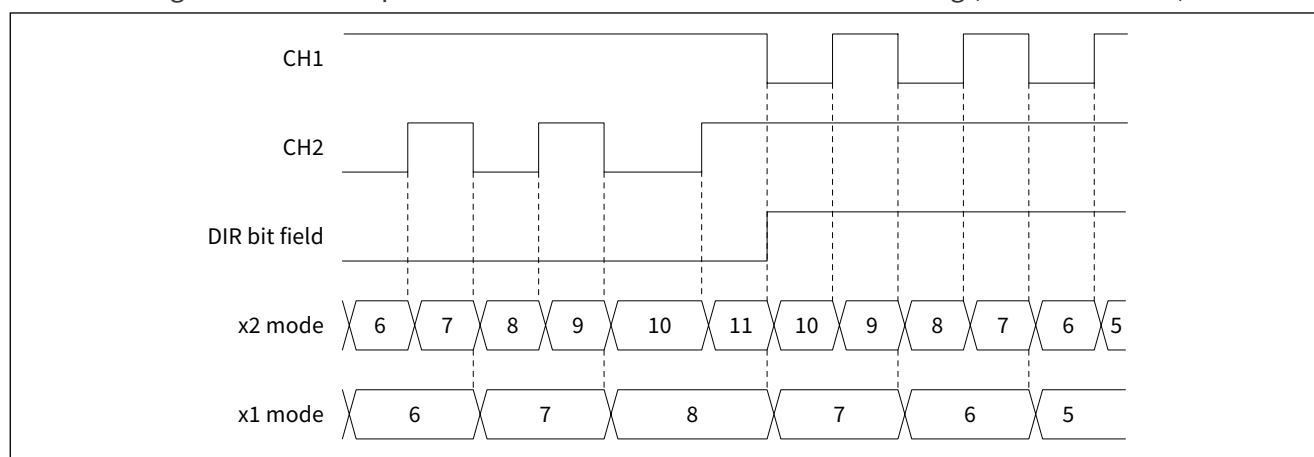
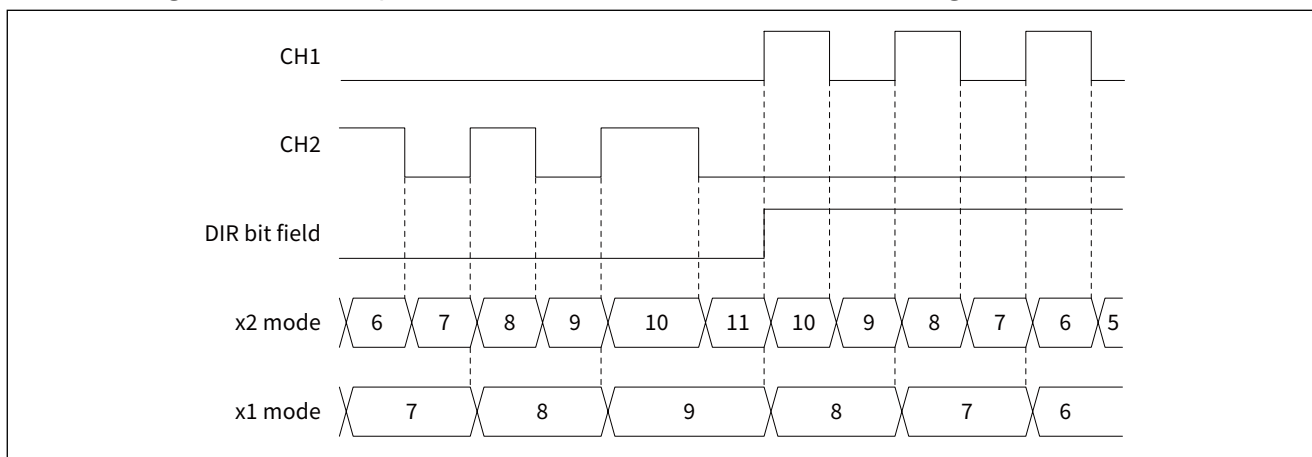


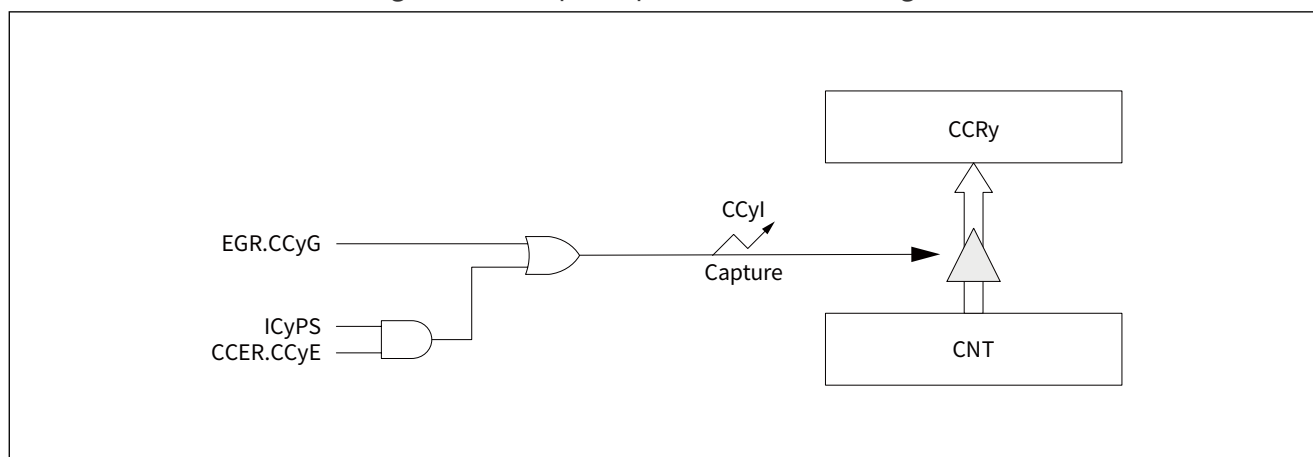
Figure 14-33 Example of directional clock encoder mode counting (CC1P = CC2P = 1)



14.3.3 Input capture function

ATIM supports the input capture function. When the CCyS bit field of the capture mode register ATIM_CCMRxCAP is set to a non-zero value, the CCy channel is configured as an input, and at the same time, the input mapping of the corresponding capture channel ICy is specified. Input capture is supported to be triggered by software or hardware, and its function block diagram is shown below:

Figure 14-34 Input capture mode block diagram



Software Trigger: When ATIM_EGR.CCyG is set to 1, a capture will be triggered by software immediately, and the value of the current counter CNT is latched into the capture/compare register CCRy of the corresponding channel to complete a capture, and the CCyG bit is cleared by the hardware automatically after the completion of the capture.

Hardware Trigger: When a valid edge is detected on the input channel TIy, the value of the current counter CNT is latched into the capture/compare register CCRy of the corresponding channel to complete a capture. The valid edge that triggers the capture is configured through the CCyNP and CCyP bit fields of the capture/compare enable register ATIM_CCER as shown in the table below:

Table 14-7 Input capture mode configuration

ATIM_CCER register		Capture mode configuration
CCyNP bit field value	CCyP bit field value	
0	0	Rising edge capture
0	1	Falling edge capture
1	1	Both rising and falling edge capture

When a capture occurs, the capture/compare interrupt flag ATIM_ISR.CCyIF of the corresponding channel is set to 1 by hardware and:

- If interrupt is enabled (setting ATIM_IER.CCyIE to 1), an interrupt request is generated.
- If the ATIM_ISR.CCyIF flag bit is already high when the capture event occurs, the repeat capture flag bit ATIM_ISR.CCyOF will be set to 1 by hardware.
- Setting ATIM_ICR.CCyIF to 0 or reading capture register CCRy clears the ATIM_ISR.CCyIF flag bit, and setting ATIM_ICR.CCyOF to 0 clears the ATIM_ISR.CCyOF flag bit.

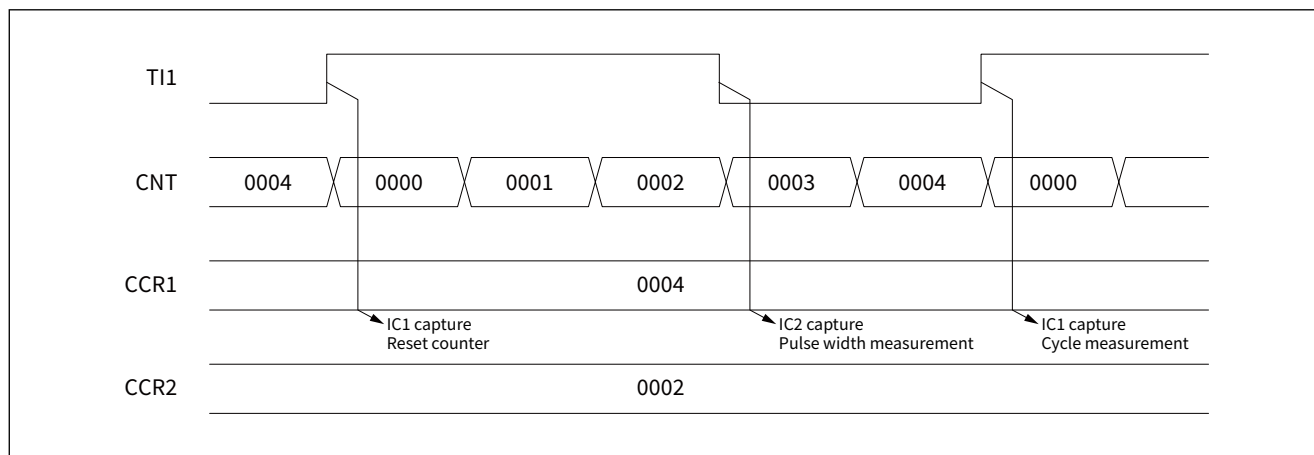
14.3.3.1 PWM input mode

PWM input mode is an application of input capture mode. The input capture function supports the same channel input signal to be captured by two capture registers. Combined with the slave mode reset function, the period and pulse width of the PWM input signal can be easily measured.

The PWM input mode can only be used with the TI1 and TI2 signals because only TI1FP1 and TI2FP2 are connected to the slave mode controller.

The following figure shows an example of the PWM signal timing when it is input from the TI1 channel:

Figure 14-35 PWM input mode timing



14.3.3.2 Input capture source

The ATIM input capture source can be the external ATIM_CHy pin or other peripherals on-chip, configured through the TlySEL bit field of the TI input select registers (ATIM_TISEL1 and ATIM_TISEL2).

When ATIM_TISELx.TlySEL is 0x0, the source of the Tly channel input capture signal is the external ATIM_CHy pin, in which case the corresponding pin needs to be configured as a multiplexing function through the GPIO multiplexing function registers (GPIOx_AFRH and GPIOx_AFRL).

When ATIM_TISELx.TlySEL is 0x1~0xD, the Tly channel input capture signals come from other peripherals on-chip, as shown in the following table:

Table 14-8 Advanced timer input capture source

TlySEL bit field value	Input capture source for the Tly channel
0000	ATIM_CHy pin
0001	VC1_OUT
0010	VC2_OUT
0011	UART1_RXD
0100	UART2_RXD
0101	MCO_OUT
0110	HSE_FAULT
0111	LSE_FAULT
1000	RTC_OUT
1001	LSI_OUT
1010	BTIM1_Trgo
1011	BTIM2_Trgo
1100	BTIM3_Trgo
1101	GTIM_Trgo



14.3.4 Output compare function

The ATIM supports the output compare function. Setting the CCyS bit field of the compare mode register ATIM_CCMRxCMP to 0 configures the CCy channel as an output. In the output compare mode, the value of the current counter CNT is compared with the value of the capture/compare register CCRy of the corresponding channel CCy. When the two match, the reference signal OCyREF is output to a settable level state, and a compare interrupt is generated at the same time. The reference signal is output by the CHy and CHyN pins after passing through the output control unit, and its output logic can be found in [14.3.1.10 Output compare channel](#).

The output action of the reference signal OCyREF is set by the OCyM bit field of the compare mode register ATIM_CCMRxCMP, as shown in the following table:

Table 14-9 Output compare mode configuration

OCyM bit field value	Compare mode configuration
0000	When comparing and matching, OCyREF keeps the original level
0001	When comparing and matching, OCyREF is set to 1
0010	When comparing and matching, OCyREF is set to 0
0011	When comparing and matching, OCyREF flips
0100	Force OCyREF to low level
0101	Force OCyREF to high level
0110	PWM mode 1
0111	PWM mode 2
1000	Retriggerable OPM mode 1
1001	Retriggerable OPM mode 2
1011	Count direction output
1100	Combined PWM mode 1
1101	Combined PWM mode 2
1110	Asymmetric PWM mode 1
1111	Asymmetric PWM mode 2

When a compare match occurs, the capture/compare interrupt flag ATIM_ISR.CCyIF of the corresponding channel is set to 1 by hardware while:

- If interrupt is enabled (setting ATIM_IER.CCyIE to 1), an interrupt request will be generated.
- Setting ATIM_ICR.CCyIF to 0 clears the ATIM_ISR.CCyIF flag bit.

The capture/compare register ATIM_CCRy has a cache function and the use of the cache function is selected through the OCyPE bit field. When OCyPE is 0, the comparison cache function of channel CHy is disabled, and the ATIM_CCRy register can be updated by software at any time; the updated value takes effect immediately and affects the output waveform; When OCyPE is 1, the comparison cache function of channel CHy is enabled, and updating the ATIM_CCRy register will not take effect immediately, and the value of ATIM_CCRy register will be updated to the valid register only when the update event UEV occurs.



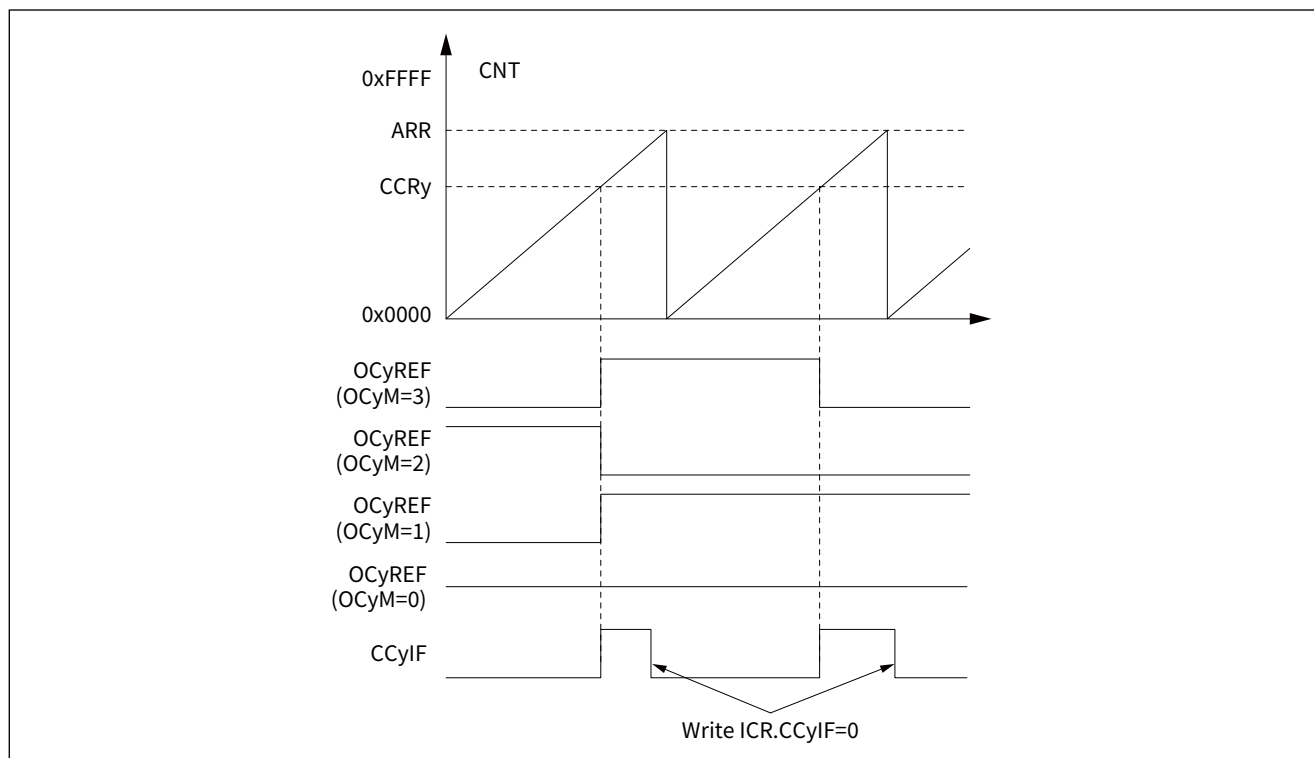
14.3.4.1 Match output

The match output mode is used to control the output waveform or to indicate that a certain time period has passed.

- Set the OCyM bit field to 0x0, when comparing and matching, the reference signal OCyREF keeps its level while the ATIM_ISR.CCyIF flag position 1.
- Set the OCyM bit field to 0x1, when comparing and matching, the reference signal OCyREF is set to high level while ATIM_ISR.CCyIF flag position 1.
- Set the OCyM bit field to 0x2, when comparing and matching, the reference signal OCyREF is set to low level while ATIM_ISR.CCyIF flag position 1.
- Set the OCyM bit field to 0x3, when comparing and matching, the reference signal OCyREF is flipped while the ATIM_ISR.CCyIF flag position 1.

The following figure shows an example of the timing for each compare-match output mode, with the example being the edge-aligned incremental mode:

Figure 14-36 Compare match output mode



14.3.4.2 Forced output

In forced output mode, the output comparison signal can be forced to a high or low state directly by software without relying on the comparison result of the capture/compare register ATIM_CCRy and the count register ATIM_CNT.

Set the OCyM bit field of the compare mode register ATIM_CCMRxCMP to 0x4 to force the reference signal OCyREF to a low level; set the OCyM bit field to 0x5 to force the reference signal OCyREF to a high level.

In forced output mode, the comparison between the ATIM_CCRy register and counter ATIM_CNT is still in progress, and the corresponding flags are set to 1. The corresponding interrupt requests are also generated.



14.3.4.3 PWM output

Pulse width modulation (PWM) mode can generate a signal with a frequency determined by the reload register ATIM_ARR and a duty cycle determined by the capture/comparison register ATIM_CCRy.

In order to prevent the output PWM signal from being incorrect when modifying the ARR register, it is necessary to set ATIM_CR1.ARPE to 1 to enable the cache function of the ARR register; in order to prevent the output PWM signal from being incorrect when modifying the CCRy register, it is necessary to set ATIM_CCMRxCMP.OCyPE to 1 to enable the cache function of the CCRy register. Only when the update event UEV occurs the value of the preloaded register will be transferred to the shadow register; before starting the timer after configuring various parameters, it is necessary to write 1 to the ATIM_EGR.UG bit to initialise all registers.

The PWM mode can be selected independently for each channel by simply writing 0x6 (PWM mode 1) or 0x7 (PWM mode 2) to the OCyM bit field of the compare mode register, ATIM_CCMRxCMP, with the output states shown in the following table:

Table 14-10 PWM mode 1/2 output state

Operating mode	Counting direction	PWM mode 1	PWM mode 2
Edge aligned mode Center aligned mode	Up count	CNT < CCRy, OCyREF is high	CNT < CCRy, OCyREF is low
	Down count	CNT > CCRy, OCyREF is low	CNT > CCRy, OCyREF is high

The polarity of OCy can be set through the CCyP bit field of the ATIM_CCER register, which can be set to active high or active low. The OCy output is enabled by a combination of the CCyE, CCyNE, MOE, OSSR, and OSSR bits (ATIM_CCER and ATIM_BDTR registers).

The following are examples of waveforms for each PWM mode in different count modes, where ATIM_ARR is 0x08:

Figure 14-37 PWM mode 1, edge-aligned mode, count up

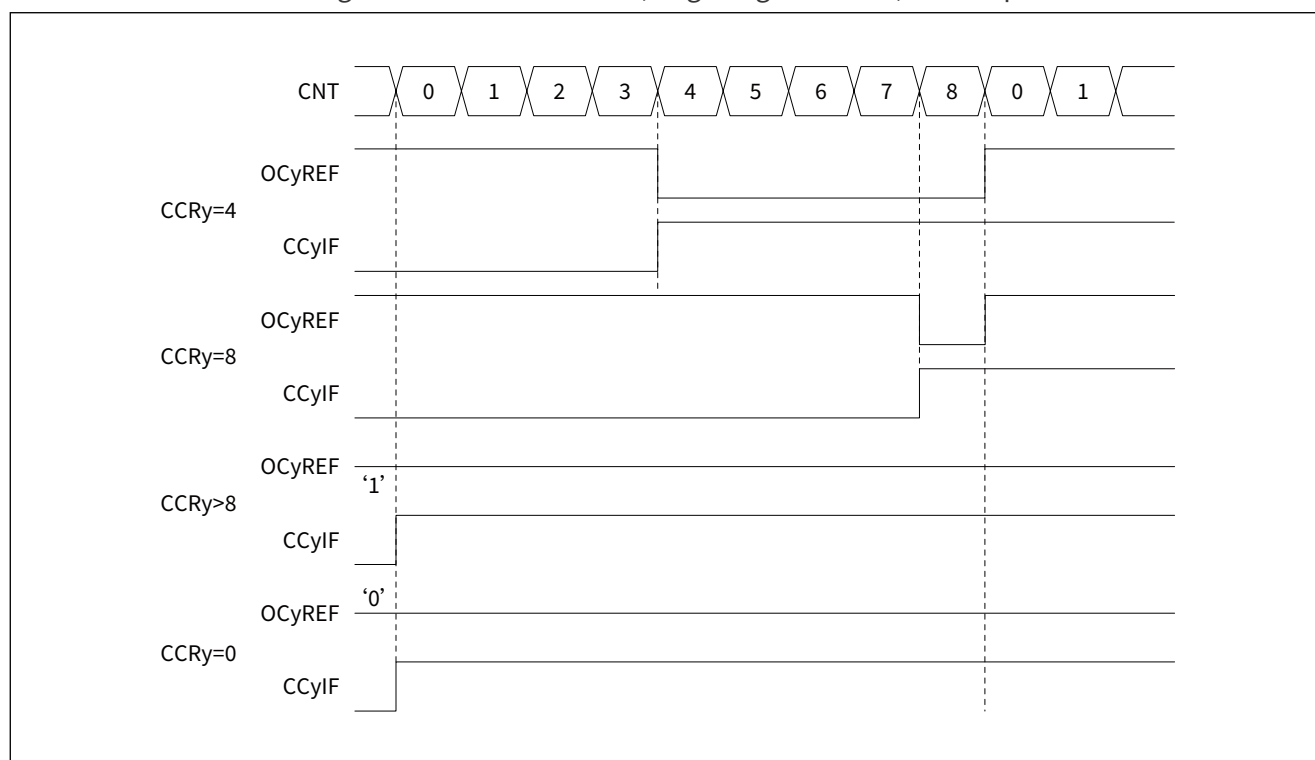


Figure 14-38 PWM mode 1, edge-aligned mode, count down

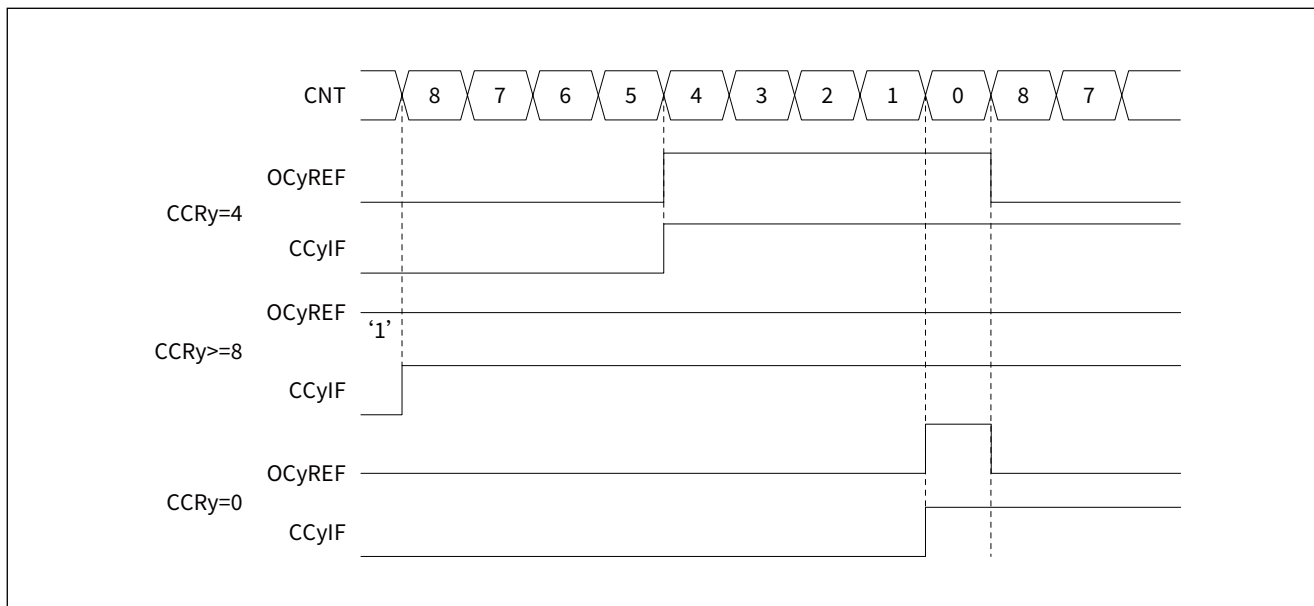
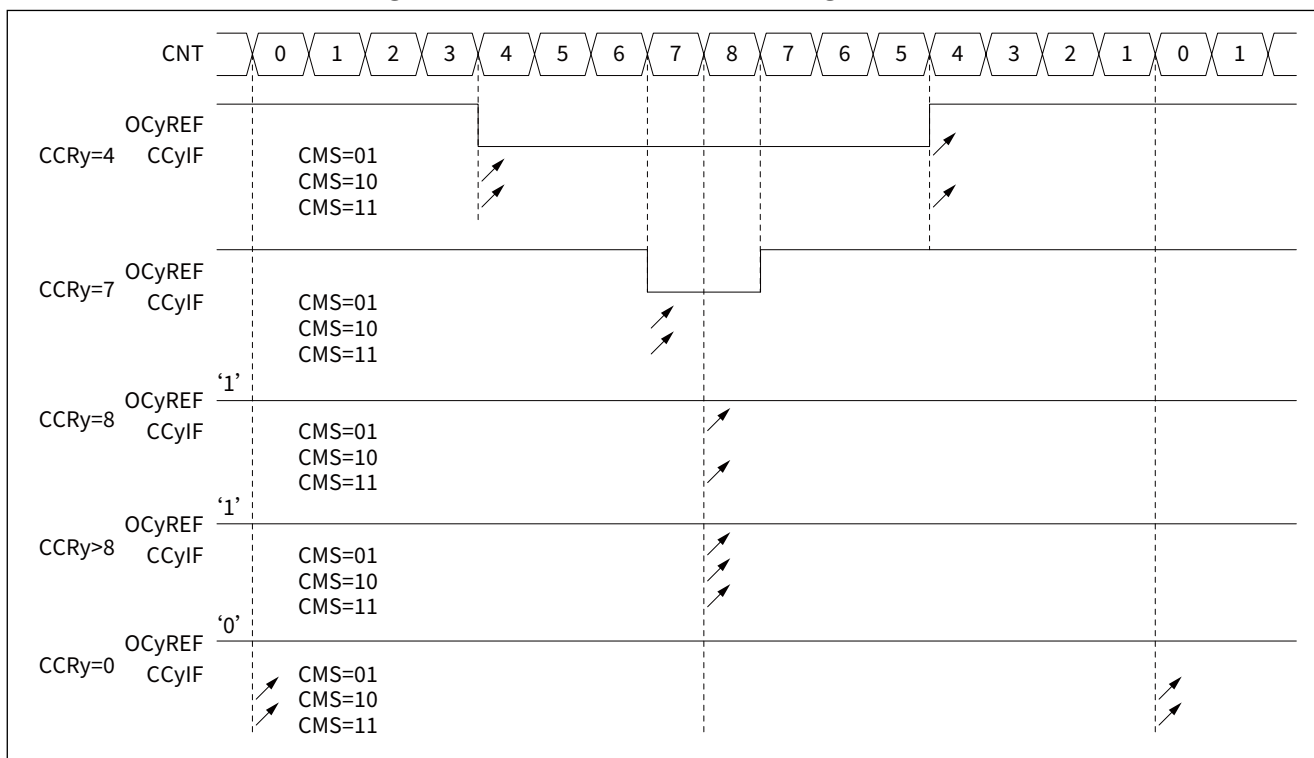


Figure 14-39 PWM mode 1, center-aligned mode



14.3.4.4 Asymmetric PWM output

The asymmetric PWM output is relative to the symmetric PWM output when counting in center-aligned mode. In the asymmetric PWM output mode, two PWM signals with programmable phase shifts are allowed to be generated in centre-aligned mode, with the frequency determined by the reload register ATIM_ARR, and the duty cycle and phase shifts determined by the two capture/comparison registers ATIM_CCRy, where one of the CCRy registers controls the PWM during counting up, and the other CCRy register controls the PWM during counting down, the details are shown below:

- OC1REFC is controlled by CCR1 (count up) and CCR2 (count down).
- OC2REFC is controlled by CCR2 (count up) and CCR1 (count down).
- OC3REFC is controlled by CCR3 (count up) and CCR4 (count down).
- OC4REFC is controlled by CCR4 (count up) and CCR3 (count down).
- OC5REFC is controlled by CCR5 (count up) and CCR6 (count down).
- OC6REFC is controlled by CCR6 (count up) and CCR5 (count down).

Each channel can independently select the asymmetric PWM mode by simply writing 0xE (asymmetric PWM mode 1) or 0xF (asymmetric PWM mode 2) to the OCyM bit field of the compare mode register ATIM_CCMRxCMP.

When the channel is set to be used as an asymmetric PWM output channel, the other auxiliary channel can still be used. For example, when channel 1 is configured to generate an OC1REFC signal in asymmetric PWM mode 1, channel 2 can still be configured to output an OC2REF signal in PWM mode 2 or to output an OC2REFC signal in asymmetric PWM mode 2.

The following figure shows an example of a waveform for asymmetric PWM mode 1/2, where ARR=8, CCR1=0, CCR2=8, CCR3=3, and CCR4=5:

Figure 14-40 Example of asymmetric PWM mode 1

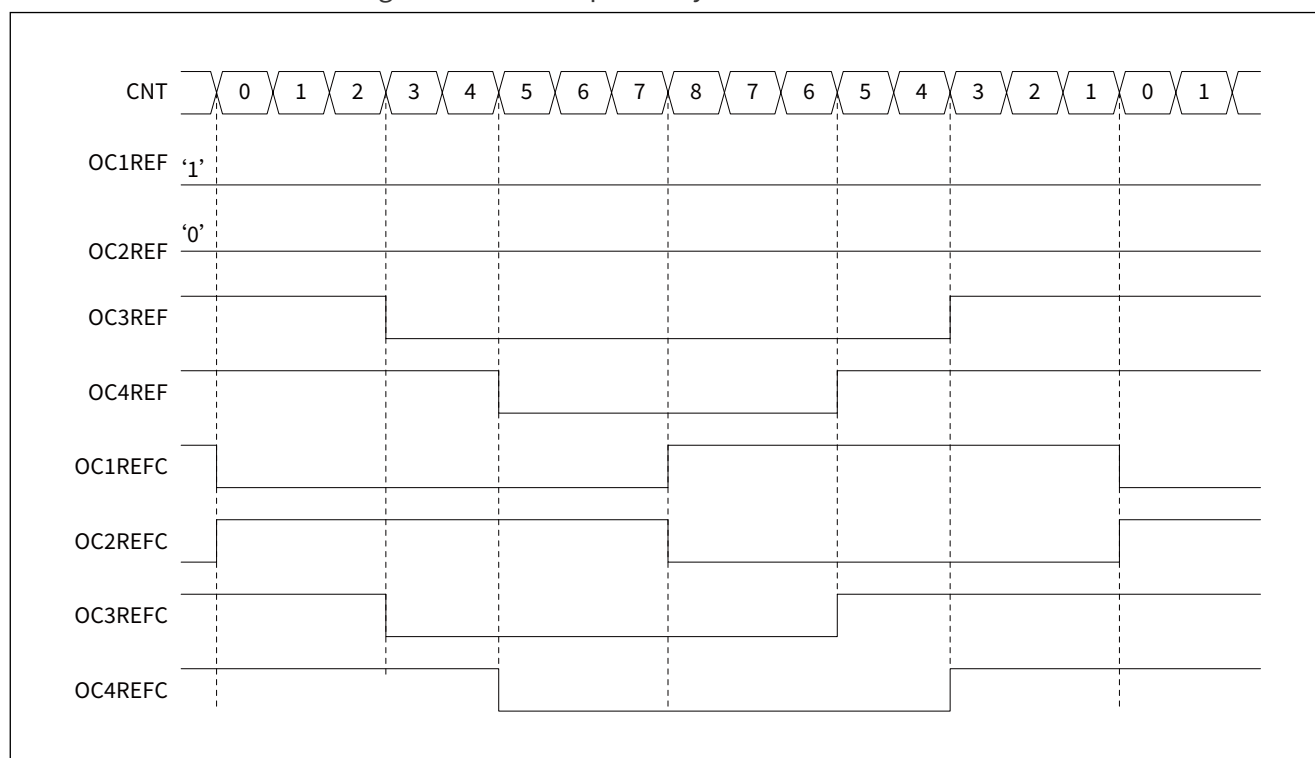
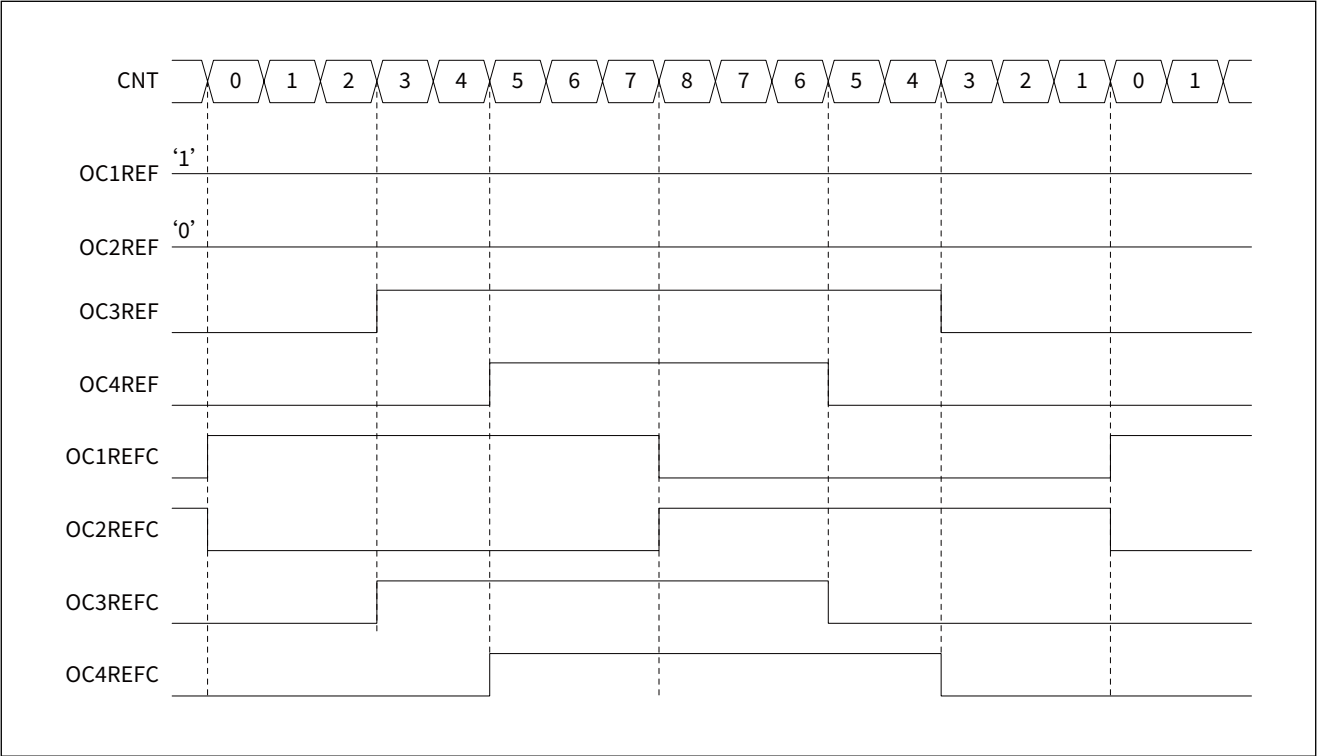


Figure 14-41 Example of asymmetric PWM mode 2



14.3.4.5 Combined PWM output

In combined PWM output mode, two PWM signals with programmable delay and phase shift are allowed to be generated in either edge-aligned or center-aligned mode, with the frequency determined by the reload register ATIM_ARR, and the duty cycle and delay determined by the two capture/comparison registers ATIM_CCRy.

In this mode, the OCyREFC signal is generated by the logic or operation or the combination of logic and operation of the two reference signals OCyREF, and the combined PWM mode can be independently selected through the OCyM bit field of the comparison mode register ATIM_CCMRxCMP as shown in the following table:

Table 14-11 Combined PWM mode configuration

OCyM bit field value	Output state
0xC (Combined PWM mode 1)	OC1REFC (or OC2REFC) is generated by the logical OR operation of OC1REF and OC2REF. OC3REFC (or OC4REFC) is generated by the logical OR operation of OC3REF and OC4REF. OC5REFC (or OC6REFC) is generated by the logical OR operation of OC5REF and OC6REF.
0xD (Combined PWM mode 2)	OC1REFC (or OC2REFC) is generated by the logical AND operation of OC1REF and OC2REF. OC3REFC (or OC4REFC) is generated by the logic AND operation of OC3REF and OC4REF. OC5REFC (or OC6REFC) is generated by the logical AND operation of OC5REF and OC6REF.

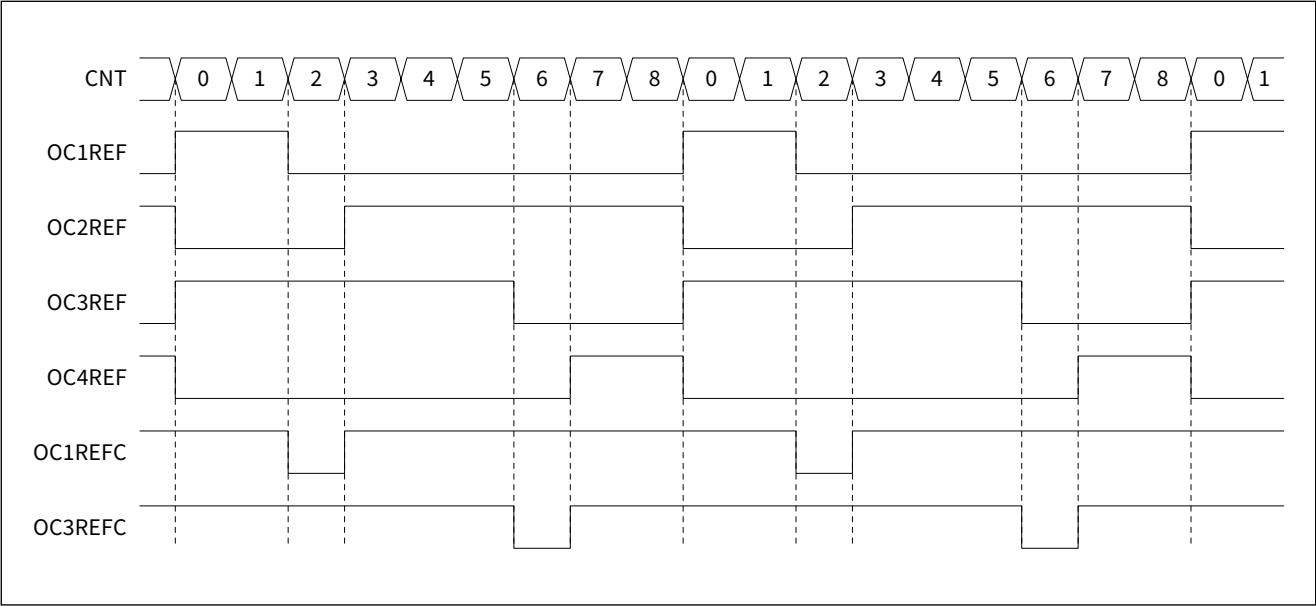
When the channel is set to be used as a combination PWM output channel, the other auxiliary channel can still be used, but must be set to the opposite PWM mode. For example, channel 2 can still be used when channel 1 is configured to generate the OC1REFC signal in combined PWM mode 1, and channel 2 can be configured to output the OC2REF signal in PWM mode 2.

The following figure shows an example of the signals for the combined PWM mode 1 in the edge-aligned incremental mode, which is configured as follows:

- The reload value ARR is 8.
- Channel 1 is configured as combined PWM mode 1, CCR1=2.
- Channel 2 is configured as PWM mode 2, CCR2=3.
- Channel 3 is configured as combined PWM mode 1, CCR3=6.
- Channel 4 is configured as PWM mode 2, CCR4=7.



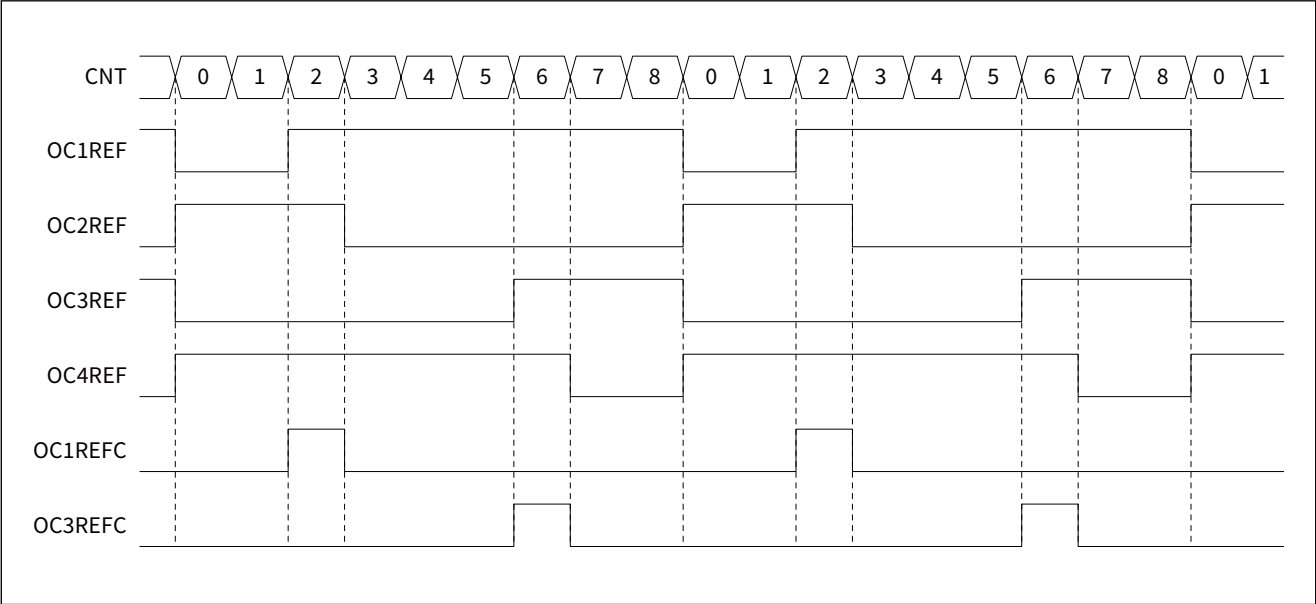
Figure 14-42 Example of combined PWM mode 1



The following figure shows an example of the signals for the combined PWM mode 2 in the edge-aligned incremental mode, which is configured as follows:

- The reload value ARR is 8.
- Channel 1 is configured as combined PWM mode 2 , CCR1=2.
- Channel 2 is configured as PWM mode 1 , CCR2=3.
- Channel 3 is configured as combined PWM mode 2, CCR3=6.
- Channel 4 is configured as PWM mode 1, CCR4=7.

Figure 14-43 Example of combined PWM mode 2



14.3.4.6 Phase shift PWM mode

In motor applications, it is necessary to sample the current through the ADC at a specific moment between the rising edges of the two PWM waveforms. When the interval between the rising edges of the two PWM waveforms is too short to complete the current sampling, the edge of one of the PWM waveforms can be adjusted through the phase-shift PWM mode to increase the interval between the rising edges of the two PWM waveforms under the condition of keeping the duty cycle unchanged.

Set the GC5Cy bit field of the ATIM_CCR5 register to 1 to enable phase-shift PWM mode 5, and the OCyREFC output states are as follows:

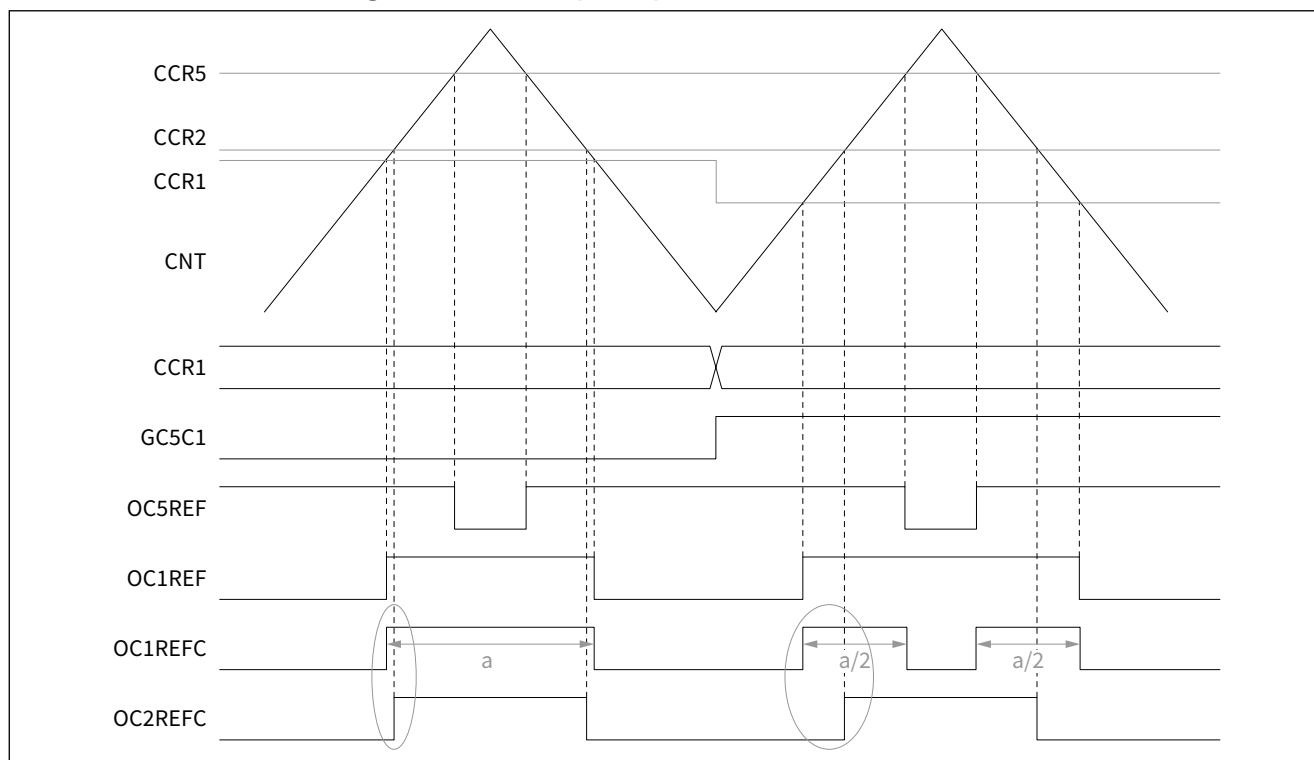
- Set GC5C1 to 1, OC1REFC outputs the logical AND of OC1REF and OC5REF
- Set GC5C2 to 1, OC2REFC outputs the logical AND of OC2REF and OC5REF
- Set GC5C3 to 1, OC3REFC outputs the logical AND of OC3REF and OC5REF
- Set GC5C4 to 1, OC4REFC outputs the logical AND of OC4REF and OC5REF
- Set GC5C6 to 1, OC6REFC outputs the logical AND of OC6REF and OC5REF

Set the GC6Cy bit field of the ATIM_CCR6 register to 1 to enable phase-shift PWM mode 6, and the OCyREFC output states are as follows:

- Set GC6C1 to 1, OC1REFC outputs OC1REF (count up) and OC6REF (count down)
- Set GC6C2 to 1, OC2REFC outputs OC2REF (count up) and OC6REF (count down)
- Set GC6C3 to 1, OC3REFC outputs OC3REF (count up) and OC6REF (count down)
- Set GC6C4 to 1, OC4REFC outputs OC4REF (count up) and OC6REF (count down)
- Set GC6C5 to 1, OC5REFC outputs OC5REF (count up) and OC6REF (count down)

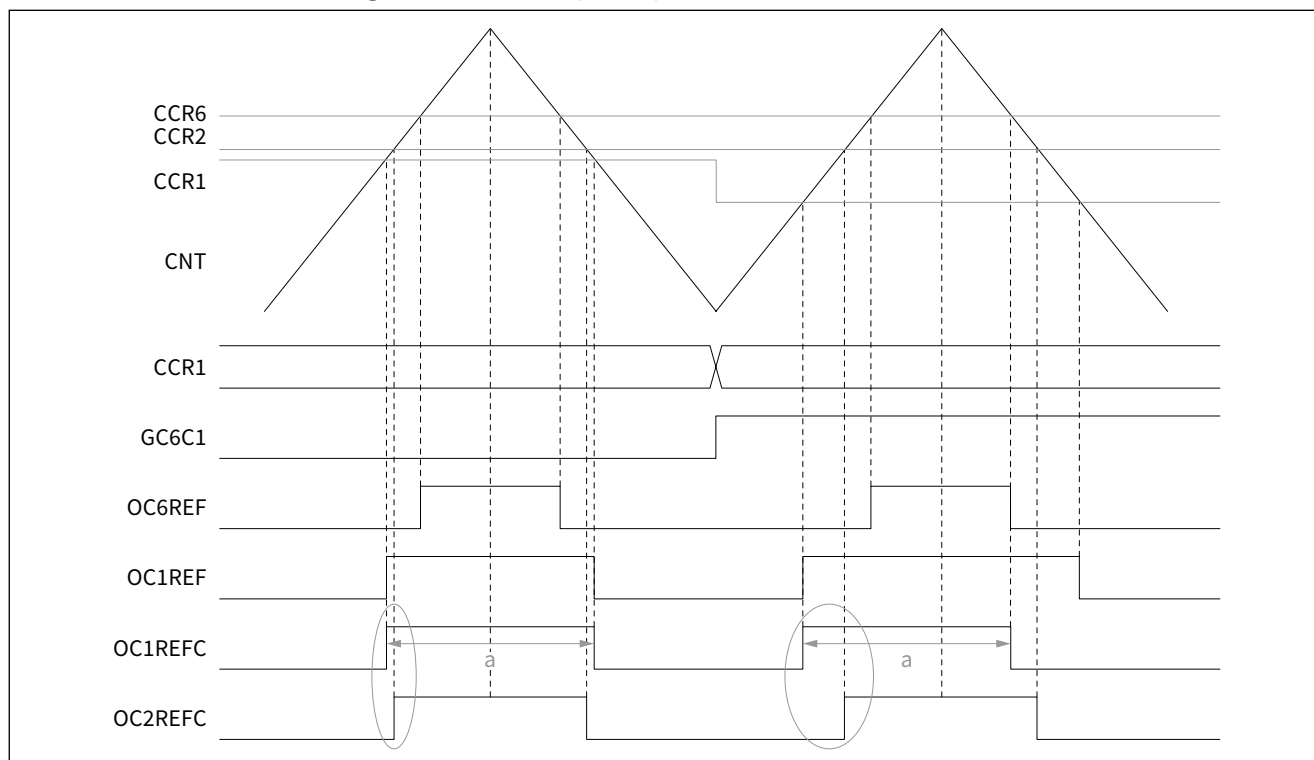
The following figure shows the timing diagram for increasing the interval between the rising edge of OC1REFC and the rising edge of OC2REFC in phase-shifted PWM mode 5 by adjusting CCR1 and GC5C1:

Figure 14-44 Example of phase-shifted PWM mode 5



The following figure shows the timing diagram for increasing the interval between the rising edge of OC1REFC and the rising edge of OC2REFC in phase-shifted PWM mode 6, achieved by adjusting CCR1 and GC6C1.

Figure 14-45 Example of phase-shifted PWM mode 6



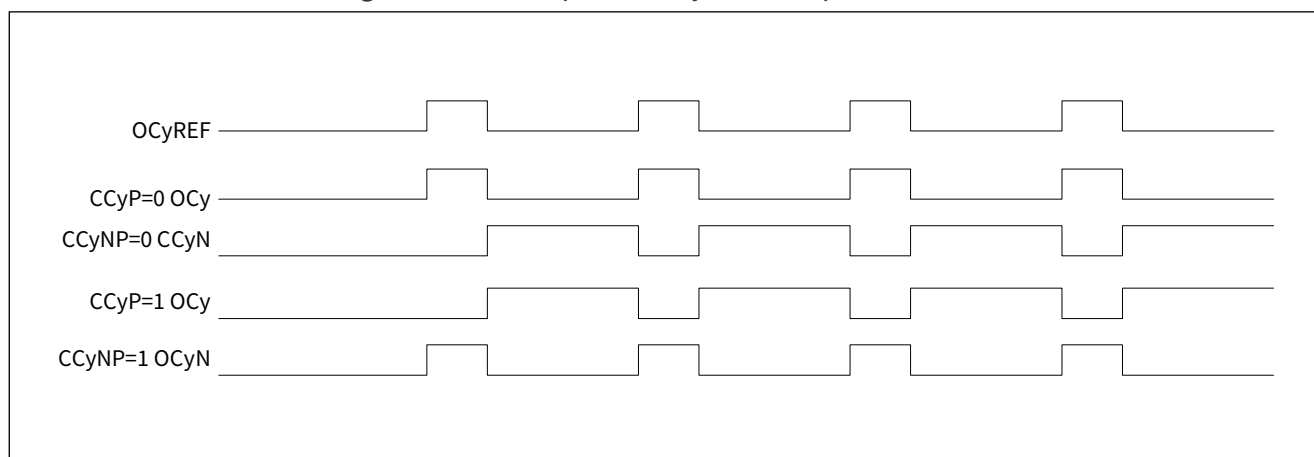
14.3.4.7 Complementary output and dead-time insertion

ATIM can output 6 pairs of complementary PWM signals and supports symmetric or asymmetric dead time insertion.

The main output OCy and the complementary output OCyN can be independently set in output polarity and are controlled via the CCyP and CCyNP bit fields of the ATIM_CCER register, respectively.

The main output OCy and complementary output OCyN are activated by a combination of several control bits: the CCyE and CCyNE bits in the ATIM_CCER register, the MOE, OSS1, and OSSR bits in the ATIM_BDTR register, and the OISy and OISyN bits in the ATIM_CR2 register. For more details, see [Table 14-13 Output control bits for complementary channels OCy and OCyN with brake function](#). It should be noted that at the moment of switching to the idle state (MOE becomes 0), the deadband is still active.

Figure 14-46 Complementary PWM output waveform



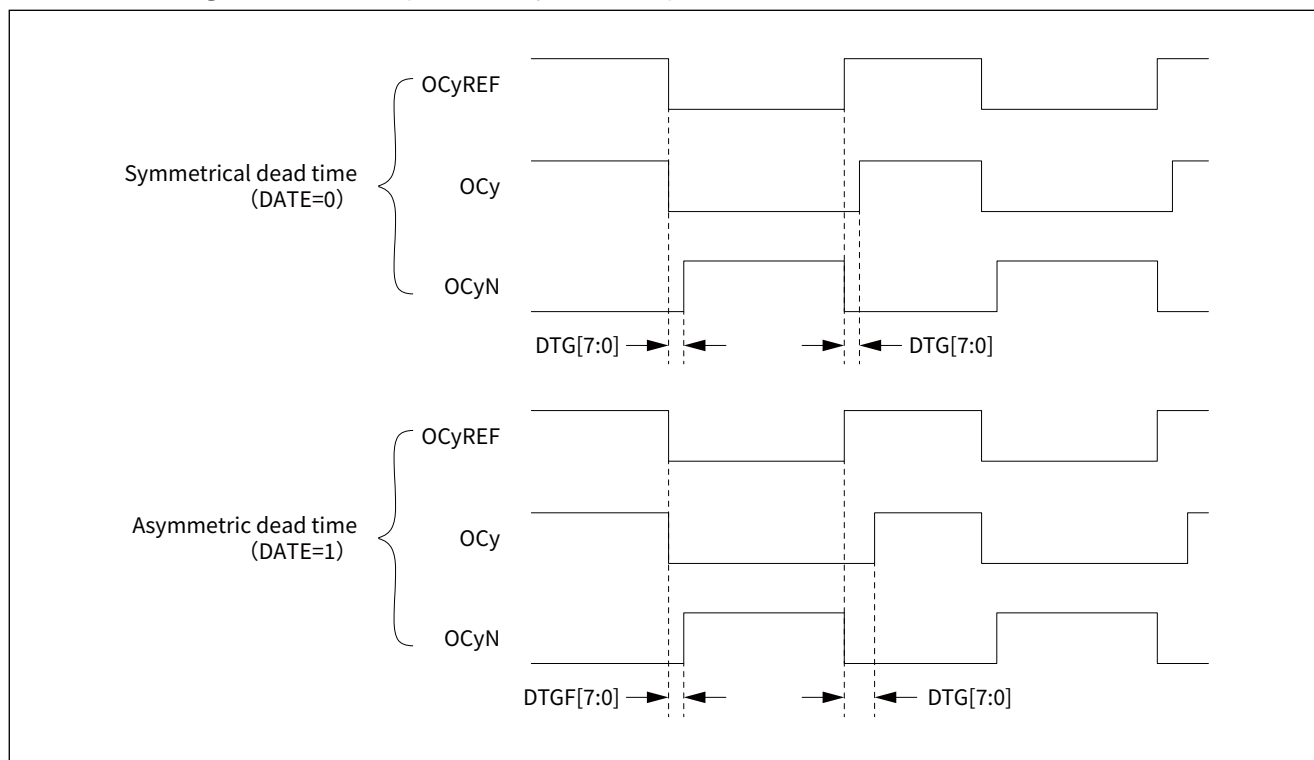
ATIM supports symmetric or asymmetric dead insertion times, which are the same for all channels. When ATIM_DTR2.DTAE is 0, the dead time of the rising edge and the falling edge is the same and is defined by DTG[7:0] in the ATIM_BDTR register; when ATIM_DTR2.DTAE is 1, the dead time of the rising edge is defined by DTG[7:0] and the dead time of the falling edge is defined by DTGF[7:0]. The DTAE bit must be written before enabling the counter and must not be modified when CEN is 1. DTG[7:0] is defined as shown in the table below and DTGF [7:0] is defined as DTG[7:0].

Table 14-12 Dead-time settings

ATIM_BDTR.DTG	Step length	Dead-time	When $t_{DTS}=125\text{ns}$, dead-time range
DTG[7:5]=0xx	$1t_{DTS}$	$\text{DTG}[7:0] \times t_{DTS}$	0~15.875 μs
DTG[7:5]=10x	$2t_{DTS}$	$(64+\text{DTG}[5:0]) \times 2 \times t_{DTS}$	16~31.750 μs
DTG[7:5]=110	$8t_{DTS}$	$(32+\text{DTG}[4:0]) \times 8 \times t_{DTS}$	32~63 μs
DTG[7:5]=111	$16t_{DTS}$	$(32+\text{DTG}[4:0]) \times 16 \times t_{DTS}$	64~126 μs

The following figure shows the relationship between the complementary PWM output signal and the reference signal OCyREF for symmetric and asymmetric deadband insertion times, with CCyP=0 and CCyNP=0 in the example:

Figure 14-47 Complementary PWM output waveform with dead-time insertion



A preload mechanism may be used to update the dead time value on the fly during PWM operation. When the DTPE bit is set to 1, the dead time value preload is enabled and the preload value is loaded into the valid register at the next update event. If the DTPE bit is enabled while the counter is enabled, any new values written since the last update are discarded and the previous values are used.

14.3.4.8 Brake function

ATIM supports two brake inputs BRK and BRK2, and when the brake signal is active, the brake circuit turns off the PWM output and forces it to a predefined safe state.

Brake channel sources include system-level faults (clock failures and parity errors, etc.) and application faults (from input pins and built-in comparators) that force the output to a predefined level (active or inactive) after a dead time duration. Brake 2 channel sources include application faults that can only force the output to an invalid state. BRK inputs have higher priority than BRK2 inputs.

Brake (BRK) trigger source:

- Software braking, set BG bit field of ATIM_EGR register to 1 to generate braking event, BG bit is automatically cleared by hardware.
- The compare output of voltage comparator VC1/2, the relevant bit field of the ATIM_AF1 register, sets the active polarity and enables the input.
- System brake request, enabled via the appropriate bit field of SYSCTRL_CR2 register:
 - Generate HardFault or Cortex-M0+ LockUp flag
 - LVD output
 - RAM parity error
 - DeepSleep
 - LSE runtime failure
 - HSE runtime failure
- External ATIM_BK pin input (GPIO function multiplex setting), the relevant bit field of the ATIM_AF1 register sets active polarity and input enable.

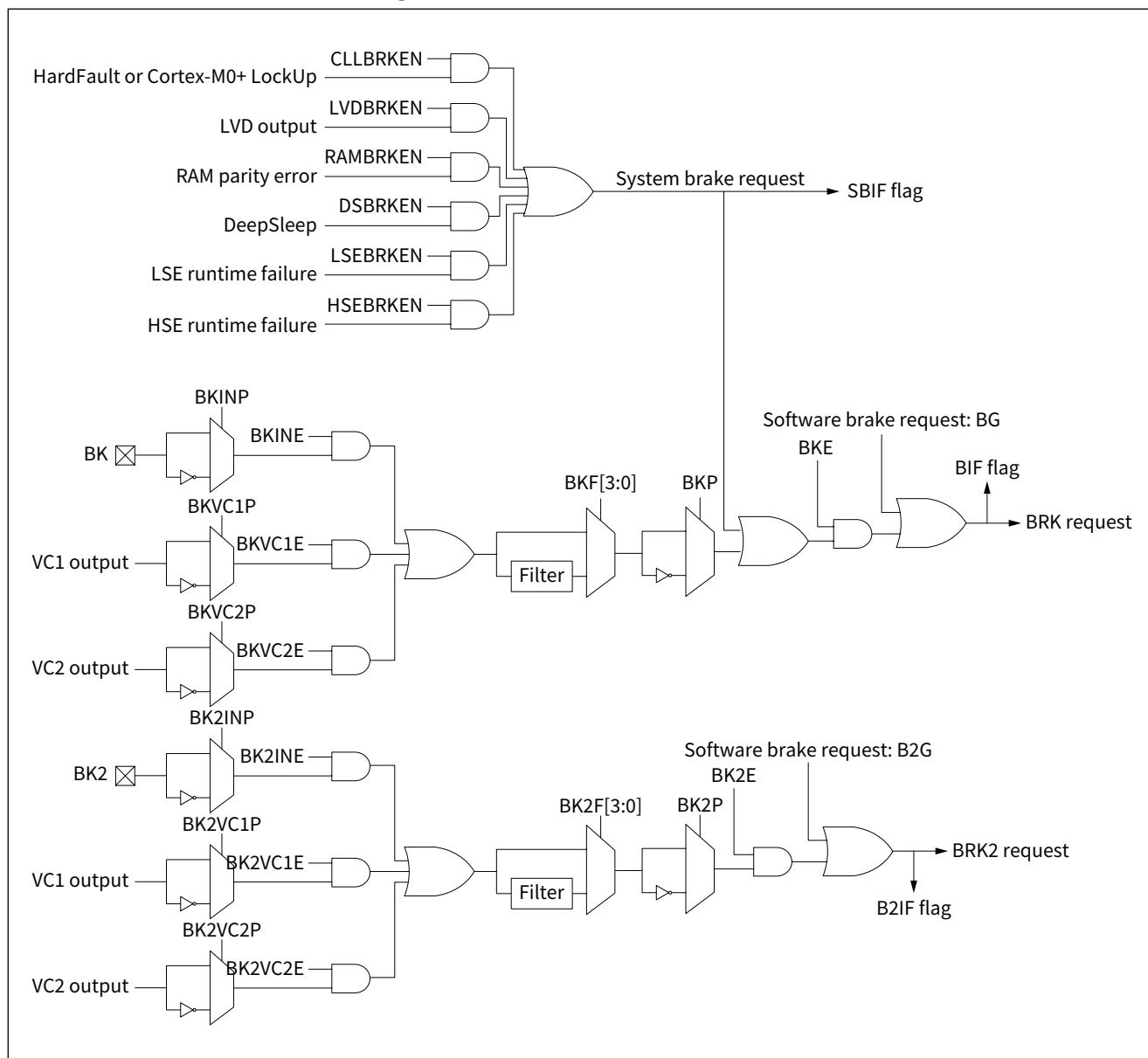
Brake 2 (BRK2) trigger source:

- Software braking, set the B2G bit field of ATIM_EGR register to 1 to generate braking event, B2G bit is automatically cleared by hardware.
- The compare output of voltage comparator VC1/2, the relevant bit field of the ATIM_AF2 register, sets the active polarity and enables the input.
- External ATIM_BK2 pin input (GPIO function multiplex setting), the relevant bit field of the ATIM_AF2 register sets active polarity and input enable.

Before all sources enter the timer BRK or BRK2 input, they are logically OR, as shown in the following figure:



Figure 14-48 Brake and brake 2 circuit

**Caution:**

Asynchronous (no clock) operation is only guaranteed when the programmable filter is disabled. If the programmable filter is enabled, fail-safe clock mode must be used to ensure that brake events can be handled.

When a braking event occurs, the MOE bit is cleared asynchronously, leaving the output in an inactive state, idle state or even releasing control to the GPIO controller (selected by the OSS1 bit, see [Table 14-13 Output control bits for complementary channels OCy and OCyN with brake function](#) for details). This feature is enabled even if the MCU oscillator is turned off. If AOE is set, the MOE bit is automatically set again on the next update event (UEV), otherwise MOE remains low until the application sets it again. Because the brake input is active, the MOE bit cannot be set (automatically or by software) and the status flags BIF and B2IF cannot be cleared when the brake input is active.

The following figure shows an example of the output behavior of OCy and OCyN when braking event BRK occurs:

Figure 14-49 Output behavior in response to braking event BRK (OSS1=1)

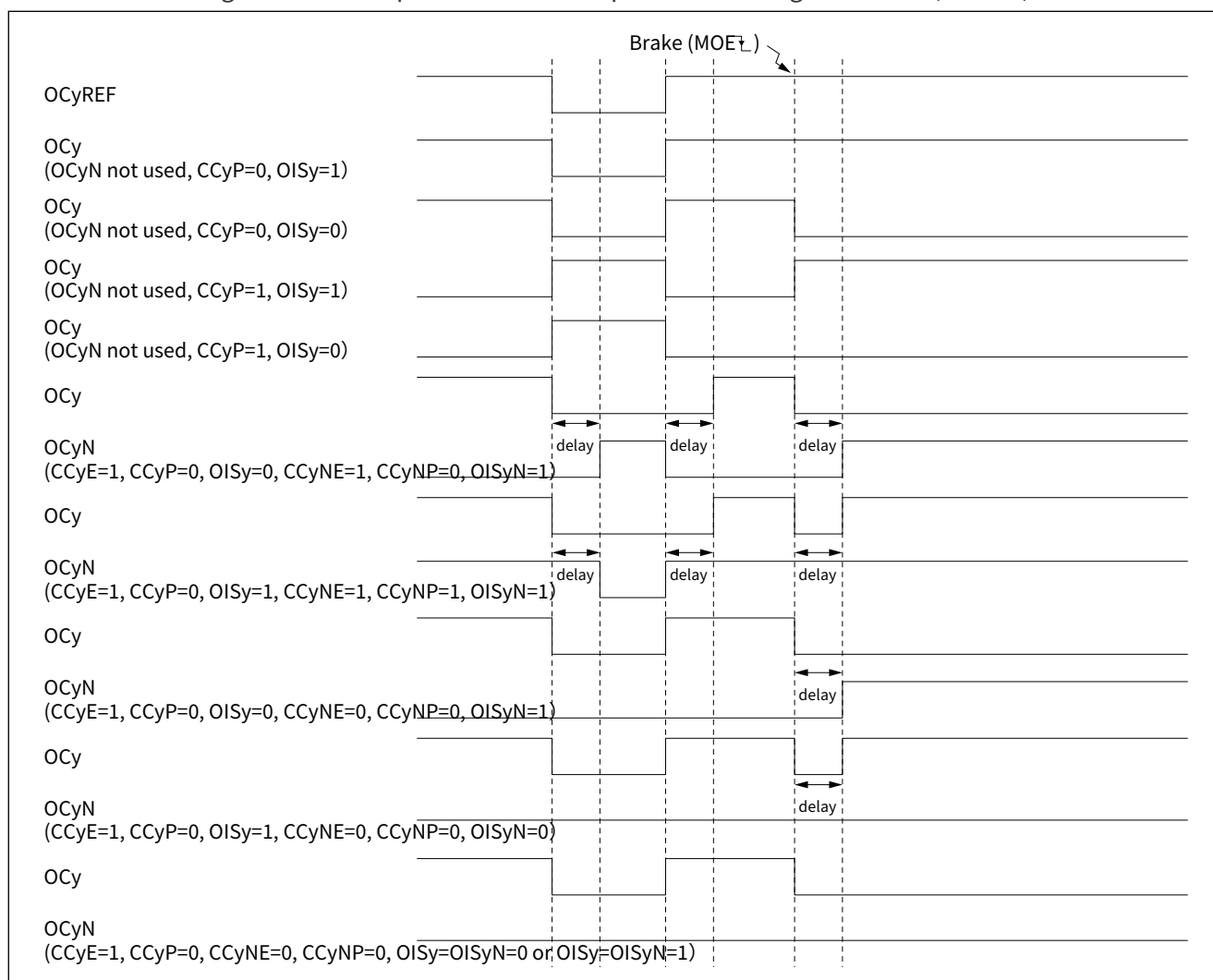
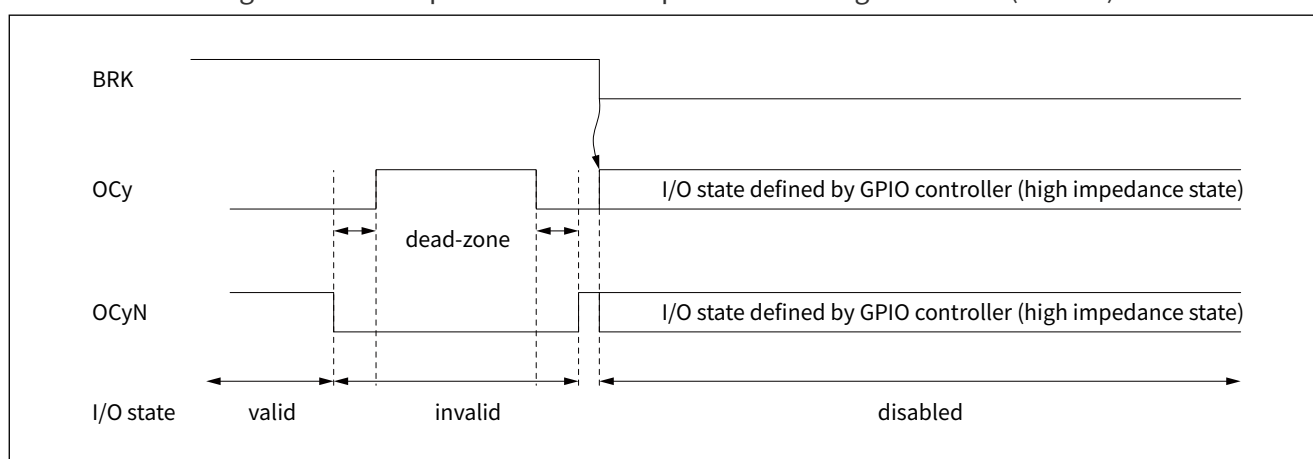


Figure 14-50 Output behavior in response to braking event BRK (OSS1=0)



The following figure shows an example of the output behavior of OCy and OCyN for braking events BRK and BRK2, where CCyP=CCyNP=0:

Figure 14-51 Output behavior in response to braking BRK and BRK2 (OSSl=1)

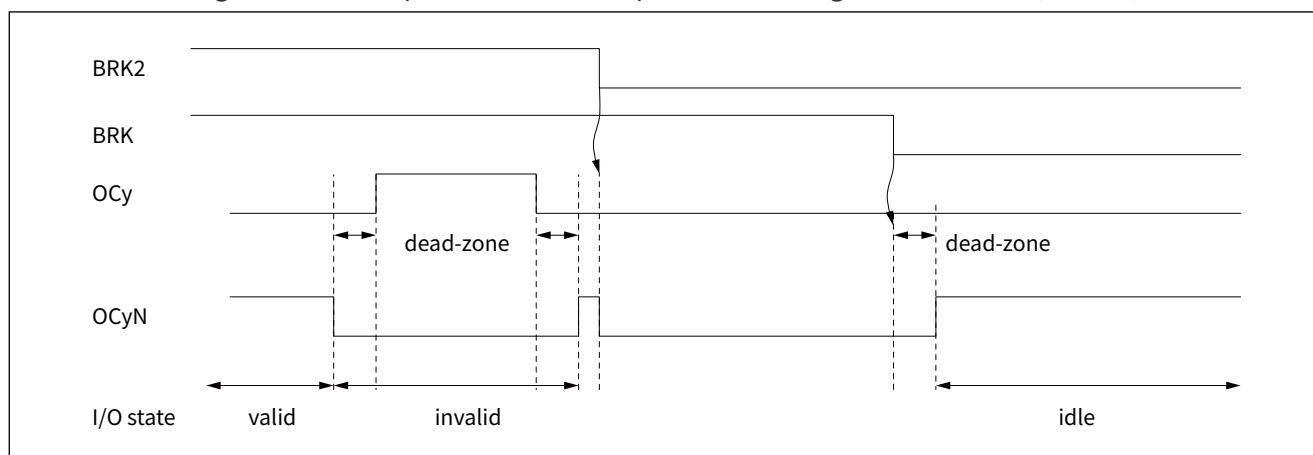


Table 14-13 Output control bits for complementary channels OCy and OCyN with brake function

Output control bit					Output state	
MOE	OSSI	OSSR	CCyE	CCyNE	OCy output state	OCyN output state
1	X	X	0	0	Disable output (no longer driven by timer); The output state is forced high impedance by GPIO logic; OCy=0, OCyN=0	
		0	0	1	Disable output (no longer driven by timer); The output state is forced high impedance by GPIO logic; OCy=0	OCyREF + polarity OCyN=OCyREF XOR CCyNP
		0	1	0	OCyREF + polarity OCy=OCyREF XOR CCyP	Disable output (no longer driven by timer); The output state is forced high impedance by GPIO logic; OCyN=0
		X	1	1	OCyREF + polarity + deadband	OCyREF complement + polarity + deadband
		1	0	1	Off state (output enable is invalid); OCy=CCyP	OCyREF + polarity OCyN=OCyREF XOR CCyNP
		1	1	0	OCyREF + polarity OCy=OCyREF XOR CCyP	Off state (output enable is invalid); OCyN=CCyNP
0	0	X	X	X	Disable output (no longer driven by timer).	
	1		0	0	The output state is forced high impedance by GPIO logic.	
			0	1	Off state (output enable is invalid).	
			1	0	Asynchronous: OCy=CCyP, OCyN=CCyNP (if BRK or BRK2 is triggered). Subsequently (valid only when BRK is triggered), if a clock is present, OCy = OISy and OCyN = OISyN after the dead time, assuming that OISy and OISyN are not both set to a valid level for OCy and OCyN (otherwise a short circuit may result when driving the switch in a half-bridge configuration).	
			1	1	OCy = OISy and OCyN = OISyN after the dead time, assuming that OISy and OISyN are not both set to a valid level for OCy and OCyN (otherwise a short circuit may result when driving the switch in a half-bridge configuration). Caution: BRK2 can only be used when OSSI=OSSR=1.	

