XHSC 小华半导体 XIAOHUA SEMICONDUCTOR

HC32F002 Series

32-bit ARM® Cortex®-M0+ Microcontroller

Reference Manual

Rev1.02 June 2023

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.

★ Information in this notification replaces and replaces information in previous versions.

Table of Contents

Table Index

Graph Index

Introduction

The HC32F002 series is a basic general-purpose 32 -bit MCU with a wide operating voltage range. The chip integrates 10 -bit 1Msps high-speed SAR ADC, high-performance PWM timer, low-power UART, SPI, I2C and other rich peripherals, and has the characteristics of high integration, high reliability and low power consumption. The HC32F002 series adopts the Cortex-M0+ core, the main frequency is up to 48MHz, cooperates with the mature Keil and IAR debugging and development software, and supports the use of C language and assembly language development.

Typical application

- Motor Control, Battery Management
- Smart home, medical equipment
- Security alarm, intelligent transportation
- Sensor modules, wireless modules, shelf labels

1 System Structure

1.1 Overview

This product system consists of the following parts:

- 1 AHB bus Master:
  - Cortex-M0+
- 4 AHB Bus Slaves:
  - FLASH memory
  - SRAM memory
  - AHB0, AHB to APB Bridge contains all APB interface peripherals
  - AHB1, contains all AHB interface peripherals

The entire system bus structure is realized by multi-level AHB-lite bus interconnection. As shown below:

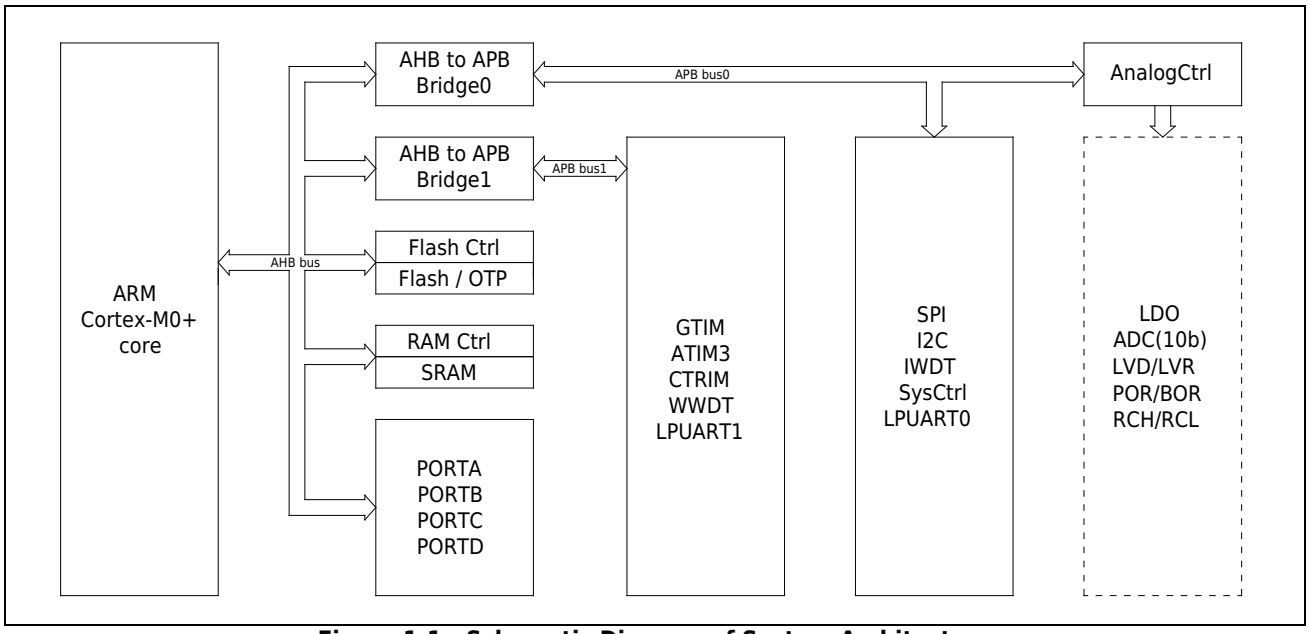


Figure 1-1 Schematic Diagram of System Architecture

1.2 Memory and Module Address Assignment

Table 1-1 Address Division Table

Categories	Boundary Address	Size	Memory Area
memory	0x0000_0000 – 0x0000_47FF	18K Byte	FLASH
	0x0010_0F00 –0x0010_0F7F	128 Byte	OTP
	0x2000_0000 – 0x2000_07FF	2K Byte	SRAM
APB0 Peripheral equipment	0x4000_0000 – 0x4000_03FF	1K Byte	LPUART0
	0x4000_0400 – 0x4000_07FF	1KByte	I2C
	0x4000_0800 – 0x4000_0BFF	1KByte	SPI
	0x4000_0F80 – 0x4000_0FFF	128Byte	IWDT
	0x4000_2000 – 0x4000_23FF	1KByte	SYSCTRL
	0x4000_2400 – 0x4000_27FF	1KByte	ANALOGCTRL
APB1 Peripheral equipment	0x4000_4000– 0x4000_43FF	1KByte	LPUART1
	0x4000_5000– 0x4000_53FF	1KByte	CTRIM
	0x4000_5800– 0x4000_5BFF	1KByte	ATIM3
	0x4000_6800 – 0x4000_6BFF	1KByte	WWDT
	0x4000_7400 – 0x4000_77FF	1KByte	GTIM
AHB Peripheral equipment	0x4002_0000 - 0x4002_03FF	1KByte	FLASH CTRL
	0x4002_0C00 - 0x4002_0FFF	1KByte	PORT CTRL

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:

- Running mode (Active Mode): CPU runs, on-chip peripherals run.
- Sleep Mode (Sleep Mode): The CPU stops and the on-chip peripherals run.
- Deep sleep mode (Deep Sleep Mode): The CPU stops, and the 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. Executing the WFI command in the running mode can make the product enter the sleep mode or deep sleep mode. Wake up from sleep mode or deep eye sleep mode via an interrupt and return to run mode.

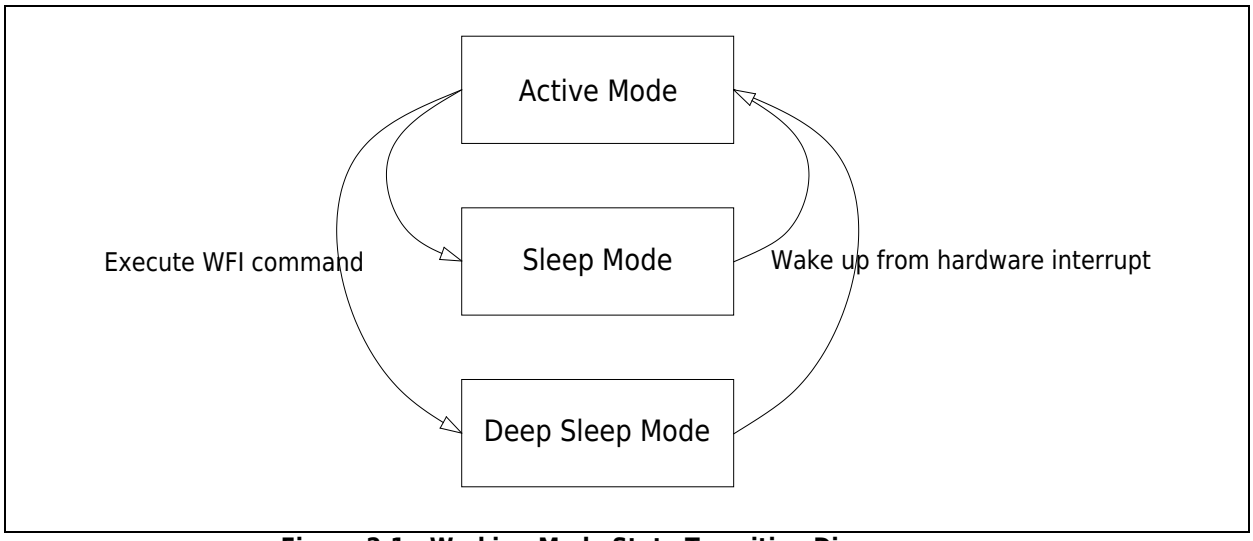


Figure 2-1 Working Mode State Transition Diagram

In various working modes, the interrupts that the CPU can respond to are shown in the table below.

Interrupt vector number	Interrupt source	Operating mode	Sleep Mode	Deep Sleep Mode
[0]	GPIO_PA	✔	✔	✔
[1]	GPIO_PB	✔	✔	✔
[2]	GPIO_PC	✔	✔	✔
[3]	GPIO_PD	✔	✔	✔
[4]	-			
[5]	ATIM3	✔	✔	
[7]	-			
[8]	LPUART0	✔	✔	✔
[9]	LPUART1	✔	✔	✔
[10]	SPI	✔	✔	
[11]	-			
[12]	I2C	✔	✔	
[13]	-			
[14]	-			
[15]	GTIM / BTIM3-5	✔	✔	
[17]	-			
[20]	-			
[21]	-			
[22]	IWDT	✔	✔	✔
	WWDT	✔	✔	
[23]	-			
[24]	ADC	✔	✔	
[27]	-			
[28]	LVD	✔	✔	✔
[30]	FLASH / RAM	✔	✔	
[31]	CTRIM	✔	✔	✔

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

	reset source	Operating mode	Sleep Mode	Deep Sleep Mode
[0]	Power-on reset POR/BOR	✔	✔	✔
[1]	hardware reset RESETB pin	✔	✔	✔
[2]	hardware reset LVD	✔	✔	✔
[3]	hardware reset IWDT	✔	✔	✔
[4]	hardware reset Cortex-M0+ LOCKUP	✔		
[5]	Software reset Cortex-M0+ SYSRESETREQ	✔		
[6]	hardware reset WWDT	✔	✔	

2.3 Active Mode

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

After a system reset or after waking up from an interrupt, the MCU works in Run mode.

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), and modify SysCtrl.CR0.PclkPrs
- Turn off the clock of unused on-chip peripherals, modify SysCtrl.PeriClkEnx

2.4 Sleep Mode

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

In the running mode, set SCB->SCR.SLEEPDEEP to 0, and execute the WFI command to make the MCU work in the 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 (PCLK), and modify SysCtrl.CR0.PclkPrs
- Turn off the clock of unused on-chip peripherals, modify SysCtrl.PeriClkEnx
- Set SCB_SCR.SLEEPONEXIT to 1, so that the MCU will automatically enter the sleep mode after executing the interrupt service routine.

2.5 Deep Sleep Mode

In this working mode, the CPU stops running, and the on-chip high-speed clock RCH is automatically turned off. If the source of HCLK is the on-chip 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 the on-chip low-speed clock RCL, all the on-chip peripherals work normally. The power consumption in this mode is much lower than that in sleep mode.

If SysCtrl.CR1.AutoSwitch is set to 1, the source of the system clock when the MCU exits 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 the high-speed clock as the system clock (SystemClk), modify SysCtrl.CR0.ClkSrc
- Turn off the clock of unused on-chip low-power peripherals, modify SysCtrl.PeriClkEnx
- Set SCB_SCR.SLEEPONEXIT to 1, so that the MCU will automatically enter the 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:

functional module	Operating mode	Sleep Mode	Deep Sleep Mode
CPU	✔		
WWDT	✔	✔	
GTIM / BTIM3-5	✔	✔	
ATIM3	✔	✔	
SPI	✔	✔	
I2C	✔	✔	
ADC	✔	✔	
GPIO_PA	✔	✔	✔
GPIO_PB	✔	✔	✔
GPIO_PC	✔	✔	✔
GPIO_PD	✔	✔	✔
IWDT	✔	✔	✔
LPUART0 / LPUART1	✔	✔	✔
CTRIM	✔	✔	✔
LVD	✔	✔	✔

2.7 Register

2.7.1 System Control Register (SCB_SCR)

Address: 0xE000ED10

Reset value: 0x0000 0000

Bit	Marking	Functional description	Read and write
31:5	RESERVED	Keep	
4	SEVONPEND	When set to 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
3	RESERVED	Keep	
2	SLEEPDEEP	When set to 1, implement WFI to enter deep sleep, and this product enters Deep Sleep mode When set to 0, execute WFI to enter sleep mode, and this product enters sleep/Idle mode	RW
1	SLEEPONEXIT	When set to 1, the processor automatically enters sleep mode (WFI) when exiting exception handling and returning to the program thread When set to 0, the feature is automatically disabled	RW
0	RESERVED	Keep	

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 three different clock sources as the system clock:

- Internal high-speed RC clock RCH (output frequency is 44~48MHz)
- Internal low-speed RC clock RCL (38.4K and 32.8K configurable)
- Clock signal input by external pin PA01 (frequency 1~16MHz)

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

The HCLK clock signal can be output through the HCLK_OUT pin; the RCH/RCL/HCLK/PCLK clock signal can be output through the TCLK_OUT pin.

3.1.1 Clock Architecture Diagram

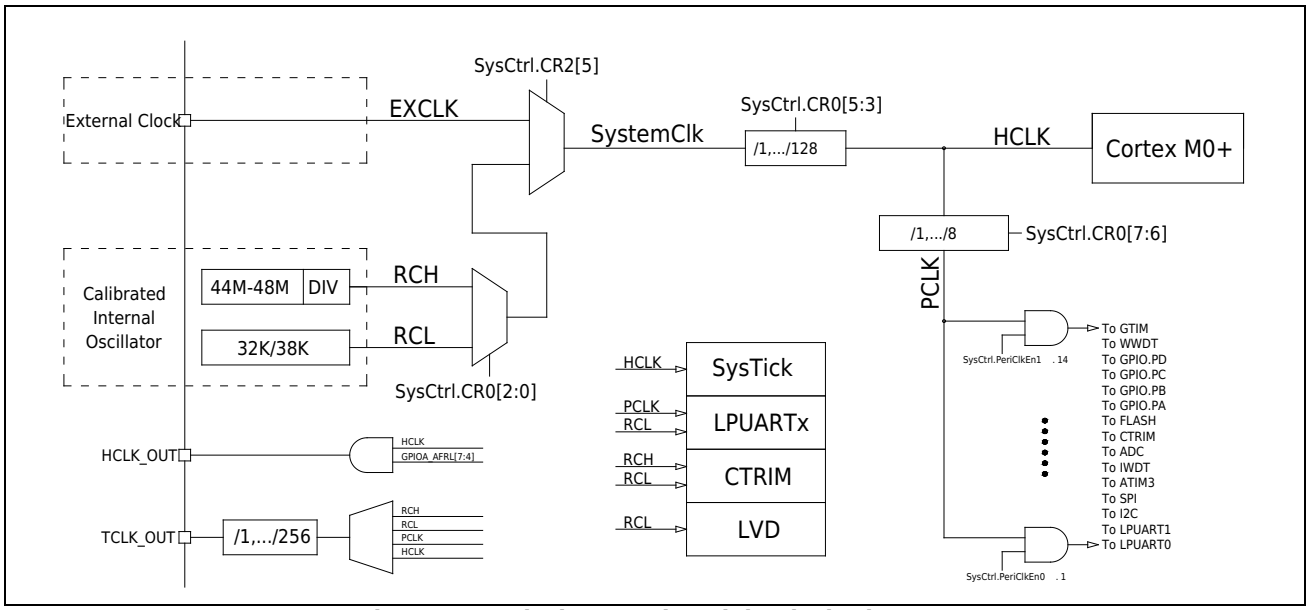


Figure 3-1 Clock Control Module Block Diagram

3.1.2 Internal high speed RC clock RCH

The default clock source after the chip is powered on or reset is RCHCLKD with a frequency of 4MHz; when the system enters DeepSleep, this high-speed clock will be automatically turned off. It only takes 5us for the internal high-speed clock to be stable from wake-up. Switching the system clock to RCHCLKD before entering deep sleep mode will allow the system to quickly respond to interrupts.

Change the value of register SysCtrl.RCH.TRIM to fine -tune the oscillation frequency of RCH. Every time the register value increases by 1, the oscillation frequency of RCH will increase by about 0.2%. The safe operating range of the oscillation frequency of RCH is 44 ~48MHz. If the oscillation frequency of RCH is adjusted to greater than 48MHz, the chip may malfunction. The chip has pre-adjusted two frequency points 44.24MHz and 48MHz before leaving the factory; if you need other frequencies, please manually adjust the value of this register. If the RCH real-time calibration function is enabled in the CTRIM module, the accuracy of the RCH in the entire working range can reach 0.5%, see the CTRIM chapter for details.

Changing the value of the register SysCtrl.RCH.DIV can adjust the frequency division ratio of RCH oscillation frequency and RCHCLKD, and the adjustment range is 1 to C 16.

3.1.3 Internal low speed RC clock RCL

The output frequency of the internal low-speed clock can be selected through the register SysCtrl.RCL.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 Externalinput clock EXTCLK

A clock signal with a frequency of 1~16MHz can be input from the PA1 pin as the system clock. The method of inputting the clock signal from the PA1 pin is: configure the pin as GPIO input; set SysCtrl.CR2.EXTEN to 1.

3.2 System Clock Switching

The system clock can choose 3 clock sources: RCHCLKD, RCL, and external pin input clock, which are controlled by registers SysCtrl.CR0.ClkSrc and SysCtrl.CR2.EXTEN. Any two of the three clock sources, RCHCLKD, RCL, and external clock, can be switched to each other. During the switching process, ensure that HCLK is not greater than 48MHz. It is recommended to perform clock switching according to the clock switching process described below to reduce the probability of clock switching errors.

The FLASH_WAIT register needs to be configured synchronously when the clock is switched. If the clock frequency is not greater than 24MHz, FLASH_WAIT should be set to 0; if the clock frequency is greater than 24MHz, FLASH_WAIT should be set to 1.

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.

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: When connecting to external clock input, GPIO input and external clock input are required.

