

HC32F420 Series

32-bit ARM® Cortex®-M4 Microcontroller

Reference Manual

Rev1.30 August 2026

Statement

- ★ Xiaohua Semiconductor Co., Ltd. (hereinafter referred to as "XHSC") reserves the right to change, correct, enhance, modify Xiaohua Semiconductor products and/or this document at any time without prior notice. Users can get the latest information before placing orders. XHSC products are sold in accordance with the terms and conditions of sale set forth in the Basic Contract for Purchase and Sales.
- ★ It is the customer's responsibility to select the appropriate XHSC product for your application and to design, validate and test your application to ensure that your application meets the appropriate standards and any safety, security or other requirements. The customer shall be solely responsible for this.
- ★ XHSC hereby acknowledges that no intellectual property license has been granted by express or implied means.
- ★ The resale of XHSC Products shall be invalidated by any warranty commitment of XHSC to such Products if its terms are different from those set forth herein.
- ★ Any graphics or words marked “®” or “™” are trademarks of XHSC. All other products or services shown on XHSC products are the property of their respective owners.
- ★ Important Notice: Before using this product, be sure to consult the latest official Errata Sheet to fully understand the product's known corrections and avoid usage risks.
- ★ Information in this notification replaces and replaces information in previous versions.

©2026 Xiaohua Semiconductor Co., Ltd. All Rights Reserved

Contents

Statement	2
Contents	3
Table Index.....	17
Figure Index	20
Overview.....	25
1 System Structure.....	26
1.1 Overview	26
1.2 System Architecture Diagram.....	26
1.3 Memory and peripheral address allocation.....	27
2 Operating mode.....	29
2.1 Overview of working mode.....	29
2.2 Working mode switching	29
2.3 Active Mode.....	30
2.4 Sleep Mode.....	30
2.5 Deep sleep Mode.....	31
2.6 Functional module working status.....	32
2.7 Register description.....	33
2.7.1 register list.....	33
2.7.2 System Control Register (SCB_SCR).....	33
3 System Controller (SYSCTRL).....	34
3.1 System Clock Introduction.....	34
3.1.1 Clock Architecture Diagram	35
3.1.2 Internal high speed RC clock RCH	35
3.1.3 Internal low speed RC clock RCL	36
3.1.4 External crystal oscillator clock XTH	36
3.1.5 Phase-locked loop clock PLL	36
3.1.6 Clock start process	37
3.2 System Clock Switching	37
3.2.1 Standard Clock Switching Process	38
3.2.2 RCHCLKD switching process between different output frequencies.....	40
3.2.3 Switching from other clocks to RCL example	40
3.2.4 Switching from other clocks to RCHCLKD example	40
3.2.5 Example of switching between PLL (reference clock is RCHCLKD) and RCHCLKD....	41
3.3 On-chip peripheral clock control.....	42
3.4 Interrupt Wakeup Control	42

3.4.1	Enable the NVIC corresponding to the module that needs to wake up the processor	43
3.4.2	Method not to execute interrupt service routine after wake-up from deep-sleep mode	43
3.4.3	Using exit hibernation feature	44
3.5	Register description.....	45
3.5.1	Register list.....	45
3.5.2	System Control Register 0 (SYSCTRL_CR0)	46
3.5.3	System Control Register 1 (SYSCTRL_CR1)	47
3.5.4	Clock Enable Control Register 2 (SYSCTRL_CR2)	48
3.5.5	Sequence Control Register 3 (SYSCTRL_CR3)	49
3.5.6	RCH Control Register (SYSCTRL_RCHCR).....	49
3.5.7	RCL Control Register (SYSCTRL_RCLCR).....	50
3.5.8	XTH Control Register (SYSCTRL_XTHCR).....	51
3.5.9	PLL Control Register (SYSCTRL_PLLCR).....	52
3.5.10	RCH Trim Value Register (SYSCTRL_RCHTRIM)	53
3.5.11	RCL Trim Value Register (SYSCTRL_RCLTRIM)	53
3.5.12	Clock Status Register (SYSCTRL_STATE)	54
3.5.13	APB0 Peripheral Clock Control Register (SYSCTRL_APBEN0).....	55
3.5.14	APB1 Peripheral Clock Control Register (SYSCTRL_APBEN1).....	56
3.5.15	AHB0 Peripheral Clock Control Register (SYSCTRL_AHBEN0).....	56
3.5.16	AHB1 Peripheral Clock Control Register (SYSCTRL_AHBEN1).....	57
3.5.17	Timer Debug Halt Control Register (SYSCTRL_DEBUG)	58
3.5.18	GPIO Auxiliary Control Register 0 (SYSCTRL_IOAUX0).....	59
3.5.19	GPIO Auxiliary Control Register 1 (SYSCTRL_IOAUX1).....	60
3.5.20	GPIO Auxiliary Control Register 2 (SYSCTRL_IOAUX2).....	60
4	Reset Controller (RESET)	61
4.1	Reset Controller Introduction.....	61
4.1.1	Power-on and power-off reset POR PDR	62
4.1.2	External reset pin reset.....	62
4.1.3	WDT reset.....	62
4.1.4	LVD low voltage reset	62
4.1.5	Cortex-M4 SYSRESETREQ reset.....	62
4.1.6	Cortex-M4 LOCKUP reset.....	62
4.2	Register description.....	63
4.2.1	Register list.....	63
4.2.2	APB0 Peripheral Reset Control Register (SYSCTRL_APBRS0).....	64
4.2.3	APB1 Peripheral Reset Control Register (SYSCTRL_APBRS1).....	65

4.2.4	AHB0 Peripheral Reset Control Register (SYSCTRL_AHBRST0)	65
4.2.5	AHB1 Peripheral Reset Control Register (SYSCTRL_AHBRST1)	66
4.2.6	Reset Flag Register (SYSCTRL_RSTFLAG)	67
5	Interrupt Controller (NVIC)	67
5.1	Overview	68
5.2	Interrupt priority	69
5.3	Interrupt digraph	70
5.4	Interrupt Input and Suspend Behavior	71
5.5	Interrupt wait	73
5.6	Interrupt source	74
5.7	Interrupt structure	76
5.8	Basic operation of the software	76
5.8.1	External interrupt enable	76
5.8.2	NVIC interrupt enable and clear enable	76
5.8.3	NVIC interrupt pending and clear pending	76
5.8.4	NVIC interrupt priority	77
5.8.5	NVIC interrupt mask	77
5.9	Register description	78
5.9.1	Register list	78
5.9.2	Interrupt Enable Setting Register0 (NVIC_ISER0)	79
5.9.3	Interrupt Enable Setting Register 1 (NVIC_ISER1)	79
5.9.4	Interrupt Enable Clear Register 0 (NVIC_ICER0)	80
5.9.5	Interrupt Enable Clear Register 1 (NVIC_ICER1)	80
5.9.6	Interrupt Pending Setup Register 0 (NVIC_ISPR0)	81
5.9.7	Interrupt Pending Setup Register 1 (NVIC_ISPR1)	81
5.9.8	Interrupt Pending Clear Register 0 (NVIC_ICPR0)	82
5.9.9	Interrupt Pending Clear Register 1 (NVIC_ICPR1)	82
5.9.10	Interrupt Priority Register (NVIC_IPRx) (x=0~15)	83
5.9.11	Interrupt Mask Register (PRIMASK)	83
6	General-Purpose Input-Output Port Controller (GPIO)	84
6.1	Introduction	84
6.2	Main characteristics	84
6.3	Functional description	85
6.3.1	Port Circuit Block Diagram	85
6.3.2	Port Mode Configuration	85
6.3.3	Port Reset Status	86
6.3.4	Port emulation function	86
6.3.5	Port general-purpose input and output functions	87

6.3.6	Port multiplexing function.....	90
6.3.7	Port clock output.....	93
6.3.8	Port external interrupt	94
6.4	Register description.....	95
6.4.1	Register list.....	95
6.4.2	Port Digital-to-Analog Configuration Register (GPIO_ADS)	96
6.4.3	Port input and output direction configuration register (GPIO_DIR).....	96
6.4.4	Port Output Type Register (GPIO_OD)	97
6.4.5	Port Output Drive Capability Register (GPIO_DR).....	97
6.4.6	Port pull-up and pull-down register (GPIO_PUPD).....	98
6.4.7	Port Data Input Register (GPIO_IN)	98
6.4.8	Port Data Output Register (GPIO_OUT)	99
6.4.9	Port Reset Register (GPIO_BRR)	99
6.4.10	Port Set Reset Register (GPIO_BSRR)	99
6.4.11	Port Alternate Function Low Register (GPIO_AFRL)	100
6.4.12	Port Alternate Function High Register (GPIO_AFRH).....	100
6.4.13	Port High Interrupt Enable Configuration Register (GPIO_HIE)	101
6.4.14	Port Low Level Interrupt Enable Configuration Register (GPIO_LIE)	101
6.4.15	Port Rising Edge Interrupt Enable Configuration Register (GPIO_RIE).....	101
6.4.16	Port Falling Edge Interrupt Enable Configuration Register (GPIO_FIE).....	102
6.4.17	Port Interrupt Status Register (GPIO_IFR).....	102
6.4.18	Port Interrupt Clear Register (GPIO_ICR)	103
7	I2C -bus (I2C).....	104
7.1	Overview	104
7.2	Main characteristics	104
7.3	Protocol description.....	105
7.3.1	Data transmission on the I2C bus.....	105
7.3.2	Acknowledgment on the I2C bus	106
7.3.3	Arbitration on the I2C bus.....	107
7.4	Functional description	108
7.4.1	Functional block diagram.....	108
7.4.2	Serial clock generator.....	109
7.4.3	Input filter	109
7.4.4	Address comparator.....	109
7.4.5	Response flag	110
7.4.6	Interrupt generator	110
7.4.7	Operating mode.....	110
7.4.8	Status code expression.....	115

7.5	Programming example	117
7.5.1	Master sending example.....	117
7.5.2	Master receive example.....	118
7.5.3	Slave receiving example.....	119
7.5.4	Slave sending example.....	120
7.6	Register description.....	121
7.6.1	Register list.....	121
7.6.2	Baud Rate Generator Enable Register (I2C_BRREN)	122
7.6.3	Baud Rate Counter Configuration Register (I2C_BRR).....	122
7.6.4	Configuration Register (I2C_CR).....	123
7.6.5	Data Register (I2C_DR)	123
7.6.6	Status Register (I2C_STAT).....	124
7.6.7	Slave Address 0 Register (I2C_ADDR0)	124
7.6.8	Slave Address 1 Register (I2C_ADDR1)	124
7.6.9	Slave Address 2 Register (I2C_ADDR2)	125
7.6.10	Slave Address Match Register (I2C_MATCH).....	125
8	Serial Peripheral Interface (SPI)	126
8.1	Overview	126
8.2	Main features.....	126
8.3	Functional description	126
8.3.1	Functional block diagram.....	126
8.3.2	Communication Format and Timing.....	127
8.3.3	Slave Select Pin Configuration	128
8.3.4	Full duplex communication	128
8.3.5	Single-wire half-duplex communication	129
8.3.6	Simplex communication	129
8.3.7	Multi-machine communication.....	131
8.3.8	Host Mode Delayed Sampling	132
8.3.9	Status Flags and Error Flags	133
8.3.10	Interrupt generation and clearing.....	137
8.4	Programming example	138
8.4.1	Example of sending and receiving data in full duplex by the host.....	138
8.4.2	Example of sending and receiving data in full duplex from the slave.....	138
8.4.3	Single-wire half-duplex sending and receiving data example	139
8.5	Register description.....	141
8.5.1	Register list.....	141
8.5.2	SPI Control Register 0 (SPI_CR0)	142
8.5.3	SPI Control Register 1 (SPI_CR1)	143

8.5.4	SPI Single-Wire Half-Duplex Output Enable Register (SPI_HDOE)	143
8.5.5	SPI Internal Slave Select Register (SPI_SSI)	144
8.5.6	SPI Status Register (SPI_SR).....	144
8.5.7	SPI Interrupt Clear Register (SPI_ICR)	145
8.5.8	SPI Data Register (SPI_DR).....	145
9	Clock Trim Module (CTRIM)	146
9.1	Overview	146
9.2	Main characteristics	146
9.3	Functional description	146
9.3.1	RCH clock calibration mode	146
9.3.2	RCL clock calibration mode.....	147
9.3.3	Low Power Timer Mode	149
9.4	Software operation process	149
9.4.1	RCH Clock Automatic Calibration Example	149
9.4.2	RCL Clock Automatic Calibration Example	150
9.5	Register description.....	151
9.5.1	Register list.....	151
9.5.2	Autoload Register (CTRIM_ARR).....	152
9.5.3	Error Counter Register (CTRIM_CNT).....	152
9.5.4	Control Register 0 (CTRIM_CR0).....	153
9.5.5	Control Register 1 (CTRIM_CR1).....	154
9.5.6	Interrupt Enable Register (CTRIM_IER).....	154
9.5.7	Interrupt and Status Register (CTRIM_ISR).....	155
9.5.8	Interrupt Status Clear Register (CTRIM_ICR)	155
9.5.9	Error Result Register (CTRIM_FCAP)	156
9.5.10	TRIM result register (CTRIM_TVAL)	156
9.5.11	Error Upper Limit Register (CTRIM_FLIM)	157
10	FLASH Controller (FLASH)	158
10.1	Overview	158
10.2	Capacity division	158
10.3	Read wait period	158
10.4	FLASH operation (read, write, erase).....	159
10.4.1	Page Erase (Sector Erase).....	159
10.4.2	Full Chip Erase (Chip Erase).....	160
10.4.3	Write operation (Program)	160
10.4.4	Read operation (Read)	162
10.4.5	Cache function.....	162
10.5	FLASH security protection	163

10.5.1	Page Erase Protection	163
10.5.2	PC address erase and write protection	163
10.5.3	Register write protection	163
10.5.4	Data Readout Protection	164
10.6	Register description.....	165
10.6.1	Register list.....	165
10.6.2	Control Register (FLASH_CR).....	166
10.6.3	Wait period setting (FLASH_WAIT)	166
10.6.4	LOCK status register (FLASH_LOCKSTATE)	166
10.6.5	CACHE control register (FLASH_CACHE)	167
10.6.6	Interrupt Enable Register (FLASH_IER).....	167
10.6.7	Interrupt Flag Register (FLASH_IFR).....	167
10.6.8	Interrupt Flag Clear Register (FLASH_ICR)	168
10.6.9	Bypass sequence register (FLASH_BYPASS)	168
10.6.10	Sector Erase and Write Protection Register (FLASH_SLOCK0).....	168
10.6.11	Sector Erase and Write Protection Register (FLASH_SLOCK1).....	169
11	RAM Controller (RAM)	170
11.1	Overview	170
11.2	Functional description	170
11.2.1	RAM address range	170
12	DMA Controller (DMAC)	171
12.1	Introduction to DMAC	171
12.2	DMAC main characteristics.....	171
12.3	DMA functional block diagram.....	172
12.4	Functional description	173
12.4.1	DMAC transfer modes	173
12.4.2	DMA software block transfer mode.....	173
12.4.3	DMA software Burst transfer mode.....	174
12.4.4	DMA hardware block transfer mode.....	175
12.4.5	DMA hardware Burst transfer mode.....	176
12.4.6	DMA transfer paused	177
12.4.7	DMA other configuration.....	177
12.4.8	DMA Interrupt	178
12.5	Registerdescription.....	179
12.5.1	Register list.....	179
12.5.2	All Channel Configuration Register (DMAC_CONF)	180
12.5.3	Channel x Configuration A Register (DMAC_CONFAx)	181
12.5.4	Channel x Configuration B Register (DMAC_CONFBx).....	183

12.5.5	Channel x Transfer Source Address Register (DMAC_SRCADR _x)	184
12.5.6	Channel x Destination Address Register (DMAC_DSTADR _x)	184
13	Composite Timer (CTIM)	185
13.1	Overview	185
13.1.1	BTIM main characteristics.....	186
13.1.2	GTIM main characteristics.....	186
13.2	BTIM function description	187
13.2.1	Functional block diagram.....	187
13.2.2	Filter unit.....	187
13.2.3	Counting unit	188
13.2.4	Timer mode	189
13.2.5	Counter mode.....	189
13.2.6	Trigger start mode	190
13.2.7	Gating mode	191
13.2.8	Timer Cascading	192
13.2.9	Interrupt.....	192
13.3	GTIM function description.....	193
13.3.1	Functional block diagram.....	193
13.3.2	filter unit.....	193
13.3.3	Counting unit	194
13.3.4	Timer mode	195
13.3.5	Counter mode.....	195
13.3.6	Trigger start mode	196
13.3.7	Gating mode	197
13.3.8	Compare capture function	198
13.3.9	DMA trigger	199
13.3.10	Interrupt.....	200
13.3.11	On-chip peripheral interconnection	200
13.4	BTIM register description.....	201
13.4.1	Register list.....	201
13.4.2	Reload register (BTIM _x _ARR) (x=0~8).....	202
13.4.3	Count register (BTIM _x _CNT) (x=0~8)	202
13.4.4	Control Register (BTIM _x _CR) (x=0~8)	203
13.4.5	Interrupt enable (BTIM _x _IER) (x=0~8).....	204
13.4.6	Interrupt Flag Register (BTIM _x _IFR) (x=0~8).....	204
13.4.7	Interrupt Flag Clear Register (BTIM _x _ICR) (x=0~8)	205
13.4.8	Composite Interrupt Flag Register (BTIM_AIFR)	205
13.4.9	Composite Interrupt Flag Clear Register (BTIM_AICR).....	206

13.5	GTIM register description.....	207
13.5.1	Register list.....	207
13.5.2	Reload Register (GTIM_ARR)	208
13.5.3	Count register (GTIM_CNT).....	208
13.5.4	Control Register 1 (GTIM_CR1).....	209
13.5.5	Control Register 0 (GTIM_CR0).....	210
13.5.6	Interrupt Enable Control Register (GTIM_IER)	211
13.5.7	Interrupt Flag Register (GTIM_IFR)	212
13.5.8	Interrupt Flag Clear Register (GTIM_ICR).....	213
13.5.9	Compare Capture Control Register (GTIM_CMMR).....	214
13.5.10	Compare Capture Register (GTIM_CCRy) (y=0,1,2,3).....	214
14	Advanced Timer (ATIM)	215
14.1	Overview	215
14.2	Main characteristics	215
14.2.1	Main features of ATIM3	215
14.2.2	Main features of ATIM0	217
14.3	Functional description	218
14.3.1	Timer clock	218
14.3.2	Timer counter	218
14.3.3	Timer Prescaler	218
14.3.4	Mode 0 count timer function.....	219
14.3.5	Mode 1 pulse width measurement PWC.....	222
14.3.6	Mode 2/3 compare capture mode.....	226
14.3.7	Mode 2/3 Slave Mode.....	248
14.3.8	Quadrature code counting function	250
14.3.9	Timer triggers ADC	252
14.3.10	Brake control	253
14.3.11	Interrupt.....	253
14.3.12	Timer interconnection.....	253
14.3.13	input interconnection.....	253
14.4	Register description.....	255
14.4.1	Register list.....	255
14.4.2	Mode 0 Register Description	256
14.4.3	Mode 1 Register Description	259
14.4.4	Mode 2,3 register description	265
15	Independent Watchdog (IWDG)	280
15.1	Overview	280
15.2	Main characteristics	280

15.3	Functional description	280
15.3.1	Functional block diagram.....	280
15.3.2	Window watchdog mode.....	280
15.3.3	Independent Watchdog Mode	281
15.3.4	Watchdog overflow handling.....	281
15.3.5	Watchdog timeout	281
15.3.6	Watchdog stop and run again	281
15.3.7	Register write protection	281
15.4	Programming example	282
15.4.1	Configure IWDG as window watchdog.....	282
15.4.2	Configure IWDG as an independent watchdog	282
15.4.3	Watchdog timeout (10kHz independent clock).....	283
15.5	Register description.....	284
15.5.1	Register list.....	284
15.5.2	Key-value register (IWDG_KR)	285
15.5.3	Control Register (IWDG_CR)	285
15.5.4	Reload Register (IWDG_ARR).....	286
15.5.5	Status Register (IWDG_SR).....	286
15.5.6	Window Register (IWDG_WINR)	287
16	Window Watchdog (WWDT)	288
16.1	Overview	288
16.2	Main characteristics	288
16.3	Functional description	288
16.3.1	Functional block diagram.....	288
16.3.2	enable watchdog	289
16.3.3	Configure watchdog timeout	289
16.3.4	Update down counter	289
16.3.5	Pre-flow interrupt	289
16.4	Programming example	289
16.5	Register description.....	290
16.5.1	Register list.....	290
16.5.2	Control Register 0 (WWDT_CR0)	291
16.5.3	Control Register 1 (WWDT_CR1)	291
16.5.4	Status Register(WWDT_SR).....	292
17	Universal Synchronous/ Asynchronous Receiver Transmitter (USART)	293
17.1	Overview	293
17.2	Main characteristics	293
17.2.1	Asynchronous Serial Port Features.....	293

17.2.2	Synchronous Serial Port Features	293
17.2.3	Smart Card Interface Features	293
17.2.4	Infrared Communication Features.....	294
17.2.5	LIN Characteristic	294
17.3	Functional description	295
17.3.1	Communication baud rate programming	295
17.3.2	Auto baud rate detection	301
17.3.3	DMA continuous communication	301
17.3.4	Hardware flow control	302
17.3.5	Multi-function timer counter	304
17.3.6	Interrupt.....	306
17.3.7	Asynchronous serial port mode	307
17.3.8	Synchronous serial port mode	313
17.3.9	Smart Card (7816) Mode	319
17.3.10	infrared mode	323
17.3.11	LIN mode	327
17.4	Registerdescription.....	333
17.4.1	Register list.....	333
17.4.2	Control Register 1 (USART_CR1)	334
17.4.3	Control Register 2 (USART_CR2)	336
17.4.4	Control Register 3 (USART_CR3)	338
17.4.5	Baud rate register (USART _ BRR).....	340
17.4.6	Timing counter register(USART_TOR).....	340
17.4.7	Status register (USART _ SR).....	341
17.4.8	Transmit Data Register(USART_TDR).....	343
17.4.9	Interrupt Enable Register(USART_IER)	344
17.4.10	Interrupt Flag Clear Register(USART_ICR)	345
17.4.11	Receive Data Register(USART_RDR)	346
17.4.12	Address Register(USART_ADDR)	346
17.4.13	Address Mask Register(USART_ADDMR)	346
17.4.14	Baud rate detection value register(USART_BDR)	347
17.4.15	Infrared Modulation Register(USART_IR)	347
18	Cyclic redundancy check (CRC)	348
18.1	Overview	348
18.2	Main characteristics	348
18.3	Functional description	349
18.3.1	Operating mode.....	349
18.3.2	Encoding	349

18.3.3	Write bit width	349
18.4	Programming example	350
18.4.1	CRC-16 encoding mode	350
18.4.2	CRC-16 check mode.....	350
18.4.3	CRC-32 encoding mode	350
18.4.4	CRC-32 check mode.....	350
18.5	Registerdescription.....	351
18.5.1	Register list.....	351
18.5.2	Control and Status Register (CRC_CSR)	352
18.5.3	Results register (CRC_RESULT).....	352
18.5.4	Data Register (CRC_DR).....	352
19	Analog-to-Digital Converters (ADC).....	353
19.1	Module Introduction.....	353
19.2	ADC block diagram.....	353
19.3	Conversion Timing and Conversion Speed	354
19.4	Single conversion mode	354
19.5	Scan conversion mode	356
19.5.1	Sequential Scan Single Conversion Mode	356
19.5.2	Sequential Scan Continuous Conversion Mode.....	359
19.5.3	Sequential scan discontinuous conversion mode	359
19.5.4	Scan conversion triggers DMA read.....	359
19.6	Continuous Conversion Accumulation Mode.....	359
19.7	ADC conversion external trigger source	362
19.8	ADC conversion result comparison.....	363
19.9	ADC interrupt	363
19.10	Measure ambient temperature using a temperature sensor	364
19.11	Registerdescription.....	366
19.11.1	Register list.....	366
19.11.2	ADC Analog Device Select Register (ADC_BGR).....	367
19.11.3	ADC Basic Configuration Register 0(ADC_CR0).....	368
19.11.4	ADC Basic Configuration Register 1 (ADC_CR1).....	370
19.11.5	ADC Sequential Scan Conversion Channel Configuration Register 0 (ADC_SQR0). 371	
19.11.6	ADC Sequential Scan Conversion Channel Configuration Register 1 (ADC_SQR1). 371	
19.11.7	ADC Sequential Scan Conversion Channel Configuration Register 2 (ADC_SQR2). 372	
19.11.8	ADC conversion result (ADC_RESULT)	372
19.11.9	ADC conversion result accumulation value (ADC_RESULTACC)	373
19.11.10	ADC Compare Upper Threshold (ADC_HT)	373
19.11.11	ADC Compare Lower Threshold (ADC_LT)	373

19.11.12	ADC Interrupt Enable (ADC_IER)	374
19.11.13	ADC Interrupt Flag Register (ADC_IFR)	375
19.11.14	ADC Interrupt Clear Register (ADC_ICR)	376
19.11.15	ADC External Interrupt Trigger Source Configuration Register 0 (ADC_EXTTRIGGER0) 377	
19.11.16	ADC Single Conversion Start Control Register (ADC_SGLSTART).....	378
19.11.17	ADC Sequential Scan Conversion Start Control Register (ADC_SQRSTART)	378
20	Analog Voltage Comparator (VC).....	379
20.1	Introduction	379
20.2	Main features.....	379
20.3	Functional description	380
20.3.1	Functional block diagram.....	380
20.3.2	Setup / Response Time	380
20.3.3	Hysteresis function	380
20.3.4	Compare Results Polarity Configuration.....	381
20.3.5	window compare function.....	381
20.3.6	Filter description	381
20.4	Registerdescription.....	382
20.4.1	register list.....	382
20.4.2	Control Register 0 (VC_CR0)	383
20.4.3	Control Register 1 (VC_CR1)	384
20.4.4	Control Register 2 (VC_CR2)	386
20.4.5	Status Register (VC_SR).....	386
21	Low Voltage Detector (LVD)	387
21.1	Introduction to LVD	387
21.2	Main characteristics	387
21.3	Functional description	387
21.3.1	Functional block diagram.....	387
21.3.2	Hysteresis function	388
21.3.3	Digital filtering	388
21.4	Programming example	389
21.4.1	LVD configured as low voltage reset.....	389
21.4.2	LVD configured as voltage change interrupt.....	389
21.5	Registerdescription.....	390
21.5.1	Register list.....	390
21.5.2	LVD Configuration Register (LVD_CR).....	391
21.5.3	LVD Interrupt Register (LVD_SR)	391
22	Debug Controller (DBGC).....	392

22.1	Introduction	392
22.2	DBGC system block diagram	392
22.3	SW-DP Debug Port(SWD)	393
22.3.1	SWD port pin.....	393
22.3.2	SW-DP debug port pin.....	393
22.3.3	Internal pull-ups on SWD pins.....	393
22.4	Register description.....	395
22.4.1	Register list.....	395
22.4.2	Timer Debug Halt Control Register (SYSCTRL_DEBUG)	395
23	Device Electronic Signature.....	396
23.1	Product Unique Identifier (UID) Register (48 bits).....	396
23.2	Product Model Register.....	396
23.3	FLASH capacity register	397
23.4	RAM capacity register.....	397
23.5	Pin Count Register	397
24	Appendix A SysTick Timer	398
24.1	Introduction to SysTick Timers	398
24.2	Set SysTick	398
24.3	SysTick register	399
24.3.1	SysTick Control and Status Register (CTRL).....	399
24.3.1	SysTick reload register (LOAD).....	399
24.3.2	SysTick current value register (VAL)	399
24.3.3	SysTick calibration value register (CALIB).....	400
	Version Revision History	401

Table Index

Table 1-1	Address Division Table.....	27
Table 1-2	Peripheral address division table.....	27
Table 2-1	Reset signal in each working mode.....	29
Table 2-2	Working status of each functional module in different working modes.....	32
Table 2-3	System Control Register List	33
Table 3-1	Behavior of processor in sleep state when interrupt request is received	42
Table 3-2	SYSCTRL register list	45
Table 4-1	Reset register list	63
Table 5-1	Cortex-M4 processor interrupt overview	68
Table 5-2	Correspondence between external interrupt and NVIC interrupt input	74
Table 5-3	Interrupt Register List.....	78
Table 6-1	Port function configuration table	85
Table 6-2	PINMUX multiplexing function	91
Table 6-3	IOAUX1/2 multiplexing function	92
Table 6-4	GPIO base address list.....	95
Table 6-5	GPIO Register List.....	95
Table 7-1	I2C clock signal baud rate	109
Table 7-2	I2C status code expression	115
Table 7-3	I2C base address list	121
Table 7-4	I2C register list.....	121
Table 8-1	SPI interrupt	137
Table 8-2	SPI base address list.....	141
Table 8-3	SPI register list	141
Table 9-1	CTRIM register list	151
Table 10-1	FLASH capacity division	158
Table 10-2	Chip Protection Level Table	164
Table 10-3	FLASH register list.....	165
Table 11-1	RAM Address Mapping	170
Table 12-1	DMA transfer mode table	173
Table 12-2	DMA register list.....	179
Table 13-1	Pin function and timer function comparison table.....	185
Table 13-2	BTIM timer interrupt.....	192
Table 13-3	List of GTIM interrupt sources.....	200
Table 13-4	GTIM trigger interconnection list	200
Table 13-5	List of GTIM ETR sources	200
Table 13-6	List of -GTIM CH3 signal sources	200

Table 13-7	BTIM base address list.....	201
Table 13-8	BTIM register list	201
Table 13-9	GTIM base address list	207
Table 13-10	GTIM register list	207
Table 14-1	ATIM PRS divider ratio list.....	218
Table 14-2	Correspondence between Buzzer output frequency and reload mode configuration value under 4MHz clock.....	221
Table 14-3	PWC mode measurement start and end edge configuration table.....	222
Table 14-4	PWC mode measurement signal source selection.....	223
Table 14-5	Capture input CH0A signal list.....	245
Table 14-6	Capture input CH0B signal list.....	245
Table 14-7	MSCR Register Slave Mode and Trigger Source List.....	248
Table 14-8	ATIM interrupt source and interrupt request list.....	253
Table 14-9	Timer Interconnect Signal List.....	253
Table 14-10	ATIM0 ETR signal source list.....	253
Table 14-11	ATIM0 CHA input signal source list	254
Table 14-12	ATIM base address list.....	255
Table 14-13	ATIM register list.....	255
Table 15-1	IWDT timeout time (10kHz independent clock).....	283
Table 15-2	IWDT register list.....	284
Table 16-1	WWDT register list.....	290
Table 17-1	USART each mode sampling count and maximum baud rate.....	295
Table 17-2	PCLK is 4MHz baud rate programming example	296
Table 17-3	PCLK is 8MHz baud rate programming example	297
Table 17-4	PCLK is 16MHz baud rate programming example	297
Table 17-5	PCLK is 24MHz baud rate programming example	298
Table 17-6	PCLK is 32MHz baud rate programming example	299
Table 17-7	PCLK is 48MHz baud rate programming example	300
Table 17-8	Multifunctional timer counter function	304
Table 17-9	Validity of interrupts in each mode of USART	306
Table 17-10	USART base address list.....	333
Table 17-11	USART Register List.....	333
Table 18-1	CRC register list.....	351
Table 19-1	ADC maximum conversion speed	354
Table 19-2	ADC interrupt	363
Table 19-3	ADC base address list.....	366
Table 19-4	ADC register list	366
Table 20-1	VC base address list.....	382

Table 20-2	VC Register List	382
Table 21-1	LVD Register List	390
Table 22-1	SWD port pin	393
Table 22-2	SW-DP debug port pin	393
Table 22-3	DBGC register list.....	395
Table 23-1	UID Definition	396
Table 24-1	SysTick register	399

Figure Index

Figure 1-1	System Architecture Diagram.....	26
Figure 2-1	Working Mode State Transition Diagram.....	29
Figure 3-1	Clock Control Module Block Diagram.....	35
Figure 3-2	PLL System Block Diagram	36
Figure 3-3	Schematic diagram of crystal oscillator clock startup	37
Figure 3-4	Schematic diagram of clock switching.....	39
Figure 4-1	Reset Source Schematic	61
Figure 5-1	Only the upper 3 bits of the priority register are used.....	69
Figure 5-2	Interrupt Vector Table	70
Figure 5-3	Interrupt active and pending states.....	71
Figure 5-4	Interrupt pending status is cleared and then reasserted	72
Figure 5-5	When the interrupt exits, if the interrupt request remains high, it will cause the interrupt processing to be executed again	72
Figure 5-6	Interrupt pending interrupts generated during interrupt processing can also be acknowledged.....	73
Figure 6-1	General port circuit diagram.....	85
Figure 6-2	HiZ - Analog Configuration Schematic	86
Figure 6-3	Schematic diagram of input floating/pull-up/pull-down configuration	88
Figure 6-4	Read port pin data synchronization diagram	88
Figure 6-5	Schematic diagram of output configuration	89
Figure 6-6	Schematic diagram of multiplexing function configuration	90
Figure 6-7	Schematic Diagram of Port External Interrupt	94
Figure 7-1	I2C transfer protocol	105
Figure 7-2	START and STOP conditions	105
Figure 7-3	Bit transfer on the I2C bus.....	106
Figure 7-4	Acknowledgment signal on I2C bus	106
Figure 7-5	Arbitration on the I2C bus.....	107
Figure 7-6	I2C function block diagram.....	108
Figure 7-7	Data synchronization diagram of master sending mode	110
Figure 7-8	I2C master sending state diagram.....	111
Figure 7-9	Data synchronization diagram of main receiving mode.....	111
Figure 7-10	I2C master receiving state diagram.....	112
Figure 7-11	Slave Receive Mode Data Synchronization Diagram.....	113
Figure 7-12	Slave receiving state diagram	113
Figure 7-13	Slave mode data synchronization diagram.....	113
Figure 7-14	I2C Slave Transmit State Diagram	114

Figure 7-15	I2C General Call State Diagram	114
Figure 8-1	SPI Block Diagram	126
Figure 8-2	SPI communication format timing diagram.....	127
Figure 8-3	Slave select NSS pin configuration	128
Figure 8-4	SPI full duplex communication.....	129
Figure 8-5	SPI single-wire half-duplex communication.....	129
Figure 8-6	SPI simplex sending communication diagram	130
Figure 8-7	SPI Simplex Receive Communication Schematic	130
Figure 8-8	SPI multi-machine system	131
Figure 8-9	Explanation diagram of general flags in SPI master mode	135
Figure 8-10	SPI slave mode general flag bit description diagram.....	136
Figure 9-1	RCH clock calibration working block diagram	146
Figure 9-2	RCL clock calibration working block diagram.....	148
Figure 9-3	Low Power Timer Working Block Diagram.....	149
Figure 12-1	DMA functional block diagram.....	172
Figure 12-2	DMA software block transfer.....	174
Figure 12-3	DMA software Burst transfer.....	174
Figure 12-4	DMA hardware block transfer	175
Figure 12-5	DMA hardware Burst transfer	176
Figure 13-1	BTIM Block Diagram.....	187
Figure 13-2	Schematic diagram of BTIM filter unit.....	187
Figure 13-3	BTIM counter waveform.....	188
Figure 13-4	BTIM Timing Mode Block Diagram	189
Figure 13-5	BTIM Counter Mode Block Diagram	189
Figure 13-6	BTIM Trigger Initiator Mode Block Diagram.....	190
Figure 13-7	Schematic diagram of BTIM trigger start mode counting	190
Figure 13-8	BTIM gating mode block diagram	191
Figure 13-9	Schematic diagram of counting in BTIM gating mode	191
Figure 13-10	Schematic diagram of cascading -BTIM timers.....	192
Figure 13-11	GTIM Block Diagram	193
Figure 13-12	Schematic diagram of GTIM filter unit	193
Figure 13-13	GTIM counter waveform.....	194
Figure 13-14	GTIM Timing Mode Block Diagram	195
Figure 13-15	GTIM Counter Mode Block Diagram	195
Figure 13-16	GTIM Trigger Initiator Mode Block Diagram.....	196
Figure 13-17	Counting diagram of GTIM trigger start mode	196
Figure 13-18	GTIM gating mode block diagram.....	197
Figure 13-19	Schematic diagram of counting in GTIM gating mode	197

Figure 13-20	Schematic diagram of GTIM capture function.....	198
Figure 13-21	Schematic diagram of GTIM comparison function.....	199
Figure 14-1	ATIM3 block diagram	216
Figure 14-2	ATIM0 block diagram	217
Figure 14-3	Free Counting Block Diagram	219
Figure 14-4	Heavy Duty Counting Waveform.....	219
Figure 14-5	16 -bit heavy load counting waveform	220
Figure 14-6	32 -bit free-counting waveform	220
Figure 14-7	Free Count Timing Diagram	221
Figure 14-8	Timing diagram of heavy load counting (prescaler is set to 2)	221
Figure 14-9	PWC Measurement Block Diagram	222
Figure 14-10	Rising edge to falling edge high level measurement.....	223
Figure 14-11	Falling edge to falling edge period measurement.....	223
Figure 14-12	Rising edge to rising edge period measurement	224
Figure 14-13	Rising edge-to-rising edge period measurement one-shot mode	225
Figure 14-14	Counter Block Diagram.....	226
Figure 14-15	Count Up Without Prescaler	227
Figure 14-16	Up Counting with Prescaler.....	228
Figure 14-17	Down counting without prescaler	228
Figure 14-18	Down counting with prescaler	229
Figure 14-19	Up and down counting with prescaler.....	229
Figure 14-20	Edge-Aligned Timer Waveform (DIR =1)	229
Figure 14-21	Edge-Aligned Timer Waveform (DIR =0)	230
Figure 14-22	Center Aligned Counter Waveform	230
Figure 14-23	Repeat counter generates update timing (RCR.UD=0 RCR.OV=0)	231
Figure 14-24	When the overflow is masked, the repetition counter generates an update timing(RCR.OV=1, RCR.UD=0)	231
Figure 14-25	When the underflow is masked, the repetition counter generates an update timing(RCR.OV=0, RCR.UD=1)	232
Figure 14-26	Triangular Wave Mode Buffer Enable	233
Figure 14-27	Triangular Wave Mode Buffer Invalidation	233
Figure 14-28	Count buffer enable on sawtooth mode.....	234
Figure 14-29	Count Buffer Invalidation on Sawtooth Mode.....	234
Figure 14-30	Count buffer enable in sawtooth mode.....	234
Figure 14-31	The counter buffer is invalid in sawtooth mode.....	235
Figure 14-32	Count compare buffer enable in sawtooth mode.....	235
Figure 14-33	OCREF output block diagram.....	236
Figure 14-34	Sawtooth counting single point comparison OCREF output waveform(OCMx=111)	237

Figure 14-35	Triangular wave counting single point comparison OCREF output waveform (OCMx=111)	237
Figure 14-36	Sawtooth wave counting double-point comparison OCREF output (OCMx=111)	238
Figure 14-37	Triangular wave counting double-point comparison OCREF output (OCMx=111)	238
Figure 14-38	Independent PWM Output Block Diagram	238
Figure 14-39	PWM output waveform when CCPx=0	239
Figure 14-40	PWM output waveform when CCPx=1	239
Figure 14-41	Complementary PWM Output Block Diagram	239
Figure 14-42	Complementary PWM output waveform	239
Figure 14-43	Complementary PWM output waveform	240
Figure 14-44	Dead time	240
Figure 14-45	Single pulse counting in triangle wave mode	241
Figure 14-46	Counting Single Pulse Mode on Ramp Waveform	242
Figure 14-47	Counting single pulse mode under sawtooth waveform	242
Figure 14-48	Interrupt diagram	243
Figure 14-49	Capturing Functional Block Diagram	244
Figure 14-50	Capture Timing Diagram	244
Figure 14-51	ATIM3 CH0A port selection	245
Figure 14-52	CHB port selection	245
Figure 14-53	Slave Mode Schematic	248
Figure 14-54	Gating function	248
Figure 14-55	Trigger and reset functions	249
Figure 14-56	Code counting function block diagram	250
Figure 14-57	Code count	251
Figure 14-58	ADC trigger	252
Figure 15-1	WDT functional block diagram	280
Figure 16-1	WWDT functional block diagram	288
Figure 17-1	Asynchronous serial port mode hardware flow control example	302
Figure 17-2	CTS hardware flow control signal	303
Figure 17-3	RTS hardware flow control signal	303
Figure 17-4	Functional block diagram of asynchronous serial port	307
Figure 17-5	Asynchronous serial port mode data format	308
Figure 17-6	Example of sending data	309
Figure 17-7	Receive data example	310
Figure 17-8	Multiprocessor communication data format	311
Figure 17-9	Synchronous serial port mode character format legend	314
Figure 17-10	Synchronous serial port mode sending data legend 1	315
Figure 17-11	Synchronous serial port mode sending data legend 2	315

Figure 17-12	Synchronous serial port mode receiving data legend 1	317
Figure 17-13	Synchronous serial port mode receiving data legend 2	317
Figure 17-14	Precautions for synchronous serial port mode	318
Figure 17-15	Smart card Pattern connected legend	319
Figure 17-16	Smart Card Pattern synchronization and sampling timing Figure	320
Figure 17-17	Smart Card Mode Data Format	320
Figure 17-18	Example of smart card mode data format	321
Figure 17-19	Smart card Pattern sending data legend	322
Figure 17-20	Smart card Pattern Receive data legend	323
Figure 17-21	Infrared Data Modulation	323
Figure 17-22	38K infrared modulation	326
Figure 17-23	LIN bus data and frame structure	327
Figure 19-1	ADC block diagram	353
Figure 19-2	ADC conversion timing diagram	354
Figure 19-3	ADC sequential scan conversion process example	357
Figure 19-4	ADC Continuous Conversion Accumulation Process Example	360
Figure 19-5	Schematic diagram of ADC single conversion or sequential scan conversion external trigger source	362
Figure 19-6	ADC conversion result comparison	363
Figure 20-1	VC functional block diagram	380
Figure 20-2	VC hysteresis function	380
Figure 20-3	VC Filter Response Time	381
Figure 21-1	LVD Block Diagram	387
Figure 21-2	LVD Hysteresis Response	388
Figure 21-3	LVD Filter Output	388
Figure 22-1	Commissioning control system	392

Overview

The HC32F420 series is a 32-bit microcontroller with a high-performance ARM® Cortex®-M4 core. The operating voltage range is 2.0~5.5V and the operating temperature range is -40~105°C. It has rich peripheral configurations and a variety of Working mode to apply to different application scenarios.

This product provides a variety of packaging forms, and the peripheral configurations are different in each packaging form.

Typical Application

- Internet of things application, smart home
- Smart meters, smart transportation, medical equipment, handheld devices
- Motor application, frequency conversion control
- Alarm systems, printers, scanners and other industrial applications

About this manual

This manual mainly introduces the function, operation and usage of the chip. For chip specifications, please refer to the corresponding "Datasheet".

1 System Structure

1.1 Overview

This product system consists of the following parts:

- 4 AHB Masters:
 - Cortex-M4 IBUS
 - Cortex-M4 DBUS
 - Cortex-M4 SBUS
 - DMA controller
- Multiple AHB Slaves:
 - FLASH memory
 - SRAM memory
 - 1 AHB to APB Bridge contains all APB interface peripherals
 - 2 AHB to AHB Bridge contains all AHB interface peripherals

The entire system bus structure is realized by multi-level AHB-lite bus interconnection.

1.2 System Architecture Diagram

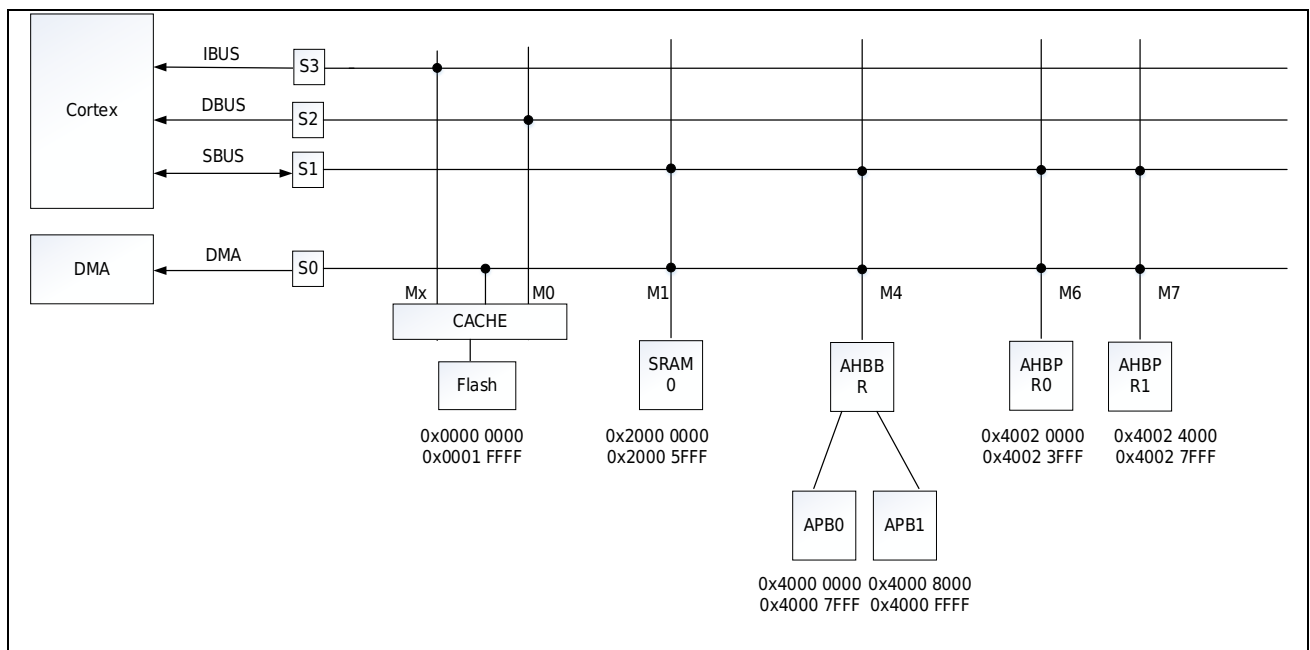


Figure 1-1 System Architecture Diagram

1.3 Memory and peripheral address allocation

Table 1-1 Address Division Table

Categories	Boundary Address	Size	Memory Area
Memory	0x0000 0000 – 0x0001 FFFF	128K Byte	FLASH
	0x2000 0000 – 0x2000 5FFF	24K Byte	SRAM
APB peripherals AHBRR	0x4000 0000 – 0x4000 7FFF	32K Byte	APB0
	0x4000 8000 – 0x4000 FFFF	32K Byte	APB1
AHB peripherals	0x4002 0000 – 0x4002 3FFF	16K Byte	AHBPR0
	0x4002 4000 – 0x4002 8FFF	16K Byte	AHBPR1

Table 1-2 Peripheral address division table

Categories	base address	Size	Peripheral equipment
APB0	0x4000 0000	0x400	USART0
	0x4000 0400	0x400	USART1
	0x4000 0800	0x400	USART2
	0x4000 0C00	0x400	USART3
	0x4000 2800	0x400	I2C0
	0x4000 2C00	0x400	I2C1
	0x4000 3800	0x400	SPI0
	0x4000 3C00	0x400	SPI1
	0x4000 5000	0x400	CTIM0
	0x4000 5400	0x400	CTIM1
	0x4000 5800	0x400	CTIM2
	0x4000 6000	0x400	ATIM0
	0x4000 6C00	0x400	ATIM3
APB1	0x4000 8000	0x400	CTRIM
	0x4000 8800	0x400	WWDT
	0x4000 8C00	0x400	IWDT
	0x4000 A000	0x400	ADC0
	0x4000 A400	0x400	ADC1
	0x4000 A800	0x400	VC
	0x4000 AC00	0x400	LVD
	0x4000 B800	0x400	SYSCTRL
	0x4000 BC00	0x400	Keep
AHBPR0	0x4002 0000	0x400	DMAC
	0x4002 0800	0x400	CRC
	0x4002 1400	0x400	FLASH
AHBPR1	0x4002 4000	0x400	GPIOA
	0x4002 4400	0x400	GPIOB
	0x4002 4800	0x400	GPIOC

Categories	base address	Size	Peripheral equipment
	0x4002 4C00	0x400	GPIO

2 Operating mode

2.1 Overview of working mode

This product can realize three working modes under the control of the power management module. The functional modules and power consumption that can work in each working mode are different.

The working mode of this product is as follows:

- Active Mode: CPU running, on-chip peripherals running
- Sleep Mode: CPU stops, on-chip peripherals run
- Deep Sleep Mode: CPU stops, low-power on-chip peripherals run

2.2 Working mode switching

The product can be freely switched among three working modes when it is working. WFI command in the running mode can make the product enter the sleep mode or deep sleep mode. Wake-up from Hibernate or Deep-Sleep mode via an interrupt and return to Run mode.

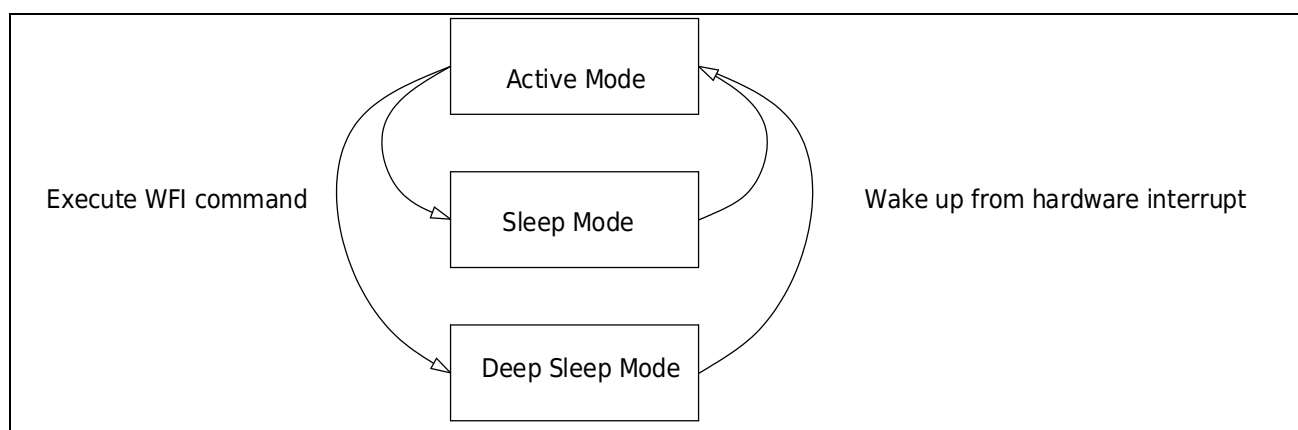


Figure 2-1 Working Mode State Transition Diagram

For the interrupts that the CPU can respond to in various working modes, see the [Interrupt source] chapter.

In various working modes, the reset signals that can be responded to are shown in the table below.

Table 2-1 Reset signal in each working mode

reset source	Operating mode	Sleep mode	Deep sleep mode
Power-on reset POR/ PDR	✓	✓	✓
Hardware reset NRESET pin	✓	✓	✓
Hardware reset LVD	✓	✓	✓
Hardware reset IWDG	✓	✓	✓
Hardware reset WWDG	✓	✓	

reset source	Operating mode	Sleep mode	Deep sleep mode
Hardware reset Cortex-M4 LOCKUP	✓		
Software reset Cortex-M4 SYSRESETREQ	✓		

2.3 Active Mode

In this working mode, all functional modules of the MCU work normally, and the power consumption is the largest.

After the system is reset or woken up from an interrupt, the MCU will work in the running state.

In the running mode, certain measures can be taken to reduce the running power consumption:

- Select a lower frequency for the system clock (SystemClk), modify SYSCTRL_CR0.CLKSRC
- Reduce the frequency of the system's core clock (HCLK), modify SYSCTRL_CR0.HCLKPRS
- Reduce the frequency of the system's on-chip peripheral clock (PCLK), modify SYSCTRL_CR0.PLCK0PRS/PLCK1PRS
- Turn off the clock of unused on-chip peripherals, modify SYSCTRL_APBEN0/APBEN1 and SYSCTRL_AHBEN0/AHBEN1

2.4 Sleep Mode

In this working mode, the CPU stops running, and other on-chip peripherals work normally, and the power consumption is slightly lower than that in the running mode.

In running mode, set SCB_SCR.SLEEPDEEP to 0 and execute WFI command to make MCU work in sleep mode. When the MCU responds to the hardware interrupt, it can exit the sleep mode and return to the running mode.

In sleep mode, certain measures can be taken to reduce operating power consumption:

- Select a lower frequency for the system clock (SystemClk), modify SYSCTRL_CR0.CLKSRC
- Reduce the frequency of the system's core clock (HCLK), modify SYSCTRL_CR0.HCLKPRS
- Reduce the frequency of the system's on-chip peripheral clock (PCLKx), modify SYSCTRL_CR0.PLCK0PRS/PLCK1PRS
- Turn off the clock of unused on-chip peripherals, modify SYSCTRL_APBEN0/APBEN1 and SYSCTRL_AHBEN0/AHBEN1
- Set SCB_SCR.SLEEPONEXIT to 1, so that the MCU will automatically enter sleep mode after executing the interrupt service routine

2.5 Deep sleep Mode

In this working mode, the CPU stops running, and the high-speed clock (RCH) is automatically turned off. If the source of HCLK is a high-speed clock, the on-chip high-speed peripherals stop running, and only the on-chip low-power peripherals work normally; if the source of HCLK is a low-speed clock (RCL), all on-chip peripherals work normally. The power consumption in this mode is much lower than that in sleep mode. This system does not support entering this mode when the system clock is PLL, it is recommended to switch the system clock to RCHCLKD before entering this mode. RCHCLKD refers to the clock generated by the 24M RCH clock configured by the SYSCTRL_RCHCR.DIV register. For example, if SYSCTRL_RCHCR.DIV=1, the RCHCLKD frequency is 6MHz.

If SYSCTRL_CR1.WAKEUPCLK is set to 1, the source of the system clock when the MCU exits the deep sleep mode is RCH, so as to realize fast wake-up.

In running mode, set SCB_SCR.SLEEPDEEP to 1 and execute WFI command to make MCU work in deep sleep mode. When the MCU responds to the hardware interrupt, it can exit the deep sleep mode and return to the running mode.

In sleep mode, certain measures can be taken to reduce operating power consumption:

- Select HCLK source as high-speed clock, modify SYSCTRL_CR0.CLKSRC
- Turn off the clock of unused on-chip low-power peripherals, modify SYSCTRL_APBEN0/APBEN1 and SYSCTRL_AHBEN0/AHBEN1
- Set SCB_SCR.SLEEPONEXIT to 1, so that the MCU will automatically enter sleep mode after executing the interrupt service routine

2.6 Functional module working status

The comparison of the working status of the functional modules in the three working modes is shown in the table below:

Table 2-2 Working status of each functional module in different working modes

Functional module	Operating mode	Sleep mode	Deep sleep mode
CM4	✓		
DMAC	✓	✓	
GPIO	✓	✓	✓
VC	✓	✓	✓
LVD	✓	✓	✓
ADC	✓	✓	
ATIM	✓	✓	
CTIM	✓	✓	
SPI	✓	✓	
I2C	✓	✓	
USART	✓	✓	
CTRIM	✓	✓	✓
WWDT	✓	✓	
IWDT	✓	✓	✓
CRC	✓	✓	

2.7 Register description

2.7.1 register list

Base address: 0xE000 ED10

Table 2-3 System Control Register List

Register name	Offset address	Reset value
System Control Register (SCB_SCR)	0x00	0x0000 0000

2.7.2 System Control Register (SCB_SCR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res											SEV ONPE ND	Res	SLEE PDEE P	SLEE PONE XIT	Res

Bit	Marking	Place name	Function	Read and write
b31~b5	Res	Reserved bit	Reserved, must remain at reset value	
b4	SEVONPEND		1: An event is generated every time a new interrupt is pending, which can be used to wake up the processor if WFE sleep is used	RW
b3	Res	Reserved bit	Reserved, must remain at reset value	
b2	SLEEPDEEP		1: Execute WFI to enter deep sleep, and this product enters Deep Sleep mode 0: Execute WFI to enter sleep mode, this product enters Sleep/Idle mode	RW
b1	SLEEPONEXIT		1: When exiting exception handling and returning to the program thread, the processor automatically enters sleep mode (WFI) 0: the feature is automatically disabled	RW
b0	Res	Reserved bit	Reserved, must remain at reset value	

3 System Controller (SYSCTRL)

3.1 System Clock Introduction

The clock control module mainly controls the system clock and the peripheral clock. Different clock sources can be configured as the system clock, different frequency divisions of the system clock can be configured, and peripheral clocks can be enabled or disabled. To ensure oscillator accuracy, the internal clocks are calibrated.

This product supports the following four different clock sources as the system clock:

- Internal high-speed RC clock RCH (output frequency is 22~24MHz)
- Internal low-speed RC clock RCL (38.4K and 32.768K configurable)
- External high-speed crystal oscillator clock XTH
- Phase-locked loop clock PLL

This product also supports the following two auxiliary clock sources:

- Internal RC10K clock, only used by IWDT
- Internal RC150K clock, only for LVD and VC modules

Note:

1: When switching the clock source of the system clock, please strictly follow the operation steps to switch, see the chapter [System Clock SwitchingSystem Clock Switching] for details.

2: XTH can not be connected to the crystal oscillator, and the clock signal can be directly input from the PD00 pin.

The PCL K clock signal can be output through the PCLKx_OUT pin; the RCH/RCL/XTH/PLL/RC150K/RC10K clock signal can be output through the MCO pin.

3.1.3 Internal low speed RC clock RCL

The output frequency of the internal low-speed clock can be selected through the register SYCTRL_RCLTRIM.TRIM, and the selectable frequencies are 38.4KHz and 32.768KHz. When the system enters DeepSleep, this low-speed clock will not be automatically turned off, and the ultra-low power peripheral module can choose RCL as its clock.

3.1.4 External crystal oscillator clock XTH

The external crystal oscillator clock needs to be connected with an 8M~32M crystal

XTH can also not be connected to the crystal oscillator, and the clock signal can be directly input from the PD00 pin. The method of inputting the clock signal from PD00 is: configure the PD00 pin as GPIO digital input; set SYCTRL_CR2.EXXTH to 1.

3.1.5 Phase-locked loop clock PLL

The built-in PLL supports 15~84MHz clock output, and the reference clock source of PLL can be selected as RCHCLKD and XTH. This system does not support entering DeepSleep when the system clock is PLL. It is recommended to switch the system clock to other clocks before entering DeepSleep.

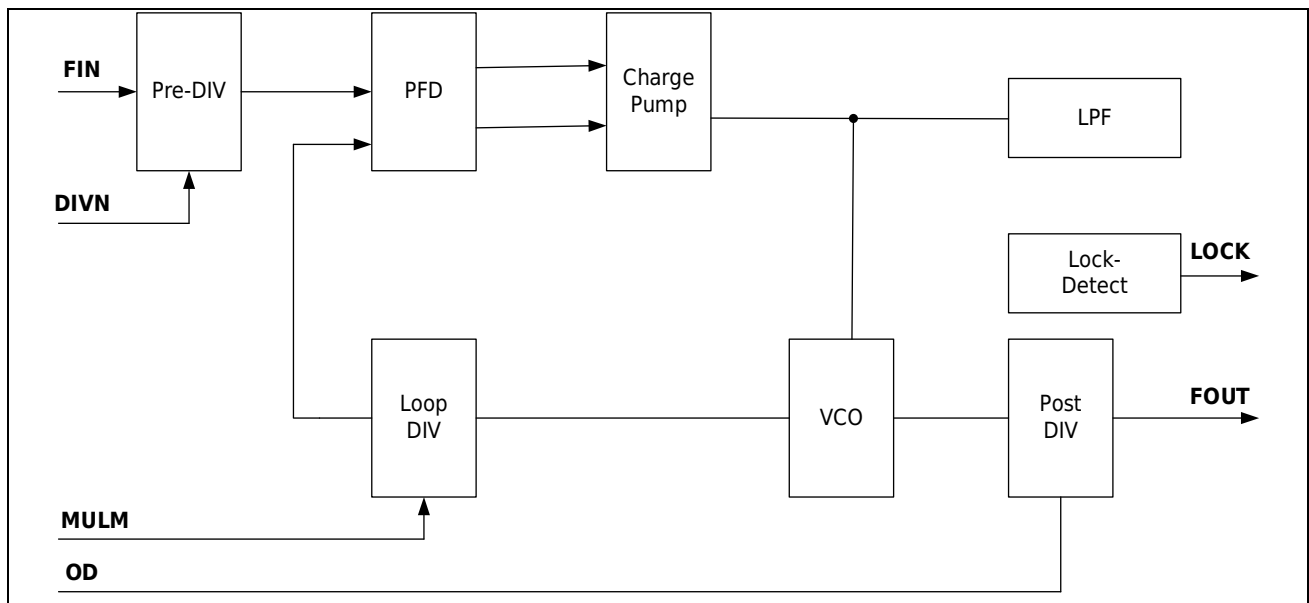


Figure 3-2 PLL System Block Diagram

FIN Input frequency range 4MHz~32MHz

FOUT Output frequency range 15MHz~84MHz

$$FOUT = FIN * \frac{MULM}{DIVN} * \frac{1}{OD}$$

FIN/DIVN frequency range 4MHz~16MHz

FIN*MULM/DIVN frequency range 120MHz~200MHz

The value range of MULM is 2~63

The value range of DIVN is 1~63

The OD value range is the frequency division value set by the OD position

Note: MULM, DIVN, and OD are the control bits of the SYSCTRL_PLLCR register

3.1.6 Clock start process

The above 4 kinds of clock sources all need a start-up stabilization time. The following figure uses the external XTH as an example to illustrate the start-up stabilization process of the clock.

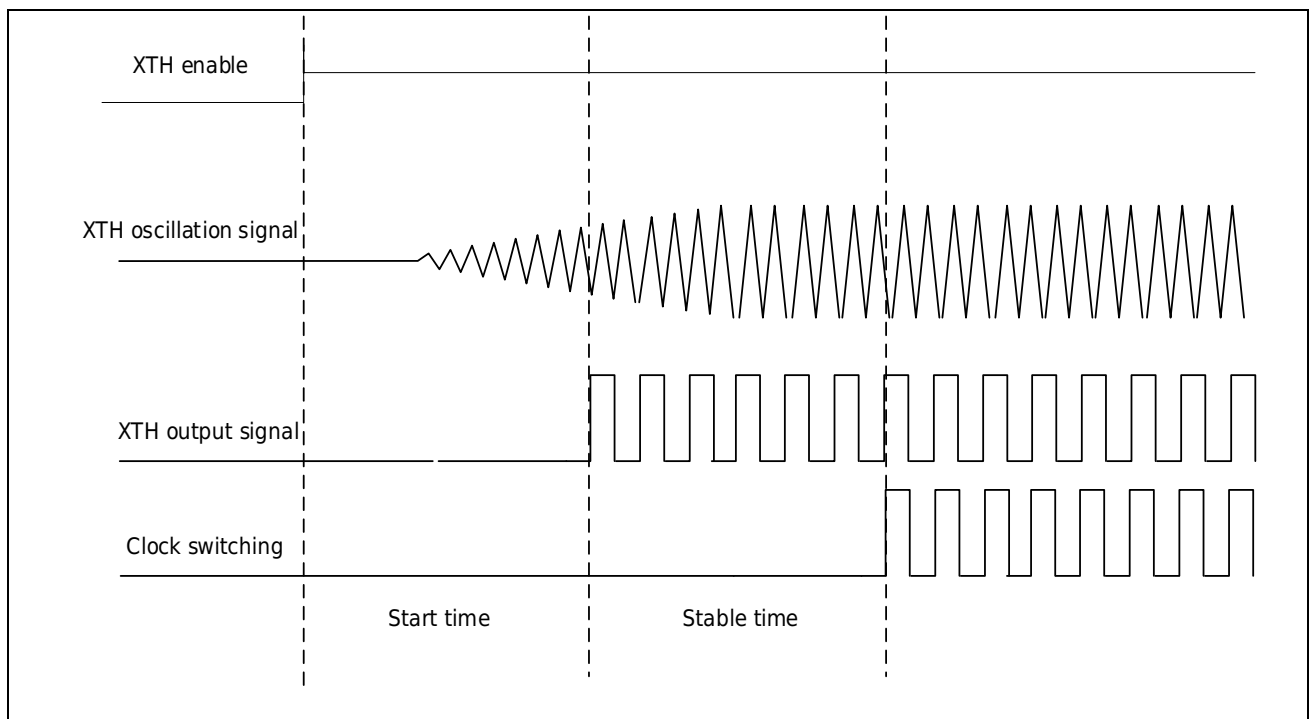


Figure 3-3 Schematic diagram of crystal oscillator clock startup

3.2 System Clock Switching

The system clock can choose 4 kinds of clock sources: RCHCLKD, RCL, XTH, PLL, controlled by the register SYSCTRL_CR0.CLKSRC. Any two of the four clock sources RCHCLKD, RCL, PLL, and XTH can be switched to each other, and it is necessary to ensure that HCLK is not greater than 24MHz during the switching process.

It is recommended to perform clock switching according to the clock switching process described below to reduce the probability of clock switching errors.

FLASH_WAIT register needs to be configured synchronously when the clock is switched:

- The clock frequency is not greater than 24MHz, set FLASH_WAIT to 0 (1 cycle)
- The clock frequency is between 24~48MHz, set FLASH_WAIT to 1 (2 cycles)

- The clock frequency is between 48~72MHz, set FLASH_WAIT to 2 (3 cycles)
- The clock frequency is between 72~84MHz, set FLASH_WAIT to 3 (4 cycles)

Note:

To set the value of FLASH_WAIT, you need to write 0x5A5A and 0xA5A5 to the FLASH_BYPASS register in sequence, and then assign a value to FLASH_WAIT. For details, see the chapter on [FLASH controller (FLASH)].

3.2.1 Standard Clock Switching Process

The operation process is as follows:

- Step1. If the new clock source requires an external pin, set the pin to the appropriate mode;
Note: An analog pin is required when connecting to an external crystal oscillator; GPIO digital input is required to input an external clock signal when connecting to an external clock input;
- Step2. Configure the oscillation parameters of the new clock source;
- Step3. Enable the oscillator for the new clock source;
- Step4. According to the higher frequency of the current clock source and the new clock source, configure the FLASH_WAIT register according to the procedures in the Flash controller chapter;
- Step5. Wait for the new clock source to output a stable frequency;
- Step6. Configure SYCTRL_CR0.CLKSRC to select the source of the system clock as the new clock source;
- Step7. According to the frequency of the new clock source, configure the FLASH_WAIT register according to the procedures in the Flash controller chapter;
- Step8. Turn off clock sources that are no longer in use.

The following figure is the clock switching timing diagram:

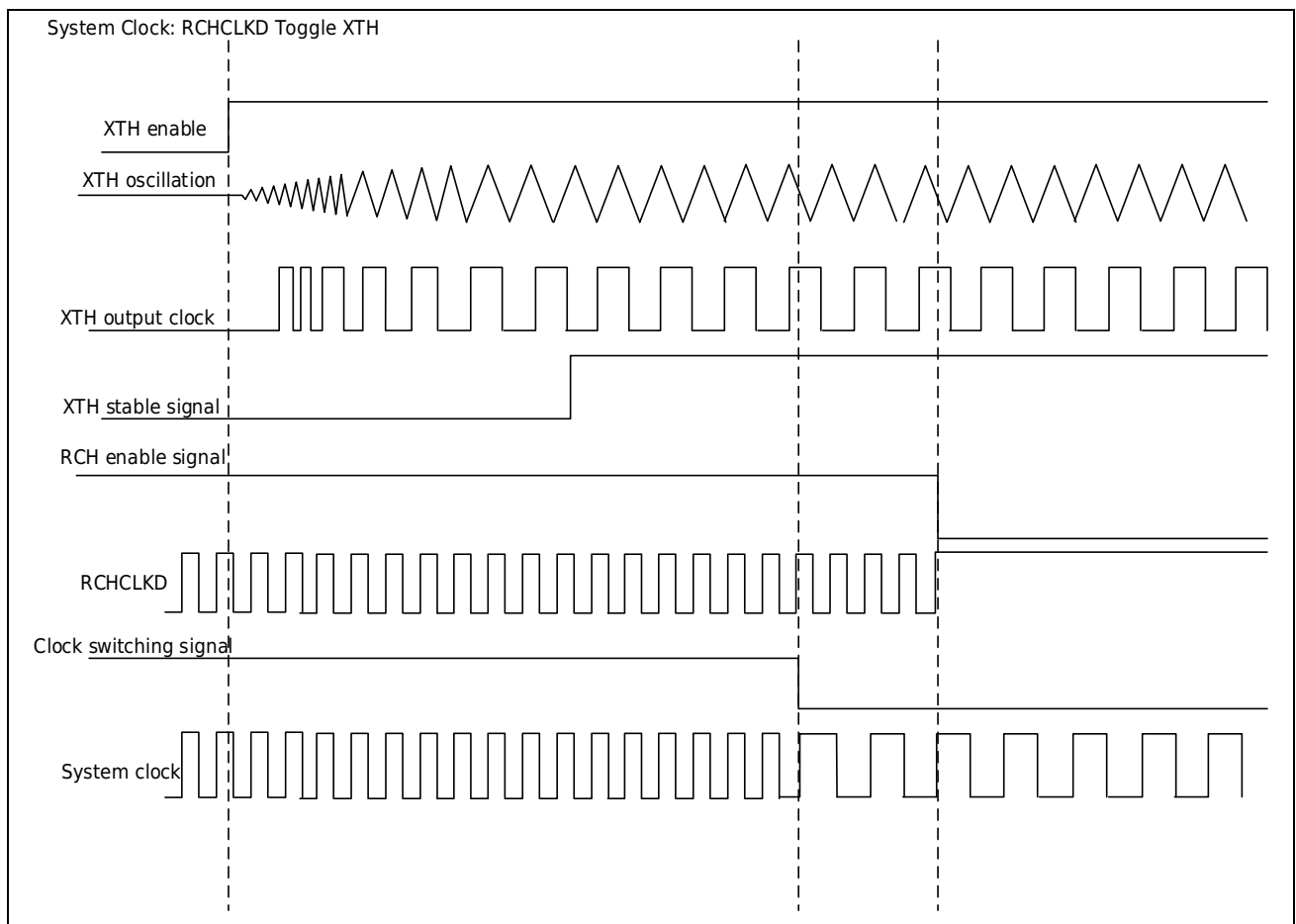


Figure 3-4 Schematic diagram of clock switching

3.2.2 RCHCLKD switching process between different output frequencies

Configure SYSTCTRL_RCHCR.DIV to make RCH output different clock frequencies.

The operation process is as follows:

- Step1. According to the higher frequency of the current RCHCLKD and the target RCHCLKD, configure the FLASH_WAIT register according to the procedures in the Flash chapter;
- Step2. Configure SYSTCTRL_RCHCR.DIV to select the frequency division ratio between RCH oscillation frequency and RCHCLKD frequency;
- Step3. According to the current frequency of RCHCLKD, configure the FLASH_WAIT register according to the procedures in the Flash chapter.

3.2.3 Switching from other clocks to RCL example

The operation process is as follows:

- Step1. Configure SYSTCTRL_RCLTRIM.TRIM and SYSTCTRL_RCLCR.WAITCYCLE;
- Step2. Write 0x5A5A and 0xA5A5 to the SYSTCTRL_CR3 register in sequence to enable register rewriting;
- Step3. Set SYSTCTRL_CR2.RCL to 1 to enable the RCL oscillator circuit;
- Step4. The query waits for the SYSTCTRL_STATE.RCL flag to become 1, and the RCL outputs a stable clock;
- Step5. Write 0x5A5A and 0xA5A5 to the SYSTCTRL_CR3 register in sequence to enable register rewriting;
- Step6. Set SYSTCTRL_CR0.CLKSRC to 2, switch the system clock to RCL;
- Step7. FLASH_WAIT to 0 according to the Flash chapter process;
- Step8. Write 0x5A5A and 0xA5A5 to the SYSTCTRL_CR3 register in sequence to enable register rewriting;
- Step9. Set SYSTCTRL_CR2.xxx to 0 to turn off unused clock sources.

3.2.4 Switching from other clocks to RCHCLKD example

The operation process is as follows:

- Step1. Configure SYSTCTRL_RCHTRIM.TRIM and SYSTCTRL_RCHCR.DIV;
- Step2. Write 0x5A5A and 0xA5A5 to the SYSTCTRL_CR3 register in sequence to enable register rewriting;
- Step3. Set SYSTCTRL_CR2.RCH to 1 to enable the RCH oscillator circuit;

- Step4. The query waits for the SYSTRL_STATE.RCH flag to become 1, and the RCH outputs a stable clock;
- Step5. According to the higher frequency of the current clock and the target RCHCLKD, configure the FLASH_WAIT register according to the procedures in the Flash chapter;
- Step6. Write 0x5A5A and 0xA5A5 to the SYSTRL_CR3 register in sequence to enable register rewriting;
- Step7. Set SYSTRL_CR0.CLKSRC to 0, switch the system clock to RCHCLKD;
- Step8. According to the frequency of RCHCLKD, configure FLASH_WAIT according to the procedures in the Flash chapter;
- Step9. Write 0x5A5A and 0xA5A5 to the SYSTRL_CR3 register in sequence to enable register rewriting;
- Step10. Set SYSTRL_CR2.xxx to 0 to turn off unused clock sources.

3.2.5 Example of switching between PLL (reference clock is RCHCLKD) and RCHCLKD

The operation process of switching from RCHCLKD to PLL is as follows:

- Step1. Set SYSTRL_PLLCR.SRC to 0, select the PLL clock source as RCHCLKD;
- Step2. Configure the SYSTRL_RCHCR.DIV value according to the PLL input frequency range so that RCHCLKD is between 4MHz~12MHz;
- Step3. According to the PLL output frequency, configure MULM, DIVN, OD, BYPASS, OEN in SYSTRL_PLLCR, and configure LKEN according to the power supply noise. Required FIN/DIVN range: 4M-16MHz, FIN*MULM/DIVN range: 120M-200MHz;
- Step4. Configure SYSTRL_PLLCR.WAITSYCLE to select an appropriate PLL stabilization time;
- Step5. Write 0x5A5A and 0xA5A5 to the SYSTRL_CR3 register in sequence to enable register rewriting;
- Step6. Set SYSTRL_CR2.PLL to 1 to enable the PLL oscillation circuit;
- Step7. According to the PLL output frequency, configure FLASH_WAIT according to the procedures in the Flash chapter;
- Step8. The query waits for the SYSTRL_STATE.PLL flag to become 1, and the PLL outputs a stable clock;
- Step9. Write 0x5A5A and 0xA5A5 to the SYSTRL_CR3 register in sequence to enable register rewriting;
- Step10. Set SYSTRL_CR0.CLKSRC to 4, switch the system clock to PLL;

Warning: *SYSCTRL_RCHCR must not be changed during switchover.*

Note: *When the input clock of PLL is selected as RCH, the input frequency cannot be 24MHz.*

The operation process of switching from PLL to RCHCLKD is as follows:

- Step1. Write 0x5A5A and 0xA5A5 to the SYSCTRL_CR3 register in sequence to enable register rewriting;
- Step2. Set SYSCTRL_CR0.CLKSRC to 0, switch the system clock to RCHCLKD;
- Step3. According to the frequency of RCHCLKD, configure FLASH_WAIT according to the procedures in the Flash chapter;
- Step4. Write 0x5A5A and 0xA5A5 to the SYSCTRL_CR3 register in sequence to enable register rewriting;
- Step5. Set SYSCTRL_CR2.xxx to 0 to turn off unused clock sources;

Warning: *SYSCTRL_RCHCR must not be changed during switchover.*

3.3 On-chip peripheral clock control

Most on-chip peripherals require a clock to function properly. When the configuration clock (PCLKx) of the on-chip peripherals is enabled, the user program can read and write the registers of the on-chip peripherals normally; when the working clock of the on-chip peripherals is enabled, the on-chip peripherals can work normally. The working clock of some on-chip peripherals is fixed to PCLKx, and the working clock of some on-chip peripherals can be configured by the user program.

By setting the control bits of SYSCTRL_APBEN0/APBEN1 and SYSCTRL_AHBEN0/AHBEN1 to 1, the configuration clock of the corresponding on-chip peripherals can be enabled.

3.4 Interrupt Wakeup Control

When the processor executes the WFI instruction to enter the sleep state, it will stop executing instructions. When an interrupt request (higher priority) occurs during sleep and needs to be serviced, the processor is woken up.

The behavior of the processor in the sleep state when receiving an interrupt request is shown in the following table:

Table 3-1 Behavior of processor in sleep state when interrupt request is received

PRIMASK status	WFI behavior	Wakeup	ISR execution
0	IRQ Priority > Current Class	Y	Y
0	IRQ Priority $\leq$ Current Level	N	N
1	IRQ Priority > Current Class	Y	N
1	IRQ Priority $\leq$ Current Level	N	N

3.4.1 Enable the NVIC corresponding to the module that needs to wake up the processor

- Step1. Enable the NVIC corresponding to the module that needs to wake up the processor
- Step2. Enable the interrupt corresponding to the module that needs to wake up the processor
- Step3. Set SCB_SCR.SLEEPDEEP to 1
- Step4. Execute WFI instruction to enter deep sleep mode
- Step5. The system enters the deep sleep mode and waits for the interrupt to wake up, and executes the interrupt service program after waking up

Routine:

```
SCB->SCR |= 0x04;
while(1)
{
    __asm("WFI");
}
```

3.4.2 Method not to execute interrupt service routine after wake-up from deep-sleep mode

- Step1. Enable the NVIC corresponding to the module that needs to wake up the processor
- Step2. Enable the interrupt corresponding to the module that needs to wake up the processor
- Step3. Set PRIMASK to 1
- Step4. Set SCB_SCR.SLEEPDEEP to 1
- Step5. Execute WFI instruction to enter deep sleep mode
- Step6. The system enters the deep sleep mode and waits for the interrupt to wake up, and executes the next instruction after waking up
- Step7. Clear interrupt flag, clear interrupt pending status
- Step8. Execute user-defined actions

Routine:

```
__asm("CPSID I"); //Set PRIMASK
SCB_SCR |= 0x04;
while(1)
{
    __asm("WFI");
    M4_TIMx->ICLR = 0x00; //Clear Int Flag
    NVIC_ClearPendingIRQ(TIMx_IRQn); //Clear Pending Flag
}
```

```
// Execute user-defined operations
```

```
}
```

3.4.3 Using exit hibernation feature

Sleep-on-exit is ideal for interrupt-driven applications. When this feature is enabled, the processor enters sleep mode whenever exception handling is complete and returns to thread mode. With the exit-from-sleep feature, the processor can be in sleep mode as much as possible.

Cortex-M4 uses the exit dormancy feature to enter dormancy. This situation is similar to the effect of executing WFI immediately after executing an abnormal exit. However, in order to avoid the need to push the stack when entering an exception next time, the processor will not perform the process of popping the stack.

- Step1. Enable the NVIC corresponding to the module that needs to wake up the processor
- Step2. Enable the interrupt corresponding to the module that needs to wake up the processor
- Step3. Set SCB_SCR.SLEEPDEEP to 1
- Step4. Set SCB_SCR.SLEEPONEXIT to 1
- Step5. Execute WFI instruction to enter deep sleep mode
- Step6. The system enters the deep sleep mode and waits for the interrupt to wake up, and executes the interrupt service subroutine after waking up
- Step7. Automatically enters sleep mode when exiting interrupt service

Routine:

```
SCB_SCR |= 0x04;  
SCB_SCR |= 0x02;  
while(1)  
{  
    __asm("WFI");  
}
```

3.5 Register description

3.5.1 Register list

Base address: 0x4000 B800

Table 3-2 SYSCTRL register list

Register name	Offset address	Reset value
SYSCTRL_CR0 (System Control Register 0)	0x00	0x0000 0000
SYSCTRL_CR1 (System Control Register 1)	0x04	0x0000 0000
SYSCTRL_CR2 (Clock Enable Control Register 2)	0x08	0x0000 0001
SYSCTRL_CR3 (Sequence Control Register)	0x0C	0x0000 0000
SYSCTRL_RCHCR (RCH Control Register)	0x10	0x0000 0002
SYSCTRL_RCLCR (RCL Control Register)	0x18	0x0000 0000
SYSCTRL_XTHCR (XTH Control Register)	0x14	0x0000 0002
SYSCTRL_PLLCR (PLL Control Register)	0x20	0x000C 02C0
SYSCTRL_RCHTRIM (RCH Trim Value Register)	0x24	24M The calibration value where the calibration value address is located
SYSCTRL_RCLTRIM (RCL Trim Value Register)	0x28	The calibration value where the 32.768k calibration value address is located
SYSCTRL_STATE (Clock Status Register)	0x2C	0x0000 0001
SYSCTRL_APBEN0 (APB0 Peripheral Clock Control Register)	0x30	0x0000 0000
SYSCTRL_APBEN1 (APB1 Peripheral Clock Control Register)	0x34	0x0000 8000
SYSCTRL_AHBEN0 (AHB0 Peripheral Clock Control Register)	0x38	0x0000 0020
SYSCTRL_AHBEN1 (AHB1 Peripheral Clock Control Register)	0x3C	0x0000 0000
SYSCTRL_DEBUG (Timer Debug Halt Control Register)	0x64	0x0000 0000
SYSCTRL_IOAUX0 (GPIO Auxiliary Control Register 0)	0x70	0x0000 0000
SYSCTRL_IOAUX1 (GPIO Auxiliary Control Register 1)	0x74	0x0000 0000
SYSCTRL_IOAUX2 (GPIO Auxiliary Control Register 2)	0x78	0x0000 0000

3.5.2 System Control Register 0 (SYSCTRL_CR0)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res				PCLK1PRS		PCLK0PRS		HCLKPRS				Res	CLKSRC		

Bit	Marking	Place name	Function	Read and write
b31~b12	Res	Reserved bit	Reserved, must remain at reset value	
b11~b10	PCLK1PRS	PCLK1 clock source selection	00: HCLK 01: HCLK/2 10: HCLK/4 11: HCLK/8	RW
b9~b8	PCLK0PRS	PCLK0 clock source selection	00: HCLK 01: HCLK/2 10: HCLK/4 11: HCLK/8	RW
b7~b4	HCLKPRS	HCLK clock source selection	0000: SystemClk 0001: SystemClk/2 0010: SystemClk/4 0011: SystemClk/8 0100: SystemClk/16 0101: SystemClk/32 0110: SystemClk/64 0111: SystemClk/128 1000: SystemClk/256 1001: SystemClk/512 Other values: SystemClk	RW
b3	Res	Reserved bit	Reserved, must remain at reset value	
b2~b0	CLKSRC	SystemClk clock source selection	000: Internal high-speed clock RCHCLKD (RCHCLKD is the value of RCH divided by SYSCTRL_RCHCR.DIV) 001: External high-speed clock XTH 010: Internal low-speed clock RCL 011: reserved 100: Internal high-speed clock PLL Note: 1) Set SYSCTRL_CR1.WAKEUPCLK=1, the clock will automatically switch to RCHCLKD after wake-up, otherwise keep the original clock 2) When writing a non-valid value, the value remains unchanged	RW

Note:

Every time you rewrite the value of SYSCTRL_CR0/CR1/CR2, you need to write 0x5A5A and 0xA5A5 to SYSCTRL_CR3 in sequence. Such steps can effectively prevent misuse of SYSCTRL_CR0/CR1/CR2 registers.

3.5.3 System Control Register 1 (SYSCTRL_CR1)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res	GTIM2CFG	GTIM1CFG	GTIM0CFG	WAKEUPCLK	LOCKUPEN	FAULTSW	XTHFLT	SPI_PCTRL	Res	SWDIO					

Bit	Marking	Place name	Function	Read and write
b31~b14	Res	Reserved bit	Reserved, must remain at reset value	
b13~b12	GTIM2CFG	GTIM2 function selection	00: GTIM is used as a general-purpose timer GTIM2 01: GTIM as the basic timer BTIM6/7/8 1x: Reserved, prohibit setting	RW
b11~b10	GTIM1CFG	GTIM1 function selection	00: GTIM is used as general-purpose timer GTIM1 01: GTIM as the basic timer BTIM3/4/5 1x: Reserved, prohibit setting	RW
b9~b8	GTIM0CFG	GTIM0 function selection	00: GTIM is used as a general-purpose timer GTIM0 01: GTIM as the basic timer BTIM0/1/2 1x: Reserved, prohibit setting	RW
b7	WAKEUPCLK	Configure the source of the system clock when DeepSleep wakes up	1: Switch the system clock source to RCHCLKD, and keep the original system clock source enabled 0: Keep the original system clock source.	RW
b6	LOCKUPEN	Cortex-M Lockup function configuration	0: off 1: Enable Note: When enabled, the CPU will reset the MCU when it reads an invalid command, which can enhance system reliability.	RW
b5	FAULTSW	When the system clock source is crystal and the crystal fails, automatically switch the system clock source configuration	0: keep the system clock source unchanged 1: switch the system clock source to RCHCLKD Note: When using this function, the hardware enables the XTH clock filtering function, and the XTH clock oscillation frequency cannot exceed 10M. When XTH is used as the reference clock of the PLL, it is forbidden to enable this function.	RW
b4	XTHFLT	XTH clock filtering enable. (Only XTH frequency less than 12M can be used)	0: Prohibited 1: Enable Note: 1) When using XTH as the system clock, it is forbidden to change this function 2) When XTH is used as the reference clock of PLL it is forbidden to enable this function 3) Requires software to turn on the RCH clock	RW
b3	SPI_PCTRL	SPI port direction control	0: SPI module control 1: GPIO control	RW
b2~b1	Res	Reserved bit	Reserved, must remain at reset value	
b0	SWDIO	Configure the SWD port function	0: PA13 and PA14 are used as SWD ports, and the GPIO function is not available. 1: PA13 and PA14 are used as GPIO ports, and the SWD function is not available.	RW

3.5.4 Clock Enable Control Register 2 (SYSCTRL_CR2)

Reset value: 0x0000 0001

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res										EXXT H	PLL	Res	RCL	XTH	RCH

Bit	Marking	Place name	Function	Read and write
b31~b6	Res	Reserved bit	Reserved, must remain at reset value	
b5	EXXTH	XTH external input clock enable	0: use internal clock, 1: Use PD0 0 to input external clock Note: When using PD00 input clock, PD00 should be set as the digital input port, and CR2.XTH is 1; When using the PD00 external input clock, the PD01 pin is forbidden to be used as a GPIO, and the PD01 port needs to be configured to simulate. PD00 has clock input before enabling this control bit.	RW
b4	PLL	Clock PLL Enable Control	0: OFF 1: Enable Note: 1) When the system enters DeepSleep, the clock will be automatically turned off 2) When the system clock is selected as the current clock, the current clock cannot be turned off	RW
b3	Res	Reserved bit	Reserved, must remain at reset value	
b2	RCL	Internal low-speed clock RCL enable control	0: OFF 1: Enable Note: 1) When the system enters DeepSleep, the clock remains in the previous state 2) When the system clock is selected as the current clock, the current clock cannot be turned off	RW
b1	XTH	External high-speed clock XTH enable control	0: OFF 1: Enable Note: 1) When the system enters DeepSleep, the clock will be automatically turned off 2) When the system clock is selected as the current clock, the current clock cannot be turned off	RW
b0	RCH	Internal high-speed clock RCH enable control	0: OFF 1: Enable Note: 1) When the system enters DeepSleep, the clock will be automatically turned off 2) When the system clock is selected as the current clock, the current clock cannot be turned off	RW

3.5.5 Sequence Control Register 3 (SYSCTRL_CR3)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CR3															

Bit	Marking	Place name	Function	Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value	
b15~b0	CR3	Sequence control	The register SYSCTRL_CR0/1/2 protects the series control register, first write 0x5A5A to SYSCTRL_CR3, and then write 0xA5A5 to start the write operation to the register SYSCTRL_CR0/1/2, as long as the register SYSCTRL_CR0/1/2 is written, this protection bit will automatically restore Protected state, need to rewrite the sequence to open the protection.	WO

3.5.6 RCH Control Register (SYSCTRL_RCHCR)

Reset value: 0x0000 0002

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res												DIV			

Bit	Marking	Place name	Function	Read and write
b31~b4	Res	Reserved bit	Reserved, must remain at reset value	
b3~b0	DIV	RCH output configuration	frequency 0000: 12.00 MHz 0001: 6.00 MHz 0010: 4.00 MHz 0011: 3.00 MHz 0100: 2.40 MHz 0101: 2.00 MHz 0110: 1.71 MHz 0111: 1.50 MHz 1000: 24.00 MHz Other values: disabled	RW

3.5.7 RCL Control Register (SYSCTRL_RCLCR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res														WAITCYCLE	

Bit	Marking	Place name	Function	Read and write
b31~b2	Res	Reserved bit	Reserved, must remain at reset value	
b1~b0	WAITCYCLE	Internal low-speed clock stabilization time selection	RCL 11: 256 cycles 10: 64 cycles 01: 16 cycles 00: 4 cycles	RW

3.5.8 XTH Control Register (SYSCTRL_XTHCR)

Reset value: 0x0000 0002

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res							DETIE	DETTIME	DETEN	WAITCYCLE		RANGE		DRIVER	

Bit	Marking	Place name	Function	Read and write
b31~b9	Res	Reserved bit	Reserved, must remain at reset value	
b8	DETIE	Interrupt Enable Control	1: Enable Fault and Fail interrupts 0: Disable Fault and Fail interrupts	RW
b7	DETTIME	Crystal failure detection time configuration	0: 2ms detection interval 1: 8ms detection interval	RW
b6	DETEN	Oscillation failure detection and work failure detection enabling control	1: Enable 0: Prohibited Note: Enabling this function requires software to enable the RCL clock enable	RW
b5~b4	WAITCYCLE	External high-speed clock XTH stabilization time selection	00: 8192 cycles 01: 32768 cycles 10: 131072 cycles 11: 262144 cycles Note: It is highly recommended to set the stabilization time of XTH to 11. If the XTH stabilization time is insufficient, the system will not work stably when performing clock switching or waking up from deep sleep.	RW
b3~b2	RANGE	External crystal oscillator operating frequency selection	11: 20M~32M 10: 16M~24M 01: 8M~16M 00: 8M~12M Note: The above configurations are recommended configurations, and users should make appropriate adjustments according to actual applications.	RW
b1~b0	DRIVER	External Crystal Oscillator Drive Capability Selection	11: The strongest driving ability 10: Default drive capacity (recommended value) 01: Weak driving ability 00: Weakest drive capability Note: It is necessary to select an appropriate driving capability according to the characteristics of the crystal oscillator, the load capacitance and the parasitic parameters of the circuit board. The greater the driving capability, the greater the power consumption; the weaker the driving capability, the smaller the power consumption.	RW

3.5.9 PLL Control Register (SYSCTRL_PLLCR)

Reset value: 0x000C 02C0

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res										MULM					
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
LKEN	BYPASS	DIVN						OEN	WAITCYCLE			OD		SRC	

Bit	Marking	Place name	Function	Read and write
b31~b22	Res	Reserved bit	Reserved, must remain at reset value	
b21~b16	MULM	Loop divider	2 ≤ MULM ≤ 63	RW
b15	LKEN	Lock enable	1: Force LOCK=1	RW
b14	BYPASS	Bypass mode	1: FOUT = FIN/DIVN	RW
b13~b8	DIVN	Pre divider	1 ≤ DIVN ≤ 63	RW
b7	OEN	Output enable	Output enable	RW
b6~b4	WAITCYCLE	PLL Stability Time Selection	000: 128 PLL cycles 001: 256 PLL cycles 010: 512 PLL cycles 011: 1024 PLL cycles 100: 2048 PLL cycles 101: 4096 PLL cycles 110: 8192 PLL cycles 111: 16384 PLL cycles	RW
b3~b2	OD	Output control	Divided by 00: 1 01: 2 10: 4 11: 8	RW
b1~b0	SRC	Input Selection	00: RCH clock 01: XTH clock (determined by SYSCTRL_CR2.EXXTH input from pin PD00 or internally generated by crystal oscillator) 1X: XTH internal clock	RW
Note: When the input clock is selected as the RCH clock, this input frequency cannot be 24MHz.				

3.5.10 RCH Trim Value Register (SYSCTRL_RCHTRIM)

Reset value: the calibration value where the 24M calibration value address is located (power-on and power-off reset and external reset pin reset will reload the value)

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res							TRIM								

Bit	Marking	Place name	Function	Read and write
b31~b9	Res	Reserved bit	Reserved, must remain at reset value	
b8~b0	TRIM	Internal high-speed clock frequency adjustment, changing the value of this register can adjust the basic frequency of RCH.	Every time the register value increases by 1, the output frequency of RCH will increase by about 0.2%, and the total adjustment range is 22 - 24 MHz. The 24MHz calibration value has been saved in FLASH, and the precise frequency can be obtained by reading the calibration value in FLASH and writing it into RCHTRIM. 24M calibration value address: 0x010011E4 - 0x010011E5 22M calibration value address: 0x010011E0 - 0x010011E1	RW

3.5.11 RCL Trim Value Register (SYSCTRL_RCLTRIM)

Reset value: the calibration value where the 32.768k calibration value address is located (power-on and power-off reset and external reset pin reset will reload the value)

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res							TRIM								

Bit	Marking	Place name	Function	Read and write
b31~b10	Res	Reserved bit	Reserved, must remain at reset value	
b9~b0	TRIM	Internal low-speed clock frequency adjustment, changing the value of this register can adjust the basic frequency of RCL.	Accurate frequency can be obtained by reading out the calibration value in FLASH and writing it into RCLTRIM. 38.4k calibration value address: 0x010011EC - 0x010011ED 32.768k calibration value address: 0x010011E8 - 0x010011E9	RW

3.5.12 Clock Status Register (SYSCTRL_STATE)

Reset value: 0x0000 0001

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res						XTHF AULT	XTHF AIL	Res		PLL OCK	PLL	Res	RCL	XTH	RCH

Bit	Marking	Place name	Function	Read and write
b31~b10	Res	Reserved bit	Reserved, must remain at reset value	
			0: works fine 1: Disabled during work, write 0 to clear	
b9	XTHFAULT	XTH working failure flag	Note: If XTH is used as the system clock and the invalid clock is automatically switched, use software to switch the clock to RCH after the invalidation and then clear the flag.	RO
b8	XTHFAIL	XTH start-up sign	0: normal vibration 1: Oscillation failure, write 0 to clear	RO
b7~b6	Res	Reserved bit	Reserved, must remain at reset value	
b5	PLLLOCK	PLL lock status	0: Unlocked 1: locked	RO
b4	PLL	PLL steady state	0: Not stable 1: Already stable	RO
b3	Res	Reserved bit	Reserved, must remain at reset value	
b2	RCL	RCL clock steady state	0: Not stable 1: Already stable	RO
b1	XTH	XTH clock steady state	0: Not stable 1: Already stable	RO
b0	RCH	RCH clock steady state	0: Not stable 1: Already stable	RO

3.5.13 APB0 Peripheral Clock Control Register (SYSCTRL_APBEN0)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res				ATIM 3	Res		ATIM 0	Res	CTIM 2	CTIM 1	CTIM 0	Res			
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
SPI1	SPI0	Res		I2C1	I2C0	Res						USAR T3	USAR T2	USAR T1	USAR T0

Bit	Marking	Place name	Function	Read and write
b31~b28	Res	Reserved bit	Reserved, must remain at reset value	
b27	ATIM3	ATIM3 module clock enable	1: Enable 0: Prohibited	RW
b26~b25	Res	Reserved bit	Reserved, must remain at reset value	
b24	ATIM0	ATIM0 module clock enable	1: Enable 0: Prohibited	RW
b23	Res	Reserved bit	Reserved, must remain at reset value	
b22	CTIM2	CTIM2 module clock enable	1: Enable 0: Prohibited	RW
b21	CTIM1	CTIM1 module clock enable	1: Enable 0: Prohibited	RW
b20	CTIM0	CTIM0 module clock enable	1: Enable 0: Prohibited	RW
b19~b16	Res	Reserved bit	Reserved, must remain at reset value	
b15	SPI1	SPI1 module clock enable	1: Enable 0: Prohibited	RW
b14	SPI0	SPI0 module clock enable	1: Enable 0: Prohibited	RW
b13~b12	Res	Reserved bit	Reserved, must remain at reset value	
b11	I2C1	I2C1 module clock enable	1: Enable 0: Prohibited	RW
b10	I2C0	I2C0 module clock enable	1: Enable 0: Prohibited	RW
b9~b4	Res	Reserved bit	Reserved, must remain at reset value	
b3	USART3	USART 3 module clock enable	1: Enable 0: Prohibited	RW
b2	USART2	USART2 module clock enable	1: Enable 0: Prohibited	RW
b1	USART1	USART1 module clock enable	1: Enable 0: Prohibited	RW
b0	USART0	USART0 module clock enable	1: Enable 0: Prohibited	RW

3.5.14 APB1 Peripheral Clock Control Register (SYSCTRL_APBEN1)

Reset value: 0x0000 8000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res				LVD	VC	ADC1	ADC0	Res				IWDT	WWD T	Res	CTRIM

Bit	Marking	Place name	Function	Read and write
b31~b12	Res	Reserved bit	Reserved, must remain at reset value	
b11	LVD	LVD module clock enable	1: Enable 0: Prohibited	RW
b10	VC	VC module clock enable	1: Enable 0: Prohibited	RW
b9	ADC1	ADC1 module clock enable	1: Enable 0: Prohibited	RW
b8	ADC0	ADC0 module clock enable	1: Enable 0: Prohibited	RW
b7~b4	Res	Reserved bit	Reserved, must remain at reset value	
b3	IWDT	IWDT module clock enable	1: Enable 0: Prohibited	RW
b2	WWD	WWD module clock enable	1: Enable 0: Prohibited	RW
b1	Res	Reserved bit	Reserved, must remain at reset value	
b0	CTRIM	CTRIM module clock enable	1: Enable 0: Prohibited	RW

3.5.15 AHB0 Peripheral Clock Control Register (SYSCTRL_AHBEN0)

Reset value: 0x0000 0020

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res										FLASH	Res	CRC	Res	DMA	C

Bit	Marking	Place name	Function	Read and write
b31~b6	Res	Reserved bit	Reserved, must remain at reset value	
b5	FLASH	FLASH control module clock enable	1: Enable 0: Prohibited	RW
b4~b3	Res	Reserved bit	Reserved, must remain at reset value	
b2	CRC	CRC module clock enable	1: Enable 0: Prohibited	RW
b1	Res	Reserved bit	Reserved, must remain at reset value	
b0	DMAC	DMAC module clock enable	1: Enable 0: Prohibited	RW

3.5.16 AHB1 Peripheral Clock Control Register (SYSCTRL_AHBEN1)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res												GPIO D	GPIO C	GPIO B	GPIO A

Bit	Marking	Place name	Function	Read and write
b31~b4	Res	Reserved bit	Reserved, must remain at reset value	
b3	GPIOD	GPIOD module clock enable	1: Enable 0: Prohibited	RW
b2	GPIOC	GPIOC module clock enable	1: Enable 0: Prohibited	RW
b1	GPIOB	GPIOB module clock enable	1: Enable 0: Prohibited	RW
b0	GPIOA	GPIOA module clock enable	1: Enable 0: Prohibited	RW

3.5.17 Timer Debug Halt Control Register (SYSCTRL_DEBUG)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res			WWD T	IWDT	CTRI M	Res			CTIM 2	CTIM 1	CTIM 0	ATIM 3	Res		ATIM 0

Bit	Marking	Place name	Function	Read and write
b31~b13	Res	Reserved bit	Reserved, must remain at reset value	
b12	WWD T	When debugging, the WWD T counting function configures the	1: In the SWD debugging interface, the counting function is suspended 0: In the SWD debugging interface, the normal counting function	RW
b11	IWDT	When debugging, the IWDT counting function configuration	1: In the SWD debugging interface, the counting function is suspended 0: In the SWD debugging interface, the normal counting function	RW
b10	CTRI M	When debugging, the CTRI M counting function configuration	1: In the SWD debugging interface, the counting function is suspended 0: In the SWD debugging interface, the normal counting function	RW
b9~b7	Res	Reserved bit	Reserved, must remain at reset value	
b6	CTIM2	When debugging, the CTIM2 counting function is configured	1: In the SWD debugging interface, the counting function is suspended 0: In the SWD debugging interface, the normal counting function	RW
b5	CTIM1	When debugging, the CTIM1 counting function configures the	1: In the SWD debugging interface, the counting function is suspended 0: In the SWD debugging interface, the normal counting function	RW
b4	CTIM0	When debugging, the CTIM0 count function configures the	1: In the SWD debugging interface, the counting function is suspended 0: In the SWD debugging interface, the normal counting function	RW
b3	ATIM3	When debugging, ATIM3 counting function configuration	1: In the SWD debugging interface, the counting function is suspended 0: In the SWD debugging interface, the normal counting function	RW
b2~b1	Res	Reserved bit	Reserved, must remain at reset value	
b0	ATIM0	When debugging, the ATIM0 counting function configuration	1: In the SWD debugging interface, the counting function is suspended 0: In the SWD debugging interface, the normal counting function	RW

3.5.18 GPIO Auxiliary Control Register 0 (SYSCTRL_IOAUX0)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res	MCOSEL			Res	MCODIV			Res	PCLK1DIV			Res	PCLK0DIV		

Bit	Marking	Place name	Function	Read and write
b31~b15	Res	Reserved bit	Reserved, must remain at reset value	
b14~b12	MCOSEL	MCO pin clock selection	000: RCH analog output 001: XTH analog output 010: RCL analog output 011: reserved 100: PLL analog output 101: RC150K analog output 110: RC10K analog output 111: reserved	RW
b11	Res	Reserved bit	Reserved, must remain at reset value	
b10~b8	MCODIV	MCO frequency division control	000: turn off the output 001: direct output 010: 8 frequency division output 011: 64 frequency division output Others: 256 frequency division output	RW
b7	Res	Reserved bit	Reserved, must remain at reset value	
b6~b4	PCLK1DIV	PCLK1 frequency division control	000: turn off the output 001: direct output 010: 2 frequency division output 011: 4 frequency division output Others: 8 frequency division output	RW
b3	Res	Reserved bit	Reserved, must remain at reset value	
b2~b0	PCLK0DIV	PCLK0 frequency division control	000: turn off the output 001: direct output 010: 2 frequency division output 011: 4 frequency division output Others: 8 frequency division output	RW

3.5.19 GPIO Auxiliary Control Register 1 (SYSCTRL_IOAUX1)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res	CTIM2ETR			Res	CTIM1ETR			Res	CTIM0ETR			Res	ATIM0ETR		

Bit	Marking	Place name	Function	Read and write
b31~b15	Res	Reserved bit	Reserved, must remain at reset value	
b14~b12	CTIM2ETR	CTIM2 ETR settings	See GPIO chapter or timer chapter description	RW
b11	Res	Reserved bit	Reserved, must remain at reset value	
b10~b8	CTIM1ETR	CTIM1 ETR settings	See GPIO chapter or timer chapter description	RW
b7	Res	Reserved bit	Reserved, must remain at reset value	
b6~b4	CTIM0ETR	CTIM0 ETR setting	See GPIO chapter or timer chapter description	RW
b3	Res	Reserved bit	Reserved, must remain at reset value	
b2~b0	ATIM0ETR	ATIM0 ETR settings	See GPIO chapter or timer chapter description	RW

3.5.20 GPIO Auxiliary Control Register 2 (SYSCTRL_IOAUX2)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res	CTIM2CAP3			Res	CTIM1CAP3			Res	CTIM0CAP3			Res	ATIM0CAP0		

Bit	Marking	Place name	Function	Read and write
b31~b15	Res	Reserved bit	Reserved, must remain at reset value	
b14~b12	CTIM2CAP3	CTIM2 CAP settings	See GPIO chapter or timer chapter description	RW
b11	Res	Reserved bit	Reserved, must remain at reset value	
b10~b8	CTIM1CAP3	CTIM1 CAP settings	See GPIO chapter or timer chapter description	RW
b7	Res	Reserved bit	Reserved, must remain at reset value	
b6~b4	CTIM0CAP3	CTIM0 CAP setting	See GPIO chapter or timer chapter description	RW
b3	Res	Reserved bit	Reserved, must remain at reset value	
b2~b0	ATIM0CAP0	ATIM0 CAP settings	See GPIO chapter or timer chapter description	RW

4 Reset Controller (RESET)

4.1 Reset Controller Introduction

This product has 7 reset signal sources, each reset signal can make the CPU run again, most of the registers will be reset to the reset value, and the program will start to execute from the reset vector.

- Power-on power-down reset POR PDR
- External Reset pin, low level is reset signal
- IWDT reset
- WWDT reset
- LVD low voltage reset
- Cortex-M4 SYSRESETREQ software reset
- Cortex-M4 LOCKUP hardware reset

Each reset source is indicated by a corresponding reset flag; query the SYSCTRL_RSTFLAG register to obtain the last reset source when the reset is completed. Whenever the polling of the SYSCTRL_RSTFLAG register is completed, this register should be cleared so that the register can normally indicate its reset source at the next reset.

The figure below describes the reset sources for each area.

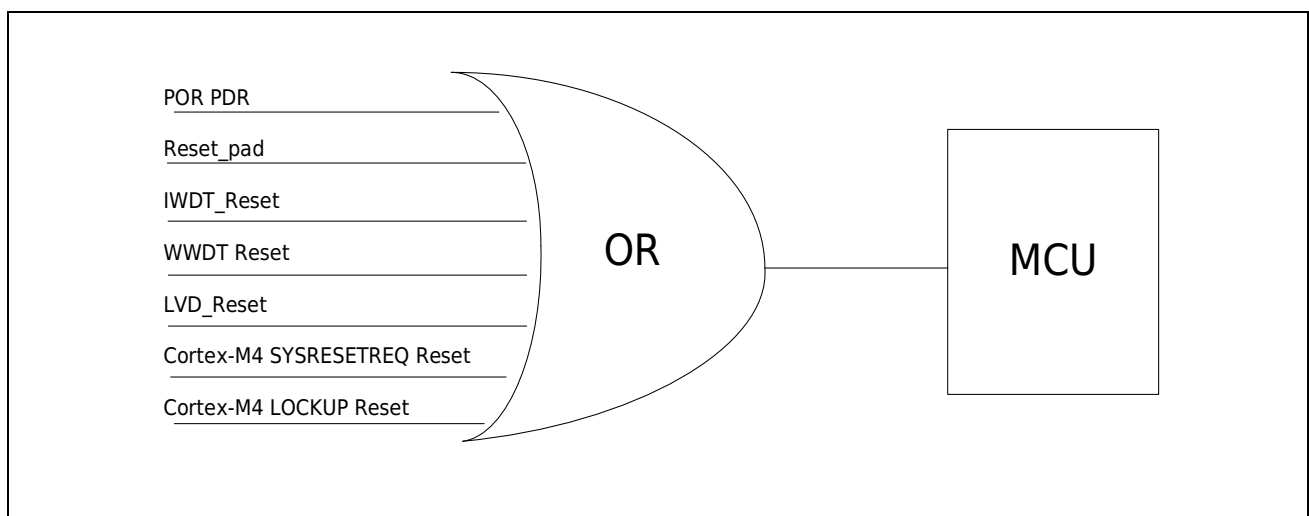


Figure 4-1 Reset Source Schematic

4.1.1 Power-on and power-off reset POR PDR

This product has two power supply areas: VCC area and VCORE area.

All analog modules and IO work in the VCC area; other modules work in the VCORE area.

When the VCC area is powered on, when the VCC voltage is lower than the POR threshold voltage, a VCC power domain reset signal will be generated; when the VCC area is powered off, when the VCC voltage is lower than the PDR threshold voltage, a VCC power domain reset signal will be generated.

When the VCORE area is powered on, when the VCORE voltage is lower than the POR threshold voltage, a VCORE domain reset signal will be generated; when the VCORE area is powered off, when the VCORE voltage is lower than the PDR threshold voltage, a VCORE domain reset signal will be generated.

Both the VCC power domain reset signal and the VCORE domain reset signal will reset the registers of the chip to the initialization state.

4.1.2 External reset pin reset

A system reset is generated when the external reset pin detects a low level. The reset pin has a built-in pull-up resistor and integrates a glitch filter circuit. The glitch filter circuit will filter the glitch signal less than 20μs (typical value), therefore, the low-level signal added to the reset pin must be greater than 20μs to ensure reliable reset of the chip.

4.1.3 WDT reset

Watchdog reset, please refer to chapter IWDT/WWDT.

4.1.4 LVD low voltage reset

LVD reset, please refer to chapter LVD.

4.1.5 Cortex-M4 SYSRESETREQ reset

Cortex-M4 software reset.

4.1.6 Cortex-M4 LOCKUP reset

When Cortex-M4 encounters a serious exception, it will stop its PC pointer at the current address, lock itself, and reset the entire CORE area after a few clock cycle delays.