Caution: BRK2 can only be used when OSSl=OSSR=1.

Caution: If both outputs of a channel are unused (controlled by GPIO), the OISy, OISyN, CCyP, and CCyNP bits must remain a clear state.



14.3.4.9 Retriggerable single pulse mode

The retriggerable single pulse mode, in conjunction with the combined reset+trigger mode (SMS=8), allows the counter to be reset and activated by a trigger signal (TRGI) and generates pulses of programmable length, with a pulse width determined by the ARR.

Compared with single pulse mode, it has the following differences:

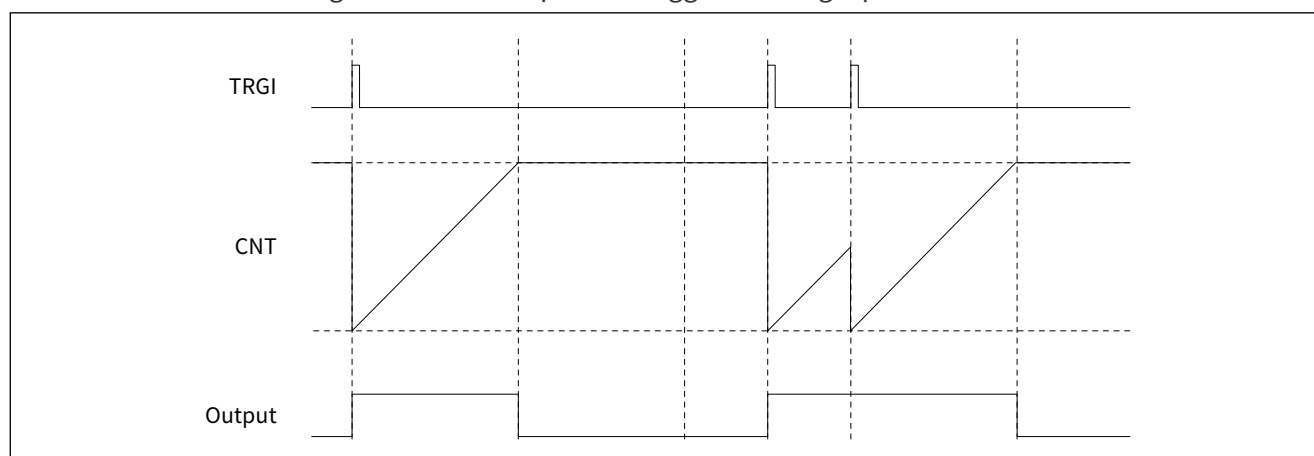
Table 14-14 Retriggerable single pulse mode and single pulse mode

Single pulse mode	Retriggerable single pulse mode
When a trigger occurs, a single pulse with a programmable pulse width is generated after a programmable delay.	When a trigger occurs, the pulse is generated immediately without programmable delay
New trigger invalidated	If the pulse generated by the previous trigger is not completed and a new trigger occurs, the pulse will be extended.

In this mode, counters can only be set to edge-aligned mode, not center-aligned mode. CCRy must be set to 0 when configured for up-count mode and greater than or equal to ARR when configured for down-count mode.

The following figure shows an example of retriggerable single pulse mode 2:

Figure 14-52 Example of retriggerable single pulse mode 2



14.3.4.10 Count direction output

Set the OCyM bit field of the ATIM_CCMRxCMP register to 0xB, which can output the count direction signal (a copy of the DIR bit in the ATIM_CR1 register) on the corresponding reference signal OCyREF.

This function can be used to monitor the count direction (or direction of rotation) in encoder mode, or to indicate up/down phase signals in center-aligned PWM mode.

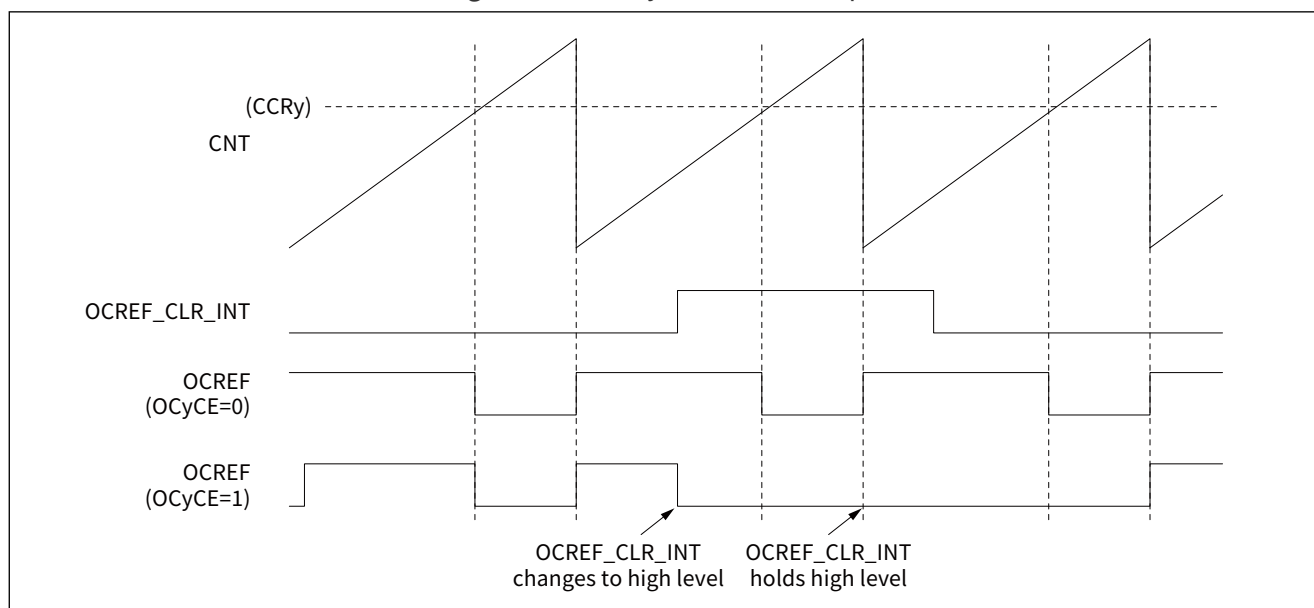
14.3.4.11 Reference signal clear function

Set the OCyCE bit field of the ATIM_CCMRxCMP register to 1, which enables the OCyREF signal clearing function of the corresponding channel, i.e., when a high level is detected on the OCREF_CLR_INT input, the OCyREF signal can be cleared to zero and the OCyREF signal will remain low until the next update event UEV occurs. This function is only available in output compare and PWM modes and is not available in forced output mode.

The clearing source OCREF_CLR_INT can be selected as either OCREF_CLR or ETRF, which is selected by the OCCS bit field of the slave mode control register ATIM_SMCR, and there are also multiple input sources for OCREF_CLR, see the description of the OCRSEL bit field in [14.9.31 ATIM_AF2 multiplexing function option register 2](#).

The following figure shows the OCyREF signal state when the OCREF_CLR_INT input goes high:

Figure 14-53 OCyREF clear example



14.3.5 UIF bit remap

Set the UIFREMAP bit in the ATIM_CR1 register to 1, which enables the UIF status bit remapping function, forcing the update interrupt flag ATIM_ISR.UIF to be copied consecutively into the UIFCPY bit field of the ATIM_CNT register. This allows automatically reading the counter value and the potential flip-flop condition issued by the UIFCPY flag. This simplifies the calculation by avoiding competing conditions when processing is shared between background tasks (counter reads) and interrupts (update interrupts).

There is no delay between UIF and UIFCPY flag enable.

When GTIMx_CR1.UIFREMAP is 0, the UIFCPY bit field is reserved and read as 0.

14.3.6 Timer cascade ITR

Timer cascading can be achieved by using the ITR signal, i.e. connecting the trigger output (TRGO) of the master timer to the ITR input of the slave timer, which in turn is connected to the trigger input (TRGI) of the slave timer. The ITR cascade can perform a variety of functions, such as using one timer as a prescaler for another timer, or the master timer performing a reset, enabling, or providing a clock to the slave timer's counter.

The source of the ATIM's cascade input ITR is the trigger output signals (TRGO) of the BTIM and GTIM, and the TXD/RXD signals of the UART, which can be selected by the TS bit field of the slave mode control register ATIM_SMCR, as configured in the table below:

Table 14-15 ITR signal source

ATIM_SMCR.TS bit field value	ITR signal name	ITR signal source
00000	ITR0	-
00001	ITR1	-
00010	ITR2	GTIM_Trgo
00011	ITR3	-
01000	ITR4	-
01001	ITR5	-
01010	ITR6	UART1_TXD
01011	ITR7	UART2_TXD
01100	ITR8	UART1_RXD
01101	ITR9	UART2_RXD
01110	ITR10	BTIM1_Trgo
01111	ITR11	BTIM2_Trgo
10000	ITR12	BTIM3_Trgo



14.3.7 On-chip peripheral interconnect ETR

The source of the ATIM's ETR signal can be the external ATIM_ETR/GTIM_ETR pins or other peripherals on-chip, which can be selected through the ETRSEL bit field of the multiplexing function option register ATIM_AF1.

When the ETR signal is input from the ETR pin, the specific external input port can be referred to [Table 8-2 GPIO multiplexing function assignment table](#), and needs to be multiplexed and configured through the GPIO multiplexing function registers (GPIOx_AFRH and GPIOx_AFRL); when the ETR signal is input from other peripheral devices on-chip, it can realise the on-chip and off-chip peripheral interconnection.

The following table shows the sources of ETR signals:

Table 14-16 ETR signal sources

AF1.ETRSEL bit field value	ETR signal source
0000	ATIM_ETR pin
0001	VC1_OUT
0010	VC2_OUT
0101	ADC_AWD
0111	HSE_FAULT
1000	LVD_OUT
1001	GTIM_ETR pin
1101	UART1_TXD
1110	UART2_TXD
1111	LSE_FAULT



14.4 ATIM interrupt

ATIM supports 16 interrupt sources. When a ATIM interrupt event occurs, the interrupt flag bit will be set by hardware, and an interrupt request will be generated if the corresponding interrupt enable control bit is set.

In the user ATIM interrupt service procedure, the relevant ATIM interrupt flag bit should be queried for corresponding processing, and the interrupt flag bit should be cleared before exiting the interrupt service procedure to avoid entering the interrupt service procedure repeatedly.

The following table shows the flag bit, interrupt enable bit, interrupt flag clear bit or clear method of each ATIM interrupt source:

Table 14-17 ATIM interrupt control

Interrupt event	Interrupt flag bit	Interrupt enable bit	Interrupt flag clear
Update interrupt	ISR.UIF	IER.UIE	Write 0 to ICR.UIF
Capture/compare 1 interrupt	ISR.CC1IF	IER.CC1IE	Write 0 to ICR.CC1IF or read CCR1 register
Capture/compare 2 interrupt	ISR.CC2IF	IER.CC2IE	Write 0 to ICR.CC2IF or read CCR2 register
Capture/compare 3 interrupt	ISR.CC3IF	IER.CC3IE	Write 0 to ICR.CC3IF or read CCR3 register
Capture/compare 4 interrupt	ISR.CC4IF	IER.CC4IE	Write 0 to ICR.CC4IF or read CCR4 register
Capture/compare 5 interrupt	ISR.CC5IF	IER.CC5IE	Write 0 to ICR.CC5IF or read CCR5 register
Capture/compare 6 interrupt	ISR.CC6IF	IER.CC6IE	Write 0 to ICR.CC6IF or read CCR6 register
COM interrupt	ISR.COMIF	IER.COMIE	Write 0 to ICR.COMIF
Trigger interrupt	ISR.TIF	IER.TIE	Write 0 to ICR.TIF
Brake interrupt	ISR.BIF	IER.BIE	Write 0 to ICR.BIF
Brake 2 interrupt	ISR.B2IF	IER.BIE	Write 0 to ICR.B2IF
System brake interrupt	ISR.SBIF	IER.BIE	Write 0 to ICR.SBIF
Encoder index interrupt	ISR.IDXF	IER.IDXIE	Write 0 to ICR.IDXF
Encoder direction change interrupt	ISR.DIRF	IER.DIRIE	Write 0 to ICR.DIRF
Encoder index error interrupt	ISR.IERRF	IER.IERRIE	Write 0 to ICR.IERRF
Encoder transition error interrupt	ISR.TERRF	IER.TERRIE	Write 0 to ICR.TERRF



14.5 Trigger ADC

The timer supports a variety of internal signals to trigger the startup of the ADC, such as capturing the compare signal and triggering the output signal TRGO and TRGO2. There are several possible events for triggering the output TRGO and TRGO2, which are selected by the MMS and MMS2 bit field of the control register ATIM_CR2. At the same time, the ADC peripherals need to configure their external trigger start registers to select the corresponding trigger source.

It should be noted that the ADC clock must be enabled before an event can be received from the main timer and that the ADC clock must not be changed in real time when the trigger signal is received from the timer.



14.6 Debugging support

ATIM supports stopping or continuing counting in debug mode, which is set up by the ATIM bit field of the debug state timer control register SYSCTRL_DEBUG:

- Set SYSCTRL_DEBUG.ATIM to 1, then the counter of ATIM is suspended in the debug state.
- Set SYSCTRL_DEBUG.ATIM to 0, then the counter of ATIM continues to count while in the debug state.



14.7 Programming examples

14.7.1 External clock mode 1 programming example

The following example configures the incremental counter to count on the rising edge of the ATIM_CH1 channel, with the following steps:

- Step 1: Set SYSCTRL_AHBEN.GPIOx to 1, SYSCTRL_APBEN1.ATIM to 1, enable the GPIO clock corresponding to the ATIM_CH1 pin and the ATIM configuration clock and operating clock;
- Step 2: Configure the GPIO corresponding to the ATIM_CH1 pin into the multiplexed input mode, for specific register configurations, please refer to [8 General-purpose input/output \(GPIO\)](#);
- Step 3: Set ATIM_TISEL1.TI1SEL to 0, select the TI1 source as the ATIM_CH1 channel;
- Step 4: Set ATIM_CCMR1CAP.IC1F to configure the input filter bandwidth;
- Step 5: Set ATIM_CCER.CC1NP to 0 and ATIM_CCER.CC1P to 0, select the rising edge count to be valid;
- Step 6: Set ATIM_SMCR.TS to 0x5, select the TRGI input as TI1FP1;
- Step 7: Set ATIM_SMCR.SMS to 0x7 to enable ATIM to operate in external clock mode 1;
- Step 8: Set ATIM_IER.UIE to 1 and configure the corresponding NVIC to enable the update interrupt;
- Step 9: Configure the counter prescaler ATIM_PSC;
- Step 10: Set the expected reload value ATIM_ARR;
- Step 11: Set ATIM_CR1.CEN to 1 to enable the counter;
- Step 12: When the counter overflows, the ATIM_ISR.UIF flag bit is set to 1 to enter the interrupt service procedure, and set ATIM_ICR.UIF to 0 to clear this interrupt flag.

14.7.2 External clock mode 2 programming example

The following is a programming example for external clock mode 2 + trigger mode, where the ATIM_CH1 input triggers to start the counter when a rising edge occurs, and the counter counts on each rising edge of the ATIM_ETR input signal, with the following steps:

- Step 1: Set SYSCTRL_AHBEN.GPIOx to 1 and SYSCTRL_APBEN1.ATIM to 1, enable the GPIO clocks corresponding to the ATIM_ETR and ATIM_CH1 pins and the ATIM configuration clock and operating clock;
- Step 2: Configure the GPIO corresponding to the ATIM_ETR and ATIM_CH1 pins into the multiplexed input mode, for specific register configurations, please refer to [8 General-purpose input/output \(GPIO\)](#);
- Step 3: Set ATIM_AF1.ETRSEL to 0, select ETR source as ATIM_ETR pin;
- Step 4: Set ATIM_SMCR.ETP to 0, select ETR rising edge valid;
- Step 5: Set ATIM_SMCR.ETPS to 0, close the prescaler of ETR;
- Step 6: Set ATIM_SMCR.ETF to 0, turn off ETR input filtering;
- Step 7: Set ATIM_SMCR.ECE to 1, enable external clock mode 2;
- Step 8: Set ATIM_TISEL1.TI1SEL to 0, select TI1 source as ATIM_CH1 channel;
- Step 9: Set ATIM_CCMR1CAP.IC1F to 0, turn off TI1 input filtering;
- Step 10: Set ATIM_CCER.CC1NP to 0 and ATIM_CCER.CC1P to 0, select detect TI1 rising edge;
- Step 11: Set ATIM_SMCR.TS to 0x5, select TRGI input to TI1FP1;
- Step 12: Set ATIM_SMCR.SMS to 0x6 to enable ATIM work in trigger mode;
- Step 13: Enabled when a rising edge occurs on the ATIM_CH1 input, the counter counts on each rising edge of the ATIM_ETR input signal.



14.7.3 Reset mode programming example

In the following example, the counter is reinitialised when a rising edge occurs on the ATIM_CH1 input, with the following steps:

- Step 1: Set SYSCTRL_AHBEN.GPIOx to 1 and SYSCTRL_APBEN1.ATIM to 1, enable the GPIO clocks corresponding to the ATIM_CH1 pin and the ATIM configuration clock and operating clock;
- Step 2: Configure the GPIO corresponding to the ATIM_CH1 pin into the multiplexed input mode, for specific register configurations, please refer to [8 General-purpose input/output \(GPIO\)](#);
- Step 3: Set ATIM_TISEL1.TI1SEL to 0, select TI1 source as ATIM_CH1 channel;
- Step 4: Set ATIM_CCMR1CAP.IC1F to configure input filter bandwidth;
- Step 5: Set ATIM_CCER.CC1NP to 0 and ATIM_CCER.CC1P to 0, select detect rising edge;
- Step 6: Set ATIM_SMCR.TS to 0x5, select TRGI input to TI1FP1;
- Step 7: Set ATIM_SMCR.SMS to 0x4 to enable ATIM to operate in reset mode;
- Step 8: Set the expected reload value ATIM_ARR;
- Step 9: Set ATIM_CR1.CEN to 1, enable counter, start normal counting;
- Step 10: When the ATIM_CH1 input has a rising edge, the counter is cleared and starts counting again from 0.

14.7.4 Gated mode programming example

In the following example, the signal input from the ATIM_CH1 pin is used as the gated signal to control the counter counting, with the following steps:

- Step 1: Set SYSCTRL_AHBEN.GPIOx to 1 and SYSCTRL_APBEN1.ATIM to 1, enable the GPIO clocks corresponding to the ATIM_CH1 pin and the ATIM configuration clock and operating clock;
- Step 2: Configure the GPIO corresponding to the ATIM_CH1 pin into the multiplexed input mode, for specific register configurations, please refer to [8 General-purpose input/output \(GPIO\)](#);
- Step 3: Set ATIM_TISEL1.TI1SEL to 0, select TI1 source as ATIM_CH1 channel;
- Step 4: Set ATIM_CCMR1CAP.IC1F to configure input filter bandwidth;
- Step 5: Set ATIM_CCER.CC1NP to 0, ATIM_CCER.CC1P to 0, select detect high level;
- Step 6: Set ATIM_SMCR.TS to 0x5, select TRGI input to TI1FP1;
- Step 7: Set ATIM_SMCR.SMS to 0x5 to enable ATIM to operate in gated mode;
- Step 8: Set the expected reload value ATIM_ARR;
- Step 9: Set ATIM_CR1.CEN to 1, enable counter;
- Step 10: The counter starts counting when the ATIM_CH1 input is high and stops counting when the ATIM_CH1 input is low.



14.7.5 Trigger mode programming example

In the following example, the counter is triggered to start when a rising edge occurs on the ATIM_CH1 input, with the following steps:

- Step 1: Set SYSCTRL_AHBEN.GPIOx to 1 and SYSCTRL_APBEN1.ATIM to 1, enable the GPIO clocks corresponding to the ATIM_CH1 pin and the ATIM configuration clock and operating clock;
- Step 2: Configure the GPIO corresponding to the ATIM_CH1 pin into the multiplexed input mode, for specific register configurations, please refer to [8 General-purpose input/output \(GPIO\)](#);
- Step 3: Set ATIM_TISEL1.TI1SEL to 0, select TI1 source as ATIM_CH1 channel;
- Step 4: Set ATIM_CCMR1CAP.IC1F to configure input filter bandwidth;
- Step 5: Set ATIM_CCER.CC1NP to 0 and ATIM_CCER.CC1P to 0, select detect rising edge;
- Step 6: Set ATIM_SMCR.TS to 0x5, select TRGI input to TI1FP1;
- Step 7: Set ATIM_SMCR.SMS to 0x6 to enable ATIM to operate in trigger mode;
- Step 8: Set the expected reload value ATIM_ARR;
- Step 9: When the ATIM_CH1 input has a rising edge, the counter starts counting and the TIF flag is set to 1.

14.7.6 Quadrature encoder mode programming example

In the following example, configure ATIM to work in quadrature encoder mode-x4 mode, with the following steps:

- Step 1: Set SYSCTRL_AHBEN.GPIOx to 1 and SYSCTRL_APBEN1.ATIM to 1, enable the GPIO clocks corresponding to the ATIM_CH1 and ATIM_CH2 pins and the ATIM configuration clock and operating clock;
- Step 2: Configure the GPIO corresponding to the ATIM_CH1 and ATIM_CH2 pins into the multiplexed input mode, for specific register configurations, please refer to [8 General-purpose input/output \(GPIO\)](#);
- Step 3: Set ATIM_TISEL1.TI1SEL to 0, select TI1 source as ATIM_CH1 channel;
- Step 4: Set ATIM_TISEL1.TI2SEL to 0, select TI2 source as ATIM_CH2 channel;
- Step 5: Set ATIM_CCMR1CAP.IC1F and ATIM_CCMR1CAP.IC2F to configure input filter bandwidth;
- Step 6: Set ATIM_CCER.CC1NP and ATIM_CCER.CC1P to 0, CH1 input not inverted;
- Step 7: Set ATIM_CCER.CC2NP, ATIM_CCER.CC2P to 0, CH2 input not inverted;
- Step 8: Set ATIM_SMCR.SMS to 0x3 to enable ATIM to operate in quadrature encoder mode-x4 mode;
- Step 9: Set the expected reload value ATIM_ARR;
- Step 10: Set ATIM_CR1.CEN to 1, enable counter.



14.7.7 Input capture programming example

14.7.7.1 Basic input capture mode

In the following example, execute an input capture when a rising edge occurs on the ATIM_CH1 input, with the following steps:

Step 1: Set `SYSCTRL_AHBEN.GPIOx` to 1 and `SYSCTRL_APBEN1.ATIM` to 1, enable the GPIO clocks corresponding to the ATIM_CH1 pin and the ATIM configuration clock and operating clock;

Step 2: Configure the GPIO corresponding to the ATIM_CH1 pin into the multiplexed input mode, for specific register configurations, please refer to [8 General-purpose input/output \(GPIO\)](#);

Step 3: Set `ATIM_TISEL1.TI1SEL` to 0, select TI1 source as ATIM_CH1 channel;

Step 4: Set `ATIM_CCMR1CAP.IC1F`, configure TI1 input filter bandwidth;

Step 5: Set `ATIM_CCER.CC1NP` to 0 and `ATIM_CCER.CC1P` to 0, select detect IC1 rising edge;

Step 6: Set `ATIM_CCMR1CAP.CC1S` to 1, CC1 channel configured as input, IC1 mapped to TI1;

Step 7: Set `ATIM_CCMR1CAP.IC1PSC` to 0, turn off IC1 prescaler;

Step 8: Set `ATIM_CCER.CC1E` to 1 to enable input capture 1 mode;

Step 9: Set `ATIM_IER.CC1IE` to 1 and configure the corresponding NVIC to enable input Capture 1 interrupt;

Step 10: Configure counter prescaler `ATIM_PSC`;

Step 11: Set the expected reload value `ATIM_ARR`;

Step 12: Set `ATIM_CR1.CEN` to 1, enable counter, start normal counting;

Step 13: When a capture occurs, the value of the current counter CNT is latched into the capture/compare register CCR1, completing a capture, while the `ATIM_ISR.CC1IF` flag bit is set to 1 to enter the interrupt service procedure, and this interrupt flag is cleared by setting `ATIM_ICR.CC1IF` to 0.



14.7.7.2 PWM input mode

The following example measures the period and duty cycle of the PWM signal input from the ATIM_CH1 pin, with the following steps:

- Step 1: Set `SYSCTRL_AHBEN.GPIOx` to 1 and `SYSCTRL_APBEN1.ATIM` to 1, enable the GPIO clocks corresponding to the ATIM_CH1 pin and the ATIM configuration clock and operating clock;
- Step 2: Configure the GPIO corresponding to the ATIM_CH1 pin into the multiplexed input mode, for specific register configurations, please refer to [8 General-purpose input/output \(GPIO\)](#);
- Step 3: Set `ATIM_TISEL1.TI1SEL` to 0, select TI1 source as ATIM_CH1 channel;
- Step 4: Set `ATIM_CCMR1CAP.IC1F`, configure TI1 input filter bandwidth;
- Step 5: Set `ATIM_CCER.CC1NP` to 0 and `ATIM_CCER.CC1P` to 0, select detect IC1 rising edge;
- Step 6: Set `ATIM_CCMR1CAP.CC1S` to 1, CC1 channel configured as input, IC1 mapped to TI1;
- Step 7: Set `ATIM_CCMR1CAP.IC1PSC` to 0, turn off IC1 prescaler;
- Step 8: Set `ATIM_CCER.CC1E` to 1 to enable input capture 1 mode;
- Step 9: Set `ATIM_IER.CC1IE` to 1 and configure the corresponding NVIC to enable input Capture 1 interrupt;
- Step 10: Set `ATIM_CCER.CC2NP` to 0, `ATIM_CCER.CC2P` to 1, select detect IC2 falling edge;
- Step 11: Set `ATIM_CCMR1CAP.CC2S` to 2, CC2 channel configured as input, IC2 mapped to TI1;
- Step 12: Set `ATIM_CCMR1CAP.IC2PSC` to 0, turn off IC2 prescaler;
- Step 13: Set `ATIM_CCER.CC2E` to 1 to enable input capture 2 mode;
- Step 14: Set `ATIM_SMCR.TS` to 0x5, select TRGI input to TI1FP1;
- Step 15: Set `ATIM_SMCR.SMS` to 0x4 to enable ATIM to operate in reset mode;
- Step 16: Configure counter prescaler `ATIM_PSC`;
- Step 17: Set the expected reload value `ATIM_ARR`;
- Step 18: Set `ATIM_CR1.CEN` to 1, enable counter, start normal counting.



14.7.8 Output compare programming example

In the following example, channel CH1 outputs the set waveform externally as follows:

- Step 1: Set `SYSCTRL_AHBEN.GPIOx` to 1 and `SYSCTRL_APBEN1.ATIM` to 1, enable the GPIO clocks corresponding to the `ATIM_CH1` pin and the ATIM configuration clock and operating clock;
- Step 2: Configure the GPIO corresponding to the `ATIM_CH1` pin into the multiplexed output mode, for specific register configurations, please refer to [8 General-purpose input/output \(GPIO\)](#);
- Step 3: Set `ATIM_CCER.CC1P` to 0, select OC1 output valid polarity;
- Step 4: Set `ATIM_CCMR1CMP.CC1S` to 0, CC1 channel configured as output;
- Step 5: Set `ATIM_CCMR1CMP.OC1M`, select output compare 1 mode;
- Step 6: Set `ATIM_CCER.CC1E` to 1 and output the OC1 signal on the corresponding output pin;
- Step 7: Set `ATIM_BDTR.MOE` to 1 to enable the main output;
- Step 8: Configure counter prescaler `ATIM_PSC`;
- Step 9: Set the expected reload value `ATIM_ARR`;
- Step 10: Set the expected compare value `ATIM_CCR1`;
- Step 11: Set `ATIM_IER.CC1IE` to 1 and configure corresponding NVIC, enable compare 1 interrupt;
- Step 12: Set `ATIM_CR1.CEN` to 1, enable counter;
- Step 13: When a comparison match occurs, the `ATIM_ISR.CC1IF` flag bit field is set to 1 to enter the interrupt service procedure, and this interrupt flag is cleared by setting `ATIM_ICR.CC1IF` to 0.



14.8 List of registers

ATIM base address: ATIM_BASE = 0x4000 1400

Table 14-18 List of ATIM registers

Register name	Register address	Register description
ATIM_CR1	ATIM_BASE + 0x00	Control register 1
ATIM_CR2	ATIM_BASE + 0x04	Control register 2
ATIM_SMCR	ATIM_BASE + 0x08	Slave mode control register
ATIM_IER	ATIM_BASE + 0x0C	Interrupt enable register
ATIM_ISR	ATIM_BASE + 0x10	Interrupt flag register
ATIM_ICR	ATIM_BASE + 0x70	Interrupt flag clear register
ATIM_EGR	ATIM_BASE + 0x14	Event generation register
ATIM_CCMR1CAP	ATIM_BASE + 0x18	Capture mode register 1
ATIM_CCMR1CMP	ATIM_BASE + 0x18	Compare mode register 1
ATIM_CCMR2CAP	ATIM_BASE + 0x1C	Capture mode register 2
ATIM_CCMR2CMP	ATIM_BASE + 0x1C	Compare mode register 2
ATIM_CCMR3CAP	ATIM_BASE + 0x50	Capture mode register 3
ATIM_CCMR3CMP	ATIM_BASE + 0x50	Compare mode register 3
ATIM_CCER	ATIM_BASE + 0x20	Capture/compare enable register
ATIM_CNT	ATIM_BASE + 0x24	Count register
ATIM_PSC	ATIM_BASE + 0x28	Prescaler register
ATIM_ARR	ATIM_BASE + 0x2C	Auto reload register
ATIM_RCR	ATIM_BASE + 0x30	Repeat count register
ATIM_CCR1	ATIM_BASE + 0x34	Capture/compare register 1
ATIM_CCR2	ATIM_BASE + 0x38	Capture/compare register 2
ATIM_CCR3	ATIM_BASE + 0x3C	Capture/compare register 3
ATIM_CCR4	ATIM_BASE + 0x40	Capture/compare register 4
ATIM_CCR5	ATIM_BASE + 0x48	Capture/compare register 5
ATIM_CCR6	ATIM_BASE + 0x4C	Capture/compare register 6
ATIM_BDTR	ATIM_BASE + 0x44	Brake and deadband register
ATIM_DTR2	ATIM_BASE + 0x54	Dead time register 2
ATIM_ECR	ATIM_BASE + 0x58	Encoder control register
ATIM_TISEL1	ATIM_BASE + 0x5C	TI input selection register 1
ATIM_TISEL2	ATIM_BASE + 0x6C	TI input selection register 2



Register name	Register address	Register description
ATIM_AF1	ATIM_BASE + 0x60	Multiplexing function option register 1
ATIM_AF2	ATIM_BASE + 0x64	Multiplexing function option register 2



14.9 Register descriptions

See section [1 Documentation conventions](#) for abbreviations used in register descriptions.

14.9.1 ATIM_CR1 Control register 1

Address offset: 0x00 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:12	RFU	-	Reserved bits, please keep the default value
11	UIFREMAP	RW	UIF status bit remap 0: Remapping is disabled, ATIM_CNT[31] remains 0 1: Remapping is enabled, ATIM_CNT[31] is equivalent to ATIM_ISR.UIF
10	RFU	-	Reserved bits, please keep the default value
9:8	CKD	RW	Configure the division ratio between PCLK clock and deadband and sampling clock (f_{DTS}) used by deadband generator and digital filter (ETR, Tly): 00: $f_{DTS} = f_{PCLK}/1$ 01: $f_{DTS} = f_{PCLK}/2$ 10: $f_{DTS} = f_{PCLK}/4$
7	ARPE	RW	Auto reload preload enable 0: Disable ARR register buffering 1: Enable ARR register buffering
6:5	CMS	RW	Counter alignment mode configuration 00: Edge-aligned mode: The counter is controlled by the CR1.DIR bit to count up or down. 01: Center-aligned mode 1: Counter alternately counts up and down. The output compare interrupt flag is set to 1 only when the counter counts down. 10: Center-aligned mode 2: Counter alternately counts up and down. The output compare interrupt flag is set to 1 only when the counter counts up. 11: Center-aligned mode 3: Counter alternately counts up and down. The output compare interrupt flag is set to 1 when either the counter counts up or down. Caution 1: Do not switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1). Caution 2: The initial count direction of the center-aligned mode is determined by the DIR bit field.
4	DIR	RW/RO	Count direction 0: Counter count up 1: Counter count down Caution: When the timer is configured in center aligned mode or encoder mode, this bit is read-only.



Bit field	Name	Permission	Function description
3	OPM	RW	Single pulse mode enable control 0: Disable single pulse mode 1: Enable single pulse mode, counter automatically stops when update event is received during running (set CEN to 0)
2	URS	RW	Update request source This bit is set to 1 and cleared to 0 by software to select the UEV event source. 0: When UEV is enabled, all of the following events generate an update interrupt. Including: - Counter overflow/underflow. - Set UG bit to 1. - Update events generated by the slave mode controller. 1: When UEV is enabled, only counter overflow/underflow generates an update interrupt.
1	UDIS	RW	Update disabled This bit is set to 1 and cleared to 0 by software to enable/disable UEV event generation. 0: UEV enabled. Update events (UEV) can be generated by one of the following events, which then updates the value of the shadow register. - Counter overflow/underflow. - Set UG bit to 1. - Update events generated by the slave mode controller. 1: UEV disabled. No update event is generated and the values of the respective shadow registers (ARR, PSC, and CCRy) remain unchanged. However, if the UG bit field is set to 1, or a hardware reset is received from the slave mode controller, the counter and prescaler are reinitialised.
0	CEN	RW	Counter enable 0: Counter disabled 1: Counter enabled <i>Caution 1: External clock, gated mode and encoder mode can only be used if the CEN bit field has been previously set to 1 by software. Trigger mode, on the other hand, automatically sets CEN position 1 via hardware.</i> <i>Caution 2: In single pulse mode, the CEN bit is automatically cleared to 0 when an update event occurs.</i>



14.9.2 ATIM_CR2 Control register 2

Address offset: 0x04 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:27	RFU	-	Reserved bits, please keep the default value
26:25	MMSH	RW	Master mode selection It is the highest 2 bits of the MMS and must be used with the MMS bit field.
24:20	MMS2	RW	Master mode selection 2 These bits select information to be sent to the ADC for synchronization (TRGO2). The combination of these bits is described in MMS bit field.
19	OIS6N	RW	Output idle state 6 (OC6N output), see OIS1N description
18	OIS6	RW	Output idle state 6 (OC6 output), see OIS1 description
17	OIS5N	RW	Output idle state 5 (OC5N output), see OIS1N description
16	OIS5	RW	Output idle state 5 (OC5 output), see OIS1 description
15	OIS4N	RW	Output idle state 4 (OC4N output), see OIS1N description
14	OIS4	RW	Output idle state 4 (OC4 output), see OIS1 description
13	OIS3N	RW	Output idle state 3 (OC3N output), see OIS1N description
12	OIS3	RW	Output idle state 3 (OC3 output), see OIS1 description
11	OIS2N	RW	Output idle state 2 (OC2N output), see OIS1N description
10	OIS2	RW	Output idle state 2 (OC2 output), see OIS1 description
9	OIS1N	RW	Output idle state 1 (OC1N output) 0: When MOE=0, OC1N=0 after dead time 1: When MOE=0, OC1N=1 after dead time Caution: This bit cannot be modified as long as LOCK (LOCK bit in ATIM_BDTR register) level 1, 2 or 3 is programmed.
8	OIS1	RW	Output idle state 1 (OC1 output) 0: When MOE=0, (after dead time if OC1N is valid) OC1=0 1: When MOE=0, (after dead time if OC1N is valid) OC1=1 Caution: This bit cannot be modified as long as LOCK (LOCK bit in ATIM_BDTR register) level 1, 2 or 3 is programmed.
7	TI1S	RW	TI1 input selection 0: ATIM_CH1 pin is connected to the TI1 input 1: XOR combination of ATIM_CH1, CH2 and CH3 pins are connected to TI1 input



Bit field	Name	Permission	Function description
6:4	MMS	RW	Master mode selection These bits select the information that will be sent to the slave timer for synchronisation in master mode (TRGO). Used in conjunction with MMSH, these bits are combined as follows: 00000: Reset - The UG bit in the ATIM_EGR register is used as the trigger output (TRGO). If the reset is generated by a trigger input (slave mode controller configured for reset mode), the signal on TRGO is delayed compared to the actual reset. 00001: Enable - The counter enable signal CNT_EN is used as a trigger output (TRGO). This trigger output can be used to start multiple timers at the same time, or to control the enabling of slave timers over a period of time. The counter enable signal consists of a logical sum operation combining the CEN control bit with the trigger input in gated mode. When the counter enable signal is controlled by a trigger input, there is a delay on the TRGO, except when master/slave mode is selected (please refer to the description of the MSM bit in 14.9.3 ATIM_SMCR slave mode control register). 00010: Update - Select the update event as the trigger output (TRGO). For example, the master timer can be used as a prescaler for the slave timer. 00011: Encoder count clock output 00100: CH1 output compare or input capture pulse 00101: CH2 output compare or input capture pulse 00110: CH3 output compare or input capture pulse 00111: CH4 output compare or input capture pulse 01000: CH5 output compare or input capture pulse 01001: CH6 output compare or input capture pulse 01010: OC1REF signal 01011: OC2REF signal 01100: OC3REF signal 01101: OC4REF signal 01110: OC5REF signal 01111: OC6REF signal 10000: OC1REFC signal 10001: OC2REFC signal 10010: OC3REFC signal 10011: OC4REFC signal 10100: OC5REFC signal 10101: OC6REFC signal 10110: OC3REFC rising edge or falling edge 10111: OC4REFC rising edge or falling edge 11000: OC5REFC rising edge or falling edge 11001: OC6REFC rising edge or falling edge



Bit field	Name	Permission	Function description
6:4 (continued)	MMS	RW	11010: OC4REFC rising edge or OC5REFC rising edge 11011: OC4REFC rising edge or OC5REFC falling edge 11100: OC4REFC rising edge or OC6REFC rising edge 11101: OC4REFC rising edge or OC6REFC falling edge 11110: OC5REFC rising edge or OC6REFC rising edge 11111: OC5REFC rising edge or OC6REFC falling edge Caution: The clock of the slave timer or ADC must be enabled before an event can be received from the master timer; and the clock of the slave timer or ADC must not be changed in real time when a trigger signal is received from the master timer.
3	RFU	-	Reserved bits, please keep the default value
2	CCUS	RW	Capture/compare control update selection 0: If the capture/compare control bits are preloaded (CCPC=1), these bits can be updated only by setting the COMG bit field to 1 1: If the capture/compare control bits are preloaded (CCPC=1), these bits can be updated by setting the COMG bit field to 1 or the rising edge of TRGI. Caution: This bit is only valid for channels with complementary outputs.
1	RFU	-	Reserved bits, please keep the default value
0	CCPC	RW	Capture/compare preload control 0: CCyE, CCyNE and OCyM bits are not preloaded. 1: The CCyE, CCyNE, and OCyM bits are preloaded. After these bits are written, they are updated only when a commutation event (COM) occurs (COMG bit field is set to 1 or a rising edge is detected on TRGI, depending on the CCUS bit). Caution: This bit is only valid for channels with complementary outputs.



14.9.3 ATIM_SMCR Slave mode control register

Address offset: 0x08 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:26	RFU	-	Reserved bits, please keep the default value
25	SMSPS	RW	Preload source This bit selects the trigger source for the transfer of the SMS[3:0] bit fields from the preload register to the valid registers. 0: Transfer is triggered by a timer update event. 1: Transfer is triggered by an index event.
24	SMSPE	RW	Preload enable This bit selects whether the SMS[3:0] bits are preloaded or not. 0: Preload disabled. 1: Preload enabled.
23:22	RFU	-	Reserved bits, please keep the default value
21:20	TSH	RW	Trigger selection It is the highest bit of the TS, and needs to be used in conjunction with the TS
19:17	RFU	-	Reserved bits, please keep the default value
16	SMSH	RW	Slave mode selection It is the highest bit of SMS, and needs to be used in conjunction with SMS.
15	ETP	RW	External trigger polarity This bit selects whether ETR or the inversion of ETR will be used for the trigger operation 0: ETR not inverted, high or rising edge is valid 1: ETR inverted, low level or falling edge is valid
14	ECE	RW	External clock enabled This bit can enable external clock mode 2 0: Disable external clock mode 2 1: Enable external clock mode 2, the counter clock is provided by any valid edge of the ETRF signal. <i>Caution 1: Set the ECE bit field to 1 which has the same effect as selecting external clock mode 1 and connecting TRGI to ETRF (SMS=111 and TS=00111).</i> <i>Caution 2: External clock mode 2 can be used in conjunction with the following slave modes: reset mode, gated mode and trigger mode. However, in these cases TRGI must not be connected to ETRF (TS bit must not be 00111).</i> <i>Caution 3: If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.</i>



Bit field	Name	Permission	Function description
13:12	ETPS	RW	External trigger prescaler The external trigger signal ETRP frequency must not exceed 1/4 of the PCLK frequency. The ETRP frequency can be reduced by enabling the prescaler, and this method is useful when a fast external clock is input. 00: Prescaler disabled 01: 2-division ETRP frequency 10: 4-division ETRP frequency 11: 8-division ETRP frequency
11:8	ETF	RW	External trigger filter This bit field defines the sampling frequency of the ETRP signal and the digital filter bandwidth applicable to the ETRP. The digital filter consists of an event counter, where only every N consecutive events are considered as a valid output edge: 0000: No filter, sampled at frequency f_{DTS} 0001: $f_{SAMPLING}=f_{DTS}/1$, $N=2$ 0010: $f_{SAMPLING}=f_{DTS}/1$, $N=4$ 0011: $f_{SAMPLING}=f_{DTS}/1$, $N=8$ 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: $f_{SAMPLING}=f_{DTS}/32$, $N=8$
7	MSM	RW	Master/slave mode 0: No action is performed 1: The action of the trigger input event (TRGI) of the current timer is postponed to allow perfect synchronisation (via TRGO) of the current timer with its slave timers. This setting applies when multiple timers are synchronised by a single external event.



Bit field	Name	Permission	Function description
6:4	TS	RW	Trigger input TRGI selection This bit field selects the trigger input that will be used to synchronise the counter. 00000: Internal trigger 0 (ITR0) 00001: Internal trigger 1 (ITR1) 00010: Internal trigger 2 (ITR2) 00011: Internal trigger 3 (ITR3) 00100: TI1 edge detector (TI1F_ED) 00101: Filtered timer input 1 (TI1FP1) 00110: Filtered timer input 2 (TI2FP2) 00111: External trigger input (ETRF) 01000: Internal trigger 4 (ITR4) 01001: Internal trigger 5 (ITR5) 01010: Internal trigger 6 (ITR6) 01011: Internal trigger 7 (ITR7) 01100: Internal trigger 8 (ITR8) 01101: Internal trigger 9 (ITR9) 01110: Internal trigger 10 (ITR10) 01111: Internal trigger 11 (ITR11) 10000: Internal trigger 12 (ITR12) Other values: Reserved Caution 1: These bits can only be changed if they are not used (e.g. SMS=0000) to avoid incorrect edge detection during transition. Caution 2: The other bit fields are in bits 21 and 20 of the same register and need to be used in conjunction with the TS. Caution 3: For specific ITRx sources, please refer to Table 14-15 ITR signal source.
3	OCCS	RW	OCREF cleared selection This bit is used to select the OCREF clear source. 0: OCREF_CLR_INT connected to OCREF_CLR input. 1: OCREF_CLR_INT connected to ETRF. Caution: OCREF_CLR input source please refer to 14.9.31 ATIM_AF2 multiplexing function option register 2 for the description of the OCRSEL bit field.
2:0	SMS	RW	Slave mode selection When an external signal is selected, the valid edge of the trigger signal (TRGI) is related to the polarity selected on the external input (see the input related control registers). 0000: Slave mode disabled - If CEN = '1', the prescaler clock is supplied directly from the internal clock. 0001: Quadrature encoder mode - x2 mode, the counter counts up/down on the TI1FP1 edge according to the TI2FP2 level.



Bit field	Name	Permission	Function description
2:0 (continued)	SMS	RW	0010: Quadrature encoder mode - x2 mode, the counter counts up/down according to the TI1FP1 level at the TI2FP2 edge. 0011: Quadrature encoder mode - x4 mode, the counter counts on the TI1FP1 and TI2FP2 edges, the direction of the count depends on the level of the other input. 0100: Reset mode - The counter is reinitialised and a register update event is generated on the occurrence of a rising edge of the selected trigger input (TRGI). 0101: Gated mode - Enables the counter clock when the trigger input (TRGI) is high. The counter stops counting (but not resetting) as soon as the trigger input goes low. Counter start and stop are controlled. 0110: Trigger mode - Starts the counter (but does not reset) on a rising edge of the trigger signal (TRGI). Only the start of the counter is controlled. 0111: External clock mode 1 - Counter clock is provided by the rising edge of the selected trigger signal (TRGI). 1000: Combined reset + trigger mode - Reinitialises the counter on the rising edge of the selected trigger input (TRGI), generates a register update event and starts the counter. 1001: Combined gated + reset mode - The counter is enabled and starts counting when the trigger input (TRGI) goes high, the counter stops counting and is reset when the trigger input goes low, in this mode the start and stop of the counter are controlled. 1010: Encoder mode - Clock + direction, x2 mode. 1011: Encoder mode - Clock+direction, x1 mode, TI2FP2's edge polarity is set by CC2P. 1100: Encoder mode - Clock with direction, x2 mode. 1101: Encoder mode - Clock with direction, x1 mode, TI1FP1 and TI2FP2 edge polarity is set by CC1P and CC2P. 1110: Quadrature encoder mode - x1 mode, counts only TI1FP1 edges, whose edge polarity is set by CC1P. 1111: Quadrature encoder mode - x1 mode, counts only TI2FP2 edges, whose polarity is set by CC2P. Caution 1: If TI1F_ED is selected as the trigger input (TS=00100), the gated mode must not be used. In fact, TI1F_ED outputs one pulse for each TI1F transition, while the gated mode checks the level of the trigger signal. Caution 2: The clock of the slave peripheral (timer, ADC, etc.) that receives the TRGO signal must be enabled before the event can be received from the master timer; and the clock frequency must not be changed in real time (prescaler) when the trigger signal is received from the master timer.



14.9.4 ATIM_IER Interrupt enable register

Address offset: 0x0C Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:24	RFU	-	Reserved bits, please keep the default value
23	TERRIE	RW	Encoder transition error interrupt enable control 0: Disable transition error interrupt 1: Enable transition error interrupt
22	IERRIE	RW	Encoder index error interrupt enable control 0: Disable index error interrupt 1: Enable index error interrupt
21	DIRIE	RW	Encoder direction change interrupt enabled 0: Disable direction change interrupt 1: Enable direction change interrupt
20	IDXIE	RW	Encoder index interrupt enable control 0: Disable index interrupt 1: Enable index interrupt
19:18	RFU	-	Reserved bits, please keep the default value
17	CC6IE	RW	Capture/compare 6 interrupt enabled 0: Disable capture/compare 6 interrupt 1: Enable capture/compare 6 interrupt
16	CC5IE	RW	Capture/compare 5 interrupt enabled 0: Disable capture/compare 5 interrupt 1: Enable capture/compare 5 interrupt
15:8	RFU	-	Reserved bits, please keep the default value
7	BIE	RW	Brake interrupt enabled 0: Disable brake interrupt 1: Enable brake interrupt
6	TIE	RW	Trigger interrupt enabled 0: Disable trigger interrupt 1: Enable trigger interrupt
5	COMIE	RW	COM interrupt enabled 0: Disable COM interrupt 1: Enable COM interrupt
4	CC4IE	RW	Capture/compare 4 interrupt enabled 0: Disable capture/compare 4 interrupt 1: Enable capture/compare 4 interrupt
3	CC3IE	RW	Capture/compare 3 interrupt enabled 0: Disable capture/compare 3 interrupt 1: Enable capture/compare 3 interrupt



Bit field	Name	Permission	Function description
2	CC2IE	RW	Capture/compare 2 interrupt enabled 0: Disable capture/compare 2 interrupt 1: Enable capture/compare 2 interrupt
1	CC1IE	RW	Capture/compare 1 interrupt enabled 0: Disable capture/compare 1 interrupt 1: Enable capture/compare 1 interrupt
0	UIE	RW	Update interrupt enabled 0: Disable update interrupt 1: Enable update interrupt



14.9.5 ATIM_ISR Interrupt flag register

Address offset: 0x10 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:24	RFU	-	Reserved bits, please keep the default value
23	TERRF	RO	Encoder transition error interrupt flag 0: No transition error occurred 1: A transition error has occurred
22	IERRF	RO	Encoder index error interrupt flag 0: No index error occurred 1: An indexing error has occurred
21	DIRF	RO	Encoder direction change interrupt flag 0: No encoding direction change has occurred 1: Encoding direction change has occurred
20	IDXF	RO	Encoder index interrupt flag 0: Index flag has not occurred 1: Index flag has occurred
19	CC6OF	RO	Capture/compare 6 repeat capture flag, see CC10F description
18	CC5OF	RO	Capture/compare 5 repeat capture flag, see CC10F description
17	CC6IF	RO	Capture/compare 6 interrupt flag, see CC1IF description
16	CC5IF	RO	Capture/compare 5 interrupt flag, see CC1IF description
15:14	RFU	-	Reserved bits, please keep the default value
13	SBIF	RO	System brake interrupt flag This flag is set by hardware to 1 whenever the system brake input becomes valid, and can be cleared by software when the system brake input becomes invalid. This flag must be reset in order for the PWM to restart operation. 0: No system brake event has occurred. 1: A valid level is detected on the system brake input. If BIE=1 in the ATIM_IER register, an interrupt is generated.
12	CC4OF	RO	Capture/compare 4 repeat capture flag, see CC10F description
11	CC3OF	RO	Capture/compare 3 repeat capture flag, see CC10F description
10	CC2OF	RO	Capture/compare 2 repeat capture flag, see CC10F description



Bit field	Name	Permission	Function description
9	CC10F	RO	Capture/compare 1 repeat capture flag This flag bit is set to 1 by hardware only when the corresponding channel is configured in input capture mode. 0: No repeat capture detected 1: Counter value has been captured in the ATIM_CCR1 register and the CC1IF flag has been set to 1
8	B2IF	RO	Brake 2 interrupt flag This flag is set by hardware to 1 whenever the brake 2 input becomes valid, and can be cleared by software after the brake 2 input becomes invalid. 0: No brake 2 event has occurred. 1: A valid level is detected on the brake 2 input. If BIE=1 in the ATIM_IER register, an interrupt is generated.
7	BIF	RO	Brake interrupt flag This flag is set by hardware to 1 whenever the brake input becomes valid, and can be cleared by software after the brake input becomes invalid. 0: No brake event has occurred. 1: A valid level is detected on the brake input. If BIE=1 in the ATIM_IER register, an interrupt is generated.
6	TIF	RO	Trigger interrupt flag In all modes except gated mode, this flag will be set by hardware to 1 when a valid edge is detected on the TRGI input after enabling the slave mode controller. When gated mode is selected, this flag will be set to 1 when the counter is started and stopped. This flag needs to be cleared by software. 0: No trigger event occurred 1: Trigger interrupt pending
5	COMIF	RO	COM interrupt flag This flag is set to 1 by hardware when a COM event occurs (when the capture/compare control bits CCyE, CCyNE, and OCyM have been updated), but needs to be cleared by software. 0: COM event has not occurred 1: COM interrupt pending
4	CC4IF	RO	Capture/compare 4 interrupt flag, see CC1IF description
3	CC3IF	RO	Capture/compare 3 interrupt flag, see CC1IF description
2	CC2IF	RO	Capture/compare 2 interrupt flag, see CC1IF description



Bit field	Name	Permission	Function description
1	CC1IF	RO	Capture/compare 1 interrupt flag If channel CC1 is configured to output: This flag is set by hardware when the counter matches the compare value, except in center-aligned mode (see 14.9.1 ATIM_CR1 controls register 1 for a description of the CMS bit). But it needs to be cleared by software. 0: No match 1: The value of the ATIM_CNT counter matches the value of the ATIM_CCR1 register. When the value of ATIM_CCR1 is greater than the value of ATIM_ARR, the CC1IF bit will go high when the counter overflows (in count up mode and center-aligned mode) or underflows (in count down mode). If channel CC1 is configured as an input: This bit is set by hardware when a capture event occurs. Clear this bit by software or by reading the ATIM_CCR1 register. 0: No input capture event occurred 1: Counter value captured in ATIM_CCR1 register (edge matching selected polarity detected on IC1)
0	UIF	RO	Update interrupt flag This bit is set to 1 by hardware when an update event occurs. But needs to be cleared by software. 0: No updates occurred 1: Update interrupt pending This bit is set to 1 by hardware when the register is updated when: - UDIS=0 in the ATIM_CR1 register and the repeat counter value underflows (updated when repeat counter= 0). - URS=0 and UDIS=0 in the ATIM_CR1 register and CNT is reinitialized by software using the UG bit in the ATIM_EGR register. - URS=0 and UDIS=0 in the ATIM_CR1 register and CNT is reinitialized by a trigger event.



14.9.6 ATIM_ICR Interrupt flag clear register

Address offset: 0x70 Reset value: 0x00FF 3FFF

Bit field	Name	Permission	Function description
31:24	RFU	-	Reserved bits, please keep the default value
23	TERRF	R1W0	Encoder transition error interrupt flag cleared 0: Clear encoder transition error interrupt flag 1: No function
22	IERRF	R1W0	Encoder index error interrupt flag cleared 0: Clear encoder index error interrupt flag 1: No function
21	DIRF	R1W0	Encoder direction change interrupt flag cleared 0: Clear encoder direction change interrupt flag 1: No function
20	IDXF	R1W0	Encoder index interrupt flag cleared 0: Clear encoder index interrupt flag 1: No function
19	CC6OF	R1W0	Capture/compare 6 repeat capture flag cleared, see CC1OF description
18	CC5OF	R1W0	Capture/compare 5 repeat capture flag cleared, see CC1OF description
17	CC6IF	R1W0	Capture/compare 6 interrupt flag cleared, see CC1IF description
16	CC5IF	R1W0	Capture/compare 5 interrupt flag cleared, see CC1IF description
15:14	RFU	-	Reserved bits, please keep the default value
13	SBIF	R1W0	System brake interrupt flag cleared 0: Clear system brake interrupt flag 1: No function
12	CC4OF	R1W0	Capture/compare 4 repeat capture flag cleared, see CC1OF description
11	CC3OF	R1W0	Capture/compare 3 repeat capture flag cleared, see CC1OF description
10	CC2OF	R1W0	Capture/compare 2 repeat capture flag cleared, see CC1OF description
9	CC1OF	R1W0	Capture/compare 1 repeat capture flag cleared 0: Clear capture/compare 1 repeat capture flag 1: No function
8	B2IF	R1W0	Brake 2 interrupt flag cleared 0: Clear brake 2 interrupt flag 1: No function



Bit field	Name	Permission	Function description
7	BIF	R1W0	Brake break flag cleared 0: Clear brake interrupt flag 1: No function
6	TIF	R1W0	Trigger interrupt flag cleared 0: Clear trigger interrupt flag 1: No function
5	COMIF	R1W0	COM interrupt flag cleared 0: Clear COM interrupt flag 1: No function
4	CC4IF	R1W0	Capture/compare 4 interrupt flag cleared, see CC1IF description
3	CC3IF	R1W0	Capture/compare 3 interrupt flag cleared, see CC1IF description
2	CC2IF	R1W0	Capture/compare 2 interrupt flag cleared, see CC1IF description
1	CC1IF	R1W0	Capture/compare 1 interrupt flag cleared 0: Clear capture/compare 1 interrupt flag 1: No function
0	UIF	R1W0	Update interrupt flag cleared 0: Clear update interrupt flag 1: No function



14.9.7 ATIM_EGR Event generation register

Address offset: 0x14 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:18	RFU	-	Reserved bits, please keep the default value
17	CC6G	RW	Capture/compare 6 generation, see CC1G description
16	CC5G	RW	Capture/compare 5 generation, see CC1G description
15:9	RFU	-	Reserved bits, please keep the default value
8	B2G	RW	Brake 2 generated. This bit is set to 1 by software to generate an event and is automatically cleared by hardware. 0: No operation is performed 1: Brake 2 event is generated. The MOE bit is cleared and the B2IF flag is set to 1. Correlation interrupts can occur when enabled.
7	BG	RW	Brake generated This bit is set to 1 by software to generate an event and is automatically cleared by hardware. 0: No operation is performed 1: Brake event is generated. The MOE bit is cleared and the BIF flag is set to 1. Correlation interrupts can occur when enabled.
6	TG	RW	Trigger generated This bit is set to 1 by software to generate an event and is automatically cleared by hardware. 0: No operation is performed 1: The TIF flag in ATIM_ISR register is set to 1. When enabled, the associated interrupt event can occur.
5	COMG	RW	Capture/compare control update generated This bit can be set to 1 by software and automatically cleared by hardware 0: No operation is performed 1: CCyE, CCyNE and OCyM bits can be updated when the CCPC bit field is set to 1 Caution: This bit is valid only for channels with complementary outputs.
4	CC4G	RW	Capture/compare 4 generation, see CC1G description
3	CC3G	RW	Capture/compare 3 generation, see CC1G description
2	CC2G	RW	Capture/compare 2 generation, see CC1G description



Bit field	Name	Permission	Function description
1	CC1G	RW	Capture/compare 1 generated This bit is set to 1 by software to generate an event and is automatically cleared by hardware. 0: No operation is performed 1: Capture/compare events are generated on channel 1 If channel CC1 is configured as output: - When enabled, the CC1IF flag is set to 1 and the corresponding interrupt request is sent. If channel CC1 is configured as input: - The current value of the counter will be captured in the ATIM_CCR1 register. When enabled, the CC1IF flag is set to 1 and the appropriate interrupt request is sent. If the CC1IF flag is already high, the CC1OF flag will be set to 1. <i>Caution: CC1E must be enabled, otherwise CCR1 register will capture the current value of the counter, but no interrupt flag will be generated, and no interrupt can be entered.</i>
0	UG	RW	Update generated This bit can be set to 1 by software and automatically cleared by hardware. 0: No operation is performed 1: Reinitialise the counter and generate a register update event. Note that the prescaler counter will also be cleared (but the prescaler ratio is not affected). The counter will be cleared if center-aligned mode or count up mode is selected; if down-count mode is selected, the counter will use the auto-reload value ATIM_ARR.

14.9.8 ATIM_CCMR1CAP Capture mode register 1

Address offset: 0x18 Reset value: 0x0000 0000

This register is the same register as ATIM_CCMR1CMP and can be used in input capture mode or output compare mode. The channel direction is defined by configuring the corresponding CCyS bit. All other bits of this register function differently in input capture mode and output compare mode.

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:12	IC2F	RW	Input capture 2 filter, see IC1F description
11:10	IC2PSC	RW	Input capture 2 prescaler, see IC1PSC description



Bit field	Name	Permission	Function description
9:8	CC2S	RW	Capture/compare 2 selection This bit field defines the channel direction (input/output) and the input mapping used. 00: CC2 channel configured as output 01: CC2 channel configured as input, IC2 mapped to TI2 10: CC2 channel configured as input, IC2 mapped to TI1 11: CC2 channel configured as input, IC2 mapped to TRC this mode is only valid when the internal trigger input is selected via the TS bit (ATIM_SMCR register) Caution: Data can be written to the CC2S bit only when the channel is closed (CC2E=0 in ATIM_CCER).
7:4	IC1F	RW	Input capture 1 filter This bit field defines the sampling frequency of the TI1 input and the sample length of the digital filter applicable to TI1. The digital filter consists of an event counter, where every N consecutive events are considered to be only one valid output edge. 0000: No filter, sampled at frequency f_{DTS} 0001: $f_{SAMPLING}=f_{PCLK}$, N=2 1001: $f_{SAMPLING}=f_{DTS}/8$, N=8 0010: $f_{SAMPLING}=f_{PCLK}$, N=4 1010: $f_{SAMPLING}=f_{DTS}/16$, N=5 0011: $f_{SAMPLING}=f_{PCLK}$, N=8 1011: $f_{SAMPLING}=f_{DTS}/16$, N=6 0100: $f_{SAMPLING}=f_{DTS}/2$, N=6 1100: $f_{SAMPLING}=f_{DTS}/16$, N=8 0101: $f_{SAMPLING}=f_{DTS}/2$, N=8 1101: $f_{SAMPLING}=f_{DTS}/32$, N=5 0110: $f_{SAMPLING}=f_{DTS}/4$, N=6 1110: $f_{SAMPLING}=f_{DTS}/32$, N=6 0111: $f_{SAMPLING}=f_{DTS}/4$, N=8 1111: $f_{SAMPLING}=f_{DTS}/32$, N=8 1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
3:2	IC1PSC	RW	Input capture 1 prescaler This bit field defines the prescaler ratio for the CC1 input (IC1). The prescaler is reset as soon as CC1E=0 (ATIM_CCER register). 00: No prescaler, capture is performed every time an edge is detected on the capture input. 01: Capture is performed every 2 events. 10: Capture is performed every 4 events. 11: Capture is performed every 8 events.
1:0	CC1S	RW	Capture/compare 1 selection This bit field defines the channel direction (input/output) and the input mapping used. 00: CC1 channel configured as output 01: CC1 channel configured as input, IC1 mapped to TI1 10: CC1 channel configured as input, IC1 mapped to TI2 11: CC1 channel configured as input, IC1 mapped to TRC, this mode is only valid when the internal trigger input is selected via the TS bit (ATIM_SMCR register) Caution: Data can be written to the CC1S bit only when the channel is closed (CC1E=0 in ATIM_CCER).



14.9.9 ATIM_CCMR1CMP Compare mode register 1

Address offset: 0x18 Reset value: 0x0000 0000

This register is the same register as ATIM_CCMR1CAP and can be used in input capture mode or output compare mode. The channel direction is defined by configuring the corresponding CCyS bit. All other bits of this register function differently in input capture mode and output compare mode.

Bit field	Name	Permission	Function description
31:25	RFU	-	Reserved bits, please keep the default value
24	OC2MH	RW	Used in conjunction with OC2M, it is the highest bit of OC2M
23:17	RFU	-	Reserved bits, please keep the default value
16	OC1MH	RW	Used in conjunction with OC1M, it is the highest bit of OC1M
15	OC2CE	RW	Output compare 2 clear enabled, see OC1CE description
14:12	OC2M	RW	Output compare 2 mode, see OC1M description
11	OC2PE	RW	Output compare 2 preload enabled, see OC1PE description
10	OC2FE	RW	Output compare 2 fast enabled, see OC1FE description
9:8	CC2S	RW	Capture/compare 2 selection This bit field defines the channel direction (input/output) and the input mapping used. 00: CC2 channel configured as output 01: CC2 channel configured as input, IC2 mapped to TI2 10: CC2 channel configured as input, IC2 mapped to TI1 11: CC2 channel configured as input, IC2 mapped to TRC, this mode is only valid when the internal trigger input is selected via the TS bit (ATIM_SMCR register) Caution: Data can be written to the CC2S bit only when the channel is closed (CC2E=0 in ATIM_CCER).
7	OC1CE	RW	Output compare 1 clear enabled 0: OC1REF is not affected by the OCREF_CLR_INT input 1: OC1REF is immediately cleared when a high level is detected on the OCREF_CLR_INT input
6:4	OC1M	RW	Output compare 1 mode These bits define the behaviour of the output reference signal OC1REF which provides OC1 and OC1N. OC1REF is valid at high level, while the valid level of OC1 and OC1N depends on the CC1P and CC1NP bit. 0000: Freeze - Comparison of output compare register ATIM_CCR1 with counter ATIM_CNT will not have any effect on the output. (This mode is used for time base generation). 0001: Set channel 1 to output a valid level when matched. The OC1REF signal is forced to high level when counter ATIM_CNT is matched with capture/compare register 1 (ATIM_CCR1). 0010: Set channel 1 to output an invalid level when matched. When counter ATIM_CNT matches capture/compare register 1 (ATIM_CCR1), the OC1REF signal is forced to low level.



6:4 (continued)	OC1M	RW	0011: Flipped - OC1REF is flipped when ATIM_CNT=ATIM_CCR1. 0100: Forced to invalid level - OC1REF is forced to low level. 0101: Forced to valid level - OC1REF is forced to high level. 0110: PWM mode 1 - In count up mode, channel 1 is valid (OC1REF=1) as long as ATIM_CNT < ATIM_CCR1, otherwise it is invalid (OC1REF=0). In count down mode, channel 1 is invalid (OC1REF=0) as long as ATIM_CNT > ATIM_CCR1, otherwise it is valid (OC1REF=1). 0111: PWM mode 2 - In count up mode, channel 1 is invalid (OC1REF=0) whenever ATIM_CNT < ATIM_CCR1, otherwise it is valid (OC1REF=1). In count down mode, channel 1 is valid (OC1REF=1) as long as ATIM_CNT > ATIM_CCR1, otherwise it is invalid (OC1REF=0). 1000: Retriggerable OPM mode 1 - In count up mode, the channel is in valid state until a trigger event is detected (on the TRGI signal). Then, a comparison is made in PWM mode 1 and the channel will become valid again at the next update. In count down mode, the channel is invalid until a trigger event is detected (on the TRGI signal). Then, a comparison is made in PWM mode 1 and the channel becomes invalid again at the next update. 1001: Retriggerable OPM mode 2 - In count up mode, the channel is invalid until a trigger event is detected (on the TRGI signal). Then, a comparison is made in PWM mode 2 and the channel becomes invalid again at the next update. In count down mode, the channel is valid until a trigger event is detected (on the TRGI signal). Then, the comparison is performed in PWM mode 2 and the channel becomes valid again at the next update. 1010: Reserved 1011: Count direction output, OC1REF signal is the count direction signal for the DIR bit. 1100: Combined PWM Mode 1 - OC1REF behaves the same as in PWM Mode 1. OC1REFC is the logical OR of OC1REF and OC2REF. 1101: Combined PWM Mode 2 - OC1REF behaves the same as in PWM Mode 2. OC1REFC is the logical AND of OC1REF and OC2REF. 1110: Asymmetric PWM mode 1 - OC1REF behaves the same as in PWM Mode 1. OC1REFC outputs OC1REF when the counter counts up and OC2REF when the counter counts down. 1111: Asymmetric PWM mode 2 - OC1REF behaves the same as in PWM Mode 2. OC1REFC outputs OC1REF when the counter counts up and OC2REF when the counter counts down. Caution 1: These bits cannot be modified as long as LOCK (LOCK bit in ATIM_BDTR register) level 3 is programmed and CC1S="00"(channel configured as output). Caution 2: In PWM mode, the OCyREF level changes only when the comparison result changes or when the output comparison mode is switched from Freeze mode to PWM mode. Caution 3: This bit field will be preloaded on channels with complementary outputs. If the CCPC bit in the ATIM_CR2 register is 1, the OC1M valid bit will get a new value from the preload bit only when a COM event is generated.



3	OC1PE	RW	Output compare 1 preload enabled 0: Disable the preload register related to ATIM_CCR1. Data can be written to ATIM_CCR1 at any time and the new value will be used immediately after writing. 1: Enable the preload register related to ATIM_CCR1. Read/write access to the preloaded registers. The ATIM_CCR1 preloaded value is loaded into the valid registers each time that an update event is generated. Caution 1: These bits cannot be modified as long as LOCK (LOCK bit in ATIM_BDTR register) level 3 is programmed and CC1S="00"(channel configured as output). Caution 2: PWM mode without validation of the preload register is only possible in single pulse mode (OPM bit field in the ATIM_CR1 register is set to 1). The behaviour is not guaranteed in other cases.
2	OC1FE	RW	Output compare 1 fast enabled This bit is used to speed up the effect of a trigger input event on the CC output. 0: Even if the trigger is on, CC1 will work properly according to the counter and CCR1 values. When an edge is present on the trigger input, the minimum delay to activate the CC1 output is 5 clock cycles. 1: The presence of a valid edge on the trigger input corresponds to a comparison match on the CC1 output. Subsequently, OC is set to the compare level regardless of the comparison result. The delay time between sampling the trigger input and activating the CC1 output is reduced to 3 clock cycles. OCyFE works only when the channel is configured for PWM1 or PWM2 mode.
1:0	CC1S	RW	Capture/compare 1 selection This bit field defines the channel direction (input/output) and the input mapping used. 00: CC1 channel configured as output 01: CC1 channel configured as input, IC1 mapped to TI1 10: CC1 channel configured as input, IC1 mapped to TI2 11: CC1 channel configured as input, IC1 mapped to TRC, this mode is only valid when the internal trigger input is selected via the TS bit (ATIM_SMCR register) Caution: Data can be written to the CC1S bit only when the channel is closed (CC1E=0 in ATIM_CCER).



14.9.10 ATIM_CCMR2CAP Capture mode register 2

Address offset: 0x1C Reset value: 0x0000 0000

This register is the same register as ATIM_CCMR2CMP and can be used in input capture mode or output compare mode. The channel direction is defined by configuring the corresponding CCyS bit. All other bits of this register function differently in input capture mode and output compare mode.

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:12	IC4F	RW	Input capture 4 filter, see IC1F description
11:10	IC4PSC	RW	Input capture 4 prescaler, see IC1PSC description
9:8	CC4S	RW	Capture/compare 4 selection This bit field defines the channel direction (input/output) and the input mapping used. 00: CC4 channel configured as output 01: CC4 channel configured as input, IC4 mapped to TI4 10: CC4 channel configured as input, IC4 mapped to TI3 11: CC4 channel configured as input, IC4 mapped to TRC, this mode is only valid when internal trigger input is selected via the TS bit (ATIM_SMCR register) Caution: Data can be written to the CC4S bit only when the channel is off (CC4E=0 in ATIM_CCER).
7:4	IC3F	RW	Input capture 3 filter, see IC1F description
3:2	IC3PSC	RW	Input capture 3 prescaler, see IC1PSC description
1:0	CC3S	RW	Capture/compare 3 selection This bit field defines the channel direction (input/output) and the input mapping used. 00: CC3 channel configured as output 01: CC3 channel configured as input, IC3 mapped to TI3 10: CC3 channel configured as input, IC3 mapped to TI4 11: CC3 channel configured as input, IC3 mapped to TRC, this mode is only valid when internal trigger input is selected via the TS bit (ATIM_SMCR register) Caution: Data can be written to the CC3S bit only when the channel is off (CC3E=0 in ATIM_CCER).