Step2. Configure the working 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 SysCtrl.CR0.ClkSrc and SysCtrl.CR2.EXTEN to select a new clock source as the system clock.

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:

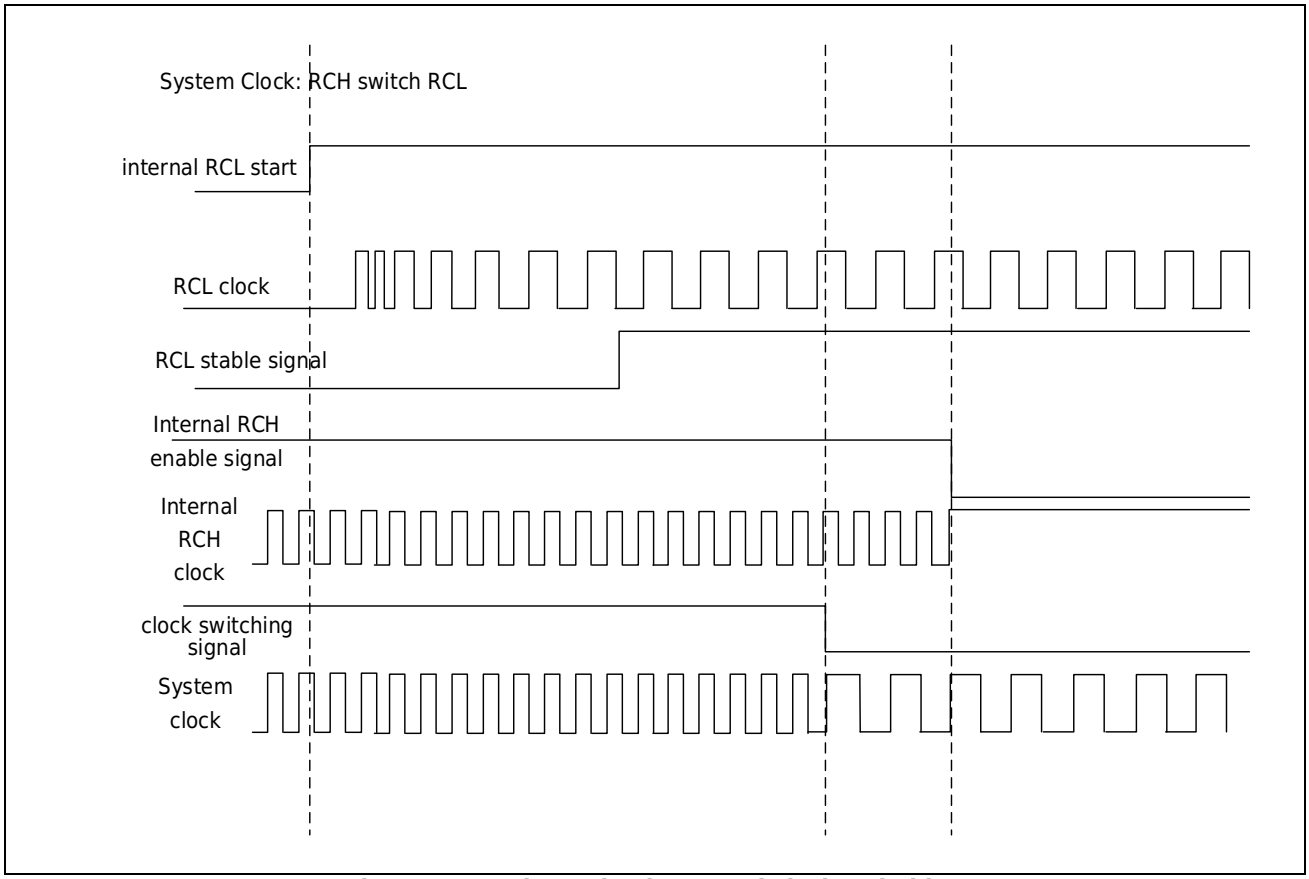


Figure 3-2 Schematic Diagram of Clock Switching

3.2.2 RCH switching process between different output frequencies

Configure SysCtrl.RCH.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 SysCtrl.RCH.DIV to select the frequency division ratio of 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 SysCtrl.RCL.TRIM and SysCtrl.RCL.WaitCycle.

Step2. Write 0x5A5A and 0xA5A5 to the SysCtrl.CR3 register in sequence to enable register rewriting.

Step3. Set SysCtrl.CR2. RCLEN to 1 to enable the RCL oscillator circuit.

Step4. The query waits for the SysCtrl.RCL.Stable flag to become 1 and the RCL output stable clock.

Step5. Write 0x5A5A and 0xA5A5 to the SysCtrl.CR3 register in sequence to enable register rewriting.

Step6. Set SysCtrl.CR0.ClkSrc to 2 to switch the system clock to RCL.

Step7. Set FLASH_WAIT to 0 according to the procedures in the Flash controller chapter.

Step8. Write 0x5A5A and 0xA5A5 to the SysCtrl.CR3 register in sequence to enable register rewriting.

Step9. Set SysCtrl.CR2.xxxEN 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 SysCtrl.RCH.TRIM and SysCtrl.RCH.DIV.

Step2. Write 0x5A5A and 0xA5A5 to the SysCtrl.CR3 register in sequence to enable register rewriting.

Step3. Set SysCtrl.CR2.RCHEN to 1 to enable the RCH oscillator circuit.

Step4. The query waits until the SysCtrl.RCH.Sable flag becomes 1 and the RCH output is 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 SysCtrl.CR3 register in sequence to enable register rewriting.

Step7. Set SysCtrl.CR0.ClkSrc to 0 to 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 SysCtrl.CR3 register in sequence to enable register rewriting.

Step10. Set SysCtrl.CR2.xxxEN to 0 to turn off unused clock sources.

3.3 On-chip peripheral clock control

Most on-chip peripherals require a clock to function properly. When the configuration clock (PCLK) 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 operating clock of some on-chip peripherals is fixed to PCLK, and the working clock of other on-chip peripherals can be configured by the user program.

By setting the control bit of SysCtrl_Peri.ClkEnx 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:

PRIMASK status	WFI behavior	wake	ISR execution
0	IRQ Priority > Current Class	Y	Y
0	IRQ Priority ≦ Current Level	N	N
1	IRQ Priority > Current Class	Y	N
1	IRQ Priority ≦ Current Level	N	N