4.2 Register description

4.2.1 Register list

Base address: 0x4000 B800

Table 4-1 Reset register list

Register name	Offset address	Reset value
SYSCTRL_APB_RST0 (APB0 Peripheral Reset Control Register)	0x40	0x0970 CC0F
SYSCTRL_APB_RST1 (APB1 Peripheral Reset Control Register)	0x44	0x0000 0F0D
SYSCTRL_AHB_RST0 (AHB0 Peripheral Reset Control Register)	0x48	0x0000 0025
SYSCTRL_AHB_RST1 (AHB1 Peripheral Reset Control Register)	0x4C	0x0000 000F
SYSCTRL_RSTFLAG (Reset Flag Register)	0x60	0x0000 0003

4.2.2 APB0 Peripheral Reset Control Register (SYSCTRL_APB0RST0)

Reset value: 0x0970 CC0F

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res				ATIM 3	Res		ATIM 0	Res	CTIM 2	CTIM 1	CTIM 0	Res			
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
SPI1	SPI0	Res		I2C1	I2C0	Res						USAR T3	USAR T2	USAR T1	USAR T0

Bit	Marking	Place name	Function	Read and write
b31~b28	Res	Reserved bit	Reserved, must remain at reset value	
b27	ATIM3	ATIM3 module reset	1: no action 0: reset	RW
b26~b25	Res	Reserved bit	Reserved, must remain at reset value	
b24	ATIM0	ATIM0 module reset	1: no action 0: reset	RW
b23	Res	Reserved bit	Reserved, must remain at reset value	
b22	CTIM2	CTIM2 module reset	1: no action 0: reset	RW
b21	CTIM1	CTIM1 module reset	1: no action 0: reset	RW
b20	CTIM0	CTIM0 module reset	1: no action 0: reset	RW
b19~b16	Res	Reserved bit	Reserved, must remain at reset value	
b15	SPI1	SPI1 module reset	1: no action 0: reset	RW
b14	SPI0	SPI0 module reset	1: no action 0: reset	RW
b13~b12	Res	Reserved bit	Reserved, must remain at reset value	
b11	I2C1	I2C1 module reset	1: no action 0: reset	RW
b10	I2C0	I2C0 module reset	1: no action 0: reset	RW
b9~b4	Res	Reserved bit	Reserved, must remain at reset value	
b3	USART3	USART3 module reset	1: no action 0: reset	RW
b2	USART2	USART2 module reset	1: no action 0: reset	RW
b1	USART1	USART1 module reset	1: no action 0: reset	RW
b0	USART0	USART0 module reset	1: no action 0: reset	RW

4.2.3 APB1 Peripheral Reset Control Register (SYSCTRL_APB1RST1)

Reset value: 0x0000 0F0D

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res				LVD	VC	ADC1	ADC0	Res				IWDT	WWD T	Res	CTRIM

Bit	Marking	Place name	Function	Read and write
b31~b12	Res	Reserved bit	Reserved, must remain at reset value	
b11	LVD	LVD module reset	1: no action 0: reset	RW
b10	VC	VC module reset	1: no action 0: reset	RW
b9	ADC1	ADC1 module reset	1: no action 0: reset	RW
b8	ADC0	ADC0 module reset	1: no action 0: reset	RW
b7~b4	Res	Reserved bit	Reserved, must remain at reset value	
b3	IWDT	IWDT module reset	1: no action 0: reset	RW
b2	WWD	WWD module reset	1: no action 0: reset	RW
b1	Res	Reserved bit	Reserved, must remain at reset value	
b0	CTRIM	CTRIM module reset	1: no action 0: reset	RW

4.2.4 AHB0 Peripheral Reset Control Register (SYSCTRL_AHB1RST0)

Reset value: 0x0000 0025

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res										FLASH	Res		CRC	Res	DMA C

Bit	Marking	Place name	Function	Read and write
b31~b6	Res	Reserved bit	Reserved, must remain at reset value	
b5	FLASH	FLASH control module reset	1: no action 0: reset	RW
b4~b3	Res	Reserved bit	Reserved, must remain at reset value	
b2	CRC	CRC module reset	1: no action 0: reset	RW
b1	Res	Reserved bit	Reserved, must remain at reset value	
b0	DMAC	DMAC module reset	1: no action 0: reset	RW

4.2.5 AHB1 Peripheral Reset Control Register (SYSCTRL_AHBRST1)

Reset value: 0x0000 000F

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res												GPIO D	GPIO C	GPIO B	GPIO A

Bit	Marking	Place name	Function	Read and write
b31~b4	Res	Reserved bit	Reserved, must remain at reset value	
b3	GPIOD	GPIOD module reset	1: no action 0: reset	RW
b2	GPIOC	GPIOC module reset	1: no action 0: reset	RW
b1	GPIOB	GPIOB module reset	1: no action 0: reset	RW
b0	GPIOA	GPIOA module reset	1: no action 0: reset	RW

4.2.6 Reset Flag Register (SYSCTRL_RSTFLAG)

Reset value: 0x0000 0003

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res								NRST	SYSREQ	LOCKUP	WWDT	IWDT	LVD	CORE	VCC

Bit	Marking	Place name	Function	Read and write
b31~b8	Res	Reserved bit	Reserved, must remain at reset value	
b7	NRST	NRESET port reset flag	1: A port reset occurred 0: No port reset occurs Write 0 to clear, write 1 to have no effect Note: When powered on, this value is related to the external reset circuit.	RW
b6	SYSREQ	Cortex-M4 CPU software reset flag	1: Cortex-M4 CPU software reset occurred 0: No Cortex-M4 CPU software reset occurs Write 0 to clear, write 1 to have no effect	RW
b5	LOCKUP	Cortex-M4 CPU Lockup reset flag	1: Cortex-M4 CPU Lockup reset occurred 0: No Cortex-M4 CPU Lockup reset occurs Write 0 to clear, write 1 to have no effect	RW
b4	WWDT	WWDT reset flag	1: WWDT reset occurs 0: No WWDT reset occurs Write 0 to clear, write 1 to have no effect	RW
b3	IWDT	IWDT reset flag	1: IWDT reset occurs 0: No IWDT reset occurs Write 0 to clear, write 1 to have no effect	RW
b2	LVD	LVD reset flag	1: LVD reset occurs 0: No LVD reset occurs Write 0 to clear, write 1 to have no effect	RW
b1	CORE	VCORE domain reset flag	1: VCORE domain reset occurred 0: No reset of VCORE domain occurs Write 0 to clear, write 1 to have no effect	RW
b0	VCC	VCC power domain reset flag	1: VCC power domain reset occurred 0: No reset occurs in VCC power domain Write 0 to clear, write 1 to have no effect	RW

Note: The reset value is 0x03 at power-on reset. After the program determines the reset source, it should clear the register so that it can determine the reset source the next time a reset occurs.

5 Interrupt Controller (NVIC)

5.1 Overview

The Cortex-M4 processor has a built-in nested vectored interrupt controller (NVIC). This product supports up to 64 interrupt request (IRQ) inputs and 1 non-maskable interrupt (NMI) input (not used in this product system.). In addition, the processor supports several internal exceptions.

Each exception source has a separate exception number, each exception type has a corresponding priority, some exceptions have a fixed priority, while others are programmable. The details are shown in the following table:

Table 5-1 Cortex-M4 processor interrupt overview

Exception number	Exception type	Priority	Description
1	Reduction	-3 (highest)	Reduction
2	NMI	-2	Non-maskable interrupt (not used in this system)
3	Hardware error	-1	Error handling exception
4-10	Keep	NA	...
11	SVC	Programmable	Invoke the hypervisor through the SVC command
12-13	Keep	NA	...
14	PendSV	Programmable	Suspendable requests for system services
15	SysTick	Programmable	SysTick timer
16	Interrupt #0	Programmable	External Interrupt #0
17	Interrupt #1	Programmable	External Interrupt #1
...	...	...	...
79	Interrupt #63	Programmable	External interrupt #63

This chapter only introduces the 64 external interrupt requests of the processor (interrupt #0 to interrupt #63) in detail, and the specific situation of the internal exception of the processor can refer to other related documents. At the same time, this chapter only discusses the interrupt handling mechanism of the NVIC in the processor core, and the interrupt generation mechanism of the peripheral module itself is not discussed here.

5.2 Interrupt priority

Each external interrupt corresponds to a priority register, each priority is 3 bits wide, and uses the highest 3 bits of the interrupt priority register, each register occupies 1 byte (8 bits). Under this setting, the available priorities are 0x00 (highest), 0x20, 0x40, 0x60, 0x80, 0xA0, 0xC0, and 0xE0 (lowest).

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Used			Not used, read as 0				

Figure 5-1 Only the upper 3 bits of the priority register are used

Preemption occurs when the processor is already running another interrupt handler and the new interrupt has a higher priority than the one currently being executed. The running interrupt processing will be suspended and the new interrupt will be executed instead. This process is usually called interrupt nesting. After the new interrupt is executed, the previous interrupt processing will continue and return to the program thread after it finishes.

If the processor is running another interrupt handler with the same or higher priority, the new interrupt will wait and enter the pending state. Pending interrupts will wait until the current interrupt level changes, for example, after the currently running interrupt handler finishes returning, the current priority is lowered to be lower than the pending interrupt.

If two interrupts occur at the same time and they have the same priority, the interrupt with the lower interrupt number will be executed first. For example, if interrupt #0 and interrupt #1 are enabled and have the same priority, when they are triggered at the same time, interrupt #0 will be executed first.

5.3 Interrupt digraph

When the Cortex-M4 processor wants to process an interrupt service request, it needs to first determine the starting address of the exception handling. The required information is called a vector table, as shown in the figure below. The vector table is stored at the beginning of the memory space and contains the exception (interrupt) vectors of the exceptions (interrupts) available in the system, as well as the initial value of the main stack pointer (MSP).

Memory address		Exception number
0x0000/ 004C	Interrupt #3 vector	19
0x0000 0048	Interrupt #2 vector	18
0x0000 0044	Interrupt #1 vector	17
0x0000 0040	Interrupt #0 vector	16
0x0000/ 003C	SysTick vector	15
0x0000 0038	PendSV vector	14
0x0000 0034	Unused	13
0x0000 0030	Unused	12
0x0000/ 002C	SVC vector	11
0x0000 0028	Unused	10
0x0000 0024	Unused	9
0x0000 0020	Unused	8
0x0000/ 001C	Unused	7
0x0000 0018	Unused	6
0x0000 0014	Unused	5
0x0000 0010	Unused	4
0x0000/ 000C	Hardware error exception	3
0x0000 0008	NMI vector	2
0x0000 0004	Reset vector	1
0x0000 0000	MSP initial value	0

Figure 5-2 Interrupt Vector Table

Among them, the storage order of the interrupt vector is consistent with the interrupt number. Since each vector is 1 word (4 bytes), the address of the interrupt vector is the interrupt number multiplied by 4, and each interrupt vector is the starting address of the interrupt processing.

5.4 Interrupt Input and Suspend Behavior

In the NVIC module of the Cortex-M4 processor, each interrupt input corresponds to a pending status register, and each register has only 1 bit, which is used to save the interrupt request, regardless of whether the request has been confirmed. When the processor starts servicing the interrupt, the hardware will automatically clear the pending status bit.

The peripherals of this system use level-triggered interrupt output. When an interrupt event occurs, the interrupt signal will be confirmed because the peripheral is connected to the NVIC. This signal remains high until the processor executes the interrupt service and clears the peripheral's interrupt signal. Inside the NVIC, when an interrupt is detected, the pending status of the interrupt will be set, and when the processor receives the interrupt and starts executing the interrupt service routine, the pending status will be cleared. The process is shown in the figure below:

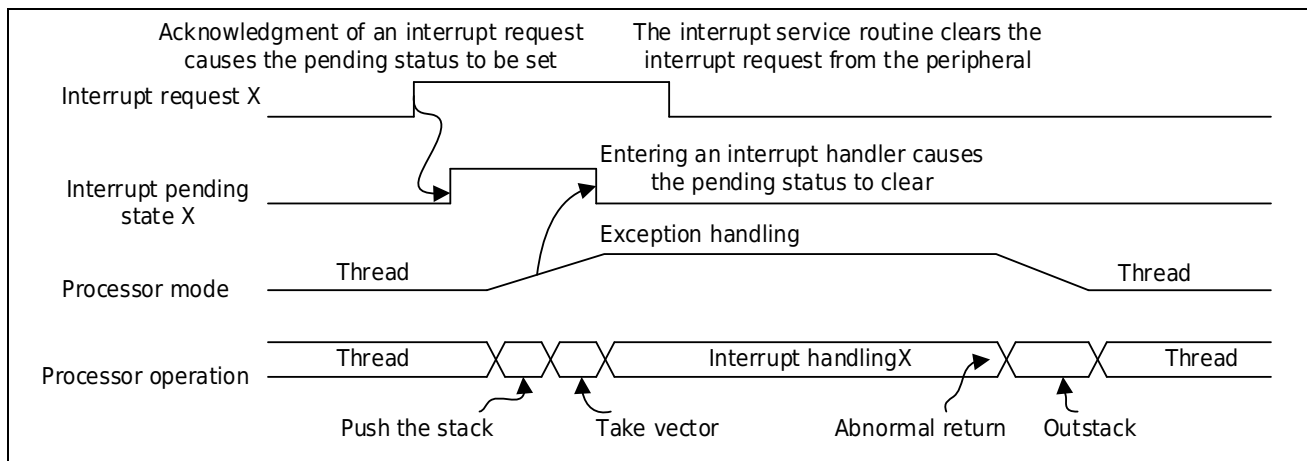


Figure 5-3 Interrupt active and pending states

If the interrupt request is not executed immediately and is cleared by software before being acknowledged, the processor ignores the request and does not perform interrupt processing. The interrupt pending status can be cleared by writing the NVIC_ICPR register, which is useful when setting up a peripheral that may have generated an interrupt request before setting it.

If the peripheral is still holding an interrupt request when software clears the pending state, the pending state is also generated immediately. The process is shown in the figure below:

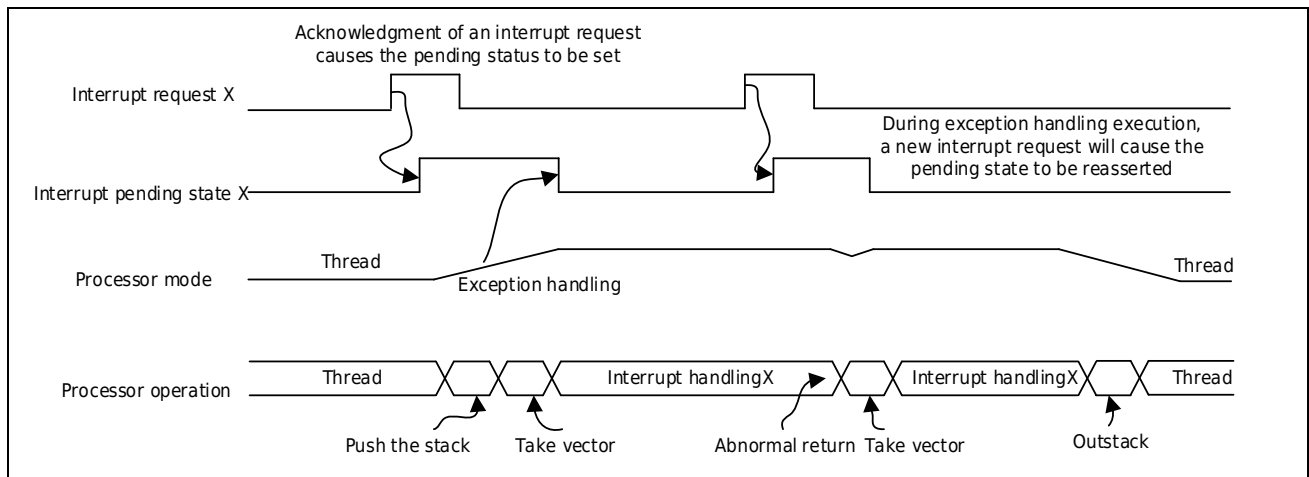


Figure 5-6 Interrupt pending interrupts generated during interrupt processing can also be acknowledged

5.5 Interrupt wait

Typically, the interrupt latency of the NVIC is 16 cycles. This wait time starts from the processor clock cycle when the interrupt is acknowledged, and ends when the interrupt handler starts executing. Computing interrupt waiting requires the following prerequisites:

- The interrupt is enabled and not masked by PRIMASK or other exception handling in progress.
- The memory system does not have any wait states. Bus transfers are used for interrupt processing, stack push, vector fetches, or instruction fetches at the start of interrupt processing. If the memory system needs to wait, the wait states generated when bus transfers occur may delay interrupts.

The following situations may cause different interrupt waits:

- The end of the interrupt is chained. If another interrupt request is generated when the interrupt returns, the processor will skip the process of popping and pushing the stack, thus reducing the interrupt waiting time.
- Delayed arrival. If an interrupt occurs, another low-priority interrupt is being pushed to the stack. Due to the existence of the delayed arrival mechanism, the high-priority interrupt will be executed first, which will also reduce the waiting time of the high-priority interrupt.

5.6 Interrupt source

Because the NVIC of the Cortex-M4 processor supports up to 128 external interrupts, and in this system, 64 external interrupt sources are used, of which NMI (non-maskable interrupt) is not used. The corresponding relationship between all external interrupt sources of this system and NVIC interrupt input is shown in the following table:

Table 5-2 Correspondence between external interrupt and NVIC interrupt input

NVIC interrupt input	External interrupt source	Operating mode	Sleep mode	Deep sleep mode
Interrupt #0	CTRIM	✓	✓	✓
Interrupt #1	VC	✓	✓	✓
Interrupt #2	LVD	✓	✓	✓
Interrupt #3	GPIOA	✓	✓	✓
Interrupt #4	GPIOB	✓	✓	✓
Interrupt #5	GPIOC	✓	✓	✓
Interrupt #6	GIOD	✓	✓	✓
Interrupt #7	IWDT	✓	✓	✓
Interrupt #8	WWDT	✓	✓	-
Interrupt #9	CLK_FAULT	✓	✓	-
Interrupt #10	ADC	✓	✓	-
Interrupt #11	FLASHCFG	✓	✓	-
Interrupt #12	USART0_RX	✓	✓	-
Interrupt #13	USART0_TX	✓	✓	-
Interrupt #14	USART0_FLAG	✓	✓	-
Interrupt #15	USART1_RX	✓	✓	-
Interrupt #16	USART1_TX	✓	✓	-
Interrupt #17	USART1_FLAG	✓	✓	-
Interrupt #18	USART2_RX	✓	✓	-
Interrupt #19	USART2_TX	✓	✓	-
Interrupt #20	USART2_FLAG	✓	✓	-
Interrupt #21	USART3_RX	✓	✓	-
Interrupt #22	USART3_TX	✓	✓	-
Interrupt #23	USART3_FLAG	✓	✓	-
Interrupt #24	I2C0	✓	✓	-
Interrupt #25	I2C1	✓	✓	-
Interrupt #26	SPI0_TX	✓	✓	-
Interrupt #27	SPI0_RX	✓	✓	-
Interrupt #28	SPI0_SSR	✓	✓	-
Interrupt #29	SPI0_ERR	✓	✓	-

NVIC interrupt input	External interrupt source	Operating mode	Sleep mode	Deep sleep mode
Interrupt #30	SPI1_TX	✓	✓	-
Interrupt #31	SPI1_RX	✓	✓	-
Interrupt #32	SPI1_SSR	✓	✓	-
Interrupt #33	SPI1_ERR	✓	✓	-
Interrupt #34	GTIM0_UEV_ETR_BTIMA	✓	✓	-
Interrupt #35	GTIM0_CH01_BTIMB	✓	✓	-
Interrupt #36	GTIM0_CH23_BTIMC	✓	✓	-
Interrupt #37	GTIM1_UEV_ETR_BTIMA	✓	✓	-
Interrupt #38	GTIM1_CH01_BTIMB	✓	✓	-
Interrupt #39	GTIM1_CH23_BTIMC	✓	✓	-
Interrupt #40	GTIM2_UEV_ETR_BTIMA	✓	✓	-
Interrupt #41	GTIM2_CH01_BTIMB	✓	✓	-
Interrupt #42	GTIM2_CH23_BTIMC	✓	✓	-
Interrupt #43	ATIM0_UEV	✓	✓	-
Interrupt #44	ATIM0_CHA	✓	✓	-
Interrupt #45	ATIM0_CHB	✓	✓	-
Interrupt #46	ATIM0_BKETR	✓	✓	-
Interrupt #47	ATIM3_UEV	✓	✓	-
Interrupt #48	ATIM3_CHYA	✓	✓	-
Interrupt #49	ATIM3_CHYB	✓	✓	-
Interrupt #50	ATIM3_BK_ETR_OV_UD_CMP3	✓	✓	-
Interrupt #51	DMA_CH0	✓	✓	-
Interrupt #52	DMA_CH1	✓	✓	-
Interrupt #53	DMA_CH2	✓	✓	-
Interrupt #54	DMA_CH3	✓	✓	-
Interrupt #55	DMA_CH4	✓	✓	-
Interrupt #56	DMA_CH5	✓	✓	-
Interrupt #57~63	NC	-	-	-

Note: Since some module interrupts are multiplexed for the same IRQ interrupt source, when the CPU enters the interrupt operation, it must first determine which module generated the interrupt, and then perform the corresponding interrupt operation.

5.7 Interrupt structure

- IRQ has more than one peripheral interrupt input multiplexing, and the interrupt flag bits of these peripherals must be read separately to determine which peripheral interrupt it is.
- If the peripheral interrupt source has a high level, regardless of whether the NVIC interrupt enable setting register NVIC_IUSER.SETENA is set or not, the interrupt pending setting register NVIC_ISPR.SETPEND will be set, indicating that the corresponding peripheral interrupt source has an interrupt.
- Only when the interrupt enable setting register NVIC_IUSER.SETENA is set, the corresponding interrupt IRQ will respond to the processor and execute the corresponding interrupt program.
- In the interrupt program, the high-level interrupt signal of the peripheral interrupt source must be cleared, and the interrupt pending setting register NVIC_ISPR.SETPEND is automatically cleared by the hardware.
- The interrupt priority registers NVIC_IPR0-NVIC_IPR15 set the priority of 64 interrupt sources, 0x00 has the highest priority, and 0xE0 has the lowest priority. When the priority is the same, the priority is determined by the interrupt number, and the smaller the number, the higher the priority.

5.8 Basic operation of the software

5.8.1 External interrupt enable

Each peripheral module has its own interrupt enable register. When an interrupt operation is required, the peripheral's own interrupt enable must be enabled first. The operation of this enable bit is not discussed in this chapter, please refer to the respective chapter descriptions of the peripheral modules.

5.8.2 NVIC interrupt enable and clear enable

The Cortex-M4 processor supports up to 128 interrupt sources, and each interrupt source corresponds to an interrupt enable bit and a clear enable bit.

You can call the CMSIS-Core function or configuration register to enable and disable interrupts

void NVIC_EnableIRQ(IRQn_Type IRQn) enables external interrupt

void NVIC_DisableIRQ(IRQn_Type IRQn) disables external interrupts

5.8.3 NVIC interrupt pending and clear pending

If an interrupt occurs but cannot be handled immediately, the interrupt request will be pending. The pending state is kept in a register and will remain valid as long as the processor's current priority has not been lowered enough to handle the pending request and the pending state has not been cleared manually.

When the processor starts to enter the interrupt service, the hardware will automatically clear the pending state.

The interrupt pending status can be accessed or modified by operating the interrupt pending set NVIC_ISPR and interrupt pending clear NVIC_ICPR registers. The Interrupt Pending Setup Register allows software to trigger interrupts.

You can call the CMSIS-Core function to clear the interrupt pending

```
void NVIC_ClearPendingIRQ(IRQn_Type IRQn);
```

5.8.4 NVIC interrupt priority

Setting the NVIC_IPRx register determines the priority of IRQx. The programming of the interrupt priority register should be done before the interrupt is enabled, which is usually done at the beginning of the program. Changing the interrupt priority after the interrupt is enabled should be avoided, the results of this situation are unpredictable and are not supported by the Cortex-M4 processor.

You can call the CMSIS-Core function to set the interrupt priority

```
void NVIC_SetPriority (IRQn_Type IRQn,uint32_t priority);
```

5.8.5 NVIC interrupt mask

Some time-sensitive applications need to disable all interrupts in a short period of time, which can be realized by using the interrupt mask register NVIC_PRIMASK. The special register NVIC_PRIMASK has only 1 bit valid and defaults to 0 after reset. When this register is 0, all interrupts and exceptions are enabled; when set to 1, only NMI (not supported by this system) and hardware error exceptions are enabled. In fact, when NVIC_PRIMASK is set to 1, the current priority of the processor is reduced to 0 (the highest priority that can be set).

The NVIC_PRIMASK register can be programmed in a variety of ways. If you use assembly language, you can refer to [Interrupt Mask Register (PRIMASK)] and use assembly instructions to set and clear the NVIC_PRIMASK register. If using C language and CMSIS device driver library, users can use the following functions to set and clear PRIMASK.

```
void enable_irq(void); // clear PRIMASK
```

```
void disable_irq(void); // set PRIMASK
```

5.9 Register description

5.9.1 Register list

Base address: 0xE000 E000

Table 5-3 Interrupt Register List

Register name	Offset address	Reset value
NVIC_ISER0 (Interrupt Enable Setting Register 0)	0x100	0x0000 0000
NVIC_ISER1 (Interrupt Enable Setting Register 1)	0x104	0x0000 0000
NVIC_ICER0 (Interrupt Enable Clear Register 0)	0x180	0x0000 0000
NVIC_ICER1 (Interrupt Enable Clear Register 1)	0x184	0x0000 0000
NVIC_ISPR0 (Interrupt Pending Setup Register 0)	0x200	0x0000 0000
NVIC_ISPR1 (Interrupt Pending Setup Register 1)	0x204	0x0000 0000
NVIC_ICPR0 (Interrupt Pending Clear Register 0)	0x280	0x0000 0000
NVIC_ICPR1 (Interrupt Pending Clear Register 1)	0x284	0x0000 0000
NVIC_IPRx (Interrupt Priority Register) (x=0~15)	0x400~0x43C	0x0000 0000
PRIMASK (Interrupt Mask Register)	-	0x0000 0000

5.9.2 Interrupt Enable Setting Register0 (NVIC_ISER0)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
SETENA[31:16]															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
SETENA[15:0]															
Bit	Marking	Place name	Function												Read and write
b31~b0	SETENA	Interrupt #0 to Interrupt #31 enable settings	Writing "1" is set, writing "0" is invalid [0]: IRQ0 [1]: IRQ1 [2]: IRQ2 [31]: IRQ31												RW

5.9.3 Interrupt Enable Setting Register 1 (NVIC_ISER1)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
SETENA[31:16]															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
SETENA[15:0]															
Bit	Marking	Place name	Function												Read and write
b31~b0	SETENA	Interrupt #32 to Interrupt #63 enable settings	Writing "1" is set, writing "0" is invalid [0]: IRQ32 [1]: IRQ33 [2]: IRQ34 [31]: IRQ63												RW

5.9.4 Interrupt Enable Clear Register 0 (NVIC_ICER0)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
CLRENA[31:16]															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CLRENA[15:0]															
Bit	Marking	Place name	Function												Read and write
b31~b0	CLRENA	Interrupt #0 to Interrupt #31 enable clear	Write "1" to clear, write "0" is invalid [0]: IRQ0 [1]: IRQ1 [2]: IRQ2 [31]: IRQ31												RW

5.9.5 Interrupt Enable Clear Register 1 (NVIC_ICER1)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
CLRENA[31:16]															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CLRENA[15:0]															
Bit	Marking	Place name	Function												Read and write
b31~b0	CLRENA	Interrupt #32 to Interrupt #63 enable clear	Write "1" to clear, write "0" is invalid [0]: IRQ32 [1]: IRQ33 [2]: IRQ34 [31]: IRQ63												RW

5.9.6 Interrupt Pending Setup Register 0 (NVIC_ISPR0)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
SETPEND[31:16]															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
SETPEND[15:0]															
Bit	Marking		Place name				Function							Read and write	
b31~b0	SETPEND		Interrupt #0 to Interrupt #31 pending setup				Writing "1" is set, writing "0" is invalid [0]: IRQ0 [1]: IRQ1 [2]: IRQ2 [31]: IRQ31							RW	

5.9.7 Interrupt Pending Setup Register 1 (NVIC_ISPR1)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
SETPEND[31:16]															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
SETPEND[15:0]															
Bit	Marking		Place name				Function							Read and write	
b31~b0	SETPEND		Interrupt #32 to Interrupt #63 pending setup				Writing "1" is set, writing "0" is invalid [0]: IRQ32 [1]: IRQ33 [2]: IRQ34 [31]: IRQ63							RW	

5.9.8 Interrupt Pending Clear Register 0 (NVIC_ICPR0)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
CLRPEND[31:16]															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CLRPEND[15:0]															
Bit	Marking		Place name				Function							Read and write	
b31~b0	CLRPEND		Interrupt #0 to Interrupt #31 Pending Clear				Write "1" to clear, write "0" is invalid [0]: IRQ0 [1]: IRQ1 [2]: IRQ2 [31]: IRQ31							RW	

5.9.9 Interrupt Pending Clear Register 1 (NVIC_ICPR1)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
CLRPEND[31:16]															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CLRPEND[15:0]															
Bit	Marking		Place name				Function							Read and write	
b31~b0	CLRPEND		Interrupt #32 to Interrupt #63 Pending Clear				Write "1" to clear, write "0" is invalid [0]: IRQ32 [1]: IRQ33 [2]: IRQ34 [31]: IRQ63							RW	

5.9.10 Interrupt Priority Register (NVIC_IPRx) (x=0~15)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
IPRx[31:16]															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
IPRx[15:0]															
Bit	Marking	Place name	Function												Read and write
b31~b0	IPRx	Interrupt priority	Priority of Interrupt #0 to Interrupt #63 [31:29]: priority of interrupt #x*4+3 [23:21]: priority of interrupt #x*4+2 [15:13]: priority of interrupt #x*4+1 [7:5]: priority of interrupt #x*4+0 Among them, 000 has the highest priority and 111 has the lowest priority												RW

5.9.11 Interrupt Mask Register (PRIMASK)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res															PRIMASK
Bit	Marking	Place name	Function												Read and write
b31~b1	Res	Reserved bit	Reserved, must remain at reset value												
b0	PRIMASK	Interrupt mask	After setting, all interrupts except NMI and hardware error exceptions will be masked; After clearing, all exceptions and interrupts will not be masked; The special register needs to be accessed through MSR and MRS special register operation instructions, and can also be accessed by changing the processor state instruction CPS. When dealing with time-sensitive applications, it is necessary to manipulate the PRIMASK register.												RW

6 General-Purpose Input-Output Port Controller (GPIO)

6.1 Introduction

GPIO is the most basic input and output channel of MCU. This product contains four sets of input and output ports, among which PA, PB, PC are 16 bits, and PD is 4 bits. For the actual number of available GPIO ports, please refer to the corresponding specification sheet for more information.

Each pin has associated control and configuration registers to meet specific application needs. By configuring the corresponding registers, the pins can be configured as the following three types of operating modes: analog peripheral input and output, digital peripheral input and output, and GPIO input and output.

6.2 Main characteristics

- Function multiplexing: each pin can be configured as multiple digital or analog functions
- Input mode: analog input, high-impedance input, pull-up input, pull-down input
- Output mode: analog output, push-pull output, open-drain output (pull-up can be enabled)
- Signal input: The input signal arrives at the data input register (GPIO_IN) and on-chip peripherals
- Signal output: The source of the output signal is the data output register (GPIO_OUT) or on-chip peripherals
- Atomic operations: bit operation registers (GPIO_BSRR, GPIO_BRR) realize atomic IO operations
- Interrupt type: rising edge interrupt, falling edge interrupt, high level interrupt, low level interrupt
- Auxiliary functions: output various internal clocks, redirect input signals of on-chip peripherals

6.3 Functional description

6.3.1 Port Circuit Block Diagram

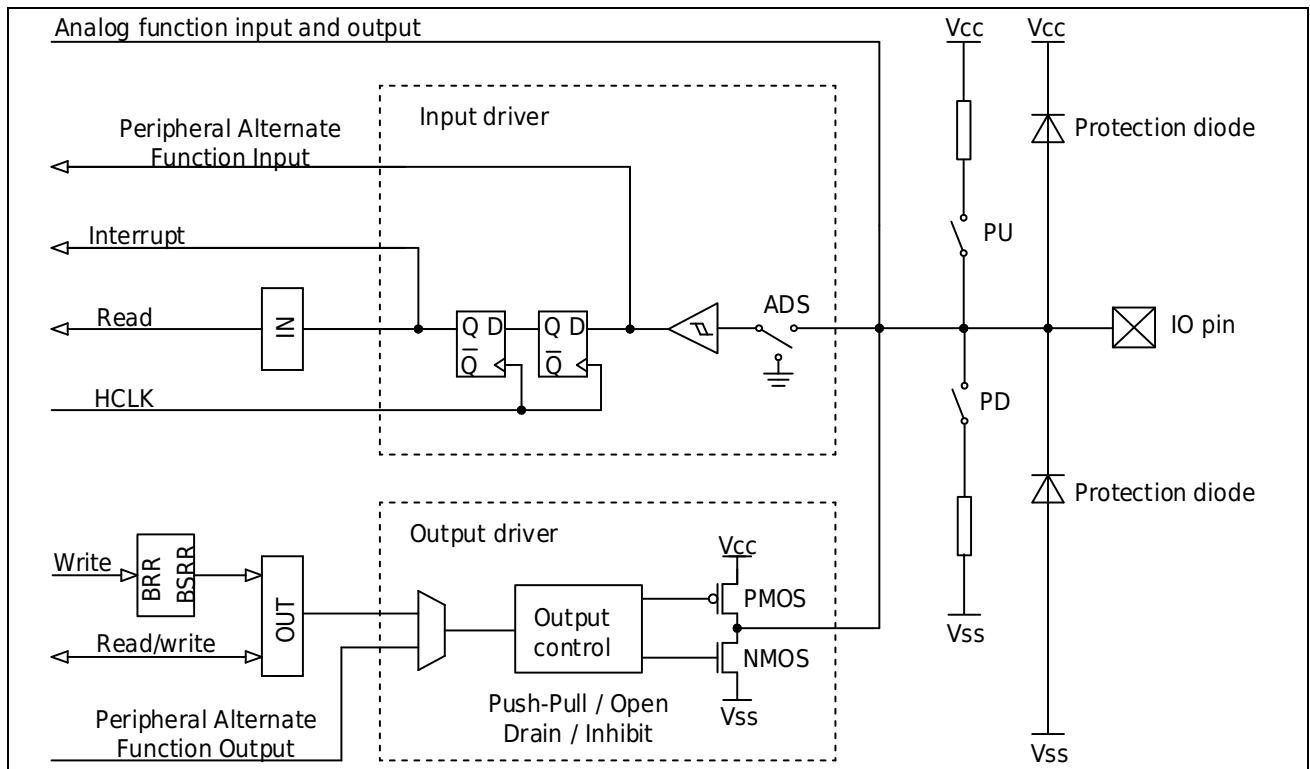


Figure 6-1 General port circuit diagram

6.3.2 Port Mode Configuration

Through the control register, the port can be configured in various working modes, and its truth table is as follows:

Table 6-1 Port function configuration table

Pin mode	IN	ADS	AFRL/AFRH	DIR	OD	OUT	PU	PD
Simulation	0	1	-	-	-	-	-	-
High impedance input	X	0	0	1	-	-	0	0
Pull-up input	1	0	0	1	-	-	1	-
Pull down input	0	0	0	1	-	-	0	1
Open-Drain Output	X	0	0	0	1	1	0	0
Output 0	0	0	0	0	0	0	-	-
Output 1	0	0	0	0	0	1	-	-
Peripheral output	X	0	>0	0	0	-	-	-
Peripheral pull-up input	X	0	>0	1	0	-	1	-
Peripheral pull-down input	X	0	>0	1	0	-	0	1

Note: - stands for any value; X stands for indeterminate

6.3.3 Port Reset Status

During reset and before the GPIO register is not configured after reset, most I/O ports are configured as analog input without pull-up and pull-down status; the status of other pins is as follows:

- SWCLK(PA14): Digital input + pull-up
- SWDIO(PA13): Digital input + pull-up
- PB11: Digital input + pull down

6.3.4 Port emulation function

When you need to use the corresponding functions of the analog peripherals (ADC, LVD.....), you need to configure the port as an analog function. The implementation method is to set the corresponding bit of the GPIO_ADS register to 1, so that the on-chip analog peripherals are connected to this port.

Under this configuration, the port has the following characteristics:

- Analog peripherals are connected to ports
- The digital function of the port is closed: the output level cannot be output; the read pin level is always 0; there is no built-in pull-up and pull-down

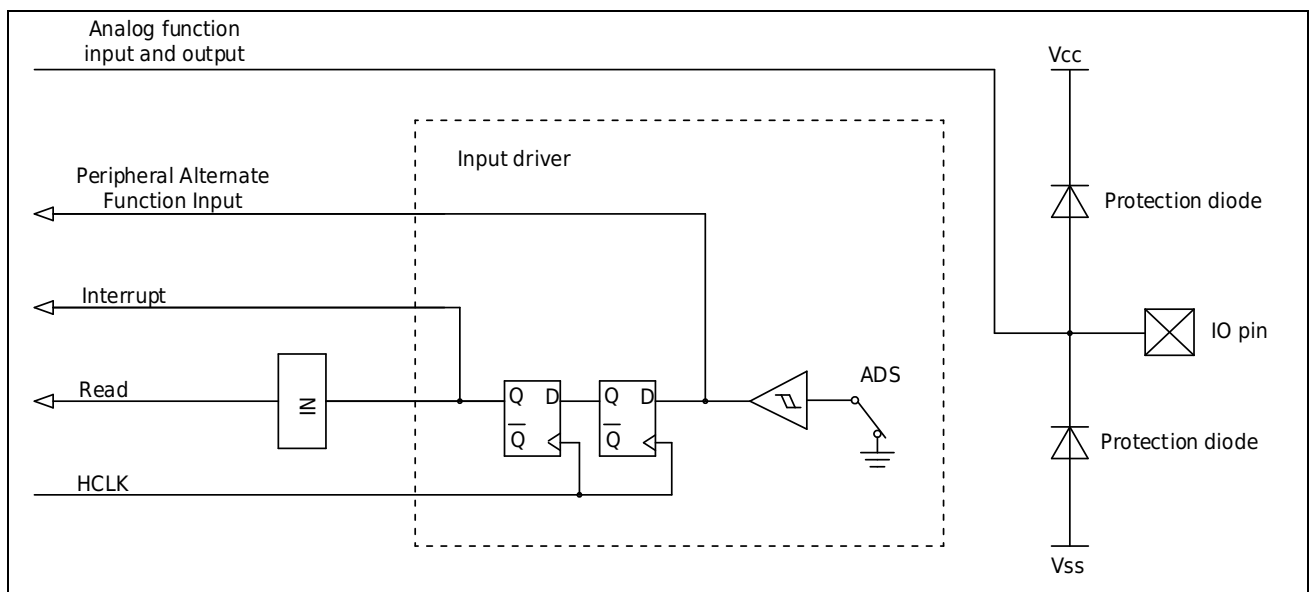


Figure 6-2 HiZ - Analog Configuration Schematic

6.3.5 Port general-purpose input and output functions

When you need to use the general-purpose input and output functions of the port, you need to configure the port as a digital function and set the value of the digital multiplexing function register to 0. The implementation method is to set the corresponding bit of the GPIO_ADS register to 0, and set the corresponding bit of the GPIO_AFRL and GPIO_AFRH registers to 0. Under this configuration, the port has the following characteristics:

- Input mode: pull-up or pull-down can be enabled
- Push-pull output: supports bit manipulation
- Open-drain output: pull-up can be enabled; bit manipulation is supported

Each GPIO has 6 control registers, whose functions are described as follows:

- GPIO_ADS Register configurable ports as digital ports
- GPIO_DIR Register configurable port as input or output
- GPIO_PUPD.PU Register to enable pull-ups on ports
- GPIO_PUPD.PD Register to enable the pull-down of the port
- GPIO_OD Register configurable port output type as push-pull or open-drain
- GPIO_DR Register output port drive capability configuration

Each GPIO has 2 data registers, and their functions are described as follows:

- GPIO_IN Registers store data input via I/O and are read only
- GPIO_OUT Registers store the data to be output and are read/write accessible

Note:

[read - modify - write] to the GPIO_OUT register is not an atomic operation, and it is not recommended to use this method to modify the port output level.

Each GPIO has 2 bit manipulation registers, and their functions are described as follows:

- GPIO_BSRR The BS bit of the register can set the corresponding bit of GPIO_OUT to 1
 The BR bit of the register can clear the corresponding bit of GPIO_OUT to 0
- GPIO_BRR The register bit can clear the corresponding bit of GPIO_OUT to 0

Note:

The process of modifying the value of GPIO_OUT through the bit operation register is an atomic operation. It is recommended to modify the port output level by bit operation.

6.3.5.1 Input mode

When GPIO is configured as input mode, its simplified block diagram is as follows:

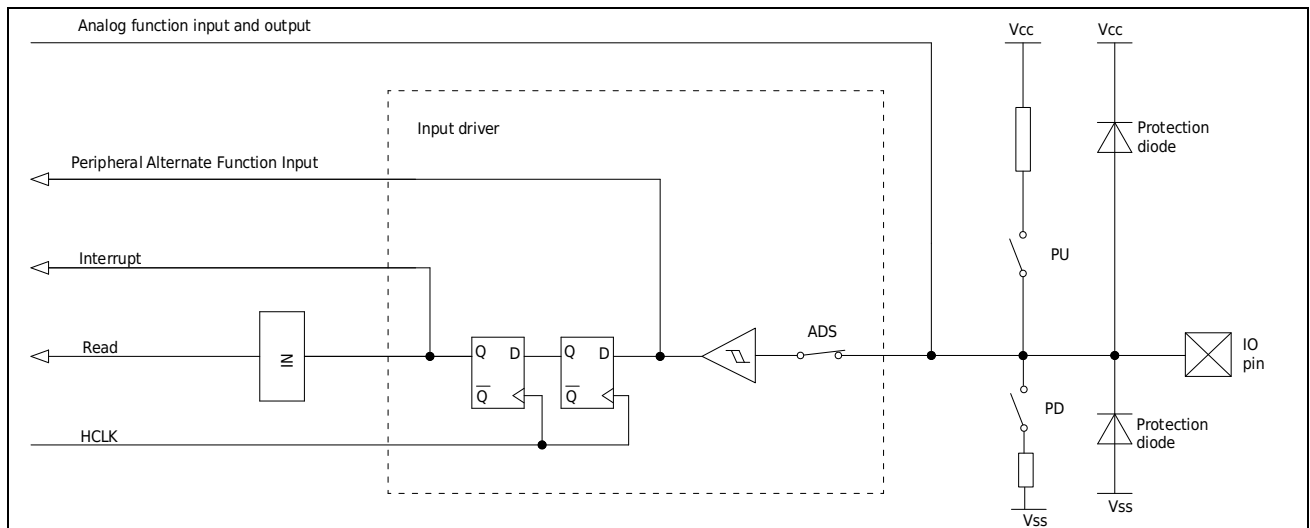


Figure 6-3 Schematic diagram of input floating/pull-up/pull-down configuration

Each port can get the port pin level by reading the GPIO_IN register. As shown in

Figure 6-3, each bit of the GPIO_IN register and the latch in front of it form a synchronizer to avoid signal instability caused by pin level changes in a short period of time when the system clock state changes, but at the same time Latency is introduced. The synchronization diagram for reading port pin data is as follows:

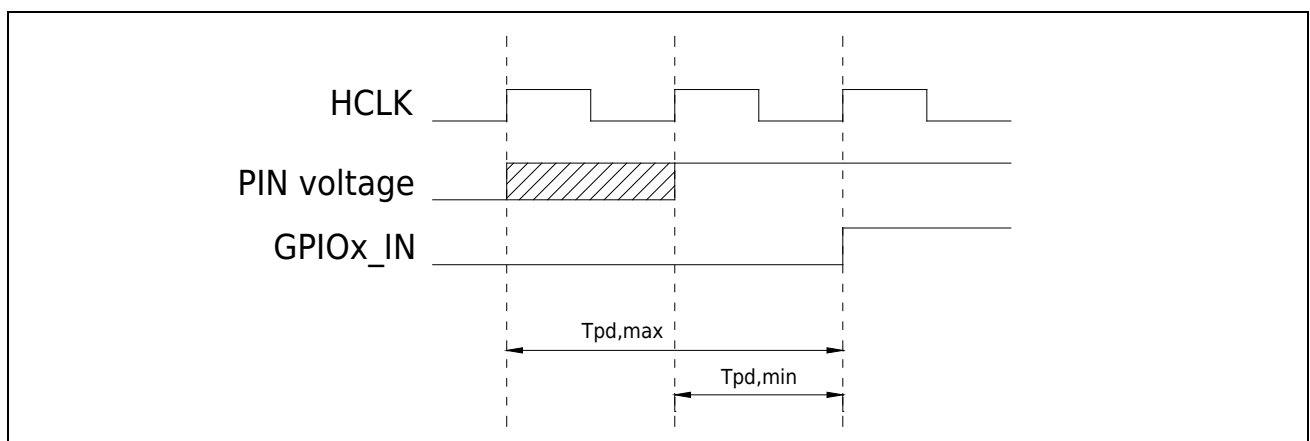


Figure 6-4 Read port pin data synchronization diagram

During the clock period 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, after the next rising edge of the system clock, the stable pin level signal can be read. GPIO_IN register on the rising edge of the system clock. The signal transition delay T_{pd} is 1-2 system clocks.

6.3.5.2 Output mode

When GPIO is configured as output mode, its simplified block diagram is as follows:

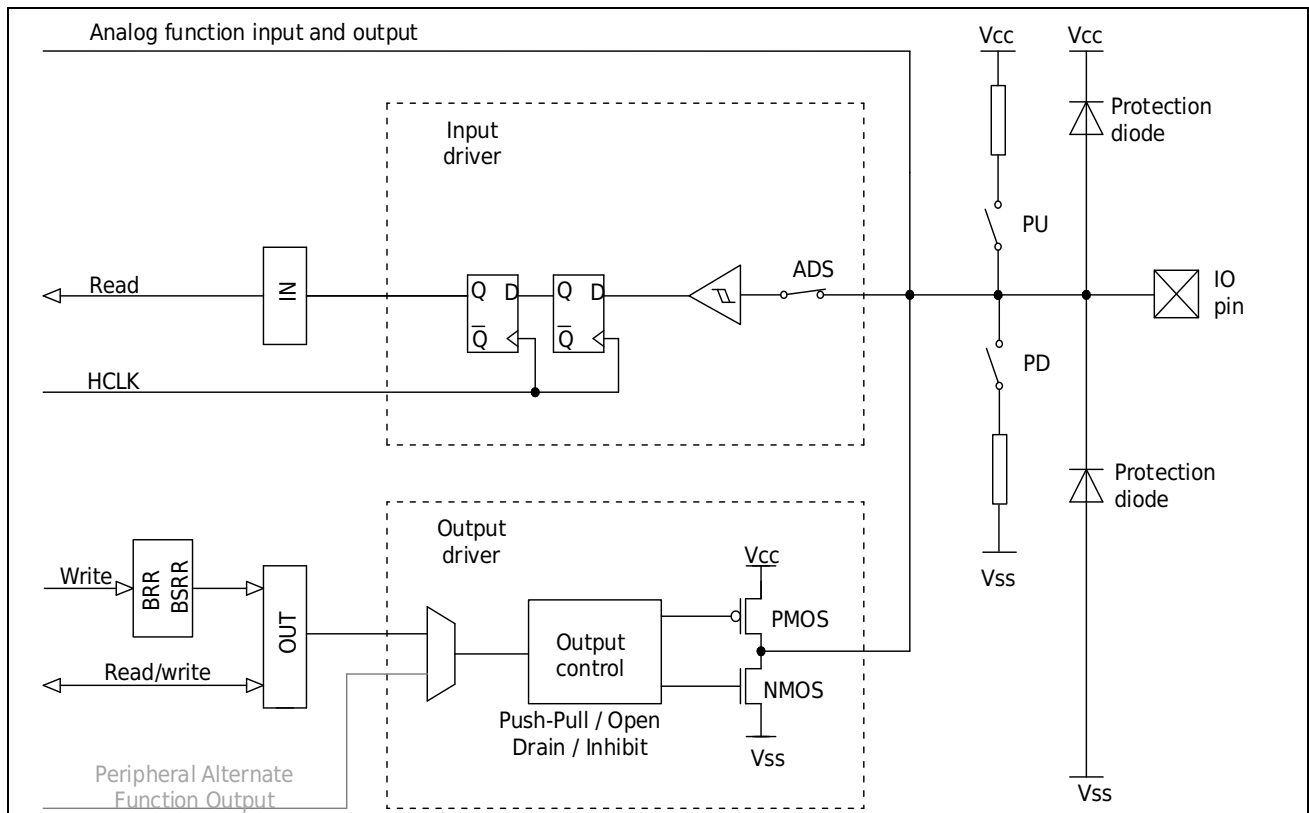


Figure 6-5 Schematic diagram of output configuration

Each port can control the output level of the IO port by writing data to GPIO_BRR, GPIO_BSRR or GPIO_OUT.

GPIO_BRR and GPIO_BSRR are bit operation registers. The number of IOs in the same group of ports can be modified from 1 to 16 each time and the unmodified IOs in the same group of ports remain unchanged. Use the above two registers to modify IOs. The process is an atomic operation. The value written to the register GPIO_OUT is the output value of the output PIN corresponding to each bit.

6.3.6 Port multiplexing function

When you need to use the corresponding functions of digital peripherals (GPIO, USART, SPI...), you need to configure the port as a digital function. The operation method is to set the corresponding bit of GPIO_ADS to 0 and configure GPIO_AFRL or GPIO_AFRH to connect a specific on-chip peripheral to the port.

Under this configuration, the port has the following characteristics:

- Input mode: pull-up or pull-down can be enabled; external signals are input to on-chip peripherals; GPIO_IN can read external signals and generate port interrupts
- Output mode: open-drain (pull-up enabled) or push-pull can be selected; on-chip peripherals output signals to ports; port interrupts cannot be generated

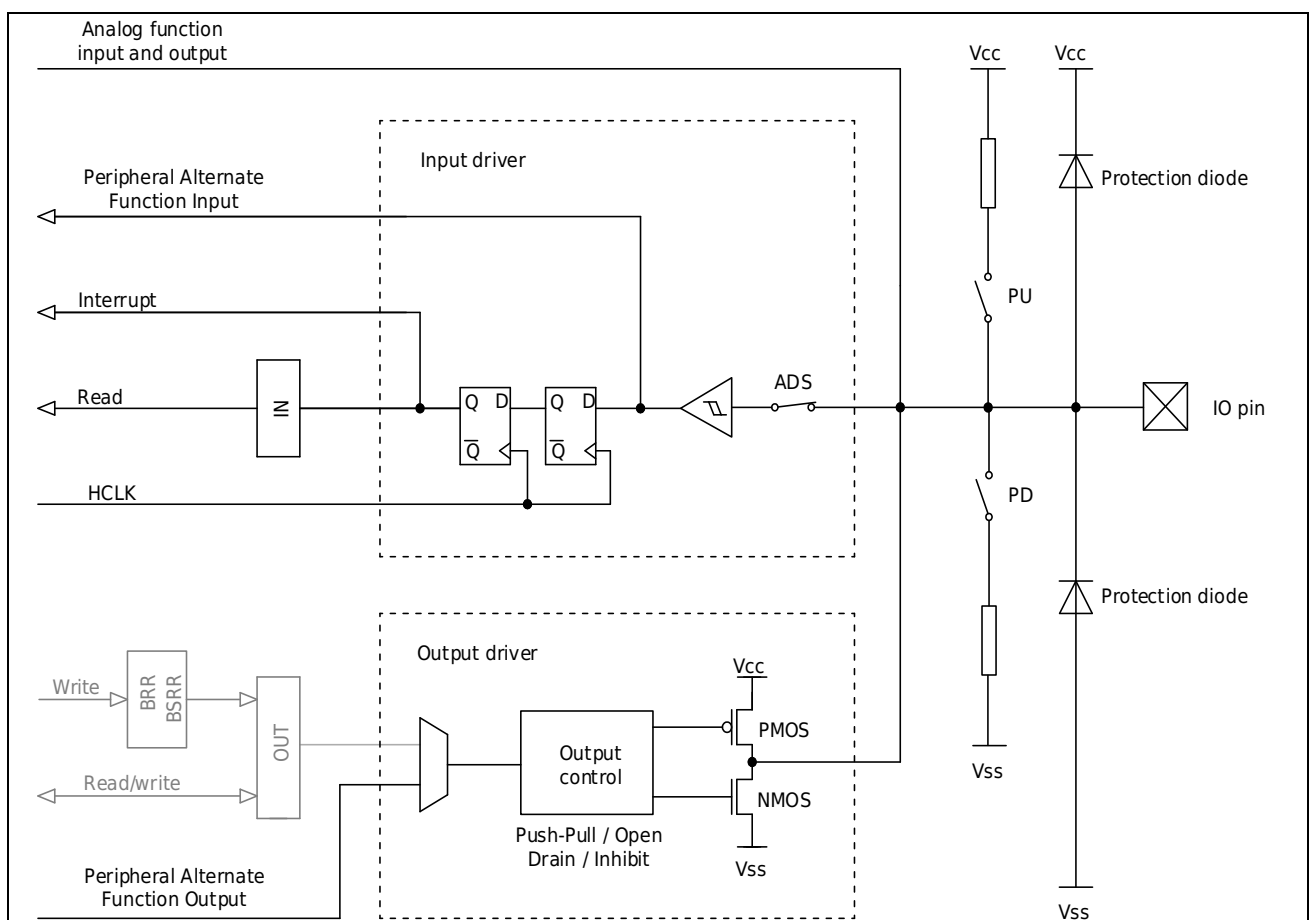


Figure 6-6 Schematic diagram of multiplexing function configuration

The digital multiplexing function of each port of this chip is shown in the table below.

Table 6-2 PINMUX multiplexing function

Pin name	Digital function					
	AF0	AF1	AF2	AF3	AF4	AF5
PD03	GPIO	ATIM0_BK	CTIM0_CH0	CTIM1_TOGP	ATIM3_BK	USART2_TXD
PC13	GPIO	SPI0_SCK	CTIM2_CH0	I2C1_SCL	ATIM3_CH0A	USART3_NCTS
PC14	GPIO	SPI0_MISO	CTIM2_CH1	I2C1_SDA	ATIM3_CH1A	USART3_NRTS
PC15	GPIO	SPI0_MOSI	CTIM2_ETR		ATIM3_CH2A	USART3_SCK
PD00	GPIO	SPI0_CS	CTIM0_CH2	I2C0_SCL	ATIM3_CH0B	I2C1_SCL
PD01	GPIO	LVD_OUT	CTIM0_CH3	I2C0_SDA	ATIM3_CH2B	I2C1_SDA
PC00	GPIO	ATIM0_BK	CTIM1_TOGP	CTIM1_ETRTOG	SPI0_MISO	
PC01	GPIO	ATIM0_CH0A	CTIM1_TOGN		SPI0_MOSI	
PC02	GPIO	ATIM0_CH0B		SPI0_CS	SPI1_MISO	USART0_NCTS
PC03	GPIO	ATIM0_GATE		SPI0_SCK	SPI1_MOSI	USART0_NRTS
PA00	GPIO	ATIM0_ETR	CTIM0_CH0	CTIM0_ETR	VC0_OUT	USART1_NCTS
PA01	GPIO	MCO	CTIM0_CH1	SPI1_CS	USART2_NCTS	USART1_NRTS
PA02	GPIO	CTIM1_TOGN	CTIM0_CH2	VC1_OUT	USART3_NRTS	USART1_TXD
PA03	GPIO	LVD_OUT	CTIM0_CH3	SPI1_SCK	USART3_TXD	USART1_RXD
PA04	GPIO	ATIM3_CH0A	PCLK0_OUT	SPI0_CS	USART3_RXD	USART1_SCK
PA05	GPIO	ATIM3_CH1A	CTIM0_CH0	CTIM0_ETR	USART3_NCTS	SPI0_SCK
PA06	GPIO	ATIM3_CH2A	USART2_RXD	CTIM1_CH0	ATIM3_BK	SPI0_MISO
PA07	GPIO	ATIM3_CH0B	USART3_TXD	CTIM1_CH1	USART2_NRTS	SPI0_MOSI
PC04	GPIO	ATIM3_CH1B		I2C0_SCL		USART0_SCK
PC05	GPIO	ATIM3_CH2B	CTIM0_TOGP	I2C0_SDA	ATIM0_ETR	
PB00	GPIO	ATIM3_CH1B	CTIM2_ETR	CTIM1_CH2	ATIM0_CH0A	USART3_SCK
PB01	GPIO	ATIM3_CH2B	CTIM2_TOGP	CTIM1_CH3	ATIM0_CH0B	PCLK1_OUT
PB02	GPIO	CTIM0_CH3	CTIM2_TOGN	ATIM3_GATE	ATIM0_GATE	USART2_RXD
PB10	GPIO	CTIM0_CH2			I2C1_SCL	USART2_TXD
PB12	GPIO	I2C0_SCL	ATIM3_BK	CTIM1_ETRTOG	SPI1_CS	USART2_SCK
PB13	GPIO	I2C0_SDA	ATIM3_CH0B		SPI1_SCK	USART2_NCTS
PB14	GPIO		ATIM3_CH1B	CTIM1_TOGP	SPI1_MISO	USART2_NRTS
PB15	GPIO	MCO	ATIM3_CH2B	CTIM1_TOGN	SPI1_MOSI	
PC06	GPIO	ATIM3_ETR	CTIM2_CH0	CTIM1_CH0		USART2_NCTS
PC07	GPIO		CTIM2_CH1	CTIM1_CH1	SPI1_SCK	USART2_NRTS
PC08	GPIO	I2C1_SCL	CTIM2_CH2	CTIM1_CH2		USART2_SCK
PC09	GPIO	I2C1_SDA	CTIM2_CH3	CTIM1_CH3		
PA08	GPIO	CTIM0_TOGP	ATIM3_CH0A	CTIM1_ETRTOG	USART3_RXD	USART0_SCK
PA09	GPIO	CTIM0_TOGN	ATIM3_CH1A	ATIM0_BK	CTIM1_CH2	USART0_TXD
PA10	GPIO	I2C1_SCL	ATIM3_CH2A	ATIM0_CH0A	CTIM1_CH3	USART0_RXD

Pin name	Digital function					
	AF0	AF1	AF2	AF3	AF4	AF5
PA11	GPIO	VC0_OUT	ATIM3_GATE	ATIM0_CH0B	USART1_NCTS	USART0_NCTS
PA12	GPIO	VC1_OUT	ATIM3_ETR	ATIM0_GATE	USART1_NRTS	USART0_NRTS
PA13	GPIO	ATIM3_BK	CTIM2_CH2	ATIM0_ETR	USART1_RXD	I2C0_SCL
PA14	GPIO	CTIM0_CH1	CTIM2_CH3	USART2_SCK	USART1_TXD	I2C0_SDA
PA15	GPIO	SPI0_CS	CTIM1_CH0	CTIM1_ETR	USART1_SCK	USART0_TXD
PC10	GPIO		USART3_TXD		ATIM3_CH0A	ATIM0_CH0A
PC11	GPIO	CTIM0_TOGN	USART3_RXD		ATIM3_CH1A	ATIM0_CH0B
PC12	GPIO		USART3_NCTS	CTIM2_ETR	ATIM3_CH2A	ATIM0_GATE
PD02	GPIO		USART3_NRTS	CTIM1_ETR	ATIM3_CH0B	ATIM0_ETR
PB03	GPIO	CTIM0_CH1	USART0_RXD	I2C1_SDA	ATIM3_CH1B	SPI0_SCK
PB04	GPIO	CTIM1_CH0	CTIM2_TOGP	PCLK0_OUT	ATIM3_CH2B	SPI0_MISO
PB05	GPIO	CTIM1_CH1	CTIM2_TOGN	USART3_SCK	USART2_TXD	SPI0_MOSI
PB06	GPIO	USART0_TXD	CTIM2_CH0	SPI1_MISO	I2C0_SCL	USART2_RXD
PB07	GPIO	USART0_RXD	CTIM2_CH1	SPI1_MOSI	I2C0_SDA	PCLK1_OUT
PB11	GPIO					
PB08	GPIO	CTIM1_ETRTOG	CTIM2_CH2		I2C0_SCL	ATIM0_BK
PB09	GPIO	SPI1_CS	CTIM2_CH3	I2C0_SDA	I2C1_SDA	

Table 6-3 IOAUX1/2 multiplexing function

ATIM0CAP0[2:0]	ATIM0_CHA	USART0_RXD	VC0_OUT	LVD_OUT	VC1_OUT	PA00	PA08	PB00
CTIM0CAP3[6:4]	CTIM0_CH3	USART1_RXD	VC0_OUT	LVD_OUT	VC1_OUT	PA01	PA09	PB01
CTIM1CAP3[10:8]	CTIM1_CH3	USART2_RXD	VC0_OUT	LVD_OUT	VC1_OUT	PA02	PA10	PB02
CTIM2CAP3[14:12]	CTIM2_CH3	USART3_RXD	VC0_OUT	LVD_OUT	VC1_OUT	PA03	PA11	PB03
ATIM0ETR[2:0]	ATIM0_ETR	USART0_RXD	VC0_OUT	LVD_OUT	VC1_OUT	PA04	PA12	PB04
CTIM0ETR[6:4]	CTIM0_ETR	USART1_RXD	VC0_OUT	LVD_OUT	VC1_OUT	PA05	PA13	PB05
CTIM1ETR[10:8]	CTIM1_ETR	USART2_RXD	VC0_OUT	LVD_OUT	VC1_OUT	PA06	PA14	PB06
CTIM2ETR[14:12]	CTIM2_ETR	USART3_RXD	VC0_OUT	LVD_OUT	VC1_OUT	PA07	PA15	PB07

6.3.7 Port clock output

6.3.7.1 Output PCLK0

Configure the GPIO register to output the 2 -frequency output of PCLK0. The following example outputs PCLK0 from PA04:

- Step1. Set the SYSCTRL_IOAUX0.PCLK0DIV of the system control module to 2, and select the frequency division by 2;
- Step2. Set GPIOA_ADS.PIN[4] to 0, and configure PA04 as a digital port;
- Step3. Set GPIOA_DIR.PIN[4] to 0, configure PA04 as output mode;
- Step4. Set GPIOA_AFRL.AFSEL4 to 2, and configure the multiplexing function of PA04 as PCLK0_OUT.

6.3.7.2 Output MCO

Configure the GPIO register to output RCH/RCL/XTH/PLL/RC10/RC150K, the example below outputs RCH/8 from PA01:

- Step1. Set SYSCTRL_IOAUX0.MCOSEL of the system control module to 0, select RCH;
- Step2. Set the SYSCTRL_IOAUX0.MCODIV of the system control module to 2, select 8 frequency division;
- Step3. Set GPIOA_ADS.PIN[1] to 0, and configure PA01 as a digital port;
- Step4. Set GPIOA_DIR.PIN[1] to 0, configure PA01 as output mode;
- Step5. Set GPIOA_AFRL.AFSEL1 to 1, and configure the multiplexing function of PA01 as MCO.

6.3.8 Port external interrupt

When the port is configured as a digital function (GPIO_ADS is 0) and works in input mode (GPIO_DIR is 1), each port has an interrupt function. Each port supports 4 types of interrupts: rising edge interrupt, falling edge interrupt, high level interrupt, low level interrupt; the corresponding interrupt enable registers are: GPIO_RIE, GPIO_FIE, GPIO_HIE, GPIO_LIE. When an interrupt is triggered, it can be determined which port triggered the interrupt by querying the interrupt status register (GPIO_IFR), and the corresponding interrupt status flag bit can be cleared through the interrupt clear register (GPIO_ICR).

The simplified block diagram related to GPIO external interrupt is as follows:

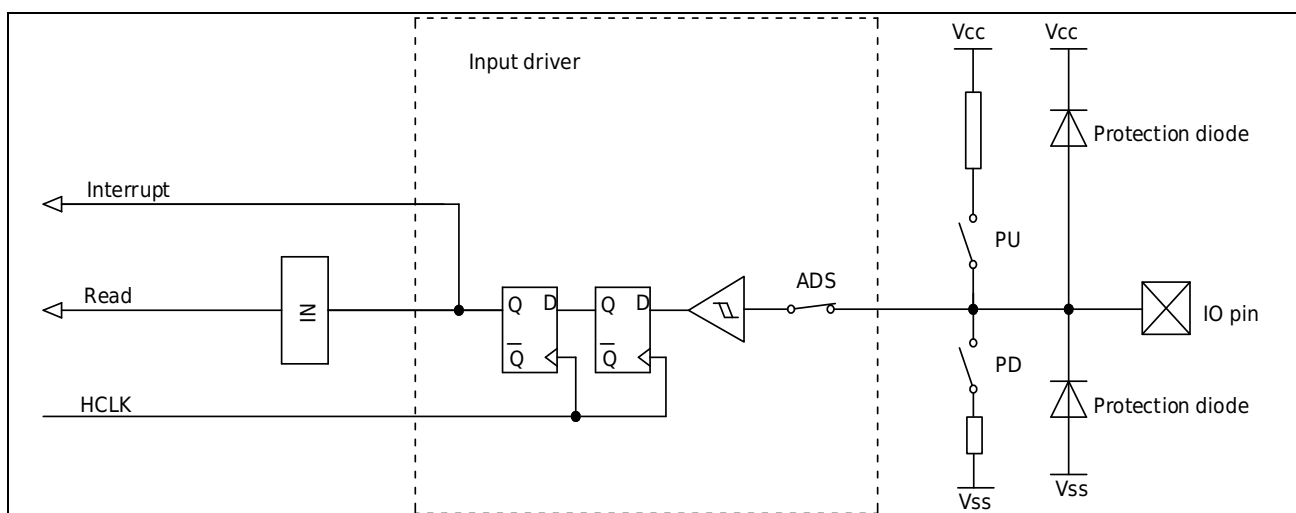


Figure 6-7 Schematic Diagram of Port External Interrupt

6.4 Register description

6.4.1 Register list

Table 6-4 GPIO base address list

Name	base address	Description
GPIOA	0x4002 4000	GPIOA base address
GPIOB	0x4002 4400	GPIOB base address
GPIOC	0x4002 4800	GPIOC base address
GPIOD	0x4002 4C00	GPIOD base address

Note:

GPIOA, GPIOB, and GPIOC have 16 effective bits, and GPIOD has 4 effective bits (PD00-PD03). For non-existing bits, the write operation is invalid, and the read value is always 0.

Table 6-5 GPIO Register List

Register name	Offset address	Reset value
Port Digital-to-Analog Configuration Register (GPIO_ADS)	0x00	(PA)0x0000 9FFF (PB)0x0000 F7FF (PC)0x0000 FFFF (PD)0x0000 000F
Port Input Output Direction Configuration Register (GPIO_DIR)	0x04	(PA)0x0000 FFFF (PB)0x0000 FFFF (PC)0x0000 FFFF (PD)0x0000 000F
Port Output Type Register (GPIO_OD)	0x08	0x0000 0000
Port Output Drive Capability Register (GPIO_DR)	0x10	0x0000 0000
Port pull-up and pull-down register (GPIO_PUPD)	0x18	(PA)0x0000 6000 (PB)0x0800 0000 (PC)0x0000 0000 (PD)0x0000 0000
Port Data Input Register (GPIO_IN)	0x1C	0x0000 xxxx
Port Data Output Register (GPIO_OUT)	0x20	0x0000 xxxx
Port Reset Register (GPIO_BRR)	0x24	0x0000 0000
Port Set Reset Register (GPIO_BSRR)	0x28	0x0000 0000
Port Alternate Function Low Register (GPIO_AFRL)	0x30	0x0000 0000
Port Alternate Function High Register (GPIO_AFRH)	0x34	0x0000 0000
Port High Interrupt Enable Configuration Register (GPIO_HIE)	0x38	0x0000 0000
Port Low Level Interrupt Enable Configuration Register (GPIO_LIE)	0x3C	0x0000 0000
Port Rising Edge Interrupt Enable Configuration Register (GPIO_RIE)	0x40	0x0000 0000
Port Falling Edge Interrupt Enable Configuration Register (GPIO_FIE)	0x44	0x0000 0000
Port Interrupt Status Register (GPIO_IFR)	0x48	0x0000 0000
Port Interrupt Clear Register (GPIO_ICR)	0x4C	(PA)0x0000 FFFF (PB)0x0000 FFFF (PC)0x0000 FFFF (PD)0x0000 000F

6.4.2 Port Digital-to-Analog Configuration Register (GPIO_ADS)

Reset value: (PA) 0x0000 9FFF

(PB) 0x0000 F7FF

(PC) 0x0000 FFFF

(PD) 0x0000 000F

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PIN															

Bit	Marking	Place name	Function	Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value	
b15~b0	PIN	Port digital function or analog function control	0: configure the port as a digital function 1: Configure the port as an analog function	RW

6.4.3 Port input and output direction configuration register (GPIO_DIR)

Reset value: (PA)0x0000 FFFF

(PB)0x0000 FFFF

(PC)0x0000 FFFF

(PD)0x0000 000F

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PIN															

Bit	Marking	Place name	Function	Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value	
b15~b0	PIN	Port input and output direction control	0: Configure the port as an output 1: Configure the port as an input	RW

6.4.4 Port Output Type Register (GPIO_OD)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PIN															

Bit	Marking	Place name	Function	Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value	
b15~b0	PIN	Port output mode control bit	0: Push-pull output (reset value) 1: Open drain output	RW

6.4.5 Port Output Drive Capability Register (GPIO_DR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PIN															

Bit	Marking	Place name	Function	Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value	
b15~b0	PIN	Port output drive capability control bit	0: High drive capability (reset value) 1: Low drive capability	RW

6.4.6 Port pull-up and pull-down register (GPIO_PUPD)

Reset value: (PA)0x0400 6000

(PB)0x0000 0000

(PC)0x0000 0000

(PD)0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
PD															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PU															
Bit	Marking	Place name		Function											Read and write
b31~b16	PD	Port	pull-down resistor enable control	0: Disable pull-down resistor 1: Enable pull-down resistor											RW
b15~b0	PU	Port	pull-up resistor enable control	0: Disable pull-up resistor 1: Enable pull-up resistor Note: PDx and PUX are 1 at the same time, then the port status is to enable the pull-up resistor.											RW

6.4.7 Port Data Input Register (GPIO_IN)

Reset value: 0x0000 XXXX

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PIN															
Bit	Marking	Place name		Function											Read and write
b31~b16	Res	Reserved bit		Reserved, must remain at reset value											
b15~b0	PIN	Port input data		-											RO

6.4.8 Port Data Output Register (GPIO_OUT)

Reset value: 0x0000 XXXX

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PIN															
Bit	Marking	Place name	Function												Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value												
b15~b0	PIN	Port output data	-												RW

6.4.9 Port Reset Register (GPIO_BRR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PIN															
Bit	Marking	Place name	Function												Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value												
b15~b0	PIN	Port Bit Clear Control	0: Does not affect the corresponding bit of the GPIO_OUT register 1: Set the corresponding bit of the GPIO_OUT register to 0												WO

6.4.10 Port Set Reset Register (GPIO_BSRR)

Reset value: 0x00000000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
BR															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
BS															
Bit	Marking	Place name	Function												Read and write
b31~b16	BR	Port Bit Clear Control	0: Does not affect the corresponding bit of the GPIO_OUT register 1: Set the corresponding bit of the GPIO_OUT register to 0												WO
b15~b0	BS	Port Bit Bit Control	0: Does not affect the corresponding bit of the GPIO_OUT register 1: Set the corresponding bit of the GPIO_OUT register to 1 Note: When BR and BS occur simultaneously, BS takes precedence.												WO

6.4.11 Port Alternate Function Low Register (GPIO_AFRL)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
AFSEL7				AFSEL6				AFSEL5				AFSEL4			
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
AFSEL3				AFSEL2				AFSEL1				AFSEL0			

Bit	Marking	Place name	Function	Read and write
b31~b0	AFSELy y = 0-7	Port multiplexing function control	0000: GPIO 0001: AF1 0010: AF2 0011: AF3 0100: AF4 0101: AF5 0110-1111: reserved bits Note 1: For the specific functions of AF1 - AF5, please refer to the chapter [Port multiplexing function]	RW

6.4.12 Port Alternate Function High Register (GPIO_AFRH)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
AFSEL15				AFSEL14				AFSEL13				AFSEL12			
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
AFSEL11				AFSEL10				AFSEL9				AFSEL8			

Bit	Marking	Place name	Function	Read and write
b31~b0	AFSELy y = 8-15	Port multiplexing function control	0000: GPIO 0001: AF1 0010: AF2 0011: AF3 0100: AF4 0101: AF5 0110-1111: reserved bits Note 1: For the specific functions of AF1 - AF5, please refer to the chapter [Port multiplexing function]	RW

6.4.13 Port High Interrupt Enable Configuration Register (GPIO_HIE)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PIN															
Bit	Marking	Place name	Function												Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value												
b15~b0	PIN	Port high level interrupt enable control	0: Disable the high level interrupt of the corresponding port 1: Enable the high-level interrupt of the corresponding port												RW

6.4.14 Port Low Level Interrupt Enable Configuration Register (GPIO_LIE)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PIN															
Bit	Marking	Place name	Function												Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value												
b15~b0	PIN	Port low level interrupt enable control	0: Disable the low-level interrupt of the corresponding port 1: Enable the low-level interrupt of the corresponding port												RW

6.4.15 Port Rising Edge Interrupt Enable Configuration Register (GPIO_RIE)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PIN															
Bit	Marking	Place name	Function												Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value												
b15~b0	PIN	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												RW

6.4.16 Port Falling Edge Interrupt Enable Configuration Register (GPIO_FIE)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PIN															
Bit	Marking	Place name	Function												Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value												
b15~b0	PIN	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												RW

6.4.17 Port Interrupt Status Register (GPIO_IFR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PIN															
Bit	Marking	Place name	Function												Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value												
b15~b0	PIN	Port interrupt status flag	0: No qualifying interrupt occurred 1: A qualifying interrupt has occurred												RO

6.4.18 Port Interrupt Clear Register (GPIO_ICR)

Reset value: (PA)0x0000 FFFF

(PB)0x0000 FFFF

(PC)0x0000 FFFF

(PD)0x0000 000F

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PIN															
Bit	Marking	Place name	Function												Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value												
b15~b0	PIN	Port interrupt flag cleared	0: Clear the corresponding interrupt flag bit 1: no function												R1W0

7 I2C -bus (I2C)

7.1 Overview

I2C is a two-wire bidirectional synchronous serial bus, which uses a clock line and a data line to transmit information between two devices connected to the bus, providing a simple and efficient method for data exchange between devices. Each device connected to the bus has a unique address, and any device can be used as both a master and a slave, but only one master is allowed at the same time. I2C standard is a real multi-master bus with a conflict detection mechanism and an arbitration mechanism. It can use the arbitration mechanism to avoid data conflicts and protect data when multiple Masters request control of the bus at the same time.

I2C bus controller can meet various specifications of the I2C bus and support all transmission modes communicating with the I2C bus. The I2C logic can handle byte transfers autonomously. It can keep track of the serial transfer, and there is a status register (I2C_STAT) that can reflect the status of the I2C bus controller and the I2C bus.

7.2 Main characteristics

I2C controller supports the following features:

- Support four working modes of master sending/receiving and slave sending/receiving
- Support standard (100Kbps) / fast (400Kbps) / high speed (1Mbps) three working rates
- The slave supports 7-bit addressing function
- Support noise filtering function
- Supports three slave addresses
- Support broadcast address
- Support interrupt status query function

7.3 Protocol description

The I2C bus transfers information using the SCL (serial clock bus) and SDA (serial data bus) that connect devices. The Master computer outputs a serial clock signal on the SCL line, and the data is transmitted on the SDA line, and each byte is transmitted (the highest bit MSB starts to be transmitted), followed by an acknowledge bit. One SCL clock pulse transfers one data bit.

7.3.1 Data transmission on the I2C bus

Usually the standard I2C transmission protocol consists of four parts: start (S) or repeated start signal (Sr), slave address and read and write bits, transmission data, and stop signal (P).

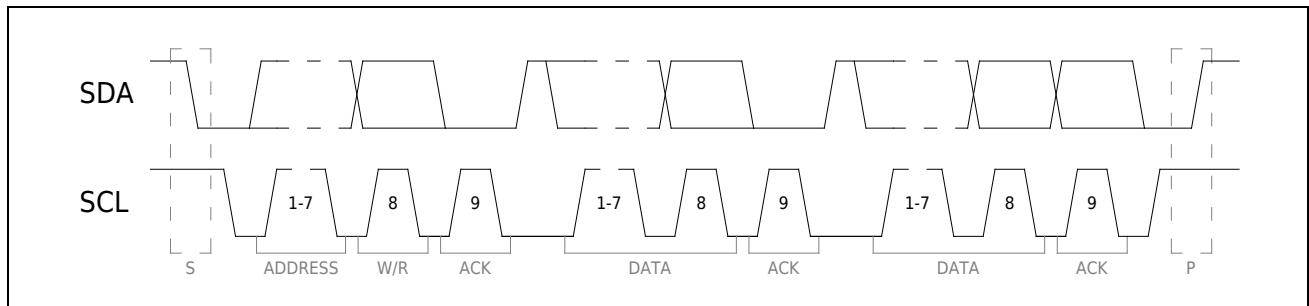


Figure 7-1 I2C transfer protocol

■ Start signal, repeat start signal, stop signal

When the bus is in an idle state (the SCL and SDA lines are high at the same time), a signal from high to low appears on the SDA line, indicating that a start signal is generated on the bus. A repeated start signal occurs when there is no stop signal between two start signals. This method is used by a master to communicate with another slave or the same slave with a different transfer direction (for example: from writing to the device to reading from the device) without releasing the bus.

When the SCL line is high, a low-to-high signal appears on the SDA line, which is defined as a stop signal. The master sends a stop signal to the bus to end the data transfer.

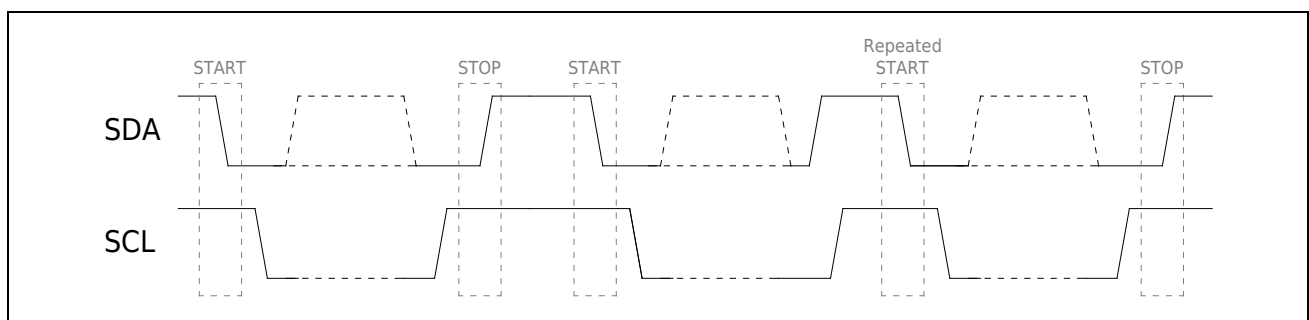


Figure 7-2 START and STOP conditions

- Slave address and read/write bits

When the start signal is generated, the host immediately transmits the first byte of data: 7 - bit slave address + read and write bits, the read and write bits control the data transmission direction of the slave (0: write; 1: read). A slave addressed by the master will acknowledge by pulling SDA low on the ninth SCL clock cycle.

- Transfer data

During data transfer, one SCL clock pulse transfers one data bit, and the SDA line can only change when SCL is low.

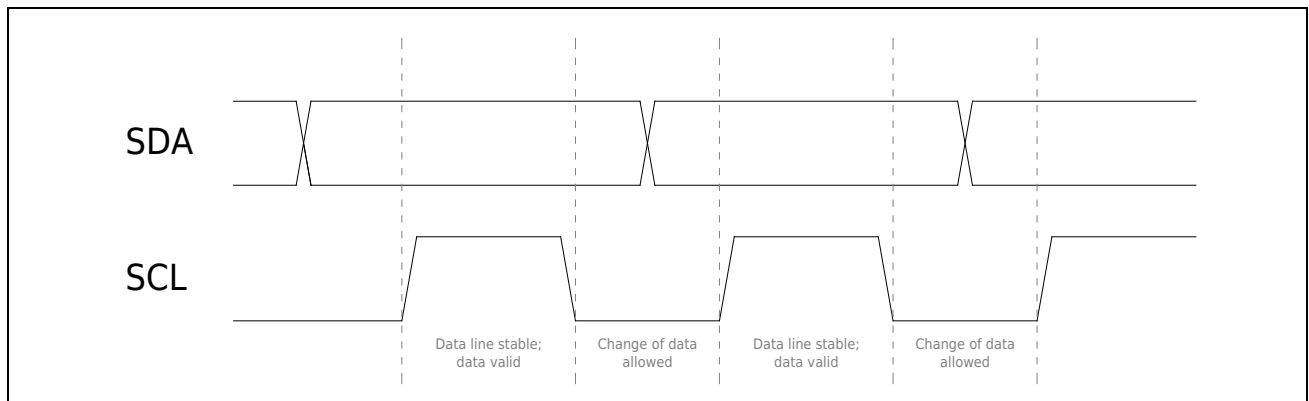


Figure 7-3 Bit transfer on the I2C bus

7.3.2 Acknowledgment on the I2C bus

Each byte transferred is followed by an acknowledge bit. After each byte is sent, the sender will release SDA, and the receiver needs to respond to the sender by pulling down the SDA line. ACK is a low-level signal. When the clock signal is high, SDA remains low to indicate that the receiving end has successfully received the data from the sending end.

(NACK) is generated on the slave, the Master can generate a stop signal to exit data transmission, or generate a repeated start signal to start a new round of data transmission. When the Master is used as a receiving device, a non-response signal (NACK) occurs, and the slave releases the SDA line, causing the Master to generate a stop signal or a repeated start signal.

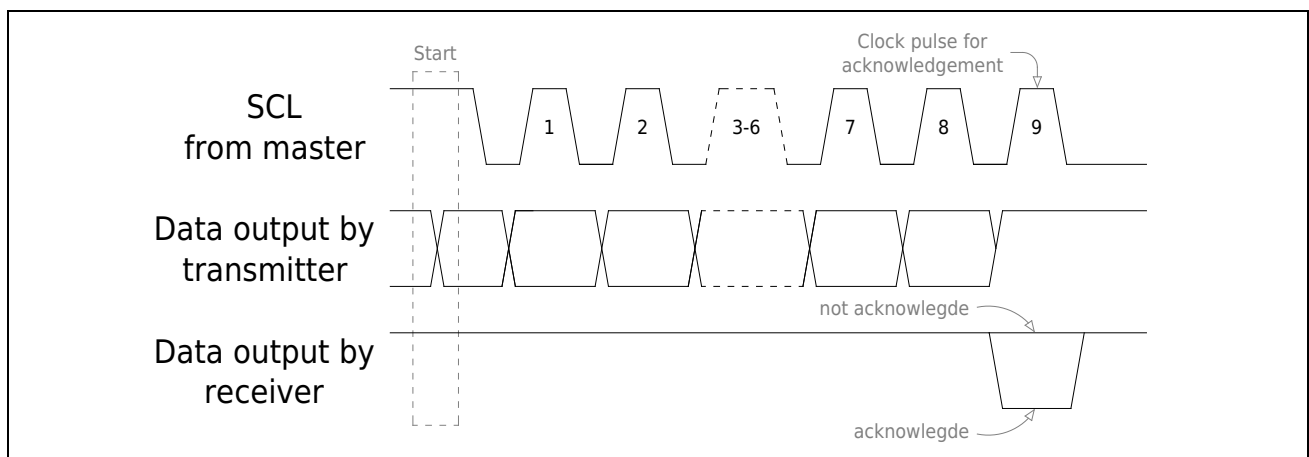


Figure 7-4 Acknowledgment signal on I2C bus

7.3.3 Arbitration on the I2C bus

Arbitration on the I2C bus is divided into two parts: synchronization on the SCL line and arbitration on the SDA line.

■ Synchronization on the SCL line (clock synchronization)

Since the I2C bus has the logic function of wired and, as long as one node on the SCL line sends a low level, the bus will show a low level. The bus can only behave as a high level when all nodes are sending a high level. Therefore, the clock low time is determined by the device with the longest clock level period, and the clock high time is determined by the device with the shortest clock high period. Due to the characteristics of I2C, when multiple Masters send clock signals at the same time, a unified clock signal is represented on the bus. If the slave wants the master to reduce the transmission speed, it can notify the master by actively pulling SCL low to extend its low level time. When the master finds that the SCL level is pulled low when preparing for the next transmission, it will wait until the slave completes. Operates and releases control of the SCL line.

■ Arbitration on SDA Online

The arbitration on the SDA line is also because the I2C bus has a logic function of line and. After the master sends data, it decides whether to exit the competition by comparing the data on the bus. The master that loses the arbitration immediately switches to the unaddressed slave state to ensure that it can be addressed by the master that wins the arbitration. A master that loses arbitration continues to output clock pulses (on SCL) until the current serial byte has been sent. This principle can ensure that the I2C bus will not lose data when multiple Masters attempt to control the bus.

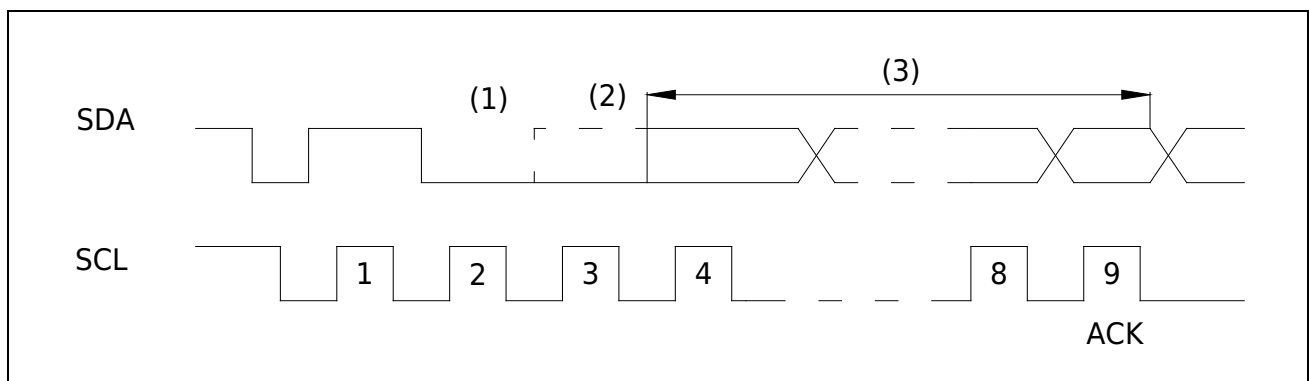


Figure 7-5 Arbitration on the I2C bus

- 1) Device 1 and Device 2 transmit serial data at the same time;
- 2) In the third SCL cycle, device 1 sends a logic 1, device 2 sends a logic 0, and the logic 0 of device 2 will pull down SDA, so that the logic 1 of device 1 is canceled, which in turn causes device 1 to lose arbitration and enter the slave receive mode;
- 3) After device 1 fails in arbitration and enters the slave receiving mode, it still generates clock pulses until the current byte is sent, and no longer generates clock pulses for the transmission

of the next byte. After Device 2 wins arbitration, the data transfer on SDA is initiated by Device 2.

7.4 Functional description

The I2C bus uses two wires to transfer information between devices connected to the bus SCL (serial clock line) and SDA (serial data line). Filtering logic can filter glitches on the data bus to protect data integrity. Since there are only non-directional ports, the I2C component requires the use of open-drain buffers to the pins. Each device connected to the bus can be addressed by a specific address using software. The I2C standard is a true multi-master bus with a collision detection mechanism and an arbitration mechanism. It prevents data collisions when two or more Masters start transmitting data at the same time. I2C bus can be queried in the status register.

7.4.1 Functional block diagram

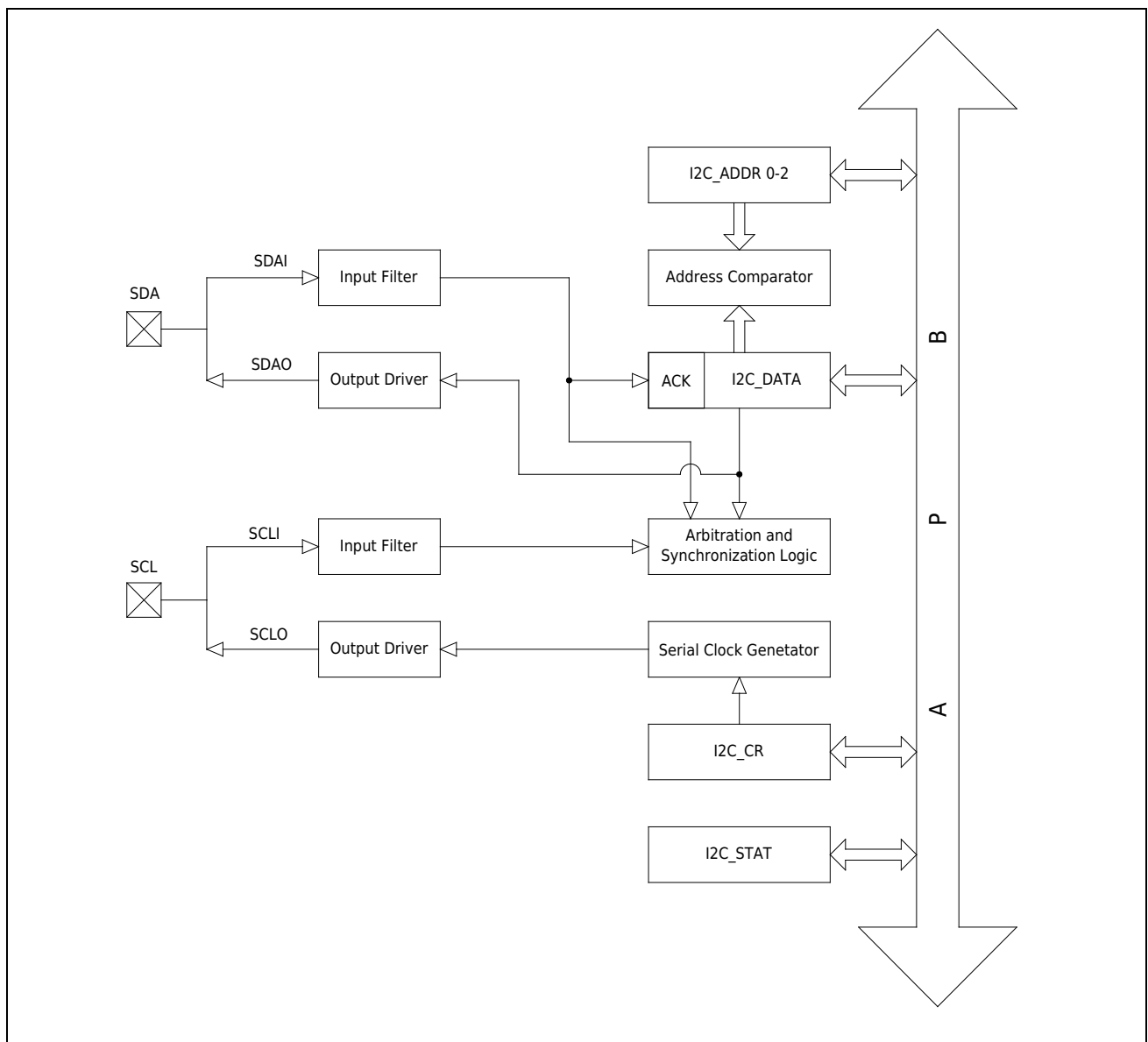


Figure 7-6 I2C function block diagram

7.4.2 Serial clock generator

The serial clock generator uses an 8-bit counter as the baud rate generator. The frequency relationship between the SCL signal and the PCLK signal is $f_{SCL} = f_{PCLK} / 8 / (I2C_BRR+1)$, where I2C_BRR should be greater than 0.

The following table lists the output frequency value of SCL signal when PCLK frequency is combined with I2C_BRR.

Table 7-1 I2C clock signal baud rate

PCLK (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	125	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

7.4.3 Input filter

The input signal is synchronous with PCLK, and the spike pulse signal lower than the period of PCLK will be filtered out. The intensity of filtering can be changed by adjusting the value of I2C_CR.FLT.

When this module is used as the host, if the value of I2C_BRR is less than or equal to 9, I2C_CR.FLT should be set to 1; if the value of I2C_BRR is greater than 9, I2C_CR.FLT should be set to 0.

When this module is used as a slave, if the ratio of PCLK to SCL frequency is less than or equal to 40, I2C_CR.FLT should be set to 1; if the ratio of PCLK to SCL frequency is greater than 40, I2C_CR.FLT should be set to 0.

7.4.4 Address comparator

The I2C address comparator compares the received 7-bit address with the three slave addresses. According to the configuration of the GC bit of the I2C_ADDR0 register, when the GC configuration is 0, the first received 8-bit byte is compared with the 7-bit Addr value of the I2C_ADDRx register. When the GC configuration is 1, the first received 8-bit The high 7 bits of the byte are compared with the broadcast address (0x00). If any one is the same, the I2C_CR.SI bit will be set and an interrupt request will be generated. Query the I2C_MATCH register to obtain the matching address as the serial number.

7.4.5 Response flag

I2C_CR.AA flag is the response flag. When the AA bit is 1, the I2C module responds with an acknowledgment bit after receiving the data, and when the AA bit is 0, the I2C module returns a non-acknowledgement bit after receiving the data.

7.4.6 Interrupt generator

I2C_CR.SI is an interrupt flag. Whenever the value of the status register (I2C_STAT) changes (except for 0xF8), the SI flag will be set to 1. When an interrupt is generated, the status of the I2C bus can be obtained by querying the status register (I2C_STAT) to determine the actual source of the interrupt. In order to proceed to the next step, the SI flag must be cleared by software.

7.4.7 Operating mode

The I2C component can realize 8-bit bidirectional data transmission, the transmission rate can reach 100Kbps in standard mode, 400Kbps in high-speed mode, and 1Mbps in ultra-high-speed mode, and can work in four modes: Master send mode, Master receiving mode, slave receiving mode, slave sending mode. There is also a special mode, general call mode, which operates in a similar way to slave receive mode.

■ Host send mode

The host sends multiple bytes to the slave, and the host generates a clock, so it is necessary to fill in the set value in I2C_BRR. I2C_CR.STA needs to be set to 1 in master send mode. When the bus is free, the master initiates a start bit START. I2C_CR.SI is set to 1 if successful. Next, write the slave address and write bit (SLA+W) into I2C_DATA, and after clearing the SI bit, SLA+W is sent on the bus.

After the master sends SLA+W and receives the slave acknowledgment bit ACK, SI is set to 1. The data is then sent according to the user-defined format. After all the data is sent, set I2C_CR.STO to 1, clear the SI bit and send a STOP signal, or send a repeated start signal for a new round of data transmission.

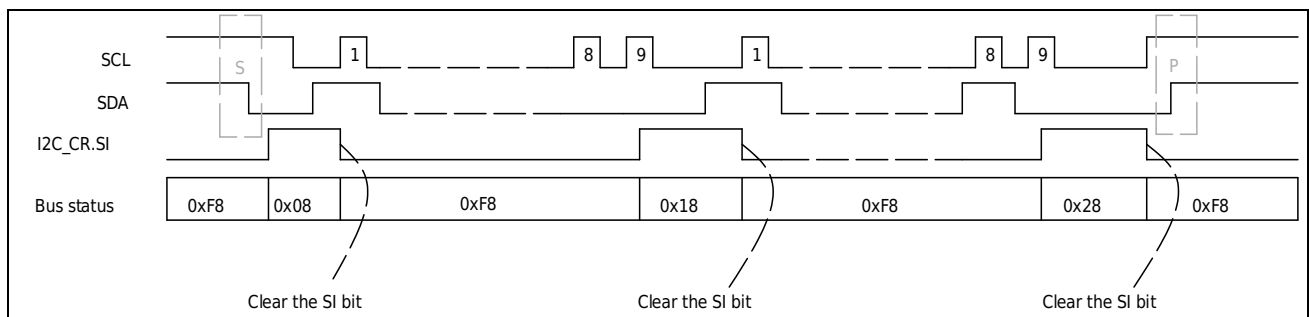


Figure 7-7 Data synchronization diagram of master sending mode

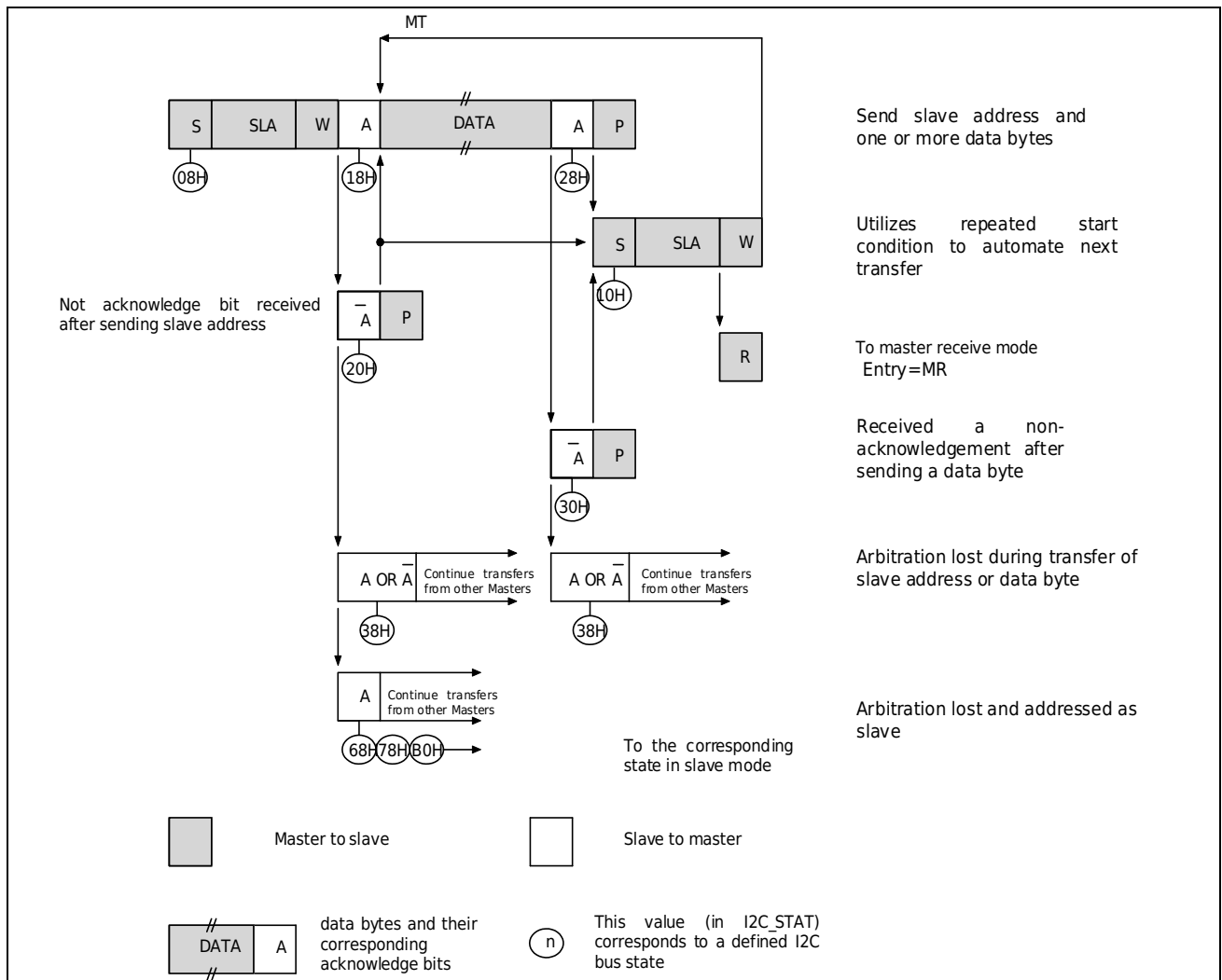


Figure 7-8 I2C master sending state diagram

■ Host receiving mode

The master receives the mode, and the data is transmitted by the slave. The initialization setting is the same as the host sending mode, after the host sends the start bit, I2C_DATA should be written into the slave address and " read bit " (SLA+R). I2C_CR.SI is set to 1 after receiving the slave acknowledge bit ACK. After SI is cleared to 0, it starts to receive slave data. If I2C_CR.AA is 1, the master responds with the acknowledge bit after receiving the data; if it is 0, the master does not respond (NACK) after receiving the data, and then the master can send a stop signal or repeat The start signal starts the next round of data transmission.

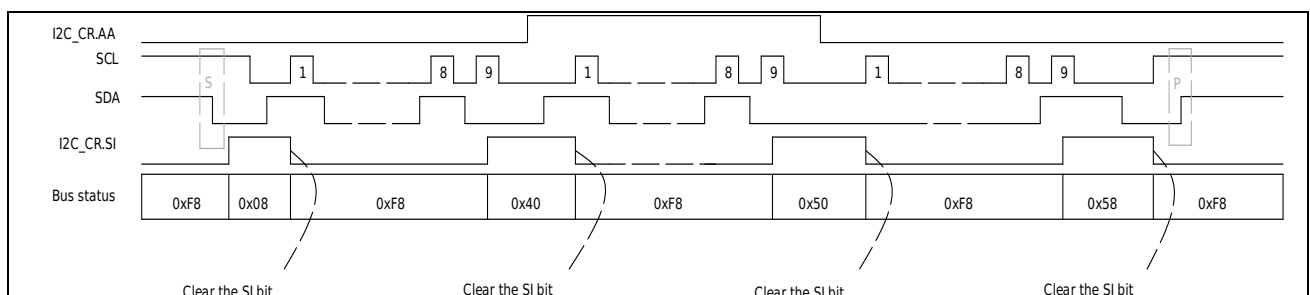


Figure 7-9 Data synchronization diagram of main receiving mode

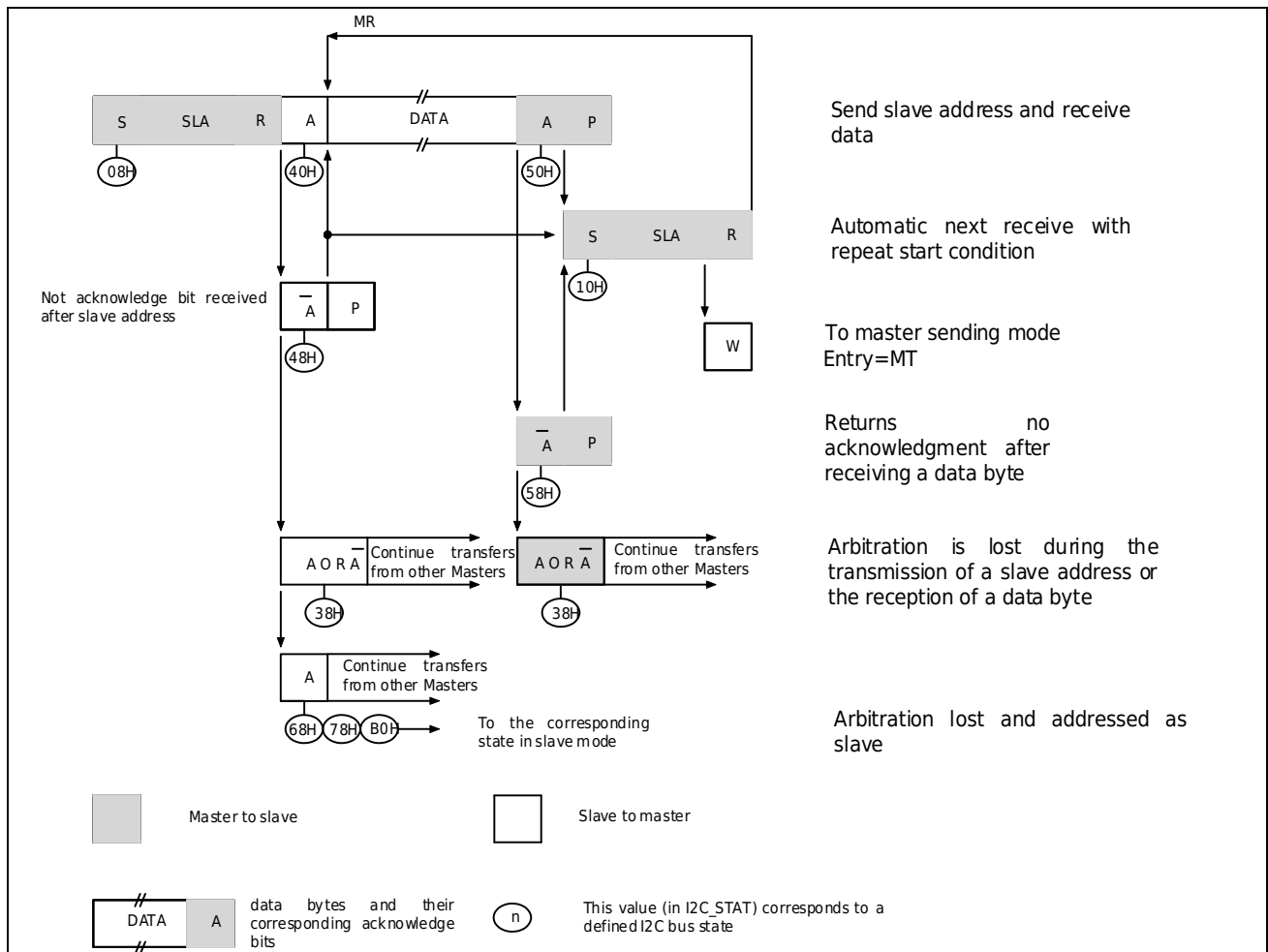


Figure 7-10 I2C master receiving state diagram

■ Slave receiving mode

In slave receiving mode, the slave receives data from the master. Before the transmission starts, I2C_ADDRx should be written to the slave address, and I2C_CR.AA is set to 1 to respond to the addressing of the master. After the above initialization, the slave enters idle mode and waits for a "write" signal (SLA+W). If the master fails to arbitrate, it will also directly enter the slave receiving mode.

When the slave is addressed by the "write" signal SLA+W, the SI bit needs to be cleared to receive data from the master. If I2C_CR.AA=0 during the transmission, the slave will return a non-acknowledgment bit NACK in the next byte, the slave will also turn into an unaddressed slave, terminate the connection with the master, no longer receive data, and I2C_DATA remains before received data. Slave address recognition can be restored by setting AA, which means that the AA bit temporarily detaches the I2C module from the I2C bus.

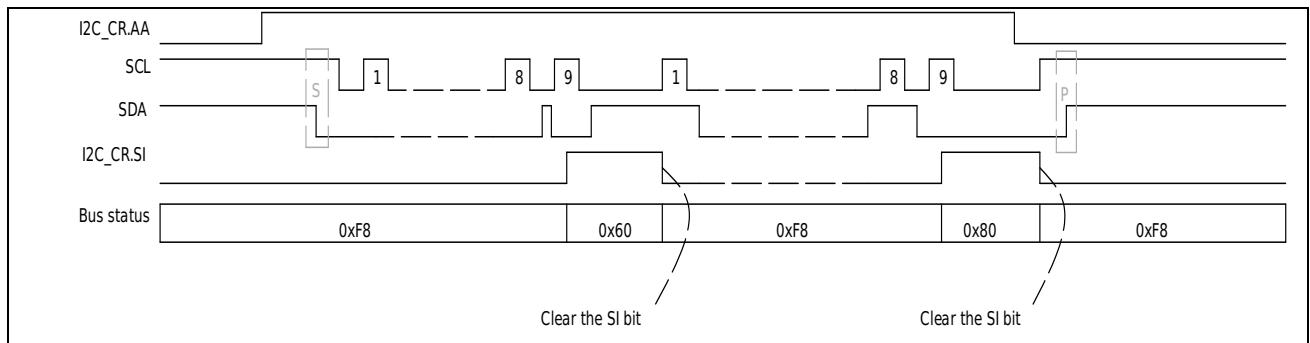


Figure 7-11 Slave Receive Mode Data Synchronization Diagram

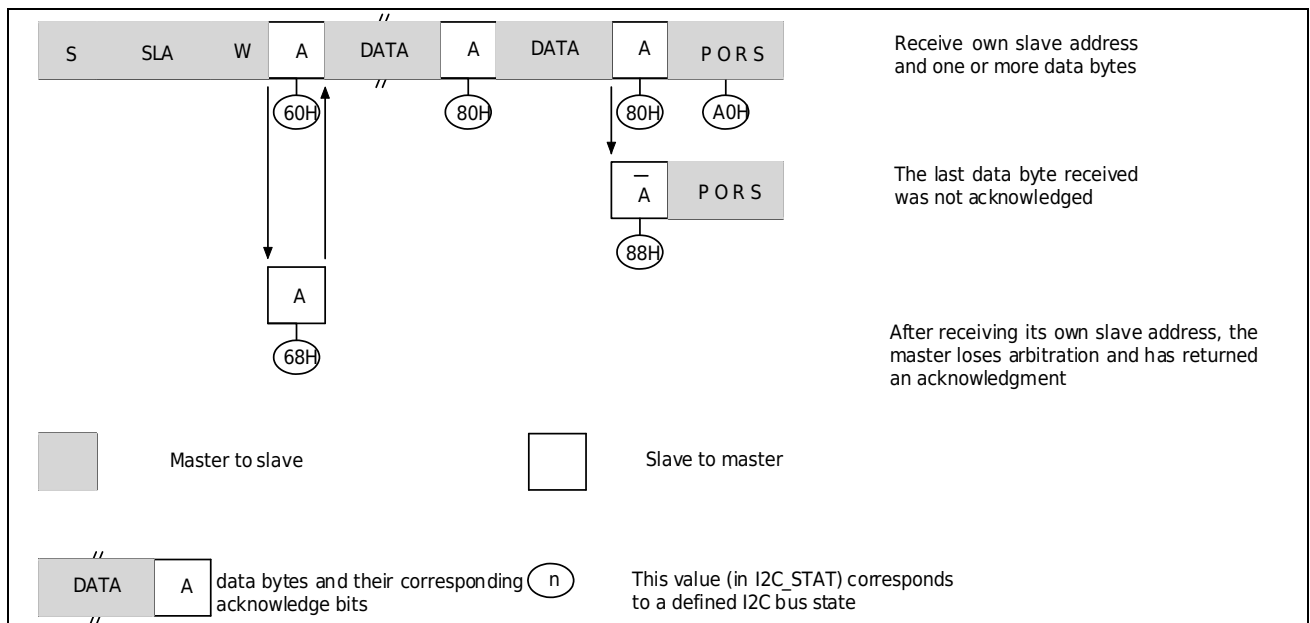


Figure 7-12 Slave receiving state diagram

■ Slave mode

In the slave sending mode, the data is sent from the slave to the master. After initializing the I2C_ADDRx and I2C_CR.AA values, the device waits until its own address is addressed by the "read" signal (SLA+R). If the master fails to arbitrate, it can also enter the slave sending mode. When the slave is addressed by the "read" signal SLA+R, it needs to clear SI to send data to the master. Normally the Master returns an acknowledge bit after each byte of data received. If I2C_CR.AA is cleared during the transmission, the slave will send the last byte of data, and send all 1 data in the next transmission, and turn itself into an unaddressed slave.