14.9.11 ATIM_CCMR2CMP Compare mode register 2

Address offset: 0x1C Reset value: 0x0000 0000

This register is the same register as ATIM_CCMR2CAP and can be used in input capture mode or output compare mode. The channel direction is defined by configuring the corresponding CCyS bit. All other bits of this register function differently in input capture mode and output compare mode.

Bit field	Name	Permission	Function description
31:25	RFU	-	Reserved bits, please keep the default value
24	OC4MH	RW	Used in conjunction with OC4M, it is the highest bit of the OC4M
23:17	RFU	-	Reserved bits, please keep the default value
16	OC3MH	RW	Used in conjunction with OC3M, it is the highest bit of the OC3M
15	OC4CE	RW	Output compare 4 clear enabled, see OC1CE description
14:12	OC4M	RW	Output compare 4 mode, see OC1M description
11	OC4PE	RW	Output compare 4 preload enabled, see OC1PE description
10	OC4FE	RW	Output compare 4 fast enabled, see OC1FE description
9:8	CC4S	RW	Capture/compare 4 selection This bit field defines the channel direction (input/output) and the input mapping used. 00: CC4 channel configured as output 01: CC4 channel configured as input, IC4 mapped to TI4 10: CC4 channel configured as input, IC4 mapped to TI3 11: CC4 channel configured as input, IC4 mapped to TRC, this mode is only valid when internal trigger input is selected via the TS bit (ATIM_SMCR register) Caution: Data can be written to the CC4S bit only when the channel is off (CC4E=0 in ATIM_CCER).
7	OC3CE	RW	Output compare 3 clear to enabled, see OC1CE description
6:4	OC3M	RW	Output compare 3 mode, see OC1M description
3	OC3PE	RW	Output compare 3 preload enabled, see OC1PE description
2	OC3FE	RW	Output compare 3 fast enabled, see OC1FE description
1:0	CC3S	RW	Capture/compare 3 selection This bit field defines the channel direction (input/output) and the input used. 00: CC3 channel configured as output 01: CC3 channel configured as input, IC3 mapped to TI3 10: CC3 channel configured as input, IC3 mapped to TI4 11: CC3 channel configured as input, IC3 mapped to TRC. This mode is only valid when the internal trigger input is selected via the TS bit (ATIM_SMCR register) Caution: Data can be written to the CC3S bit only when the channel is off (CC3E=0 in ATIM_CCER).



14.9.12 ATIM_CCMR3CAP Capture mode register 3

Address offset: 0x50 Reset value: 0x0000 0000

This register is the same register as ATIM_CCMR3CMP and can be used in input capture mode or output compare mode. The channel direction is defined by configuring the corresponding CCyS bit. All other bits of this register function differently in input capture mode and output compare mode.

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:12	IC6F	RW	Input capture 6 filter, see IC1F description
11:10	IC6PSC	RW	Input capture 6 prescaler, see IC1PSC description
9:8	CC6S	RW	Capture/compare 6 selection This bit field defines the channel direction (input/output) and the input mapping used. 00: CC6 channel configured as output 01: CC6 channel configured as input, IC6 mapped to TI6 10: CC6 channel configured as input, IC6 mapped to TI5 11: CC6 channel configured as input, IC6 mapped to TRC. This mode is only valid when the internal trigger input is selected via the TS bit (ATIM_SMCR register) Caution: Data can be written to the CC6S bit only when the channel is off (CC6E=0 in ATIM_CCER).
7:4	IC5F	RW	Input capture 5 filter, see IC1F description
3:2	IC5PSC	RW	Input capture 5 prescaler, see IC1PSC description
1:0	CC5S	RW	Capture/compare 5 selection This bit field defines the channel direction (input/output) and the input mapping used. 00: CC5 channel configured as output 01: CC5 channel configured as input, IC5 mapped to TI5 10: CC5 channel configured as input, IC5 mapped to TI6 11: CC5 channel configured as input, IC5 mapped to TRC. This mode is only valid when the internal trigger input is selected via the TS bit (ATIM_SMCR register) Caution: Data can be written to the CC5S bit only when the channel is off (CC5E=0 in ATIM_CCER).



14.9.13 ATIM_CCMR3CMP Compare mode register 3

Address offset: 0x50 Reset value: 0x0000 0000

This register is the same register as ATIM_CCMR3CAP and can be used in input capture mode or output compare mode. The channel direction is defined by configuring the corresponding CCyS bit. All other bits of this register function differently in input capture mode and output compare mode.

Bit field	Name	Permission	Function description
31:25	RFU	-	Reserved bits, please keep the default value
24	OC6MH	RW	Used in conjunction with OC6M, it is the highest bit of the OC6M
23:17	RFU	-	Reserved bits, please keep the default value
16	OC5MH	RW	Used in conjunction with OC5M, it is the highest bit of the OC5M
15	OC6CE	RW	Output compare 6 clear enabled, see OC1CE description
14:12	OC6M	RW	Output compare 6 mode, see OC1M description
11	OC6PE	RW	Output compare 6 preload enabled, see OC1PE description
10	OC6FE	RW	Output compare 6 fast enabled, see OC1FE description
9:8	CC6S	RW	Capture/compare 6 selection This bit field defines the channel direction (input/output) and the input used. 00: CC6 channel configured as output 01: CC6 channel configured as input, IC6 mapped to TI6 10: CC6 channel configured as input, IC6 mapped to TI5 11: CC6 channel configured as input, IC6 mapped to TRC. This mode is only valid when the internal trigger input is selected via the TS bit (ATIM_SMCR register) Caution: Data can be written to the CC6S bit only when the channel is off (CC6E=0 in ATIM_CCER).
7	OC5CE	RW	Output compare 5 clear enabled, see OC1CE description
6:4	OC5M	RW	Output compare 5 mode, see OC1M description
3	OC5PE	RW	Output compare 5 preload enabled, see OC1PE description
2	OC5FE	RW	Output compare 5 fast enabled, see OC1FE description
1:0	CC5S	RW	Capture/compare 5 selection This bit field defines the channel direction (input/output) and the input used. 00: CC5 channel configured as output 01: CC5 channel configured as input, IC5 mapped to TI5 10: CC5 channel configured as input, IC5 mapped to TI6 11: CC5 channel configured as input, IC5 mapped to TRC. This mode is only valid when the internal trigger input is selected via the TS bit (ATIM_SMCR register) Caution: Data can be written to the CC5S bit only when the channel is off (CC5E=0 in ATIM_CCER).



14.9.14 ATIM_CCER Capture/compare enable register

Address offset: 0x20 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:24	RFU	-	Reserved bits, please keep the default value
23	CC6NP	RW	Capture/compare 6 complementary output polarity, see CC1NP
22	CC6NE	RW	Capture/compare 6 complementary output enabled, see CC1NE
21	CC6P	RW	Capture/compare 6 output polarity, see CC1P
20	CC6E	RW	Capture/compare 6 output enabled, see CC1E
19	CC5NP	RW	Capture/compare 5 complementary output polarity, see CC1NP
18	CC5NE	RW	Capture/compare 5 complementary output enabled, see CC1NE
17	CC5P	RW	Capture/compare 5 output polarity, see CC1P
16	CC5E	RW	Capture/compare 5 output enabled, see CC1E
15	CC4NP	RW	Capture/compare 4 complementary output polarity, see CC1NP
14	CC4NE	RW	Capture/compare 4 complementary output enabled, see CC1NE
13	CC4P	RW	Capture/compare 4 output polarity, see CC1P
12	CC4E	RW	Capture/compare 4 output enabled, see CC1E
11	CC3NP	RW	Capture/compare 3 complementary output polarity, see CC1NP
10	CC3NE	EW	Capture/compare 3 complementary output enabled, see CC1NE
9	CC3P	RW	Capture/compare 3 output polarity, see CC1P
8	CC3E	RW	Capture/compare 3 output enabled, see CC1E
7	CC2NP	RW	Capture/compare 2 complementary output polarity, see CC1NP
6	CC2NE	RW	Capture/compare 2 complementary output enabled, see CC1NE
5	CC2P	RW	Capture/compare 2 output polarity, see CC1P
4	CC2E	RW	Capture/compare 2 output enabled, see CC1E



Bit field	Name	Permission	Function description
3	CC1NP	RW	Capture/compare 1 complementary output polarity CC1 channel configured as output: 0: OC1N is valid at high level 1: OC1N is valid at low level CC1 channel configured as input: This bit is used in conjunction with CC1P to define TI1FP1/ TI2FP1 polarity. See CC1P description. <i>Caution 1: This bit becomes unwritable as soon as LOCK (LOCK bit in ATIM_BDTR register) level 2 or 3 is programmed and CC1S="00"(channel configured as output).</i> <i>Caution 2: This bit will preload on channels with complementary outputs.If the CCPC bit in the ATIM_CR2 register is set to 1, the CC1NP valid bit takes a new value from the preload bit only when a commutation event is generated.</i>
2	CC1NE	RW	Capture/compare 1 complementary output enabled 0: OFF - OC1N is inactive.The OC1N level depends on the MOE, OSSI, OSSR, OIS1, OIS1N, and CC1E bits. 1: ON - OC1N signal is output on the corresponding output pin, depending on the MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits. <i>Caution: This bit will preload on channels with complementary outputs.If the CCPC bit in the ATIM_CR2 register is set to 1, the CC1NE valid bit takes a new value from the preload bit only when a commutation event is generated.</i>
1	CC1P	RW	Capture/compare 1 output polarity CC1 channel configured as output: 0: OC1 is valid at high level 1: OC1 is valid at low level CC1 channel configured as input: The CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations. 00: Non-inverted/rising edge trigger. The circuit acts on the rising edge of TIxFP1 (capture or trigger operation in reset mode, external clock mode or trigger mode), TIxFP1 is not inverted (trigger operation in gate mode or encoder mode). 01: Inverted/falling edge trigger. The circuit acts on the falling edge of TIxFP1 (performs capture or trigger operation in reset mode, external clock mode, or trigger mode) and TIxFP1 inverts (performs trigger operation in gate mode or encoder mode). 10: Reserved, this configuration is not used.



Bit field	Name	Permission	Function description
1 (continued)	CC1P	RW	11: Non-inverted/triggered on both rising and falling edges. The circuit acts on the rising and falling edges of TlxFP1 (capture or trigger operation in reset mode, external clock mode or trigger mode), TlxFP1 is not inverted (trigger operation in gated mode). This configuration should not be used in encoder mode. Caution 1: <i>This bit becomes unwritable as soon as LOCK (LOCK bit in the ATIM_BDTR register) level 2 or 3 is programmed.</i> Caution 2: <i>This bit will preload on channels with complementary outputs. If the CCPC bit in the ATIM_CR2 register is set to 1, the CC1P valid bit takes a new value from the preload bit only when a commutation event is generated.</i>
0	CC1E	RW	Capture/compare 1 output enabled CC1 channel configured as output: 0: OFF - OC1 is inactive. The OC1 level depends on the MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits. 1: ON - OC1 signal is output on the corresponding output pin, depending on the MOE, OSSI, OSSR, OIS1, OIS1 and CC1NE bits. CC1 channel configured as input: This bit determines whether the counter value can actually be captured in Input Capture/Compare Register 1(ATIM_CCR1). 0: Capture disabled 1: Capture enabled Caution 1: <i>This bit will preload on channels with complementary outputs. If the CCPC bit in the ATIM_CR2 register is set to 1, the CC1E valid bit takes a new value from the preload bit only when a commutation event is generated.</i> Caution 2: <i>This bit must be configured to 1 when using the input capture feature.</i>



14.9.15 ATIM_CNT Count register

Address offset: 0x24 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31	UIFCPY	RO	According to the value of the UIFREMAP bit field in the ATIM_CR1 register, this bit field indicates different meanings: UIFREMAP = 0, this bit field is reserved and read as 0. UIFREMAP = 1, this bit field indicates a read-only copy of the UIF bit of the ATIM_ISR register.
30:16	RFU	-	Reserved bits, please keep the default value
15:0	CNT	RW	Count value

14.9.16 ATIM_PSC Prescaler register

Address offset: 0x28 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	PSC	RW	Prescaler value The counter clock frequency CK_CNT is equal to $f_{CK\_PSC} / (PSC[15:0] + 1)$. The PSC contains the value to be loaded into the valid prescaler register each time an update event occurs, including when the counter is cleared by the UG bit in the ATIM_EGR register, or by the trigger controller when configured for "reset mode".

14.9.17 ATIM_ARR Auto reload register

Address offset: 0x2C Reset value: 0x0000 FFFF

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	ARR	RW	Auto reload value ARR is the value to be loaded into the real auto reload register, the counter does not work when the auto reload value is empty.



14.9.18 ATIM_RCR Repeat count register

Address offset: 0x30 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	REP	RW	Repeat counter value These bits allow the user to set the update frequency of the compare register (i.e., the periodic transfer of data from the preload register to the active register) when the preload register is enabled, or the generation rate of update interrupts when update interrupts are enabled. Each time the down counter associated with REP_CNT counts to 0, an update event is generated and the counter restarts counting from the REP value. Because REP_CNT overloads the REP value only when a repeat update event U_RC is generated, whatever value is written to the ATIM_RCR register has no effect until the next repeat update event is generated. This means that (REP+1) in PWM mode is equivalent to: - The number of PWM cycles in edge-aligned mode. - The number of PWM half-cycles in center-aligned mode.

14.9.19 ATIM_CCR1 Capture/compare register 1

Address offset: 0x34 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	CCR1	RW	Capture/compare 1 value If Channel CC1 is configured to output: CCR1 is the value to be loaded into the valid capture/compare 1 register (preload value). This value takes effect immediately if the preload function is not enabled via the OC1PE bit in the ATIM_CCMR1CMP register; otherwise, it takes effect only on an update event (copied to the capture/compare register 1 that is actually active). The actual capture/compare register contains the value to be compared to counter ATIM_CNT and signaled on the OC1 output. If channel CC1 is configured as input: CCR1 is the counter value when the last Input Capture 1 event (IC1) occurred. The ATIM_CCR1 register can only be read and cannot be programmed.



14.9.20 ATIM_CCR2 Capture/compare register 2

Address offset: 0x38 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	CCR2	RW	Capture/compare 2 value, see CCR1 description

14.9.21 ATIM_CCR3 Capture/compare register 3

Address offset: 0x3C Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	CCR3	RW	Capture/compare 3 value, see CCR1 description

14.9.22 ATIM_CCR4 Capture/compare register 4

Address offset: 0x40 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	CCR4	RW	Capture/compare 4 value, see CCR1 description



14.9.23 ATIM_CCR5 Capture/compare register 5

Address offset: 0x48 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31	GC5C6	RW	CH6/CH5 combined waveform configuration, see GC5C1 description
30	GC5C5	RW	CH5/CH5 combined waveform configuration, see GC5C1 description
29	GC5C4	RW	CH4/CH5 combined waveform configuration, see GC5C1 description
28	GC5C3	RW	CH3/CH5 combined waveform configuration, see GC5C1 description
27	GC5C2	RW	CH2/CH5 combined waveform configuration, see GC5C1 description
26	GC5C1	RW	CH1/CH5 combined waveform configuration 0: OC1REFC is independent of OC5REF 1: Logic AND of OC1REF and OC5REF from OC1REFC outputs
25:16	RFU	-	Reserved bits, please keep the default value
15:0	CCR5	RW	Capture/compare 5 value, see CCR1 description



14.9.24 ATIM_CCR6 Capture/compare register 6

Address offset: 0x4C Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31	GC6C6	RW	CH6/CH6 combined waveform configuration, see GC6C1 description
30	GC6C5	RW	CH5/CH6 combined waveform configuration, see GC6C1 description
29	GC6C4	RW	CH4/CH6 combined waveform configuration, see GC6C1 description
28	GC6C3	RW	CH3/CH6 combined waveform configuration, see GC6C1 description
27	GC6C2	RW	CH2/CH6 combined waveform configuration, see GC6C1 description
26	GC6C1	RW	CH1/CH6 combined waveform configuration 0: OC1REFC is independent of OC6REF 1: OC1REFC output OC1REF when counting up; OC1REFC output OC6REF when counting down
25:16	RFU	-	Reserved bits, please keep the default value
15:0	CCR6	RW	Capture/compare 6 value, see CCR1 description

14.9.25 ATIM_BDTR Brake and deadband register

Address offset: 0x44 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:26	RFU	-	Reserved bits, please keep the default value
25	BK2P	RW	Brake 2 polarity 0: Brake input BRK2 is valid at low level 1: Brake input BRK2 is valid at high level Caution 1: This bit cannot be modified as long as LOCK (LOCK bit in the ATIM_BDTR register) level 1 is programmed. Caution 2: Any write to this bit requires a delay of 1 APB clock cycle to take effect.
24	BK2E	RW	Brake 2 enabled 0: Disable brake input BRK2 1: Enable brake input BRK2 Caution 1: BRK2 must only be used when OSSR=OSS1=1. Caution 2: This bit cannot be modified as long as LOCK (LOCK bit in the ATIM_BDTR register) level 1 is programmed. Caution 3: Any write to this bit requires a delay of 1 APB clock cycle to take effect.



Bit field	Name	Permission	Function description
23:20	BK2F	RW	Brake 2 filter, this bit field defines the sampling frequency of the BRK2 input and the digital filter bandwidth applicable to the BRK2. The digital filter consists of event counters, and every N consecutive events are regarded as a valid output edge. 0000: No filter, BRK2 operates asynchronously 0001: $f_{\text{SAMPLING}} = f_{\text{PCLK}}, N=2$ 0010: $f_{\text{SAMPLING}} = f_{\text{PCLK}}, N=4$ 0011: $f_{\text{SAMPLING}} = f_{\text{PCLK}}, N=8$ 0100: $f_{\text{SAMPLING}} = f_{\text{DTS}}/2, N=6$ 0101: $f_{\text{SAMPLING}} = f_{\text{DTS}}/2, N=8$ 0110: $f_{\text{SAMPLING}} = f_{\text{DTS}}/4, N=6$ 0111: $f_{\text{SAMPLING}} = f_{\text{DTS}}/4, N=8$ 1000: $f_{\text{SAMPLING}} = f_{\text{DTS}}/8, N=6$ 1001: $f_{\text{SAMPLING}} = f_{\text{DTS}}/8, N=8$ 1010: $f_{\text{SAMPLING}} = f_{\text{DTS}}/16, N=5$ 1011: $f_{\text{SAMPLING}} = f_{\text{DTS}}/16, N=6$ 1100: $f_{\text{SAMPLING}} = f_{\text{DTS}}/16, N=8$ 1101: $f_{\text{SAMPLING}} = f_{\text{DTS}}/32, N=5$ 1110: $f_{\text{SAMPLING}} = f_{\text{DTS}}/32, N=6$ 1111: $f_{\text{SAMPLING}} = f_{\text{DTS}}/32, N=8$ Caution: This bit cannot be modified when LOCK (LOCK bit in the ATIM_BDTR register) level 1 is programmed.
19:16	BKF	RW	Brake filter, this bit field defines the sampling frequency of the BRK input and the digital filter bandwidth applicable to the BRK. The digital filter consists of event counters, and every N consecutive events are regarded as a valid output edge. 0000: No filter, BRK operates asynchronously 0001: $f_{\text{SAMPLING}} = f_{\text{PCLK}}, N=2$ 0010: $f_{\text{SAMPLING}} = f_{\text{PCLK}}, N=4$ 0011: $f_{\text{SAMPLING}} = f_{\text{PCLK}}, N=8$ 0100: $f_{\text{SAMPLING}} = f_{\text{DTS}}/2, N=6$ 0101: $f_{\text{SAMPLING}} = f_{\text{DTS}}/2, N=8$ 0110: $f_{\text{SAMPLING}} = f_{\text{DTS}}/4, N=6$ 0111: $f_{\text{SAMPLING}} = f_{\text{DTS}}/4, N=8$ 1000: $f_{\text{SAMPLING}} = f_{\text{DTS}}/8, N=6$ 1001: $f_{\text{SAMPLING}} = f_{\text{DTS}}/8, N=8$ 1010: $f_{\text{SAMPLING}} = f_{\text{DTS}}/16, N=5$ 1011: $f_{\text{SAMPLING}} = f_{\text{DTS}}/16, N=6$ 1100: $f_{\text{SAMPLING}} = f_{\text{DTS}}/16, N=8$ 1101: $f_{\text{SAMPLING}} = f_{\text{DTS}}/32, N=5$ 1110: $f_{\text{SAMPLING}} = f_{\text{DTS}}/32, N=6$ 1111: $f_{\text{SAMPLING}} = f_{\text{DTS}}/32, N=8$ Caution: This bit cannot be modified when LOCK (LOCK bit in the ATIM_BDTR register) level 1 is programmed.



Bit field	Name	Permission	Function description
15	MOE	RW	Master output enabled This bit is cleared asynchronously by hardware whenever the brake input (BRK or BRK2) is valid. This bit is set to 1 by software or can be set automatically based on the state of the AOE bit. This bit is only valid for channels configured as outputs. 0: Responding to braking events (2). OC and OCN outputs disabled Responding to a braking event or writing a 0 to MOE: OC and OCN outputs are disabled or forced to idle, depending on the OSS1 bit. 1: The OC and OCN outputs are enabled if their respective enable bits (CCyE and CCyNE bits in the ATIM_CCER register) are both set to 1.
14	AOE	RW	Auto output enabled 0: MOE can only be set to 1 by software 1: MOE can be set to 1 by software or automatically when the next update event occurs (this bit is invalid if the brake inputs BRK and BRK2 are valid). Caution: This bit cannot be modified as long as LOCK (LOCK bit in ATIM_BDTR register) level 1 is programmed.
13	BKP	RW	Brake polarity 0: Brake input BRK is valid at low level 1: Brake input BRK is valid at high level Caution 1: This bit cannot be modified as long as LOCK (LOCK bit in the ATIM_BDTR register) level 1 is programmed. Caution 2: Any write to this bit takes effect with a delay of 1 APB clock cycle.
12	BKE	RW	Brake enabled This bit enables complete brake protection (including all internal sources and corresponding BKIN sources). 0: Disable brake function 1: Enable brake function Caution 1: This bit cannot be modified as long as LOCK (LOCK bit in the ATIM_BDTR register) level 1 is programmed. Caution 2: Any write to this bit takes effect with a delay of 1 APB clock cycle.



Bit field	Name	Permission	Function description
11	OSSR	RW	Off state selection in operating mode This bit works on channels configured in output mode with complementary outputs when MOE=1. If there is no complementary output in the timer, then there is no OSSR. 0: When inactive, OC/OCN output is disabled (timer releases output control, taken over by GPIO logic forcing high impedance state). 1: In the inactive state, once CCyE=1 or CCyNE=1, enable OC/OCN output and set it to inactive level (output is still controlled by timer). Caution: This bit cannot be modified as long as LOCK (LOCK bit in the ATIM_BDTR register) level 2 is programmed.
10	OSSI	RW	Off state selection in idle mode When MOE=0 due to a braking event or software write operation, this bit acts on channels configured as outputs. 0: When inactive, OC/OCN output is disabled (timer releases output control, taken over by GPIO logic forcing high impedance state). 1: In the inactive state, force the OC/OCN output to its inactive level first and then to its idle level after the dead time. The timer always controls the output. Caution: This bit cannot be modified as long as LOCK (LOCK bit in the ATIM_BDTR register) level 2 is programmed.
9:8	LOCK	RW	Locked configuration These bits are used to provide write protection against software errors. 00: Off lock - Does not write protect any bits. 01: Lock level 1, where no write operation can be performed to the DTG bits in the ATIM_BDTR register, the OISy and OISyN bits in the ATIM_CR2 register, and the BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSSI, OSSR, and DTG[7:0] bits in the ATIM_BDTR register. 10: Lock level 2, where no write operation can be performed to the applicable bits in lock level 1, CC polarity bits (CCyP/CCyNP bits in the ATIM_CCER register, provided the associated channel is configured as output via the CCyS bit), and OSSR and OSSI bits. 11: Lock level 3, where no write operation can be performed to the applicable bits in lock level 2, CC control bits (OCyM and OCyPE bits in the ATIM_CCMRxCMP register, as long as the associated channel is configured as output via the CCyS bit). Caution: Only one write to the LOCK bit can be performed after reset. A write to the ATIM_BDTR register freezes its contents until the next reset.



Bit field	Name	Permission	Function description
7:0	DTG	RW	Configure deadband generator This bit field defines the dead time duration inserted between complementary outputs. DT corresponds to this duration. $DTG[7:5]=0xx \Rightarrow DT=DTG[7:0] * t_{dtg}$, where $t_{dtg}=t_{DTS}$. $DTG[7:5]=10x \Rightarrow DT=(64+DTG[5:0]) * t_{dtg}$, where $t_{dtg}=2 * t_{DTS}$. $DTG[7:5]=110 \Rightarrow DT=(32+DTG[4:0]) * t_{dtg}$, where $t_{dtg}=8 * t_{DTS}$. $DTG[7:5]=111 \Rightarrow DT=(32+DTG[4:0]) * t_{dtg}$, where $t_{dtg}=16 * t_{DTS}$. Example: If $t_{DTS} = 125ns$ (8MHz), the possible deadband values are: 0 to 15875ns (in 125ns steps) 16μs to 31750ns (in 250ns steps) 32μs to 63μs (in 1μs steps) 64μs to 126μs (in 2 μs steps) Caution: This bit field cannot be modified as long as LOCK (LOCK bit in the ATIM_BDTR register) level 1, 2, or 3 is programmed.

Caution:

Since the write to the lock bits BK2P, BK2E, BK2F [3:0], BKF[3:0], AOE, BKP, BKE, OSS1, OSSR, and DTG[7:0] can be configured according to LOCK, these bits must be configured the first time a write access is performed to the ATIM_BDTR register.



14.9.26 ATIM_DTR2 Dead time register 2

Address offset: 0x54 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:18	RFU	-	Reserved bits, please keep the default value
17	DTPE	RW	Dead time value preload enabled 0: Disable dead time value preload 1: Enable dead time value preload
16	DTAE	RW	Dead time asymmetry enabled 0: The dead time of the rising and falling edges is the same, defined by DTG[7:0]. 1: Dead time of rising edge is defined by DTG[7:0] and dead time of falling edge is defined by DTGF[7:0]. Caution: This bit field cannot be modified as long as LOCK (LOCK bit in ATIM_BDTR register) level 1, 2, or 3 is programmed.
15:8	RFU	-	Reserved bits, please keep the default value
7:0	DTGF	RW	Dead time generator for falling edge This bit field defines the dead time duration inserted between falling edges of complementary outputs. DTGF[7:5]=0xx => DTF=DTGF[7:0] * t_{dtg} , where $t_{dtg}=t_{DTS}$. DTGF[7:5]=10x => DTF=(64+DTG[5:0]) * t_{dtg} , where $t_{dtg}=2 * t_{DTS}$. DTGF[7:5]=110 => DTF=(32+DTG[4:0]) * t_{dtg} , where $t_{dtg}=8 * t_{DTS}$. DTGF[7:5]=111 => DTF=(32+DTG[4:0]) * t_{dtg} , where $t_{dtg}=16 * t_{DTS}$. Example: If $t_{DTS} = 125\text{ns}$ (8MHz), the possible deadband values are: - 0 to 15875ns (in 125ns steps) - 16μs to 31750ns (in 250ns steps) - 32 μs to 63μs (in 1μs steps) - 64μs to 126μs (in 2 μs steps) Caution: This bit field cannot be modified as long as LOCK (LOCK bit in ATIM_BDTR register) level 1, 2, or 3 is programmed.



14.9.27 ATIM_ECR Encoder control register

Address offset: 0x58 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:8	RFU	-	Reserved bits, please keep the default value
7:6	IPOS	RW	Coding index position In quadrature encoder mode (SMS[3:0] = 0001, 0010, 0011, 1110, 1111), this bit indicates in which AB input configuration the index event resets the counter. 00: When AB = 00, the index resets the counter 01: When AB = 01, the index resets the counter 10: When AB = 10, the index resets the counter 11: When AB = 11, the index resets the counter In directional clock encoder mode or clock plus direction encoder mode (SMS[3:0] = 1010, 1011, 1100, 1101), these bits indicate at which level the index event resets the counter. In directional clock encoder mode, this applies to both clock inputs. x0: when the clock signal is low, the index resets the counter x1: When the clock signal is high, the index resets the counter
5	FIDX	RW	This bit indicates whether only the first index is considered 0: Index is always active 1: Only the first index resets the counter
4:3	RFU	-	Reserved bits, please keep the default value
2:1	IDIR	RW	Index direction This bit field indicates in which count direction the index resets the counter 00: Reset counter in any counting direction 01: Reset counter only when counting up 10: Reset counter only when counting down 11: Reserved
0	IE	RW	Index enabled 0: Index disabled 1: Index enabled



14.9.28 ATIM_TISEL1 TI input selection register 1

Address offset: 0x5C Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:28	RFU	-	Reserved bits, please keep the default value
27:24	TI4SEL	RW	TI4 channel input capture signal source selection, see TI1SEL description
23:20	RFU	-	Reserved bits, please keep the default value
19:16	TI3SEL	RW	TI3 channel input capture signal source selection, see TI1SEL description
15:12	RFU	-	Reserved bits, please keep the default value
11:8	TI2SEL	RW	TI2 channel input capture signal source selection, see TI1SEL description
7:4	RFU	-	Reserved bits, please keep the default value
3:0	TI1SEL	RW	TI1 channel input capture signal source selection 0000: ATIM_CH1 (Caution: TIx corresponds to ATIM CHx) 0001: VC1_OUT 0010: VC2_OUT 0011: UART1_RXD 0100: UART2_RXD 0101: MCO_OUT 0110: HSE_FAULT 0111: LSE_FAULT 1000: RTC_OUT 1001: LSI_OUT 1010: BTIM1_Trgo 1011: BTIM2_Trgo 1100: BTIM3_Trgo 1101: GTIM_Trgo

14.9.29 ATIM_TISEL2 TI input selection register 2

Address offset: 0x6C Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:12	RFU	-	Reserved bits, please keep the default value
11:8	TI6SEL	RW	TI6 channel input capture signal source selection, see TI1SEL description
7:4	RFU	-	Reserved bits, please keep the default value
3:0	TI5SEL	RW	TI5 channel input capture signal source selection, see TI1SEL description



14.9.30 ATIM_AF1 Multiplexing function option register 1

Address offset: 0x60 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:18	RFU	-	Reserved bits, please keep the default value
17:14	ETRSEL	RW	ETR source selection 0000: ATIM_ETR pin 0001: VC1_OUT 0010: VC2_OUT 0101: ADC_AWD 0111: HSE_FAULT 1000: LVD_OUT 1001: GTIM_ETR pin 1101: UART1_TXD 1110: UART2_TXD 1111: LSE_FAULT <i>Caution: These bits cannot be modified as long as LOCK (LOCK bit in the ATIM_BDTR register) level 1 is programmed.</i>
13:12	RFU	-	Reserved bits, please keep the default value
11	BKVC2P	RW	BRK VC2 input polarity This bit selects the VC2 input level sensitivity and must be programmed with the BKP polarity bit. 0: VC2 input is valid at high level 1: VC2 input is valid at low level <i>Caution: This bit cannot be modified as long as LOCK (LOCK bit in the ATIM_BDTR register) level 1 is programmed.</i>
10	BKVC1P	RW	BRK VC1 input polarity This bit selects the VC1 input level sensitivity and must be programmed with the BKP polarity bit. 0: VC1 input is valid at high level 1: VC1 input is valid at low level <i>Caution: This bit cannot be modified as long as LOCK (LOCK bit in the ATIM_BDTR register) level 1 is programmed.</i>
9	BKINP	RW	BRK BKIN input polarity This bit selects the BKIN multiplex function input level sensitivity and must be programmed with the BKP polarity bit. 0: BKIN input is valid at high level 1: BKIN input is valid at low level <i>Caution: This bit cannot be modified as long as LOCK (LOCK bit in the ATIM_BDTR register) level 1 is programmed.</i>
8:3	RFU	-	Reserved bits, please keep the default value



Bit field	Name	Permission	Function description
2	BKVC2E	RW	BRK VC2 enabled This bit enables VC2 input by the timer BRK, and the VC2 output performs an "OR" operation with other BRK sources. 0: Disable VC2 input 1: Enable VC2 input <i>Caution: This bit cannot be modified as long as LOCK (LOCK bit in the ATIM_BDTR register) level 1 is programmed.</i>
1	BKVC1E	RW	BRK VC1 enabled This bit enables VC1 input by the timer BRK, and the VC1 output performs an "OR" operation with other BRK sources. 0: Disable VC1 input 1: Enable VC1 input <i>Caution: This bit cannot be modified as long as LOCK (LOCK bit in the ATIM_BDTR register) level 1 is programmed.</i>
0	BKINE	RW	BRK BKIN input enabled This bit enables the BKIN multiplexing function input by the timer BRK, and the BKIN input performs an "OR" operation with other BRK sources. 0: Disable BKIN input 1: Enable BKIN input <i>Caution: This bit cannot be modified as long as LOCK (LOCK bit in the ATIM_BDTR register) level 1 is programmed.</i>



14.9.31 ATIM_AF2 Multiplexing function option register 2

Address offset: 0x64 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:19	RFU	-	Reserved bits, please keep the default value
18:16	OCRSEL	RW	OCREF_CLR source selection 000: VC1_OUT 001: VC2_OUT 100: ADC_AWD 111: ETRF
15:12	RFU	-	Reserved bits, please keep the default value
11	BK2VC2P	RW	BRK2 VC2 input polarity This bit selects the VC2 input level sensitivity and must be programmed with the BK2P polarity bit. 0: VC2 input is valid at low level 1: VC2 input is valid at highlevel Caution: This bit cannot be modified as long as LOCK (LOCK bit in the ATIM_BDTR register) level 1 is programmed.
10	BK2VC1P	RW	BRK2 VC1 input polarity This bit selects the VC1 input level sensitivity and must be programmed with the BK2P polarity bit. 0: VC1 input is valid at low level 1: VC1 input is valid at high level Caution: This bit cannot be modified as long as LOCK (LOCK bit in the ATIM_BDTR register) level 1 is programmed.
9	BK2INP	RW	BRK2 BKIN2 input polarity This bit selects the BKIN2 multiplex function input level sensitivity and must be programmed with the BK2P polarity bit. 0: BKIN2 input is valid at low level 1: BKIN2 input is valid at high level Caution: This bit cannot be modified as long as LOCK (LOCK bit in the ATIM_BDTR register) level 1 is programmed.
8:3	RFU	-	Reserved bits, please keep the default value
2	BK2VC2E	RW	BRK2 VC2 enabled This bit enables VC2 input by the timer BRK2, and the VC2 output performs an "OR" operation with other BRK2 sources. 0: Disable VC2 input 1: Enable VC2 input Caution: This bit cannot be modified as long as LOCK (LOCK bit in the ATIM_BDTR register) level 1 is programmed.



Bit field	Name	Permission	Function description
1	BK2VC1E	RW	BRK2 VC1 enabled This bit enables VC1 input by the timer BRK2, and the VC1 output performs an "OR" operation with other BRK2 sources. 0: Disable VC1 input 1: Enable VC1 input Caution: This bit cannot be modified as long as LOCK (LOCK bit in the ATIM_BDTR register) level 1 is programmed.
0	BK2INE	RW	BRK2 BKIN input enabled This bit enables the BKIN2 multiplexing function input by the timer BRK2, and the BKIN2 input performs an "OR" operation with other BRK2 sources. 0: Disable BKIN2 input 1: Enable BKIN2 input Caution: This bit cannot be modified as long as LOCK (LOCK bit in the ATIM_BDTR register) level 1 is programmed.



15 Independent watchdog timer (IWDG)

15.1 Overview

The CW32L010 integrates an independent watchdog timer (IWDG). Once the IWDG is started, the user needs to reload the IWDG counter within the specified time interval, otherwise the counter overflow will trigger a reset or generate an interrupt signal. After the IWDG is started, the counting can be stopped. The IWDG can be selected to keep running or suspend counting during DeepSleep mode.

The specially set key-value register can lock the key registers of the IWDG to prevent the registers from being accidentally modified.

15.2 Main Features

- 12-bit down counter
- Programmable clock prescale period
- Overflow can trigger interrupt or reset
- Register protection lock function
- Counting can be suspended in DeepSleep mode

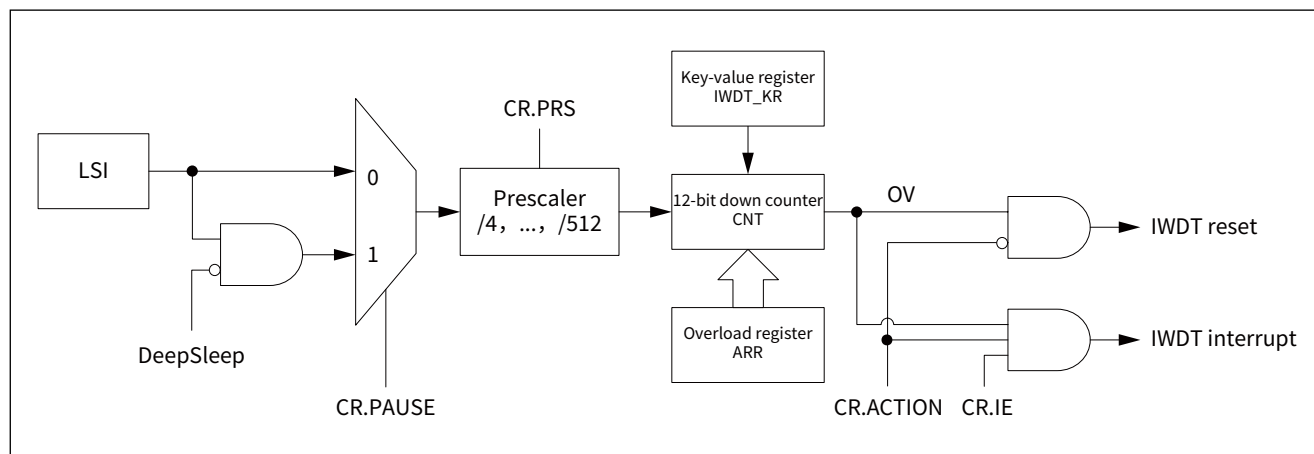


15.3 Functional description

15.3.1 Functional block diagram

The functional block diagram of IWDT is shown in the following figure:

Figure 15-1 IWDT functional block diagram



The IWDT is implemented by a 12-bit reloadable down-counter, and its counting clock source is the internal low-speed LSI clock. The clock source LSI signal can be prescaled by 4 to 512 through the PRS bit field of the control register IWDT_CR. When the IWDT counter overflows, it can optionally generate interrupt and reset signals.

15.3.2 Operating modes

To start the IWDT counter, you need to write 0xCCCC to the key value register IWDT_KR, and the counter starts to count down from 0xFFFF.

Before the counter reduces to 0, writing 0xAAAA to the IWDT_KR register will trigger a counter reload, loading the ARR register value into the counter. When the IWDT counter value is reduced to 0, an overflow event will be generated. The overflow event can trigger the MCU reset or generate the IWDT interrupt signal, and trigger the counter reload at the same time. The ACTION and IE bit fields of the control register IWDT_CR are used to control whether interrupts and resets are generated when the watchdog overflows, as shown in the following table:

Table 15-1 Operating modes

IWDT_CR.ACTION	IWDT_CR.IE	Action after IWDT overflows
0	X	Reset
1	1	No reset, only interrupt
1	0	No reset, no interrupt

When the MCU enters the DeepSleep mode, the IWDT can choose to suspend the IWDT counting, so as to achieve lower overall system power consumption: the PAUSE bit field of the control register IWDT_CR is 0, and the IWDT timer is kept running in the DeepSleep mode; when it is 1, the counting is suspended, the IWDT automatically resumes counting when the MCU exits the DeepSleep mode.

15.3.3 Window options

Write a value smaller than the reload register IWDT_ARR to the window register IWDT_WINR to make the IWDT work in the window watchdog mode. The default value of the IWDT_WINR register is 0x0FFF, that is, the window option is closed by default.

Modify the value of the IWDT_WINR register will trigger a reload operation to load the ARR register value into the counter.

When using the IWDT window function, the user should perform a reload operation before the counter value is less than or equal to the IWDT_WINR window value and decrements to 0 to avoid reset or watchdog timer overflow. A reload operation when the watchdog counter value is greater than the window value will trigger a system reset.

15.3.4 Register lock function

Through the key value register IWDT_KR of the IWDT, the important registers IWDT_CR, IWDT_ARR, and IWDT_WINR of the IWDT can be configured to be locked or unlocked. The registers cannot be modified in the locked state.

Write 0x5555 to the IWDT_KR register to unlock the IWDT_CR, IWDT_ARR, and IWDT_WINR registers; write any other value to the IWDT_KR register to enable lock protection.

After the CW32L010 is powered on, the IWDT_CR, IWDT_ARR, and IWDT_WINR registers are locked by default, and the user needs to unlock them before modifying them.

15.3.5 Start refresh and stop

Configure the IWDT_KR register to start, refresh and stop the IWDT:

- Write 0xCCCC to start IWDT
- Write 0xAAAA, reload the counter, that is, refresh the IWDT
- Write 0x5A5A, 0xA5A5 sequentially, stop IWDT

Caution:

The above operations will simultaneously activate the IWDT register lock protection.



15.3.6 Status register

The status register IWDT_SR indicates the current running status or register update status of the IWDT, as shown in the following table:

Table 15-2 IWDT status indication

IWDT_SR bit	Name	Description	1	0
4	RUN	Run flag	Running	Not running
3	OV	Overflow flag	Generate overflow	Not overflow
5	RELOAD	Counter reload flag	Reloading	Reload completed
2	WINRF	WINR register update flag	Updating	Update completed
1	ARRF	ARR register update flag	Updating	Update completed
0	CRF	CR register update flag	Updating	Update completed

To ensure the correct operation of the IWDT register, after updating IWDT_WINR, IWDT_ARR, and IWDT_CR, the user needs to check whether the WINRF, ARRF, and CRF flags are 0 to confirm whether the operation is completed.

To ensure the reload operation of the IWDT, the user should check whether the RELOAD flag is 0 after the reload operation.



15.3.7 Timing duration setting

The counting clock source of the IWDT is the internal low-speed LSI clock (the clock frequency is approximately 32.8KHz, please refer to the datasheet for details). By controlling the PRS bit field of the IWDT_CR register, the frequency of the clock source LSI signal can be divided, as shown in the following table:

Table 15-3 IWDT prescaler divider table

IWDT_CR.PRS	Prescaler value
000	4
001	8
010	16
011	32
100	64
101	128
110	256
111	512

Watchdog timer duration calculation formula:

$$T = (4 \times 2^{\text{PRS}} / f) \times (\text{ARR} + 1)$$

Where, f is the frequency of the clock source LSI, PRS is the prescaler divider, and ARR is the reload value.

Therefore, when the frequency of the clock source LSI is 32800Hz, the longest and shortest timing ranges of the IWDT are:

$$\text{IWDT shortest timing} = (4 \times 2^0 / 32800) \times (0x000 + 1) \approx 122\mu\text{s}$$

$$\text{IWDT longest timing} = (4 \times 2^7 / 32800) \times (0xFF + 1) \approx 63.9\text{s}$$

Example: When the frequency of the clock source LSI is 32800Hz, set the prescale value to 64 and the reload value to 512, then:

$$\text{IWDT timing duration} = (4 \times 2^4 / 32800) \times (512 + 1) = 1\text{s}$$



15.4 Programming examples

15.4.1 Configure IWDG as independent watchdog

Step 1: Set SYSCTRL_APBEN2.IWDG to 1 to enable the configuration clock of IWDG;

Step 2: Write 0xCCCC to the IWDG_KR register to start the IWDG;

Caution:

It is necessary to start IWDG before modifying the relevant registers.

Step 3: Write 0x5555 to the IWDG_KR register to release the lock function of the IWDG register;

Step 4: Configure IWDG_CR, configure the prescale value of watchdog count clock and LSI oscillator, action after overflow, whether to automatically suspend in DeepSleep mode;

Step 5: Configure IWDG_ARR and configure the overflow period of the watchdog;

Step 6: Wait for IWDG_SR.ARRF and IWDG_SR.CRF to become 0, wait for the reload value and CR register update to complete;

Step 7: Write 0xAAAA to the IWDG_KR register to load the ARR into the IWDG counter.

15.4.2 Configure IWDG as window watchdog

Step 1: Set SYSCTRL_APBEN2.IWDG to 1 to enable the configuration clock of IWDG;

Step 2: Write 0xCCCC to the IWDG_KR register to start the IWDG;

Caution:

It is necessary to start IWDG before modifying the relevant registers.

Step 3: Write 0x5555 to the IWDG_KR register to release the lock function of the IWDG register;

Step 4: Configure IWDG_CR, configure the prescale value of watchdog count clock and LSI oscillator, action after overflow, whether to automatically suspend in DeepSleep mode;

Step 5: Configure IWDG_ARR, configure the reload value of the watchdog;

Step 6: Configure IWDG_WINR, configure the window size. Caution that IWDG_WINR must be less than the overloaded value of IWDG_ARR;

Step 7: Wait for IWDG_SR.ARRF, IWDG_SR.WINRF and IWDG_SR.CRF to become 0, wait for the reload value, window register and CR register update to complete;

Step 8: Write 0xAAAA to the IWDG_KR register to load the ARR into the IWDG counter.

15.4.3 Refresh IWDG (feed dog operation)

Step 1: Write 0xAAAA to the IWDG_KR register to load ARR into the counter;

Step 2: Wait for IWDG_SR.RELOAD to become 0, and wait for the reload operation to complete.



15.5 List of registers

IWDT base address: IWDT_BASE = 0x4000 5000

Table 15-4 List of IWDT registers

Register name	Register address	Register description
IWDT_KR	IWDT_BASE + 0x00	Key-value register
IWDT_CR	IWDT_BASE + 0x04	Control register
IWDT_ARR	IWDT_BASE + 0x08	Reload register
IWDT_SR	IWDT_BASE + 0x0C	Status register
IWDT_WINR	IWDT_BASE + 0x10	Window register
IWDT_CNT	IWDT_BASE + 0x24	Count value register



15.6 Register descriptions

See section [1 Documentation conventions](#) for abbreviations used in register descriptions.

15.6.1 IWDT_KR Key-value register

Address offset: 0x00 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	KR	WO	Write 0xCCCC: start the IWDT counter Write 0xAAAA: reload the IWDT count value Write 0x5A5A, 0xA5A5 in sequence: stop watchdog Write 0x5555: release the write protection of IWDT_ARR, IWDT_WINR, IWDT_CR Write non-0x5555: enable write protection of IWDT_ARR, IWDT_WINR, IWDT_CR



15.6.2 IWDT_CR Control register

Address offset: 0x04 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:6	RFU	-	Reserved bits, please keep the default value
5	PAUSE	RW	IWDT suspension control in DeepSleep mode 0: IWDT continues to run in DeepSleep mode 1: IWDT automatically suspends in DeepSleep mode
4	IE	RW	IWDT interrupt enable control 0: Watchdog interrupt disabled 1: Watchdog interrupt enabled
3	ACTION	RW	Action configuration after IWDT overflow 0: Reset after watchdog overflow 1: Interrupt after watchdog overflow
2:0	PRS	RW	Configure the divider ratio of the watchdog count clock to the LSI oscillator 000: 4 frequency division 001: 8 frequency division 010: 16 frequency division 011: 32 frequency division 100: 64 frequency division 101: 128 frequency division 110: 256 frequency division 111: 512 frequency division

Caution:

This register can be modified only after writing 0x5555 to the IWDT_KR register; IWDT_CR can only be modified when IWDT_SR.CRF is 0.

15.6.3 IWDT_ARR Reload register

Address offset: 0x08 Reset value: 0x0000 0FFF

Bit field	Name	Permission	Function description
31:12	RFU	-	Reserved bits, please keep the default value
11:0	ARR	RW	IWDT reload value

Caution:

This register can be modified only after writing 0x5555 to the IWDT_KR register; IWDT_ARR can only be modified when IWDT_SR.CRF is 0.



15.6.4 IWDT_CNT Count value register

Address offset: 0x24 Reset value: 0x0000 0FFF

Bit field	Name	Permission	Function description
31:12	RFU	-	Reserved bits, please keep the default value
11:0	CNT	RO	Count value register <i>Caution: The count value can only be considered to be read if the value read twice in a row is the same.</i>

15.6.5 IWDT_WINR Window register

Address offset: 0x10 Reset value: 0x0000 0FFF

Bit field	Name	Permission	Function description
31:12	RFU	-	Reserved bits, please keep the default value
11:0	WINR	RW	Window comparator compare value



15.6.6 IWDT_SR Status register

Address offset: 0x0C Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:6	RFU	-	Reserved bits, please keep the default value
5	RELOAD	RO	WDT counter reload flag 0: WDT counter has completed reload operation 1: WDT counter is reloading <i>Caution: In window mode, you need to wait for the flag to become 0 after reloading.</i>
4	RUN	RO	WDT running flag 0: WDT is not running 1: WDT is running
3	OV	RW0	WDT overflow flag R0: WDT does not overflow R1: WDT overflow W0: Clear the WDT overflow flag
2	WINRF	RO	WINR register update flag 0: WINR register update completed 1: WINR register is updating <i>Caution: After writing the WINR register, you need to wait for this flag to become 0.</i>
1	ARRF	RO	ARR register update flag 0: ARR register update completed 1: ARR register is updating <i>Caution: After writing the ARR register, you need to wait for this flag to become 0.</i>
0	CRF	RO	CR register update flag 0: CR register update completed 1: CR register is updating <i>Caution: After writing the CR register, you need to wait for the flag to become 0.</i>



16 Universal asynchronous receiver/transmitter (UART)

16.1 Overview

CW32L010 integrates 2 universal asynchronous receiver/transmitter (UART), supports asynchronous full-duplex, synchronous half-duplex and single-wire half-duplex modes, supports hardware data flow control and multi-computer communication, also supports LIN (Local Interconnect Network); programmable data frame structure can provide wide range of baud rate selection through wavelet baud rate generator. Built-in timer module supporting receive timeout detection, receive idle detection, automatic baud rate detection and general timing functions.

The UART controller works in a dual clock domain, allowing data reception in DeepSleep mode, and the reception completion interrupt can wake the MCU back to Active mode.

Caution: Only UART1 supports LIN and timer functions; UART2 enables timeout timer related functions through on-chip peripheral interconnect working in conjunction with slave mode of BTIM/GTIM/ATIM.



16.2 Main features

- Supports dual clock domain drive:
 - Configure clock PCLK
 - Transmit clock UCLK
- Programmable data frame structure
 - Data word length: 8/9 bits, LSB/MSB first
 - Parity bit: no parity/odd parity/even parity
 - Stop bit length: 1/1.5/2 bits
- 16-bit integer, 4-bit fractional baud rate generator
- Supports asynchronous full-duplex, synchronous half-duplex, single-wire half-duplex
- Separate transmitter and receiver enable bits
- Separate transmit and receive signal polarity control
- TXD/RXD pin configurations are interchangeable
- Support LoopBack mode
- Support received data match detection
- Interval frame and idle character transmission and reception functions
- Hardware flow control RTS, CTS and RS485 driver enable
- Transmit buffer empty/receive data complete trigger start ADC
- Supports multi-machine communication, automatic address recognition
- 13 interrupt sources with interrupt flags
- Auto baud rate detection mode 1/2
- Receive timeout detection/idle detection
- Built-in timer module supports universal timer function
- Error detection: parity error, frame structure error, overflow error, noise error
- Transmit and receive data in low power mode, interrupt wake-up MCU
- Support communication with devices operating at lower voltages than the MCU (via VC)
- LIN master synchronous break send capability and LIN slave break detection capability:
 - Support length-configurable Break Field transmission
 - Support 10/11 bit Break Field detection



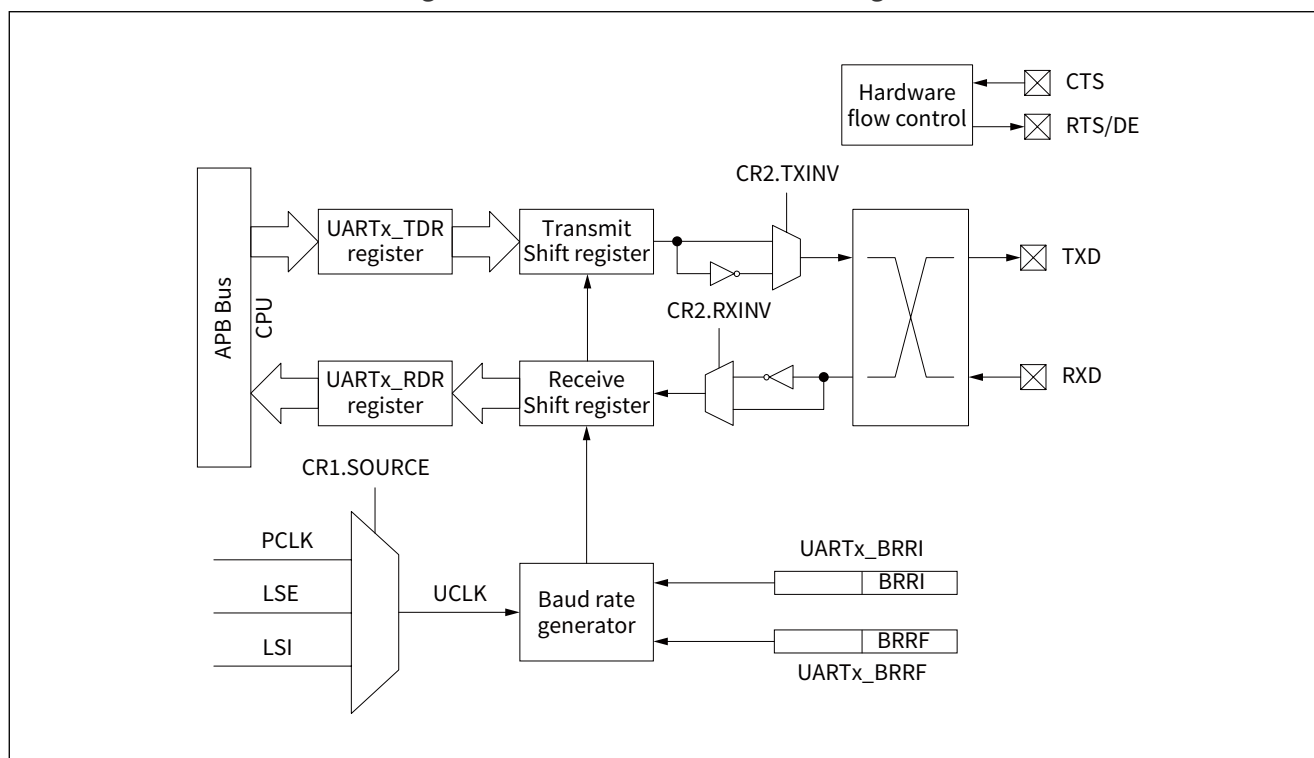
16.3 Functional description

16.3.1 Functional block diagram

The UART controller is mounted on the APB bus, and the clock domain PCLK is configured, which is fixed to the APB bus clock PCLK, which is used for register configuration logic work; transfer clock domain UCLK is used for data transceiver logic work, and its source can be selected from PCLK clock, external low-speed clock (LSE) and internal low-speed clock (LSI). The dual clock domain design facilitates baud rate setting and supports waking up the controller from DeepSleep mode.

UART transmitters and receivers have separate enable bits that support separate signal polarity control for transmit and receive, and the TXD/RXD pin configurations are interchangeable for easy user configuration. The functional block diagram of the UART controller is shown in the following figure:

Figure 16-1 UART functional block diagram



The UART controller supports multiple operating modes. In different operating modes, each pin has different function and different pin configuration, as shown in the following table (where x = 1, 2):

Table 16-1 UART ports configuration

Operating mode	UART pin	Function	GPIO configuration
Asynchronous full duplex	UARTx_TXD	Data serial output	Digital, push-pull output/open-drain output, multiplexed
	UARTx_RXD	Data serial input	Digital, pull-up input, multiplexed
	UARTx_RTS ¹	Request to transmit	Digital, push-pull output/open-drain output, multiplexed
	UARTx_CTS	Allow to transmit	Digital, pull-up input, multiplexed
Synchronous half duplex	UARTx_TXD	Clock signal output	Digital, push-pull output/open-drain output, multiplexed
	UARTx_RXD	Data transmitting and receiving	Digital, push-pull output/open-drain output, multiplexed
Single-wire half duplex	UARTx_TXD	Data transmitting and receiving	Digital, open-drain output, multiplexed (external pull-up required)
	UARTx_RXD	Do not use	Can be used as general IO

Caution 1:

When the RS485 driver enable signal is set, the RTS pin is used as the DE function.



16.3.2 Synchronous mode

The UART supports the synchronous half-duplex working mode. In this mode, the UARTx_TXD pin outputs a synchronous shift clock signal, and the UARTx_RXD pin transmits and receives data. See [Table 16-1 UART ports configuration](#) for the specific configuration of the UARTx_TXD and UARTx_RXD pins.

The synchronous mode of the UART can communicate with the single-wire half-duplex mode of the SPI. At this time, the SPI must be configured in a fixed level mode: the clock polarity CPOL is 1, and the clock phase CPHA is 1.

Set the SYNC bit field of the control register UARTx_CR1 to 1 to make the UART work in synchronous half-duplex mode. At this time, the UART can only be used as a master, the data word length can only be 8 bits, and UARTx_CR1.SIGNAL and UARTx_CR2.ADDREN must be cleared.

16.3.2.1 Baud rate setting

In synchronous half-duplex mode, the baud rate calculation formula:

$$\text{BaudRate} = \text{UCLK} / 12$$

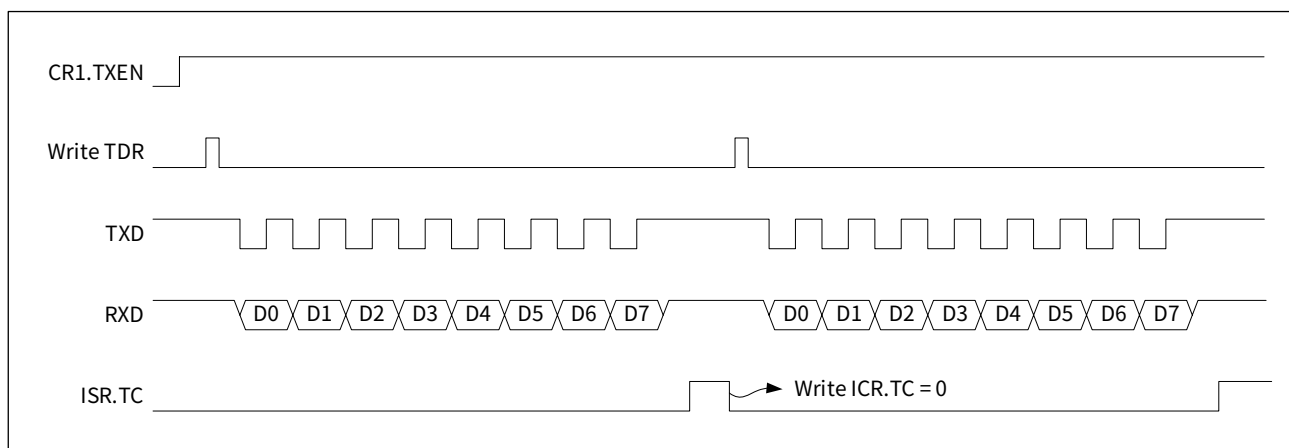
Where, UCLK is the transmission clock of the UART, and its source can be PCLK, LSE or LSI, which is selected by the SOURCE bit field of the control register UARTx_CR1.



16.3.2.2 Data transmission and reception

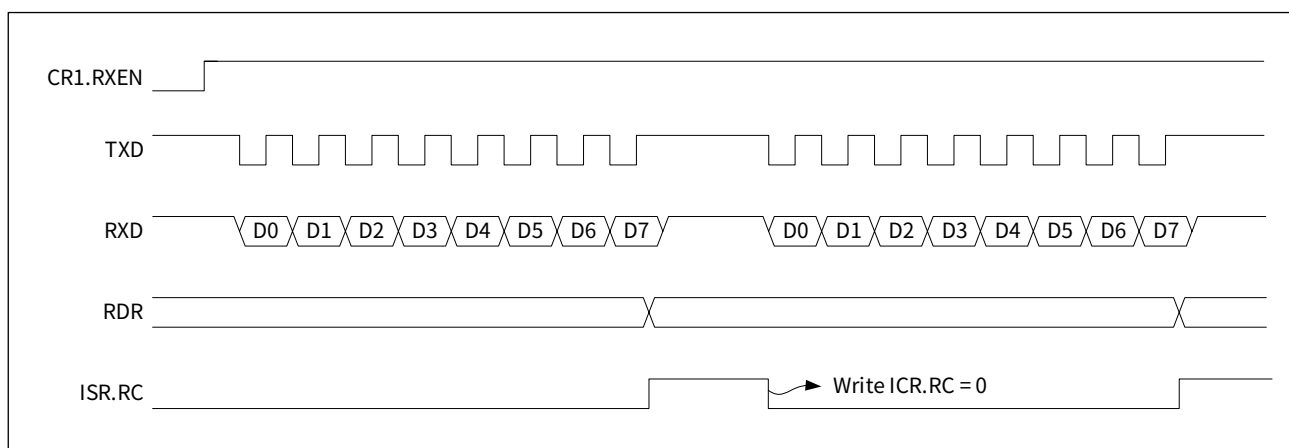
Set the UART transmitting enable bit `UARTx_CR1.TXEN` to 1 to make the UART work in the transmitting state. After writing the data to the `UARTx_TDR` register, the data will be serially output from the `UARTx_RXD` pin, and the `UARTx_TXD` pin will output a synchronous shift clock signal. When the data in the `UARTx_TDR` register and shift register have been transmitted, the transmit completion interrupt bit `UARTx_ISR.TC` is set by hardware.

Figure 16-2 Transmit timing in synchronous half-duplex mode (transmit two bytes of data)



Set the UART receiving enable bit `UARTx_CR1.RXEN` to 1 to make the UART work in the receiving state, the `UARTx_TXD` pin will output a synchronous shift clock signal, and the data will be serially input from the `UARTx_RXD` pin. After the data reception is completed, the reception completion interrupt flag `UARTx_ISR.RC` will be set by hardware, and the `UARTx_RDR` register can be read at this time. Before receiving the next byte, the `UARTx_ISR.RC` flag must be cleared to 0. After the RC flag is cleared, the next byte of data will be received immediately.

Figure 16-3 Receive timing in synchronous half-duplex mode (receive two bytes of data)



16.3.3 Asynchronous mode

16.3.3.1 Data frame format

In UART asynchronous communication, data is transmitted and received in the form of data frames, and a frame of data includes a start bit, a data field, an optional parity bit and a stop bit with a programmable width.

Start bit

The length of the start bit is fixed at 1 bit, and the judgment method of the start bit can be selected by falling edge or low level, which is selected by the START bit field of the control register UARTx_CR1. Generally, the falling edge is selected as the judgment method of the start bit. The low-level method to determine the start bit is mainly used in the low-power mode. Please refer to section [16.4 Low power mode](#).

Data field

The data word length can be set to 8 or 9 bits and is configured by controlling the CHLEN bit field of register UARTx_CR1. When parity is disabled (UARTx_CR1.PARITYEN is set to 0), the data word length can be set to 8 or 9 bits; when parity is enabled (UARTx_CR1.PARITYEN is set to 1), the data word length must be set to 9 bits, and if the character length is set to 8 bits, the PARITYEN bit field is automatically cleared. Please refer to [Table 16-2 Data frame structure](#).

The order in which the data bits are transmitted can be selected MSB or LSB first, selected by controlling the MSBF bit field of register UARTx_CR1.

Parity bit

The parity mode supports even parity and odd parity, which is selected by the PARITY bit field of the control register UARTx_CR1.

- Odd parity: The parity bit makes the total number of "1"s in the data bits and parity bits in a frame of data an odd number.
- Even parity: The parity bit makes the total number of "1" in the data bits and the parity bit in a frame of data an even number.

Stop bit

The length of the stop bit can be configured as 1, 1.5 or 2 bits, which is configured through the STOP bit field of the control register UARTx_CR1.

Table 16-2 Data frame structure

UARTx_CR1.CHLEN	UARTx_CR1.PARITYEN	Data frame
0	X	Start bit + 8-bit data + stop bit
1	0	Start bit + 9-bit data + stop bit
1	1	Start bit + 8-bit data + parity bit + stop bit



16.3.3.2 Fractional baud rate generator

Generation of baud rate

The receive and transmit baud rates of the UART are the same, generated by the same baud rate generator.

The baud rate generator supports 4 sampling modes: 16 times sampling, 8 times sampling, 4 times sampling and dedicated sampling. The specific sampling mode is selected by the OVER bit field of the control register UARTx_CR1.

1. OVER=00, set 16 times sampling, baud rate calculation formula:

$$\text{BaudRate} = \text{UCLK} / (16 \times \text{BRRI} + \text{BRRF})$$

UCLK is the transmission clock of UART, and its source can be PCLK, LSE or LSI. The specific source is selected by UARTx_CR1.SOURCE.

BRRI (UARTx_BRRI[15:0]) is the integer part of the baud rate counter and can be set in the range of 1 ~ 65535.

BRRF (UARTx_BRRF[3:0]) is the fractional part of the baud rate counter, and the settable range is 0 ~ 15.

Example 1:

When the frequency of the transmission clock UCLK is 24MHz, set BRRI=156 (ie UARTx_BRRI=0x9C), BRRF=4 (ie UARTx_BRRF=0x04), then:

$$\text{BaudRate} = 24000000 / (16 \times 156 + 4) = 9600 \text{ bps}$$

Example 2:

When the frequency of the transmission clock UCLK is 24MHz, it is required to configure BaudRate=115200bps, and calculate

$$16 \times \text{BRRI} + \text{BRRF} = 24000000 / 115200 = 208.33$$

then:

$$\text{BRRI} = 208.33 / 16 = 13.02, \text{ the nearest integer is: } 13 (0x0D)$$

$$\text{BRRF} = 0.02 \times 16 = 0.32, \text{ the nearest integer is: } 0 (0x00)$$

That is, you need to set UARTx_BRRI to 0x0D and UARTx_BRRF to 0x00

At this time, the actual baud rate BaudRate=115384.62bps, the error rate is 0.16%

2. OVER = 01, set 8 times sampling, baud rate calculation formula:

$$\text{BaudRate} = \text{UCLK} / (8 \times \text{BRRI})$$

3. OVER = 10, set 4 times sampling, baud rate calculation formula:

$$\text{BaudRate} = \text{UCLK} / (4 \times \text{BRRI})$$

4. OVER = 11, set dedicated sampling, baud rate calculation formula:

$$\text{BaudRate} = (256 \times \text{UCLK}) / \text{BRRI}$$

Caution:

Dedicated sampling is only suitable for UART communication at 2400bps, 4800bps or 9600bps baud rate when the transmission clock source is LSE or LSI.



Example 1:

The transmission clock UCLK selects LSE, the frequency is 32.768KHz, and the baud rate of 9600bps is required to be configured, then:

$$\text{BRR1} = 256 \times 32768 / 9600 = 873.81, \text{ the nearest integer is: } 874 \text{ (0x36A)}$$

That is, you need to set UARTx_BRR1 to 0x36A

At this time, the actual baud rate BaudRate=9597.95bps, the error is 0.02%

Example of baud rate setting

Table 16-3 to Table 16-9 list the setting values of the baud rate count register and the baud rate error when the frequency of the common MCU is used.

Table 16-3 UCLK is 4MHz baud rate setting example (OVER=00)

Baud rate (bps)	BRR1	BRRF	Actual baud rate	Error rate
4800	52	1	4801.92	0.04%
9600	26	1	9592.33	-0.08%
14400	17	6	14388.49	-0.08%
19200	13	0	19230.77	0.16%
38400	6	8	38461.54	0.16%
56000	4	7	56338.03	0.60%
57600	4	5	57971.01	0.64%
115200	2	3	114285.71	-0.79%
256000	1	0	250000	-2.34%

Table 16-4 UCLK is 8MHz baud rate setting example (OVER=00)

Baud rate (bps)	BRR1	BRRF	Actual baud rate	Error rate
4800	104	3	4799.04	-0.02%
9600	52	1	9603.84	0.04%
14400	34	12	14388.49	-0.08%
19200	26	1	19184.65	-0.08%
38400	13	0	38461.54	0.16%
56000	8	15	55944.06	-0.10%
57600	8	11	57553.96	-0.08%
115200	4	5	115942.03	0.64%
500000	1	0	500000	0.00%



Table 16-5 UCLK is 16MHz baud rate setting example (OVER=00)

Baud rate (bps)	BRR1	BRRF	Actual baud rate	Error rate
4800	208	5	4800.48	0.01%
9600	104	3	9598.08	-0.02%
14400	69	7	14401.44	0.01%
19200	52	1	19207.68	0.04%
38400	26	1	38369.30	-0.08%
56000	17	14	55944.06	-0.10%
57600	17	6	57553.96	-0.08%
115200	8	11	115107.91	-0.08%
1000000	1	0	1000000	0.00%

Table 16-6 UCLK is 24MHz baud rate setting example (OVER=00)

Baud rate (bps)	BRR1	BRRF	Actual baud rate	Error rate
4800	312	8	4800.00	0.00%
9600	156	4	9600.00	0.00%
14400	104	3	14397.12	-0.02%
19200	78	2	19200.00	0.00%
38400	39	1	38400.00	0.00%
56000	26	13	55944.06	-0.10%
57600	26	1	57553.96	-0.08%
115200	13	0	115384.62	0.16%
1500000	1	0	1500000	0.00%



Table 16-7 UCLK is 32MHz baud rate setting example (OVER=00)

Baud rate (bps)	BRR1	BRRF	Actual baud rate	Error rate
4800	416	11	4799.76	0.00%
9600	208	5	9600.96	0.01%
14400	138	14	14401.44	0.01%
19200	104	3	19196.16	-0.02%
38400	52	1	38415.37	0.04%
56000	35	11	56042.03	0.08%
57600	34	12	57553.96	-0.08%
115200	17	6	115107.91	-0.08%
2000000	1	0	2000000	0.00%

Table 16-8 UCLK is 48MHz baud rate setting example (OVER=00)

Baud rate (bps)	BRR1	BRRF	Actual baud rate	Error rate
4800	625	0	4800.00	0.00%
9600	312	8	9600.00	0.00%
14400	208	5	14401.44	0.01%
19200	156	4	19200.00	0.00%
38400	78	2	38400.00	0.00%
56000	53	9	56009.33	0.02%
57600	52	1	57623.05	0.04%
115200	26	1	115107.91	-0.08%
3000000	1	0	3000000	0.00%

Table 16-9 UCLK is 32.768KHz baud rate setting example (OVER=11)

Baud rate (bps)	BRR1	Actual baud rate	Error rate
2400	3495	2400.17	0.01%
4800	1748	4798.97	-0.02%
9600	874	9597.95	-0.02%



16.3.3.3 Transmit control

Data transmission

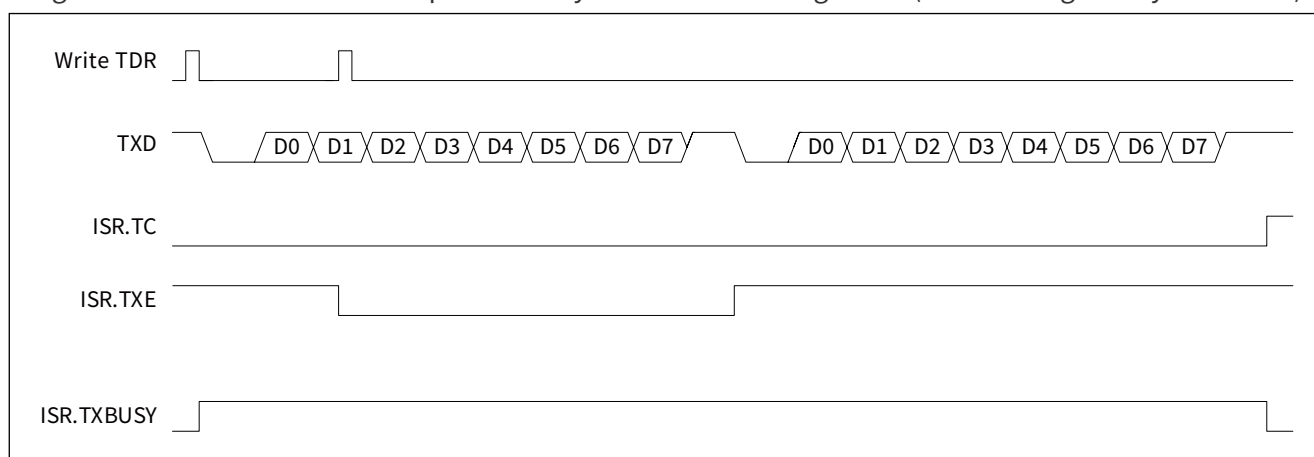
Set UARTx_CR1.TXEN to 1 to enable the transmit circuit. After writing data to the UARTx_TDR register, the data is transferred by hardware to the transmit shift register, which shifts the data out serially (MSB or LSB first). For the register configuration of the specific transmission process, please refer to section [16.6 Programming examples](#).