3.4.1 Method to Execute Interrupt Service Routine After Waking 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 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");
M0P_TIMx->ICLR = 0x00; //Clear Int Flag
NVIC_ClearPendingIRQ(TIMx_IRQn); //Clear Pending Flag
 .. // perform user-defined operations
}
```

3.4.3 exit hibernation

Exiting hibernation (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-M0 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.

1. Enable the NVIC corresponding to the module that needs to wake up the processor
2. Enable the interrupt corresponding to the module that needs to wake up the processor
3. Set SCB->SCR.SLEEPDEEP to 1
4. Set SCB->SCR.SLEEPONEXIT to 1
5. Execute WFI instruction to enter deep sleep mode
6. The system enters the deep sleep mode and waits for the interrupt to wake up, and executes the interrupt service subroutine after waking up
7. Automatically enters sleep mode when exiting interrupt service

Routine:

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

3.5 Register

Base Address: 0x40002000

Table 3-1 System Control Register Table

Register	Offset address	Description
CR0	0x000	System Control Register 0
CR1	0x004	System Control Register 1
CR2	0x008	System Control Register 2
CR3	0x00C	System Control Register 3
RCH	0x010	RCH control register
RCL	0x018	RCL control register
PeriClkEN0	0x030	On-chip peripheral clock control register 0
PeriClkEN1	0x034	On-chip peripheral clock control register 1
DebugActive	0x044	Debug mode module work status control register

3.5.1 System Control Register 0 (CR0)

Offset address: 0x000

Reset value: 0x0000 0000

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

Bit	Marking	Functional description
31:8	Reserved	Keep
7:6	PclkPrs	PCLK clock source selection 00: HCLK 01: HCLK / 2 10: HCLK / 4 11: HCLK / 8
5:3	HclkPrs	HCLK clock source selection 000: SystemClk 001: SystemClk / 2 010: SystemClk / 4 011: SystemClk / 8 100: SystemClk / 16 101: SystemClk / 32 110: SystemClk / 64 111: SystemClk / 128
2:0	ClkSrc	When CR2.EXTEN is 0, SystemClk clock source selection x0x: internal high-speed clock RCHCLKD x1x: internal low-speed clock RCL

Note:

– Every time you rewrite the value of CR0 / CR1 / CR2, you need to write 0x5A5A and 0xA5A5 to CR3 in sequence. Such steps can effectively prevent misoperation of CR0/CR1/CR2 registers.

3.5.2 System Control Register 1 (CR1)

Offset address: 0x004

Reset value: 0x0000 0200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							GTIM CFG	Reserved				RESI O	Wake UpClk	Lock UpEn	SWD IO
							RW					RW	RW	RW	RW

Bit	Marking	Functional description
31:9	Reserved	Internally reserved bit, do not change its value.
8	GTIMCFG	GTIM/BTIM3-5 timer function configuration 0: GTIM is valid, BTIM3/4/5 is invalid 1: BTIM3/4/5 is valid, GTIM is invalid
7:4	Reserved	reserved bit
3	RESIO	RESET pin function configuration 1: The RESET pin is configured as an IO input pin and no longer has a reset function 0: The RESET pin is configured as a reset function, without IO input function
2	WakeUpClk	When DeepSleep wakes up, the source configuration of the system clock 1: Switch the system clock source to RCHCLKD, and keep the original system clock source enabled 0: keep the original system clock source
1	LockUpEn	Cortex-M0+ LockUp function configuration 0: off 1: Enable Note: If this function is enabled, the CPU will reset the MCU when it reads an invalid command.
0	SWDIO	SWD port function configuration 0: PC07 and PD01 are used as SWD port, GPIO function is not available. 1: PC07 and PD01 are used as GPIO ports, SWD function is not available.

Note:

– Every time you rewrite the value of CR0 / CR1 / CR2, you need to write 0x5A5A and 0xA5A5 to CR3 in sequence. Such steps can effectively prevent misoperation of CR0/CR1/CR2 registers.

3.5.3 System Control Register 2 (CR2)

Offset address: 0x008

Reset value: 0x0000 0081

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

Bit	Marking	Functional description
31:6	Reserved	Keep
5	EXTCLKEN	SystemClk clock source selection 0: SystemClk source is determined by CR0.ClkSrc 1: SystemClk comes from the clock input by the PA1 pin Note: Before enabling this control bit, make sure that PA1 has been set as a digital input and a clock signal has been input
4:3	Reserved	Keep
2	RCLEN	Internal low-speed clock RCL enable control 0: off 1: Enable Note: When the system enters DeepSleep, the low-speed clock remains the previous state.
1	Reserved	Keep
0	RCHEN	Internal high-speed clock RCH enable control 0: off 1: Enable Note: When the system enters DeepSleep, this high-speed clock will be automatically turned off.

Note:

– Every time you rewrite the value of CR0 / CR1 / CR2, you need to write 0x5A5A and 0xA5A5 to CR3 in sequence. Such steps can effectively prevent misoperation of CR0/CR1/CR2 registers.

3.5.4 System Control Register 3 (CR3)

Offset address: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR3															
WO															

Bit	Marking	Functional description
31:16	Reserved	reserved bit
15:0	CR3	Register CR0 / CR1 / CR2 write protection register, write 0x5A5A, 0xA5A5 to CR3 in sequence, the write protection of CR0 / CR1 / CR2 can be released; when CR0 / CR1 / CR2 is written, the write protection will be reset automatically. Before writing data to CR0 / CR1 / CR2 each time, it is necessary to write 0x5A5A and 0xA5A5 sequences to CR3 in sequence.

3.5.5 RCH Control Register (RCH)

Offset address: 0x010

Reset value: 0x0000 0B47

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

Bit	Marking	Functional description
31:16	Reserved	reserved bit
15	Stable	RCH clock stable flag bit 1: It means that RCH is stable and can be used by internal circuits 0: Represents that RCH is not stable and cannot be used by internal circuits
14:13	Reserved	reserved bit
12:9	DIV	RCHCLKD output frequency and RCH oscillation frequency division ratio configuration 0000: RCHCLKD = RCH / 2 0001: RCHCLKD = RCH / 4 0010: RCHCLKD = RCH / 6 0011: RCHCLKD = RCH / 8 0100: RCHCLKD = RCH / 10 0101: RCHCLKD = RCH / 12 0110: RCHCLKD = RCH / 14 0111: RCHCLKD = RCH / 16 1000: RCHCLKD = RCH / 1
8:0	TRIM	Clock frequency adjustment, changing the value of this register can adjust the oscillation frequency of RCH. Every time the register value increases by 1, the oscillation frequency of RCH increases by about 0.2%, and the total adjustment range is 44 - 48MHz. The calibration values of 44.24MHz and 48MHz have been saved in Flash, and the precise frequency can be obtained by reading out the calibration values in Flash and writing them into SysCtrl.RCH.TRIM. 44.24M calibration value address: 0x00100DF0 - 0x00100DF1 48.00M calibration value address: 0x00100DF2 - 0x00100DF3

3.5.6 RCL Control Register (RCL)

Offset address: 0x018

Reset value: 0x0000 033F

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

Bit	Marking	Functional description
31:16	Reserved	reserved bit
15	Stable	RCL clock stable flag bit 1: Represents that RCL is stable and can be used by internal circuits 0: Represents that RCL is not yet stable and cannot be used by internal circuits
14:12	Reserved	reserved bit
11:10	WaitCycle	Internal low-speed clock RCL stabilization time selection 11: 256 cycles 10: 64 cycles 01: 16 cycles 00: 4 cycles
9:0	TRIM	Internal low-speed clock frequency adjustment, 2 sets of frequency calibration values are stored in Flash. Accurate frequency can be obtained by reading out the calibration value in Flash and writing it into RCL.TRIM. 32.8K calibration value address: 0x00100DF4 - 0x00100DF5 38.4K calibration value address: 0x00100DF6 - 0x00100DF7

3.5.7 On -chip Peripheral Clock Control Register 0 (PeriClkEn0)

Reset value: 0x8080_0000

Offset address: 0x030

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IWDT	Reserved			ATIM3	Res.	Res.	Reserved		SPI	Res.	I2C	LPUART1	LPUART0	Reserved	
RW				RW		RW			RW		RW	RW	RW		

Bit	Marking	Functional description
31	FLASH	FLASH module configuration clock enable control 1: FLASH register can be read and written 0: FLASH register cannot be read or written
30:22	Reserved	Keep
21	CTRIM	CTRIM module configuration clock enable control 1: Enable 0: off
20:17	Reserved	Keep
16	ADC	ADC module configuration clock and working clock enable control 1: Enable 0: off
15	IWDT	IWDT module configuration clock enable control 1: Enable 0: off
14:12	Reserved	reserved bit
11	ATIM3	ATIM3 module configuration clock and working clock enable control 1: Enable 0: off
10:7	Reserved	reserved bit
6	SPI	SPI module configuration clock and working clock enable. 1: Enable 0: off
5	Reserved	reserved bit
4	I2C	I2C module configuration clock and working clock enable control 1: Enable 0: off
3	LPUART1	LPUART1 module configuration clock enable control 1: Enable 0: off
2	LPUART0	LPUART0 module configuration clock enable control 1: Enable 0: off
1:0	Reserved	reserved bit

3.5.8 On -chip Peripheral Clock Control Register 1 (PeriClkEn1)

Reset value: 0x0000_0000

Offset address: 0x034

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	GTIM	Reserved			WWDT	Reserved						PD	PC	PB	PA
	R/W				R/W							R/W	R/W	R/W	R/W

Bit	Marking	Functional description
31:15	Reserved	reserved bit
14	GTIM	GTIM/BTIM3-5 module configuration clock and working clock enable control 1: Enable 0: off
13:11	Reserved	reserved bit
10	WWDT	WWDT module configuration clock and working clock enable control 1: Enable 0: off
9:4	Reserved	reserved bit
3	PD	PD port configuration clock clock enable control 1: Enable 0: off
2	PC	PC port configuration clock clock enable control 1: Enable 0: off
1	PB	PB port configuration clock clock enable control 1: Enable 0: off
0	PA	PA port configuration clock clock enable control 1: Enable 0: off

3.5.9 Debug mode module work status control register (DebugActive)

Reset value 0x0001_4980 (this register setting works only in SWD debug mode)

Offset address: 0x4000_2044

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	GTIM	Reserved		ATIM3	Reserved		IWDT	Ctrim	Reserved						
	RW			RW			RW	RW							

Bit	Marking	Functional description
31:17	Reserved	Keep
16	WWDT	When debugging, the WWDT counting function configures the 1: In the SWD debugging interface, suspend the WWDT counting function 0: In the SWD debugging interface, WWDT counts normally
15	Reserved	Keep
14	GTIM	When debugging, the GTIM/BTIM3-5 count function is configured 1: In the SWD debugging interface, suspend the GTIM/BTIM3 - BTIM5 counting function 0: Under the SWD debugging interface, GTIM / BTIM3 - BTIM5 normal counting function
13:12	Reserved	Keep
11	ATIM3	When debugging, the ATIM3 counting function is configured 1: In the SWD debugging interface, suspend the counting function of ATIM3 0: In the SWD debugging interface, ATIM3 counts normally
10:9	Reserved	Keep
8	IWDT	When debugging, the IWDT count function configures the 1: In the SWD debugging interface, suspend the IWDT counting function 0: In the SWD debugging interface, the IWDT counts normally
7	CTRIM	When debugging, the CTRIM count function configures the 1: In the SWD debugging interface, suspend the CTRIM counting function 0: Under the SWD debugging interface, CTRIM counts normally