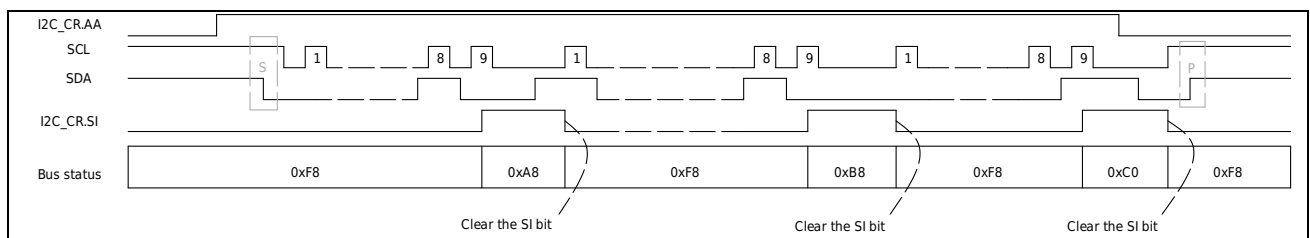


Figure 7-13 Slave mode data synchronization diagram

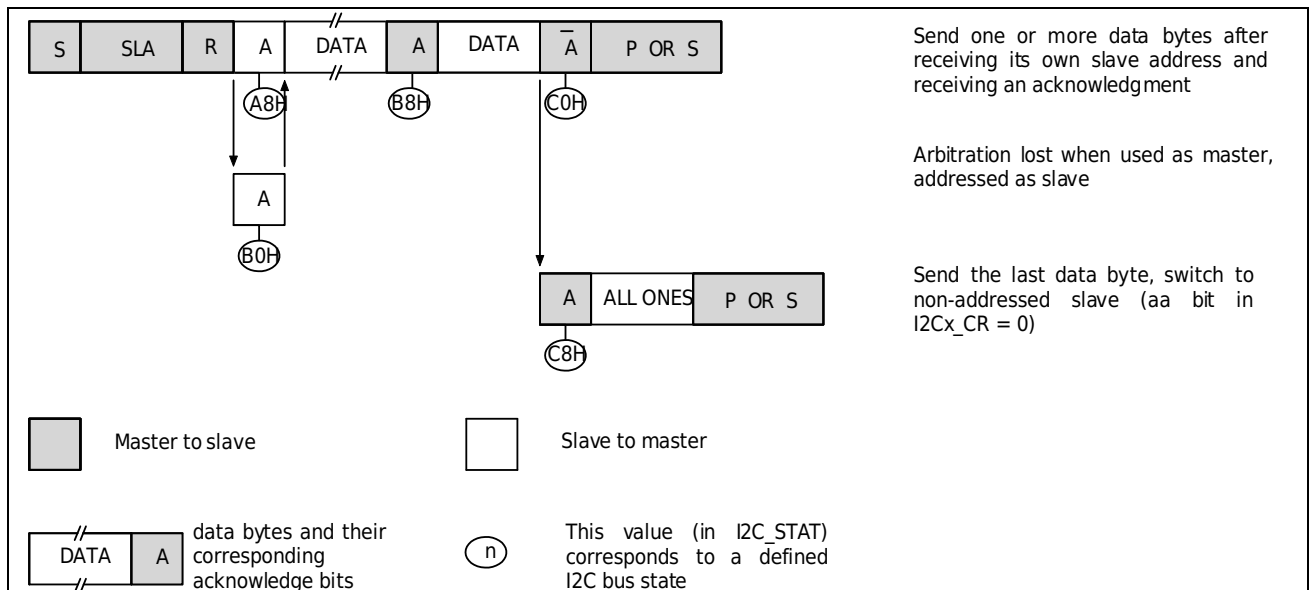


Figure 7-14 I2C Slave Transmit State Diagram

General call mode

The general call mode is a special slave receiving mode, the addressing mode is 0x00, and the slave address is 0. When both I2C_ADDR0.GC and I2C_CR.AA are set to 1, the general call mode is enabled. In this mode, the I2C_STAT value is different from the normal slave receiver mode I2C_STAT value. Arbitration failure may also enter general call mode.

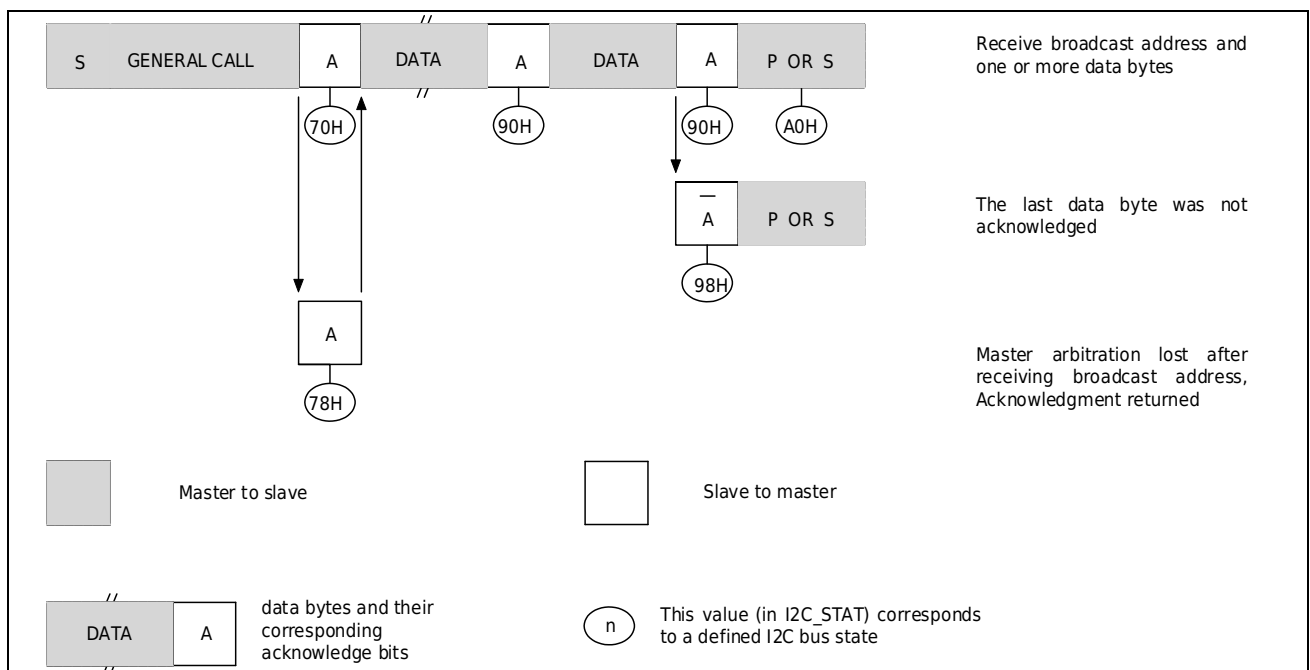


Figure 7-15 I2C General Call State Diagram

7.4.8 Status code expression

There are two special states in the I2C status register: F8H and 00H.

F8H: This status code indicates that no relevant information is available, because the serial interrupt flag "SI" has not been set. This condition occurs between other states and before the I2C module has started to perform a serial transfer.

00H: This status code indicates that a bus error occurred during I2C serial transmission. A bus error occurs when a START or STOP condition occurs at an illegal position in a format frame. These illegal locations refer to address bytes, data bytes, or acknowledge bits during serial transfers. when external disturbances affect the internal I2C block signals. SI is set when a bus error occurs.

Table 7-2 I2C status code expression

Status code	Description
Master sending mode	
08H	Start condition sent
10H	Repeated start condition sent
18H	SLA+W sent, ACK received
20H	SLA+W sent, not ACK received
28H	Data in I2C_DATA has been sent, ACK has been received
30H	Data in I2C_DATA has been sent, non-ACK has been received
38H	Loss of Arbitration when SLA+ reads or writes data bytes
Main receiving mode	
08H	Start condition sent
10H	Repeated start condition sent
38H	Lost arbitration in non-ACK (that is, the host finds that SDA has been pulled low by other devices when replying ACK)
40H	SLA+R sent, ACK received
48H	SLA+R sent, not ACK received
50H	Data byte has been received, ACK has been returned
58H	Data bytes received, non- ACK returned
Slave receive mode	
60H	Has received its own SLA+W and returned ACK
68H	When the master is in SLA+ read and write lost arbitration, has received its own SLA+W, and has returned ACK
80H	The previous addressing used its own slave address, received data bytes, and returned ACK
88H	The previous addressing used its own slave address, received data bytes, and returned non- ACK
A0H	After receiving a stop condition or a repeated start condition, it enters a non-addressed slave state and can still be addressed
Slave send mode	
A8H	Received its own SLA+R and returned ACK
B0H	When the Master loses arbitration, has received its own SLA+R, and has returned ACK

Status code	Description
B8H	ACK received
C0H	First data byte sent, not ACK received
C8H	Loaded data bytes have been sent, ACK has been received
General call mode	
70H	Received broadcast address (0x00), returned ACK
78H	When the master is in SLA+ read and write lost arbitration, the broadcast address has been received, and ACK has been returned
90H	The previous addressing used the broadcast address, data bytes have been received, and ACK has been returned
98H	The previous addressing used the broadcast address, the data byte has been received, and a non- ACK has been returned
A0H	After receiving a stop condition or a repeated start condition, it enters a non-addressed slave state and can still be addressed
Other miscellaneous status	
F8H	No relevant status information available, SI=0
00H	A bus error occurs during transmission, or external interference causes I2C to enter an undefined state

7.5 Programming example

7.5.1 Master sending example

Step1. SCL and SDA to the required pins according to the relevant description of the pin digital multiplexing function in the GPIO chapter; and configure the SCL and SDA pins to open-drain output mode;

Note: *SCL and SDA need to be externally connected with pull-up resistors according to the requirements of the I2C protocol.*

Step2. Set SysCtrl_PeriClkEn0.I2C to 1, enable I2C configuration clock and working clock;

Step3. Write 0 and 1 to PeriReset0.I2C in sequence to reset the I2C module;

Step4. Configure I2C_BRR so that the clock rate of SCL meets the application requirements;

Step5. Set I2C_BRREN to 1 to enable the SCL clock generator;

Step6. Set I2C_CR.EN to 1 to enable the I2C module;

Step7. Set I2C_CR.STA to 1, the bus tries to send the Start signal;

Step8. Wait for I2C_CR.SI to become 1 the Start signal has been sent to the bus, set I2C_CR.STA to 0;

Step9. Query I2C_STAT, if the register value is 0x08 or 0x10, continue to the next step, otherwise, perform error handling;

Step10. Write SLA+W to I2C_DATA set I2C_CR.SI to 0 and send SLA+W;

Step11. Wait for I2C_CR.SI to become 1, SLA+W has been sent to the bus;

Step12. Query I2C_STAT, if the register value is 0x18, proceed to the next step. Otherwise, perform error handling;

Step13. Write the data to be sent to I2C_DATA, set I2C_CR.SI to 0, and send the data;

Step14. Wait for I2C_CR.SI to become 1, the data has been sent to the bus;

Step15. Query I2C_STAT, if the register value is 0x28, proceed to the next step. Otherwise, perform error handling;

Step16. If the data to be sent is not completed, then jump to Step13 to continue execution;

Step17. Set I2C_CR.STO to 1, set I2C_CR.SI to 0, and the bus tries to send a Stop signal.

7.5.2 Master receive example

Step1. SCL and SDA to the required pins according to the relevant description of the pin digital multiplexing function in the GPIO chapter; and configure the SCL and SDA pins to open-drain output mode;

Note: SCL and SDA need to be externally connected with pull-up resistors according to the requirements of the I2C protocol.

Step2. Set SysCtrl_PeriClkEn0.I2C to 1, enable I2C configuration clock and working clock;

Step3. Write 0 and 1 to PeriReset0.I2C in sequence to reset the I2C module;

Step4. Configure I2C_BRR so that the clock rate of SCL meets the application requirements;

Step5. Set I2C_BRREN to 1 to enable the SCL clock generator;

Step6. Set I2C_CR.EN to 1 to enable the I2C module;

Step7. Set I2C_CR.STA to 1, the bus tries to send the Start signal;

Step8. Wait for I2C_CR.SI to become 1 the Start signal has been sent to the bus, set I2C_CR.STA to 0;

Step9. Query I2C_STAT, if the register value is 0x08 or 0x10, continue to the next step, otherwise, perform error handling;

Step10. Write SLA+R to I2C_DATA, set I2C_CR.SI to 0, and send SLA+R;

Step11. Wait for I2C_CR.SI to become 1, SLA+R has been sent to the bus;

Step12. Query I2C_STAT, if the register value is 0x40, continue to the next step, otherwise, perform error handling;

Step13. Set I2C_CR.AA to 1 to enable the response flag;

Step14. Set I2C_CR.SI to 0, the slave sends data, and the host sends ACK or NACK according to I2C_CR.AA;

Step15. Wait for I2C_CR.SI to become 1, read the received data from I2C_DATA;

Step16. Query I2C_STAT, if the register value is 0x50 or 0x58, continue to the next step, otherwise, perform error handling;

Step17. If the data to be received is only the last byte, set I2C_CR.AA to 0 and enable the non-response flag;

Step18. If the data to be received is not completed, jump to Step14 to continue execution;

Step19. Set I2C_CR.STO to 1, set I2C_CR.SI to 0, and the bus tries to send a Stop signal.

7.5.3 Slave receiving example

- Step1. SCL and SDA to the required pins according to the relevant description of the pin digital multiplexing function in the GPIO chapter; and configure the SCL and SDA pins to open-drain output mode;
- Step2. Set SysCtrl_PeriClkEn0.I2C to 1, enable I2C configuration clock and working clock;
- Step3. Write 0 and 1 to PeriReset0.I2C in sequence to reset the I2C module;
- Step4. Set I2C_CR.EN to 1 to enable the I2C module;
- Step5. Configure I2C_ADDRx as the slave address;
- Step6. Set I2C_CR.AA to 1 to enable the response flag;
- Step7. Wait for I2C_CR.SI to become 1 and be addressed by SLA+W;
- Step8. Query I2C_STAT, if the register value is 0x60, continue to the next step, otherwise, perform error handling;
- Step9. Set I2C_CR.SI to 0, the host sends data, and the slave returns ACK or NACK according to I2C_CR.AA;
- Step10. Wait for I2C_CR.SI to become 1, read the received data from I2C_DATA;
- Step11. Query I2C_STAT, if the register value is 0x80, continue to the next step, otherwise, perform error handling;
- Step12. If the data to be received is not completed, then jump to Step9 to continue execution;
- Step13. Set I2C_CR.AA to 0 and set I2C_CR.SI to 0.

7.5.4 Slave sending example

- Step1. SCL and SDA to the required pins according to the relevant description of the pin digital multiplexing function in the GPIO chapter; and configure the SCL and SDA pins to open-drain output mode;
- Step2. Set SysCtrl_PeriClkEn0.I2C to 1, enable I2C configuration clock and working clock;
- Step3. Write 0 and 1 to PeriReset0.I2C in sequence to reset the I2C module;
- Step4. Set I2C_CR.EN to 1 to enable the I2C module;
- Step5. Configure I2C_ADDRx as the slave address;
- Step6. Set I2C_CR.AA to 1 to enable the response flag;
- Step7. Wait for I2C_CR.SI to become 1 and be addressed by SLA+R;
- Step8. Query I2C_STAT, if the value of this register is 0xA8, continue to the next step, otherwise, perform error handling;
- Step9. Write the data to be sent to I2C_DATA, set I2C_CR.SI to 0, and send the data;
- Step10. Wait for I2C_CR.SI to become 1, the data has been sent to the bus;
- Step11. Query I2C_STAT, if the value of this register is 0xB8 or 0xC0, continue to the next step, otherwise, perform error handling;
- Step12. If the data to be sent is not completed, then jump to Step9 to continue execution;
- Step13. Set I2C_CR.AA to 0 and set I2C_CR.SI to 0.

7.6 Register description

7.6.1 Register list

Table 7-3 I2C base address list

Name	base address	Description
I2C0	0x4000 2800	I2C0 base address
I2C1	0x4000 2c00	I2C1 base address

Table 7-4 I2C register list

Register name	Offset address	Reset value
I2C Baud Rate Generator Enable Register (I2C_BRREN)	0x00	0x0000 0000
I2C Baud Rate Counter Configuration Register (I2C_BRR)	0x04	0x0000 0000
I2C Configuration Register (I2C_CR)	0x08	0x0000 0000
I2C Data Register (I2C_DR)	0x0c	0x0000 0000
I2C Status Register (I2C_STAT)	0x14	0x0000 0000
I2C Slave Address 0 Register (I2C_ADDR0)	0x10	0x0000 0000
I2C Slave Address 1 Register (I2C_ADDR1)	0x20	0x0000 0000
I2C Slave Address 2 Register (I2C_ADDR2)	0x24	0x0000 0000
I2C Slave Address Match Register (I2C_MATCH)	0x28	0x0000 0000

7.6.2 Baud Rate Generator Enable Register (I2C_BRREN)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res														EN	

Bit	Marking	Place name	Function	Read and write
b31~b1	Res	Reserved bit	Reserved, must remain at reset value	
b0	EN	Baud Rate Generator Enable Control	1: Enable 0: Prohibited	RW

7.6.3 Baud Rate Counter Configuration Register (I2C_BRR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res								BRR							

Bit	Marking	Place name	Function	Read and write
b31~b8	Res	Reserved bit	Reserved, must remain at reset value	
b7~b0	BRR	I2C bus SCL baud rate configuration	$f_{SCL} = f_{PCLK} / 8 / (BRR+1)$ where $BRR > 0$	RW

7.6.4 Configuration Register (I2C_CR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res									EN	STA	STO	SI	AA	Res	FLT

Bit	Marking	Place name	Function	Read and write
b31~b7	Res	Reserved bit	Reserved, must remain at reset value	
b6	EN	Module Enable Control	1: Enable 0: Prohibited	RW
b5	STA	Bus state control	1: send START to the bus 0: no function Note: User program should set this bit to 0 after START transmission is completed.	RW
b4	STO	Bus state control	1: Send STOP to the bus 0: no function Note: The hardware automatically sets this bit to 0 after the STOP transmission is completed.	RW
b3	SI	I2C interrupt flag	Read 1: I2C interrupt has occurred Read 0: No I2C interrupt occurred Write 1: no function Write 0: clear the I2C interrupt flag and make the state machine execute the next action	RW
b2	AA	response control	1: Send ACK during the answer phase 0: Send NAK during the answer phase	RW
b1	Res	Reserved bit	Reserved, must remain at reset value	
b0	FLT	I2C filter parameter configuration	1: Simple filtering, faster communication rate 0: Advanced filtering, higher anti-interference performance Note: See the chapter on input filter for details.	RW

7.6.5 Data Register (I2C_DR)

Reset value: 0x00000000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res									DR						

Bit	Marking	Place name	Function	Read and write
b31~b8	Res	Reserved bit	Reserved, must remain at reset value	
b7~b0	DR	Data register	In send mode, write the data to be sent In receive mode, read out received data	RW

7.6.6 Status Register (I2C_STAT)

Reset value: 0x00000000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res								STAT							
Bit	Marking	Place name	Function											Read and write	
b31~b8	Res	Reserved bit	Reserved, must remain at reset value												
b7~b0	STAT	I2C state register.	For the specific definition of the status value, see the chapter [Status code expression]											RO	

7.6.7 Slave Address 0 Register (I2C_ADDR0)

Reset value: 0x00000000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res								Addr							GC
Bit	Marking	Place name				Function									Read and write
b31~b8	Res	Reserved bit				Reserved, must remain at reset value									
b7~b1	Addr	I2C slave mode address 0													RW
b0	GC	Broadcast Address Acknowledgment Enable				1: Enable 0: Prohibited									RW

7.6.8 Slave Address 1 Register (I2C_ADDR1)

Reset value: 0x00000000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res								Addr							Res.
Bit	Marking	Place name				Function									Read and write
b31~b8	Res	Reserved bit				Reserved, must remain at reset value									
b7~b1	Addr	I2C slave mode address 1													RW
b0	Res	Reserved bit				Reserved, must remain at reset value									

7.6.9 Slave Address 2 Register (I2C_ADDR2)

Reset value: 0x00000000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res								Addr						Res.	

Bit	Marking	Place name	Function	Read and write
b31~b8	Res	Reserved bit	Reserved, must remain at reset value	
b7~b1	Addr	I2C slave mode address 2		RW
b0	Res	Reserved bit	Reserved, must remain at reset value	

7.6.10 Slave Address Match Register (I2C_MATCH)

Reset value: 0x00000000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res												AD2F	AD1F	AD0F	

Bit	Marking	Place name	Function	Read and write
b31~b3	Res	Reserved bit	Reserved, must remain at reset value	
b2	AD2F	I2C slave mode address 2 match flag	1: The device address received from the bus is the same as ADDR2 0: The device address received from the bus is not the same as ADDR2	RO
b1	AD1F	I2C slave mode address 1 match flag	1: The device address received from the bus is the same as ADDR1 0: The device address received from the bus is not the same as ADDR1	RO
b0	AD0F	I2C slave mode address 0 match flag	1: The device address received from the bus is the same as ADDR0 0: The device address received from the bus is not the same as ADDR0	RO

8 Serial Peripheral Interface (SPI)

8.1 Overview

The serial peripheral interface (SPI) provides data transmission function conforming to the SPI communication protocol in the master-slave mode, which can be used for synchronous serial communication between the MCU and external devices. This module supports full-duplex and single-line half-duplex and simplex communication.

8.2 Main features

- Can be configured as master or slave, support multi-machine mode
- The maximum frequency division factor of the host mode is PCLK/2
- The maximum frequency division factor of the slave mode is PCLK/4
- Multiple communication modes: full-duplex, single-wire half-duplex, simplex
- Two transmission orders: send and receive MSB first or send and receive LSB first
- Various data frame lengths: 4bits~16bits
- Two NSS methods: hardware control, software control
- Configurable serial clock polarity and phase
- DMA support

8.3 Functional description

8.3.1 Functional block diagram

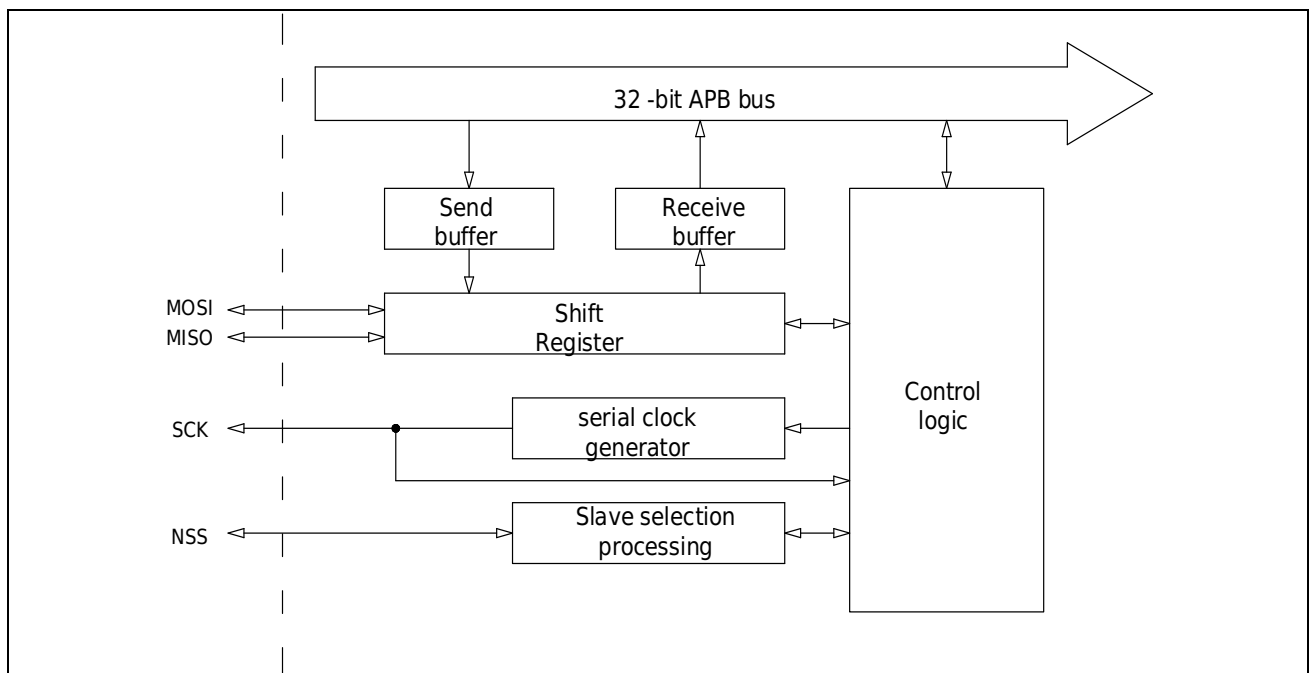


Figure 8-1 SPI Block Diagram

8.3.2 Communication Format and Timing

SPI serial frame format depends on serial clock polarity CPOL, serial clock phase CPHA and data frame format. The normal communication of the SPI interface requires that all masters and slaves must be configured to the same communication format.

The clock polarity CPOL refers to the level state of the serial clock in the idle state where the SPI does not transmit data. When SPI_CR0.CPOL is 0, the serial clock SCK is low when idle; when SPI_CR0.CPOL is 1, The serial clock SCK is high when idle.

The clock phase CPHA refers to the sequence in which the master and slave of the SPI sample and shift according to the serial clock SCK. When SPI_CR0.CPHA is 0, the master and slave sample data on the front edge and shift data on the rear edge; When CPHA is 1, the master-slave machine shifts data on the front edge, and samples data on the back edge.

The width of each frame of data can be configured from 4bits to 16bits by setting SPI_CR0.WIDTH.

By setting SPI_CR0.LSBF, the transmission sequence of each frame of data can be configured as LSB first or MSB first.

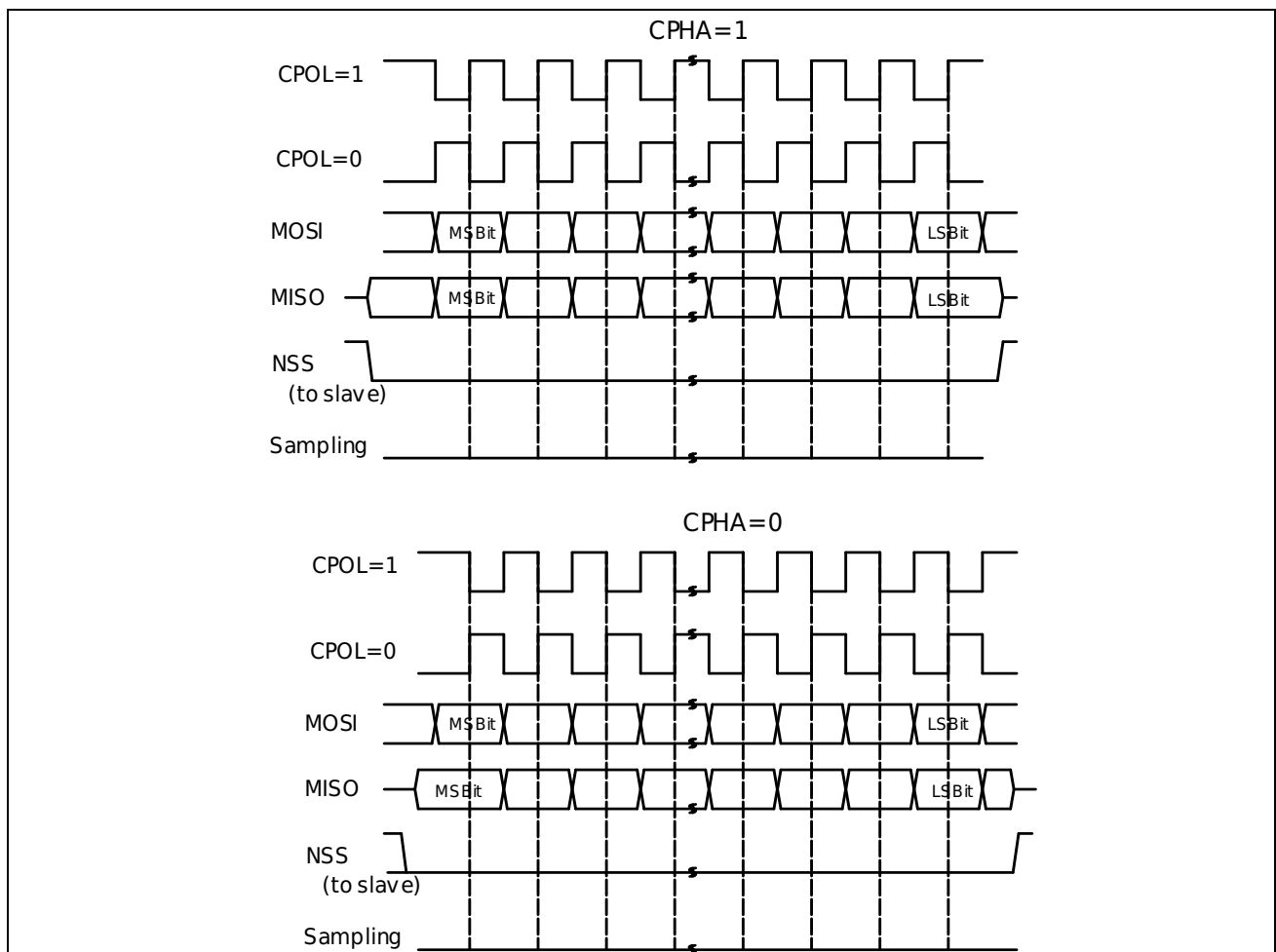


Figure 8-2 SPI communication format timing diagram

8.3.3 Slave Select Pin Configuration

This module provides a variety of slave selection methods to adapt to different application scenarios.

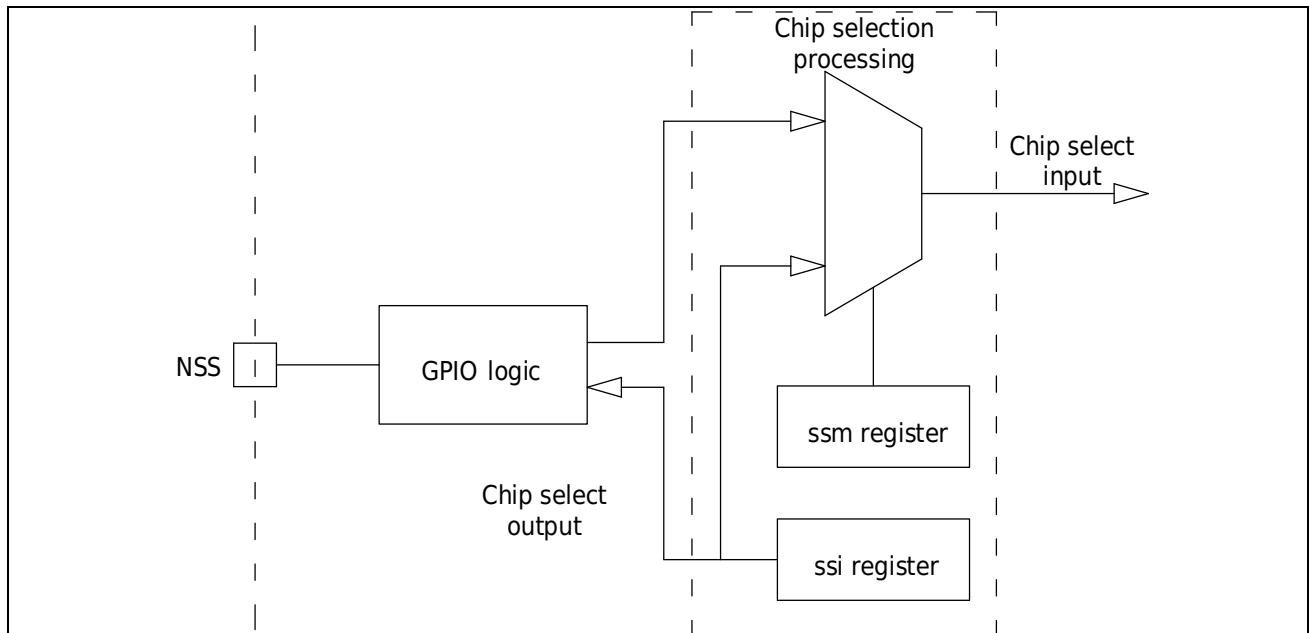


Figure 8-3 Slave select NSS pin configuration

When the SPI works in master mode and SPI_CR0.SSM is 0, the slave selects NSS as an input, and its state is determined by the level of the NSS pin.

When the SPI works in master mode and SPI_CR0.SSM is 1, the slave selects NSS as an output, and the output level of the NSS pin is determined by the internal register SPI_SSI.SSI. When SPI_SSI.SSI is 0, the NSS pin outputs a low level; when SPI_SSI.SSI is 1, the NSS pin outputs a high level.

When SPI works in slave mode and SPI_CR0.SSM is 0, the low level of NSS pin can select this module.

When the SPI works in slave mode and SPI_CR0.SSM is 1, the NSS pin no longer has the chip selection function. Only when the register SPI_SSI.SSI of this module is configured as 0 can this module be selected.

8.3.4 Full duplex communication

The full-duplex communication method allows the master and slave to perform two-way communication of sending and receiving at the same time relying on two unidirectional data lines. In this mode, the host provides a clock signal through the SCK pin to synchronize data input and output. The master and slave MOSI pins are connected to each other, and the MISO pins are connected to each other. The master sends data to the slave through the MOSI pin, and the slave returns the data through the MISO pin. Whenever the user writes data to the SPI_DR register, the SPI master will send data to the SPI slave through SCK and MOSI; and receive data from the slave through SCK and MISO.

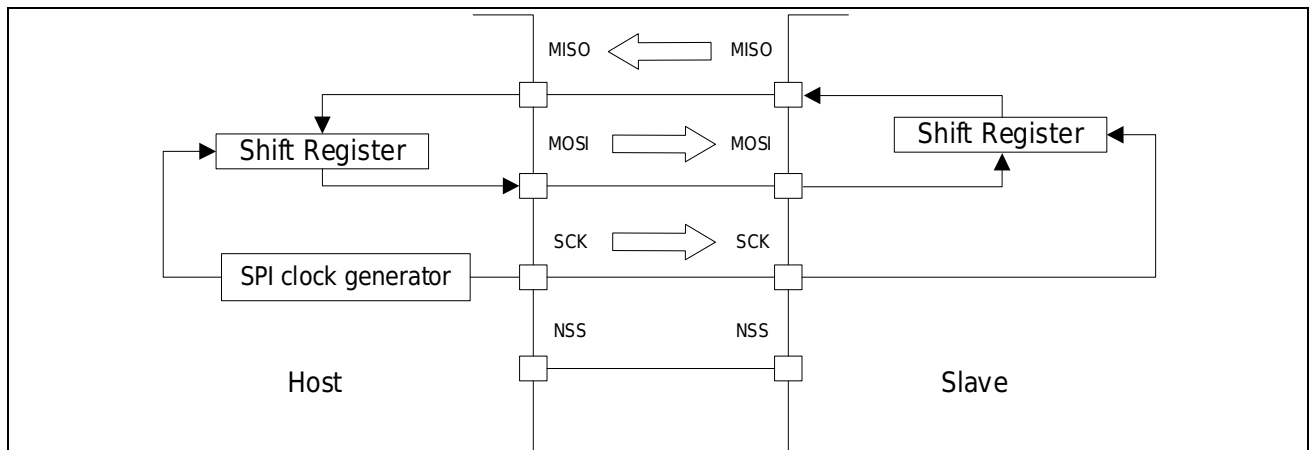


Figure 8-4 SPI full duplex communication

8.3.5 Single-wire half-duplex communication

The single-wire half-duplex communication method allows the master and slave to perform time-sharing two-way communication based on a two-way data line. The pin used as a master is MOSI, and the pin used as a slave is MISO. The direction of data transmission is determined by HDOE. When HDOE is 1, the data line is sent. Unused pins do not participate in SPI communication and can be used by other functions. When the host is in the receiving state, it is necessary to write any data to the DR register to trigger the SPI host to send SCK for data transmission.

When the master-slave performs single-wire half-duplex communication, since the switching of sending and receiving may not be synchronized, they may have a driving conflict for the bidirectional shared data line. In order to avoid possible damage to the device due to such situations, it is recommended to insert a current-limiting resistor in series on the data line to ensure Protect chip pins.

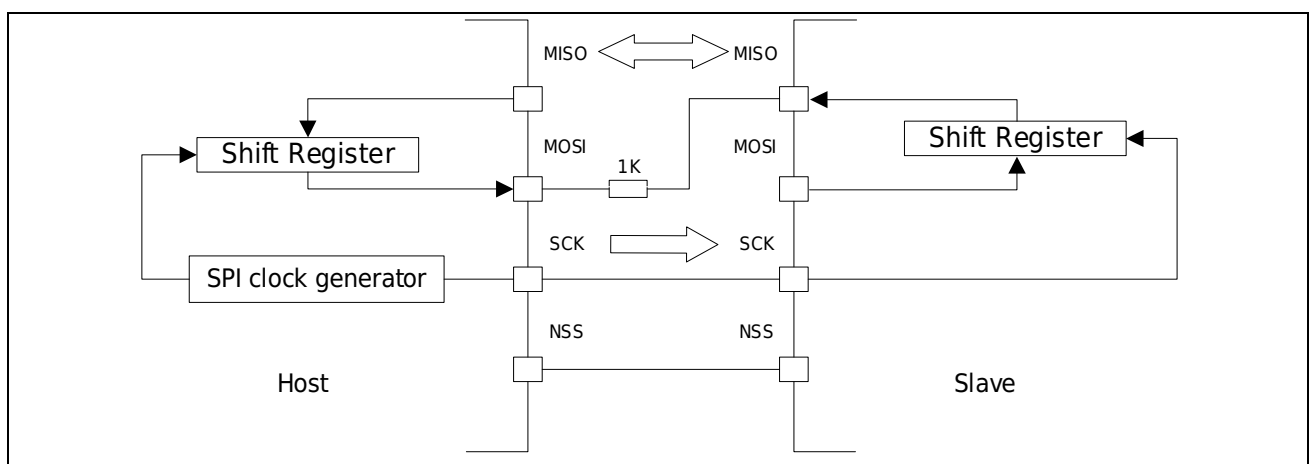


Figure 8-5 SPI single-wire half-duplex communication

8.3.6 Simplex communication

The simplex communication method allows the master and slave to rely on a one-way data line to perform one-way communication of fixed reception or fixed transmission. By setting SPI_CR0.CM,

select the simplex only send or simplex only receive mode. Unused pins do not participate in SPI communication and can be used by other functions.

When the master works in the sending state and the slave works in the receiving state, the master writes data to the DR register to trigger the SPI master to send SCK for data transmission.

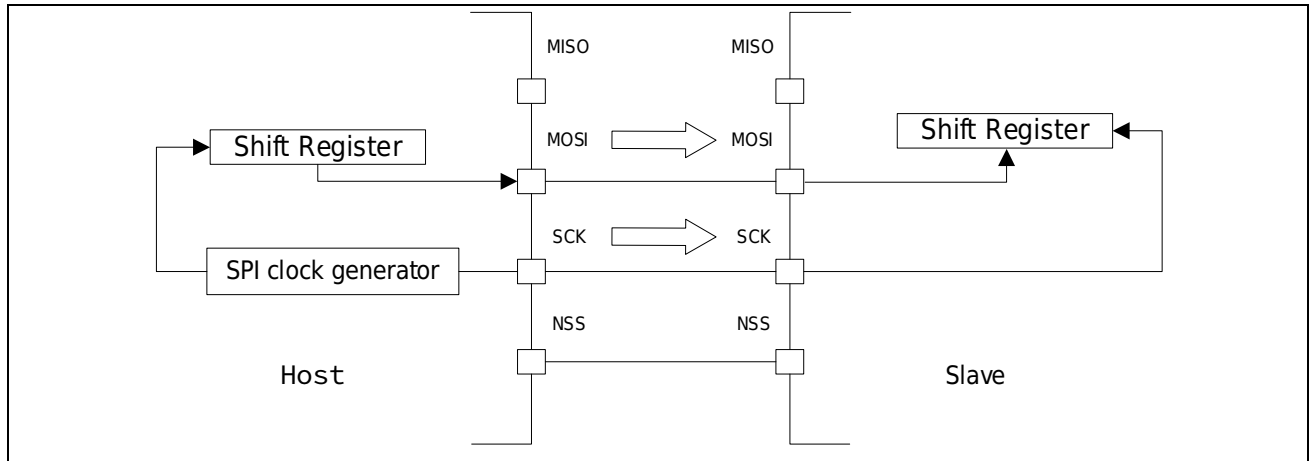


Figure 8-6 SPI simplex sending communication diagram

When the master works in the receiving state and the slave works in the sending state, the master also needs to write arbitrary data to the DR register to trigger the SPI master to send SCK for data transmission.

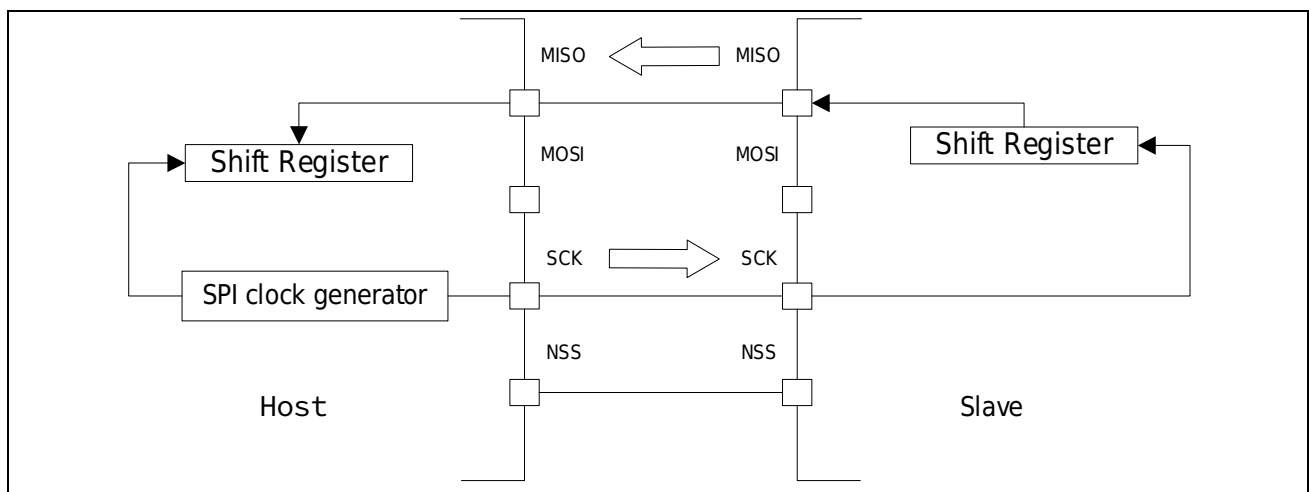


Figure 8-7 SPI Simplex Receive Communication Schematic

Note:

1. NSS pin can be used to provide hardware control flow between master and slave devices. Peripherals can also choose not to use these pins. After that, the hardware control flow must be handled internally for the master and slave.
2. In this configuration, two idle pins (MOSI or MISO) can be used as GPIO.

8.3.7 Multi-machine communication

In a multi-machine communication system, the slave selection NSS pin of the host is configured as a hardware input (SPI_CR0.SSM is 0), and the GPIO is responsible for outputting the slave selection signal, and the NSS input of the host is driven by the GPIO of another host. Once the slave selection NSS pin of the host is pulled low by the GPIO of other hosts, a mode error MODF will be generated, which will cause the SPI of this module to be forced to return to the disabled state (that is, SPI_CR0.EN is cleared), and the master-slave mode The selection register (SPI_CR0.MSTR) will also be cleared (that is, switched to slave mode). When the system needs to use SPI communication again, it needs to reset and initialize the module again. Therefore, when the host tries to obtain the SPI bus control right again, it needs to detect whether the NSS input is high. If it is high and there is no MODF, the bus control right can be successfully obtained. In a system with multiple slaves, each slave shares the serial clock and data lines, but requires its own independent slave selection signal, which requires the host to output the corresponding slave selection signal to each slave through different GPIOs.

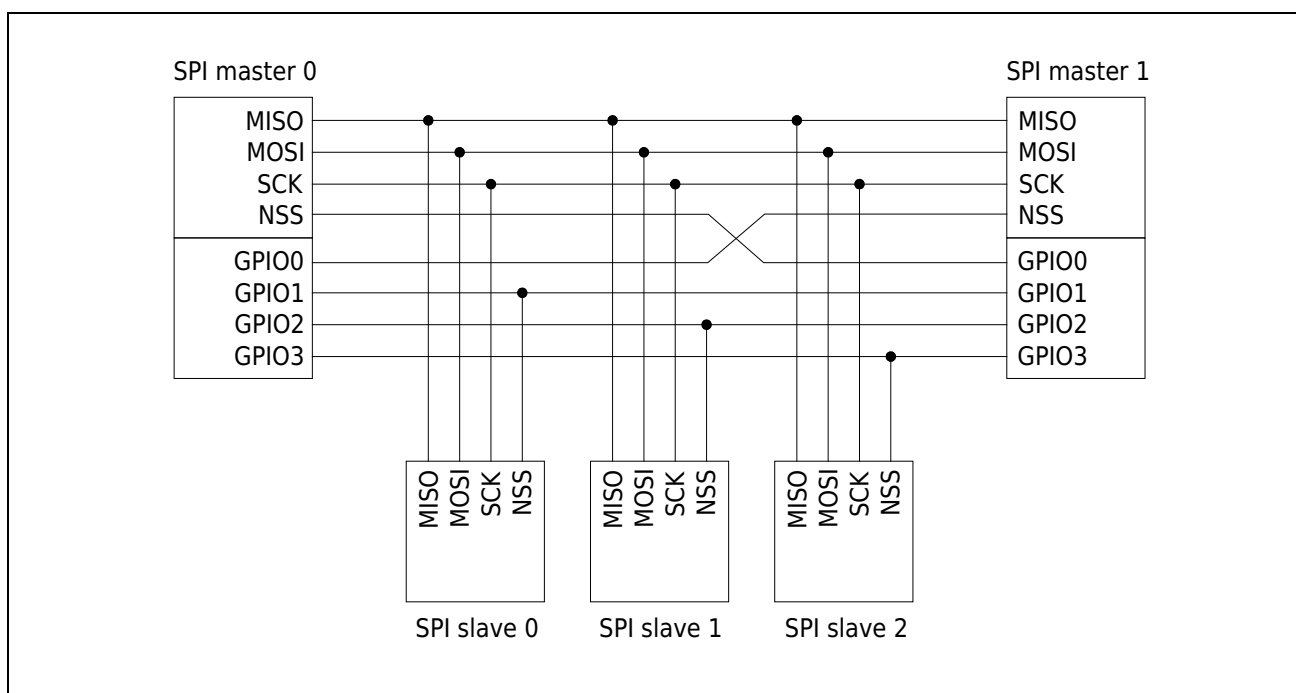


Figure 8-8 SPI multi-machine system

8.3.8 Host Mode Delayed Sampling

In SPI communication, due to the following factors:

- The time required for the slave to send data from the shift edge of the serial clock SCK to MISO
- Propagation time for data from slave data output MISO to master data input MISO

When the host receives valid data, there is a certain delay compared to the shift edge of the serial clock. In the traditional SPI protocol, the master needs to sample the MISO data at the sampling edge of half an SCK cycle after the shift edge, which requires that the above-mentioned delay must be less than half an SCK cycle, and the half SCK cycle is equal to the delay. The difference is to meet the requirement of MISO setup time. Therefore, when the above-mentioned delay is obvious, the rate of SPI communication will be restricted, and data errors will occur when the rate is high.

This module provides the function of delayed sampling in the host mode, which can effectively alleviate the communication speed bottleneck problem caused by the above problems. By setting the register SPI_CR0.SMP to 1, enable the master mode to delay sampling. SPI master, this module will take the sampling edge of the traditional SPI standard as the reference, and delay half the SCK cycle for sampling. This not only provides a higher tolerance to delay for the entire system, but also does not affect the data transmission of the next bit, thereby achieving a higher communication rate.

Note:

1. *This function is only valid in master mode, not in slave mode;*
2. *SPI_CR0.SMP does not affect the configuration of the communication format of the master and slave — whether this function is enabled or not, the master and slave must be configured with the same CPOL and CPHA according to the requirements of conventional SPI.*

8.3.9 Status Flags and Error Flags

SPI provides a variety of status flags to indicate the current working status, including general status flags and error flags.

Slave Select Input (NSS) state

The slave selection input status indicates the level of the currently input slave selection signal, which corresponds to the register SPI_SR.SSLVL. When SPI_SR.SSLVL is 0, the slave selection input is low; when SPI_SR.SSLVL is 1, the slave selection input is high.

Bus busy flag bit

The bus busy flag indicates whether the current SPI interface is busy or idle, corresponding to the register SPI_SR.BUSY. When SPI_SR.BUSY is 0, the SPI interface is in an idle state, and there is no data transmission currently; when SPI_SR.BUSY is 1, the SPI interface is in a busy state, and data transmission is currently being prepared or in progress.

Slave select input rising edge flag

The slave selection input rising edge flag indicates whether the current input slave selection signal has a rising edge, corresponding to the register SPI_SR.SSR. When SPI_SR.SSR is 0, there is no rising edge on the slave selection input; when SPI_SR.SSR is 1, there is a rising edge on the slave selection input.

Slave select input falling edge flag

The slave selection input falling edge flag indicates whether the current input slave selection signal has a falling edge, corresponding to the register SPI_SR.SSF. When SPI_SR.SSF is 0, there is no falling edge on the slave select input; when SPI_SR.SSF is 1, there is a falling edge on the slave select input.

Send buffer empty flag

The send buffer empty flag indicates whether the current SPI send buffer is loaded with data to be sent, corresponding to the register SPI_SR.TXE. When SPI_SR.TXE is 0, the SPI send buffer has been loaded with data to be sent; when SPI_SR.TXE is 1, the SPI send buffer is empty, and the next data to be sent can be written.

Receive buffer not empty flag

The receiving buffer non-empty flag indicates whether there is received data in the receiving buffer of the current SPI, corresponding to the register SPI_SR.RXNE. When SPI_SR.RXNE is 0, the SPI receive buffer has not been loaded with valid received data; when SPI_SR.RXNE is 1, the SPI receive buffer is loaded with received data, which is available for reading.

Slave selection error flag in slave mode

When the SPI is working in slave mode and valid data transmission is in progress, the input slave selection signal should remain selected (low level) to ensure normal communication. If the slave selection is pulled high during this period, it will When a slave selection error occurs, the corresponding register SPI_SR.SSERR will be set.

If a slave selection error occurs, the corresponding data transmission error will occur, and the SPI interface will also enter the unselected state due to the slave selection being pulled high. After a slave selection error occurs, the corresponding error flag is set and does not prevent subsequent normal data transmission, but the error flag will remain set until the user program clears it or the SPI is reset by the system control module.

It should be noted that the slave selection error SSERR is an error that only occurs in the slave mode, and it will only be triggered when the slave selection is pulled high when valid data is still being transmitted. When there is no valid data transmission between frames A normal slave select pull high does not trigger this error.

Receive buffer overflow error flag

The size of the SPI receive buffer is one frame. When the data in the receive buffer has not been read out and the data of the new frame has been received, a receive buffer overflow error occurs, and the corresponding register SPI_SR.OVF will be set.

When a receive buffer overflow error occurs, the original data in the buffer will be overwritten by the newly received data, resulting in data loss. After a receive buffer overflow error occurs, the corresponding error flag bit does not prevent subsequent normal data transmission, but the error flag bit will remain set until the user program clears it or the SPI is reset by the system control module.

Transmit buffer underflow error flag in slave mode

SPI send buffer is one frame. When the send buffer is empty in slave mode and the transmission of a new frame starts, a send buffer underflow error occurs, and the corresponding register SPI_SR.UDF will be set.

The occurrence of a send buffer underflow error in slave mode means that the send buffer is not filled with data in time, and the data sent to the master is missed. After a buffer underflow error occurs in the slave mode, the corresponding error flag bit does not prevent subsequent normal data transmission, but the error flag bit will remain set until the user program clears it or the SPI is reset by the system control module.

Mode error flag bit in host mode

When the SPI works in master mode and the slave selection pin is configured as an input, the input slave selection signal must remain unselected (high level) to ensure normal communication. If the

slave selection is pulled low, it will A mode error will occur and the corresponding register SPI_SR.MODF will be set.

A mode error will cause the SPI to be forced to return to the disabled state (that is, SPI_CR0.EN is cleared), and the master-slave mode selection register (SPI_CR0.MSTR) will also be cleared (that is, switch to slave mode). After a mode error occurs, setting the corresponding error flag does not prevent the user program from re-enabling the SPI and performing normal data transmission, but the error flag will remain set until the user program clears it or the SPI is reset by the system control module.

It should be noted that the mode error MODF is an error that only appears in the specific configuration of the host mode. It will only be triggered when the slave selection is configured as an input and pulled low, and the error will not be triggered in other cases.

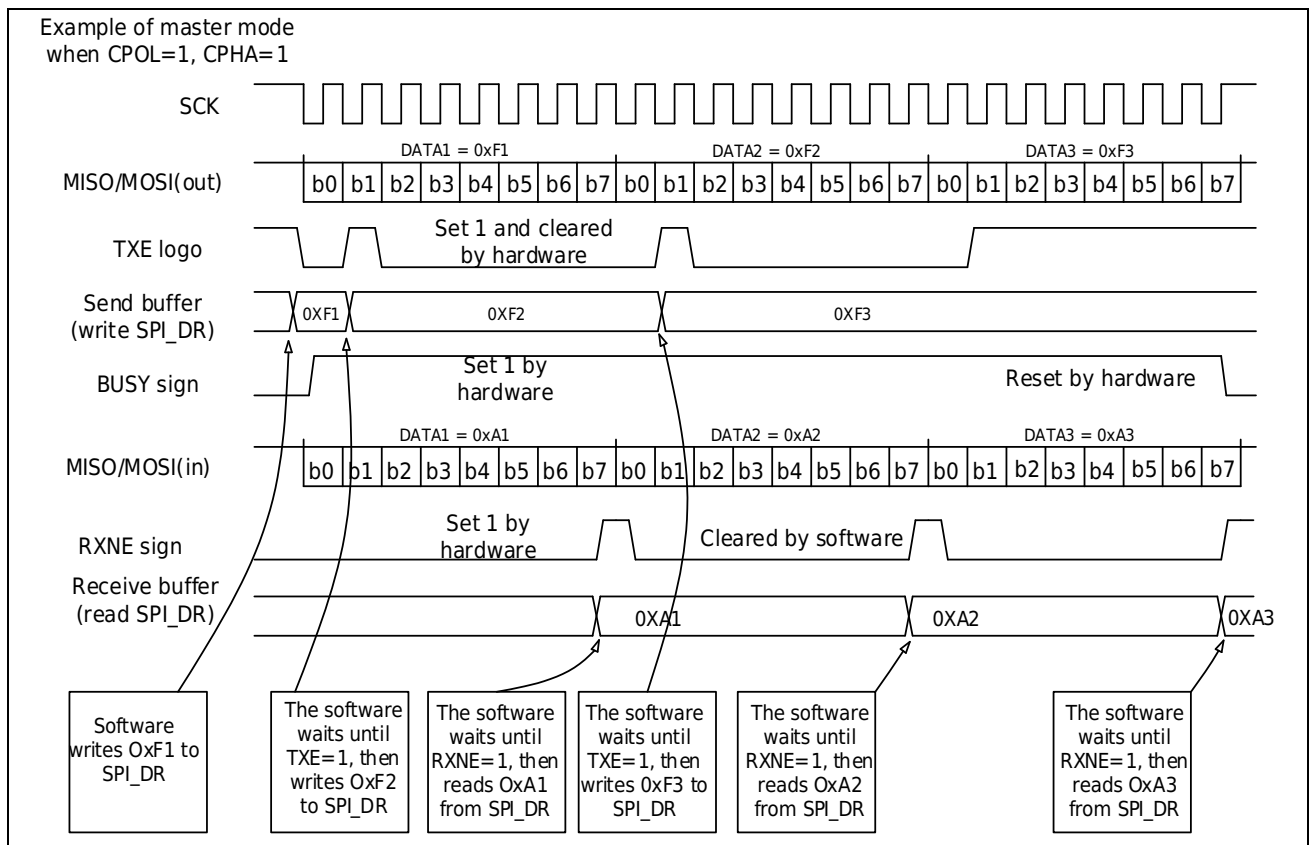


Figure 8-9 Explanation diagram of general flags in SPI master mode

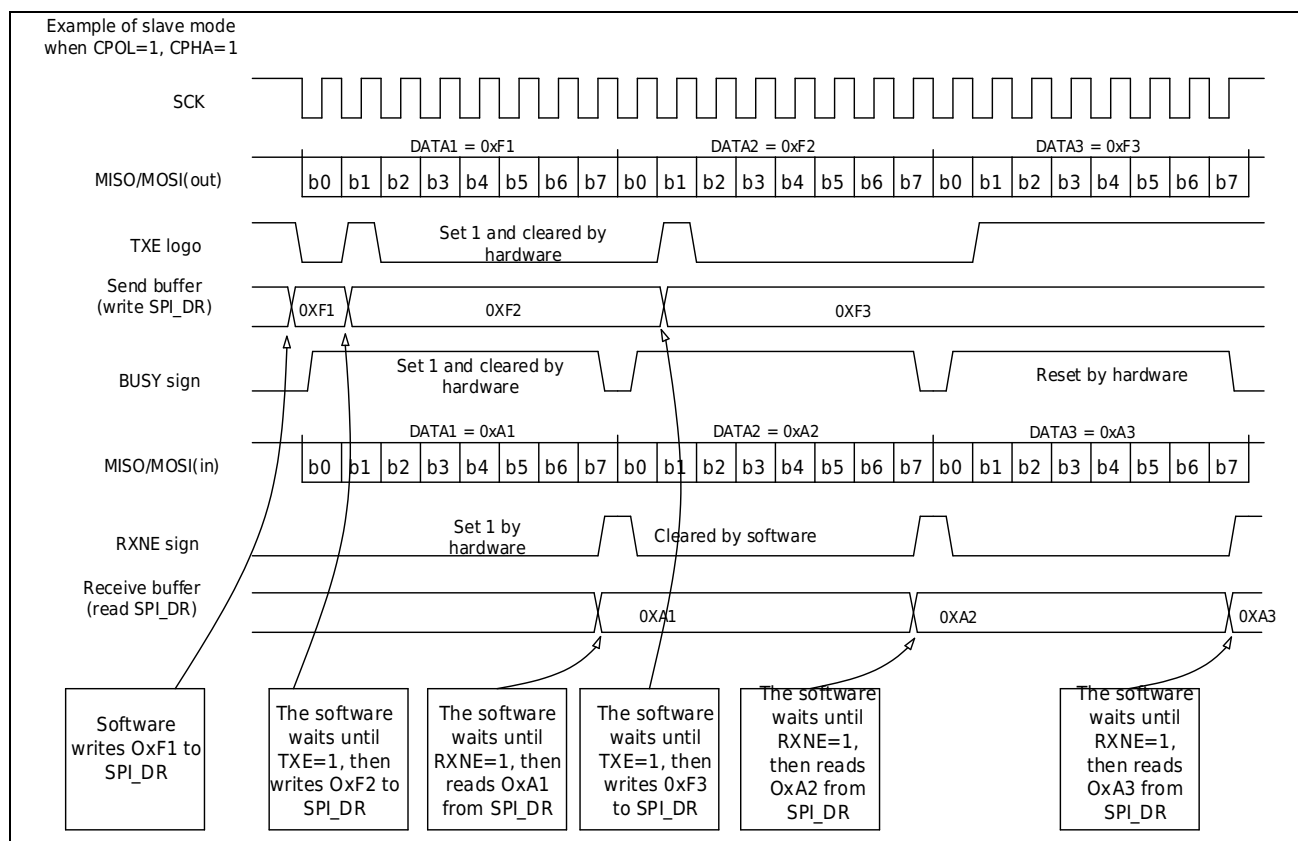


Figure 8-10 SPI slave mode general flag bit description diagram

8.3.10 Interrupt generation and clearing

SPI supports generating interrupt requests from different events, the specific information is as follows:

Table 8-1 SPI interrupt

Interrupt event	Corresponding flag (SPI_SR)	Corresponding interrupt enable (SPI_CR1)	How to clear the interrupt flag
Slave select input rising edge	SSR	SSRIE	Write 0 to SPI_ICR.SSRC
Slave select input falling edge	SSF	SSFIE	Write 0 to SPI_ICR.SSFC
Send buffer empty	TXE	TXEIE	Write data to the data register DR
Receive buffer is not empty	RXNE	RXNEIE	Read data from data register DR Or write 0 to SPI_ICR.RXNEC
Slave selection error in slave mode	SSERR	SSERRIE	Write 0 to SPI_ICR.SSERRC
receive buffer overflow error	OVF	OVFIE	Write 0 to SPI_ICR.OVFC
Slave mode transmit buffer underflow error	UDF	UDFIE	Write 0 to SPI_ICR.UDFC
Mode error in host mode	MODF	MODFIE	Write 0 to SPI_ICR.MODFC

As shown in the above table, each interrupt event has an independent interrupt enable control, and the interrupt generated by the error event also has an independent interrupt clear register. For the above interrupt requests, no matter whether the interrupt is enabled or not, the method described in the table can be used to clear the interrupt flag.

8.4 Programming example

8.4.1 Example of sending and receiving data in full duplex by the host

- Step1. Map NSS, SCK, MOSI, and MISO to the required pins according to the relevant description of the pin digital multiplexing function in the GPIO chapter;
- Step2. Set SYSCTRL_APB0EN.SPI to 1 to enable SPI configuration clock and working clock;
- Step3. Set CR0.MSTR to 1, select the host mode;
- Step4. Set CR0.BRR to configure the communication baud rate;
- Step5. Set CR0.CM to 0, select full-duplex communication mode;
- Step6. Set CR0.SSM to 1 to configure the slave selection mode;
- Step7. Set CR0.CPOL, CR0.CPHA, select serial clock polarity and phase;
- Step8. Set CR0.WIDTH to configure the data width of each frame;
- Step9. Set CR0.LSBF to configure the data sequence order when sending and receiving data;
- Step10. Set CR0.EN to 1 to enable the SPI module;
- Step11. Set SSI.SSI to 0, select the corresponding slave;
- Step12. Write the data to be sent into the DR register, and the SPI automatically starts data transmission;
- Step13. Query and wait for SR.RXNE to become 1 to complete the sending and receiving of a frame of data;
- Step14. Read the received data from the DR register and save it;
- Step15. If the data to be sent and received has not been completed, jump to Step12 to continue execution.

8.4.2 Example of sending and receiving data in full duplex from the slave

- Step1. Map NSS, SCK, MOSI, and MISO to the required pins according to the relevant description of the pin digital multiplexing function in the GPIO chapter;
- Step2. Set SYSCTRL_APB0EN.SPI to 1 to enable SPI configuration clock and working clock;
- Step3. Set CR0.MSTR to 0, select slave mode;
- Step4. Set CR0.CM to 0, select full-duplex communication mode;
- Step5. Set CR0.SSM to configure the slave selection mode;
- Step6. Set CR0.CPOL, CR0.CPHA, select serial clock polarity and phase;

- Step7. Set CR0.WIDTH to configure the data width of each frame;
- Step8. Set CR0.LSBF to configure the data sequence order when sending and receiving data;
- Step9. Set CR0.EN to 1 to enable the SPI module;
- Step10. Set ICR.TXES to 0 to clear the send buffer;
- Step11. The query waits for SR.SSLVL to become 0, and the SPI host selects this module;
- Step12. Write the data to be sent into the DR register, and prepare the data to be sent;
- Step13. Query and wait for SR.RXNE to become 1 to complete the sending and receiving of a frame of data;
- Step14. Read the received data from the DR register and save it;
- Step15. If the data to be sent and received has not been completed, jump to Step12 to continue execution;
- Step16. Wait for SR.SSLVL to become 1 to complete this communication.

8.4.3 Single-wire half-duplex sending and receiving data example

- Step1. Map NSS, SCK, MOSI of the host and NSS, SCK, MISO of the slave to the required pins according to the relevant description of the pin digital multiplexing function in the GPIO chapter;
- Step2. Set SYSCTRL_APB0EN.SPI to 1 to enable SPI configuration clock and working clock;
- Step3. The master sets CR0.MSTR to 1, and the slave sets CR0.MSTR to 0;
- Step4. The host sets CR0.BRR to configure the communication baud rate;
- Step5. Set CR0.CM to 11, select single-line half-duplex communication mode;
- Step6. Configure CR0.SSM to configure the slave selection mode;
- Step7. Configure CR0.CPOL, CR0.CPHA, select serial clock polarity and phase;
- Step8. Configure CR0.WIDTH to configure the data width of each frame;
- Step9. Configure CR0.LSBF to configure the data sequence order when sending data;
- Step10. Set CR0.EN to 1 to enable the SPI module;
- Step11. The master sets SSI.SSI to 0 and selects the corresponding slave; the slave waits for SR.SSLVL to become 0 to select this module;
- Step12. The master sets HDOE.HDOE to 1 and switches to the sending mode; the slave sets HDOE.HDOE to 0 and switches to the receiving mode;

- Step13. The host writes the data to be sent into the DR register, and the SPI automatically starts data transmission;
- Step14. The host query waits for SR.BUSY to become 0 to complete the transmission of a frame of data; the slave query waits for SR.RXNE to become 1 to complete the reception of a frame of data;
- Step15. The slave reads the received data from the DR register and saves it;
- Step16. If the data to be sent by the host is not completed, then jump to Step13 to continue execution;
- Step17. The master sets HDOE.HDOE to 0 and switches to the receiving mode; the slave sets HDOE.HDOE to 1 and switches to the sending mode;
- Step18. The slave query waits for SR.TXE to become 1, and the sending buffer is idle;
- Step19. If the data to be sent by the slave is not completed, write a frame of data to the DR register;
- Step20. writes arbitrary data to the DR register, and the SPI host sends SCK for data transmission;
- Step21. The host query waits for SR.RXNE to become 1, and completes the reception of a frame of data;
- Step22. The host reads the received data from the DR register and saves it;
- Step23. If the data to be sent by the slave is not completed, then jump to Step18 to continue execution;
- Step24. The slave waits for SR.SSLVL to become 1, sets HDOE.HDOE to 0, and completes this communication.

8.5 Register description

8.5.1 Register list

Table 8-2 SPI base address list

Name	base address	Description
SPI0	0x4000 3800	SPI0 base address
SPI1	0x4000 3C00	SPI1 base address

Table 8-3 SPI register list

Register name	Offset	Reset value
SPI Control Register 0 (SPI_CR0)	0x00	0x0000 0700
SPI Control Register 1 (SPI_CR1)	0x04	0x0000 0000
SPI Single-Wire Half-Duplex Output Enable Register (SPI_HDOE)	0x08	0x0000 0000
SPI Internal Slave Select Register (SPI_SSI)	0x0C	0x0000 0001
SPI Status Register (SPI_SR)	0x10	0x0000 0001
SPI Interrupt Clear Register (SPI_ICR)	0x14	0x0000 00FF
SPI Data Register (SPI_DR)	0x18	0x0000 0000

8.5.2 SPI Control Register 0 (SPI_CR0)

Reset value: 0x0000 0700

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
SMP	CM		SSM	WIDTH				LSBF	MSTR	CPOL	CPHA	BRR		EN	

Bit	Marking	Place name	Function	Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value	
b15	SMP	Host Mode Delayed Sample Bit	0: In traditional SPI mode, the data received by the host is sampled according to the settings of CPOL and CPHA 1: Delayed sampling method, the sampling time is delayed by half an SCK cycle compared with the traditional method	RW
b14~b13	CM	Communication configuration mode	00: full duplex bidirectional 01: simplex only send 10: simplex only 11: Single-wire half-duplex	RW
b12	SSM	Slave selection configuration	Host mode 1: SSI register determines slave select output value 0: NSS pin determines slave select input value Slave mode 1: SSI register determines slave select input value 0: NSS pin determines slave select input value	RW
b11~b8	WIDTH	Frame data width configuration	0011: 4bit 0100: 5bit ... 1110: 15bit 1111: 16bit	RW
b7	LSBF	Data frame high and low order selection	1: Transmit and receive the least significant bit LSB first 0: Transmit and receive the most significant bit MSB first	RW
b6	MSTR	Working mode configuration	1: Host mode 0: Slave mode	RW
b5	CPOL	Serial Clock Configuration	1: High level during standby 0: low level in standby	RW
b4	CPHA	Serial Clock Phase Configuration	1: Front edge shift / back edge sampling 0: sampling on the leading edge / shifting on the trailing edge	RW
b3~b1	BRR	Host mode baud rate configuration	000: PCLK/2 001: PCLK/4 010: PCLK/8 011: PCLK/16 ... 110: PCLK/128 111: reserved	RW
b0	EN	enable control	1: Enable SPI module 0: disable SPI module	RW

8.5.3 SPI Control Register 1 (SPI_CR1)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res						DMATXEN	DMA RXEN	MOD FIE	SSERRIE	OVFIE	UDFIE	SSRIE	SSFIE	RXNEIE	TXEIE

Bit	Marking	Place name	Function	Read and write
b31~b10	Res	Reserved bit	Reserved, must remain at reset value	
b9	DMATXEN	Transmit DMA enable	1: Enable 0: disabled	RW
b8	DMARXEN	Receive DMA enable	1: Enable 0: disabled	RW
b7	MODFIE	Mode Error Interrupt Enable Control	1: Enable 0: disabled	RW
b6	SSERRIE	Slave Select Error Interrupt Enable Control in Slave Mode	1: Enable 0: disabled	RW
b5	OVFIE	Receive buffer overflow error interrupt enable control	1: Enable 0: disabled	RW
b4	UDFIE	Transmit buffer underflow error interrupt enable control in slave mode	1: Enable 0: disabled	RW
b3	SSRIE	Slave select input rising edge interrupt enable control	1: Enable 0: disabled	RW
b2	SSFIE	Slave select input falling edge interrupt enable control	1: Enable 0: disabled	RW
b1	RXNEIE	Receive buffer non-empty interrupt enable control	1: Enable 0: disabled	RW
b0	TXEIE	Transmit buffer empty interrupt enable control	1: Enable 0: disabled	RW

8.5.4 SPI Single-Wire Half-Duplex Output Enable Register (SPI_HDOE)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res															HDOE

Bit	Marking	Place name	Function	Read and write
b31~b1	Res	Reserved bit	Reserved, must remain at reset value	
b0	HDOE	MOSI/MISO transceiver switching control	0: receive only 1: send only Note: only valid in single-line half-duplex communication	RW

8.5.5 SPI Internal Slave Select Register (SPI_SSI)

Reset value: 0x0000 0001

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res															SSI
Bit	Marking		Place name		Function										Read and write
b31~b1	Res		Reserved bit		Reserved, must remain at reset value										
b0	SSI		Internal Select	Slave	Valid when the SPI_CR0.SSM register is 1 (The register determines the slave selection output value in the master mode / the register determines the slave selection input value in the slave mode)										RW

8.5.6 SPI Status Register (SPI_SR)

Reset value: 0x0000 0001

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res						SSLVL	BUSY	MOD F	SSERR	OVF	UDF	SSR	SSF	RXNE	TXE

Bit	Marking	Place name	Function	Read and write
b31~b10	Res	Reserved bit	Reserved, must remain at reset value	
b9	SSLVL	State of slave select input	1: Slave select input is high 0: Slave select input is low Note: When the master mode and SPI_CR0.SSM=1 this bit is meaningless.	RO
b8	BUSY	Bus busy flag bit	1: Bus busy 0: Bus is idle	RO
b7	MODF	Mode error flag MODF is triggered when the slave select input is low in master mode)	1: Error 0: Normal	RO
b6	SSERR	Slave selection error flag in slave mode	1: Error 0: Normal	RO
b5	OVF	Receive buffer overflow error flag	1: Error 0: Normal	RO
b4	UDF	Transmit buffer underflow error flag in slave mode	1: Error 0: Normal	RO
b3	SSR	Slave select input rising edge flag	1: A rising edge has occurred 0: No rising edge occurs Note: When the master mode and SPI_CR0.SSM=1 this bit is meaningless.	RO
b2	SSF	Slave select input falling edge flag	1: A falling edge has occurred 0: No falling edge occurs Note: When the master mode and SPI_CR0.SSM=1 this bit is meaningless.	RO
b1	RXNE	Receive buffer not empty flag	1: Receive buffer is not empty 0: receive buffer is empty	RO
b0	TXE	Send buffer empty flag	1: send buffer empty 0: send buffer is not empty	RO

8.5.7 SPI Interrupt Clear Register (SPI_ICR)

Reset value: 0x0000 00FF

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res								MODF	SSERR	OVF	UDF	SSR	SSF	RXNE	TXE

Bit	Marking	Place name	Function	Read and write
b31~b8	Res	Reserved bit	Reserved, must remain at reset value	
b7	MODF	ISR.MODF flag clear bit	Write 0 to clear the ISR.MODF flag, read as 1.	R1W0
b6	SSERR	ISR.SSERR flag clear bit	Write 0 to clear the ISR.SSERR flag, read as 1.	R1W0
b5	OVF	ISR.OVF flag clear bit	Write 0 to clear the ISR.OVF flag, read as 1.	R1W0
b4	UDF	ISR.UDF flag clear bit	Write 0 to clear the ISR.UDF flag, read as 1.	R1W0
b3	SSR	ISR.SSR flag clear bit	Write 0 to clear the ISR.SSR flag, read as 1.	R1W0
b2	SSF	ISR.SSF flag clear bit	Write 0 to clear the ISR.SSF flag, read as 1.	R1W0
b1	RXNE	ISR.RXNE flag clear bit	Write 0 to clear ISR.RXNE flag, read as 1.	R1W0
b0	TXE	ISR.TXE clear bit	Writing 0 sets the ISR.TXE flag to 1 and reading it to 1.	R1W0

8.5.8 SPI Data Register (SPI_DR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
DR															

Bit	Marking	Place name	Function	Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value	
b15~b0	DR	SPI data bits	When sending data, write the data to be sent into this register When receiving data, read the received data from this register	RW

9 Clock Trim Module (CTRIM)

9.1 Overview

When the module works in calibration mode, it can automatically calibrate the output frequency of RCH/RCL in real time, so that the accuracy of RCH/RCL output frequency is no longer affected by environmental changes. When the module works in timer mode, it has general timing function and can still work normally under DeepSleep.

9.2 Main characteristics

- Calibration mode: support RCH/RCL automatic real-time calibration / single calibration
- Calibration mode: multiple calibration steps, quickly approaching the target
- Timing mode: can work in DeepSleep mode
- Timing mode: support automatic wake-up, the cycle is about 60us ~ 65536s (at 32.768KHz frequency)
- Timing mode: 5 clock sources to choose from
- Timing Mode: Toggle Output

9.3 Functional description

9.3.1 RCH clock calibration mode

When configuring CTRIM.CR1.MD as 0x00, the module works in RCH calibration mode. This mode can automatically calibrate the output frequency of the RCH in real time, so that the accuracy of the RCH output frequency is no longer affected by environmental changes.

RCH clock calibration is shown below.

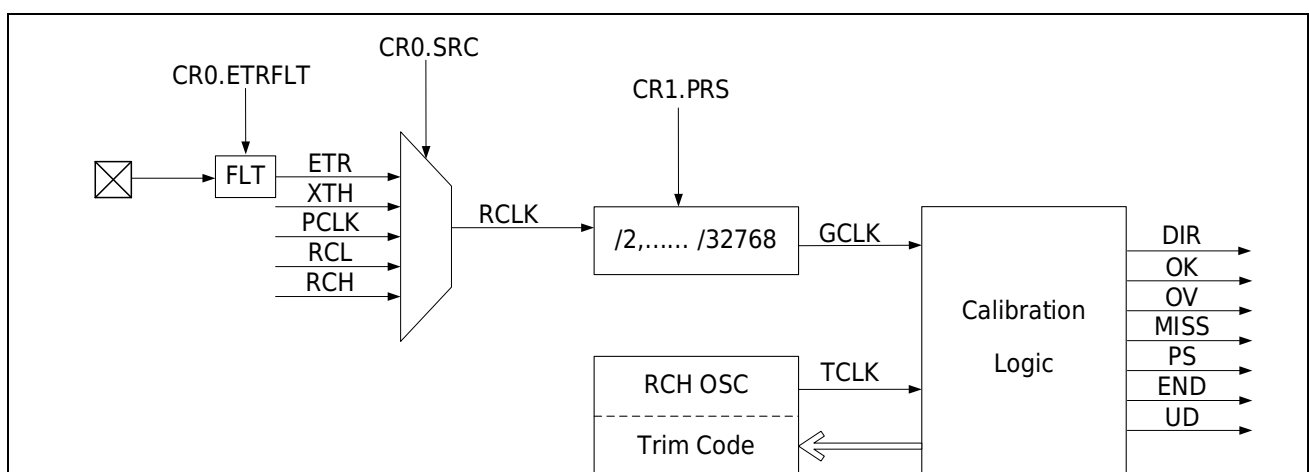


Figure 9-1 RCH clock calibration working block diagram

When calibrating RCH, it is necessary to provide an accurate low-frequency clock signal to the module, and the clock source can be selected through CTRIM_CR0.SRC. Note that the frequency of

the low-frequency clock signal should be less than 1/2 of the RCH frequency. The low-frequency clock signal generates GCLK after the frequency division output set by CTRIM_CR1.PRS, and GCLK is used as the input of the calibration logic unit.

When the automatic calibration starts to work, the error counter counts down the TCLK signal (RCH oscillation frequency) from CTRIM.ARR in each GCLK cycle, and counts down with the value set by CTRIM_CR0.STEP as the initial step amount, and counts to 0. Start counting up.

If the count value of the error counter in the GCLK cycle is less than or equal to the value of the CTRIM_FLIM register, then set CTRIM_ISR.OK to indicate that the RCH output frequency has reached the expectation, and set CTRIM_ISR.END;

If the count value of the error counter in the GCLK cycle is greater than the value of the CTRIM_FLIM register, the Trim Code of the RCH will be automatically adjusted and the next round of counting will be performed. The automatic adjustment rule is: at the end of each GCLK cycle, if the counting direction is down counting, then increase the step amount for the Trim Code of RCH; if the counting mode is up counting, then decrease the step amount for the RCH Trim Code.

The calibration logic will automatically adjust the value of the step to achieve the frequency of the RCH output approaching the target value successively. When the step size is adjusted to 1, set CTRIM_ISR.END to mark the end of automatic calibration.

When the error counter counts down to zero, generate a UD (underflow) event and set CTRIM_ISR.UD; when the error counter counts up to CTRIM_ARR, stop counting and generate a MISS event, set CTRIM_ISR.MISS. Whenever the rising edge of GCLK arrives, the error counter starts counting again.

After a single calibration is completed, the user needs to write CTRIM_TVAL[8:0] to the SYSCTRL_RCHTRIM register and close the calibration module as required.

9.3.2 RCL clock calibration mode

When configuring CTRIM.CR1.MD as 0x01, the module works in RCL calibration mode. This mode can automatically calibrate the output frequency of RCL in real time, so that the accuracy of RCL output frequency will no longer be affected by environmental changes.

RCL clock calibration is shown below.

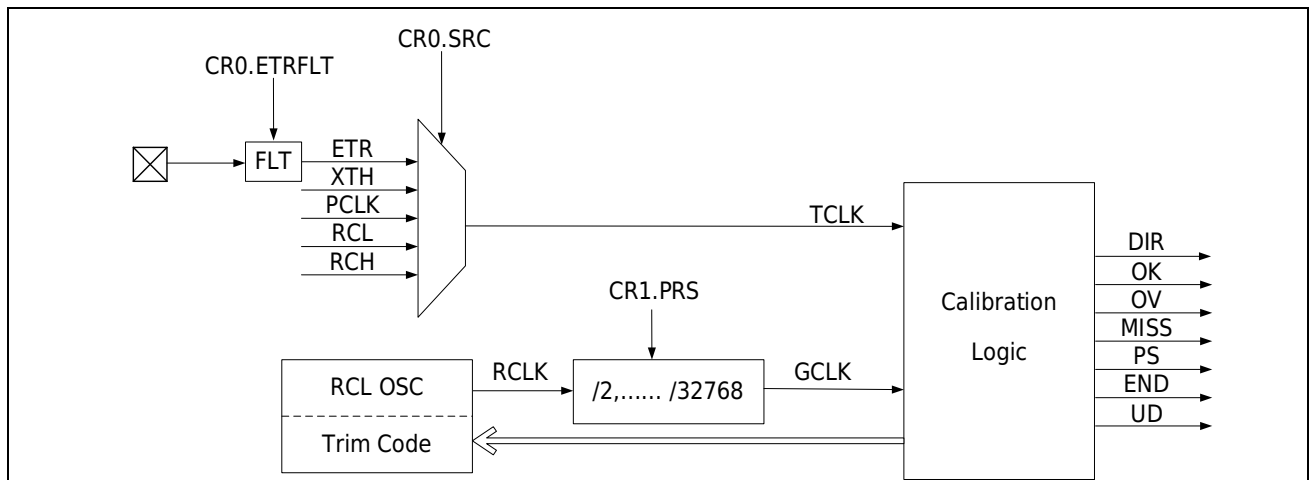


Figure 9-2 RCL clock calibration working block diagram

When calibrating RCL, it is necessary to provide an accurate high-frequency clock signal to the module, and the clock source can be selected through CTRIM_CR0.SRC. This clock source is input to the calibration logic as TCLK.

When the automatic calibration starts to work, the error counter counts down the TCLK clock signal starting from CTRIM.ARR in each GCLK (the RCL oscillation frequency after frequency division) and taking the value set by CTRIM_CR0.STEP as the initial step amount. , start counting up after counting to 0.

If the count value of the error counter in the GCLK period is less than or equal to the value of the CTRIM_FLIM register, then set CTRIM_ISR.OK to indicate that the RCL output frequency has reached the expectation, and set CTRIM_ISR.END;

If the count value of the error counter in the GCLK cycle is greater than the value of the CTRIM_FLIM register, the Trim Code of the RCL will be automatically adjusted and the next round of counting will be performed. The automatic adjustment rule is: at the end of each GCLK cycle, if the counting direction is down counting, the step amount is reduced for the Trim Code of RCL; if the counting mode is up counting, the step amount is increased for the RCL Trim Code.

The calibration logic will automatically adjust the value of the step to achieve the frequency of the RCL output approaching the target value successively. When the step size is adjusted to 1, set CTRIM_ISR.END to mark the end of automatic calibration.

When the error counter counts down to zero, generate a UD (underflow) event and set CTRIM_ISR.UD; when the error counter counts up to CTRIM_ARR, stop counting and generate a MISS event, set CTRIM_ISR.MISS. Whenever the rising edge of GCLK arrives, the error counter starts counting again.

After a single calibration is complete, write CTRIM_TVAL[8:0] to the SYSCTRL_RCLTRIM[8:0] register and turn off the calibration module as needed.

Note: `SYSCtrl_RCLTrim[9]` is the frequency selection bit and should remain unchanged.

9.3.3 Low Power Timer Mode

When `CTRIM.CR1.MD` is configured as 0x02 or 0x03, the module works in general low-power timer mode. In this mode, `CTIRM` has the timing and interrupt functions of a general-purpose timer. If the timer clock is selected as `RCL`, it can work normally in DeepSleep, supports automatic wake-up, and the cycle is 60us ~ 65536s. The operation block diagram of the general-purpose low-power timer is shown below.

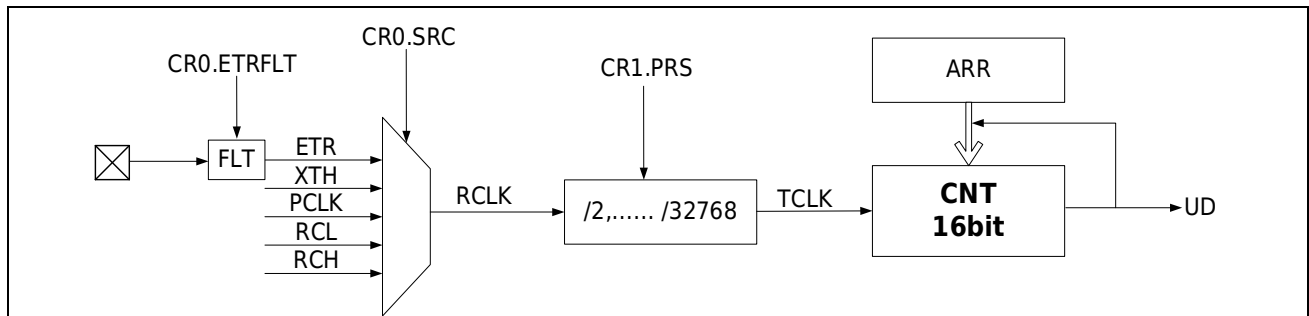


Figure 9-3 Low Power Timer Working Block Diagram

The timer count value is counted down from `ARR`. When the count reaches 0, a `UD` (underflow) event is generated and `CTRIM_ISR.UD` is set. If the interrupt (`CTRIM_IER.UD`) is enabled, a `UD` (underflow) interrupt will be generated. The counting cycle is `ARR+1`.

`TOG` output level flips once every time the timer overflows; the user can output the `TOG` signal to the pin through the port multiplexing function of the `GPIO` to drive an external circuit. **Note:** `TOG` output is not available when using external `ETR` input.

9.4 Software operation process

9.4.1 RCH Clock Automatic Calibration Example

- Step1. According to the port multiplexing function, configure the function of the corresponding port as `CTRIM_ETRTOG`, and provide it with a 1KHz square wave;
- Step2. Set `CTRIM_CR1.MD` to 0, select `RCH` clock calibration mode;
- Step3. Set `CTRIM_CR0.SRC` to 0, and the external `ETR` pin input signal is used as `RCLK`;
- Step4. Set `CTRIM_CR1.PRS` to 1, divide `RCLK` by 2, and `GCLK` frequency is 500Hz;
- Step5. Configure `CTRIM_ARR` to 47999, and the target count value in a `GCLK` count cycle is 48000;

Note: `RCH` target calibration value is 24MHz, `GCLK` frequency is 500Hz,

The theoretical counting value of the error counter to the `RCH` output frequency within 0.5s (500Hz) = 24M/ 500 = 48000,

That is, $CTRIM_ARR=48000-1=47999$;

Step6. Set `CTRIM_FLIM` to 240, select RCH correction accuracy to 0.5%;

Note: $48000 * 0.005 = 240$.

Step7. Set `CTRIM_CR0.STEP` to 4, and select the initial calibration step to be 16;

Step8. Set `CTRIM_CR1.AUTO` to 1 to enable automatic calibration;

Step9. Set `CTRIM_CR1.EN` to 1 to start clock calibration;

Step10. Wait for `CTRIM_ISR.END` to be 1, and the automatic calibration is completed.

Note: Query `CTRIM_ISR.OK` to determine whether the accuracy meets the setting requirements when Trim is completed.

9.4.2 RCL Clock Automatic Calibration Example

Step1. According to the port multiplexing function, configure the function of the corresponding port as `CTRIM_ETRTOG`, and provide it with a 1MHz square wave;

Step2. Set `CTRIM_CR1.MD` to 1, select RCL clock calibration mode;

Step3. Set `CTRIM_CR0.SRC` to 0, and the external ETR pin input signal is `TCLK`;

Step4. Set `CTRIM_CR1.PRS` to 0x0A, divide `RCLK` (`RCL` output frequency 32.768KHz) by 1024, and `GCLK` target frequency is 32Hz;

Step5. Configure `CTRIM_ARR` to 31249, and the target count value in a `GCLK` count cycle is 31250;

Note: RCL target calibration value is 32.768KHz, `GCLK` frequency is 32Hz,

*The theoretical count value of the error counter to `TCLK` (1MHz) within 31.25ms (32Hz)
 $= 1M / 32 = 31250$,*

That is, $CTRIM_ARR=31250-1=31249$;

Step6. Set `CTRIM_FLIM` to 156, select RCL correction accuracy to 0.5%;

Note: $31250 * 0.005 = 156$.

Step7. Set `CTRIM_CR0.STEP` to 4, and select the initial calibration step to be 16;

Step8. Set `CTRIM_CR1.AUTO` to 1 to enable automatic calibration;

Step9. Set `CTRIM_CR1.EN` to 1 to start clock calibration;

Step10. Wait for `CTRIM_ISR.END` to be 1, and the automatic calibration is completed.

Note: Query `CTRIM_ISR.OK` to determine whether the accuracy meets the setting requirements when Trim is completed.

9.5 Register description

9.5.1 Register list

Base address: 0x4000 8000

Table 9-1 CTRIM register list

Register name	Offset address	Reset value
Autoload Register (CTRIM_ARR)	0x00	0x0000 FFFF
Error Counter Register (CTRIM_CNT)	0x04	0x0000 0000
Control Register 0 (CTRIM_CR0)	0x0C	0x0000 0000
Control Register 1 (CTRIM_CR1)	0x10	0x0000 0000
Interrupt Enable Register (CTRIM_IER)	0x14	0x0000 0000
Interrupt and Status Register (CTRIM_ISR)	0x18	0x0000 0000
Interrupt Flag Clear Register (CTRIM_ICR)	0x1C	0x0000 003F
Error Result Register (CTRIM_FCAP)	0x20	0x0000 FFFF
TRIM result register (CTRIM_TVAL)	0x24	0x0000 0100
Error Upper Limit Register (CTRIM_FLIM)	0x28	0x0000 0010

9.5.2 Autoload Register (CTRIM_ARR)

Reset value: 0x0000 FFFF

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
ARR															
Bit	Marking	Place name	Function												Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value												
b15~b0	ARR	Error counter reload value	Error counter reload value												RW

9.5.3 Error Counter Register (CTRIM_CNT)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CNT															
Bit	Marking	Place name	Function												Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value												
b15~b0	CNT	Error counter count value	Error counter count value												RO

9.5.4 Control Register 0 (CTRIM_CR0)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res					SRC			Res	ETRFLT			Res	STEP		

Bit	Marking	Place name	Function	Read and write
b31~b11	Res	Reserved bit	Reserved, must remain at reset value	
b10~b8	SRC	Clock source selection	000: ETR 001: XTH 010: reserved 011: PCLK 100: RCL 101: RCH 110: reserved 111: reserved	RW
b7	Res	Reserved bit	Reserved, must remain at reset value	
b6~b4	ETRFLT	External input pin (ETR) filter configuration	000: no filtering 001: $F_{sample} = PCLK$, N=2 010: $F_{sample} = PCLK$, N=4 011: $F_{sample} = PCLK$, N=6 100: $F_{sample} = PCLK/4$, N=4 101: $F_{sample} = PCLK/4$, N=6 110: $F_{sample} = PCLK/8$, N=4 111: $F_{sample} = PCLK/8$, N=6	RW
b3	Res	Reserved bit	Reserved, must remain at reset value	
b2~b0	STEP	TRIM initial step configuration	000: 0x10 001: 0x02 010: 0x04 011: 0x08 100: 0x10 101: 0x20 110: 0x40 111: 0x80	RW

Note: This register can only be changed when EN is 0.

9.5.5 Control Register 1 (CTRIM_CR1)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res							OST	PRS				AUTO	MD		EN

Bit	Marking	Place name	Function	Read and write
b31~b9	Res	Reserved bit	Reserved, must remain at reset value	
b8	OST	Calibration Mode Configuration	0: real-time calibration mode, the TRIM process continues 1: One-shot calibration mode, end when STEP becomes 1	RW
b7~b4	PRS	Reference Clock Prescaler Configuration	0000: Reserved, not configurable 1000: 256 frequency division 0001: 2 frequency division 1001: 512 frequency division 0010: 4 frequency division 1010: 1024 frequency division 0011: 8 frequency division 1011: 2048 frequency division 0100: 16 frequency division 1100: 4096 frequency division 0101: 32 frequency division 1101: 8192 frequency division 0110: 64 frequency division 1110: 16384 frequency division 0111: 128 frequency division 1111: 32768 frequency division	RW
b3	AUTO	RCH/RCL auto- TRIM enable configuration	1: Enable 0: Prohibited	RW
b2~b1	MD	Working mode configuration	00: RCH calibration mode, CR0.SRC needs to select a low-speed precision clock 01: RCL calibration mode, CR0.SRC needs to select high-speed and precise clock 1x: Timer mode	RW
b0	EN	Timer Enable Control	1: Enable 0: Prohibited	RW

9.5.6 Interrupt Enable Register (CTRIM_IER)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res										OK	OV	MISS	PS	END	UD

Bit	Marking	Place name	Function	Read and write
b31~b6	Res	Reserved bit	Reserved, must remain at reset value	
b5	OK	Auto Trim results in range interrupt enable configuration	1: Enable 0: Prohibited	RW
b4	OV	TrimCode overflow interrupt enable configuration	1: Enable 0: Prohibited	RW
b3	MISS	Count failure interrupt enable configuration	1: Enable 0: Prohibited	RW
b2	PS	GCLK reference clock rising edge interrupt enable configuration	1: Enable 0: Prohibited	RW
b1	END	Auto Trim End Interrupt Enable Configuration	1: Enable 0: Prohibited	RW
b0	UD	Error Counter Underflow Interrupt Enable Configuration	1: Enable 0: Prohibited	RW

9.5.7 Interrupt and Status Register (CTRIM_ISR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res									DIR	OK	OV	MISS	PS	END	UD

Bit	Marking	Place name	Function	Read and write
b31~b7	Res	Reserved bit	Reserved, must remain at reset value	
b6	DIR	Error counter counting direction	0: Counting up, RCH output frequency is higher than the target frequency, RCL output frequency is lower than the target frequency 1: count down, RCH output frequency is lower than the target frequency, RCL output frequency is higher than the target frequency	RO
b5	OK	Trim precision mark	1: When Trim is completed, the accuracy meets the requirements of FLIM 0: Accuracy does not meet FLIM requirements when Trim is completed	RO
b4	OV	TrimCode overflow flag	1: TrimCode of RCH/RCL has been adjusted to 0x000 or 0x1FF 0: TrimCode of RCH/RCL is not 0x000 or 0x1FF	RO
b3	MISS	Count failure flag	1: The error counter has counted up to ARR during one cycle of GCLK 0: The error counter does not count up to ARR during one cycle of GCLK	RO
b2	PS	GCLK reference clock rising edge flag	1: GCLK reference clock rising edge detected 0: GCLK reference clock rising edge not detected	RO
b1	END	Auto Trim End Flag	1: Auto Trim is over 0: Auto Trim is in progress	RO
b0	UD	Error Counter Underflow Flag	1: The error counter has counted to 0 0: The error counter has not counted to 0	RO

9.5.8 Interrupt Status Clear Register (CTRIM_ICR)

Reset value: 0x0000 003F

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res									OK	OV	MISS	PS	END	UD	

Bit	Marking	Place name	Function	Read and write
b31~b6	Res	Reserved bit	Reserved, must remain at reset value	
b5	OK	Clear the OK flag	Write 0 to clear the calibration success OK flag	R1W0
b4	OV	Clear the OV flag bit	Write 0 to clear the TrimCode overflow flag	R1W0
b3	MISS	Clear the MISS flag bit	Write 0 to clear count failure flag	R1W0
b2	PS	Clear the PS flag bit	Write 0 to clear the GCLK reference clock rising edge flag	R1W0
b1	END	Clear the END flag bit	Write 0 to clear auto- trim end flag	R1W0
b0	UD	Clear the UD flag bit	Write 0 to clear the error counter underflow flag	R1W0

9.5.9 Error Result Register (CTRIM_FCAP)

Reset value: 0x0000 FFFF

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
FCAP															
Bit	Marking	Place name	Function												Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value												
			Frequency error capture value												
b15~b0	FCAP	Frequency error capture value	Note: 1) At the end of each GCLK cycle, the count value of the automatic capture error counter is stored in the FCAP register												RO

9.5.10 TRIM result register (CTRIM_TVAL)

Reset value: 0x0000 0100

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
TVAL															
Bit	Marking	Place name	Function												Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value												
			Auto TRIM results												
b15~b0	TVAL	Auto TRIM results	Note: 1) After the end of automatic TRIM, the value of TVAL needs to be written into the corresponding TRIM register 2) If the software reads during the TRIM process, it needs to be read twice to be the same valid												RO

9.5.11 Error Upper Limit Register (CTRIM_FLIM)

Reset value: 0x0000 0010

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res								FLIM							

Bit	Marking	Place name	Function	Read and write
b31~b8	Res	Reserved bit	Reserved, must remain at reset value	
b7~b0	FLIM	Error tolerance	Error tolerance	RW

10 FLASH Controller (FLASH)

10.1 Overview

This system includes a FLASH memory with a capacity of 128K bytes (Byte), which is divided into 256 pages (Sector) in total, and the capacity of each page (Sector) is 512 bytes (Byte).

This controller supports erasing, programming and reading operations on FLASH memory. The read and write bit widths supported by this controller are half-word (16 bits) and word (32 bits). **Note:** The target address of the half-word operation must be half-word aligned (the lowest bit of the address is 0), and the address of the word operation must be word-aligned (the lowest two bits of the address are both 0). If the target address is not aligned according to the bit width, the operation is invalid and the CPU will enter the Hard Fault interrupt.

10.2 Capacity division

FLASH capacity supported by this controller is 128KB, and the capacity of each page (Sector) is 512 bytes (Byte), and its composition is shown below.

Table 10-1 FLASH capacity division

Address	Serial number	Address	Serial number
0x00000 - 0x001FF	Sector0		
0x00200 - 0x003FF	Sector1	0x1F000 - 0x1F1FF	Sector248
0x00400 - 0x005FF	Sector2	0x1F200 - 0x1F3FF	Sector249
0x00600 - 0x007FF	Sector3	0x1F400 - 0x1F5FF	Sector250
0x00800 - 0x009FF	Sector4	0x1F600 - 0x1F7FF	Sector251
0x00A00 - 0x00BFF	Sector5	0x1F800 - 0x1F9FF	Sector252
0x00C00 - 0x00DFF	Sector6	0x1FA00 - 0x1FBFF	Sector253
0x00E00 - 0x00FFF	Sector7	0x1FC00 - 0x1FDFF	Sector254
.....		0x1FE00 - 0x1FFFF	Sector255

10.3 Read wait period

The fastest instruction fetch frequency supported by the built-in FLASH of this device is 24MHz, so when the CPU frequency (HCLK) is greater than 24MHz, the CPU must insert a wait cycle (set FLASH_WAIT greater than 0) to fetch instructions from the FLASH. When FLASH_WAIT is equal to 1, every two HCLKs can complete a FLASH fetch operation.

10.4 FLASH operation (read, write, erase)

10.4.1 Page Erase (Sector Erase)

Sector) specified by the user at a time. After the erase operation is completed, the data in the page (Sector) are all 0xFF. If the erase operation is performed from FLASH, the CPU will stop fetching instructions, and the hardware will automatically wait for the operation to complete (FLASH_IFR.BUSY becomes 0); if the erase operation is performed from RAM, the CPU will not stop fetching means, user software should wait for the operation to complete (FLASH_IFR.BUSY becomes 0).

Sector erase operation steps are as follows:

- Step1. Write 0x5A5A and 0xA5A5 to the FLASH_BYPASS register in sequence to enable register rewriting;
- Step2. Configure FLASH_CR.OP to 2, and set the Flash operation mode to Sector Erase;
- Step3. Check if FLASH_CR.OP is 2, if not, jump to Step1;
- Step4. Write 0x5A5A and 0xA5A5 to the FLASH_BYPASS register in sequence to enable register rewriting;
- Step5. Set the corresponding bit of FLASH_SLOCKx to 1 to remove the erase protection of the Sector;
- Step6. Check whether the corresponding bit of FLASH_SLOCKx is 1, if not 1, jump to Step4;
- Step7. Write arbitrary data to any address in the Sector to be erased, triggering Sector erasure;

Example: *((unsigned int *)0x0000 0200) = 0x0000 0000.
- Step8. Wait for FLASH_IFR.BUSY to become 0, and the Sector erase operation is completed;
- Step9. To erase other sectors, repeat Step4 – Step8. If the erasure is completed, jump to Step10;
- Step10. Write 0x5A5A and 0xA5A5 successively to the FLASH_BYPASS register to enable register rewriting;
- Step11. Configure FLASH_CR.OP to 0 and set the Flash operation mode to read;
- Step12. Check whether FLASH_CR.OP is 0. If it is not 0, jump to Step10; If it is 0, end the operation.

10.4.2 Full Chip Erase (Chip Erase)

Full chip erase can erase all pages (Sector) at one time. After the erase operation is completed, the data in all pages (Sector) are 0xFF. If the erase operation is performed from FLASH, the operation will be prohibited and the corresponding error flag will be set; if the erase operation is performed from RAM, the CPU will not stop fetching instructions, and the user software should wait The operation completes (FLASH_IFR.BUSY becomes 0).

Chip erase operation steps are as follows:

- Step1. Write 0x5A5A and 0xA5A5 to the FLASH_BYPASS register in sequence to enable register rewriting;
- Step2. Configure FLASH_CR.OP to 3, set the Flash operation mode to Chip erase;
- Step3. Check whether FLASH_CR.OP is 3, if not, jump to Step1;
- Step4. Write 0x5A5A and 0xA5A5 to the FLASH_BYPASS register in sequence to enable register rewriting;
- Step5. Set FLASH_SLOCKx to 0xFFFF FFFF to remove the erase protection of all Sectors;
- Step6. Check whether FLASH_SLOCKx is 0xFFFF FFFF, if not 0xFFFF FFFF, jump to Step4;
- Step7. Write any address in the Chip to be erased to trigger Chip erasure;
Example: $* ((\text{unsigned int } *) 0x0000\ 0000) = 0x00$.
- Step8. Wait for FLASH_IFR.BUSY to become 0, and the Chip erase operation is completed;
- Step9. Write 0x5A5A and 0xA5A5 successively to the FLASH_BYPASS register to enable register rewriting;
- Step10. Configure FLASH_CR.OP to 0 and set the Flash operation mode to read;
- Step11. Check whether FLASH_CR.OP is 0. If it is not 0, jump to Step10; If it is 0, end the operation.

10.4.3 Write operation (Program)

The write operation can only change the bit data in FLASH from 1 to 0, so before writing data, make sure that the data in the address to be written is 0xFF. Support writing 2 kinds of data length: Half-word(16bits), Word(32bits), the written data is stored in FLASH in little endian mode, that is, the low byte of the data is stored in the low address. If the write operation is executed from FLASH, the CPU will stop fetching instructions, and the hardware will automatically wait for the operation to complete (FLASH_IFR. BUSY becomes 0); if the write operation is executed from RAM, the CPU will not stop fetching instructions, and the user Software should wait for the operation to complete (FLASH_IFR.BUSY goes to 0).

Half-word write operation steps are as follows:

- Step1. Write 0x5A5A and 0xA5A5 to the FLASH_BYPASS register in sequence to enable register rewriting;
- Step2. Configure FLASH_CR.OP to 1, set the Flash operation mode to write;
- Step3. Check if FLASH_CR.OP is 1, if it is not 1, jump to Step1;
- Step4. Write 0x5A5A and 0xA5A5 to the FLASH_BYPASS register in sequence to enable register rewriting;
- Step5. Set the corresponding bit of FLASH_SLOCKx to 1 to remove the erase protection;
- Step6. Check whether the corresponding bit of FLASH_SLOCKx is 1, if not 1, jump to Step4;
- Step7. Perform a Half-word write operation on the target address to be written to trigger the write operation;

Example: * ((unsigned short int *) 0x0000 1232) = 0xABCD.
- Step8. Wait for FLASH_IFR.BUSY to become 0, and the write operation is completed;
- Step9. To write Half-word to other addresses, repeat Step7 – Step8. If the writing is completed, jump to Step10;
- Step10. Write 0x5A5A and 0xA5A5 successively to the FLASH_BYPASS register to enable register rewriting;
- Step11. Configure FLASH_CR.OP to 0 and set the Flash operation mode to read;
- Step12. Check whether FLASH_CR.OP is 0. If it is not 0, jump to Step10; If it is 0, end the operation.

Word write operation steps are as follows:

- Step1. Write 0x5A5A and 0xA5A5 to the FLASH_BYPASS register in sequence to enable register rewriting;
- Step2. Configure FLASH_CR.OP to 1, set the Flash operation mode to write;
- Step3. Check if FLASH_CR.OP is 1, if it is not 1, jump to Step1;
- Step4. Write 0x5A5A and 0xA5A5 to the FLASH_BYPASS register in sequence to enable register rewriting;
- Step5. Set the corresponding bit of FLASH_SLOCK to 1 to remove the erase protection;
- Step6. Check whether the corresponding bit of FLASH_SLOCK is 1, if it is not 1, jump to Step4;
- Step7. Perform a Word write operation on the target address to be written to trigger the write operation;

Example: $*((\text{unsigned int } *) 0x0000\ 1234) = 0x5566\ 7788$.

- Step8. Wait for FLASH_IFR.BUSY to become 0, and the write operation is completed;
- Step9. If you need to write Word to other addresses, repeat Step7 - Step8. If the writing is completed, jump to Step10;
- Step10. Write 0x5A5A and 0xA5A5 successively to the FLASH_BYPASS register to enable register rewriting;
- Step11. Configure FLASH_CR.OP to 0 and set the Flash operation mode to read;
- Step12. Check whether FLASH_CR.OP is 0. If it is not 0, jump to Step10; If it is 0, end the operation.

10.4.4 Read operation (Read)

Support to read 3 kinds of data length: Byte(8bits), Half-word(16bits), Word(32bits), the read data is little-endian mode, that is, the low byte of the data is stored in the low address. the data in the FLASH can be read at any time.

example:

Byte read operation: $\text{temp} = *((\text{unsigned char } *)0x0000\ 1231)$

Half-word read operation $\text{temp} = *((\text{unsigned short int } *)0x0000\ 1232)$

Word read operation $\text{temp} = *((\text{unsigned int } *)0x0000\ 1234)$

10.4.5 Cache function

Support the cache function of data and instructions. If the system operating frequency is greater than 24M and the wait cycle FLASH_WAIT is greater than 0, enabling the cache instruction and data cache function will improve system performance.

Note: For FLASH erase and write operations, the cache must be turned off.

10.5 FLASH security protection

10.5.1 Page Erase Protection

FLASH memory has a total of 128K bytes, which is divided into 256 pages, and every 4 pages share an erase and write protection bit. When the page is protected, the erasing and writing operations on the page are invalid and an alarm flag and interrupt signal are generated. When any page in the FLASH memory is protected, the whole-chip erasing of the FLASH is invalid, and an alarm flag and an interrupt signal are generated.

10.5.2 PC address erase and write protection

If the CPU executes the program in FLASH and the current PC pointer just falls within the address range of the page to be erased, then the erase operation is invalid and an alarm flag and an interrupt signal are generated. Based on this feature, only when the code is executed from RAM, can chip erase operation be performed on FLASH.

10.5.3 Register write protection

The important controller of this module shields ordinary write operations, and must be modified by writing sequence.

Registers that require a write sequence to be changed are as follows:

FLASH_CR, FLASH_SLOCKx, FLASH_WAIT, FLASH_CACHE.

Registers that can be changed without a write sequence are as follows:

FLASH_ICLR, FLASH_BYPASS.

The specific operation steps to modify the register value by writing sequence are as follows:

- Step1. Write 0x5A5A to the FLASH_BYPASS register;
- Step2. Write 0xA5A5 to the FLASH_BYPASS register;
- Step3. Write the target value to the register to be modified;
- Step4. Verify whether the current value of the register to be modified is the same as the target, if not, jump to Step1;
- Step5. Perform other operations.

Note:

- Do not insert any write operation between the two steps of writing 0x5A5A and writing 0xA5A5, otherwise the value of the target register cannot be rewritten. If rewriting fails, you need to perform these three steps again.

10.5.4 Data Readout Protection

The chip's built-in FLASH supports 4 -level read protection, and the current protection level can be read through the FLASH_LOCKSTATE register. When programming the chip with a programmer, the read protection level of FLASH can be configured.

The functions of the chip under each protection level are shown in the table below.