When the data is transferred to the transmission shift register by hardware, the transmit buffer empty interrupt flag UARTx_ISR.TXE will be set by hardware, indicating that the UARTx_TDR register is empty, and new data to be sent is allowed to be written to the UARTx_TDR register. While writing data to the UARTx_TDR register, the UARTx_ISR.TXE flag will be automatically cleared. If there is still untransmitted data in the transmission shift register at this time, the newly written data will be temporarily stored in the UARTx_TDR register. After the current data transmission is completed, the data in the UARTx_TDR register will be transferred by hardware to the transmission shift register. continue to transmit in the register.

When the data frame transmission in the transmission shift register is completed, if there is no data to be transmitted in the UARTx_TDR register (i.e. UARTx_ISR.TXE = 1), the transmission completion interrupt flag bit UARTx_ISR.TC will be set by hardware, and the UART transmission busy flag bit UARTx_ISR.TXBUSY will be cleared by hardware, indicating that all data transmission has been completed, and the UART transmitter is idle.

The timing diagram of the UART transmission control link is shown in the following figure:

Figure 16-4 Transmit control sequence in asynchronous working mode (transmitting two bytes of data)



Break frame

An break frame is a complete data frame consisting entirely of '0's (also '0' during the stop bit). At the end of the break frame, the transmitter inserts another 2 bits of logic '1' to ensure that the start bit of the next frame is recognised.

Setting UARTx_TDR.BREAK to 1 will send the break frame, the length of which is determined by the UART data word length. If there is data being sent when BREAK is written, an break frame will be sent on TXD after waiting for the current data to be sent.

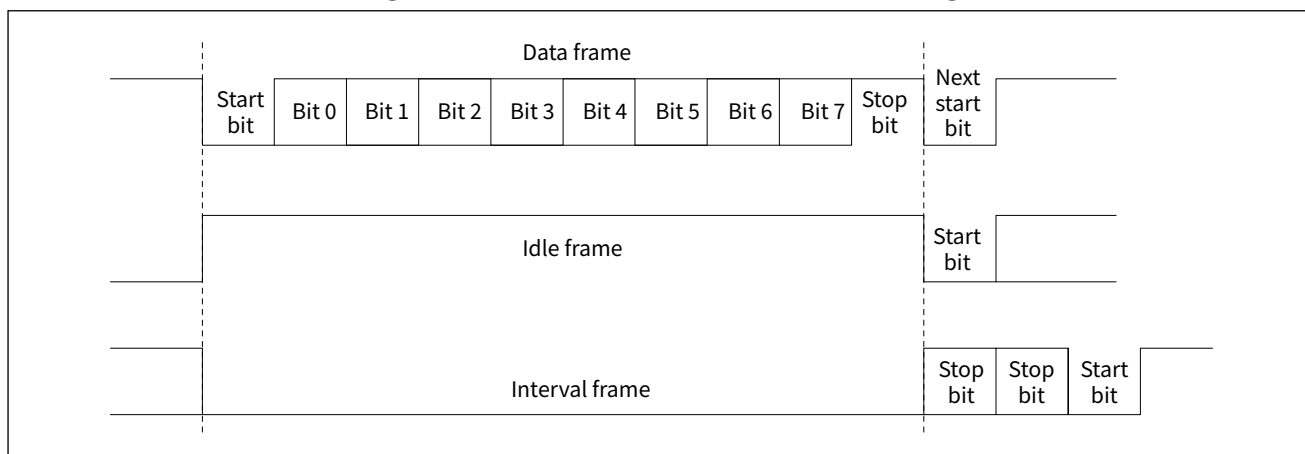
Idle frame

An idle frame is a complete data frame consisting entirely of '1's (also a '1' during the stop bit) followed by the start bit of the next frame containing the data.

Setting UARTx_TDR.IDLE to 1 will send an idle frame, the length of which is determined by the UART data word length. If there is data being sent when IDLE is written, an idle frame will be sent on TXD after waiting for the current data to be sent.

The following diagram shows the timing of break frame and idle frame, where the data word length is 8 bits:

Figure 16-5 Interval frame and idle frame timing



16.3.3.4 Receive control

Data reception

Set UARTx_CR1.RXEN to 1 to enable receive circuitry. When the receiver detects the start bit, it starts to receive data. During reception, the data is serially shifted into the receive shift register from the lowest bit or the highest bit, and transferred to the UARTx_RDR register in parallel. At this time, the data can be read out. For the register configuration of the specific receiving process, please refer to [16.6 Programming examples](#).

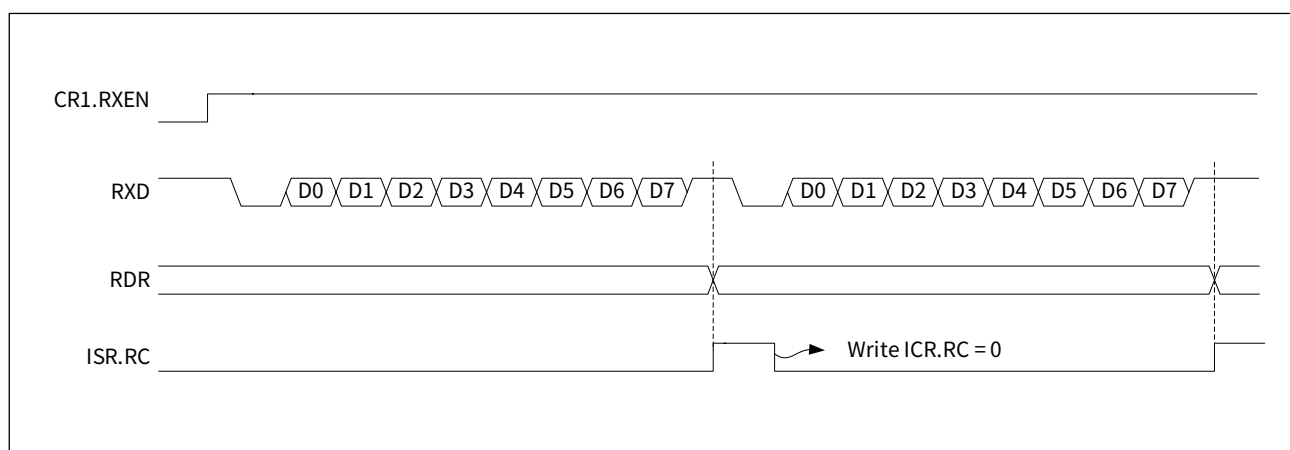
When the received data is transferred from the shift register to the UARTx_RDR register, the reception completion interrupt flag bit UARTx_ISR.RC will be set by hardware, indicating that the reception of a frame of data has been completed. In the user program, once the UARTx_ISR.RC flag is detected as 1, the UARTx_RDR register should be read as soon as possible, and the UARTx_ISR.RC flag should be cleared. If the UARTx_RDR register is not read in time, the newly received data will overwrite the data in the UARTx_RDR register, resulting in data loss.

When a data frame is received, the parity check will be performed automatically. If a parity check error is detected, the UARTx_ISR.PE flag will be set by hardware, indicating a parity check error.

When receiving a data frame, if the correct stop bit is not recognized within the expected time, it is determined that a frame structure error has occurred, and the UARTx_ISR.FE flag will be set by hardware.

The timing diagram of the UART reception control circuit is shown in the figure below:

Figure 16-6 Receive control timing in asynchronous working mode (receive two bytes of data)



Overflow error

If the UARTx_ISR.RC flag bit is not cleared and another character is received, an overflow error occurs and the UARTx_ISR.ORE flag bit is set by hardware. When an overflow error occurs, newly received data overwrites the data in the UARTx_RDR register, causing data loss.

If UARTx_IER.ORE is set to 1, an overflow interrupt is generated, and UARTx_ICR.ORE is set to 0 to clear the overflow flag.

Noise error

When noise is detected in a received frame, the UARTx_ISR.NE flag bit is hardware set on the rising edge of the UARTx_ISR.RC flag bit, and invalid data is transferred from the shift register to the UARTx_RDR register. If UARTx_IER.NE is set to 1, a noise interrupt is generated, and UARTx_ICR.NE is set to 0 to clear the noise flag. Note that noise errors are only valid when sampling by 16 or sampling by 8.

Receive data match

Set UARTx_CR2.RXMATCHEN to 1 to enable receive data match detection. When the received data matches the target received data configured in the UARTx_RXMATCH register, the UARTx_ISR.RXMATCH flag is set by hardware. If UARTx_IER.RXMATCH is set to 1, a receive data match interrupt is generated. If UARTx_ICR.RXMATCH is set to 0, the flag is cleared.

Break frame

When the UART receives a break frame, the RXBRK, RC, and FE flags are set, and the UARTx_RDR register receives a 0. If the corresponding interrupt is enabled, a corresponding interrupt request will be generated.

Idle frame

Idle character detection is initiated only when the reception of a character is completed. When an idle character is detected, the RXIDLE flag is set and an interrupt request will be generated if the corresponding interrupt is enabled.



16.3.3.5 Single-wire half-duplex mode

Set UARTx_CR1.SIGNAL to 1 to make the UART work in single-wire half-duplex mode. In this mode, the UARTx_TXD pin is used for data transmission and reception without occupying the UARTx_RXD pin (UARTx_RXD can be used as general IO). See [Table 16-1 UART ports configuration](#) for configuration.

After writing data to the UARTx_TDR register, the UARTx_TXD pin immediately enters the transmit state and the output UARTx_TDR data in the register. After the data transmission is complete, the UARTx_TXD pin returns to its normal receive state.

When no data is transmitted, the UARTx_TXD pin is in the receiving state. After the data is received, the reception completion flag UARTx_ISR.RC will be set by hardware. At this time, the UARTx_RDR register should be read as soon as possible, and the UARTx_ISR.RC flag should be cleared.

Caution:

Users should take appropriate application layer protection mechanisms to ensure that multiple masters do not send data to the bus at the same time.

16.3.3.6 Multi-machine communication

UART supports multi-machine communication mode. In this mode, there is one master and multiple slaves on the UART bus. Each slave has a unique slave address. During communication, the master will first send an address frame to address the slave. Only the slave with the matching address is activated and receives the data frame sent by the subsequent master.

Master transmits

In the multi-machine communication mode, the master needs to set the frame data length to 9 bits and disable parity check.

The highest bit of the UARTx_TDR.TDR determines whether the master sends an address frame or a data frame. When UARTx_TDR[8] is 1, it means that the master sends an address frame. When UARTx_TDR[8] is 0, it means that the master sends a data frame.

Slave receives

In the multi-machine communication mode, the slave needs to set the frame data length to 9 bits, disable parity check, and set UARTx_CR2.ADDREN to 1 to enable the slave address recognition, and the slave hardware automatically detects the address sent by the master and the local machine. whether the address matches.

If the address matches, the slave will save the received address frame into the UARTx_RDR register, the UARTx_ISR.RC flag is set by hardware, and the UARTx_ISR.SLVMATCH flag is set by hardware, and the slave receives the subsequent data frame sent by the master. During the communication process, the slave should pay attention to:

1. The application program queries UARTx_RDR[8] in the reception completion interrupt RC to determine whether it is an address frame or a data frame.
2. When the slave sends a data frame, it needs to set UARTx_TDR[8] to 0 to avoid being regarded as an address frame by other slaves.

If the addresses do not match, the slave will not receive the data frame sent by the master, nor will it generate a reception completion interrupt, and the set UARTx_ISR.SLVMATCH flag will be cleared.



Slave address and address mask

The slave address is configured by the UARTx_ADDR register, and the slave address should be configured to be unique. The UARTx_MASK register is an address mask. The slave address bit corresponding to the bit '1' in UARTx_MASK[7:0] participates in the slave address matching operation, and the slave address bit corresponding to the bit '0' does not participate in the slave address matching operation.

When all bits in UARTx_MASK[7:0] are set to '1', that is, write 0xFF to the UARTx_MASK register, then all 8-bit address bits of the slave address participate in the slave address matching, and the master can uniquely identify the slave.

When some bits in UARTx_MASK[7:0] are set to '0', the corresponding slave address bits do not participate in slave address matching, so as to realize that multiple slaves respond to the same address frame sent by the master, that is, the master addresses multiple slaves at the same time.

Example 1:

Set the slave A: UARTx_ADDR = 0xA0, UARTx_MASK = 0xFE

Set the slave B: UARTx_ADDR = 0xA1, UARTx_MASK = 0xFE

Set the slave C: UARTx_ADDR = 0xA2, UARTx_MASK = 0xFC

Set the slave D: UARTx_ADDR = 0xA3, UARTx_MASK = 0xFC

Set the slave E: UARTx_ADDR = 0xA5, UARTx_MASK = 0xFF

When the master transmits a 0xA0 or 0xA1 address frame, it can address slave A, slave B, slave C, and slave D

When the master transmits 0xA2 or 0xA3 address frame, it can address slave C, slave D

When the master transmits a 0xA4 address frame, no slave is addressed

When the master transmits a 0xA5 address frame, it can address the slave E

Broadcast address

In multi-machine communication, the address 0xFF is defined as the broadcast address. When the master transmits the broadcast address frame, the slaves of all addresses are addressed.

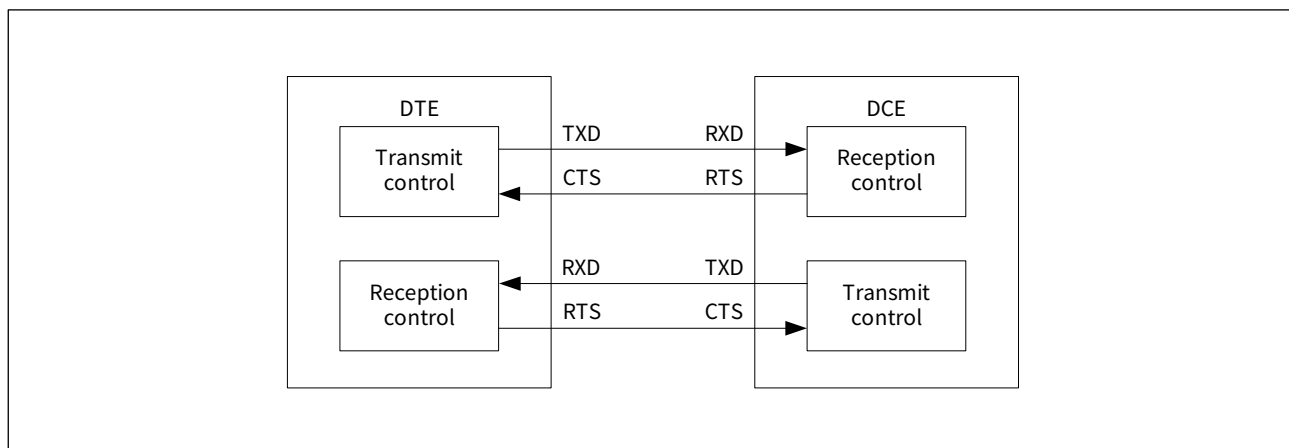


16.3.3.7 Hardware flow control

CW32L010 supports UART hardware flow control mode, that is, it supports request to transmit RTS and allow to transmit CTS, which is used to automatically control the transmission and reception of data.

In the data communication system, the schematic diagram of the communication hardware flow control between the data terminal equipment DTE and the data circuit terminating equipment DCE is shown in the following figure.

Figure 16-7 Schematic diagram of hardware flow control



Request to transmit RTS

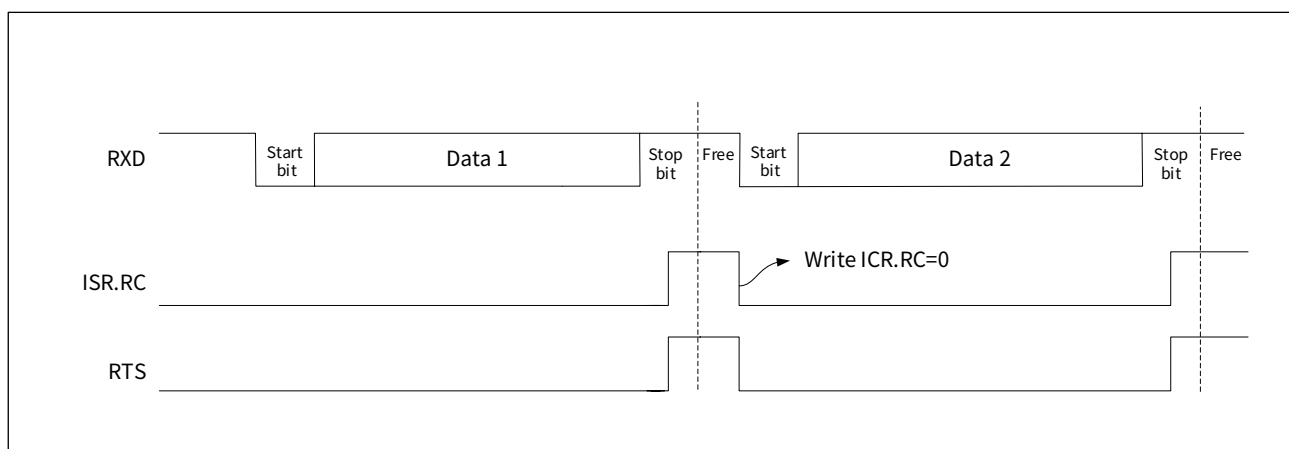
Set `UARTx_CR2.RTSN` to 1 to enable RTS flow control, RTS means request to transmit, and low level is active.

When the receiver is ready to receive new data (ie `UARTx_ISR.RC=0`), the RTS pin will output a low level, requesting the transmitting terminal to transmit a frame of data.

When the receiving terminal receives a frame of data (ie `UARTx_ISR.RC=1`), the RTS pin will output a high level to notify the transmitting terminal to suspend transmitting.

In the user application, after reading the data in the `UARTx_RDR` register, if you want to continue to receive the next frame of data, the software must clear the `UARTx_ISR.RC` flag bit. At this time, the RTS pin will output a low level, requesting the transmitting terminal to transmit the next frame of data.

Figure 16-8 RTS flow control sequence



Allow transmitting CTS

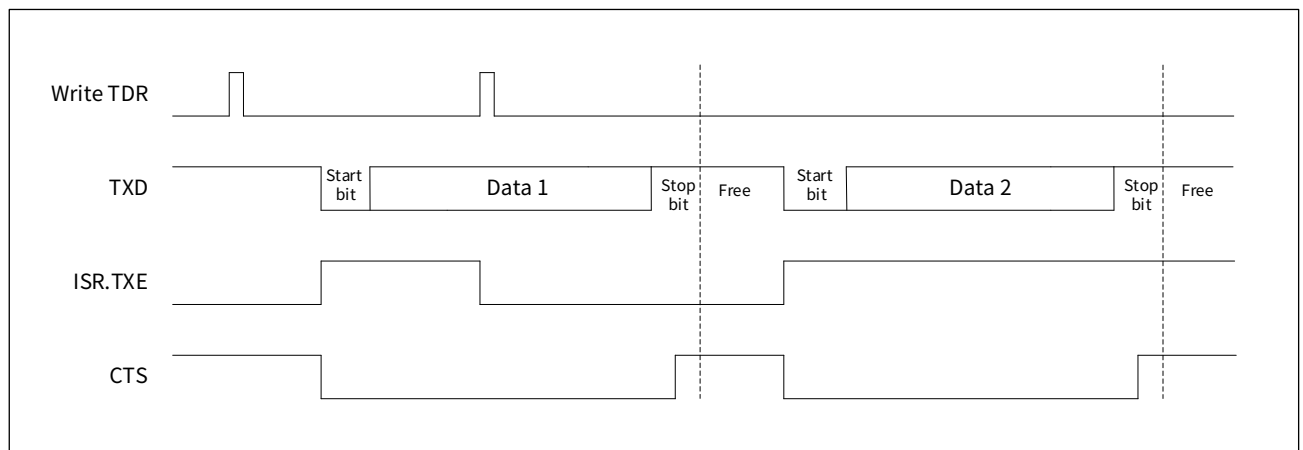
Set UARTx_CR.CTSEN to 1 to enable CTS flow control, CTS means transmission is allowed, and low level is active.

When the transmitting terminal detects that the CTS signal is low, if there is data to be sent (ie UARTx_ISR.TXE=0), and there is no data currently being transmitted, this frame of data will be transmitted.

When the transmitting terminal detects that the CTS signal is at a high level, if there is ongoing data transmission, the current transmission will continue, and the transmission will be stopped after the current transmission is completed.

In the CTS flow control process, the CTS signal level flag bit `UARTx_ISR.CTSLV` indicates the current level state of the CTS pin; the CTS signal change flag bit `UARTx_ISR.CTS` indicates whether the CTS signal has changed, when the CTS signal changes, The `UARTx_ISR.CTS` flag will be set by hardware. When the CTS interrupt is enabled (that is, set `UARTx_IER.CTS` to 1), an interrupt will be generated. Write `UARTx_ICR.CTS` to 0 to clear the interrupt flag.

Figure 16-9 CTS flow control sequence



16.3.3.8 RS485 driver enable

Set the DEM bit field of control register UARTx_CR3 to 1 to enable the RS485 driver enable function, at this time, the RTS pin will be DE (Driver Enable) function, the user needs to configure this pin for the multiplexing function of the digital output.

After enabling this function, user can activate the transmit mode of the external transceiver by DE signal, the polarity of DE signal can be configured through the DEP bit field of UARTx_CR3 register, and user can select the DE signal to be valid at high level or low level.

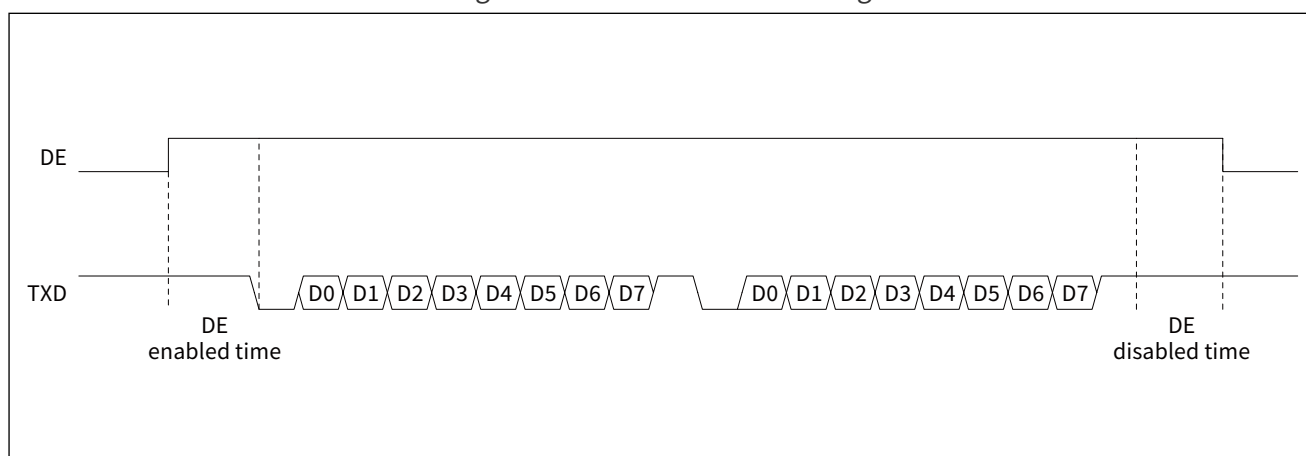
The driver enable time is the time between the valid level of output DE and the start of sending START bits, and the disable time is the time between the completion of sending the last STOP bit and the invalid level of output DE. Both the driver enable time and disable time are set through the DETIME bit field of the UARTx_CR3 register, and the calculation formula is as follows:

$$\text{Driver enable and disable time} = \text{Sample time} * \text{DETIME}$$

where the sample time is 1/4, 1/8, and 1/16 bit time, and the DE function is not supported with dedicated sampling (OVER=3).

When several consecutive data are sent, the DE signal will become invalid level only after the last STOP bit is sent. The following figure shows the timing when DE is valid at high level:

Figure 16-10 Driver enable timing



16.3.3.9 LoopBack mode

Set UARTx_CR2.LOOP to 1 to cause UART to operate in LoopBack mode. In this mode, the RXD signal is sourced from the internal TXD signal and does not occupy the external UARTx_TXD and UARTx_RXD pins (UARTx_TXD and UARTx_RXD can be used as general purpose GPIOs).

After writing data to the UARTx_RDR register, the RXD receives data internally to the UARTx_RDR register. After transmission and reception are complete, the TC and RC flag bits are set to 1. If interrupt is enabled, the corresponding interrupt request will be generated.

16.3.3.10 Input signal source/level shift

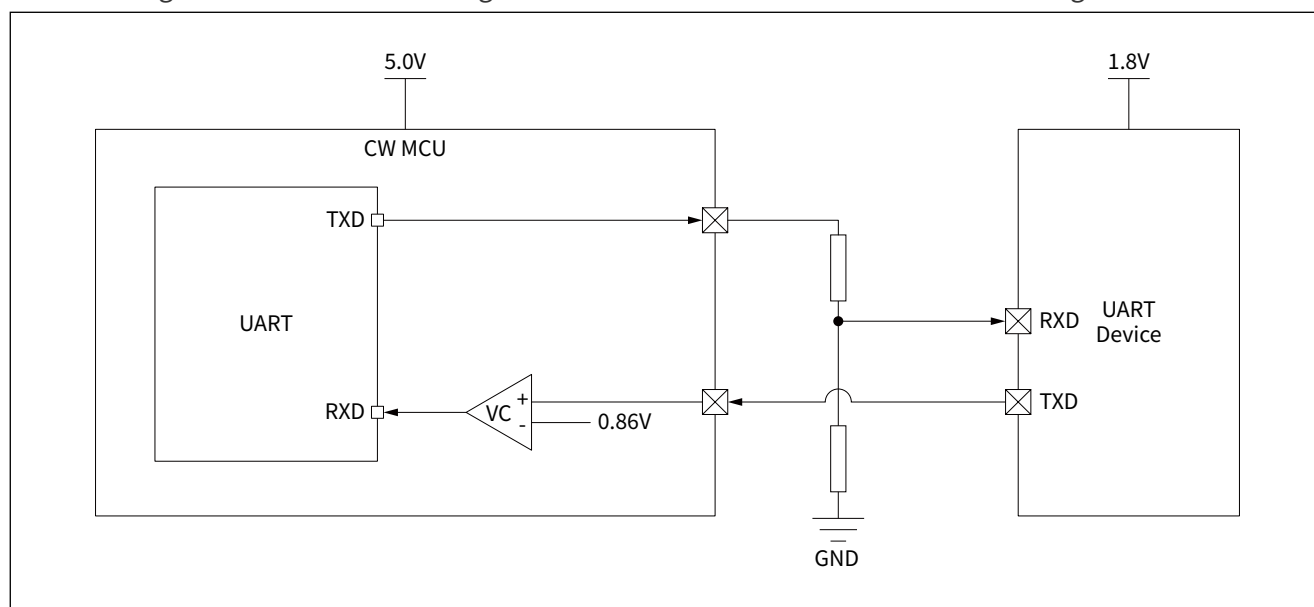
The RXD input signal of this module can be configured for multiple sources as shown in the following table:

Table 16-10 RXD input signal source

UARTx_CR2.RXSRC	UARTx_CR1.SIGNAL	UARTx_CR2.SWAP	RXD input signal source
000	0	0	RXD pin
		1	TXD pin
	1	0	TXD pin
		1	RXD pin
001	-	-	VC1_OUT
010			VC2_OUT

When the input signal source of RXD is configured as VCx_OUT, this module can communicate with UART devices that are lower than the operating voltage of this chip, and VCx acts as a level shifter in the communication link, and the typical working circuit is shown in the following figure:

Figure 16-11 Schematic diagram of communication between different voltage devices

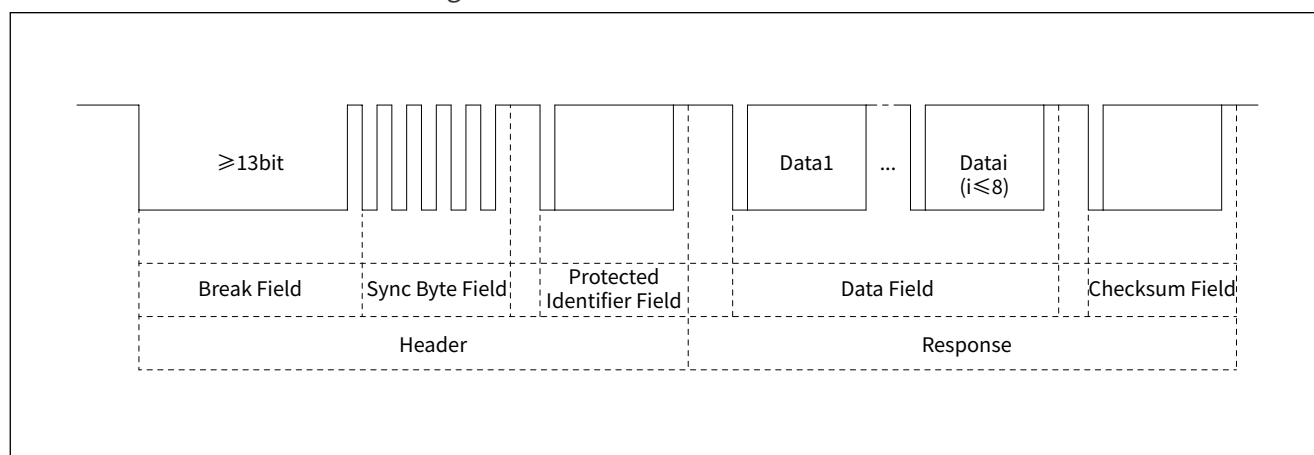


16.3.4 LIN Mode

LIN mode is selected by setting the LIN bit field in the UARTx_CR3 register to 1. In LIN mode, the SIGNAL, SYNC, STOP, and PARITY bits in the UARTx_CR1 register must be cleared.

A frame of LIN data contains two parts: Header and Response. The master task is responsible for transmitting the frame header, and the slave task receives the frame header and parses the information contained in the frame header, and reacts to the frame header (receive/send/ignore the response part). The frame structure of LIN is shown in the following figure:

Figure 16-12 Structure of the LIN frame



In a LIN frame, except for the Break Field, all other fields are transmitted in the format of a byte field. The byte field includes 1 start bit, 8 data bits, and 1 stop bit, and the data transmission is preceded by the least significant bit (LSB).

16.3.4.1 LIN transmit

The transmission of Break Field

The Break Field indicates the beginning of a frame of data. The number of bits in the LIN Break Field is set by UARTx_TDR.TDR. Note that the Break Field is a dominant level that lasts for at least 13 bits (based on the bit rate of the master node).

In LIN mode, set UARTx_TDR.BREAK to 1 and write the number of bits of the Break Field to UARTx_TDR.TDR to immediately transmit the Break Field of the specified length.

The transmission of the Sync Byte Field

Writing 0x55 to the UARTx_TDR register will send the Sync Byte Field.

The transmission of the Protected Identifier Field

The first 6 bits of the Protected Identifier Field are the frame ID, plus two parity bits called the Protected ID.

According to the parity formula, the parity bits corresponding to the frame ID are calculated, and then the protected ID is written to the data register UARTx_TDR, which will send the protected ID field.

16.3.4.2 LIN receive

Detection of the Break Field

Enabling LIN mode activates the Break Field detection circuit. The length threshold of the receive Break Field is set via the BRKL bit field of the control register UARTx_CR3.

After enabling the receive circuit, the hardware starts monitoring the start signal on the RXD pin input. After the start bit is detected, the circuit will sample the next bits. When the low width of the RXD communication line is detected to be greater than or equal to the Break Field length threshold set in the BRKL bit field and followed by the break delimiter, the UARTx_ISR.RXBRK flag bit will be set by the hardware, indicating that the Break Field reception is complete. If UARTx_IER.RXBRK is 1, an interrupt request will be generated.

If a high level has been sampled before the set Break Field length threshold, the detection circuitry will cancel the current detection and searches for the start bit again.

In LIN mode, when a frame structure error occurs, the receiver stops immediately until the detection circuit receives a '1' (Break Field incomplete) or receives a spacer (Break Field detected).

During Break Field detection, the frame structure error flag bit UARTx_ISR.FE is set in hardware. While clearing the UARTx_ISR.RXBRK flag bit, the UARTx_ISR.FE flag needs to be cleared and the UARTx_RDR register needs to be read to confirm that the data received is 0x00.

Detection of the Sync Byte Field

The slave node calculates the master node bit rate by receiving the Sync Byte Field from the master node, and readjusts its own bit rate according to the calculation result. The slave node should set the TIMCR bit field of the control register UARTx_CR2 to 0x6, configure it for automatic baud rate detection mode 2 and measure the frequency of the Sync Byte Field by counting the clock UCLK. The bit rate calculation formula is as follows:

$$1 \text{ bit time} = (\text{TIMCNT} + (\text{TIMARR} + 1) \times \text{ARR overflow times}) / (8 \times \text{UCLK})$$

If the UART is set to 16 times sampling (OVER=00), the adjusted baud rate counter value will be as follows:

$$\text{BRR1} = (\text{uint16\_t}) ((\text{TIMCNT} + (\text{TIMARR} + 1) \times \text{ARR overflow times}) / 8) \gg 4$$

$$\text{BRRF} = (\text{uint16\_t}) ((\text{TIMCNT} + (\text{TIMARR} + 1) \times \text{ARR overflow times}) / 8) \& 0x0F$$



16.3.5 Timer operating mode

The UART has a built-in timer module containing a 24bit up counter TIMCNT and a 24bit auto reload register TIMARR. The timer supports five operating modes, as shown in the table below:

Table 16-11 Timer operating modes

UARTx_CR2. TIMCR	Operating mode
010	Waiting for timeout detection mode
011	Receive idle detection mode
101	Auto baud rate detection mode 1
110	Auto baud rate detection mode 2
111	General purpose timer mode

16.3.5.1 Waiting for timeout detection mode

Set the TIMCR bit field of control register UARTx_CR2 to 0x2, and the timer operates in wait timeout detection mode with a timeout time:

$$(TIMARR+1)/f_{BaudRate}$$

Setting UARTx_CR2.TIMCR to 0x2 will immediately start the counter TIMCNT counting up from 0. The counting clock is the baud rate clock, and re-counts from 0 at every stop bit received; When the count reaches the reload value TIMARR the timer overflows, and at the same time the timer overflow flag bit UARTx_ISR.TIMOV is set by hardware, indicating that the receive wait timeout is reached and the timer stops automatically; If the timer overflow interrupt is enabled (set UARTx_IER.TIMOV to 1), the corresponding interrupt service routine will be executed, and the corresponding flag bit should be cleared before exiting the interrupt service routine to avoid entering the interrupt service routine repeatedly.

16.3.5.2 Receive idle detection mode

Set the TIMCR bit field of control register UARTx_CR2 to 0x3, and the timer operates in receive idle detection mode, with a detection idle time :

$$(TIMARR+1)/f_{BaudRate}$$

Whenever a stop bit is received, the counter TIMCNT is driven by the baud rate clock to count up from 0; When the count reaches the reloaded value TIMARR the timer overflows, and at the same time the timer overflow flag bit UARTx_ISR.TIMOV is set by the hardware, indicating that the receive idle timeout is reached, and the timer stops automatically; If the timer overflow interrupt is enabled (set UARTx_IER.TIMOV to 1), the corresponding interrupt service routine will be executed, and the corresponding flag bit should be cleared before exiting the interrupt service routine to avoid entering the interrupt service routine repeatedly.



16.3.5.3 Auto baud rate detection mode

The UART supports two auto baud rate detection modes. Set the TIMCR bit field of control register UARTx_CR2 to 0x5 to configure auto baud rate detection mode 1; set the TIMCR bit field of control register UARTx_CR2 to 0x6 to configure auto baud rate detection mode 2.

Auto baud rate detection mode 1

The falling edge of UARTx_RXD pin triggers the timer to start, the counter TIMCNT is driven by the counting clock UCLK to count up from 0, the timer overflows after counting to the reload value TIMARR, at the same time the timer overflow flag bit UARTx_ISR.TIMOV is set by hardware, and the counter keeps the TIMARR value unchanged.

The rising edge of the UARTx_RXD pin stops the timer counting while the UARTx_ISR.BAUD flag bit will be set by hardware, indicating that the auto baud rate detection has been completed.

Auto baud rate detection mode 2

Auto baud rate detection mode 2 is mainly used for clock synchronization in LIN mode, please refer to [16.3.4 LIN Mode](#).

The falling edge of the UARTx_RXD pin triggers the timer to start, the counter TIMCNT is driven by the counting clock UCLK to count up from 0, the timer overflows after counting to the reload value TIMARR, at the same time the timer overflow flag bit UARTx_ISR.TIMOV is set by hardware, and the counter keeps the TIMARR value unchanged.

The 4th falling edge of the UARTx_RXD pin stops the timer counting, while the UARTx_ISR.BAUD flag bit will be set by hardware, indicating that the automatic baud rate detection has been completed.

16.3.5.4 General purpose timer mode

The timer module built into the UART can be used as a general purpose timer, contains a 24bit up counter and is driven by the count clock UCLK.

Setting the TIMCR bit field of control register UARTx_CR2 to 0x7 will start the timer immediately and the counter TIMCNT will accumulate counts driven by the count clock UCLK. The timer will stop automatically after a timeout (TIMARR+1) of the UCLK clock.



16.4 Low-power mode

The UART controller works in a dual clock domain, supports normal data transmission and reception in DeepSleep mode, and wakes up the MCU back to Active mode through a reception completion interrupt.

If the source of the transmission clock UCLK is set as the low-speed clock, when the system enters the DeepSleep mode, the high-speed clock will stop, the low-speed clock will keep running, and the UART can still transmit and receive data normally (the baud rate only supports 2400bps, 4800bps and 9600bps).

To use the UART wake-up function in DeepSleep mode, you need to enable the UART reception completion interrupt (that is, set UARTx_IER.RC to 1) before entering DeepSleep mode. When data reception is completed, the reception completion interrupt will wake up the MCU and return to Active mode.

If the source of the transmission clock UCLK is set as the high-speed clock, when the system enters the DeepSleep mode, the high-speed clock will stop running and the UART will not receive data. At this time, the MCU can still be woken up through the GPIO interrupt to receive data in DeepSleep mode. The reference configuration steps are as follows:

Step 1: Enable the GPIO falling edge interrupt of the corresponding pin of UARTx_RXD;

Step 2: Set UARTx_CR1.START to 1, and select the RXD signal start bit determination method to be low level;

Step 3: Enable UART reception (that is, set UARTx_CR1.RXEN to 1);

Step 4: Enter DeepSleep mode;

Step 5: Wait for the master to transmit data, generate a GPIO falling edge interrupt, and wake up the MCU;

Step 6: Closed the GPIO interrupt function of the corresponding pin to RXD and wait for the completion of RXD reception.

For more information on the optimized design of low power mode entry and wake-up, please refer to chapter [3 Power Control \(PWR\) and Power Consumption](#).



16.5 UART interrupt

The UART controller supports 13 interrupt sources. When the UART interrupt trigger event occurs, the interrupt flag bit will be set by hardware. If the corresponding interrupt enable control bit is set, an interrupt request will be generated.

A UART module of CW32L010 uses a same system UART interrupt. Whether the UART interrupt generates interrupt jump is controlled by the corresponding bit of the interrupt enable setting register NVIC_IUSER of the Nested Vectored Interrupt Controller (NVIC).

In the user's UART interrupt service routine, the relevant UART interrupt flag bit should be queried for corresponding processing. Before exiting the interrupt service routine, the interrupt flag bit should be cleared to avoid repeatedly entering the interrupt routine.

The flag bits, interrupt enable bits, interrupt flag clear bits and clear methods of each interrupt source of the UART are shown in the following table:

Table 16-12 UART interrupt control

Interrupt event	Interrupt flag bit	Interrupt enable bit	Flag clear method
Transmit buffer empty	ISR.TXE	IER.TXE	Write UARTx_TDR register
Transmit completed	ISR.TC	IER.TC	Write 0 to ICR.TC
Receive completed	ISR.RC	IER.RC	Write 0 to ICR.RC
Idle character reception completed	ISR.RXIDLE	IER.RXIDLE	Write 0 to ICR.RXIDLE
Break Field reception completed	ISR.RXBRK	IER.RXBRK	Write 0 to ICR. RXBRK
Automatic baud rate detection completed	ISR.BAUD	IER.BAUD	Write 0 to ICR. BAUD
Timer overflow	ISR.TIMOV	IER.TIMOV	Write 0 to ICR. TIMOV
CTS signal change	ISR.CTS	IER.CTS	Write 0 to ICR.CTS
Frame structure error	ISR.FE	IER.FE	Write 0 to ICR.FE
Parity error	ISR.PE	IER.PE	Write 0 to ICR.PE
Noise flag	ISR.NE	IER.NE	Write 0 to ICR.NE
Overflow error	ISR.ORE	IER.ORE	Write 0 to ICR.ORE
Received data match	ISR.RXMATCH	IER.RXMATCH	Write 0 to ICR.RXMATCH



16.6 Programming examples

16.6.1 Asynchronous full-duplex programming examples

16.6.1.1 Transmit data in query mode

- Step 1: Set SYSCTRL_AHBEN.GPIOx to 1, SYSCTRL_APBENx.UARTx to 1, enable the GPIO clock and UART configuration clock corresponding to the UART pin;
- Step 2: Configure the UARTx_TXD pin to the push-pull multiplexed output mode. For the specific register configuration steps, please refer to chapter [8 General-purpose input/output \(GPIO\)](#);
- Step 3: Set UARTx_CR1.SYNC to 0, and configure UARTx to be asynchronous full-duplex communication mode;
- Step 4: Configure the data frame;
1. Start bit determination method: Configure UARTx_CR1.START.
 2. Data word length: Configure UARTx_CR1.CHLEN.
 3. Check bit: Configure UARTx_CR1.PARITYEN and UARTx_CR1.PARITY.
 4. Stop bit: Configure UARTx_CR1.STOP.
- Step 5: Configure UARTx_CR1.SOURCE and select the transmission clock source;
- Step 6: Configure UARTx_CR1.OVER and select sampling mode;
- Step 7: Configure the UARTx_BRRI and UARTx_BRRF registers and configure the baud rate. For details, please refer to section [16.3.3.2 Fractional baud rate generator](#);
- Step 8: Set UARTx_CR1.TXEN to 1 to enable transmit;
- Step 9: Set UARTx_ICR.TC to 0, clear the transmit completion flag;
- Step 10: Write a frame of data to be transmitted into the UARTx_TDR register;
- Step 11: Query and wait for the UARTx_ISR.TC flag to be set to 1 to confirm that one frame of data is transmitted;
- Step 12: Repeat steps 9 ~ 12 to transmit the next frame of data.



16.6.1.2 Receive data in query mode

- Step 1: Set SYSCTRL_AHBEN.GPIOx to 1, SYSCTRL_APBENx.UARTx to 1, enable the GPIO clock and UART configuration clock corresponding to the UART pin;
- Step 2: Configure the RXD pin as the pull-up input multiplexed mode. For the specific register configuration steps, please refer to chapter [8 General-purpose input/output \(GPIO\)](#);
- Step 3: Set UARTx_CR1.SYNC to 0, and configure UARTx to be asynchronous full-duplex communication mode;
- Step 4: Configure the data frame;
1. Start bit determination method: Configure UARTx_CR1.START.
 2. Data word length: Configure UARTx_CR1.CHLEN.
 3. Check bit: Configure UARTx_CR1.PARITYEN and UARTx_CR1.PARITY.
 4. Stop bit: Configure UARTx_CR1.STOP.
- Step 5: Configure UARTx_CR1.SOURCE and select the transmission clock source;
- Step 6: Configure UARTx_CR1.OVER and select sampling mode;
- Step 7: Configure the UARTx_BRRI and UARTx_BRRF registers and configure the baud rate. For details, please refer to section [16.3.3.2 Fractional baud rate generator](#);
- Step 8: Write 0x00 to the UARTx_ICR register to clear all flags;
- Step 9: Set UARTx_CR1.RXEN to 1 to enable receive;
- Step 10: Query and wait for the UARTx_ISR.RC flag to be set to 1 to confirm that one frame of data has been received;
- Step 11: Query the error flags UARTx_ISR.PE and UARTx_ISR.FE to confirm whether the data is valid. If the data is invalid, perform error processing. If the data is valid, read the UARTx_RDR register and save the data;
- Step 12: Set UARTx_ICR.RC to 0, clear the reception completion flag;
- Step 13: Repeat steps 10 ~ 13 to receive the next frame of data.



16.6.1.3 Transmit data in interrupt mode

- Step 1: Set SYSCTRL_AHBEN.GPIOx to 1, SYSCTRL_APBENx.UARTx to 1, enable GPIO clock and UART configuration clock corresponding to UART pins;
- Step 2: Configure the TXD pin to the push-pull multiplexed output mode. For the specific register configuration steps, please refer to chapter [8 General-purpose input/output \(GPIO\)](#);
- Step 3: Set UARTx_CR1.SYNC to 0, and configure UARTx to be asynchronous full-duplex communication mode;
- Step 4: Configure the data frame;
1. Start bit determination method: Configure UARTx_CR1.START.
 2. Data word length: Configure UARTx_CR1.CHLEN.
 3. Check bit: Configure UARTx_CR1.PARITYEN and UARTx_CR1.PARITY.
 4. Stop bit: Configure UARTx_CR1.STOP.
- Step 5: Configure UARTx_CR1.SOURCE and select the transmission clock source;
- Step 6: Configure UARTx_CR1.OVER and select sampling mode;
- Step 7: Configure the UARTx_BRRI and UARTx_BRRF registers and configure the baud rate. For details, For details, please refer to section [16.3.3.2 Fractional baud rate generator](#);
- Step 8: Configure the NVIC controller, see chapter [5 Interrupt](#);
- Step 9: Set UARTx_IER.TXE to 1 to enable transmit buffer empty interrupt;
- Step 10: Set UARTx_CR1.TXEN to 1 to enable transmit;
- Step 11: Write a frame of data to be transmitted into the UARTx_TDR register;
- Step 12: The data starts to transmit, the transmit buffer is empty, and the interrupt service function is entered: query and judge the UARTx_ISR.TXE flag bit, if the flag bit is 1, write a new frame of data to the UARTx_TDR register;
- Step 13: Query and wait for the UARTx_ISR.TXBUSY flag to be cleared and turn off the UART.



16.6.1.4 Receive data in interrupt mode

- Step 1: Set SYSCTRL_AHBEN.GPIOx to 1, SYSCTRL_APBENx.UARTx to 1, enable the GPIO clock and UART configuration clock corresponding to the UART pin;
- Step 2: Configure the RXD pin as the pull-up input multiplexed mode. For the specific register configuration steps, please refer to chapter [8 General-purpose input/output \(GPIO\)](#);
- Step 3: Set UARTx_CR1.SYNC to 0, and configure UARTx to be asynchronous full-duplex communication mode;
- Step 4: Configure the data frame;
1. Start bit determination method: Configure UARTx_CR1.START.
 2. Data word length: Configure UARTx_CR1.CHLEN.
 3. Check bit: Configure UARTx_CR1.PARITYEN and UARTx_CR1.PARITY.
 4. Stop bit: Configure UARTx_CR1.STOP.
- Step 5: Configure UARTx_CR1.SOURCE and select the transmission clock source;
- Step 6: Configure UARTx_CR1.OVER and select sampling mode;
- Step 7: Configure the UARTx_BRR1 and UARTx_BRRF registers and configure the baud rate. For details, please refer to section [16.3.3.2 Fractional baud rate generator](#);
- Step 8: Configure the NVIC controller, see chapter [5 Interrupt](#);
- Step 9: Set UARTx_IER.RC to 1 to enable receive completion interrupt;
- Step 10: Set UARTx_CR1.RXEN to 1 to enable receive;
- Step 11: Wait for the completion of receiving a frame of data, and enter the interrupt service function: query the UARTx_ISR.PE and UARTx_ISR.FE flag bits to confirm whether the data is valid, if invalid, perform error handling; if valid, query and judge the UARTx_ISR.RC flag bit, If the flag bit is 1, read the UARTx_RDR register and save the data, and clear the flag bit.



16.6.2 Synchronous half-duplex programming examples

16.6.2.1 Transmit data in query mode

- Step 1: Set `SYSCTRL_AHBEN.GPIOx` to 1, `SYSCTRL_APBENx.UARTx` to 1, enable GPIO clock and UART configuration clock;
- Step 2: Configure the `UARTx_TXD` and `UARTx_RXD` pins as push-pull multiplexed output mode, and configure the slave chip select pin `NCS` as push-pull output mode. For the specific register configuration steps, please refer to chapter [8 General-purpose input/output \(GPIO\)](#);
- Step 3: Set `UARTx_CR1.SYNC` to 1, and configure UARTx to be synchronous half-duplex communication mode;
- Step 4: Configure `UARTx_CR1.SOURCE`, select the transmission clock source, and configure the baud rate. For details, please refer to [16.3.2 Synchronous mode](#);
- Step 5: Pull down the `NCS` signal and select a slave;
- Step 6: Set `UARTx_CR1.TXEN` to 1 to enable transmit;
- Step 7: Set `UARTx_ICR.TC` to 0, and clear the transmission completion flag;
- Step 8: Write the 8-bit data to be transmitted to the `UARTx_TDR` register;
- Step 9: Query and wait for the `UARTx_ISR.TC` flag bit to be set to 1 to confirm that the data transmission is completed;
- Step 10: Repeat steps 7 ~ 10 to transmit the next frame of data;
- Step 11: Pull up the `NCS` signal to end the communication.

16.6.2.2 Receive data in query mode

- Step 1: Set `SYSCTRL_AHBEN.GPIOx` to 1, `SYSCTRL_APBENx.UARTx` to 1, enable GPIO clock and UART configuration clock;
- Step 2: Configure the `UARTx_TXD` and `UARTx_RXD` pins as push-pull multiplexed output mode, and configure the slave chip select pin `NCS` as push-pull output mode. For the specific register configuration steps, please refer to chapter [8 General-purpose input/output \(GPIO\)](#);
- Step 3: Set `UARTx_CR1.SYNC` to 1, and configure UARTx to be synchronous half-duplex communication mode;
- Step 4: Configure `UARTx_CR1.SOURCE`, select the transmission clock source, and configure the baud rate. For details, please refer to section [16.3.2 Synchronous mode](#);
- Step 5: Pull down the `NCS` signal and select a slave;
- Step 6: Write 0x00 to `UARTx_ICR` to clear all flags;
- Step 7: Set `UARTx_CR1.RXEN` to 1 to enable reception;
- Step 8: Query and wait for the `UARTx_ISR.RC` flag to be set to 1 to confirm that the data reception is completed;
- Step 9: Read the `UARTx_RDR` register and save the data;
- Step 10: Set `UARTx_ICR.RC` to 0, clear the reception completion flag;
- Step 11: Repeat steps 8 ~ 11 to receive the next 8-bit data;
- Step 12: Pull up the `NCS` signal to end the communication.



16.6.3 Single-wire half-duplex programming examples

16.6.3.1 Transmit data in query mode

- Step 1: Set SYSCTRL_AHBEN.GPIOx to 1, SYSCTRL_APBENx.UARTx to 1, enable the GPIO clock and UART configuration clock corresponding to the UART pin;
- Step 2: Configure the UARTx_TXD pin to open-drain multiplexed output mode, and connect an external pull-up resistor. For the specific register configuration steps, please refer to chapter [8 General-purpose input/output \(GPIO\)](#);
- Step 3: Set UARTx_CR1.SIGNAL to 1, and configure UARTx to be single-wire half-duplex communication mode;
- Step 4: Configure the data frame;
1. Start bit determination method: Configure UARTx_CR1.START.
 2. Data word length: Configure UARTx_CR1.CHLEN.
 3. Check bit: Configure UARTx_CR1.PARITYEN and UARTx_CR1.PARITY.
 4. Stop bit: Configure UARTx_CR1.STOP.
- Step 5: Configure UARTx_CR1.SOURCE and select the transmission clock source;
- Step 6: Configure UARTx_CR1.OVER and select sampling mode;
- Step 7: Configure the UARTx_BRRI and UARTx_BRRF registers and configure the baud rate. For details, please refer to [16.3.3.2 Fractional baud rate generator](#);
- Step 8: Set UARTx_CR1.TXEN to 1 and UARTx_CR1.RXEN to 1 to enable transmit and receive;
- Step 9: Set UARTx_ICR.TC to 0, and clear the transmission completion flag;
- Step 10: Write a frame of data to be transmitted into the UARTx_TDR register;
- Step 11: Query and wait for the UARTx_ISR.TC flag bit to be set to 1 to confirm that the data transmission is completed;
- Step 12: Repeat steps 9 ~ 12 to transmit the next frame of data.



16.6.3.2 Receive data in query mode

- Step 1: Set SYSCTRL_AHBEN.GPIOx to 1, SYSCTRL_APBENx.UARTx to 1, enable the GPIO clock and UART configuration clock corresponding to the UART pin;
- Step 2: Configure the UARTx_TXD pin to open-drain multiplexed output mode, and connect an external pull-up resistor. For the specific register configuration steps, please refer to chapter [8 General-purpose input/output \(GPIO\)](#);
- Step 3: Set UARTx_CR1.SIGNAL to 1, and configure UARTx to be single-wire half-duplex communication mode;
- Step 4: Configure the data frame;
1. Start bit determination method: Configure UARTx_CR1.START.
 2. Data word length: Configure UARTx_CR1.CHLEN.
 3. Check bit: Configure UARTx_CR1.PARITYEN and UARTx_CR1.PARITY.
 4. Stop bit: Configure UARTx_CR1.STOP.
- Step 5: Configure UARTx_CR1.SOURCE and select the transmission clock source;
- Step 6: Configure UARTx_CR1.OVER and select sampling mode;
- Step 7: Configure the UARTx_BRR1 and UARTx_BRRF registers and configure the baud rate. For details, please refer to section [16.3.3.2 Fractional baud rate generator](#);
- Step 8: Write 0x00 to UARTx_ICR to clear all flags;
- Step 9: Set UARTx_CR1.TXEN to 1 and UARTx_CR1.RXEN to 1 to enable transmit and receive;
- Step 10: Query and wait for the UARTx_ISR.RC flag to be set to 1 to confirm that one frame of data has been received;
- Step 11: Query the error flags UARTx_ISR.PE and UARTx_ISR.FE to confirm whether the data is valid. If the data is invalid, perform error processing. If the data is valid, read the UARTx_RDR register and save the data;
- Step 12: Set UARTx_ICR.RC to 0, clear the reception completion flag;
- Step 13: Repeat steps 10 ~ 13 to receive the next frame of data.



16.7 List of registers

UART1 base address: UART1_BASE = 0x4000 0C00

UART2 base address: UART2_BASE = 0x4000 1000

UART3 base address: UART3_BASE = 0x4000 2000

Table 16-13 List of UART registers

Register name	Register address	Register description
UARTx_CR1	UARTx_BASE + 0x00	Control register 1
UARTx_CR2	UARTx_BASE + 0x04	Control register 2
UARTx_CR3	UARTx_BASE + 0x38	Control register 3
UARTx_BRR1	UARTx_BASE + 0x0C	Baud rate counter integer part register
UARTx_BRRF	UARTx_BASE + 0x10	Baud rate counter fractional part register
UARTx_TIMARR	UARTx_BASE + 0x14	Timer reload register
UARTx_TIMCNT	UARTx_BASE + 0x18	Timer count register
UARTx_IER	UARTx_BASE + 0x08	Interrupt enable register
UARTx_ISR	UARTx_BASE + 0x1C	Interrupt flag register
UARTx_ICR	UARTx_BASE + 0x20	Interrupt flag clear register
UARTx_RDR	UARTx_BASE + 0x24	Receive data register
UARTx_TDR	UARTx_BASE + 0x28	Transmit data register
UARTx_ADDR	UARTx_BASE + 0x30	Slave address register
UARTx_MASK	UARTx_BASE + 0x34	Slave address mask register
UARTx_RXMATCH	UARTx_BASE + 0x3C	Receive data match register



16.8 Register descriptions

See section [1 Documentation conventions](#) for abbreviations used in register descriptions.

16.8.1 UARTx_CR1 Control register 1

Address offset: 0x00 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:15	RFU	-	Reserved bits, please keep the default value
14	SYNC	RW	Transmission sync-async mode settings 0: Working in asynchronous mode 1: Working in synchronous mode Caution: SYNC bit must be cleared in LIN mode.
13:12	SOURCE	RW	UART transmission clock source configuration 00: UCLK from PCLK 01: UCLK from PCLK 10: UCLK from LSE 11: UCLK from LSI
11	SIGNAL	RW	Single bus mode enable control 0: Disable, transmit data via TXD port, receive data via RXD port 1: Enable, transmit and receive data through TXD port Caution: In LIN mode, this control bit must be kept 0.
10:9	OVER	RW	Sampling mode settings 00: 16 times sampling, Baud = $UCLK / (BRR1 \times 16 + BRRF)$ 01: 8 times sampling, Baud = $UCLK / (BRR1 \times 8)$ 10: 4 times sampling, Baud = $UCLK / (BRR1 \times 4)$ 11: dedicated sampling, Baud = $UCLK \times 256 / BRR1$ Caution 1: Dedicated sampling is only suitable for 2400, 4800, 9600 communication when the transmission clock is LSI/LSE Caution 2: When BRRF is a non-zero value, 16 times sampling is automatically adopted
8	START	RW	RXD signal START bit determination mode settings 0: RXD signal falling edge 1: RXD signal low level
7	MSBF	RW	Data frame high and low bit order selection 0: The least significant bit (LSB) is sent and received first 1: The most significant bit (MSB) is sent and received first
6	CHLEN	RW	Character length 0: 8bit 1: 9bit Caution: If parity is enabled, the last 1Bit of the character is the parity bit.



Bit field	Name	Permission	Function description
5:4	STOP	RW	Stop bit length settings 00: 1 bit 01: 1.5 bit 10: 2 bit 11: Reserved <i>Caution 1: STOP bit must be cleared in synchronous mode.</i> <i>Caution 2: STOP bit must be cleared in LIN mode.</i>
3	PARITYEN	RW	Parity enable 0: Disabled 1: Enabled <i>Caution: Parity enable is automatically cleared when the character length is 8bit.</i>
2	PARITY	RW	Parity selection 0: Even check 1: Odd check
1	RXEN	RW	Receive circuit enable control 0: Disabled 1: Enabled
0	TXEN	RW	Transmit circuit enable control 0: Disabled 1: Enabled



16.8.2 UARTx_CR2 Control register 2

Address offset: 0x04 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:18	RFU	-	Reserved bits, please keep the default value
17:15	RXSRC	RW	RXD input signal source configuration 000: UARTx_RXD pin 001: VC1_OUT 010: VC2_OUT Other: Reserved
14	LOOP	RW	LoopBack mode enable control 0: RXD select external RXD Pin 1: RXD select internal TXD
13	ADCTX	RW	Transmit buffer empty triggers ADC enable control 0: Disabled 1: Enabled
12	ADCRX	RW	Receive data completion trigger ADC enable control 0: Disabled 1: Enabled
11	SWAP	RW	TXD/RXD pin-swap enable control 0: No pin-swap 1: Pin swap
10:8	TIMCR	RW	Timer operating mode configuration 000: Stop timer immediately 010: Wait timeout detection mode Start timer immediately starts counting from 0 and starts counting from 0 every time a STOP bit is received; count clock is baudrate clock; automatically stops after timeout. 011: Receive idle detection mode Counts from 0 for each STOP bit received; count clock is baudrate clock; automatically stops after timeout. 101: Auto baud rate detection mode 1 RXD falling edge causes timer to start counting from 0, rising edge stops timer; counting clock is UCLK; detection is completed automatically stops. 110: Auto baud rate detection mode 2 RXD falling edge causes timer to start counting from 0, 4th falling edge stops timer; counting clock is UCLK; detection is completed automatically stops. 111: General timing function Start timer immediately starts counting from 0; count clock is UCLK; automatically stops after timeout (TIMARR+1) UCLK.
7:6	RFU	-	Reserved bits, please keep the default value



Bit field	Name	Permission	Function description
5	TXINV	RW	TXD pin outputs signal inversion control 0: The TXD pin outputs the non-inverting signal 1: The TXD pin outputs the inverted signal
4	RXINV	RW	RXD pin inputs signal inversion control 0: The RXD pin signal is directly transmitted to the receiving circuit 1: The RXD pin signal is inverted and transmitted to the receiving circuit
3	RTSEN	RW	RTS hardware signal enable control 0: Disabled 1: Enabled
2	CTSEN	RW	CTS hardware signal enable control 0: Disabled 1: Enabled
1	RXMATCHEN	RW	Receive data matching detection enable control 0: Disabled 1: Enabled
0	ADDREN	RW	Slave address identification enable control 0: Disabled 1: Enabled



16.8.3 UARTx_CR3 Control register 3

Address offset: 0x38 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:9	RFU	-	Reserved bits, please keep the default value
8	BRKL	RW	LIN mode, receive break field length threshold 0: 10bit 1: 11bit
7	LIN	RW	LIN operating mode enabled 0: Working in UART mode 1: Working in LIN mode
6:2	DETIME	RW	Driver enabled and disabled time After outputting the DE active level and waiting for this time, the START bit starts to be transmitted; After the STOP bit is transmitted and waiting for this time, output DE invalid level. This time = Sampling time * DETIME <i>Caution 1: The sampling time is 1/4, 1/8, 1/16 bit time;</i> <i>Caution 2: The DE function is not supported when OVER = 3 (dedicated sampling).</i>
1	DEP	RW	Driver output signal polarity configuration 0: DE signal is valid at high level 1: DE signal is valid at low level
0	DEM	RW	Driver enabled configuration 0: Disable driver function, RTS pin for RTS function 1: Enable driver function, RTS pin for DE function <i>Caution: This control bit has higher priority than UARTx_CR2.RTSEN.</i>

16.8.4 UARTx_BRRI Baud rate counter integer part register

Address offset: 0x0C Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	BRRI	RW	Baud rate counter integer part



16.8.5 UARTx_BRRF Baud rate counter fractional register

Address offset: 0x10 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:4	RFU	-	Reserved bits, please keep the default value
3:0	BRRF	RW	Baud rate counter fractional part

16.8.6 UARTx_TIMARR Timer reload register

Address offset: 0x14 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:24	RFU	-	Reserved bits, please keep the default value
23:0	TIMARR	RW	Timer reload values

16.8.7 UARTx_TIMCNT Timer count register

Address offset: 0x18 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:24	RFU	-	Reserved bits, please keep the default value
23:0	TIMCNT	RO	Timer count values



16.8.8 UARTx_TDR Transmit data register

Address offset: 0x28 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:11	RFU	-	Reserved bits, please keep the default value
10	BREAK	WO	Transmit break frame 0: Transmit normal data 1: Transmit break frame <i>Caution: The Break Field length for LIN mode is determined by the value of UARTx_TDR[8:0]; the Break Field length for UART mode is determined by the UART word length.</i>
9	IDLE	WO	Transmit idle frame 0: Transmit normal data 1: Transmit idle frame <i>Caution 1: Idle frame length is determined by the UART word length.</i> <i>Caution 2: Idle frame has higher priority than Break Frame.</i>
8:0	TDR	WO	Length of data to be transmitted or LIN mode Break Field <i>Caution: When sending Break Field in LIN mode, the data written to this register is used as the number of bits in the Break Field.</i>

16.8.9 UARTx_RDR Receive data register

Address offset: 0x24 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:9	RFU	-	Reserved bits, please keep the default value
8:0	RDR	RO	Received data



16.8.10 UARTx_IER Interrupt enable register

Address offset: 0x08 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:13	RFU	-	Reserved bits, please keep the default value
12	RXMATCH	RW	Receive data match interrupt enable control 0: Disabled 1: Enabled
11	ORE	RW	Overflow interrupt enable control 0: Disabled 1: Enabled
10	NE	RW	Noise interrupt enable control 0: Disabled 1: Enabled
9	PE	RW	Parity error interrupt enable control 0: Disabled 1: Enabled
8	FE	RW	Frame structure error interrupt enable control 0: Disabled 1: Enabled
7	CTS	RW	CTS signal change interrupt enable control 0: Disabled 1: Enabled
6	TIMOV	RW	Timer overflow interrupt enable control 0: Disabled 1: Enabled
5	BAUD	RW	Automatic baud rate detection completion interrupt enable control 0: Disabled 1: Enabled
4	RXBRK	RW	Break Field reception completion interrupt enable control 0: Disabled 1: Enabled
3	RXIDLE	RW	Idle character reception completion interrupt enable control 0: Disabled 1: Enabled
2	RC	RW	Receive completion interrupt enable control 0: Disabled 1: Enabled
1	TC	RW	Transmit completion interrupt enable control 0: Disabled 1: Enabled



Bit field	Name	Permission	Function description
0	TXE	RW	Transmit buffer empty interrupt enable control 0: Disabled 1: Enabled

16.8.11 UARTx_ISR Interrupt flag register

Address offset: 0x1C Reset value: 0x0000 0001

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15	CTSLV	RO	CTS signal level status 0: CTS signal is low level 1: CTS signal is high level
14	TXBUSY	RO	TXD transmit status 0: TDR register and transmit shift register are empty 1: TDR register or transmit shift register is not empty
13	SLVMATCH	RO	Receive slave address match flag 0: No matching address detected 1: Matching address detected
12	RXMATCH	RO	Receive data match flag 0: Received data is different from UARTx_RXMATCH 1: Received data is the same as UARTx_RXMATCH
11	ORE	RO	Overflow flag When UARTx_ISR.RC is 1, hardware sets this bit when the data currently being received in the shift register is ready for transfer to the UARTx_RDR register. 0: No overflow error 1: Overflow error detected <i>Caution: After an overflow error occurs, the newly received data overwrites the data in the UARTx_RDR register.</i>
10	NE	RO	Noise flag 0: No noise detected 1: Noise detected <i>Caution: When OVER is 2 or 3, this bit is invalid and remains 0.</i>
9	PE	RO	Parity error interrupt flag 0: No parity error detected 1: Parity error detected
8	FE	RO	Frame structure error interrupt flag 0: No frame structure errors detected 1: Frame structure error detected



Bit field	Name	Permission	Function description
7	CTS	RO	CTS signal change interrupt flag 0: No CTS level change detected 1: CTS level change detected
6	TIMOV	RO	Timer overflow flag 0: Timer does not overflow ($TIMCNT < TIMARR$) 1: Timer has overflowed ($TIMCNT \geq TIMARR$)
5	BAUD	RO	Auto-baud rate detection complete flag 0: Automatic baud rate detection is not completed 1: Automatic baud rate detection has been completed
4	RXBK	RO	Break Field reception completion flag 0: No Break Field received 1: Break Field has been received
3	RXIDLE	RO	Idle character reception completion flag 0: No idle characters received 1: Idle character received <i>Caution: Idle character detection is initiated each time a character is received.</i>
2	RC	RO	Receive completion interrupt flag 0: One frame of data has not been received 1: One frame of data has been received <i>Caution: STOP is 1bit, RC receives 1bit STOP bit field set to 1; STOP is 1.5bit, RC receives 0.5bit STOP bit field set to 1; STOP is 2bit, RC receives 0.5bit STOP bit field set to 1.</i>
1	TC	RO	Transmit completion interrupt flag 0: Data in transmit buffer or shift register has not been transmitted. 1: The data in the transmit buffer and shift register have been transmitted.
0	TXE	RO	Transmit buffer empty flag 0: There is data in the transmit buffer 1: No data in transmit buffer <i>Caution: Writing to the UARTx_TDR register clears this flag bit.</i>



16.8.12 UARTx_ICR Interrupt flag clear register

Address offset: 0x20 Reset value: 0x0000 0FFF

Bit field	Name	Permission	Function description
31:13	RFU	-	Reserved bits, please keep the default value
12	RXMATCH	R1W0	Receive data match flag clear control W0: Clear received data match flag W1: No function
11	ORE	R1W0	Overflow flag clear control W0: Clear overflow flag W1: No function
10	NE	R1W0	Noise flag clear control W0: Clear noise flag W1: No function
9	PE	R1W0	Parity error interrupt flag clear control W0: Clear parity error interrupt flag W1: No function
8	FE	R1W0	Frame structure error interrupt flag clear control W0: Clear frame structure error interrupt flag W1: No function
7	CTS	R1W0	CTS signal change interrupt flag clear control W0: Clear CTS signal change interrupt flag W1: No function
6	TIMOV	R1W0	Timer overflow flag clear control W0: Clear timer overflow flag W1: No function
5	BAUD	R1W0	Automatic baud rate detection complete flag clear control W0: Clear Auto-Baud rate detection complete flag W1: No function
4	RXBRK	R1W0	Break Field receive completion flag clear control W0: Clear Break Field reception completion flag W1: No function
3	RXIDLE	R1W0	Idle character receive completion flag clear control W0: Clear idle character reception completion flag W1: No function
2	RC	R1W0	Receive completion interrupt flag clear control W0: Clear receive completion interrupt flag W1: No function
1	TC	R1W0	Transmit completion interrupt flag clear control W0: Clear transmit completion interrupt flag W1: No function
0	RFU	-	Reserved bits, please keep the default value



16.8.13 UARTx_ADDR Slave address register

Address offset: 0x30 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:8	RFU	-	Reserved bits, please keep the default value
7:0	ADDR	RW	Slave address register

16.8.14 UARTx_MASK Slave address mask register

Address offset: 0x34 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:8	RFU	-	Reserved bits, please keep the default value
7:0	MASK	RW	Slave address mask register

16.8.15 UARTx_RXMATCH Receive data match register

Address offset: 0x3C Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:9	RFU	-	Reserved bits, please keep the default value
8:0	RXMATCH	RW	Target to be matched receives data



17 Serial peripheral interface (SPI)

17.1 Overview

Serial peripheral interface (SPI) is a synchronous serial data communication interface, which is often used for synchronous serial communication between MCU and external devices. CW32L010 integrates one serial peripheral SPI interfaces, supports bidirectional full-duplex, single-wire half-duplex and simplex communication modes, can configure MCU as master or slave, supports multi-master communication mode.

17.2 Main features

- Supports master mode, slave mode
- Supports full-duplex, single-wire half-duplex, simplex
- Selectable 4 to 16 bit data frame width
- Supports sending and receiving data LSB or MSB first
- Programmable clock polarity and clock phase
- Communication rates up to PCLK/2 in master mode
- Supports multi-computer communication mode
- 8 interrupt sources with flag bits
- Master mode supports break frame adjustment
- Transmit buffer empty/receive data completion trigger to start ADC

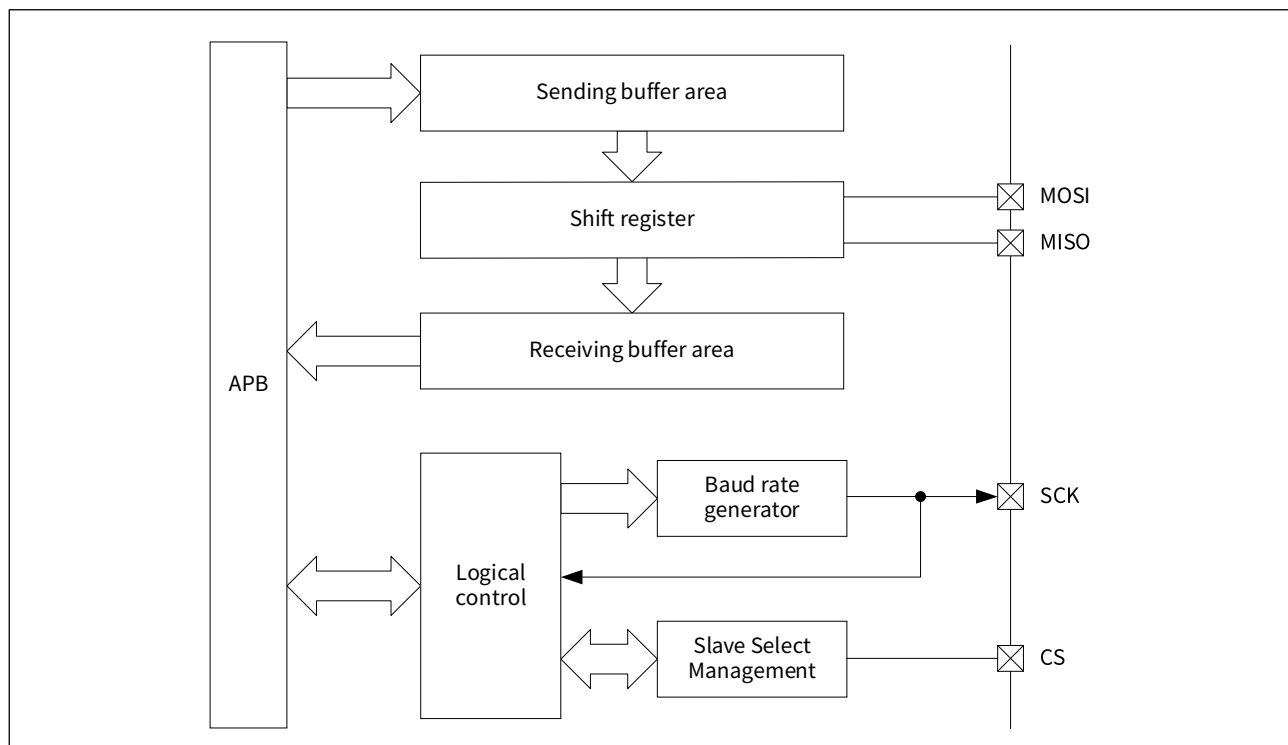


17.3 Functional description

17.3.1 Functional block diagram

The functional block diagram of the SPI controller is shown in the following figure:

Figure 17-1 SPI functional block diagram



SPI is generally connected to external devices through 4 pins:

- MOSI
Master output/slave input, used for data transmission in master mode and data reception in slave mode;
- MISO
Master input/slave output, used for data reception in master mode and data transmission in slave mode;
- SCK
Synchronous serial clock, master clock output and slave clock input, transmit or receive master synchronous clock;
- CS
Slave selection is also used for bus conflict detection in multi-computer communication. Refer to section [17.3.3 Slave select pin CS](#).