6:0	Reserved	Keep

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.

- Digital area power-on power-off reset POR
- External Reset PAD, low level is reset signal
- IWDT reset
- WWDT reset
- LVD low voltage reset
- Cortex-M0+ SYSRESETREQ software reset
- Cortex-M0+ LOCKUP hardware reset

Each reset source is indicated by a corresponding reset flag; when the reset is completed, query the ResetFlag register to obtain the last reset source. Whenever the query of the ResetFlag register is completed, the 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.

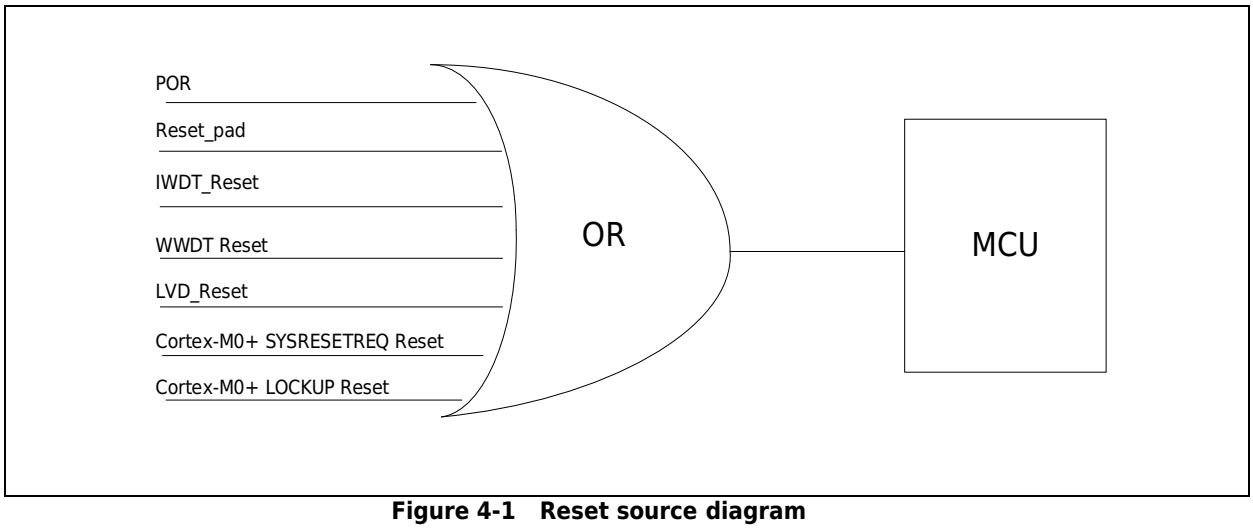


Figure 4-1 Reset source diagram

4.1.1 Power-on and power-off reset POR

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 (typically 1.65V), a POR5V signal will be generated; when the VCC area is powered off, when the VCC voltage is lower than the BOR threshold voltage (typically 1.5V), will generate a POR5V signal.

When the Vcore area is powered on, when the Vcore voltage is lower than the POR threshold voltage, a POR15V signal will be generated; when the Vcore area is powered off, when the Vcore voltage is lower than the BOR threshold voltage, a POR15V signal will be generated.

Both the POR5V signal and the POR15V signal will reset the registers of the chip to the initialization state.

4.1.2 External reset pin reset

A system reset will be 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 20uS (typical value), therefore, the low level signal added to the reset pin must be greater than 20uS to ensure reliable chip reset.

4.1.3 WDT reset

Watchdog reset, please refer to the IWDT / WWDT chapter description.

4.1.4 LVD low voltage reset

For LVD reset, please refer to the LVD chapter.

4.1.5 Cortex-M0+ SYSRESETREQ software reset

Cortex-M0+ software reset

4.1.6 Cortex-M0+ LOCKUP reset

When Cortex-M0+ encounters a serious abnormality, it stops its PC pointer at the current address and locks itself to reset the entire chip after several clock cycles delay.

4.2 Register

4.2.1 Reset flag register (Reset_Flag)

Reset value: 0x0000 0003

Address: 0x40002040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RSTB	SysReq	Lockup	WWDT	IWDT	LVD	Vcore	Vcc
								RW0	RW0	RW0	RW0	RW0	RW0	RW0	RW0

Bit	Marking	Functional description
31:8	Reserved	reserved bit
7	RSTB	RESETB port reset flag 1: A port reset has occurred 0: No port reset has occurred Write 0 to clear, write 1 to have no effect
6	SysReq	Cotrex-M0+ CPU software reset flag 1: A Cortrex-M0+ CPU software reset has occurred 0: No Cotrex-M0+ CPU software reset occurred Write 0 to clear, write 1 to have no effect
5	Lockup	Cotrex-M0+ CPU Lockup reset flag 1: A Cortrex-M0+ CPU Lockup reset has occurred 0: No Cortrex-M0+ CPU Lockup reset occurred Write 0 to clear, write 1 to have no effect
4	WWDT	WWDT reset flag, needs software initialization and clearing 1: A WWDT reset has occurred 0: No WWDT reset occurred Write 0 to clear, write 1 to have no effect
3	IWDT	IWDT reset flag 1: IWDT reset has occurred 0: No IWDT reset occurred Write 0 to clear, write 1 to have no effect
2	LVD	LVD reset flag 1: LVD reset has occurred 0: No LVD reset occurred Write 0 to clear, write 1 to have no effect
1	Vcore	Vcore domain reset flag 1: Vcore domain reset has occurred 0: Vcore domain occurred without reset Write 0 to clear, write 1 to have no effect
0	Vcc	Vcc power domain reset flag 1: VCC power domain reset has occurred 0: No reset of VCC power domain occurred Write 0 to clear, write 1 to have no effect

Note: The reset value is 0x03 at power-on reset. After the program judges the reset source this time, this register should be cleared, so that the reset source can be judged after the next reset occurs.

4.2.2 On -chip Peripheral Reset Control Register 0 (PeriReset0)

Reset value: 0x0021_0ADC

Address: 0x40002038

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.				ATIM3	Res.				SPI	Res.	I2C	LPUART1	LPUART0	Res.	
				RW					RW		RW	RW	RW		

Bit	Marking	Functional description
31:22	Res.	reserved bit
21	CTrim	CTRIM module reset control 1: The module is working normally 0: The module is in reset state
20:17	Res.	reserved bit
16	ADC	ADC module reset control 1: The module is working normally 0: The module is in reset state
15:12	Res.	reserved bit
11	ATIM3	ATIM3 module reset control 1: The module is working normally 0: The module is in reset state
10:7	Res.	reserved bit
6	SPI	SPI module reset control 1: The module is working normally 0: The module is in reset state
5	Res.	reserved bit
4	I2C	I2C module reset control 1: The module is working normally 0: The module is in reset state
3	LPUART1	LPUART1 module reset control 1: The module is working normally 0: The module is in reset state
2	LPUART0	LPUART0 module reset control 1: The module is working normally 0: The module is in reset state
1:0	Res.	reserved bit

4.2.3 On -chip Peripheral Reset Control Register 1 (PeriReset1)

Reset value: 0x0000 440F

Address: 0x4000203C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	GTIM	Reserved			WWDT	Reserved						PD	PC	PB	PA
	R/W				RW							R/W	R/W	R/W	R/W

Bit	Marking	Functional description
31:15	Reseved	reserved bit
14	GTIM	GTIM/BTIM3-5 module reset control 1: The module is working normally 0: The module is in reset state
13:11	Reseved	reserved bit
10:	WWDT	WWDT module reset control 1: The module is working normally 0: The module is in reset state
9:4	Reseved	reserved bit
3	PD	PD port reset control 1: The module is working normally 0: The module is in reset state
2	PC	PC port reset control 1: The module is working normally 0: The module is in reset state
1	PB	PB port reset control 1: The module is working normally 0: The module is in reset state
0	PA	PA port reset control 1: The module is working normally 0: The module is in reset state

5 Interrupt Controller (NVIC)

5.1 Overview

The Cortex-M0+ processor has a built-in nested vectored interrupt controller (NVIC), which supports up to 32 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-M0+ 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
…	…	…	…
47	Interrupt #31	programmable	External Interrupt #31

This chapter only introduces the 32 external interrupt requests of the processor (interrupt #0 to interrupt #31) 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 2 bits wide, and uses the highest two bits of the interrupt priority register, and each register occupies 1 byte (8 bits). Under this setting, the available priorities are 0x00 (highest), 0x40, 0x80 and 0xc0 (lowest).

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

Figure 5-1 Only the upper two 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-M0+ processor wants to process an interrupt service request, it needs to first determine the start address of the exception handling, and the required information is called a vector table, as shown in Figure 5-2. 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
0x0000004C	Interrupt #3 vector	19
0x00000048	Interrupt #2 vector	18
0x00000044	Interrupt #1 vector	17
0x00000040	Interrupt #0 vector	16
0x0000003C	SysTick vector	15
0x00000038	PendSV vector	14
0x00000034	unused	13
0x00000030	unused	12
0x0000002C	SVC vector	11
0x00000028	unused	10
0x00000024	unused	9
0x00000020	unused	8
0x0000001C	unused	7
0x00000018	unused	6
0x00000014	unused	5
0x00000010	unused	4
0x0000000C	hardware error exception	3
0x00000008	NMI vector	2
0x00000004	reset vector	1
0x00000000	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-M0+ 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 5-3:

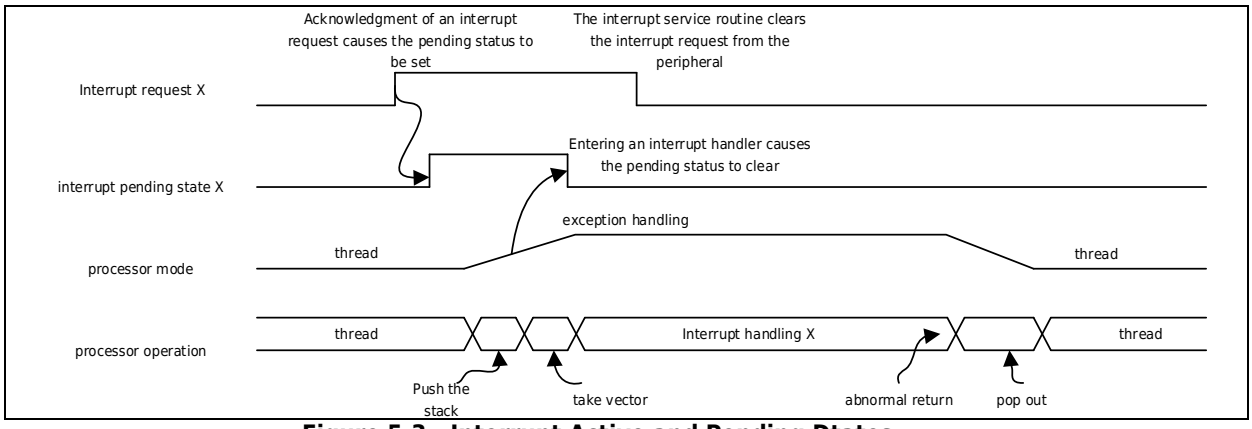


Figure 5-3 Interrupt Active and Pending Dtates

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 5-4:

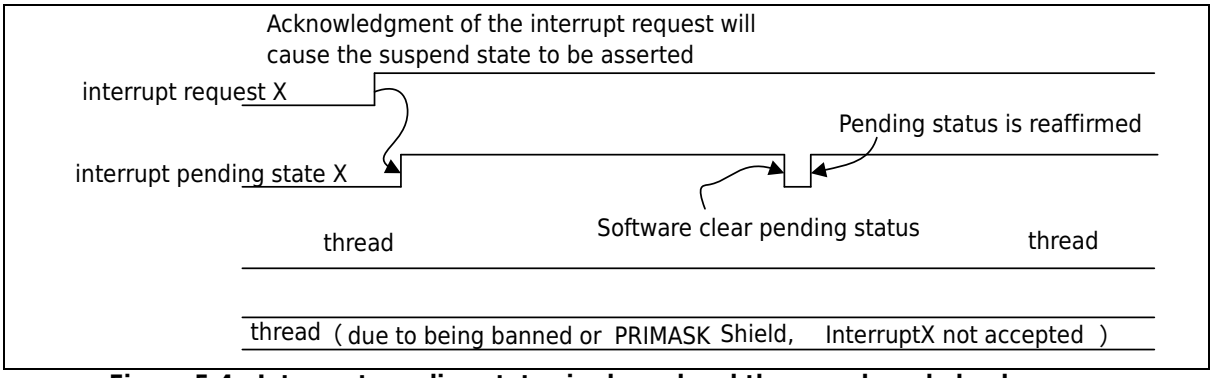


Figure 5-4 Interrupt pending status is cleared and then reacknowledged

If the interrupt request generated by the peripheral is not cleared when the exception is handled, the suspended state will be activated again after the exception returns, so that the interrupt service routine will be executed again. The process is shown in the Figure 5-5:

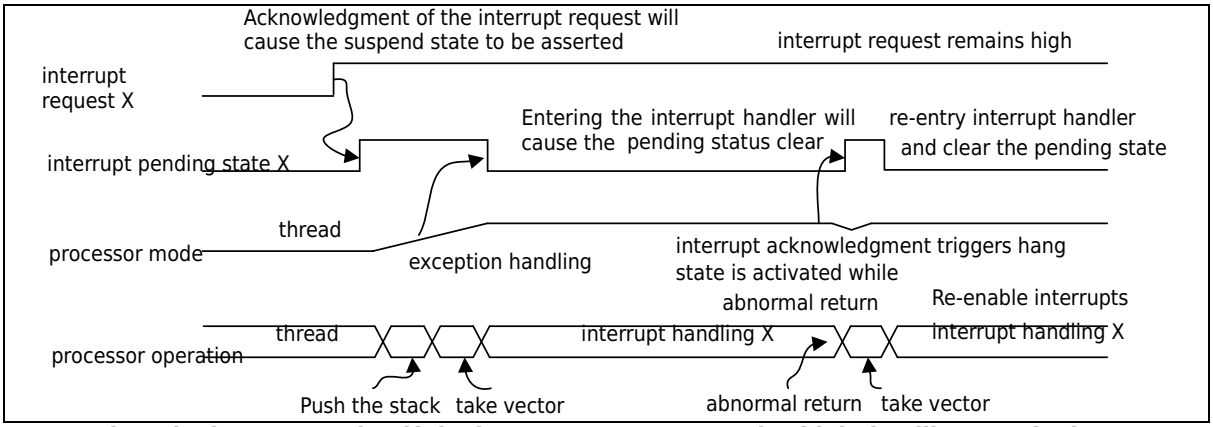


Figure 5-5 When the interrupt exits, if the interrupt request remains high, it will cause the interrupt processing to be executed again

If a peripheral interrupt request is generated during the execution of the interrupt service routine, the request will be regarded as a new interrupt request, and after this interrupt exits, the interrupt service routine will be executed again. The process is shown in theFigure 5-6:

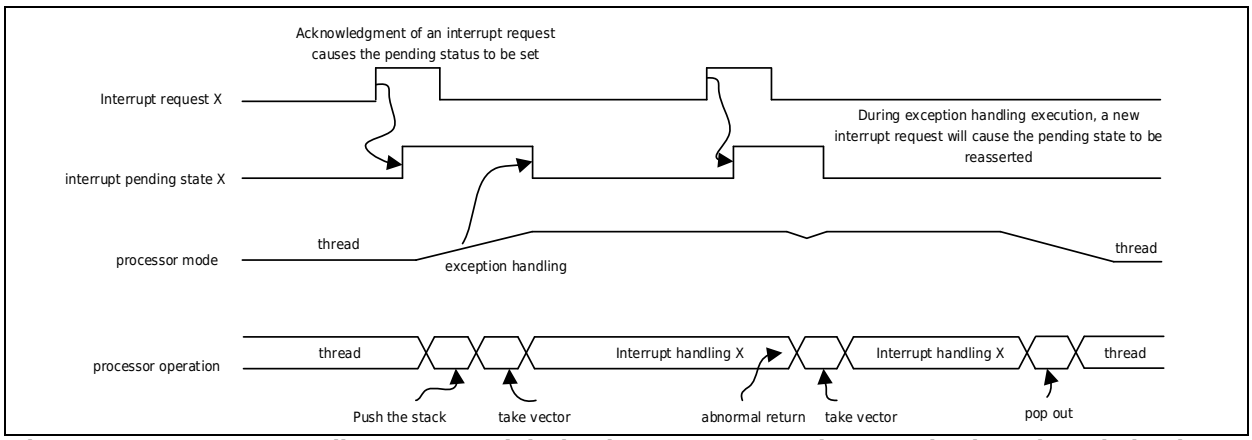


Figure 5-6 Interrupt pending generated during interrupt processing can also be acknowledged

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-M0+ processor supports up to 32 external interrupts, and in this system, there are 15 external interrupt sources, so some external interrupts are multiplexed on the same NVIC interrupt input, where NMI (non-maskable interrupt) and did not use. The corresponding relationship between all external interrupt sources of this system and NVIC interrupt input is shown in the following table:

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

NVIC interrupt input	external interrupt source	Active mode	sleep mode	DeepSleep mode
Interrupt #0	PORTA	✔	✔	✔
Interrupt #1	PORTB	✔	✔	✔
Interrupt #2	PORTC	✔	✔	✔
Interrupt #3	PORTD	✔	✔	✔
Interrupt #4	-	-	-	-
Interrupt #5	ATIM3	✔	✔	-
Interruption #6 - #7	-	-	-	-
Interrupt #8	LPUART0	✔	✔	✔
Interrupt #9	LPUART1	✔	✔	✔
Interrupt #10	SPI	✔	✔	-
Interrupt #11	-	-	-	-

Interrupt #12	I2C	✔	✔	-
Interruption #13 - #14	-	-	-	-
Interrupt #15	GTIM / BTIM3 - 5	✔	✔	-
Interruption #16 - #21	-	-	-	-
Interrupt #22	IWDT	✔	✔	✔
	WWDT	✔	✔	
Interrupt #23	-	-	-	-
Interrupt #24	ADC	✔	✔	-
Interruption #25 - #27	-	-	-	-
Interrupt #28	LVD	✔	✔	-
Interrupt #29	-	-	-	-
Interrupt #30	FLASH	✔	✔	-
Interrupt #31	CTRIM	✔	✔	✔

Note:

– Because some module interrupts are reused 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 diagram

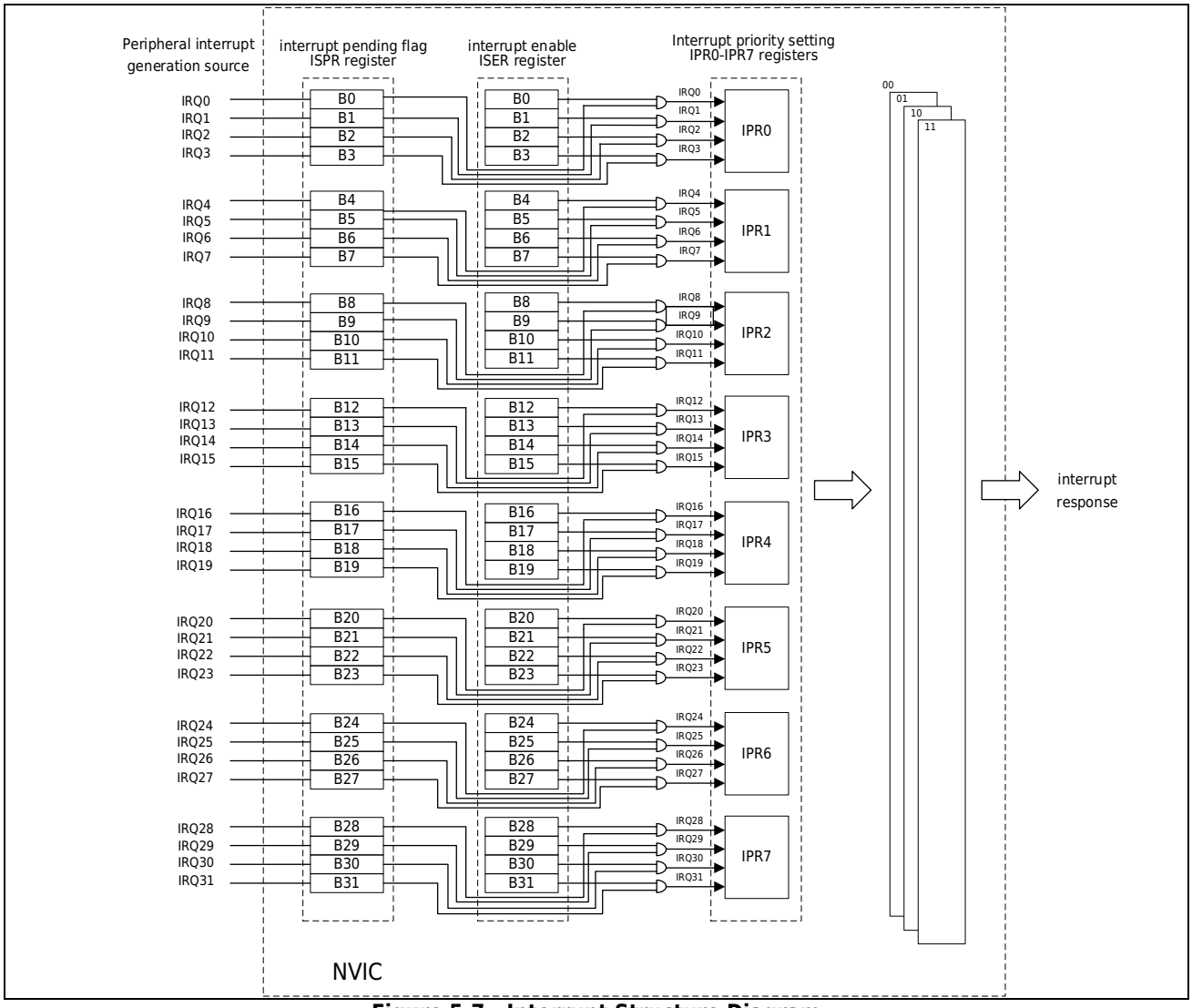


Figure 5-7 Interrupt Structure Diagram

The block diagram of the interrupt structure of the system is shown in the Figure 5-7. A few points to note:

- The respective interrupt enables of the peripheral interrupt sources are not marked in the figure, and only the logic block diagram of the interrupt signal after the peripheral interrupt is generated is included here.
- 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 register SCS_SETENA is set or not, the interrupt pending register SCS_SEPEND will be set, indicating that the corresponding peripheral interrupt source has an interrupt.
- Only when the interrupt enables register NVIC_ISER 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 register NVIC_ISPR is automatically cleared by the hardware.
- The interrupt priority registers NVIC_IPR0-NVIC_IPR7 set the priority of 32 interrupt sources, 00 has the highest priority and 11 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 enables