Table 10-2 Chip Protection Level Table

Protection level	Chip function
Level0	FLASH can be read and written through ISP and SWD
Level1	Data cannot be read be downgraded to Level0 through ISP and SWD. After downgrading, the FLASH content is all FF
Level2	Data cannot be read be downgraded to Level0 through ISP. After downgrading, the FLASH content is all FF The SWD interface is disabled and the data cannot be read out
Level3	Data cannot be read the ISP and SWD interfaces are prohibited, that is, the chip can only be programmed once

10.6 Register description

10.6.1 Register list

Base address: 0x4002 1400

Table 10-3 FLASH register list

Register name	Offset address	Reset value
Control Register(FLASH_CR)	0x00	0x0080 (high 16 bits reserved)
Wait period setting(FLASH_WAIT)	0x04	0x0000 (upper 16 bits reserved)
LOCK status register(FLASH_LOCKSTATE)	0x08	0x0000 (upper 16 bits reserved)
CACHE control register(FLASH_CACHE)	0x0C	0x0000 (upper 16 bits reserved)
Interrupt Enable Register(FLASH_IER)	0x10	0x0000 (upper 16 bits reserved)
Interrupt Flag Register(FLASH_IFR)	0x14	0x000F (upper 16 bits reserved)
Interrupt Flag Clear Register(FLASH_ICR)	0x18	0x0000 (upper 16 bits reserved)
Bypass sequence register(FLASH_BYPASS)	0x1C	0x0000 (upper 16 bits reserved)
Sector Erase and Write Protection Register(FLASH_SLOCK0)	0x30	0x0000 0000
Sector Erase and Write Protection Register(FLASH_SLOCK1)	0x34	0x0000 0000

10.6.2 Control Register (FLASH_CR)

Reset value: 0x0080

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res								DPSTB_EN	Res					OP	
Bit	Marking		Place name			Function								Read and write	
b15~b8	Res		Reserved bit			Reserved, must remain at reset value									
b7	DPSTB_EN		FLASH dpstb enable Mask bit			0: When the system enters deepsleep mode, FLASH does not enter low power consumption mode; 1: When the system enters deepsleep mode, FLASH enters low power consumption mode;								RW	
b6~b2	Res		Reserved bit			Reserved, must remain at reset value									
b1~b0	OP		FLASH operation			00: read(read); 01: write(program); 10: page erase(sector erase); 11: full chip erase(chip erase) Note: After the software performs an erase or write operation on the Flash, the control bit of FLASH CR.OP needs to be reset to 0.								RW	

10.6.3 Wait period setting (FLASH_WAIT)

Reset value: 0x0000

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res												WAIT			
Bit	Marking	Place name	Function												Read and write
b15~b4	Res	Reserved bit	Reserved, must remain at reset value												
b3~b0	WAIT	Read number	FLASH cycle	0~24MHz: 0000, 1 cycle; 24~48MHz: 0001: 2 cycles; 48~72MHz: 0010: 3 cycles; 72~96MHz: 0011: 4 cycles; It is forbidden to set other values (The highest clock of this series of products is 84MHz)											RW

10.6.4 LOCK status register (FLASH_LOCKSTATE)

Reset value: 0x0000

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res														STATE	
Bit	Marking	Place name	Function												Read and write
b15~b2	Res	Reserved bit	Reserved, must remain at reset value												
b1~b0	STATE	FLASH protection status	00: Level0, ISP can read and write, SWD can read and write 01: Level1, ISP can be downgraded, SWD can be downgraded; data cannot be read out 10: Level2, ISP can be downgraded, SWD has no function; data cannot be read out 11: Level3, ISP has no function, SWD has no function; data cannot be read out												RO

10.6.5 CACHE control register (FLASH_CACHE)

Reset value: 0x0000

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res													INVALID	DEN	IEN
Bit	Marking	Place name	Function											Read and write	
b15~b3	Res	Reserved bit	Reserved, must remain at reset value												
b2	INVALID	cache reset	Write 1 first, then write 0 to realize cache reset											RW	
b1	DEN	data cache enable	1: enable data cache 0: close the data cache											RW	
b0	IEN	instruction cache enable	1: enable instruction cache 0: Close instruction cache											RW	

10.6.6 Interrupt Enable Register (FLASH_IER)

Reset value: 0x0000

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res														PROT	PC
Bit	Marking	Place name	Function											Read and write	
b15~b2	Res	Reserved bit	Reserved, must remain at reset value												
b1	PROT	FLASH erasing protection alarm interrupt enable	1: enable FLASH erasing protection alarm interrupt 0: disable FLASH erasing protection alarm interrupt											RW	
b0	PC	FLASH erase PC address alarm interrupt enable	1: enable FLASH erase PC address alarm interrupt 0: Disable FLASH erasing and writing PC address alarm interrupt											RW	

10.6.7 Interrupt Flag Register (FLASH_IFR)

Reset value: 0x0000

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res											BUSY	Res		PROT	PC
Bit	Marking	Place name					Function							Read and write	
b15~b5	Res	Reserved bit					Reserved, must remain at reset value								
b4	BUSY	FLASH bus busy flag					1: The FLASH bus is currently busy 0: FLASH bus is currently idle							RO	
b3~b2	Res	Reserved bit					Reserved, must remain at reset value								
b1	PROT	FLASH erasing protection interrupt flag bit		alarm		1: FLASH erasing protection alarm interrupt occurs 0: FLASH erasing protection alarm interrupt does not occur							RO		
b0	PC	FLASH erase PC address alarm interrupt flag bit		alarm		1: FLASH erasing and writing PC address alarm interrupt 0: No FLASH erasing and writing PC address alarm interrupt							RO		

10.6.8 Interrupt Flag Clear Register (FLASH_ICR)

Reset value: 0x000F

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res														PROT	PC

Bit	Marking	Place name	Function	Read and write
b15~b2	Res	Reserved bit	Reserved, must remain at reset value	
b1	PROT	Clear protection alarm interrupt flag bit	1:Invalid 0: clear	R1W0
b0	PC	Clear the PC address alarm interrupt flag bit	1:Invalid 0: clear	R1W0

10.6.9 Bypass sequence register (FLASH_BYPASS)

Reset value: 0x0000

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
BYSEQ															

Bit	Marking	Place name	Function	Read and write
b15~b0	BYSEQ	Bypass sequence bit	Before modifying the registers of this module, the 0x5a5a-0xa5a5 sequence must be written to the BYSEQ[15:0] register. each write to the Bypass sequence, the register can only be modified once. If you need to modify the register again, you must enter the 0x5a5a-0xa5a5 sequence again.	RW

10.6.10 Sector Erase and Write Protection Register (FLASH_SLOCK0)

Reset value: 0x00000000 (Peripheral reset is invalid)

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
SLOCK[31:16]															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
SLOCK[15:0]															

Bit	Marking	Place name	Function	Read and write
b31~b0	SLOCK	Sector protection bit	0: Erase and write not allowed; 1: Erase and write allowed Bit[0] corresponds to: Sector0~3 Bit [1] corresponds to: Sector4~7 Bit [2] corresponds to: Sector8~11 Bit [3] corresponds to: Sector12~15 Bit [30] corresponds to: Sector120~123 Bit [31] corresponds to: Sector124~127	RW

10.6.11 Sector Erase and Write Protection Register (FLASH_SLOCK1)

Reset value: 0x00000000 (Peripheral reset is invalid)

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
SLOCK[31:16]															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
SLOCK[15:0]															

Bit	Marking	Place name	Function	Read and write
b31~b0	SLOCK	Sector protection bit	0: Erase and write not allowed; 1: Erase and write allowed Bit[0] corresponds to: Sector128~131 Bit[1] corresponds to: Sector132~135 Bit[2] corresponds to: Sector136~139 Bit[30] corresponds to: Sector248~251 Bit[31] corresponds to: Sector252~255	RW

11 RAM Controller (RAM)

11.1 Overview

This product contains a piece of SRAM with a starting address of 0x0x2000_0000 and a capacity of up to 24K bytes (Byte), which supports three read and write operations of byte (8 bits), half word (16 bits), and word (32 bits), zero wait cycle.

11.2 Functional description

11.2.1 RAM address range

RAM in the system map is shown in the table below:

Table 11-1 RAM Address Mapping

Address range	Size	Memory type
0x2000 0000 - 0x2000 1FFF	8KByte	SRAM
0x2000 0000 - 0x2000 3FFF	16KByte	
0x2000 0000 - 0x2000 5FFF	24KByte	

12 DMA Controller (DMAC)

12.1 Introduction to DMAC

Direct memory access (DMA) is used to provide high-speed data transmission between peripherals and memory or between memory and memory. The CPU does not need to participate in the transmission process, so the CPU can perform other operations synchronously. DMAC has 6 independent DMA channels, each channel is used to manage requests from peripheral hardware or memory access, and an arbiter is used to coordinate the priority of each DMA request.

12.2 DMAC main characteristics

- 6 independent DMA channels, support priority configuration
- 3 data transfer widths: 8-bit, 16-bit, 32-bit
- 4 transmission modes: Software Block, Software Burst, Hardware Block, Hardware Burst
- 2 source address types: peripheral, memory
- 2 target address types: peripheral, memory
- 2 address change modes: fixed, auto-increment
- Transmission address addressing range: 0x0000 0000 ~ 0xFFFF FFFF
- The number of data blocks to be transmitted is configurable: 1~65536
- The size of the data block to be transmitted is configurable: 1~16
- Support address and transfer number reload function

12.3 DMA functional block diagram

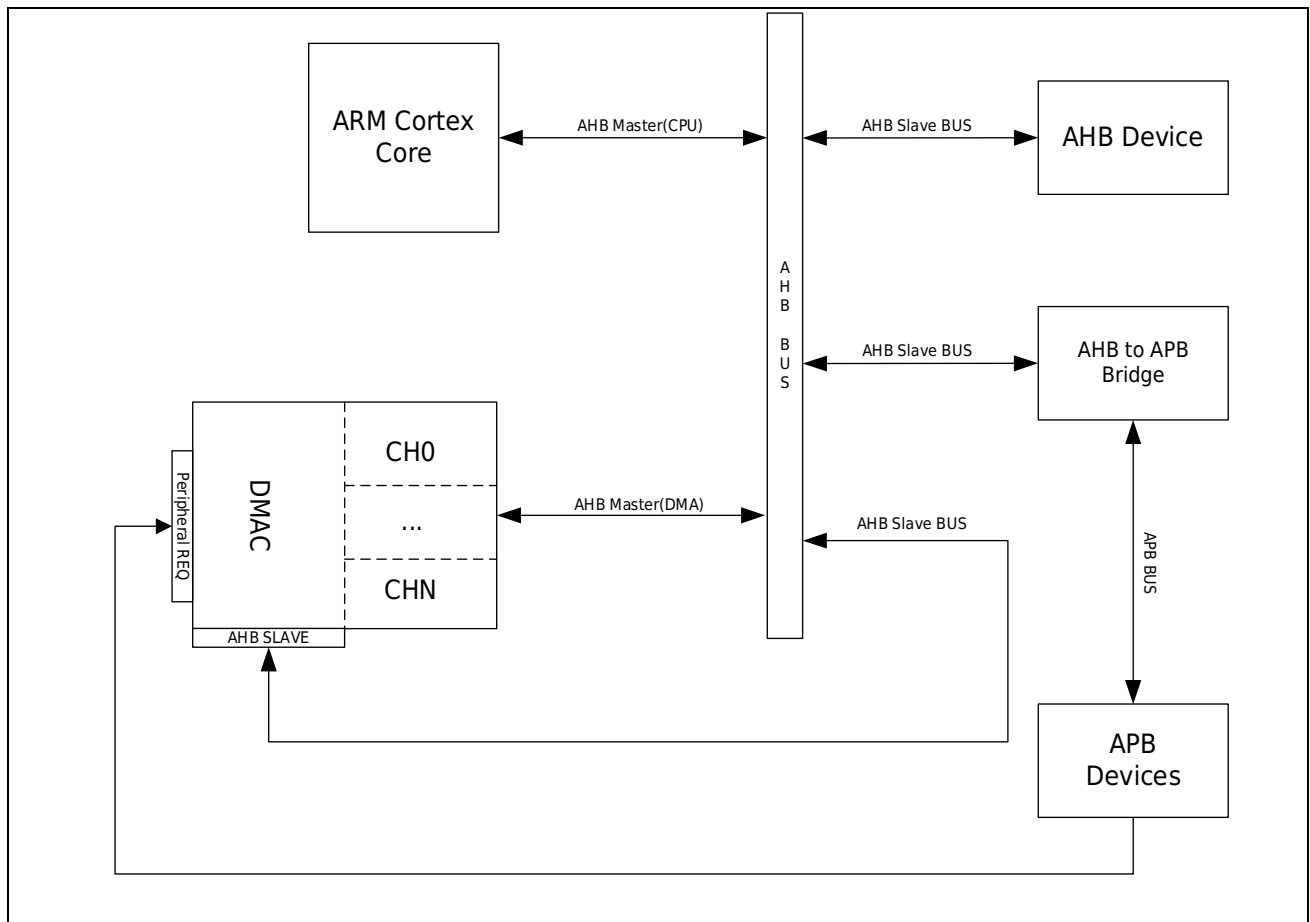


Figure 12-1 DMA functional block diagram

- **DMAC**

DMAC has 6 DMA channels, and when conflicts occur between channels, arbitration is performed by the priority controller.

- **DMA requests**

If the peripheral supports hardware DMA request, the DMA channel can be configured as a hardware trigger; otherwise, it can only be configured as a software trigger.

12.4 Functional description

12.4.1 DMAC transfer modes

The DMA controller supports four transfer modes: software Block transfer mode, software Burst transfer mode, hardware Block transfer mode, and hardware Burst transfer mode.

The 4 transmission modes is shown in the table below:

Table 12-1 DMA transfer mode table

Compare Items	Software Block Transmission	Software Burst Transmission	Hardware Block Transmission	Hardware Burst Transmission
Interrupted by higher priority DMA channel or CPU	Can	Can't	Can	Can't
Conditions that trigger the start of the transfer	Write DMAC Register		Peripheral interrupt flag	
The amount of data that triggers a transfer	$(BC+1) * (TC+1)$		BC+1	$(BC+1) * (TC+1)$
The total amount of data transferred after configuration	$(BC+1) * (TC+1)$			
Main application scenarios	- Memory to memory - Memory to peripherals without DMA requests		- Memory to peripherals with DMA requests - With DMA requests to memory	- Memory to memory - Memory to peripherals without DMA requests

12.4.2 DMA software block transfer mode

When configuring DMAC_CONFAx.TRI_SEL=0 and DMAC_CONFBx.MODE=0, DMAC works in software Block transfer mode.

In this mode, the user writes 1 to DMAC_CONFAx.ST, which triggers the DMAC to start the software block transmission mode, and stops after transmitting $(BC+1) * (TC+1)$ data. each(BC+1) data transmission is completed, the DMAC will insert a transmission gap, and the priority arbiter performs priority arbitration in the gap. The CPU or a higher priority DMA channel requests the bus during this gap, the bus ownership is transferred to the CPU or a higher priority DMA channel; when the CPU or a higher priority DMA channel releases the bus, the unfinished data transfer will continue to be completed.

The software Block transmission diagram is shown below, where SA represents the source address and DA represents the target address; The size of the data block is 3 (BC=2), the number of data blocks is 3 (TC=2), the data width is 16bit, and the address increases automatically.

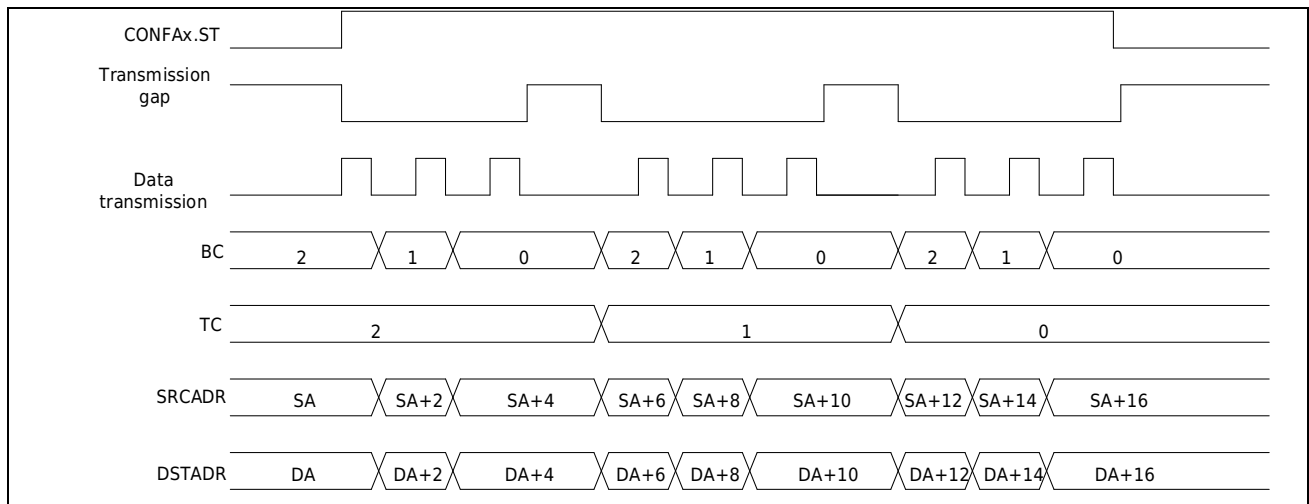


Figure 12-2 DMA software block transfer

12.4.3 DMA software Burst transfer mode

When configuring DMAC_CONFAX.TRI_SEL=0 and DMAC_CONFBx.MODE=1, DMAC works in software Burst transfer mode.

In this mode, the user writes 1 to DMAC_CONFAX.ST, which triggers the DMAC to start the software Burst transmission mode, and stops after transmitting $(BC+1) * (TC+1)$ data. There is no transmission gap in this transfer. Before all data transfers are completed, the currently working DMA channel will always occupy the bus. The CPU or a higher priority DMA channel can only access the bus after the operation is completed. The CPU may not work completely for a period of time.

The software Burst transmission diagram is as follows, where SA represents the source address, DA represents the destination address; the data block size is 3 (BC=2), the number of data blocks is 3 (TC=2), the data width is 16bit, and the address is self-incrementing.

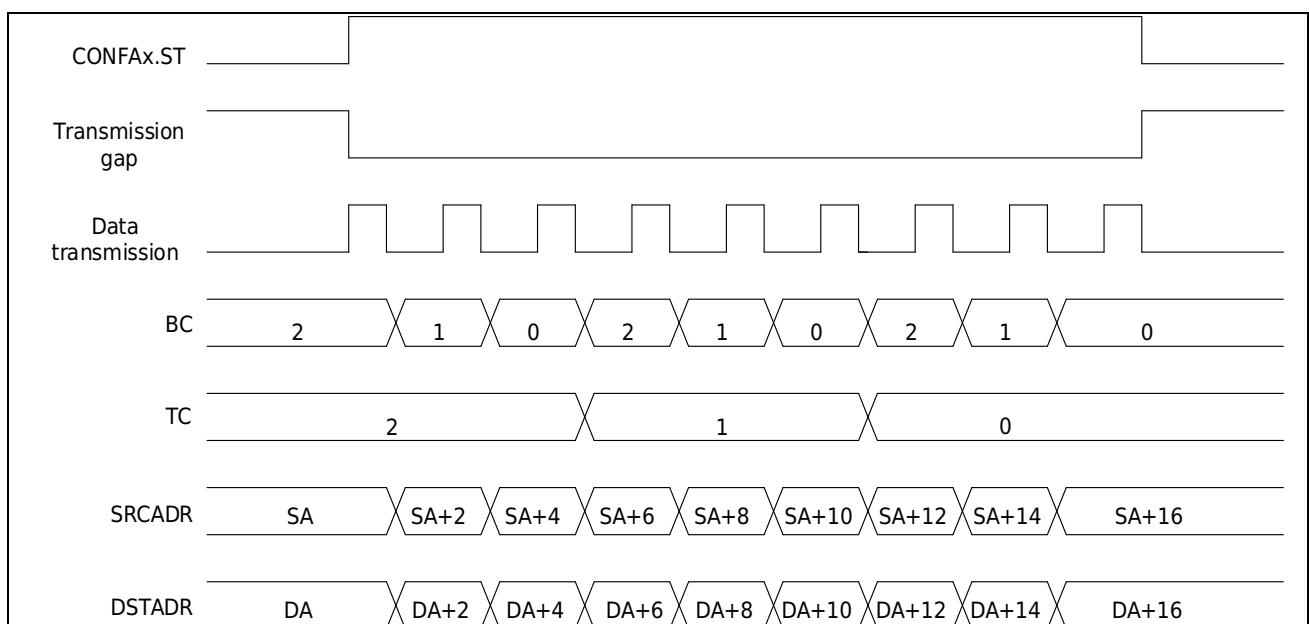


Figure 12-3 DMA software Burst transfer

12.4.4 DMA hardware block transfer mode

When configuring DMAC_CONF_x.TRI_SEL as the corresponding trigger source and DMAC_CONF_x.MODE=0, the DMAC works in the hardware Block transfer mode.

In this mode, whenever the peripheral DMA request signal appears once, the DMAC is triggered to start a hardware block transfer, and (BC+1) data is transferred. It needs (TC+1) peripheral DMA request signals to complete all DMA data transmission. Each (BC+1) data is completed, the DMAC will insert a transmission gap, and the priority arbiter performs priority arbitration in the gap. The CPU or a higher priority DMA channel is also requesting the bus during this gap, the bus ownership is transferred to the CPU or a higher priority DMA channel; when the CPU or a higher priority DMA channel releases the bus, the unfinished data transfer will continue. Finish.

Note: In this mode, generally set BC to 0 to obtain a data from the peripheral or provide a data to the peripheral.

Request signals supported by DMA are as follows:

- UART_x receiving Buf is not empty, sending Buf is empty
- SPI_x receiving Buf is not empty, sending Buf is empty
- GTIM_x capture complete, comparison match
- ATIM_x capture complete, compare match
- ADC_x sequential single conversion complete

The hardware block transmission diagram is as follows, where SA represents the source address, DA represents the target address; the data block size is 2 (BC=1), the number of data blocks is 3 (TC=2), the data width is 8bit, and the target address is self-incrementing, the source address is fixed, and the transmission is automatically disabled after completion.

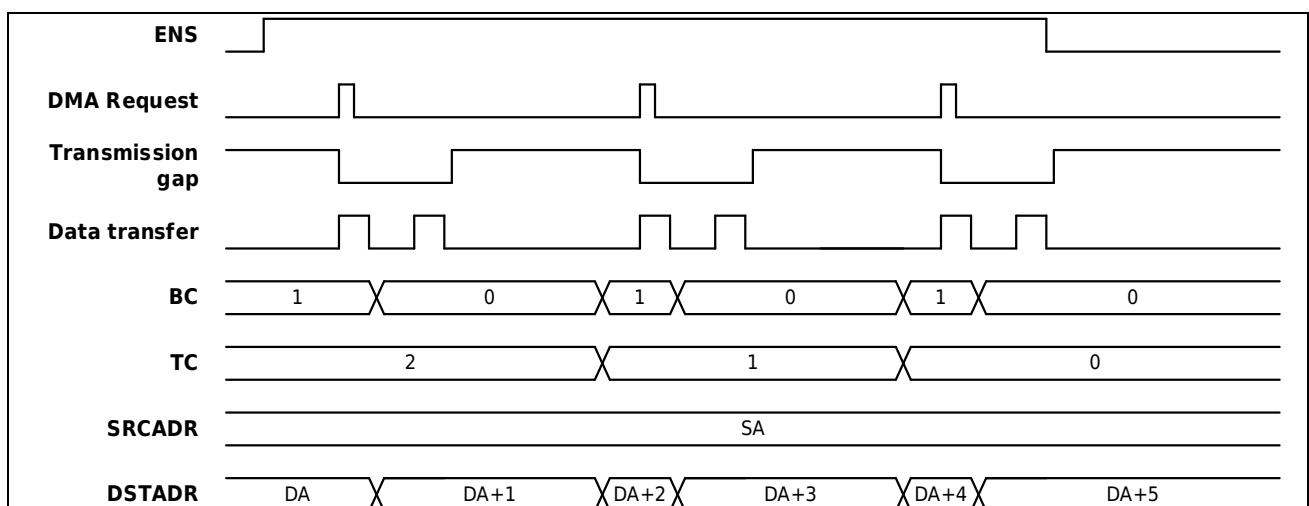


Figure 12-4 DMA hardware block transfer

12.4.5 DMA hardware Burst transfer mode

When configuring `DMAC_CONFx.TRI_SEL= non-zero` and `DMAC_CONFBx.MODE=1`, DMAC works in hardware Burst transfer mode.

In this mode, once the peripheral DMA request signal appears, the DMAC will be triggered to start the hardware Burst transmission mode, and the transmission will stop after completing $(BC+1) * (TC+1)$ data. There is no transmission gap in this transfer. Before all data transfers are completed, the currently working DMA channel will always occupy the bus. The CPU or a higher priority DMA channel can only access the bus after the operation is completed. The CPU may not work completely for a period of time.

Request signals supported by DMAC are as follows:

- UARTx receiving Buf is not empty, sending Buf is empty
- SPIx receiving Buf is not empty, sending Buf is empty
- GTIMx capture complete, comparison match
- ATIMx capture complete, compare match
- ADC sequential single conversion complete

The hardware Burst transmission diagram is as follows, where SA represents the source address, DA represents the target address; the data block size is 3 ($BC=2$), the number of data blocks is 3 ($TC=2$), the data width is 8bit, and the source address is auto-incremented, the target address is fixed, and the transfer is automatically disabled after completion.

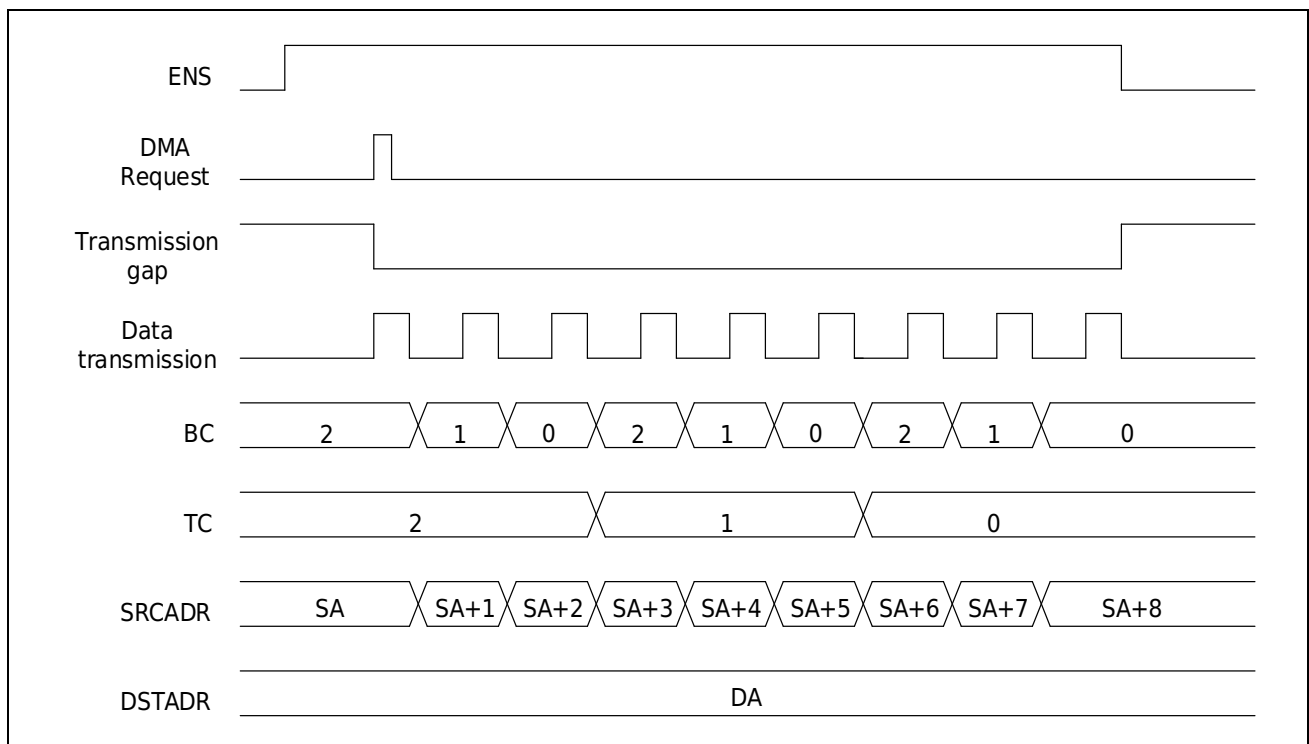


Figure 12-5 DMA hardware Burst transfer

12.4.6 DMA transfer paused

When DMAC works in Block transfer mode, if DMAC_CONF.HALT is set to a non-zero value, all channels that are being transferred will enter the pause state during the transfer interval, and no more data transfer will be performed. When DMAC_CONFx.PAS is configured as 1, the channel corresponding to x that is being transmitted will enter a pause state during the transmission interval, and no data transmission will be performed. DMAC_CONFx.PAS is configured as 1, the channel corresponding to x that is being transmitted will enter a pause state during the transmission interval and no longer transmit data. configuring DMAC_CONFx.PAS to 0, DMAC needs to receive the next trigger signal before continuing the unfinished transmission.

When the DMAC works in the Burst transfer mode and no data transfer is performed, configure DMAC_CONF.HALT to a non-zero value, and the DMA channel enters the pause state and no longer performs data transfer. After configuring DMAC_CONF.HALT to 0, DMAC will start a new complete transmission when it receives the next trigger signal. When the DMAC works in the Burst transfer mode and no data transfer is performed, when DMAC_CONFx.PAS is set to 1, the DMA channel enters the pause state and no data transfer is performed. After configuring DMAC_CONFx.PAS to 0, DMAC will start a new complete transmission when it receives the next trigger signal.

When the DMAC is working in the Burst transfer mode and data transfer is in progress, writing a non-zero value to DMAC_CONF.HALT or writing 1 to DMAC_CONFx.PAS cannot suspend the ongoing transfer, and the DMA channel currently in data transfer is in the data. After the transmission is completed, it will not enter the pause state.

12.4.7 DMA other configuration

12.4.7.1 Data width

Configure DMAC_CONFBx.WIDTH, you can configure the width of each data to be transmitted as 8bit, 16bit, 32bit, note that the bit width should be completely consistent with the bit width of the DMA source address and target address.

12.4.7.2 Block size

Configure DMAC_CONFx.BC, you can configure the number of data to be transmitted in each data block as 1~16. Note that the value of BC should match the number of consecutive operations provided by the source and destination addresses.

If the source address or destination address can only provide or receive one data at a time, then BC can only be configured as 0. For example, UART/SPI can only provide one data to DMA at a time, and UART/SPI can only receive one data from DMA at a time, so BC can only be configured as 0.

If both the source address and the destination address can provide or receive multiple data, BC can be configured to a larger value to speed up the transmission speed. For example, when transferring

from FLASH to RAM or from FLASH to CRC, the source address and target address can be operated continuously, so BC can be configured as 15 to achieve fast transfer.

12.4.7.3 Number of data blocks

Configure DMAC_CONFx.TC, you can configure the number of data blocks to be transferred from 1 to 65536 after each DMA is started.

12.4.7.4 channel priority

When DMAC_CONF.PRIO=0, DMAC works in fixed priority mode. At this time, the priority order is: CH0>CH1>CH2>CH3>CH4>CH5. Only when the high priority channel does not transmit data, the low priority channel can transmit data.

When DMAC_CONF.PRIO=1, DMAC works in cycle priority mode. At this time, CH0, CH1, CH2, CH3, CH4 and CH5 occupy the bus in turn for data transmission.

12.4.7.5 Auto reload

If the automatic reload function is enabled, the data block size, number of data blocks, source address, and destination address will be automatically loaded to the last configuration value after the transmission is completed, and there is no need to configure it every time.

12.4.8 DMA Interrupt

DMA supports two types of interrupts: transfer error and transfer complete. When an interrupt is triggered, the status register DMAC_CONFBx.STAT can be checked to determine what condition triggered the interrupt, and the built-in interrupt status flag can be cleared by clearing the register DMAC_CONFBx.STAT.

The five conditions that trigger the DMA interrupt are as follows:

- Transport address out of addressing range
- Peripheral request to stop DMA
- Error accessing the transfer source address
- Access transfer destination address error
- Transmission completion

12.5 Register description

12.5.1 Register list

Base address: 0x4002 0000

Table 12-2 DMA register list

Register name	Offset address	Reset value
All Channel Configuration Register(DMAC_CONF)	0x00	0x0000 0000
Channel 0 Configuration A Register(DMAC_CONFA0)	0x10	0x0000 0000
Channel 0 Configuration B Register(DMAC_CONFB0)	0x14	0x0000 0000
Channel 0 Transfer Source Address Register(DMAC_SRCADR0)	0x18	0x0000 0000
Channel 0 Transfer Destination Address Register(DMAC_DSTADR0)	0x1C	0x0000 0000
Channel 1 Configuration A Register(DMAC_CONFA1)	0x20	0x0000 0000
Channel 1 Configuration B Register(DMAC_CONFB1)	0x24	0x0000 0000
Channel 1 Transfer Source Address Register(DMAC_SRCADR1)	0x28	0x0000 0000
Channel 1 Transfer Destination Address Register(DMAC_DSTADR1)	0x2C	0x0000 0000
Channel 2 Configuration A Register(DMAC_CONFA2)	0x30	0x0000 0000
Channel 2 Configuration B Register(DMAC_CONFB2)	0x34	0x0000 0000
Channel 2 Transfer Source Address Register(DMAC_SRCADR2)	0x38	0x0000 0000
Channel 2 Transfer Destination Address Register(DMAC_DSTADR2)	0x3C	0x0000 0000
Channel 3 Configuration A Register(DMAC_CONFA3)	0x40	0x0000 0000
Channel 3 Configuration B Register(DMAC_CONFB3)	0x44	0x0000 0000
Channel 3 Transfer Source Address Register(DMAC_SRCADR3)	0x48	0x0000 0000
Channel 3 Transfer Destination Address Register(DMAC_DSTADR3)	0x4C	0x0000 0000
Channel 4 Configuration A Register(DMAC_CONFA4)	0x50	0x0000 0000
Channel 4 Configuration B Register(DMAC_CONFB4)	0x54	0x0000 0000
Channel 4 Transfer Source Address Register(DMAC_SRCADR4)	0x58	0x0000 0000
Channel 4 Transfer Destination Address Register(DMAC_DSTADR4)	0x5C	0x0000 0000
Channel 5 Configuration A Register(DMAC_CONFA5)	0x60	0x0000 0000
Channel 5 Configuration B Register(DMAC_CONFB5)	0x64	0x0000 0000
Channel 5 Transfer Source Address Register(DMAC_SRCADR5)	0x68	0x0000 0000
Channel 5 Transfer Destination Address Register(DMAC_DSTADR5)	0x6C	0x0000 0000

12.5.2 All Channel Configuration Register (DMAC_CONF)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
EN	Res	PRIO	HALT					Res							
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res															

Bit	Marking	Place name	Function	Read and write
b31	EN	DMA controller enable	1: Enable DMA controller 0: disable DMA controller Note: When the DMA controller is disabled, ongoing transfers will be forcibly stopped, causing unpredictable results.	RW
b30~b29	Res	Reserved bit	Reserved, must remain at reset value	
b28	PRIO	DMA channel priority configuration	DMA channel priority configuration 1: Cycle priority mode, CH0/CH1/CH2/CH3/CH4/CH5 take turns to obtain bus access 0: Fixed priority mode, CH0>CH1>CH2>CH3>CH4>CH5 priority	RW
b27~b24	HALT	DMA all channel transfer pause control	DMA all channel transfer pause control 0000: Resume all channel data transmission xxxx: suspend all channel data transmission	RW
b23~b0	Res	Reserved bit	Reserved, must remain at reset value	

12.5.3 Channel x Configuration A Register (DMAC_CONFAX)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
ENS	PAS	ST	TRI_SEL						Res		BC				
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
TC															

Bit	Marking	Place name	Function	Read and write
b31	ENS	DMA channel enable control	1: Enable the current DMA channel; if CONFBx.MSK is 0, this bit is automatically cleared when the transfer is completed 0: disable the current DMA channel Note: When a DMA channel is disabled, ongoing transfers will be forcibly stopped, resulting in unpredictable results.	RW
b30	PAS	DMA channel data transfer pause control	1: Suspend DMA channel data transfer 0: Resume DMA channel data transfer	RW
b29	ST	DMA channel software trigger control	1: Trigger DMA channel to start software Block/Burst transfer, this bit is automatically cleared when the transfer is completed 0: Stop DMA channel software Block/Burst transfer	RW
b28~b22	TRI_SEL	DMA channel trigger source configuration	0x00: software trigger 0x40: USART0 sends Buf empty 0x41: USART0 receiving Buf is not empty 0x42: USART1 sends Buf empty 0x43: USART1 receiving Buf is not empty 0x44: USART2 sends Buf empty 0x45: USART2 receive Buf is not empty 0x46: USART3 sends Buf empty 0x47: USART3 receiving Buf is not empty 0x48: GTIM0 capture event or compare event 0x49: GTIM1 capture event or compare event 0x4A: GTIM2 capture event or compare event 0x4B: Reserved 0x4C: Reserved 0x4D: ADC0 sequential single conversion completed 0x4E: Reserved 0x4F: ADC1 sequential single conversion completed 0x50: SPI0 sends Buf empty 0x51: SPI0 receiving Buf is not empty 0x52: SPI1 sends Buf empty 0x53: SPI1 receiving Buf is not empty 0x54: Reserved 0x55: Reserved 0x56: Reserved 0x57: Reserved 0x58: ATIM0 channel A, capture event or compare event or triggered externally occurs 0x59: ATIM0 channel B, capture event or compare event or event update occurs 0x5A: Reserved 0x5B: Reserved 0x5C: Reserved 0x5D: Reserved 0x5E: ATIM3 channel A, capture event or compare event or triggered externally occurs 0x5F: ATIM3 channel B, capture event or compare event or event update occurs 0x60~0x7F: Reserved	RW
b21~b20	Res	Reserved bit	Reserved, must remain at reset value	
b19~b16	BC	DMA channel transfer data block size	Configure the size of the data block to be transmitted by the DMA channel as BC+1 Note: When the source address or destination address can only provide or receive a data every once in a while, the BC value can only be set to 0	RW
b15~b0	TC	Number of data blocks transferred by DMA channel	Configure the number of data blocks to be transmitted by the DMA channel as TC+1 During transmission, every time a data block is transmitted, the value of this register will be decremented by 1 Note: The total amount of data transferred by DMA is (TC + 1)	RW

12.5.4 Channel x Configuration B Register (DMAC_CONFbX)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res	MODE	WIDTH	FS	FD	RC	RS	RD	ERR_I E	FIS_I E	STAT					
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res														MSK	

Bit	Marking	Place name	Function	Read and write
b31~b30	Res	Reserved bit	Reserved, must remain at reset value	
b29~b28	MODE	DMA channel transfer mode configuration	00: Block transmission 01: Burst transmission 10: Reserved 11: Reserved	RW
b27~b26	WIDTH	DMA transfer data bit width	00: 8bit 01: 16bit 10: 32bit 11: Reserved	RW
b25	FS	DMA channel source address auto-increment enable	1: The source address remains unchanged during transmission 0: The source address will be incremented after each data transmission is completed; The auto-increment corresponding to the data width of 8bit / 16bit / 32bit is 1 / 2 / 4 respectively	RW
b24	FD	DMA channel target address self-increment enable	1: Destination address remains unchanged during transfer 0: The destination address will be incremented after each data transmission is completed; The auto-increment corresponding to the data width of 8bit / 16bit / 32bit is 1 / 2 / 4 respectively	RW
b23	RC	DMA channel BC and TC reload enable	1: Enable BC/TC reload, and the value of BC/TC will automatically restore to the initial value written after the transmission is completed 0: Disable BC/TC reloading, and the values of BC/TC are all 0 after the transmission is completed	RW
b22	RS	DMA channel source address reload enable	1: Enable the source address to be reloaded, and the value of the source address is the initial value written after the transmission is completed 0: Disable source address overloading, the value of the source address is uncertain after the transmission is completed	RW
b21	RD	DMA channel target address reload enable	1: Enable target address reloading, the value of the target address will automatically restore to the initial value written after the transfer is completed 0: Disable destination address overloading, the value of the destination address is uncertain after the transfer is completed	RW
b20	ERR_IE	DMA channel transfer error interrupt enable	1: When a transmission error occurs, an interrupt is generated 0: disable interrupt when transmission error Note: In the interrupt service routine, you need to write 0x00 to STAT to clear the interrupt status	RW
b19	FIS_IE	DMA channel transfer complete interrupt enable	1: Generate an interrupt when the transfer is complete 0: When the transfer is complete, disable the interrupt Note: In the interrupt service routine, you need to write 0x00 to STAT to clear the interrupt status	RW
b18~b16	STAT	DMA channel current transfer status	000: initial value 001: Transmission error, the transmission address exceeds the addressing range 010: Transmission error, peripheral request to stop DMA 011: Transmission error, error in accessing transmission source address 100: transmission error, access transmission destination address error 101: Transfer complete 110: reserved 111: Transmission paused	RW
b15~b1	Res	Reserved bit	Reserved, must remain at reset value	
b0	MSK	Auto-disable configuration when DMA channel transfer completes	1: CONFAX.ENS remains unchanged when DMA channel transfer is complete 0: CONFAX.ENS is automatically cleared when the DMA channel transfer is complete	RW

12.5.5 Channel x Transfer Source Address Register (DMAC_SRCADR_x)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
SRCADR[31:16]															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
SRCADR[15:0]															
Bit	Marking	Place name	Function										Read and write		
b31~b0	SRCADR	DMA channel source address configuration	Its auto-reload can be configured via CONFBx.RS Note: The address alignment should match the data width, otherwise it will cause address access errors										RW		

12.5.6 Channel x Destination Address Register (DMAC_DSTADR_x)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
DSTADR[31:16]															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
DSTADR[15:0]															
Bit	Marking	Place name	Function										Read and write		
b31~b0	DSTADR	DMA channel target address configuration	Its auto-reload can be configured via CONFBx.RD Note: The address alignment should match the data width, otherwise it will cause address access errors										RW		

13 Composite Timer (CTIM)

13.1 Overview

CTIM is a compound timer, which can be configured as a general-purpose timer GTIM with 4 groups of comparison capture functions, or as a group (three) of basic timers BTIM with timing functions only. by SysCtrl_CR1.GTIMxCFG, BTIM is valid when this bit is 1, and GTIM is valid when this bit is 0. Each CTIMs of this product can be independently and flexibly configured as GTIM or BTIM

BTIM is a group of timing counters with basic timing functions, each group contains three 16 -bit timers with reload function. This product can be configured with up to 9 basic timers: BTIM0~8.

GTIM is a 16 -bit timer with 4 -way compare capture function. This product can be configured with up to 3 general-purpose timers: GTIM0~2.

Table 13-1 Pin function and timer function comparison table

CTIM(x=0,1,2)	GTIM(x=0,1,2)	BTIM(y=0,3,6)
CTIMx_ETR	GTIMx_ETR	BTIMy_ETR
CTIMx_TOGP	GTIMx_TOGP	BTIMy+0_TOGP
CTIMx_TOGN	GTIMx_TOGN	BTIMy+0_TOGN
CTIMx_CH0	GTIMx_CH0	BTIMy+1_TOGP
CTIMx_CH1	GTIMx_CH1	BTIMy+1_TOGN
CTIMx_CH2	GTIMx_CH2	BTIMy+2_TOGP
CTIMx_CH3	GTIMx_CH3	BTIMy+2_TOGN

Note:

1. BTIM0/1/2 share CTIM0_ETR; BTIM3/4/5 share CTIM1_ETR; BTIM6/7/8 share CTIM2_ETR.
2. When x=0, y=0; when x=1, y=3; when x=2, y=6.

13.1.1 BTIM main characteristics

- 3 16bit timers _
- 2 toggle outputs per timer
- Single pulse mode
- External input count function
- Gating function
- External trigger function
- Cascade function

13.1.2 GTIM main characteristics

- 16bit timer
- 2 toggle outputs
- Single pulse mode
- External input count function
- Gating function
- External trigger function
- 4 independent PWM outputs, adjustment range 0% ~ 100%
- 4- way independent capture
- Cascade function

13.2 BTIM function description

13.2.1 Functional block diagram

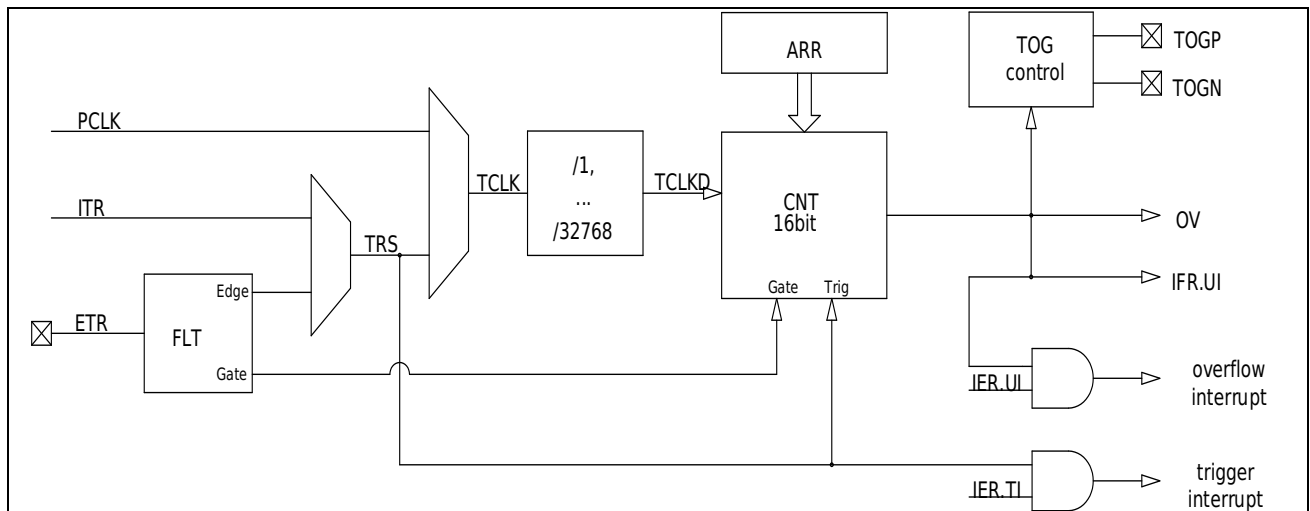


Figure 13-1 BTIM Block Diagram

Note:

1. A group of 3 BTIMs share one ETR.
2. For the source of ITR signal, please refer to the [Timer Cascading] function chapter.

13.2.2 Filter unit

ETR signal input by the external pin is filtered, polarity selected, and edge detected, the gate control signal and edge signal are output for use by the counting unit and the triggering unit. Since the ETR signal uses PCLK for synchronous filtering, the frequency of the input ETR signal should be lower than 1/2 of the frequency of PCLK.

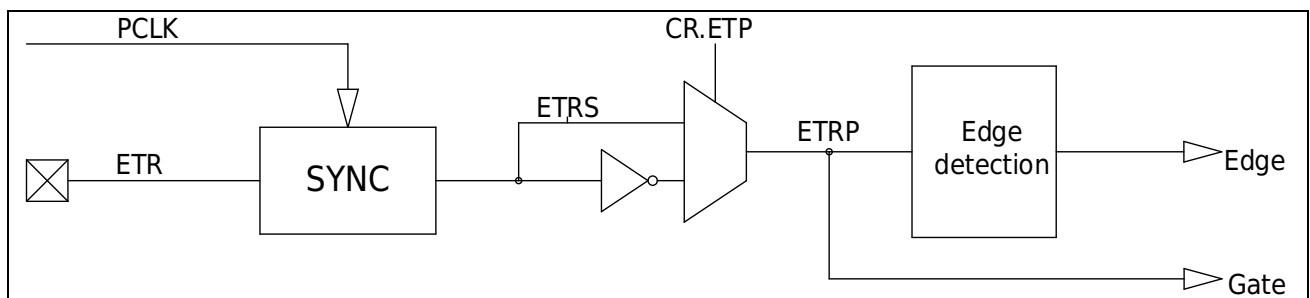


Figure 13-2 Schematic diagram of BTIM filter unit

13.2.3 Counting unit

On each rising edge of the TCLKD signal, the internal counter CNT is automatically incremented by 1. counting to ARR, an overflow signal OV (width of one PCLK) is generated, and the internal count value is reset to 0 and counting starts again. The counting range of the counter is 0 to ARR, and the counting period of the counting unit is $ARR+1$.

The timer supports 2 counting modes, One-Shot mode and continuous counting mode. When BTIMx_CR.OST is set to 1, the timer works in One-Shot mode; the timer stops counting immediately after an overflow occurs, and the hardware automatically clears CEN. When BTIMx_CR.OST is set to 0, the timer works in continuous counting mode; as long as BTIMx_CR.CEN is 1, the timer will keep counting.

Every time there is an overflow, BTIMx_IFR.UI will be set to 1 by hardware. If the corresponding interrupt is enabled, an overflow interrupt can be generated. The user should clear the overflow flag in the interrupt service routine.

If BTIMx_CR.TOGEN is 1, the output level of TOGP/TOGN is flipped every time overflow; the user can output the TOGP/TOGN signal to the pin through the multiplexing function of GPIO to drive the external circuit.

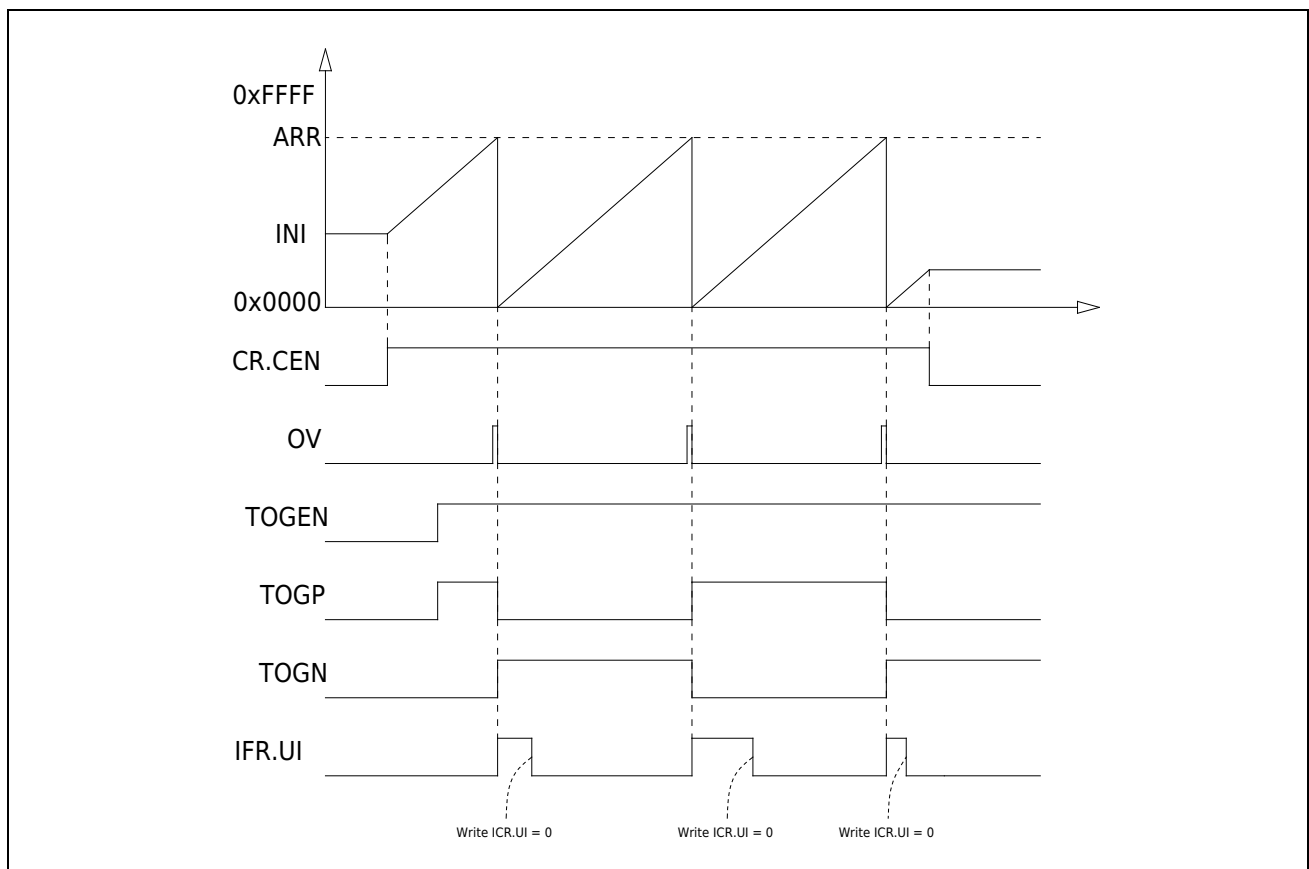


Figure 13-3 BTIM counter waveform

13.2.4 Timer mode

When setting BTIMx_CR.MD to 0x00, BTIMx works in timer mode. In this mode, the counting clock is the clock TCLKD after PCLK frequency division. When BTIMx_CR.CEN is set to 1, the timer starts counting up from the initial value, and overflows after counting to ARR and restarts counting from 0.

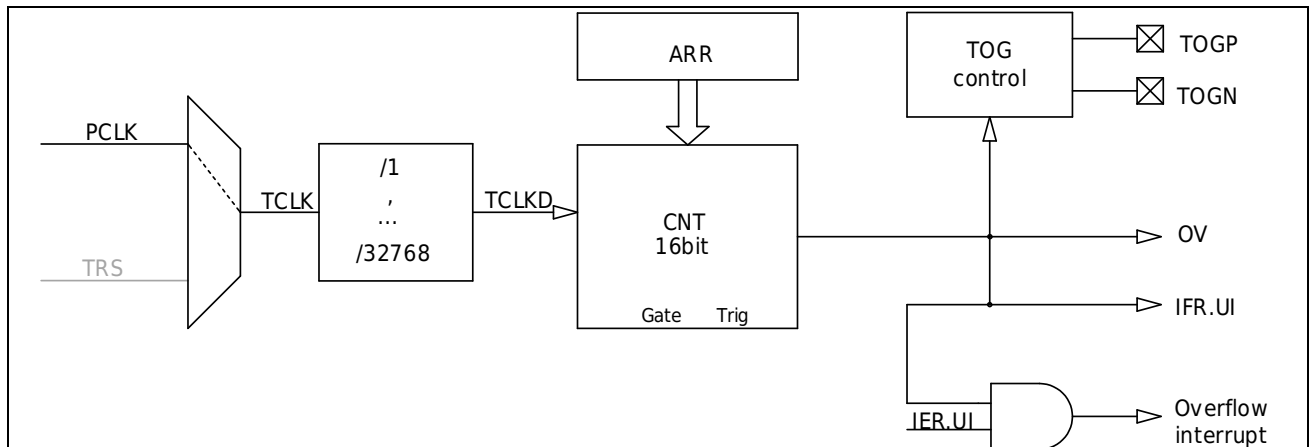


Figure 13-4 BTIM Timing Mode Block Diagram

13.2.5 Counter mode

When setting BTIMx_CR.MD to 0x01, BTIMx works in counter mode. the signal input by the ITR or ETR pin after frequency division. When BTIMx_CR.CEN is set to 1, the counter starts counting up from the initial value, and overflows after counting to ARR and starts counting from 0 again.

The counting clock source can be selected as ITR or ETR through BTIMx_CR.TRS; through BTIMx_CR.ETP, the rising edge or falling edge of the ETR pin input signal can be selected to be counted.

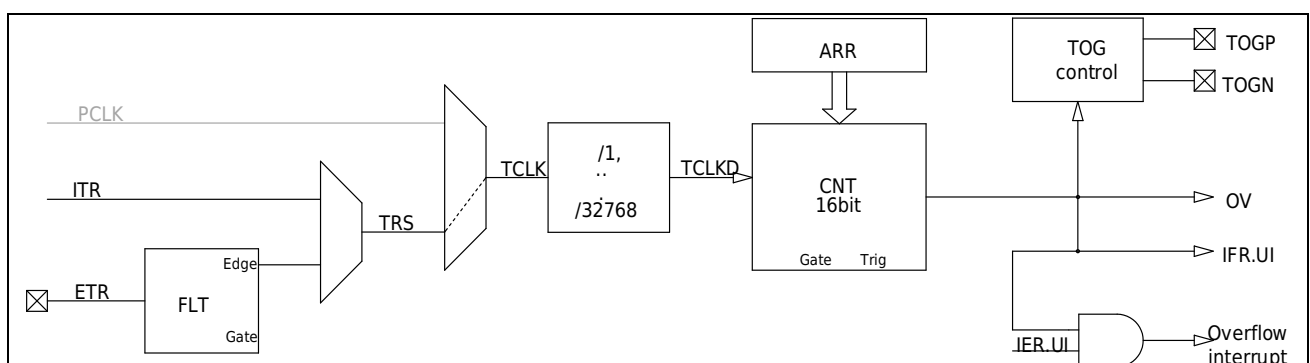


Figure 13-5 BTIM Counter Mode Block Diagram

13.2.6 Trigger start mode

When BTIMx_CR.MD is set to 0x02, BTIMx works in the trigger start mode, and the counting clock is the clock TCLKD after PCLK frequency division. In this mode, the timer has two start methods: set BTIMx_CR.CEN to 1 or there is an expected level change on the TRS signal.

After starting, the counter starts counting up from the initial value, overflows after counting to ARR and restarts counting from 0; set BTIMx_CR.CEN to 0 at any time, you can stop the timer.

Select TRS signal source as ETR or ITR through BTIMx_CR.TRS.

The rising edge or falling edge of the ETR pin input signal can be selected to start the timer through BTIMx_CR.ETP.

Every time an expected trigger signal is detected, BTIMx_IFR.TI will be set to 1 by hardware. If the corresponding interrupt is enabled, a trigger interrupt can be generated. The user should clear the trigger flag in the interrupt service routine.

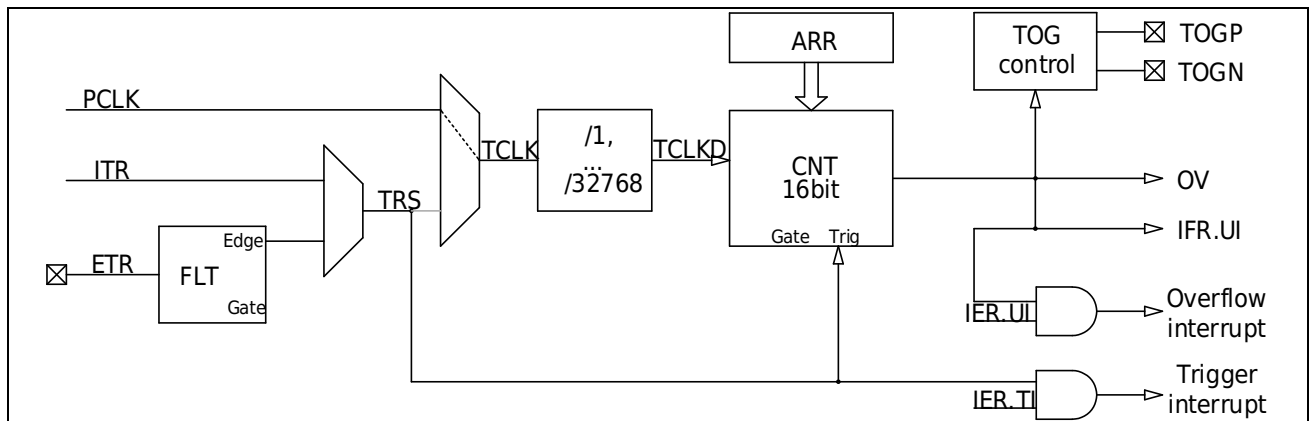


Figure 13-6 BTIM Trigger Initiator Mode Block Diagram

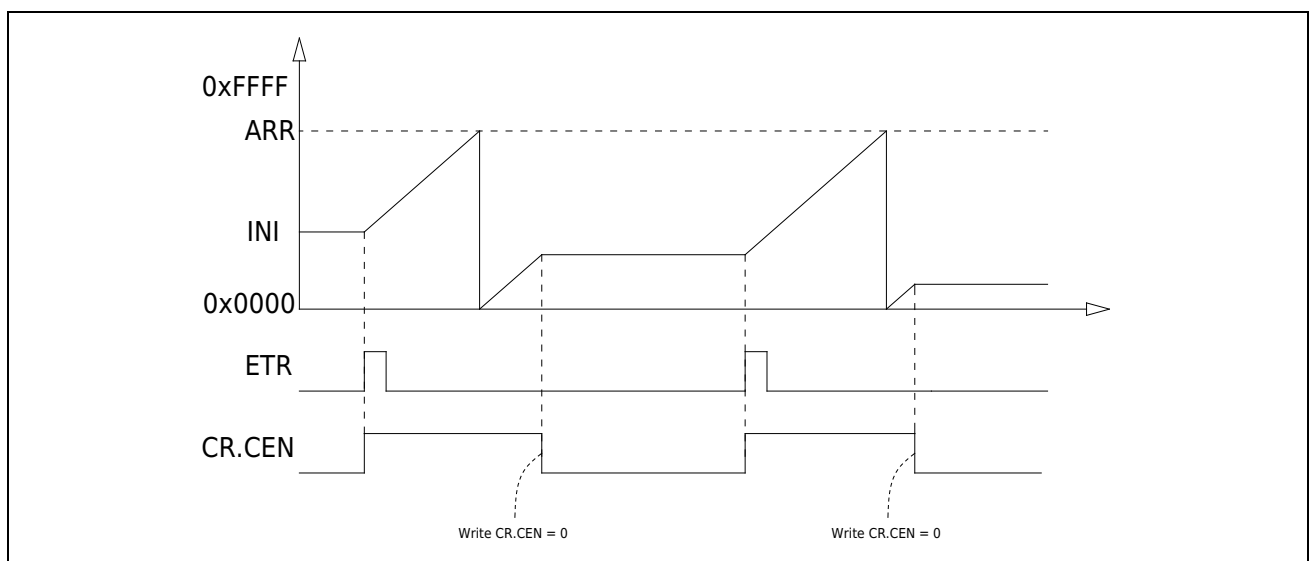


Figure 13-7 Schematic diagram of BTIM trigger start mode counting

13.2.7 Gating mode

When BTIMx_CR.MD is set to 0x03, BTIMx works in gated mode, and the counting clock is the clock TCLKD after PCLK frequency division. In this mode, the timer needs to meet two conditions before it can count: BTIMx_CR.CEN is 1, and the level on the ETR pin meets expectations. When the timer is running, the counter starts counting up from the initial value, overflows after counting to ARR and starts counting from 0 again.

The high level or low level of the input signal of the ETR pin can be selected as the counting level through BTIMx_CR.ETP.

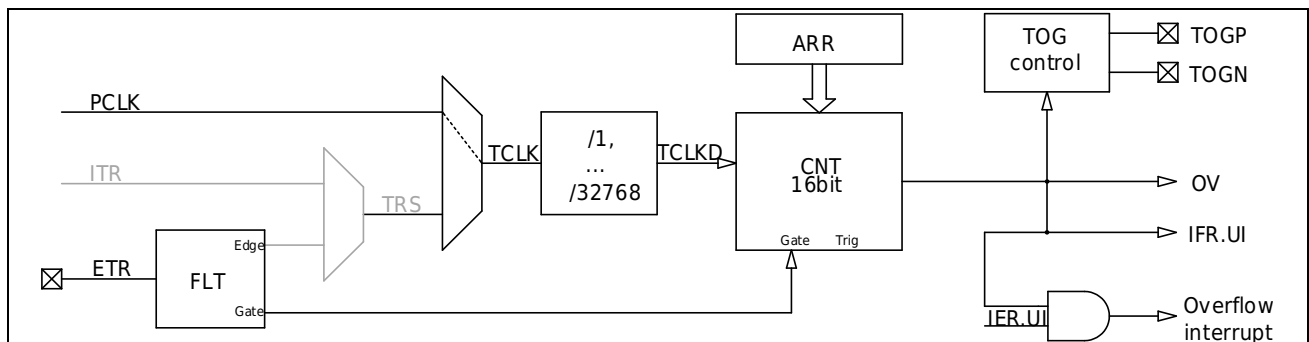


Figure 13-8 BTIM gating mode block diagram

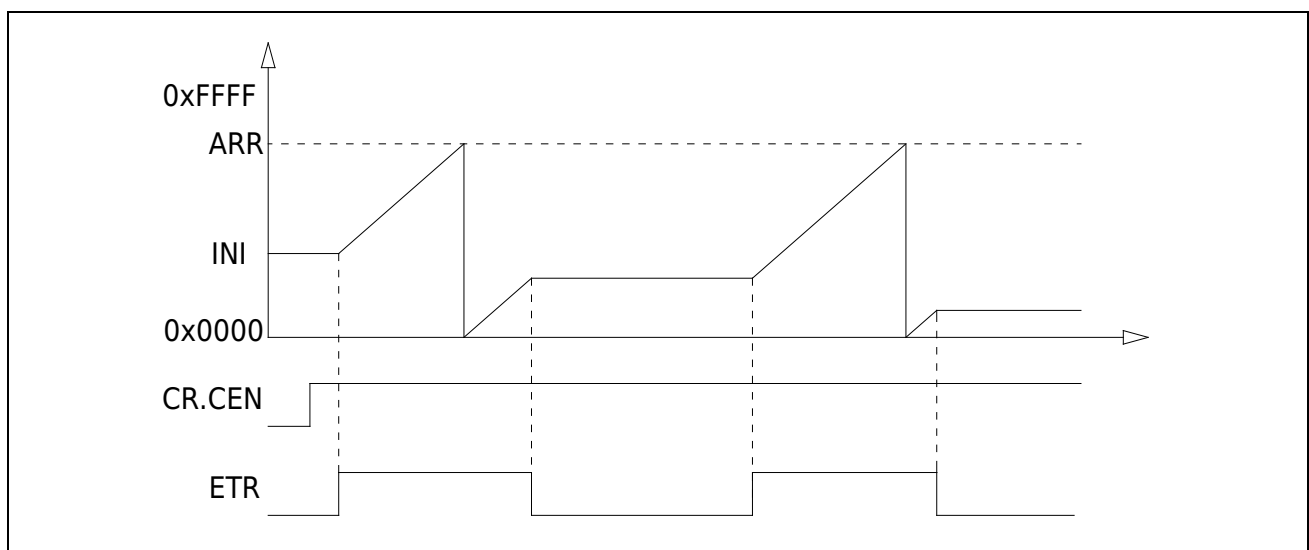


Figure 13-9 Schematic diagram of counting in BTIM gating mode

13.2.8 Timer Cascading

BTIM can be used in cascade through appropriate configuration, such as combining two 16-bit timers into a 32-bit timer. When used in cascade, the count clock of the first stage timer can be PCLK or the input signal from the ETR pin, and the count clock of the subsequent stage timer is the overflow signal of the previous stage timer. The signal cascading has been solidified in the chip, and its cascading method is as follows.

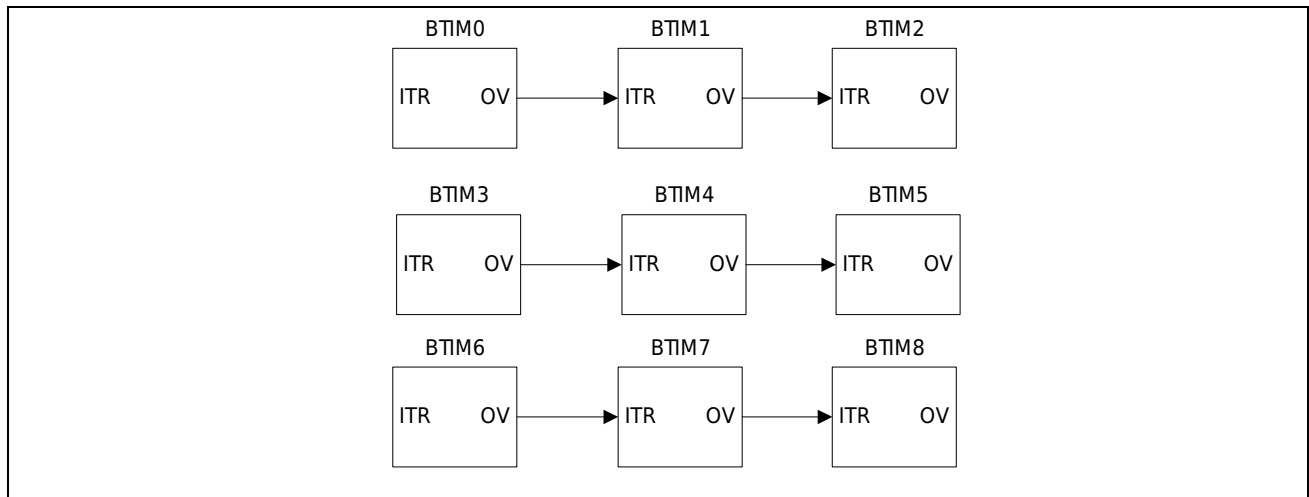


Figure 13-10 Schematic diagram of cascading -BTIM timers

13.2.9 Interrupt

Table 13-2 BTIM timer interrupt

Interrupt	Note
GTIM0_UEV_ETR_BTIMA	BTIM0 overflow / trigger interrupt
GTIM0_CH01_BTIMB	BTIM1 overflow / trigger interrupt
GTIM0_CH23_BTIMC	BTIM2 overflow / trigger interrupt
GTIM1_UEV_ETR_BTIMA	BTIM3 overflow / trigger interrupt
GTIM1_CH01_BTIMB	BTIM4 overflow / trigger interrupt
GTIM1_CH23_BTIMC	BTIM5 overflow / trigger interrupt
GTIM2_UEV_ETR_BTIMA	BTIM6 overflow / trigger interrupt
GTIM2_CH01_BTIMB	BTIM7 overflow / trigger interrupt
GTIM2_CH23_BTIMC	BTIM8 overflow / trigger interrupt

13.3 GTIM function description

13.3.1 Functional block diagram

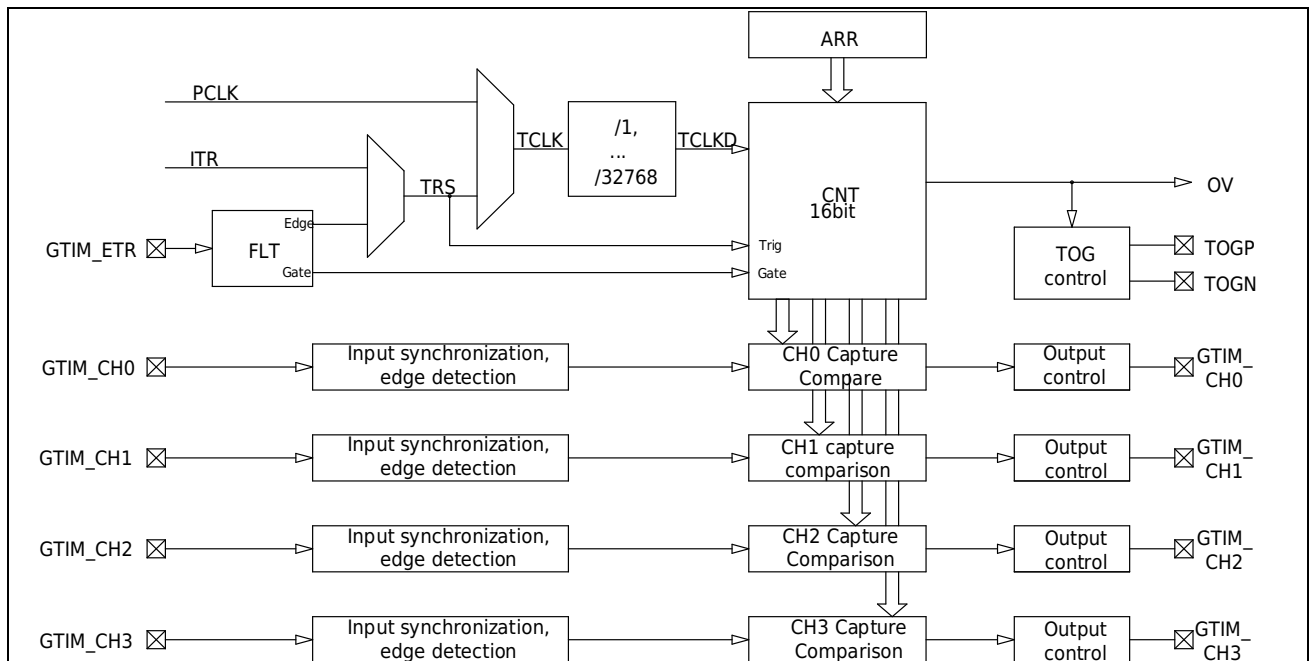


Figure 13-11 GTIM Block Diagram

Note: For the source of ITR signal, please refer to the cascading function chapter.

13.3.2 filter unit

After synchronous filtering, polarity selection, and edge detection of the ETR signal input by the external pin, the gate control signal and edge signal are output for use by the counting unit and the triggering unit. Since the ETR signal uses a filter clock for synchronous filtering, the frequency of the input ETR signal should be lower than one-half of the frequency of the filter clock.

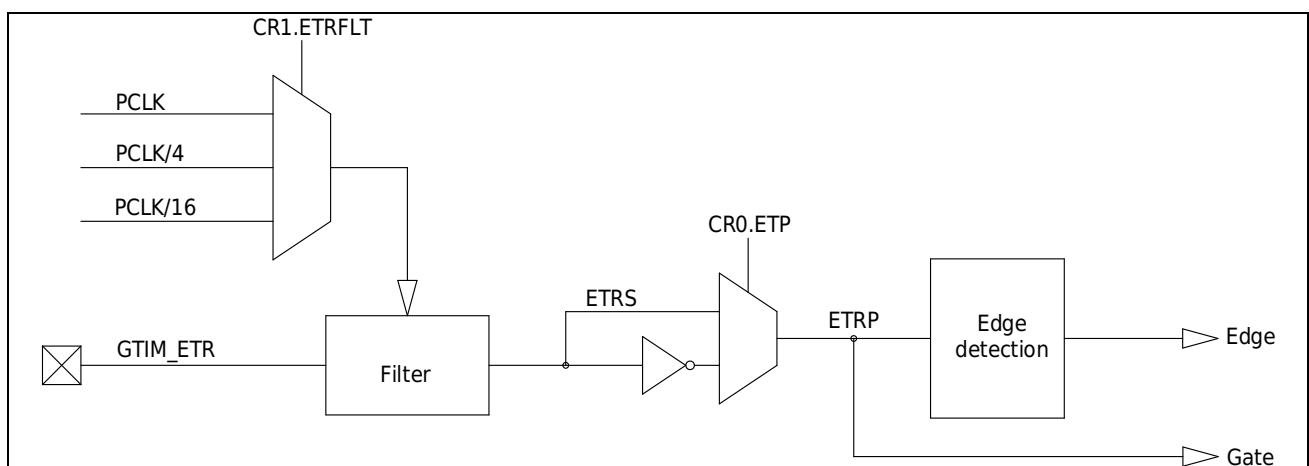


Figure 13-12 Schematic diagram of GTIM filter unit

13.3.3 Counting unit

On each rising edge of the TCLKD signal, the internal counter CNT is automatically incremented by 1. counting to ARR, an overflow signal OV (width of one PCLK) is generated, and the internal count value is reset to 0 and counting starts again. The counting range of the counter is 0 to ARR, and the counting period of the counting unit is $ARR+1$.

The timer supports 2 counting modes, One-Shot mode and continuous counting mode. When GTIM_CR0.OST is set to 1, the timer works in One-Shot mode; the timer stops counting immediately after an overflow occurs, and the hardware automatically clears CEN. When setting GTIM_CR0.OST to 0, the timer works in continuous counting mode; as long as GTIM_CR0.CEN is 1, the timer will keep counting.

Every time there is an overflow, GTIM_IFR.UI will be set to 1 by hardware. If the corresponding interrupt is enabled, an overflow interrupt can be generated. The user should clear the overflow flag in the interrupt service routine.

If GTIM_CR0.TOGEN is 1, the output level of TOGP/TOGN is flipped every time overflow; the user can output the TOGP/TOGN signal to the pin through the multiplexing function of GPIO to drive the external circuit.

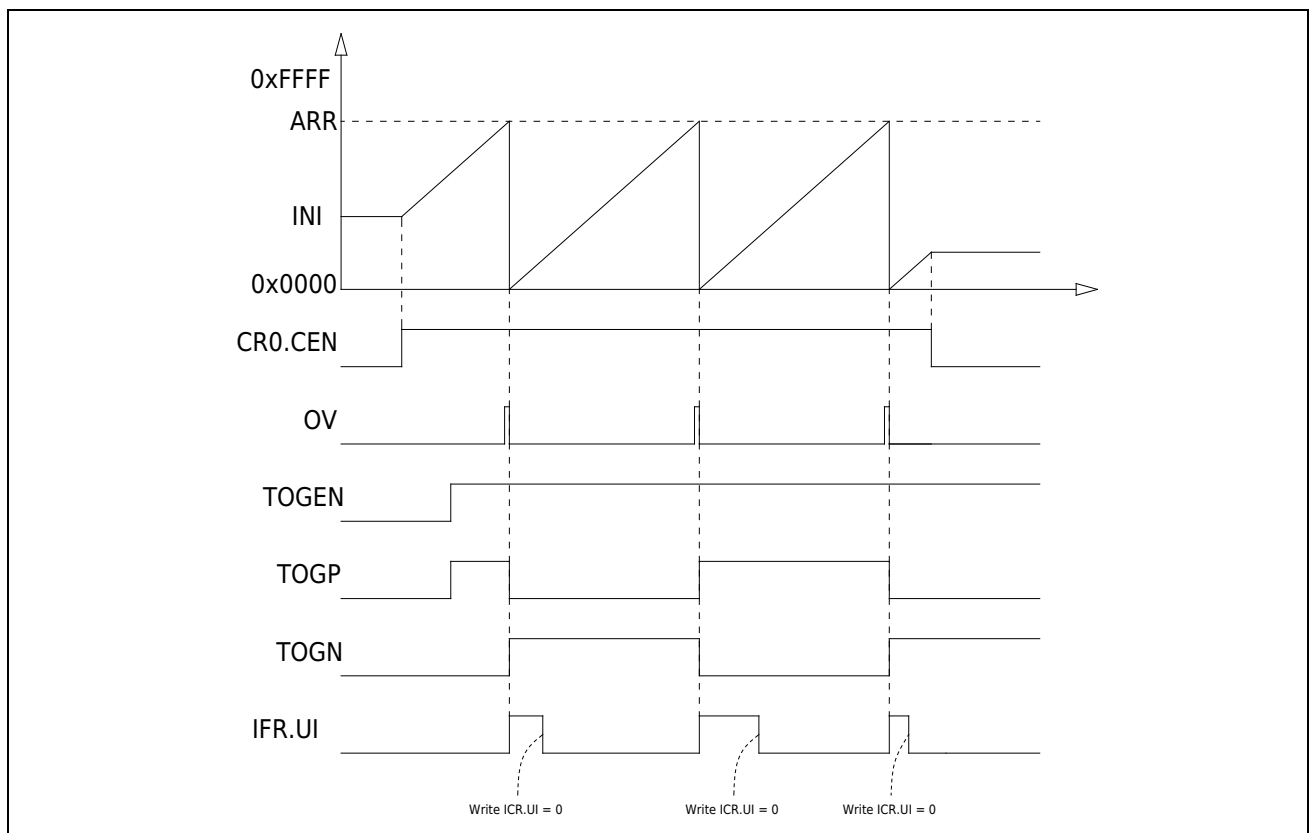


Figure 13-13 GTIM counter waveform

13.3.4 Timer mode

GTIM_CR0.MD is set to 0x00, GTIM works in timer mode. In this mode, the count clock is TCLKD after PCLK frequency division. When GTIM_CR0.CEN is set to 1, the timer starts counting up from the initial value, and overflows after counting to ARR and restarts counting from 0.

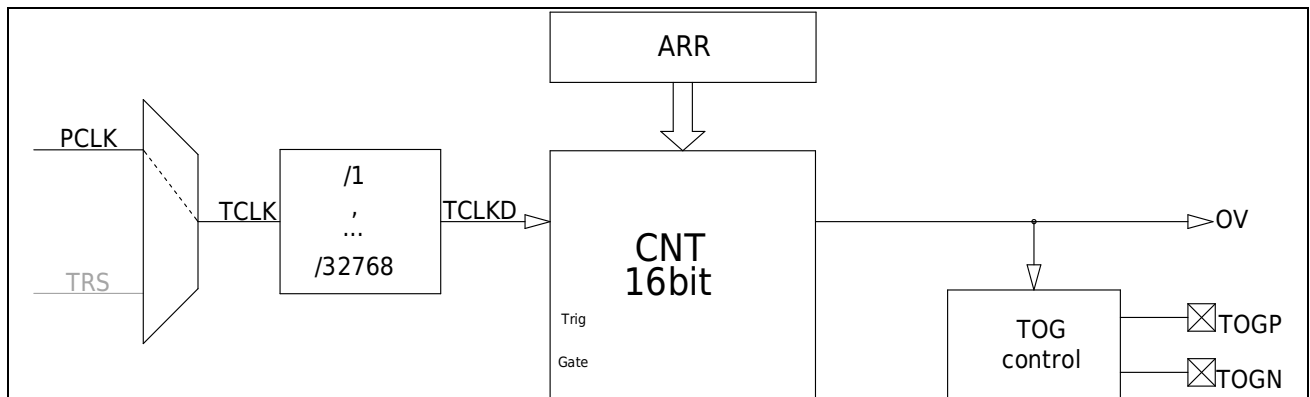


Figure 13-14 GTIM Timing Mode Block Diagram

13.3.5 Counter mode

GTIM_CR0.MD is set to 0x01, GTIM works in counter mode. the signal input by the ITR or ETR pin after frequency division. When GTIM_CR0.CEN is set to 1, the counter starts counting up from the initial value, and overflows after counting to ARR and starts counting from 0 again.

Through GTIM_CR0.TRS, you can select the counting clock source as ITR or ETR; through GTIM_CR0.ETP, you can choose to count the rising edge or falling edge of the ETR pin input signal.

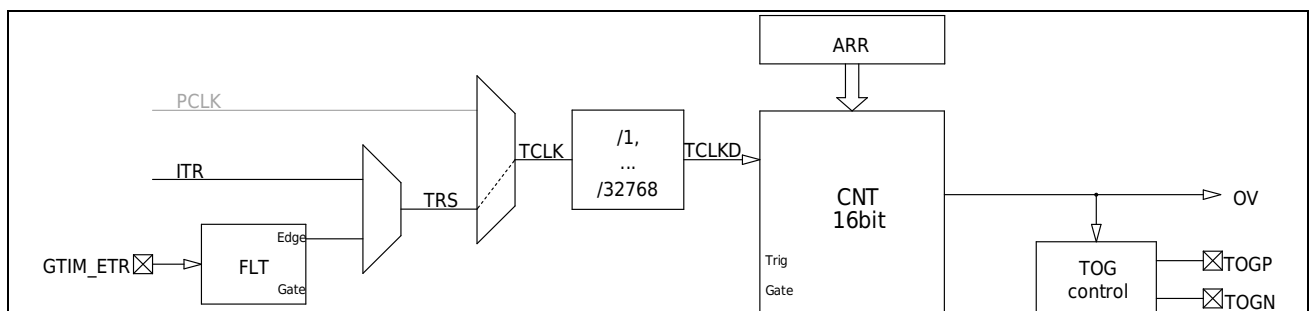


Figure 13-15 GTIM Counter Mode Block Diagram

13.3.6 Trigger start mode

When GTIM_CR0.MD is set to 0x02, GTIM works in trigger start mode, and the count clock is TCLKD after PCLK frequency division. In this mode, the timer has two start methods: set GTIM_CR0.CEN to 1 or the expected level change on the TRS signal.

After starting, the counter starts counting up from the initial value, overflows after counting to ARR, and restarts counting from 0; set GTIM_CR0.CEN to 0 at any time, you can stop the timer.

Select the TRS signal source as ETR or ITR through GTIM_CR0.TRS.

the ETR pin input signal can be selected to start the timer through GTIM_CR0.ETP.

Every time an expected trigger signal is detected, GTIM_IFR.TI will be set to 1 by hardware. If the corresponding interrupt is enabled, a trigger interrupt can be generated. The user should clear the trigger flag in the interrupt service routine.

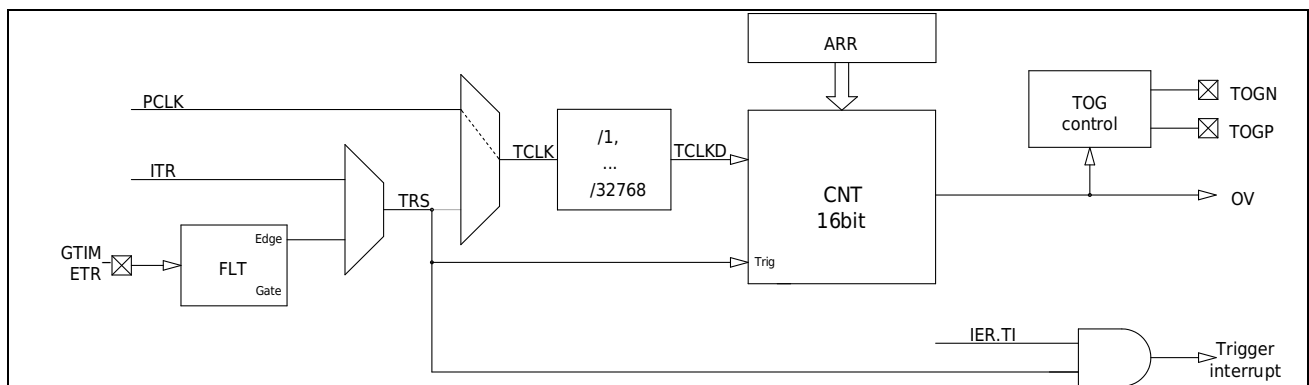


Figure 13-16 GTIM Trigger Initiator Mode Block Diagram

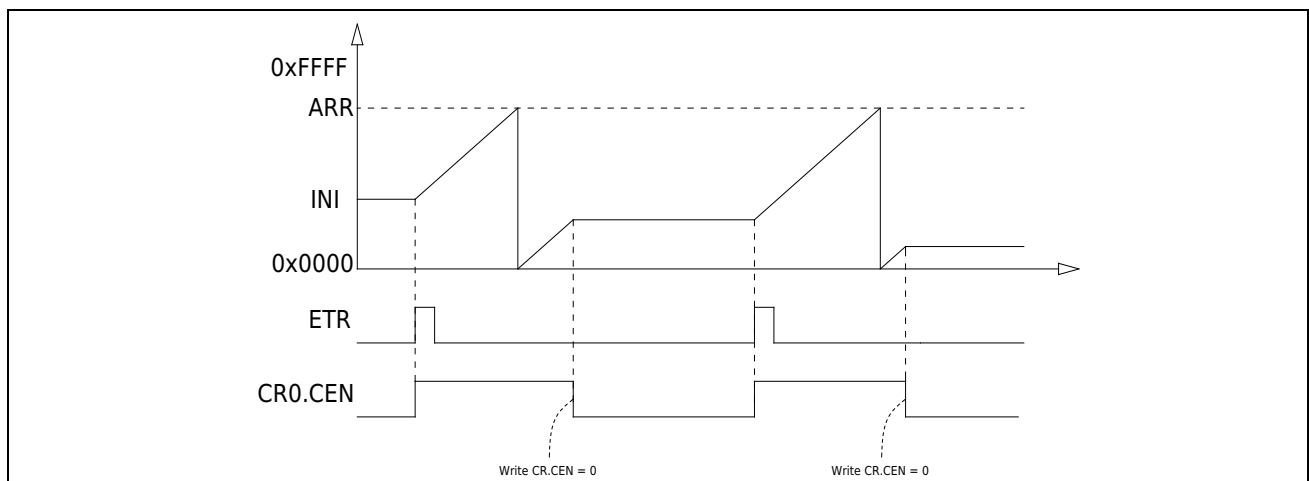


Figure 13-17 Counting diagram of GTIM trigger start mode

13.3.7 Gating mode

When GTIM_CR0.MD is set to 0x03, GTIM works in gated mode, and the count clock is TCLKD after PCLK frequency division. In this mode, the timer needs to meet two conditions before it can count: GTIM_CR0.CEN is 1, and the level on the ETR pin meets expectations. When the timer is running, the counter starts counting up from the initial value, overflows after counting to ARR and starts counting from 0 again.

The high level or low level of the input signal of the ETR pin can be selected as the counting level through GTIM_CR0.ETP.

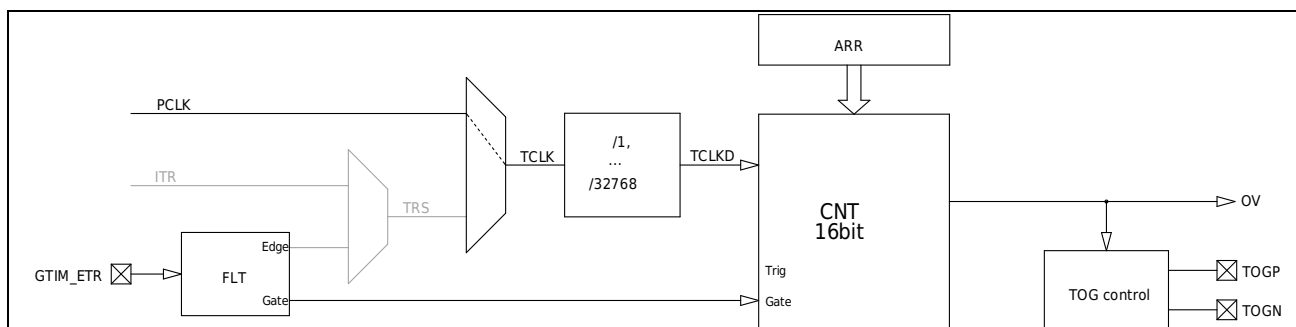


Figure 13-18 GTIM gating mode block diagram

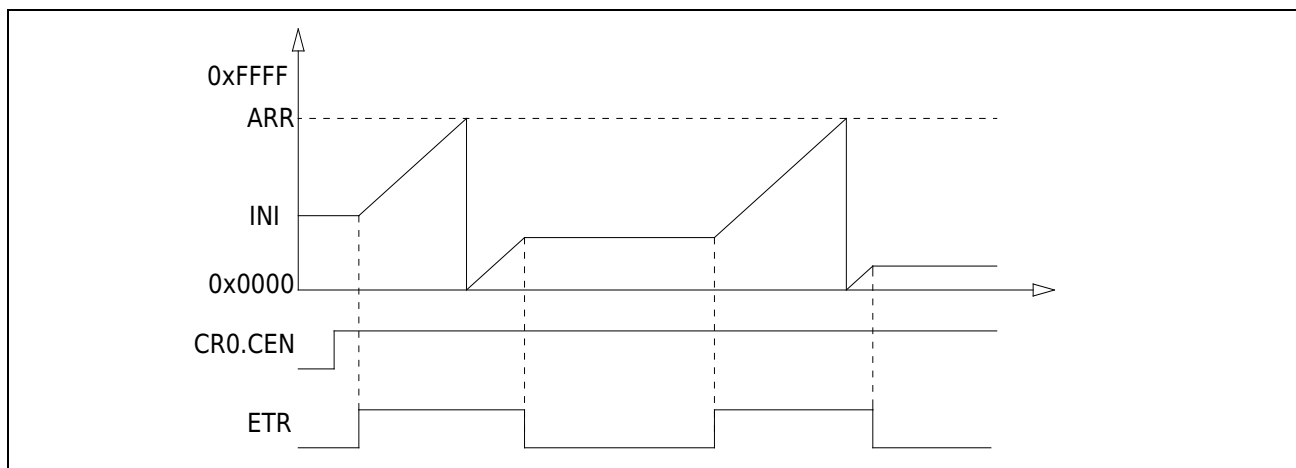


Figure 13-19 Schematic diagram of counting in GTIM gating mode

13.3.8 Compare capture function

GTIM has 4 comparison capture channels, CH0 ~ CH3.

13.3.8.1 Capture function

When configuring CMMR.CCyM as 1 ~ 3, CHy works in capture mode and supports three capture methods:

- 001: rising edge capture
- 010: Falling edge capture
- 011: Simultaneous capture of rising and falling edges

When the desired pulse edge appears on the CHy channel, the hardware automatically writes the current count value into the CCRy register and sets IFR.CCyF; if the interrupt of the corresponding channel is enabled, a comparison capture interrupt will be generated, and the user should set it in the interrupt service routine clear the interrupt flag.

The figure below shows the schematic of two channels for input capture. Among them, CH0 is captured on the rising edge; CH1 is captured on the rising and falling edges at the same time.

When using the capture mode, the pre-division CR0.PRS must be less than 4, that is, the pre-division can only use 1, 2, 4, and 8. Other pre-frequency division values are prohibited from being configured. When using external ETR counting, the capture function cannot be used.

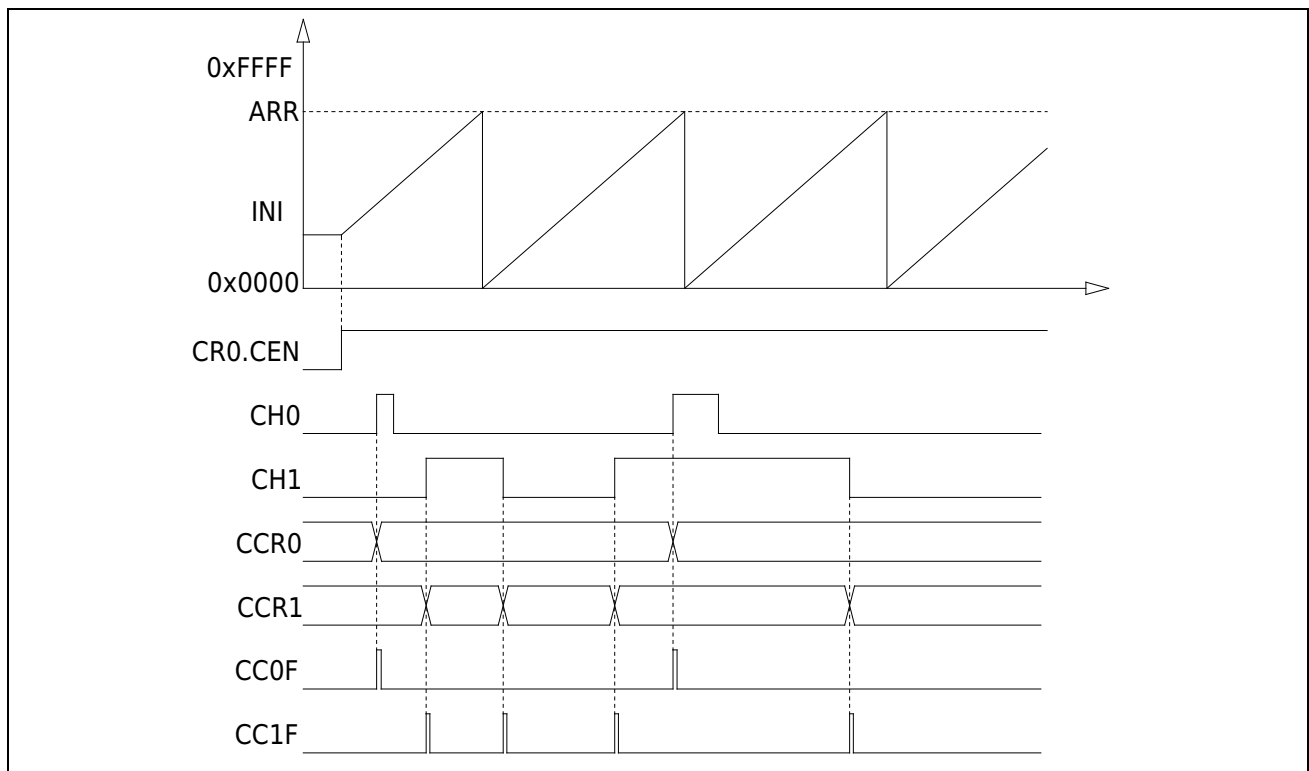


Figure 13-20 Schematic diagram of GTIM capture function

13.3.8.2 Compare function

CMMR.CCyM is set to 4 ~ 7, CHy works in comparison mode, and four waveforms can be compared:

- 100: forced output low level
- 101: Force output high level
- 110: PWM positive output(CNT $\geq$ CCRy output high level)
- 111: PWM reverse output(CNT < CCRy output high level)

Note: When the ARR value is not 0xFFFF, the PWM adjustment range can be 0% ~ 100%.

Example: In forward output mode, when ARR is 99, CCR is 0~100 and the corresponding duty cycle is 100%~0%.

The figure below shows the schematic diagram of the comparison output of two channels. Among them, CH0 adopts PWM forward output; CH1 adopts PWM reverse output.

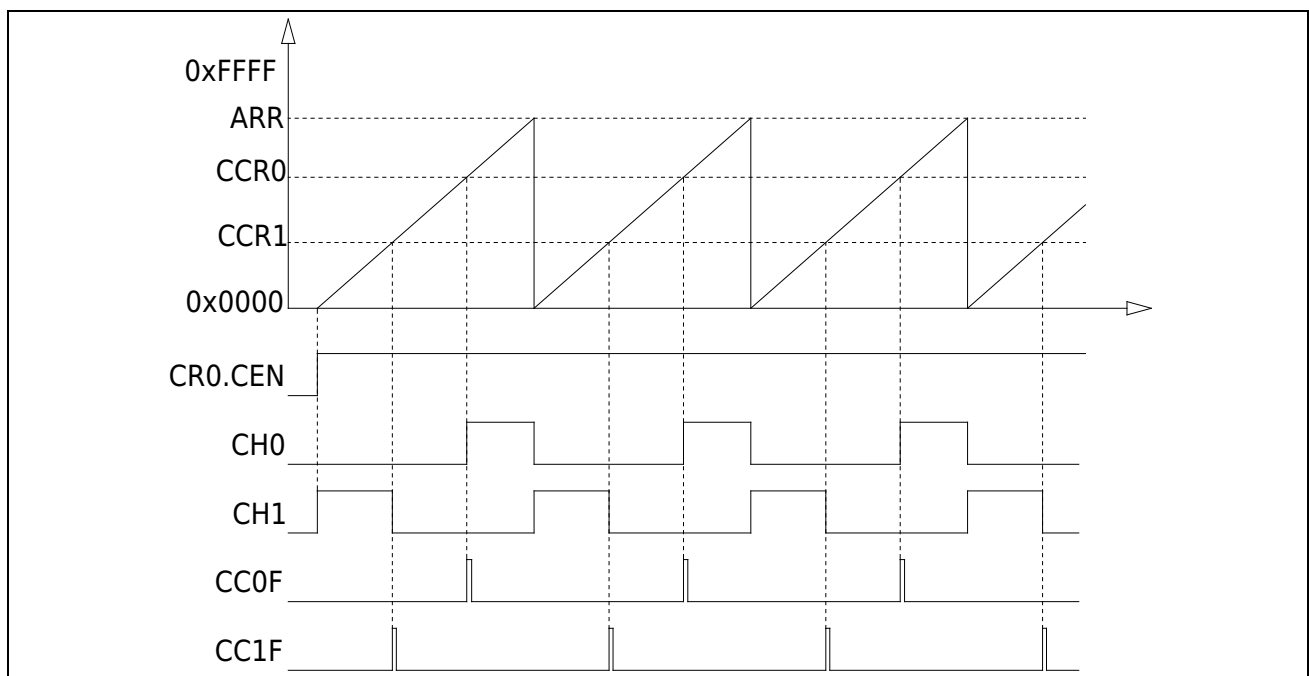


Figure 13-21 Schematic diagram of GTIM comparison function

13.3.9 DMA trigger

Compare capture can generate trigger signal to trigger DMA. Enable to use GTIM_IER.DCCx(x=0~3) control.

13.3.10 Interrupt

Table 13-3 List of GTIM interrupt sources

Interrupt	Note
GTIM0_UEV_ETR_BTIMA	GTIM0 overflow / trigger interrupt
GTIM0_CH01_BTIMB	GTIM0 channel 01 compare capture interrupt
GTIM0_CH23_BTIMC	GTIM0 channel 23 compare capture interrupt
GTIM1_UEV_ETR_BTIMA	GTIM1 overflow / trigger interrupt
GTIM1_CH01_BTIMB	GTIM1 channel 01 compare capture interrupt
GTIM1_CH23_BTIMC	GTIM1 channel 23 compare capture interrupt
GTIM2_UEV_ETR_BTIMA	GTIM2 overflow / trigger interrupt
GTIM2_CH01_BTIMB	GTIM2 channel 01 compare capture interrupt
GTIM2_CH23_BTIMC	GTIM2 channel 23 compare capture interrupt

13.3.11 On-chip peripheral interconnection

Trigger interconnection

Table 13-4 GTIM trigger interconnection list

CR0.TRS	00	01	10	11
GTIM0	ETR	ATIM0_TRGO	ADC0_ExtTrigger0.TRIG.	ADC0_ExtTrigger1.TRIG.
GTIM1	ETR	ATIM3_TRGO	ADC1_ExtTrigger0.TRIG.	ADC1_ExtTrigger1.TRIG.
GTIM2	ETR	ATIM0_TRGO	GTIM1 overflow.	ATIM3_TRGO.

When SYSCTRL.IOAUX1.ETRx is 0, the source of GTIM_ETR signal is determined by GPIO_AFRH/ GPIO_AFRL; when SYSCTRL.IOAUX1.ETRx is 1~7, the source of GTIM_ETR signal is determined by the following table.

Table 13-5 List of GTIM ETR sources

SYSCTRL.IOAUX1	0	1	2	3	4	5	6	7
CTIM0ETR[6:4]	GTIM0_ETR	USART1_RXD	VC0_OUT	LVD_OUT	VC1_OUT	PA05	PA13	PB05
CTIM1ETR[10:8]	GTIM1_ETR	USART2_RXD	VC0_OUT	LVD_OUT	VC1_OUT	PA06	PA14	PB06
CTIM2ETR[14:12]	GTIM2_ETR	USART3_RXD	VC0_OUT	LVD_OUT	VC1_OUT	PA07	PA15	PB07

When SYSCTRL.IOAUX2.CAPx is 0, the capture signal source of GTIM_CH3 is determined by GPIO_AFRH/ GPIO_AFRL; when SYSCTRL.IOAUX2.CAPx is 1~7, the source of GTIM_CH3 signal is determined by the following table.

Table 13-6 List of -GTIM CH3 signal sources

SYSCTRL.IOAUX2	0	1	2	3	4	5	6	7
CTIM0CAP[6:4]	GTIM0_CH3	USART1_RXD	VC0_OUT	LVD_OUT	VC1_OUT	PA01	PA09	PB01
CTIM1CAP[10:8]	GTIM1_CH3	USART2_RXD	VC0_OUT	LVD_OUT	VC1_OUT	PA02	PA10	PB02
CTIM2CAP[14:12]	GTIM2_CH3	USART3_RXD	VC0_OUT	LVD_OUT	VC1_OUT	PA03	PA11	PB03

13.4 BTIM register description

13.4.1 Register list

Table 13-7 BTIM base address list

Name	base address	Description
BTIM0	0x4000 5000	BTIM0 base address
BTIM1	0x4000 5100	BTIM1 base address
BTIM2	0x4000 5200	BTIM2 base address
BTIM3	0x4000 5400	BTIM3 base address
BTIM4	0x4000 5500	BTIM4 base address
BTIM5	0x4000 5600	BTIM5 base address
BTIM6	0x4000 5800	BTIM6 base address
BTIM7	0x4000 5900	BTIM7 base address
BTIM8	0x4000 5A00	BTIM8 base address

Note: BTIM is valid only when SysCtrl_CR1.GTIMxCFG is 1.

Table 13-8 BTIM register list

(x=0~8)

Register name	Offset address	Reset value
Reload Register(BTIMx_ARR)	0x00	0x0000 FFFF
Counter Register(BTIMx_CNT)	0x04	0x0000 0000
Control Register(BTIMx_CR)	0x10	0x0000 0000
Interrupt Enable Register(BTIMx_IER)	0x14	0x0000 0000
Interrupt Flag Register(BTIMx_IFR)	0x18	0x0000 0000
Interrupt Clear Register(BTIMx_ICR)	0x1C	0x0000 0003
Composite Interrupt Flag Register(BTIM_AIFR)	0x48	0x0000 0000
Composite Interrupt Clear Register(BTIM_AICR)	0x4C	0x0000 003F

13.4.2 Reload register (BTIMx_ARR) (x=0~8)

Reset value: 0x0000 FFFF

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
ARR															

Bit	Marking	Place name	Function	Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value	
b15~b0	ARR	Timer reload value	-	RW

13.4.3 Count register (BTIMx_CNT) (x=0~8)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CNT															

Bit	Marking	Place name	Function	Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value	
b15~b0	CNT	Timer count value	-	RW

13.4.4 Control Register (BTIMx_CR) (x=0~8)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res				ETP	TRS			OST	PRS			TOGEN	MD		CEN

Bit	Marking	Place name	Function	Read and write
b31~b12	Res	Reserved bit	Reserved, must remain at reset value	
b11	ETP	ETR signal polarity selection for external pin input	0: ETR positive(trigger mode rising edge is valid, gating mode high level is valid) 1: ETR reverse(trigger mode falling edge is valid, gating mode is low level valid)	RW
b10~b9	TRS	Trigger source selection	00: ETR pin input signal 01: ITR, please refer to the chapter of timer cascade function for details Note: BTIM0~2 share one ETR pin; BTIM3~5 share another ETR pin, BTIM6~8 share another ETR pin	RW
b8	OST	Single / continuous count mode control	0: continuous counting mode 1: Single count mode	RW
b7~b4	PRS	Prescaler Frequency Division Configuration	0: DIV1 4: DIV16 8: DIV256 12: DIV4096 1: DIV2 5: DIV32 9: DIV512 13: DIV8192 2: DIV4 6: DIV64 10: DIV1024 14: DIV16384 3: DIV8 7: DIV128 11: DIV2048 15: DIV32768	RW
b3	TOGEN	TOG pin output enable control	0: TOGP, TOGN output levels are both 0 1: TOGP, TOGN output signals with opposite levels	RW
b2~b1	MD	Timer working mode configuration	00: Timer mode, the count clock is PCLK 01: Counter mode, counting clock is TRS signal 10: Trigger start mode, the count clock is PCLK, TRS signal triggers the counter to start 11: Gating mode, counting clock is PCLK, ETR pin input signal is used as gating	RW
b0	CEN	Timer Enable Control	0: timer stopped 1: Timer enabled Note: After the trigger start mode is triggered, CEN is set to 1 by hardware	RW

13.4.5 Interrupt enable (BTIMx_IER) (x=0~8)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res														TI	UI

Bit	Marking	Place name	Function	Read and write
b31~b2	Res	Reserved bit	Reserved, must remain at reset value	
b1	TI	Trigger interrupt enable control	0: disable trigger interrupt 1: enable trigger interrupt	RW
b0	UI	Overflow interrupt enable control	0: disable overflow interrupt 1: enable overflow interrupt	RW

13.4.6 Interrupt Flag Register (BTIMx_IFR) (x=0~8)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res														TI	UI

Bit	Marking	Place name	Function	Read and write
b31~b2	Res	Reserved bit	Reserved, must remain at reset value	
b1	TI	trigger flag	0: No trigger event occurred 1: A trigger event has occurred	RO
b0	UI	counter overflow flag	0: The counter has not overflowed 1: The counter has overflowed	RO

13.4.7 Interrupt Flag Clear Register (BTIMx_ICR) (x=0~8)

Reset value: 0x0000 0003

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res														TI	UI

Bit	Marking	Place name	Function	Read and write
b31~b2	Res	Reserved bit	Reserved, must remain at reset value	
1	TI	trigger flag clear	0: clear trigger flag 1: no function	R1W0
0	UI	Counter overflow flag cleared	0: Clear counter overflow flag 1: no function	R1W0

13.4.8 Composite Interrupt Flag Register (BTIM_AIFR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res										TI2	UI2	TI1	UI1	TI0	UI0

Bit	Marking	Place name	Function	Read and write
b31~b2	Res	Reserved bit	Reserved, must remain at reset value	
b5	TI2	BTIM2/5/8 trigger flag	0: No trigger event occurred 1: A trigger event has occurred	R0
b4	UI2	BTIM2/5/8 overflow flag	0: The counter has not overflowed 1: The counter has overflowed	R0
b3	TI1	BTIM1/4/7 trigger flag	0: No trigger event occurred 1: A trigger event has occurred	R0
b2	UI1	BTIM1/4/7 overflow flag	0: The counter has not overflowed 1: The counter has overflowed	R0
b1	TI0	BTIM0/3/6 trigger flag	0: No trigger event occurred 1: A trigger event has occurred	R0
b0	UI0	BTIM0/3/6 overflow flag	0: The counter has not overflowed 1: The counter has overflowed	R0

13.4.9 Composite Interrupt Flag Clear Register (BTIM_AICR)

Reset value: 0x0000 003F

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res										TI2	UI2	TI1	UI1	TI0	UI0

Bit	Marking	Place name	Function	Read and write
b31~b2	Res	Reserved bit	Reserved, must remain at reset value	
b5	TI2	BTIM2/5/8 trigger flag clear	0: clear trigger flag 1: no function	R1W0
b4	UI2	BTIM2/5/8 overflow flag clear	0: clear overflow flag 1: no function	R1W0
b3	TI1	BTIM1/4/7 Trigger Flag Clear	0: clear trigger flag 1: no function	R1W0
b2	UI1	BTIM1/4/7 overflow flag cleared	0: clear overflow flag 1: no function	R1W0
b1	TI0	BTIM0/3/6 trigger flag clear	0: clear trigger flag 1: no function	R1W0
b0	UI0	BTIM0/3/6 overflow flag clear	0: clear overflow flag 1: no function	R1W0

13.5 GTIM register description

13.5.1 Register list

Table 13-9 GTIM base address list

Timer	base address	Description
GTIM0	0x4000 5000	GTIM0 base address
GTIM1	0x4000 5400	GTIM1 base address
GTIM2	0x4000 5800	GTIM2 base address

Note: GTIM is valid when SysCtrl_CR1.GTIMxCFG is 0, and invalid when it is 1.