CW32L010 supports the user to flexibly select GPIO as the SPI communication pin, which is realized by multiplexing the AFR function, please refer to the pin definitions in the Datasheet.

SPI supports a variety of operating modes. In different operating modes, each pin has different functions. The pin configuration is shown in the following table:

Table 17-1 SPI ports configuration

SPI pins	Mode configuration	Function	GPIO configuration
SPI_SCK	Master	Serial clock output	Digital, push-pull output, multiplexed
	Slave	Serial clock input	Digital, floating input, multiplexed
SPI_MOSI	Full duplex/master	Master data output	Digital, push-pull output, multiplexed
	Full duplex/slave	Slave data input	Digital, floating input, multiplexed
	Single-wire half-duplex/ master	Master data transmitting and receiving	Digital, push-pull output, multiplexed
	Single-wire half-duplex/ slave	Do not use	Can be used as general purpose I/O
	Simplex single transmitting/ master	Master data output	Digital, push-pull output, multiplexed
	Simplex single transmitting/slave	Do not use	Can be used as general purpose I/O
	Simplex single receiving/ master	Do not use	Can be used as general purpose I/O
	Simplex single receiving/ slave	Slave data input	Digital, floating input, multiplexed
SPI_MISO	Full duplex/master	Master data input	Digital, floating input, multiplexed
	Full duplex/slave	Slave data output	Digital, push-pull output, multiplexed
	Single-wire half-duplex/ master	Do not use	Can be used as general purpose I/O
	Single-wire half-duplex/ slave	Slave data transmission and reception	Digital, push-pull output, multiplexed
	Simplex single transmitting/master	Do not use	Can be used as general purpose I/O
	Simplex single transmitting/slave	Slave data output	Digital, push-pull output, multiplexed
	Simplex single receiving/ master	Master data output	Digital, push-pull output, multiplexed
	Simplex single receiving/ slave	Do not use	Can be used as general purpose I/O
SPI_CS	Master (SSM=0)	Slave select input	Digital, floating input, multiplexed
	Master (SSM=1)	Slave select output	Digital, push-pull output, multiplexed
	Slave (SSM=0)	Slave select input	Digital, floating input, multiplexed
	Slave (SSM=1)	Do not use	Can be used as general purpose I/O

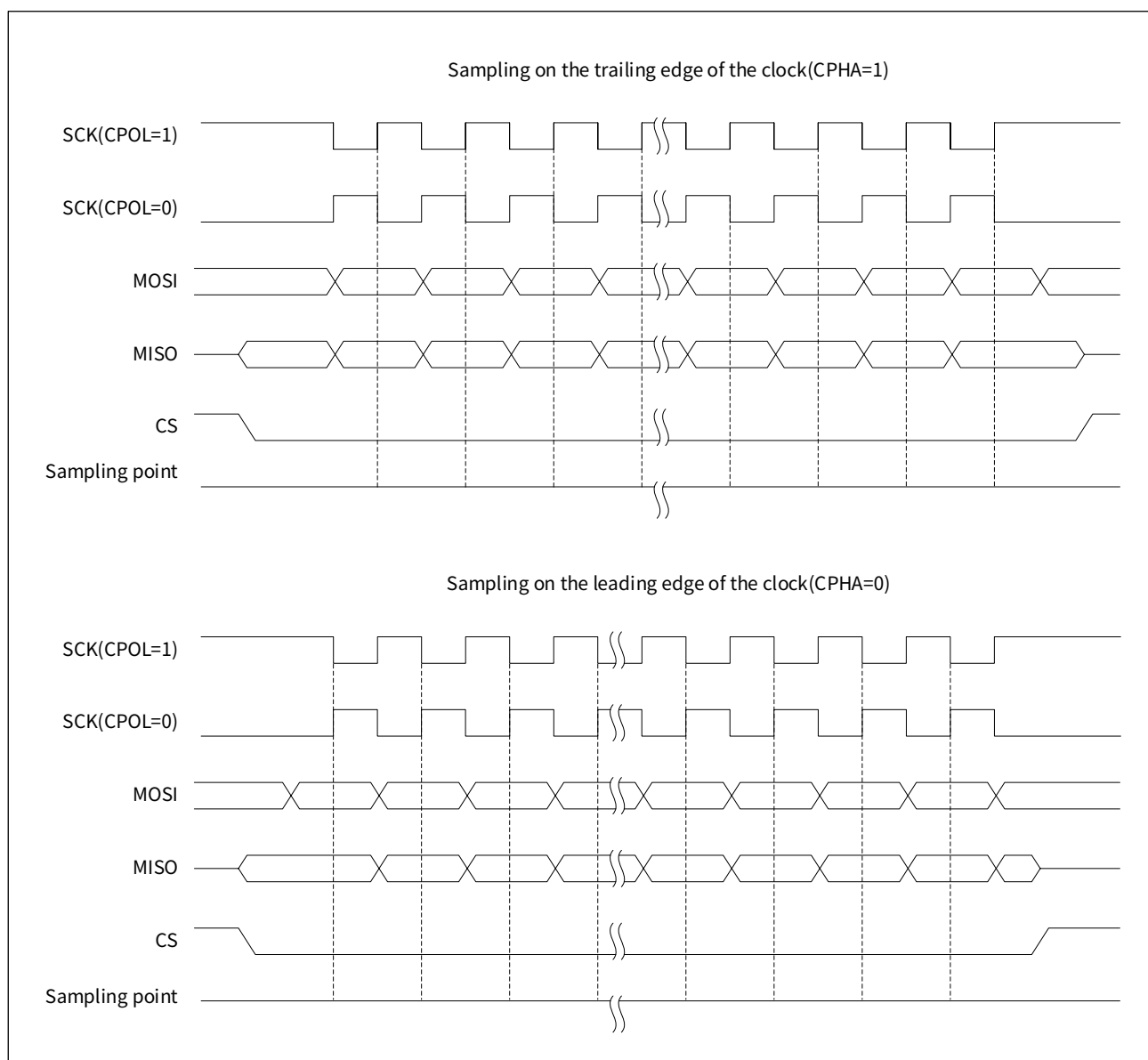


17.3.2 Communication timing and data format

Communication timing

The following figure shows the communication timing of SPI. The CS, SCK, and MOSI signals are all controlled by the master, and the MISO signal is the slave response signal. The slave selects the CS signal from high level to low level, which is the start signal of SPI communication. When the slave detects the start signal, it starts to communicate with the master. In each SCK clock cycle, the MOSI and MISO signal lines transmit one bit of data. CS changes from low level to high, which is the stop signal of communication, indicating the end of this communication.

Figure 17-2 SPI communication timing



Data frame format

The data frame width is configured by the WIDTH bit field of the control register SPI_CR1, which can be set to any data bit width from 4 to 16 bits.

The size of the data is configured by the LSBF bit field of the control register SPI_CR1, and the most significant bit first (MSB) or the least significant bit first (LSB) can be selected.

The master mode frame interval is configured by the GAP bit field of control register SPI_CR1, and can be selected from 0 to 15 SCK clock cycle intervals.

Clock frequency

The synchronous serial clock SCK signal is controlled and generated by the SPI master, and its clock source is PCLK. The frequency division factor is set by configuring the BR bit field of the control register SPI_CR1, and the frequency division can be selected from 2 to 256. For SPI slaves, configuring SPI_CR1.BR has no effect.

When the SPI communication rate is significantly lower than the PCLK clock frequency ($f_{SCK} \leq f_{PCLK}/16$), SPI_CR1.FLTEN can be set to 1 to enable the SPI filter function.

Clock polarity, clock phase

Clock polarity CPOL refers to the level state of the SCK serial clock line when the device is in an idle state without data transmission. Configure through the CPOL bit field of the control register SPI_CR1: set SPI_CR1.CPOL to 0, and the SCK clock line is low when it is idle; set SPI_CR1.CPOL to 1, and the SCK clock line is high when it is idle.

The clock phase CPHA refers to the sampling and shifting moment of the data. Configure through the CPHA of the control register SPI_CR1: set SPI_CR1.CPHA to 0, sample on the front edge of SCK (the clock edge when SCK changes from idle state to non-idle state), and the trailing edge (SCK changes from non-idle state to idle state). Clock edge) shift; set SPI_CR1.CPHA to 1, shift on the leading edge of SCK, and sample on the trailing edge.

According to the different configurations of clock polarity CPOL and clock phase CPHA, SPI can set 4 level modes. The master and slave need to be configured to the same level mode to ensure normal communication.

Table 17-2 SPI level mode

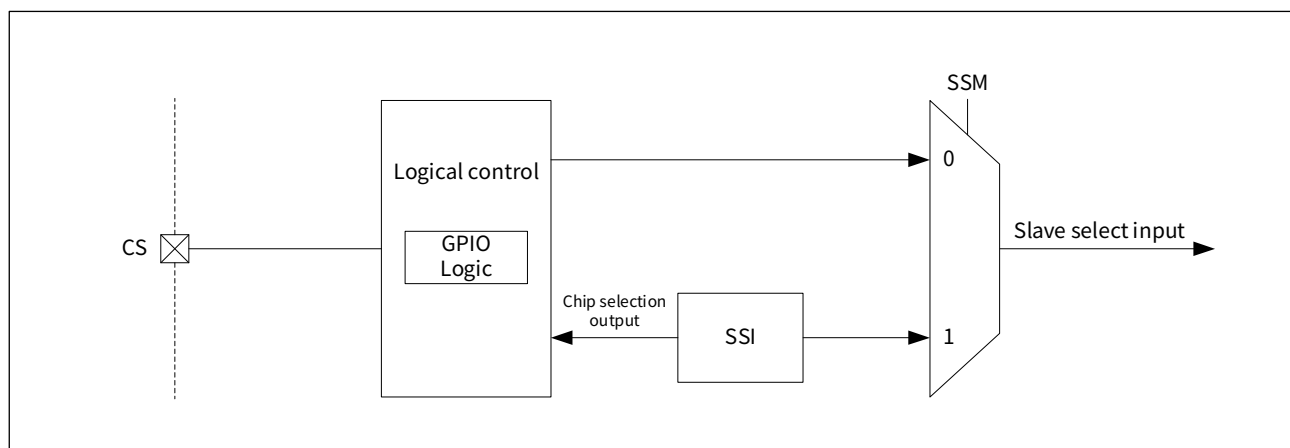
CPOL	CPHA	SCK clock when idle	Sampling time
0	0	Low level	Front edge
0	1	Low level	Back edge
1	0	High level	Front edge
1	1	High level	Back edge



17.3.3 Slave select pin CS

The following figure shows a schematic diagram of the configuration management of slave select pin CS. In master mode, CS can be selected as input or output. When used as input, it is used to detect bus conflicts in multi-master mode, and when used as output, it is used to generate slave chip select signals. In slave mode, CS acts as a device chip select input.

Figure 17-3 Slave select pin management



In master and slave mode, the slave select CS is managed by the SSM bit field of the control register SPI_CR1.

CS pin in master mode

Set SPI_CR1.SSM to 0, and CS is configured as an input to detect whether there is a bus conflict in the multi-computer communication system (**Caution that the master needs to select the slave to communicate through other GPIOs at this time**), please refer to section [17.3.7 Multi-machine communication](#).

Setting SPI_CR1.SSM to 1, the slave select pin CS is configured as an output for generating the slave select signal. The CS pin output level is determined by the SSI bit field of the slave select register SPI_SSI: set SPI_SSI.SSI to 1, CS outputs a high level; set SPI_SSI.SSI to 0, CS outputs a low level.

CS pin in slave mode

Set SPI_CR1.SSM to 0 to use CS signal as device select. The CS level determines whether this unit is selected: when CS is low level, this unit is selected; when CS is high level, this unit is deselected.

Set SPI_CR1.SSM to 1, do not use CS as the device selection, the slave select is determined by the SSI bit field value of the SPI_SSI register. Set SPI_SSI.SSI to 0, this machine is selected; set SPI_SSI.SSI to 1, this machine is unchecked. In this configuration, the user program sets whether the machine is selected by software, and the CS pin can be used as a common IO.

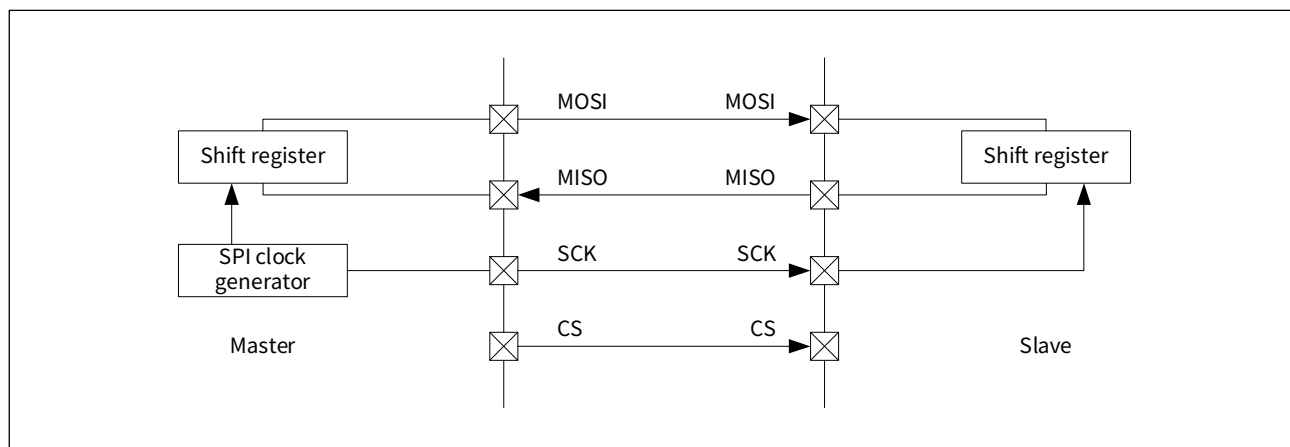
17.3.4 Full-duplex mode

SPI supports full-duplex communication mode. In this mode, the shift registers of the master and slave are connected through two unidirectional data lines, MOSI and MISO, and two lines of data are simultaneously transmitted under the control of the SCK clock signal provided by the master.

Set the MODE bit field of the control register SPI_CR1 to 0x0 to make the SPI work in full-duplex communication mode.

A typical application block diagram of full-duplex mode is shown in the following figure:

Figure 17-4 Full-duplex communication mode



Master transmits and receives

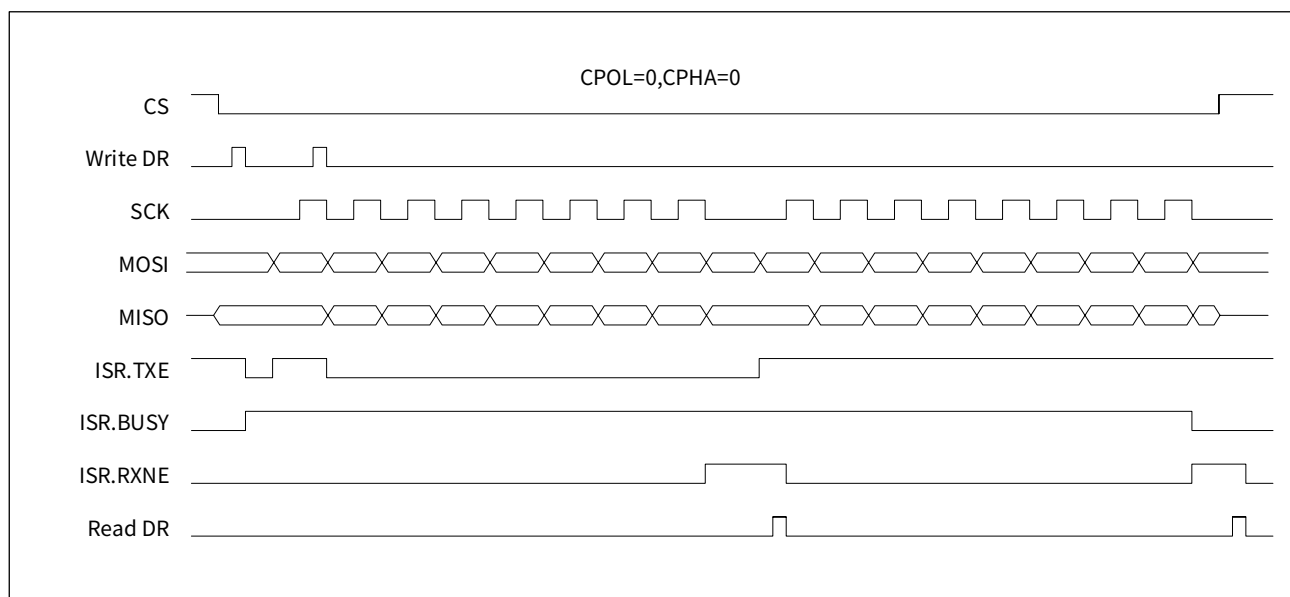
Set SPI_CR1.MSTR to 1, SPI works in master mode.

The master sets SPI_SSI.SSI to 0, and the slave selects the CS pin to output a low level as the communication start signal.

When the transmit buffer is empty, that is, the SPI_ISR.TXE flag is 1, write a frame of data to be sent into the SPI_DR register, the data is output from the MOSI pin under the control of the synchronous shift clock signal, and the MISO pin is simultaneously pin data is received into the shift register. At the end of transmitting a data frame, the non-empty flag bit SPI_ISR.RXNE of the receive buffer is set to 1 by hardware, indicating that a frame of data has been received, and the SPI_DR register can be read at this time.

Set SPI_SSI.SSI to 1, and the slave select pin CS outputs a high level to end this communication.

Figure 17-5 Master transmits and receives timing in full-duplex mode (two bytes)



Slave transmits and receives

Set SPI_CR1.MSTR to 0, SPI works in slave mode.

Before the slave select CS signal is pulled low, the slave needs to set SPI_ICR.FLUSH to 0 to clear the transmit buffer and shift register, and write the first frame data to be transmitted into the SPI_DR register.

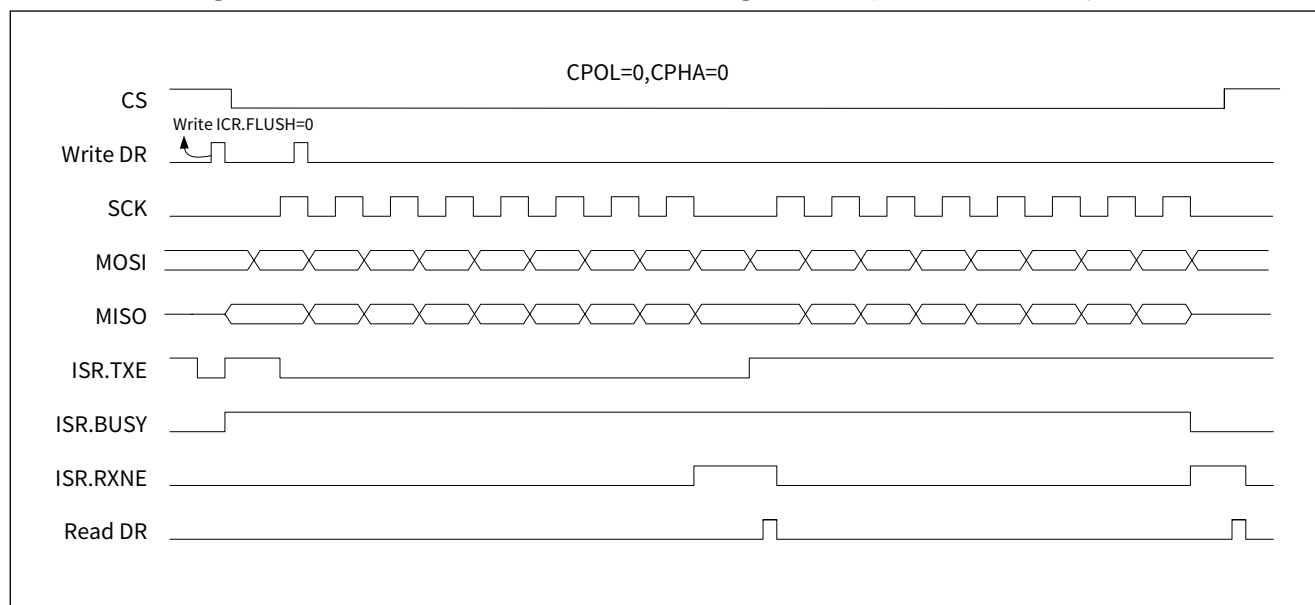
When the CS signal is pulled low, the written data will be output from the MISO pin under the control of the master's synchronous shift clock signal, and at the same time receive the data from the MOSI pin to the shift register.

If it is continuous communication of multiple data frames, the user should continuously query the SPI_ISR.TXE flag bit. Once the flag is 1, immediately write the data to be transmitted into the SPI_DR register to avoid data leakage.

When the receive buffer is not empty, that is, the SPI_ISR.RXNE flag is 1, indicating that a frame of data has been received, and the SPI_DR register can be read at this time.

This communication ends when it is detected that the CS pin changes to a high level.

Figure 17-6 Slave transmit and receive timing in full-duplex mode (two bytes)



17.3.5 Single-wire half-duplex mode

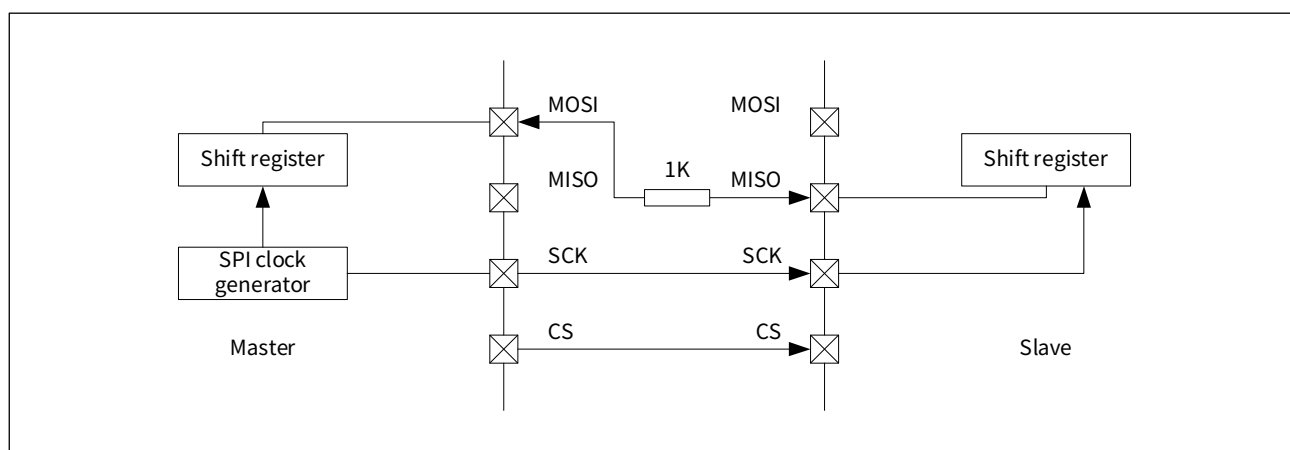
SPI supports single-wire half-duplex communication mode. In this mode, the master and slave communicate data through a bidirectional data line. The master uses MOSI, the slave uses MISO, and other unused SPI signal lines can be used by other functions.

Set the MODE bit field of the control register SPI_CR1 to 0x3 to make the SPI work in single-wire half-duplex mode.

The HDOE bit field of the control register SPI_CR3 selects the transmitting and receiving direction. When HDOE is 0, it is only received, and when HDOE is 1, it is only transmitted.

A typical application block diagram of single-wire half-duplex mode is shown in the following figure:

Figure 17-7 Single-wire half-duplex mode



During single-wire half-duplex communication, since the communication directions of the master and the slave cannot be guaranteed to switch synchronously, there may be a driving conflict and cause device damage. It is recommended to connect a resistor in series on the data line between the master and the slave to limit the current. .

When the SPI is used as a slave, the single-wire half-duplex mode can be used to communicate with a standard UART transceiver in synchronous mode. At this time, the SPI must be configured as a fixed level mode: the clock polarity CPOL is 1, and the clock phase is CPHA. is 1.

Master transmits

Set SPI_CR1.MSTR to 1, SPI works in master mode; set SPI_CR3.HDOE to 1, the master only transmits data.

The master sets SPI_SSI.SSI to 0, and outputs a low level on the slave selection pin CS as a communication start signal.

When the transmit buffer is empty, that is, the SPI_ISR.TXE flag bit is 1, a frame of data to be transmitted is written into the SPI_DR register, and the data is output from the MOSI pin under the control of the synchronous shift clock signal.

After writing the last frame of data, you must wait for the transmit buffer to be empty, that is, the SPI_ISR.TXE flag becomes 1, and the SPI_ISR.BUSY flag becomes 0 to ensure that the data is transmitted. Then set SPI_SSI.SSI to 1, select the CS pin to output a high level in the slave, and end this communication.

Master receives

Set SPI_CR1.MSTR to 1, SPI works in master mode; set SPI_CR3.HDOE to 0, the master only receives data.

The master sets SPI_SSI.SSI to 0, and the slave selects the CS pin to output a low level as the communication start signal.

When the transmit buffer is empty, that is, the SPI_ISR.TXE flag is 1, write a frame of dummy data to the SPI_DR register to start the transmission.

When the receive buffer is not empty, that is, the SPI_ISR.RXNE flag is 1, indicating that a frame of data has been received, and the SPI_DR register can be read at this time.

If you want to receive multiple frames of data, repeat the above steps to write to SPI_DR and read the received data from SPI_DR.

When all data frames are received, set SPI_SSI.SSI to 1, select the CS pin to output a high level in the slave, and end this communication.

Slave transmits

Set SPI_CR1.MSTR to 0, SPI works in slave mode; set SPI_CR3.HDOE to 1, the slave only transmits data.

Before the slave select CS signal is pulled low, the slave needs to set SPI_ICR.FLUSH to 0 to clear the transmit buffer and shift register, and write the first frame data to be transmitted into the SPI_DR register.

When the CS signal is pulled low, the written data will be output from the MISO pin under the control of the master's synchronous shift clock signal.

If it is continuous communication of multiple data frames, the user should continuously query the SPI_ISR.TXE flag bit. Once the flag is 1, immediately write the data to be transmitted into the SPI_DR register to avoid data leakage.

This communication ends when it is detected that the CS pin changes to a high level.

Slave receives

Set SPI_CR1.MSTR to 0, SPI works in slave mode; set SPI_CR3.HDOE to 0, the slave only receives data.

When the CS pin is detected to go low, the slave starts to communicate with the master.

When the receive buffer is not empty, that is, the SPI_ISR.RXNE flag becomes 1, indicating that a frame of data has been received, and the SPI_DR register can be read at this time.

This communication ends when it is detected that the CS pin changes to a high level.



17.3.6 Simplex mode

SPI supports simplex communication mode, and the master and slave communicate with single transmitting or single receiving through a unidirectional data line.

The master uses the MOSI signal line for single transmitting communication, and the MISO signal line for single receiving communication; the slave uses the MOSI signal line for single receiving communication, and the MISO signal line for single transmitting communication, and the unused signal line can be used for other functions.

Set the MODE bit field of the control register SPI_CR1 to 0x1, and the SPI works in the simplex single transmitting communication mode; set the MODE bit field to 0x2, and the SPI works in the simplex single receiving communication mode.

Caution:

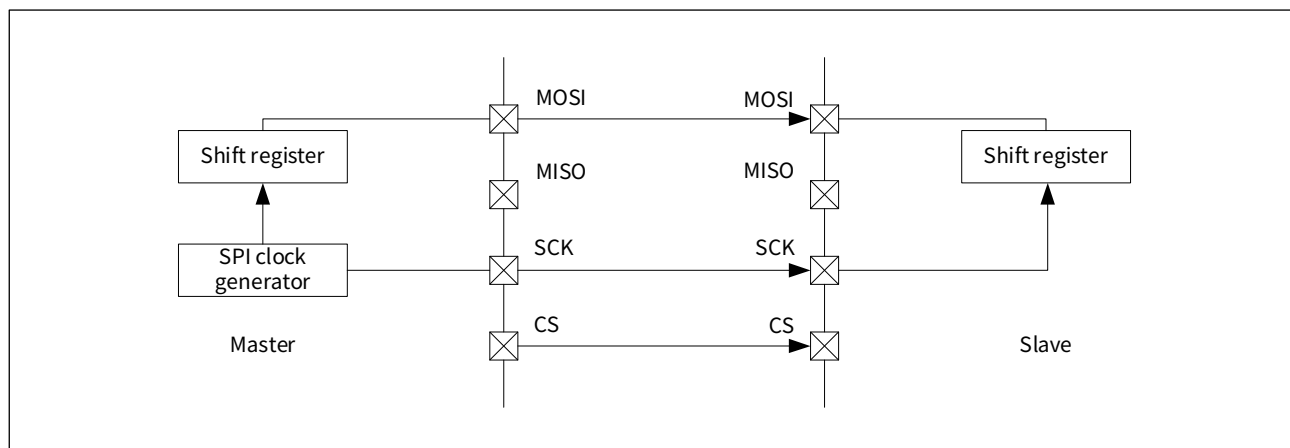
In simplex single receiving mode, all events related to the receive stream on the transmitter side, such as the receive buffer overflow error flag bit SPI_ISR.OV, must be ignored.



17.3.6.1 Master single transmitting/slave single receiving

In the application scenario of single transmitting by the master and single receiving of the slave, the master and the slave use a MOSI data line to communicate. The application block diagram is shown in the following figure:

Figure 17-8 Master single transmitting, slave single receiving mode



Master single transmitting

Set SPI_CR1.MODE to 0x1, SPI works in simplex single transmitting communication mode; set SPI_CR1.MSTR to 1, SPI works in master mode.

Set SPI_SSI.SSI to 0, and select the CS pin to output a low level in the slave as the communication start signal.

When the transmit buffer is empty, that is, the SPI_ISR.TXE flag bit is 1, a frame of data to be sent is written into the SPI_DR register, and the data is output from the MOSI pin under the control of the synchronous shift clock signal.

After writing the last frame of data, you must wait for transmit buffer empty flag SPI_ISR.TXE to become 1, and the SPI bus busy flag SPI_ISR.BUSY to become 0 to ensure that the data is transmitted. Then set SPI_SSI.SSI to 1, so that the slave selects the CS pin to output a high level, and ends this communication.

Slave single receiving

Set SPI_CR1.MODE to 0x2, SPI works in simplex single receiving communication mode; set SPI_CR1.MSTR to 0, SPI works in slave mode.

When the CS pin is detected to go low, the slave starts to communicate with the master.

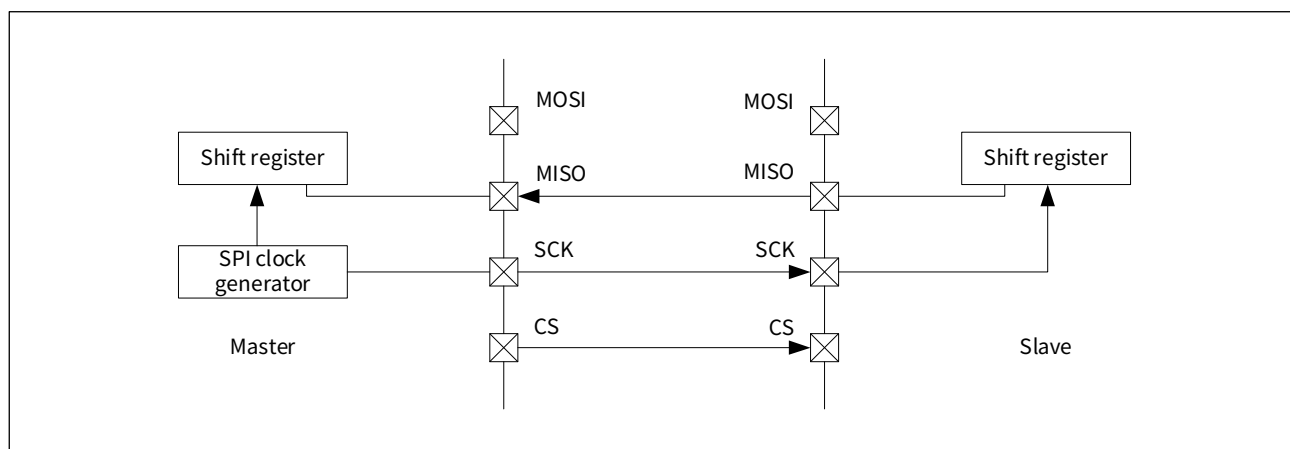
When the receive buffer is not empty, that is, the SPI_ISR.RXNE flag is 1, indicating that a frame of data has been received, and the SPI_DR register can be read at this time.

This communication ends when it is detected that the CS pin changes to a high level.

17.3.6.2 Master single receiving/slave single transmitting

In the application scenario of the master single receiving and the slave single transmitting, the master and the slave use a MISO data line to communicate. The application block diagram is shown in the following figure:

Figure 17-9 Master single receiving, slave single transmitting mode



Master single receiving

Set SPI_CR1.MODE to 0x2, SPI works in simplex single receiving communication mode; set SPI_CR1.MSTR to 1, SPI works in master mode.

Set SPI_SSI.SSI to 0, and select the CS pin to output a low level in the slave as the communication start signal.

When the transmit buffer is empty, that is, the SPI_ISR.TXE flag is 1, write a frame of dummy data to the SPI_DR register to start the transmission.

When the receive buffer is not empty, that is, the SPI_ISR.RXNE flag is 1, indicating that a frame of data has been received, and the SPI_DR register can be read at this time.

If you want to receive multiple frames of data, repeat the above steps to write to SPI_DR and read the received data from SPI_DR.

When all data frames are received, set SPI_SSI.SSI to 1, and the slave selects the CS pin to output a high level to end this communication.

Slave single transmitting

Set SPI_CR1.MODE to 0x1, SPI works in simplex single transmitting communication mode; set SPI_CR1.MSTR to 0, SPI works in slave mode.

Before the slave selection signal CS is pulled low, the slave needs to set SPI_ICR.FLUSH to 0 to clear the transmit buffer and shift register, and write the first frame data to be transmitted into the SPI_DR register.

When the CS signal is pulled low, the written data is output from the MISO pin under the control of the master's synchronous shift clock signal.

If it is continuous communication of multiple data frames, the user should continuously query the SPI_ISR.TXE flag bit. Once the flag is 1, immediately write the data to be transmitted into the SPI_DR register to avoid data leakage.

This communication ends when it is detected that the CS pin changes to a high level.

17.3.7 Multi-machine communication

SPI supports multi-machine communication mode. In this mode, the slave select CS pin of the master should be configured as an input and connected to the bus application signal of other masters to detect whether there is a conflict on the SPI bus.

If the slave select CS pin of a master is pulled to a low level, it means that another master is occupying the bus, the master will generate a mode error (please refer to section [17.3.9 Error flags](#)), the master SPI will be disabled, and the SPI will be disabled. Avoid SPI bus conflicts.

If the slave of a master selects the CS pin to detect a high level, it means that the SPI bus is idle. At this time, if the master wants to communicate with the slave, it must first apply for a signal on the bus to output a low level to other masters. CS signal, get SPI bus control, and then select a slave to start communication.

[Figure 17-10 Multi-machine communication system](#) shows a typical multi-machine communication system. Taking the communication between master A and slave 1 as an example, the operation process is as follows:

Step 1: Master A detects the CS pin until a high level occurs;

Step 2: GPIO0 of master A outputs a low level to obtain bus control, and master B is disabled;

Step 3: GPIO1 of master A outputs low level, and slave 1 is selected;

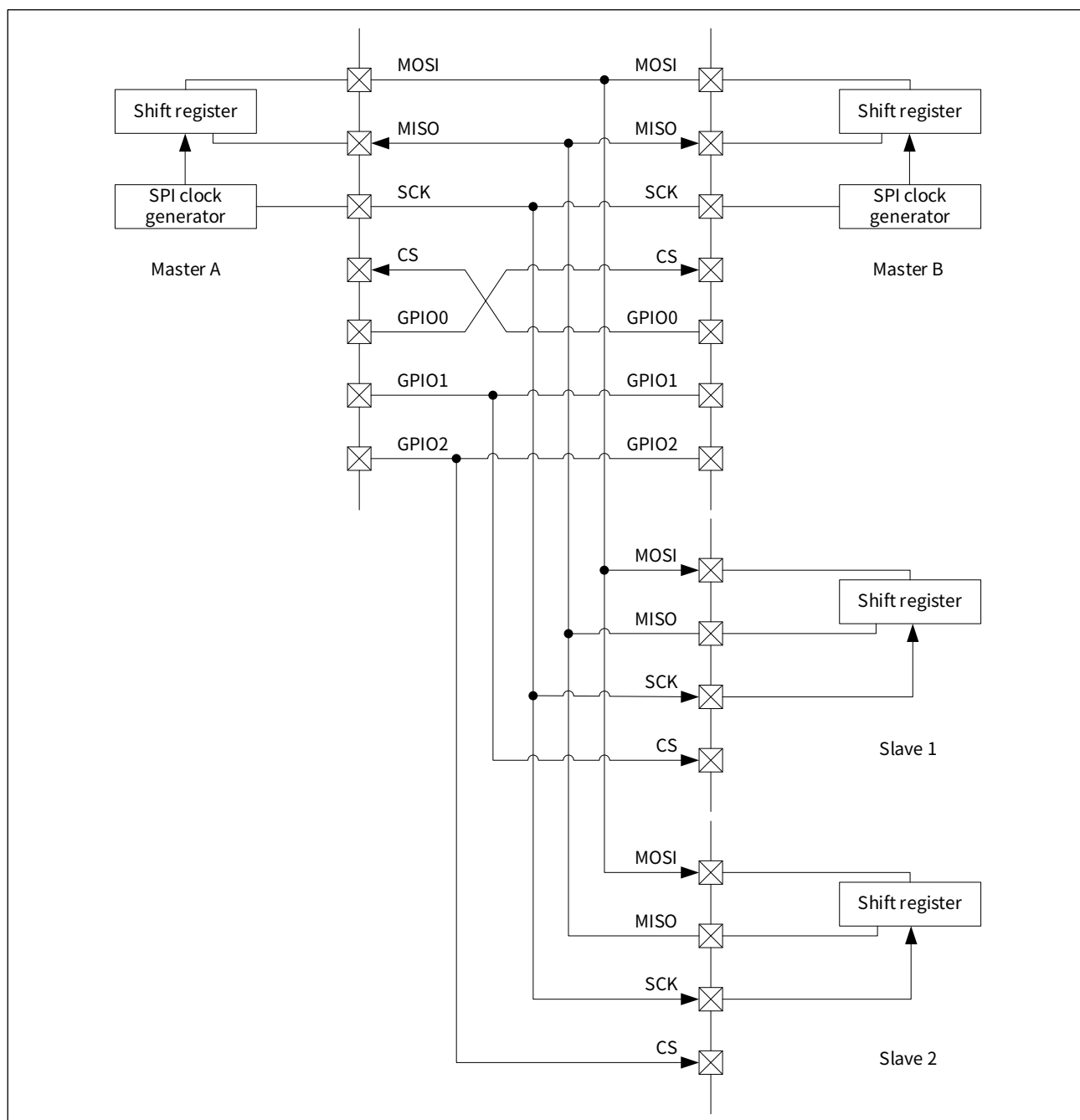
Step 4: The master A communicates with the slave 1;

Step 5: GPIO1 of master A outputs a high level, releasing slave 1;

Step 6: GPIO0 of master A outputs a high level, releasing the SPI bus.



Figure 17-10 Multi-machine communication system



In order to ensure the communication reliability in the multi-machine communication mode, the MISOHD bit field of the slave control register SPI_CR1 should be set to 1. When the slave is selected, the MISO pin is CMOS output; when it is not selected, the MISO pin is high resistance output.

17.3.8 Status flags

There are 6 status flags in the SPI controller to indicate the general working status of the SPI.

Transmit buffer empty flag (SPI_ISR.TXE)

When the transmit data is transferred from the SPI_DR register to the shift register, the SPI_ISR.TXE flag will be set by hardware, indicating that the transmit buffer is empty, and new data to be transmitted is allowed to be written to the SPI_DR register. While writing data to the SPI_DR register, the SPI_ISR.TXE flag will be automatically cleared.

Receive buffer not empty flag (SPI_ISR.RXNE)

When the received data is transferred from the shift register to the SPI_DR register, the SPI_ISR.RXNE flag will be set by hardware, indicating that the reception of a frame of data has been completed. At this time, the SPI_DR register can be read, and the SPI_ISR will be cleared while reading the SPI_DR register. RXNE flag bit.

Users can also manually clear the SPI_ISR.RXNE flag by software.

Bus busy flag bit (SPI_ISR.BUSY)

The SPI_ISR.BUSY flag bit, set and cleared by hardware, is used to indicate the current SPI communication status.

In master mode, when preparing or performing data transmission (there is data in the transmit buffer or data in the shift register), the BUSY flag will be set by hardware, indicating that the SPI interface is in a busy state; When there is no data transmission (no data in the send buffer and no data in the shift register), the BUSY flag is automatically cleared by hardware, indicating that the SPI interface is in an idle state. When the user code inquires that the flag is 0, the CS signal can be pulled high to end the current round of communication.

This flag is not recommended in slave mode.

Slave select signal status (SPI_ISR.SSLVL)

The SPI_ISR.SSLVL status bit, set and cleared by hardware, indicates the level status of the slave select input signal.

SPI_ISR.SSLVL is 0, the slave select CS input is low, and the slave is selected; SPI_ISR.SSLVL is 1, the slave select CS input is high, and the slave is not selected.

Slave select input rising edge flag (SPI_ISR.SSR)

The SPI_ISR.SSR flag bit is used to indicate whether the slave select CS input signal has a rising edge. It only makes sense when CS uses hardware pins in slave mode (MSTR=0 and SSM=0).

SPI_ISR.SSR is 0, indicating that the slave select CS input has no rising edge; SPI_ISR.SSR is 1, indicating that the slave select input has a rising edge.

Slave select input falling edge flag (SPI_ISR.SSF)

The SPI_ISR.SSF flag bit is used to indicate whether the slave select CS input signal has a falling edge. It only makes sense when CS uses hardware pins in slave mode (MSTR=0 and SSM=0).

SPI_ISR.SSF is 0, indicating that the slave select CS input has no falling edge; SPI_ISR.SSF is 1, indicating that the slave select CS input has a falling edge.



17.3.9 Error flags

There are 4 error flags in the SPI controller to indicate whether an error has occurred in the SPI.

Mode error flag (SPI_ISR.MODF)

SPI_ISR.MODF flag bit, used to detect bus conflict in SPI multi-machine communication mode.

When the SPI works in master mode and the slave select CS pin is configured as an input, if the CS pin input is low, a mode error will occur, and the SPI_ISR.MODF flag is set by hardware, indicating that other SPI masters are occupying the bus at this time.

After a mode error occurs, SPI_CR2.EN is cleared and the SPI is forced to shut down.

Setting SPI_ICR.MODF to 0 can clear the SPI_ISR.MODF flag. After clearing, the user needs to reconfigure the SPI controller.

After a mode error occurs, if the SPI controller is re-enabled, the SPI can still perform normal data transmission, but the SPI_ISR.MODF flag will remain set until cleared by user software.

Slave select error flag in slave mode (SPI_ISR.SSERR)

When the SPI is working in slave mode and a valid data transfer is in progress, if the slave select CS input is pulled high, a slave select error will occur, and the SPI_ISR.SSERR flag will be set by hardware.

The occurrence of a slave select error will cause an error in the current data transmission, and the slave will enter the unselected state.

After a slave select error occurs, if the slave select CS input is pulled low again, the SPI can still perform normal data transmission, but the SPI_ISR.SSERR flag will remain set until cleared by user software.

Receive buffer overflow error flag (SPI_ISR.OV)

When the master or slave does not clear the SPI_ISR.RXNE flag after receiving a frame of data, that is, neither read the SPI_DR register nor clear the SPI_ICR.RXNE to 0 by software, and then when a new frame of data is received, a receive buffer overflow error will occur, and the SPI_ISR.OV flag is set by hardware.

After a receive buffer overflow error occurs, the newly received data will overwrite the original data in the buffer.

After a receive buffer overflow error occurs, the SPI can still perform normal data transmission, but the SPI_ISR.OV flag will remain set until cleared by user software.

Transmit buffer underflow error flag in slave mode (SPI_ISR.UD)

When the SPI works in slave mode, if the transmit buffer is empty and the transmission of a new frame has already started, a transmit buffer underflow error will occur, and the SPI_ISR.UD flag will be set by hardware, indicating that the slave did not send the data in time. The SPI_DR register writes data and misses transmitting data to the master.

After the transmit buffer underflow error occurs, the SPI can still perform normal data transmission, but the SPI_ISR.UD flag will remain set until cleared by user software.

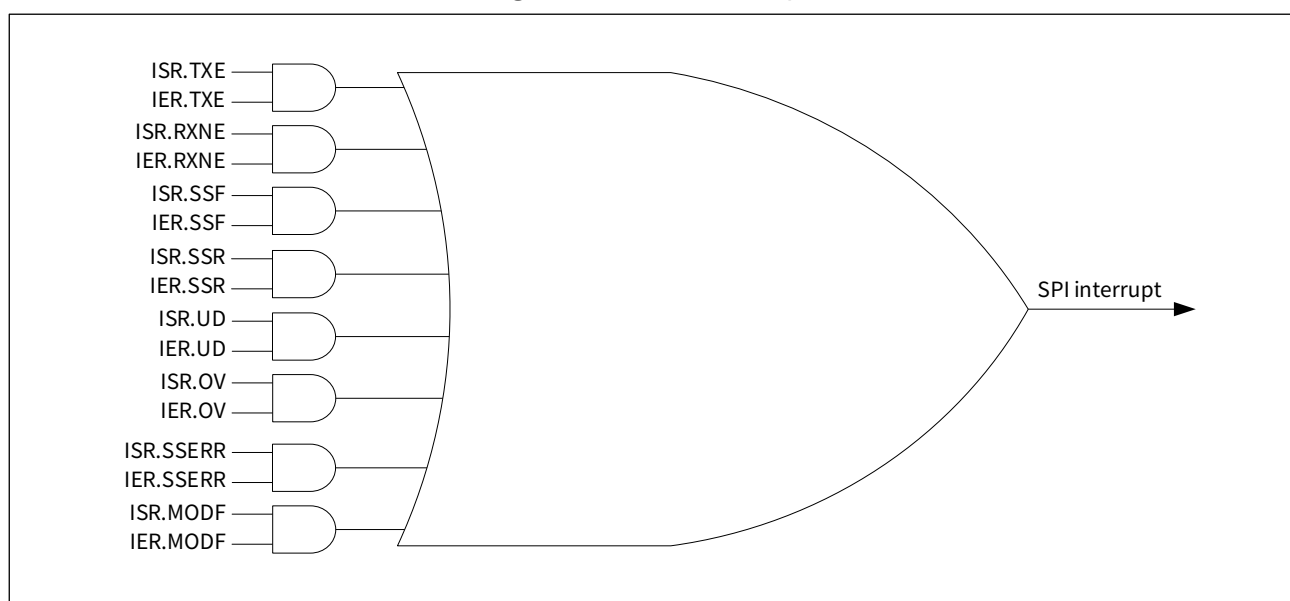


17.4 SPI interrupts

The SPI controller supports 8 interrupt sources. When the SPI interrupt trigger event occurs, the interrupt flag bit will be set by hardware. If the corresponding interrupt enable control bit is set, an interrupt request will be generated.

An SPI module of CW32L010 uses a same system SPI interrupt, whether the SPI interrupt generates an interrupt jump is controlled by the corresponding bit of the interrupt enable setting register NVIC_ISER. The schematic diagram of the system SPI interrupt is shown in the following figure:

Figure 17-11 SPI interrupts



In the user's SPI interrupt service routine, the relevant SPI interrupt flag bit should be queried for corresponding processing. Before exiting the interrupt service routine, the interrupt flag bit should be cleared to avoid repeatedly entering the interrupt routine.

The flag bit, interrupt enable bit, interrupt flag clear bit and clear method of each interrupt source of SPI are shown in the following table:

Table 17-3 SPI interrupt control

Interrupt event	Interrupt flag bit	Interrupt enable bit	Flag clear method
Transmit buffer empty	ISR.TXE	IER.TXE	Write SPI_DR register
Receive buffer non-empty	ISR.RXNE	IER.RXNE	Read SPI_DR register, or write 0 to ICR.RXNE
Falling edge appears on the slave select input	ISR.SSF	IER.SSF	Write 0 to ICR.SSF
Rising edge appears on the slave select input	ISR.SSR	IER.SSR	Write 0 to ICR.SSR
Transmit buffer underflow in slave mode	ISR.UD	IER.UD	Write 0 to ICR.UD
Receive buffer overflow	ISR.OV	IER.OV	Write 0 to ICR.OV
Slave select error in slave mode	ISR.SSERR	IER.SSERR	Write 0 to ICR.SSERR
Mode error	ISR.MODF	IER.MODF	Write 0 to ICR.MODF

17.5 Programming examples

17.5.1 Full duplex/master mode

17.5.1.1 Transmit and receive in query mode

- Step 1: Set `SYSCTRL_AHBEN.GPIOx` to 1, `SYSCTRL_APBEN1.SPI` to 1, and enable the GPIO clock and SPI working clock corresponding to the SPI pin;
- Step 2: Configure the `SPI_SCK`, `SPI_MOSI` and `SPI_CS` pins as push-pull multiplexed output mode, and configure the `SPI_MISO` pin as floating input and multiplexed mode. For specific register configuration steps, please refer to chapter [8 General-purpose input/output \(GPIO\)](#);
- Step 3: Set `SPI_CR1.MODE` to 0x0, and configure SPI to be bidirectional full-duplex communication mode;
- Step 4: Set `SPI_CR1.MSTR` to 1, and configure SPI to master mode;
- Step 5: Configure `SPI_CR1.WIDTH` and set the width of each frame of data;
- Step 6: Configure `SPI_CR1.LSBF` and set the data bit sending and receiving sequence;
- Step 7: Configure `SPI_CR1.CPOL` and set the clock polarity;
- Step 8: Configure `SPI_CR1.CPHA` and set the clock phase;
- Step 9: Set `SPI_CR1.SSM` to 1, and determine the slave select output value through the SSI register;
- Step 10: Configure `SPI_CR1.BR` and set the SCK baud rate;
- Step 11: Set `SPI_CR2.EN` to 1 to enable the SPI controller;
- Step 12: Set `SPI_SSI.SSI` to 0, communication starts;
- Step 13: Query and wait for the `SPI_ISR.TXE` flag to be set to 1, and confirm that the transmit buffer is empty;
- Step 14: Write a frame of data to be transmitted into the `SPI_DR` register;
- Step 15: Query and wait for the `SPI_ISR.RXNE` flag to be set to 1 to confirm that one frame of data has been received;
- Step 16: Read the `SPI_DR` register and save the data;
- Step 17: Repeat steps 13~17 to send and receive the next frame of data until all data is transmitted and received;
- Step 18: Set `SPI_SSI.SSI` to 1, communication ends.



17.5.1.2 Transmit and receive in interrupt mode

- Step 1: Set SYSCTRL_AHBEN.GPIOx to 1, SYSCTRL_APBEN1.SPI to 1, and enable the GPIO clock and SPI working clock corresponding to the SPI pin;
- Step 2: Configure the SPI_SCK, SPI_MOSI and SPI_CS pins as push-pull multiplexed output mode, and configure the SPI_MISO pin as floating input and multiplexed mode. For specific register configuration steps, please refer to chapter [8 General-purpose input/output \(GPIO\)](#);
- Step 3: Set SPI_CR1.MODE to 0x0, and configure SPI to be bidirectional full-duplex communication mode;
- Step 4: Set SPI_CR1.MSTR to 1, and configure SPI to master mode;
- Step 5: Configure SPI_CR1.WIDTH and set the width of each frame of data;
- Step 6: Configure SPI_CR1.LSBF and set the data bit sending and receiving sequence;
- Step 7: Configure SPI_CR1.CPOL and set the clock polarity;
- Step 8: Configure SPI_CR1.CPHA and set the clock phase;
- Step 9: Set SPI_CR1.SSM to 1, and determine the slave selection output value through the SSI register;
- Step 10: Configure SPI_CR1.BR and set the SCK baud rate;
- Step 11: Set SPI_CR2.EN to 1 to enable the SPI controller;
- Step 12: Write 0x00 to SPI_ICR to clear all flags;
- Step 13: Configure the NVIC controller, please refer to chapter [5 Interrupt](#);
- Step 14: Set SPI_SSI.SSI to 0, communication starts;
- Step 15: Set SPI_IER.RXNE to 1 to enable receive buffer non-empty interrupt;
- Step 16: Set SPI_IER.TXE to 1 to enable transmit buffer empty interrupt;
- Step 17: Enter the interrupt service function: query and judge the SPI_ISR.TXE flag bit, if the flag bit is 1, write a frame of data to be transmitted into the SPI_DR register; query and judge the SPI_ISR.RXNE flag bit, if the flag bit is 1, then read the SPI_DR register and save the data;
- Step 18: Set SPI_SSI.SSI to 1, communication ends.



17.5.2 Single-wire half-duplex/master mode

17.5.2.1 Transmit in query mode

- Step 1: Set `SYSCTRL_AHBEN.GPIOx` to 1, `SYSCTRL_APBEN1.SPI` to 1, and enable the GPIO clock and SPI working clock corresponding to the SPI pin;
- Step 2: Configure the `SPI_SCK`, `SPI_MOSI` and `SPI_CS` pins as push-pull multiplexed output mode. For specific register configuration steps, please refer to chapter [8 General-purpose input/output \(GPIO\)](#);
- Step 3: Set `SPI_CR1.MODE` to 0x3, and configure SPI to be single-wire half-duplex communication mode;
- Step 4: Set `SPI_CR1.MSTR` to 1, and configure SPI to master mode;
- Step 5: Configure `SPI_CR1.WIDTH` and set the width of each frame of data;
- Step 6: Configure `SPI_CR1.LSBF` and set the data bit transmitting and receiving sequence;
- Step 7: Configure `SPI_CR1.CPOL` and set the clock polarity;
- Step 8: Configure `SPI_CR1.CPHA` and set the clock phase;
- Step 9: Set `SPI_CR1.SSM` to 1, and determine the slave select output value through the SSI register;
- Step 10: Configure `SPI_CR1.BR` and set the SCK baud rate;
- Step 11: Set `SPI_CR2.EN` to 1 to enable the SPI controller;
- Step 12: Set `SPI_SSI.SSI` to 0, communication starts;
- Step 13: Set `SPI_CR3.HDOE` to 1, only transmit;
- Step 14: Query and wait for the `SPI_ISR.TXE` flag to be set to 1, and confirm that the transmit buffer is empty;
- Step 15: Write a frame of data to be transmitted into the `SPI_DR` register;
- Step 16: If there is still data to be transmitted, repeat steps 14-16 to transmit the next frame of data;
- Step 17: Query and wait for the `SPI_ISR.TXE` flag bit to be 1, and the last frame of data starts to be transmitted;
- Step 18: Query and wait for the `SPI_ISR.BUSY` flag to become 0, and the SPI bus is idle;
- Step 19: Set `SPI_SSI.SSI` to 1, communication ends.



17.5.2.2 Receive in query mode

- Step 1: Set SYSCTRL_AHBEN.GPIOx to 1, SYSCTRL_APBEN1.SPI to 1, and enable the GPIO clock and SPI working clock corresponding to the SPI pin;
- Step 2: Configure the SPI_SCK, SPI_MOSI and SPI_CS pins as push-pull multiplexed output mode. For specific register configuration steps, please refer to chapter [8 General-purpose input/output \(GPIO\)](#);
- Step 3: Set SPI_CR1.MODE to 11, and configure SPI to be single-wire half-duplex communication mode;
- Step 4: Set SPI_CR1.MSTR to 1, and configure SPI to master mode;
- Step 5: Configure SPI_CR1.WIDTH and set the width of each frame of data;
- Step 6: Configure SPI_CR1.LSBF and set the data bit transmitting and receiving sequence;
- Step 7: Configure SPI_CR1.CPOL and set the clock polarity;
- Step 8: Configure SPI_CR1.CPHA and set the clock phase;
- Step 9: Set SPI_CR1.SSM to 1, and determine the slave select output value through the SSI register;
- Step 10: Configure SPI_CR1.BR and set the SCK baud rate;
- Step 11: Set SPI_CR2.EN to 1 to enable the SPI controller;
- Step 12: Set SPI_SSI.SSI to 0, communication starts;
- Step 13: Set SPI_CR3.HDOE to 0, only receive;
- Step 14: Query and wait for the SPI_ISR.TXE flag to be set to 1, and confirm that the transmit buffer is empty;
- Step 15: Write a frame of dummy data into the SPI_DR register;
- Step 16: Query and wait for the SPI_ISR.RXNE flag to be set to 1 to confirm that one frame of data has been received;
- Step 17: Read the SPI_DR register and save the data;
- Step 18: Repeat steps 14~18 to receive the next frame of data until all data is received;
- Step 19: Set SPI_SSI.SSI to 1, communication ends.



17.5.3 Simplex mode/master mode

17.5.3.1 Transmit in query mode

- Step 1: Set SYSCTRL_AHBEN.GPIOx to 1, SYSCTRL_APBEN1.SPI to 1, and enable the GPIO clock and SPI working clock corresponding to the SPI pin;
- Step 2: Configure the SPI_SCK, SPI_MOSI and SPI_CS pins as push-pull multiplexed output mode. For specific register configuration steps, please refer to chapter [8 General-purpose input/output \(GPIO\)](#);
- Step 3: Set SPI_CR1.MODE to 01, and configure SPI as simplex single-transmit communication mode;
- Step 4: Set SPI_CR1.MSTR to 1, and configure SPI to master mode;
- Step 5: Configure SPI_CR1.WIDTH and set the width of each frame of data;
- Step 6: Configure SPI_CR1.LSBF and set the data bit transmitting and receiving sequence;
- Step 7: Configure SPI_CR1.CPOL and set the clock polarity;
- Step 8: Configure SPI_CR1.CPHA and set the clock phase;
- Step 9: Set SPI_CR1.SSM to 1, and determine the slave select output value through the SSI register;
- Step 10: Configure SPI_CR1.BR and set the SCK baud rate;
- Step 11: Set SPI_CR2.EN to 1 to enable the SPI controller;
- Step 12: Set SPI_SSI.SSI to 0, communication starts;
- Step 13: Query and wait for the SPI_ISR.TXE flag to be set to 1, and confirm that the transmit buffer is empty;
- Step 14: Write a frame of data to be transmitted into the SPI_DR register;
- Step 15: If there is still data to be transmitted, repeat steps 13~15 to transmit the next frame of data;
- Step 16: Query and wait for the SPI_ISR.TXE flag bit to be 1, and the last frame of data starts to be transmitted;
- Step 17: Query and wait for the SPI_ISR.BUSY flag to become 0, and the SPI bus is idle;
- Step 18: Set SPI_SSI.SSI to 1, communication ends.



17.5.3.2 Receive in query mode

- Step 1: Set SYSCTRL_AHBEN.GPIOx to 1, SYSCTRL_APBEN1.SPI to 1, and enable the GPIO clock and SPI working clock corresponding to the SPI pin;
- Step 2: Configure the SPI_SCK and SPI_CS pins as push-pull multiplexing output mode, and configure the SPI_MISO pin as floating input and multiplexing mode. For specific register configuration steps, please refer to chapter [8 General-purpose input/output \(GPIO\)](#);
- Step 3: Set SPI_CR1.MODE to 10, and configure SPI as simplex single receiving communication mode;
- Step 4: Set SPI_CR1.MSTR to 1, and configure SPI to master mode;
- Step 5: Configure SPI_CR1.WIDTH and set the width of each frame of data;
- Step 6: Configure SPI_CR1.LSBF and set the data bit transmitting and receiving sequence;
- Step 7: Configure SPI_CR1.CPOL and set the clock polarity;
- Step 8: Configure SPI_CR1.CPHA and set the clock phase;
- Step 9: Set SPI_CR1.SSM to 1, and determine the slave select output value through the SSI register;
- Step 10: Configure SPI_CR1.BR and set the SCK baud rate;
- Step 11: Set SPI_CR2.EN to 1 to enable the SPI controller;
- Step 12: Set SPI_SSI.SSI to 0, communication starts;
- Step 13: Query and wait for the SPI_ISR.TXE flag to be set to 1, and confirm that the transmit buffer is empty;
- Step 14: Write a frame of dummy data into the SPI_DR register;
- Step 15: Query and wait for the SPI_ISR.RXNE flag to be set to 1 to confirm that one frame of data has been received;
- Step 16: Read the SPI_DR register and save the data;
- Step 17: Repeat steps 13~17 to receive the next frame of data until all data is received;
- Step 18: Set SPI_SSI.SSI to 1, communication ends.



17.6 List of registers

SPI base address: SPI_BASE = 0x4000 0800

Table 17-4 List of SPI registers

Register name	Register address	Register description
SPI_CR1	SPI_BASE + 0x00	Control register 1
SPI_CR2	SPI_BASE + 0x04	Control register 2
SPI_CR3	SPI_BASE + 0x08	Control register 3
SPI_SSI	SPI_BASE + 0x0C	Slave select register
SPI_IER	SPI_BASE + 0x10	Interrupt enable register
SPI_ISR	SPI_BASE + 0x14	Interrupt flag register
SPI_ICR	SPI_BASE + 0x18	Interrupt flag clear register
SPI_DR	SPI_BASE + 0x1C	Data register



17.7 Register descriptions

See section [1 Documentation conventions](#) for abbreviations used in register descriptions.

17.7.1 SPI_CR1 Control register 1

Address offset: 0x00 Reset value: 0x0000 1C04

Bit field	Name	Permission	Function description
31:21	RFU	-	Reserved bits, please keep the default value
20	SMP	RW	Master mode delayed sampling 0: Normal sampling mode, the sampling time is determined by CPOL and CPHA. 1: Delayed sampling mode, sampling time is delayed about 20ns compared to normal sampling mode.
19	MISOHD	RW	Slave MISO output configuration 0: MISO is always COMS output 1: MISO is COMS output when the slave is selected, high resistance output when it is unselected
18	FLTEN	RW	Filter function enabled 0: Disabled 1: Enabled <i>Caution: Digital filter circuit can filter out small pulses less than one PCLK period on SCK, MISO, MOSI signals, but it will add an additional delay of about 2 PCLK clocks. Digital filtering is only used when SPI communication rate is significantly lower than PCLK clock frequency ($f_{SCK} < f_{PCLK}/8$).</i>
17:16	MODE	RW	Communication mode configuration 00: Bidirectional full duplex 01: Simplex single transmitting 10: Simplex single receiving 11: Single-wire half-duplex
15:12	GAP	RW	Master mode frame gap 0000: No gap 0001: $1T_{SCK}$ 0010: $2T_{SCK}$ 1111: $15T_{SCK}$
11:8	WIDTH	RW	The width of each frame of data is WIDTH+1, for example 0011: 4bit 0100: 5bit ... 1110: 15bit 1111: 16bit <i>Caution: Write 0~2 automatically becomes 8 bits.</i>



Bit field	Name	Permission	Function description
7	LSBF	RW	Data frame high and low sequence selection 0: Most significant bit (MSB) is sent and received first 1: Least significant bit (LSB) is sent and received first
6:4	BR	RW	Master mode SCK baud rate configuration 000: PCLK/2 001: PCLK/4 010: PCLK/8 011: PCLK/16 100: PCLK/32 101: PCLK/64 110: PCLK/128 111: PCLK/256
3	CPOL	RW	Serial clock polarity configuration 0: Low level in standby 1: High level in standby
2	CPHA	RW	Serial clock phase configuration 0: Front edge sampling/back edge shifting 1: Front edge shifting/back edge sampling
1	SSM	RW	Slave select configuration (master mode) 0: SPI_CS pin is in input mode, pin being at low level to select the local machine 1: SPI_CS pin is in output mode, output level from SSI Slave select configuration (slave mode) 0: SPI_CS pin level determines whether this machine is selected or not 1: SSI register value determines whether this machine is selected or not
0	MSTR	RW	Operating mode configuration 0: Slave mode 1: Master mode

Caution:

This register can only be modified if SPI_CR2.EN is 0.



17.7.2 SPI_CR2 Control register 2

Address offset: 0x04 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:5	RFU	-	Reserved bits, please keep the default value
4	ADCTX	RW	Transmit buffer null trigger ADC enable control 0: Disabled 1: Enabled
3	ADCRX	RW	Receive data completion triggers ADC enable control 0: Disabled 1: Enabled
2:1	RFU	-	Reserved bits, please keep the default value
0	EN	RW	Enable control 0: Disable SPI module 1: Enable SPI module

17.7.3 SPI_CR3 Control register 3

Address offset: 0x08 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:1	RFU	-	Reserved bits, please keep the default value
0	HDOE	RW	When single-wire half-duplex, data transmission/reception status control 0: Only transmitted 1: Only received <i>Caution: Only valid for single-wire half-duplex communication.</i>



17.7.4 SPI_SSI Slave select register

Address offset: 0x0C Reset value: 0x0000 0001

Bit field	Name	Permission	Function description			
31:1	RFU	-	Reserved bits, please keep the default value			
0	SSI	RW	Slave select, valid when SPI_CR1.SSM is 1			
			SSM	MSTR	SSI	Description
			0	-	-	Invalid
			1	0(Slave)	0	Set the machine to selected state
					1	Set this machine to unselected state
				1(Master)	0	SPI_CS pins output low
					1	SPI_CS pins output high



17.7.5 SPI_IER Interrupt enable register

Address offset: 0x10 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:8	RFU	-	Reserved bits, please keep the default value
7	MODF	RW	Mode error interrupt enable control 0: Disabled 1: Enabled
6	SSERR	RW	Slave select error interrupt enable control in slave mode 0: Disabled 1: Enabled
5	OV	RW	Receive buffer overflow error interrupt enable control 0: Disabled 1: Enabled
4	UD	RW	Transmit buffer underflow error interrupt enable control in slave mode 0: Disabled 1: Enabled
3	SSR	RW	Slave select input rising edge interrupt enable control 0: Disabled 1: Enabled
2	SSF	RW	Slave select input falling edge interrupt enable control 0: Disabled 1: Enabled
1	RXNE	RW	Receive buffer non-empty interrupt enable control 0: Disabled 1: Enabled
0	TXE	RW	Transmit buffer empty interrupt enable control 0: Disabled 1: Enabled



17.7.6 SPI_ISR Interrupt flag register

Address offset: 0x14 Reset value: 0x0000 0001

Bit field	Name	Permission	Function description
31:10	RFU	-	Reserved bits, please keep the default value
9	SSLVL	RO	Slave select signal state 0: Slave select signal is low level 1: Slave select signal is high level
8	BUSY	RO	Bus busy flag 0: Bus idle 1: Bus busy
7	MODF	RO	Mode error flag 0: Normal 1: Error <i>Caution: In master mode, the slave select input is low to trigger MODF.</i>
6	SSERR	RO	Slave select error flag in slave mode 0: Normal 1: Error
5	OV	RO	Receive buffer overflow error flag bit 0: Normal 1: Error
4	UD	RO	Transmit buffer underflow error flag in slave mode 0: Normal 1: Error
3	SSR	RO	Slave select input rising edge flag bit 0: No rising edge occurred 1: Rising edge has occurred <i>Caution: It only makes sense when CS uses hardware pins in slave mode (MSTR=0 and SSM=0).</i>
2	SSF	RO	Slave select input falling edge flag bit 0: No falling edge occurred 1: Falling edge occurred <i>Caution: It only makes sense when CS uses hardware pins in slave mode (MSTR=0 and SSM=0).</i>
1	RXNE	RO	Receive buffer not empty flag bit 0: Receive buffer is empty 1: Receive buffer is non-empty <i>Caution: This flag can be cleared by reading the DR register or writing 0 to ICR.RXNE.</i>
0	TXE	RO	Send buffer empty flag bit 0: Send buffer is non-empty 1: Send buffer is empty



17.7.7 SPI_ICR Interrupt flag clear register

Address offset: 0x18 Reset value: 0x0000 00FF

Bit field	Name	Permission	Function description
31:8	RFU	-	Reserved bits, please keep the default value
7	MODF	R1W0	Mode error flag cleared W0: Mode error flag cleared W1: No function
6	SSERR	R1W0	Slave select error flag cleared in slave mode W0: Slave select error flag in slave mode cleared W1: No function
5	OV	R1W0	Receive buffer overflow error flag cleared W0: Receive buffer overflow error flag cleared W1: No function
4	UD	R1W0	Transmit buffer underflow error flag cleared in slave mode W0: Transmit buffer underflow error flag in slave mode cleared W1: No function
3	SSR	R1W0	Slave select input rising edge flag cleared W0: Slave select input rising edge flag cleared W1: No function
2	SSF	R1W0	Slave select input falling edge flag cleared W0: Slave select input falling edge flag cleared W1: No function
1	RXNE	R1W0	Receive buffer not empty flag cleared W0: Receive buffer non-empty flag cleared W1: No function
0	FLUSH	R1W0	Clear transmit buffer and shift register W0: The transmit buffer and shift register (ISR.TXE is set to 1 after clearing) cleared W1: No function

17.7.8 SPI_DR Data register

Address offset: 0x1C Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value
15:0	DR	RW	Write data to be transmitted Read as received data

18 I2C interface

18.1 Overview

The CW32L010 integrates a I2C controller, which can transmit data serially to the I2C bus according to the I2C specification at the set transmission rate (standard, fast, high speed), detect the status during communication, and support bus conflict and arbitration processing in multi-master communication.

18.2 Main features

- Supports master transmit/receive and slave transmit/receive four working modes
- Supports clock stretching (clock synchronization) and multi-master communication conflict arbitration
- Supports standard (100Kbps)/fast (400Kbps)/high speed (1Mbps) three operating rates
- Supports 7bit addressing function
- Supports 3 slave addresses
- Supports broadcast address
- Supports input signal noise filtering function
- Supports interrupt status query function
- Support SMBUS timeout detection
- Support communication with devices operating at lower voltages than MCUs (via VC)



18.3 Protocol description

The I2C bus uses two signal lines (the data line SDA and the clock line SCL) to transfer data between devices. SCL is a unidirectional clock line, which is fixedly driven by the master. SDA is a bidirectional data line, which is driven by the transmitting and receiving terminals in time sharing during data transmission.

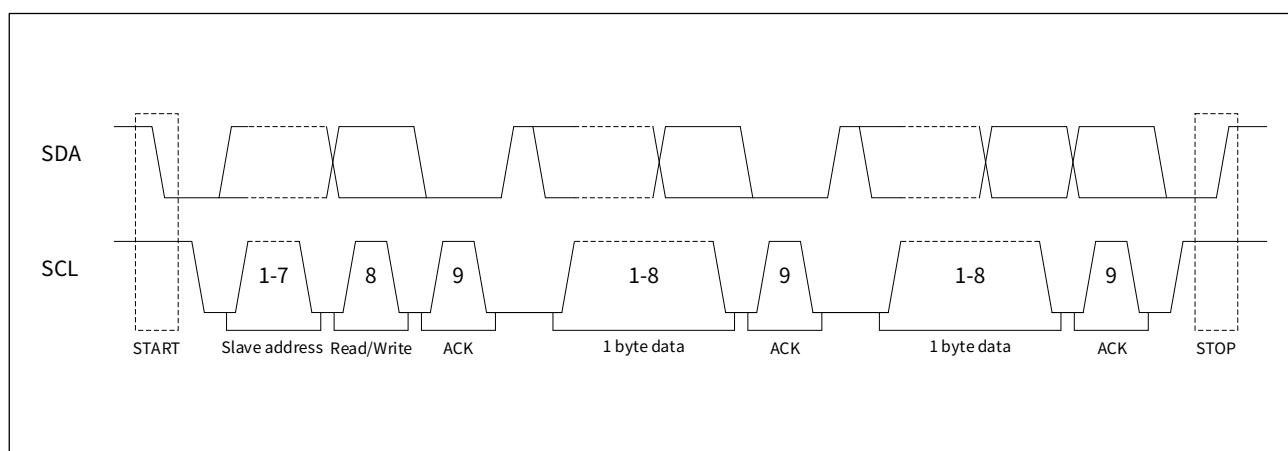
Multiple devices can be connected to the I2C bus. All devices are in an idle state when they are not transmitting data (the slave receiving mode is not addressed). Any device can send a START signal as a master to start data transmission. The bus is occupied until the STOP signal appears on the bus.

I2C communication adopts the master-slave structure, and the communication is initiated and terminated by the master. The master initiates communication by transmitting the START signal, then transmits a total of 8 bits of SLA+W/R data (where SLA is the 7-bit slave address, and W/R is the read-write bit), and releases the SDA bus at the ninth SCL clock, the corresponding slave occupies the SDA bus at the ninth SCL clock and outputs an ACK response signal to complete the slave addressing. After that, the transmitter and receiver of data communication are determined according to the W/R bit of the first byte transmitted by the master. For each byte of data transmitted by the transmitter, the receiver must respond with an ACK response signal. After the data transmission is completed, the master transmits a STOP signal to end the communication.