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-M0+ processor supports up to 32 interrupt sources, and each interrupt source corresponds to an interrupt enable bit and a clear enable bit. In this way, there are 32 -bit interrupt enable register NVIC_ISER and 32 -bit interrupt request clear register NVIC_ICER. If you want to enable an interrupt, set the corresponding bit of the NVIC_ISER register to 1. If you want to clear an interrupt, set the corresponding bit in the NVIC_ICER register.

Note that the interrupt enable mentioned here is only for the processor NVIC. Whether the interrupt of each peripheral is generated or not is determined by the interrupt control register of the peripheral, and has nothing to do with NVIC_ISER and NVIC_ICER.

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.

You can access or modify the interrupt pending status by manipulating the interrupt setting pending NVIC_ISPR and interrupt clearing pending NVIC_ICPR registers. The Interrupt Pending Status Register allows software to trigger interrupts.

5.8.4 NVIC interrupt priority

Setting the NVIC_IPR0-NVIC_IPR7 registers determines the priority of SCS_IRQ0 -SCS_IRQ32. 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 result of this situation is unpredictable and is not supported by the Cortex-M0+ processor.

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 SCS_PRIMASK. Only 1 bit of the special register SCS_PRIMASK is valid, and it 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 SCS_PRIMASK is set to 1, the current priority of the processor is reduced to 0 (the highest priority that can be assigned).

The SCS_PRIMASK register can be programmed in a variety of ways. Using assembly language, the MS R instruction can be used to set and clear the SCS_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

Base address: 0xE000 E000

Register	Offset address	Description
NVIC_ISER	0x100	Interrupt Enable Setting Register
NVIC_ICER	0x180	Interrupt Enable Clear Register
NVIC_ISPR	0x200	Interrupt Pending Status Setting Register
NVIC_ICPR	0x280	Interrupt Pending Status Clear Register
NVIC_IPR0	0x400	Interrupt Priority Register 0
NVIC_IPR1	0x404	Interrupt Priority Register 1
NVIC_IPR2	0x408	Interrupt Priority Register 2
NVIC_IPR3	0x40C	Interrupt Priority Register 3
NVIC_IPR4	0x410	Interrupt Priority Register 4
NVIC_IPR5	0x414	Interrupt Priority Register 5
NVIC_IPR6	0x418	Interrupt Priority Register 6
NVIC_IPR7	0x41C	Interrupt Priority Register 7
PRIMASK	-	Interrupt Mask Special Register

5.9.1 Interrupt Enable Setting Register (NVIC_ISER)

Offset address: 0x100

Reset value: 0x0000 0000

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

Bit	Marking	Functional description
31:0	SETENA [31:0]	Set enable interrupt #0 to interrupt #31; write "1" set, write "0" invalid [0]: IRQ0 [1]: IRQ1 [2]: IRQ2 ..... [31]: IRQ31

5.9.2 Interrupt Enable Clear Register (NVIC_ICER)

Offset address: 0x180

Reset value: 0x0000 0000

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

Bit	Marking	Functional description
31:0	CLRENA	Clear Enable Interrupt #0 to Interrupt #31; Write "1" to clear, write "0" is invalid

5.9.3 Interrupt Pending Status Set Register (NVIC_ISPR)

Offset address: 0x200

Reset value: 0x0000 0000

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

Bit	Marking	Functional description
31:0	SETPEND	Set the pending status of interrupt #0 to interrupt #31; write "1" to set, write "0" to have no effect [0]: IRQ0 [1]: IRQ1 [2]: IRQ2 ..... [31]: IRQ31

5.9.4 Interrupt Pending Status Clear Register (NVIC_ICPR)

Offset address: 0x280

Reset value: 0x0000 0000

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

Bit	Marking	Description
31:0	CLRPEND	Clear pending status of Interrupt #0 to Interrupt #31; write "1" to clear, write "0" to have no effect [0]: IRQ0 [1]: IRQ1 [2]: IRQ2 ..... [31]: IRQ31

5.9.5 Interrupt Priority Register (NVIC_IPR0)

Offset address: 0x400

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPR0[31:16]															
RW															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IPR0[15:0]]															
RW															

Bit	Marking	Functional description
31:0	IPR0[31:0]	Priority of interrupt #0 to interrupt #3; [31:30]: Priority of interrupt #3 [23:22]: Priority of interrupt #2 [15:14]: Priority of interrupt #1 [7:6]: Priority of interrupt #0 Among them, 00 has the highest priority and 11 has the lowest priority

5.9.6 Interrupt Priority Register (NVIC_IPR1)

Offset address: 0x404

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPR1[31:16]															
RW															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IPR1[15:0]]															
RW															

Bit	Marking	Functional description
31:0	IPR1[31:0]	Priority of interrupt #4 to interrupt #7; [31:30]: Priority of interrupt #7 [23:22]: Priority of interrupt #6 [15:14]: Priority of interrupt #5 [7:6]: Priority of interrupt #4 Among them, 00 has the highest priority and 11 has the lowest priority

5.9.7 Interrupt Priority Register (NVIC_IPR2)

Offset address: 0x408

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPR2[31:16]															
RW															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IPR2[15:0]]															
RW															

Bit	Marking	Functional description
31:0	IPR2[31:0]	Priority of interrupt #8 to interrupt #11; [31:30]: Priority of interrupt #11 [23:22]: Priority of interrupt #10 [15:14]: Priority of interrupt #9 [7:6]: Priority of interrupt #8 Among them, 00 has the highest priority and 11 has the lowest priority

5.9.8 Interrupt Priority Register (NVIC_IPR3)

Offset address: 0x40C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPR3[31:16]															
RW															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IPR3[15:0]]															
RW															

Bit	Marking	Functional description
31:0	IPR3[31:0]	Priority of interrupt #12 to interrupt #15; [31:30]: Priority of interrupt #15 [23:22]: Priority of interrupt #14 [15:14]: Priority of interrupt #13 [7:6]: Priority of interrupt #12 Among them, 00 has the highest priority and 11 has the lowest priority

5.9.9 Interrupt Priority Register (NVIC_IPR4)

Offset address: 0x410

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPR4[31:16]															
RW															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IPR4[15:0]]															
RW															

Bit	Marking	Functional description
31:0	IPR4[31:0]	Priority of interrupt #16 to interrupt #19; [31:30]: Priority of interrupt #19 [23:22]: Priority of interrupt #18 [15:14]: Priority of interrupt #17 [7:6]: Priority of interrupt #16 Among them, 00 has the highest priority and 11 has the lowest priority

5.9.10 Interrupt Priority Register (NVIC_IPR5)

Offset address: 0x414

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPR5[31:16]															
RW															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IPR5[15:0]]															
RW															

Bit	Marking	Functional description
31:0	IPR5[31:0]	Priority of interrupt #20 to interrupt #23; [31:30]: Priority of interrupt #23 [23:22]: Priority of interrupt #22 [15:14]: Priority of interrupt #21 [7:6]: Priority of interrupt #20 Among them, 00 has the highest priority and 11 has the lowest priority

5.9.11 Interrupt Priority Register (NVIC_IPR6)

Offset address: 0x418

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPR6[31:16]															
RW															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IPR6[15:0]]															
RW															

Bit	Marking	Functional description
31:0	IPR6[31:0]	Priority of interrupt #24 to interrupt #27; [31:30]: Priority of interrupt #27 [23:22]: Priority of interrupt #26 [15:14]: Priority of interrupt #25 [7:6]: Priority of interrupt #24 Among them, 00 has the highest priority and 11 has the lowest priority

5.9.12 Interrupt Priority Register (NVIC_IPR7)

Offset address: 0x41C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPR7[31:16]															
RW															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IPR7[15:0]															
RW															

Bit	Marking	Functional description
31:0	IPR7[31:0]	Priority of interrupt #28 to interrupt #31; [31:30]: Priority of interrupt #31 [23:22]: Priority of interrupt #30 [15:14]: Priority of interrupt #29 [7:6]: Priority of interrupt #28 Among them, 00 has the highest priority and 11 has the lowest priority

5.9.13 Interrupt Mask Special Register (PRIMASK)

Offset address: ---

Reset value: 0x0000 0000

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

Bit	Marking	Functional description
31:1	Reserved	
0	PRIMASK	When set, 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.

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 is 5 bits, PB is 6 bits, PC is 5 bits, and PD is 6 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, except PA00
- Input mode: analog input, high-impedance input, pull-up 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 (GPIOx_IN) and on-chip peripherals
- Signal output: The source of the output signal is the data output register (GPIOx_OUT) or on-chip peripherals
- Atomic operations: bit operation registers (GPIOx_BSRR, GPIOx_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
- Double drive: PB04-05, PC03-04 of TSSOP20 and QFN20 have double drive capability

6.3 Functional description

6.3.1 Port Circuit Block Diagram

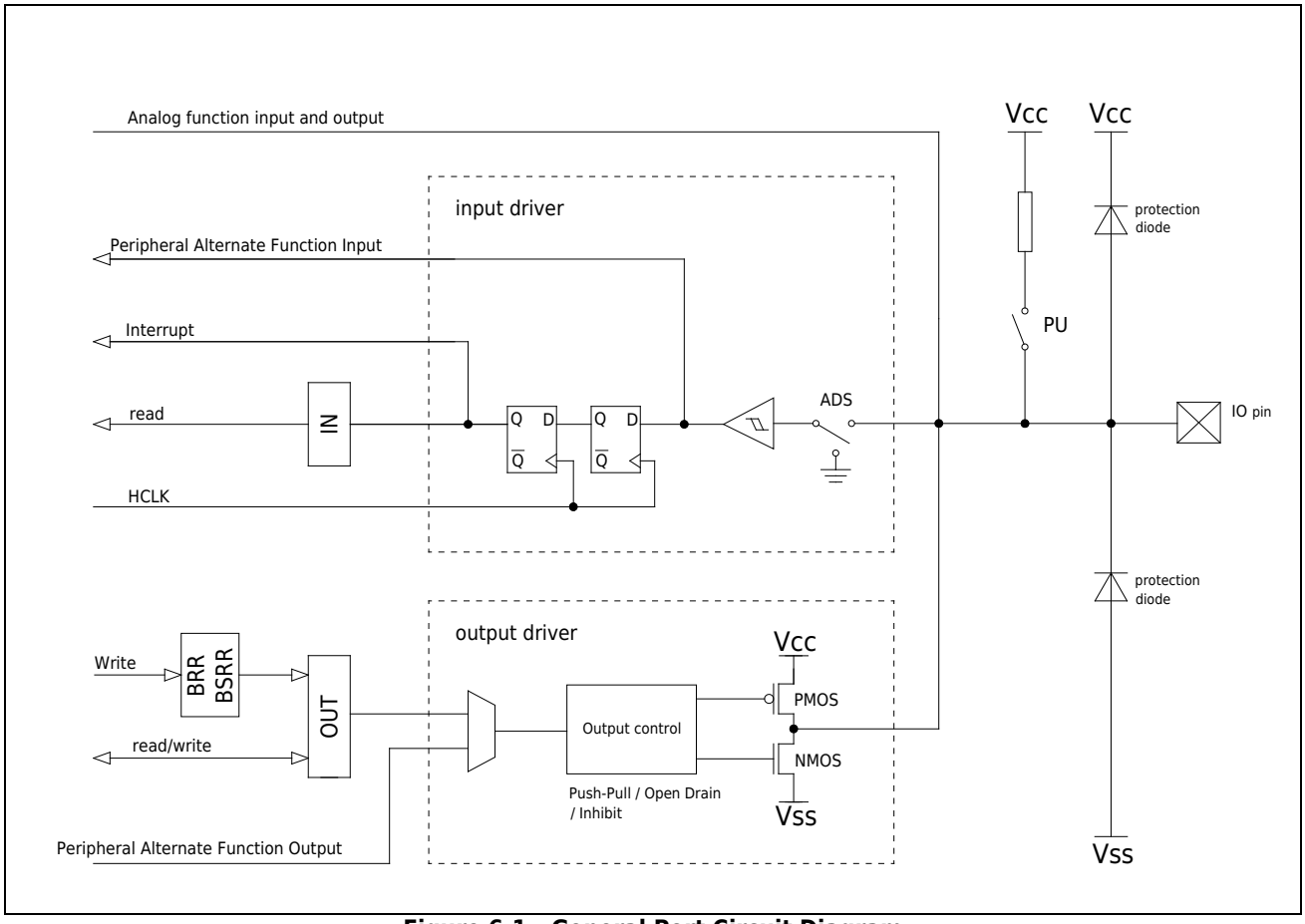


Figure 6-1 General Port Circuit Diagram

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	DIR	Open Drain	OUT	BRR.BR	BSRR.BR	BSRR.BS	PU
simulation	0	1	-	-	-	-	-	-	-	-
high impedance input	X	0	0	1	-	-	-	-	-	0
pull-up input	1	0	0	1	-	-	-	-	-	1
High impedance output	X	0	0	0	1	1	0	0	0	0
High impedance output	X	0	0	0	1	-	0	0	1	0
High impedance output	X	0	0	0	1	-	0	1	1	0
pull-up output	1	0	0	0	1	1	0	0	0	1
pull-up output	1	0	0	0	1	-	0	0	1	1
pull-up output	1	0	0	0	1	-	0	1	1	1
output 0	0	0	0	0	1	0	0	0	0	-
output 0	0	0	0	0	1	-	1	0	0	-
output 0	0	0	0	0	1	-	0	1	0	-
output 0	0	0	0	0	0	0	0	0	0	-
output 0	0	0	0	0	0	-	1	0	0	-
output 0	0	0	0	0	0	-	0	1	0	-
output 1	0	0	0	0	0	1	0	0	0	-
output 1	0	0	0	0	0	-	0	0	1	-
output 1	0	0	0	0	0	-	0	1	1	-
peripheral output	X	0	>0	0	0	-	-	-	-	-
Peripheral pull-up input	X	0	>0	1	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(PD01): digital input pull-up
- SWDIO(PC07): Digital input pull-up
- PC06: Digital input without pull-up

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 GPIOx_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 level cannot be output; the level of the read pin is always 0; there is no built-in pull-up and pull-down

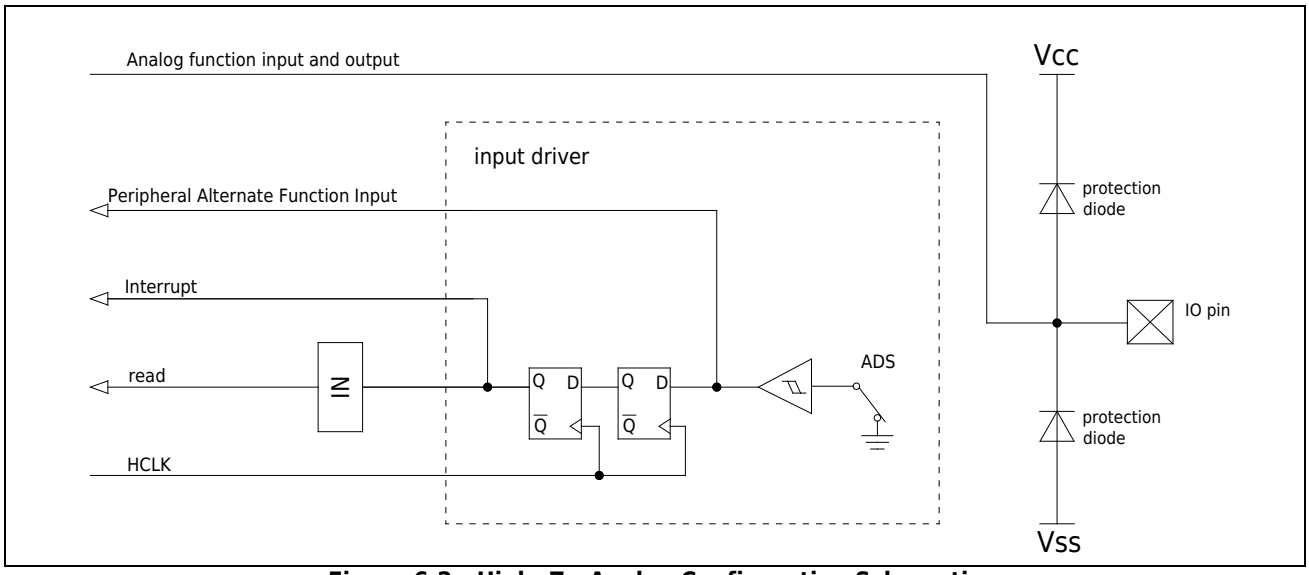


Figure 6-2 High- Z - Analog Configuration Schematic

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 GPIOx_AFRL to 0. The implementation method is to set the corresponding bit of the GPIOx_ADS register to 0, and set the corresponding bit of the GPIOx_AFRL register to 0. Under this configuration, the port has the following characteristics:

- Input mode: pull-up enabled

- Push-pull output: enable pull-up; support bit operation
- Open-drain output: pull-up can be enabled; bit manipulation is supported