Table 13-10 GTIM register list

Register name	Offset address	Reset value
Reload Register(GTIM_ARR)	0x300	0x0000 FFFF
count register(GTIM_CNT)	0x304	0x0000 0000
Compare Capture Control Register(GTIM_CMMR)	0x308	0x0000 0000
Control Register 1 (GTIM_CR1)	0x30C	0x0000 0000
Control Register 0 (GTIM_CR0)	0x310	0x0000 0000
Interrupt Enable Control Register(GTIM_IER)	0x314	0x0000 0000
Interrupt Flag Register(GTIM_IFR)	0x318	0x0000 0000
Interrupt Flag Clear Register(GTIMx_ICR)	0x31C	0x0000 003F
CH0 Compare / Capture Register(GTIM_CCR0)	0x320	0x0000 FFFF
CH1 Compare / Capture Register(GTIM_CCR1)	0x324	0x0000 0000
CH2 Compare / Capture Register(GTIM_CCR2)	0x328	0x0000 FFFF
CH3 Compare / Capture Register(GTIM_CCR3)	0x32C	0x0000 0000

13.5.2 Reload Register (GTIM_ARR)

Reset value: 0x0000 FFFF

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
ARR															
Bit	Marking	Place name	Function												Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value												
b15~b0	ARR	timer reload value													RW

13.5.3 Count register (GTIM_CNT)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CNT															
Bit	Marking	Place name	Function												Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value												
b15~b0	CNT	Timer count value	-												RW

13.5.4 Control Register 1 (GTIM_CR1)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res								ETRFLT			Res				

Bit	Marking	Place name	Function	Read and write
b31~b7	Res	Reserved bit	Reserved, must remain at reset value	
b6~b4	ETRFLT	ETR pin input signal filter configuration	The sampling clock is PCLK or PCLK frequency division, and the number of sampling points is N 000: no filtering 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: $F_{\text{sample}} = \text{PCLK}/8$, $N=6$	RW
b3~b0	Res	Reserved bit	Reserved, must remain at reset value	

13.5.5 Control Register 0 (GTIM_CR0)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res				ETP	TRS			OST	PRS			TOGEN	MD		CEN

Bit	Marking	Place name	Function	Read and write
b31~b12	Res	Reserved bit	Reserved, must remain at reset value	
b11	ETP	External input ETR polarity selection	0: external input ETR positive(trigger mode rising edge is valid, gating mode high level is valid) 1: The external input ETR is reversed(the falling edge of the trigger mode is valid, and the low level of the gate control mode is valid)	RW
b10~b9	TRS	Trigger source selection	00: ETR pin input signal 01: Internal interconnection 1 10: Internal interconnection 2 11: Internal interconnection 3 Note: For details on the internal interconnection, see the function description of the interconnection of internal and external devices on the chip.	RW
b8	OST	Single / continuous count mode control	0: continuous counting mode 1: Single count mode	RW
b7~b4	PRS	Prescaler Frequency Division Configuration	0: DIV1 4: DIV16 8: DIV256 12: DIV4096 1: DIV2 5: DIV32 9: DIV512 13: DIV8192 2: DIV4 6: DIV64 10: DIV1024 14: DIV16384 3: DIV8 7: DIV128 11: DIV2048 15: DIV32768 Note: The pre-division coefficients in the capture mode can only be set to 0,1,2, and 3.	RW
b3	TOGEN	TOG pin output enable control	0: TOGP, TOGN output levels are both 0 1: TOGP, TOGN output signals with opposite levels	RW
b2~b1	MD	Timer working mode configuration	00: Timer mode, the count clock is PCLK 01: Counter mode, counting clock is TRS signal 10: Trigger start mode, the count clock is PCLK, TRS signal triggers the counter to start 11: Gating mode, counting clock is PCLK, ETR pin input signal is used as gating	RW
b0	CEN	Timer Enable Control	0: timer stopped 1: Timer enabled Note: After the trigger start mode is triggered, CEN is set to 1 by hardware	RW

13.5.6 Interrupt Enable Control Register (GTIM_IER)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res				C0TRAD	OVTRAD	DCC3	DCC2	DCC1	DCC0	CC3	CC2	CC1	CC0	TI	UI

Bit	Marking	Place name	Function	Read and write
b31~b12	Res	Reserved bit	Reserved, must remain at reset value	
b11	C0TRAD	Compare capture 0 trigger ADC enable	0: Prohibited 1: Enable	RW
b10	OVTRAD	Overflow trigger ADC enable control	0: Prohibited 1: Enable	RW
b9	DCC3	CH3 triggers DMA enable control	0: Prohibited 1: Enable	RW
b8	DCC2	CH2 triggers DMA enable control	0: Prohibited 1: Enable	RW
b7	DCC1	CH1 triggers DMA enable control	0: Prohibited 1: Enable	RW
b6	DCC0	CH0 triggers DMA enable control	0: Prohibited 1: Enable	RW
b5	CC3	CH3 capture compare interrupt enable control	0: Prohibited 1: Enable	RW
b4	CC2	CH2 capture compare interrupt enable control	0: Prohibited 1: Enable	RW
b3	CC1	CH1 capture compare interrupt enable control	0: Prohibited 1: Enable	RW
b2	CC0	CH0 capture compare interrupt enable control	0: Prohibited 1: Enable	RW
b1	TI	Trigger interrupt enable control	0: Prohibited 1: Enable	RW
b0	UI	Overflow interrupt enable control	0: Prohibited 1: Enable	RW

13.5.7 Interrupt Flag Register (GTIM_IFR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res										CC3	CC2	CC1	CC0	TI	UI

Bit	Marking	Place name	Function	Read and write
b31~b6	Res	Reserved bit	Reserved, must remain at reset value	
b5	CC3	CH3 comparison capture interrupt flag	0: no compare capture event occurs 1: A compare capture event has occurred	RW
b4	CC2	CH2 comparison capture interrupt flag	0: no compare capture event occurs 1: A compare capture event has occurred	RW
b3	CC1	CH1 comparison capture interrupt flag	0: no compare capture event occurs 1: A compare capture event has occurred	RW
b2	CC0	CH0 comparison capture interrupt flag	0: no compare capture event occurs 1: A compare capture event has occurred	RW
b1	TI	trigger interrupt flag	0: No trigger event occurred 1: A trigger event has occurred	RW
b0	UI	Overflow interrupt flag	0: The counter has not overflowed 1: The counter has overflowed	RW

13.5.8 Interrupt Flag Clear Register (GTIM_ICR)

Reset value: 0x0000 003F

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res										CC3	CC2	CC1	CC0	TI	UI

Bit	Marking	Place name	Function	Read and write
b31~b6	Res	Reserved bit	Reserved, must remain at reset value	
b5	CC3	CH3 compare capture interrupt flag clear	0: Clear compare capture interrupt flag 1: no function	R1W0
b4	CC2	CH2 compare capture interrupt flag clear	0: Clear compare capture interrupt flag 1: no function	R1W0
b3	CC1	CH1 compare capture interrupt flag clear	0: Clear compare capture interrupt flag 1: no function	R1W0
b2	CC0	CH0 compare capture interrupt flag clear	0: Clear compare capture interrupt flag 1: no function	R1W0
b1	TI	trigger interrupt flag clear	0: Clear compare capture interrupt flag 1: no function	R1W0
b0	UI	overflow interrupt flag clear	0: Clear counter overflow flag 1: no function	R1W0

13.5.9 Compare Capture Control Register (GTIM_CMMR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res	CC3M			Res	CC2M			Res.	CC1M			Res	CC0M		

Bit	Marking	Place name	Function	Read and write
b31~b15	Res	Reserved bit	Reserved, must remain at reset value	
b14~b12	CC3M	CH3 capture compare mode configuration	CC0M for function description	RW
b11	Res	Reserved bit	Reserved, must remain at reset value	
b10~b8	CC2M	CH2 capture compare mode configuration	CC0M for function description	RW
b7	Res	Reserved bit	Reserved, must remain at reset value	
b6~b4	CC1M	CH1 capture compare mode configuration	CC0M for function description	RW
b3	Res	Reserved bit	Reserved, must remain at reset value	
b2~b0	CC0M	CH0 capture compare mode configuration	000: no function 001: rising edge capture 010: Falling edge capture 011: Simultaneous capture of rising and falling edges 100: forced output low level 101: Force output high level 110: PWM positive output(CNT ≥ CCR0 output high level) 111: PWM reverse output(CNT < CCR output high level)	RW

13.5.10 Compare Capture Register (GTIM_CCRy) (y=0,1,2,3)

Reset value: 0x0000 FFFF/ 0x0000 0000/ 0x0000 FFFF/ 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CCR															

Bit	Marking	Place name	Function	Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value	
b15~b0	CCR	16 -bit compare capture register		RW

14 Advanced Timer (ATIM)

14.1 Overview

ATIM3 is a timer composed of 1 counting unit and 7 comparing units, and supports 6 independent PWM outputs or 3 pairs of complementary PWM outputs. Supports 6 capture inputs.

Using timer prescaler, system prescaler and system clock selection, the pulse width and waveform period can be flexibly adjusted; the pulse width can be measured conveniently.

ATIM0 is a timer composed of 1 counting unit and 2 comparing units, and supports 2 independent PWM outputs or 1 pair of complementary PWM outputs. Supports 2 capture inputs.

Using timer prescaler, system prescaler and system clock selection, the pulse width and waveform period can be flexibly adjusted; the pulse width can be measured conveniently.

14.2 Main characteristics

14.2.1 Main features of ATIM3

- 6 independent PWM outputs CH0A, CH0B, CH1A, CH1B, CH2A, CH2B
- 3 complementary PWM outputs(CHxA, CHxB) + 1 independent PWM output(gate)
- 3- way complementary PWM output(CHxA, CHxB) + 1 -way capture function(gate)
- Up to 6 capture inputs
- Pulse Width Measurement
- Dead zone control
- Brake control
- Edge alignment, symmetric center alignment and asymmetric center alignment PWM output
- Quadrature code counting function
- Single pulse mode
- External input count function

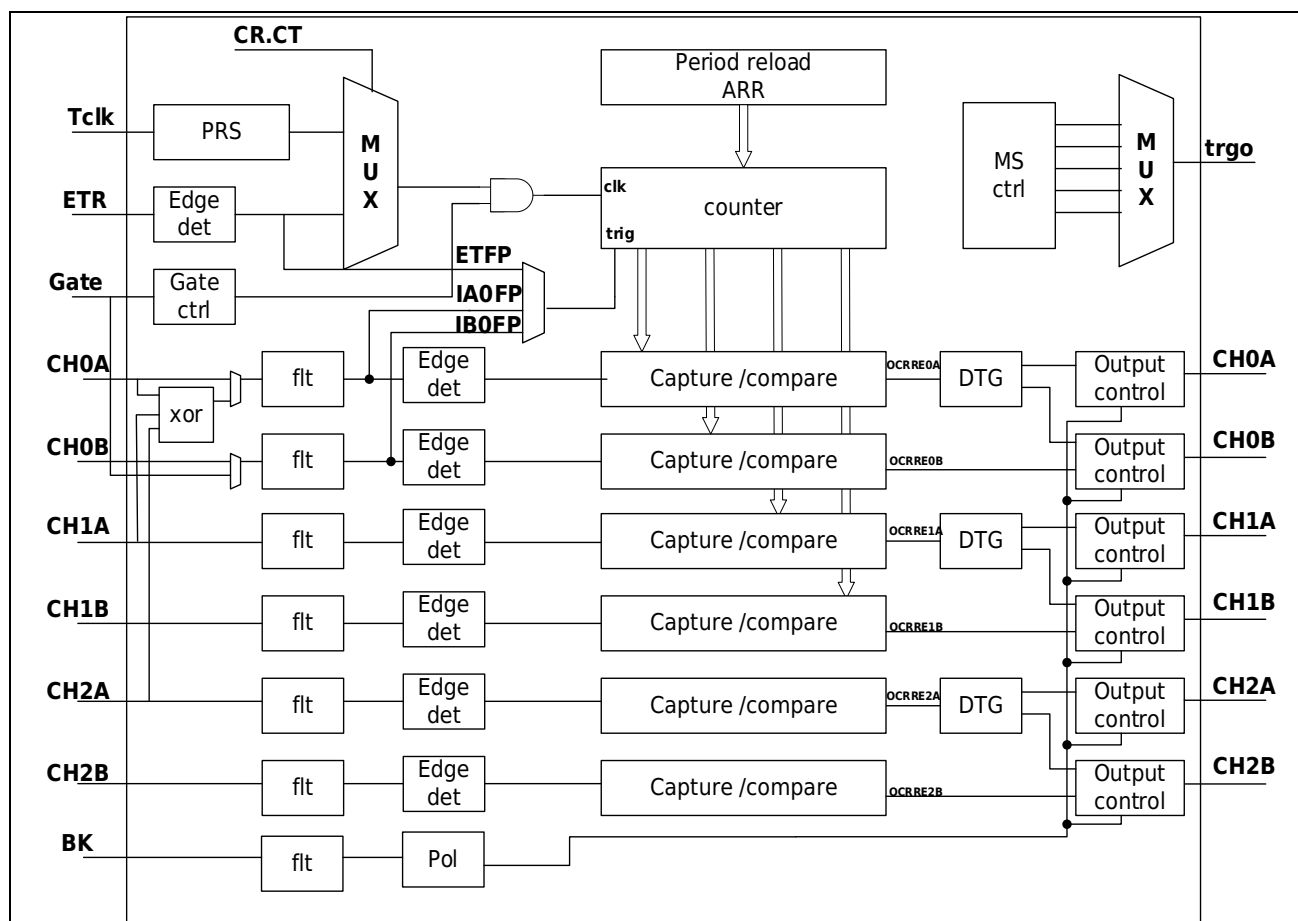


Figure 14-1 ATIM3 block diagram

14.2.2 Main features of ATIM0

- 2 independent PWM outputs CHA, CHB
- 1 channel complementary PWM output(CHA, CHB) + 1 channel independent PWM output(gate)
- 1 channel complementary PWM output(CHA, CHB) + 1 channel capture function(gate)
- Up to 2 capture inputs
- Pulse Width Measurement
- Dead zone control
- Brake control
- Edge alignment, symmetric center alignment and asymmetric center alignment PWM output
- Quadrature code counting function
- Single pulse mode
- External counting function

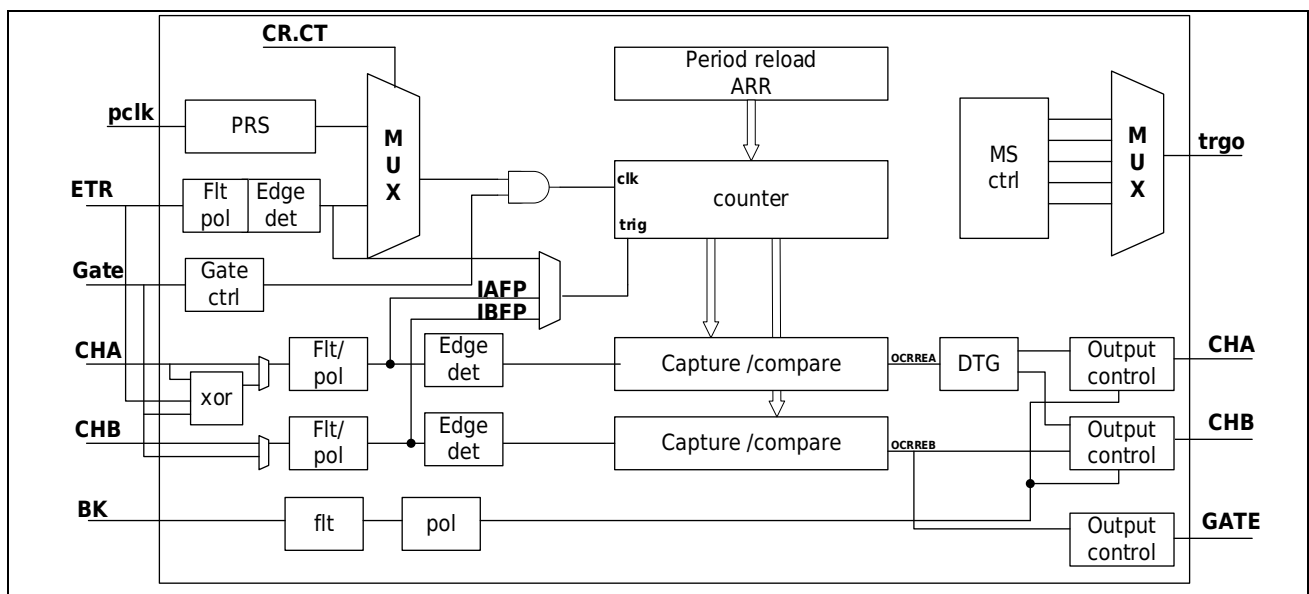


Figure 14-2 ATIM0 block diagram

14.3 Functional description

14.3.1 Timer clock

The timer uses PCLK as the timer clock, and the timer clock is marked with TCLK below.

14.3.2 Timer counter

The main building blocks of the Advanced Timer are a 16-bit counter and its associated auto-reload register. The counter can count up, count down, or alternately count up and down. The counter clock can be divided by a prescaler.

The counter, auto-reload register and prescaler register can be read and written by software. Read and write operations can be performed even while the counter is running.

14.3.3 Timer Prescaler

Using TCLK as the timer clock, prescaler can be used. The prescaler settings are as follows:

Table 14-1 ATIM PRS divider ratio list

PRS	0b000	0b 001	0b 010	0b 011	0b 100	0b 101	0b 110	0b 111
Frequency division ratio	1	2	4	8	16	32	64	256

The prescaler does not have a preloaded buffer, so any changes to the prescaler will take effect immediately.

14.3.4 Mode 0 count timer function

In this mode, the counter counts up. The counter supports two counting modes, free counting mode and reload mode, and can choose external clock ETR counting or system clock counting. Counting to the maximum overflow generates an interrupt. Invert the output port CHA, CHB, in this mode, CHA, CHB are reversed.

The free count mode overflows after counting from the set count value to 0xFFFFFFFF, and the count value restarts counting from 0x0 after overflow.

The counting range of the reload mode is from ARR counting up to 0xFFFF overflow, and then counting from ARR again, and the counting cycle is 0xFFFF-ARR+1.

14.3.4.1 Functional block diagram

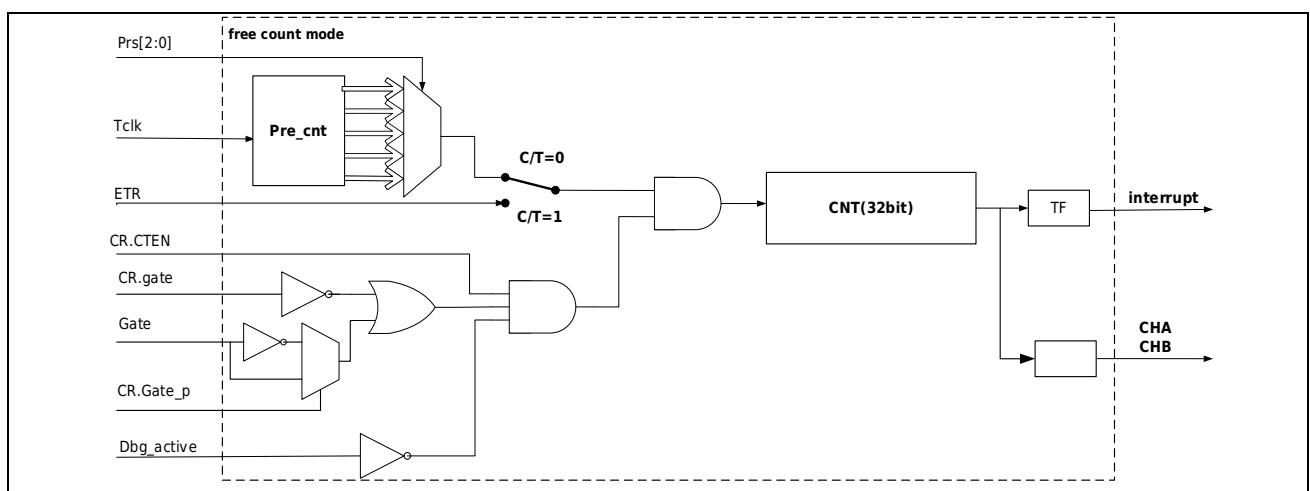


Figure 14-3 Free Counting Block Diagram

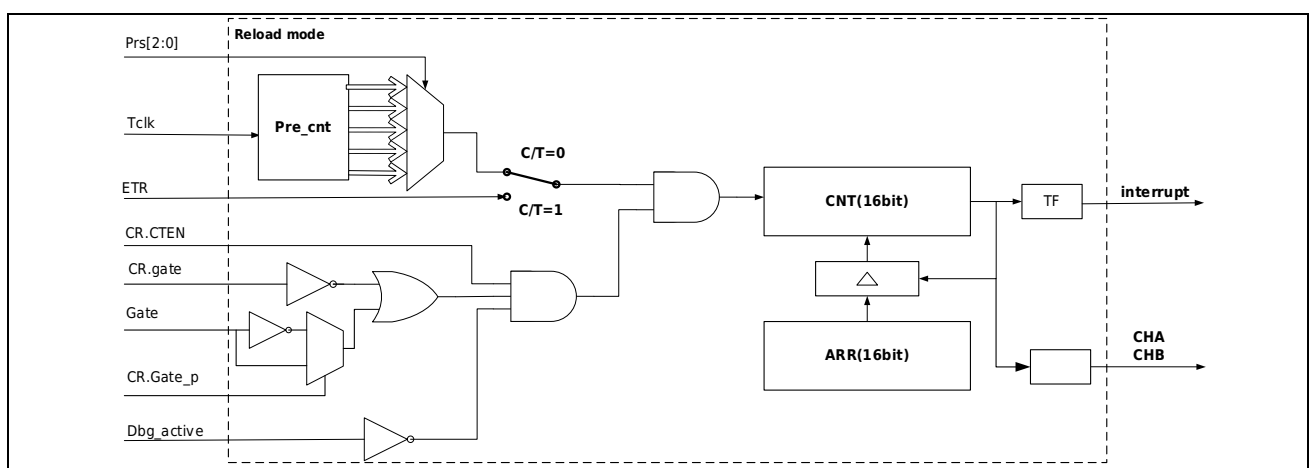


Figure 14-4 Heavy Duty Counting Waveform

14.3.4.2 Count waveform

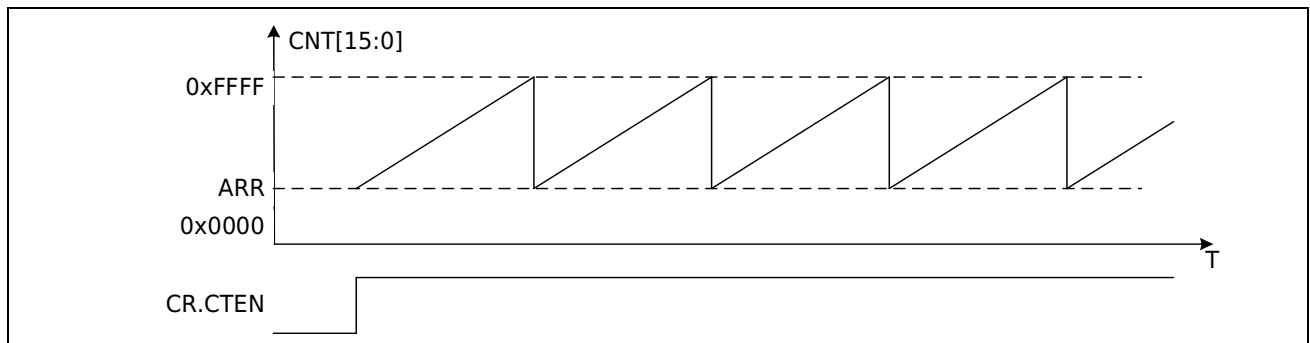


Figure 14-5 16-bit heavy load counting waveform

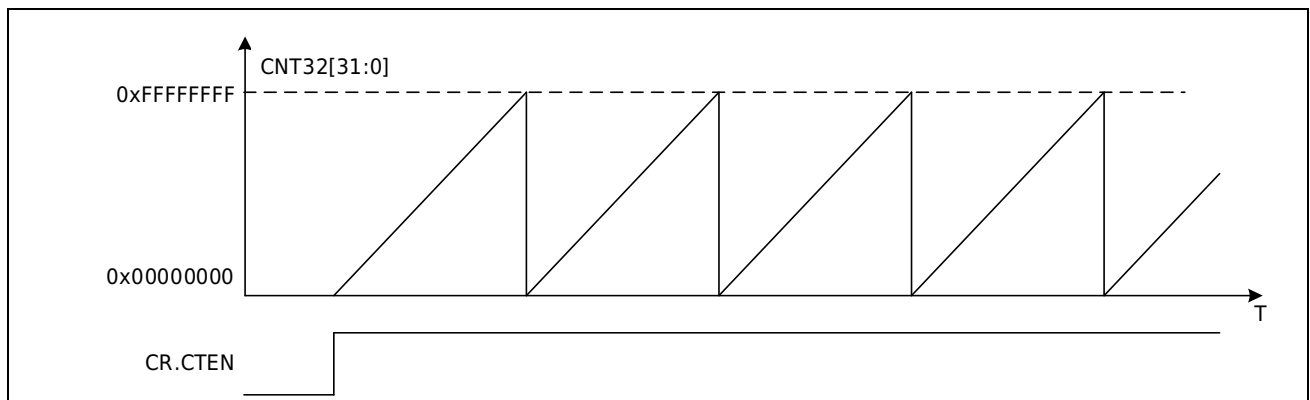


Figure 14-6 32-bit free-counting waveform

14.3.4.3 Counting function

Counting functions are used to determine the number of times an event occurs. In the count function, the counter increments every falling edge of the corresponding input clock. The input signal is sampled by the internal TCLK, so the external input clock frequency cannot exceed the system TCLK clock. Counting to the maximum value will overflow and generate an interrupt. The interrupt flag needs to be cleared by software.

14.3.4.4 Timing function

The timing function is used to generate interval timing. In the timing function, the timer has a pre-divided frequency, and the timer accumulates once per clock of each pre-divided frequency. When counting to the maximum value, it will overflow and generate an interrupt. The interrupt flag needs to be cleared by software.

14.3.4.5 Timing diagram

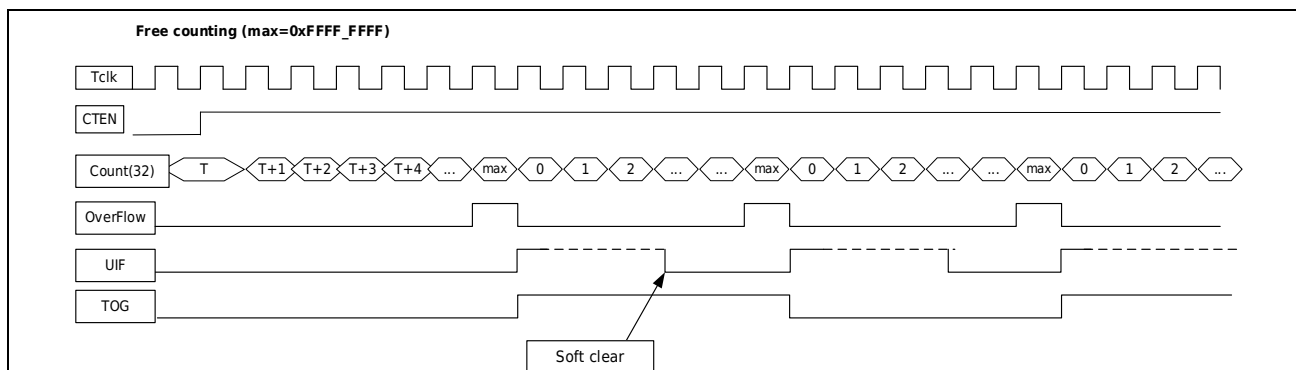


Figure 14-7 Free Count Timing Diagram

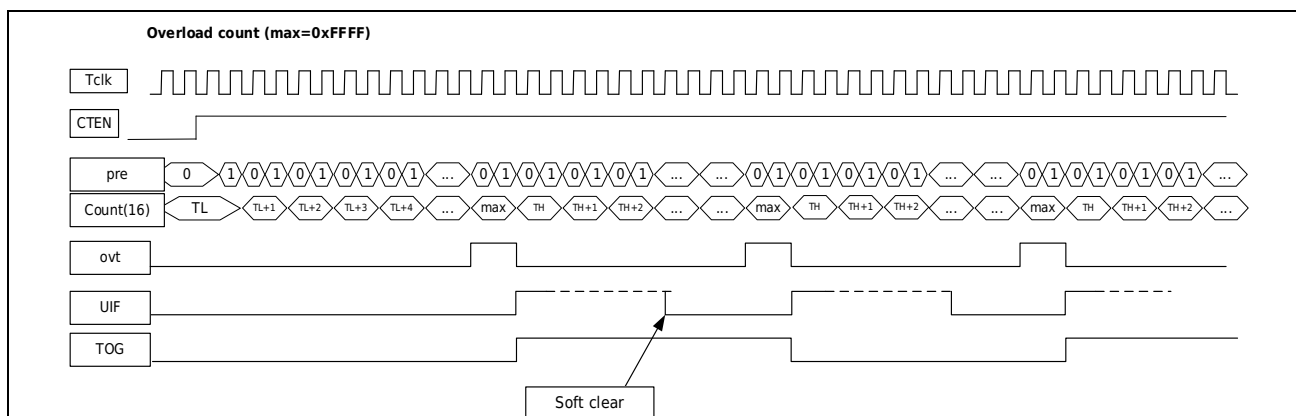


Figure 14-8 Timing diagram of heavy load counting (prescaler is set to 2)

14.3.4.6 Buzzer function

The Buzzer can be realized through the flipping output function of the timer. To use the toggle output requires enabling the DTR.MOE control bit. Setting CR.TOGEN to 0 can set port CHA, CHB output to 0 at the same time. When the counting clock is 4M, the timer reload mode configuration of Buzzer outputting different frequencies is as follows:

Table 14-2 Correspondence between Buzzer output frequency and reload mode configuration value under 4MHz clock

Buzzer frequency	Counter period	Counter value count	Counter value reload	CNTL value initial	CNTH value reload
1000Hz	0.5ms	2000	63536	0xF830	0xF830
2000Hz	0.25ms	1000	64536	0xFC18	0xFC18
4000Hz	0.125ms	500	65036	0xFE0C	0xFE0C

14.3.4.7 Setup example

Reload Timer Settings

1. Set timer mode M0CR.MODE=0

2. Set load value ARR
3. Set the counter initial value CNT
4. Clear interrupt flag
5. Enable interrupt M0CR.UIE
6. Enable reload mode M0CR.MD
7. Start timer M0CR.CTEN

BUZZER output control

1. Set the appropriate ARR value according to the output frequency
2. Set the timer to reload mode, refer to reload timer setting
3. Enable Output Enable DTR.MOE
4. Start another timer to control M0CR.TOGEN to realize frequency interval output

14.3.5 Mode 1 pulse width measurement PWC

In this mode, the high level and low level or period width of the input pulse can be automatically measured.

The first valid edge counter is initialized to 0x0001, the second valid edge will stop counting, and the current count value will be stored in CCR0A, and a pulse width measurement interrupt flag PWC will be generated. If the counter overflows, an overflow flag will be generated. Setting overflow interrupt enable will generate overflow interrupt.

Table 14-3 PWC mode measurement start and end edge configuration table

M1CR.edg1st	0	0	1	1
M1CR.edg2nd	0	1	0	1
Pulse width measurement	Upper edge ~ Upper edge Cycle width	Upper edge ~ lower edge High level width	Lower edge ~ upper edge Low level width	Lower edge ~ lower edge Cycle width

During period measurement, a period is measured at intervals of one period.

14.3.5.1 PWC functional block diagram

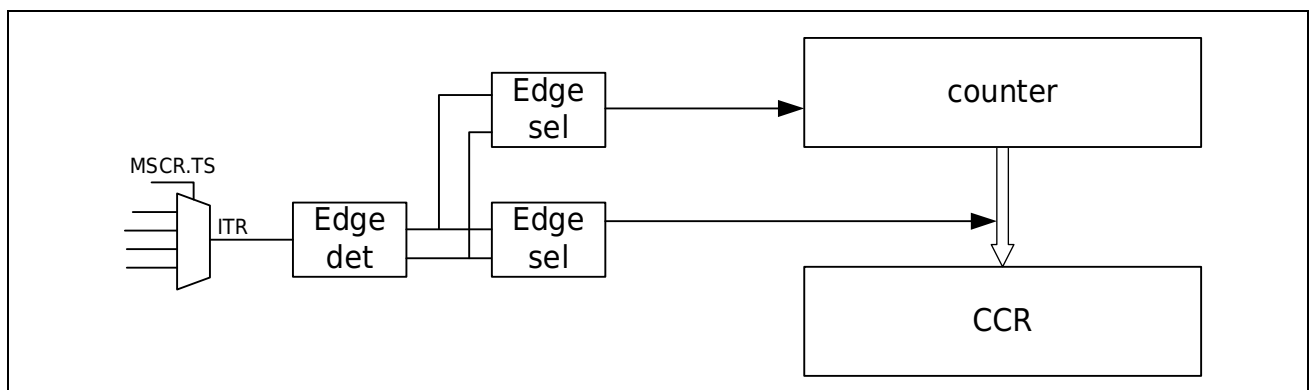


Figure 14-9 PWC Measurement Block Diagram

Table 14-4 PWC mode measurement signal source selection

MSCR.TS	Trigger selection 000: the signal ETRP after the filter phase selection of the port ETR; 001: Internal interconnection signal ITR0 010: internal interconnection signal ITR1; 011: internal interconnection signal ITR2; 100: internal interconnection signal ITR3; 101:Invalid 110: Filtered signal IAFP of port CH0A (polarity selection is invalid in pulse width measurement mode) 111: Filtered signal IBFP of port CH0B (polarity selection is invalid in pulse width measurement mode)
---------	--

14.3.5.2 PWC waveform measurement timing diagram

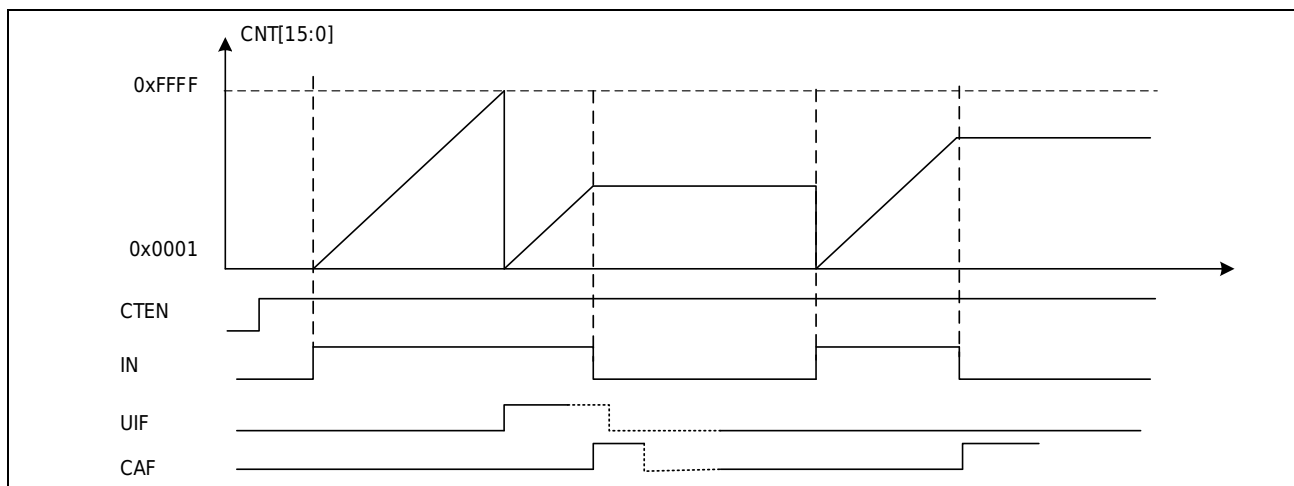


Figure 14-10 Rising edge to falling edge high level measurement

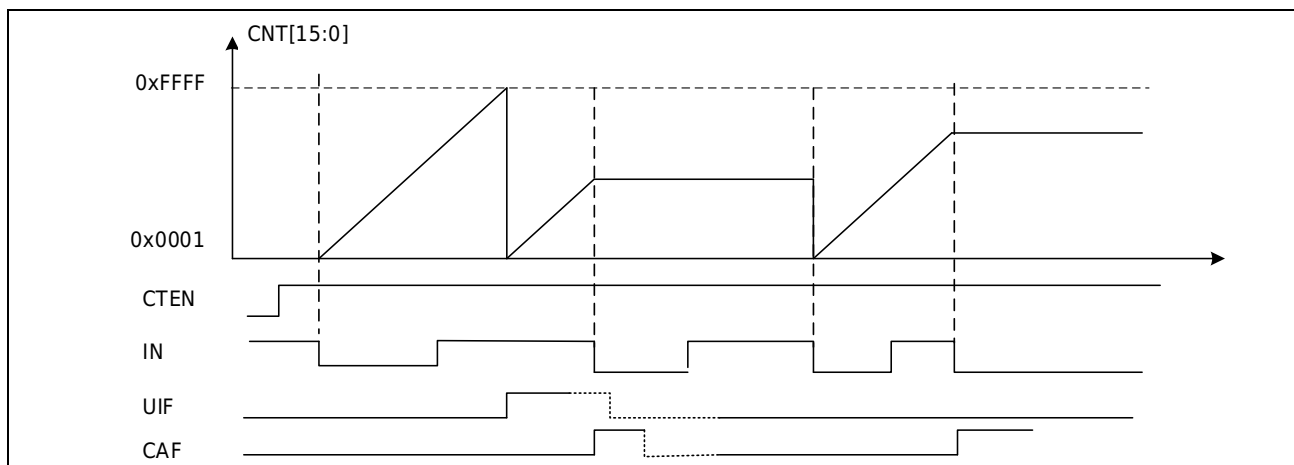


Figure 14-11 Falling edge to falling edge period measurement

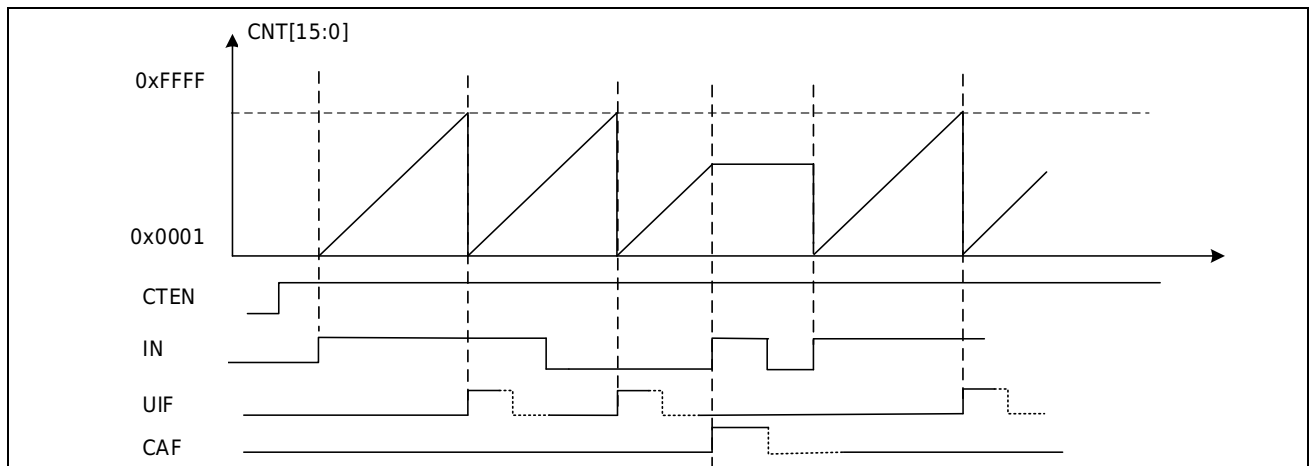


Figure 14-12 Rising edge to rising edge period measurement

Select the measurement signal source by registering MSCR.TS.

000 ETFP: ETR external input and input filtered phase selection signal, external filter and input reverse can be selected

001 ITR0: Timer internal interconnection signal 0, TRGO output of other timers

010 ITR1: Timer internal interconnection signal 1, TRGO output of other timers

011 ITR2: Timer internal interconnection signal 2, TRGO output of other timers

100 ITR3: Timer internal interconnection signal 3, TRGO output of other timers

101 IA0ED: Invalid

110 IAFP: CH0A external input and input filtered phase selection signal, external filter can be selected

111 IBFP: CH0B external input and input filtered phase selection signal, external filter can be selected

14.3.5.3 PWC one-shot mode

Set M1CR.ONESHOT=1 to set PWC single measurement, and CTEN will be cleared after the measurement is completed.

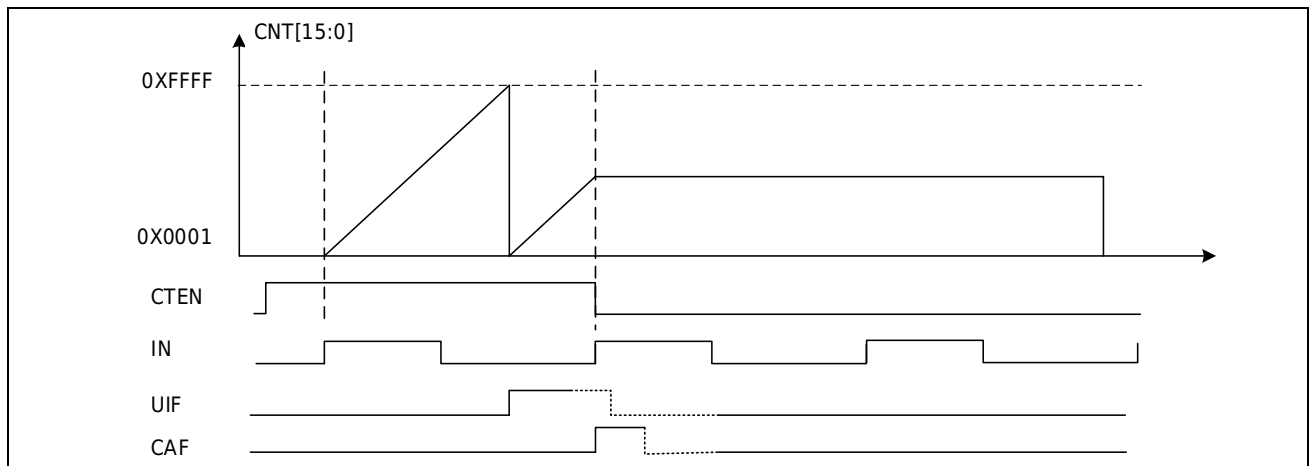


Figure 14-13 Rising edge-to-rising edge period measurement one-shot mode

14.3.5.4 Setup example

Pulse Low Level Measurement Setup

1. Set to pulse measurement mode M1CR.MODE=1
2. Set MSCR.TS to select the signal to measure
3. Set M1CR.edg2dn=0, M1CR.edg1st=1 to choose to measure low level
4. Clear interrupt flag
5. Enable overflow interrupt M1CR.UIE
6. Enable measurement end interrupt CR0.CIEA
7. Enable timer M1CR.CTEN
8. Read CCR0A and the number of overflows in the interrupt service routine and clear the interrupt flag
9. Waiting for next measurement

Pulse high level single measurement setup

1. Set to pulse measurement mode M1CR.MODE=1
2. Set MSCR.TS to select the signal to measure
3. Set pulse single measurement mode M1CR.ONESHOT=1
4. Set M1CR.edg2dn=1, M1CR.edg1st=0 to choose to measure high level
5. Clear interrupt flag
6. Enable overflow interrupt M1CR.UIE
7. Enable measurement end interrupt CR0.CIEA
8. Enable timer M1CR.CTEN

9. Read CCR0A and the number of overflows in the interrupt service routine and clear the interrupt flag
10. End of measurement

14.3.6 Mode 2/3 compare capture mode

14.3.6.1 Counter

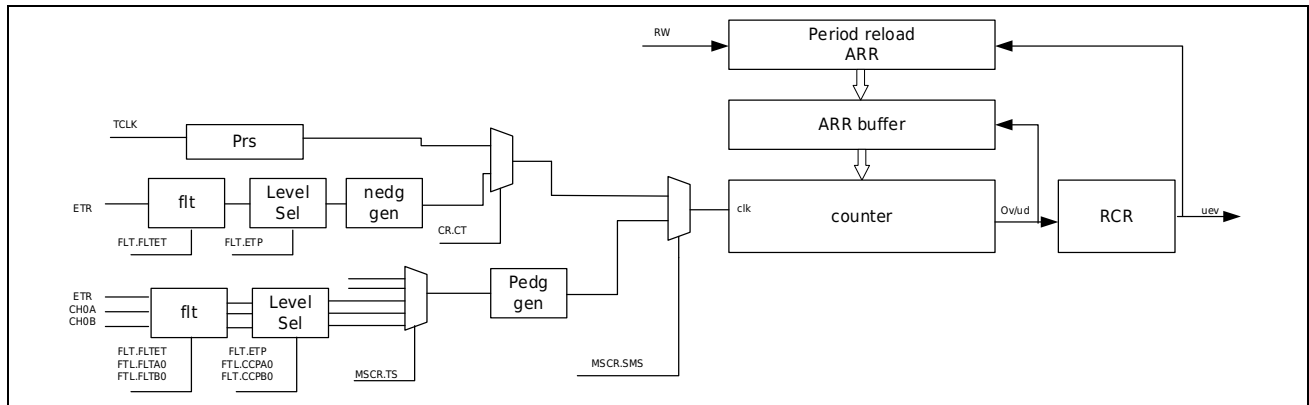


Figure 14-14 Counter Block Diagram

The main part of the counter is a 16-bit counter with associated autoloader registers. This counter can count up (mode 2), count down (mode 2) or count up and down in both directions (mode 3). The clock of the counter can be divided by the prescaler PRS, or ETR can be selected to input an external clock or the external input signal and internal interconnection signal selected by MSCR.TS.

Counter basic units include:

- Counter Register CNT
- Prescaler register CR.PRS
- Automatic Reload Register ARR
- Repeat Count Register RCR
- Clock Selection Control Registers FLT, CR0, MSCR, CR

The automatic reload register ARR has a cache function, and the reload value is updated from the cache register to the counter after the counter generates an event update. When the counter is stopped or the cache function is disabled, the auto-reload register is immediately updated to the cache register. When the timer is running and the cache function is valid, the value written to the auto-reload register will not be updated to the cache register immediately, and will be updated from the auto-reload register to the cache register only after the event is updated.

Clock selection and gating function, trigger function, and reset function refer to the [Mode 2/3 Slave Mode] chapter.

14.3.6.2 Counter waveform

Mode 2 is a sawtooth counting waveform, and the counting direction can be changed by setting CR.DIR.

CR.DIR is set to 0, the counter is in up-counting mode. In this mode, the counter counts from 0 to the auto-reload value(ATIMx_ARR), and then starts counting from 0 again and generates a counter overflow event. (UEV) is generated when the count-up repeats to the number programmed in the repeat count register plus one (ATIMx_RCR+1). Otherwise, an update event will be generated every time the counter overflows.

generated when the UG bit of the ATIMx_CR register is set (by software or using a slave mode controller).

When an update event occurs, all registers are updated and the update flag(UIF bit in the ATIMx_IFR register) is set(depending on the URS bit):

- The auto-reload cache value will be updated with the ARR register value
- The compare buffer value will be updated with the compare register CCRxy

The figure below shows the counter waveforms of different counting directions when ARR=0x2C:

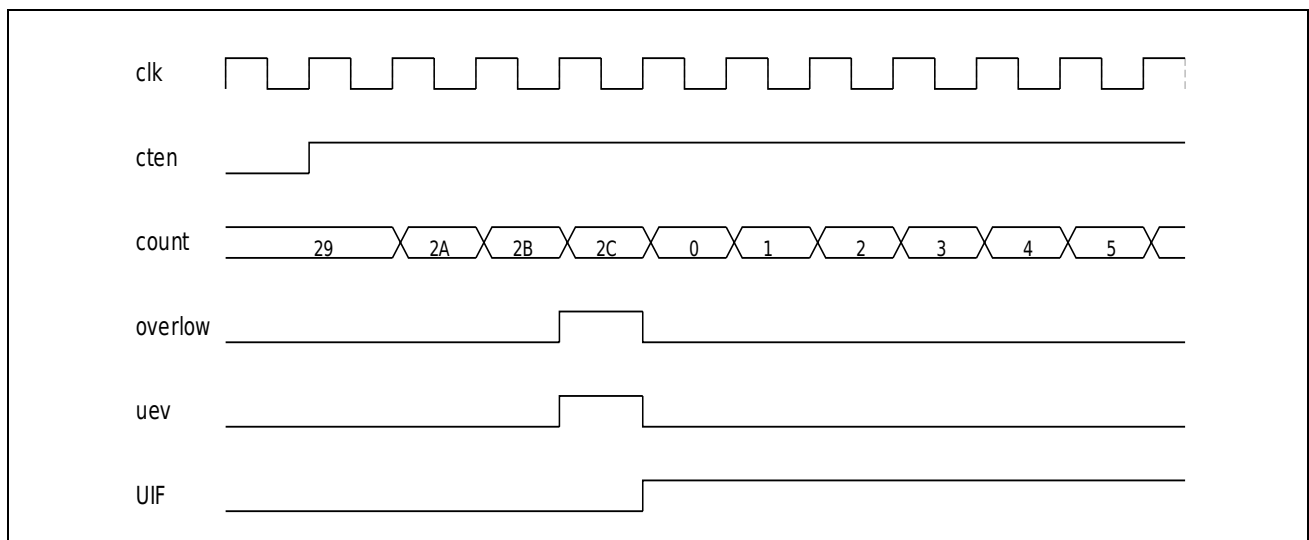


Figure 14-15 Count Up Without Prescaler

The number of counting overflow cycle clocks is ARR+1,

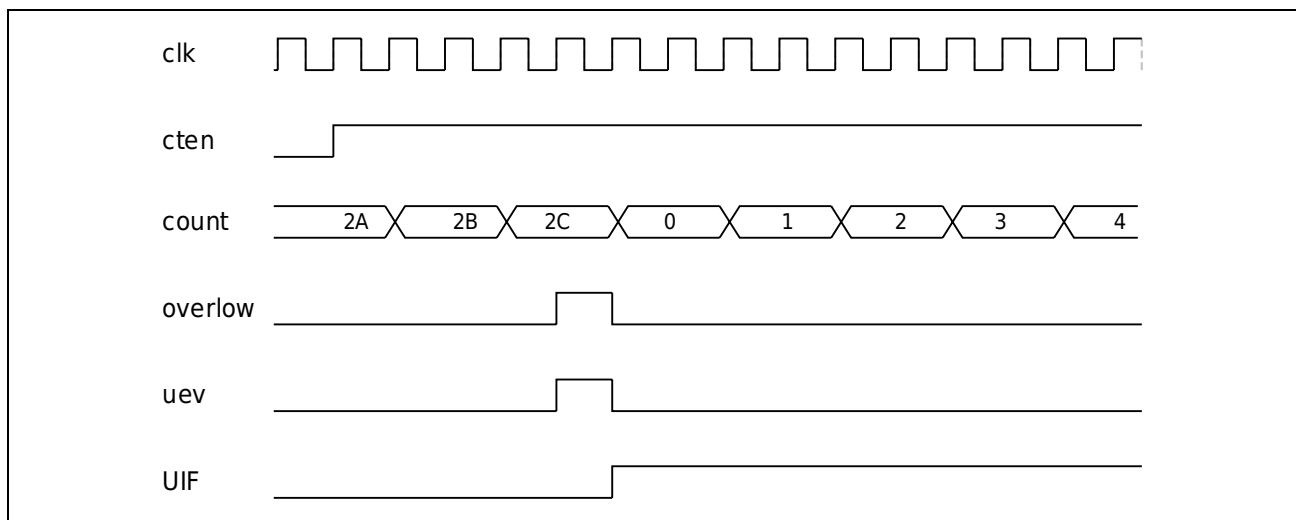


Figure 14-16 Up Counting with Prescaler

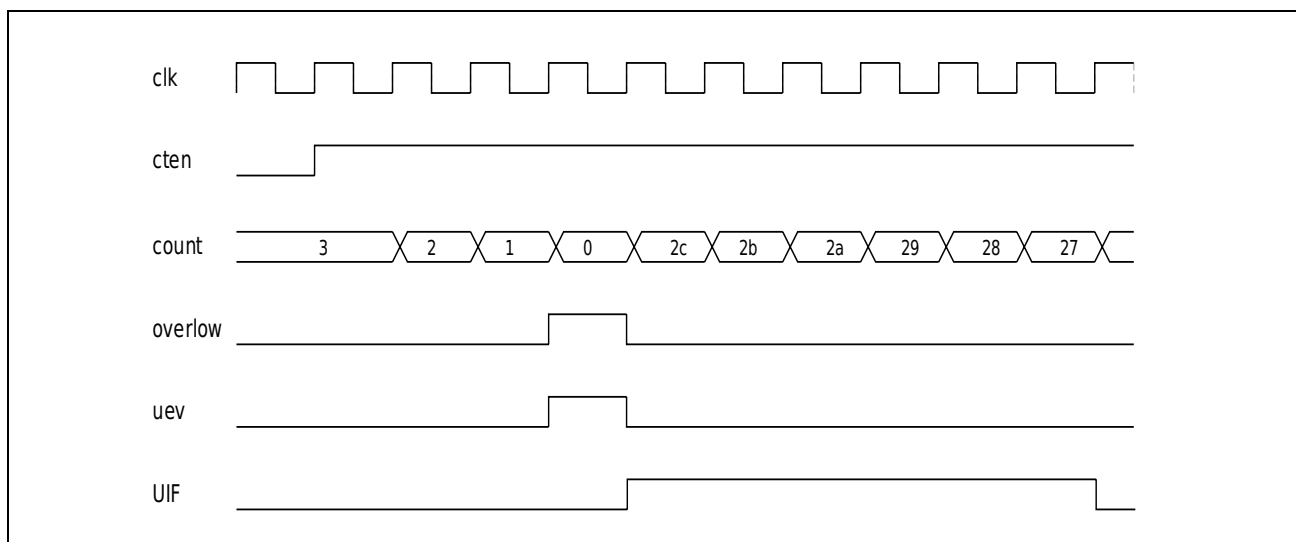


Figure 14-17 Down counting without prescaler

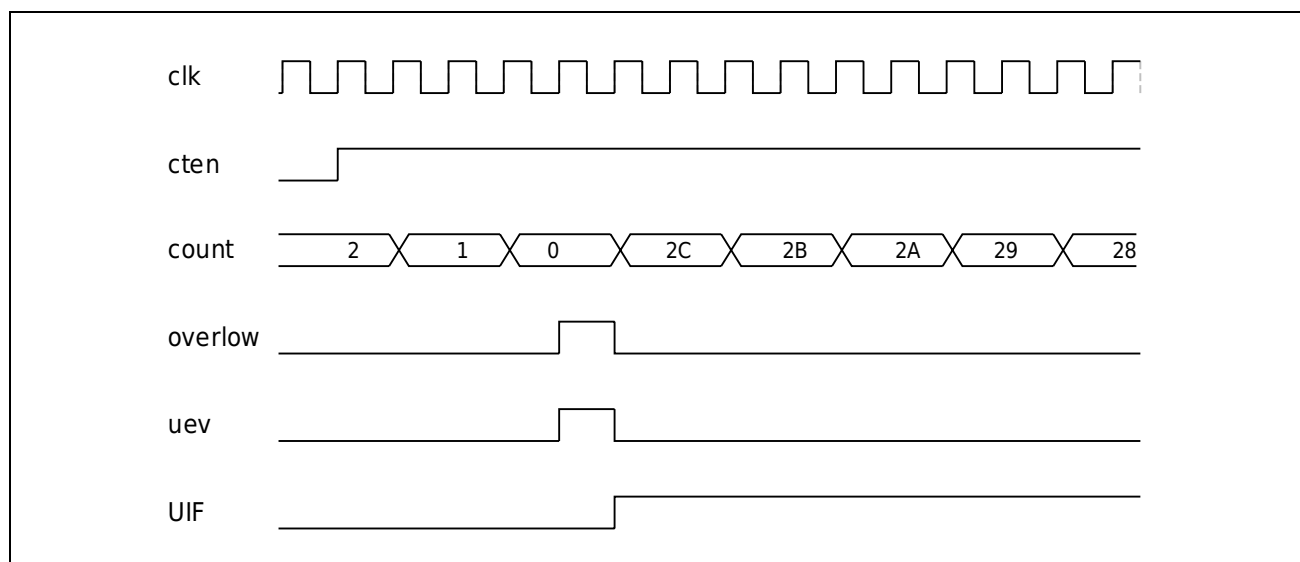


Figure 14-18 Down counting with prescaler

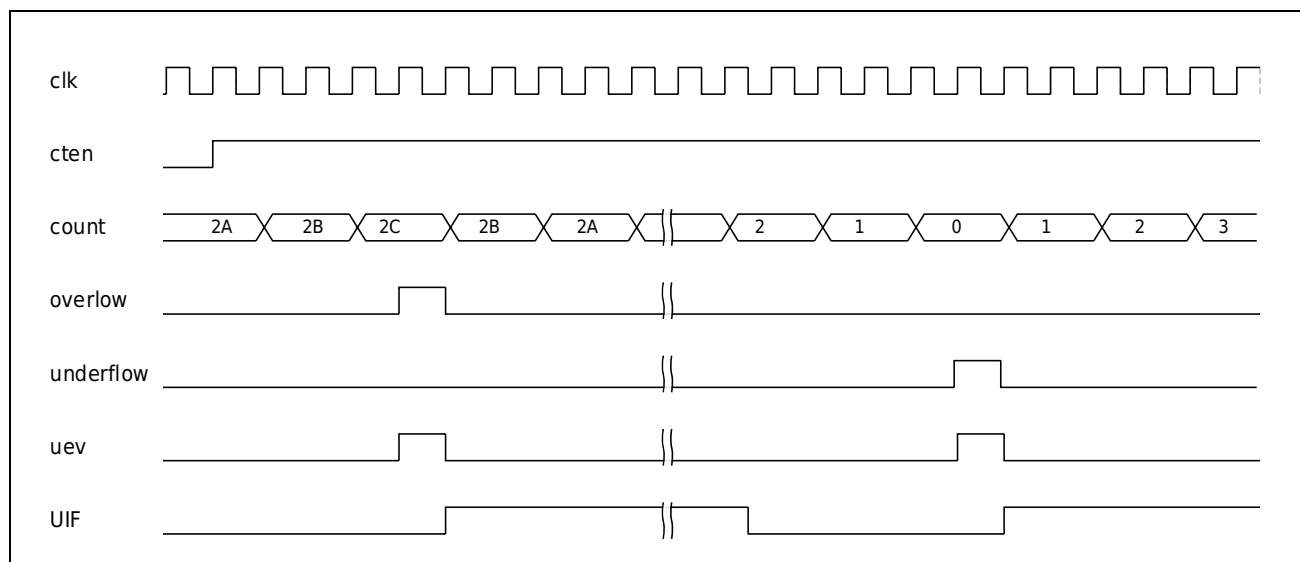


Figure 14-19 Up and down counting with prescaler

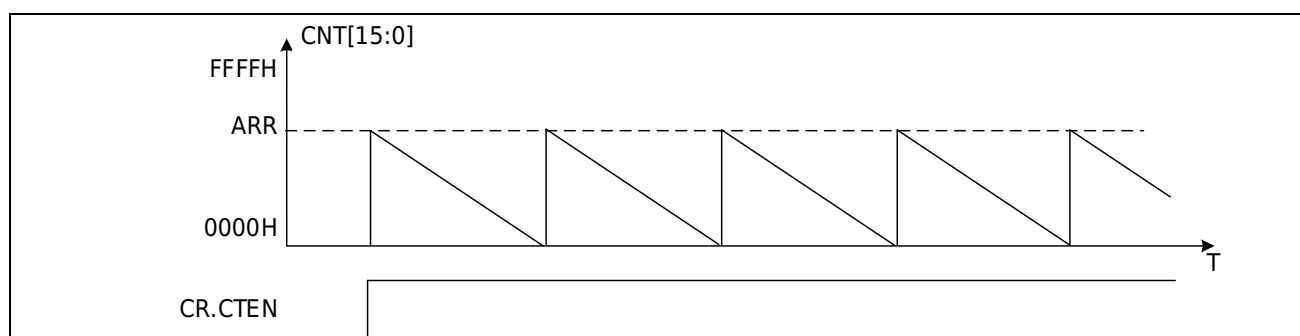


Figure 14-20 Edge-Aligned Timer Waveform (DIR =1)

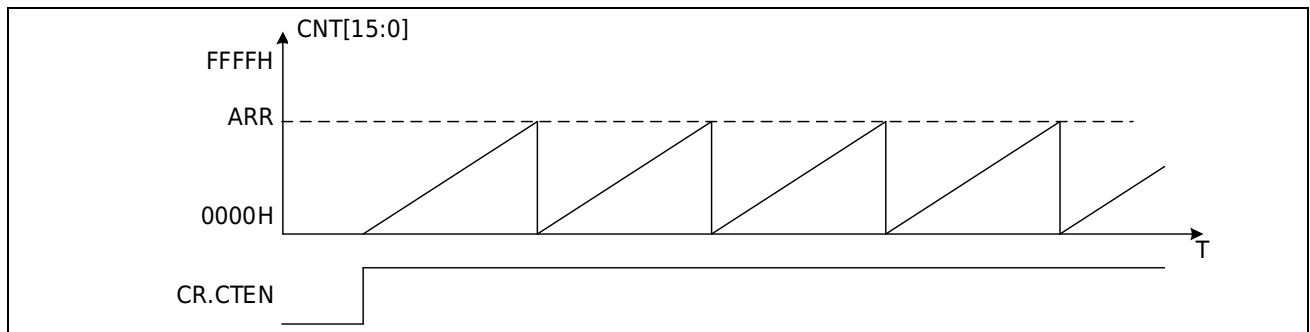


Figure 14-21 Edge-Aligned Timer Waveform (DIR = 0)

Mode 3 is a triangular wave counting waveform, the counting direction control bit is read-only, and the counting direction cannot be changed.

The CR.DIR direction bit is read-only in center-aligned (triangular wave) mode, and the write value is invalid. Switching from other modes to center-aligned mode DIR is automatically cleared to 0. DIR is automatically cleared in reset mode by software event update and external trigger in slave mode.

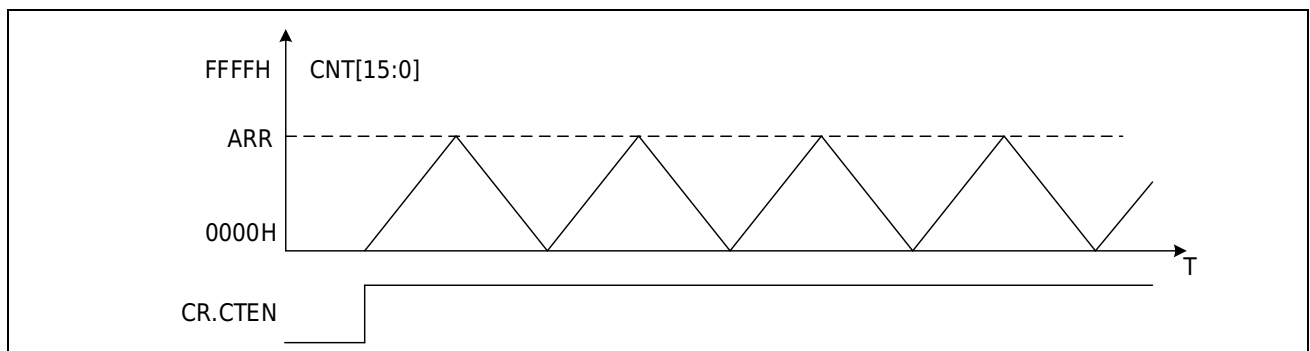


Figure 14-22 Center Aligned Counter Waveform

14.3.6.3 Repeat count

A repeating counter uses the counter's overflow to count down. counting to 0, that is, when the counter overflows once the value set by the repeat register is added. When the cache register is enabled, the auto-reload register is updated to the auto-reload cache register. In compare mode, the value of the compare register is updated to the compare buffer register.

The repetition counter is decremented when the following conditions are true (when RCR.UD = 0 and RCR.OV = 0):

- Each time the counter overflows in up counting mode
- Each time the counter underflows in down counting mode
- Every overflow and every underflow in triangle wave mode

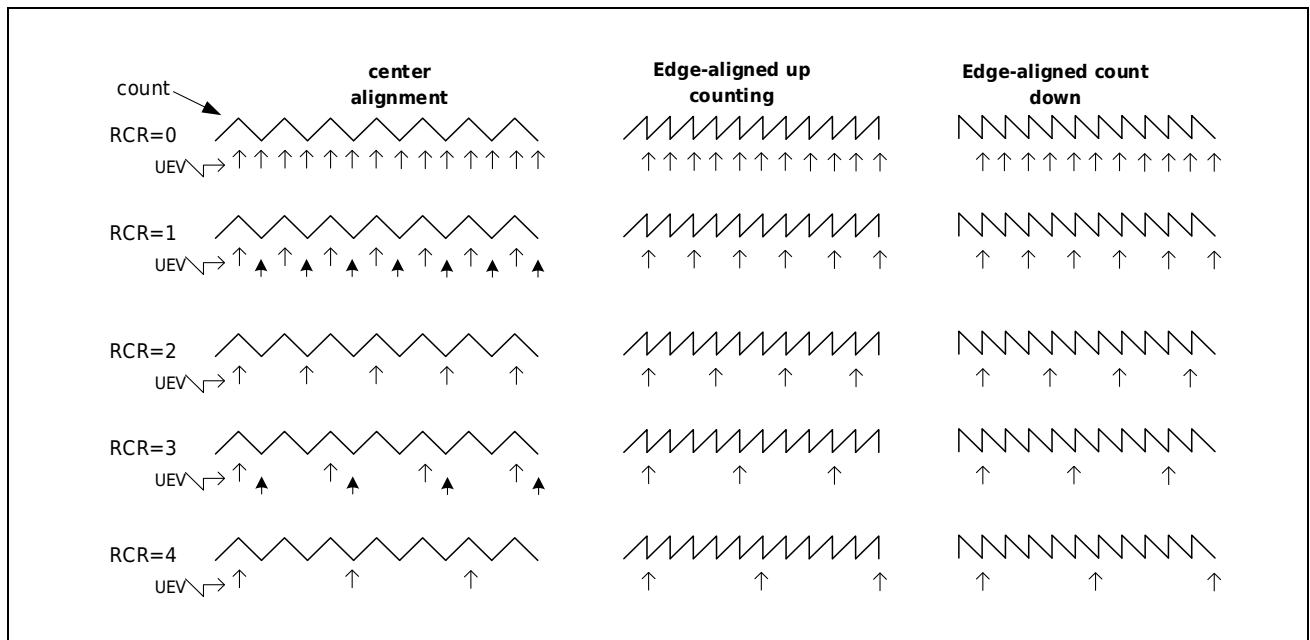


Figure 14-23 Repeat counter generates update timing (RCR.UD=0 RCR.OV=0)

RCR.OV is the masking function bit for repeat counting control overflow. If RCR.OV is 1, the overflow event will be ignored when performing repeat counting;

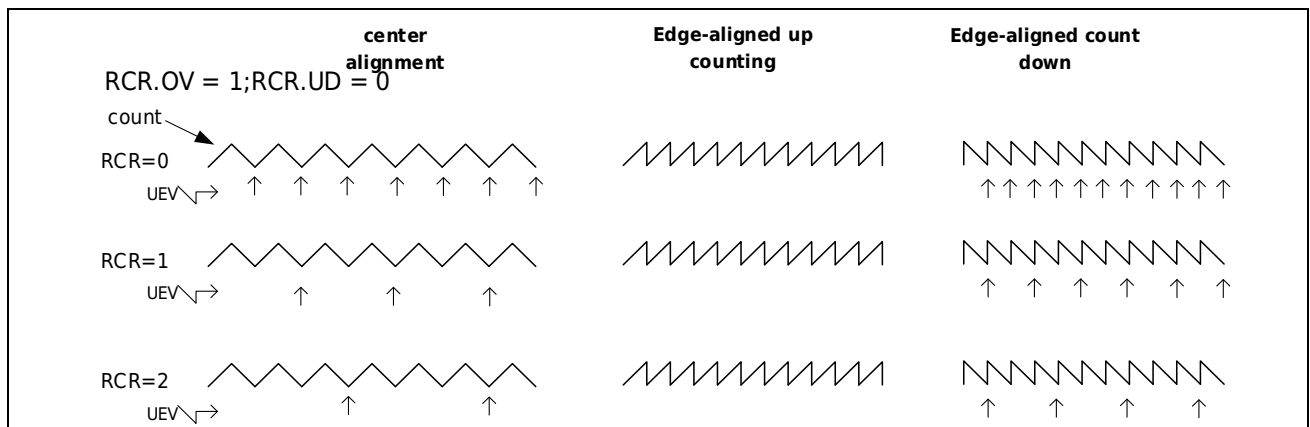


Figure 14-24 When the overflow is masked, the repetition counter generates an update timing (RCR.OV=1, RCR.UD=0)

RCR.UD is the underflow mask function bit for repeat counting control. If RCR.UD is 1, the underflow event will be ignored when performing repeat counting;

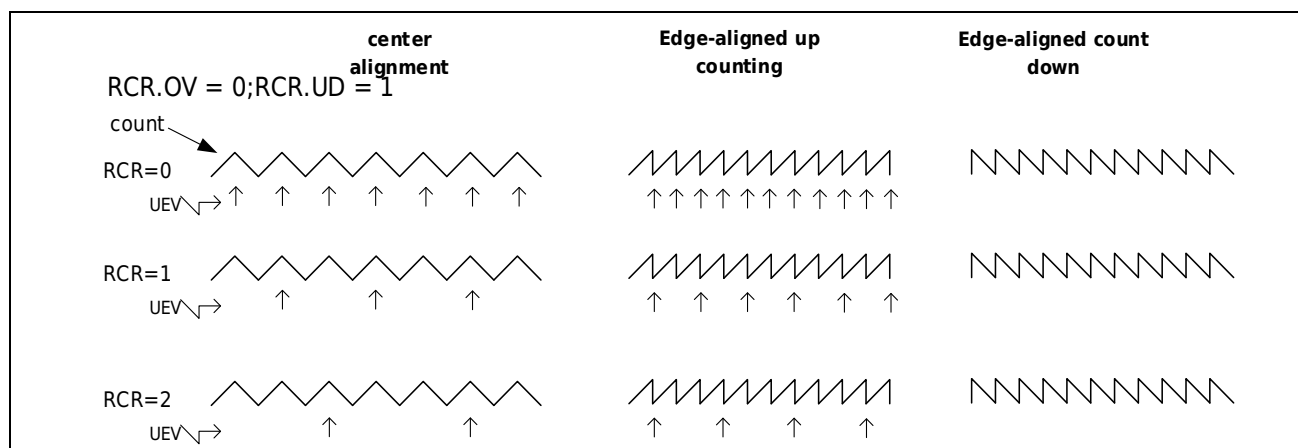


Figure 14-25 When the underflow is masked, the repetition counter generates an update timing(RCR.OV=0, RCR.UD=1)

In addition to generating events to update UEV through repeated counters on overflow and overflow, you can also generate software and slave mode reset events to update UEV by writing register CR.UG; at this time, you need to configure CR.URS.

14.3.6.4 Data cache

Both the auto-reload register ARR and the comparison register CCRxy can be configured with a cache function. When the cache function is valid, the written period value ARR and comparison value CCRxy will only take effect when the UEV event is updated.

The update timing diagram of the auto-reload value in different counting modes is as follows:

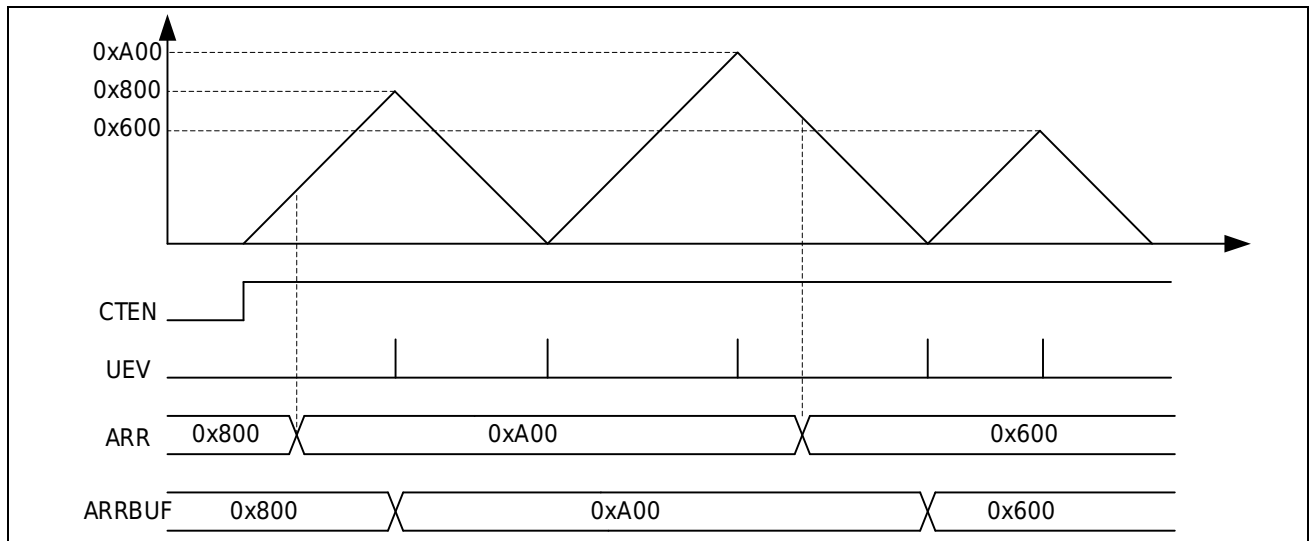


Figure 14-26 Triangular Wave Mode Buffer Enable

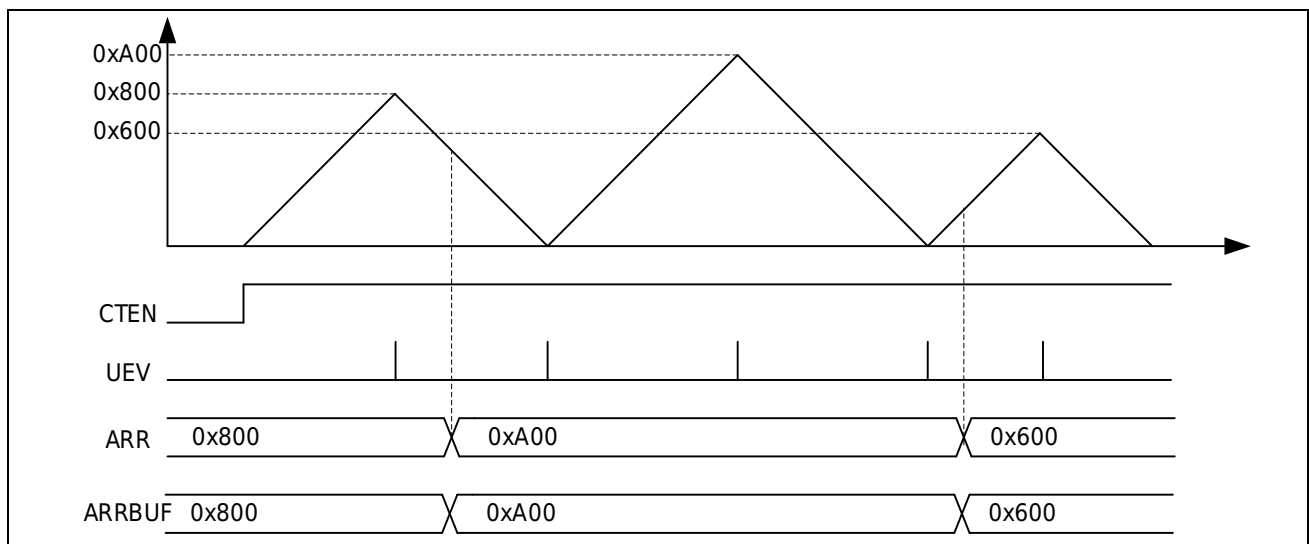


Figure 14-27 Triangular Wave Mode Buffer Invalidation

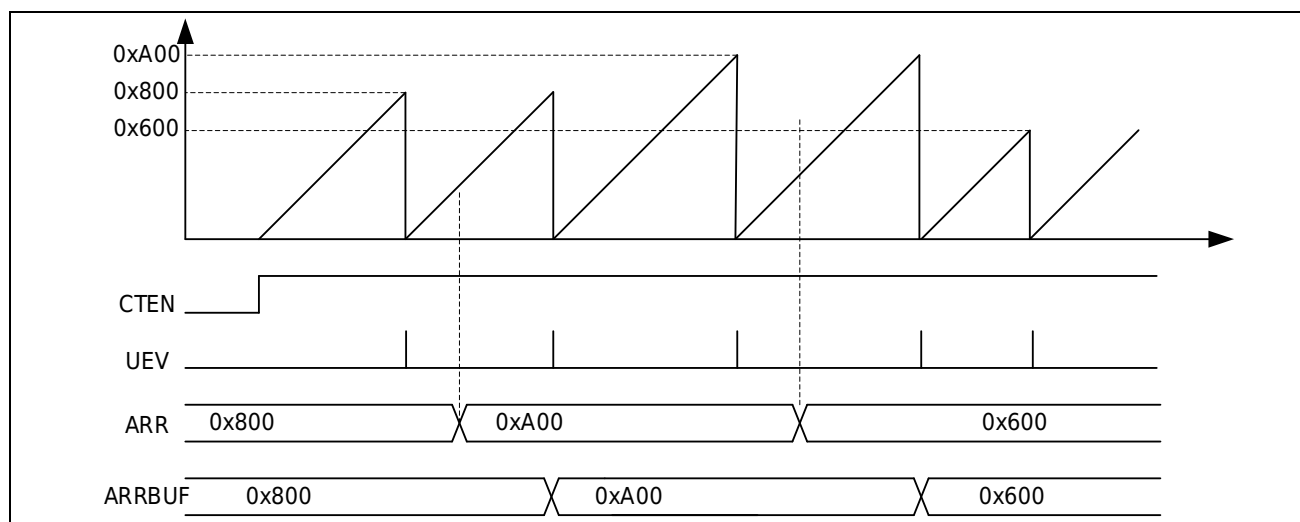


Figure 14-28 Count buffer enable on sawtooth mode

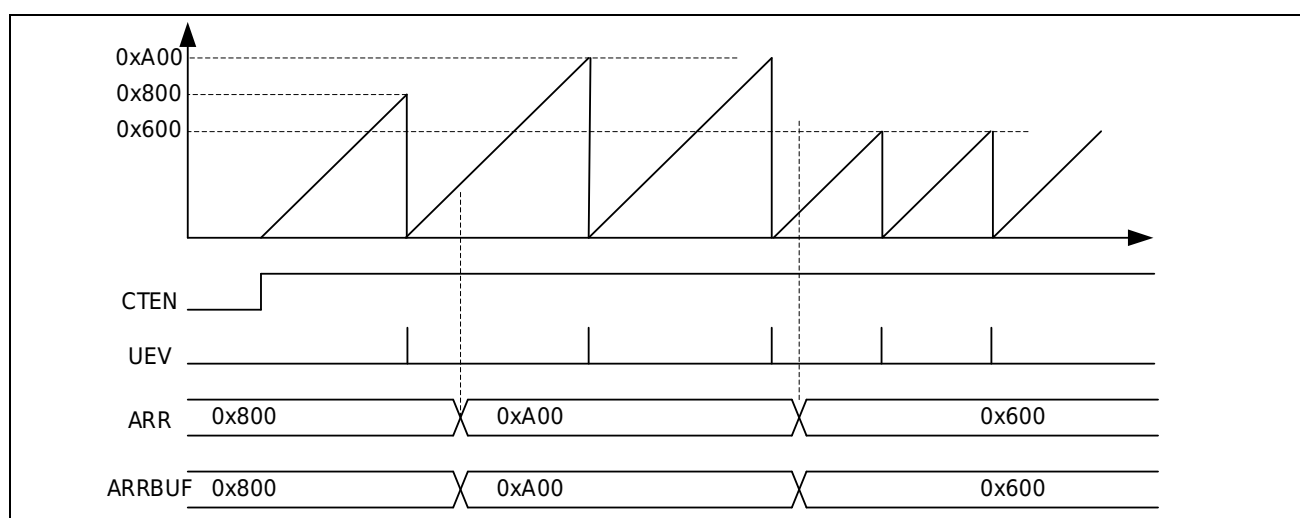


Figure 14-29 Count Buffer Invalidation on Sawtooth Mode

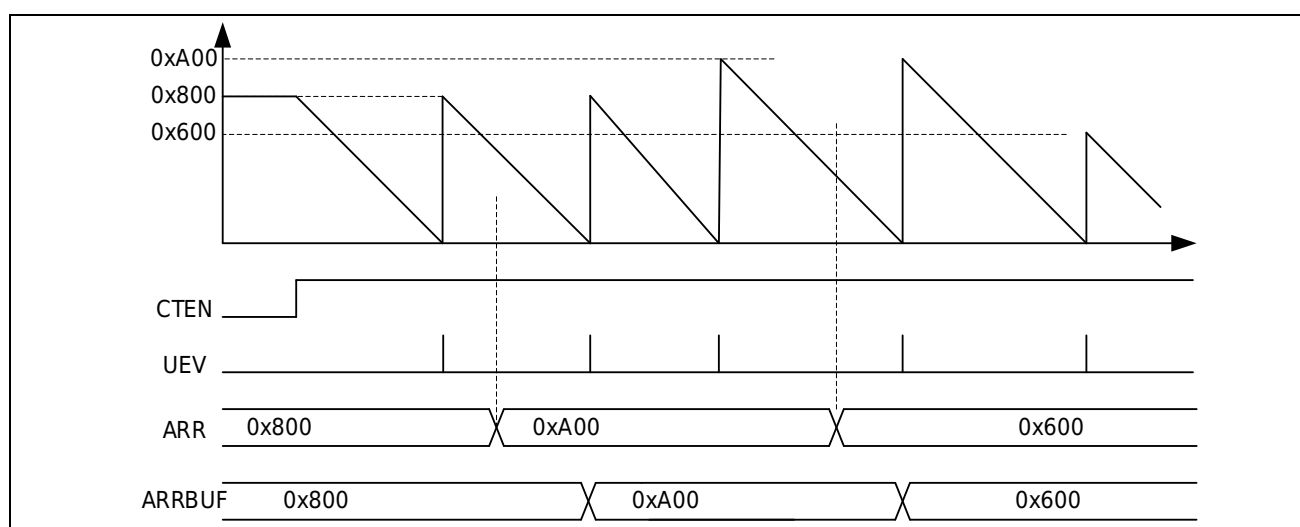


Figure 14-30 Count buffer enable in sawtooth mode

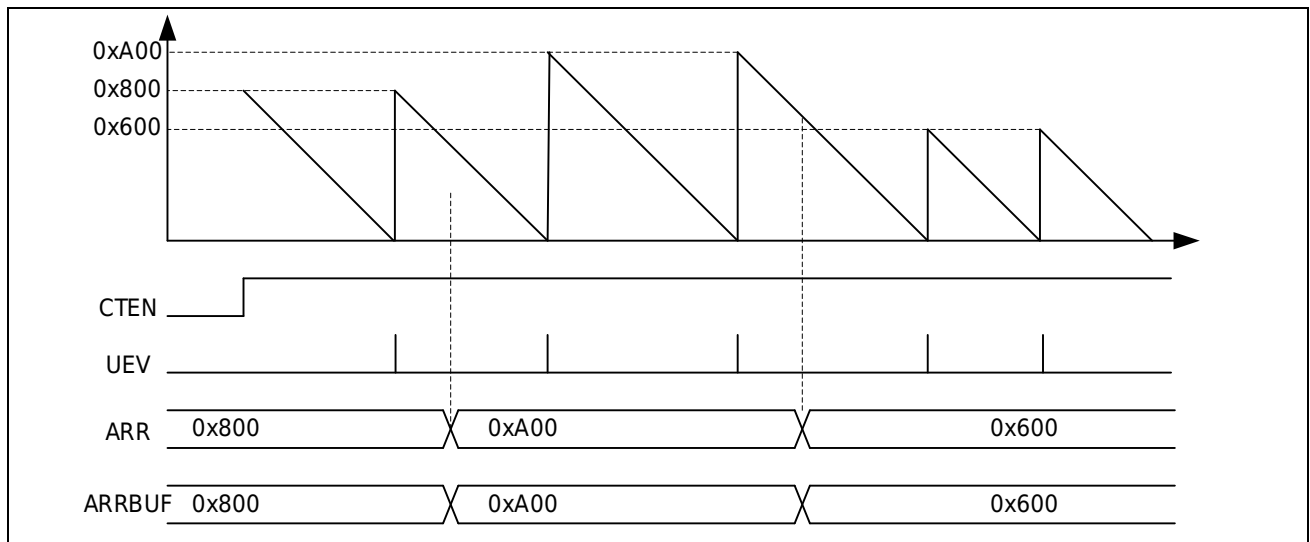


Figure 14-31 The counter buffer is invalid in sawtooth mode

Note: In triangular wave mode and sawtooth wave counting mode, if the cache is disabled, when changing ARR, the current counter value should be less than the ARR cycle value to be changed, otherwise the current cycle will count to 0xFFFF.

The update status of the comparison cache is consistent with that of the periodic cache, and the sequence diagrams are not listed here.

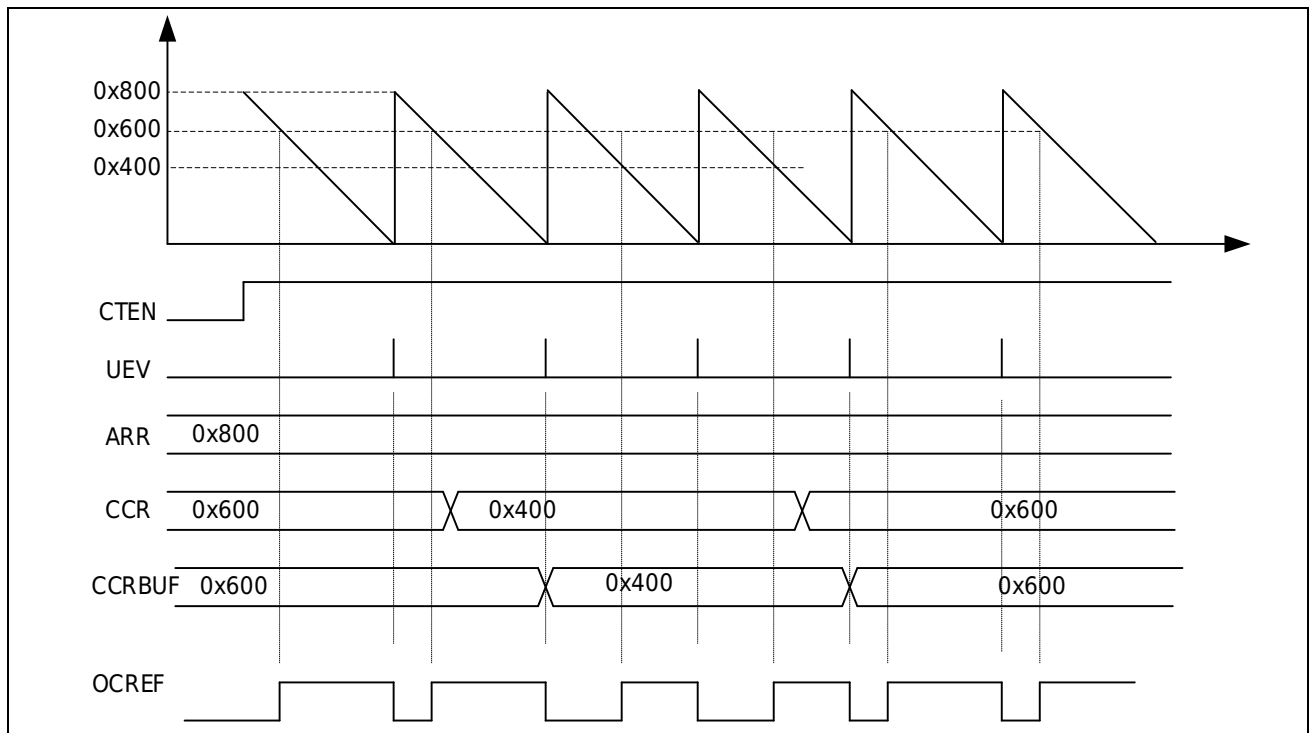


Figure 14-32 Count compare buffer enable in sawtooth mode

14.3.6.5 Compare output OCREF

The comparison output OCREFA can be configured as a single-point comparison, and the comparison register CCRA is used to control the output of OCREFA; the comparison output of OCREFA can also be configured as a two-point comparison, and the comparison registers CCRA and CCRB are used to control the comparison output of OCREFA.

OCREFB can only use single-point comparison, and the comparison register CCRB is used to control the comparison output of OCREFB.

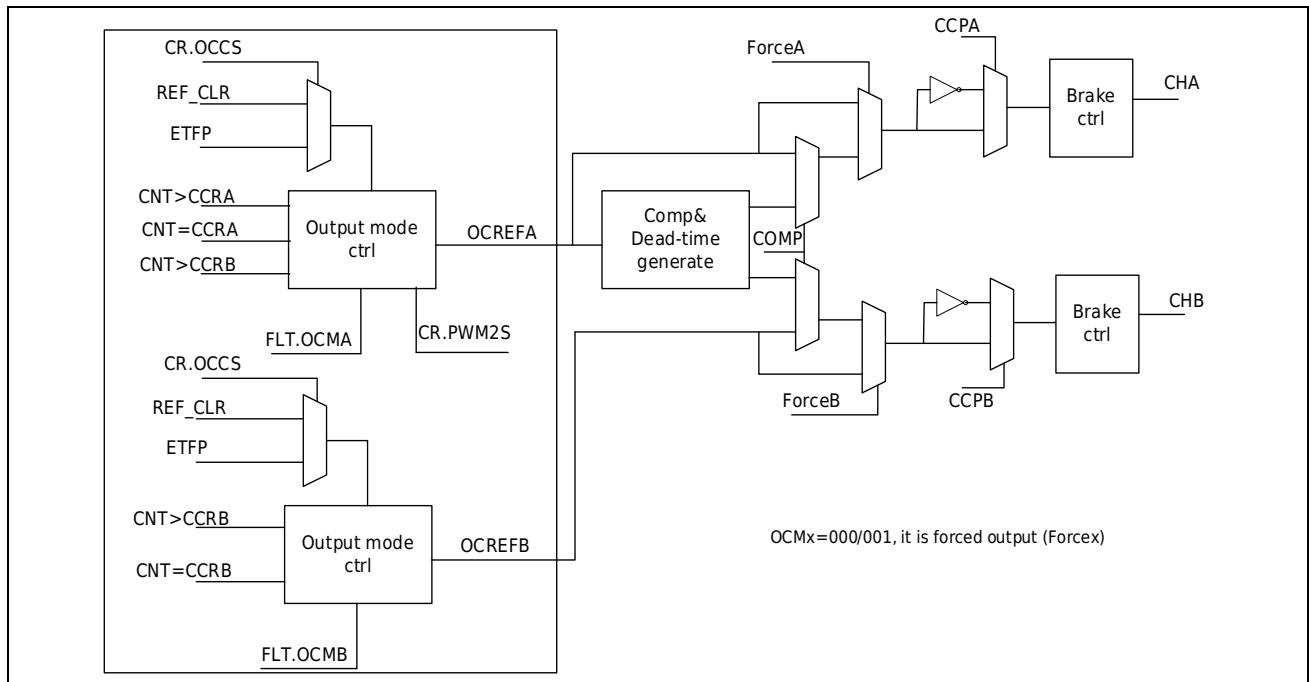


Figure 14-33 OCREF output block diagram

OCREF output is selected using OCMx

000: forced to 0

001: Forced to 1

010: Forced to 0 when comparing matches

011: Forced to 1 when comparing matches

100: flip when comparison matches

101: Output a high level for one count period when comparing matches

110: PWM mode 1

Single point comparison:

When counting up, $CNT < CCR_{xy}$ outputs high, and when counting down, $CNT > CCR_{xy}$ outputs low level

Two-point comparison:

- 1) Counting $CCRxA < CNT \leq CCRxB$ output on the sawtooth wave is low level
- 2) Counting under sawtooth wave $CCRxA < CNT \leq CCRxB$ output is high level
- 3) Triangular wave up count $CNT < CCRxA$ output high, down count $CNT > CCRxB$ is low level

111: PWM mode 2

Single point comparison:

When counting up, $CNT < CCRxy$ output is low, when counting down, $CNT > CCRxy$ output is high level

Two-point comparison:

- 1) Counting $CCRxA \leq CNT < CCRxB$ output on the sawtooth wave is high level
- 2) Counting under sawtooth wave $CCRxA \leq CNT < CCRxB$ output is low level
- 3) Triangular wave up count $CNT < CCRxA$ output low, down count $CNT > CCRxB$ is high level

Note: Forced output has high priority, when forced output is valid, complementary output control is invalid.

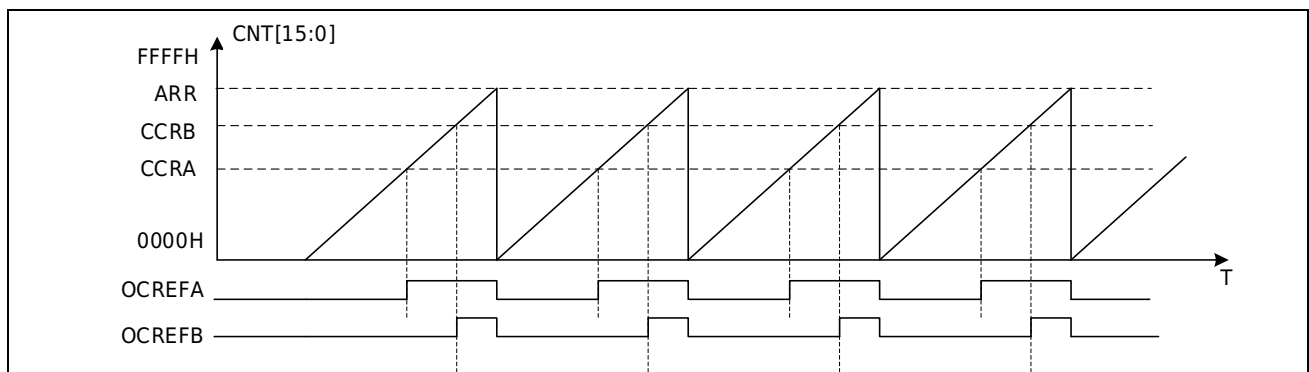


Figure 14-34 Sawtooth counting single point comparison OCREF output waveform(OCMx=111)

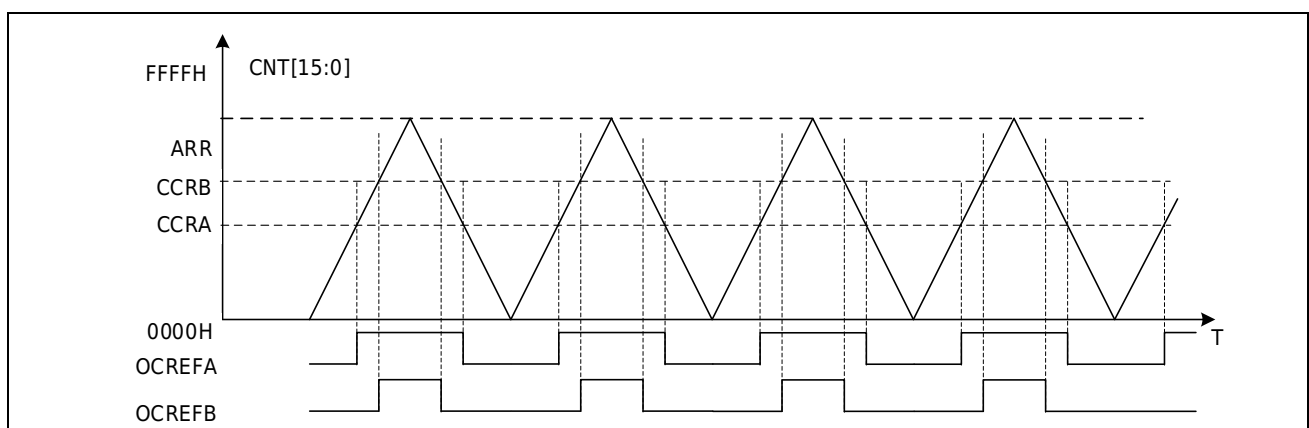


Figure 14-35 Triangular wave counting single point comparison OCREF output waveform (OCMx=111)

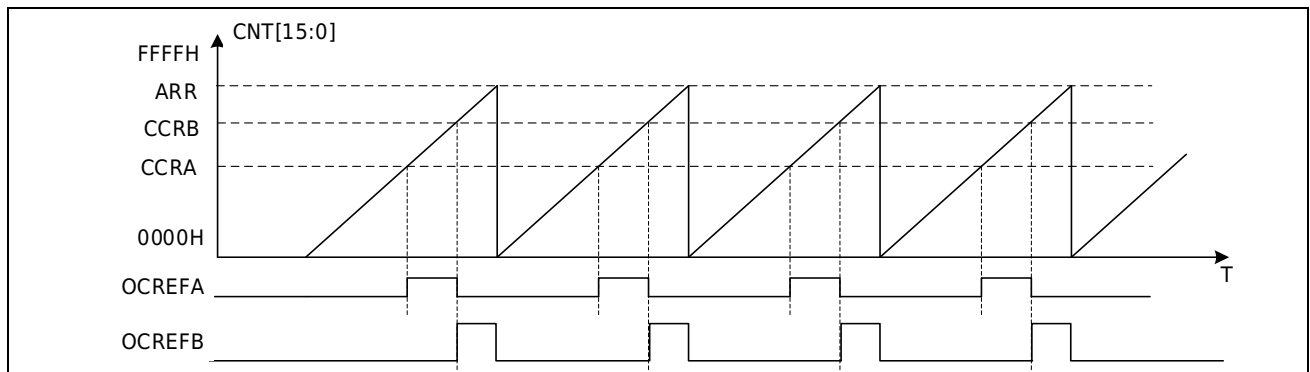


Figure 14-36 Sawtooth wave counting double-point comparison OCREF output (OCMx=111)

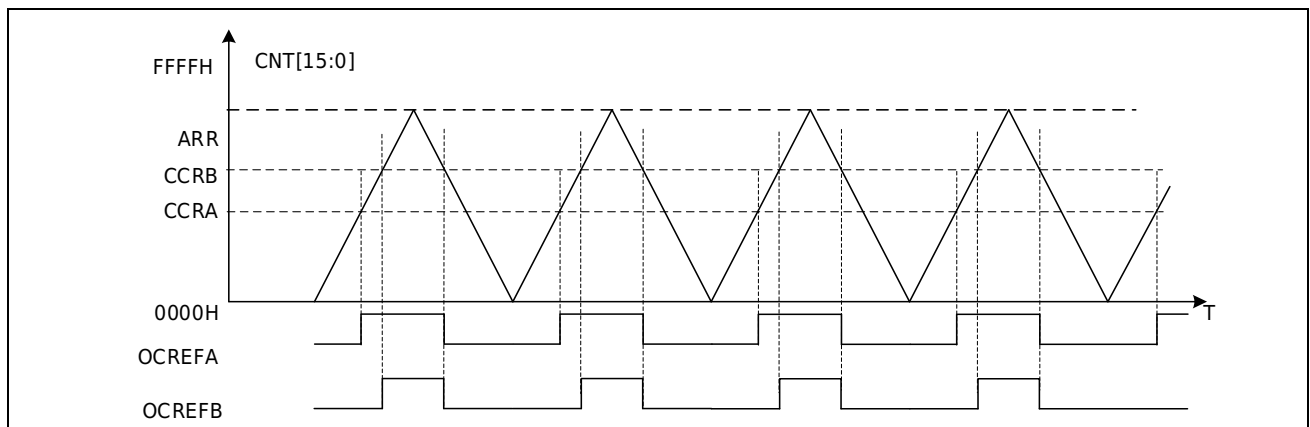


Figure 14-37 Triangular wave counting double-point comparison OCREF output (OCMx=111)

14.3.6.6 Independent PWM output

CHA is controlled by OCREFA, and the output of CHB is controlled by OCREFB. Through FLTR.CCPA, FLTR.CCPB can control the reverse of CHA and CHB output.

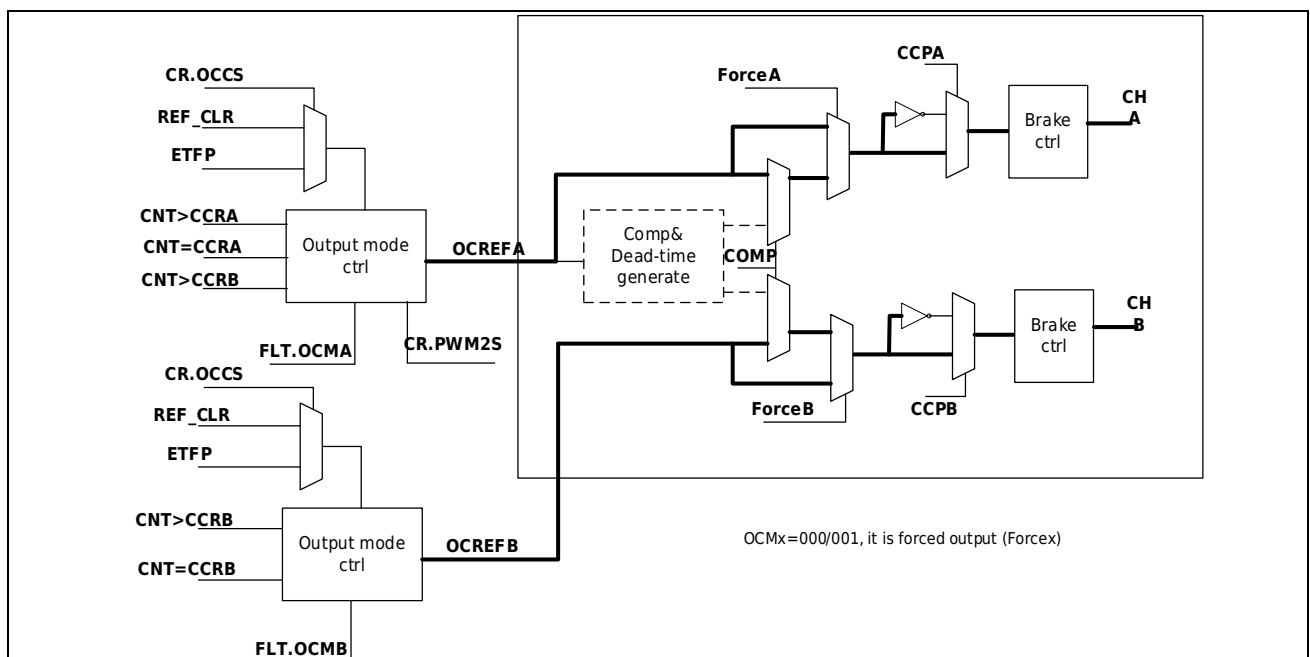


Figure 14-38 Independent PWM Output Block Diagram

Relationship between PWM output and OCREF

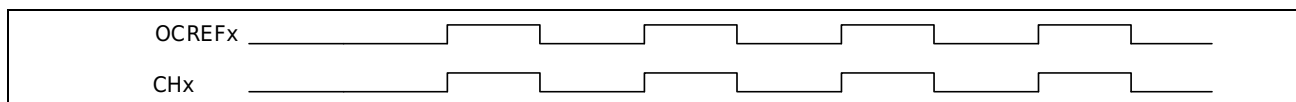


Figure 14-39 PWM output waveform when CCPx=0

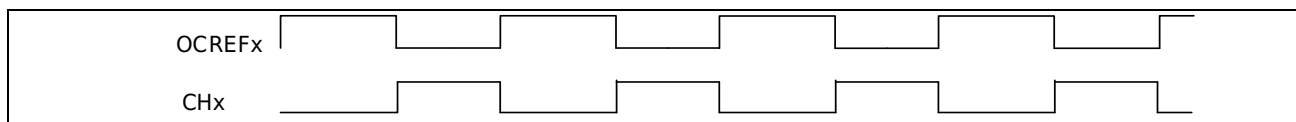


Figure 14-40 PWM output waveform when CCPx=1

14.3.6.7 Complementary PWM output

CHA is controlled by OCREFA, and OCREFA controls the output of CHB at the same time. The compare register CCRxB can be used as a dedicated compare control ADC trigger.

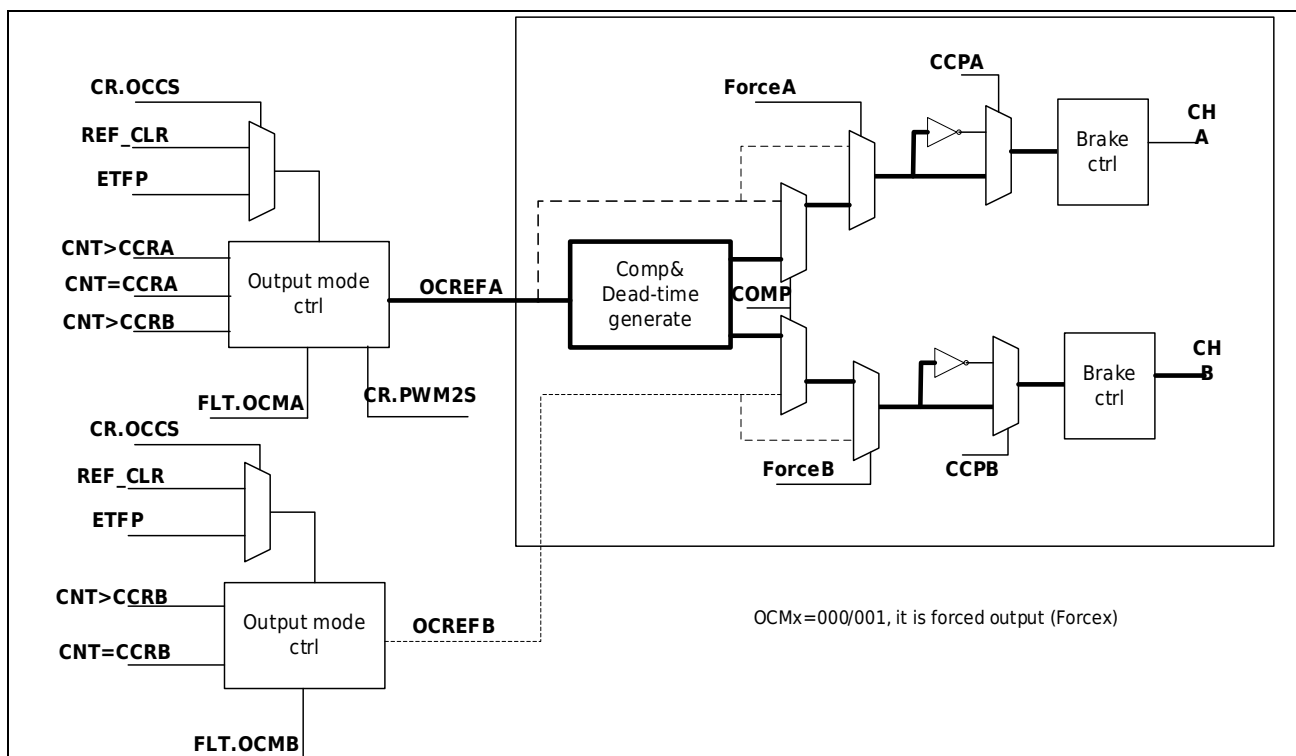


Figure 14-41 Complementary PWM Output Block Diagram

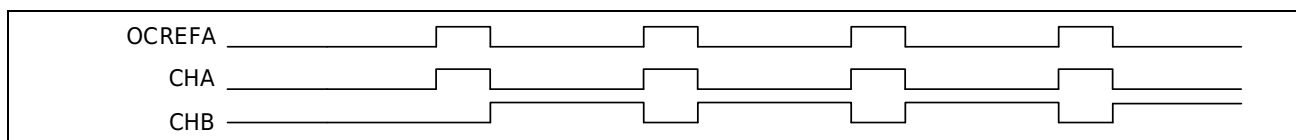


Figure 14-42 Complementary PWM output waveform

14.3.6.8 PWM output with dead zone

The dead-time function can be set in complementary PWM output mode.

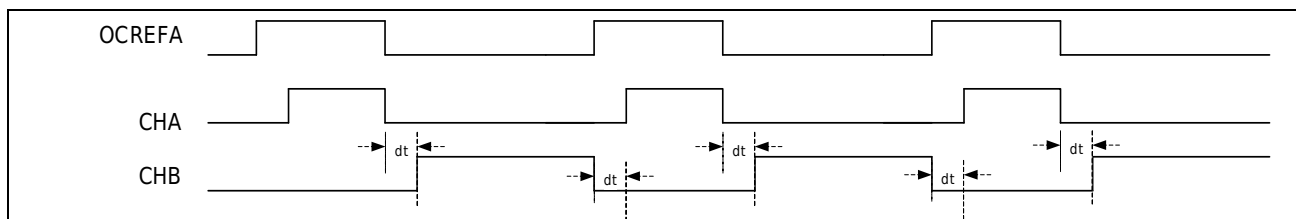


Figure 14-43 Complementary PWM output waveform

The dead time is controlled by 8-bit DTR, and the relationship between the dead time dt and DTR is as follows

DTR[7] =0	$T=DTR[6:0]+2$	2-129	step=1
DTR[7:6] =10	$T=\{DTR[5:0]+64\}*2+2$	130-256	step=2
DTR[7:5] =110	$T=\{DTR[4:0]+32\}*8+2$	258-506	step=8
DTR[7:5] =111	$T=\{DTR[4:0]+32\}*16+2$	514-1010	step=16

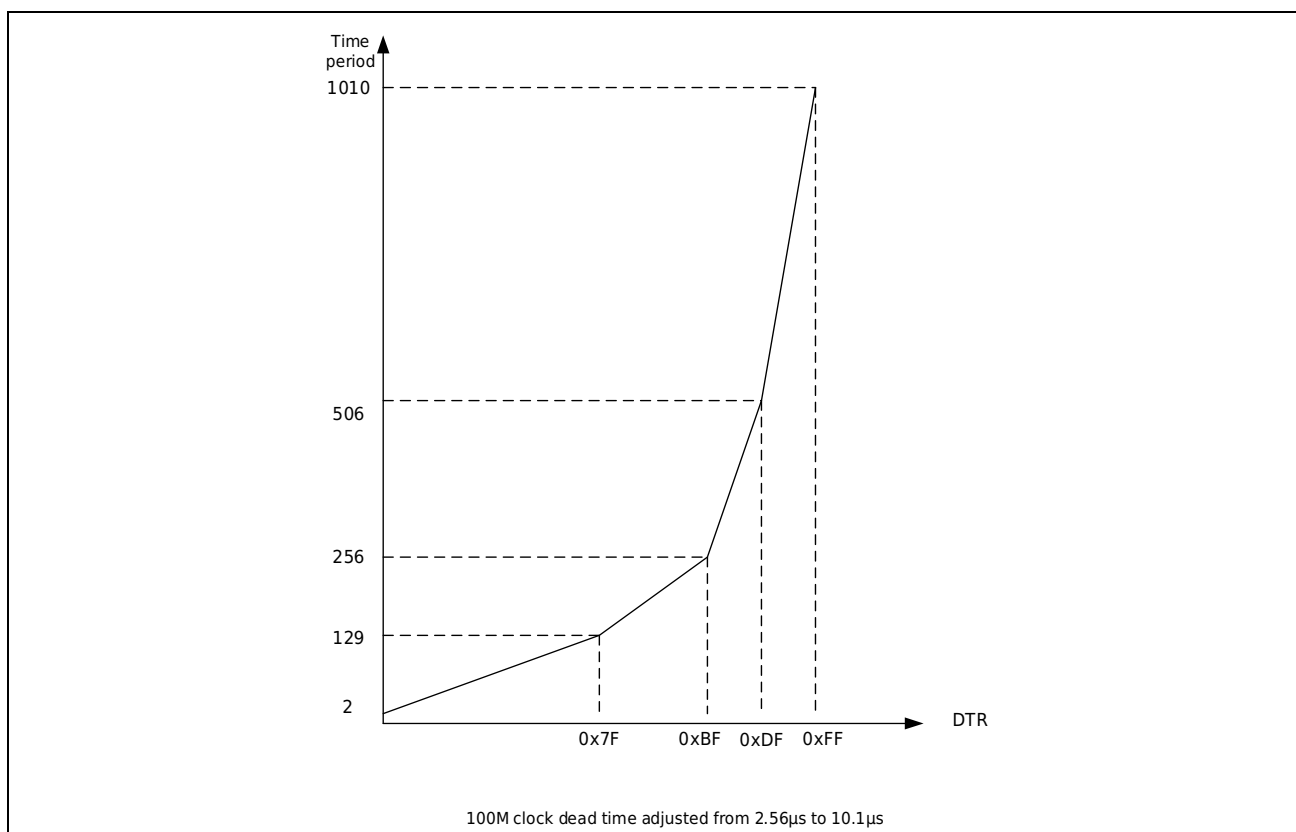


Figure 14-44 Dead time

14.3.6.9 Single pulse output

Single pulse mode (ONE SHOT) is a special case of the above PWM mode. In this mode, the counter can be triggered to start by an excitation signal and generate a programmable width pulse, as shown in the following diagram.