18.3.1 Protocol frame format

The standard I2C transmission protocol frame consists of four parts: start signal (START) or repeated start signal (Repeated START), slave address and read and write bits, data transmission, and stop signal (STOP). As shown in the following figure:

Figure 18-1 I2C protocol frame

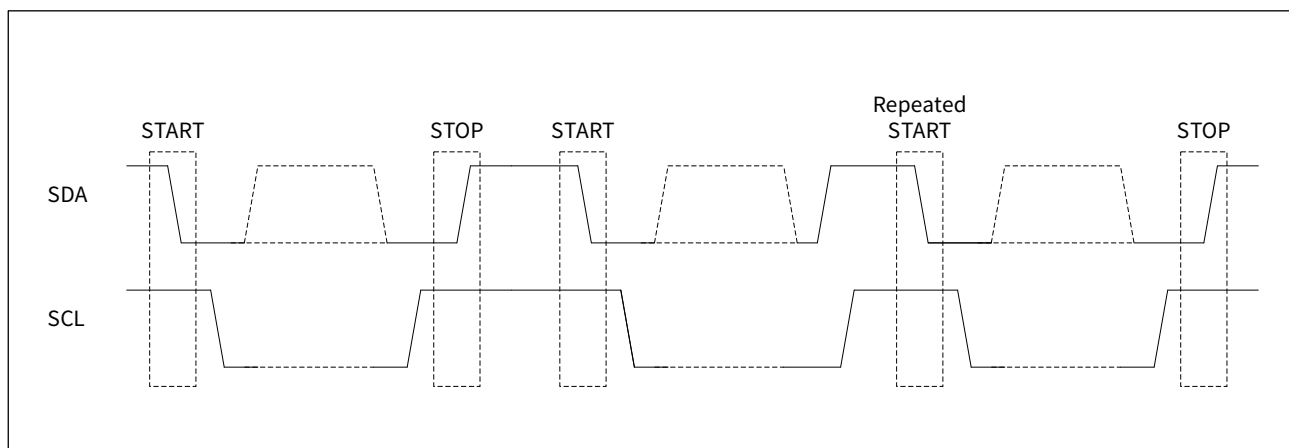


- Start signal (START)
When the bus is in an idle state (SCL and SDA lines are high at the same time), a falling edge signal from high to low appears on the SDA line, which is defined as a start signal. After the master transmits a start signal to the bus, data transmission starts and the bus is occupied.
- Repeated start signal (Repeated START)
When a start signal does not appear before a stop signal, a new start signal appears, and the new start signal is defined as a repeated start signal. Before the master transmits the stop signal, the SDA bus is always in the occupied state, and other masters cannot occupy the bus.

- Stop signal (STOP)

When the SCL line is high, a rising edge signal from low to high appears on the SDA line, which is defined as a stop signal. The master transmits a stop signal to the bus to end the data transfer and releases the bus.

Figure 18-2 Start and stop signals



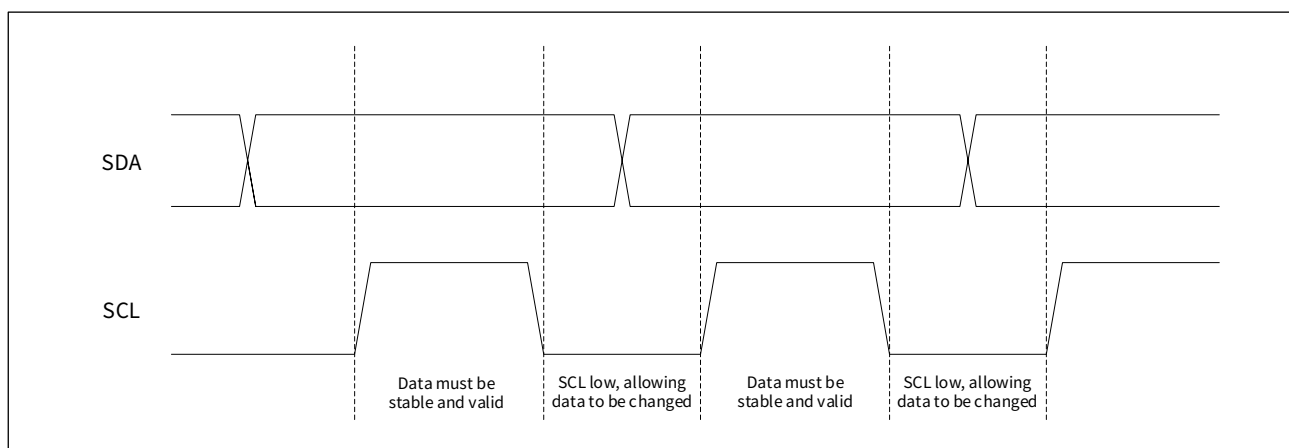
- Slave address and read/write bit

When the start signal is generated, the master starts to transmit the first byte of data: 7 bit slave address + read and write bits. The read and write bits (1: read; 0: write) control the direction of data transfer on the bus. The addressed slave occupies the SDA bus during the 9th SCL clock cycle and acknowledges the ACK by asserting SDA low.

- Data transmission

The master outputs the serial clock signal on the SCL line, and the master and slave devices transmit data through the SDA line. During data transmission, 1 SCL clock pulse transmits 1 data bit (the most significant bit MSB is first), and the data on the SDA line changes only when SCL is low, and each transmitted 1 byte is followed by an acknowledge bit. As shown in the following figure:

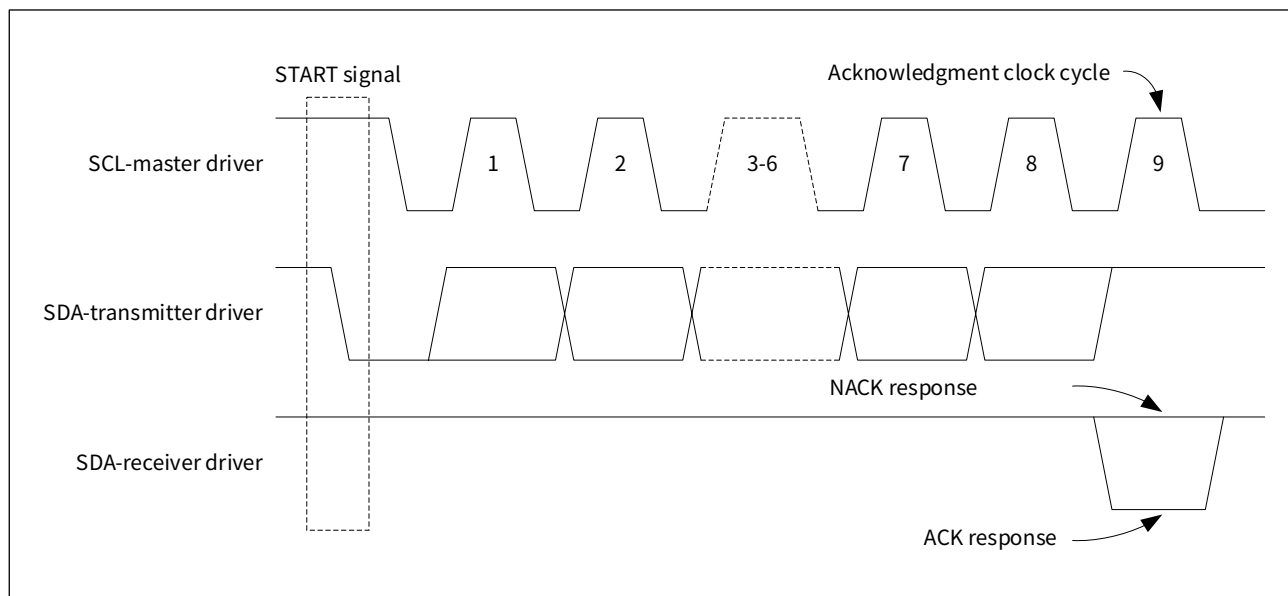
Figure 18-3 Data transmission



18.3.2 Transmission response

When transmitting data on the bus, the transmitter gives up the control of SDA in the 9th SCL clock cycle after transmitting 1 byte of data, and the receiver must reply a response bit in the 9th SCL clock cycle: the reception is successful, Send ACK response, receive exception send NACK response.

Figure 18-4 Transmission response



18.3.3 Conflict detection and arbitration

In a multi-master communication system, each node on the bus has a slave address. Each node can be accessed by other nodes as a slave node, and can also be used as a master node to send control bytes and transmit data to other nodes. A bus collision occurs when two or more nodes send a start signal to the bus at the same time and start transmitting data. The I2C controller has a built-in arbiter that can detect and arbitrate I2C bus conflicts to ensure the reliability and integrity of data communication.

- Conflict detection principle

In the physical realization, the SDA and SCL pins have the same circuit structure, and the output drive of the pin is connected with the input buffer. The output structure is an open-drain field effect transistor, and the input structure is an inverter with high input impedance. Based on this structure:

1. Since SDA and SCL are open-drain structures, the "wire-and" logic of the signal is realized by means of an external pull-up resistor;
2. The device reads data while writing data to the bus, which can be used to detect bus conflicts and realize "clock synchronization" and "bus arbitration".

According to the "wire-and" logic, if two masters send logic 1 or logic 0 at the same time, neither of the two masters can detect the conflict, and it needs to wait until the next bit of data is transmitted before continuing to detect the conflict; if one of the two hosts sends a logic 1, When a logic 0 is transmitted, the bus is logic 0 at this time, the master transmitting logic 1 detects a conflict, and the master transmitting logic 0 does not detect a conflict.

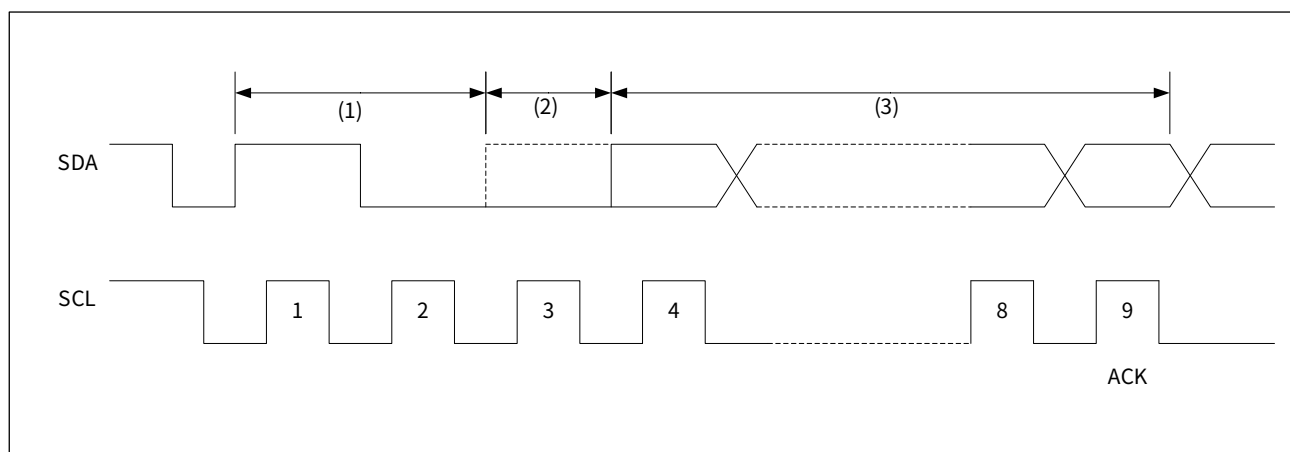
- Conflict arbitration principle

When the master detects a bus conflict, the master loses arbitration, exits the master transmit mode, enters the unaddressed slave mode, releases the SDA data line, and returns to the address detection state, and then enters the state according to the received SLA+W/R corresponding slave mode (SLA address match enters addressed slave mode, SLA address mismatch enters unaddressed slave mode). The master that loses arbitration will still send the SCL serial clock until the end of the current byte transfer.

When the master does not detect a bus conflict, the master wins the arbitration and continues to dominate the data transfer until the communication is complete.

The following figure is a schematic diagram of transmitting conflict and arbitration between master A and master B on an I2C bus (assuming that the transmitting clocks of the two masters A and B are synchronized):

Figure 18-5 Conflict detection and arbitration



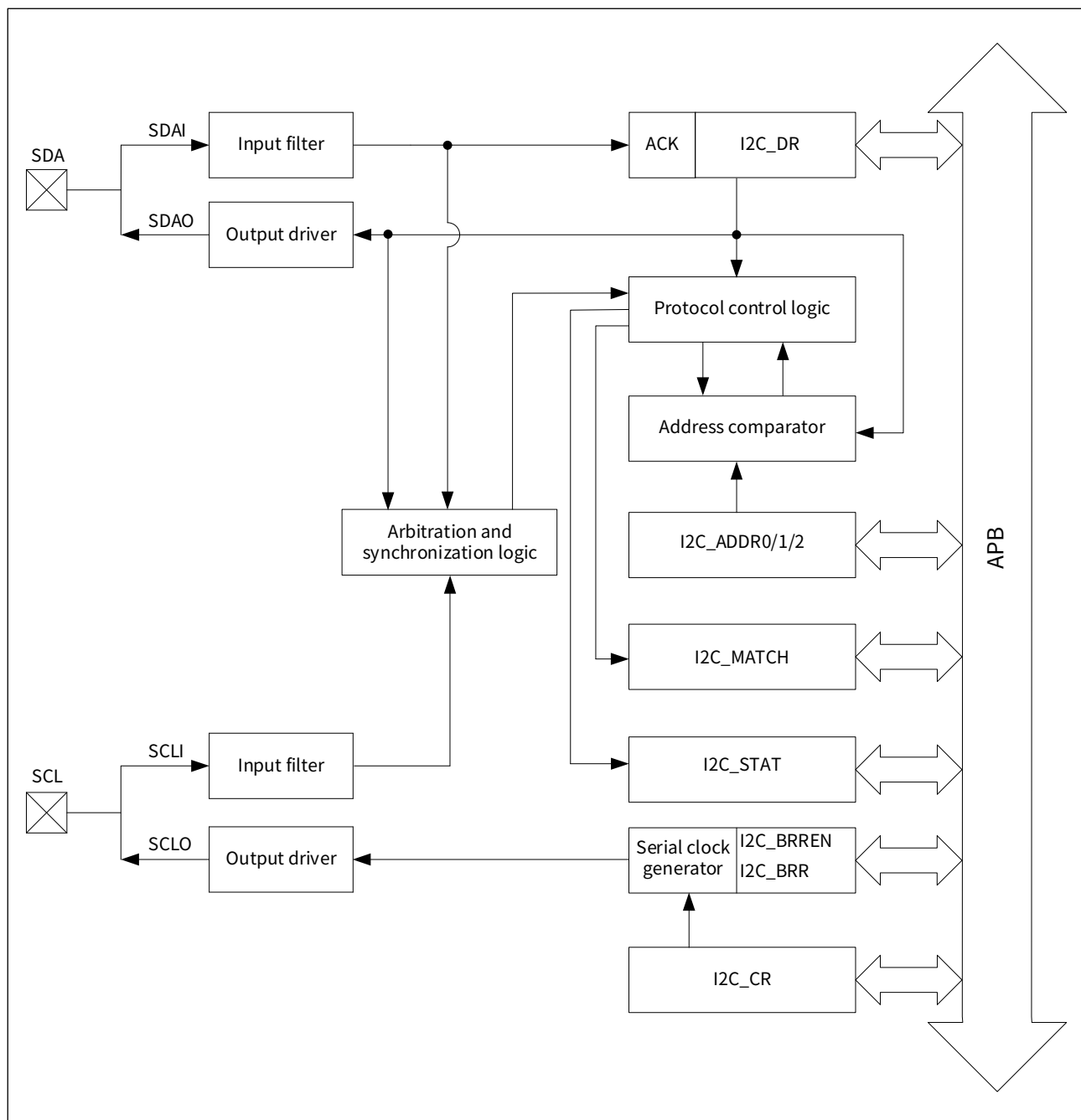
1. In stage (1), both master A and master B transmit logic 1 and logic 0 successively. Since the data is the same and synchronous, neither host will detect a bus conflict.
2. In stage (2), master A transmits a logic 1, master B transmits a logic 0, and the data on the SDA bus is a logic 0. The A host controller detects a data error and exits the transmission competition, that is, the arbitration is lost, and the A master enters the unaddressed slave receiver mode. Host B does not detect a bus conflict, wins the arbitration, and continues this data transfer.
3. In stage (3), master A is in unaddressed slave receive mode, but still generates SCL clock pulses until the end of the current byte transfer. After that, the A host I2C controller no longer transmits the clock signal. Since the B master wins the arbitration, both SCL and SDA are dominated by the B master to control the transfer.

18.4 Functional description

18.4.1 Functional block diagram

The I2C module mainly includes clock generator, input filter, address comparator, protocol control logic, arbitration and synchronization logic, and related registers. Its functional block diagram is shown in the following figure:

Figure 18-6 I2C functional block diagram



CW32L010 supports users to flexibly select GPIO as I2C communication pin, as shown in the following table:

Table 18-1 I2C pin selection

I2C	Pin
I2C_SCL	PA01/PA06/PA08/PB01/PB06
I2C_SDA	PA00/PA05/PA07/PB00/PB05

18.4.2 Serial clock generator

The serial clock generator is used to generate the baud rate clock SCL for I2C communication. The serial clock generator uses PCLK as the input clock, counts through an 8-bit counter, and outputs the I2C clock signal with the required baud rate.

SCL clock frequency calculation formula:

$$f_{SCL} = f_{PCLK} / 8 / (BRR + 1)$$

Where, BRR is configured through the baud rate counter configuration register I2C_BRR, and the effective range of BRR is 1~255.

The counter count of the serial clock generator is enabled by the EN bit field of the I2C_BRREN register. EN is 1 to enable and 0 to prohibit. When the master is used, EN should be set to 1, and this bit has no effect when the slave is used.

The corresponding relationship between the combination of PCLK and BRR and the SCL clock frequency is shown in the following table:

Table 18-2 I2C transfer rate and configuration example

$f_{PCLK}(kHz)$ \ $f_{SCL}(kHz)$	I2C_BRR						
	1	2	3	4	5	6	7
1000	62	41	31	25	20	17	15
2000	125	83	62	50	41	35	31
4000	250	166	125	100	83	71	62
6000	375	250	187	150	124	107	93
8000	500	333	250	200	166	142	125
10000	625	416	312	250	208	178	156
12000	750	500	375	300	250	214	187
14000	875	583	437	350	291	250	218
16000	1000	666	500	400	333	285	250



18.4.3 Input filter

The input filter can filter the SDA and SCL input signals, and the spike signal less than 1 PCLK cycle will be filtered out.

The input filter can be configured in two modes: simple filter mode and advanced filter mode. Simple filtering has faster communication rate; advanced mode has higher anti-jamming performance. The filtering mode is configured through the FLT bit field of the I2C control register I2C_CR. Setting I2C_CR.FLT to 1 is the simple filtering mode, and setting I2C_CR.FLT to 0 is the advanced filtering mode.

As a master, if the value of BRR is less than or equal to 9, I2C_CR.FLT should be set to 1; if the value of BRR is greater than 9, then I2C_CR.FLT should be set to 0.

As a slave, if the frequency ratio of PCLK to SCL is less than or equal to 40, then I2C_CR.FLT should be set to 1; if the frequency ratio of PCLK to SCL is greater than 40, then I2C_CR.FLT should be set to 0.

18.4.4 Address comparator

Each device on the I2C bus has a slave address, and each slave address is different. The master addresses the slave according to the slave address, and the slave automatically detects whether the 7bit addressing address transmitted by the master matches the local address through the address comparator to determine whether to communicate with the master.

The I2C controller supports 3 programmable slave addresses, and the specific address information is configured through the slave address registers I2C_ADDR0/I2C_ADDR1/I2C_ADDR2.

The address comparator of the slave machine compares the received 7-bit addressing address with the three slave addresses and the broadcast address (0x00). If it matches any one of the 4 addresses, it is considered that the address matches, and the I2C interrupt flag bit I2C_CR.SI will be set to 1, and an interrupt request will be generated. The application program can obtain the successfully matched address serial number by querying the slave address matching register I2C_MATCH.

If the address matches I2C_ADDR0/1/2, the slave enters the corresponding addressed slave receive mode (receives SLA+W) or addressed slave transmit mode (receives SLA+R); if the address matches to If the broadcast address is 0x00 (SLA+W is received), the slave enters the broadcast receiving mode.



18.4.5 Arbitration and synchronization logic

18.4.5.1 SCL synchronization

I2C supports clock synchronization (clock stretching) function, the time of SCL clock low level is determined by the device with the longest SCL clock low level width, and the SCL clock high level time is determined by the device with the shortest clock high level width.

If the slave wants the master to reduce the transmission speed, it can keep I2C_CR.SI 1 after receiving the data and responding to ACK, and the slave I2C controller will keep SCL low to notify the master. When the master is ready for the next byte data transfer (transmit or receive), it detects that the level of SCL is pulled low, and waits until the slave completes the operation and clears I2C_CR.SI to 0, and the slave controller releases the pull of SCL. Low control, the master continues to transmit the next byte of data after detecting that SCL is high.

18.4.5.2 SDA arbitration

I2C supports SDA conflict detection and arbitration, which can ensure that the data on the I2C bus is not destroyed when multiple masters attempt to control the I2C bus.

When each master transmits data, it will simultaneously compare whether the data on the bus is consistent with the data transmitted by itself. If they are inconsistent, it is considered that a bus conflict is detected, and the transmitting competition will be withdrawn, that is, the arbitration will be lost. A master that loses arbitration will immediately switch to an unaddressed slave state to ensure that it can be addressed by a master that succeeds in arbitration. A master that loses arbitration continues to output the SCL serial clock until the current byte transfer is complete.

SDA arbitration generally occurs when the master transmits SLA+W/R data. If two masters transmit data to a slave at the same time, that is, the slave addresses transmitted by the two masters are the same, the arbitration will continue in the second byte.



18.4.6 Response control

The receiver of the I2C data transmission must give an ACK or NACK response to the transmitter in the 9th SCL clock cycle of each byte. The receiver obtains the current state of the receiver through the response bit: responding with an ACK response means that the receiver has received it correctly. This byte of data can continue to receive the next byte of data; responding to NACK generally indicates that the receiver no longer receives any data.

Whether the receiver sends ACK or NACK is controlled by the AA bit field of the receiver's I2C control register I2C_CR. When I2C_CR.AA is set to 1, the I2C module will respond with an ACK response every time it receives 1 byte of data. When I2C_CR.AA is set to 0, the I2C module will respond with a NACK response every time it receives 1 byte of data.

In the process of receiving data by the master, the master as the initiator of communication, controls the number of bytes transmitted and received. The master (receiver) responds to the NACK response to the slave (transmitter) after receiving the last byte of data, and the slave receives the NACK. After the slave receives the NACK response, it will switch to unaddressed slave receive mode and release the SDA bus so that the master can transmit a STOP signal or a Repeated START signal.

In the process of transmitting data from the slave, if its own I2C_CR.AA response control bit is cleared by the application program, the slave will switch itself to the unaddressed slave receiving mode after transmitting the last 1 byte of valid data, and release the SDA bus, the master will get 0xFF when reading data from the bus. At this time, the master should be able to judge that the slave is in an unresponsive state, and send a STOP signal or a Repeated START signal.

In the process of receiving data from the slave, if it responds to NACK to the master (transmitter), it means that the slave (receiver) actively ends the communication, no longer receives the data transmits by the master, and switches itself to the unaddressed slave to receive mode, and release the SDA bus, at this time the master should send a STOP signal or a Repeated START signal.



18.4.7 Input signal source/level shift

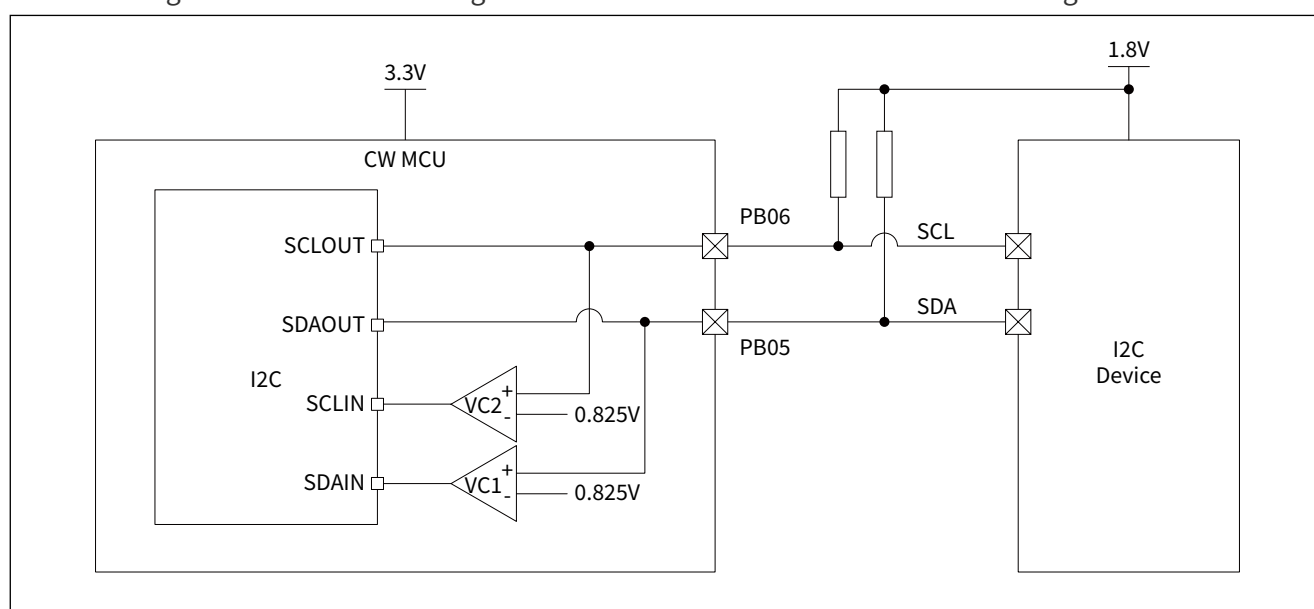
The SCL/SDA input signal of this module can be configured to a variety of sources, as shown in the following table:

Table 18-3 SCL/SDA input signal source

I2C_CR.SCLINSRC	SCL input signal source	I2C_CR.SDAINSRC	SDA input signal source
000	I2C_SCL pin	000	I2C_SDA pin
001	VC1_OUT	001	VC1_OUT
010	VC2_OUT	010	VC2_OUT

When the input signal source of SCL/SDA is VCx_OUT, this module can communicate with I2C devices lower than the operating voltage of this chip. VCx plays the role of level conversion in the communication link. The typical operating circuit is shown in the following figure:

Figure 18-7 Schematic diagram of communication between different voltage devices



Note that the PB06 pin shown above has the I2C_SCL function and the VC2_CH0 channel, and the PB05 pin has the I2C_SDA function and the VC1_CH0 channel. When in use, the digital function of PB06 needs to be configured as I2C_SCL and open drain is enabled, and the digital function of PB05 needs to be configured as I2C_SDA and open drain is enabled; the input signal source of I2C_SCL needs to be configured as VC2_OUT, and the input signal source of I2C_SDA needs to be configured as VC1_OUT; the positive terminal of VC2 needs to be configured as CH0 (PB06 pin), and the positive terminal of VC1 needs to be configured as CH0 (PB05 pin); The negative terminals of VC1/VC2 shall be configured to be 0.825V of resistor voltage divider output.

18.4.8 SMBUS timeout detection

I2C and BTIM1 work together to detect SMBUS timeouts via internal interconnect signals. BTIM1 clears the count whenever it detects a falling edge of SCL, accumulates the count whenever SCL is low, and overflows to indicate SMBUS timeout detection when the low lasts longer than expected by ARR.

The main configuration steps are as follows:

- Step 1: Set BITM1_SMCR.SMS to 010 to enable BTIM1 to work in gated counting mode;
- Step 2: Set BITM1_SMCR.TRGISRC to 0100, BTIM1 gating signal source is I2C_SCL;
- Step 3: Set BITM1_SMCR.TRGIPOL to 1, BTIM1 gating level is valid at low level;
- Step 4: Set BITM1_SMCR.RSTISRC to 0001, BTIM1 count value clear signal source is I2C_SCL;
- Step 5: Set BITM1_SMCR.RSTIPOL to 1, the BTIM1 count clear signal is valid on falling edge;
- Step 6: Configure BTIM1_ARR appropriately according to SCL low timeout specified by SMBUS.

18.4.9 I2C interrupts

The SI bit field of the I2C control register I2C_CR is the interrupt flag bit. When the value of the STAT bit field of the I2C status register I2C_STAT changes (except for 0xF8), the I2C_CR.SI flag will be set and an interrupt request will be generated.

In the user I2C interrupt service routine, you should query the STAT bit field value of the I2C status register I2C_STAT to obtain the status of the I2C bus to determine the cause of the interrupt. Set I2C_CR.SI to 0 to clear this flag.



18.4.10 Operating modes

The I2C controller supports 4 operating modes: master transmit mode, master receive mode, slave transmit mode, and slave receive mode. In addition, the broadcast receive mode is also supported, and its working method is similar to that of the slave receive mode.

18.4.10.1 Master transmit mode

In master transmit mode, the master actively transmits multiple bytes to the slave.

The SCL serial clock is controlled by the master, so it is necessary to set the I2C_BRR register according to the transmission baud rate and set the EN bit field of the I2C_BRREN register to 1, and then start the transfer.

The master sets I2C_CR.STA to 1, and informs the controller to transmit a START signal. After the controller receives the notification, it detects whether the bus is idle. When the bus is idle, the master controller transmits a START signal to the bus. If the transmission is successful, the status code I2C_STAT becomes 0x08, and the interrupt flag bit I2C_CR.SI is set to 1.

After the master transmits the START signal, software needs to set I2C_CR.STA to 0, and then write the slave address and write flag bit (SLA+W) to the I2C data register I2C_DR; clear the I2C_CR.SI bit, the master controller SLA+W will be transmitted to the I2C bus; when the master transmits SLA+W and receives the ACK response signal from the slave, the status code I2C_STAT becomes 0x18, and the interrupt flag bit I2C_CR.SI is set to 1.

After that, the master transmits multiple bytes of user-defined data and detects the ACK response signal according to the needs of the application. The transmitting process of each byte is similar to that of sending SLA+W data frames.

In the process of transmitting data, if the master receives a NACK response signal, it indicates that the slave no longer receives the data sent by the master (I2C_CR.AA of the slave is cleared), the master should send a STOP signal to end the data transfer, or send Repeated START repeats the start signal to start a new round of transfer.

When the master completes transmitting all the data, set I2C_CR.STO to 1 to notify the controller that the data has been transmitted and the STOP signal is to be transmitted. Clear the I2C_CR.SI bit, and the master controller will transmit a STOP signal to the I2C bus to complete this data transfer and release the bus.

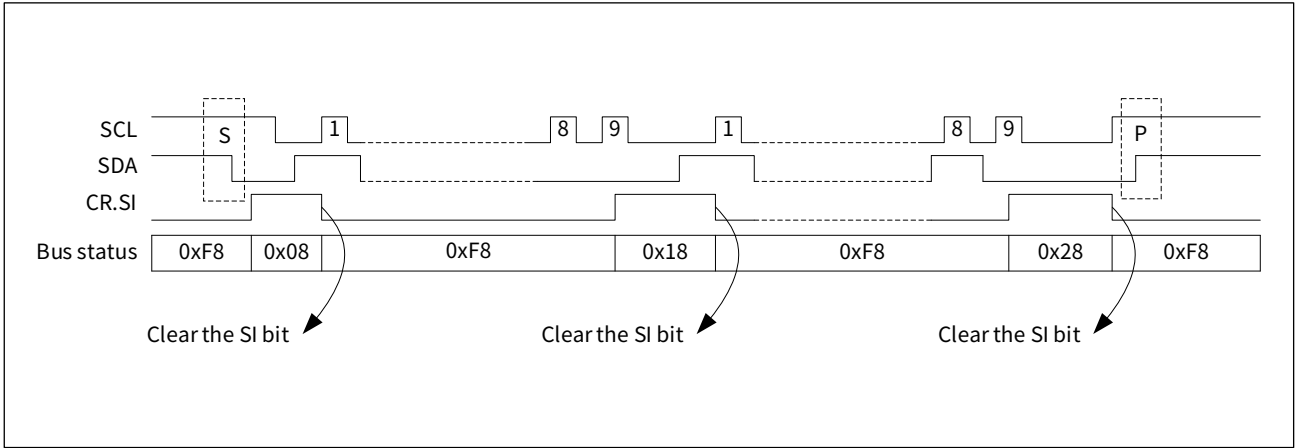
When the master has finished transmitting all the data, it can also not transmit the STOP signal, but directly transmit the Repeated START signal, and continue to occupy the bus for a new round of data transmission.

When the master loses arbitration due to a bus conflict, it enters the unaddressed slave receive mode (status code I2C_STAT=0x38).

The state synchronization diagram in the master transmit mode is shown in the following figure:



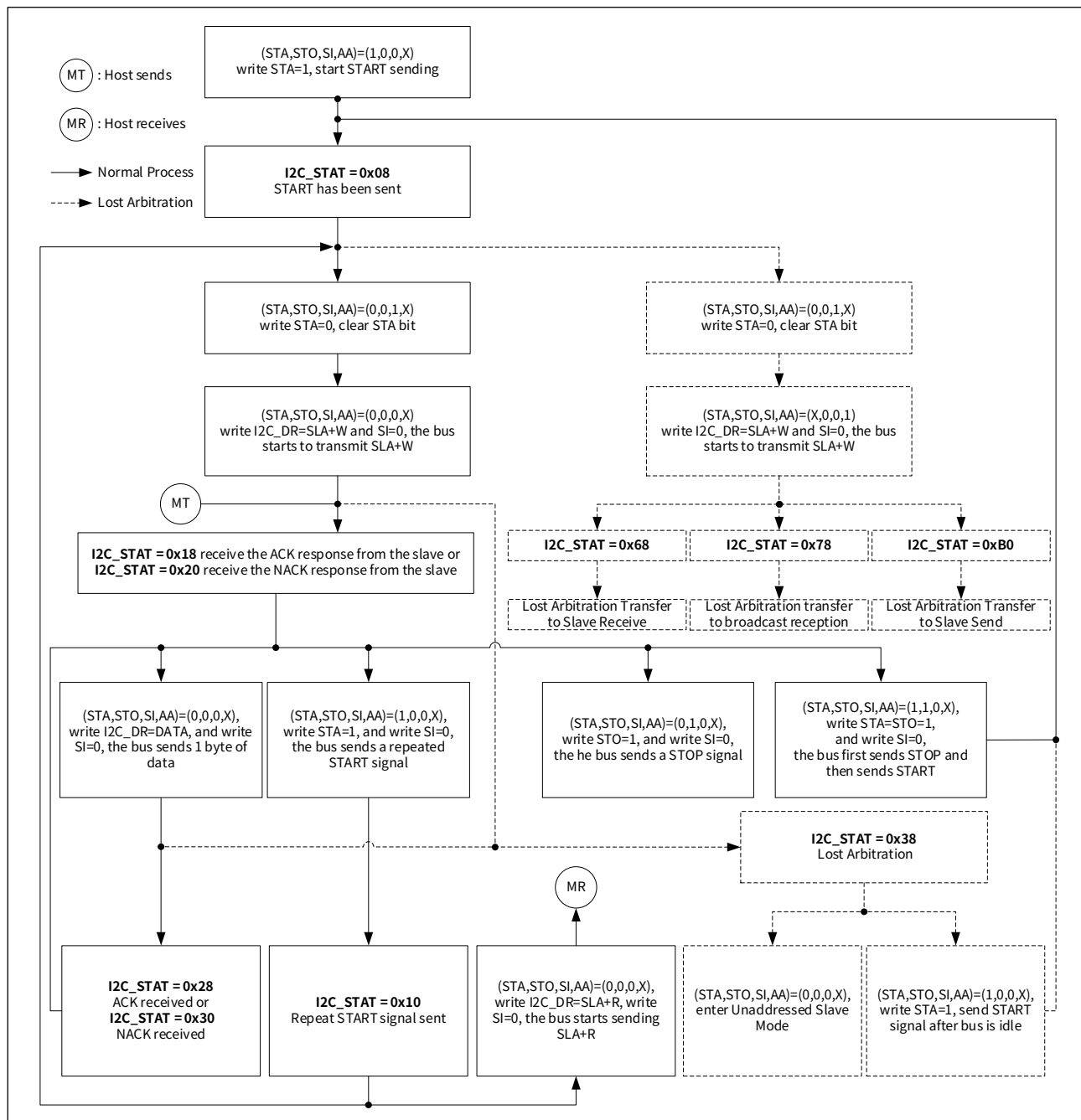
Figure 18-8 The status synchronization diagram of master transmit mode



The data transmission flow chart and status register values in the master transmit mode are shown in the following figure:



Figure 18-9 Master transmit mode flow and register status



18.4.10.2 Master receive mode

The master receive mode is used to receive multiple data transmitted by the slave, and the master will respond with an ACK response signal every time it receives 1 byte of data.

The SCL serial clock is controlled and generated by the master, so it is necessary to set the I2C_BRR register of the master according to the transmission baud rate and set the EN bit field of the I2C_BRREN register to 1, and then start the transfer.

The master sets I2C_CR.STA to 1, and informs the controller to send a START signal. After the controller receives the notification, it detects whether the bus is idle. When the bus is idle, the master controller sends a START signal to the bus. If the transmission is successful, the status code I2C_STAT becomes 0x08, and the interrupt flag bit I2C_CR.SI is set to 1.

After the master sends the START signal, the software needs to set I2C_CR.STA to 0, and then write the slave address and read flag bit (SLA+R) to the I2C data register I2C_DR; clear the I2C_CR.SI flag bit, the master controls the controller will send SLA+R to the I2C bus; when the master sends SLA+R and receives the ACK response signal from the slave, the status code I2C_STAT becomes 0x40, and the interrupt flag bit I2C_CR.SI is set to 1.

After that, the master sets I2C_CR.AA to 1, and clears the I2C_CR.SI bit, and starts to receive the data transmitted by the slave. After receiving 1 byte of data, it will reply an ACK response signal. During the master receiving process, it should be cautioned that:

1. In order to ensure that the I2C_CR.SI interrupt signal can be correctly generated after receiving 1 byte of data, it is necessary to clear the I2C_CR.SI bit in time after receiving 1 byte of data.
2. Before receiving the last byte, I2C_CR.AA needs to be cleared, that is, the master does not generate an ACK response signal when receiving the last byte, so as to notify the slave to stop data transmission.

In the process of receiving data by the master, if the slave no longer transmits the data required by the master for some reason (the I2C_CR.AA of the slave is cleared), the master will receive all 1 signals later. At this time, the master needs to be in the application layer. Judgment is made on the data, and it is determined that the slave is in an unresponsive state. It should send a STOP signal to end the current transmission, or send a Repeated START signal to start a new round of transmission.

When the master completes receiving all the data, set I2C_CR.STO to 1 to notify the controller that the data has been received and to send a STOP signal. Clear the I2C_CR.SI bit, and the master controller sends a STOP signal to the I2C bus to complete the data transfer and release the bus.

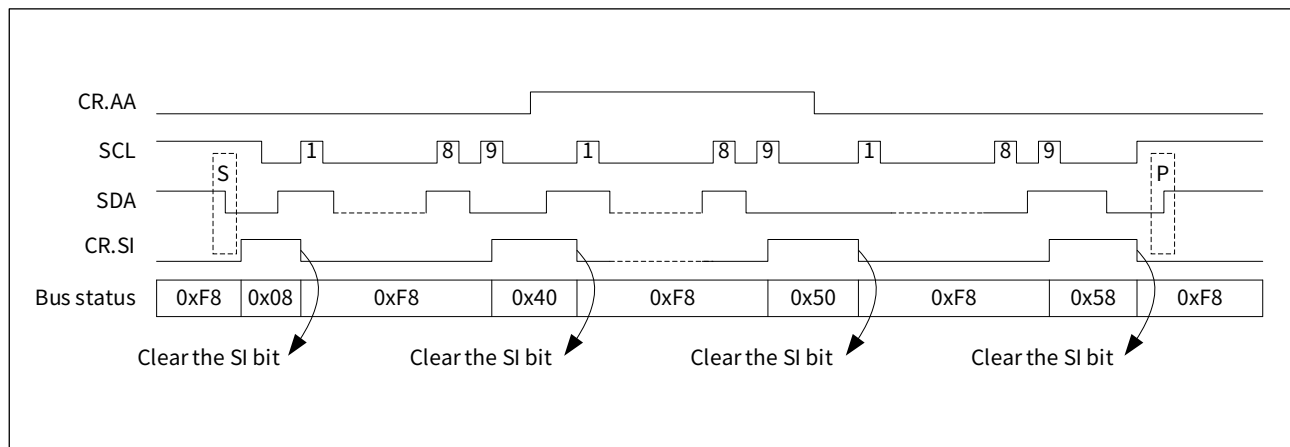
When the master has finished receiving all the data, it can also not send the STOP signal, but directly send the Repeated START signal, and continue to occupy the bus for a new round of data transmission.

When the master loses arbitration due to a bus conflict, it enters the unaddressed slave receive mode (status code I2C_STAT=0x38).

The state synchronization diagram in the master receive mode is shown in the following figure:

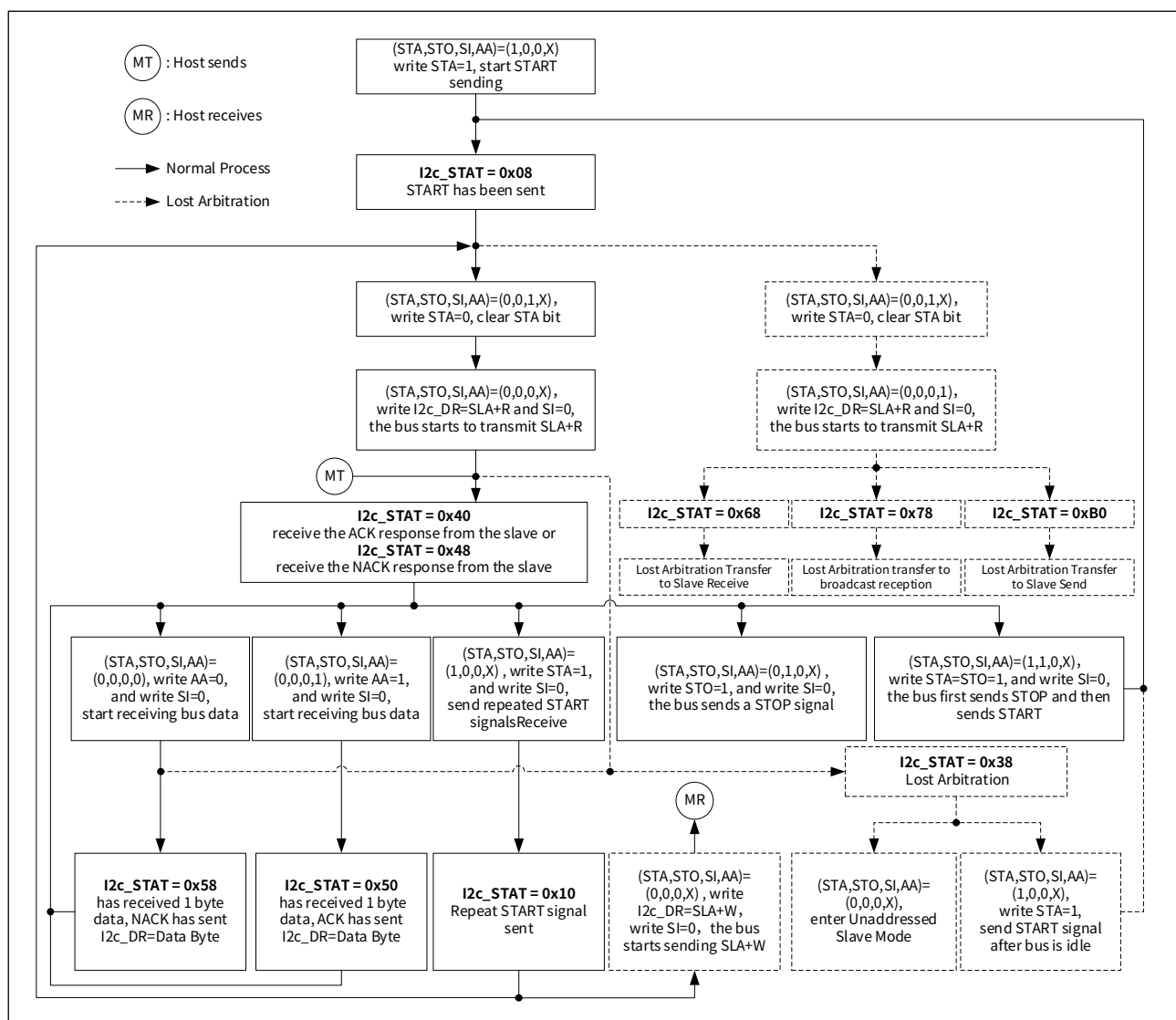


Figure 18-10 The status synchronization diagram of master receive mode



The data reception flow chart and status register value in the master receive mode are shown in the following figure:

Figure 18-11 Master receive mode flow and register status



18.4.10.3 Slave receive mode

In slave receive mode, the slave receives the data transmitted by the master.

Before transfer, the slave needs to set the slave address: write the slave address to any one of the three slave address registers I2C_ADDR0, I2C_ADDR1, I2C_ADDR2; set I2C_CR.AA to 1 to respond to the master's addressing; The set value of the I2C_BRR register is invalid, so it is not necessary to set it.

After completing the above initialization work, the slave machine enters the idle state (the slave receiving mode is not addressed), waiting to be addressed by the write signal (SLA+W) sent by the master. After the slave receives SLA+W, if the address matches I2C_ADDR0/1/2, the slave responds with ACK and enters the addressed slave receiving mode, the status code I2C_STAT becomes 0x60, and the interrupt flag bit I2C_CR.SI is also set to 1. The I2C_CR.SI bit must be cleared at this time in order to receive data from the master from the bus.

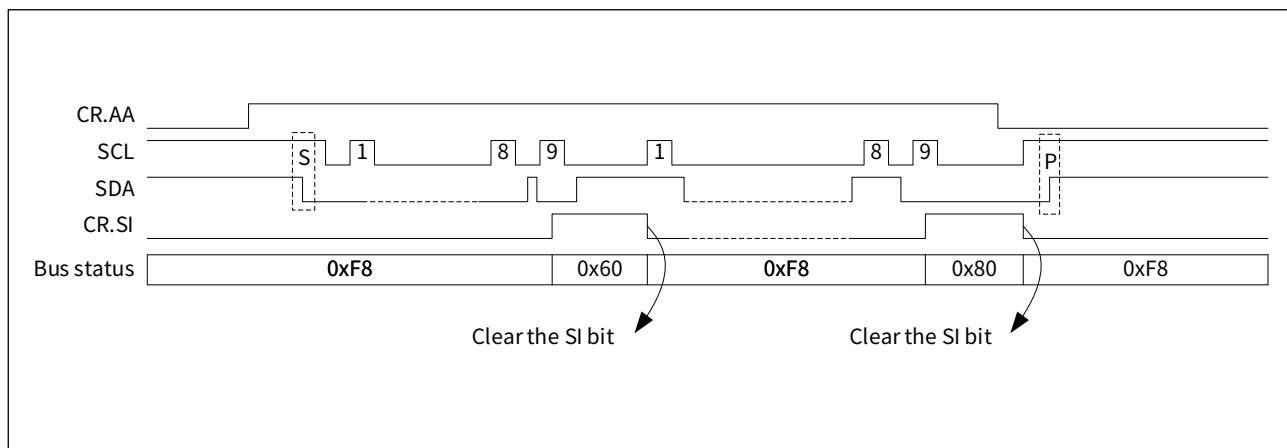
Every time the slave receives 1 byte of data, it must respond with an ACK response. After the application reads the byte of data, it must clear the I2C_CR.SI bit to prepare for receiving the next byte of data.

In the process of receiving data from the slave, if I2C_CR.AA is cleared, the slave will return a NACK signal when it receives the next byte, and the state of the slave will also switch to the unaddressed slave receiving mode, ending with the master communication, no more data is received, and the I2C_DR register holds the previously received data. It can be seen from this feature that the slave application can actively switch the slave from the addressed slave receive mode to the unaddressed slave receive mode by setting I2C_CR.AA to 0.

When the master loses arbitration due to bus conflict in the SLA+ read and write stage, it will enter the unaddressed slave receiving mode, and then if it receives SLA+W that matches the local address and responds with ACK (status code I2C_STAT=0x68), it will enter addressed slave receive mode.

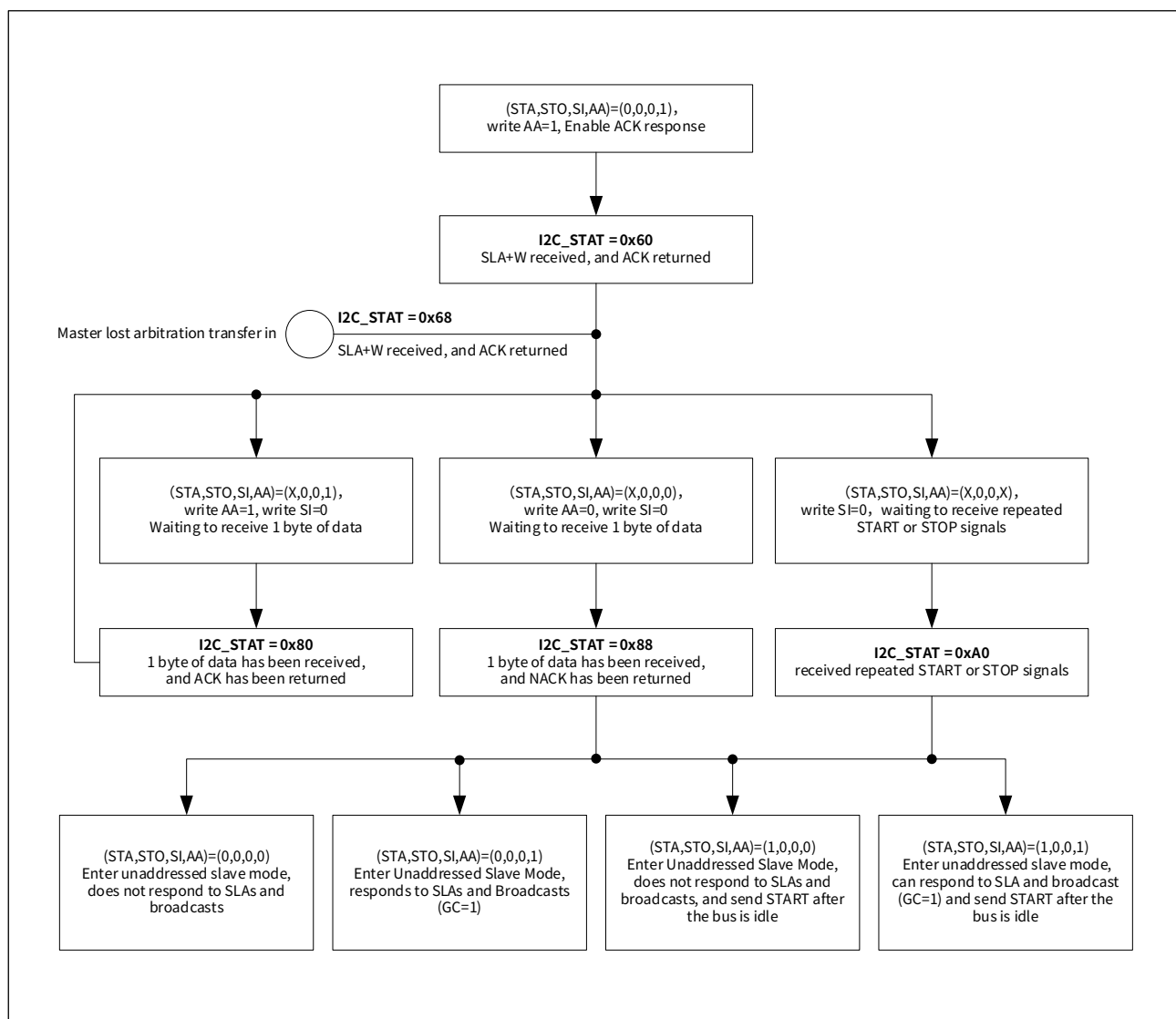
The state synchronization diagram in slave receive mode is shown in the following figure:

Figure 18-12 The status synchronization diagram of slave receive mode



The data receive flow chart and status register value in slave receive mode are shown in the following figure:

Figure 18-13 Slave receive mode flow and register status



18.4.10.4 Slave transmit mode

In slave transmit mode, data is transmitted from the slave to the master.

Before transfer, the slave needs to set the slave address: write the slave address to any one of the three slave address registers I2C_ADDR0, I2C_ADDR1, I2C_ADDR2; set I2C_CR.AA to 1 to respond to the master's addressing; The set value of the I2C_BRR register is invalid, so it is not necessary to set it.

After completing the above initialization work, the slave machine enters the idle state (the slave receive mode is not addressed), waiting to be addressed by the read signal (SLA+R) transmitted by the master. After the slave receives SLA+R, if the address matches I2C_ADDR0/1/2, the slave responds with ACK and enters the addressed slave transmit mode, the status code I2C_STAT becomes 0xA8, and the interrupt flag bit I2C_CR.SI is also set to 1. At this time, the data to be transmitted should be written into the I2C_DR register in time, and the I2C_CR.SI bit should be cleared, waiting for 1 byte of data to be transmitted, and ACK confirmation after each byte of data is transmitted until all data is transmitted.

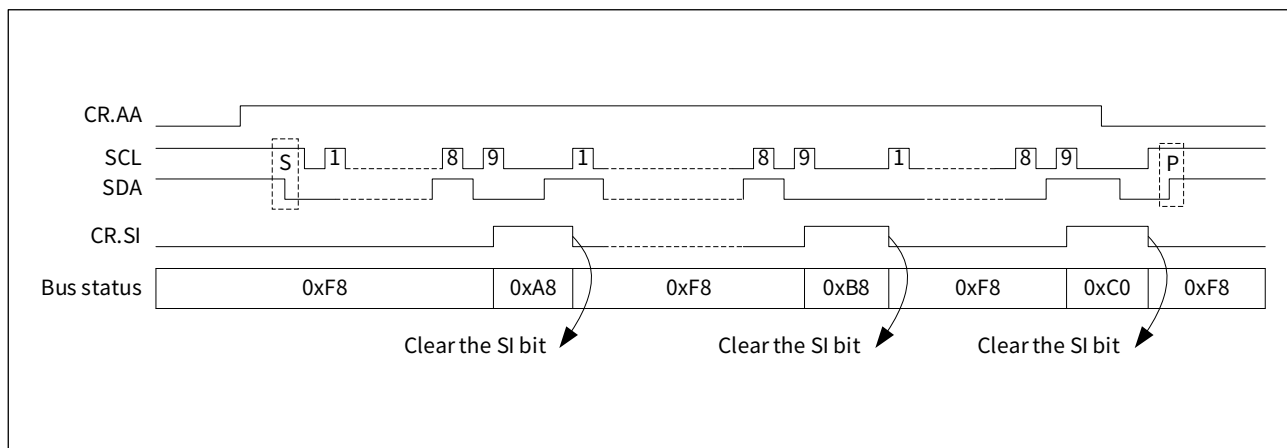
If the slave receives a NACK response during transfer, the slave no longer transmits data and enters the unaddressed slave receive mode.

If the slave actively sets I2C_CR.AA to 0 during the transmission process, the slave switches itself to the unaddressed slave receive mode after transmitting the last 1 byte of valid data, after which the master will get 0xFF when reading data from the bus.

When the master loses arbitration due to bus conflict in the SLA+ read and write stage, it will enter the unaddressed slave receive mode, and then if it receives SLA+R that matches the local address and responds with ACK (status code I2C_STAT=0xB0), it will enter addressed slave transmit mode.

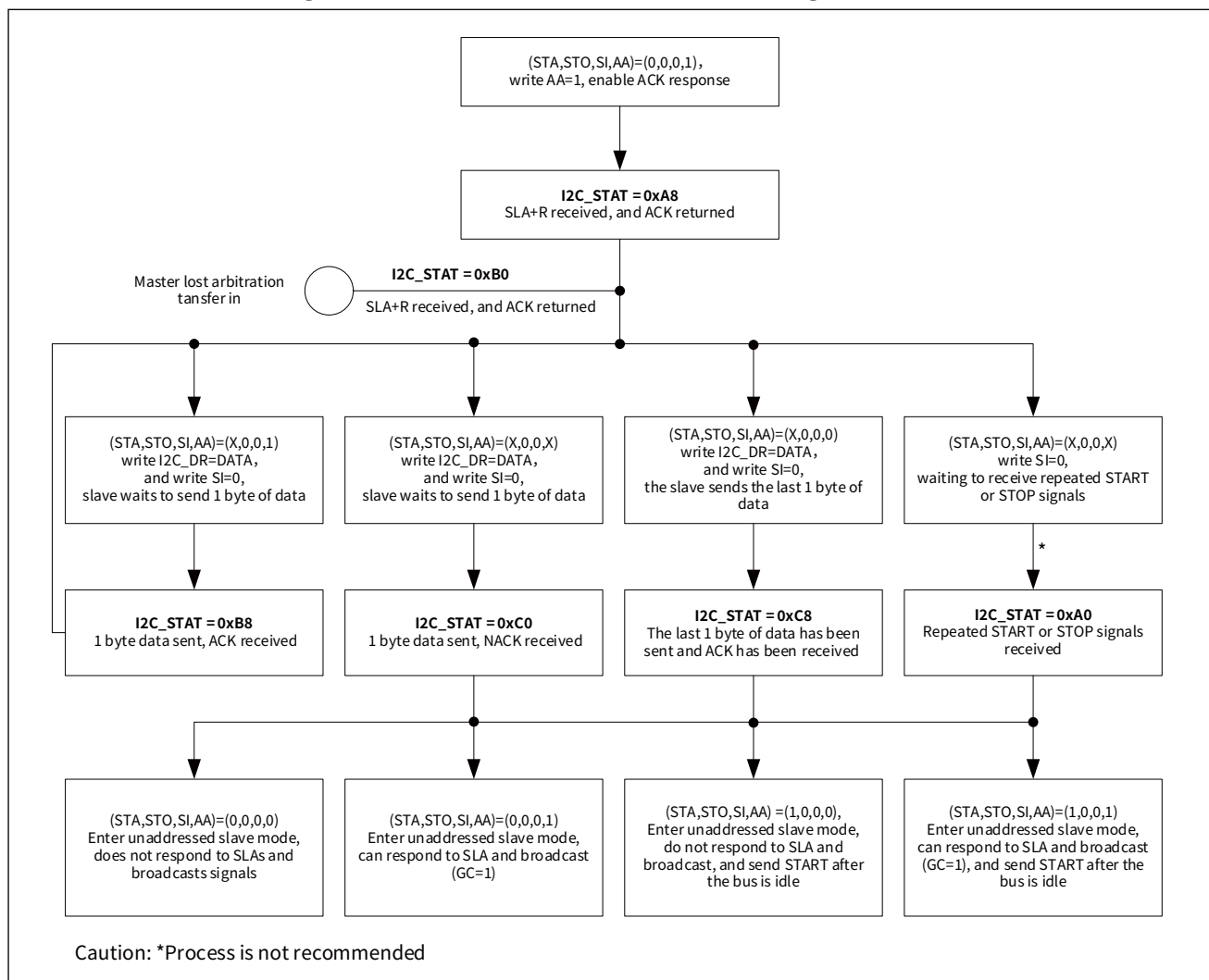
The state synchronization diagram in slave transmit mode is shown in the following figure:

Figure 18-14 The status synchronization diagram of slave transmit mode



The data transmission flow chart and status register value in slave transmit mode are shown in the following figure:

Figure 18-15 Slave transmit mode flow and register status



18.4.10.5 Broadcast receive mode

The broadcast receive mode is a special slave receive mode. When the slave is in an idle state (the slave receive mode is not addressed), the broadcast receive mode is entered when the SLA+W transmitted by the master is received, and the SLA is broadcast address 0x00. In this mode, the data reception process is the same as the slave receive mode, but the I2C bus status code is different.

To receive the broadcast information transmitted by the master, the slave needs to set I2C_CR.AA to 1 and I2C_ADDR0.GC to 1 to respond to the master's broadcast addressing and broadcast data; the three slave address registers I2C_ADDR0/1/2 do not need to be set; The value of I2C_BRR is invalid and does not need to be set.

After completing the above initialization work, the slave machine enters the idle state (the slave receive mode is not addressed), waiting to be addressed by the write signal (SLA+W) transmitted by the master. When the slave receives SLA+W, if the address matches the broadcast address 0x00, the slave responds with an ACK response and enters the broadcast receive mode, the status code I2C_STAT becomes 0x70, and the interrupt flag bit I2C_CR.SI is set to 1 at the same time. At this time, the I2C_CR.SI bit must be cleared in time to receive the data transmitted by the master from the bus.

Every time the slave receives 1 byte of data, it must respond with an ACK response. After the application reads the byte of data, it must clear the I2C_CR.SI bit to prepare for receiving the next byte of data.

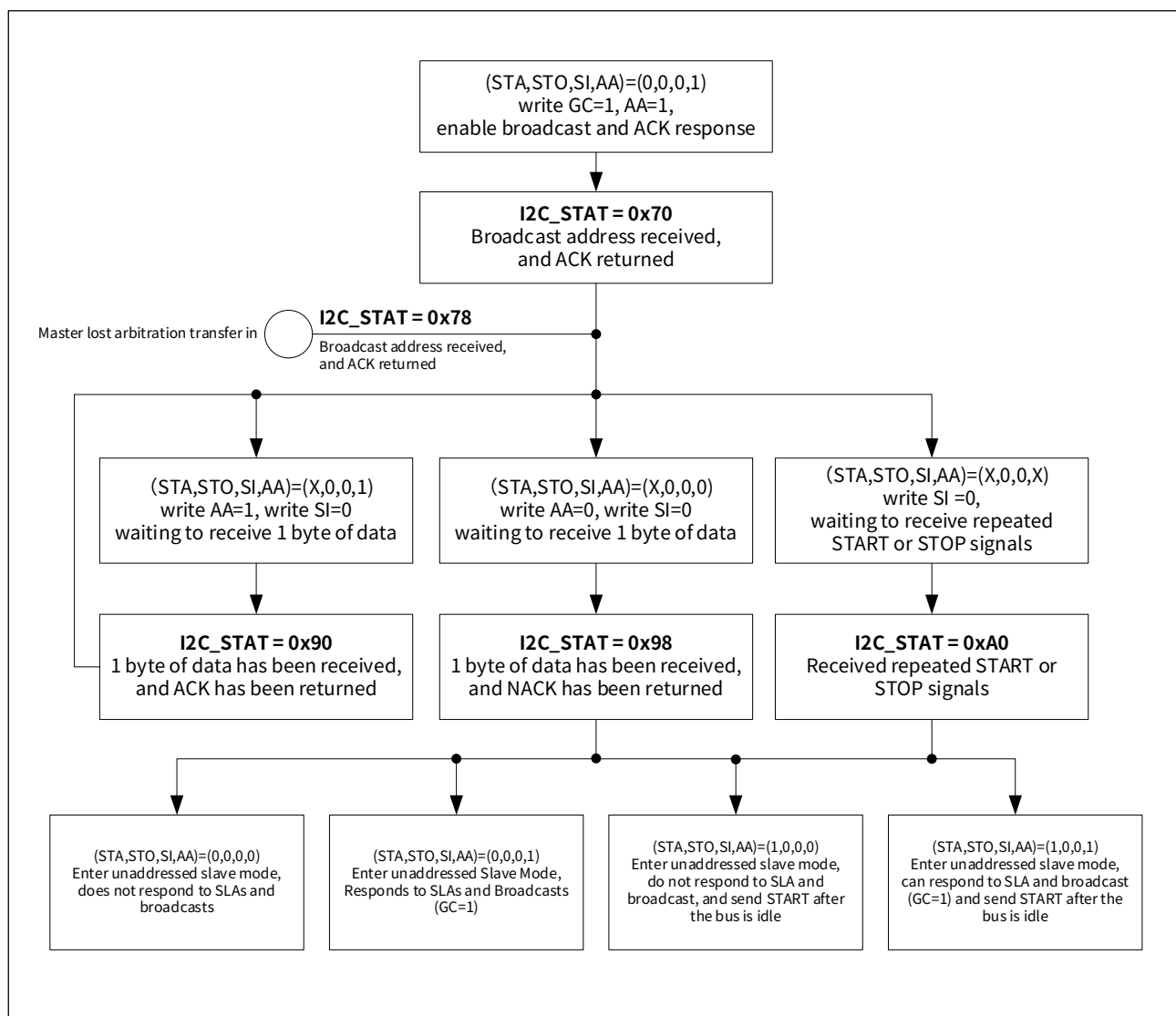
In the process of receiving data from the slave, if I2C_CR.AA is cleared, the slave will return the NACK signal when it receives the next byte, and the state of the slave will also switch to the unaddressed slave receive mode, ending communication with the master, no more data is received, and the I2C_DR register holds the previously received data. It can be seen from this feature that the slave application can actively switch from the addressed broadcast reception mode to the unaddressed slave receive mode by setting I2C_CR.AA to 0.

When the master loses arbitration due to bus conflict in the SLA+ read and write stage, it will enter the unaddressed slave receive mode, and then if it receives SLA+W that conforms to the broadcast address and responds with ACK (status code I2C_STAT=0x78), it will enter the addressed broadcast receive mode.

The data reception flow chart and status register value in broadcast receive mode are shown in the following figure:



Figure 18-16 Broadcast receive mode flow and register status



18.4.11 Multi-master communication

In some applications, there are two or more masters accessing the slave at the same time on one I2C bus, and it is possible to transmit data at the same time. At this time, there will be data conflicts on the SDA bus.

The I2C of the CW32L010 can perform data conflict detection and arbitration on the SDA bus to achieve multi-master applications. If two masters transmit data at the same time, the master that detects the conflict will lose arbitration and enter the unaddressed slave mode, and the master that has not detected the conflict will win the arbitration and continue to dominate the data communication flow.

18.4.12 I2C status codes

The I2C bus status is identified by the I2C status register I2C_STAT, with a total of 26 normal receiving or transmitting statuses, and 2 special statuses (0xF8: no information available on the I2C bus; 0x00: bus error).

Whether I2C is in master transmit, master receive, slave receive, slave transmit or broadcast receive mode, when the content of the status register I2C_STAT changes, I2C_CR.SI will be set and an I2C interrupt will be generated.

The I2C status codes are shown in the following table:

Table 18-4 I2C status codes

Operating mode	Status code	Meaning
Master transmit mode	08H	Start signal has been transmitted
	10H	Repeated start signal has been transmitted
	18H	SLA+W has been transmitted, ACK has been received
	20H	SLA+W has been transmitted, NACK has been received
	28H	Data in I2C_DR has been transmitted, ACK has been received
	30H	Data in I2C_DR has been transmitted, NACK has been received
	38H	The master loses arbitration during sending SLA+W phase or during transmitting data phase
Master receive mode	08H	Start signal has been transmitted
	10H	Repeated start signal has been transmitted
	38H	Master loses arbitration during transmitting SLA+R phase or responding to NACK phase
	40H	SLA+R has been transmitted, ACK has been received
	48H	SLA+R has been transmitted, NACK has been received
	50H	Data bytes have been received, ACK has been returned
	58H	Data bytes have been received, NACK has been returned

Operating mode	Status code	Meaning
Slave receive mode	60H	Its own SLA+W has been received, ACK has been returned
	68H	When the master loses arbitration in the SLA+read-write phase, its own SLA+W has been received, and ACK has been returned
	80H	The previous addressing used its own slave address, the data byte has been received, and ACK has been returned
	88H	The previous addressing used its own slave address, the data byte has been received, and NACK has been returned
	A0H	STOP or Repeated START condition is received while the addressed slave is waiting to receive data
Slave transmit mode	A8H	Its own SLA+R has been received, ACK has been returned
	B0H	When the master loses arbitration in the SLA+read-write phase, its own SLA+R has been received, and ACK has been returned
	B8H	Data bytes have been transmitted, ACK has been received
	C0H	Data bytes have been transmitted, NACK has been received
	C8H	Slave last data byte has been transmitted and ACK has been received
Broadcast receive mode	70H	The broadcast address (0x00) has been received, ACK has been returned
	78H	When the master loses arbitration in the SLA + read and write phase, the broadcast address has been received, and ACK has been returned
	90H	The previous addressing used the broadcast address, the data byte has been received, and the ACK has been returned
	98H	The previous addressing used the broadcast address, the data byte has been received, and NACK has been returned
	A0H	STOP or Repeated START condition is received while the addressed slave is waiting to receive data
Others	F8H	No relevant status information available, I2C_CR.SI=0
	00H	A bus error occurred during the transfer, or external interference caused the I2C to enter an undefined state

The special status code F8H indicates that there is no useful information at the current moment, and the current bus status cannot be determined, because I2C_CR.SI has not been set, and no interrupt is generated. This condition occurs before other states and the I2C module has started to perform a serial transfer.

The special status code 00H indicates that there is a bus error during the I2C serial transmission, such as the START or STOP signal appears in the wrong position of the data frame (including the address byte, data byte or response bit in the serial transmission process) or when external interference affects the internal I2C module signal, etc. When a bus error occurs, the I2C_CR.SI flag is immediately set, and the device is immediately switched to unaddressed slave receive mode, releasing SDA and SCL, and clearing the I2C_DR register.



When it is detected that the STAT of the I2C bus is a bus error (status code is 00H), since the I2C bus is continuously enabled, and the error has not been cleared for the I2C module, SI will continue to remain 1, that is, SI cannot be cleared.

Bus error clearing method:

1. Transmit a STOP signal to the bus: set the STO bit and clear the SI bit (due to the current bus error state, the controller does not actually transmit the STOP signal to the bus), the STO bit will be automatically cleared by hardware, releasing the bus to normal idle state.
2. If the SI bit cannot be cleared by setting the STO bit, it means that the timing has been disrupted. It is necessary to set I2C_CR.EN to 0 and 1 in turn, that is, shut down and restart the I2C module, and then set SI to 0 to clear the SI bit.

In each operating mode, the I2C bus state transition diagram is shown in the following figure. Caution that when transitioning between the two normal states, when the action is not completed (such as in the process of transmitting SLA), that is, before entering a new state, the status code will appear a short-term transition state, 0xF8. The user does not care, and no interruption is generated.