Each GPIO has 4 control registers, and their functions are described as follows:

- GPIOx_ADS: Register configurable port as a digital port
- GPIOx_DIR: Register configurable port as input or output
- GPIOx_PU: Register to enable pull-ups for ports
- GPIOx_OpenDrain: Register configurable port output type as push-pull or open-drain

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

- GPIOx_IN: The register stores data input through I/O and can only be read
- GPIOx_OUT: The register stores the data to be output, which can be read / written

Note: The process of [read - modify – write] to the GPIOx_OUT register is not an atomic operation, and it is not recommended to use this method to modify the value of the GPIOx_OUT register.

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

- The BS bit of the GPIOx_BSRR register can set the corresponding bit of GPIOx_OUT to 1, and the BR bit of the register can clear the corresponding bit of GPIOx_OUT to 0
- The GPIOx_BRR register bit can clear the corresponding bit of GPIOx_OUT to 0

Note: The process of modifying the value of GPIOx_OUT through the bit operation register is an atomic operation. It is recommended to modify the value of the GPIOx_OUT register by bit operation.

6.3.5.1 input mode

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

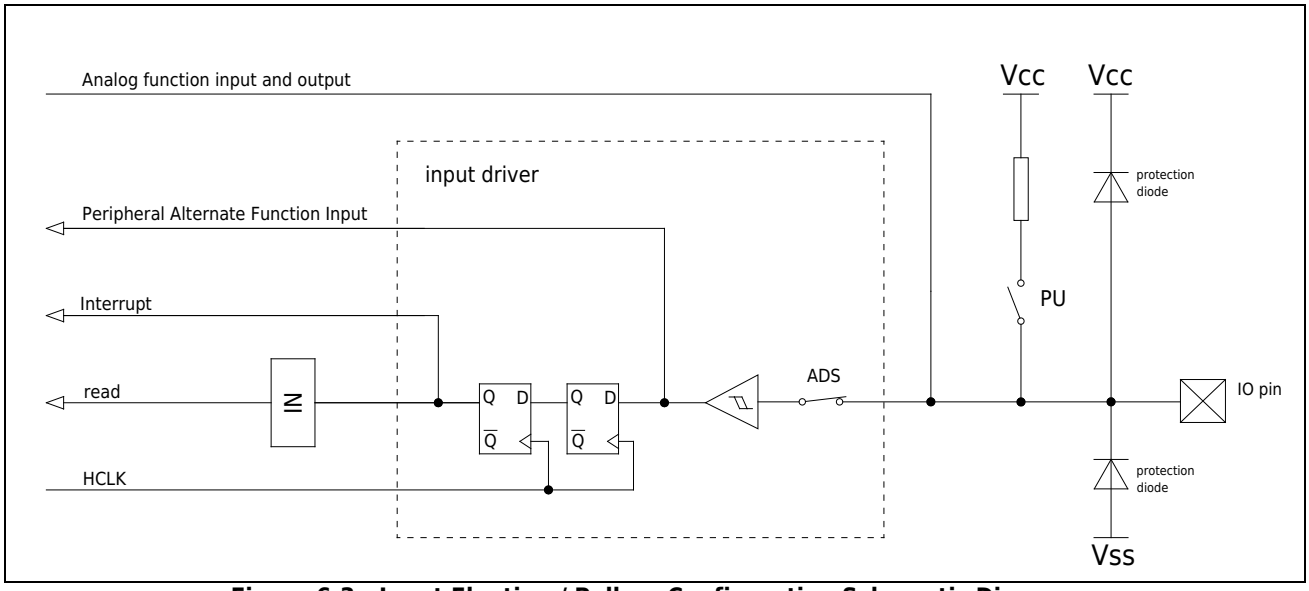


Figure 6-3 Input Floating / Pull-up Configuration Schematic Diagram

Each port can get the port pin level by reading the GPIOx_IN register. As shown in the Figure 6-4, each bit of the GPIOx_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 it also introduces Delay. The synchronization diagram for reading port pin data is as follows:

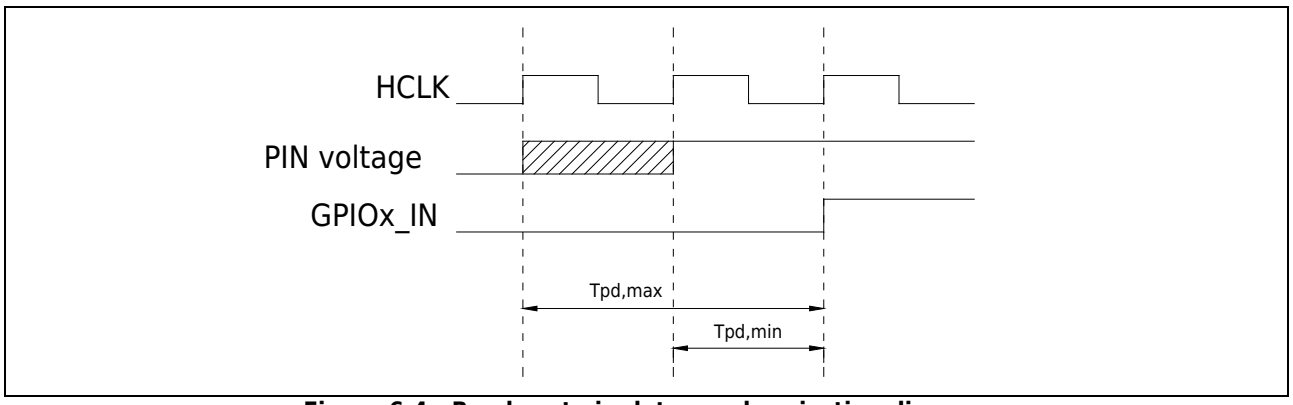


Figure 6-4 Read port pin data synchronization diagram

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. The data is then latched into the GPIOx_IN register on the rising edge of the system clock. The signal transition delay Tpd 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:

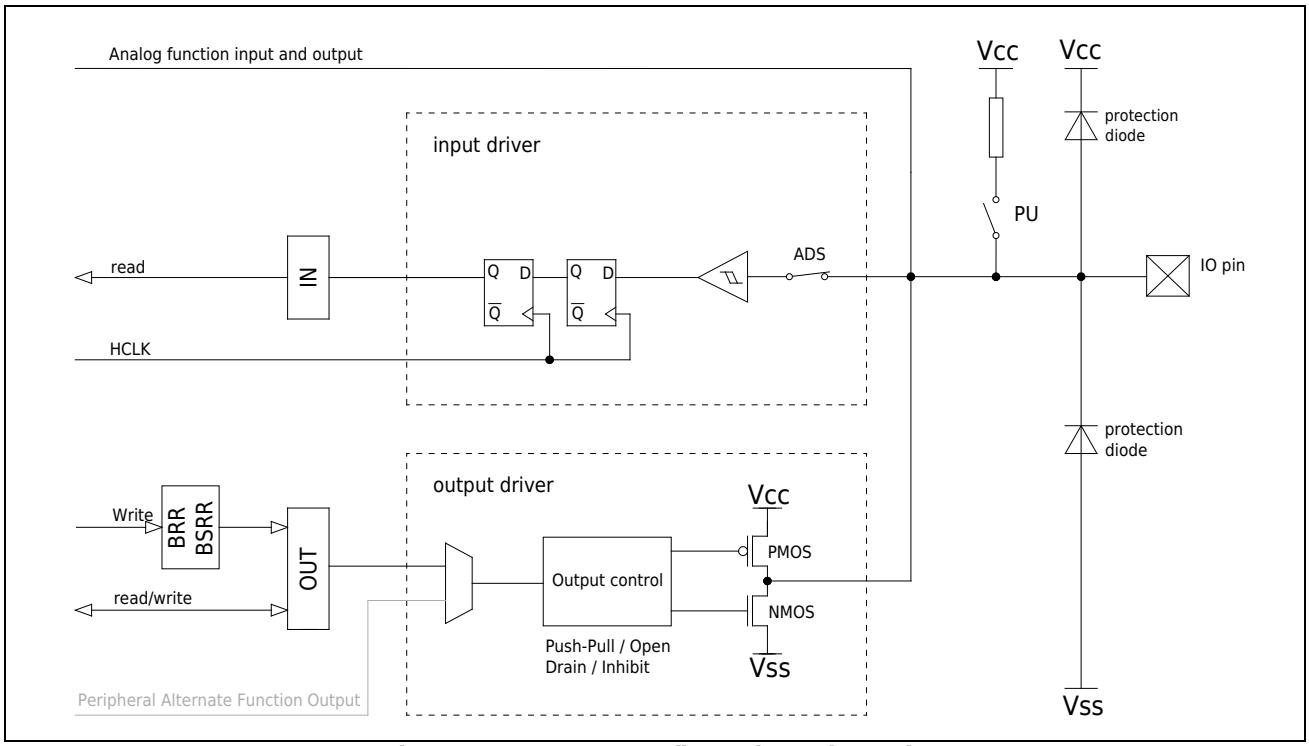


Figure 6-5 Output Configuration Schematic

Each port can control the output level of the IO port by writing data to GPIOx_BRR, GPIOx_BSRR or GPIOx_OUT.

GPIOx_BRR and GPIOx_BSRR are bit operation registers. The number of IOs in the same group of ports can be modified from 1 to 6 at a time, and the unmodified IOs in the same group of ports remain unchanged. Use these two registers to modify IOs The process is an atomic operation. The value written to the register GPIOx_OUT is the output value of the output PIN pin corresponding to each bit.

6.3.6 Port multiplexing function

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

Under this configuration, the port has the following characteristics:

- Input mode: pull-up can be enabled; external signals are input to on-chip peripherals; GPIOx_IN can read external signals

  Port interrupt can be generated
- Output mode:open-drain or push-pull selectable; pull-up can be enabled

  On-chip peripherals output signals to ports; port interrupts cannot be generated

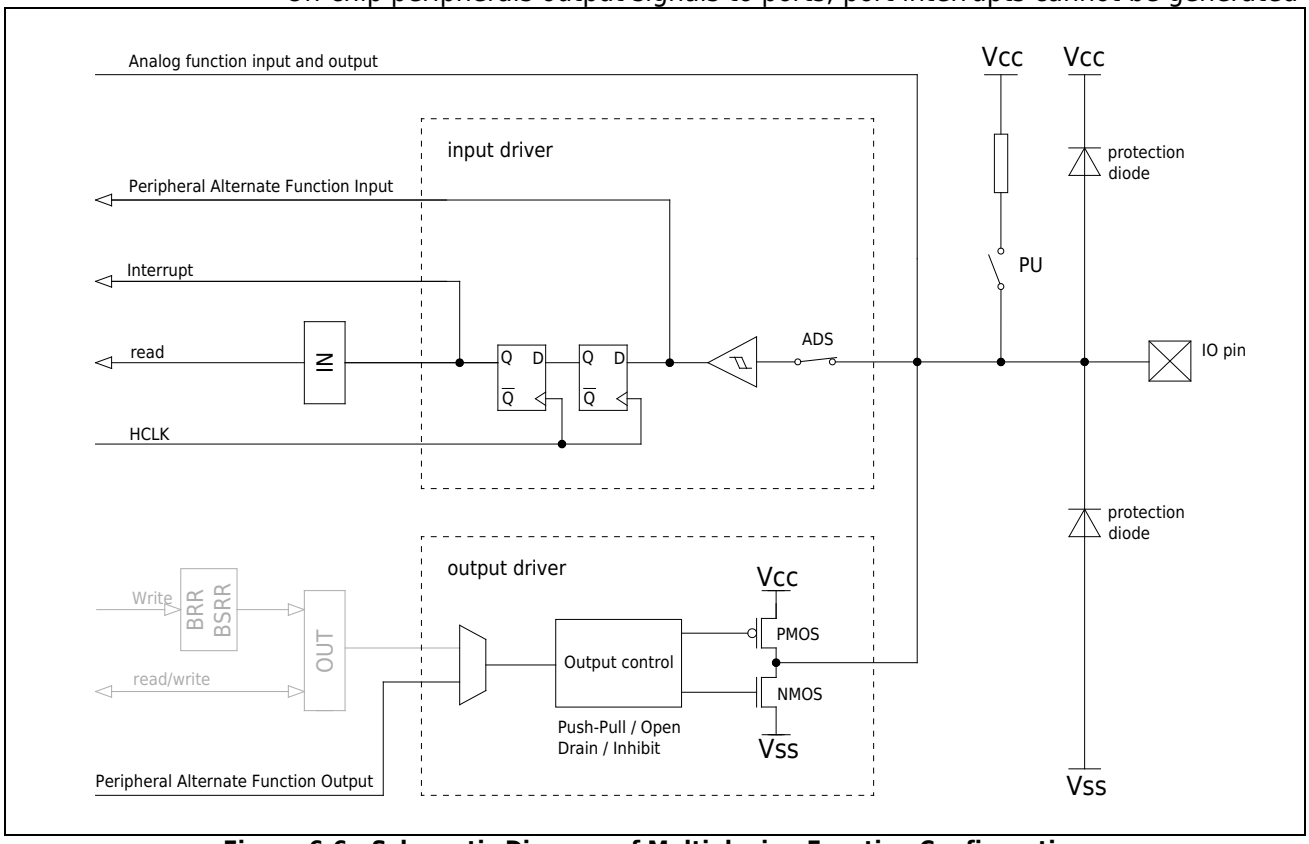


Figure 6-6 Schematic Diagram of Multiplexing Function Configuration

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

Pin name	digital function				
	AF0	AF1	AF2	AF3	AF4
PA01	GPIO	LPUART0_RXD	I2C_SDA	GTIM_TOGP	HCLK_OUT
PA02	GPIO	LPUART0_TXD	I2C_SCL	GTIM_TOGN	LVD_OUT
PA03	GPIO	GTIM_CH2	SPI_NSS	CTRIM_ETR CTRIM_TOG	
PA04	GPIO	GTIM_TOGP	LVD_OUT	SPI_NSS	
PB00	GPIO	ATIM3_CH0B	GTIM_CH3	LPUART0_RTS	
PB01	GPIO	ATIM3_CH1B	GTIM_CH2	LPUART0_CTS	
PB02	GPIO	ATIM3_CH2B	GTIM_CH1	LPUART1_RTS	
PB03	GPIO	ATIM3_ETR	GTIM_CH0	LPUART1_CTS	
PB04	GPIO	I2C_SCL	ATIM3_CH2B	LPUART0_CTS	
PB05	GPIO	I2C_SDA	ATIM3_BK	CTRIM_ETR CTRIM_TOG	
PC03	GPIO	ATIM3_CH2A	ATIM3_CH0B	LPUART0_RTS	
PC04	GPIO	ATIM3_CH0B	ATIM3_CH1B	TCLK_OUT	
PC05	GPIO	SPI_SCK	GTIM_CH3	ATIM3_ETR	
PC06	GPIO	SPI_MOSI	ATIM3_CH0A	LPUART1_RTS	
PC07	GPIO	SPI_MISO	ATIM3_CH1A	LPUART0_RXD	
PD01	GPIO	GTIM_ETR	ATIM3_BK	LPUART0_TXD	
PD02	GPIO	GTIM_CH2	ATIM3_CH2A	LPUART1_CTS	
PD03	GPIO	GTIM_CH1	ATIM3_CH1A	LPUART1_RXD	
PD04	GPIO	GTIM_CH0	LPUART1_TXD	SPI_SCK	
PD05	GPIO	LPUART1_TXD	ATIM3_CH0A	SPI_MISO	
PD06	GPIO	LPUART1_RXD	GTIM_ETR	SPI_MOSI	

6.3.7 port clock output

6.3.7.1 Output HCLK

Configure the GPIO register to output HCLK. The following example outputs HCLK from PA01:

Step1. Set GPIOA_ADS.PIN1 to 0, and configure PA01 as a digital port.

Step2. Set GPIOA_DIR.PIN1 to 0, configure PA01 as output mode.

Step3. Set GPIOA_AFRL.AFSEL1 to 4, and configure the multiplexing function of PA01 as HCLK.

6.3.7.2 Output TCLK

Configure the GPIO register to output RCH/RCL/PCLK/HCLK. The example below outputs RCH/8 from PC04:

Step1. Set GPIOC_ADS.PIN4 to 0, and configure PC04 as a digital port.

Step2. Set GPIOC_DIR.PIN4 to 0, configure PC04 as output mode.

Step3. Set GPIOC_AFRL.AFSEL4 to 3, and configure the multiplexing function of PC04 as TCLK.

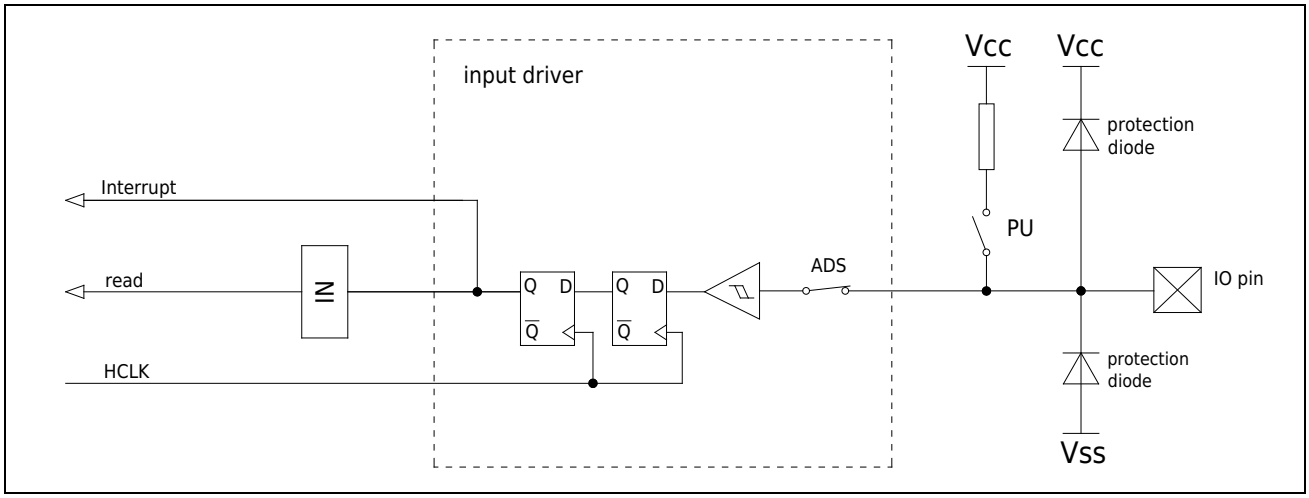
Step4. Set GPIOC_CR1.TCLKSRC to 0, select the TCLK clock source as RCH.

Step5. Set GPIOC_CR1.TCLKDIV to 1, output TCLK divided by 8.

6.3.8 port external interrupt

Each port has an interrupt function when the port is configured as a digital function (GPIOx_ADS is 0) and works in input mode (GPIOx_DIR is 1). 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: GPIOx_RISEIE, GPIOx_FALLIE, GPIOx_HIGHIE, GPIOx_LOWIE. When an interrupt is triggered, it can be determined which port triggered the interrupt by querying the interrupt status register (GPIOx_IFR), and the corresponding interrupt status flag bit can be cleared through the interrupt clear register (GPIOx_ICR).

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



6.4 Register description

6.4.1 register list

Base address:

GPIOA：0x4002 0C00

GPIOB：0x4002 0D00

GPIOC：0x4002 0E00

GPIOD：0x4002 0F00

Note:

- GPIOA has 5 effective bits (PA00-04), GPIOB has 6 effective bits (PB00-05), GPIOC has 5 effective bits (PC03-07), and GPIOD has 6 effective bits (PD01-06). For non-existing bits, the write operation is invalid, and the read value is always 0.
  Among them, 2 auxiliary function registers (CR1, CR4) are shared registers, GPIOA / GPIOB / GPIOC / GPIOD can be used and are valid for 4 ports, that is, 0x4002 0CA4, 0x4002 0DA4, 0x4002 0EA4 and 0x4002 0FA4 are the same register; 0x4002 0CB0, 0x4002 0D B0, 0x4002 0E B0 and 0x4002 0F B0 are the same register.
  In the table below x = A 、B 、C 、D

Table 6-2 List of GPIO registers

Register name	Offset address	Reset value
Port Digital-to-Analog Configuration Register (GPIOx_ADS)	0x00	(PA)0x0000 001E (PB)0x0000 003F (PC)0x0000 0038 (PD)0x0000 007C
Port Input Output Configuration Register (GPIOx_DIR)	0x04	(PA)0x0000 001F (PB)0x0000 003F (PC)0x0000 00F8 (PD)0x0000 007E
Port Output Type Register (GPIOx_OpenDrain)	0x08	0x0000 0000
Port pull-up register (GPIOx_PU)	0x18	(PA)0x0000 0001 (PB/C/D)0x0000 0000
Port Data Input Register (GPIOx_IN)	0x1C	0x0000 xxxx
Port Data Output Register (GPIOx_OUT)	0x20	0x0000 xxxx
Port Reset Register (GPIOx_BRR)	0x24	0x0000 0000
Port Set / Reset Register (GPIOx_BSRR)	0x28	0x0000 0000
Port Alternate Function Low Register (GPIOx_AFRL)	0x30	0x0000 0000
Port High Interrupt Enable Configuration Register (GPIOx_HIGHIE)	0x38	0x0000 0000
Port Low Interrupt Enable Configuration Register (GPIOx_LOWIE)	0x3C	0x0000 0000
Port Rising Edge Interrupt Enable Configuration Register (GPIOx_RISEIE)	0x40	0x0000 0000
Port Falling Edge Interrupt Enable Configuration Register (GPIOx_FALLIE)	0x44	0x0000 0000
Port Interrupt Status Register (GPIOx_IFR)	0x48	0x0000 0000
Port Interrupt Clear Register (GPIOx_ICR)	0x4C	(PA)0x0000 001F (PB)0x0000 003F (PC)0x0000 00F8 (PD)0x0000 007E
Port Miscellaneous Function Configuration Register 1 (GPIOx_CR1)	0xA4	0x0000 0000
Port Miscellaneous Function Configuration Register 4 (GPIOx_CR4)	0xB0	0x0000 0000

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

Address offset: 0x00

Reset value: (PA) 0x0000 001E

(PB) 0x0000 003F

(PC) 0x0000 0038

(PD) 0x0000 007C

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

Bit	Symbol	Functional description
31:8	Reserved	Keep
7:0	PIN	Port digital function or analog function control 1: Configure the port as an analog function 0: configure the port as a digital function Note: PA port valid bits [4:1], PA[0] are only used as digital ports, PB port valid bits [5:0], PC port valid bits [7:3], PD port valid bits [6:1]]

6.4.3 Port Direction Configuration Register (GPIOx_DIR)

Address offset: 0x04

Reset value: (PA)0x0000 001F

(PB)0x0000 003F

(PC)0x0000 00F8

(PD)0x0000 007E

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

Bit	Symbol	Functional description
31:8	Reserved	Keep
7:0	PIN	Port input and output direction control 1: Configure the port as an input 0: Configure the port as an output Note: PA port valid bits [4:1], PA[0] are input only, PB port valid bits [5:0], PC port valid bits [7:3], PD port valid bits [6:1]

6.4.4 Port Output Type Register (GPIOx_OpenDrain)

Address offset: 0x08

Reset value: 0x0000 0000

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

Bit	Symbol	Functional description
31:8	Reserved	Keep
7:0	PIN	Port output mode control bit 0: Push-pull output (reset value) 1: Open drain output Note: PA port valid bits [4:1], PB port valid bits [5:0], PC port valid bits [7:3], PD port valid bits [6:1]

6.4.5 Port pull-up register (GPIOx_PU)

Address offset: 0x18

Reset value: (PA)0x0000 0001

(PB/PC/PD)0x0000 0000

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

Bit	Symbol	Functional description
31:8	Reserved	Keep
7:0	PUx	Port upper resistor enable control 0: Disable pull-up resistor 1: Enable pull-up resistor Note: PA port valid bits [4:0], PA[0] will be pulled up automatically when it is used as RESET PAD, PB port valid bits [5:0], PC port valid bits [7:3], PD port valid bits [6]:1]

6.4.6 Port Data Input Register (GPIOx_IN)

Address offset: 0x1C

Reset value: 0x0000 XXXX

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

Bit	Symbol	Functional description
31:8	Reserved	Keep
7:0	PIN	port input data PA port valid bits [4:0], PB port valid bits [5:0], PC port valid bits [7:3], PD port valid bits [6:1]

6.4.7 Port Data Output Register (GPIOx_OUT)

Address offset: 0x20

Reset value: 0x0000 XXXX

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

Bit	Symbol	Functional description
31:8	Reserved	Keep
7:0	PIN	port output data Note: PA port valid bits [4:1], PB port valid bits [5:0], PC port valid bits [7:3], PD port valid bits [6:1]

6.4.8 Port Reset Register (GPIOx_BRR)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bit	Symbol	Functional description
31:8	Reserved	Keep
7:0	PIN	Port Bit Clear Control 0: Do not affect the corresponding bit of the GPIOx_OUT register 1: Set the corresponding bit of the GPIOx_OUT register to 0 Note: PA port valid bits [4:1], PB port valid bits [5:0], PC port valid bits [7:3], PD port valid bits [6:1]

6.4.9 Port Set Reset Register (GPIOx_BSRR)

Address offset: 0x28

Reset value: 0x00000000