The counter can be started by the slave mode controller. can be generated in output compare mode or PWM mode. One-shot mode is selected by setting the ONESHOT bit in the ATIMx_M23CR register to 1. the counter will automatically stop when the next update event UEV occurs.

Because the counter will stop when the next update event UEV occurs, so when the initial value of the RCR setting is not 0, the timer will stop after the RCR value + 1 overflow event. If the overflow or underflow mask function of the RCR register is enabled, the corresponding mask overflow event needs to be ignored when RCR is counted.

Do not set the initial value of the counter in the sawtooth counting mode to 0 in the single pulse mode, and do not set the counting value in the up counting mode to be greater than or equal to ARR.

The output pulses are only generated correctly when the comparison value differs from the counter initial value. Before starting (when the timer is waiting to be triggered), the following configurations must be made:

- When counting up: $CNT < CCRxy \leq ARR$ (special attention, $0 < CCRxy$)
- When counting down: $CNT > CCRxy$

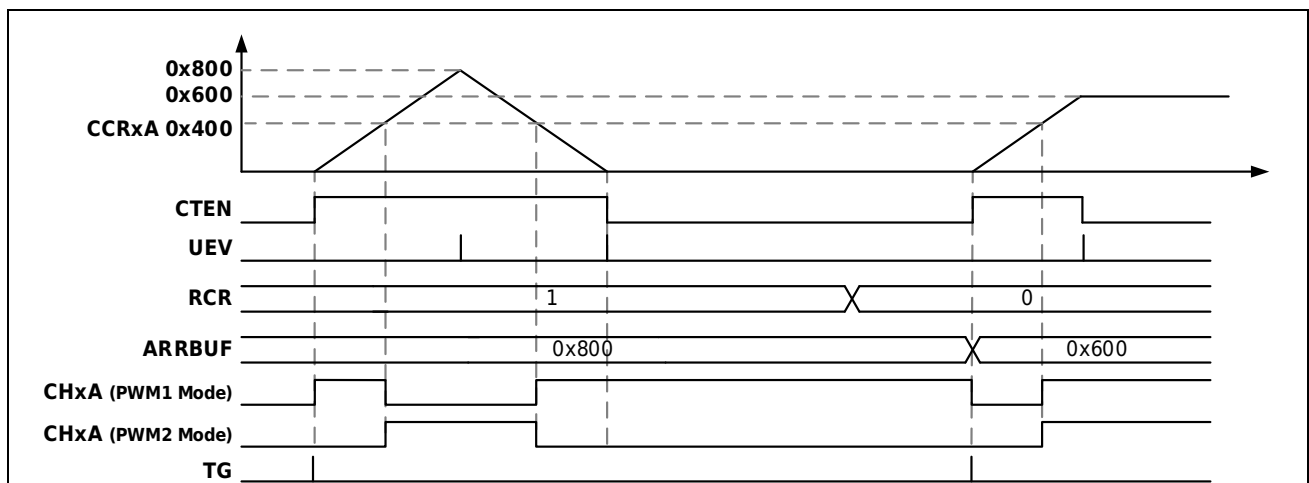


Figure 14-45 Single pulse counting in triangle wave mode

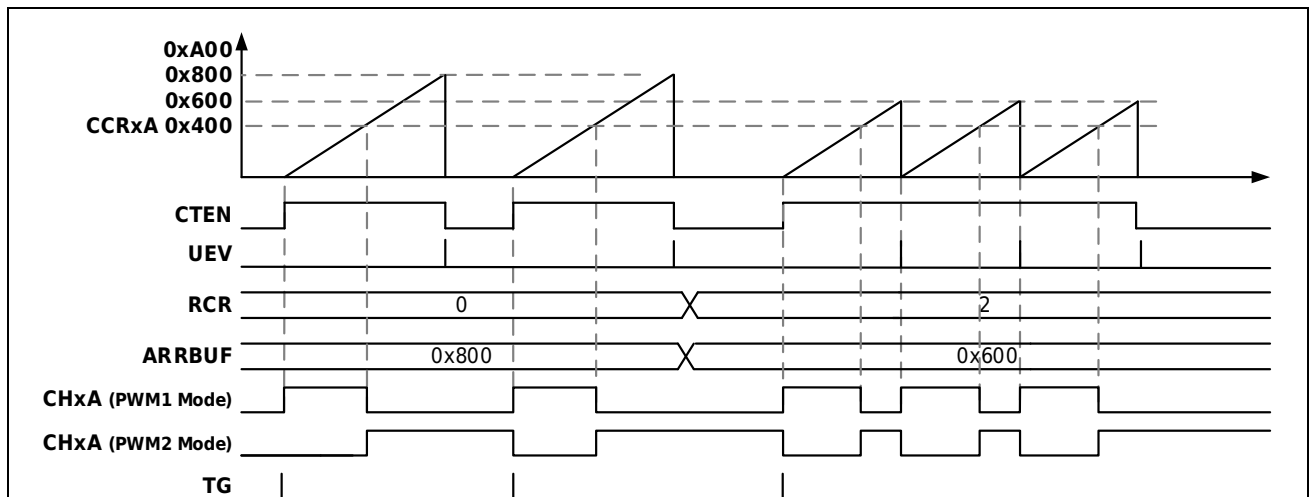


Figure 14-46 Counting Single Pulse Mode on Ramp Waveform

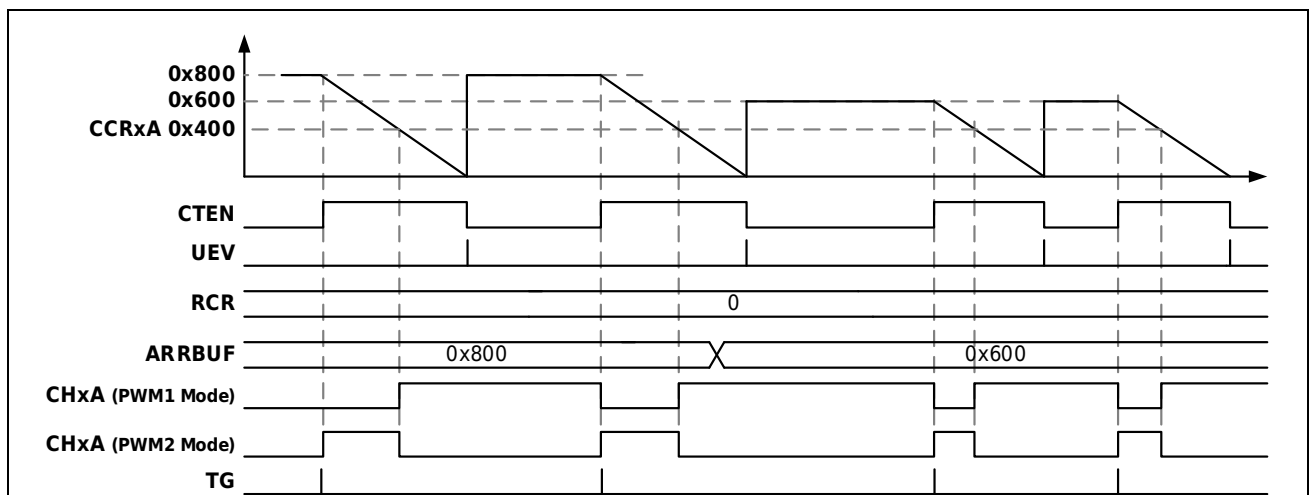


Figure 14-47 Counting single pulse mode under sawtooth waveform

Note: In up-counting, when configured as PWM2 mode, since the channel initial output is low level, the first trigger will generate a rising edge and maintain a high level. Upon the second trigger, a programmable width low level pulse can be output. In down-counting, when configured as PWM1 mode, since the channel initial output is low level, the first trigger will generate a rising edge and maintain a high level. Upon the second trigger, a programmable width low level pulse can be output.

14.3.6.10 Compare interrupt

The sawtooth comparison match will set the corresponding flag of the IFR register to 1, and if the interrupt is enabled CRCHx.CIEy (x=0,1,2;y=A,B) will trigger an interrupt.

For triangular wave comparison match, you can select up-count comparison match, down-count comparison match or both comparison matches separately.

Compare A Compare Match When the count value is equal to CCRxA, use M23CR.CIS control uniformly.

When M23CR.CIS=2'b00 compare match no output

When M23CR.CIS=2'b01 compare match when counting up

When M23CR.CIS=2'b10 compare match when counting down

When M23CR.CIS=2'b11 Comparing matches when counting up and down

Compare B Compare Match When the count value is equal to CCRxB, different channels can be controlled individually using CRCHx.CISB.

When CRCHx.CISB=2'b00 Channel x compare match no output

When CRCHx.CISB=2'b01 compare match while counting on channel x

When CRCHx.CISB=2'b10 Compare match when counting down channel x

When CRCHx.CISB=2'b11 Comparing matches when channel x counts up and down

The B channel comparison match is controlled separately for more flexible triggering of the ADC. For details, see the [Timer Trigger ADCTimer triggers ADC] chapter.

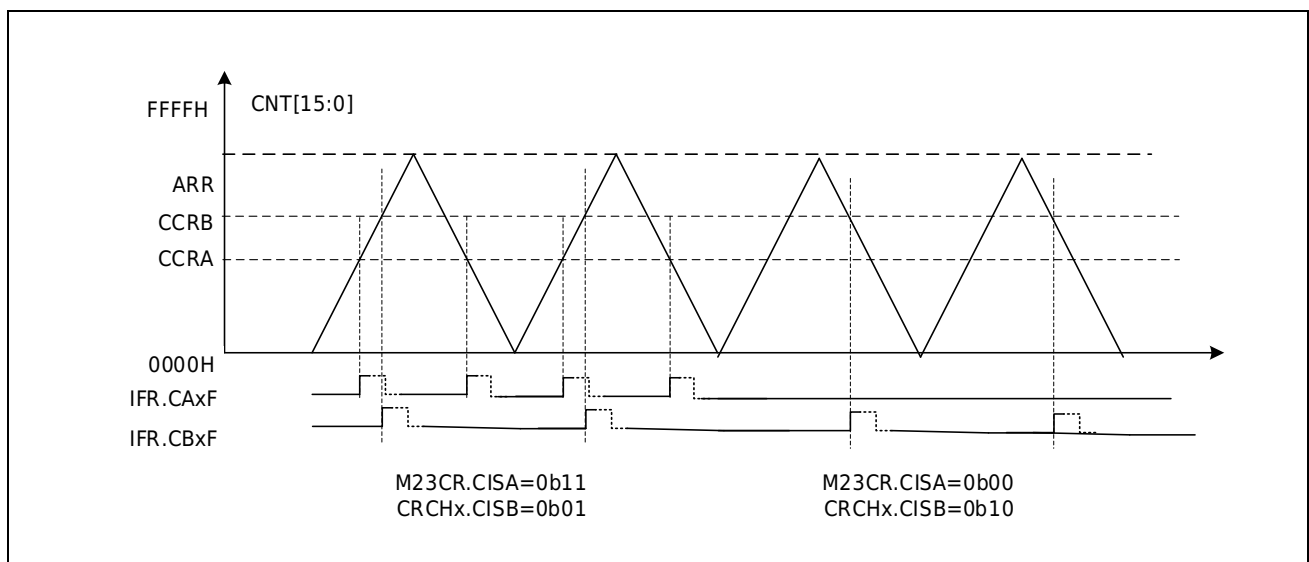


Figure 14-48 Interrupt diagram

14.3.6.11 Capture input

M23CR.MODE=2/3

The capture function can be set in the triangular wave counting or sawtooth wave counting mode, and the level edge of the capture can be set. When the capture occurs, the captured value is stored in the comparison capture register and a capture interrupt is generated.

The compare capture function of each channel can be set individually, selected by the register CRCHx.CSA/CSB.

The capture edge of each channel can be set independently, and the edge of the capture trigger can be selected through the register CRCHx.CFy/CRy (x=0/1/2;y=A/B).

When the capture action occurs again before the capture flag is cleared after the capture occurs, the capture data overwrite flag will be generated.

When the timer is not started, if there is a valid capture edge, the capture flag and capture action will also be generated.

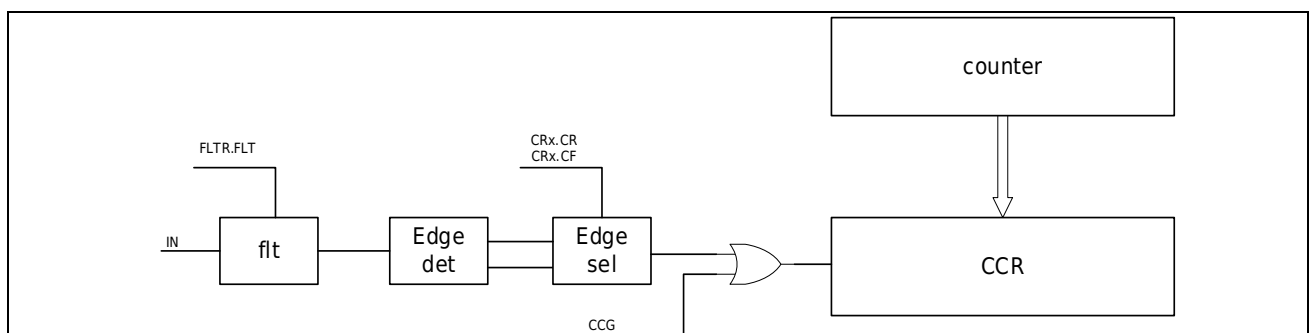


Figure 14-49 Capturing Functional Block Diagram

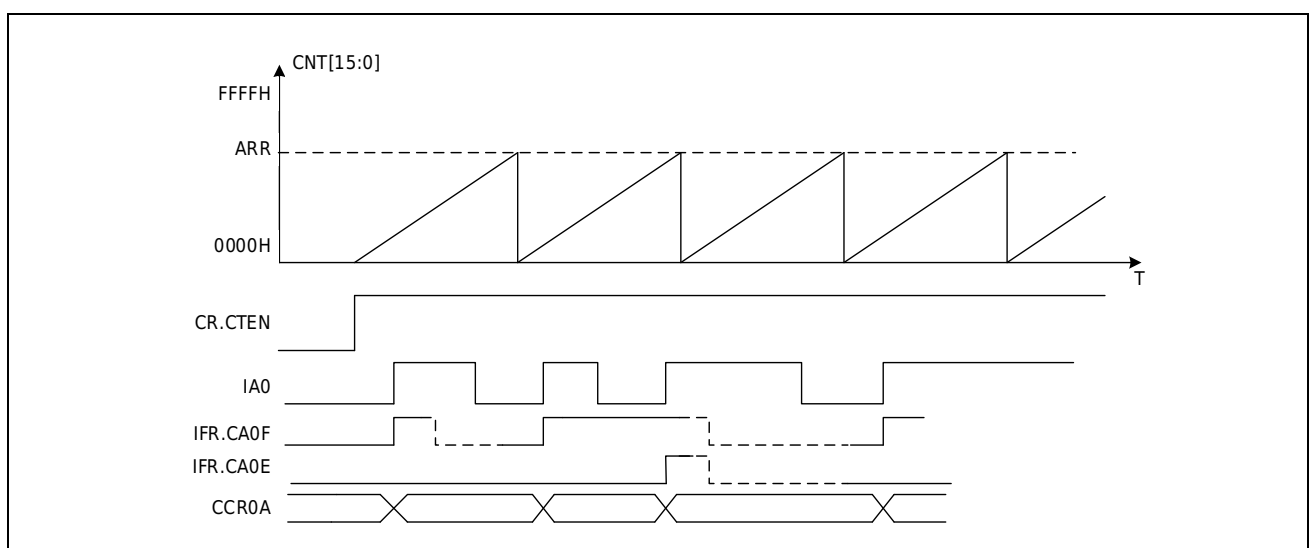


Figure 14-50 Capture Timing Diagram

CH0A selects CH0A input or the XOR input of CH0A, CH1A, and CH2A through MSCR.IA0S.

Table 14-5 Capture input CH0A signal list

IA0S	0	1
ATIM3	CH0A	CH0A CH1A CH2A XOR input
ATIM0	CHA	CHA ETR GATE XOR input

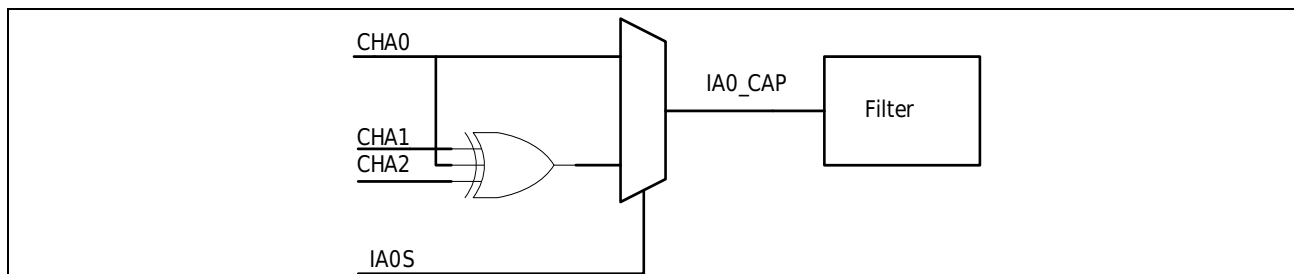


Figure 14-51 ATIM3 CH0A port selection

CH0B selects the CH0B input or the internal trigger signal selected by MSCR.TS through MSCR.IB0S.

Table 14-6 Capture input CH0B signal list

IB0S	0	1	
ATIM3	CHB0	Internal trigger MSCR.TS select signal	
		MSCR.TS	Channel B capture input
		000	ETR filter phase selection output signal, filter phase selection can be configured
		001	Interconnect ITR0
		010	Interconnect ITR1
		011	Interconnect ITR2
		100	Interconnect ITR3
		101	CH0A edge
		110	CH0A filter output signal, filter function can be configured
		111	Reserved

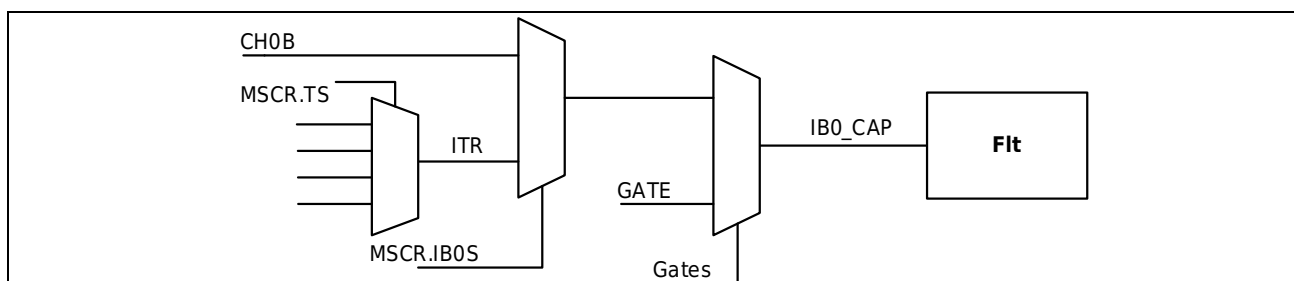


Figure 14-52 CHB port selection

The Gates signal is generated for selecting the PWM complementary output function. When PWM is the single point comparison complementary output, the compare capture register CCR0B is unused, and automatically switches to using GATE as the capture compare output.

Note: When GATE is configured for compare output, its output mode only supports being set to force the output to 0 or 1 and having the PWM mode active (that is, the GATE output is valid when OCMB0 is configured as 0/1/6/7).

14.3.6.12 Setup example

Edge-Aligned Independent PWM Output Settings

1. Set mode M23CR.MODE=2
2. Set sawtooth wave counting direction M23CR.DIR
3. Set PWM period value ARR
4. Set the initial value of the counter (the initial value must be less than the period value)
5. Set PWM comparison value CCRxA, CCRxB
6. Set PWM output comparison mode FLT.OCMA FLT.OCMB to 6 or 7
7. Clear the associated interrupt flag
8. Enable the corresponding interrupt
9. Set output polarity FLT.CCPAx FLT.CCPBx
10. Enable output DTR.MOE
11. Enable timer M23CR.CTEN

Center-Aligned Complementary PWM Output Settings

1. Set mode M23CR.MODE=3
2. Set complementary output M23CR.COMP
3. Set PWM period value ARR
4. Set the initial value of the counter (the initial value must be less than the period value)
5. Set the PWM comparison value CCRxA (CCRxB does not need to be set, the PWM output has nothing to do with this register)
6. Set PWM output comparison mode FLT.OCMA FLT.OCMB to 6 or 7
7. Clear the associated interrupt flag
8. Enable the corresponding interrupt
9. Set output polarity FLT.CCPAx FLT.CCPBx
10. Enable output DTR.MOE
11. Enable timer M23CR.CTEN

Triangular wave non-center alignment with dead zone complementary PWM output setting

1. Set mode M23CR.MODE=3
2. Set complementary output M23CR.COMP
3. Set two comparison enable M23CR.PWM2S

4. Set PWM period value ARR
5. Set the initial value of the counter (the initial value must be less than the period value)
6. Set the PWM comparison value CCRxA, CCRxB, the upper counting comparison point is CCRxA, and the lower counting comparison point is CCRxB
7. Set PWM output comparison mode FLT.OCMA FLT.OCMB to 6 or 7
8. Clear the associated interrupt flag
9. Enable the corresponding interrupt
10. Set output polarity FLT.CCPAx FLT.CCPBx
11. Set dead zone enable DTR.DTEN
12. Set dead time DTR.DT
13. Enable output DTR.MOE
14. Enable timer M23CR.CTEN

Capture function settings

Set CH0B as rising edge capture function

1. Select the counter counting method, set mode=2
2. Select CH0B as capture mode CRCHx.CSB=1
3. FLT.FLTB0 as required
4. Select the captured edge CRCHx.CRB=1
5. Set ARR to change the period value
6. Start timer M23CR.CTEN
7. Clear the associated interrupt flag
8. Clear the capture flag and enable the corresponding interrupt
9. After querying the capture flag, read CCR0B to obtain the capture value

Note: If the capture edge is valid when the timer is not started, the capture flag and capture action will also be generated.

14.3.7 Mode 2/3 Slave Mode

The timer can be synchronized to an external trigger in several modes: reset mode, gated mode and triggered mode.

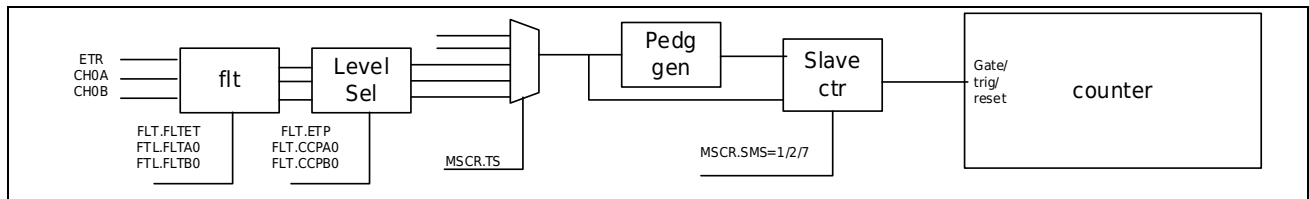


Figure 14-53 Slave Mode Schematic

Select from mode function by MSCR.SMS;

Table 14-7 MSCR Register Slave Mode and Trigger Source List

MSCR.SMS	Select from mode function 001: Reset function 010: trigger mode 111: Gating function
MSCR.TS	Trigger selection 000: The signal ETFP after the filter phase selection of port ETR, the filter function can be configured 001: Internal interconnection signal ITR0 010: Internal interconnection signal ITR1 011: Internal interconnection signal ITR2 100: Internal interconnection signal ITR3 101: edge signal of port CH0A; 110: Port CH0A filters the signal IAFP after phase selection, the filter function can be configured 111: The signal IBFP after the filter phase selection of port CH0B, the filter function can be configured

14.3.7.1 Gated count

Enables the counter according to the selected input level. In the following example, the counter only counts up when CH0A is low:

- Configure channel CH0A to detect a low level on CH0A. Configure the input filter bandwidth (in this example, no filtering is required, so keep FLT.FLTA0=000). Set CCPA0=1 in the FLTR register to determine the polarity (only detect low level).
- Set SMS=111 in the MSCR register to configure the timer as gated mode; set TS=110 in the MSCR register to select CH0A as the input source.
- Set CTEN=1 in the CR register to start the counter. When CH0A is low, the counter starts counting according to the internal clock, and stops counting once CH0A becomes high. In gated mode, if CTEN=0, the counter cannot be started regardless of the trigger input level.

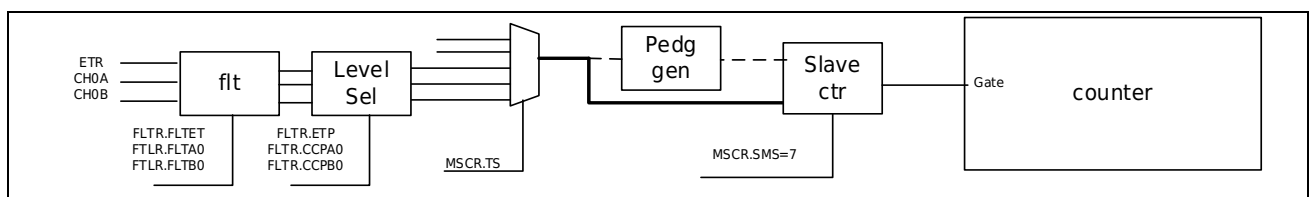


Figure 14-54 Gating function

14.3.7.2 Trigger function

Using external triggers (CH0A, CH0B, ETR) can start the timer synchronously, and using the internal interconnection signal of the timer combined with MSCR.MMS can also configure the timer to start synchronously. The trigger signal is the rising edge of the input signal. You can also use software to write CR.TG to start the software trigger function.

A selected event on the input enables the counter. In the example below, the counter starts counting up on the rising edge of the CH0B input:

- Configure channel CH0B to detect the rising edge of CH0B. Configure the input filter bandwidth (in this example, no filter is needed, keep FLT.FLTB0=000). Set CCPB0=0 in the FLTR register to determine polarity (no inversion).
- Set SMS=010 in the MSCR register to configure the timer as trigger mode; set TS=111 in the MSCR register to select CH0B as the input source. When a rising edge appears on CH0B, the counter starts counting driven by the internal clock and sets the TIF flag at the same time. The delay between the rising edge of CH0B and the start of the counter depends on the synchronization circuit at the CH0B input.

Note: If you use the falling edge trigger, first select the trigger polarity, and then select the mode, otherwise false triggers will occur.

14.3.7.3 Reset count

When a trigger input event occurs, the counter and its prescaler can be re-initialized; at the same time, if the URS bit of the CR register is low, an update event UEV is also generated; then all preload registers (ARR, CCRx) are updated.

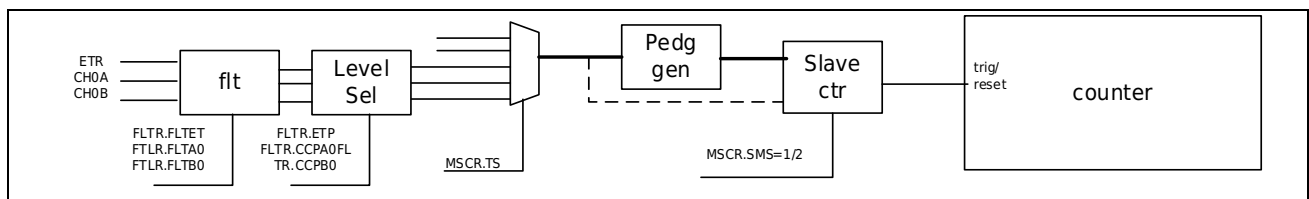


Figure 14-55 Trigger and reset functions

14.3.8 Quadrature code counting function

MSCR.SMS=4/5/6, corresponding to the mode 1/2/3 of the orthogonal coding mode. At this time, the counter encodes and counts according to the phase relationship of IAFP and IBFP. IAFP, IBFP is the port input CH0A, CH0B filter phase selection signal.

Mode 1 uses the edge count of CH0A. Mode 2 uses the edge count of CH0B. Mode 3 uses both CH0A and CH0B edges to count.

In order to ensure the correctness of the counting phase, it is necessary to ensure that the phase difference of the A/B input is greater than the phase difference of one pulse width, and the A/B input pulse width needs to be greater than two pulse widths.

	Quadrature Signal Level Conditions		Quadrature signal IAFP edge condition		Quadrature signal IBFP edge condition	
	IBFP	IAFP	Rising	falling	Rising	falling
MOD1	High		Down	Up	-	-
	Low		up	Down	-	-
MOD2		High	-	-	up	Down
		Low	-	-	Down	Up
MOD3	High	High	Down	Up	up	Down
	Low	Low	up	Down	Down	Up

Note: The white bottom in the above table is the CNT counting action

CHAF/CHBF is the signal filtered by port CH0A/CH0B, and IAFP/IBFP is the signal after port filter phase selection.

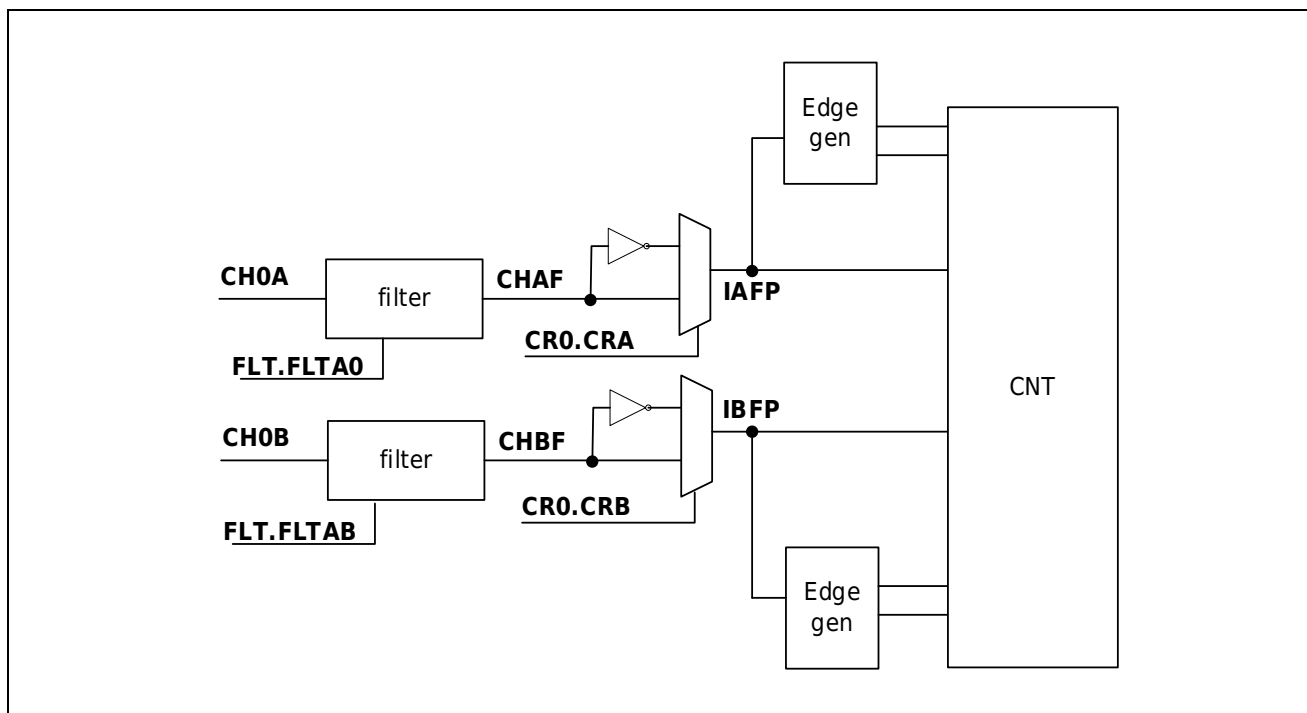


Figure 14-56 Code counting function block diagram

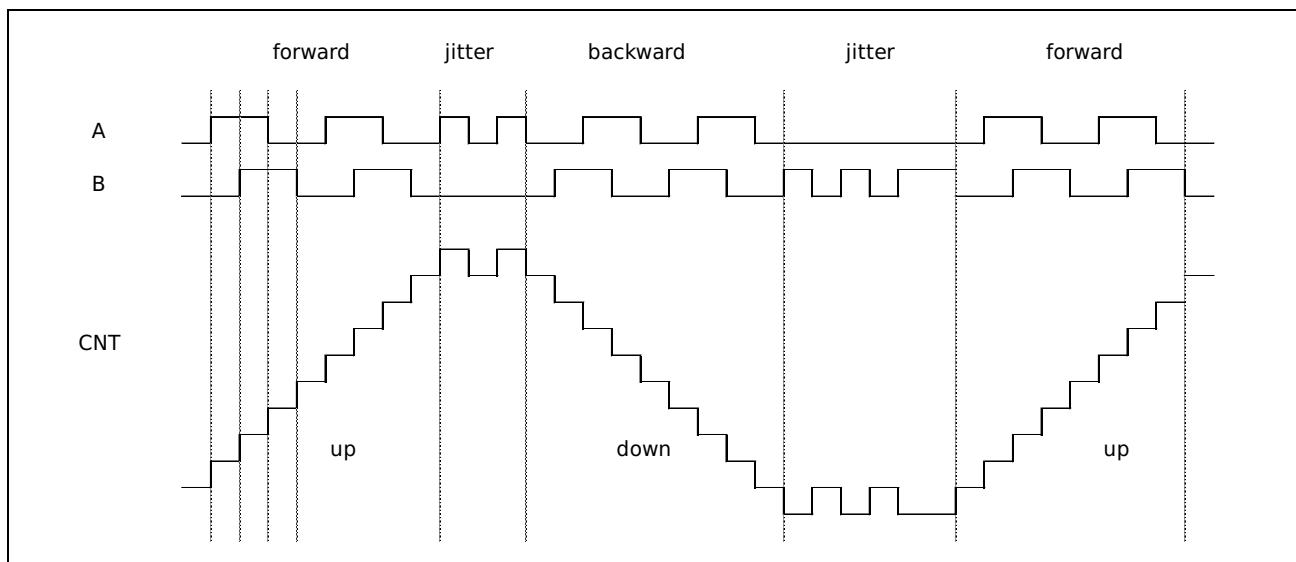


Figure 14-57 Code count

14.3.9 Timer triggers ADC

CCR0A, CCR1A, CCR2A comparison match can be configured to trigger the ADC, and the center-aligned PWM can only be selected through the control register M23CR.CIS to trigger on the rising match or falling match.

CCR0B, CCR1B, CCR2B, CCR3 comparison match can be configured to trigger ADC. When center-aligned PWM, three matching trigger points (rising, falling, rising and falling) can be controlled respectively through the register CRCHx.CISB.

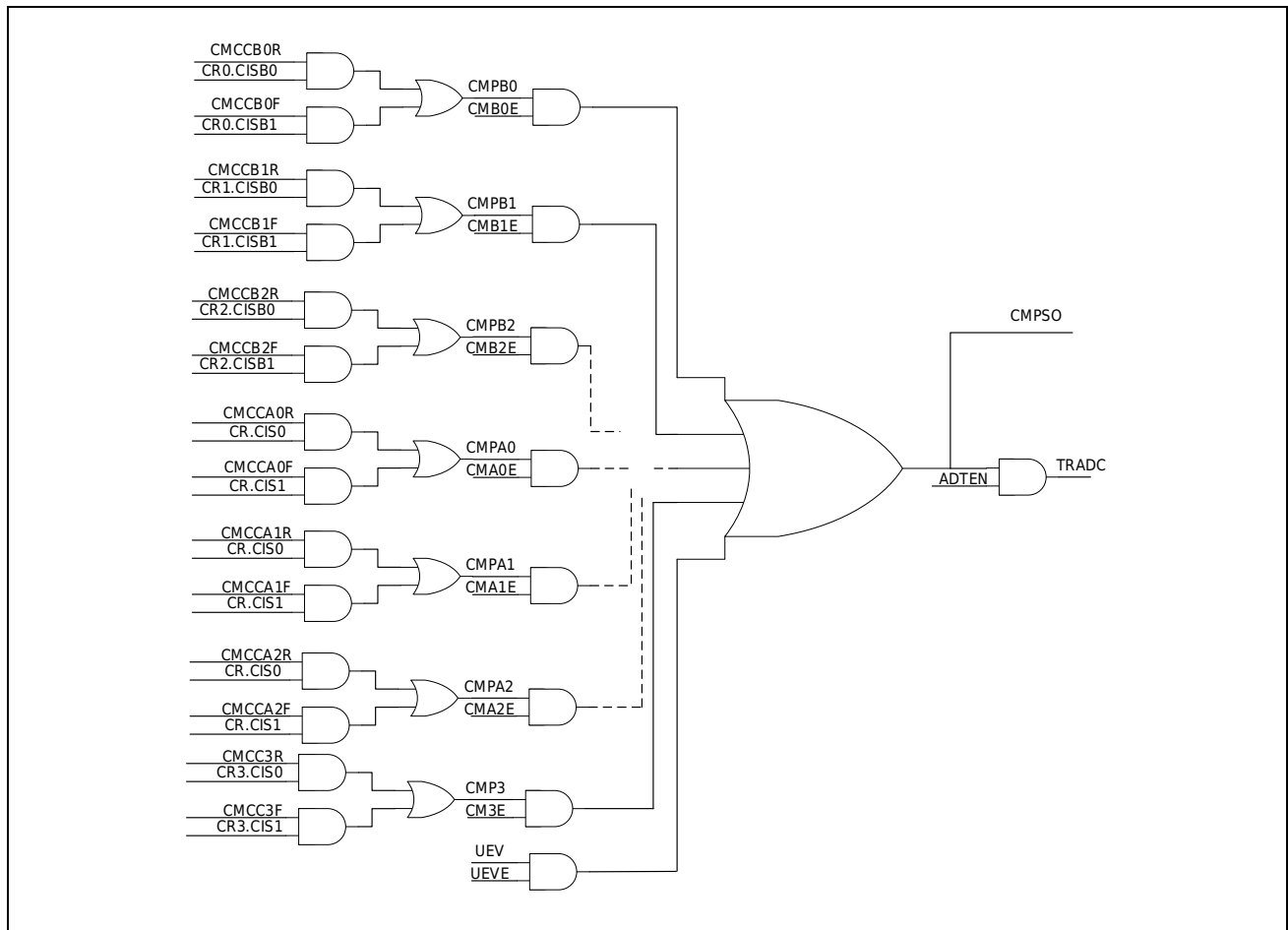


Figure 14-58 ADC trigger

Complementary PWM output B compare as ADC trigger function

1. Set to PWM complementary output mode, refer to " Center Aligned Complementary PWM Output Setting "
2. Set CCRxB to set ADC trigger comparison point
3. Set CRCHx.CISB to select up-counting and down-counting comparison matching (sawtooth wave does not need to be selected)
4. Select the source ADTR for ADC trigger comparison
5. Enable ADC trigger ADTR.ADTE

14.3.10 Brake control

The external BK port input can control the brake, and the system fail can control the brake function. Through CR.BG, the software brake function can be realized, and the output port can be controlled to the set state.

14.3.11 Interrupt

Table 14-8 ATIM interrupt source and interrupt request list

Interrupt	Note
ATIM0_UEV	ATIM0 event update
ATIM0_CHA	ATIM0 channel A compare capture interrupt
ATIM0_CHB	ATIM0 channel B compare capture interrupt
ATIM0_BKETR	ATIM0 brake / trigger interrupt
ATIM3_UEV	ATIM3 Event Update
ATIM3_CHYA	ATIM3 channel A compare capture interrupt
ATIM3_CHYB	ATIM3 channel B compare capture interrupt
ATIM3_BK_ETR_OV_UD_CMP3	ATIM3 brake / trigger interrupt / channel 3 compare / overflow / underflow

14.3.12 Timer interconnection

The TRGO output signal can be connected to the ITR signal of other timers. The connection relationship is as follows:

Table 14-9 Timer Interconnect Signal List

	ITR0	ITR1	ITR2	ITR3
ATIM0	GTIM0 overflow	GTIM1 overflow	GTIM2 overflow	ATIM3 TRGO
ATIM3	GTIM0 overflow	GTIM1 overflow	GTIM2 overflow	ATIM0 TRGO

14.3.13 input interconnection

The ETR input of ATIM0 can be directly input from the port, and can also be connected to other modules or ports through the selection of the port function register SYSCTRL->IOAUX1.

When ATIM0ETR=0x0, the capture input is the port input selected by AFRH/AFRL. When ATIM0ETR=0x1~0x7, it can also be selected through the port function register SYSCTRL->IOAUX1 to connect to the input or output of other modules.

Table 14-10 ATIM0 ETR signal source list

SYSCTRL->IOAUX1	0	1	2	3	4	5	6	7
ATIM0ETR [2:0]	ATIM0_ETR	USART0_RXD	VC0_OUT	LVD_OUT	VC1_OUT	PA04	PA12	PB04

The input signal of the CHA of ATIM0 can be input from the port selected by AFRH/AFRL, or can be connected to other modules or ports through the port function register SYSCTRL->IOAUX2 selection.

When CAP0=0x0, the capture input is the port input selected by AFRH/AFRL. When CAP0=0x1~0x7, it is connected to the input or output of other modules.

Table 14-11 ATIM0 CHA input signal source list

SYSCTRL->IOAUX2	0	1	2	3	4	5	6	7
ATIM0CAP0[2:0]	ATIM0_CHA	USART0_RXD	VC0_OUT	LVD_OUT	VC1_OUT	PA00	PA08	PB00

14.4 Register description

14.4.1 Register list

Table 14-12 ATIM base address list

Name	base address	Description
ATIM0	0x4000 6000	ATIM0 base address
ATIM3	0x4000 6C00	ATIM3 base address

Table 14-13 ATIM register list

Register name	Offset address	Reset value
Auto-Reload Register(ATIMx_ARR)	0x000	0x0000 0000
16 -bit Mode Count Register(ATIMx_CNT)	0x004	0x0000 0000
32- bit Mode Count Register(ATIMx_CNT32)	0x008	0x0000 0000
Mode 0 Control Register (described by mode)(ATIMx_M0CR)	0x00C	0x0060 0008
Mode 1 Control Register (described by mode)(ATIMx_M1CR)	0x00C	0x0060 0008
Mode 23 Control Register (described by mode)(ATIMx_M23CR)	0x00C	0x0060 0008
Interrupt Flag Register(ATIMx_IFR)	0x010	0x0000 0000
Interrupt Clear Register(ATIMx_ICLR)	0x014	0x0000 FFFF
Master-Slave Mode Control Register(ATIMx_MSCR)	0x018	0x0000 0000
Filter Control Register(ATIMx_FLTR)	0x01C	0x0000 0000
ADC Trigger Control Register(ATIMx_ADTR)	0x020	0x0000 0000
Compare Unit 0 Control Register(ATIMx_CR0/ATIMx_CRCH0)	0x024	0x0000 3000
Compare Unit 1 Control Register(ATIM3_CRCH1)	0x028	0x0000 3000
Compare Unit 2 Control Register(ATIM3_CRCH2)	0x02C	0x0000 3000
Dead Time Register(ATIMx_DTR)	0x030	0x0000 0000
Repeat Count Register(ATIMx_RCR)	0x034	0x0000 0000
Timer Reload Register / Cycle Map Address(ATIMx_ARRDM)	0x038	0x0000 0000
Compare 0A Register(ATIMx_CCR0A)	0x03C	0x0000 0000
Compare 0B Register(ATIMx_CCR0B)	0x040	0x0000 0000
Compare 1A Register(ATIM3_CCR1A)	0x044	0x0000 0000
Compare 1B Register(ATIM3_CCR1B)	0x048	0x0000 0000
Compare 2A Register(ATIM3_CCR2A)	0x04C	0x0000 0000
Compare 2B Register(ATIM3_CCR2B)	0x050	0x0000 0000
Compare 3A Register(ATIM3_CCR3)	0x054	0x0000 0000
Compare Unit 3A Control Register(ATIM3_CRCH3)	0x058	0x0000 0000

(x=0,3)

14.4.2 Mode 0 Register Description

14.4.2.1 Auto-Reload Register (ATIMx_ARR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
ARR															

Bit	Marking	Place name	Function	Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value	
b15~b0	ARR	16 -bit reload timer reload value register	-	RW

14.4.2.2 16 -bit Mode Count Register (ATIMx_CNT)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CNT															

Bit	Marking	Place name	Function	Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value	
b15~b0	CNT	16 -bit reload timer Count value register	-	RW

14.4.2.3 32- bit Mode Count Register (ATIMx_CNT32)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
CNT32[31:16]															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CNT32[15:0]															

Bit	Marking	Place name	Function	Read and write
b31~b0	CNT32	32- bit timer Count value register	valid in mode 0 32 -bit timer free counting mode, other modes prohibit writing this register	RW

14.4.2.4 Control Register (ATIMx_MOCR)

Reset value: 0x0060 0008

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res	MODE		Res	UI	GATE P	GATE	Res	PRS				TOGE N	CT	MD	CEN

Bit	Marking	Place name	Function	Read and write
b31~b14	Res	Reserved bit	Reserved, must remain at reset value	
b13~b12	MODE	Operating mode	00 timer mode 0; 01 PWC mode 10 sawtooth wave mode; 11 triangle wave mode	RW
b11	Res	Reserved bit	Reserved, must remain at reset value	
b10	UI	Overflow interrupt enable control	0: disable overflow interrupt 1: enable overflow interrupt	RW
b9	GATEP	Port GATE polarity control	0: Port GATE active high 1: Port GATE active low	RW
b8	GATE	Timer Gate Enable	0: No gate control, timer works when CTEN=1 1: Gating is enabled, the timer works only when the port GATE is valid and CTEN=1	RW
b7	Res	Reserved bit	Reserved, must remain at reset value	
b6~b4	PRS	Internal clock frequency division selection	000: 1; 001: 2; 010: 4; 011: 8; 100: 16; 101: 32; 110: 64; 111: 256;	RW
b3	TOGEN	Toggle output enable in mode 0	0: Inversion output is off, CHA, CHB output is low level 1: Toggle output enable	RW
b2	CT	Counting clock selection	0: internal count clock TCLK 1: External count clock ETR	RW
b1	MD	Mode selection 32 timing /16 timing mode selection	0: 32 -bit free count 1: 16 -bit reload count	RW
b0	CEN	Timer enable	0: timer stopped; 1: Timer enabled	RW

14.4.2.5 Interrupt Flag Register (ATIMx_IFR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res														UI	

Bit	Marking	Place name	Function	Read and write
b31~b1	Res	Reserved bit	Reserved, must remain at reset value	
b0	UI	Overflow interrupt flag	0: The counter has not overflowed 1: The counter has overflowed	RO

14.4.2.6 Interrupt Flag Clear Register (ATIMx_ICLR)

Reset value: 0x0000 FFFF

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res														UI	

Bit	Marking	Place name	Function	Read and write
b31~b1	Res	Reserved bit	Reserved, must remain at reset value	
b0	UI	overflow interrupt clear	0: clear overflow flag 1: no function	R1W0

14.4.2.7 Dead Time Register (ATIMx_DTR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res			MO	Res											

Bit	Marking	Place name	Function	Read and write
b31~b13	Res	Reserved bit	Reserved, must remain at reset value	
b12	MO	Toggle output enable	0: flip port to input state 1: Toggle port output enable	RW
b11~b0	Res	Reserved bit	Reserved, must remain at reset value	

14.4.3 Mode 1 Register Description

14.4.3.1 16-bit Mode Count Register (ATIMx_CNT)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CNT															

Bit	Marking	Place name	Function	Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value	
b15~b0	CNT	Register count value	-	RW

14.4.3.2 Control Register (ATIMx_M1CR)

Reset value: 0x0060 0008

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res	Ones hot	Mode		Res	UI	Edg2nd	Edg1st	Res	PRS			Res	CT	Res	CEN

Bit	Marking	Place name	Function					Read and write
b31~b15	Res	Reserved bit	Reserved, must remain at reset value					
b14	Oneshot	Single trigger mode selection	0: Cyclic measurement 1: After completing a pulse measurement, it will end automatically, and you need to re-enable CTEN for another measurement.					RW
b13~b12	MODE	Operating mode	00 Timer mode 0; 01 PWC mode 10 sawtooth wave mode; 11 triangle wave mode					RW
b11	Res	Reserved bit	Reserved, must remain at reset value					
b10	UI	Overflow interrupt enable control	0: disable overflow interrupt 1: enable overflow interrupt Note: Counting to 0xFFFF will overflow and generate an overflow flag					RW
b9	Edg2nd	Pulse width measurement end edge selection						
b8	Edg1st	Pulse width measurement start edge selection	edg2nd	0	0	1	1	RW
			Edg1st	0	1	0	1	
			Measurement	Rising - upperiod	Low level width	High level width	Falling edge - falling edge cycle	
b7	Res	Reserved bit	Reserved, must remain at reset value					
b6~b4	PRS	Internal clock frequency division selection	000: 1; 001: 2; 010: 4; 011: 8; 100: 16; 101: 32; 110: 64; 111: 256;					RW
b3	Res	Reserved bit	Reserved, must remain at reset value					
b2	CT	Counting clock selection	0: Internal count clock TCLK 1: External count clock ETR;					RW
b1	Res	Reserved bit	Reserved, must remain at reset value					
b0	CEN	Pulse Width Measurement Enable	0: forbidden; 1: Enable					RW

14.4.3.3 Interrupt Flag Register (ATIMx_IFR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res												PWC	Res.	UI	

Bit	Marking	Place name	Function	Read and write
b31~b3	Res	Reserved bit	Reserved, must remain at reset value	
b2	PWC	Pulse Width Measurement Interrupt Flag	0: Pulse width measurement not completed 1: Pulse width measurement is complete	RO
b1	Res	Reserved bit	Reserved, must remain at reset value	
b0	UI	Overflow interrupt flag	0: The counter has not overflowed 1: The counter has overflowed	RO

14.4.3.4 Interrupt Flag Clear Register (ATIMx_ICLR)

Reset value: 0x0000 FFFF

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res												PWC	Res.	UI	

Bit	Marking	Place name	Function	Read and write
b31~b3	Res	Reserved bit	Reserved, must remain at reset value	
b2	PWC	Pulse width measurement interrupt flag cleared	0: Clear pulse width measurement interrupt flag 1: no function	R1W0
b1	Res	Reserved bit	Reserved, must remain at reset value	
b0	UI	overflow interrupt clear	0: clear overflow flag 1: no function	R1W0

14.4.3.5 Master-Slave Mode Control Register (ATIMx_MSCR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res		IB0S	IA0S	Res			TS			Res					

Bit	Marking	Place name	Function	Read and write
b31~b13	Res	Reserved bit	Reserved, must remain at reset value	
b12	IB0S	CH0B input selection	0: CH0B; 1: internal trigger TS selection signal;	RW
b11	IA0S	IA0 input selection	0: CH0A; 1: CH0A CH1A CH2A XOR(TIM3) Note: After setting to 1, any port change of the port will cause the input to change	RW
b10~b8	Res	Reserved bit	Reserved, must remain at reset value	
b7~b5	TS	Trigger selection	000: the signal ETRP after the filter phase selection of the port ETR; 001: Internal interconnection signal ITR0 010: internal interconnection signal ITR1; 011: internal interconnection signal ITR2; 100: internal interconnection signal ITR3; 101:Invalid; 110: Filtered phase-selected signal IAFP of port CH0A 111: signal IBFP after filter phase selection of port CH0B;	RW
b4~b0	Res	Reserved bit	Reserved, must remain at reset value	

14.4.3.6 Output Control Filtering (ATIMx_FLTR)

Reset value: 0x00000000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
ETP	FLTET			Res											
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res									FLTBO			Res	FLTA0		

Bit	Marking	Place name	Function	Read and write
b31	ETP	ETR input phase selection	0: same phase; 1: reverse input;	RW
b30~b28	FLTET	ETR filter control	0xx: Filter invalid 100: pclk/3 continuous effective; 101: pclk/4 continuous effective; 110: pclk/16 continuous effective; 111: pclk/64 continuous effective	RW
b27~b7	Res	Reserved bit	Reserved, must remain at reset value	
b6~b4	FLTBO	CHB input filter control	0xx: Filter invalid 100: pclk/3 continuous effective; 101: pclk/4 continuous effective; 110: pclk/16 continuous effective; 111: pclk/64 continuous effective Note: CRCH0.CSB needs to be configured as 1	RW
b3	Res	Reserved bit	Reserved, must remain at reset value	
b2~b0	FLTA0	CHA input filter control	0xx: Filter invalid 100: pclk/3 continuous effective; 101: pclk/4 continuous effective; 110: pclk/16 continuous effective; 111: pclk/64 continuous effective Note: CR0.CSA needs to be configured as 1	RW

14.4.3.7 Control Register (ATIMx_CR0)

Reset value: 0x0000 3000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res							CIEA	Res		CSB	CSA	Res			

Bit	Marking	Place name	Function	Read and write
b31~b9	Res	Reserved bit	Reserved, must remain at reset value	
b8	CIEA	Pulse width measurement complete interrupt enable	0: Prohibited 1: Enable	RW
b7~b6	Res	Reserved bit	Reserved, must remain at reset value	
b5	CSB	B channel capture / compare function selection	0: Compare Mode 1: capture mode	RW
b4	CSA	A channel capture / compare function selection	0: Compare Mode 1: capture mode	RW
b3~b0	Res	Reserved bit	Reserved, must remain at reset value	

14.4.3.8 Compare Capture Register (ATIMx_CCR0A)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CCR0A															

Bit	Marking	Place name	Function	Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value	
b15~b0	CCR0A	Pulse Width Measurement Results		RO

14.4.4 Mode 2,3 register description

14.4.4.1 Auto-Reload Register (ATIMx_ARR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
ARR															

Bit	Marking	Place name	Function	Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value	
b15~b0	ARR	Reload Register / Period Register	With cache function, when the counter is not enabled or the cache is not enabled, the cache register can be updated immediately	RW

14.4.4.2 16 -bit Mode Count Register (ATIMx_CNT)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CNT															

Bit	Marking	Place name	Function	Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value	
b15~b0	CNT	reload timer Count value register	-	RW

Note: When using PWM comparison output and **counting in triangle wave mode or sawtooth wave mode**, the initial CNT value needs to be less than the value of ARR.

14.4.4.3 Control Register (ATIMx_M23CR)

Reset value: 0x0060 0008

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res	UND	OV	DIR	BG	UG	TG	OCC	CIS	BI	TI	TD	URS	OCCS		
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CSG	Ones hot	Mode	UD	UI	CFG	CRG	BUFP	PRS		Pwm 2s	CT	COM P	CTEN		

Bit	Marking	Place name	Function	Read and write
b31~b30	Res	Reserved bit	Reserved, must remain at reset value	
b29	UND	Underflow interrupt enable	0: interrupt disabled 1: interrupt enable	RW
b28	OV	Overflow interrupt enable	0: interrupt disabled 1: interrupt enable	RW
b27	DIR	Counting direction	0: count up 1: count down Note: It can only be written in sawtooth wave mode. In other modes, read-only, write prohibition. Switching from other modes to center-aligned mode DIR is automatically cleared to 0. DIR is automatically cleared in reset mode by software event update and external trigger in slave mode.	RW/RO
b26	BG	Software brake, automatic reset	0: Writing 0 is invalid 1: Write 1 to generate software brake	W1
b25	UG	Software update, automatic reset	0: Writing 0 is invalid 1: Writing a 1 generates a software update Note: Initialize the counter and update the cache register to the corresponding register (cache enable), the pre-divider counter will also be cleared.	W1
b24	TG	Triggered by software, automatically cleared	0: Writing 0 is invalid 1: Write 1 to generate software trigger to be triggered under the trigger mode SMS=2 and mode=2/3	W1
b23	OCC	OCREF clear enable	0: OCREF output is not affected by OCREF_CLR 1: OCREF_CLR signal can clear OCREF output	RW
b22~b21	CIS	Channel A compare interrupt mode	00: no interrupt, 01: rising edge interrupt, 10: Falling edge interrupt, 11: Both the upper and lower edges are interrupted Note: The channel B compare interrupt is individually controlled in the CRCHx register CISB	RW
b20	BI	Brake Interrupt Enable	0: interrupt disabled 1: interrupt enable	RW
b19	TI	Trigger interrupt enable	0: interrupt disabled 1: interrupt enable	RW
b18	TD	Trigger DMA enable	0: trigger DMA disabled 1: trigger DMA enable	RW
b17	URS	Update source	0: overflow / underflow / software update UG/ slave mode reset; 1: overflow / underflow	RW
b16	OCCS	OCREF Clear Source Selection	0: Voltage comparator VC output, VC selection is set in VCx_OUTCFG register 1: ETR port filters the signal after phase selection When OCCE is valid, OC_clr can clear the comparison output signal of OCREF to zero, (valid when OCMx>1), and continue to compare the output after the next uev event	RW
b15	CSG	GATE capture / compare selection in PWM complementary mode	0: Compare 1: capture Note: Using CCR0B as the compare or capture channel of GATE is only valid for PWM complementary output.	RW
b14	Oneshot	Single trigger mode selection	0: loop count 1: Timer stops after an event update occurs.	RW
b13~b12	MODE	Operating mode	00 Timer mode 0; 01 PWC mode 10 sawtooth wave mode; 11 triangle wave mode	RW
b11	UD	Update Trigger DMA Enable	0: disable update trigger DMA 1: Enable update triggered DMA	RW
b10	UI	update interrupt enable	0: Disable update interrupt 1: Enable update interrupt Note: Counting to 0xFFFF will overflow and generate an overflow flag	RW

b9	CFG	When GATE is used as a capture input, the falling edge capture is effectively controlled	0: Falling edge capture is invalid 1: Falling edge capture is valid Note: only valid when PWM complementary output	RW
b8	CRG	When GATE is used as a capture input, the rising edge capture is effectively controlled	0: The rising edge capture is invalid 1: The upper edge capture is valid Note: only valid when PWM complementary output	RW
b7	BUFP	Auto-reload ARR cache enable	0: The period cache is invalid, and the period value will be affected immediately after writing 1: The period cache is enabled, and the period value will not be affected until the next event is updated after writing.	RW
b6~b4	PRS	Internal clock frequency division selection	000: 1; 001: 2; 010: 4; 011: 8; 100: 16; 101: 32; 110: 64; 111: 256;	RW
b3	PWM2S	OCREFA two-point comparison selection (default is 1)	0: Dual-point comparison is enabled, use CCRA, CCRB comparison to control OCREFA output 1: Single-point compare enable, only use CCRA compare to control OCREFA output Note: OCREFB is not affected, still use CCRB to control OCREFB output	RW
b2	CT	Counting clock selection	0: Internal count clock TCLK 1: External count clock ETR;	RW
b1	COMP	PWM complementary output mode selection	0: independent PWM output 1: Complementary PWM output	RW
b0	CTEN	Timer enable	0: forbidden; 1: Enable It can be enabled by external trigger, and this bit is automatically cleared at the end of oneshot mode	RW

14.4.4.4 Interrupt Flag Register (ATIMx_IFR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res													CA3	Und	OV
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
TI	BI	CB2E	CB1E	CB0E	CA2E	CA1E	CA0E	CB2F	CB1F	CB0F	CA2F	CA1F	CA0F	Res	UIF

Bit	Marking	Place name	Function	Read and write
b31~b19	Res	Reserved bit	Reserved, must remain at reset value	RO
b18	CA3	Channel CH3 compare match flag	0: nothing happened 1: occurs	RO
b17	UND	underflow interrupt flag	0: nothing happened 1: occurs	RO
b16	OV	overflow interrupt flag	0: nothing happened 1: occurs	RO
b15	TI	trigger interrupt flag	0: nothing happened 1: occurs	RO
b14	BI	Brake Interrupt Flag	0: nothing happened 1: occurs	RO
b13	CB2E	Channel CH2B capture data loss flag	0: no data loss 1: data loss	RO
b12	CB1E	Channel CH1B Capture Data Loss Flag	0: no data loss 1: data loss	RO
b11	CB0E	Channel CH0B Capture Data Loss Flag	0: no data loss 1: data loss	RO
b10	CA2E	Channel CH2A Capture Data Loss Flag	0: no data loss 1: data loss	RO
b9	CA1E	Channel CH1A Capture Data Loss Flag	0: no data loss 1: data loss	RO
b8	CA0E	Channel CH0A Capture Data Loss Flag	0: no data loss 1: data loss	RO
b7	CB2F	Channel CH2B capture / compare match flag	0: nothing happened 1: occurs	RO
b6	CB1F	Channel CH1B capture / compare match flag	0: nothing happened 1: occurs	RO
b5	CB0F	Channel CH0B capture / compare match flag	0: nothing happened 1: occurs	RO
b4	CA2F	Channel CH2A capture / compare match flag	0: nothing happened 1: occurs	RO
b3	CA1F	Channel CH1A capture / compare match flag	0: nothing happened 1: occurs	RO
b2	CA0F	Channel CH0A capture / compare match flag	0: nothing happened 1: occurs	RO
b1	Res	Reserved bit	Reserved, must remain at reset value	
b0	UIF	Event Update Interrupt Flag	0: nothing happened 1: occurs	RO

14.4.4.5 Interrupt Flag Clear Register (ATIMx_ICLR)

Reset value: 0x0000 FFFF

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res													CA3	Und	OV
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
TI	BI	CB2E	CB1E	CB0E	CA2E	CA1E	CA0E	CB2F	CB1F	CB0F	CA2F	CA1F	CA0F	Res.	UIF

Bit	Marking	Place name	Function	Read write and
b31~b19	Res	Reserved bit	Reserved, must remain at reset value	R1W0
b18	CA3	Channel CH3 compare match flag clear	0: clear channel CH3A comparison match flag 1: no function	R1W0
b17	UND	Underflow interrupt flag cleared	0: Clear the underflow interrupt flag 1: no function	R1W0
b16	OV	overflow interrupt flag clear	0: Clear overflow interrupt flag 1: no function	R1W0
b15	TI	trigger interrupt flag clear	0: Clear trigger interrupt flag 1: no function	R1W0
b14	BI	Brake Interrupt Flag Clear	0: Clear brake interrupt flag 1: no function	R1W0
b13	CB2E	Channel CH2B capture data loss flag cleared	0: clear channel CH2B capture data loss flag 1: no function	R1W0
b12	CB1E	Channel CH1B capture data loss flag cleared	0: clear channel CH1B capture data loss flag 1: no function	R1W0
b11	CB0E	Channel CH0B capture data loss flag cleared	0: clear channel CH0B capture data loss flag 1: no function	R1W0
b10	CA2E	Channel CH2A capture data loss flag cleared	0: clear channel CH2A capture data loss flag 1: no function	R1W0
b9	CA1E	Channel CH1A capture data loss flag cleared	0: clear channel CH1A capture data loss flag 1: no function	R1W0
b8	CA0E	Channel CH0A capture data loss flag cleared	0: Clear channel CH0A capture data loss flag 1: no function	R1W0
b7	CB2F	Channel CH2B capture / compare match flag clear	0: clear channel CH2B capture / compare match flag 1: no function	R1W0
b6	CB1F	Channel CH1B capture / compare match flag clear	0: clear channel CH1B capture/compare match flag 1: no function	R1W0
b5	CB0F	Channel CH0B capture / compare match flag clear	0: clear channel CH0B capture/compare match flag 1: no function	R1W0
b4	CA2F	Channel CH2A capture / compare match flag clear	0: clear channel CH2A capture/compare match flag 1: no function	R1W0
b3	CA1F	Channel CH1A capture / compare match flag clear	0: clear channel CH1A capture/compare match flag 1: no function	R1W0
b2	CA0F	Channel CH0A capture / compare match flag clear	0: Clear channel CH0A capture / compare match flag 1: no function	R1W0
b1	Res	Reserved bit	Reserved, must remain at reset value	
b0	UIF	event update interrupt clear	0: Clear event update interrupt flag 1: no function	R1W0

14.4.4.6 Master-Slave Mode Control Register (ATIMx_MSCR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res		IB0S	IA0S	SMS			TS			MSM	CCDS	MMS			

Bit	Marking	Place name	Function	Read and write
b31~b13	Res	Reserved bit	Reserved, must remain at reset value	RW
b12	IB0S	CH0B input selection	0: CH0B; 1: internal trigger TS selection signal;	RW
b11	IA0S	IA0 input selection	0: CH0A; 1: CH0A CH1A CH2A XOR(ATIM3) 0: CHA; 1: CHA ETR GATE XOR(ATIM0) Note: After setting to 1, any port change of the port will cause the input to change	RW
b10~b8	SMS	Select from mode function	000: use the internal clock; 001: Reset function; 010: Trigger mode; 011: External Clock Mode 100: Orthogonal encoding counting mode 1; 101: Orthogonal encoding counting mode 2; 110: Orthogonal encoding counting mode 3; 111: Gating function	RW
b7~b5	TS	Trigger selection	000: the signal ETRP after the filter phase selection of the port ETR; 001: Internal interconnection signal ITR0 010: internal interconnection signal ITR1; 011: internal interconnection signal ITR2; 100: internal interconnection signal ITR3; 101: edge signal of port CH0A; 110: Filtered phase-selected signal IAFP of port CH0A 111: signal IBFP after filter phase selection of port CH0B;	RW
b4	MSM	master-slave selection	0: no delay 1: Delay enabled, so that the main sending counter starts at the same time. Note: When using the trigger mode, the slave mode is set to 0, and the master mode is set to 1, so that the master and slave counts can be started at the same time	RW
b3	CCDS	DMA compare trigger selection in compare mode	0: compare match triggers DMA; 1: Compare match does not trigger DMA, event update instead of compare match triggers DMA	RW
b2~b0	MMS	Master mode output selection	For internal interconnection, connected to ITRx of other timers 000: software update UG, write CR.UG 001: Timer enable CTEN 010: Timer event updates UEV; 011: compare match select output CMPSO; 100: Timer comparison parameter output OCREFOA 101: ATIM0: Timer comparison parameter output OCREFOB ATIM3: Timer comparison parameter output OCREF1A 110: ATIM0: Timer comparison parameter output OCREFOB ATIM3: Timer comparison parameter output OCREF2A 111: Timer comparison parameter output OCREFOB	RW

14.4.4.7 Output Control/ Input Filtering (ATIMx_FLTR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
ETP	FLTET			BKP	FLTBK			CCPB2	OCMB2			CCPA2	OCMA2		
-	-			-	-			-	FLTB2			-	FLTA2		
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CCPB1	OCMB1			CCPA1	OCMA1			CCPB0	OCMB0			CCPA0	OCMA0		
-	FLTB1			-	FLTA1			CCPB0	FLTB0			CCPA0	FLTA0		

Bit	Marking	Place name	Function	Read and write
b31	ETP	ETR input phase selection	0: same phase; 1: reverse input;	RW
b30~b28	FLTET	ETR filter control	0xx: Filter invalid 100: pclk/3 continuous effective; 101: pclk/4 continuous effective 110: pclk/16 continuous effective; 111: pclk/64 continuous effective	RW
b27	BKP	Brake BK input phase selection	0: same-phase input 1: reverse input;	RW
b26~b24	FLTBK	Brake input filter control	Filter settings 0xx: Filter invalid 100: pclk/3 continuous effective; 101: pclk/4 continuous effective 110: pclk/16 continuous effective; 111: pclk/64 continuous effective	RW
b23	CCPB2	CH2B channel comparison output polarity control	0: normal output; 1: Reverse output	RW
b22~b20	OCMB2 FLTB2	CH2B channel comparison control / CH2B channel input filter setting	Comparison function: CH2B channel comparison control; refer to OCMB0 Capture function: CH2B channel input filter setting, refer to FLTBK	RW
b19	CCPA2	CH2A channel comparison output polarity control	0: normal output; 1: Reverse output	RW
b18~b16	OCMA2 FLTA2	CH2A channel comparison control capture function / CH2A channel input filter setting	Comparison function: CH2A channel comparison control; refer to OCMB0 Capture function: CH2A channel input filter setting, refer to FLTBK	RW
b15	CCPB1	CH1B channel comparison output polarity control	0: normal output; 1: Reverse output	RW
b14~b12	OCMB1 FLTB1	CH1B channel comparison control / CH1B channel input filter setting	Comparison function: CH1B channel comparison control; refer to OCMB0 Capture function: CH1B channel input filter setting, refer to FLTBK	RW
b11	CCPA1	CH1A channel comparison output polarity control	0: normal output; 1: Reverse output	RW
b10~b8	OCMA1 FLTA1	CH1A channel comparison control capture function / CH1A channel input filter setting	Comparison function: CH1A channel comparison control; refer to OCMB0 Capture function: CH1A channel input filter setting, refer to FLTBK	RW
b7	CCPB0	CH0B port comparison output polarity control / CH0B port input phase and polarity control	Compare function: output compare mode CH0B port comparison output polarity control 0: normal output; 1: Reverse output Encode Count and Slave Mode Gating Function: Input Phase Control Slave mode gating, reset, external trigger, external clock using CH0B port input polarity control 0: normal input; 1: reverse input	RW
b6~b4	OCMB0 FLTB0	CH0B channel compare control / CH0B channel input filter setting	Comparison function: CHxB channel comparison control: 000: Forced to 0 001: Forced to 1 010: Forced to 0 when comparing matches 011: Forced to 1 when comparing matches 100: flip when comparison matches	RW

			101: Output a high level for one count period when comparing matches 110: PWM mode 1 Single point comparison: When counting up, $CNT < CCR_{xy}$ outputs high, and when counting down, $CNT > CCR_{xy}$ outputs low level Two-point comparison: 1) Counting $CCR_A < CNT \leq CCR_B$ output on the sawtooth wave is low level 2) Counting under sawtooth wave $CCR_A < CNT \leq CCR_B$ output is high level 3) Triangular wave up counting $CNT < CCR_A$ output high, down counting $CNT > CCR_B$ is low level 111: PWM mode 2 Single point comparison: When counting up, $CNT < CCR_{xy}$ output is low, when counting down, $CNT > CCR_{xy}$ output is high level Two-point comparison: 1) Counting $CCR_A \leq CNT < CCR_B$ output on the sawtooth wave is high level 2) Counting under sawtooth wave $CCR_A \leq CNT < CCR_B$ output is low level 3) Triangular wave up count $CNT < CCR_A$ output low, down count $CNT > CCR_B$ is high level Capture function: CH0B input channel filter setting, refer to FLTBK	
b3	CCPA0	CH0A port comparison output polarity control / CH0B input phase and polarity control	Compare function: CH0A port comparison output polarity control: 0: normal output; 1: Reverse output Encode Count and Slave Mode Gating Function: Input Phase Control Slave mode gating, reset, external trigger, external clock using CH0A port input polarity control: 0: normal input; 1: reverse input	RW
b2~b0	OCMA0/ FLTA0	CH0A channel compare control / CH0A channel input filter setting	Comparison function: A channel comparison control; refer to OCMB0 Capture function: CH0A input channel filter setting, refer to FLTBK	RW

14.4.4.8 ADC Trigger Control Register (ATIMx_ADTR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res								ADT	CMB2	CMB1	CMB0	CMA2	CMA1	CMA0	UEV

Bit	Marking	Place name	Function	Read and write
b31~b8	Res	Reserved bit	Reserved, must remain at reset value	
b7	ADT	Enable ADC trigger global control	0: Prohibited 1: Enable	RW
b6	CMB2	Channel CH2B comparison match triggers ADC enable	0: Prohibited 1: Enable	RW
b5	CMB1	Channel CH1B comparison match triggers ADC enable	0: Prohibited 1: Enable	RW
b4	CMB0	Channel CH0B comparison match triggers ADC enable	0: Prohibited 1: Enable	RW
b3	CMA2	Channel CH2A comparison match triggers ADC enable	0: Prohibited 1: Enable	RW
b2	CMA1	Channel CH1A compare match trigger ADC enable	0: Prohibited 1: Enable	RW
b1	CMA0	Channel CH0A comparison match triggers ADC enable	0: Prohibited 1: Enable	RW
b0	UEV	Event update triggers ADC enable	0: Prohibited 1: Enable	RW

14.4.4.9 Channel 0 Control Register (ATIMx_CRCH0)

Reset value: 0x0000 3000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CCGB	CCGA	CISB	CDEB	CDEA	CIEB	CIEA	-	-	CSB	CSA	CFB	CRB	CFA	CRA	
CCGB	CCGA	CISB	CDEB	CDEA	CIEB	CIEA	BUFEB	BUFEA	CSB	CSA	bksb		bkas		

Bit	Marking	Place name	Function	Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value	
b15	CCGB	Capture compare B software trigger, hardware automatic clear	Only interrupts are generated in compare mode; In capture mode an interrupt is generated and the counter value is captured into the capture register.	RW
b14	CCGA	Capture comparison A is triggered by software and automatically cleared by hardware	Only interrupts are generated in compare mode; In capture mode an interrupt is generated and the counter value is captured into the capture register.	RW
b13~b12	CISB	B channel compare match setting	00 no match 01 rising match 10 drop matches 11 double matches	RW
b11	CDEB	B capture compare trigger DMA enable	0: Prohibited 1: Enable	RW
b10	CDEA	A capture compare trigger DMA enable	0: Prohibited 1: Enable	RW
b9	CIEB	B capture compare trigger interrupt enable	0: Prohibited 1: Enable	RW
b8	CIEA	A capture compare trigger interrupt enable	0: Prohibited 1: Enable	RW
b7	BUFEB	Compare Function: B Compare Cache Enable Control	0: Prohibited 1: Enable	RW
b6	BUFEA	Compare Function: A Compare Cache Enable Control	0: Prohibited 1: Enable	RW
b5	CSB	B channel capture / compare function selection	0: Compare Mode 1: capture mode	RW
b4	CSA	A channel capture / compare function selection	0: Compare Mode 1: capture mode	RW
b3	CFB	B channel falling edge capture enable	0: Prohibited 1: Enable	RW
b2	CRB	B channel rising edge capture enable	0: Prohibited 1: Enable	RW
b3~b2	BKSB	B channel comparison function output brake level control	00: High impedance output 01: keep the previous output 10: forced output low level 11: forced output high level; Note: no debug stop mode	RW
b1	CFA	A channel falling edge capture enable	0: Prohibited 1: Enable	RW
b0	CRA	A channel rising edge capture enable	0: Prohibited 1: Enable	RW
b1~b0	BKSA	A channel comparison function output brake level control	00: High impedance output 01: keep the previous output 10: forced output low level 11: Force output high level Note:(In debug stop mode, the brake level control of A and B channels are controlled by BKSA)	RW

14.4.4.10 Channel 1/2 Control Register (ATIM3_CRCH1/2)

Reset value: 0x0000 3000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CCGB	CCGA	CISB	CDEB	CDEA	CIEB	CIEA	-	-	CSB	CSA	CFB	CRB	CFA	CRA	
CCGB	CCGA	CISB	CDEB	CDEA	CIEB	CIEA	BUFEB	BUFEA	CSB	CSA	bksb		bkas		

Bit	Marking	Place name	Function	Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value	
b15	CCGB	Capture compare B software trigger, hardware automatic clear	Only interrupts are generated in compare mode; In capture mode an interrupt is generated and the counter value is captured into the capture register.	RW
b14	CCGA	Capture comparison A is triggered by software and automatically cleared by hardware	Only interrupts are generated in compare mode; In capture mode an interrupt is generated and the counter value is captured into the capture register.	RW
b13~b12	CISB	B channel compare match setting	00 no match 01 rising match 10 drop matches 11 double matches	RW
b11	CDEB	Reserved bit	Reserved, must remain at reset value	
b10	CDEA	Reserved bit	Reserved, must remain at reset value	
b9	CIEB	B capture compare trigger interrupt enable	0: Prohibited 1: Enable	RW
b8	CIEA	A capture compare trigger interrupt enable	0: Prohibited 1: Enable	RW
b7	BUFEB	Compare Function: B Compare Cache Enable Control	0: Prohibited 1: Enable	RW
b6	BUFEA	Compare Function: A Compare Cache Enable Control	0: Prohibited 1: Enable	RW
b5	CSB	B channel capture / compare function selection	0: Compare Mode 1: capture mode	RW
b4	CSA	A channel capture / compare function selection	0: Compare Mode 1: capture mode	RW
b3	CFB	B channel falling edge capture enable	0: Prohibited 1: Enable	RW
b2	CRB	B channel rising edge capture enable	0: Prohibited 1: Enable	RW
b3~b2	BKSB	B channel comparison function output brake level control	00: High impedance output 01: keep the previous output 10: forced output low level 11: forced output high level; Note: no debug_stop mode	RW
b1	CFA	A channel falling edge capture enable	0: Prohibited 1: Enable	RW
b0	CRA	A channel rising edge capture enable	0: Prohibited 1: Enable	RW
b1~b0	BKSA	A channel comparison function output brake level control	00: High impedance output 01: keep the previous output 10: forced output low level 11: Force output high level Note:(In debug stop mode, the brake level control of A and B channels are controlled by BKSA)	RW

14.4.4.11 Dead Time Register (ATIMx_DTR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res.	VCE	Safe	MO	AO	BK	DT	Bksel	DTR							

Bit	Marking	Place name	Function	Read and write	
b31~b15	Res	Reserved bit	Reserved, must remain at reset value		
b14	VCE	VC brake enable	0: Prohibited 1: Enable	RW	
b13	Safeen	Safety brake enable (osc fail,brown down,lockup)	0: Prohibited 1: Enable	RW	
b12	MO	PWM output enable	0: Prohibited 1: Enable	RW	
b11	AO	PWM output is automatically enabled	0: Prohibited 1: Enable	RW	
b10	BK	Port brake enable	0: Prohibited 1: Enable	RW	
b9	DT	Dead zone control enable	0: Prohibited 1: Enable	RW	
b8	bksel	Reserved bit	Reserved, must remain at reset value	RW	
b7~b0	DTR	Dead Time Register	DTR[7] =0 DTR[7:6] =10 DTR[7:5] =110 DTR[7:5] =111 T=DTR[6:0]+2 T={DTR[5:0]+64}*2 +2 T={DTR[4:0]+32}*8 +2 T={DTR[4:0]+32}*16 +2	2-129 step=1 130-256 step=2 258-506 step=8 514-1010 step=16	RW

14.4.4.12 Repeat Count Register (ATIMx_RCR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res						UD	OV	RCR							

Bit	Marking	Place name	Function	Read and write
b31~b10	Res	Reserved bit	Reserved, must remain at reset value	
b9	UD	Overflow masking under repeat count control	0: Down counting overflow enable 1: down count overflow mask	RW
b8	OV	Overflow masking on repeat count control	0: Up counting overflow enable 1: upper count overflow mask	RW
b7~b0	RCR	Recurrence count value	Set RCR+1 cycle, the overflow controlled by [9:8] generates an event update, the internal RCR_CNT is decremented by 1 when the counter overflows or underflows, when the count reaches zero, RCR_CNT reloads the value of RCR, and generates an event to update the UEV signal	RW

14.4.4.13 Channel 0 Compare Capture Register (ATIMx_CCR0A/B)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CCR0y															

Bit	Marking	Place name	Function	Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value	
b15~b0	CCR0y	compare capture register	As a comparison, it has a cache function (y=A,B)	RW

14.4.4.14 Channel 1/2 Compare Capture Register (ATIM3_CCR1/2 A/B)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CCRxy															

Bit	Marking	Place name	Function	Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value	
b15~b0	CCR0y	compare capture register	As a comparison, it has a cache function (x=1,2; y=A,B)	RW

14.4.4.15 Channel 3 Compare Capture Register (ATIM3_CCR3)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CCR3															

Bit	Marking	Place name	Function	Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value	
b15~b0	CCR3	compare capture register	As a comparison with a cache function	RW

14.4.4.16 Channel 3 Control Register (ATIM3_CRCH3)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res										C3	CIS		CD	CI	BUF

Bit	Marking	Place name	Function	Read and write
b31~b6	Res	Reserved bit	Reserved, must remain at reset value	
b5	C3	Channel Compare Enable	0: Prohibited 1: Enable	RW
b4~b3	CIS	Channel Compare Match Settings	00 no match 01 rising match 10 drop matches 11 double matches	RW
b2	CD	Compare Trigger DMA Enable	0: Prohibited 1: Enable	RW
b1	CI	Compare Trigger Interrupt Enable	0: Prohibited 1: Enable	RW
b0	BUF	Compare Cache Enable Control	0: Prohibited 1: Enable	RW

15 Independent Watchdog (IWDT)

15.1 Overview

The device features an embedded stand-alone watchdog peripheral for high security and flexibility of use. This independent watchdog detects and resolves failures caused by software errors, triggering an interrupt or reset when its count reaches a programmed timeout value. The watchdog is driven by the internal low-speed clock (RC10K), and it can still work normally even if the main clock fails.

15.2 Main characteristics

- Independent RC Oscillator
- Overflow period 400 μ s - 209s
- Trigger interrupt or reset on overflow
- A running watchdog can be stopped by a specific write sequence
- Automatically pause counting in DeepSleep mode

15.3 Functional description

15.3.1 Functional block diagram

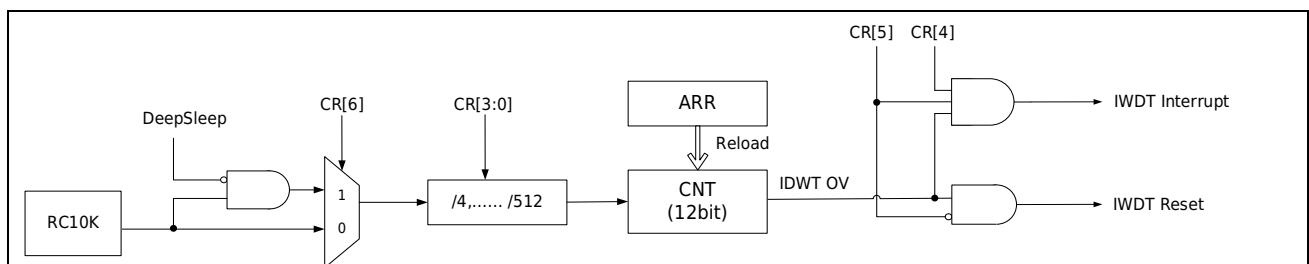


Figure 15-1 WDT functional block diagram

After starting the watchdog, the counter starts counting down from 0xFF. 0xAAAA is written to the IWDT_KR register, the value of IWDT_ARR is reloaded to the counter. When the count value is decremented to 0x000, the watchdog will generate an overflow signal; when the counter is reloaded, the count value is greater than the value of the IWDT_WINR register and an overflow signal will also be generated. Configure the corresponding control bit of IWDT_CR to make reset or interrupt after watchdog overflow. By configuring the corresponding control bit of IWDT_CR, the watchdog can automatically suspend counting during DeepSleep.

15.3.2 Window watchdog mode

When the configured value of IWDT_WINR register is less than IWDT_ARR, IWDT works in window watchdog mode. In this mode, there are two conditions to trigger the watchdog overflow: the count value is decremented to 0x000, and the watchdog is reloaded when the count value is greater than

the window value. When the IWDT_WINR register value is changed, the hardware will automatically reload the watchdog to reset the down counter value to IWDT_ARR. The user program should reload the watchdog before the count value is less than or equal to the IWDT_WINR register value and the count value is decremented to 0x000.

15.3.3 Independent Watchdog Mode

When the value of IWDT_WINR register is greater than or equal to IWDT_ARR, IWDT works in independent watchdog mode. In this mode, there is only one condition that triggers a watchdog overflow: the count value decrements to 0x000. The user program should reload the watchdog before the count value is decremented to 0x000.

15.3.4 Watchdog overflow handling

The overflow signal of the watchdog can trigger an interrupt or a reset. IWDT_CR.ACTION is set to 0, the watchdog overflow signal will reset the MCU; when IWDT_CR.ACTION is set to 1 and IWDT_CR.IE is 1, the watchdog overflow signal will trigger an interrupt.

15.3.5 Watchdog timeout

If IWDT_CR.PAUSE is set to 1, the watchdog automatically pauses when the MCU enters DeepSleep; when the MCU exits DeepSleep, the watchdog continues to run. Using this feature, the MCU does not need to wake up frequently to feed the dog when it is in a low power consumption state, and can achieve lower standby power consumption.

15.3.6 Watchdog stop and run again

Write 0x1234, 0x5678 to the IWDT_KR register in sequence to stop the running watchdog.

Writing 0xCCCC to the IWDT_KR register restarts the watchdog with its configuration unchanged.

15.3.7 Register write protection

The IWDT_CR, IWDT_ARR, and IWDT_WINR registers are write-protected. If you need to modify these three registers, you should write 0x5555 to the IWDT_KR register to release the write protection of these registers; writing other values to the IWDT_KR register will enable the write protection of these registers.

15.4 Programming example

15.4.1 Configure IWDT as window watchdog

- Step1. Write 0xCCCC to the IWDT_KR register to enable the IWDT.
- Step2. Delay 500μs.
- Step3. Write 0x5555 to the IWDT_KR register to release the write protection of the register.
- Step4. Configure IWDT_CR, select the counting clock frequency of the watchdog, the action after overflow, and the DeepSleep feature.
- Step5. Wait for IWDT_SR.PRSF to be 0.
- Step6. Configure IWDT_ARR to select the overflow period of the watchdog.
- Step7. Wait for IWDT_SR.ARRF to be 0.
- Step8. Write the required value to IWDT_WINR to configure the window size.
- Step9. Wait for IWDT_SR.WINRF to become 0, and the auto-refresh counter value is IWDT_ARR.
- Step10. Before the count value is less than or equal to the IWDT_WINR register value and the count value is decremented to 0, send to IWDT_KR 0xAAAA to the register to reload the IWDT.

15.4.2 Configure IWDT as an independent watchdog

- Step1. Write 0xCCCC to the IWDT_KR register to enable the IWDT.
- Step2. Delay 500μs.
- Step3. Write 0x5555 to the IWDT_KR register to release the write protection of the register.
- Step4. Configure IWDT_CR, select the counting clock frequency of the watchdog, the action after overflow, and the DeepSleep feature.
- Step5. Wait for IWDT_SR.PRSF to be 0.
- Step6. Configure IWDT_ARR to select the overflow period of the watchdog.
- Step7. Wait for IWDT_SR.ARRF to be 0.
- Step8. The refresh counter value is the value of IWDT_ARR (IWDT_KR = 0x0000 AAAA).
- Step9. the count value is decremented to 0, write 0xAAAA to the IWDT_KR register to reload the IWDT.

15.4.3 Watchdog timeout (10kHz independent clock)

Table 15-1 IWDt timeout time (10kHz independent clock)

Prescaler coefficient	PRS[2:0] bits	Minimum time (ms) ARR[11:0]=0x000	Maximum time (ms) ARR[11:0]=0xFFF
4 frequency divisions	000	$0.1 \times 4 = 0.4$	$0.1 \times 4 \times 4096$
8 frequency division	001	0.1×8	$0.1 \times 8 \times 4096$
16 frequency division	010	0.1×16	$0.1 \times 16 \times 4096$
32 frequency division	011	0.1×32	$0.1 \times 32 \times 4096$
64 frequency division	100	0.1×64	$0.1 \times 64 \times 4096$
128 frequency division	101	0.1×128	$0.1 \times 128 \times 4096$
256 frequency division	110	0.1×256	$0.1 \times 256 \times 4096$
512 frequency division	111	0.1×512	$0.1 \times 512 \times 4096 = 209715.2$

15.5 Register description

15.5.1 Register list

IWDT base address: 0x4000 8C80

Table 15-2 IWDT register list

Register name	Offset address	Reset value
Key-value register(IWDT_KR)	0x00	0x0000 0000
Control Register(IWDT_CR)	0x04	0x0000 0000
Reload Register(IWDT_ARR)	0x08	0x0000 0FFF
Status Register(IWDT_SR)	0x0C	0x0000 0000
Window Register(IWDT_WINR)	0x10	0x0000 0FFF

15.5.2 Key-value register (IWDT_KR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
KR															

Bit	Marking	Place name	Function	Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value	
b15~b0	KR	IWDT key-value register	Write 0xCCCC: start IWDT counter Write 0xAAAA: reload IWDT count value Write 0x1234 - 0x5678 sequentially: 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	WO

Note:

- After starting the IWDT counter, that is, after writing 0xCCCC to the IWDT_KR register to enable IWDT, a delay of at least 500μs should be added to wait for the clock to stabilize before reading and writing other registers of IWDT.

15.5.3 Control Register (IWDT_CR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res									PAUSE	ACTION	IE	Res	PRS		

Bit	Marking	Place name	Function	Read and write
b31~b7	Res	Reserved bit	Reserved, must remain at reset value	
b6	PAUSE	WDT pause control in DeepSleep mode	0: IWDT continues to run in DeepSleep mode 1: IWDT automatically pauses in DeepSleep mode	RW
b5	ACTION	Action configuration after IWDT overflow	0: A reset is generated after watchdog time-out 1: Generate interrupt after watchdog overflow	RW
b4	IE	IWDT interrupt enable control	0: Disable watchdog interrupt 1: Enable watchdog interrupt	RW
b3	Res	Reserved bit	Reserved, must remain at reset value	
b2~b0	PRS	the frequency division ratio of watchdog count clock and RC10K oscillator	111: 512 frequency division 011: 32 frequency division 110: 256 frequency division 010: 16 frequency division 101: 128 frequency division 001: 8 frequency division 100: divide by 64 000: divide by 4	RW

Note:

- This register can only be modified after writing 0x5555 to the IWDT_KR register.
- This register can only be modified when IWDT_SR.PRSF is 0.

15.5.4 Reload Register (IWDT_ARR)

Reset value: 0x0000 0FFF

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res				ARR											

Bit	Marking	Place name	Function	Read and write
b31~b12	Res	Reserved bit	Reserved, must remain at reset value	
b11~b0	ARR	IWDT reload value	IWDT reload value	RW

Note:

1. writing 0x5555 to the IWDT_KR register.
2. This register can only be modified when IWDT_SR.ARRF is 0.

15.5.5 Status Register (IWDT_SR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res											RUN	OV	WINRF	ARRF	PRSF

Bit	Marking	Place name	Function	Read and write
b31~b5	Res	Reserved bit	Reserved, must remain at reset value	
b4	RUN	WDT running flag	1: WDT is running 0: WDT is not running	RO
b3	OV	WDT overflow flag	1: WDT overflow, write 0 to clear 0: WDT has not overflowed	RW0
b2	WINRF	WINR register update flag	1: WINR register is being updated 0: WINR register update complete Note: The WINR register can only be updated when the WINRF bit is 0.	RO
b1	ARRF	ARR register update flag	1: ARR register is being updated 0: ARR register update completed Note: The ARR register can only be updated when the ARRF bit is 0.	RO
b0	PRSF	CR register update flag	1: CR register is being updated 0: CR register update completed Note: The CR register can be updated only when the PRSF is 0.	RO

15.5.6 Window Register (IWDT_WINR)

Reset value: 0x00000FFF

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res				WINR											

Bit	Marking	Place name	Function	Read and write
b31~b12	Res	Reserved bit	Reserved, must remain at reset value	
b11~b0	WINR	window comparator comparison value	window comparator comparison value	RW

Note:

1. writing 0x5555 to the IWDT_KR register.
2. IWDT_SR.WINRF is 0.

16 Window Watchdog (WWDT)

16.1 Overview

The window watchdog (WWDT) is often used to monitor software failures triggered by unexpected errors. An interrupt is generated when the counter value is decremented to 0x40, and a reset is generated when the counter value is decremented to 0x3F; if the user program updates the counter value when the counter value is greater than the window value, a reset is generated. WWDT counter within a limited window (the counter value is less than or equal to the window value and greater than 0x3F).

16.2 Main characteristics

- 6- bit down counter
- PCLK divided clock as count clock
- When PCLK is 48MHz, its timeout time is 85us ~ 699ms
- An interrupt is generated when the count value decrements to 0x40
- A reset is generated when the count value is decremented to 0x3F
- When the count value is greater than the window value, update the counter to generate a reset

16.3 Functional description

16.3.1 Functional block diagram

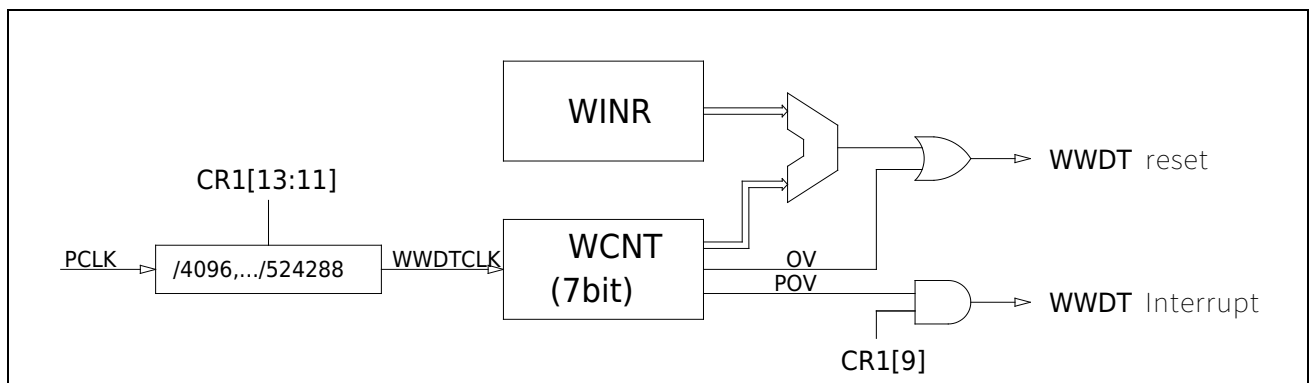


Figure 16-1 WWDT functional block diagram

After starting the watchdog, the value of the counter WCNT is decremented by 1 every rising edge of WWDTCLK. When the counter is decremented to 0x40, the watchdog will generate a pre-overflow signal POV, enabling WWDT_CR1.IE can make the watchdog generate a pre-overflow interrupt; when the counter is decremented to 0x3F, the watchdog will generate a reset signal. User program can write WWDT_CR0.WCNT to update WCNT to prevent watchdog from generating reset signal. If the value of WWDT_CR0.WCNT is greater than the value of WWDT_CR1.WINR when the user updates WWDT_CR0.WCNT, the watchdog will also generate a reset signal.

16.3.2 enable watchdog

After the system is reset, the watchdog is turned off. Set WWDT_CR0.EN to 1 to enable the watchdog. The user program cannot disable the enabled watchdog by writing 0 to WWDT_CR0.EN. writing 0xFF to WWDT_CR0 to ensure that the reset will not be triggered when the watchdog is enabled.

16.3.3 Configure watchdog timeout

The required watchdog timeout period can be obtained by properly configuring WWDT_CR0.WCNT and WWDT_CR1.PRS. When PCLK is 48MHz, the range of its timeout period is 85us ~ 699ms.

The formula for calculating the timeout period is as follows:

$$T_{\text{WWDT}} = T_{\text{PCLK}} \times 4096 \times 2^{\text{CR1.PRS}} \times (\text{CR0.WCNT} - 0x3F)$$

16.3.4 Update down counter

to WWDT_CR0.WCNT updates the WWDT down counter. If the written value is less than or equal to 0x3F, a reset will occur directly; if the current count value is greater than the window value when the counter is updated, a reset will occur directly. Therefore, the user program can only update the down counter when the value of the down counter is less than or equal to the window value and greater than 0x3F.

16.3.5 Pre-flow interrupt

If the decrement counter decrements to 0x40 and WWDT_CR1.IE is 1, WWDT will generate a pre-overflow interrupt. The user program can update the decrement counter in the interrupt service routine to prevent WWDT from being reset; the user program should write 0x00 to the WWDT_SR register before exiting the interrupt service to clear the WWDT pre-overflow flag.

16.4 Programming example

The following example configures the overflow period of the watchdog as 786432 PCLKs, and the window is 1/3 of the overflow period.

- Step1. Configure WWDT_CR1.PRS to 2, the count clock of WWDT is PCLK/16384;
- Step2. Configure WWDT_CR1.WINR as 0x4F, WWDT update window as 0x4F~0x40;
- Step3. Configure WWDT_CR0.WCNT as 0x6F, and the overflow period of WWDT is 786432 PCLKs;
- Step4. Set WWDT_CR0.EN to 1 to start the watchdog;
- Step5. When the down counter value is less than or equal to the window value and greater than 0x3F, write 0x6F to WWDT_CR0.WCNT to reload the watchdog counter.

16.5 Register description

16.5.1 Register list

WWDT base address: 0x4000 8800

Table 16-1 WWDT register list

Register name	Offset address	Reset value
Control Register 0 (WWDT_CR0)	0x00	0x0000 007F
Control Register 1 (WWDT_CR1)	0x04	0x0000 007F
Status Register(WWDT_SR)	0x08	0x0000 0000

Note: WWDT affects register value after reset.

16.5.2 Control Register 0 (WWDT_CR0)

Reset value: 0x0000 007F

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res								EN	WCNT						

Bit	Marking	Place name	Function	Read and write
b31~b8	Res	Reserved bit	Reserved, must remain at reset value	
b7	EN	WWDT enable control	After this bit is enabled, it cannot be cleared by software 1: Enable WWDT 0: Disable WWDT (status)	RW1
b6~b0	WCNT	WWDT counter	Read: WWDT counter current count value Write: Update the current count value of the WWDT counter	RW

16.5.3 Control Register 1 (WWDT_CR1)

Reset value: 0x0000 007F

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res		PRS			Res	IE	Res		WINR						

Bit	Marking	Place name	Function	Read and write
b31~b14	Res	Reserved bit	Reserved, must remain at reset value	
b13~b11	PRS	WWDT count clock configuration	000: PCLK / 4096 001: PCLK / 8192 010: PCLK / 16384 011: PCLK / 32768 100: PCLK / 65536 101: PCLK / 131072 110: PCLK / 262144 111: PCLK / 524288	RW
b10	Res	Reserved bit	Reserved, must remain at reset value	
b9	IE	WWDT pre-overflow interrupt enable control	1: Enable pre-overflow interrupt 0: Disable pre-overflow interrupt	RW
b8~b7	Res	Reserved bit	Reserved, must remain at reset value	
b6~b0	WINR	Watchdog window value	Watchdog window value	RW

16.5.4 Status Register(WWDT_SR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res															POV

Bit	Marking	Place name	Function	Read and write
b31~b1	Res	Reserved bit	Reserved, must remain at reset value	
b0	POV	WWDT pre-flow flag	1: WWDT has pre-flown, write 0 to clear this flag 0: WWDT pre-flow has not occurred	RW0

17 Universal Synchronous/ Asynchronous Receiver Transmitter (USART)

17.1 Overview

This product is equipped with 4 universal serial synchronous asynchronous transceiver modules (USART), which can flexibly exchange data with external devices.

The module supports general asynchronous serial communication, general synchronous serial communication, smart card protocol, infrared communication protocol and LIN protocol.

17.2 Main characteristics

- Two baud rate generation methods: integer frequency division, fractional frequency division
- Two data sequence order: MSB first, LSB first
- Built-in multi-function timer counter
- Support TXD, RXD pin swap
- Support TXD, RXD level inversion
- Support DMA continuous communication

17.2.1 Asynchronous Serial Port Features

- transfer data asynchronously
- Two frame lengths: 8 bits, 9 bits
- Frame check: odd check, even check, no check
- Two kinds of stop bits: 1 bit, 2 bits
- Support hardware flow control(CTS/RTS)
- Support single-wire half-duplex communication
- Support baud rate automatic detection
- Support for inter-processor communications
- Support sleep mode wake-up function

17.2.2 Synchronous Serial Port Features

- transfer data synchronously
- The data length is fixed at 8 bits
- Support hardware flow control(CTS/RTS)
- Two communication clock sources: internal clock source, external clock source

17.2.3 Smart Card Interface Features

- Communication protocol: ISO/IEC7816-3
- 8 ETU lengths: 16 ~ 512

- Support hardware error detection and retransmission
- Support IO signal edge integer
- Support host interface

17.2.4 Infrared Communication Features

- Support $\frac{3}{16}$ no carrier mode
- Support 38K carrier modulation mode, adjustable duty cycle, configurable modulation level
- Support sleep mode wake-up function

17.2.5 LIN Characteristic

- Support synchronous interval transmission and detection
- Support synchronous segment transmission and baud rate automatic detection
- Support sleep mode wake-up function

17.3 Functional description

This chapter will describe in detail the asynchronous serial mode, synchronous serial mode, smart card mode, infrared mode and LIN mode.

Asynchronous serial port, synchronous serial port, smart card, infrared and LIN functions can only activate one of them at a time.

Note: This module supports output signal inversion control(USART_CR2.TXINV) and input signal inversion control(USART_CR2.RXINV). Unless otherwise specified, the figures, tables and texts involved in this chapter are output inversion control and input inversion control. Phase control is off.

17.3.1 Communication baud rate programming

This module supports two baud rate generation methods: integer baud rate generation method and fractional baud rate generation method.

The formulas for the two baud rate generation methods are as follows:

$$\text{Baud}_{\text{integer}} = \frac{\text{PCLK}}{4^{\text{psc}} \times \text{sample count} \times (\text{DIVI} + 1)}$$

$$\text{Baud}_{\text{decimal}} = \frac{\text{PCLK} \times (128 + \text{DIVF})}{4^{\text{psc}} \times \text{sample count} \times (\text{DIVI} + 1) \times 256}$$

Table 17-1 USART each mode sampling count and maximum baud rate

Functional mode	Sample count	Maximum baud rate (unit: bps)
Asynchronous serial port	Sample Count = $8 \times (2 - \text{OVER8})$	PCLK/8
Synchronous serial port	Sample Count = 4	When internal clock source: PCLK/8 When external clock source: PCLK/6 (The maximum frequency requirement of the external input clock is PCLK/6)
smart card interface	Sample Count = $2 \times \text{BCN}$	PCLK/32
infrared communication	Sample Count = 8×2	Determined by IR modulation mode
LIN	Sample Count = $8 \times (2 - \text{OVER8})$	20K

Baud: Baud rate
PCLK: Timer
PSC: USART_BRR.PSC setting value
OVER8: USART_CR1.OVER8 setting value
BCN: The number of basic clocks set by USART_CR3.BCN
DIVI: USART_BRR.DIVI setting value
DIVF: USART_BRR.DIVF setting value

Note: Communication baud rate programming is applicable to asynchronous serial port mode, synchronous serial port mode, smart card mode, infrared mode and LIN mode

Knowing the communication baud rate, the steps to calculate the USART_BRR register value are as follows:

- Step1. Configure USART_BRR.PSC so that the calculation result of the following expression is less than or equal to 256;

$$\frac{PCLK}{4^{PSC} \times \text{Baud} \times \text{Sample count}}$$

- Step2. USART_BRR.DIVI according to the formula below, where the INT function represents rounding down;

$$\text{DIVI} = \text{INT} \left(\frac{PCLK}{4^{PSC} \times \text{Baud} \times \text{Sample count}} - 1 \right)$$

- Step3. USART_BRR.DIVF according to the formula below, where the INT function represents rounding down;

$$\text{DIVF} = \text{INT} \left(\frac{\text{Baud} \times \text{Sample count} \times (\text{DIVI} + 1) \times 256 \times 4^{PSC} \times 1.0}{PCLK} - 128 + 0.5 \right)$$

- Step4. When DIVF is less than 127.5, set USART_BRR.FBME to 1 to enable the fractional baud rate generator; when DIVF is greater than 127.5, set USART_BRR.FBME to 0 to disable the fractional baud rate generator.

17.3.1.1 Asynchronous serial communication baud rate programming example

Table 17-2 PCLK is 4MHz baud rate programming example

Target baud rate	PCLK=4MHz									
	OVER8=1					OVER8=0				
	PSC	DIVI	DIVF	Actual baud rate	Error%	PSC	DIVI	DIVF	Actual baud rate	Error%
2400	0	207	NA	2404	0.16%	0	103	NA	2404	0.16%
4800	0	103	NA	4808	0.16%	0	51	NA	4808	0.16%
9600	0	51	NA	9615	0.16%	0	25	NA	9615	0.16%
19200	0	25	NA	19231	0.16%	0	12	NA	19231	0.16%
76800	0	5	108	76823	0.03%	0	2	108	76823	0.03%
115200	0	3	108	115234	0.03%	0	1	108	115234	0.03%
230400	0	1	108	230469	0.03%	0	0	108	230469	0.03%
250000	0	1	NA	250000	0.00%	0	0	NA	250000	0.00%
500000	0	0	NA	500000	0.00%					

Table 17-3 PCLK is 8MHz baud rate programming example

Target baud rate	PCLK=8MHz									
	OVER8=1					OVER8=0				
	PSC	DIVI	DIVF	Actual baud rate	Error%	PSC	DIVI	DIVF	Actual baud rate	Error%
2400	1	103	NA	2404	0.16%	0	207	NA	2404	0.16%
4800	0	207	NA	4808	0.16%	0	103	NA	4808	0.16%
9600	0	103	NA	9615	0.16%	0	51	NA	9615	0.16%
19200	0	51	NA	19231	0.16%	0	25	NA	19231	0.16%
76800	0	12	NA	76923	0.16%	0	5	108	76823	0.03%
115200	0	7	108	115234	0.03%	0	3	108	115234	0.03%
230400	0	3	108	230469	0.03%	0	1	108	230469	0.03%
250000	0	3	NA	250000	0.00%	0	1	NA	250000	0.00%
500000	0	1	NA	500000	0.00%	0	0	NA	500000	0.00%
750000	0	0	64	750000	0.00%					
1000000	0	0	NA	1000000	0.00%					

Table 17-4 PCLK is 16MHz baud rate programming example

Target baud rate	PCLK=16MHz									
	OVER8=1					OVER8=0				
	PSC	DIVI	DIVF	Actual baud rate	Error%	PSC	DIVI	DIVF	Actual baud rate	Error%
2400	2	51	NA	2404	0.16%	1	103	NA	2404	0.16%
4800	1	103	NA	4808	0.16%	0	207	NA	4808	0.16%
9600	0	207	NA	9615	0.16%	0	103	NA	9615	0.16%
19200	0	103	NA	19231	0.16%	0	51	NA	19231	0.16%
76800	0	25	NA	76923	0.16%	0	12	NA	76923	0.16%
115200	0	16	123	115349	0.13%	0	7	108	115234	0.03%
230400	0	7	108	230469	0.03%	0	3	108	230469	0.03%
250000	0	7	NA	250000	0.00%	0	3	NA	250000	0.00%
500000	0	3	NA	500000	0.00%	0	1	NA	500000	0.00%
750000	0	1	64	750000	0.00%	0	0	64	750000	0.00%
1000000	0	1	NA	1000000	0.00%	0	0	NA	1000000	0.00%
1500000	0	0	64	1500000	0.00%					
2000000	0	0	NA	2000000	0.00%					

Table 17-5 PCLK is 24MHz baud rate programming example

Target baud rate	PCLK=24MHz									
	OVER8=1					OVER8=0				
	PSC	DIVI	DIVF	Actual baud rate	Error%	PSC	DIVI	DIVF	Actual baud rate	Error%
2400	2	77	NA	2404	0.16%	2	38	NA	2404	0.16%
4800	2	38	NA	4808	0.16%	1	77	NA	4808	0.16%
9600	1	77	NA	9615	0.16%	0	155	NA	9615	0.16%
19200	0	155	NA	19231	0.16%	0	77	NA	19231	0.16%
76800	0	38	NA	76923	0.16%	0	18	121	76789	-0.01%
115200	0	25	NA	115385	0.16%	0	12	NA	115385	0.16%
230400	0	12	NA	230769	0.16%	0	5	108	230469	0.03%
250000	0	11	NA	250000	0.00%	0	5	NA	250000	0.00%
500000	0	5	NA	500000	0.00%	0	2	NA	500000	0.00%
750000	0	3	NA	750000	0.00%	0	1	NA	750000	0.00%
1000000	0	2	NA	1000000	0.00%	0	0	43	1001953	0.20%
1500000	0	1	NA	1500000	0.00%	0	0	NA	1500000	0.00%
2000000	0	0	43	2003906	0.20%					
2500000	0	0	85	2496094	-0.16%					
3000000	0	0	NA	3000000	0.00%					

Table 17-6 PCLK is 32MHz baud rate programming example

Target baud rate	PCLK=32MHz									
	OVER8=1					OVER8=0				
	PSC	DIVI	DIVF	Actual baud rate	Error%	PSC	DIVI	DIVF	Actual baud rate	Error%
2400	2	103	NA	2404	0.16%	2	51	NA	2404	0.16%
4800	2	51	NA	4808	0.16%	1	103	NA	4808	0.16%
9600	1	103	NA	9615	0.16%	0	207	NA	9615	0.16%
19200	0	207	NA	19231	0.16%	0	103	NA	19231	0.16%
76800	0	51	NA	76923	0.16%	0	25	NA	76923	0.16%
115200	0	33	123	115349	0.13%	0	16	123	115349	0.13%
230400	0	16	123	230699	0.13%	0	7	108	230469	0.03%
250000	0	15	NA	250000	0.00%	0	7	NA	250000	0.00%
500000	0	7	NA	500000	0.00%	0	3	NA	500000	0.00%
750000	0	4	112	750000	0.00%	0	1	64	750000	0.00%
1000000	0	3	NA	1000000	0.00%	0	1	NA	1000000	0.00%
1500000	0	1	64	1500000	0.00%	0	0	64	1500000	0.00%
2000000	0	1	NA	2000000	0.00%	0	0	NA	2000000	0.00%
2500000	0	0	32	2500000	0.00%					
3000000	0	0	64	3000000	0.00%					
4000000	0	0	NA	4000000	0.00%					

Table 17-7 PCLK is 48MHz baud rate programming example

Target baud rate	PCLK=48MHz									
	OVER8=1					OVER8=0				
	PSC	DIVI	DIVF	Actual baud rate	Error %	PSC	DIVI	DIVF	Actual Baud rate	Error %
2400	2	155	NA	2404	0.16%	2	77	NA	2404	0.16%
4800	2	77	NA	4808	0.16%	2	38	NA	4808	0.16%
9600	2	38	NA	9615	0.16%	1	77	NA	9615	0.16%
19200	1	77	NA	19231	0.16%	0	155	NA	19231	0.16%
76800	0	77	NA	76923	0.16%	0	38	NA	76923	0.16%
115200	0	51	NA	115385	0.16%	0	25	NA	115385	0.16%
230400	0	25	NA	230769	0.16%	0	12	NA	230769	0.16%
250000	0	23	NA	250000	0.00%	0	11	NA	250000	0.00%
500000	0	11	NA	500000	0.00%	0	5	NA	500000	0.00%
750000	0	7	NA	750000	0.00%	0	3	NA	750000	0.00%
1000000	0	5	NA	1000000	0.00%	0	2	NA	1000000	0.00%
1500000	0	3	NA	1500000	0.00%	0	1	NA	1500000	0.00%
2000000	0	2	NA	2000000	0.00%	0	0	43	2003906	0.20%
2500000	0	1	85	2496094	-0.16%	0	0	85	2496094	-0.16%
3000000	0	1	NA	3000000	0.00%	0	0	NA	3000000	0.00%
4000000	0	0	43	4007812.5	0.20%					
6000000	0	0	NA	6000000	0.00%					

17.3.2 Auto baud rate detection

This function needs to be combined with a multi-function timer counter to realize automatic baud rate detection.