Figure 18-17 The status transition diagram of master transmit/receive mode

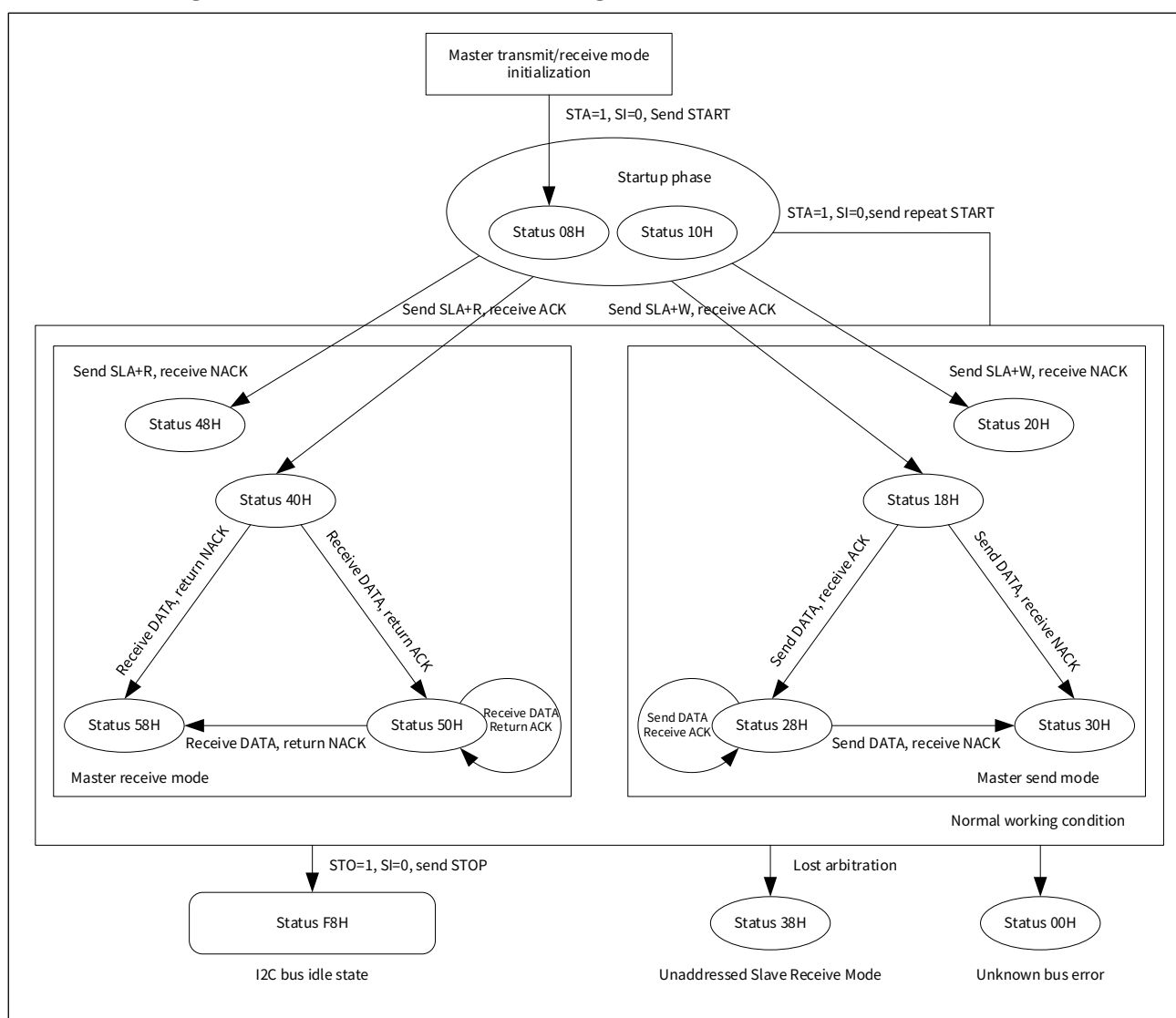


Figure 18-18 The status transition diagram of slave transmit/receive mode

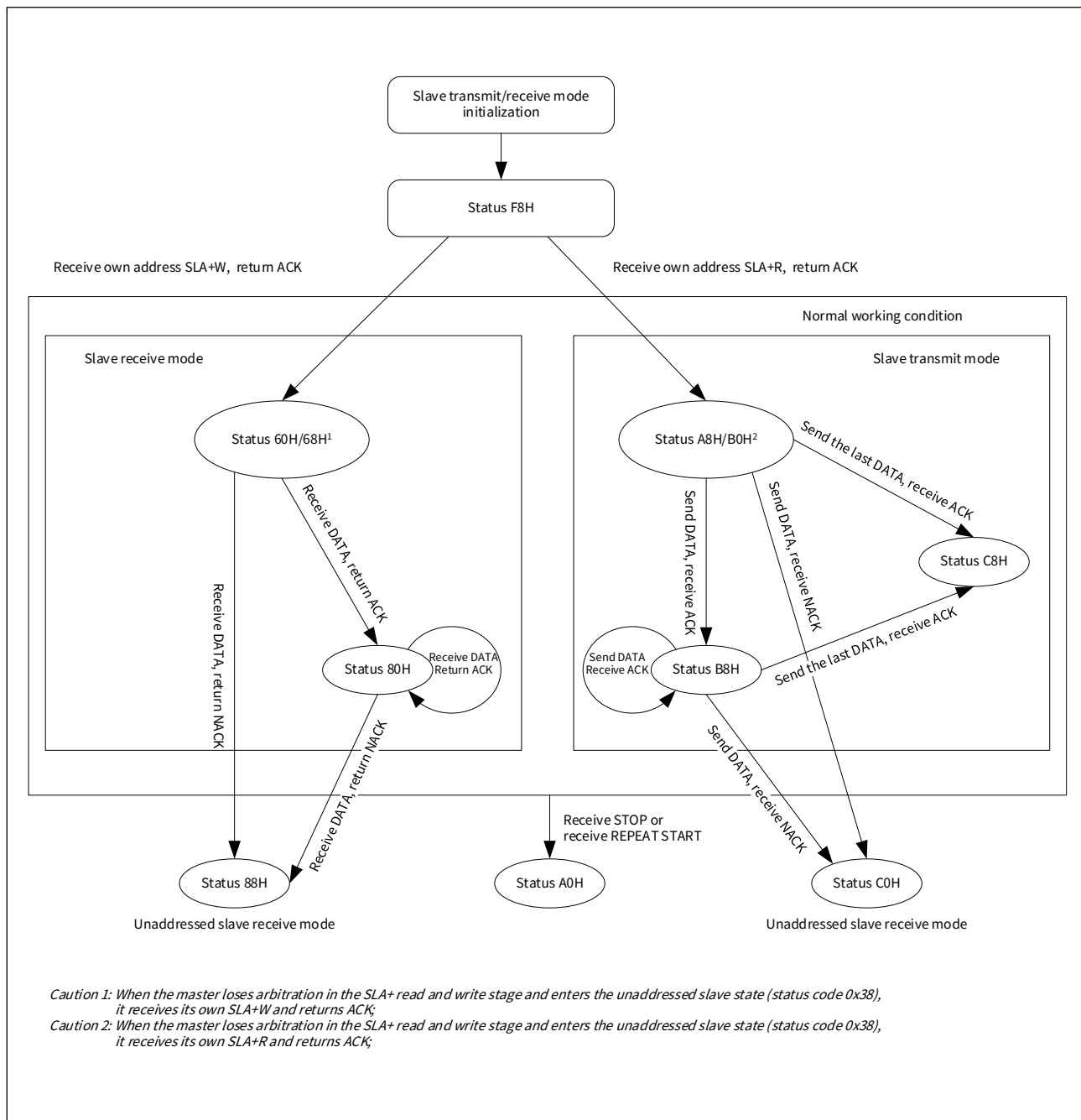
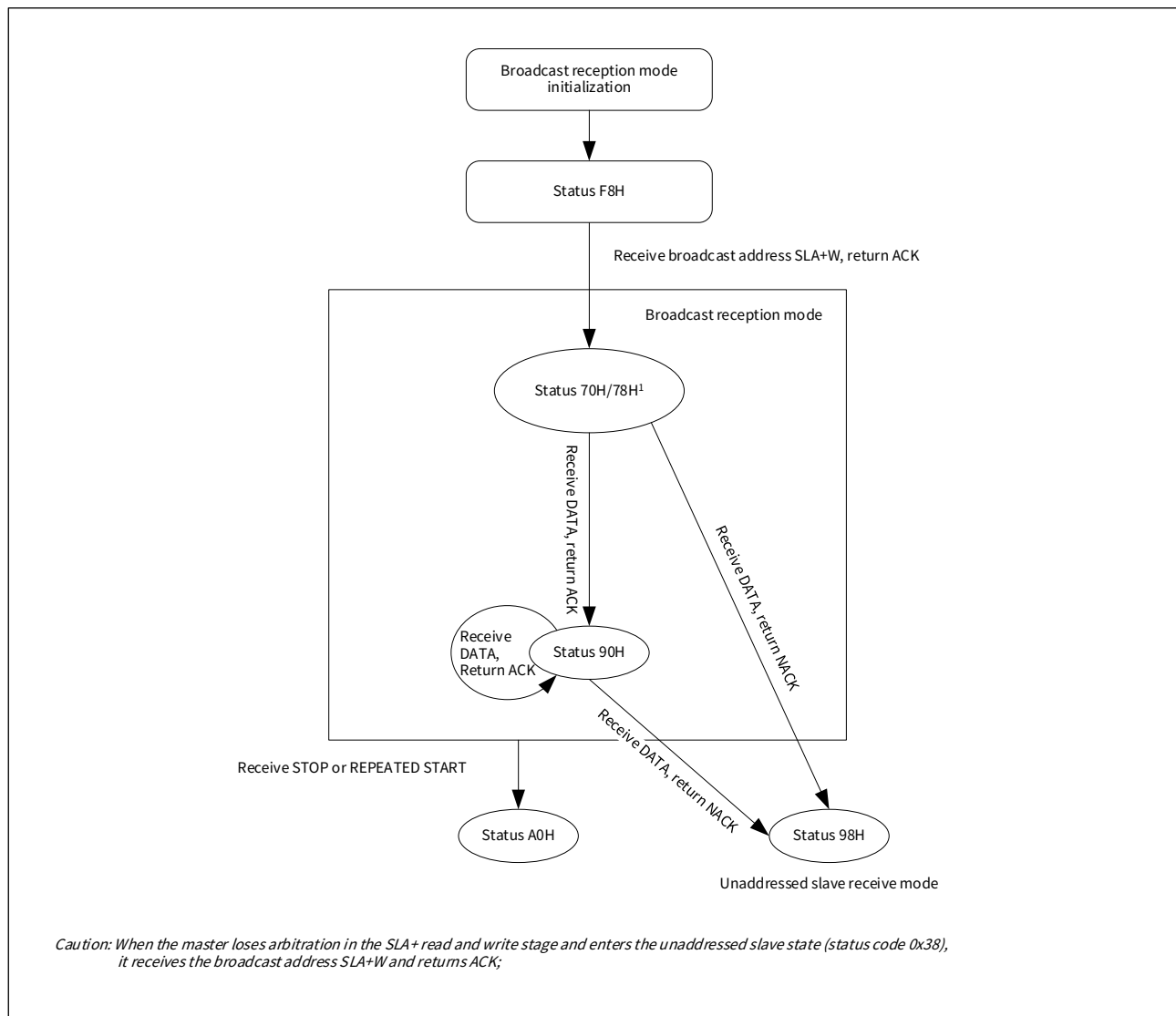


Figure 18-19 The status transition diagram of broadcast receive mode



18.5 Programming examples

18.5.1 Master transmits example

- Step 1: According to the relevant description of the pin digital multiplexed function in the GPIO chapter, map SCL and SDA to the required pins, and configure the SCL and SDA pins as open-drain output mode;
- Step 2: Set `SYSCTRL_APBEN2.I2C` to 1 to enable the I2C module clock;
- Step 3: Write 0 and 1 to `SYSCTRL_APBRS2.I2C` in turn to reset the I2C module;
- Step 4: Configure `I2C_BRR` to make the clock rate of SCL meet the application requirements;
- Step 5: Set `I2C_BRREN` to 1 to enable the SCL clock generator;
- Step 6: Set `I2C_CR.EN` to 1 to enable the I2C module;
- Step 7: Set `I2C_CR.STA` to 1, the bus tries to transmit a START signal;
- Step 8: Wait for `I2C_CR.SI` to become 1, START signal has been transmitted to the bus;
- Step 9: Query `I2C_STAT`, if the register value is 0x08 or 0x10, continue to the next step, otherwise, perform error processing;
- Step 10: Write `SLA+W` to `I2C_DR`, set `I2C_CR.STA` to 0, set `I2C_CR.SI` to 0, and transmit `SLA+W`;
- Step 11: Wait for `I2C_CR.SI` to become 1, `SLA+W` has been transmitted to the bus;
- Step 12: Query `I2C_STAT`, if the register value is 0x18, continue to the next step, otherwise, perform error processing;
- Step 13: Write the data to be transmitted to `I2C_DR`, set `I2C_CR.SI` to 0, and transmit the data;
- Step 14: Wait for `I2C_CR.SI` to become 1, the data has been transmitted to the bus;
- Step 15: Query `I2C_STAT`, if the register value is 0x28, continue to the next step, otherwise, perform error processing;
- Step 16: If the data to be transmitted is not completed, then jump to step 13 and continue to execute;
- Step 17: Set `I2C_CR.STO` to 1, set `I2C_CR.SI` to 0, transmit STOP signal, and end this data transfer.



18.5.2 Master receives example

- Step 1: According to the relevant description of the pin digital multiplexed function in the GPIO chapter, map SCL and SDA to the required pins, and configure the SCL and SDA pins as open-drain output mode;
- Step 2: Set `SYSCTRL_APBEN2.I2C` to 1 to enable the I2C module clock;
- Step 3: Write 0 and 1 to `SYSCTRL_APBRS2.I2C` in turn to reset the I2C module;
- Step 4: Configure `I2C_BRR` to make the clock rate of SCL meet the application requirements;
- Step 5: Set `I2C_BRREN` to 1 to enable the SCL clock generator;
- Step 6: Set `I2C_CR.EN` to 1 to enable the I2C module;
- Step 7: Set `I2C_CR.STA` to 1, the bus tries to transmit a START signal;
- Step 8: Wait for `I2C_CR.SI` to become 1, START signal has been transmitted to the bus;
- Step 9: Query `I2C_STAT`, if the register value is 0x08 or 0x10, continue to the next step, otherwise, perform error processing;
- Step 10: Write `SLA+R` to `I2C_DR`, set `I2C_CR.STA` to 0, set `I2C_CR.SI` to 0, and transmit `SLA+R`;
- Step 11: Wait for `I2C_CR.SI` to become 1, `SLA+R` has been transmitted to the bus;
- Step 12: Query `I2C_STAT`, if the register value is 0x40 (ACK has been received), continue to the next step, otherwise, perform error processing;
- Step 13: Set `I2C_CR.AA` to 1, enable the response flag;
- Step 14: Set `I2C_CR.SI` to 0, waiting to receive 1 byte of data (the master transmits the clock, and the slave transmits data under the action of the clock);
- Step 15: Wait for `I2C_CR.SI` to become 1 (the master has completed the 1-byte data reception and has responded to the ACK signal), and read the received data from `I2C_DR`;
- Step 16: Query `I2C_STAT`, if the register value is 0x50 or 0x58, continue to the next step, otherwise, perform error processing;
- Step 17: If the data to be received is only missing the last byte, set `I2C_CR.AA` to 0, and enable the non-response flag;
- Step 18: If the data to be received is not completed, then jump to step 14 and continue to execute;
- Step 19: Set `I2C_CR.STO` to 1, set `I2C_CR.SI` to 0, transmit STOP signal, and end this data transfer.



18.5.3 Slave receives example

- Step 1: According to the relevant description of the pin digital multiplexed function in the GPIO chapter, map SCL and SDA to the required pins, and configure the SCL and SDA pins as open-drain output mode;
- Step 2: Set `SYSCTRL_APBEN2.I2C` to 1 to enable the I2C module clock;
- Step 3: Write 0 and 1 to `SYSCTRL_APBRS2.I2C` in turn to reset the I2C module;
- Step 4: Set `I2C_CR.EN` to 1 to enable the I2C module;
- Step 5: Configure `I2C_ADDR0` as the slave address;
- Step 6: Set `I2C_CR.AA` to 1, enable the response flag;
- Step 7: Wait for `I2C_CR.SI` to become 1 and be addressed by `SLA+W`;
- Step 8: Query `I2C_STAT`, if the register value is 0x60, continue to the next step, otherwise, perform error processing;
- Step 9: Set `I2C_CR.SI` to 0, wait for the master to transmit data, and respond to the ACK signal;
- Step 10: Wait for `I2C_CR.SI` to become 1, and read the received data from `I2C_DR`;
- Step 11: Query `I2C_STAT`, if the register value is 0x80, continue to the next step, otherwise, perform error processing;
- Step 12: If the data to be received is not completed, then jump to step 9 and continue to execute;
- Step 13: Set `I2C_CR.AA` to 0, set `I2C_CR.SI` to 0, the slave switches to the unaddressed slave receive mode, and does not respond to master addressing.

18.5.4 Slave transmits example

- Step 1: According to the relevant description of the pin digital multiplexed function in the GPIO chapter, map SCL and SDA to the required pins, and configure the SCL and SDA pins as open-drain output mode;
- Step 2: Set `SYSCTRL_APBEN2.I2C` to 1 to enable the I2C module clock;
- Step 3: Write 0 and 1 to `SYSCTRL_APBRS2.I2C` in turn to reset the I2C module;
- Step 4: Set `I2C_CR.EN` to 1 to enable the I2C module;
- Step 5: Configure `I2C_ADDR0` as the slave address;
- Step 6: Set `I2C_CR.AA` to 1, enable the response flag;
- Step 7: Wait for `I2C_CR.SI` to become 1 and be addressed by `SLA+R`;
- Step 8: Query `I2C_STAT`, if the value of this register is 0xA8, continue to the next step, otherwise, perform error processing;
- Step 9: Write the data to be transmitted to `I2C_DR`, set `I2C_CR.SI` to 0, and prepare to transmit data;
- Step 10: Wait for `I2C_CR.SI` to become 1, indicating that the data has been transmitted to the bus and received an ACK or NACK response;
- Step 11: Query `I2C_STAT`, if the value of this register is 0xB8, continue to the next step, otherwise, perform error processing;
- Step 12: If the data to be transmitted is not completed, then jump to step 9 to continue execution;
- Step 13: Set `I2C_CR.AA` to 0, set `I2C_CR.SI` to 0, the slave switches to the unaddressed slave receive mode, and does not respond to master addressing.



18.6 List of registers

I2C base address: I2C_BASE = 0x4000 5800

Table 18-5 List of I2C registers

Register name	Register address	Register description
I2C_BRREN	I2C_BASE + 0x00	Baud rate counter enable register
I2C_BRR	I2C_BASE + 0x04	Baud rate counter configuration register
I2C_CR	I2C_BASE + 0x08	Control register
I2C_DR	I2C_BASE + 0x0C	Data register
I2C_ADDR0	I2C_BASE + 0x10	Slave address 0 register
I2C_STAT	I2C_BASE + 0x14	Status register
I2C_ADDR1	I2C_BASE + 0x20	Slave address 1 register
I2C_ADDR2	I2C_BASE + 0x24	Slave address 2 register
I2C_MATCH	I2C_BASE + 0x28	Slave address match flag register



18.7 Register descriptions

See section [1 Documentation conventions](#) for abbreviations used in register descriptions.

18.7.1 I2C_BRREN Baud rate counter enable register

Address offset: 0x00 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:1	RFU	-	Reserved bits, please keep the default value
0	EN	RW	I2C bus SCL baud rate counter enable control 0: Disabled 1: Enabled Caution: EN should be enabled when acting as the master, and this bit has no effect when acting as the slave.

18.7.2 I2C_BRR Baud rate counter configuration register

Address offset: 0x04 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:8	RFU	-	Reserved bits, please keep the default value
7:0	BRR	RW	I2C bus SCL baud rate configuration $f_{SCL} = f_{PCLK} / 8 / (BRR+1)$, where $BRR > 0$



18.7.3 I2C_CR Control register

Address offset: 0x08 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:14	RFU	-	Reserved bits, please keep the default value
13:11	SDAINSRC	RW	SDA input signal source configuration 000: I2C_SDA pin 001: VC1_OUT 010: VC2_OUT Others: Reserved
10:8	SCLINSRC	RW	SCL input signal source configuration 000: I2C_SCL pin 001: VC1_OUT 010: VC2_OUT Others: Reserved
7	RFU	-	Reserved bits, please keep the default value
6	EN	RW	Module enable control 0: Disabled 1: Enabled
5	STA	RW	Bus status control W0: Clear STA W1: Send START signal to the bus Caution 1: After setting STA to 1, if the bus is idle, send the START signal, if the bus is busy, wait for the I2C to stop, and then send the START signal. Caution 2: If the device is already in master mode and has sent one or more bytes, and then set STA at this time, the I2C bus will generate a Repeated START signal. Caution 3: STA can be set at any time, including slave mode. However, the hardware will not automatically clear 0 after the completion of the START or repeat START signal transmission, and the user needs to manually clear STA.
4	STO	RW	Bus status control W0: No function W1: Send STOP signal to bus Caution 1: The hardware will automatically clear STO to 0 after the completion of the STOP signal transmission. Caution 2: If STA and STO are set at the same time in master mode, the I2C bus sends START immediately after sending STOP. In slave mode, it is forbidden to set STA and STO at the same time to avoid sending illegal I2C frames. Caution 3: STO is also set to 1 when an error state occurs on the bus (STAT status word is 00H), but the I2C bus will not send a STOP signal in this case.



Bit field	Name	Permission	Function description
3	SI	RW	I2C interrupt flags R0: No I2C interrupt occurred R1: I2C interrupt has occurred W0: Clear the I2C interrupt flag and cause the status machine to execute the next action W1: No function Caution 1: If one of the 26 states of I2C occurs, the hardware will set this bit (except F8H) to 1. At this time, the software confirms the current status of the bus through the I2C_STAT register value. Caution 2: SI needs to be cleared by software. Caution 3: Before SI is cleared to 0, the SCL low level period is extended and the transmission is suspended. This status is very useful for the slave to process the received data, which can ensure that the previous data is accurately processed before the next data is received. Caution 4: Before software clears SI, software should prepare appropriate register settings. After SI is cleared, the I2C bus will perform the corresponding operation according to the register settings.
2	AA	RW	Response control 0: Transmit NACK in response phase 1: Transmit ACK in response phase Caution 1: For the addressed slave, if no ACK response bit is returned in slave receive mode or no ACK response bit is received in slave transmit mode, the slave will switch to unaddressed slave receive mode, cannot receive data until its AA is set to 1 and re-addressed by the master. Caution 2: Special case: in slave transmit mode, before the slave transmits the last byte to the master, clear AA. After transmitting the last byte, the slave will switch to the unaddressed slave mode and be disconnected from the master, the status register I2C_STAT is C8H. If the master reads data from the bus again, it will get 0xFF.
1	RFU	-	Reserved bits, please keep the default value
0	FLT	RW	I2C filter parameter configuration 0: Advanced filtering, higher anti-interference performance 1: Simple filtering, faster communication rate Caution: See section 18.4.3 Input filter.



18.7.4 I2C_DR Data register

Address offset: 0x0C Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:8	RFU	-	Reserved bits, please keep the default value
7:0	DR	RW	Data register In transmit mode, write the transmitted data In receive mode, read the received data

18.7.5 I2C_STAT status register

Address offset: 0x14 Reset value: 0x0000 00F8

Bit field	Name	Permission	Function description
31:8	RFU	-	Reserved bits, please keep the default value
7:0	STAT	RO	I2C status register, see section 18.4.12 I2C status codes for the specific definition of status value; when STAT = F8H, it means that there is no relevant status information available, and SI will remain at 0. For the other 26 states, SI will be set to 1 and an interrupt request will be generated.

18.7.6 I2C_ADDR0 Slave address 0 register

Address offset: 0x10 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:8	RFU	-	Reserved bits, please keep the default value
7:1	ADDR0	RW	Slave mode address 0 Caution 1: The master mode is invalid. Caution 2: The master needs to address the slave, and the address information must be the same as this address through the first byte value address information after START or Repeated START. If AA is 1, the slave responds to the master and becomes the addressed slave, otherwise the master broadcast addressing information will be ignored. Caution 3: I2C_ADDR0[7:1] cannot be written as all 0, because 0x00 is reserved for broadcast addressing.
0	GC	RW	Broadcast address response enabled 0: Disabled 1: Enabled Caution 1: The master mode is invalid. Caution 2: After enabling GC, if AA is set to 1, the broadcast receive mode is enabled, and if AA is cleared to 0, the broadcast addressing information on the bus is ignored.



18.7.7 I2C_ADDR1 Slave address 1 register

Address offset: 0x20 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:8	RFU	-	Reserved bits, please keep the default value
7:1	ADDR1	RW	Slave mode address 1 Caution: The master mode is invalid.
0	RFU	-	Reserved bits, please keep the default value

18.7.8 I2C_ADDR2 Slave address 2 register

Address offset: 0x24 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:8	RFU	-	Reserved bits, please keep the default value
7:1	ADDR2	RW	Slave mode address 2 Caution: The master mode is invalid.
0	RFU	-	Reserved bits, please keep the default value



18.7.9 I2C_MATCH Slave address match register

Address offset: 0x28 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:3	RFU	-	Reserved bits, please keep the default value
2	ADDR2	RO	I2C slave mode address 2 match flag bit 0: The device address received from the bus is not the same as ADDR2 1: The device address received from the bus is the same as ADDR2
1	ADDR1	RO	I2C slave mode address 1 match flag bit 0: The device address received from the bus is not the same as ADDR1 1: The device address received from the bus is the same as ADDR1
0	ADDR0	RO	I2C slave mode address 0 match flag bit 0: The device address received from the bus is not the same as ADDR0 1: The device address received from the bus is the same as ADDR0

Caution:*The address match flag bit will be cleared in the following three cases:*

- When the module is reset;
- When START/STOP is transmitted.



19 Infrared modulation transmitter (IR)

19.1 Overview

The CW32L010 integrates an internal infrared modulated transmitter (IR), which can be used in conjunction with the timer, UART and IRSW soft control bits to easily realize various standard PWM or PPM encoding methods, as well as infrared modulated transmission of UART data.

19.2 Main features

- Supports SIR of IrDA standard 1.0
- Maximum data rate 115.2kbps
- Can adapt to high and low level infrared emission tube



19.3 Functional description

When implementing an infrared modulation transmitter, a timer channel is used to generate a square wave signal with a fixed frequency, and another timer or UART is used to generate modulated data, which is output from the IR_OUT pin after 'AND' or 'OR' operation with IRSW soft control bits. For specific IR_OUT pins, please refer to the pin definition in the data manual.

The MOD bit field of the IR modulation control register IRMOD_CR is used to select the source of the carrier signal and the data signal, and the 'AND' 'OR' operation with the IRSW soft control bit. The selection of the 'AND' 'OR' is determined by the drive level of the user's hardware IR emitter. The IR_OUT output can be controlled inversely through the INV bit field.

The carrier signal frequency can be set by the user, the most common carrier frequency is 38KHz.

In order to reduce the transmission power consumption, the duty cycle of the carrier is generally set to about 20%, and the timer that generates the carrier can use the PWM mode, please refer to chapter [13 General-purpose timer \(GTIM\)](#). In order to increase the transmission distance, the duty cycle can be appropriately increased. In the IrDA standard 1.0, in order to achieve a fast IR communication rate, the width of the pulse is specified as 3/16 of a bit period or a fixed 1.63μs, and the minimum cannot be lower than 1.41μs.

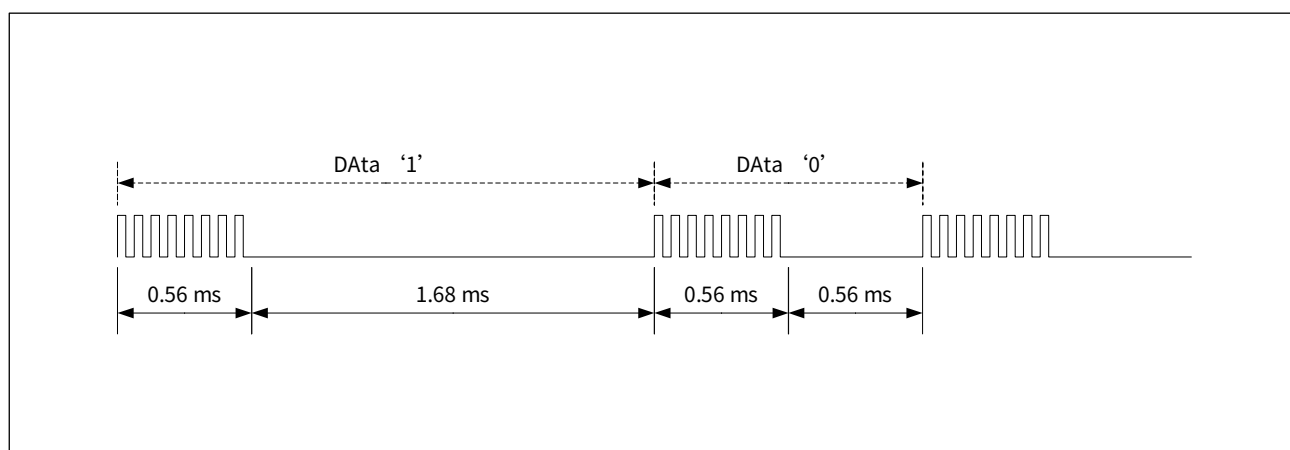


19.3.1 Infrared modulation mode

There are two common IR infrared encoding methods, pulse width modulation PWM and pulse position modulation PPM.

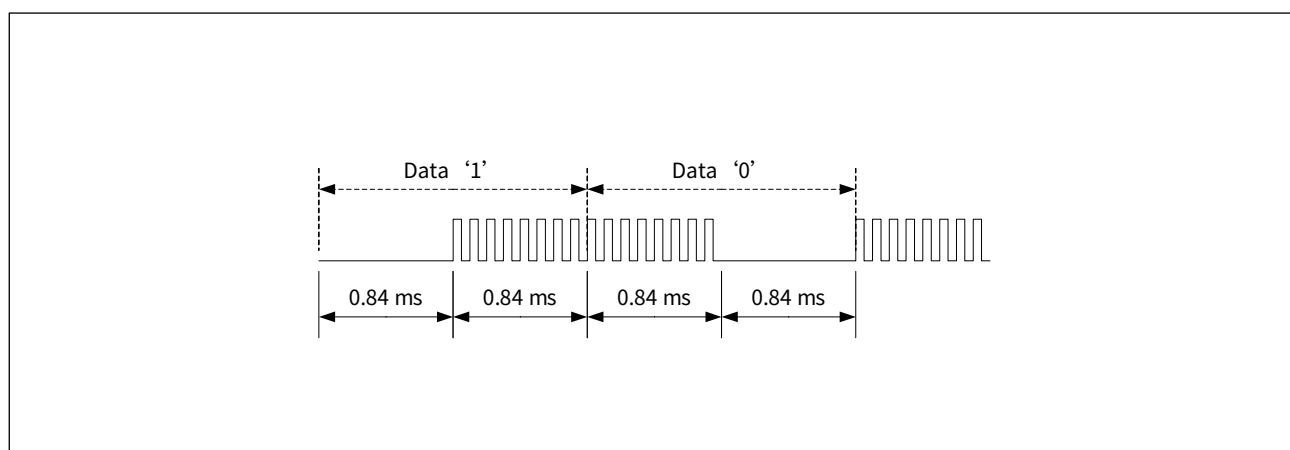
PWM pulse width modulation to represent '0' and '1' with the duty cycle of the transmitted infrared carrier. For example, UPD6121, the carrier transmits 0.56ms, and does not transmit 0.56ms, which means '0'; the carrier transmits 0.56ms, and does not transmit 1.68ms, which means '1'. The reference diagram is shown in the following figure:

Figure 19-1 PWM modulation mode



PPM pulse position modulation to represent '0' and '1' with the position of the transmitted carrier. It is '0' from transmitting carrier to not transmitting carrier, and '1' from never transmitting carrier to transmitting carrier. For example, SAA3010, the carrier transmits 0.84ms, and does not transmit 0.84ms, which means '0'; does not transmit 0.84ms, and the carrier transmits 0.84ms, means '1'. The reference diagram is shown in the following figure:

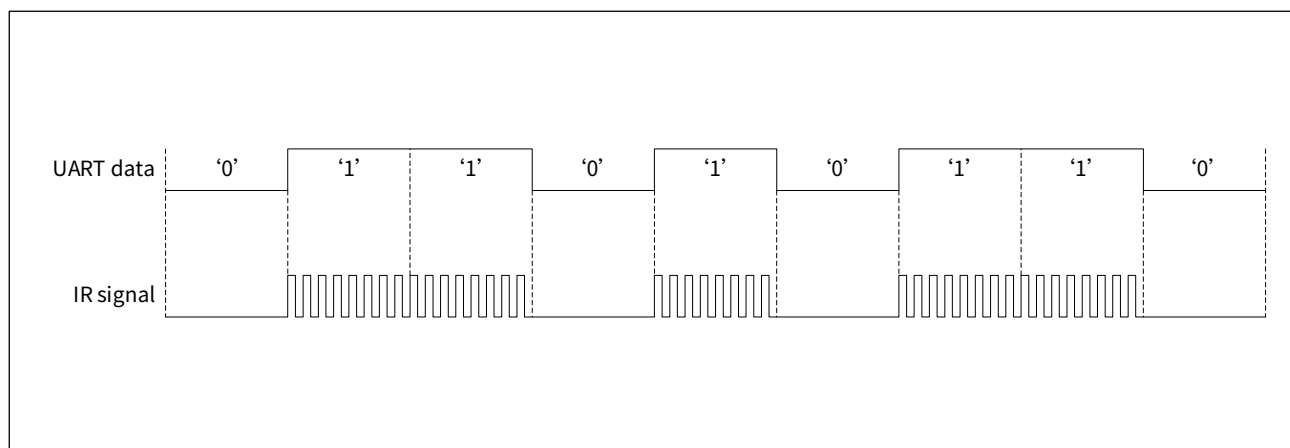
Figure 19-2 PPM modulation mode



UART modulation is a relatively simple infrared modulation method, which directly performs 'AND' or 'OR' operation on serial data and carrier signal. When the UART serial baud rate is lower than the carrier frequency, the reference schematic diagram is shown in the following figure:

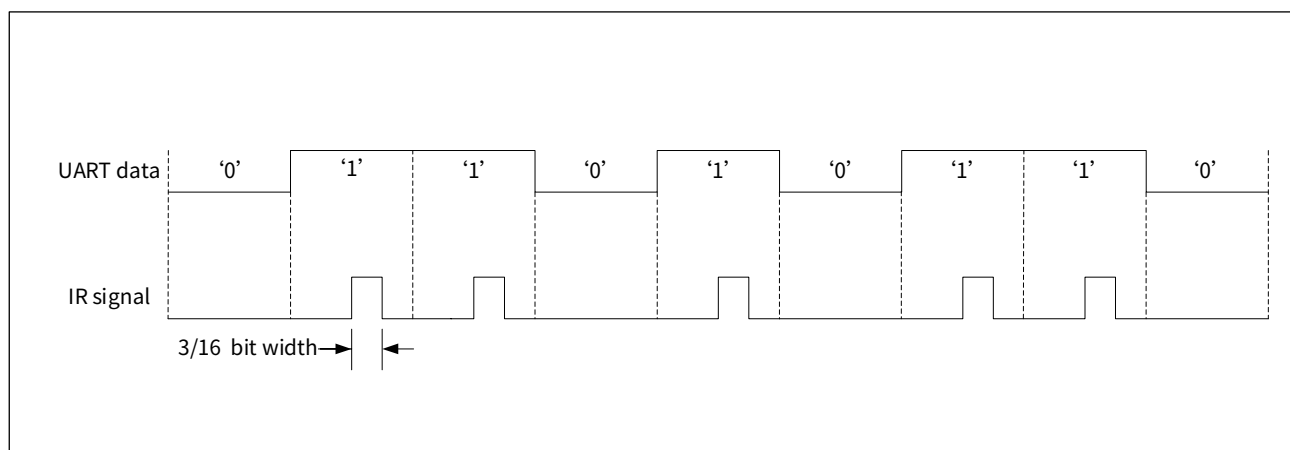


Figure 19-3 UART modulation mode



When the UART serial baud rate is higher than the carrier frequency, the carrier pulse width should be set to 3/16 of the bit width, and the carrier frequency should be kept consistent with the serial frequency. For example, when the UART baud rate is set to 115.2kbps, the carrier frequency is also set to 115.2kHz, and the width of the carrier pulse is $1.63\mu\text{s}$ ($3/16 \times 1/115200\text{Hz} = 1.63\mu\text{s}$). The reference schematic diagram is shown in the following figure:

Figure 19-4 High-speed UART modulation mode



For the convenience of debugging, usually a standard data frame also includes boot code, end bit, retransmission code, etc. Please refer to the relevant data manual by yourself.

19.3.2 Infrared modulation initialization configuration

The initialization process of a complete IR infrared modulation transmitter should include the configuration of timer, UART, and GPIO. The reference steps are as follows:

- Step 1: Set the SYSCTRL_APBEN1 and SYSCTRL_APBEN2 registers to enable the timer and UART peripheral clock to be selected;
- Step 2: Set the allowed bits of the GPIO clock corresponding to the IR_OUT pin in the SYSCTRL_AHBEN register.
- Step 3: Configure the timer selected as the carrier, set the carrier frequency, duty cycle (common frequency 38KHz, 1/3 duty cycle), and set the timer working mode (PWM output is recommended);
- Step 4: Configure the timer or UART selected as data, set the bit width, and UART working mode (no need to define output pins);
- Step 5: Set GPIOx_ANALOG as IR_OUT output port as digital mode, set GPIOx_DIR as output, and set appropriate GPIOx_OPENDRAIN, GPIOx_PUR;
- Step 6: Start the timer and UART as carrier and data;
- Step 7: Set the MOD bit field of the IR infrared modulation control register IRMOD_CR, and set the infrared modulation mode configuration;
- Step 8: Set the GPIOx_AFRH/GPIOx_AFRL of the IR_OUT output port to IR_OUT, and enable the IR multiplexing function.

19.3.3 Infrared reception

CW32L010 does not have an IR receiving demodulation module inside. In IR receiving applications, it is necessary to use an integrated infrared receiving head with demodulation function. With the capture function of GTIM (UART mode can directly use the RXD pin input), it can be easily realized IR receiving function.



19.4 IRMOD_CR Infrared modulation control register

Address: 0x4000 4000 + 0x80 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:6	RFU	-	Reserved bits, please keep the default value
5	INV	RW	IR output reverse control 0: Positive output 1: Reverse output
4	IRSW	RW	Infrared modulation soft control bit Functions detailed in MOD bit field
3:0	MOD	RW	Infrared modulation mode configuration 0000: GTIM_CH1 & ATIM_CH1 & IRSW 0001: GTIM_CH1 & ATIM_CH2 & IRSW 0010: GTIM_CH2 & ATIM_CH1 & IRSW 0011: GTIM_CH2 & ATIM_CH2 & IRSW 0100: GTIM_CH1 ATIM_CH1 IRSW 0101: GTIM_CH1 ATIM_CH2 IRSW 0110: GTIM_CH2 ATIM_CH1 IRSW 0111: GTIM_CH2 ATIM_CH2 IRSW 1000: UART1_TXD & GTIM_CH1 & IRSW 1001: UART1_TXD GTIM_CH1 IRSW 1010: UART1_TXD & GTIM_CH2 & IRSW 1011: UART1_TXD GTIM_CH2 IRSW 1100: UART2_TXD & ATIM_CH1 & IRSW 1101: UART2_TXD ATIM_CH1 IRSW 1110: UART2_TXD & ATIM_CH2 & IRSW 1111: UART2_TXD ATIM_CH2 IRSW



20 Analog-to-digital converter (ADC)

20.1 Overview

CW32L010 integrates a successive approximation analog-to-digital converter (SAR ADC) with a 12-bit precision and a maximum conversion speed of 2M SPS, which can convert up to 16 analog signals into digital signals. The vast majority of signals in the real world are analog, such as light, electricity, sound, image signals, etc., which must be converted into digital signals by ADC before they can be digitally processed by MCU.

20.2 Main features

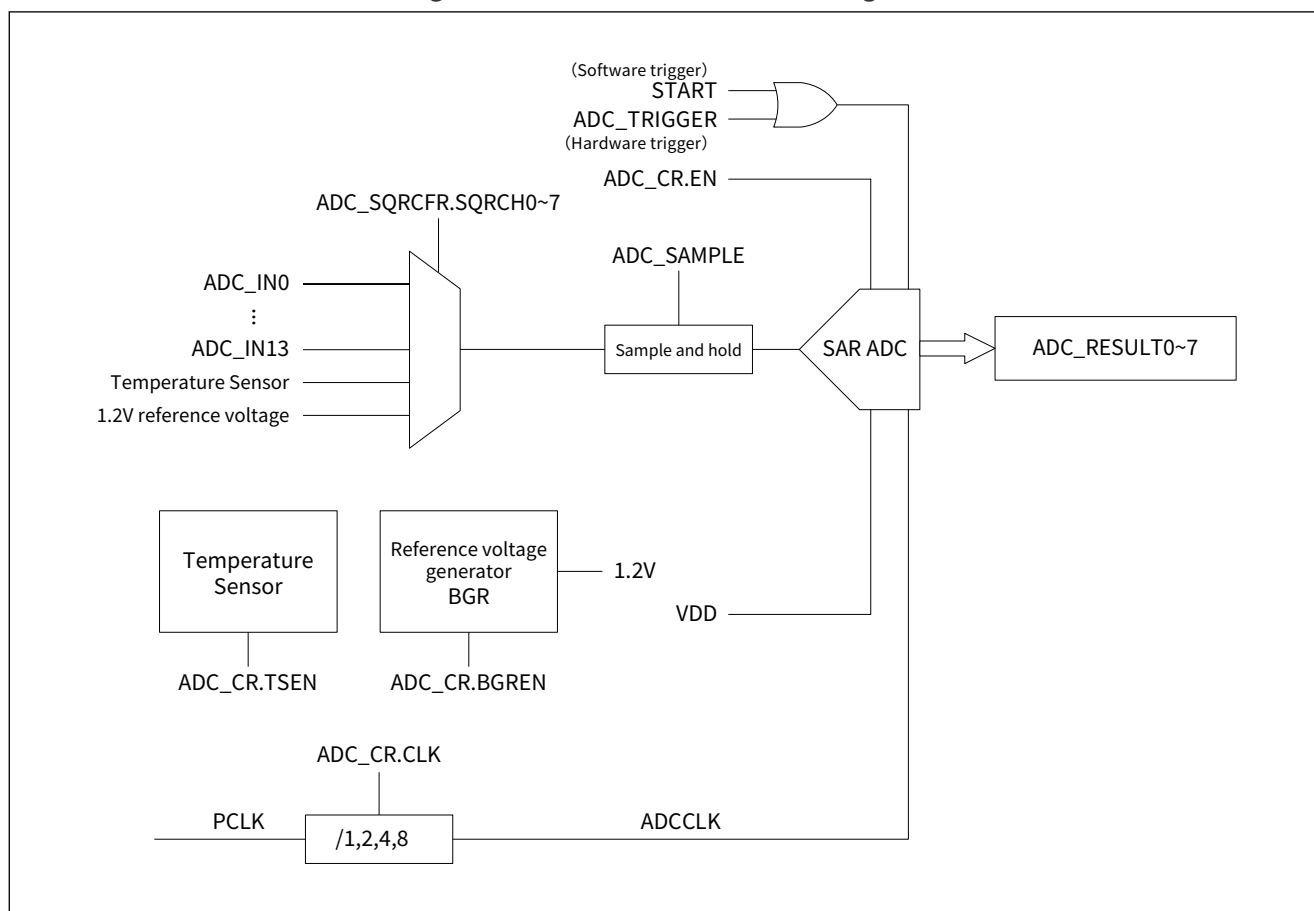
- 12-bit precision
- Individually configurable conversion speeds for each sequence channel, up to 2M SPS
- 16 channels of input conversion
 - 14 external pins input
 - Built-in temperature sensor
 - Built-in BGR 1.2V reference
- VDD supply voltage as reference voltage source (V_{ref})
- Sampling voltage input range: $0 \sim V_{ref}$
- Supports sequence channel conversion modes
 - Supports single and continuous conversions
 - Supports up to 8 conversion channels, each channel can select one of the 16 conversion sources
- Supports input channel voltage threshold monitoring
- Supports on-chip peripherals to automatically trigger ADC conversion



20.3 Functional block diagram

The ADC functional block diagram is shown in the following figure:

Figure 20-1 ADC functional block diagram

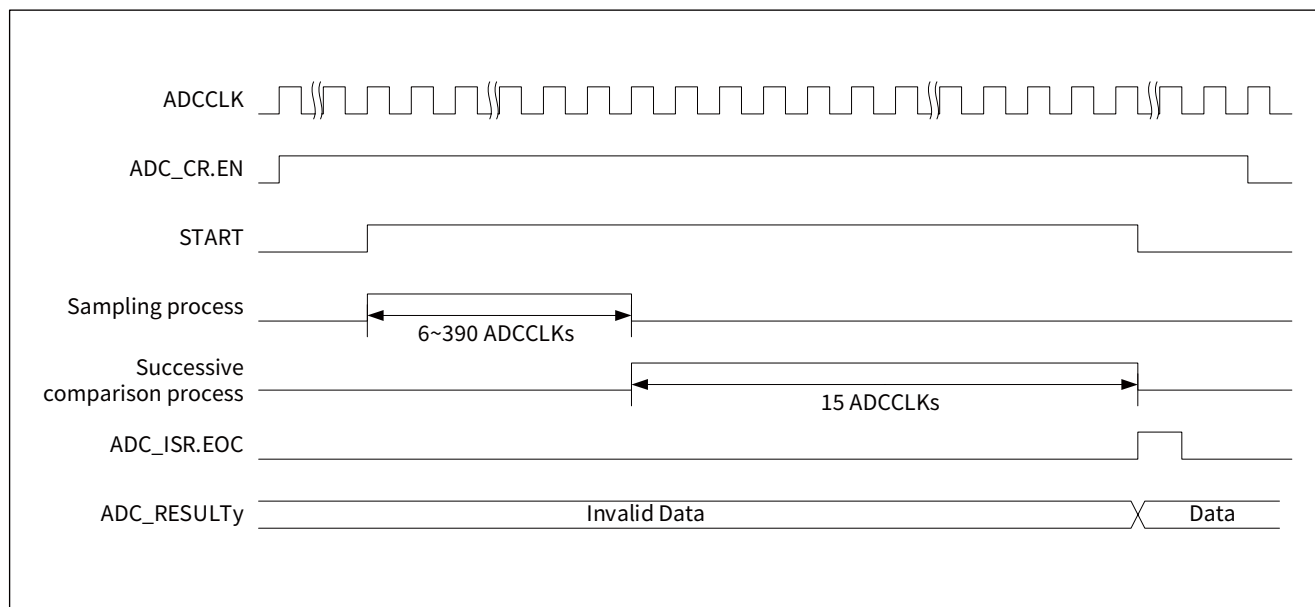


20.4 Conversion timing, conversion speed, conversion accuracy, and conversion results

20.4.1 Conversion timing

The conversion timing of the ADC is shown in the following figure:

Figure 20-2 ADC conversion timing diagram



Write 1 to the EN bit field of the ADC control register ADC_CR to enable the ADC module.

About 1μs after ADC_CR.EN changes from 0 to 1, the initialization of the analog circuit is completed, and the ADC conversion can be started.

Write 1 to the START bit field of the ADC start register ADC_START to start the ADC conversion, and the hardware will automatically clear it after the conversion is completed.

The ADC working clock ADCCLK is obtained by dividing the system clock PCLK by the prescaler. The CLK bit field of the control register ADC_CR can be selected to divide the frequency from 1/2/4/8, as shown in the following table:

Table 20-1 ADC clock configuration table

ADC_CR.CLK	ADCCLK
000	PCLK
001	PCLK/2
010	PCLK/4
011	PCLK/8

A complete ADC conversion requires 21~405 ADCCLK clock cycles, including sampling phase and successive comparison phase:

1. Sampling phase: 6 ~ 390 ADCCLK clock cycles are required. The sample period for each sequence conversion channel can be configured individually and is set through the SQRCHy bit field of the sample period configuration register ADC_SAMPLE as shown in the following table:

Table 20-2 ADC sampling period of each sequence channel

ADC_CR0.SAM	Fixed sampling period (number of ADCCLKs)
0000	6
0001	7
0010	9
0011	12
0100	18
0101	24
0110	30
0111	42
1000	54
1001	70
1010	102
1011	134
1100	166
1101	198
1110	262
1111	390

The sampling moment and sampling period of ADC can be output through the external ADC_SAM pin, please refer to the Datasheet Pin Definition for the specific pin.

ADC sampling period length is determined by the user's speed requirement for sampling and the electrical characteristics of the sampled signal, the user should choose the appropriate sampling period to achieve the best conversion effect.

2. Successive comparison phase: 15 ADCCLK clock cycles are required.

After each ADC conversion is completed, the conversion completion flag bit ADC_ISR.EOC will be set to 1 by hardware, and the ADC conversion result is stored in the corresponding sequence conversion result register ADC_RESULTy (y=0, 1, 2, 3, 4, 5, 6, 7), the user can clear this flag bit by setting ADC_ICR.EOC to 0.



20.4.2 Conversion speed

ADC conversion speed is closely related to VDD supply voltage, and the maximum conversion speed under various conditions is shown in the following table:

Table 20-3 ADC conversion speed and voltage comparison table

VDD voltage	Maximum conversion speed	Maximum ADCCLK frequency
1.62V ~ 1.8V	200K SPS	4MHz
1.8V ~ 2.8V	500K SPS	12MHz
2.8V ~ 3.3V	1M SPS	24MHz
3.3V ~ 5.5V	2M SPS	48MHz

The corresponding relationship between the conversion speed of the ADC and the working clock ADCCLK is as follows:

$$\text{ADC conversion speed} = f_{\text{ADCCLK}} / N_T$$

Where, f_{ADCCLK} is the clock frequency of ADCCLK, and N_T is ADCCLK number required for one ADC conversion.

20.4.3 Conversion results

After the ADC completes each conversion, the conversion result of the currently converted sequence channel SQRCH_y (y=0, 1, 2, 3, 4, 5, 6, 7) is stored in the corresponding ADC_RESULT_y (y=0, 1, 2, 3, 4, 5, 6, 7) register.

The sequence conversion result register is 16 bits wide, the conversion result is right-aligned, and the valid value is stored in the low 12 bits (Bits 11:0) of the ADC_RESULT_y register, and the high bits (Bits 15:12) are automatically filled with 0.



20.5 Operating modes

The ADC supports sequence channel conversion mode, which can convert up to 8 sequence channels (SQRCH0~SQRCH7) in turn, and the number of sequence channels to be converted is configured by the ENS bit field of the ADC_CR register. Each sequence channel SQRCHy (y=0, 1, 2, 3, 4, 5, 6, 7) can select one of the 16 conversion sources, which is configured by the SQRCHy bit field of the sequence conversion channel configuration register ADC_SQRCFR, as shown in the following table:

Table 20-4 Sequence channel to be converted source

ADC_SQRCFR.SQRCHy	Source to be converted	GPIO
0000	ADC_IN0	PA00
0001	ADC_IN1	PA01
0010	ADC_IN2	PA02
0011	ADC_IN3	PA03
0100	ADC_IN4	PA04
0101	ADC_IN5	PA05
0110	ADC_IN6	PA06
0111	ADC_IN7	PB00
1000	ADC_IN8	PB01
1001	ADC_IN9	PB02
1010	ADC_IN10	PB03
1011	ADC_IN11	PB04
1100	ADC_IN12	PB05
1101	ADC_IN13	PB06
1110	TS built-in temperature sensor	-
1111	1.2V core voltage reference	-

ADC conversion is initiated by writing a 1 to the START bit field of the ADC start register ADC_START or by other peripherals, see [20.6 External trigger source](#).

The ADC supports single and continuous modes. In single mode (set ADC_CR.CONT to 0), the conversion configured in the ENS bit field is performed only once; in continuous mode (set ADC_CR.CONT to 1), the conversion configured in the ENS bit field is performed continuously.



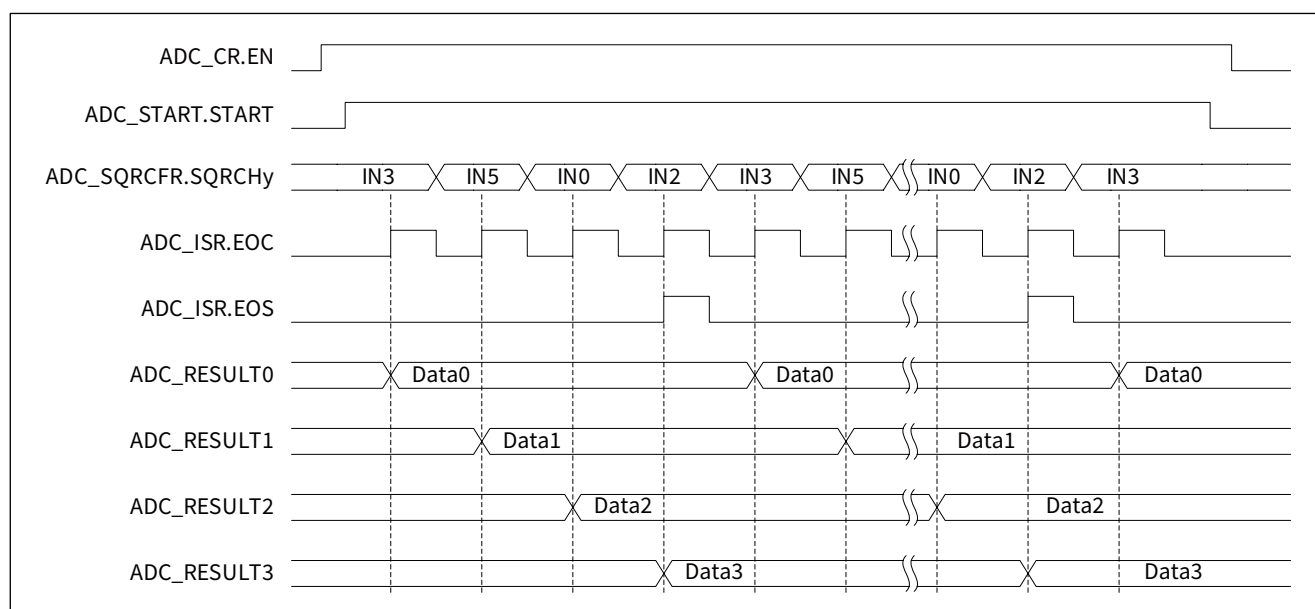
20.5.1 Sequence continuous conversion mode (CONT=1)

Set ADC_CR.CONT to 1 to enable continuous conversion mode. In this mode, whether the ADC is enabled by the software START bit or by an external trigger, once the ADC is enabled, the sequence conversion channel configured in the ENS bit field will continue to convert until the ADC_START.START bit is cleared.

After each ADC conversion is completed, the ADC_ISR.EOC flag bit is automatically set to 1, and the conversion results are stored in the sequence conversion result registers ADC_RESULT0 to ADC_RESULT7 of the same number as the sequence conversion channels SQRCH0 to SQRCH7. When all conversion of the selected sequence channel is completed, the sequence conversion completion flag bit ADC_ISR.EOS is set to 1. Users should read the conversion results in time to avoid overflow of conversion results. Clearing the ADC_START.START bit stops the conversion and resets the conversion channel of the sequence (i.e., starting the ADC again will restart the conversion from channel 0 of the sequence).

Taking the sequence conversion channels SQRCH0 ~ SQRCH3 as an example, the timing of the sequence continuous conversion mode is shown in the following figure:

Figure 20-3 Sequence continuous conversion timing diagram



The ADC sequence continuous conversion is started by the START bit, and the reference operation flow is as follows:

- Step 1: Set SYSCTRL_AHBEN.GPIOx to 1, SYSCTRL_APBEN1.ADC to 1, enable the GPIO clock corresponding to the ADC channel, and the ADC working clock;
- Step 2: Set the GPIO pin corresponding to the ADC channel to the analog function. For the specific register configuration, please refer to chapter [8 General-purpose input/output \(GPIO\)](#);
- Step 3: Set ADC_CR.EN to 1 to enable ADC module;
- Step 4: Delay about 1μs to wait for the completion of initialization of analog circuit;
- Step 5: Set ADC_CR.CONT to 1, select continuous conversion mode;
- Step 6: Configure the corresponding bit field of the ADC_SAMPLE register and ADC_CR.CLK, set ADC sampling speed and clock selection;
- Step 7: Configure ADC_CR.ENS, select the number of channels to-be-converted in sequence, as shown in [Figure 20-3](#), set ADC_CR.ENS to 3, and the sequence converted channels are SQRCH0~SQRCH3;
- Step 8: Configure ADC_SQRCFR.SQRCH0, select the sequence channel 0 to-be-converted source, as shown in [Figure 20-3](#), set ADC_SQRCFR.SQRCH0 to 3, and the channel 0 to-be-converted source is ADC_IN3;

- Step 9: Configure ADC_SQRCFR.SQRCH1, select the sequence channel 1 to-be-converted source, as shown in [Figure 20-3](#), set ADC_SQRCFR.SQRCH1 to 5, and the channel 1 to-be-converted source is ADC_IN5;
- Step 10: Configure ADC_SQRCFR.SQRCH2, select the sequence channel 2 to-be-converted source, as shown in [Figure 20-3](#), set ADC_SQRCFR.SQRCH2 to 0, and the channel 2 to-be-converted source is ADC_IN0;
- Step 11: Configure ADC_SQRCFR.SQRCH3, select the sequence 1 channel 3 to-be-converted source, as shown in [Figure 20-3](#), set ADC_SQRCFR.SQRCH3 to 2, and the channel 3 to-be-converted source is ADC_IN2;
- Step 12: Set ADC_ICR to 0, clear ADC interrupt flag;
- Step 13: Set ADC_START.START to 1 to start ADC conversion;
- Step 14: Wait for ADC_ISR.EOS to become 1, and read ADC_RESULT0 to ADC_RESULT3 registers in turn to obtain ADC conversion results for the corresponding channels. When ADC_ISR.EOS becomes 1, it indicates that the sequence conversion of 4 channels at a time is completed;
- Step 15: Set ADC_START.START to 0 to stop ADC conversion;
- Step 16: If you need to convert other channels, repeat steps 6 to 15;
- Step 17: Set ADC_CR.EN to 0 to turn off the ADC module.



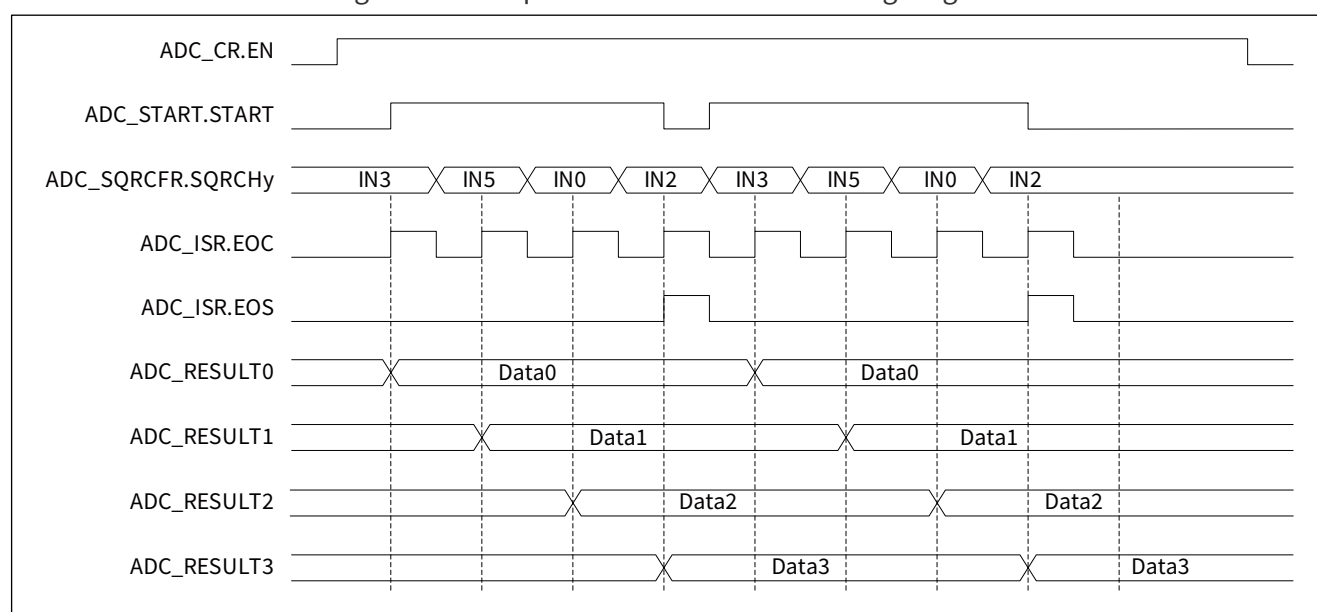
20.5.2 Sequence scan conversion mode (CONT=0)

Set ADC_CR.CONT to 0 to enable single conversion mode. In this mode, whether the ADC is enabled by the software START bit or by an external trigger, enabling the ADC once will only convert all of the sequence conversion channels configured in the ENS bit field once.

After each ADC conversion is completed, the ADC_ISR.EOC flag bit is automatically set to 1, and the conversion results are stored in the sequence conversion result registers ADC_RESULT0 to ADC_RESULT7 of the same number as the sequence conversion channels SQRCH0 to SQRCH7. When all the selected sequence channels are converted, the sequence conversion completion flag bit ADC_ISR.EOS is set to 1, the ADC_START.START bit is automatically cleared to 0, and the ADC stops conversion.

Taking the sequence conversion channels SQRCH0 ~ SQRCH3 as an example, the timing of the sequence scan conversion mode is shown in the following figure:

Figure 20-4 Sequence scan conversion timing diagram



The ADC sequence scan conversion is started by the START bit, and the reference operation flow is as follows:

- Step 1: Set `SYSCTRL_AHBEN.GPIOx` to 1, `SYSCTRL_APBEN1.ADC` to 1, enable the GPIO clock corresponding to the ADC channel, and the ADC working clock;
- Step 2: Set the GPIO pin corresponding to the ADC channel to the analog function. For the specific register configuration, please refer to chapter [8 General-purpose input/output \(GPIO\)](#);
- Step 3: Set `ADC_CR.EN` to 1 to enable ADC module;
- Step 4: Delay about 1μs to wait for the completion of initialization of analog circuit;
- Step 5: Set `ADC_CR.CONT` to 0, select single conversion mode;
- Step 6: Configure the corresponding bit field of the `ADC_SAMPLE` register and `ADC_CR.CLK`, set ADC sampling speed and clock selection;
- Step 7: Configure `ADC_CR.ENS`, select the number of channels to be converted in sequence, as shown in [Figure 20-4](#), set `ADC_CR.ENS` to 3, and the sequence converted channels are `SQRCH0 ~ SQRCH3`;
- Step 8: Configure `ADC_SQRCFR.SQRCH0`, select the sequence channel 0 to-be-converted source, as shown in [Figure 20-4](#), set `ADC_SQRCFR.SQRCH0` to 3, and the channel 0 to-be-converted source is `ADC_IN3`;
- Step 9: Configure `ADC_SQRCFR.SQRCH1`, select the sequence channel 1 to-be-converted source, as shown in [Figure 20-4](#), set `ADC_SQRCFR.SQRCH1` to 5, and the channel 1 to-be-converted source is `ADC_IN5`;
- Step 10: Configure `ADC_SQRCFR.SQRCH2`, select the sequence channel 2 to-be-converted source, as shown in [Figure 20-4](#), set `ADC_SQRCFR.SQRCH2` to 0, and the channel 2 to-be-converted source is `ADC_IN0`;
- Step 11: Configure `ADC_SQRCFR.SQRCH3`, select the sequence channel 3 to-be-converted source, as shown in [Figure 20-4](#), set `ADC_SQRCFR.SQRCH3` to 2, and the channel 3 to-be-converted source is `ADC_IN2`;
- Step 12: Set `ADC_ICR` to 0, clear ADC interrupt flag;
- Step 13: Set `ADC_START.START` to 1 to start ADC conversion;
- Step 14: Wait for `ADC_ISR.EOS` to become 1, and read `ADC_RESULT0` to `ADC_RESULT3` registers in turn to obtain ADC conversion results for the corresponding channels. When `ADC_ISR.EOS` becomes 1, it indicates that the sequence conversion of 4 channels at a time is completed, the `ADC_START.START` bit is automatically cleared to 0, and the ADC conversion stops;
- Step 15: If you need to convert other channels, repeat steps 6 to 14;
- Step 16: Set `ADC_CR.EN` to 0 to turn off the ADC module.



20.6 External trigger source

ADC conversion can be started either by software (i.e., ADC_START.START is set to 1) or by external trigger. The trigger source is selected by the external trigger start register ADC_TRIGGER. There are 19 ways to trigger ADC. See the relevant register description for details.



20.7 Analog watchdog

The analog watchdog function is often used for automatic monitoring of analog quantities, i.e., comparing ADC conversion results with user-set thresholds and generating interrupts.

The analog watchdog protects multiple selected channels by programming the INy (y=0~15) bit field of the ADC_AWDTCR register, i.e., when the INy bit field is set to 1, the watchdog for the corresponding channel is enabled.

The analog watchdog supports upper and lower threshold comparisons, which are set via the VTH and VT bit fields of the ADC_AWDTR register, respectively. When the analog voltage converted by ADC is lower than the lower threshold or higher than the upper threshold, the corresponding flag bit of analog watchdog will be set to 1. If the corresponding bit field (AWDH, AWDL) of interrupt enable register ADC_IER is set, an interrupt request will be generated. The upper and lower thresholds are compared as follows:

Upper threshold comparison: When the conversion result is within the interval (AWDTR.VTH, 4095], the ADC_ISR.AWDH flag bit is set to 1.

Lower threshold comparison: When the conversion result is within the interval [0, AWDTR.VTL), the ADC_ISR.AWDL flag bit is set to 1.

The internal hardware signal ADC_AWD at the analog watchdog output can also be connected directly to the ETR inputs of GTIM and ATIM or as a clear source for the timer reference signal OCREF, see the corresponding timer section.



20.8 BGR module

The nominal output voltage of the built-in BGR module of this chip is 1.2V. The voltage output by BGR module of each chip is slightly different. The voltage output by the BGR module has been accurately measured at the factory and stored in FLASH. The voltage value can be read out by using the expression $((\text{uint16\_t}) 0x001007D2)$, and its unit is mV.

The internal resistance of the output voltage is relatively large, and the duration of the sampling process should be configured not less than 40μs when converting the voltage. By converting the voltage at the BGR output, the exact value of the current V_{DD} supply voltage can be calculated using the formula shown below:

$$V_{DD} = V_{BGR} * 4095 / \text{RESULT}$$

The specific operation steps are as follows:

Step 1: Set SYCTRL_APBEN1.ADC to 1 to enable ADC configuration clock and operating clock;

Step 2: Set ADC_CR.EN to 1 to enable ADC module;

Step 3: Set ADC_CR.BGREN to 1 to enable BGR module;

Step 4: Configure ADC_SQRCFR.SQRCH0 as 0xF, select the channel to be converted as BGR 1.2V;

Step 5: Configure ADC_CR.CLK to 3, select ADCCLK source to PCLK/8;

Step 6: Configure ADC_SAMPLE.SQRCH0 to 0xE and configure sampling time to 262 ADCCLK clock cycles;

Caution:

Here, PCLK frequency is assumed to be 48MHz, ensuring that the duration of the sampling process is not less than 40μs.

Step 7: Configure ADC_CR.ENS to 0 to convert only one channel;

Step 8: Configure ADC_CR.CONT to 0, single conversion mode;

Step 9: Set ADC_START.START to 1 to start ADC conversion;

Step 10: Query and wait for ADC_START.START to become 0, ADC conversion completed;

Step 11: Read out the conversion result from ADC_RESULT0;

Step 12: Set ADC_CR.EN to 0, turn off ADC module;

Step 13: Calculate the exact value of V_{DD} in mV as shown below.

$$V_{DD} = ((\text{uint16\_t}) 0x001007D2) * 4095 / \text{RESULT0}$$



20.9 Temperature sensor

CW32L010 has a built-in temperature sensor module. The output voltage of the sensor changes with temperature. Set the sampling channel of the ADC module as an internal temperature sensor, and the current ambient temperature can be calculated through the ADC measurement results.

The temperature sensor is off by default. By setting the TSEN bit field of the control register ADC_CR to 1, the temperature sensor is enabled, and enable for approximately 30μs before switching TS channels.

The formula for calculating the ambient temperature is as follows:

$$\text{Ambient temperature} = T_0 * 0.5 + 0.095765 * (V_{\text{ref}} * \text{AdcValue} - 4.095 * \text{Trim})$$

Where:

V_{ref} is the reference voltage of the current ADC module, which takes the value of V_{DD} in V.

T_0 is the 8-bit initial calibration temperature value, which is recorded in the FLASH memory of the chip. Its address is 0x0001 07CD, and the unit is 0.5 degrees Celsius. The read value needs to be divided by 2 to be the actual temperature.

AdcValue is the ADC conversion result of the ADC module measuring the output voltage of the temperature sensor, and the value range is 0 ~ 4095.

Trim is a 16-bit calibration value, which needs to be read from the FLASH memory of the chip during calculation, and its storage address is shown in the following table:

Table 20-5 ADC calibration value address

ADC reference voltage	Calibration value storage address	Calibration value accuracy
V_{DD}	0x0010 07CE - 0x0010 07CF	$\pm 3^{\circ}\text{C}$

A calculation example is as follows:

Condition 1: $V_{\text{ref}} = 3.3$, AdcValue = 0x3CD, Trim = 0x311, $T_0 = 0x32$

Temperature 1: $0x32 * 0.5 + 0.095765 * (3.3 * 0x3CD - 4.095 * 0x311) = 24.65^{\circ}\text{C}$

The reference operation flow for measuring the ambient temperature through the ADC is as follows:

Step 1: Set SYCTRL_APBEN1.ADC to 1, enable ADC configuration clock and working clock;

Step 2: Set ADC_CR.EN to 1 to enable the ADC module;

Step 3: Delay about 1μs to wait for the completion of initialization of analog circuit;

Step 4: Set ADC_CR.ENS to 0, convert SQRCH to 0;

Step 5: Set ADC_SQRCFR.SQRCH0 to 0xE, select sequence channel 0 to-be-converted source as built-in temperature sensor;

Step 6: Configure ADC_SAMPLE.SQRCH0 and ADC_CR.CLK to set ADC conversion speed;

Step 7: Set ADC_CR.TSEN to 1 to enable the temperature sensor and wait approximately 30μs for it to stabilize;

Step 8: Set ADC_ICR.EOC to 0, clear ADC_ISR.EOC flag;

Step 9: Set ADC_START.START to 1 to start ADC conversion;

Step 10: Wait for ADC_ISR.EOC to become 1, and read the ADC_RESULT0 register to obtain the ADC conversion result;

Step 11: Set ADC_CR.EN to 0, turn off ADC module;

Step 12: Read T_0 and Trim, and calculate the current ambient temperature according to the formula.



20.10 ADC interrupts

ADC interrupt request, as shown in the following table:

Table 20-6 ADC interrupt sources and interrupt flags

Interrupt source	Interrupt flag bit	Interrupt enable	Flag clear
Conversion result > ADC_AWDTR.VTH	ADC_ISR.AWDH	ADC_IER.AWDH is set to 1	ADC_ICR.AWDH is cleared to 0
Conversion result < ADC_AWDTR.VTL	ADC_ISR.AWDL	ADC_IER.AWDL is set to 1	ADC_ICR.AWDL is cleared to 0
ADC sequence conversion completed	ADC_ISR.EOS	ADC_IER.EOS is set to 1	ADC_ICR.EOS is cleared to 0
ADC conversion completed	ADC_ISR.EOC	ADC_IER.EOC is set to 1	ADC_ICR.EOC is cleared to 0



20.11 List of registers

ADC base address: ADC_BASE = 0x4000 0000

Table 20-7 List of ADC registers

Register name	Register address	Register description
ADC_CR	ADC_BASE + 0x00	ADC Control register
ADC_START	ADC_BASE + 0x08	ADC Startup register
ADC_SAMPLE	ADC_BASE + 0x28	ADC Sample cycle configuration register
ADC_SQRCFR	ADC_BASE + 0x2C	ADC Sequence conversion channel configuration register
ADC_AWDTR	ADC_BASE + 0x10	ADC Analogue watchdog threshold register
ADC_AWDCR	ADC_BASE + 0x20	ADC Analogue watchdog configuration register
ADC_TRIGGER	ADC_BASE + 0x18	ADC External trigger register
ADC_IER	ADC_BASE + 0x74	ADC Interrupt enable register
ADC_ISR	ADC_BASE + 0x7C	ADC Interrupt flag register
ADC_ICR	ADC_BASE + 0x78	ADC Interrupt flag clear register
ADC_RESULT0	ADC_BASE + 0x40	ADC Sequence conversion result register 0
ADC_RESULT1	ADC_BASE + 0x44	ADC sequence conversion result register 1
ADC_RESULT2	ADC_BASE + 0x48	ADC sequence conversion result register 2
ADC_RESULT3	ADC_BASE + 0x4C	ADC sequence conversion result register 3
ADC_RESULT4	ADC_BASE + 0x50	ADC sequence conversion result register 4
ADC_RESULT5	ADC_BASE + 0x54	ADC sequence conversion result register 5
ADC_RESULT6	ADC_BASE + 0x58	ADC sequence conversion result register 6
ADC_RESULT7	ADC_BASE + 0x5C	ADC sequence conversion result register 7



20.12 Register descriptions

See section [1 Documentation conventions](#) for abbreviations used in register descriptions.

20.12.1 ADC_CR Control register

Address offset: 0x00 Reset value: 0x0000 00C0

Bit field	Name	Permission	Function description
31:9	RFU	-	Reserved bits, please keep the default value.
8:6	ENS	RW	Number of channels to be converted in sequence configuration 000: Convert SQRCH0 001: Convert SQRCH0 – SQRCH1 010: Convert SQRCH0 – SQRCH2 011: Convert SQRCH0 – SQRCH3 100: Convert SQRCH0 – SQRCH4 101: Convert SQRCH0 – SQRCH5 110: Convert SQRCH0 – SQRCH6 111: Convert SQRCH0 – SQRCH7
5:4	CLK	RW	ADC working clock ADCCLK source configuration 00: ADCCLK = PCLK / 1 01: ADCCLK = PCLK / 2 10: ADCCLK = PCLK / 4 11: ADCCLK = PCLK / 8
3	CONT	RW	Continuous conversion enable configuration 0: Single mode (performs the conversion configured by the ENS once) 1: Continuous mode (performs ENS-configured conversions continuously)
2	TSEN	RW	Built-in temperature sensor enable control 0: Disable built-in temperature sensor 1: Enable built-in temperature sensor Caution: TS channels cannot be converted until approximately 30μs after enabled.
1	BGREN	RW	Built-in BGR reference enable control 0: Disable built-in BGR reference 1: Enable built-in BGR reference Caution: The BGR1.2V channel cannot be converted until approximately 30μs after enabled.
0	EN	RW	ADC enable control 0: Disable ADC 1: Enable ADC Caution: External channels cannot be converted until approximately 1μs after enabled.