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

Bit	Symbol	Functional description
31:24	Reserved	Keep
23:16	BR	Port Bit Clear Control 0: Do not affect the corresponding bit of the GPIOx_OUT register 1: Set the corresponding bit of the GPIOx_OUT register to 0 Note: PA port valid bits [20:17], PB port valid bits [21:16], PC port valid bits [23:19], PD port valid bits [22:17]
15:8	Reserved	Keep
7:0	BS	Port Bit Bit Control 0: Do not affect the corresponding bit of the GPIOx_OUT register 1: Set the corresponding bit of the GPIOx_OUT register to 1 Note: PA port valid bits [4:1], PB port valid bits [5:0], PC port valid bits [7:3], PD port valid bits [6:1] Note: When BR and BS occur at the same time, BS takes precedence

6.4.10 Port Alternate Function Low Register (GPIOx_AFRL)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL7				AFSEL6				AFSEL5				AFSEL4			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL3				AFSEL2				AFSEL1				AFSEL0			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Symbol	Functional description
31:0	AFSELy y = 0-7	Port multiplexing function control 0000: GPIO 0001: AF1 0010: AF2 0011: AF3 0100: AF4 0101-1111: reserved bits Note 1: For the specific functions of AF1 – C AF4, please refer to the chapter [Port multiplexing function] Note 2: PA port valid bits [19:4], PB port valid bits [23:0], PC port valid bits [31:12], PD port valid bits [27:4]

6.4.11 Port High Level Interrupt Enable Configuration Register (GPIOx_HIGHIE)

Address offset: 0x38

Reset value: 0x0000 0000

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

Bit	Marking	Functional description
31:8	Reserved	Keep
7:0	PIN	Port high level interrupt enable control 1: Enable the high-level interrupt of the corresponding port 0: Disable the high level interrupt of the corresponding port Note: PA port valid bits [4:0], PB port valid bits [5:0], PC port valid bits [7:3], PD port valid bits [6:1]

6.4.12 Port Low Level Interrupt Enable Configuration Register (GPIOx_LOWIE)

Address offset: 0x3C

Reset value: 0x0000 0000

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

Bit	Marking	Functional description
31:8	Reserved	Keep
7:0	PIN	Port low level interrupt enable control 1: Enable the low-level interrupt of the corresponding port 0: Disable the low-level interrupt of the corresponding port Note: PA port valid bits [4:0], PB port valid bits [5:0], PC port valid bits [7:3], PD port valid bits [6:1]

6.4.13 Port rising edge interrupt enable configuration register (GPIOx_RISEIE)

Address offset: 0x40

Reset value: 0x0000 0000

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

Bit	Marking	Functional description
31:8	Reserved	Keep
7:0	PIN	Port rising edge interrupt enable control 1: Enable the rising edge interrupt of the corresponding port 0: Disable the rising edge interrupt of the corresponding port Note: PA port valid bits [4:0], PB port valid bits [5:0], PC port valid bits [7:3], PD port valid bits [6:1]

6.4.14 Port Falling Edge Interrupt Enable Configuration Register (GPIOx_FALLIE)

Address offset: 0x44

Reset value: 0x0000 0000

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

Bit	Marking	Functional description
31:8	Reserved	Keep
7:0	PIN	Port falling edge interrupt enable control 1: Enable the falling edge interrupt of the corresponding port 0: Disable the falling edge interrupt of the corresponding port Note: PA port valid bits [4:0], PB port valid bits [5:0], PC port valid bits [7:3], PD port valid bits [6:1]

6.4.15 Port Interrupt Status Register (GPIOx_IFR)

Address offset: 0x48

Reset value: 0x0000 0000

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

Bit	Marking	Functional description
31:8	Reserved	Keep
7:0	PIN	Port interrupt status flag 1: A qualifying interrupt has occurred 0: No qualifying interrupt occurred Note: PA port valid bits [4:0], PB port valid bits [5:0], PC port valid bits [7:3], PD port valid bits [6:1]

6.4.16 Port Interrupt Clear Register (GPIOx_ICR)

Address offset: 0x4C

Reset value: (PA)0x0000 001F

(PB)0x0000 003F

(PC)0x0000 00F8

(PD)0x0000 007E

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

Bit	Marking	Functional description
31:8	Reserved	Keep
7:0	PIN	Port interrupt flag cleared 1: no function 0: Clear the corresponding interrupt flag bit Note: PA port valid bits [4:0], PB port valid bits [5:0], PC port valid bits [7:3], PD port valid bits [6:1]

6.4.17 Port auxiliary function configuration register 1 (GPIOx_CR1)

Address offset: 0xA4

Reset value: 0x0000 0000

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

Bit	Marking	Functional description
31:5	Reserved	Keep
4:3	TCLKDIV	TCLK_OUT output frequency division control 00: TCLK 01: TCLK/8 10: TCLK/64 11: TCLK/256
2	Reserved	Keep
1:0	TCLKSRC	TCLK_OUT output signal selection 00: RCH 01: RCL 10: PCLK 11: HCLK

6.4.18 Port Miscellaneous Function Configuration Register 4 (GPIOx_CR4)

Address offset: 0xB0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						GTIMCH0		Reserved		ATIM3CH0B		Reserved			
						RW				RW					

Bit	Marking	Functional description
31:10	Reserved	Keep
9:8	GTIMCH0	GTimer timer CH0 input source configuration See GPIO_CR4[5:4] for details
7:6	Reserved	Keep
5:4	ATIM3CH0B	ATimer3 timer CH0B input source configuration 00: determined by GPIOx_AFRL 01: LPUART0_RXD 10: LPUART1_RXD 11: LVD_OUT
3:0	Reserved	Keep

7 FLASH controller (FLASH)

7.1 Overview

This system includes a piece of FLASH memory with a capacity of 18k bytes (Byte), which is divided into 36 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 Byte (8 bits), half-word (16 bits) and word (32 bits). Note that the target address of the Byte operation must be byte-aligned, 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.

7.2 Capacity division

The FLASH capacity supported by this controller is 18KB, and the capacity of each page (Secotr) is 512 bytes (Byte), and its composition is shown below.

Table 7-1 FLASH capacity division

Address	Serial number		Address	Serial number
0x0E00 – 0x0FFF	Sector7	.....	0x4600 – 0x47FF	Sector35
0x0C00 – 0x0DFF	Sector6	.....	0x4200 – 0x43FF	Sector34
0x0A00 – 0x0BFF	Sector5	.....	0x4400 – 0x45FF	Sector33
0x0800 – 0x09FF	Sector4	.....	0x4000 – 0x41FF	Sector32
0x0600 – 0x07FF	Sector3	.....	0x3E00 – 0x3FFF	Sector31
0x0400 – 0x05FF	Sector2	.....	0x3C00 – 0x3DFF	Sector30
0x0200 – 0x03FF	Sector1	.....	0x3A00 – 0x3BFF	Sector29
0x0000 – 0x01FF	Sector0	.....	0x3800 – 0x39FF	Sector28

7.3 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.

7.4 FLASH operation (read, write, erase)

7.4.1 Page Erase (Sector Erase)

Page erase can erase a page (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 executed from FLASH, the CPU will stop fetching instructions, and the hardware will automatically wait for the operation to complete (FLASH_CR.BUSY becomes 0); if the erase operation is executed from RAM, the CPU will not stop fetching means, user software should wait for the operation to complete (FLASH_CR.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_SLOCK to 1 to remove the sector 's erasure and write protection.
Step6. Check whether the corresponding bit of FLASH_SLOCK is 1, if it is not 1, jump to Step4.
Step7. Write any data to any address in the Sector to be erased to trigger Sector erasure.
Example: * ((unsigned char *) 0x00000200) = 0x00.
Step8. Wait for FLASH_CR.BUSY to become 0, Sector erase operation is completed.
Step9. To erase other sectors, repeat Step4 - C Step8.

7.4.2 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 is complete (FLASH_CR. BUSY goes to 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, and set the Flash operation mode to Chip erase.
Step3. Check if 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_SLOCK to 0xFFFFFFFF to remove the erase and write protection of all Sectors.

Step6. Check if FLASH_SLOCK is 0xFFFFFFFF, if it is not 0xFFFFFFFF, then jump to Step4.

Step7. Write any address in the Chip to be erased to trigger Chip erasure.
Example: * ((unsigned char *) 0x00000000) = 0x00.

Step8. Wait for FLASH_CR.BUSY to become 0, and the Chip erase operation is completed.

7.4.3 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 3 data lengths: Byte (8bits), 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_CR. 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_CR.BUSY goes to 0).

Byte 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, and set the Flash operation mode to write.

Step3. Check whether FLASH_CR.OP is 1, 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_SLOCK to 1 to remove the erase protection.

Step6. Check whether the corresponding bit of FLASH_SLOCK is 1, if not, jump to Step4.

Step7. Perform a Byte write operation on the target address to be written to trigger the write operation.
Example: * ((unsigned char *) 0x00001231) = 0x5A.

Step8. Wait for FLASH_CR. BUSY to become 0, and the write operation is completed.

Step9. If you need to write Byte to other addresses, repeat Step7 - C Step8.

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, and set the Flash operation mode to write.

Step3. Check whether FLASH_CR.OP is 1, 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_SLOCK to 1 to remove the erase protection.

Step6. Check whether the corresponding bit of FLASH_SLOCK is 1, if not, 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 *) 0x00001232) = 0xABCD.

Step8. Wait for FLASH_CR.BUSY to become 0, and the write operation is completed.

Step9. To write Half-word to other address, repeat Step7 - C Step8.

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, and set the Flash operation mode to write.

Step3. Check whether FLASH_CR.OP is 1, 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_SLOCK to 1 to remove the erase protection.

Step6. Check whether the corresponding bit of FLASH_SLOCK is 1, if not, jump to Step4.

Step7. Perform a Word write operation on the target address to be written to trigger the write operation.

Example: * ((unsigned int *) 0x00001234) = 0x55667788.

Step8. Wait for FLASH_CR.BUSY to become 0, and the write operation is completed.

Step9. If you need to write Word to other addresses, repeat Step7-C Step8.

7.4.4 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 read operation does not require any operation steps, and the data in the FLASH can be read at any time.

example:

Byte read operation:	*temp = * ((unsigned char *) 0x00001231)*
Half-word read operation	*temp = * ((unsigned short int *) 0x00001232)*
Word read operation	*temp = * ((unsigned int *) 0x00001234)*

7.5 Flash Security Protection

7.5.1 Page Erase Protection

The FLASH memory has a total of 18K bytes, which is divided into 36 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.

7.5.2 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.

7.5.3 Register write protection

The important controller of this module shields the normal write operation and must be modified by writing sequence.

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

FLASH_CR, FLASH_SLOCK, FLASH_WAIT.

Other registers are registers that can be changed without requiring a write sequence.