By setting USART_CR1.TOE=5 and USART_CR3.ABRE=1, the USART module enters the automatic baud rate detection mode. the hardware detects that there is an input data edge on RXD, the counter starts counting from 0, and the communication baud rate can be detected when the data frame of 0x55 is received. When the automatic baud rate detection is completed, stop counting, and the hardware automatically sets USART_SR.ABR to 1 and stores the baud rate detection value in the USART_BDR register. If USART_IER.ABR is set to 1, automatic baud rate detection can be generated interruption.

The communication baud rate can be calculated by the following formula:

$$\text{Baud} = \frac{f_{\text{PCLK}}}{4^{\text{PSC}} \times (\text{BDR} + 1)}$$

f_{PCLK} : PCLK frequency

PSC: the value of USART_BRR.PSC

BDR: The value of the USART_BDR register

When it is no longer necessary to detect the baud rate, set USART_CR3.ABRE to 0 to exit the automatic baud rate detection mode.

Note:

1. When the detected baud rate does not match the automatic baud rate detection range, the value of USART_BDR may overflow.
2. If the counter overflows during the automatic baud rate detection process, the hardware automatically sets USART_SR.TO to 1, and if the corresponding interrupt is enabled, the multi-function timer counter interrupt can be generated.
3. Communication baud rate detection is applicable to asynchronous serial port mode and LIN mode.

17.3.3 DMA continuous communication

The USART module supports the hardware handshaking logic of the DMAC.

- Write USART_CR2.TXDMAEN=1 to start the DMAC hardware handshake logic sent by USART. When the send buffer is empty, the USART will send a data transfer request to the DMAC. DMAC receives the sending request signal, it transfers the sending data of one frame from the DMAC source address to the USART_TDR. The above steps are repeated until all the data lengths configured in the DMAC are transferred.
- Write USART_CR2.RXDMAEN=1 to start the DMAC hardware handshake logic for USART reception. When a frame is received, the USART will send a data transfer request to the

DMAC. the DMAC receives the receive request signal, it transfers the received data from the USART_RDR to the target address of the DMAC. The above steps are repeated until all the data lengths configured in the DMAC are transferred.

Note:

1. DMA continuous communication is applicable to asynchronous serial mode, synchronous serial mode, smart card mode, infrared mode and LIN mode.
2. DMA mode is not supported when working in single-wire half-duplex.

17.3.4 Hardware flow control

Hardware flow control refers to controlling the serial data flow between two USART devices through CTS and RTS signals, and the connection method is shown in the figure below.

The hardware flow control function is divided into two types, RTS flow control function and CTS flow control function.

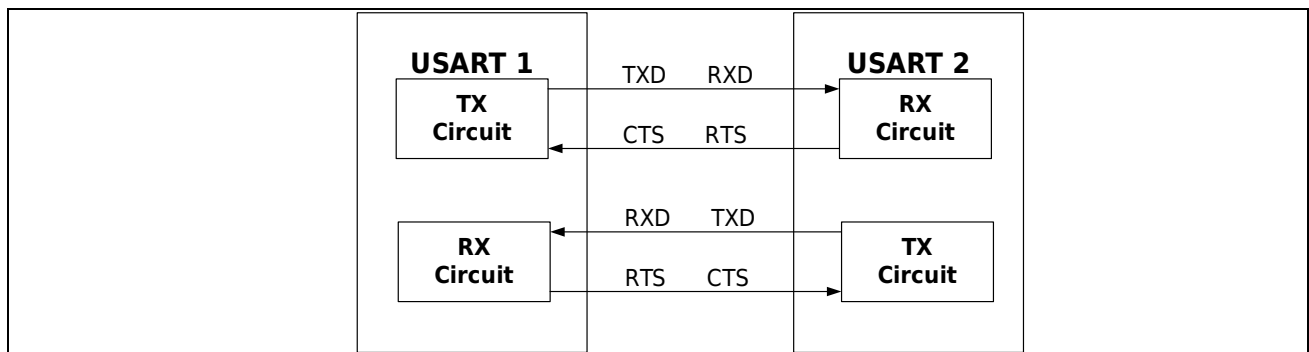


Figure 17-1 Asynchronous serial port mode hardware flow control example

CTS flow control function

When USART_CR2.CTSEN is set to 1, this module has CTS flow control function.

- When the CTS pin input is high level, the TXD pin will not send data out;
- When the CTS pin input is low level, the TXD pin sends out data;

Note:

If the input signal of the CTS pin becomes high level when the data is being sent, the data frame being sent will not be affected. After the current data frame is sent, the sending of new data will be suspended.

RTS flow control function

When USART_CR2.RTSN is set to 1, this module has RTS flow control function.

- When the receive buffer is empty, the RTS pin outputs a low level;
- When the receive buffer is full, the RTS pin outputs a high level;

RTS pin will output low level only when all the following conditions are met.

- 1) USART_CR1.RE is 1, the receiving circuit is enabled

- 2) USART_SR.RC is 0, the data in the USART_RDR register has been read
- 3) USART_SR.PE, USART_SR.FE and USART_SR.OR are 0, no receive data check error, receive frame error and overflow error

If all the above conditions cannot be met at the same time, the RTS pin will output a high level.

RTS pin will output low level only when all the following conditions are met.

- 1) Receive enable(USART_CR1.RE=1) and not receiving data
- 2) There is no unread data in the USART_RDR register(when USART_CR1.RE=1)
- 3) USART_TDR update completed(if USART_CR1.TE= 1)
- 4) USART_SR.OR is 0, no receive data overflow error

If all the above conditions cannot be met at the same time, the RTS pin will output a high level.

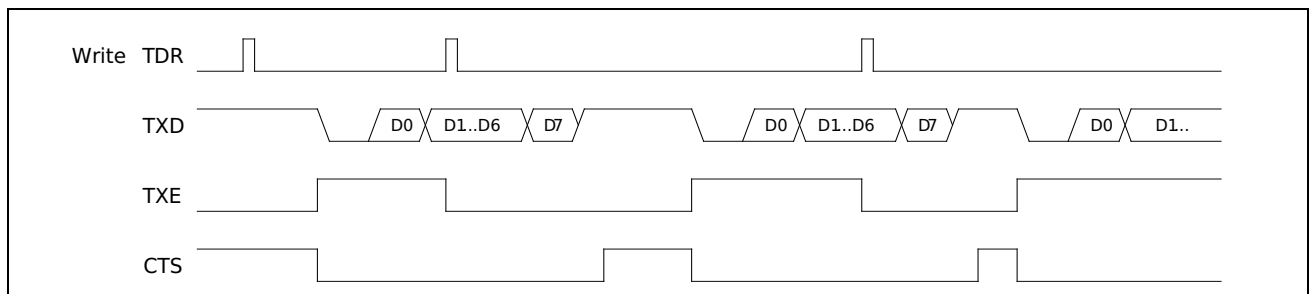


Figure 17-2 CTS hardware flow control signal

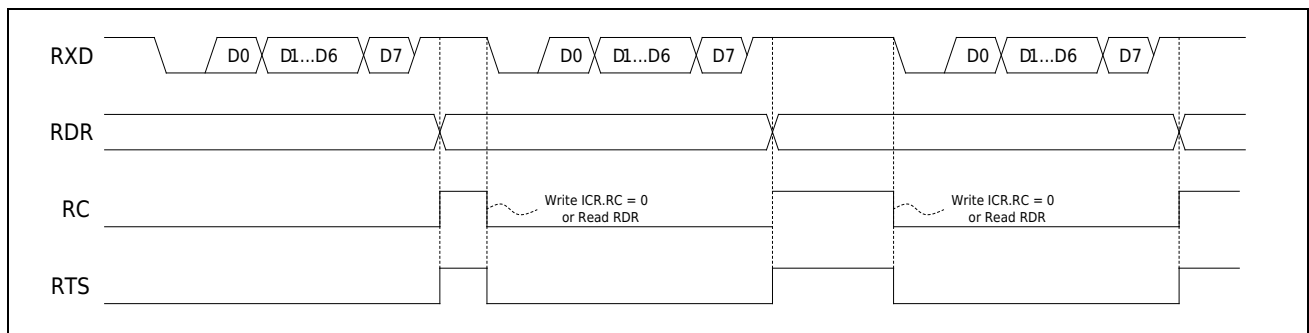


Figure 17-3 RTS hardware flow control signal

Note: Hardware flow control is applicable to asynchronous serial port mode and synchronous serial port mode.

17.3.5 Multi-function timer counter

This module is embedded with a multi-functional timing counter, which can realize five timing and counting functions.

Its regular counting cycle formula is $t_{PCLK} \times 4^{PSC} \times (TOR+1)$.

t_{PCLK} : Period of PCLK

PSC: the value of USART_BRR.PSC

TOR: the value of the USART_TOR register

When the PCLK frequency is 48MHz, the minimum value of its timing counting range is 0.02us, and the maximum value is about 1.4s.

This timer counter can realize five functions, and the comparison of their functions is as follows:

Table 17-8 Multifunctional timer counter function

Function	Counter start condition	Application Scenario
Universal timing	Start immediately after writing 1 to USART_CR1.TOE	Universal timer
Receive Timeout Count	Start immediately after writing 2 to USART_CR1.TOE	Wait for another host to reply with a message within a limited time
Receive inter-frame timeout count	Write 3 to USART_CR1.TOE and start after receiving the first frame of data	Wait for variable-length packet transmission to complete
Synchronous Interval Measurements	Write 4 to USART_CR1.TOE, after USART_CR3.LBDE=1, the hardware detects that there is a falling edge on RXD	LIN mode
Auto baud rate detection	Write 5 to USART_CR1.TOE, after USART_CR3.ABRE=1, the hardware detects that there is an input data edge on RXD	Baud rate detection (asynchronous serial port mode /LIN mode)

Note:

1. Only when USART_CR1.TOE is 0, other values can be written to start the timer counter.
2. When USART_SR.TO is 1, the hardware automatically sets USART_CR1.TOE to 0.

17.3.5.1 General purpose timing function

USART_CR1.TOE is 1, it works in general timing function. When 1 is written to USART_CR1.TOE, the counter starts immediately and starts counting from 0. If the count value is equal to USART_TOR, the hardware automatically sets USART_SR.TO to 1. If the corresponding interrupt is enabled, the multi-function timer counter interrupt can be generated.

17.3.5.2 Receive timeout count function

USART_CR1.TOE is 2, it works on receive timeout counting function. When 2 is written to USART_CR1.TOE, the counter starts immediately and starts counting from 0. If the count value is equal to USART_TOR and RXD has not received new data, it stops counting and the hardware automatically sets USART_SR.TO to 1. If the If the corresponding interrupt can be interrupted, the multi-function timer counter interrupt can be generated.

If the count value is less than USART_TOR when RXD receives new data(USART_SR.RC=1), the counter is cleared and the counting function is started again, counting from 0.

17.3.5.3 Receive inter-frame timeout counting function

USART_CR1.TOE is 3, it works on the timeout counting function between receiving frames. After writing 3 to USART_CR1.TOE, the counter starts counting from 0 after receiving the stop bit of the first byte. If the count value is equal to USART_TOR and RXD has not received new data, it stops counting and the hardware automatically Set USART_SR.TO to 1, if the corresponding interrupt is enabled, the multi-function timer counter interrupt can be generated.

If the count value is less than USART_TOR when RXD receives new data(USART_SR.RC=1), the counter is cleared and the counting function is started again, counting from 0.

17.3.5.4 Synchronous interval measurement function

USART_CR1.TOE is 4, it works in the synchronization interval measurement function. Write 4 to USART_CR1.TOE and write 1 to USART_CR2.LBDE, the hardware automatically recognizes the falling edge of the input signal RXD and counts the low level from 0, if the count value is equal to USART_TOR, then stop counting, and the hardware automatically sets USART_SR.LBD is 1, if the corresponding interrupt is enabled, the synchronization interval segment detection interrupt can be generated.

If the count value of the low-level length count is less than USART_TOR, wait for the next falling edge of RXD to enter a new round of synchronization interval measurement.

17.3.5.5 Auto baud rate detection function

USART_CR1.TOE is 5, it works in automatic baud rate detection function.

Refer to the Auto-Baud Detection chapter to learn more about the Auto-Baud Detection function.

17.3.6 Interrupt

Refer to the "External Interrupt and NVIC Interrupt Input Correspondence Table" in the chapter [Interrupt Controller (NVIC)]. The four USARTs in this chapter each provide three types of interrupts: TX interrupt, RX interrupt, and FLAG interrupt.

- 1) TX interrupt corresponds to USART_IER.TXE or USART_IER.TC interrupt enable
- 2) RX interrupt corresponding to USART_IER.RC interrupt enable
- 3) FLAG interrupt corresponds to any one or more interrupt enable in USART_IER[14:3]

Table 17-9 Validity of interrupts in each mode of USART

Functional name	Interrupt enable bit (USART_IER)	Asynchronous serial port	Synchronous serial port	Smart card	Infrared	LIN
Send Data Empty Interrupt	TXE	•	•	•	•	•
Send complete interrupt	TC	•	•	•	•	•
receive complete interrupt	RC	•	•	•	•	•
Receive data verification error interrupt	PE	•	X	•	•	X
Receive frame error interrupt	FE	•	X	•	•	•
Receive data overflow error interrupt	OR	•	•	•	•	•
Multi-function timing count interrupt	TO	•	•	•	•	•
Address match interrupt in multi-device communication mode	AM	•	X	X	X	X
CTS interrupt	CTS	•	•	X	X	X
Auto-baud detect interrupt	ABR	•	X	X	X	•
Sync Interval Detection Interrupt	LBD	X	X	X	X	•
Input data wake-up interrupt	DWK	•	X	X	•	•

Note:

•: valid

X: invalid

17.3.7 Asynchronous serial port mode

Asynchronous serial interface, which provides a flexible full-duplex data exchange interface for asynchronous transmission, is usually also used as RS232 standard communication.

The interface is connected with other devices through 1 to 4 pins, and usually only two pins are used in the application: RXD and TXD.

RXD: Receive data input, and distinguish data and noise through oversampling technology.

TXD: send data output, when not sending data, TXD pin keeps high level.

Select asynchronous serial port mode by setting USART_CR2.SYNC=0.

In asynchronous serial mode, the following bits must be cleared

USART_CR3.SCEN、USART_CR3.LINEN 和 USART_CR3.IREN

17.3.7.1 Functional block diagram

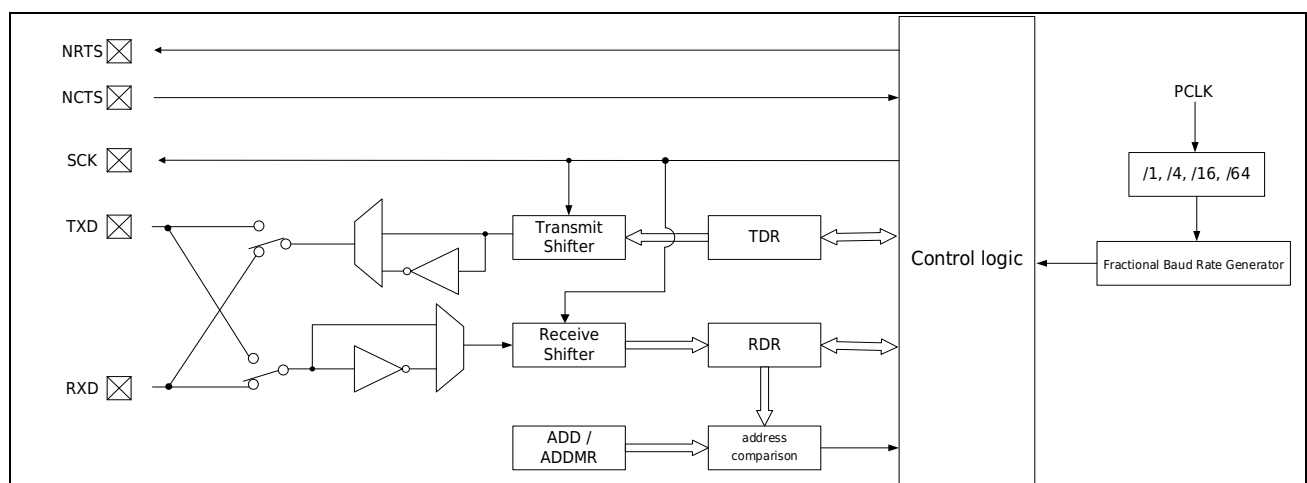


Figure 17-4 Functional block diagram of asynchronous serial port

The function description of each pin is as follows:

SCK: USART working clock output

TXD: USART send data pin

RXD: USART receive data pin

NRTS: USART hardware flow control RTS output pin

NCTS: USART hardware flow control CTS input pin

17.3.7.2 Data format

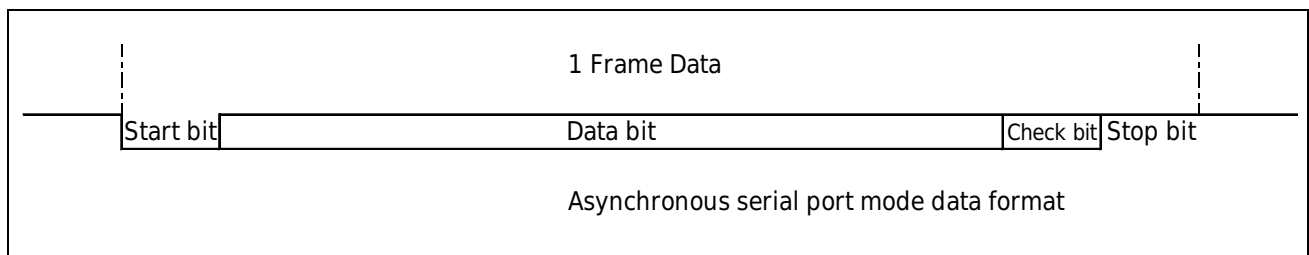


Figure 17-5 Asynchronous serial port mode data format

The format of the USART serial data frame is: start bit, 8 ~ 9 data bits, parity bit (optional), 1 ~ 2 stop bits.

The polarity of the start bit is opposite to the data line idle state, and the stop bit is the same as the data line idle state.

Both start bit and stop bit are used for data synchronization during asynchronous data transfer.

The module supports two data bit orders: LSB first or MSB first.

When sending, the USART module reads the data in the sending buffer, and then outputs it from the TXD pin in the above data frame format.

When receiving, the USART module searches for the start bit on the RXD pin, and receives the data bit, parity bit, and stop bit according to the above data frame format after finding the start bit. If no stop bit is found, the USART module considers the transmission of the entire data frame to have failed.

This module supports two ways to find the start bit: the falling edge on the RXD pin or the low level on the RXD pin (when USART_CR2.RXINV=0).

17.3.7.3 Transmitter

When setting USART_CR1.TE to 1, the TXD pin can send data externally. When there is data in the USART_TDR register, the hardware automatically loads the data into the transmit shift register, and shifts and outputs it from the TXD pin according to the data frame format configured by the user. The USART_SR.TXE flag is set to 1 by hardware whenever data in the USART_TDR register is loaded into the shift register. the user finds that the USART_SR.TXE flag is 1, the user can write the next data to be sent to the USART_TDR register. Whenever the TXD pin sends a frame of data, the USART_SR.TC flag will be set to 1 by hardware to indicate the current sending progress.

The figure below is an example of sending two bytes; the data length is 8 bits, no parity, and one STOP bit.

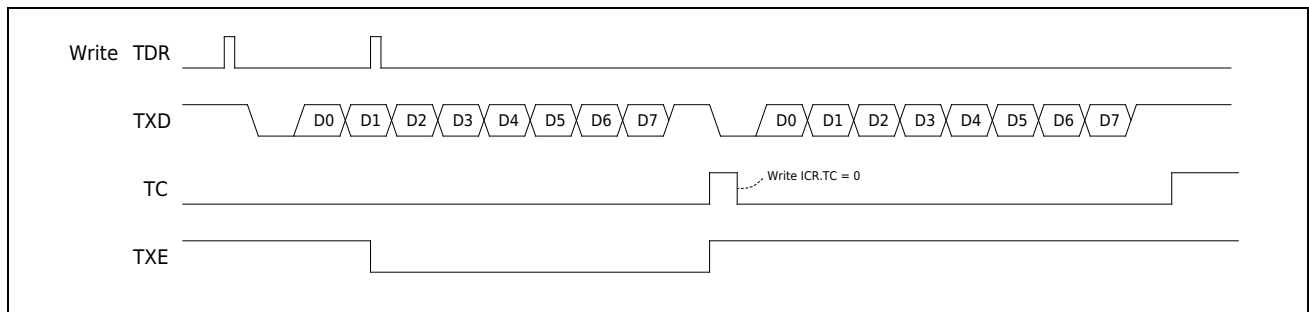


Figure 17-6 Example of sending data

Note:

1. Before writing data to be sent into `USART_TDR`, the `USART_CR1.TE` bit must be set to 1 first.
2. Do not write `USART_CR1.TE=0` during data transmission, otherwise the baud rate counter will be frozen, causing the TXD pin to output unexpected waveforms.

17.3.7.4 Receiver

When setting `USART_CR1.RE` to 1, this module can receive the data received from the RXD pin. Whenever a frame of data is received, the `USART_SR.RC` flag will be set to 1 by hardware, and the received data can be read from the `USART_RDR` register.

This module supports two start bit detection methods: detecting the level on the RXD pin as the start bit, and detecting the edge on the RXD pin as the start bit, see `USART_CR1.SBS` description for details.

This module supports two sampling methods: 8 times sampling and 16 times sampling. 8 times sampling means that the signal on the RXD pin is sampled 8 times per baud rate cycle; 16 times sampling means that the signal on the RXD pin is sampled 16 times per baud rate cycle.

When the module receives data, it can detect three kinds of errors: overflow error, parity error, frame error. After any kind of error occurs, the module continues to receive data, and the user program should handle the error as soon as possible, and clear the error flag to restart data reception.

- Overflow error: When `USART_SR.RC` is 1 (the data in `USART_RDR` is not read) and a new frame of data is received, an overflow error occurs. When this error occurs, `USART_SR.OR` will be set to 1 by hardware, and the newly received data will overwrite the previous frame data. Write 0 to `USART_ICR.OR` to clear this flag. If the data in `USART_RDR` is not read after `USART_SR.RC` is 1, but `USART_SR.RC` is cleared to 0 by writing `USART_ICR.RC = 0`, then `USART_SR.OR` will not be set to 1.
- Parity Error: A parity error occurs when an incorrect parity of a data frame is detected. When this error occurs, `USART_SR.PE` will be set to 1 by hardware, and the received data will be sent to `USART_RDR`. Write 0 to `USART_ICR.PE` to clear this flag.

- Framing Error: A framing error occurs when a received data frame does not detect a stop bit at the expected time. When this frame error occurs, USART_SR.FE will be set to 1 by hardware, and the received data will be sent to USART_RDR. Write 0 to USART_ICR.FE to clear this flag.

The following figure is an example of receiving two bytes: the data length is 8 bits, no parity, and one STOP bit.

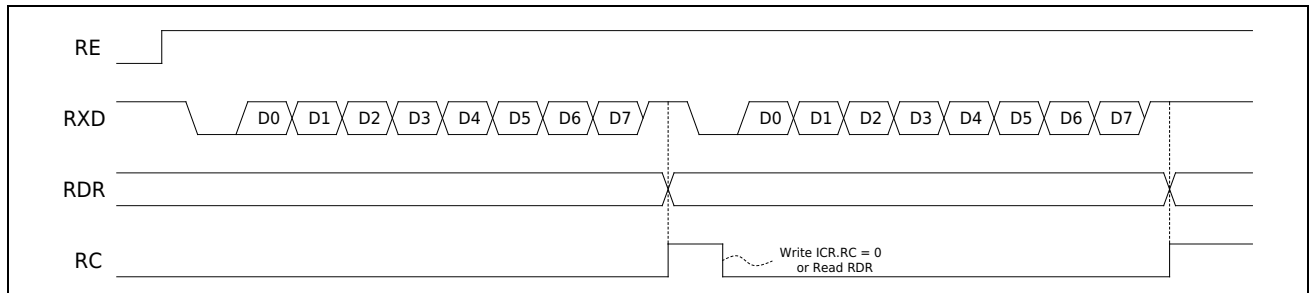


Figure 17-7 Receive data example

17.3.7.5 Single wire half duplex

When setting USART_CR1.COMB and USART_CR1.RE to 1 and USART_CR1.TE to 0, this module works in single-wire half-duplex mode, and only uses the TXD pin for data transmission and reception.

When no data is sent, the TXD pin is in the input state, waiting for other hosts on the receiving bus to send data. Whenever the user writes data to the USART_TDR register, the TXD pin automatically changes to the output state to send the data written by the user to the USART_TDR internal register to the bus.

To prevent collisions on a single bus, the user should take appropriate application-layer protection measures to ensure that multiple masters do not send data to the bus at the same time.

Note: DMA mode is not supported when single-wire half-duplex works.

17.3.7.6 Multiprocessor communication

When setting USART_CR1.MPE to 1, USART works in multiprocessor communication mode. The master can send address frames and data frames, and the slave can receive data frames only after receiving matching address frames.

Data format

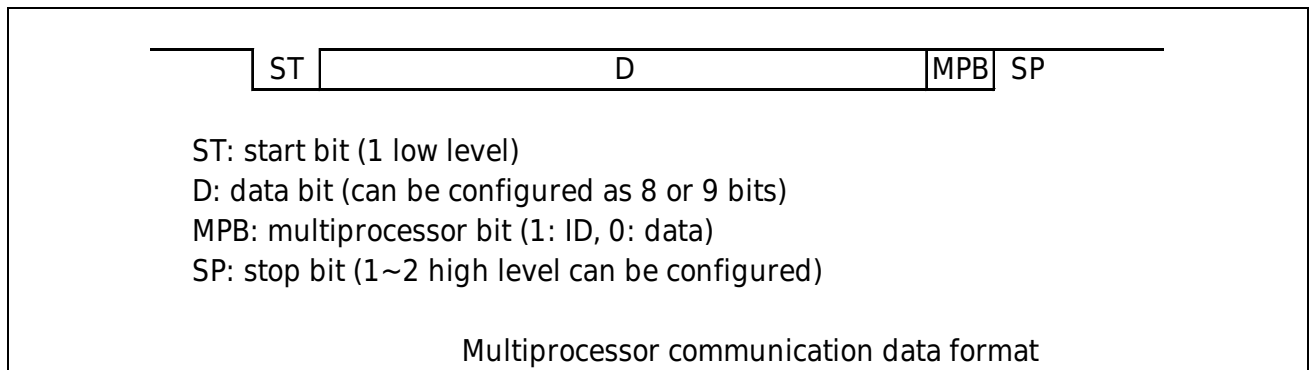


Figure 17-8 Multiprocessor communication data format

Host sends address frame and data frame

- If the word length is 8, ie USART_CR1.CHLEN=0
When USART_TDR.MPID is 1, the address frame is sent, and USART_TDR[7:0] is the address of the target slave.
When USART_TDR.MPID is 0, the data frame is sent, and USART_TDR[7:0] is the data to be sent.
- If the word length is 9, ie USART_CR1.CHLEN=1
When USART_TDR.MPID is 1, the address frame is sent, and USART_TDR[7:0] is the address of the target slave.
When USART_TDR.MPID is 0, the data frame is sent, and USART_TDR[8:0] is the data to be sent.

Slave receives address frame and data frame

USART_CR1.SLME is set to 1, the slave only receives address frames that match its own address, and ignores unmatched address frames and data frames. When the slave receives the address frame from RXD, the hardware automatically compares the address in the address frame with the local address; only when the two match, the received address will be stored in the USART_RDR register and USART_SR.RC and USART_SR.AM will be set.

Slave address configuration

The slave address consists of the AND of the user-configured address(USART_ADDR) and the address mask(USART_ADDMR).

Slave address: USART_ADDR & USART_ADDMR

When the address frame is received, it is detected that the address received from the bus matches the value of USART_ADDMR and the value is the same as the address configured by the user, then it is considered that the address frame matching the slave address has been received.

Example: USART_ADDR = 0x55, USART_ADDMR = 0xF0, then the address that the slave can match is 0x50 ~ 0x5F(user configuration address).

Note: Multiprocessor communication is only applicable to asynchronous serial port mode.

17.3.7.7 Send data programming example

- Step1. According to the relevant description of the pin digital multiplexing function in the GPIO chapter, configure TXD as a digital input with pull-up when USART_CR2.TXINV=0, configure TXD as a digital input with pull-down when USART_CR2.TXINV=1, and set GPIOx_AFRH/AFRL to map TXD to required pins;
- Step2. Set APBEN0.USARTn(n=0/1/2/3) to 1 to enable USART configuration clock and working clock;
- Step3. Write 0x00 to USART_CR1 to make the USART module return to the initial state;
- Step4. Configure USART_CR2.SYNC as 0 to make the USART module work in asynchronous serial port mode;
- Step5. Configure the USART_CR1.OVER8 and USART_BRR registers to configure the communication baud rate;
- Step6. Configure USART_CR1.CHLEN to configure the number of data bits in the frame;
- Step7. Configure USART_CR1.STOP, USART_CR1.PCE and USART_CR1.PS, configure stop bit length and parity check mode;
- Step8. Configure USART_CR1.MSB to configure the data sequence order when sending data;
- Step9. Set USART_CR1.TE to 1 to enable TXD sending function;
- Step10. Write 0x00 to USART_ICR.TC to clear the sending completion flag;
- Step11. Write a frame of data to be sent to the USART_TDR register;
- Step12. The query waits for USART_SR.TC to become 1, and the data frame written this time is sent;
- Step13. If the data to be sent is not completed, jump to Step10 to continue execution.

17.3.7.8 Receive data programming example

- Step1. RXD to the required pins according to the relevant description of the pin digital multiplexing function in the GPIO chapter; configure RXD as a digital input with pull-up when USART_CR2.RXINV=0, and configure RXD as a digital input with pull-down when USART_CR2.RXINV=1, to prevent false triggering caused by the floating RXD line when the sending end is not enabled and the receiving end is enabled first;
- Step2. Set APBEN0.USARTn(n=0/1/2/3) to 1 to enable USART configuration clock and working clock;
- Step3. Write 0x00 to USART_CR1 to make the USART module return to the initial state;

- Step4. Configure USART_CR2.SYNC as 0 to make the USART module work in asynchronous serial port mode;
- Step5. Configure the USART_CR1.OVER8 and USART_BRR registers to configure the communication baud rate;
- Step6. Configure USART_CR1.CHLEN to configure the number of data bits in the frame;
- Step7. Configure USART_CR1.STOP, USART_CR1.PCE and USART_CR1.PS, configure stop bit length and parity check mode;
- Step8. Configure USART_CR1.MSB to configure the data sequence order when receiving data;
- Step9. Configure USART_CR1.SBS to select the judgment method of the start bit;
- Step10. Write 0x00 to the USART_ICR register to clear all status flags;
- Step11. Set USART_CR1.RE to 1;
- Step12. The query waits for USART_SR.RC to become 1, and one frame of data is received;
- Step13. Query USART_SR.PE, USART_SR.FE, USART_SR.OR to confirm whether the received data frame is valid. If it is invalid, perform error handling, if it is valid, read the data from the USART_RDR register and save it;
- Step14. Write 0x00 to USART_ICR.RC to clear the receiving completion flag;
- Step15. If the data to be received is not completed, jump to Step12 to continue execution.

17.3.8 Synchronous serial port mode

The synchronous serial port mode means that the sending and receiving of data is synchronized with the SCK clock pin.

Select synchronous serial port mode by setting USART_CR2.SYNC=1.

In synchronous serial port mode, the following bits must be cleared:

- USART_CR1.COMB, USART_CR1.SLME, USART_CR1.MPE, USART_CR1.OVER8, USART_CR1.STOP, USART_CR1.CHLEN, USART_CR1.PCE
- USART_CR3.SCEN, USART_CR3.LINEN and USART_CR3.IREN

17.3.8.1 Clock

There are two options for the clock source of the synchronous serial port mode, which can be set by USART_CR2.CLKC:

- Internal clock source: clock generated by internal baud rate generator

The synchronous clock is output from the CK pin, and a frame of data outputs 8 clock pulses. When neither sending data nor receiving data, the clock output is fixed at high level.

- External clock source: CK pin input clock

The external clock source is the clock input from the CK pin as the communication clock.

17.3.8.2 Data format

In the synchronous serial port mode, a frame of data is fixedly composed of 8 bits, and the sending and receiving of a frame of data requires 8 synchronous clock pulses.

Data is sent at the falling edge of the synchronization clock when data is sent and sampled at the rising edge of the synchronization clock when data is received.

Except for the continuous transmission mode, the synchronous clock is fixed at high level when there is no data transmission; when there is data transmission, after the last bit of the sent data is sent, the data line keeps the value of the last bit.

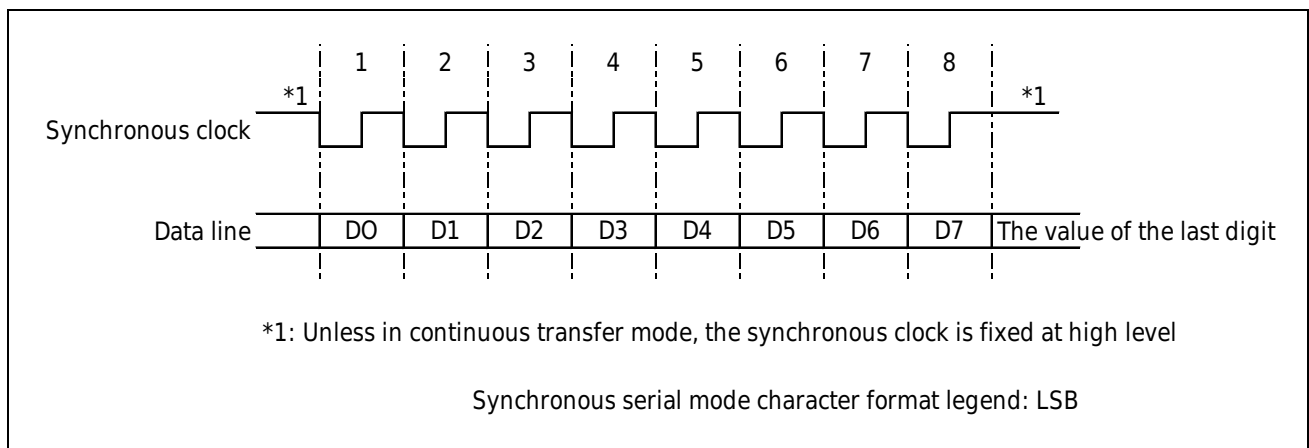


Figure 17-9 Synchronous serial port mode character format legend

17.3.8.3 Transmitter

When the transmit enable bit USART_CR1.TE=1, the data in the transmit shift register is output on the TXD pin, and the corresponding clock is output on the CK pin.

The sending data register USART_TDR and the internal sending shift register form a double buffer structure, which can send data continuously.

The writing of the sending data register USART_TDR can be completed through an interrupt program (CPU or DMA), and one interrupt can only write data once.

Send data operation steps

- Step1. Set USART_CR1, USART_SR registers to reset values;
- Step2. Initialize the port, configure TXD as a digital input with pull-up; set GPIOx_AFRH/AFRL to map TXD to the required pin;
- Step3. Select the clock source through the USART_CR2.CLKC bit;
- Step4. Set USART_CR1 (except TE), USART_CR2, USART_CR3 registers;

- Step5. Set USART_BRR.PSC to select the prescaler value, and set the USART_BRR communication baud rate;
- Step6. Set USART_CR1.TE=1, USART_IER.TXE=1;
- Step7. USART_TDR through the interrupt program, the data is transferred to the sending shift register, and the sending starts (when the CTS function is valid, when the CTS pin inputs low level, the data is transferred to the sending shift register, and the sending starts);
- Step8. If you need to send data continuously, repeat step 7;
- Step9. Write the last sent data through the interrupt program, and write USART_IER.TXE=0, USART_IER.TC=1, after the last data is sent, a send completion interrupt occurs.

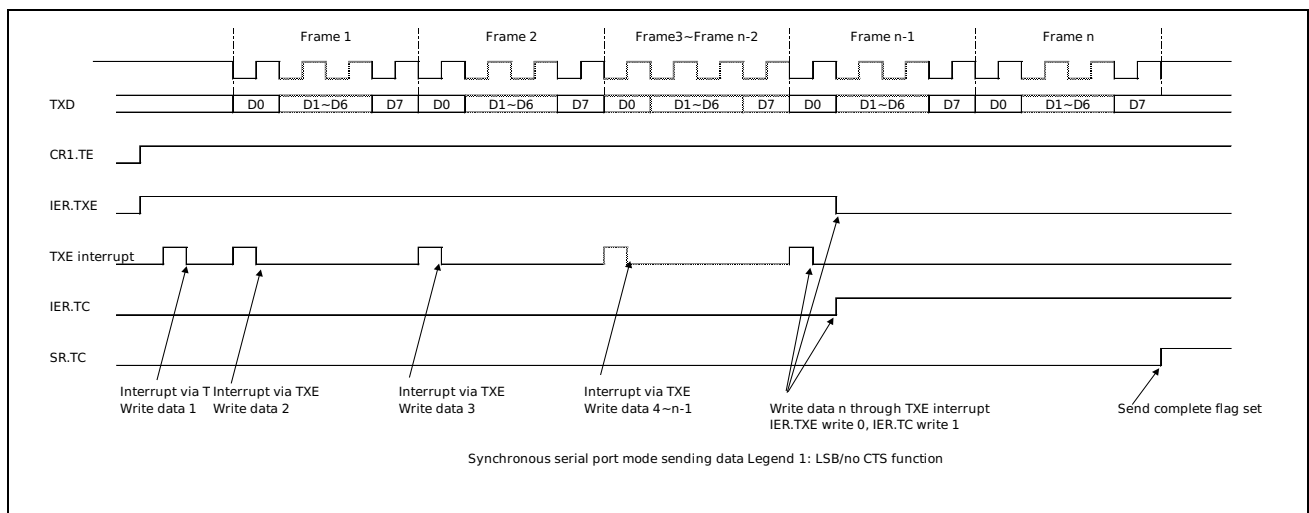


Figure 17-10 Synchronous serial port mode sending data legend 1

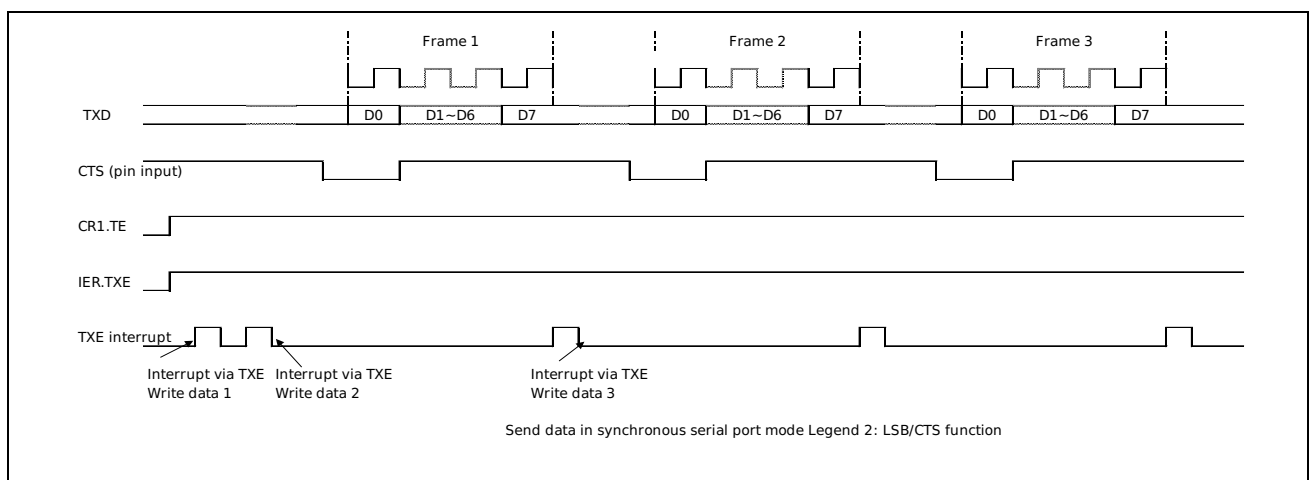


Figure 17-11 Synchronous serial port mode sending data legend 2

Transmitter interrupt

The synchronous serial port mode transmitter supports two interrupts, the transmit data register empty interrupt and the transmit complete interrupt.

USART_IER.TXE=1, an interrupt occurs when the value of the USART_TDR register is transferred to the transmit shift register.

USART_IER.TC=1, when the last bit of data is sent, the USART_TDR register is not updated and an interrupt occurs.

17.3.8.4 Receiver

Steps to receive data

- Step1. Set USART_CR1, USART_SR registers to reset values;
- Step2. Initialize the port, configure RXD as a digital input with pull-up; set GPIOx_AFRH/AFRL to map RXD to the required pin;
- Step3. Select the clock source through the USART_CR2.CLKC bit;
- Step4. Set USART_CR1 (except RE), USART_CR2, USART_CR3 registers;
- Step5. Set USART_BRR.PSC to select the prescaler value, and set the USART_BRR communication baud rate;
- Step6. Set USART_IER.RC=1, USART_IER.OR=1, USART_CR1.RE=1 (when using the RTS function, the RTS pin outputs a low level after USART_CR1.RE=1);
- Step7. When the synchronous clock is input or the internally generated synchronous clock is working, the data is received, and the data is received into the shift register.
 - 1) When an overflow error occurs, the data is overwritten and an overflow error interrupt occurs, and error handling is performed
 - 2) When no overflow error occurs, the received data is transferred to the USART_RDR register, and a reception completion interrupt occurs. Through the reception completion interrupt program, the received data is read before the last bit of the next frame of data is received, and then step 7 is repeated. Realize the function of continuously receiving data

(When using the RTS function, the RTS pin outputs a low level after the data is read)

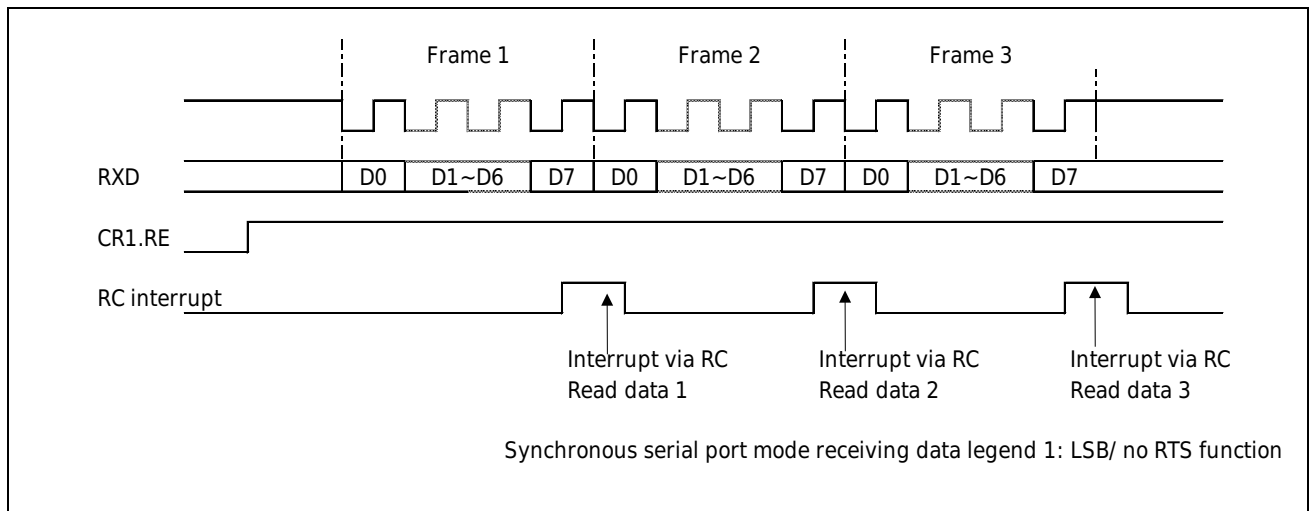


Figure 17-12 Synchronous serial port mode receiving data legend 1

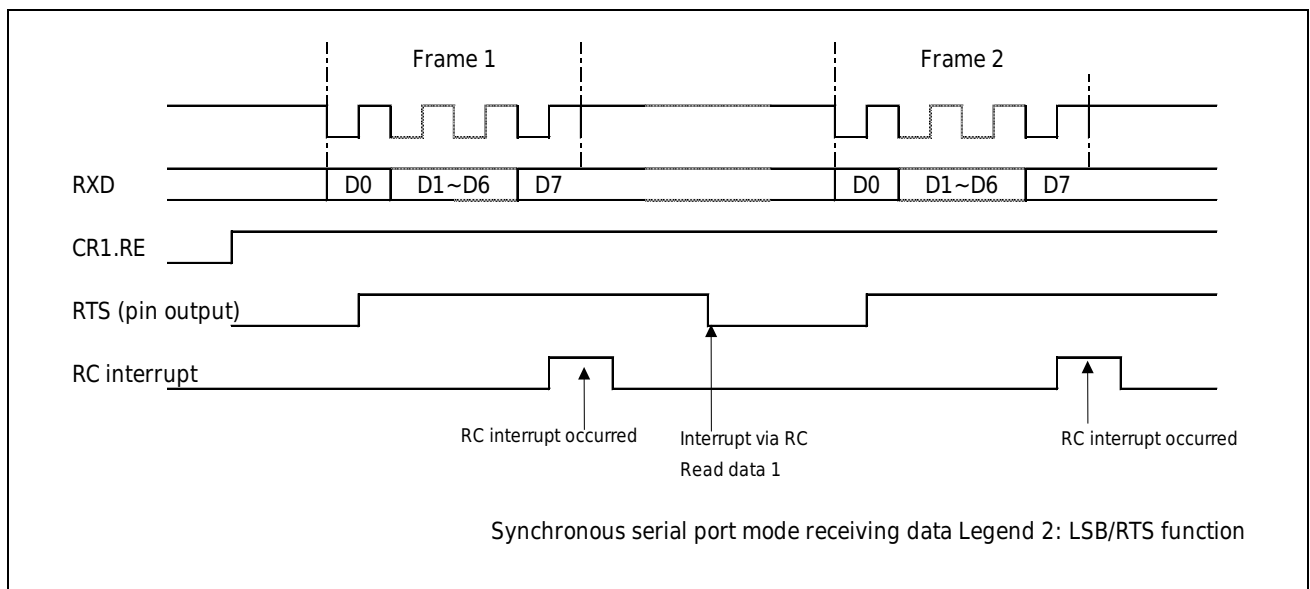


Figure 17-13 Synchronous serial port mode receiving data legend 2

Error handling

When receiving data in synchronous serial port mode, the receiving error is an overflow error (USART_SR.OR). The overflow error occurs when new data is received when the value of USART_RDR has not been read. Data not read after an overflow error occurs is overwritten. The data transfer can be restarted by clearing the error flag to zero by writing the corresponding clear register bit.

In order to avoid overflow errors, the data received in the previous frame should be read before the last bit of the current frame is received.

Receiver interrupt

In the synchronous serial port mode, the receiver supports two kinds of interrupts, receiving complete interrupt and overflow error interrupt.

USART_IER.RC=1, an interrupt occurs when data is transferred from the shift register to USART_RDR.

USART_IER.OR=1, the receive data overflow error interrupt occurs.

The above interrupts are all generated after the eighth CK pin rising edge.

17.3.8.5 Send and receive data at the same time

The USART synchronous serial port mode supports full-duplex operation, sending and receiving data at the same time.

When sending and receiving data at the same time, a command is required to write 1 to USART_CR1.RE and USART_CR1.TE at the same time, and set the interrupt enable bit. Other setting procedures are the same as those of the transmitter and receiver.

17.3.8.6 Precautions for synchronous serial port mode

- 1) When sending data using an external input clock, wait for more than 6 PCLKs before inputting the clock after USART_TDR is updated. total period of the input clock must be greater than or equal to 6 PCLKs, and the periods of high and low levels must be greater than or equal to 2 PCLKs.
- 2) When data is sent consecutively, the next frame of data needs to be updated before the last bit of the current frame is sent.

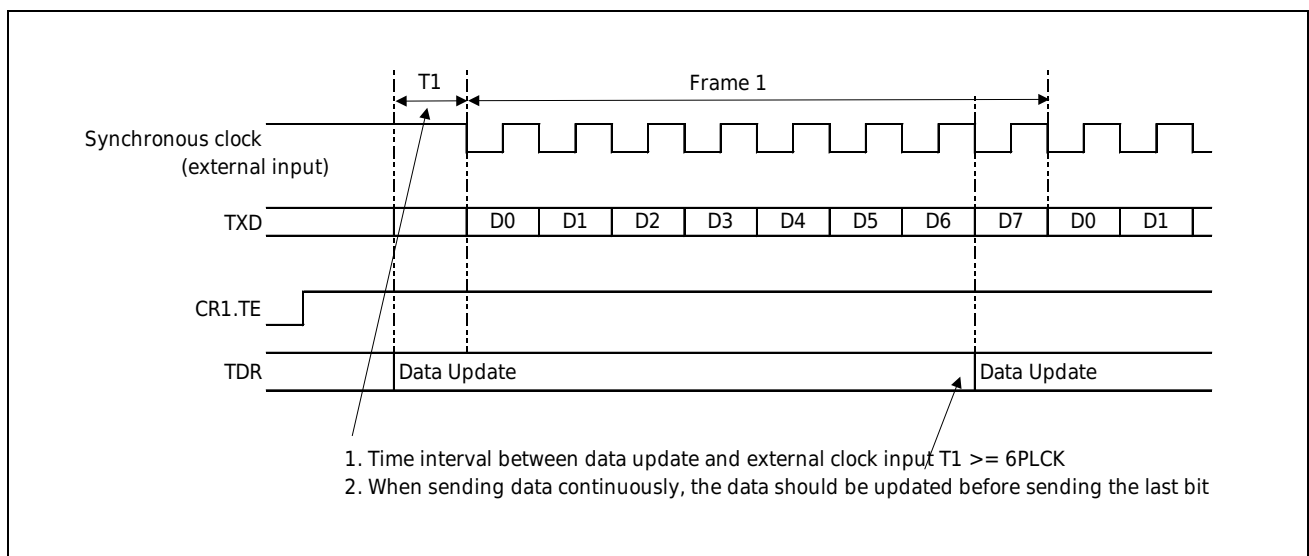


Figure 17-14 Precautions for synchronous serial port mode

17.3.9 Smart Card (7816) Mode

ISO7816 is a contact card reading protocol. This module is only used as a smart card host interface, and the smart card mode can be selected by writing USART_CR3.SCEN=1. In smart card mode, the following bits must be cleared:

- USART_CR1.COMB, USART_CR1.SLME, USART_CR1.MPE, USART_CR1.SBS, USART_CR1.OVER8
- USART_CR2.SYNC
- USART_CR3.LINEN and USART_CR3.IREN

In addition, the USART_CR2.CLKC bit needs to be set to provide the clock for the smart card.

17.3.9.1 Connection Diagram

The smart card interface supports the smart card communication protocol stipulated in ISO/IEC 7816-3, and supports T=0 (character mode) and T=1 (block mode) at the same time. The following figure is the connection diagram of the smart card mode:

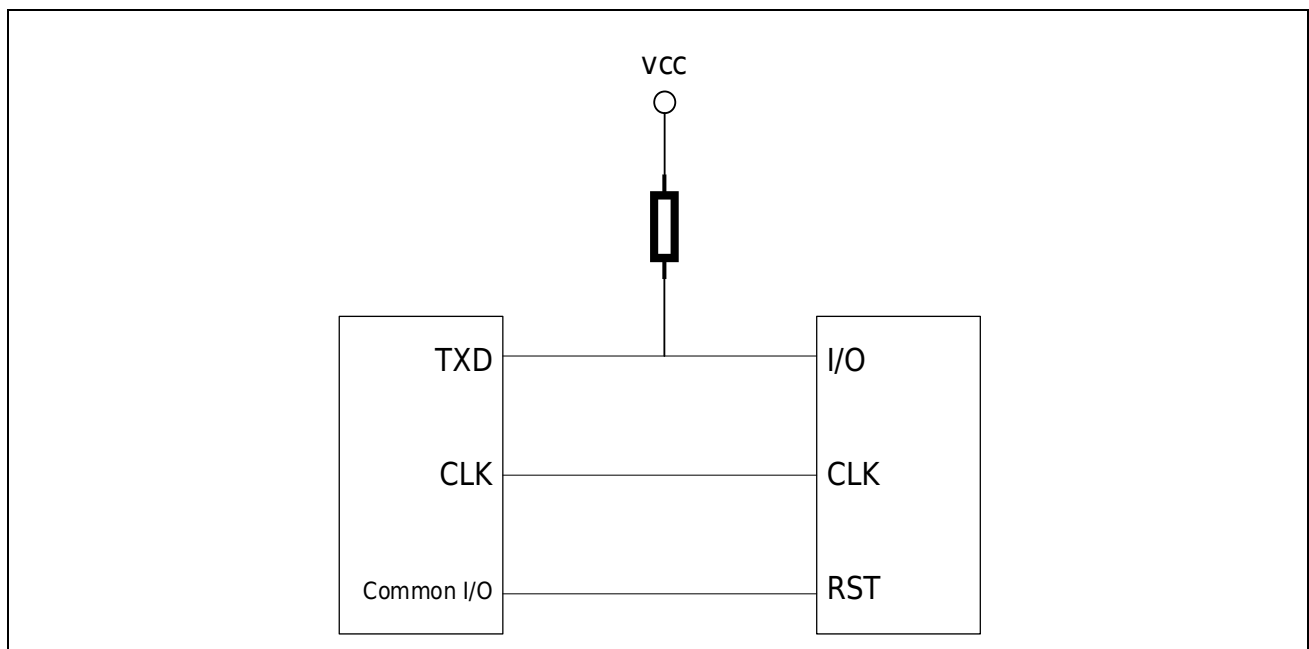


Figure 17-15 Smart card Pattern connected legend

17.3.9.2 Clock

Internal timer

Only the clock generated by the internal baud rate generator can be used as the clock source in smart card mode.

The basic clock number for one-bit data transmission is the setting value of USART_CR3.BCN.

Register USART_CR2.CLKC bits and USART_BRR are used to set the clock output in smart card mode.

Sampling and reception tolerance

After detecting the falling edge of the receiving pin (TXD), the USART will clock the received data based on the internal basic clock to start data reception.

Each bit of received data is sampled three times for a majority vote and then sent to the shift register.

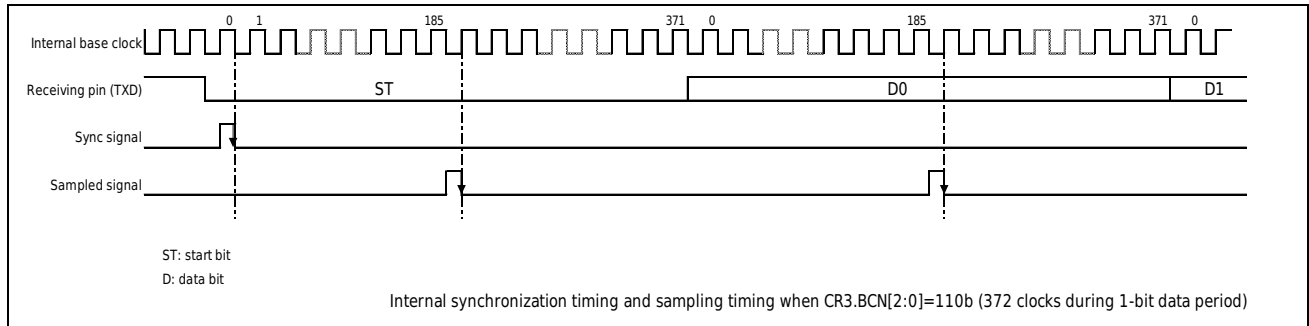


Figure 17-16 Smart Card Pattern synchronization and sampling timing Figure

The following formula is used to calculate the receiving tolerance:

$$M[\%] = \left| \left(0.5 - \frac{1}{2N} \right) - (L-0.5)F - \frac{|D-0.5|}{N} (1+F) \right| \times 100\%$$

M: Receive tolerance

N: The number of clocks required for one-bit data transmission (BCN[2:0] setting value)

L: the length of the frame (10)

D: Clock duty cycle

F: clock skew

17.3.9.3 Data format

In smart card mode, a frame of data consists of start bit, data bit and parity bit.

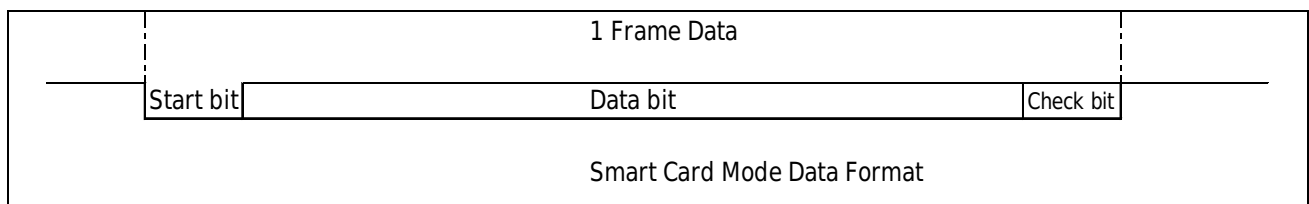


Figure 17-17 Smart Card Mode Data Format

Start bit

The start bit is fixed at a low level.

Data bit

The data bits are fixed to 8-bit data.

Check bit

The parity bit needs to be configured as 1-bit even parity.

The figure below shows the legend of data 0x3B data format.

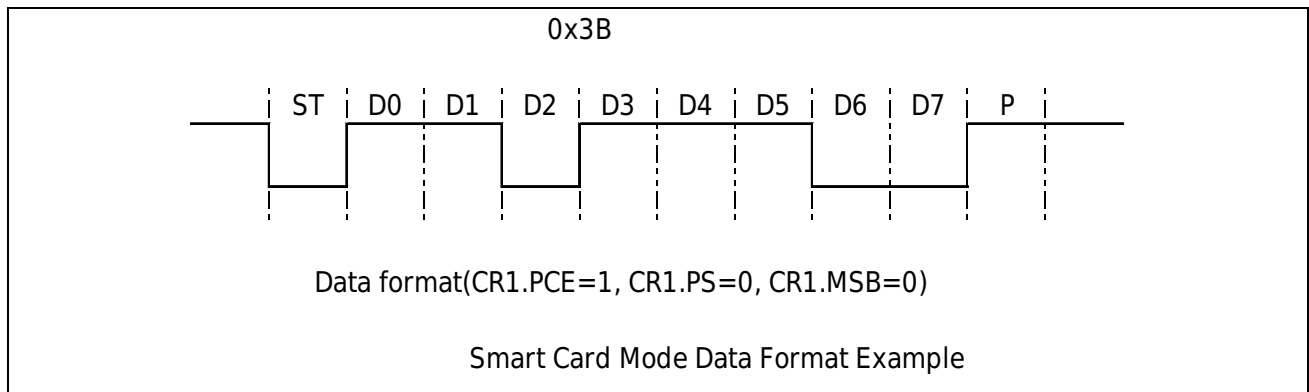


Figure 17-18 Example of smart card mode data format

17.3.9.4 Initial setting procedure of smart card

- Step1. Set the USART_CR1 register to the reset value;
- Step2. Initialize ports, all set as digital input with pull-up; set the pins to be used;
- Step3. The status register is confirmed, and the USART_SR register is set to the reset value;
- Step4. Set USART_CR1 (except TE, RE), USART_CR2, USART_CR3 registers;
- Step5. Set USART_BRR.PSC to select the prescaler value, and set the USART_BRR communication baud rate;
- Step6. Set the clock control of the USART_CR2.CLKC bit;
- Step7. Set the USART_CR1 register (TE, RE bits), set TE and RE as required, and do not set TE and RE to 1 at the same time.

When the sending mode is switched to receiving mode, or the receiving mode is switched to sending mode, step 7 needs to be reset.

When using the smart card function, the word length USART_CR1.CHLEN, the check control bit USART_CR1.PCE stop bit length USART_CR1.STOP can be any value, the parity check bit USART_CR1.PS must be set to odd or even as required, and the reverse control bit USART_CR2.TXINV=0. Both data reception and transmission are transmitted through TXD, regardless of the USART_CR1.COMB value.

17.3.9.5 Smart card mode action description

In smart card mode, the interrupt(send data empty interrupt) flag is USART_SR.TXE. USART_SR.TXE=1 and USART_IER.TXE=1.

Functional Overview

When sending data, there is a protection time of more than 2 etu (Elementary Time Unit) between two frames of data (from the end of the parity bit to the start bit of the next frame).

When sending data, if an error signal sent by the receiver is detected, the data will be resent automatically after 2 etu.

When a verification error occurs in the received data, a low level of 1etu is sent, which is an error signal. The timing of sending the error signal is 10.5etu after the start of reception.

1etu is the number of clocks on the SCK pin determined by the value of USART_CR3.BCN.

Send instructions

- 1) After a frame of data is sent, if an error signal sent by the receiver is detected, set USART_SR.FE=1 to generate a frame error interrupt (when USART_IER.FE=1), the USART_SR.TC flag is not set to 1, and the data is automatically resent. The USART_SR.FE bit must be cleared before receiving the next frame parity bit.
- 2) A frame of data is sent without error, set USART_SR.TC=1, and an interrupt occurs (USART_IER.TC=1). Data can be sent continuously by writing data through the interrupt program.

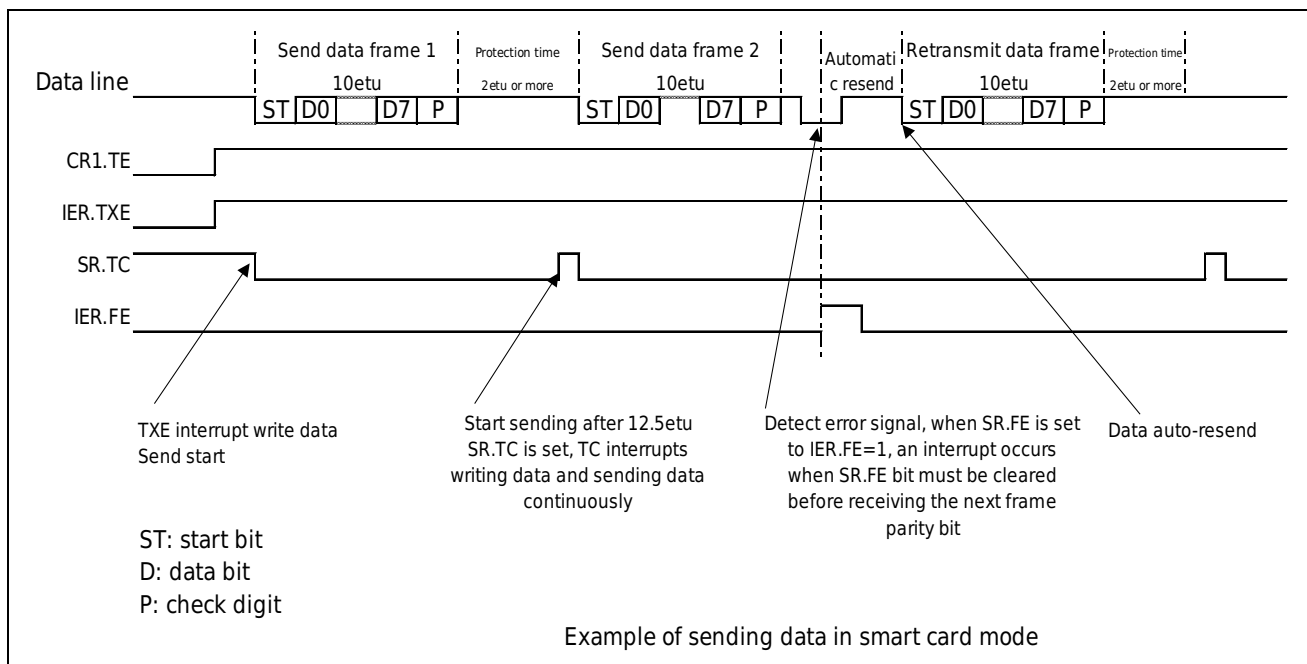


Figure 17-19 Smart card Pattern sending data legend

Receiving Instructions

- 1) When receiving data, if a parity error is detected, then USART_SR.PE=1, if USART_IER.PE=1, a parity error interrupt occurs. USART_SR.PE needs to be cleared before receiving the next frame parity bit.
- 2) When a verification error occurs, a low level of 1etu will be sent, that is, an error signal, requiring the sender to resend the data.

- 3) To receive data normally, you can read the received data through the interrupt program and receive continuously.
- 4) When receiving data, an overflow error is detected.

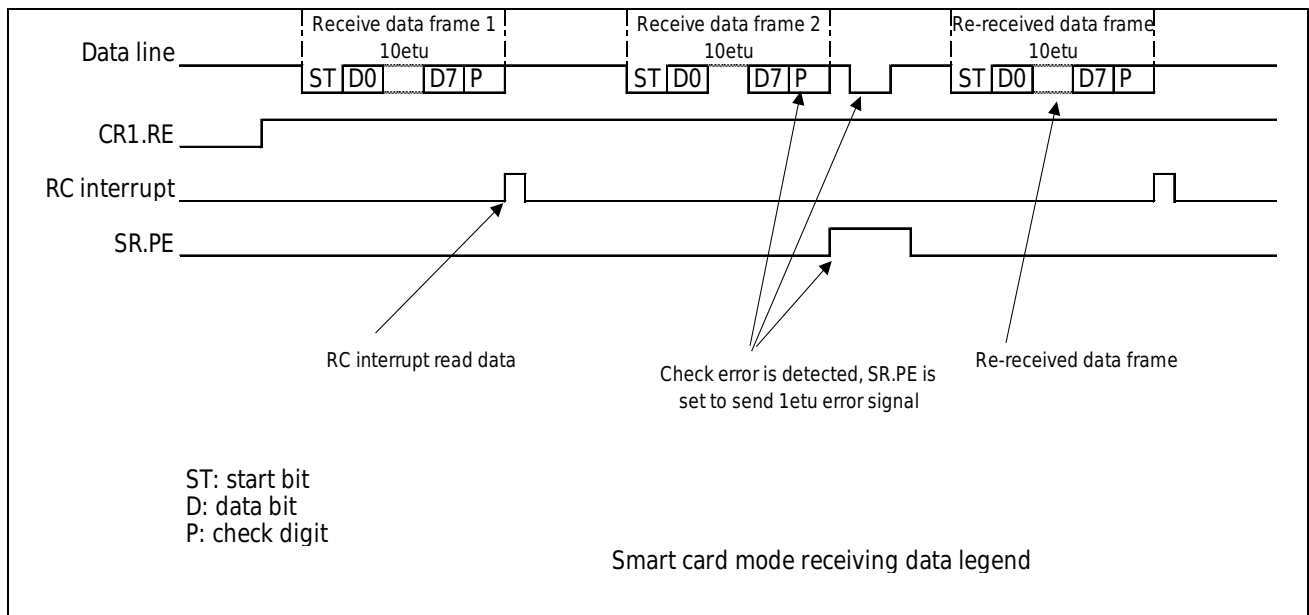


Figure 17-20 Smart card Pattern Receive data legend

17.3.10 infrared mode

Select infrared mode by writing `USART_CR3.IREN=1`. In infrared mode, the following bits must be cleared:

- `USART_CR1.COMB`, `USART_CR1.SLME`, `USART_CR1.MPE`
- `USART_CR3.SCEN` and `USART_CR3.LINEN`

This module contains 2 infrared modes: $\frac{3}{16}$ infrared pulse and 38K modulation output with baud rate period. 2 modes of the same module cannot work at the same time.

Reasonably configure the word length `USART_CR1.CHLEN`, check control bits `USART_CR1.PCE`, `USART_CR1.PS`, stop bit length `USART_CR1.STOP`, inversion control bits `USART_CR2.TXINV`, `USART_CR2.RXINV` and other register bits to meet the needs of infrared applications.

17.3.10.1 IR mode 1: IR pulse width for baud rate period $\frac{3}{16}$

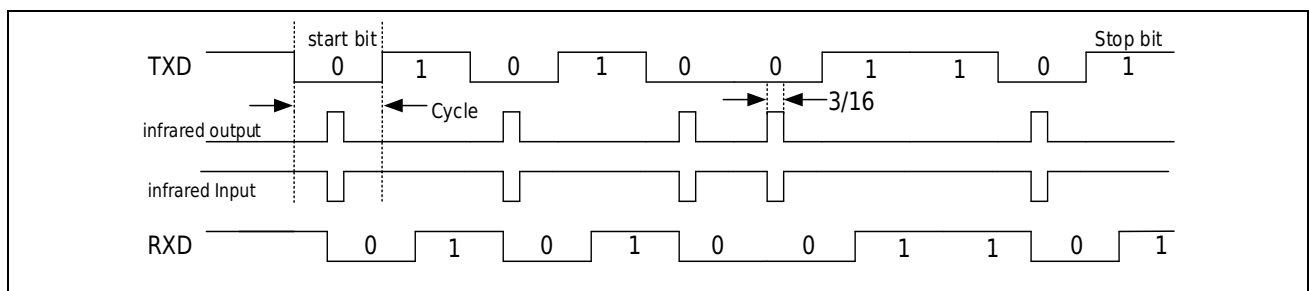


Figure 17-21 Infrared Data Modulation

Select $\frac{3}{16}$ baud rate period mode by writing USART_CR3.IREN=1 and USART_CR3.IRCW=0. In infrared mode, the USART_CR3.LINEN and USART_CR3.SCEN bits must be cleared.

The infrared protocol physical layer specifies the use of a return-to-zero-inverted (RZI) modulation scheme, which represents a logic 0 with pulses of infrared light (pictured above).

The infrared output pulse stream of the USART is sent to the infrared LED outside the chip. SIR ENDEC bit rate supported by USART is up to 115.2Kbps. In normal mode, the transmitted pulse width is specified as one bit period $\frac{3}{16}$.

The infrared receiver decoder outside the chip outputs the received non-return-to-zero (NRZ) serial bit stream to the USART. In the idle state, the decoder input is normally high (flag state). Send the encoder output with the opposite polarity to the decoder input. A start bit is detected when the decoder input is low.

- 1) Infrared communication is a half-duplex communication protocol. To prevent data corruption, simultaneous sending and receiving should be avoided.
- 2) A "0" is sent as a high pulse, and a "1" is sent as a "0". In normal mode, the pulse width is specified for the selected bit period $\frac{3}{16}$ (see figure above).
- 3) The specification requires that the pulse width value should be greater than 1.41us.
- 4) In infrared mode, some bits in the USART_CR1 and USART_CR2 registers are masked as invalid bits, and writing 1 or 0 by the user will not affect the normal operation of the infrared, see the notes in the register section for details.
- 5) In the infrared mode, the input and output signals can be inverted by USART_CR2.RXINV and USART_CR2.TXINV and then input to the USART module or output from the USART module.

Sending data setting procedure

- Step1. Set the USART_CR1 register to the reset value;
- Step2. The initialization ports are all set as digital input with pull-up; set the pins that need to be used by infrared;
- Step3. Set values other than TE in USART_CR1, set USART_CR2, USART_CR3 registers;
- Step4. Set USART_BRR.PSC to select the prescaler value, and set the USART_BRR communication baud rate;
- Step5. Set USART_CR3.IREN=1(USART_CR2.TXINV is configured as needed), USART_CR1.TE=1, set USART_IER.TXE=1;

- Step6. Write the communication data to USART_TDR by sending the interrupt program, and the sending starts;
- Step7. If you need to send data continuously, repeat step 6;
- Step8. Write the last sent data through the sending interrupt program, and write USART_IER.TXE=0, USART_IER.TC=1, after the last data is sent, a sending completion interrupt occurs;
- Step9. Write USART_CR1.TE=0.

Receiving data setting procedure

- Step1. Set the USART_CR1 register to the reset value;
- Step2. Initialize ports, all set to input with pull-up; set the pins that need to be used by infrared;
- Step3. set USART_IER.RC=1;
- Step4. Set values other than RE in USART_CR1, set USART_CR2, USART_CR3 registers;
- Step5. Set USART_BRR.PSC to select the prescaler value, and set the USART_BRR communication baud rate;
- Step6. USART_CR3.IREN=1(USART_CR2.RXINV is configured as needed), USART_CR1.RE=1;
- Step7. When the start bit is detected, the data is received into the shift register after internal synchronization, and the check bit and stop bit are checked;
 - 1) When the stop bit is low, a frame error occurs, the received data is transferred to the USART_RDR register, and a receive frame error interrupt occurs, and the error interrupt handler is executed
 - 2) When a data overflow error occurs, the previous frame data is overwritten and a data overflow error interrupt occurs, and the error interrupt handler is executed
 - 3) When an error occurs, a parity error occurs, the received data is transferred to USART_RDR, and a parity error interrupt occurs, and the error interrupt handler is executed
 - 4) When no error occurs, the received data is transferred to the USART_RDR register, and a reception completion interrupt occurs. After reading the received data through the interrupt program, repeat step 7 to receive data continuously
- Step8. Write USART_CR1.RE=0.

17.3.10.2 Infrared mode 2: 38K modulation with adjustable duty cycle

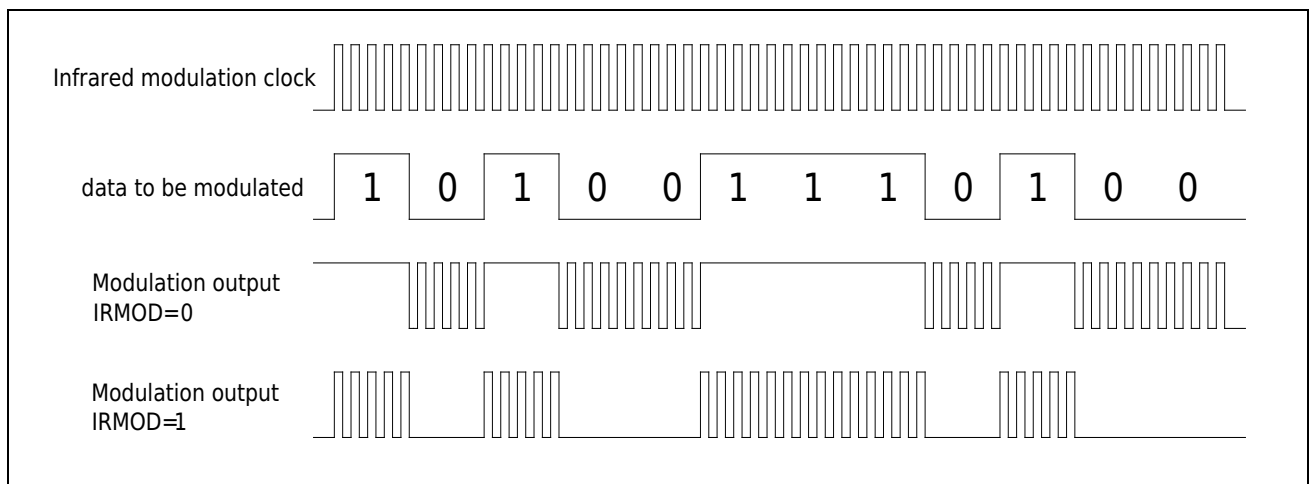


Figure 17-22 38K infrared modulation

The way of infrared modulation is: output infrared modulation when sending data 0, and not modulate when sending data 1. The 38K infrared modulation frequency is determined by the value of the register USART_IR and the current operating frequency of the module, and its calculation formula is:

$$F = PCLK / (USART_IR.DCNTH + USART_IR.DCNTL + 2)$$

Where: PCLK is the module operating frequency,

USART_IR.DCNTH determines the high level duration of the 38K modulation frequency,

USART_IR.DCNTL determines the low level duration of the 38K modulation frequency.

Considering the needs of PNP and NPN infrared tubes in application, USART_CR2.RXINV and USART_CR2.TXINV can be set to invert the received and sent data. USART_IR.DCNTH and USART_IR.DCNTL are the duty cycle settings when not inverting. If the sending data is inverting, the two data need to be exchanged.

17.3.11 LIN mode

Select LIN mode by writing USART_CR3.LINEN=1. In LIN mode, the following bits must be cleared:

- USART_CR1.COMB and USART_CR1.STOP, USART_CR1.SLME, USART_CR1.MPE, USART_CR1.PCE
- USART_CR2.SYNC
- USART_CR3.SCEN and USART_CR3.IREN

17.3.11.1 Feature Brief

LIN is the abbreviation of Local Interconnect Network (Local Interconnect Network), which is a low-speed(1~20Kbps) serial communication protocol for reducing the cost of automotive networks.

LIN communication is a single-master communication, and a master device can connect up to 15 devices.

"wire-AND" relationship between the devices on the bus, and the dominant level plays a leading role:

- When any node on the bus sends a dominant level (low level), the bus is at a dominant level;
- Only when all nodes send recessive level (high level) or do not send information (when no information is sent, the bus defaults to recessive level), the bus will show recessive level.

The LIN bus supports ISO9141; the LIN slave device is connected to the master control device through LIN, which is mainly used for the control of switches, transmissions and sensors.

The USART in this product integrates a multi-function timer to facilitate the realization of LIN bus functions, including data transmission, baud rate setting, synchronization interval detection, synchronization measurement and baud rate calculation, etc. RXD pin edge interrupt (USART_IER.DWK) and I/O port external interrupt input function can be used for wake-up signal detection.

According to the application, when USART is used as LIN, it is RXD/TXD two-wire mode, and the corresponding chip needs to be configured externally for boosting and bus conversion.

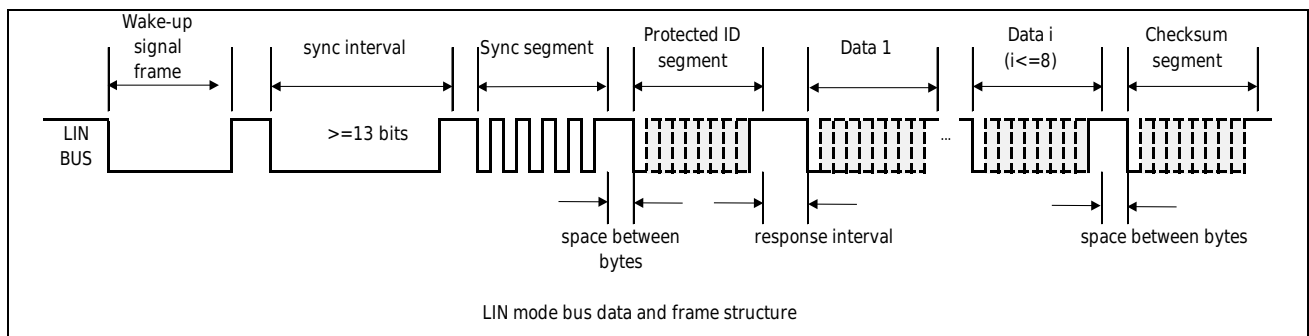


Figure 17-23 LIN bus data and frame structure

Note: Wake-up signal frame is only required when the device needs to wake up

17.3.11.2 LIN data format

One frame of LIN data consists of 1 start bit, 8 data bits and 1 stop bit, and sends and receives data in LSB mode.

17.3.11.3 LIN send

Sending of Wake Signal Frames

Set the appropriate baud rate, and realize the sending of the wake-up signal frame by sending data 0x80.

Synchronous Interval Sent

The synchronization interval segment indicates the beginning of a frame of data, and lasts at least 13 bits (based on the host node speed) of the dominant level (low voltage). This product uses USART_CR3.BREAK and USART_TDR as the synchronization interval time for sending dominant level, and the number of "0" s for sending continuous bits is set in the USART_TDR register.

Synchronization segment sending

Sending Synchronous Segment Data by Sending Data 0x55.

LIN master control device software

- Step1. Set the USART_CR1 register to the reset value;
- Step2. According to the relevant description of the pin digital multiplexing function in the GPIO chapter, configure TXD and RXD as digital input with pull-up, and set GPIOx_AFRH/AFRL to map TXD and RXD to the required pins;
- Step3. Set USART_CR1 (except TE), USART_CR2, USART_CR3 (except LINEN) registers;
- Step4. Set USART_BRR.PSC to select the prescaler value;
- Step5. Set the USART_BRR register to set the baud rate of the signal frame;

(Steps 6-8 below are for sending the wake-up signal frame, skipped if not necessary)

- Step6. set USART_CR1.TE=1;
- Step7. Write communication data 0x80 to USART_TDR, the data is transferred to the shift register, and the transmission starts;
- Step8. Wait for the sending to complete;

(The following steps 9~12 are the sending of the synchronous interval segment)

- Step9. Set USART_CR3.BREAK=1, USART_CR3.LINEN=1;
- Step10. If you skip the above steps 6~8, set USART_CR1.TE=1, otherwise skip this step;

Step11. Write the length of the synchronization interval to USART_TDR, and send the low level of the synchronization interval to the bus;

Step12. Wait for the sending to complete;

(The following steps 13~14 are the transmission of the synchronous segment, and the process is the same as that of the asynchronous serial port transmission)

Step13. Write data 0x55 to USART_TDR;

Step14. Wait for the sending to complete;

(The following steps 15~17 are the interval between bytes)

Step15. Write the length of waiting time between bytes to USART_TOR;

Step16. Write USART_CR1.TOE=1 to start software timing;

Step17. wait for USART_SR.TO=1;

Step18. Protected ID segment: write protected ID segment data(ID) to USART_TDR;

Step19. Wait for the sending to complete;

(The following 20~22 steps are response intervals)

Step20. Write the response waiting time length to USART_TOR;

Step21. Write USART_CR1.TOE=1 to start software timing;

Step22. wait for USART_SR.TO=1;

Step23. Data segment 1: The data sender may be the master or the slave, depending on the application and the parameters of the protected ID segment. If the master sends data, write the data to USART_TDR and wait for the sending to complete(USART_SR.TC= 1);

Step24. Inter-byte interval of data segment 1, refer to steps 15~17 to execute;

Step25. Data segment 2: refer to the data segment 1 of step 23;

Step26. Inter-byte interval of data segment 2, refer to step 24;

Step27.

Step28. Data segment n: refer to the data segment 1 of step 23;

Step29. Interbyte interval of data segment n: refer to step 24;

Step30. Checksum section(checksum is implemented by software): if the data section is the data sent by the master, then refer to step 23;

Step31. Wait for the transmission to complete, the LIN operation ends, and the control register is cleared.

Note: Check bit `USART_CR1.PCE=0`, stop bit `USART_CR1.STOP=0`, `USART_CR1.COMB=0` must be set.

17.3.11.4 LIN Receive

Detection of Wake Signal Frames

The detection of the wake-up signal frame is realized by detecting the falling edge through the external interrupt input function of the I/O port or setting `USART_IER.DWK=1` to generate an interrupt; if LIN is active, skip this step.

Detection of Synchronous Interval Segments

The detection of the synchronization interval is realized by writing the synchronization interval detection enable bit `USART_CR3.LBDE=1` and writing `USART_CR1.TOE=4` to set the synchronization interval detection function.

Write the desired low-level width in `USART_TOR` (for example, 11 baud rate bit width, `PCLK=48M`, LIN baud rate 20K, `USART_BRR.PSC=1`, then $TOR = 48M / 20K / 4 * 11 = 6600 = 0x19C8$). If the falling edge of `RXD` is detected after `USART_CR3.LBDE=1`, the internal timer starts counting:

- If the counter value is equal to the `USART_TOR` value during `RXD=0`, the hardware sets `USART_SR.LBD=1` and generates an interrupt (if `USART_IER.LBD=1`);
- `RXD` rising edge is detected when the counter value is less than the `USART_TOR` value, stop counting and return to the idle state to continue monitoring the arrival of the next synchronization interval.

Baud Rate Measurements for Sync Segments

Start the USART internal timer by writing the baud rate automatic detection enable bit `USART_CR3.ABRE=1` (the count clock of the timer is the `PCLK` of the USART) to measure the baud rate:

The hardware continuously measures the period between the 5 falling edges of the `RXD` pin, accumulates the measured 4 period values and divides it by 8 to obtain the communication baud rate value. This process is completed by the hardware without software participation.

$$\text{Baud rate} = 1 / (tPCLK * 4PSC * (BDR + 1))$$

`tPCLK`: module working clock cycle

`PSC`: `USART_BRR.PSC` value

`BDR`: `USART_BDR` value

After the measurement is completed, the hardware writes the baud rate value into `USART_BDR`, and at the same time, the hardware sets the automatic baud rate detection flag `USART_SR.ABR`; when `USART_SR.ABR=0`, it is not recommended to read the value of `USART_BDR`, and the software can

set USART_IER.ABR=1, a hardware interrupt is generated when USART_SR.ABR=1. The software can adjust the current USART baud rate according to the USART_BDR value for more precise synchronization. For the setting of the baud rate, refer to [Communication baud rate programming].

LIN slave device software operation steps

- Step1. Set the USART_CR1 register to the reset value;
- Step2. According to the relevant description of the pin digital multiplexing function in the GPIO chapter, configure TXD and RXD as digital input with pull-up, and set GPIOx_AFRH/AFRL to map TXD and RXD to the required pins;
- Step3. Set USART_CR1 (except RE), USART_CR2, USART_CR3, USART_IER registers and USART_CR3.LINEN=1 (if there is step 5, then USART_CR3.LINEN=0);
- Step4. Set the prescaler and baud rate initial value of USART_BRR;

(The following step 5 is to wait for the wake-up signal to arrive, skip it if it is not necessary)

- Step5. Set the I/O port external interrupt to falling edge detection or set USART_IER.DWK=1, and wait for the CPU to be woken up;
- Step6. Set the low-level width of the synchronization interval segment, for example, expect 13 baud rates, then calculate the TOR value according to the formula in the USART_TOR register and write it into the TOR register (if there is step 5, set USART_CR3.LINEN=1);
- Step7. Set USART_CR3.LBDE=1, USART_IER.LBD, USART_CR1.TOE=4 (synchronous interval measurement) and USART_CR1.RE=1. If USART_IER.LBD=0, go to step 8, otherwise go to step 8 after the interrupt occurs;
- Step8. The software queries the USART_SR.LBD value, if it is 1, it means that the synchronization interval segment is detected, and then enters the synchronization segment measurement. Otherwise, continue to query the USART_SR.LBD value;

(The following step 9 is the baud rate measurement of the synchronization segment)

- Step9. Set USART_CR3.LBDE=0, set USART_IER.ABR value and USART_CR3.ABRE=1, USART_CR1.TOE=5. If USART_IER.ABR=0, loop through the USART_SR.ABR value, if USART_IER.ABR=1, wait for the interrupt to occur and go to step 10;
- Step10. Set USART_CR3.ABRE=0, according to the USART_BDR value, if you need to adjust the baud rate, set USART_CR1.RE=0, then reset USART_BRR, the baud rate setting refers to "communication clock source and baud rate programming"; finally set USART_CR1.RE=1; If you do not need to adjust the baud rate, then skip to step 11;

(The following steps 11~13 are to receive the protected ID segment, data and checksum, the process is the same as the asynchronous serial port receiving data process; if the slave is required to send data after parsing the protected ID segment, ignore the following steps 12 and 13, Set USART_CR1.TE=1 and refer to steps 23~31 of [LIN sendLIN send] for operation)

- Step11. Receive the protected ID segment: query the USART_SR.RC value, if it is 1, read the USART_RDR value and perform the operation specified by this value according to the agreement; if USART_SR.RC=0, continue to query. This step can also be done with an interrupt;
- Step12. Receive data: query USART_SR.RC value, if it is 1, read USART_RDR, otherwise continue to query;
- Step13. Receive checksum: Query USART_SR.RC value, if it is 1, read USART_RDR, otherwise continue to query. check the N data received before, then compare with the received checksum, and perform corresponding operations according to the comparison result;
- Step14. After the reception is over, it enters the dormant state and waits for the next wake-up.

17.4 Register description

17.4.1 Register list

Table 17-10 USART base address list

Name	Base address	Description
USART0	0x4000 0000	USART0 base address
USART1	0x4000 0400	USART1 base address
USART2	0x4000 0800	USART2 base address
USART3	0x4000 0C00	USART3 base address

Register bit access permission description:

- 1) USART_CR1.RE=0 and USART_CR1.TE=0, software can write all registers.
- 2) As long as one of USART_CR1.RE and USART_CR1.TE is 1, the only software writable registers are USART_IER, USART_ICR, USART_TDR, USART_TOR, USART_BRR and USART_CR1.TOE, USART_CR1.SLME, USART_CR2.TXDMAEN, USART_CR2.RXDMAEN.
- 3) As long as one of USART_CR1.RE and USART_CR1.TE is 1, the condition that USART_CR1 can be written into a new value by software is: USART_CR1.TE=0 and USART_CR1.RE=0 in the newly written value.
- 4) In LIN mode, the value of the USART_BRR register can be updated when the value of USART_CR1.TE is 1.
- 5) There are no restrictions on reading and writing to the USART_CR3 register.

Table 17-11 USART Register List

Register name	Offset address	Reset value
Control register1 (USART_CR1)	0x00	0x0000 0000
Control register2 (USART_CR2)	0x04	0x0000 0000
Control register3 (USART_CR3)	0x08	0x0000 0000
Baud rate register (USART_BRR)	0x0C	0x0000 FF7F
Timing counter register(USART_TOR)	0x10	0x0000 0000
Status register (USART_SR)	0x14	0x0000 0001
Transmit Data Register(USART_TDR)	0x18	0x0000 01FF
Interrupt Enable Register(USART_IER)	0x1C	0x0000 0000
Interrupt Flag Clear Register(USART_ICR)	0x20	0x0000 7A7E
Receive Data Register(USART_RDR)	0x24	0x0000 0000
Address Register(USART_ADDR)	0x28	0x0000 0000
Address Mask Register(USART_ADDMR)	0x2C	0x0000 0000
Baud rate detection value register(USART_BDR)	0x30	0x0000 0000
Infrared Modulation Register(USART_IR)	0xF0	0x0000 0000

17.4.2 Control Register 1 (USART_CR1)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res	SLME	MPE	COMB	SBS	TOE			OVER8	MSB	STOP	CHLEN	PCE	PS	TE	RE

Bit	Marking	Place name	Function	Read and write
b31~b15	Res	Reserved bit	Reserved, must remain at reset value	
b14	SLME	When multiprocessor communication, silent mode enable bit	0: Normal mode 1: Silent mode Note: 1) When SLME=1, the data whose MPB bit is 0 will not be read from the receiving shift register to the receiving data register, and the error flags PE and FE will not be set. receiving ID data whose MPB is 1 and the address matches, SLME is automatically cleared and starts normal data receiving action. 2) The SLME bit is only valid in multiprocessor communication in asynchronous serial port mode, and this bit must be set to 0 in other modes	RW
b13	MPE	Multiprocessor communication function enable bit (only applicable to asynchronous serial port mode)	0: Invalid 1: Valid Note: 1) The non-asynchronous serial port mode must be set to 0. 2) This bit can only be written when USART_CR1.TE and USART_CR1.RE are both 0	RW
b12	COMB	In asynchronous serial port mode, the composite enable bit of the transceiver port	0: TXD and RXD are used separately 1: Asynchronous serial port single-bus mode, TXD and RXD share TXD port Note: 1) Non-asynchronous serial port mode must be set to 0 2) This bit can only be written when USART_CR1.TE and USART_CR1.RE are both 0	RW
b11	SBS	When receiving data in asynchronous serial port mode, the start bit detection mode setting bit	0: Receive data RXD pin edge(USART_CR2.RXINV=0 is the falling edge, USART_CR2.RXINV=1 is the rising edge) as the start bit 1: Receive data RXD pin level(USART_CR2.RXINV=0 is low level, USART_CR2.RXINV=1 is high level) as the start bit Note: 1) Non-asynchronous serial port mode must be set to 0 2) This bit can only be written when USART_CR1.TE and USART_CR1.RE are both 0	RW
b10~b8	TOE	Multi-function timer working mode configuration bit	000: stop timer counter 001: General timing function 010: Receive timing function 011: Timing function between data frames 100: In LIN mode, synchronous interval segment counting is enabled 101: In LIN mode or asynchronous serial mode, baud rate automatic detection is enabled 110~111: Reserved, same as 000, counting function off Note: 1) TOE of all functions are cleared by hardware 2) If switching from one function (such as TOE=010) to another function (such as TOE=001), the software needs to write TOE=000 first and then write TOE=001	RW
b7	OVER8	asynchronous serial port mode /LIN mode, the sampling mode setting bit	0: 16 times sampling 1: 8x sampling Note: 1) Synchronous serial port mode and smart card mode must be set to 0, infrared mode setting is invalid 2) This bit can only be written when USART_CR1.TE and USART_CR1.RE are both 0	RW
b6	MSB	MSB/LSB mode select bit	0: LSB mode 1: MSB mode Note: 1) This bit can only be written when USART_CR1.TE and USART_CR1.RE are both 0	RW

b5	STOP	stop bit length setting bit	0: 1 stop bit 1: 2 stop bit Note: 1) Synchronous serial port mode and LIN mode must be set to 0 2) This bit can only be written when USART_CR1.TE and USART_CR1.RE are both 0	RW
b4	CHLEN	Send / receive data length setting bit	0: 8 bits 1: 9 bits Note: 1) Synchronous serial port mode and LIN mode must be set to 0 2) This bit can only be written when USART_CR1.TE and USART_CR1.RE are both 0	RW
b3	PCE	parity valid bit	0: No check 1: Verification Note: 1) Synchronous serial port mode and LIN mode must be set to 0 2) This bit can only be written when USART_CR1.TE and USART_CR1.RE are both 0	RW
b2	PS	parity select bit	0: Even check 1: Parity Note: 1) The PS bit is only valid when PCE=1 2) This bit can only be written when both USART_CR1.TE and USART_CR1.RE are 0.	RW
b1	TE	send enable bit	0: Send disabled 1: Send enable Note: 1) In synchronous serial port mode(USART_CR2.SYNC=1), the TE bit can only be written to 1 when TE=0 and RE=0(send / receive disabled). If it is full-duplex in synchronous mode, TE and RE must be at the same time Write 1; when TE=1 or RE=1, this bit can only write 0 with the RE bit at the same time; when TE=1 and RE=1, this bit can only write 0 with the RE bit at the same time. Unlimited when USART_CR2.SYNC=0 2) TE setting is invalid when single-line half-duplex work	RW
b0	RE	receive enable bit	0: Reception disabled 1: Receive enable Note: 1) In synchronous serial port mode(USART_CR2.SYNC=1), the RE bit can only be written to 1 when TE=0 and RE=0(send / receive disabled). If it is full-duplex in synchronous mode, RE and TE must be at the same time Write 1; when TE=1 or RE=1, this bit can only write 0 with the TE bit at the same time; when TE=1 and RE=1, this bit can only write 0 with the TE bit at the same time. USART_CR2.SYNC=0.	RW

17.4.3 Control Register 2 (USART_CR2)

Reset value: 0x0000 0000

Note: [11:10] are reserved bits, the written data must be the reset value

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res						CLKC	SWAP	SYNC	TXINV	RXINV	CTSEN	RTSEN	TXDMAEN	RXDMAEN	

Bit	Marking	Place name	Function	Read and write
b31~b10	Res	Reserved bit	Reserved, must remain at reset value	
b9~b8	CLKC	In asynchronous serial mode / synchronous serial mode / smart card mode, the clock control bit	Asynchronous serial port mode 00: The clock source is the clock generated by the internal baud rate generator, the clock is not output to the CK pin, and the CK pin can be used as a normal IO 01: The clock source is the clock generated by the internal baud rate generator, the clock is output to the CK pin, and the output clock frequency is the same as the baud rate Synchronous serial port mode 00, 01: The clock source is the clock generated by the internal baud rate generator, output to the CK pin 10, 11: The clock source is an external input clock, and the frequency of the input clock is the same as the baud rate Smart card mode 00: The clock source is the clock generated by the internal baud rate generator, the clock is not output to the CK pin, and the CK pin can be used as a normal IO 01: The clock source is the clock generated by the internal baud rate generator, and the clock is output to the CK pin 10, 11: Setting prohibited Note: 1) Infrared mode and LIN mode settings are invalid, neither CK can be output to CK pin 2) This bit can only be written when USART_CR1.TE and USART_CR1.RE are both 0	RW
b7	SWAP	TXD/RXD pin swap control bit	0: not interchangeable 1: exchange Note: 1) This bit can only be written when USART_CR1.TE and USART_CR1.RE are both 0	RW
b6	SYNC	Synchronous mode select bit	0: asynchronous serial port mode 1: synchronous serial port mode Note: 1) Asynchronous serial port mode, smart card mode and LIN mode must write 0 2) Synchronous serial port mode must write 1 3) The infrared mode setting is invalid 4) This bit can only be written when USART_CR1.TE and USART_CR1.RE are both 0	RW
b5	TXINV	Output signal inversion control bit	0: no inversion 1: TXD output pin signal inversion Note: 1) This bit can only be written when USART_CR1.TE and USART_CR1.RE are both 0	RW
b4	RXINV	Input signal inversion control bit	0: no inversion 1: RXD input pin signal inversion Note: 1) This bit can only be written when USART_CR1.TE and USART_CR1.RE are both 0	RW
b3	CTSEN	asynchronous serial port mode / synchronous serial port mode, CTS function enable bit	0: Prohibited 1: Enable Note: 1) Non-asynchronous serial port mode and non-synchronous serial port mode must be set to 0 2) This bit can only be written when USART_CR1.TE and USART_CR1.RE are both 0	RW

b2	RTSEN	asynchronous serial port mode / synchronous serial port mode, the RTS function enable bit	0: Prohibition 1: Enable Note: 1) Non-asynchronous serial port mode and non-synchronous serial port mode must be set to 0 2) This bit can only be written when USART_CR1.TE and USART_CR1.RE are both 0	RW
b1	TXDMAEN	TX DMAC hardware handshake signal enable bit	0: Turn off hardware handshake signal 1: Turn on the hardware handshake signal Note: 1) DMA mode is not supported when single-line half-duplex operation	RW
b0	RXDMAEN	RX DMAC hardware handshake signal enable bit	0: Turn off hardware handshake signal 1: Turn on the hardware handshake signal Note: 1) DMA mode is not supported when single-line half-duplex operation	RW

17.4.4 Control Register 3 (USART_CR3)

Reset value: 0x0000 0000

Note: [12:10] are reserved bits, the written data must be the reset value

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
IRMOD	IRCW	IREN	Res			SMCPU		BCN			SCE	BRE	ABRE	LBDE	LINE

Bit	Marking	Place name	Function	Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value	
b15	IRMOD	In infrared mode, 38K modulation mode	0: Output 38K carrier when data is 0 1: Output 38K carrier when data is 1 Note: 1) Non-infrared mode settings are invalid	RW
b14	IRCW	In infrared mode, infrared carrier selection	0: $\frac{3}{16}$ mode 1: 38K carrier mode Note: 1) Non-infrared mode settings are invalid	RW
b13	IREN	In infrared mode, the infrared mode enable bit	0: Infrared function module off 1: The infrared function module is turned on Note: 1) Non-infrared mode must be set to 0	RW
b12~10	Res	Reserved bit	Reserved, must remain at reset value	
b9~8	SMCPU	In smart card mode, output edge shaping	00: Output without edge shaping when open drain 01: When open-drain, strong output CMOS high level for 1 PCLK cycle 10: When open-drain, strong output CMOS high level for 2 PCLK cycle 11: When open-drain, strong output CMOS high level for 4 PCLK cycle Note: 1) The non-smart card mode setting is invalid	RW
b7~5	BCN	In smart card mode, sets the number of base clocks during a one-bit data transfer	000: 16 001: 32 010: 64 011: 128 100: 186 101: 256 110: 372 111: 512 Note: 1) The non-smart card mode setting is invalid	RW
b4	SCEN	In smart card mode, the mode enable bit	0: disable smart card mode 1: Enable Smart Card Pattern Note: 1) Non-smart card mode must be set to 0	RW
b3	BREAK	LIN mode, send sync interval enable	1: The value written into the USART_TDR register represents the number of dominant levels that need to be sent in the synchronization interval. The value should not be less than 13, and it will be automatically cleared by hardware after the transmission is completed. 0: The value written to the USART_TDR register represents the data sent normally Note: 1) Non- LIN mode must be set to 0	RW
b2	ABRE	asynchronous serial port mode /LIN mode, automatic baud rate detection is enabled	0: stop detecting baud rate 1: start detecting baud rate Note: 1) Non-asynchronous serial port mode and non- LIN mode must be set to 0	RW
b1	LBDE	LIN mode, the synchronization interval segment detection enable bit	0: Disable synchronization interval segment detection 1: Enable synchronization interval segment detection Note: 1) Non- LIN mode must be set to 0	RW

b0	LINEN	LIN mode, the mode enable bit	0: disable LIN mode 1: Enable LIN mode Note: 1) Non- LIN mode must be set to 0	RW
----	-------	-------------------------------	---	----

17.4.5 Baud rate register (USART _ BRR)

Reset value: 0x0000 FF7F

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res														PSC	
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
DIVI								FBME	DIVF						

Bit	Marking	Place name	Function	Read and write
b31~b18	Res	Reserved bit	Reserved, must remain at reset value	
b17~b16	PSC	When the internal clock source is used, the prescaler select bit	00: PCLK 01: PCLK/4 10: PCLK/16 11: PCLK/64	RW
b15~b8	DIVI	Integer frequency division	Integer frequency division register	RW
b7	FBME	Fractional baud rate generation enable bit when internal baud rate generator is selected	0: Prohibited 1: Enable	RW
b6~b0	DIVF	Fractional division	Fractional divider register of DIVF is only valid when FBME=1; it is recommended to enable the fractional frequency division when the module works at 8M and below.	RW

17.4.6 Timing counter register(USART_TOR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res												TOR[19:16]			
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
TOR[15:0]															

Bit	Marking	Place name	Function	Read and write
b31~b20	Res	Reserved bit	Reserved, must remain at reset value	
b19~b0	TOR	Timing counting time	Timing counting time= $t_{PCLK} * 4^{PSC} * (TOR+1)$	RW

17.4.7 Status register (USART_SR)

Reset value: 0x0000 0001

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res	DWK	LBD	ABR	CTS	CTSV	AM	MPB	Res	TO	OR	FE	PE	RC	TC	TXE

Bit	Marking	Place name	Function	Read and write
b31~b15	Res	Reserved bit	Reserved, must remain at reset value	
b14	DWK	In asynchronous serial port mode / infrared mode /LIN mode, RXD line wake-up flag	Set by hardware, cleared by software 0: no wake up 1: A wakeup has occurred	RO
b13	LBD	LIN mode, sync interval detection flag	Set by hardware, cleared by software 0: Sync interval not detected 1: Sync interval detected	RO
b12	ABR	Automatic baud rate detection flag in asynchronous serial port mode /LIN mode	Set by hardware, cleared by software 0: detection not complete 1: detection completed	RO
b11	CTS	asynchronous serial port mode / synchronous serial port mode, CTS interrupt flag bit	Set by hardware, cleared by software 0: CTS signal does not toggle 1: CTS signal flipped	RO
b10	CTSV	asynchronous serial port mode / synchronous serial port mode, the current level value of CTS pin	0: CTS pin is low 1: CTS pin is high level	RO
b9	AM	When multiprocessors communicate, the address match flag	0: address does not match 1: address match	RO
b8	MPB	During multiprocessor communication, the multiprocessor flag bit	0: receive data as data 1: Receive data as ID Note: 1) The MPB bit is only valid for multiprocessor communication	RO
b7	Res	Reserved bit	Reserved, must remain at reset value	
b6	TO	Timer match or overflow flag	0: Timer did not match or overflow 1: Timer has matched or overflowed	RO
b5	OR	overflow error flag	Set by hardware, cleared by software 0: no overflow 1: overflow occurs Note: 1) OR set condition: When the received data register has a value, new data is received. 2) RE=0 does not reset the OR bit. 3) The data received after OR=1 will overwrite the previous data.	RO
b4	FE	In asynchronous serial port mode / smart card mode / infrared mode /LIN mode, receive frame error flag	Set by hardware, cleared by software 0: No received frame error 1: Receive frame error Note: 1) Setting condition in asynchronous serial port mode / infrared mode /LIN mode: the stop bit of the received data frame is 0. 2) Setting condition in smart card mode: low level error signal flag is sampled. 3) RE=0 does not reset the FE bit. 4) The data received after FE=1 will overwrite the previous data.	RO
b3	PE	In asynchronous serial port mode / smart card mode / infrared mode, receive data verification error flag	Set by hardware, cleared by software 0: no parity error 1: A parity error occurred Note: 1) PE setting condition: a parity error occurs in the received data. 2) RE=0 does not reset the PE bit. 3) The data received after PE=1 will overwrite the previous data.	RO
b2	RC	Data receiving complete flag	0: non-receive complete status 1: Receive completed Note: 1) After receiving the stop bit, it is set to 1 by the hardware, and the data is automatically cleared after being read by the software. The software can also write USART_ICR.RC=0 to clear.	RO

b1	TC	Send completion flag bit	Set by hardware, cleared by software 0: Send data 1: Completion of data transmission Note: 1) Setting conditions in asynchronous mode and clock synchronous mode: When a frame of data transmission is completed (including stop bit), the transmit data register is empty(TXE=1). TC clearing condition: write 0 to USART_ICR.TC. 2) Setting condition in smart card mode: After a certain time has elapsed after the last 1 byte of data is sent, FE=0 and the sending data register is empty(TXE=1). The specific timing of TC setting is: 2.5 digits after the check digit is sent. TC clearing condition: write 0 to USART_ICR.TC.	RO
b0	TXE	Send data register null flag	0: Data is not transferred to the shift register 1: Data is transferred to the shift register Note: 1) TXE is set to 1 and cleared to 0 by hardware: the hardware sets to 1 after the data is transmitted to the shift register, and the hardware clears to 0 after the data is written into the transmit data register.	RO

17.4.8 Transmit Data Register(USART_TDR)

Reset value: 0x0000 01FF

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res						MPID	TDR								

Bit	Marking	Place name	Function	Read and write
b31~b10	Res	Reserved bit	Reserved, must remain at reset value	
b9	MPID	Multiprocessor ID bit	0: Send data 1: Sending ID Note: 1) The MPID bit is only valid in multiprocessor communication, and must be set to 0 in other modes.	RW
b8~b0	TDR	Send data	Send data Note: 1) The highest bit TDR[8] is only valid in the asynchronous serial port mode and the data length is set to 9 bits.	RW

17.4.9 Interrupt Enable Register(USART_IER)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res	DWK	LBD	ABR	CTS	Res	AM	Res	TO	OR	FE	PE	RC	TC	TXE	

Bit	Marking	Place name	Function	Read and write
b31~b15	Res	Reserved bit	Reserved, must remain at reset value	
b14	DWK	asynchronous serial port mode / infrared mode /LIN mode, RXD pin edge interrupt enable	RXD pin edge interrupt enable (USART_CR2.RXINV=0 is falling edge, USART_CR2.RXINV=1 is rising edge) 0: interrupt disabled 1: interrupt enabled	RW
b13	LBD	LIN mode, the synchronous interval segment interrupt enable	0: interrupt disabled 1: interrupt enable	RW
b12	ABR	Interrupt enable for auto-baud rate detection in asynchronous serial mode /LIN mode	0: interrupt disabled 1: interrupt enable	RW
b11	CTS	asynchronous serial port mode / synchronous serial port mode, CTS signal toggle interrupt enable	0: interrupt disabled 1: interrupt enable	RW
b10	Res	Reserved bit	Reserved, must remain at reset value	
b9	AM	In the multi-machine communication mode, the interrupt enable after the address match	0: interrupt disabled 1: interrupt enable	RW
b8~b7	Res	Reserved bit	Reserved, must remain at reset value	
b6	TO	Timer / Event Counter Overflow Interrupt Enable	0: interrupt disabled 1: interrupt enable	RW
b5	OR	The error interrupt enable when the receive data register is not read and new data is received	0: interrupt disabled 1: interrupt enable	RW
b4	FE	asynchronous serial port mode / smart card mode / infrared mode /LIN mode, the interrupt enable of receiving frame error	0: interrupt disabled 1: interrupt enable	RW
b3	PE	asynchronous serial port mode / smart card mode / infrared mode, the check error interrupt enable after a frame of data is received	0: interrupt disabled 1: interrupt enable	RW
b2	RC	Interrupt request enable bit after a frame of data is received and written to the receive data register	0: interrupt disabled 1: interrupt enabled	RW
b1	TC	Interrupt request enable bit when a frame of data transmission is completed	0: interrupt disabled 1: interrupt enabled	RW
b0	TXE	Transmit buffer empty status interrupt enable	0: interrupt disabled 1: interrupt enable	RW

17.4.10 Interrupt Flag Clear Register(USART_ICR)

Reset value: 0x0000 7A7E

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res	DWK	LBD	ABR	CTS	Res	AM	Res	TO	OR	FE	PE	RC	TC	Res	

Bit	Marking	Place name	Function	Read and write
b31~b15	Res	Reserved bit	Reserved, must remain at reset value	
b14	DWK	DWK clear bit	0: Clear DWK 1: Does not affect DWK value	WO
b13	LBD	LBD clear bit	0: Clear LBD 1: Does not affect LBD value	WO
b12	ABR	ABR clear bit	0: Clear ABR 1: Does not affect ABR value	WO
b11	CTS	CTS clear bit	0: Clear CTS 1: Does not affect CTS value	WO
b10	Res	Reserved bit	Reserved, must remain at reset value	
b9	AM	AM clear bit	0: Clear AM 1: Does not affect the AM value	WO
b8~b7	Res	Reserved bit	Reserved, must remain at reset value	
b6	TO	TO clear bit	0: Clear TO 1: Does not affect TO value	WO
b5	OR	OR clear bit	0: Clear OR 1: Does not affect the OR value	WO
b4	FE	FE clear bit	0: clear FE 1: Does not affect FE value	WO
b3	PE	PE clear bit	0: clear PE 1: Does not affect the PE value	WO
b2	RC	RC clear bit	0: clear RC 1: Does not affect the RC value	WO
b1	TC	TC clear bit	0: clear TC 1: Does not affect the TC value	WO
b0	Res	Reserved bit	Reserved, must remain at reset value	

17.4.11 Receive Data Register(USART_RDR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res							RDR								

Bit	Marking	Place name	Function	Read and write
b31~b9	Res	Reserved bit	Reserved, must remain at reset value	
b8~b0	RDR	Receiving data	The highest bit RDR[8] is valid only in the asynchronous serial port mode and the data length is set to 9 bits.	RO

17.4.12 Address Register(USART_ADDR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res							ADDR								

Bit	Marking	Place name	Function	Read and write
b31~b8	Res	Reserved bit	Reserved, must remain at reset value	
b7~b0	ADDR	Slave configuration address	This bit can only be written when both USART_CR1.TE and USART_CR1.RE are 0.	RW

17.4.13 Address Mask Register(USART_ADDMR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res							ADDM								

Bit	Marking	Place name	Function	Read and write
b31~b8	Res	Reserved bit	Reserved, must remain at reset value	
b7~b0	ADDM	Slave address mask	0: Corresponding bits do not participate in matching 1: Corresponding bit participates in matching Note: 1) Slave address = USART_ADDR.ADDR & USART_ADDMR.ADDM. 2) This bit can only be written when both USART_CR1.TE and USART_CR1.RE are 0.	RW

17.4.14 Baud rate detection value register(USART_BDR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res				BDR											

Bit	Marking	Place name	Function	Read and write
b31~b12	Res	Reserved bit	Reserved, must remain at reset value	
b11~b0	BDR	Baud rate detection value in asynchronous serial port mode /LIN mode	The communication baud rate can be calculated by the following formula $\text{Baud} = \frac{f_{\text{PCLK}}}{4^{\text{PSC}} \times (\text{BDR} + 1)}$ Note: 1) Only applicable to LIN mode or asynchronous serial mode.	RO

17.4.15 Infrared Modulation Register(USART_IR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res					DCNTH										
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res					DCNTL										

Bit	Marking	Place name	Function	Read and write
b31~b27	Res	Reserved bit	Reserved, must remain at reset value	
b26~b16	DCNTH	In infrared mode, high level count value	38K high level hold time $t = t_{\text{PCLK}} * (\text{DCNTH} + 1)$	RW
b15~b11	Res	Reserved bit	Reserved, must remain at reset value	
b10~b0	DCNTL	In infrared mode, low level count value	38K low level maintenance time $t = t_{\text{PCLK}} * (\text{DCNTL} + 1)$	RW

18 Cyclic redundancy check (CRC)

18.1 Overview

A cyclic redundancy check(CRC) calculation unit takes a data stream or block of data as input and produces an output number under the control of a generator polynomial. This output number is often used to verify the correctness and integrity of data transmission or storage. This module supports calculating CRC value and checking CRC value.