20.12.2 ADC_START Startup register

Address offset: 0x08 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:1	RFU	-	Reserved bits, please keep the default value.
0	START	RW	ADC start conversion control 0: Stop ADC conversion 1: Start ADC conversion <i>Caution: Software write 0 to stop conversion and reset the conversion channel of sequence.</i>

20.12.3 ADC_SAMPLE Sample cycle configuration register

Address offset: 0x28 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:28	SQRCH7	RW	ADC sequence channel 7 sample time configuration: See SQRCH0 bit field.
27:24	SQRCH6	RW	ADC sequence channel 6 sample time configuration: See SQRCH0 bit field.
23:20	SQRCH5	RW	ADC sequence channel 5 sample time configuration: See SQRCH0 bit field.
19:16	SQRCH4	RW	ADC sequence channel 4 sample time configuration: See SQRCH0 bit field.
15:12	SQRCH3	RW	ADC sequence channel 3 sample time configuration: See SQRCH0 bit field.
11:8	SQRCH2	RW	ADC sequence channel 2 sample time configuration: See SQRCH0 bit field.
7:4	SQRCH1	RW	ADC sequence channel 1 sample time configuration: See SQRCH0 bit field.
3:0	SQRCH0	RW	ADC sequence channel 0 sample time configuration: 0000: 6 ADCCLK clock cycles 1000: 54 ADCCLK clock cycles 0001: 7 ADCCLK clock cycles 1001: 70 ADCCLK clock cycles 0010: 9 ADCCLK clock cycles 1010: 102 ADCCLK clock cycles 0011: 12 ADCCLK clock cycles 1011: 134 ADCCLK clock cycles 0100: 18 ADCCLK clock cycles 1100: 166 ADCCLK clock cycles 0101: 24 ADCCLK clock cycles 1101: 198 ADCCLK clock cycles 0110: 30 ADCCLK clock cycles 1110: 262 ADCCLK clock cycles 0111: 42 ADCCLK clock cycles 1111: 390 ADCCLK clock cycles <i>Caution 1: Conversion time = sampling time + successive comparison time (fixed at 15 ADCCLKs).</i> <i>Caution 2: When the converted channel is BGR or TS, the sampling duration needs at least 40μs.</i>



20.12.4 ADC_SQRCFR Sequence conversion channel configuration register

Address offset: 0x2C Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:28	SQRCH7	RW	Sequence channel 7 to-be-converted source configuration: See SQRCH0.
27:24	SQRCH6	RW	Sequence channel 6 to-be-converted source configuration: See SQRCH0.
23:20	SQRCH5	RW	Sequence channel 5 to-be-converted source configuration: See SQRCH0.
19:16	SQRCH4	RW	Sequence channel 4 to-be-converted source configuration: See SQRCH0.
15:12	SQRCH3	RW	Sequence channel 3 to-be-converted source configuration: See SQRCH0.
11:8	SQRCH2	RW	Sequence channel 2 to-be-converted source configuration: See SQRCH0.
7:4	SQRCH1	RW	Sequence channel 1 to-be-converted source configuration: See SQRCH0.
3:0	SQRCH0	RW	Sequence channel 0 to-be-converted source configuration: 0000: ADC_IN0 1000: ADC_IN8 0001: ADC_IN1 1001: ADC_IN9 0010: ADC_IN2 1010: ADC_IN10 0011: ADC_IN3 1011: ADC_IN11 0100: ADC_IN4 1100: ADC_IN12 0101: ADC_IN5 1101: ADC_IN13 0110: ADC_IN6 1110: Built-in temperature sensor 0111: ADC_IN7 1111: 1.2V core voltage reference <i>Caution: When channels 1110~1111 are selected, the sampling duration requires at least 40μs.</i>

20.12.5 ADC_AWDTR Analogue watchdog threshold register

Address offset: 0x10 Reset value: 0x0FFF 0000

Bit field	Name	Permission	Function description
31:28	RFU	-	Reserved bits, please keep the default value.
27:16	VTH	RW	Analogue watchdog threshold high limit value
15:12	RFU	-	Reserved bits, please keep the default value.
11:0	VTL	RW	Analogue watchdog threshold low limit value



20.12.6 ADC_AWDCR Analogue watchdog configuration register

Address offset: 0x20 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value.
15	IN15	RW	Analog watchdog monitoring ADC_IN15 enable control (i.e, 1.2V core voltage reference) 0: Disabled 1: Enabled
14	IN14	RW	Analog watchdog monitoring ADC_IN14 enable control (i.e, built-in temperature sensor output signal) 0: Disabled 1: Enabled
13	IN13	RW	Analog watchdog monitoring ADC_IN13 enable control 0: Disabled 1: Enabled
12	IN12	RW	Analogue watchdog monitoring ADC_IN12 enable control 0: Disabled 1: Enabled
11	IN11	RW	Analogue watchdog monitoring ADC_IN11 enable control 0: Disabled 1: Enabled
10	IN10	RW	Analogue watchdog monitoring ADC_IN10 enable control 0: Disabled 1: Enabled
9	IN9	RW	Analogue watchdog monitoring ADC_IN9 enable control 0: Disabled 1: Enabled
8	IN8	RW	Analogue watchdog monitoring ADC_IN8 enable control 0: Disabled 1: Enabled
7	IN7	RW	Analogue watchdog monitoring ADC_IN7 enable control: 0: Disabled 1: Enabled
6	IN6	RW	Analogue watchdog monitoring ADC_IN6 enable control 0: Disabled 1: Enabled
5	IN5	RW	Analogue watchdog monitoring ADC_IN5 enable control 0: Disabled 1: Enabled



Bit field	Name	Permission	Function description
4	IN4	RW	Analogue watchdog monitoring ADC_IN4 enable control 0: Disabled 1: Enabled
3	IN3	RW	Analogue watchdog monitoring ADC_IN3 enable control 0: Disabled 1: Enabled
2	IN2	RW	Analogue watchdog monitoring ADC_IN2 enable control 0: Disabled 1: Enabled
1	IN1	RW	Analogue watchdog monitoring ADC_IN1 enable control 0: Disabled 1: Enabled
0	IN0	RW	Analogue watchdog monitoring ADC_IN0 enable control 0: Disabled 1: Enabled

20.12.7 ADC_TRIGGER External trigger register

Address offset: 0x18 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:19	RFU	-	Reserved bits, please keep the default value.
18	UART2	RW	UART2 triggers ADC startup 0: Disabled 1: Enabled
17	UART1	RW	UART1 triggers ADC startup 0: Disabled 1: Enabled
16	SPI	RW	SPI triggers ADC startup 0: Disabled 1: Enabled
15	BTIM3TRGO	RW	BTIM3 TRGO singal triggers ADC startup 0: Disabled 1: Enabled
14	BTIM2TRGO	RW	BTIM2 TRGO singal triggers ADC startup 0: Disabled 1: Enabled
13	BTIM1TRGO	RW	BTIM1 TRGO singal triggers ADC startup 0: Disabled 1: Enabled



Bit field	Name	Permission	Function description
12	GTIMCC4	RW	GTIM output compare CC4 signal trigger ADC startup 0: Disabled 1: Enabled
11	GTIMCC3	RW	GTIM output compare CC3 signal trigger ADC startup 0: Disabled 1: Enabled
10	GTIMCC2	RW	GTIM output compare CC2 signal trigger ADC startup 0: Disabled 1: Enabled
9	GTIMCC1	RW	GTIM output compare CC1 signal trigger ADC startup 0: Disabled 1: Enabled
8	GTIMTRGO	RW	GTIM TRGO singal triggers ADC startup 0: Disabled 1: Enabled
7	ATIMCC6	RW	ATIM output compare CC6 signal trigger ADC startup 0: Disabled 1: Enabled
6	ATIMCC5	RW	ATIM output compare CC5 signal trigger ADC startup 0: Disabled 1: Enabled
5	ATIMCC4	RW	ATIM output compare CC4 signal trigger ADC startup 0: Disabled 1: Enabled
4	ATIMCC3	RW	ATIM output compare CC3 signal trigger ADC startup 0: Disabled 1: Enabled
3	ATIMCC2	RW	ATIM output compare CC2 signal trigger ADC startup 0: Disabled 1: Enabled
2	ATIMCC1	RW	ATIM output compare CC1 signal trigger ADC startup 0: Disabled 1: Enabled
1	ATIMTRGO2	RW	ATIM TRGO2 singal triggers ADC startup 0: Disabled 1: Enabled
0	ATIMTRGO	RW	ATIM TRGO singal triggers ADC startup 0: Disabled 1: Enabled



20.12.8 ADC_IER Interrupt enable register

Address offset: 0x74 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:4	RFU	-	Reserved bits, please keep the default value.
3	AWDH	RW	Analogue watchdog threshold high limit interrupt enable control 0: Disabled 1: Enabled
2	AWDL	RW	Analogue watchdog threshold low limit interrupt enable control 0: Disabled 1: Enabled
1	EOS	RW	Sequence conversion completion interrupt enable control 0: Disabled 1: Enabled
0	EOC	RW	Single conversion completion interrupt enable control 0: Disabled 1: Enabled

20.12.9 ADC_ISR Interrupt flag register

Address offset: 0x7C Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:4	RFU	-	Reserved bits, please keep the default value.
3	AWDH	RO	Analog watchdog threshold high limit flag 0: The conversion result is in the interval [0, AWDTR.VTH] 1: The conversion result is in the interval (AWDTR.VTH, 4095]
2	AWDL	RO	Analog watchdog threshold low limit flag 0: The conversion result is in the interval [AWDTR.VTL, 4095] 1: The conversion result is in the interval [0, AWDTR.VTL)
1	EOS	RO	Sequence conversion completion flag 0: Sequence conversion not completed 1: Sequence conversion completed
0	EOC	RO	Conversion completion flag 0: One ADC conversion not completed 1: One ADC conversion completed



20.12.10 ADC_ICR Interrupt flag clear register

Address offset: 0x78 Reset value: 0x0000 000F

Bit field	Name	Permission	Function description
31:4	RFU	-	Reserved bits, please keep the default value.
3	AWDH	R1W0	Analogue watchdog threshold high limit flag clear 0 control W0: Clear the corresponding flag in the ISR register W1: No function
2	AWDL	R1W0	Analogue watchdog threshold low limit flag clear 0 control W0: Clear the corresponding flags in the ISR register W1: No function
1	EOS	R1W0	Sequence conversion completion flag clear 0 control W0: Clear the corresponding flags in the ISR register W1: No function
0	EOC	R1W0	Conversion completion flag clear 0 control W0: Clear the corresponding flags in the ISR register W1: No function

20.12.11 ADC_RESULT0 Sequence conversion result register 0

Address offset: 0x40 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value.
15:0	RESULT	RO	ADC sequence conversion result 0

20.12.12 ADC_RESULT1 Sequence conversion result register 1

Address offset: 0x44 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value.
15:0	RESULT	RO	ADC sequence conversion result 1



20.12.13 ADC_RESULT2 Sequence conversion result register 2

Address offset: 0x48 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value.
15:0	RESULT	RO	ADC sequence conversion result 2

20.12.14 ADC_RESULT3 Sequence conversion result register 3

Address offset: 0x4C Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value.
15:0	RESULT	RO	ADC sequence conversion result 3

20.12.15 ADC_RESULT4 Sequence conversion result register 4

Address offset: 0x50 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value.
15:0	RESULT	RO	ADC sequence conversion result 4

20.12.16 ADC_RESULT5 Sequence conversion result register 5

Address offset: 0x54 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value.
15:0	RESULT	RO	ADC sequence conversion result 5

20.12.17 ADC_RESULT6 Sequence conversion result register 6

Address offset: 0x58 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value.
15:0	RESULT	RO	ADC sequence conversion result 6



20.12.18 ADC_RESULT7 Sequence conversion result register 7

Address offset: 0x5C Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:16	RFU	-	Reserved bits, please keep the default value.
15:0	RESULT	RO	ADC sequence conversion result 7



21 Analog voltage comparator (VC)

21.1 Overview

CW32L010 integrates two analog voltage comparators (VC), which are used to compare two analog input voltages, and output the comparison results from the pins. The positive terminal input of the voltage comparator supports 4 external analog inputs, and the negative terminal supports not only 2 external analog inputs, but also internal 1.2V voltage reference, internal resistance voltage divider output voltage. The comparison result output has filtering function, hysteresis window function, and polarity selection. Support compare interrupt, which can be used to wake up MCU in low power mode.

21.2 Main features

- Dual-channel analog voltage comparators VC1, VC2
- Internal 8-step resistor divider
- 4 external analog signal inputs
- 2 on-chip analog input signals
 - Built-in resistor divider output voltage
 - Built-in 1.2V reference voltage
- Selectable output polarity
- Supports hysteresis window compare function
- Programmable filters and filter time
- 3 interrupt trigger modes, which can be used in combination
 - High level trigger
 - Rising edge trigger
 - Falling edge trigger
- Supports running in low power mode, interrupt wake-up MCU

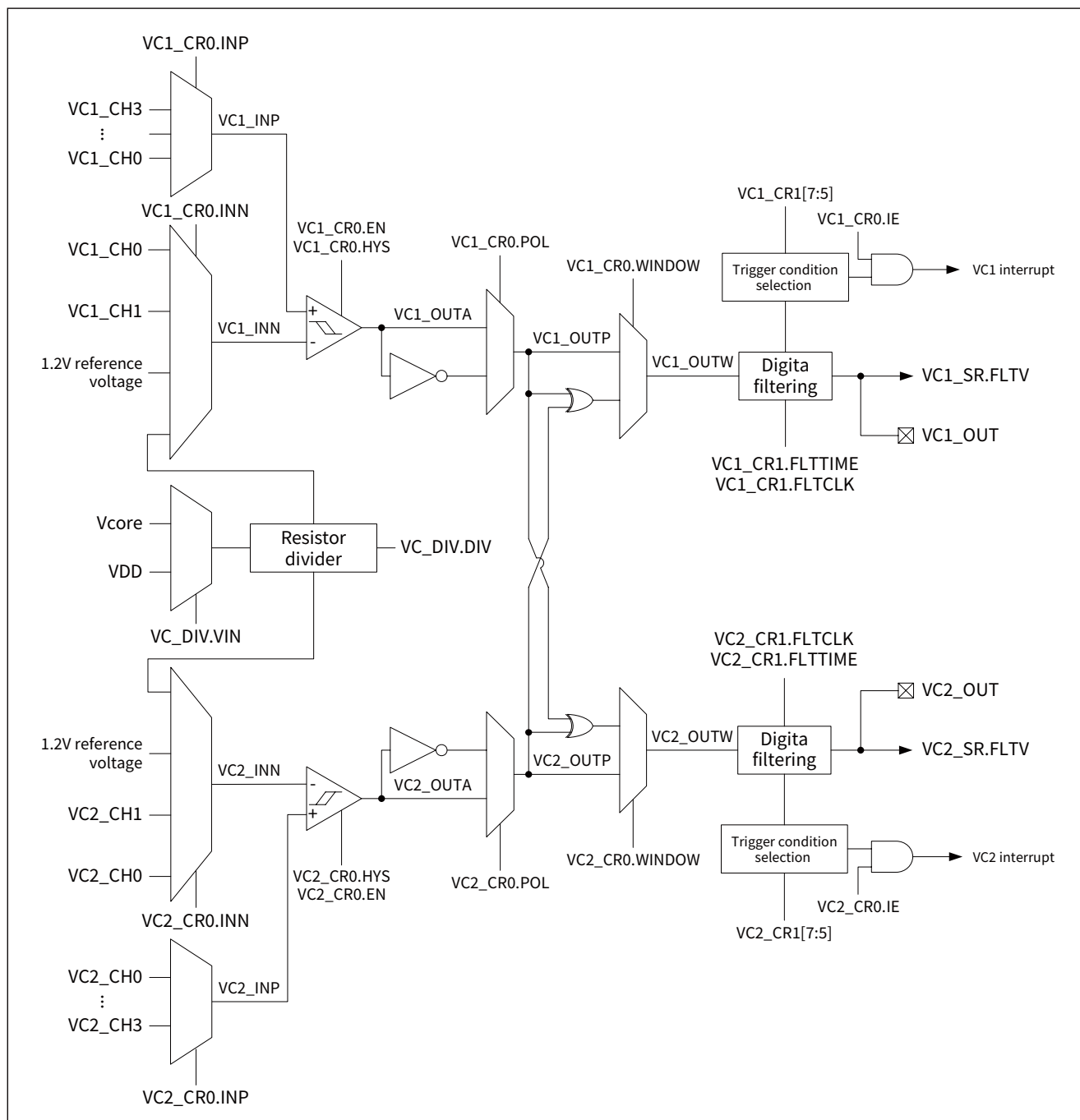


21.3 Functional description

21.3.1 Functional block diagram

The functional block diagram of the analog voltage comparator (VC) is shown in the following figure:

Figure 21-1 VC functional block diagram



The input selection of the positive and negative terminals of VC1 and VC2 is selected by the INP and INN bit fields of the control register VCx_CR0, as shown in the following table:

Table 21-1 VC positive and negative input signal configuration

VCx_CR0.INP	VCx positive terminal input signal	VCx_CR0.INN	VCx negative terminal input signal
00	VCx_CH0	00	VCx_CH0
01	VCx_CH1	01	VCx_CH1
10	VCx_CH2	10	Built-in 1.2V reference voltage
11	VCx_CH3	11	Built-in resistor divider output voltage

The built-in resistor divider configuration is controlled by three bit fields in the VC_DIV register¹:

- EN bit field, enable resistor divider
- VIN bit field, select the input voltage of the voltage divider
 - VIN is 0, select V_{DD}
 - VIN is 1, select V_{core} , about 1.6V
- DIV bit field, voltage divider proportional coefficient, valid range is 0~7, select the output voltage of the voltage divider, the calculation formula is as follows:

$$\text{Voltage divider output voltage} = \text{input voltage} \times (1 + \text{DIV}) / 8$$

Caution 1:

VC1 and VC2 share the resistor divider circuit.

When selecting the built-in 1.2V reference voltage as the negative input of the comparator, it is necessary to enable the BGR module inside the chip. The startup time of the internal BGR is about 30μs, the VC voltage comparator needs to wait for the internal BGR to stabilize before it can work normally.



21.3.2 I/O pins

The analog voltage comparator supports 8 external analog signal inputs, and the user must configure the corresponding GPIO port as an analog function (GPIOx_ANALOG.PINy = 1).

The analog voltage comparator supports outputting the comparison result from the pin. The user must configure the corresponding GPIO port as a digital output and select the function multiplexing.

The I/O pins supported by VC1 and VC2 are shown in the following table:

Table 21-2 VC I/O pins configuration

VC1 I/O	GPIO	Configuration	VC2 I/O	GPIO	Configuration
VC1_CH0	PB05	Analog	VC2_CH0	PB06	Analog
VC1_CH1	PA00	Analog	VC2_CH1	PA05	Analog
VC1_CH2	PA03	Analog	VC2_CH2	PB02	Analog
VC1_CH3	PB03	Analog	VC2_CH3	PB04	Analog
VC1_OUT	PA02	Digital output (AFR=4)	VC2_OUT	PA04	Digital output (AFR=5)
	PA08	Digital output (AFR=4)		PB02	Digital output (AFR=4)

21.3.3 Latency/Response time

Set the EN bit field of the control register VCx_CR0 to 1 to enable the VC module.

The time from when VC is enabled or the input voltage at the positive and negative ends of VC changes to the time when the voltage comparator outputs the correct comparison result is defined as the delay/response time of the comparator. The delay/response time is configured by the RESP bit field of the control register VCx_CR0. The shorter the response time, the greater the power consumption of the VC module.



21.3.4 Polarity selection

The polarity of the output signals of the voltage comparators VC1 and VC2 is set by the POL bit field of the control register VCx_CR0:

- POL is 1, the polarity of VCx_OUTP signal is opposite to that of VCx_OUTA signal, that is, when the positive terminal is greater than the negative terminal, VCx outputs a low level.
- POL is 0, and the VCx_OUTP signal has the same polarity as the VCx_OUTA signal, that is, when the positive terminal is greater than the negative terminal, VCx outputs a high level.

21.3.5 Digital filter

The built-in digital filter of the voltage comparator is used to digitally filter the output signal of the voltage comparator. Users can use the filtering function to filter system noise, such as high current noise when the motor is stopped, to avoid system malfunction caused by the noise output of the comparator.

The clock for the digital filter is selected by the FLTCLK bit field in the control register VCx_CR1:

- FLTCLK is 1, use PCLK as filter clock.
- FLTCLK is 0, the built-in low speed RC oscillator clock LSI is used as the filter clock.

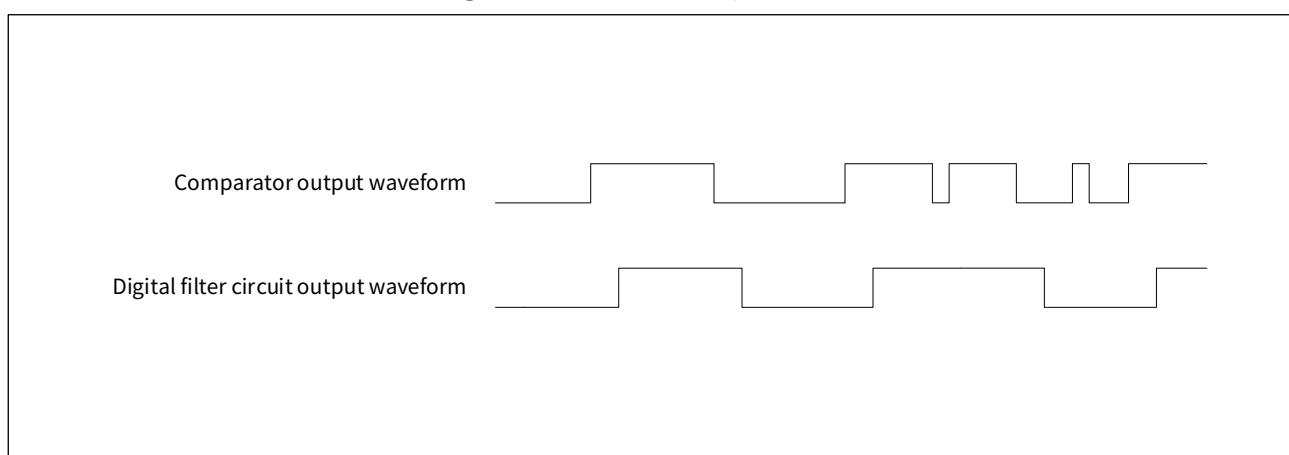
The filter samples the output signal with a certain sampling frequency. When the continuous sampling reaches N same levels, the signal is valid, otherwise the signal is invalid, so as to filter out high-frequency clutter signals. The sampling clock of the filter unit is FLTCLK or the frequency division of FLTCLK, and the frequency division ratio of FLTCLK and the number of sampling points N can be selected through the FLTIME bit field of the VCx_CR1 register.

The digitally filtered output level of the voltage comparator can be read out through the FLTV bit field of the status register VCx_SR.

When the user sets the VCx_OUT pin as a digital output and selects the multiplexing function of the VC comparison output, the VCx_OUT pin will output the digitally filtered output level of the voltage comparator.

The filter response waveform of VC is shown in the following figure:

Figure 21-2 VC filter response time



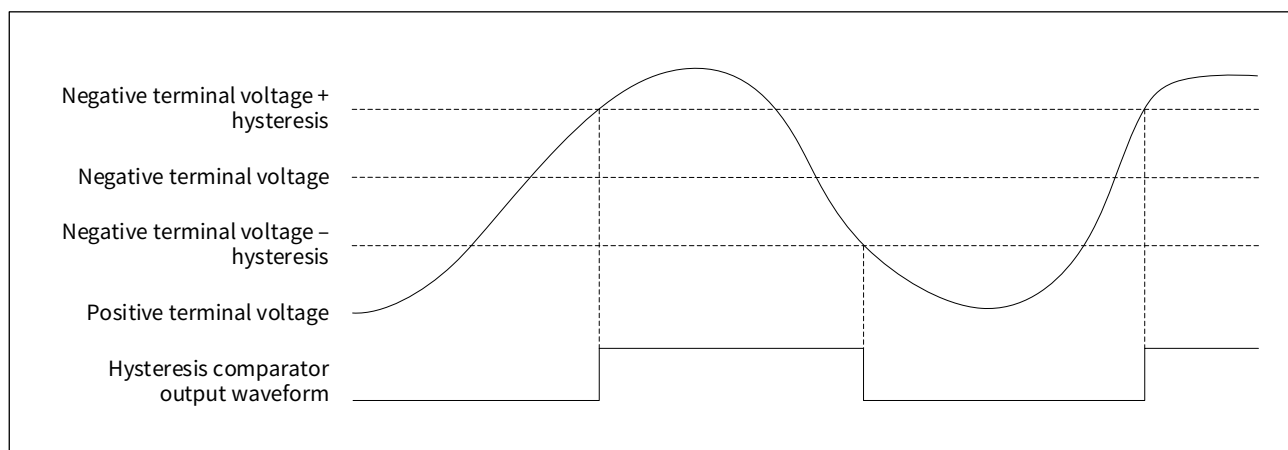
The output of the voltage comparator can be used as the brake input or OCREF_CLR input of the Advanced Timer (ATIM), for details about the brake and OCREF_CLR, please refer to [14 Advanced-control timer \(ATIM\)](#). This function is usually used in motor protection applications, please refer to the relevant application notes for details.

21.3.6 Hysteresis function

The analog voltage comparator supports the hysteresis function. After the hysteresis function is used, the output result of the comparator will not be reversed immediately with the change of the input signal, but will reverse only after the offset value of the two input signals is higher or lower than the hysteresis threshold voltage.

The waveform diagram of VC hysteresis function is shown in the following figure:

Figure 21-3 VC hysteresis function



The hysteresis function can effectively enhance the anti-interference ability of the chip, and avoid unnecessary frequent flipping of the output end of the comparator due to the small amplitude jitter of the input signal.

The hysteresis threshold voltage is determined by the HYS bit field of the control register VCx_CR0, as shown in the following table:

Table 21-3 VC hysteresis threshold voltage

VCx_CR0.HYS	Hysteresis window configuration
0	No hysteresis
1	Hysteresis window about 20mV

21.3.7 Window compare function

The analog voltage comparator supports the window compare function, which can output the comparison results of VC1 and VC2 after XOR operation, which is enabled by the WINDOW bit field of the control register VCx_CR0:

- When WINDOW is 1, the VCx_OUTW signal is the XOR value of the VC1_OUTP signal and the VC2_OUTP signal.
- When WINDOW is 0, the VCx_OUTW signal is at the same level as the VCx_OUTP signal.

21.3.8 BLANK window function

While keeping the VCx module working, if you want to temporarily stop the voltage comparison function, or in some application systems (such as motor control), the short-term reasonable fluctuation of the monitored signal may cause unnecessary reverse of output levels of voltage comparator, the voltage comparator of this chip has added the BLANK window function. That is, when the BLANK window is activated by a specified external trigger condition, no voltage comparison is performed during the set BLANK window period, and the voltage comparator outputs a set high or low level. level status. After the BLANK window period, the voltage comparator resumes normal operation.

The BLANK window duration is configured by the BLANKTIME bit field of the control register VCx_CR1. The window duration is

$$(2^{(\text{BLANKTIME} + 2)} \sim 2^{(\text{BLANKTIME} + 2)} + 2) \text{ PCLK cycles}$$

As shown in the following table:

Table 21-4 BLANK window duration configuration

VCx_CR1.BLANKTIME	The number of PCLK clocks in the BLANK window
000	4 ~ 6
001	8 ~ 10
010	16 ~ 18
011	32 ~ 34
100	64 ~ 66
101	128 ~ 130
110	256 ~ 258
111	512 ~ 514

The output level of the voltage comparator during the BLANK window duration is set by the BLANKLVL bit field of control register VCx_CR1. BLANKLVL outputs a low level when BLANKLVL is 0, and outputs a high level when BLANKLVL is 1.

The trigger start condition of the BLANK window is configured by the BLANKCH1, BLANKCH2, BLANKCH3, BLANKCH4, BLANKCH5, and BLANKCH6 bit fields of the control register VCx_CR1, and the BLANK window is triggered by the rising edges of the ATIM's OC1REFC, OC2REFC, OC3REFC, OC4REFC, OC5REFC, and OC6REFC respectively.



21.4 VC interrupts

The voltage comparator of CW32L010 supports working in low power consumption mode, and the comparison interrupt can wake up the chip from low power consumption mode.

Set the IE bit field of the control register VCx_CR0 to 1 to enable the VCx interrupt. When an interrupt occurs, the interrupt flag bit INTF of the status register VCx_SR will be set to 1 by hardware. The user can write 0 to the INTF bit to clear the interrupt flag.

Set the HIGHIE, RISEIE, FALLIE bit fields of the control register VCx_CR1 to select different interrupt triggering methods:

- HIGHIE is 1, VCx_OUT output signal high level triggers interrupt
- RISEIE is 1, the rising edge of VCx_OUT output signal triggers interrupt
- FALLIE is 1, the falling edge of VCx_OUT output signal triggers interrupt

21.5 Programming examples

In this example, the digital filtering function is used, and the compare interrupt is configured as follows:

Step 1: Configure VC_DIV.EN to 1, enable the voltage divider;

Step 2: Configure VC_DIV.DIV and set the voltage division coefficient;

Step 3: Configure VCx_CR0.INP and select the voltage source to be monitored at the positive terminal;

Step 4: Configure VCx_CR0.INN and select the voltage source to be monitored at the negative terminal;

Step 5: Configure VCx_CR1.FLTTIME and select the filter time;

Step 6: Configure VCx_CR1.FLTCLK, select filter clock;

Step 7: Set the HIGHIE, RISEIE, and FALLIE of the VCx_CR1 register to 1, and select the interrupt trigger mode;

Step 8: Set VCx_CR0.IE to 1 to enable VCx interrupt;

Step 9: Set VCx_CR0.EN to 1 to enable VCx;

Step 10: In the initialization routine of the VC module and the interrupt service routine of the VC module, write 0 to the INTF bit of the VCx_SR register, clear the interrupt flag, and enable the generation of the VCx interrupt.



21.6 List of registers

VC base address: VC_BASE = 0x4000 00A0

Table 21-5 List of VC registers

Register name	Register address	Register description
VC_DIV	VC_BASE + 0x00	Resistor divider control register
VCx_CR0	VC_BASE + 0x04 + 16 * (x-1)	Control register 0, x=1~2
VCx_CR1	VC_BASE + 0x08 + 16 * (x-1)	Control register 1, x=1~2
VCx_SR	VC_BASE + 0x0C + 16 * (x-1)	Status register, x=1~2



21.7 Register descriptions

See section [1 Documentation conventions](#) for abbreviations used in register descriptions.

21.7.1 VC_DIV Resistor divider control register

Address offset: 0x00 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:6	RFU	-	Reserved bits, please keep the default value
5	VIN	RW	Resistor divider circuit input voltage setting 0: VDD 1: Vcore (about 1.6V)
4	EN	RW	Resistor divider circuit enable control 0: Disabled 1: Enabled
3	RFU	-	Reserved bits, please keep the default value
2:0	DIV	RW	Resistor divider circuit output voltage configuration Output voltage = Input voltage * (1 + DIV) / 8



21.7.2 VCx_CR0 Control register 0

Address offset: $0x04 + 16 * (x-1)$ ($x=1\sim2$) Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:10	RFU	-	Reserved bits, please keep the default value
9:8	INN	RW	VCx negative terminal input signal configuration 00: VCx_CH0 01: VCx_CH1 10: Built-in 1.2V reference voltage 11: Built-in voltage divider output voltage
7:6	INP	RW	VCx positive terminal input signal configuration 00: VCx_CH0 01: VCx_CH1 10: VCx_CH2 11: VCx_CH3
5	WINDOW	RW	Window compare function configuration 0: Window function disabled, VCx_OUTW signal equals VCx_OUTP signal 1: Window function enabled, VCx_OUTW signal equals to the XOR of VC1_OUTP and VC2_OUTP
4	POL	RW	VCx output signal polarity settings 0: When the positive terminal is greater than the negative terminal, VCx outputs a high level 1: When the positive terminal is greater than the negative terminal, VCx outputs a low level
3	IE	RW	VCx interrupt enable configuration 0: Disabled 1: Enabled
2	HYS	RW	VCx hysteresis window configuration 0: No hysteresis 1: Hysteresis window about 20mV
1	RESP	RW	VCx response speed configuration 0: Low speed 1: High speed
0	EN	RW	VCx enable control 0: Disabled 1: Enabled



21.7.3 VCx_CR1 Control register 1

Address offset: $0x08 + 16 * (x-1)$ ($x=1\sim2$) Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:18	RFU	-	Reserved bits, please keep the default value
17	BLANKLVL	RW	BLANK window duration output level configuration 0: Output low level 1: Output high level
16:14	BLANKTIME	RW	BLANK window duration configuration: $(2^{(BLANKTIME+2)} \sim 2^{(BLANKTIME+2)+2})$ PCLK cycles <i>Caution: Any rising edge of OC1REFC, OC2REFC, OC3REFC, OC4REFC, OC5REFC, OC6REFC of ATIM will trigger the start of the BLANK window. During the duration of the BLANK window, no voltage comparison will be performed.</i>
13	BLANKCH6	RW	ATIM_OC6REFC rising edge triggers VCx to start BLANK window configuration 0: Disabled 1: Enabled
12	BLANKCH5	RW	ATIM_OC5REFC rising edge triggers VCx to start BLANK window configuration 0: Disabled 1: Enabled
11	BLANKCH4	RW	ATIM_OC4REFC rising edge triggers VCx to start BLANK window configuration 0: Disabled 1: Enabled
10	BLANKCH3	RW	ATIM_OC3REFC rising edge triggers VCx to start BLANK window configuration 0: Disabled 1: Enabled
9	BLANKCH2	RW	ATIM_OC2REFC rising edge triggers VCx to start BLANK window configuration 0: Disabled 1: Enabled
8	BLANKCH1	RW	ATIM_OC1REFC rising edge triggers VCx to start BLANK window configuration 0: Disabled 1: Enabled
7	HIGHIE	RW	VCx output signal high level trigger interrupt enabled 0: Disabled 1: Enabled



Bit field	Name	Permission	Function description
6	RISEIE	RW	VCx output signal rising edge trigger interrupt enabled 0: Disabled 1: Enabled
5	FALLIE	RW	VCx output signal falling edge trigger interrupt enabled 0: Disabled 1: Enabled
4	FLTCLK	RW	Digital filter module filter clock settings 0: LSI 1: PCLK <i>Caution: Before entering DeepSleep mode, when the digital filtering function is not required, please set the filtering clock source to PCLK, otherwise the LSI will generate power consumption when running in DeepSleep mode.</i>
3:0	FLTTIME	RW	Digital filter module filter time configuration 0000: No filter 0001: $f_{\text{SAMPLING}} = f_{\text{FLTCLK}}/1$, N=2 0010: $f_{\text{SAMPLING}} = f_{\text{FLTCLK}}/1$, N=4 0011: $f_{\text{SAMPLING}} = f_{\text{FLTCLK}}/1$, N=8 0100: $f_{\text{SAMPLING}} = f_{\text{FLTCLK}}/2$, N=6 0101: $f_{\text{SAMPLING}} = f_{\text{FLTCLK}}/2$, N=8 0110: $f_{\text{SAMPLING}} = f_{\text{FLTCLK}}/4$, N=6 0111: $f_{\text{SAMPLING}} = f_{\text{FLTCLK}}/4$, N=8 1000: $f_{\text{SAMPLING}} = f_{\text{FLTCLK}}/8$, N=6 1001: $f_{\text{SAMPLING}} = f_{\text{FLTCLK}}/8$, N=8 1010: $f_{\text{SAMPLING}} = f_{\text{FLTCLK}}/16$, N=5 1011: $f_{\text{SAMPLING}} = f_{\text{FLTCLK}}/16$, N=6 1100: $f_{\text{SAMPLING}} = f_{\text{FLTCLK}}/16$, N=8 1101: $f_{\text{SAMPLING}} = f_{\text{FLTCLK}}/32$, N=5 1110: $f_{\text{SAMPLING}} = f_{\text{FLTCLK}}/32$, N=6 1111: $f_{\text{SAMPLING}} = f_{\text{FLTCLK}}/32$, N=8



21.7.4 VCx_SR Status register

Address offset: $0x0C + 16 * (x-1)$ ($x=1\sim2$) Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:2	RFU	-	Reserved bits, please keep the default value
1	FLTV	RO	Level value of digital filter output 0: Digital filter output low level 1: Digital filter output high level
0	INTF	RW0	Interrupt flag R0: No VC interrupt occurred R1: VC interrupt has occurred W0: VC interrupt flag cleared W1: No function



22 Low voltage detector (LVD)

22.1 Overview

Low voltage detector (LVD) is used to monitor VDD power supply voltage or external pin input voltage. When the comparison result between the monitored voltage and the LVD threshold meets the trigger condition, an LVD interrupt or reset signal will be generated, which is usually used to handle some urgent tasks.

The interrupt and reset flags generated by the LVD can only be cleared by software; only after the interrupt or reset flag is cleared and the trigger condition is reached again, the LVD can generate an interrupt or reset signal again.

22.2 Main features

- 2-channel monitoring voltage source: VDD power supply voltage, PA03 pin input
- 8-step threshold voltage, range 1.8V ~ 4.6V
- 3 trigger conditions, which can be used in combination
 - Level triggered: voltage below threshold.
 - Falling edge trigger: The falling edge when the voltage drops below the threshold.
 - Rising edge trigger: The rising edge when the voltage rises above the threshold.
- Can trigger to generate interrupt or reset signal, both cannot be generated at the same time
- Programmable filter and filter time
- Hysteresis function supported
- Supports running in low power mode, interrupt wake-up MCU

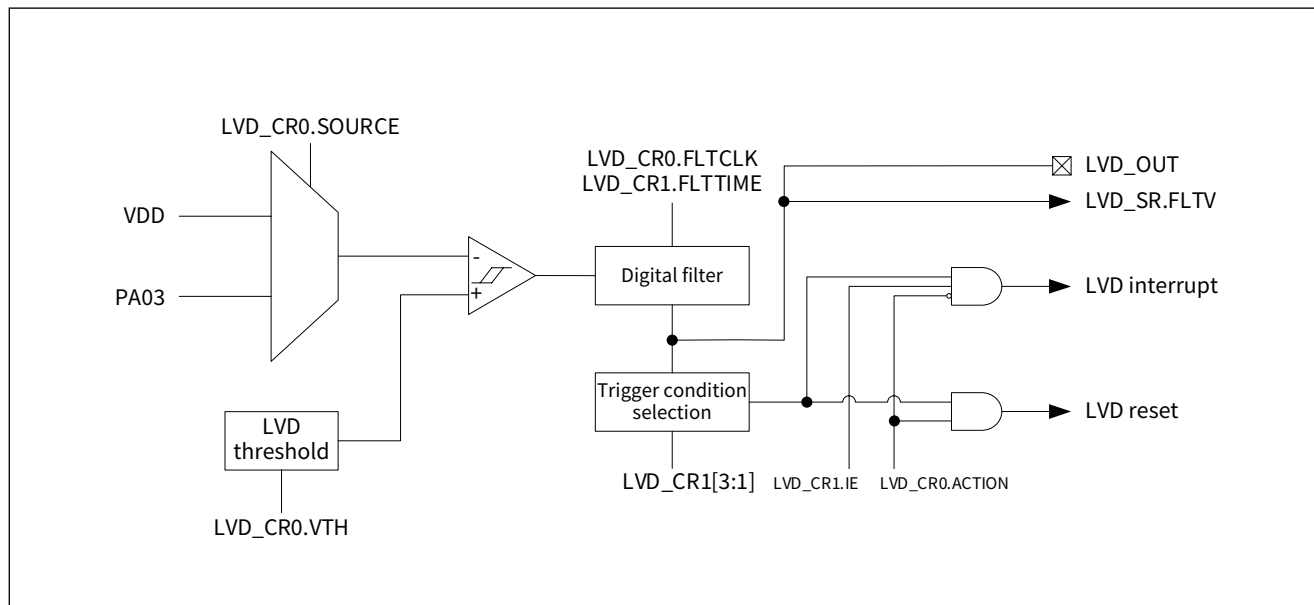


22.3 Functional description

22.3.1 Functional block diagram

The functional block diagram of the Low voltage detector (LVD) is shown in the following figure:

Figure 22-1 LVD functional block diagram



The LVD can monitor the VDD power supply voltage, as well as the external pin input voltage (PA03), which is selected by the SOURCE bit field of the control register LVD_CR0. When using external analog signal input, the user must configure the corresponding GPIO port as analog function (GPIOx_ANALOG.PINy = 1).

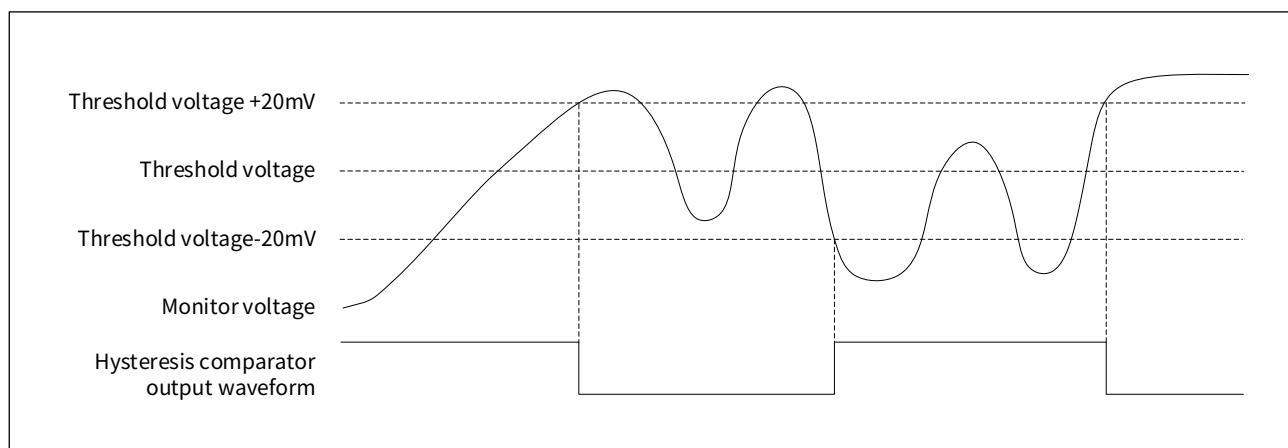
The LVD output result can be output from the PB00 pin, the user must configure the corresponding GPIO port as a digital output, and select function multiplexing at the same time.

22.3.2 Hysteresis function

The built-in voltage comparator of the LVD has a hysteresis function, which can avoid frequent flipping of the output result of the voltage comparator when the monitored voltage of the LVD is near the threshold voltage, and enhance the anti-interference ability of the system.

The comparator output signal toggles only when the monitored voltage is 20mV above or below the threshold voltage. The specific waveform is shown in the following figure:

Figure 22-2 LVD hysteretic response



The threshold voltage of LVD is determined by the VTH bit value of the control register LVD_CR0, and the effective value is 0 ~ 7, as shown in the following table:

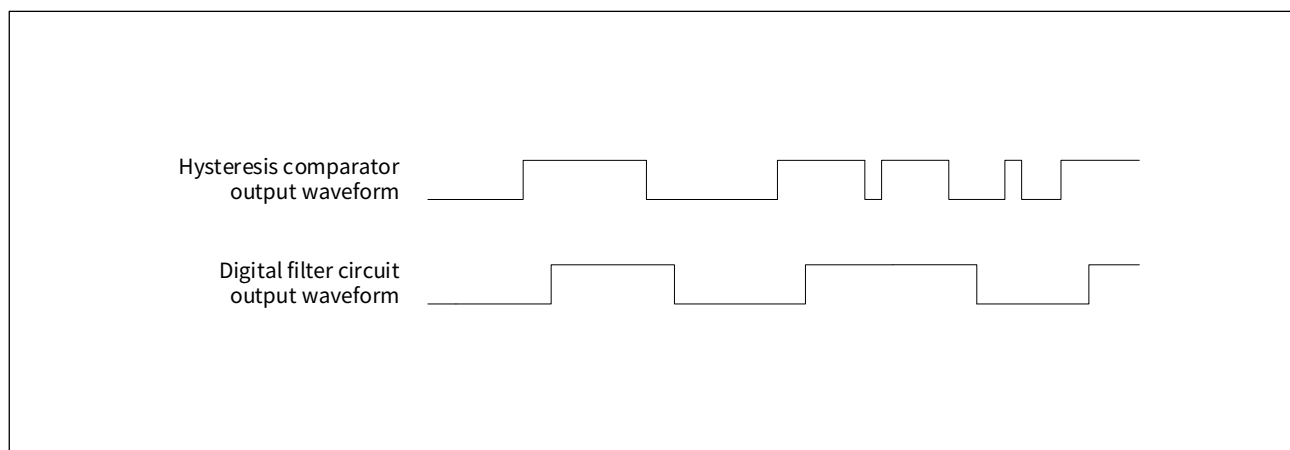
Table 22-1 LVD threshold voltage

LVD_CR0.VTH	Threshold voltage (unit: V)	LVD_CR0.VTH	Threshold voltage (unit: V)
000	1.8	100	3.4
001	2.2	101	3.8
010	2.6	110	4.2
011	3.0	111	4.6

22.3.3 Digital filter

In order to enhance the robustness of the system, the LVD supports the digital filtering function, which can digitally filter the output result signal of the LVD voltage comparison, and the signal smaller than the filtering width is filtered out without triggering an interrupt or reset. The specific waveform is shown in the following figure:

Figure 22-3 LVD filter output



The FLTCLK bit field of the control register LVD_CR0 is used to select the clock for digital filter:

- FLTCLK bit is 1, SYSCLK is selected as filter clock.
- FLTCLK bit is 0, and the built-in low speed RC oscillator clock LSI is selected as the filter clock.

The filter samples the output signal with a certain sampling frequency. When the continuous sampling reaches N same levels, the signal is valid, otherwise the signal is invalid, so as to filter out high-frequency clutter signals. The sampling clock of the filter unit is FLTCLK or the frequency division of FLTCLK, and the frequency division ratio of FLTCLK and the number of sampling points N can be selected through the FLTIME bit field of LVD_CR1 register.

From the FLTV bit field of the LVD status register LVD_SR, the signal level after LVD digital filter can be read out; When the function of GPIO is multiplexed as LVD_OUT, the digitally filtered signal can be output from GPIO to facilitate observation and measurement.

22.4 LVD interrupts

LVD supports working in low power mode, and interrupt output can wake up the chip from low power mode.

When the comparison result of the monitored voltage and the LVD threshold meets the trigger condition, an interrupt or reset signal can be generated. Whether to generate an interrupt or a reset signal is controlled by the ACTION bit field of the control register LVD_CR0:

- When ACTION is 1, LVD triggers a reset.
- When ACTION is 0, LVD triggers an interrupt.

Set the IE bit field of the control register LVD_CR1 to 1 to enable the LVD interrupt. When the trigger condition is met, an LVD interrupt will be generated. The interrupt flag bit LVD_SR.INTF will be set to 1 by hardware. The user can write 0 to the INTF bit to clear the interrupt flag.

Set the LEVEL, FALL, and RISE bit fields of the control register LVD_CR1 to select different interrupt or reset trigger methods. The three can be used in combination:

- LEVEL is 1, when the monitored voltage is lower than the threshold, an interrupt is triggered or a reset is generated.
- FALL is 1, the falling edge of the monitored voltage falls below the threshold to triggers an interrupt or generates a reset.
- RISE is 1, the rising edge of the monitored voltage rises above the threshold to trigger an interrupt or generate a reset.



22.5 Programming examples

22.5.1 Brown-out reset programming example

In this example, the MCU is reset when the monitored voltage falls below the threshold voltage, using digital filtering.

The configuration method is as follows:

Step 1: Configure LVD_CR0.SOURCE and select the voltage source to be monitored;

Step 2: Configure LVD_CR0.VTH and set the threshold voltage;

Step 3: Configure LVD_CR1.FLTTIME and select LVD filter time;

Step 4: Configure LVD_CR0.FLTCLK and select the filter clock;

Step 5: Set LVD_CR1.LEVEL to 1, and select to trigger LVD action when the monitored voltage is lower than the threshold;

Step 6: Set LVD_CR0.ACTION to 1, and select LVD trigger action as system reset;

Step 7: Set LVD_CR0.EN to 1 to enable LVD.

22.5.2 Interrupt programming example

In this example, an interrupt is generated when the monitored voltage is above or below a threshold voltage, using digital filtering.

The configuration method is as follows:

Step 1: Configure LVD_CR0.SOURCE and select the voltage source to be monitored;

Step 2: Configure LVD_CR0.VTH and select the threshold voltage;

Step 3: Configure LVD_CR1.FLTTIME and select LVD filter time;

Step 4: Configure LVD_CR0.FLTCLK and select the filter clock;

Step 5: Set both LVD_CR1.RISE and FALL to 1, and select rising and falling edge triggers;

Step 6: Set LVD_CR0.ACTION to 0, and select LVD trigger action as interrupt;

Step 7: Set LVD_CR1.IE to 1 to enable LVD interrupt;

Step 8: Enable the LVD interrupt in the NVIC interrupt vector table;

Step 9: Set LVD_CR0.EN to 1 to enable LVD;

Step 10: In the LVD initialization routine and the LVD interrupt service routine, write 0 to the LVD_ISR.INTF bit to clear the interrupt flag and allow a new LVD interrupt to be generated.



22.6 List of registers

LVD base address: LVD_BASE = 0x4000 00D0

Table 22-2 List of LVD registers

Register name	Register address	Register description
LVD_CR0	LVD_BASE + 0x00	Control register 0
LVD_CR1	LVD_BASE + 0x04	Control register 1
LVD_SR	LVD_BASE + 0x08	Status register



22.7 Register descriptions

See section [1 Documentation conventions](#) for abbreviations used in register descriptions.

22.7.1 LVD_CR0 Control register 0

Address offset: 0x00 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:9	RFU	-	Reserved bits, please keep the default value
8	FLTCLK	RW	Digital filter module filter clock setting 0: LSI 1: SYSCLK <i>Caution: Before entering DeepSleep mode, when the digital filter function is not required, please set the filter clock source to SYSCLK, otherwise the LSI will generate power consumption when running under DeepSleep mode.</i>
7	RFU	-	Reserved bits, please keep the default value
6:4	VTH	RW	Threshold voltage selection 000: 1.8V 001: 2.2V 010: 2.6V 011: 3.0V 100: 3.4V 101: 3.8V 110: 4.2V 111: 4.6V
3	RFU	-	Reserved bits, please keep the default value
2	SOURCE	RW	Monitor source configuration 0: VDD supply voltage 1: PA03 port input voltage
1	ACTION	RW	Trigger action configuration 0: NVIC interrupts 1: System resets
0	EN	RW	Enable control 0: LVD disabled 1: LVD enabled



22.7.2 LVD_CR1 Control register 1

Address offset: 0x04 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:8	RFU	-	Reserved bits, please keep the default value
7:4	FLTTIME	RW	Digital filter module filter time configuration 0000: No filter 0001: $f_{\text{SAMPLING}} = f_{\text{FLTCLK}} / 1$, $N=2$ 0010: $f_{\text{SAMPLING}} = f_{\text{FLTCLK}} / 1$, $N=4$ 0011: $f_{\text{SAMPLING}} = f_{\text{FLTCLK}} / 1$, $N=8$ 0100: $f_{\text{SAMPLING}} = f_{\text{FLTCLK}} / 2$, $N=6$ 0101: $f_{\text{SAMPLING}} = f_{\text{FLTCLK}} / 2$, $N=8$ 0110: $f_{\text{SAMPLING}} = f_{\text{FLTCLK}} / 4$, $N=6$ 0111: $f_{\text{SAMPLING}} = f_{\text{FLTCLK}} / 4$, $N=8$ 1000: $f_{\text{SAMPLING}} = f_{\text{FLTCLK}} / 8$, $N=6$ 1001: $f_{\text{SAMPLING}} = f_{\text{FLTCLK}} / 8$, $N=8$ 1010: $f_{\text{SAMPLING}} = f_{\text{FLTCLK}} / 16$, $N=5$ 1011: $f_{\text{SAMPLING}} = f_{\text{FLTCLK}} / 16$, $N=6$ 1100: $f_{\text{SAMPLING}} = f_{\text{FLTCLK}} / 16$, $N=8$ 1101: $f_{\text{SAMPLING}} = f_{\text{FLTCLK}} / 32$, $N=5$ 1110: $f_{\text{SAMPLING}} = f_{\text{FLTCLK}} / 32$, $N=6$ 1111: $f_{\text{SAMPLING}} = f_{\text{FLTCLK}} / 32$, $N=8$
3	FALL	RW	Triggered by the falling edge of the monitored voltage falling below the threshold (i.e., triggered by the rising edge of the LVD output signal) 0: Disabled 1: Enabled
2	RISE	RW	Triggered by the rising edge of the monitored voltage rising above the threshold (i.e., triggered by the falling edge of the LVD output signal) 0: Disabled 1: Enabled
1	LEVEL	RW	Triggered when the monitored voltage is lower than the threshold voltage (i.e., triggered by a high-level LVD output signal) 0: Disabled 1: Enabled
0	IE	RW	LVD interrupt enable control 0: Disabled 1: Enabled



22.7.3 LVD_SR Status register

Address offset: 0x08 Reset value: 0x0000 0000

Bit field	Name	Permission	Function description
31:2	RFU	-	Reserved bits, please keep the default value
1	FLTV	RO	Level value of digital filter output 0: Digital filter output low level 1: Digital filter output high level
0	INTF	RW0	Interrupt flags R0: LVD interrupt did not occur R1: LVD interrupt has occurred W0: LVD interrupt flag cleared W1: No function



23 Debug interface (DBG)

23.1 Overview

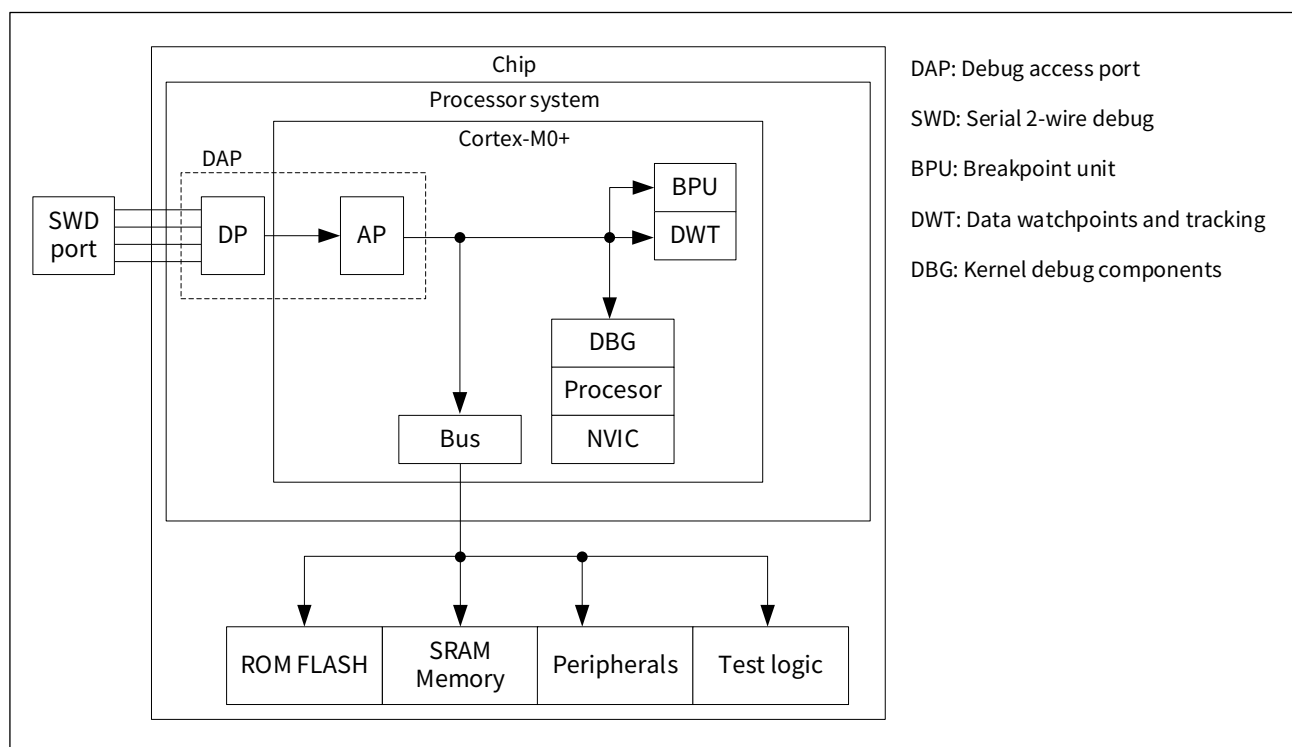
The core of CW32L010 is ARM® Cortex®-M0+, the core has built-in DAP hardware debug module, and supports SWD mode debugging. The hardware debug module can be suspended when fetching instructions (instruction breakpoints) or accessing data (data breakpoints), and the program stops running. query; and the core and peripherals can be restored, and program execution continues.

When using the debugging simulation tool to connect to the CW32L010 through the SWD interface, enter the debugging mode, and perform debugging operations through the DAP hardware debugging module in the chip core.

23.2 Serial wire debug port SWD

Use the SWD interface of the CW special debugger or the general debugging simulation tool to connect with the DAP debugging module inside the target chip, and carry out data exchange through the packet transmission protocol to realize the debugging operation. The SWD mode is a 2-wire serial communication, including a clock line SWCLK and a bidirectional data line SWDIO. As shown in the following figure:

Figure 23-1 SWD functional block diagram



The pin assignment of the SWD interface of the CW32L010 series is shown in the following table. The PA07/PA08 pin is the SWD function by default when the chip leaves the factory.

Table 23-1 SWD pin assignment

SWD port name	Pin function	Pin assignment
SWCLK	Serial clock input	PA08
SWDIO	Serial data input and output	PA07

The PA07/PA08 pins can be configured as SWD function or GPIO/ISP function, and the function configuration is performed by the SWDIO bit field of the system control register SYSCTRL_CR2. When SWDIO is 0, the PA07/PA08 pin is configured as SWD function; when SWDIO is 1, it is configured as GPIO/ISP function.

It is generally recommended to use a 10KΩ pull-up resistor for the SWD pin. When the PA07/PA08 of the CW32L010 is used as the SWD function, there is a built-in pull-up resistor with a resistance value between 55kΩ ~ 70kΩ. Users can add a pull-up resistor externally to improve the anti-interference performance.

The PA07/PA08 pin of the CW32L010 series is the SWD function by default. If the user has set the encryption level, it is necessary to judge whether the SWD function is supported according to the set level. The configuration and function of the SWD pin are shown in the following table:

Table 23-2 SWD pin function

Encryption level	SYSCTRL_CR2.SWDIO	PA07/PA08 function
Level0/1 encryption	0 (default)	SWD function
	1	GPIO/ISP function
Level2 encryption	0 (default)	NA, no function
	1	GPIO/ISP function
Level3 encryption	0 (default)	NA, no function
	1	GPIO function

Cautions :

1. The encryption level can be set through the ISP communication method. For details, please refer to the ISP communication protocol document;
2. After the chip is powered on, both SWDIO and SWCLK are internal pull-ups by default, and users do not need to connect external pull-up resistors;
3. When the chip encryption level is set to 2, the SWD function is disabled and can only be programmed by ISP;
4. When the chip encryption level is set to 3, the SWD function and the ISP function are disabled, and the chip cannot be programmed with new programs again.



23.3 Debug communication protocol

The debugger and the DAP debug module of the target chip communicate through the SWD packet transmission protocol. The packet transmission protocol is a 2-wire synchronous serial protocol, using the SWCLK clock signal and the SWDIO data signal:

- SWCLK is a one-way clock signal, which is output by the debugger to the target chip
- SWDIO is a bidirectional data signal, which is driven by the debugger and the target chip bidirectionally time-sharing

The protocol defines the conversion time of the transceiver and the transceiver with a length of one SWCLK cycle. During the conversion time of the transceiver and the target chip, neither the debugger nor the target chip drives SWDIO, and SWDIO is pulled up to a high level by a pull-up resistor.

When transmitting data on the SWDIO signal line, follow the principle that the lowest LSB is transmitted first, and the highest MSB is transmitted last.

Through the packet transfer protocol, the debugger can read and write access to the DP register and AP register in the DAP debug module of the target chip, hereinafter referred to as SW-DP and SW-AP.

23.3.1 Transmission protocol format

The DAP debug module includes the DP debug port register and the AP access port register. The communication between the debugger and the DAP debug module of the target chip is actually the read and write operations on the DP register and the AP register.

The transmission communication frame usually contains three fields:

- Package request
The length is 8-bit, the debugger to the target chip;
- Response
The length is 3-bit; the target chip to the debugger;
- Data transmission
The length is 33-bit, the target chip to the debugger or the debugger to the target chip, optional field.

The function definition of each request package is as follows:

Table 23-3 Definition of packet request bits

Bit	Name	Description
0	Start up	Must be 1
1	APnDP	0: DP access; 1: AP access
2	RnW	0: Write request; 1: Read request
3:4	A[3:2]	Address field of DP or AP register, bit3 is A2, bit4 is A3
5	Parity check	Bit parity bit of the previous 5-bit
6	Stop	0
7	Reside	Not driven by the master. Target chip reads as 1 due to pull-up



The request packet is always followed by the conversion time of the transceiver (default 1bit). At this time, neither the master nor the target drives SWDIO.

The response bits transmitted by the target are defined as follows:

Table 23-4 Definition of target transmit response bits

Bit	Name	Description
0:2	ACK	001: FAULT (bit0:0, bit1:0, bit2:1) 010: WAIT (bit0:0, bit1:1, bit2:0) 100: OK (bit0:1, bit1:0, bit2:0)

After the ACK response is the conversion time of the transceiver (default 1bit), at this time, neither the master nor the target will drive SWDIO.

Data transmitted by the debugger or target chip, the bits are defined as follows:

Table 23-5 Definition of data transmission bits

Bit	Name	Description
0:31	WDATA or RDATA	Write or read data
32	Parity check	Bit parity bit of 32-bit data

The data transmission phase is not necessary, and data transmission occurs only in the following two cases:

- The response received after the data read or write request is the OK response
- The ORUNDETECT flag bit of the DP-CTRL/STAT register is set to 1. At this time, no matter what kind of response (including WAIT and FAULT) is received, data transmission is required

23.3.2 SW-DP state machine

There is an ID CODE register inside SW-DP with a fixed value of 0x0BB11477 (Cortex®-M0+ identification code of ARM company). Before using the SWD packet protocol to read and write the target register, the ID number must be read to activate the SW-DP logic of the target chip, otherwise the SW-DP state machine of the target chip will not work.

The operation sequence is as follows:

Step 1: After power-on reset or after the SWDIO line is at a high level for more than 50 cycles, the SW-DP state machine enters the reset state;

Step 2: After entering the reset state, keep the SWDIO line at a low level for at least 2 cycles, and the SW-DP state machine enters the idle state;

Step 3: After entering the idle state, perform read access to the ID CODE register of DP-SW;

Caution:

If this operation is not performed, the target board will send a FAULT response in the subsequent packet communication response phase.

Step 4: Read and write access to the registers to be accessed according to the packet protocol.



23.3.3 Read and write access for SW-DP and SW-AP

- Write access for SW-DP/SW-DP
After the debugger receives the ACK, the data must be transmitted immediately after the conversion time of 1 clock cycle.
- Read access for SW-DP
 - When the target chip is ready to transmit data, it transmits an OK response, and then transmits data immediately;
 - When the target chip DAP is not ready for data transmission, it transmits a WAIT response to end the communication;
 - When the DAP status of the target chip is wrong, a FAULT response is transmitted to end the communication.
- Read access for SW-AP
The read access to the AP is registered, that is, the read AP operation cannot return the required result, but the result of the read operation cannot be returned until the next AP operation. If the next access to be performed is not an AP read access operation, the DP-RDBUFF register must be read to obtain the result of this read operation.
- The READOK flag in the DP-CTRL/STAT register of the M0 core's debug control/status register is updated every time a SW-AP read access or DP-RDBUFF read request is made to indicate whether the AP read access is successful.
- SW-DP has write buffers (for SW-DP or SW-AP writes) that can accept the next write operation while the read and write operations are not complete. If the SW-DP's write buffer is full, the chip SW-DP logic will reply with a WAIT response to notify the master to suspend the operation. The exception is special operations such as IDCODE read, DP-CTRL/STAT read or ABORT write operations, which are also accepted when the write buffer is full.
- Since SWCLK and HCLK are not synchronous clocks and belong to the asynchronous clock domain, two additional SWCLK cycles are required after the write operation (after the parity bit of the data transmission) to ensure that the write operation takes effect. The SWDIO line should be driven low (idle state) while the master is driving these 2 SWCLK cycles. This is especially important when writing power-on request operations to the DP-CTRL/STAT registers, otherwise the next operation (one that is valid after the core is powered on) will be executed immediately and will fail.



23.3.4 SW-DP register

When APnDP is 0, access the DP register, which is addressed by A[3:2]. When the address is the same, the meaning of the register may be different due to different read and write operations, as shown in the following table:

Table 23-6 List and definition of DP registers

A[3:2]	Read/write	R register name	Register content description
00	Read	IDCODE	Fixed at 0x0BB1 1477 (to identify SW-DP), ARM Cortex [®] -M0+ identification code
	Write	ABORT	See Table 23-7 Definition of DP ABORT register bits
01	Read/write	DP-CTRL/STAT (SELECT.CTRLSEL = 0)	Mainly used for: 1. Request system or debug power-on; 2. Configure transfer operation for AP access; 3. Control comparison and verification operations; 4. Read some status flags (overflow and power-up acknowledgment)
		WIRE CONTROL (SELECT.CTRLSEL = 1)	Used to configure the physical serial port protocol (such as the duration of transition times, etc.)
10	Read	READ RESEND	Allow recovery of read data from corrupted debug transfers without repeating the original AP transfer.
	Write	SELECT	Used to select the current access AP port and the 4-word register BANK in the AP port. See Table 23-8 DP SELECT register
11	Read/write	READ BUFFER	Since an AP access has been issued, this read buffer is useful (providing the result of reading this AP request when executing the next AP transaction). This read buffer captures the data in the AP, displayed as the result of the previous read, without starting a new operation.

Table 23-7 Definition of DP ABORT register bits

Bit	Name	Description
31:5	RESERVE	
4	ORUNERRCLR	Write 1 to clear the STICKYORUN error flag
3	WDATAERR	Write 1 to clear the WDATAERR error flag
2	STICKYERR	Write 1 to clear the STICKYERR error flag
1	STICKYCMP	Write 1 to clear the STICKYCMP flag
0	DAPABORT	Write 1 to generate the DAP ABORT signal and abandon the current access operation; when the target chip responds to the WAIT response, this operation must be performed to abort this operation.



Table 23-8 DP SELECT register

Bit	Name	Description
31:24	APSEL	Select the current AP port, fixed as b00000000
23:8	RESERVE	
7:4	APBANKSEL	Select the active 4-word register BANK on the current AP
3:1	RESERVE	
0	CTRLSEL	DP port register selection: The default is 0 after reset, select the CTRL/STAT register; Set to 1 to select the WCR register (Wire Control Register)

23.3.5 SW-AP register

When APnDP is 1, access the AP register. Since there are many SW-AP registers, it is necessary to combine A[3:2] and the APBANKSEL bit field value of the SW-DP selection register SELECT to uniquely address the SW-AP register.

This article will not go into details about SW-AP one by one, but only introduces a common IDR public register (address 0xFC), as shown in the following table:

Table 23-9 AP IDR public register

Bit	Definition
31:28	AP designed version Rrevision.
27:24	AP designer identification code, this chip is fixed to 0x04.
23:17	AP designer identification code, this chip is fixed to 0x3B.
16:13	AP type. For example, the memory access port type identification code is b1000, and the undefined type access port type identification code is b0000.
12:8	Reserved
7:0	AP ID. Such as MEM-AP or JTAG-AP and so on.



23.4 Kernel debugging

The core can be debugged by manipulating the core debug registers, and the debugger accesses these registers through DAP debugging.

The kernel debug registers mainly include 4 registers, as shown in the following table:

Table 23-10 Kernel debug registers

Register	Description
DHCSR	Debug suspend control and status register, 32-bit; Control processor suspend, single step and restart actions.
DCRSR	Debug kernel register selector register, 17-bit; Controls reading and writing of kernel registers during suspend.
DCRDR	Debug kernel register data register, 32-bit; Data transfer register for reading and writing kernel registers during pause.
DEMCR	Debug exception monitoring control register, 32-bit; Used to enable data watchpoint unit and vector capture feature, using vector capture, the debugger can suspend the processor when the processor is reset or a hardware error occurs.

These registers can only be reset by a power-on reset. See 《Cortex®-M0+ Technical Reference Manual》 for more details.

23.5 Breakpoint unit BPU

The Cortex®-M0+ BPU breakpoint unit provides four breakpoint registers to implement the PC pointer-based breakpoint function.

For more information on the identification registers and access types of the BPU CoreSight component, see 《ARMv6-M Architecture Reference Manual》 and the ARM® CoreSight Component Technical Reference Manual.

23.6 Data observation points and tracking DWT

Cortex®-M0+ DWT provides two observation point register sets. Realize the following functions:

- Set data watchpoints:
Data or peripheral addresses can be marked as watch variables, and access to this address will generate a debug event and suspend program execution.
- Optional DWT program counter sampling register (DWT_PCSR) feature in ARMv6-M. Allows the debugger to periodically sample the PC pointer, providing a rough analysis without stopping the processor.

See 《ARMv6-M Architecture Reference Manual》 for more information.



23.7 Debug components DBG

The debugger implements clock control support for peripherals such as timers, watchdogs, and RTCs during breakpoints through the debug components DBG. Through the setting of the debug state timer control register SYSCTRL_DEBUG, you can control the timer, watchdog timer, RTC, etc. to run normally or pause during the breakpoint in the debug state, as shown in the following table:

Table 23-11 Debug status timer control register SYSCTRL_DEBUG

Bit	Name	Vaule	Counter function configuration during breakpoints in debug state
10	WWDT	1	In debug state, the WWDT counter pauses counting
		0	In debug state, the WWDT counter counts normally
9	IWDT	1	In debug state, the IWDT counter pauses counting
		0	In debug state, the IWDT counter counts normally
8	RTC	1	In debug state, the RTC counter pauses counting
		0	In debug state, the RTC counter counts normally
6	LPTIM	1	In debug state, the LPTIM counter pauses counting
		0	In debug state, the LPTIM counter counts normally
5	BTIM123	1	In debug state, the BTIM1, BTIM 2, BTIM 3 counters pause counting
		0	In debug state, the BTIM1, BTIM 2, BTIM 3 counters count normally
1	GTIM	1	In debug state, the GTIM counter pauses counting
		0	In debug state, the GTIM counter counts normally
0	ATIM	1	In debug state, the ATIM counter pauses counting
		0	In debug state, the ATIM counter counts normally

For example, when the timer outputs PWM for motor control, the timer cannot be stopped, otherwise the motor will be damaged. For another example, the watchdog timer should stop counting during the breakpoint to prevent an undesired reset of the system.



23.8 Cautions

CW32L010 needs to use HCLK for debugging connection during debugging. It is not allowed to turn off HCLK during the debugging session. Therefore, executing WFI in the debugging environment will turn off the kernel HCLK, causing the debugging function to fail.

If the target chip has entered the DeepSleep mode, the DAP hardware debugging module does not work at this time, the debugger cannot connect with the DAP hardware debugging module of the target chip, and the debugging function cannot be realized.

The user must ensure that the chip is in Active mode or Sleep mode to use the debugger for debugging.

Therefore, it is recommended that users complete all functional debugging of the device in non-DeepSleep mode before merging the relevant code of DeepSleep, and then debug the code related to DeepSleep.



24 Digital signature

24.1 Overview

The digital signature is mainly used to store information such as the unique identification (UID) of the chip, product model, FLASH capacity, SRAM capacity, and the number of pins in the chip package, which can be read by SWD or CPU. The digital signature-related information is programmed at the factory, and user firmware or external devices can verify the legitimacy of the chip by reading the digital signature.

24.2 Product unique identification (UID) register (80bit)

The UID register stores the unique identifier of the chip, and its address is 0x0010 07B0 - 0x0010 07B9, a total of 80 bits. The UID is written during chip production and cannot be modified by the user. UID registers support reading in single-byte/half-word/full-word, etc., and then concatenated using custom algorithms.

Typical application scenarios of unique identifiers:

- Used as device serial number
- Device legitimacy verification to prevent piracy
When the device is produced, the user uses the private key to encrypt the UID, and stores the calculation result in the main FLASH memory or OTP memory. If the result is compared with the previously stored calculation result, it is considered that the device is legal, otherwise the program will not start, which can effectively prevent the user's device from being illegally copied (piracy).
- Used as a security key
Combined with UID and private algorithm, users can perform security verification before users program FLASH to improve the security of the code in FLASH.
- Activate the secure boot process, etc.

24.3 Product model register

The product model register stores the ASCII code of the product model, and its address is 0x0010 07E0 - 0x0010 07EF, with a total of 16 bytes. When the product model is less than 16 bytes, it is filled with 0x00.

For example, the model of the chip is CW32L010F8U6, the corresponding data stored (starting from the low address) is: 0x43 0x57 0x33 0x32 0x4C 0x30 0x31 0x30 0x46 0x38 0x55 0x36 0x00 0x00 0x00 0x00.

Caution:

If there are letters in the product model, the letters need to be capitalized.

24.4 FLASH capacity register

The FLASH capacity register stores the capacity of the built-in FLASH memory of the chip, and its address is 0x0010 07D8 - 0x0010 07DB, a total of 4 bytes.

The FLASH capacity size read from the FLASH capacity register is in bytes, such as 0x8000 for 32KB.



24.5 SRAM capacity register

The SRAM capacity register stores the capacity of the built-in SRAM memory in the chip, and its address is 0x0010 07D6 - 0x0010 07D7, a total of 2 bytes.

The SRAM capacity size read from the SRAM capacity register is in bytes, such as 0x1000 for 4KB.

24.6 Pin number register

The pin number register stores the chip pin number, its address is 0x0010 07D5, a total of 1 bytes. For example, 0x14 represents 20Pin.



25 Revision history

Table 25-1 Document revision history

Date	Revision	Changes
September 26, 2024	Rev 1.0	Initial release.