18.2 Main characteristics

- One implementation standard: ISO/IEC13239
- Two encoding methods: CRC-16, CRC-32
- Three write bit widths: 8bit, 16bit, 32bit
- Two working modes: CRC encoding mode, CRC check mode
- CRC-16/ CRC-16 polynomial: $x^{16}+x^{12}+x^5+1$
- CRC-32 polynomial: $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

18.3 Functional description

18.3.1 Operating mode

This module supports two working modes: CRC encoding mode and CRC checking mode.

The CRC encoding mode is to input a certain amount of raw data to the CRC module and obtain the output value (CRC_RESULT) generated by the CRC module. The CRC check mode is to point to the CRC module to input a certain number of encoded data sequences (original data + CRC check value), and verify whether the original data matches the CRC check value (CRC_CSR.FLAG).

18.3.2 Encoding

This module supports two encoding methods CRC-16 and CRC-32, and the calculation results are 16 bits and 32 bits respectively. Configure the encoding method to be used through CRC_CSR.CODE.

CRC-16 polynomial: $x^{16}+x^{12}+x^5+1$.

CRC-32 polynomial: $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$.

18.3.3 Write bit width

This module supports three write bit widths: 8bit, 16bit, 32bit. The writing of different bit widths needs to comply with the principle of "consistent bit width, low first and then high", that is, "every time data is written, it must be written into a register equal to the effective data bit width of this time, and the lower bit Data is written before higher bit data".

The following shows how the same sequence of data is written using three bit widths, and the output results are the same.

- 8bit bit width write: 0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77
- 16bit bit width write: 0x1100, 0x3322, 0x5544, 0x7766
- 32bit bit width write: 0x33221100, 0x77665544

18.4 Programming example

18.4.1 CRC-16 encoding mode

- Step1. Write 0x00 to CRC_CSR.CODE, select CRC-16;
- Step2. Write 0xFFFF to CRC_RESULT[15:0] to initialize CRC calculation;
- Step3. Write the original data to be encoded into the CRC_DR register in sequence, and the write bit width can be selected from 8bit, 16bit, and 32bit;
- Step4. Query CRC_CSR.BUSY until CRC_CSR.BUSY becomes zero;
- Step5. Read CRC_RESULT[15:0] to get the CRC-16 calculation result.

18.4.2 CRC-16 check mode

- Step1. Write 0x00 to CRC_CSR.CODE, select CRC-16;
- Step2. Write 0xFFFF to CRC_RESULT[15:0] to initialize CRC calculation;
- Step3. Write the encoded data sequence (original data + CRC check value) into the CRC_DR register in sequence, and the write bit width can be selected from 8bit, 16bit, 32bit;
- Step4. Query CRC_CSR.BUSY until CRC_CSR.BUSY becomes zero;
- Step5. Determine whether the encoded data sequence has been tampered with according to the value of CRC_CSR.FLAG.

18.4.3 CRC-32 encoding mode

- Step1. Write 0x01 to CRC_CSR.CODE, select CRC-32;
- Step2. Write 0xFFFFFFFF to CRC_RESULT[31:0] to initialize CRC calculation;
- Step3. Write the original data to be encoded into the CRC_DR register in sequence, and the write bit width can be selected from 8bit, 16bit, and 32bit;
- Step4. Query CRC_CSR.BUSY until CRC_CSR.BUSY becomes zero;
- Step5. Read CRC_RESULT[31:0] to get the CRC-32 calculation result.

18.4.4 CRC-32 check mode

- Step1. Write 0x01 to CRC_CSR.CODE, select CRC-32;
- Step2. Write 0xFFFFFFFF to CRC_RESULT[31:0] to initialize CRC calculation;
- Step3. Write the encoded data sequence (original data + CRC check value) into the CRC_DR register in sequence, and the write bit width can be selected from 8bit, 16bit, 32bit;
- Step4. Query CRC_CSR.BUSY until CRC_CSR.BUSY becomes zero;
- Step5. Determine whether the encoded data sequence has been tampered with according to the value of CRC_CSR.FLAG.

18.5 Register description

18.5.1 Register list

Base address: 0x4002 0800

Table 18-1 CRC register list

Register name	Offset address	Reset value
Control and Status Register(CRC_CSR)	0x00	0x0000 0001
Results register (CRC_RESULT)	0x04	0x0000 0000
Data Register(CRC_DR)	0x80	0x0000 0000

18.5.2 Control and Status Register (CRC_CSR)

Reset value: 0x0000 0001

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res													BUSY	FLAG	CODE

Bit	Marking	Place name	Function	Read and write
b31~b3	Res	Reserved bit	Reserved, must remain at reset value	
b2	BUSY	CRC busy flag	0: The current CRC is idle or the calculation is complete 1: The current CRC is busy or calculating	RO
b1	FLAG	CRC check result	0: The current CRC check error 1: The current CRC check is correct	RO
b0	CODE	CRC encoding method selection	0: CRC-16 encoding 1: CRC-32 encoding	RW

18.5.3 Results register (CRC_RESULT)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
RESULT[31:16]															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
RESULT[15:0]															

Bit	Marking	Place name	Function	Read and write
b31~b0	RESULT	CRC calculation result	Read RESULT[15:0] to get the calculation result of CRC-16 Read RESULT[31:0] to get the calculation result of CRC-32 Write 0xFFFF to RESULT[15:0] to initialize CRC-16 calculation Write 0xFFFFFFFF to RESULT[31:0] to initialize CRC-32 calculation	RW

18.5.4 Data Register (CRC_DR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
DATA[31:16]															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
DATA[15:0]															

Bit	Marking	Place name	Function	Read and write
b31~b0	DATA	CRC write data	This register is used to write data that needs to be calculated, and supports 3 write bit widths 8bit writing method: *((uint8_t *)0x40020880) = 0xxx 16bit writing method: *((uint16_t *)0x40020880) = 0xxxxx 32bit writing method: *((uint32_t *)0x40020880) = 0xxxxxxxxx	WO

19 Analog-to-Digital Converters (ADC)

19.1 Module Introduction

External analog signals need to be converted into digital signals to be further processed by the MCU. It integrates two 12-bit successive approximation analog-to-digital converter (SAR ADC) modules with high precision and high conversion rate. Has the following properties:

- 12-bit conversion accuracy
- 1M SPS conversion speed
- 18 input channels, including 16 external pin inputs, 1 internal temperature sensor voltage, and 1 1/3 AVCC voltage
- 4 reference sources: AVCC voltage, ExRef pin, built-in 1.5V reference voltage, built-in 2.5V reference voltage
- ADC voltage input range: 0~Vref
- Multiple conversion modes: single conversion, sequential scan single conversion, sequential scan continuous conversion, sequential scan intermittent conversion, sequential scan single conversion accumulation
- Input channel voltage threshold monitoring
- Software can configure ADC conversion rate
- Built-in signal follower, which can convert high-impedance signals
- Support on-chip peripherals to automatically trigger ADC conversion, effectively reducing chip power consumption and improving real-time conversion

19.2 ADC block diagram

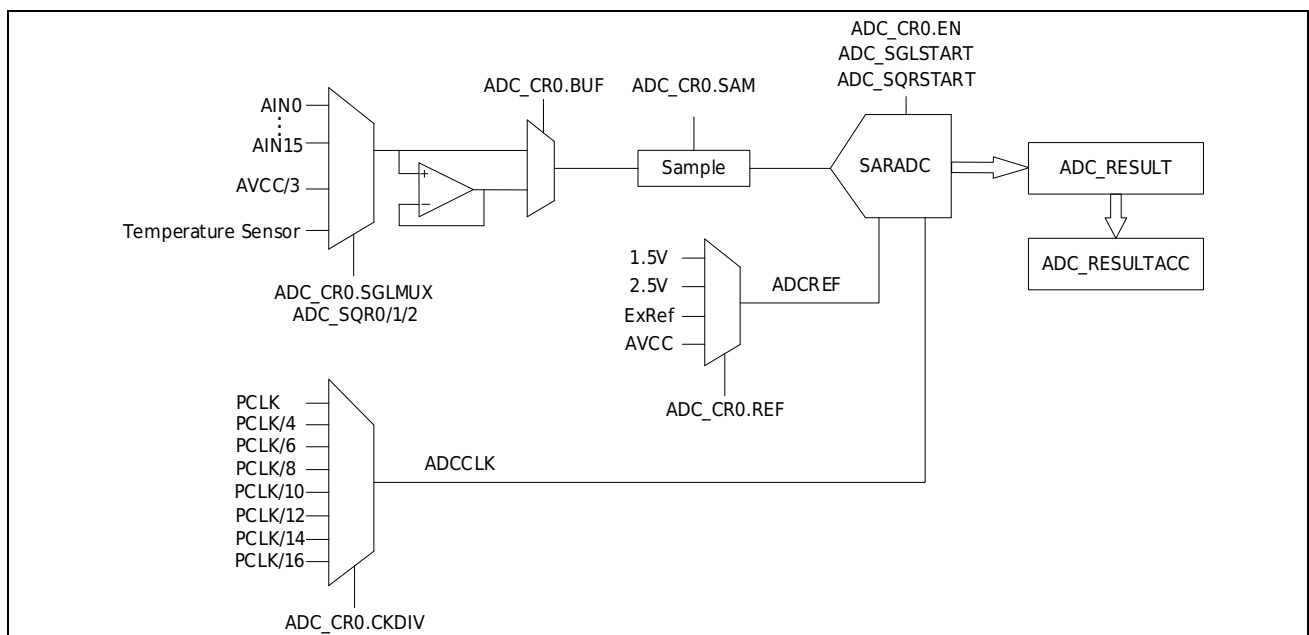


Figure 19-1 ADC block diagram

19.3 Conversion Timing and Conversion Speed

The ADC conversion timing is shown in the figure below: a complete ADC conversion consists of a conversion process and a successive comparison process. , the conversion process requires 3~10 ADCCLKs, which are configured by ADC_CR0.SAM; the successive comparison process requires 15 ADCCLKs. Therefore, an ADC conversion requires a total of 18~ 25 ADCCLKs.

ADC conversion speed is SPS, that is, how many ADC conversions are performed per second. the ADC conversion speed is: the frequency of ADCCLK / the number of ADCCLKs required for one ADC conversion.

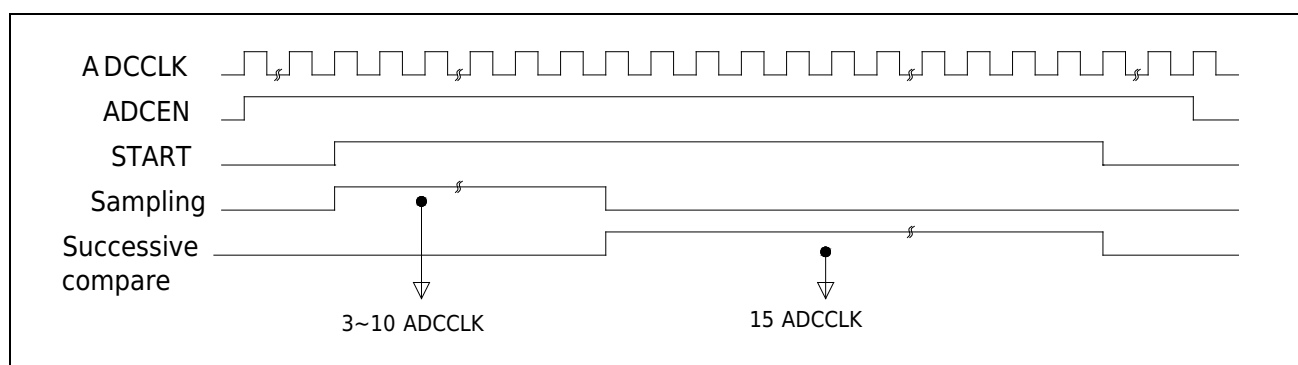


Figure 19-2 ADC conversion timing diagram

The ADC conversion speed is related to the ADC reference voltage and AVCC voltage. The maximum conversion speed is shown in the following table:

Table 19-1 ADC maximum conversion speed

ADC reference voltage	AVCC voltage	Maximum conversion speed	Maximum ADCCLK frequency
Internal 1.5V	1.8~5.5V	200K SPS	5MHz
Internal 2.5V	2.8~5.5V	200K SPS	5MHz
AVCC / ExRef	1.8~2.4V	200K SPS	5MHz
AVCC / ExRef	2.4~2.7V	500K SPS	12.5MHz
AVCC / ExRef	2.7~5.5V	1M SPS	25MHz

19.4 Single conversion mode

In the single conversion mode, only one conversion is performed after the ADC starts, and all 19 ADC channels can be converted. This mode can be started either by setting the ADC_SGLSTART.START bit or by setting the external trigger of ADC_EXTTRIG0. Once the ADC conversion of the selected channel is completed, the ADC_IFR.ESG bit is automatically set to 1, and the conversion result is saved in the ADC_RESULT register.

Start the ADC single conversion operation process through the ADC_SGLSTART.START bit:

Step1. Configure the corresponding bit of GPIOx_ADS to configure the ADC channel to be converted as an analog port;

Step2. Set GPIOB_ADS.bit1 to 1, and configure the ADC external reference voltage pin(PB01) as an analog port;

Note: If the ADC reference voltage does not select an external reference voltage pin, this step can be skipped.

Step3. Set ADC_BGR.BGR_EN to 1 to enable the BGR module;

Step4. Set ADC_CR0.EN to 1 to enable the ADC module;

Step5. Delay 20us, wait for the ADC and BGR module to start up;

Step6. Set ADC_CR1.Mode to 0, select single conversion mode;

Step7. Configure ADC_CR0.REF to select the reference voltage of ADC;

Step8. Configure ADC_CR0.SAM and ADC_CR0.CKDIV to set the conversion speed of ADC;

Step9. Configure ADC_CR0.SGLMUX to select the channel to be converted;

Step10. Set ADC_ICR.ESG to 0, clear the ADC_IFR.ESG flag;

Step11. Set ADC_SGLSTART.START to 1 to start ADC single conversion;

Step12. Wait for ADC_IFR. ESG to become 1, read the ADC_RESULT register to get the ADC conversion result;

Step13. If you need to convert other channels, repeat Step9~Step12;

Step14. Set ADC_CR0.EN and ADC_BGR.BGR_EN to 0, turn off the ADC module and BGR module.

ADC single conversion operation process through an external trigger:

Step1. Configure the corresponding bit of GPIOx_ADS to configure the ADC channel to be converted as an analog port;

Step2. Set GPIOB_ADS.bit1 to 1, and configure the ADC external reference voltage pin(PB01) as an analog port;

Note: If the ADC reference voltage does not select an external reference voltage pin, this step can be skipped.

Step3. Set ADC_BGR.BGR_EN to 1 to enable the BGR module;

Step4. Set ADC_CR0.EN to 1 to enable the ADC module;

Step5. Delay 20us, wait for the ADC and BGR module to start up;

Step6. Set ADC_CR1.MODE to 0, select single conversion mode;

Step7. Set ADC_IER.ESG to 1 to enable ADC interrupt;

- Step8. Enable the ADC interrupt in the NVIC interrupt vector table;
- Step9. Configure ADC_CR0.REF to select the reference voltage of ADC;
- Step10. Configure ADC_CR0.SAM and ADC_CR0.CKDIV to set the conversion speed of ADC;
- Step11. Configure ADC_CR0.SGLMUX to select the channel to be converted;
- Step12. Set ADC_ICR to 0x0, clear the ADC interrupt flag;
- Step13. Configure ADC_EXTTRIGGER0 to select the external trigger condition;
- Step14. When an external trigger condition triggers the ADC to complete the conversion, the ADC module will generate an interrupt. The user can read the ADC_RESULT register in the ADC interrupt service program to obtain the ADC conversion result;
- Step15. If you need to convert other channels, repeat Step11~Step14;
- Step16. Set ADC_CR0.EN and ADC_BGR.BGR_EN to 0, turn off the ADC module and BGR module.

19.5 Scan conversion mode

The scan conversion mode is divided into three modes: sequential scan single conversion, sequential scan continuous conversion and sequential scan intermittent conversion. When the three modes work independently, they can perform multiple conversions on multiple channels continuously.

19.5.1 Sequential Scan Single Conversion Mode

The sequential scan single conversion mode can perform up to 16 consecutive conversions, and the total number of conversions is configured by ADC_SQR2.CNT; all 20 channels can be configured for conversion, and the channel to be converted is configured by ADC_SQRx.CHxMUX. This mode can be started either by setting the ADC_SQRSTART.START bit or by setting the external trigger of ADC_EXTTRIGGER0. After starting the conversion, the ADC module converts the channels configured in CH0MUX~CHxMUX sequentially until the total conversion times are completed. ADC_RESULT register after each conversion, and the ADC_IFR.EOC bit is automatically set to 1. The conversion result needs to be read out in time. The ADC_IFR.EOS bit is automatically set to 1 after the ADC module completes the total number of conversions. The figure below demonstrates the sequential scan conversion process with 8 conversions for AIN0 AIN1 AIN5. Among them, the sequential scanning conversion channels 7, 4 and 1 are configured as AIN0, the conversion channels 6, 3 and 0 are configured as AIN1, and the conversion channels 5 and 2 are configured as AIN5. ADC_SQRSTART.START is set to 1, the ADC module will convert sequential scan conversion channels 0~7 in turn.

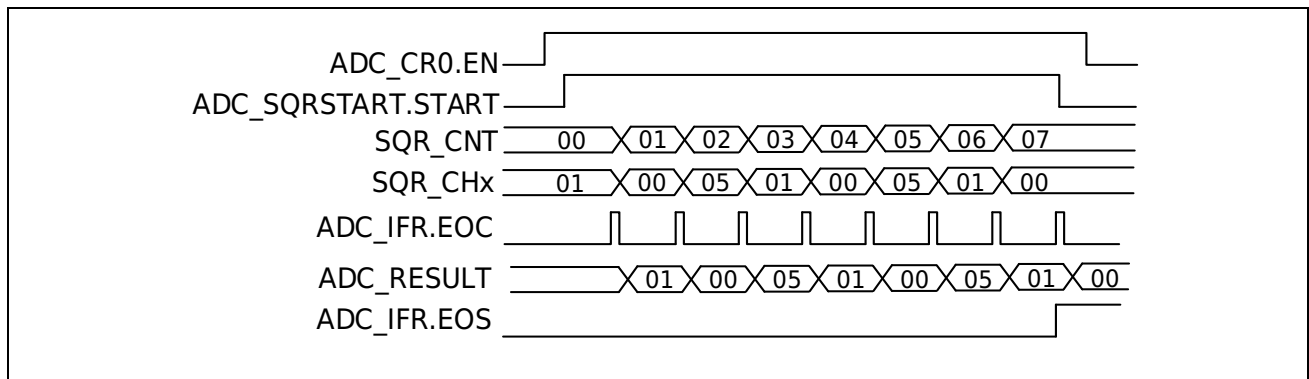


Figure 19-3 ADC sequential scan conversion process example

Start the ADC sequential scan conversion operation process through the ADC_SQRSTART.START bit:

- Step1. Configure the corresponding bit of GPIOx_ADS to configure the ADC channel to be converted as an analog port;
- Step2. Set GPIOB_ADS.bit1 to 1, and configure the ADC external reference voltage pin(PB01) as an analog port;

Note: If the ADC reference voltage does not select an external reference voltage pin, this step can be skipped.

- Step3. Set ADC_BGR.BGR_EN to 1 to enable the BGR module;
- Step4. Set ADC_CR0.EN to 1 to enable the ADC module;
- Step5. Delay 20us, wait for the ADC and BGR module to start up;
- Step6. Set ADC_CR1.MODE to 1 to select the scan conversion mode;
- Step7. Configure ADC_CR0.REF to select the reference voltage of ADC;
- Step8. Configure ADC_CR0.SAM and ADC_CR0.CKDIV to set the conversion speed of ADC;
- Step9. Configure ADC_SQRx.CHxMUX to select sequential scan conversion channels;
- Step10. Configure ADC_SQR2.CNT to select the total number of conversions for sequential scan conversions;

Note: The total number of conversions needs to be consistent with the number of conversion channels configured in Step9.

- Step11. Set ADC_ICR.EOS/EOC to 0, clear the ADC_IFR.EOC/EOS flag;
- Step12. Set ADC_SQRSTART.START to 1 to start ADC sequential scan conversion;
- Step13. Wait for ADC_IFR.EOC to become 1, read the ADC_RESULT register to get the conversion result of the corresponding channel. Wait for ADC_IFR.EOS to become 1 to end the sequential scan conversion;

Step14. If you need to convert other channels, repeat Step9~Step13;

Step15. Set ADC_CR0.EN and ADC_BGR.BGR_EN to 0, turn off the ADC module and BGR module.

Start the ADC sequential scan conversion operation process through an external trigger:

Step1. Configure the corresponding bit of GPIOx_ADS to configure the ADC channel to be converted as an analog port;

Step2. Set GPIOB_ADS.bit1 to 1, and configure the ADC external reference voltage pin(PB01) as an analog port;

Note: If the ADC reference voltage does not select an external reference voltage pin, this step can be skipped.

Step3. Set ADC_BGR.BGR_EN to 1 to enable the BGR module;

Step4. Set ADC_CR0.EN to 1 to enable the ADC module;

Step5. Delay 20us, wait for the ADC and BGR module to start up;

Step6. Set ADC_CR1.MODE to 1 to select the scan conversion mode;

Step7. Set ADC_CR0.IE to 1 to enable ADC interrupt;

Step8. Enable the ADC interrupt in the NVIC interrupt vector table;

Step9. Configure ADC_CR0.REF to select the reference voltage of ADC;

Step10. Configure ADC_CR0.SAM and ADC_CR0.CKDIV to set the conversion speed of ADC;

Step11. Configure ADC_SQRx.CHxMUX to select sequential scan conversion channels;

Step12. Configure ADC_SQR2.CNT to select the total number of conversions for sequential scan conversions;

Note: The total number of conversions needs to be consistent with the number of conversion channels configured in Step11.

Step13. Set ADC_IFR to 0x0, clear the ADC interrupt flag;

Step14. Configure ADC_EXTTRIGGER0 to select the external trigger condition;

Step15. When the external trigger condition triggers the ADC to complete the conversion, the ADC module will generate an EOC interrupt. Users can read the ADC_RESULT register in the ADC interrupt service routine to get the conversion result of the corresponding channel. When the EOS is interrupted, the current sequence conversion ends;

Step16. If you need to convert other channels, repeat Step12~Step15;

Step17. Set ADC_CR0.EN and ADC_BGR.BGR_EN to 0, turn off the ADC module and BGR module.

19.5.2 Sequential Scan Continuous Conversion Mode

The sequential scan continuous conversion mode can continuously convert the ADC, and the number of single conversions is configured by ADC_SQR2.CNT; it can be configured to select any channel for conversion, and the channel to be converted is configured by ADC_SQRx.CHxMUX. This mode can be started either by setting the ADC_SQRSTART.START bit or by setting the external trigger of ADC_EXTTRIGGER0. After starting the conversion, the ADC module converts the channel conversions configured in CH0MUX~CHxMUX in turn. After a channel conversion is completed, the ADC_IFR.EOC bit will be automatically set to 1, and the conversion result is saved in the ADC_RESULT register, and then switch to the next channel. The ADC module completes ADC_SQR2 After the number of conversions configured in .CNT, the ADC_IFR.EOS bit is automatically set to 1. And automatically start the next conversion until the software writes 0 to ADC_SQRSTART.START, and stops after converting the channel currently being converted.

19.5.3 Sequential scan discontinuous conversion mode

In the sequential scan discontinuous conversion mode, the channels configured for ADC_SQR2.CNT and ADC_SQRx.CHxMUX can be converted sequentially. This mode can be started either by setting the ADC_SQRSTART.START bit or by setting the external trigger of ADC_EXTTRIGGER0. Convert one channel each time, and then wait for the next channel to be converted, the ADC_IFR.EOC bit will be automatically set to 1, and the conversion result will be saved in the ADC_RESULT register.

19.5.4 Scan conversion triggers DMA read

After the sequential scan conversion is completed, the DMA can be automatically triggered to read the conversion result. In sequential scan mode, enable this function by configuring ADC_IER. DEOC to 1.

19.6 Continuous Conversion Accumulation Mode

In the continuous single-conversion accumulation mode, starting an ADC can perform multiple conversions on multiple channels and accumulate the results of each conversion; all 19 channels can be configured for conversion. The total number of ADC conversions is configured by ADC_SQR2.CNT; the channel to be converted is configured by ADC_SQRx.CHxMUX. This mode can be started either by setting the ADC_SQRSTART.START bit or by setting the external trigger of ADC_EXTTRIGGER0. After starting the continuous conversion, the ADC module converts the channels configured in CHxMUX~CH0MUX sequentially until the total conversion times are completed. After the ADC module completes the total number of conversions, the ADC_IFR.EOS bit is automatically set to 1, and the accumulated value of the conversion result is stored in the ADC_RESULTACC register.

The figure below demonstrates the process of accumulating 10 consecutive conversions of AIN0, AIN1, and AIN5. Sequential scan conversion channels 9, 6, 3, 0 are configured as AIN0, conversion channels 8, 5, 2 are configured as AIN5, and conversion channels 7, 4, 1 are configured as AIN1. After ADC_SQRSTART.START is set to 1, the ADC module will convert sequentially according to the sequential scan conversion channel configuration until the count value of SQR_CNT becomes 0. The ADC_RESULTACC register is automatically incremented each time a conversion is complete. The conversion results of AIN0, AIN1, and AIN5 given in the figure are 0x010, 0x020, and 0x040 in turn.

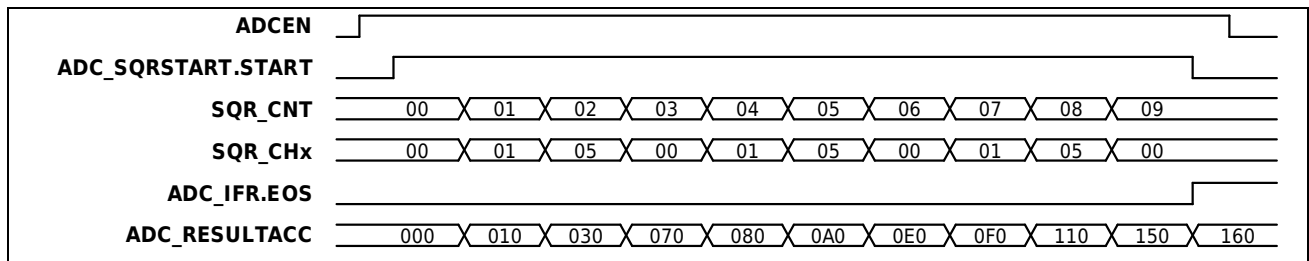


Figure 19-4 ADC Continuous Conversion Accumulation Process Example

Start the ADC continuous conversion and accumulation operation process through the ADC_SQRSTART.START bit:

- Step1. Configure the corresponding bit of GPIOx_ADS to configure the ADC channel to be converted as an analog port;
- Step2. Set GPIOB_ADS.bit1 to 1, and configure the ADC external reference voltage pin(PB01) as an analog port;

Note: If the ADC reference voltage does not select an external reference voltage pin, this step can be skipped.

- Step3. Set ADC_BGR.BGR_EN to 1 to enable the BGR module;
- Step4. Set ADC_CR0.EN to 1 to enable the ADC module;
- Step5. Delay 20us, wait for the ADC and BGR module to start up;
- Step6. Set ADC_CR1.MODE to 1 to select the scan conversion mode;
- Step7. Set ADC_CR1.RACCEN to 1 to enable the automatic accumulation function of ADC conversion;
- Step8. Configure ADC_CR0.REF to select the reference voltage of ADC;
- Step9. Configure ADC_CR0.SAM and ADC_CR0.CKDIV to set the conversion speed of ADC;
- Step10. Configure ADC_SQRx.CHxMUX to select sequential scan conversion channels;
- Step11. Configure ADC_SQR2.CNT to select the total number of conversions for sequential scan conversions;

Note: The total number of conversions needs to be consistent with the number of conversion channels configured in Step10.

Step12. Set ADC_ICR.EOS to 0, clear the ADC_IFR.EOS flag;

Step13. Set ADC_CR1.RACCCLR to 0, clear the ADC_RESULTACC register;

Step14. Set ADC_SQRSTART.START to 1 to start ADC sequential scan conversion;

Step15. Wait for ADC_IFR.EOS to become 1, read the ADC_RESULTACC register to obtain the accumulated value of the conversion result;

Step16. If you need to convert other channels, repeat Step11~Step15;

Step17. Set ADC_CR0.EN and ADC_BGR.BGR_EN to 0, turn off the ADC module and BGR module.

Start ADC continuous conversion and accumulation operation process through external trigger:

Step1. Configure the corresponding bit of GPIOx_ADS to configure the ADC channel to be converted as an analog port;

Step2. Set GPIOBADs.bit1 to 1, and configure the ADC external reference voltage pin(PB01) as an analog port;

Note: If the ADC reference voltage does not select an external reference voltage pin, this step can be skipped.

Step3. Set ADC_BGR.BGR_EN to 1 to enable the BGR module;

Step4. Set ADC_CR0.EN to 1 to enable the ADC module;

Step5. Delay 20us, wait for the ADC and BGR module to start up;

Step6. Set ADC_CR1.MODE to 1, select the scan single conversion mode;

Step7. Set ADC_IER.EOS to 1 to enable ADC interrupt;

Step8. Enable the ADC interrupt in the NVIC interrupt vector table;

Step9. Set ADC_CR1.RACCEN to 1 to enable the automatic accumulation function of ADC conversion;

Step10. Configure ADC_CR0.REF to select the reference voltage of ADC;

Step11. Configure ADC_CR0.SAM and ADC_CR0.CKDIV to set the conversion speed of ADC;

Step12. Configure ADC_SQRx.CHxMUX to select sequential scan conversion channels;

Step13. Configure ADC_SQR2.CNT to select the total number of conversions for sequential scan conversions;

Note: The total number of conversions needs to be consistent with the number of conversion channels configured in Step12.

Step14. Set ADC_IFR to 0x0, clear the ADC interrupt flag;

Step15. Set ADC_CR1.RACCCLR to 0, clear the ADC_RESULTACC register;

Step16. Configure ADC_EXTTRIGGER0 to select the external trigger condition;

Step17. When an external trigger condition triggers the ADC to complete the conversion, the ADC module will generate an interrupt. The user can read the ADC_RESULTACC register in the ADC interrupt service program to obtain the accumulated value of the conversion result;

Step18. If you need to convert other channels, repeat Step12~Step17;

Step19. Set ADC_CR0.EN and ADC_BGR.BGR_EN to 0, turn off the ADC module and BGR module.

19.7 ADC conversion external trigger source

ADC conversions can be initiated either by software configuration or by an external trigger.

Configure the ADC_EXTTRIGGER0 register to set the external trigger source for ADC single conversion or sequential scan conversion.

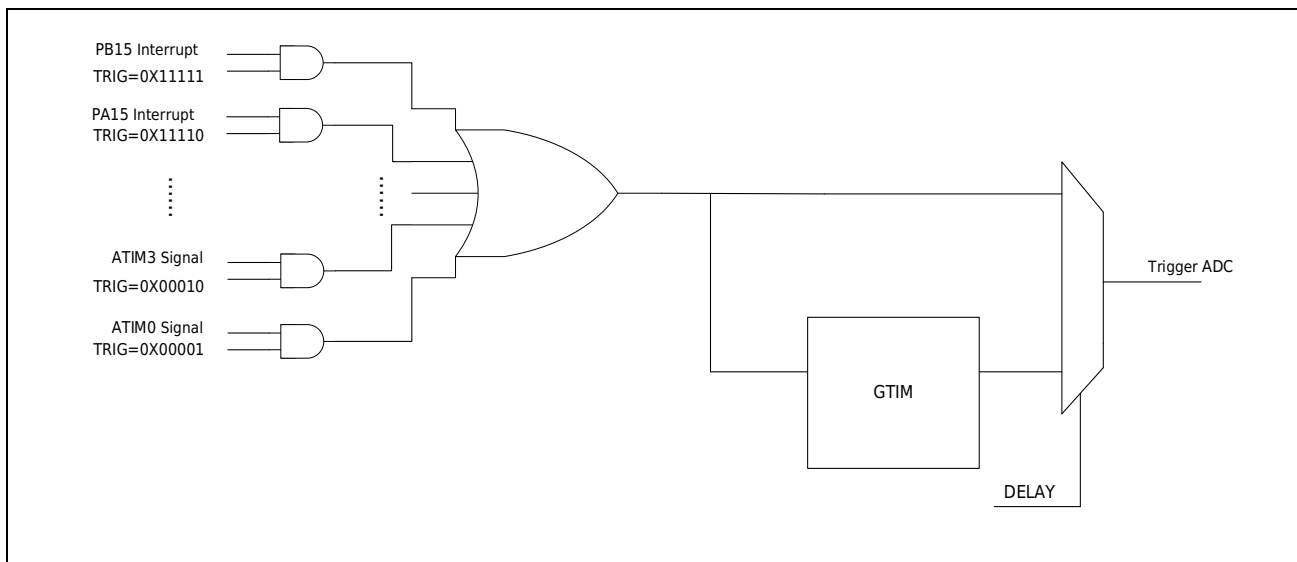


Figure 19-5 Schematic diagram of ADC single conversion or sequential scan conversion external trigger source

19.8 ADC conversion result comparison

When the ADC conversion is completed, the ADC conversion result can be compared with the threshold set by the user, supporting upper threshold comparison, lower threshold comparison, and interval value comparison. This function needs to set the corresponding control bits HTCMP, LTCMP, REGCMP to 1. This function can realize the automatic monitoring of the analog quantity, until the ADC conversion result meets the user's expectations, an interrupt is generated to apply for user program interface entry. The monitoring channel selection is configured through ADC_CR1.THCH.

Upper threshold comparison: When the ADC conversion result is within the range [ADC_HT, 4095], ADC_IFR.HTI is set to 1; writing 0 to ADC_ICR.HTI clears ADC_IFR.HTI.

Lower threshold comparison: when the ADC conversion result is within the range [0, ADC_LT), ADC_IFR.LTI is set to 1; writing 0 to ADC_ICR.LTI clears ADC_IFR.LTI.

Interval value comparison: When the ADC conversion result is within the [ADC_LT, ADC_HT) interval, ADC_IFR.RGI is set to 1; writing 0 to ADC_ICR.RGI clears ADC_IFR.RGI.

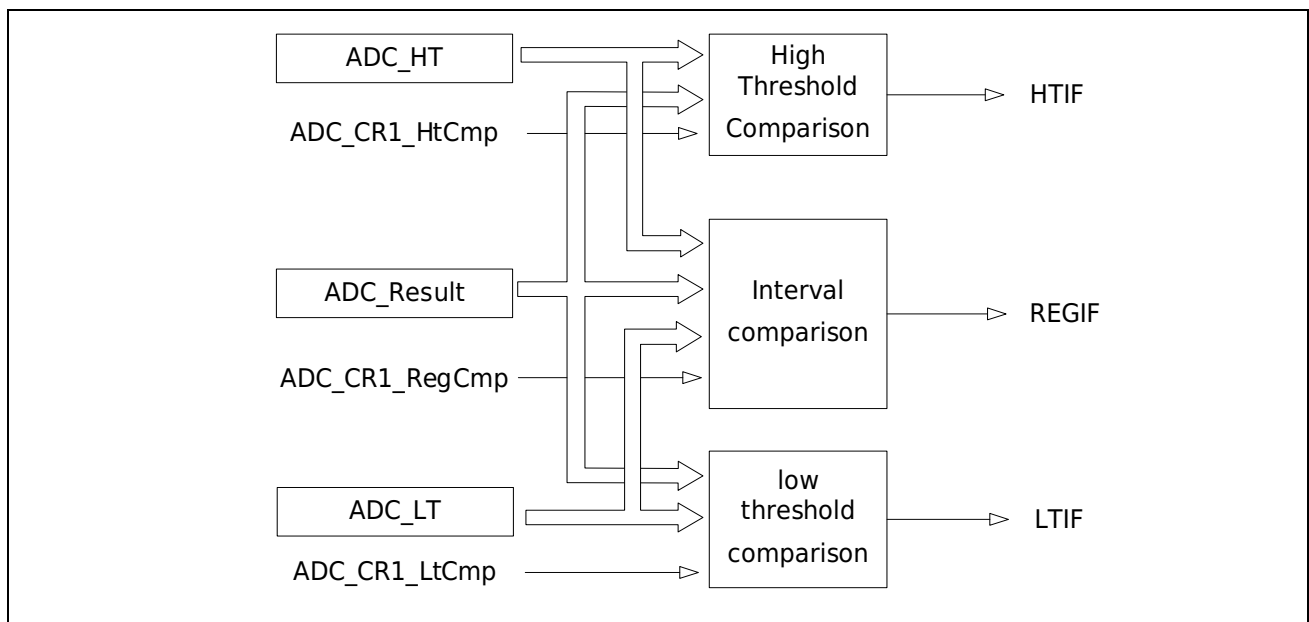


Figure 19-6 ADC conversion result comparison

19.9 ADC interrupt

The ADC interrupt request is shown in the table below:

Table 19-2 ADC interrupt

Interrupt source	Interrupt logo	Interrupt enable
ADC conversion result overflow	ADC_IFR.OV	ADC_IER.OV
ADC scan single channel conversion completed	ADC_IFR.EOC	ADC_IER.EOC
ADC sequential scan conversion complete	ADC_IFR.EOS	ADC_IER.EOS

Interrupt source	Interrupt logo	Interrupt enable
The ADC conversion result is in the interval value area (greater than or equal to the lower threshold and less than the upper threshold)	ADC_IFR.RGI	ADC_IER. RGI
The ADC conversion result is in the upper threshold region (greater than or equal to the upper threshold)	ADC_IFR.HTI	ADC_IER.HTI
ADC conversion result comparison lower threshold area (less than lower threshold)	ADC_IFR.LTI	ADC_IER.LTI
ADC single conversion completed	ADC_IFR.ESG	ADC_IER.ESG

19.10 Measure ambient temperature using a temperature sensor

The output voltage of the temperature sensor will change with the change of the ambient temperature, so the corresponding ambient temperature can be calculated according to the output voltage of the temperature sensor. When the measurement channel of the ADC module selects the output voltage of the temperature sensor, the ambient temperature can be measured.

Calculated as follows:

$$\text{Ambient Temperature} = \text{TS_ENV} / 2 + 0.09354 \times \text{Vref} \times (\text{AdcValue} - \text{Trim})$$

Of which:

TS_ENV: The ambient temperature corresponding to the calibration value *2. It needs to be read from the Flash memory during calculation, and its storage address is 0x0100 1234 to 0x0100 1237.

Vref is the reference voltage of the current ADC module, and the value is 1.5 or 2.5.

AdcValue is the result of measuring the output voltage of the temperature sensor by the ADC module, and the value is 0~4095.

Trim is a calibration value of 32Bits, which needs to be read from the Flash memory during calculation, and its storage address is detailed in the table below.

ADC reference voltage	Calibration value storage address
Internal 1.5V	0x0100 122C ~ 0x0100 122F
Internal 2.5V	0x0100 1230 ~ 0x0100 1233

An example calculation is as follows:

Condition 1: Vref=2.5, AdcValue=0x7E5, Trim=0x76C, TS_ENV=0x32:

Temperature 1: $0x32/2 + 0.09354 \times 2.5 \times (0x7E5 - 0x76C) = 53.30^{\circ}\text{C}$.

Condition 2: Vref=1.5, AdcValue=0x72D, Trim=0x76C, TS_ENV=0x32:

Temperature 2: $0x32/2 + 0.09354 \times 1.5 \times (0x72D - 0x76C) = 16.16^{\circ}\text{C}$.

Operation process of measuring ambient temperature through ADC:

- Step1. Set ADC_BGR to 3, enable BGR module and temperature sensor module;
- Step2. Set ADC_CR0.EN to 1 to enable the ADC module;
- Step3. Delay 20 μ s, wait for ADC, BGR module and temperature sensor module to start up;
- Step4. Set ADC_CR1.MODE to 0, select single conversion mode;
- Step5. Configure ADC_CR0.REF, select the reference voltage of ADC as internal 1.5V or internal 2.5V;
- Step6. Configure ADC_CR0.SAM and ADC_CR0.CKDIV to set the conversion speed of ADC;
- Step7. Set ADC_CR0.SGLMUX to 0x11, select the channel to be converted as the output of the temperature sensor;
- Step8. Set ADC_CR0.BUF to 1 to enable the input signal amplifier;
- Step9. Set ADC_ICR.ESG to 0, clear the ADC_IFR.ESG flag;
- Step10. Set ADC_SGLSTART.START to 1 to start ADC single conversion;
- Step11. Wait for ADC_IFR.ESG to become 1, read the ADC_RESULT register to get the ADC conversion result;
- Step12. Set ADC_CR0.EN and ADC_BGR to 0, turn off the ADC module, BGR module, and temperature sensor module;
- Step13. Read the calibration value of the temperature sensor and calculate the current ambient temperature according to the formula.

19.11 Registerdescription

19.11.1 Register list

Table 19-3 ADC base address list

Name	base address	Description
ADC0	0x4000 A000	ADC0 base address
ADC1	0x4000 A400	ADC1 base address

Table 19-4 ADC register list

Register name	Offset address	Reset value
ADC Analog Device Select Register(ADC_BGR)	0x000	0x0000 0000
ADC Basic Configuration Register 0 (ADC_CR0)	0x004	0x0000 27F0
ADC Basic Configuration Register 1 (ADC_CR1)	0x008	0x0000 8000
ADC interrupt and DMA trigger enable(ADC_IER)	0x030	0x0000 0000
ADC Sequential Scan Conversion Channel Configuration Register 0 (ADC_SQR0)	0x040	0x0000 0000
ADC Sequential Scan Conversion Channel Configuration Register 1 (ADC_SQR1)	0x044	0x0000 0000
ADC Sequential Scan Conversion Channel Configuration Register 2 (ADC_SQR2)	0x048	0x0000 0000
ADC conversion result (ADC_RESULT)	0x0A0	0x0000 0000
ADC conversion result accumulation value (ADC_RESULTACC)	0x0A4	0x0000 0000
ADC Compare Upper Threshold (ADC_HT)	0x0A8	0x0000 0FFF
ADC Compare Lower Threshold (ADC_LT)	0x0AC	0x0000 0000
ADC Interrupt Flag Register (ADC_IFR)	0x0B0	0x0000 0000
ADC Interrupt Clear Register (ADC_ICR)	0x0B4	0x0000 03FF
ADC External Interrupt Trigger Source Configuration Register 0(ADC_EXTTRIG0)	0x0B8	0x0000 0000
ADC Single Conversion Start Control Register(ADC_SGLSTART)	0x0C0	0x0000 0000
ADC Sequential Scan Conversion Start Control Register(ADC_SQRSTART)	0x0C4	0x0000 0000

19.11.2 ADC Analog Device Select Register (ADC_BGR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res														TS_EN	BGR_EN
Bit	Marking	Place name	Function										Read and write		
b31~b2	Res	Reserved bit	Reserved, must remain at reset value												
b1	TS_EN	TS module enable	1: TS module is enabled 0: TS module off										RW		
b0	BGR_EN	BGR module enable	1: BGR module enable 0: BGR module off										RW		

Note: This register only exists in ADC0, not in ADC1.

19.11.3 ADC Basic Configuration Register 0(ADC_CR0)

Reset value: 0x0000 27F0

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res		SAM		BUF	REF		SGLMUX					CKDIV		EN	

Bit	Marking	Place name	Function	Read and write
b31~b14	Res	Reserved bit	Reserved, must remain at reset value	
b13~b12	SAM	ADC sampling period selection	00: 3 sampling periods 01: 6 sampling periods 10: 9 sampling periods 11: 10 sampling periods, Note: The ADC conversion period is fixed at 15	RW
b11	BUF	Built-in follower control	0: Disable the follower, and the external input signal is directly connected to the ADC. 1: Turn on the follower, and the external input signal is connected to the ADC after passing through the follower to measure high-impedance signals. Note 1: When this function is enabled, the maximum rate is 200Ksps Note 2: The following conditions must enable this function The output impedance of the signal to be converted is high The signal to be converted is: AVCC/3, built-in temperature sensor output voltage, DAC output	RW
b10~b9	REF	ADC reference voltage selection (only registers in ADC0 are valid, shared by ADC0/1)	00: Internal 1.5V 01: Internal 2.5V 10: External reference voltage ExRef (PB01) 11: AVCC voltage	RW
b8~b4	SGLMUX	Single Conversion Mode Selection	00000: AIN0 (PA00) 00001: AIN1 (PA01) 00010: AIN2 (PA02) 00011: AIN3 (PA03) 00100: AIN4 (PA04) 00101: AIN5 (PA05) 00110: AIN6 (PA06) 00111: AIN7 (PA07) 01000: AIN8 (PB00) 01001: AIN9 (PB01) 01010: AIN10 (PC00) 01011: AIN11 (PC01) 01100: AIN12 (PC02) 01101: AIN13 (PC03) 01110: AIN14 (PC04) 01111: AIN15 (PC05) 10000: AVCC/3 Note: ADC_CR0.Buf must be 1 10001: Built-in temperature sensor output voltage Note: ADC_CR0.Buf must be 1 10010: Reserved	RW
b3~b1	CKDIV	ADC clock selection	000: PCLK clock 001: PCLK clock divided by 4 010: PCLK clock divided by 6 011: PCLK clock divided by 8 100: PCLK clock divided by 10 101: PCLK clock divided by 12 110: PCLK clock divided by 14 111: PCLK clock divided by 16	RW

b0	EN	ADC enable control	1: enable ADC 0: disable ADC	RW
----	----	--------------------	---------------------------------	----

19.11.4 ADC Basic Configuration Register 1 (ADC_CR1)

Reset value: 0x0000 8000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
RACCCLR	REGCMP	HTCMP	LTCMP	RACCEN	OVMD	Res		THCH				ALIGN		MODE	

Bit	Marking	Place name	Function	Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value	
b15	RACCCLR	ADC conversion result accumulation register is cleared	1: no effect; 0: ADC conversion result accumulation register (ADC_ResultAcc) is cleared.	R1W0
b14	REGCMP	ADC interval comparison control	1: enable range comparison 0: interval comparison disabled	RW
b13	HTCMP	ADC high threshold compare control	1: enable high threshold compare 0: disable high threshold comparison	RW
b12	LTCMP	ADC low threshold compare control	1: enable low threshold compare 0: disable low threshold comparison	RW
b11	RACCEN	ADC conversion result automatic accumulation control	1: enable the automatic accumulation function of ADC conversion results 0: disable the automatic accumulation function of ADC conversion results	RW
b10	OVMD	the ADC is over-converted, the Adc_Result storage method	0: Keep the last data and discard the new data 1: Overwrite the last data and keep the new data	RW
b9~b8	Res	Reserved bit	Reserved, must remain at reset value	
b7~b3	THCH	Threshold comparison channel selection	0000: select channel 0 for threshold comparison 00001: Select channel 1 for threshold comparison 00010: Select channel 2 for threshold comparison 10010: Select channel 18 for threshold comparison	RW
b2	ALIGN	Conversion result alignment control	1: The conversion result is left-aligned and stored in 16Bits 0: conversion result 16Bits right-aligned storage	RW
b1~b0	MODE	Conversion mode	00 single 01 sequential scan single sequence 10 Sequential Scan Consecutive Sequences 11 sequential scan intermittent sequence	RW

19.11.5 ADC Sequential Scan Conversion Channel Configuration Register 0 (ADC_SQR0)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res		CH5MUX					CH4MUX					CH3MUX[4:1]			
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CH3MUX[0]		CH2MUX					CH1MUX					CH0MUX			

Bit	Marking	Place name	Function	Read and write
b31~b30	Res	Reserved bit	Reserved, must remain at reset value	
b29~b25	CH5MUX	Sequential scan conversion channel 5 selection	ADC_CR0.SGLMUX for setting	RW
b24~b20	CH4MUX	Sequential scan conversion channel 4 selection	ADC_CR0.SGLMUX for setting	RW
b19~b15	CH3MUX	Sequential scan conversion channel 3 selection	ADC_CR0.SGLMUX for setting	RW
b14~b10	CH2MUX	Sequential scan conversion channel 2 selection	ADC_CR0.SGLMUX for setting	RW
b9~b5	CH1MUX	Sequential scan conversion channel 1 selection	ADC_CR0.SGLMUX for setting	RW
b4~b0	CH0MUX	Sequential scan conversion channel 0 selection	ADC_CR0.SGLMUX for setting	RW

19.11.6 ADC Sequential Scan Conversion Channel Configuration Register 1 (ADC_SQR1)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res		CH11MUX					CH10MUX					CH9MUX[4:1]			
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CH9MUX[0]		CH8MUX					CH7MUX					CH6MUX			

Bit	Marking	Place name	Function	Read and write
b31~b30	Res	Reserved bit	Reserved, must remain at reset value	
b29~b25	CH11MUX	Sequential scan conversion channel 11 selection	ADC_CR0.SGLMUX for setting	RW
b24~b20	CH10MUX	Sequential scan conversion channel 10 selection	ADC_CR0.SGLMUX for setting	RW
b19~b15	CH9MUX	Sequential scan conversion channel 9 selection	ADC_CR0.SGLMUX for setting	RW
b14~b10	CH8MUX	Sequential scan conversion channel 8 selection	ADC_CR0.SGLMUX for setting	RW
b9~b5	CH7MUX	Sequential scan conversion channel 7 selection	ADC_CR0.SGLMUX for setting	RW
b4~b0	CH6MUX	Sequential scan conversion channel 6 selection	ADC_CR0.SGLMUX for setting	RW

19.11.7 ADC Sequential Scan Conversion Channel Configuration Register 2 (ADC_SQR2)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res								CNT				CH15MUX[4:1]			
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CH15MUX[0]	CH14MUX					CH13MUX					CH12MUX				

Bit	Marking	Place name	Function	Read and write
b31~b24	Res	Reserved bit	Reserved, must remain at reset value	
b23~b20	CNT	Sequential Scan Conversions		RW
b19~b15	CH15MUX	Sequential scan conversion channel 15 selection	ADC_CR0.SGLMUX for setting	RW
b14~b10	CH14MUX	Sequential scan conversion channel 14 selection	ADC_CR0.SGLMUX for setting	RW
b9~b5	CH13MUX	Sequential scan conversion channel 13 selection	ADC_CR0.SGLMUX for setting	RW
b4~b0	CH12MUX	Sequential scan conversion channel 12 selection	ADC_CR0.SGLMUX for setting	RW

19.11.8 ADC conversion result (ADC_RESULT)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
RESULT															

Bit	Marking	Place name	Function	Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value	RO
b15~b0	RESULT	ADC conversion results		RO

19.11.9 ADC conversion result accumulation value (ADC_RESULTACC)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res												RESULTACC[19:16]			
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
RESULTACC[15:0]															
Bit	Marking	Place name	Function										Read and write		
b31~b20	Res	Reserved bit	Reserved, must remain at reset value												
b19~b0	RESULTACC	ADC conversion result accumulated value											RO		

19.11.10 ADC Compare Upper Threshold (ADC_HT)

Reset value: 0x0000 0FFF

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res				HT											
Bit	Marking	Place name	Function										Read and write		
b31~b12	Res	Reserved bit	Reserved, must remain at reset value												
b11~b0	HT	ADC conversion result comparison upper threshold											RW		

19.11.11 ADC Compare Lower Threshold (ADC_LT)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res				LT											
Bit	Marking	Place name	Function										Read and write		
b31~b12	Res	Reserved bit	Reserved, must remain at reset value												
b11~b0	LT	ADC conversion result comparison lower threshold											RW		

19.11.12 ADC Interrupt Enable (ADC_IER)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res						OV	Res	DEOC	EOC	Res	EOS	RGI	HTI	LTI	ESG

Bit	Marking	Place name	Function	Read and write
b31~b10	Res	Reserved bit	Reserved, must remain at reset value	
b9	OV	ADC conversion result overflow interrupt enable	0: disable 1: enable	RW
b8	Res	Reserved bit	Reserved, must remain at reset value	
b7	DEOC	Sequential conversion completion triggers DMA	0: disable 1: enable	RW
b6	EOC	Sequential conversion complete interrupt enable	0: disable 1: enable	RW
b5	Res	Reserved bit	Reserved, must remain at reset value	
b4	EOS	Sequential Conversion Complete Interrupt Enable	0: disable 1: enable	RW
b3	RGI	Range comparison interrupt enable	0: disable 1: enable	RW
b2	HTI	High Threshold Compare Interrupt Enable	0: disable 1: enable	RW
b1	LTI	Low Threshold Compare Interrupt Enable	0: disable 1: enable	RW
b0	ESG	One-shot conversion interrupt enable	0: disable 1: enable	RW

19.11.13 ADC Interrupt Flag Register (ADC_IFR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res						OV	Res		EOC	Res	EOS	RGI	HTI	LTI	ESG

Bit	Marking	Place name	Function	Read and write
b31~b10	Res	Reserved bit	Reserved, must remain at reset value	
b9	OV	ADC conversion result overflow flag		RO
b8~b7	Res	Reserved bit	Reserved, must remain at reset value	
b6	EOC	ADC sequential scan conversion single completion flag	1: ADC sequential scan single conversion completed 0: ADC sequential scan single conversion not completed	RO
b5	Res	Reserved bit	Reserved, must remain at reset value	
b4	EOS	ADC sequential scan conversion complete flag	1: ADC sequential scan conversion complete 0: ADC sequential scan conversion not completed	RO
b3	RGI	ADC conversion result comparison interval flag	1: The ADC conversion result is within the range of [ADC_LT, ADC_HT) 0: The ADC conversion result is outside the [ADC_LT, ADC_HT) range	RO
b2	HTI	ADC conversion result comparison upper threshold flag	1: The ADC conversion result is within the range [ADC_HT, 4095] 0: ADC conversion result is outside [ADC_HT, 4095] range	RO
b1	LTI	ADC conversion result comparison lower threshold flag	1: The ADC conversion result is in the range [0, ADC_LT) 0: The ADC conversion result is outside the interval [0, ADC_LT)	RO
b0	ESG	ADC single conversion complete flag	1: ADC single conversion completed 0: ADC single conversion not completed	RO

19.11.14 ADC Interrupt Clear Register (ADC_ICR)

Reset value: 0x0000 03FF

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res						OV	Res		EOC	Res	EOS	RGI	HTI	LTI	ESG

Bit	Marking	Place name	Function	Read and write
b31~b10	Res	Reserved bit	Reserved, must remain at reset value	
b9	OV	ADC conversion result overflow flag clear bit	Write 0 to clear data overwrite flag clear Write 1 has no effect	R1W0
b8~b7	Res	Reserved bit	Reserved, must remain at reset value	
b6	EOC	ADC sequential scan conversion single completion flag clear bit	Write 0 to clear the ADC sequential scan conversion single completion flag Write 1 has no effect	R1W0
b5	Res	Reserved bit	Reserved, must remain at reset value	
b4	EOS	ADC sequential scan conversion complete flag clear bit	Write 0 to clear the ADC sequential scan conversion complete flag Write 1 has no effect	R1W0
b3	RGI	ADC conversion result comparison interval flag clear bit	Write 0 to clear the ADC conversion result comparison interval flag Write 1 has no effect	R1W0
b2	HTI	ADC conversion result comparison upper threshold flag clear bit	Write 0 to clear ADC conversion result comparison upper threshold Write 1 has no effect	R1W0
b1	LTI	ADC conversion result comparison lower threshold flag clear bit	Write 0 to clear ADC conversion result comparison lower threshold flag Write 1 has no effect	R1W0
b0	ESG	ADC single conversion complete flag clear bit	Write 0 to clear the ADC single conversion complete flag Write 1 has no effect	R1W0

19.11.15 ADC External Interrupt Trigger Source Configuration Register 0 (ADC_EXTTRIGGER0)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res										DELAY	TRIG				

Bit	Marking	Place name	Function	Read and write
b31~b6	Res	Reserved bit	Reserved, must remain at reset value	
b5	DELAY	ADC single conversion or sequential scan conversion external interrupt trigger delay control	1: The trigger signal selected by Trig triggers the GTIM, and the GTIM signal triggers the ADC (ADC0 triggers GTIM0. ADC1 triggers GTIM1) 0: directly use the TRIG selection signal to trigger the ADC	RW
b4~b0	TRIG	ADC single conversion or sequential scan conversion external interrupt trigger signal selection	11111 PB15 interrupt triggers ADC conversion 11110 PA15 interrupt triggers ADC conversion 11101 PB13 interrupt triggers ADC conversion 11100 PA13 interrupt triggers ADC conversion (note: PA13 is swd by default and needs to be multiplexed as GPIO) 11011 PB11 interrupt triggers ADC conversion 11010 PA11 interrupt triggers ADC conversion 11001 PB09 interrupt triggers ADC conversion 11000 PA09 interrupt triggers ADC conversion 10111 PB07 interrupt triggers ADC conversion 10110 PA07 interrupt triggers ADC conversion 10101 PB05 interrupt triggers ADC conversion 10100 PA05 interrupt triggers ADC conversion 10011 PB03 interrupt triggers ADC conversion 10010 PA03 interrupt triggers ADC conversion 10001 PB01 interrupt triggers ADC conversion 10000 PA01 interrupt triggers ADC conversion 01111 LVD interrupt triggers ADC conversion 01110 VC1 interrupt triggers ADC conversion 01101 VC0 interrupt triggers ADC conversion 01100 DMA interrupt triggers ADC conversion 01011 SPI1 interrupt triggers ADC conversion 01010 SPI0 interrupt triggers ADC conversion 01001 USART3 interrupt triggers ADC conversion 01000 USART2 interrupt triggers ADC conversion 00111 USART1 interrupt triggers ADC conversion 00110 USART0 interrupt triggers ADC conversion 00101 GTIM2 signal triggers ADC conversion 00100 GTIM1 signal triggers ADC conversion 00011 GTIM0 signal triggers ADC conversion 00010 ATIM3 signal triggers ADC conversion 00001 ATIM0 signal triggers ADC conversion 00000 No peripheral trigger	RW

Note:

1. The timer trigger ADC has a dedicated control signal, which is controlled inside the timer.
2. The ADC is triggered using the rising edge of each interrupt flag bit. If repeated triggering is required, the interrupt flag needs to be cleared. If you do not need to enter the interrupt service routine, please do not enable the interrupt enable of the NVIC.

19.11.16 ADC Single Conversion Start Control Register (ADC_SGLSTART)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res														START	
Bit	Marking	Place name	Function											Read and write	
b31~b1	Res	Reserved bit	Reserved, must remain at reset value												
b0	START	ADC single conversion start control	CR0.EN=1 can it be read back correctly 1: start ADC single conversion 0: stop ADC single conversion											RW	

19.11.17 ADC Sequential Scan Conversion Start Control Register (ADC_SQRSTART)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res														START	
Bit	Marking	Place name	Function											Read and write	
b31~b1	Res	Reserved bit	Reserved, must remain at reset value												
b0	START	ADC sequential scan conversion start control	CR0.EN=1 can it be read back correctly 1: start ADC sequential scan conversion 0: stop ADC sequential scan conversion											RW	

20 Analog Voltage Comparator (VC)

20.1 Introduction

The analog voltage comparator VC is used to compare the size of two input analog voltages, and output high / low level according to the comparison result.

- When the " + " input voltage is higher than the " - " input voltage, the voltage comparator output is high
- When the " + " input voltage is lower than the " - " input voltage, the voltage comparator output is low

20.2 Main features

- Support 64 -order voltage divider reference
- Support 11 external input ports and 4 on-chip input signals
- Supports 2 output polarities
- Support window compare function
- A variety of filter time configurations can effectively enhance the anti-interference ability of the chip
- Support 3 interrupt trigger modes, high level / rising edge / falling edge
- The output of the voltage comparator can be used as the input of the timer
- Support working in DeepSleep mode

20.3 Functional description

20.3.1 Functional block diagram

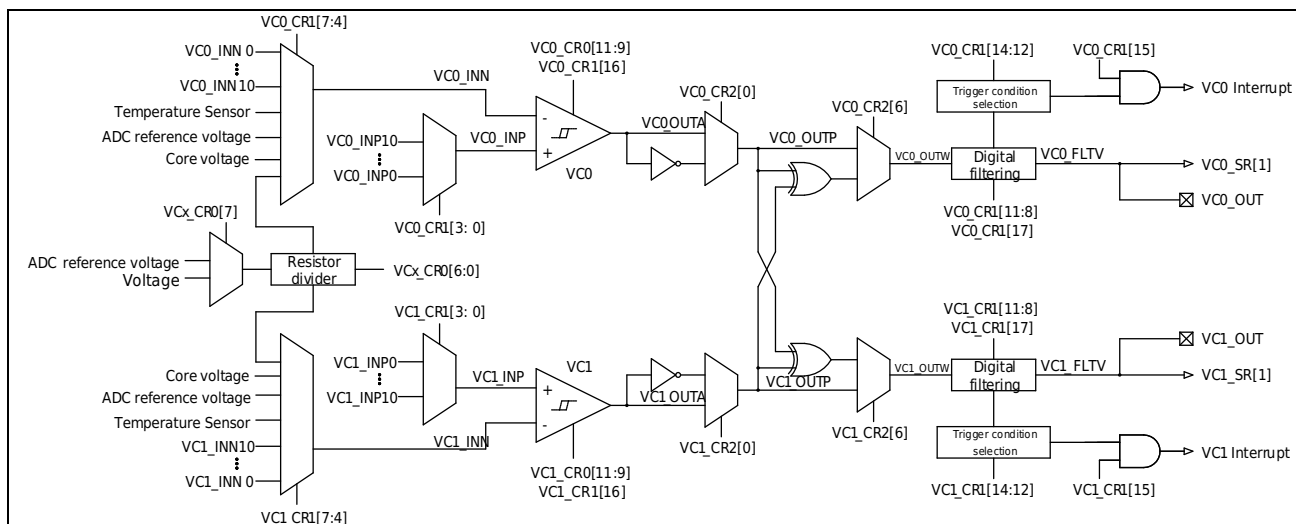


Figure 20-1 VC functional block diagram

20.3.2 Setup / Response Time

When using a voltage comparator, the time from when VC is enabled or when the input voltage at both ends of VC changes to when the correct result is output is determined by the VCx_CR0.BIAS control bit. The larger the current, the faster the VC response. The typical value ranges from 200ns to 20us. adjustable.

If the ADC reference voltage is selected as the comparison reference voltage or CR0.BIAS is 2 or 3, the BGR module needs to be enabled. The startup time of the BGR module is Tstable(BGR), and the voltage comparator needs to wait for the BGR module to stabilize before outputting normally.

20.3.3 Hysteresis function

the VC working diagram after the hysteresis function is enabled is as follows:

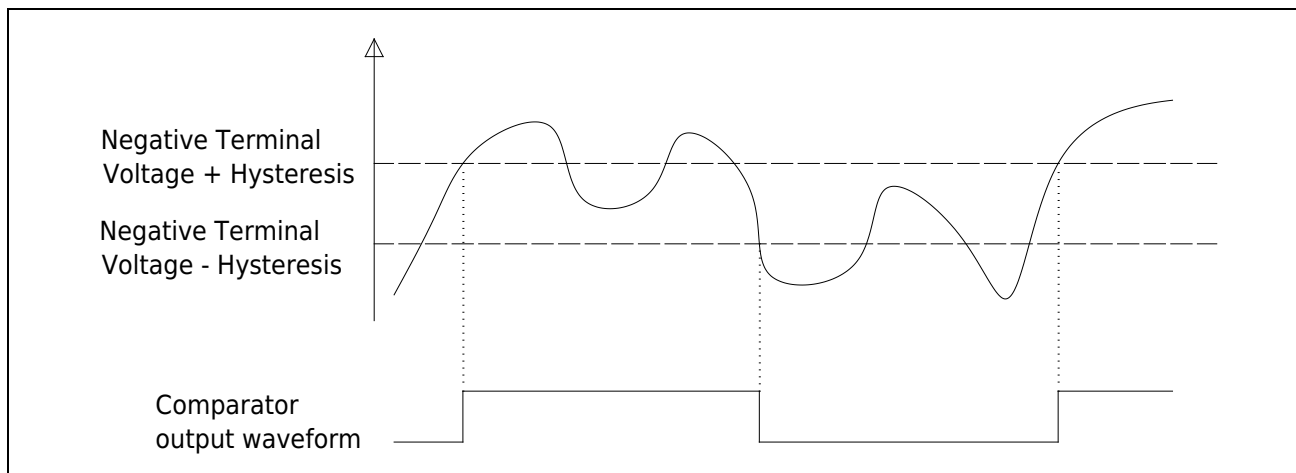


Figure 20-2 VC hysteresis function

20.3.4 Compare Results Polarity Configuration

VCx_CR2.POL is configured as 0, the level of VCx_OUTA and VCx_OUTP is the same.

When VCx_CR2.POL is configured as 1, the level of VCx_OUTA and VCx_OUTP is opposite.

20.3.5 window compare function

VCx_CR2.Window is configured as 0, the VCx_OUTW signal is the same as the VCx_OUTP signal.

VCx_CR2.Window is configured as 1, the VCx_OUTW signal is the XOR value of VC0_OUTP and VC1_OUTP.

20.3.6 Filter description

In addition to the inherent settling/response time of the voltage comparator, the user can set a longer filter time to filter out system noise, such as high current noise when the motor stops.

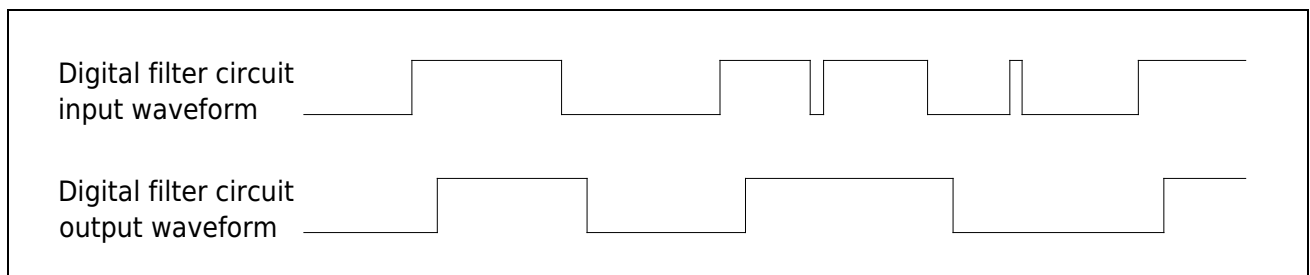


Figure 20-3 VC Filter Response Time

20.4 Register description

20.4.1 register list

Table 20-1 VC base address list

Name	Base address	Description
VC0	0x4000 A800	VC0 base address
VC1	0x4000 A810	VC1 base address

Table 20-2 VC Register List

Register name	Offset address	Reset value
VC Control Register 0 (VC_CR0)	0x00	0x0000 0020
VC Control Register 1 (VC_CR1)	0x04	0x0000 0000
VC Control Register 2 (VC_CR2)	0x08	0x0000 0000
VC Status Register(VC_SR)	0x0C	0x0000 0000

20.4.2 Control Register 0 (VC_CR0)

Reset value: 0x0000 0020

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res				HYS		BIAS		REF	Div En	DIV					

Bit	Marking	Place name	Function	Read and write
b31~b12	Reserved	Reserved bit	Reserved, must keep the default value	
b11~b10	HYS	Hysteresis function configuration	11: The hysteresis voltage is about 30mV 10: The hysteresis voltage is about 20mV 01: The hysteresis voltage is about 10mV 00: no hysteresis	RW
b9~b8	BIAS	power selection	11: High power consumption mode; BGR needs to be turned on, please refer to the data sheet for the BGR startup time 10: Medium power consumption mode; BGR needs to be turned on, please refer to the data sheet for the BGR startup time 01: Medium and low power consumption mode 00: low power consumption mode Note: The greater the power consumption, the faster the response	RW
b7	REF	Selection of Reference Voltage for Resistor Voltage Divider Circuit	1: ADC reference voltage, need to enable ADC, BGR 0: AVCC Note: VC0/VC1 shared	RW
b6	DivEn	Resistor voltage divider circuit enable control	1: Enable 0: Prohibited Note: VC0/VC1 shared	RW
b5~b0	DIV	Resistor divider circuit output voltage configuration	111111: 64/64Vref 111110: 63/64 Vref ... 000001: 2/64 Vref 000000: 1/64 Vref Note: VC0/VC1 shared, the source of Vref is determined by the REF control bit	RW

20.4.3 Control Register 1 (VC_CR1)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16	
Res														FltClk	EN	
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	
IE	Level	Rising	Falling	FltTime			FLTEN	NSEL				PSEL				
Bit	Marking		Place name					Function						Read and write		
b31~b18		Res		Keep					Reserved, must remain at reset value							
b17		FltClk		Filter Clock Selection					0: built-in RC oscillator, frequency about 150KHz 1: PCLK Note: When the filter clock is PCLK, VC cannot wake up the MCU in DeepSleep mode						RW	
b16		EN		enable control					0: disable 1: enable						RW	
b15		IE		Interrupt Enable Control					0: disable 1: enable						RW	
b14		Level		Filter signal high level trigger interrupt enable control					0: disable 1: enable						RW	
b13		Rising		Filter signal rising edge trigger interrupt enable control					0: disable 1: enable						RW	
b12		Falling		Filter signal falling edge trigger interrupt enable control					0: disable 1: enable						RW	
b11~b9		FltTime		Filter time configuration					000: Filter out signals whose width is less than 1 filter clock 001: Filter out signals whose width is less than 3 filter clock 010: Filter out signals whose width is less than 7 filter clock 011: Filter out signals whose width is less than 15 filter clock 100: Filter out signals whose width is less than 63 filter clock 101: Filter out signals whose width is less than 255 filter clock 110: Filter out signals whose width is less than 1023 filter clock 111: Filter out signals whose width is less than 4095 filter clock						RW	
b8		FLTEN		Filter function enable configuration					1: Enable, the filter width is controlled by FltTime 0: disabled, the filter output waveform is the same as the input waveform						RW	
b7~b4		NSEL		Voltage Comparator Negative Input Signal Selection					0000: VCx_INN0 (PA00) 0001: VCx_INN1 (PA01) 0010: VCx_INN2 (PA02) 0011: VCx_INN3 (PA03) 0100: VCx_INN4 (PA04) 0101: VCx_INN5 (PA05) 0110: VCx_INN6 (PB00) 0111: VCx_INN7 (PB01) 1000: VCx_INN8 (PB12) 1001: VCx_INN9 (PB04) 1010: VCx_INN10 (PB05) 1011: Built-in resistor divider output voltage 1100: Built-in temperature sensor output voltage 1101: Reserved 1110: The reference voltage configured by the ADC module (ADC needs to be enabled)						RW	

			1111: core voltage	
			0000: VCx_INP0 (PA00)	
			0001: VCx_INP1 (PA01)	
			0010: VCx_INP2 (PA02)	
			0011: VCx_INP3 (PA03)	
			0100: VCx_INP4 (PA04)	
b3~b0	PSEL	Voltage Comparator Positive Input Signal Selection	0101: VCx_INP5 (PA05)	RW
			0110: VCx_INP6 (PB00)	
			0111: VCx_INP7 (PB01)	
			1000: VCx_INP8 (PB12)	
			1001: VCx_INP9 (PB04)	
			1010: VCx_INP10 (PB05)	
			Other: reserved	

20.4.4 Control Register 2 (VC_CR2)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res									Wind ow	ATIM BK	ATIM 3CLR	Res		ATIM 0CLR	POL
Bit	Marking	Place name							Function				Read and write		
b31~b7	Res	Reserved bit							Reserved, must remain at reset value						
b6	Window	Window comparison function configuration							1: Enable 0: Disable				RW		
b5	ATIMBK	VCx filtered signal output to ATIM0/3 brake enable control							1: Enable 0: Disable				RW		
b4	ATIM3CLR	VCx filtered signal output to ATIM3 REFCLR enable control							1: Enable 0: Disable				RW		
b3~b2	Res	Reserved bit							Reserved, must remain at reset value						
b1	ATIM0CLR	VCx filtered signal output to ATIM0 REFCLR enable control							1: Enable 0: Disable				RW		
b0	POL	VCx_OUTP signal polarity configuration							1: VCx_OUTP is low when the positive terminal is greater than the negative terminal 0: VCx_OUTP is high when the positive terminal is greater than the negative terminal				RW		

20.4.5 Status Register (VC_SR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res														FLT V	INT F
Bit	Marking	Place name		Function		Read and write									
b31~b2	Res	Keep		Reserved, must remain at reset value											
b1	FLTV	Filtered signal level		1: high level 0: low level		RO									
b0	INTF	Interrupt logo		1: An interrupt has occurred; write 0 to clear the flag 0: No interrupt occurred		RW									

21 Low Voltage Detector (LVD)

21.1 Introduction to LVD

LVD can be used to monitor the power supply voltage and the voltage of chip pins. When the comparison result of the monitored voltage and the LVD threshold meets the trigger condition, the LVD will generate an interrupt or reset signal, and the user can perform some urgent tasks according to the signal.

21.2 Main characteristics

- 4-channel monitoring sources, AVCC, PC13, PB08, PB07
- 16-step threshold voltage, optional
- 8 trigger conditions, combinations of high level, rising edge and falling edge
- 2 trigger results, reset and interrupt
- 8-stage filter configuration to prevent false triggering
- With hysteresis function, strong anti-interference
- Supports operation in DeepSleep mode

21.3 Functional description

21.3.1 Functional block diagram

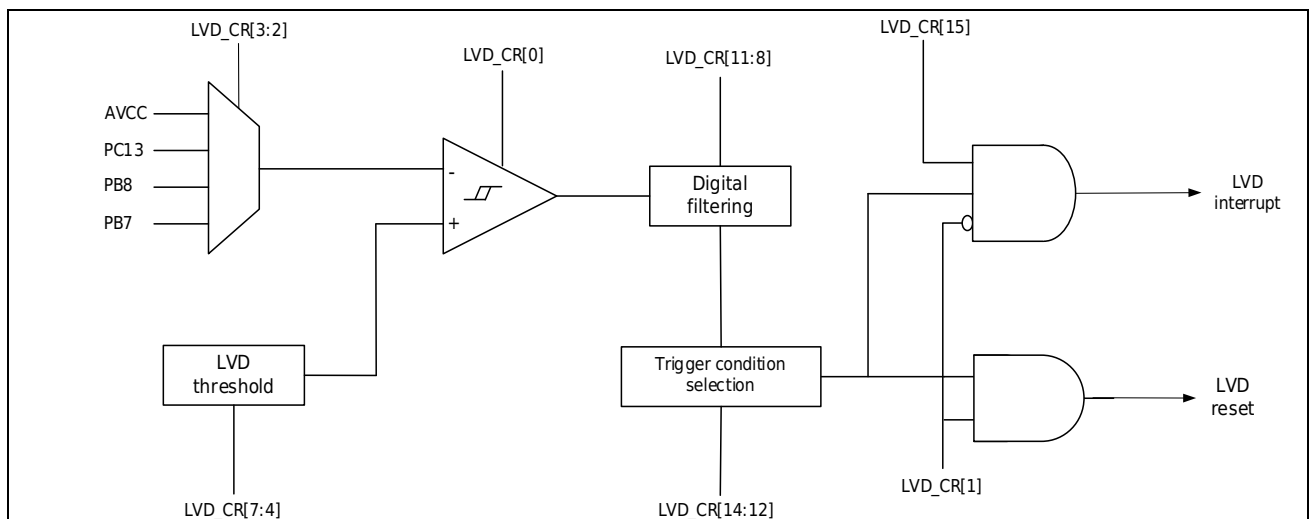


Figure 21-1 LVD Block Diagram

21.3.2 Hysteresis function

The built-in voltage comparator of LVD has a hysteresis function, which can enhance the anti-interference ability of the chip. LVD will not flip until the input signal is higher or lower than the threshold voltage of 20mV. The input and output signals are as follows:

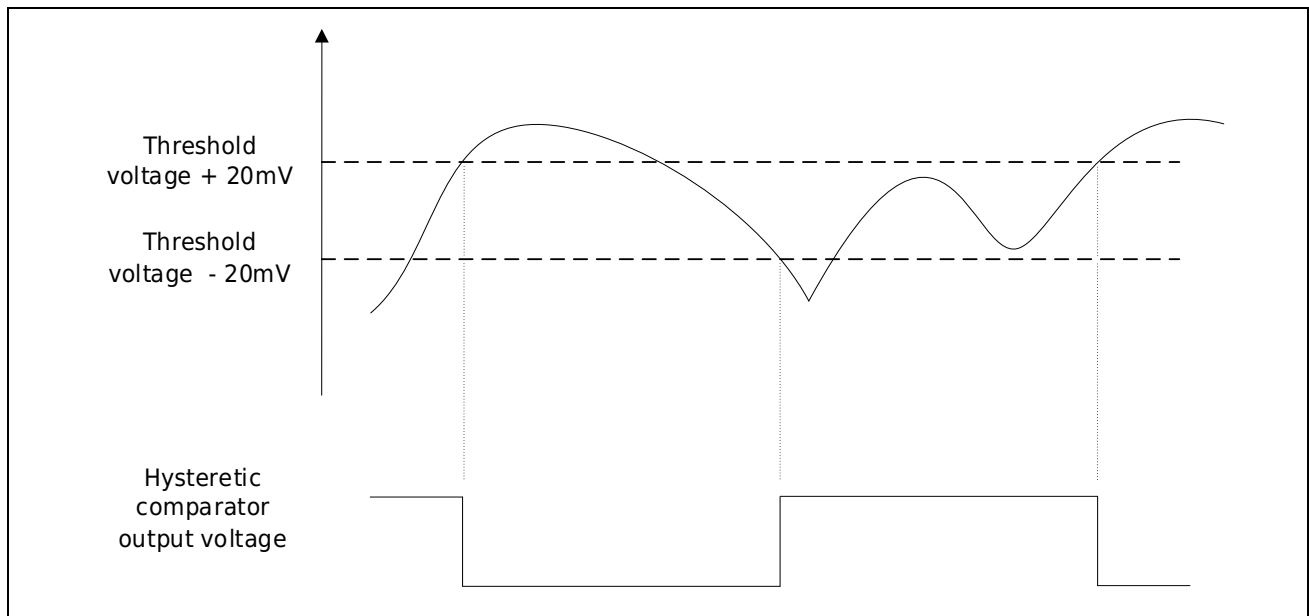


Figure 21-2 LVD Hysteresis Response

21.3.3 Digital filtering

If the working environment of the chip is bad, the output of the hysteresis comparator will appear noise signal. the noise signal whose pulse width is less than LVD_CR.FltTime in the output waveform of the hysteresis comparator can be filtered out. If the digital filter module is disabled, the input and output signals of the digital filter module are the same. The filter clock used by the digital filter module is a built-in RC oscillator with an oscillation frequency of about 150KHz.

When the digital filter module is enabled, the input and output waveforms of the filter are as follows:

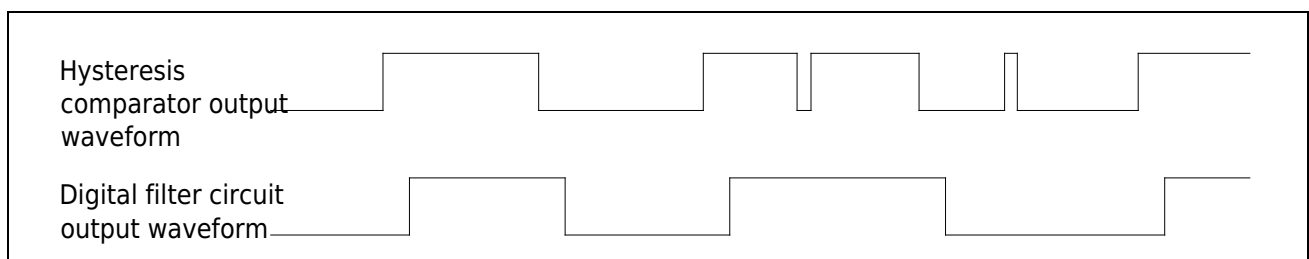


Figure 21-3 LVD Filter Output

21.4 Programming example

21.4.1 LVD configured as low voltage reset

In this mode, the MCU is reset when the monitored voltage is lower than the threshold voltage.

The configuration method is as follows:

- Step1. Configure LVD_CR.Source to select the voltage source to be monitored;
- Step2. Configure LVD_CR.VTDS to select the threshold voltage of LVD;
- Step3. Configure LVD_CR.FltTime to select the LVD filter time;
- Step4. Set LVD_CR.FltEn to 1 to enable LVD filtering;
- Step5. Set LVD_CR.HTEN to 1, select high level to trigger LVD action;
- Step6. Set LVD_CR.ACT to 1, select LVD action as reset;
- Step7. Set LVD_CR.EN to 1 to enable LVD.

21.4.2 LVD configured as voltage change interrupt

In this mode, an interrupt is generated when the monitored voltage goes above or below the threshold voltage.

The configuration method is as follows:

- Step1. Configure LVD_CR.Source to select the voltage source to be monitored;
- Step2. Configure LVD_CR.VTDS to select the threshold voltage of LVD;
- Step3. Configure LVD_CR.FltTime to select the LVD filter time;
- Step4. Set LVD_CR.FltEn to 1 to enable LVD filtering;
- Step5. Set LVD_CR.RTEN and LVD_CR.FTEN to 1, select level change to trigger LVD action;
- Step6. Set LVD_CR.ACT to 0, select LVD action as interrupt;
- Step7. Set LVD_CR.IE to 1 to enable LVD interrupt;
- Step8. Enable the LVD interrupt in the NVIC interrupt vector table;
- Step9. Set LVD_CR.EN to 1 to enable LVD;
- Step10. Execute the operations required by the user in the LVD interrupt service routine; write 0x00 to LVD_SR to clear the interrupt flag before exiting the interrupt service routine.

21.5 Register description

21.5.1 Register list

Base address: 0x4000 AA00

Table 21-1 LVD Register List

Register name	Offset address	Reset value
LVD Configuration Register(LVD_CR)	0x00	0x0000 0000
LVD Interrupt Flag Register(LVD_SR)	0x04	0x0000 0000

21.5.2 LVD Configuration Register (LVD_CR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
IE	HTEN	RTEN	FTEN	FltTime			FltEn	VTDS				Source		ACT	EN

Bit	Marking	Place name	Function	Read and write
b31~b16	Res	Reserved bit	Reserved, must remain at reset value	
b15	IE	LVD interrupt enable	1: Enable; 0: Disabled.	RW
b14	HTEN	High level trigger enable (The monitored voltage is lower than the threshold voltage)	1: Enable; 0: Disabled.	RW
b13	RTEN	Rising edge trigger enable (The monitored voltage changes from above the threshold voltage to below the threshold voltage)	1: Enable; 0: Disabled.	RW
b12	FTEN	Falling edge trigger enable (The monitored voltage changes from below the threshold voltage to above the threshold voltage)	1: Enable; 0: Disabled.	RW
b11~b9	FltTime	Digital filter time configuration (The filter clock is 150K clock)	Filter time data refer to data sheet Note: The filter time is only valid when FLTEN is 1.	RW
b8	FltEn	Digital filter enable configuration	1: Enable digital filtering 0: disable digital filtering Note: When the digital filter is enabled, the system will automatically enable the RCL clock.	RW
b7~b4	VTDS	LVD Threshold Voltage Selection	Threshold voltage data refer to the data sheet	RW
b3~b2	Source	LVD monitoring source selection	00: AVCC voltage 01: The voltage of the PC13 port input signal 10: The voltage of the input signal at the PB08 port 11: Voltage of input signal at PB07 port	RW
b1	ACT	LVD trigger action selection	1: system reset 0: NVIC interrupt	RW
b0	EN	LVD enable control	1: enable LVD 0: disable LVD	RW

21.5.3 LVD Interrupt Register (LVD_SR)

Reset value: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
Res															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Res														FLTV	INTF

Bit	Marking	Place name	Function	Read and write
b31~b2	Res	Reserved bit	Reserved, must remain at reset value	
b1	FLTV	Level value of digital filter output	1: digital filter output high level 0: Digital filter output low level	RO
b0	INTF	LVD interrupt flag	1: LVD interrupt flag has been generated, write 0 to clear 0: LVD interrupt flag not generated	RW0

22 Debug Controller (DBGC)

This product refers to the following ARM technical documents:

- Cortex™-M4F r0p1 Technical Reference Manual (TRM)
- ARM debug interface V5
- ARM CoreSight Design Suite Version r0p1 Technical Reference Manual

22.1 Introduction

The core of this MCU is ARM® Cortex®-M4, which includes hardware for advanced debugging functions. With these debugging features, you can stop the kernel when fetching fingers (instruction breakpoints) or accessing data (data breakpoints). When the kernel stops, you can query the internal state of the kernel and the external state of the system. After the query completes, the kernel and system are restored and program execution is restored.

Debug interface:

- Serial debug interface SWD

22.2 DBGC system block diagram

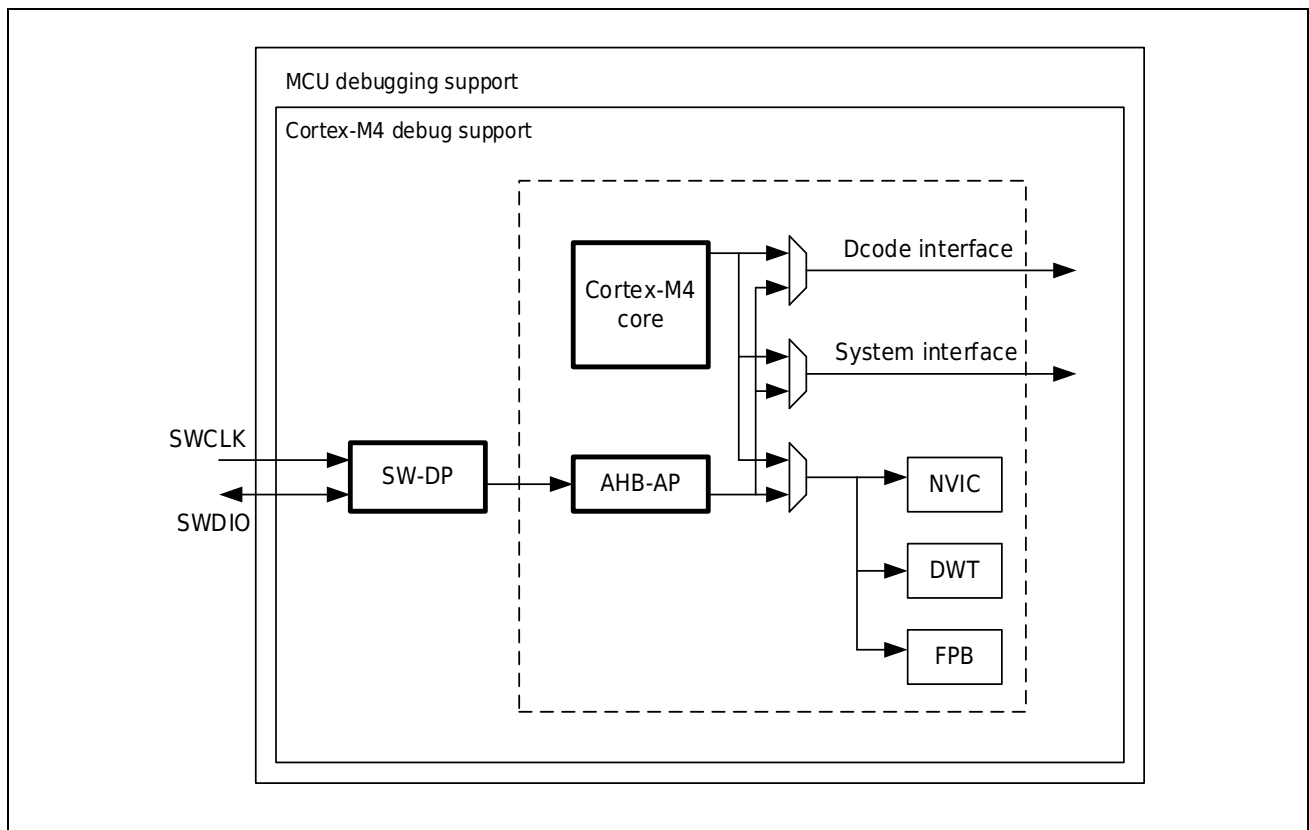


Figure 22-1 Commissioning control system

The ARM Cortex™-M4F core provides integrated on-chip debugging support. It includes:

- SW-DP: SWD debug port
- AHP-AP: AHB access port
- FPB: Flash instruction breakpoint
- DWT: data watchpoint trigger

Note: For more information on the debug features supported by the ARM Cortex™-M4F core, see the Cortex™-M4F-r0p1 Technical Reference Manual and the CoreSight Design Suite r0p1 Technical Reference Manual.

22.3 SW-DP Debug Port(SWD)

This MCU core integrates SWD debug port. This port is the ARM standard CoreSight debug port.

22.3.1 SWD port pin

SWD interface of this series requires 2 pins, as shown in the following table:

Table 22-1 SWD port pin

SWD port name	Debug function	Pin assignment
SWCLK	Serial clock	PA14
SWDIO	Serial data input / output	PA13

22.3.2 SW-DP debug port pin

If the [Encryption Chip] option is enabled when programming the program, the SWD debugging function will be disabled after power-on. If the [encrypted chip] option is not enabled when programming the program, the PA13/PA14 pins are initialized as dedicated pins that can be used by the debugger after power-on. Users can set the SYSCTRL_CR1.SWDIO register to disable the SWD pin debug function, and the SWD pin will be released to be used as a normal GPIO. SWD pins are summarized in the following table:

Table 22-2 SW-DP debug port pin

[Encryption chip] option	SYSCTRL_CR1.SWDIO configuration	PA13/PA14 function
Encryption	0	NA
Encryption	1	GPIO
No encryption	0	SWD
No encryption	1	GPIO

22.3.3 Internal pull-ups on SWD pins

user software disables the debug function of the SWD pins, the GPIO controller takes control of these pins. GPIO control register places the I/O in the following equivalent states:

- SWDIO: input pull-up

- SWCLK: input pull-up

No external resistors need to be added due to built-in pull-up resistors.

22.4 Register description

22.4.1 Register list

Base address: 0x4000 B800

Table 22-3 DBGCR register list

Register name	Offset address	Reset value
SYSCTRL_DEBUG (Timer Debug Halt Control Register)	0x64	0x0000 0000

22.4.2 Timer Debug Halt Control Register (SYSCTRL_DEBUG)

For specific register content, please refer to the [Timer Debug Halt Control Register (SYSCTRL_DEBUG)] chapter.

23 Device Electronic Signature

The electronic signature is stored in the system storage area of the flash memory module and can be read by SWD or CPU. HC32Fxxx / HC32Lxxx microcontrollers with different configurations.

23.1 Product Unique Identifier (UID) Register (48 bits)

Typical application scenarios for unique identifiers:

- Used as a serial number
- UID is used as a security key when used in conjunction with software cryptographic primitives and protocols to increase the security of code in Flash prior to programming the internal Flash
- Activation of the secure bootstrap process, etc.

The 48-bit unique device identifier provides a reference number that is unique to any device and in any context. The user can never change these bits. The 48-bit UID can also be read in different ways like single byte/halfword/word and then concatenated using a custom algorithm.

Base address: 0x0100 14E0

Table 23-1 UID Definition

Offset address	Description	UID Bits (48 bits)							
		7	6	5	4	3	2	1	0
0	Y Coordinate on the wafer	UID[7:0]							
1	X Coordinate on the wafer	UID[15:8]							
2	Wafer Number	UID[23:16]							
3	Lot Number	UID[31:24]							
4		UID[39:32]							
5		UID[47:40]							

23.2 Product Model Register

0x0100 1140 to 0x0100 114F store the ASCII codes of the product models. If the product model is less than 16 bytes, it should be filled with 0x00.

Example: 48433332463432304B41544200000000 models for HC32F420KATB products.

23.3 FLASH capacity register

Base address: 0x0100 1160

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
FLASHSIZE[31:16]															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
FLASHSIZE [15:0]															

Bit	Marking	Place name	Function	Read and write
b31~b0	FLASHSIZE	The capacity of the product's built-in Flash	Note: The unit is Byte 0x00008000 represents 32KB 0x00010000 represents 64KB 0x00020000 represents 128KB	RO

23.4 RAM capacity register

Base address: 0x0100 1164

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
RAMSIZE[31:16]															
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
RAMSIZE [15:0]															

Bit	Marking	Place name	Function	Read and write
b31~b0	RAMSIZE	The capacity of the built-in RAM of the product	Note: The unit is Byte 0x00000800 represents 2KB 0x00001000 represents 4KB 0x00002000 represents 8KB	RO

23.5 Pin Count Register

Base address: 0x0100 116A

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PINCOUNT [15:0]															

Bit	Marking	Place name	Function	Read and write
b15~b0	PINCOUNT	Number of product pins	0x0020 represents 32 pins 0x0030 represents 48 pins	RO

24 Appendix A SysTick Timer

24.1 Introduction to SysTick Timers

If the OS wants to support multitasking, it needs to perform context switching periodically, so hardware resources such as timers are needed to interrupt program execution. When the timer interrupt is generated, the processor will perform OS task scheduling in exception handling, and will also perform OS maintenance work at the same time. There is a simple timer called SysTick in the Cortex-M4 processor, which is used to generate periodic interrupt requests.

SysTick is a 24-bit timer and counts down. After the count of the timer is reduced to 0, a programmable value will be reloaded, and a SysTick exception (exception number 15) will be generated at the same time. This abnormal event will cause the execution of SysTick exception handling, which is a part of the OS.

For systems that do not require an OS, the SysTick timer can also be used for other purposes, such as timing, timing, or providing an interrupt source for tasks that need to be executed periodically. The generation of SysTick exception is controllable. If the exception is disabled, the SysTick timer can still be used by polling, such as checking the current count value or polling the count flag.

24.2 Set SysTick

Since the reload value and current value of the SysTick timer are undefined at reset, in order to prevent abnormal results, the configuration of SysTick needs to follow a certain process:

- Step1. Configure SysTick->CTRL. ENABLE is 0, disable SysTick.
- Step2. Configure SysTick->CTRL. CLKSOURCE to select the clock source of SysTick.
- Step3. Configure SysTick->LOAD, select the overflow period of SysTick.
- Step4. Write any value to SysTick->VAL, clear SysTick->VAL and SysTick->CTRL. COUNTFLAG.
- Step5. Configure SysTick->CTRL. TICKINT is 1, enable SysTick interrupt.
- Step6. Set Sys Tick->CTRL. ENABLE to 1 to enable SysTick.
- Step7. Read SysTick->CTRL in the interrupt service routine to clear the overflow flag.

Note: Systick overflow period is SysTick->LOAD+1, the configuration example is as follows:

Clock source	SysTick->LOAD	Overflow cycle
HCLK: For example, 4MHz	3999	1ms
XTL: For example, 32.768KHz	327	10.01ms

24.3 SysTick register

Table 24-1 SysTick register

Register name	Address	CMSIS symbol
SysTick Control and Status Register (SYS_CSR)	0XE000 E010	SysTick->CTRL
SysTick reload register (SYS_RVR)	0XE000 E014	SysTick->LOAD
SysTick current value register (SYS_CVR)	0XE000 E018	SysTick->VAL
SysTick calibration value register (SYS_CALIR)	0XE000 E01C	SysTick->CALIB

24.3.1 SysTick Control and Status Register (CTRL)

Bit	Symbol	Functional description	Types	Reset value
31:17	Reserved	-	-	-
16	COUNTFLAG	SysTick timer overflow flag 1: SysTick timer underflow occurred 0: SysTick timer has not overflowed Reading this register clears the COUNTFLAG flag	RO	0
15:3	Reserved	-	-	-
2	CLKSOURCE	SysTick clock source selection 1: Use the core clock (HCLK) 0: Use the reference clock (XTL) Note: When the reference clock is selected as the clock source of SysTick, it is necessary to synchronize and enable the XTL clock source and SysCtrl.PERI_CLKEN0.TICK	RW	0
1	TICKINT	SysTick interrupt enable 1: enable interrupt 0: disable interrupt	RW	0
0	ENABLE	SysTick timer enable 1: enable SysTick 0: disable SysTick	RW	0

24.3.1 SysTick reload register (LOAD)

Bit	Symbol	Functional description	Types	Reset value
31:24	Reserved	-	-	-
23:0	RELOAD	SysTick timer reload value	RW	Undefined

24.3.2 SysTick current value register (VAL)

Bit	Symbol	Functional description	Types	Reset value
31:24	Reserved	-	-	-
23:0	CURRENT	Read this register to get the current count value of the SysTick timer Write any value to this register, clear this register and COUNTFLAG	RW	Undefined

24.3.3 SysTick calibration value register (CALIB)

Bit	Symbol	Functional description	Types	Reset value
31	NOREF	SysTick current count clock flag 1: The current count clock is the core clock 0: The current count clock is external reference clock (XTL)	RO	-
30	SKEW	TENMS accuracy Indication 1: TENMS value represents roughly 10ms 0: TENMS value represents exactly 10ms	RO	
29:24	Reserved	-	-	
23:0	TENMS	10 millisecond calibration value	RO	-

Version Revision History

Version Number	Revision Date	Modify the content
Rev1.00	2023/03/31	First edition release.
Rev1.10	2024/01/17	1) Reset Controller (RESET): Modified the reset values of SYSCTRL_APBRS1, SYSCTRL_AHBRST0, and SYSCTRL_RSTFLAG; removed the description "requires software initialization and clearing" from bits b2 to b7 of SYSCTRL_RSTFLAG, and added a note to bit b7. 2) Interrupt Controller (NVIC): Modified the text in "Figure 5-6 Interrupt pending interrupts generated during interrupt processing can also be acknowledged" to "Interrupt Service Routine". 3) General-Purpose Input-Output Port Controller (GPIO): Modified the description of the number of GPIO control registers in Section 6.3.5 "Port general-purpose input and output functions" to 6. 4) Clock Trim Module (CTRM): Added the division value for the PRS bit of CTRM_CR1 being 0b1000. 5) Flash Controller (FLASH): Added a step to write 0 to OP after erase and write operations of FLASH; added a note to the OP bit of FLASH_CR. 6) DMA Controller (DMAC): Added descriptions for the external trigger and event update of ATIM0 and ATIM3 for the TRI_SEL bit of DMAC_CONFx. 7) Composite Timer (CTIM): Modified the description of the number of CTIMs in Section 13.1 "Overview" to "Each CTIM can be independently and flexibly configured as GTIM or BTIM"; added the pre-scaler value limit for the capture function of GTIM, and noted that the capture function cannot be used when counting with an external ETR; added a note to the PRS bit of GTIM_CR0: the pre-scaler coefficient for capture mode can only be set to 0, 1, 2, or 3; modified the division values for the ETRFLT bit of GTIM_CR1 being 0b110 and 0b111. 8) Advanced Timer (ATIM): Modified the content of Table 14-5 "Capture Input CH0A Signal List"; modified the name of Figure 14-51 to "ATIM3 CH0A Port Selection"; modified the description of the DIR bit of ATIMx_M23CR to write prohibited; added the signal description of ATIM0 to the IA0S bit of ATIMx_MSCR, and modified TIM3 to ATIM3; added the description for ATIM3 to the MMS bit. 9) Independent Watchdog (IWDG): Added a 500us delay after writing 0xCCCC to the IWDG_KR register in Section 15.4 "Programming Examples"; added a note to delay 500us for the IWDG_KR register. 10) Universal Synchronous/ Asynchronous Receiver Transmitter (USART): 17.3.9.2 In the description of clock sampling and receiving tolerance, the receiving pin in smart card (7816) mode is modified to TXD ; modified the name of the receive pin in Figure 17-16 Smart Card Pattern synchronization and sampling timing Figure to "Receive Pin (TXD)"; modified the protection time in Figure 17-19 Smart card Pattern sending data legend to be more than 2etu; removed the description "The data behavior on the LIN bus is as shown in the following figure" in Section 17.3.11.2 LIN data format. 11) Analog-to-Digital Converters (ADC): Modified the text in Section 19.6 Conversion Accumulation Mode to "conversion channels 8, 5, 2 are configured as AIN5, and conversion channels 7, 4, 1 are configured as AIN1".

Version Number	Revision Date	Modify the content
		12) Low Voltage Detector (LVD): Added the description that LVD supports operation in DeepSleep mode. 13) Added new sections on Device Electronic Signature and SysTick Timer.
Rev1.20	2024/05/06	1) System Controller (SYSCTRL): Removed the description "and enable XTH" from 3.1.4 External crystal oscillator clock XTH; 3.1.5 Phase-locked loop clock PLL: Updated the parameter description content; 3.2.5 Example of switching between PLL (reference clock is RCHCLKD) and RCHCLKD: Modified the range from 4 MHz to 24 MHz in Step 2 to 4 MHz to 12 MHz, and added a note; 3.5.4 Clock enable control register 2 (SYSCTRL_CR2): Modified the note in the function description of b5; 3.5.9 PLL control register (SYSCTRL_PLLCR): Added a note in the function description of b1 to b0. 2) Analog-to-Digital Converters (ADC): 19.1 Module introduction: Changed the 19 input channels to 18, and removed the description "one built-in BGR 1.2V voltage"; 19.2 ADC block diagram: Removed the description of 1.2V Ref in the diagram; 19.10 Measure ambient temperature using the temperature sensor: Modified the formula and corresponding address values, modified the numerical values in the calculation example, and modified the register names in the operation process; 19.11.3 ADC basic configuration register 0 (ADC_CR0): Removed the description "one built-in BGR 1.2V voltage" in the function description of b11; Removed the description "10010 internal reference 1.2V output voltage" in the function description of b8 to b4. 3) Analog Voltage Comparator (VC): 20.2 Main features: Changed the 5 on-chip input signals to 4; 20.3.1 Functional block diagram: Removed the description of 1.2V reference voltage in the diagram; 20.4.3 Control register 1 (VC_CR1): Removed the description "1101 built-in reference voltage 1.2V" in the function description of b7 to b4.
Rev1.21	2025/01/14	1) System Controller (SYSCTRL): 3.5.4 Clock Enable Control Register 2 (SYSCTRL_CR2): Revise the note in the functional description of the EXXTH bit.
Rev1.22	2025/05/07	1) Serial Peripheral Interface (SPI): 8.2 Main features: Removed the description of the maximum communication rate between the master and slave machines.
Rev1.23	2026/02/11	1) Reset Controller (RESET): 4.2.6 Reset Flag Register (SYSCTRL_RSTFLAG): Added a new note "The reset value is 0x03 at power-on reset. After the program determines the reset source, it should clear the register so that it can determine the reset source the next time a reset occurs". 2) Advanced Timer (ATIM): 14.3.6 Mode 2/3 compare capture mode: "Single pulse output" section, optimized the details of pulse count timing diagram, and added relevant instructions; "Capture Input" section, added a new note "When GATE is configured for compare output, its output mode only supports being set to force the output to 0 or 1 and having the PWM mode active (that is, the GATE output is valid when OCMB0 is configured as 0/1/6/7). 3) Independent Watchdog (IWDG): 15.5.3 Control Register (IWDG_CR): note point 2, revised the description to "This register can only be modified when IWDG_SR.PRSF is 0"; 15.5.5 Status Register (IWDG_SR), changed the bit0 name from "IWDG_CR.PRS register update flag" to "CR register update flag" with corresponding

Version Number	Revision Date	Modify the content
		functional descriptions updated to reflect the new naming, and revised the explanation to "The CR register can be updated only when the PRSF bit is 0". 4) Analog-to-Digital Converters (ADC): 19.6 Continuous Conversion Accumulation Mode: changed ADC_IFR.SQRIF to ADC_IFR.EOS, and ADC_ICR.SQRIC to ADC_ICR.EOS. In Figure 19-4, modified SQRIF to ADC_IFR.EOS; 19.9 ADC Interrupt: added descriptions related to the ADC conversion result overflow interrupt OV; Standardized the bit name descriptions for ADC_IER.OV, ADC_IFR.OV, and ADC_ICR.OV to "ADC conversion result overflow XXX".
Rev1.30	2026/08/19	1) DMA Controller (DMAC): In the section "12.4.4 DMA Hardware Block Transmission Mode", modified the hardware block transmission schematic. 2) Advanced Timer (ATIM): In the section "14.3.6 Mode 2/3 Comparison Capture Mode - Capture Input", modified the constraint for selecting the input signal for CH0B.