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.

7.5.4 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.

Protection Level	ISP interface			SWD interface		
	FLASH			FLASH		
	erase	to write	read out	erase	to write	read out
Level0	Y	Y	Y	Y	Y	Y
Level1	Y	Y	N	Y	N	N
Level2	Y	Y	N	N	N	N
Level3	N	N	N	N	N	N

Note:

1) Level1 can be downgraded to Level0 through ISP and SWD. After downgrading, the FLASH content is all FF
2) Level2 can be downgraded to Level0 through ISP. After downgrading, the FLASH content is all FF
3) Level3 ISP and SWD interfaces are prohibited, that is, the chip can only be programmed once

7.6 Registerdescription

7.6.1 Control register list

Base address: 0x4002 0000

Register	Offset address	Description
FLASH_CR	0x20	control register
FLASH_IFR	0x24	Interrupt Flag Register
FLASH_ICLR	0x28	Interrupt Flag Clear Register
FLASH_BYPASS	0x2C	sequence register
FLASH_SLOCK	0x30	Erase and write protection register
FLASH_WAIT	0x50	Read wait period register
FLASH_LockState	0x54	Read Protection Status Register

7.6.2 Control Register(FLASH_CR)

Offset address: 0x20

Reset value: 0x0000 0200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						Lp Mode	Reserved		FLASH IE	PCIE	BUSY	Reserved		OP	
						RW			RW	RW	RO			RW	

Bit	Marking	Functional description
31:10	Reserved	reserved bit
9	LpMode	FLASH low power consumption control 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; Note: Keep this register value at 1
8:7	Reserved	reserved bit
6	FLASHIE	FLASH protected address erase interrupt enable control 0: Prohibited 1: Enable
5	PCIE	Interrupt enable control of the FLASH page where the PC is located 0: Prohibited 1: Enable
4	BUSY	FLASH erase status flag 0: The Erase and Write operation is complete 1: The Erase and Write operation is not completed
3:2	Reserved	reserved bit
1:0	OP	FLASH operation mode configuration 00: read (Read) 01: Write (Program) 10: Page Erase (Sector Erase) 11: Full Chip Erase (Chip Erase)

7.6.3 Interrupt Flag Register (FLASH_IFR)

Offset address: 0x24

Reset value: 0x0000 0000

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

Bit	Marking	Description
31:2	Reserved	reserved bit
1	FLASH	FLASH erasing protection alarm interrupt flag bit
0	PC	FLASH erase PC address alarm interrupt flag bit

7.6.4 Interrupt Flag Clear Register (FLASH_ICLR)

Offset address: 0x28

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														FLASH	PC
														R1W0	R1W0

Bit	Marking	Functional description
31:2	Reserved	reserved bit
1	FLASH	Clear protection alarm interrupt flag bit write 0 to clear Writing 1 is invalid
0	PC	Clear the PC address alarm interrupt flag bit write 0 to clear Writing 1 is invalid

7.6.5 Sequence register (FLASH_BYPASS)

Offset address: 0x2C

Reset value: 0x0000 0000

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

Bit	Marking	Description
31:16	Reserved	reserved bit
15:0	BYPASS	Before modifying this module-specific register, the 0x5A5A - 0xA5A5 sequence must be written to this register. After each write to the Bypass sequence, the register can only be modified once. To modify the register again, the sequence 0x5A5A - 0xA5A5 must be entered again.

7.6.6 Erase and write protection register (FLASH_SLOCK)

Offset address: 0x30

Reset value: 0x0000 0000

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

Bit	Marking	Description
31:9	Reserved	reserved bit
8:0	SLOCK	FLASH page erase protection configuration 0: Erase and write not allowed 1: Erase and write allowed Bit0 corresponds to: Sector0 - Sector3 Bit1 corresponds to: Sector4 - Sector7 Bit2 corresponds to: Sector8 - Sector11 ..... Bit8 corresponds to: Sector32 – C Sector35

7.6.7 Read wait period register (FLASH_WAIT)

Offset address: 0x50

Reset value: 0x0000 0000

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

Bit	Marking	Functional description
31:2	Reserved	reserved bit
1:0	wait	FLASH instruction fetch wait cycle configuration 00: 1 cycle, for HCLK <= 24MHz 01: 2 cycles, for 24MHz < HCLK <= 48MHz 10: reserved 11: reserved

7.6.8 Read Protection Status Register (FLASH_LockState)

Offset address: 0x54

Reset value: 0x0000 0000

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

Bit	Marking	Description
31:2	Reserved	reserved bit
1:0	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

8 RAM controller (RAM)

8.1 Overview

This system contains a piece of SRAM with a capacity of 2k bytes (Byte), which supports three kinds of read and write operations: byte (8 bits), half word (16 bits), and word (32 bits). Read and write operations can be performed at the system clock frequency without waiting for cycles.

8.2 Functional description

8.2.1 RAM address range

The address range of RAM in the system map is shown in the table below:

Table 8-1 RAM Address Mapping

Address range	size	Memory type
0x20000000 – 0x200007FF	2K Byte	SRAM

8.2.2 Read and write bit width

This controller supports byte (8 -bit), half-word (16 -bit), and word (32 -bit) three bit-width read and write operations. The address of the byte operation must be byte aligned, the target address of the halfword operation must be halfword 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 of the read and write operation is not aligned according to the bit width, the operation is invalid, and the system will generate a Hard Fault error interrupt.

9 Basic Timer (BTIM)

9.1 Overview

BTIM is a group of timing counters with basic timing functions, each group contains three 16 -bit timers with reload function. This product contains a set of basic timers: BTIM3/4/5. BTIM3/4/5 and GTIM functions cannot be used at the same time, controlled by SysCtrl_CR1.GTIMCFG, when this bit is 1, BTIM3/4/5 is valid.

9.2 Main characteristics

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

9.3 Functional description

9.3.1 Functional block diagram

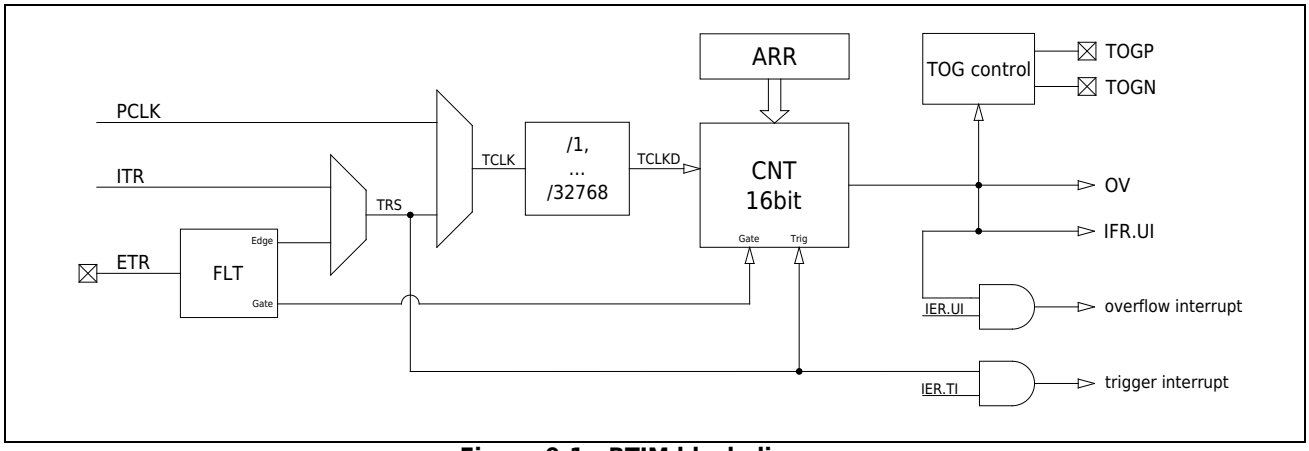


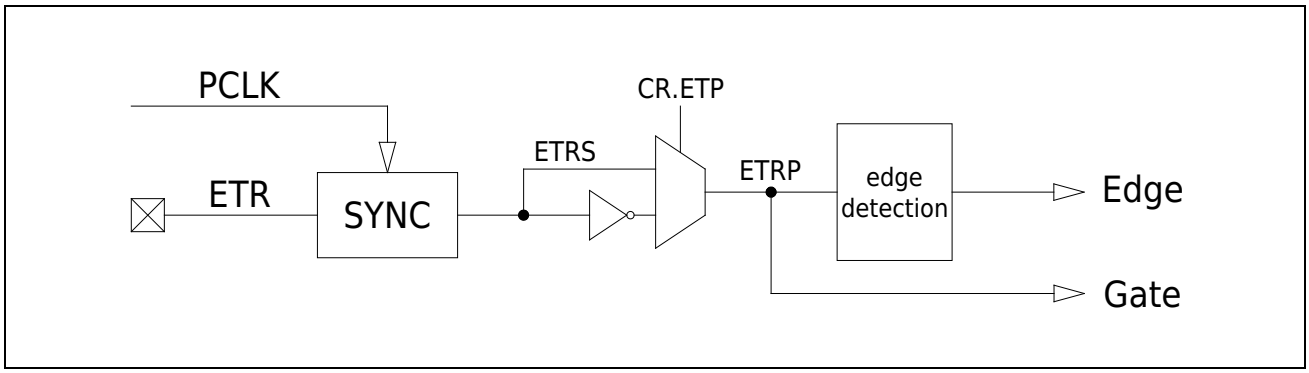
Figure 9-1 BTIM block diagram

Note:

- BIM3/4/5 share one ETR
- For the source of ITR signal, please refer to the function chapter 9.3.8Timer Cascading

9.3.2 filter unit

After the 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.



9.3.3 counting unit

On each rising edge of the TCLKD signal, the internal counter CNT is automatically incremented by 1. After 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 auxiliary function of GPIO to drive the external circuit.

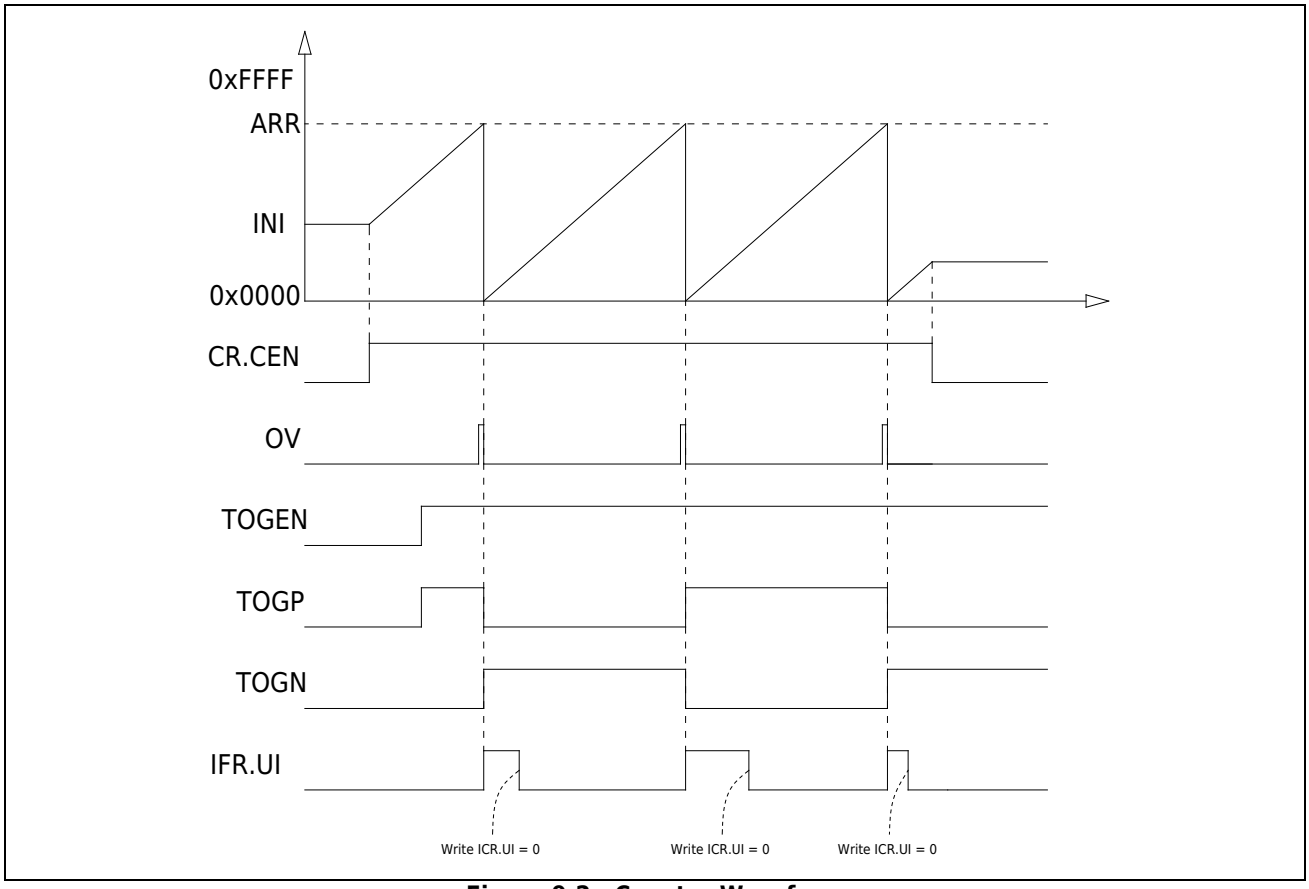


Figure 9-2 Counter Waveform

9.3.4 timer mode

When setting BTIMx_CR.MD to 0x00, BTIMx works in timer mode. In this mode, the count 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.

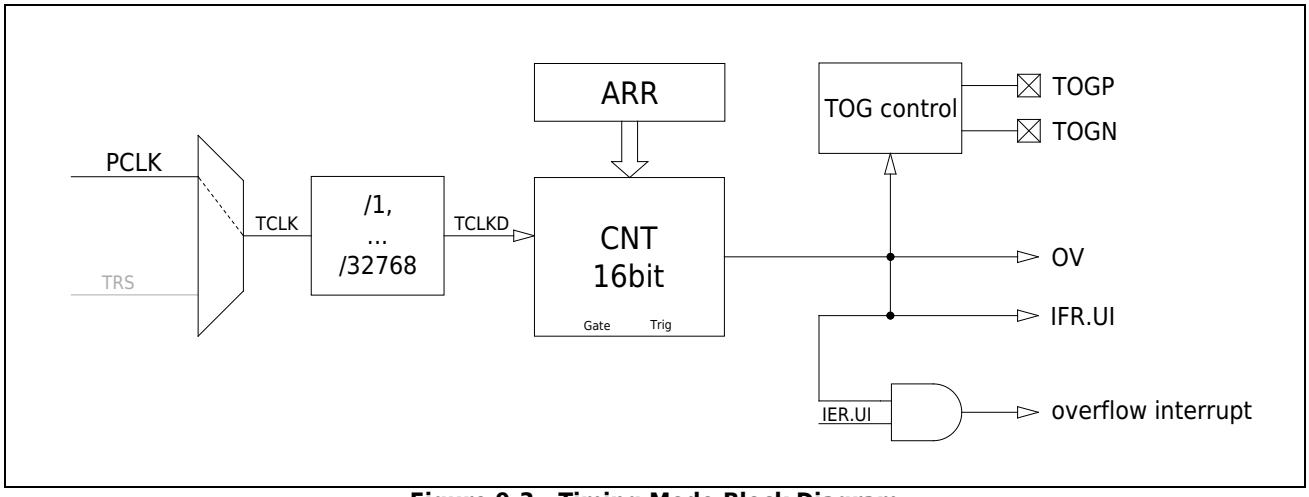


Figure 9-3 Timing Mode Block Diagram

9.3.5 counter mode

When setting BTIMx_CR.MD to 0x01, BTIMx works in counter mode. In this mode, the count clock is 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.

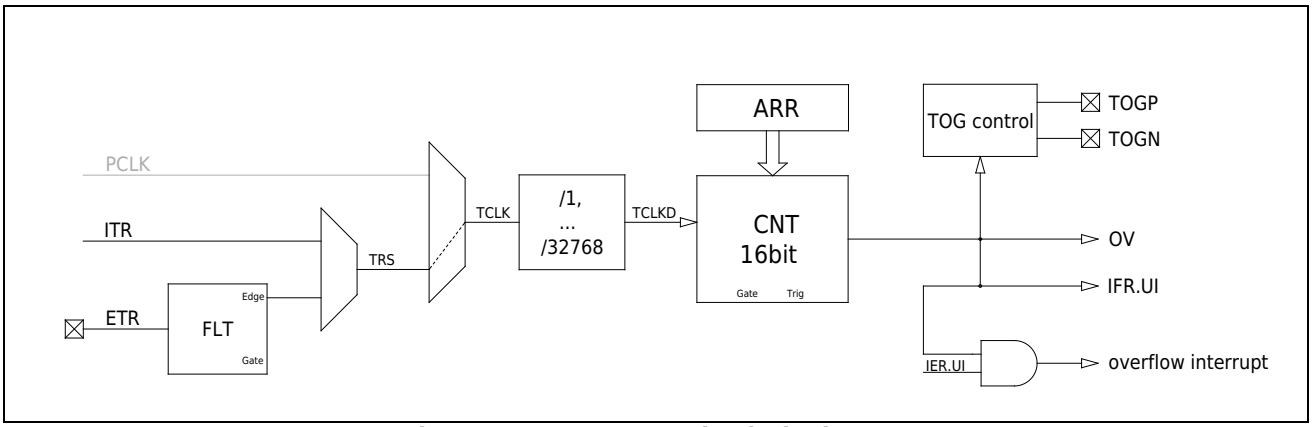


Figure 9-4 Counter Mode Block Diagram

9.3.6 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.

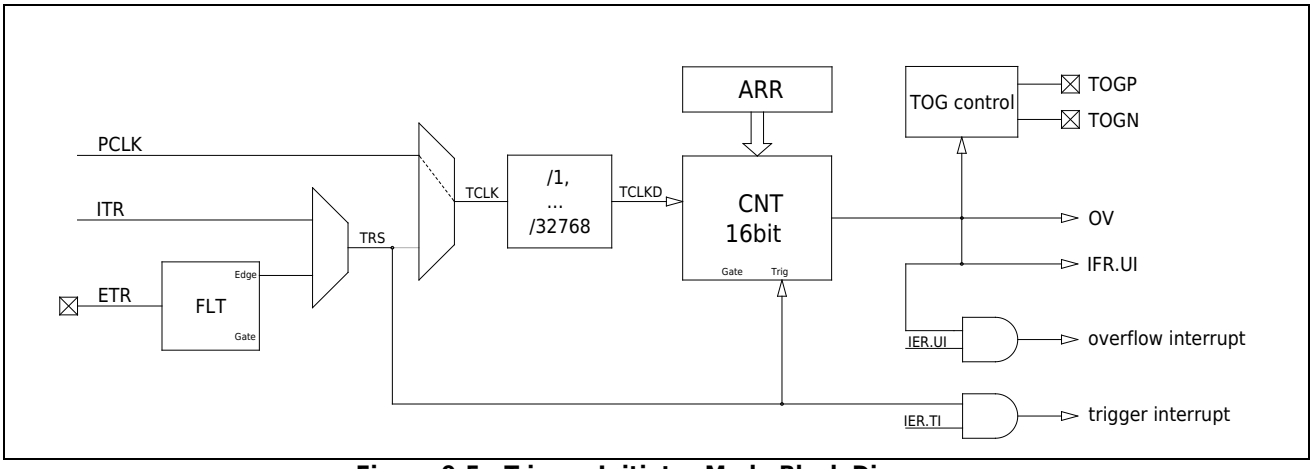


Figure 9-5 Trigger Initiator Mode Block Diagram

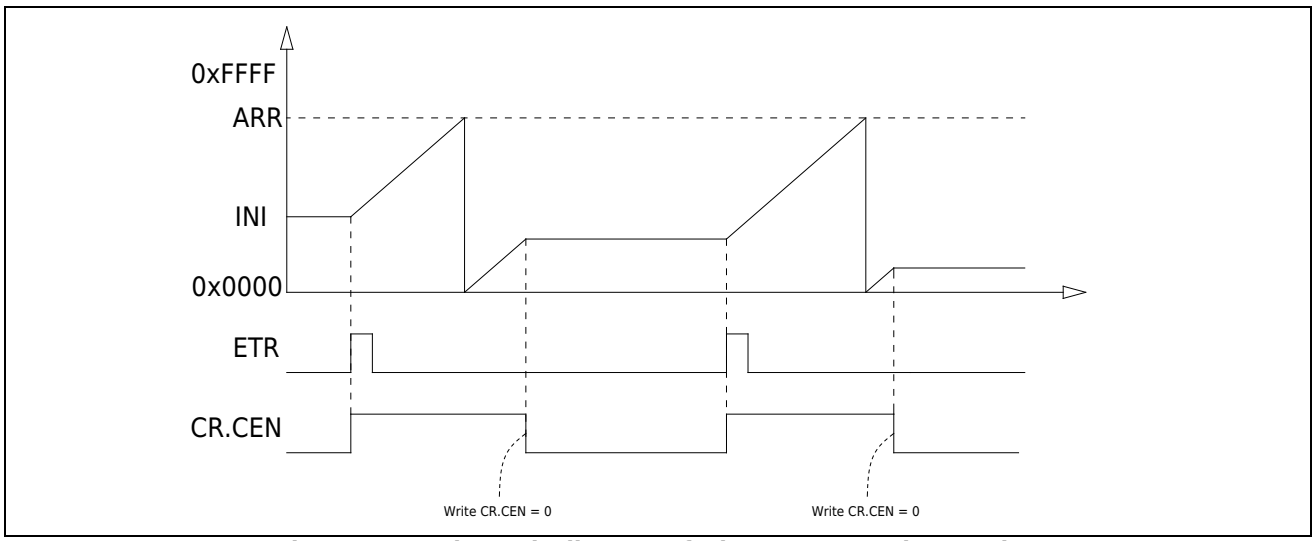


Figure 9-6 Schematic diagram of trigger start mode counting

9.3.7 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, and 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.

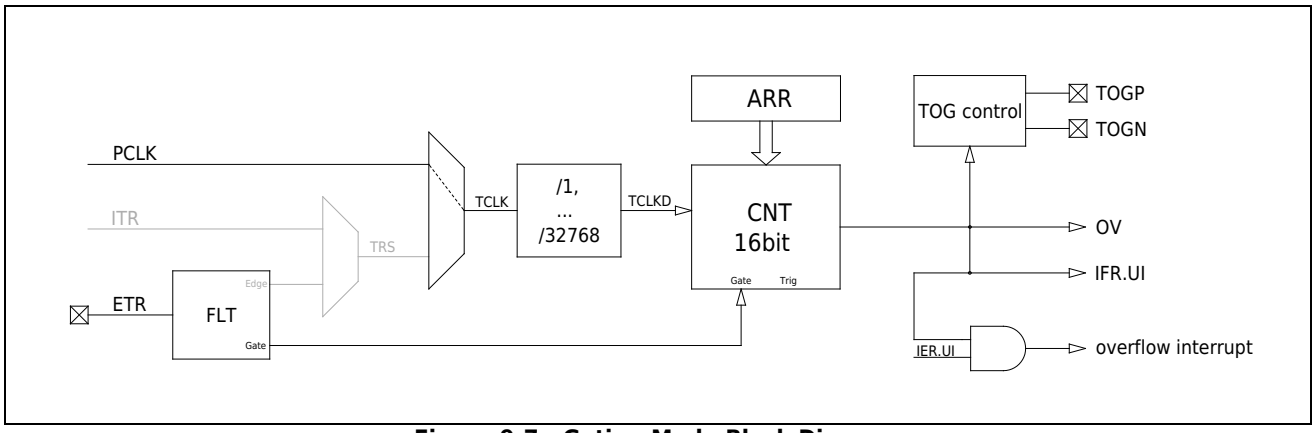


Figure 9-7 Gating Mode Block Diagram

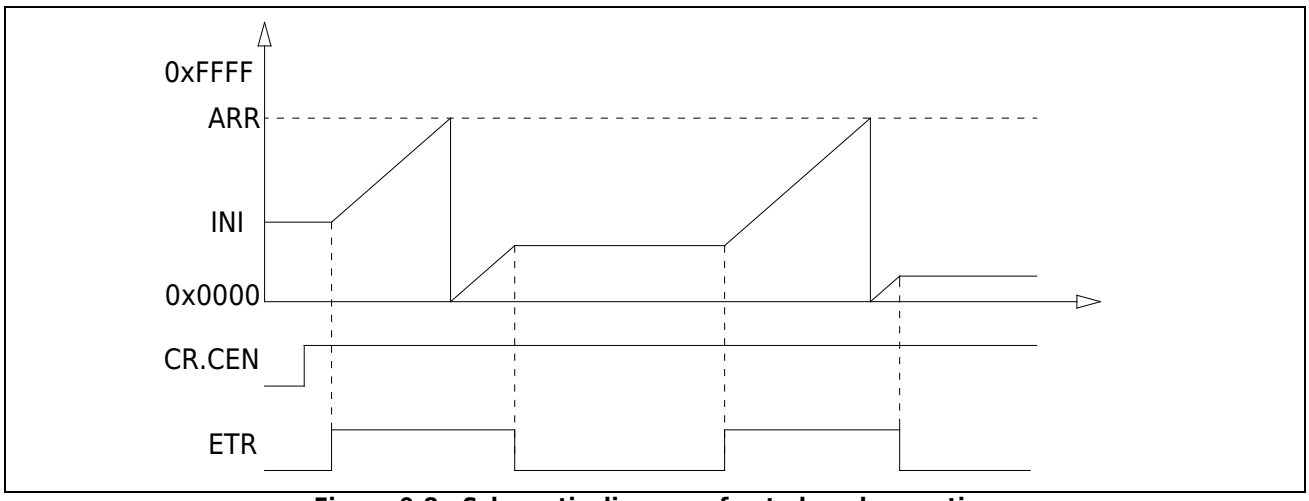
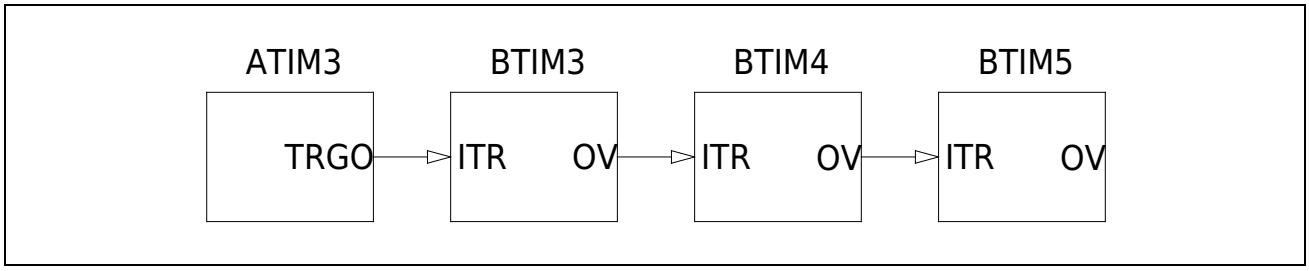


Figure 9-8 Schematic diagram of gated mode counting

9.3.8 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.



9.4 Register description

Table 9-1 BTIM Register List

Name	base address	Description
BTIM3	0x40007400	BTIM3 base address
BTIM4	0x40007500	BTIM4 base address
BTIM5	0x40007600	BTIM5 base address

Note: BTIM3, BTIM4, and BTIM5 are only valid when SysCtrl_CR1.GTIMCFG is 1.

Register	Offset address	Description
BTIMx_ARR	0x00	BTIMx reload register
BTIMx_CNT	0x04	BTIMx Counter Register
BTIMx_CR	0x10	BTIMx Control Register
BTIMx_IER	0x14	BTIMx Interrupt Enable Register
BTIMx_IFR	0x18	BTIMx Interrupt Flag Register
BTIMx_ICR	0x1C	BTIMx Interrupt Clear Register
BTIM345_AIFR	0x48	BTIM345 Composite Interrupt Flag Register
BTIM345_AICR	0x4C	BTIM345 Composite Interrupt Clear Register

9.4.1 Reload Register (BTIMx_ARR) (x=3,4,5)

Offset address: 0x000

Reset Value: 0x0000 FFFF

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

Bit	Symbol	Description
31:16	Reserved	reserved bit
15:0	ARR	timer reload value

9.4.2 Count Register (BTIMx_CNT) (x=3,4,5)

Offset address: 0x004

Reset Value: 0x0000 0000

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

Bit	Symbol	Description
31:16	Reserved	reserved bit
15:0	CNT	Timer count value

9.4.3 Control Register (BTIMx_CR) (x=3,4,5)

Offset address: 0x010

Reset Value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				ETP	TRS		OST	PRS				TOGEN	MD		CEN
				RW	RW		RW	RW				RW	RW		RW

Bit	Symbol	Description
31:12	Reserved	reserved bit
11	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)
10:9	TRS	Trigger source selection 00: ETR pin input signal 01: ITR, refer to the cascading function chapter for details Note: BTIM3-5 share another ETR pin
8	OST	Single / continuous count mode control 0: continuous counting mode 1: Single count mode
7:4	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
3	TOGEN	TOG pin output enable control 1: TOGP, TOGN output signals with opposite levels 0: TOGP, TOGN output levels are both 0
2:1	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
0	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

9.4.4 Interrupt Enable (BTIMx_IER) (x=3,4,5)

Offset address: 0x014

Reset Value: 0x0000 0000

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

Bit	Symbol	Description
31:2	Reserved	reserved bit
1	TI	Trigger interrupt enable control 1: enable trigger interrupt 0: disable trigger interrupt
0	UI	Overflow interrupt enable control 1: enable overflow interrupt 0: disable overflow interrupt

9.4.5 Interrupt Flag Register (BTIMx_IFR) (x=3,4,5)

Offset address: 0x018

Reset Value: 0x0000 0000

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

Bit	Symbol	Description
31:2	Reserved	reserved bit
1	TI	trigger flag 1: A trigger event has occurred 0: No trigger event occurred
0	UI	counter overflow flag 1: The counter has overflowed 0: The counter has not overflowed

9.4.6 Interrupt Flag Clear Register (BTIMx_ICR) (x=3,4,5)

Offset address: 0x01C

Reset Value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														TI	UI
														R1W0	R1W0

Bit	Symbol	Description
31:2	Reserved	reserved bit
1	TI	trigger flag clear 1: no function 0: clear trigger flag
0	UI	Counter overflow flag cleared 1: no function 0: Clear counter overflow flag

9.4.7 Composite Interrupt Flag Register (BTIM345_AIFR)

Offset address: 0x048

Reset Value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										TI5	UI5	TI4	UI4	TI3	UI3
										RO	RO	RO	RO	RO	RO

Bit	Symbol	Description
31:6	Reserved	reserved bit
5	TI5	BTIM5 trigger flag 1: A trigger event has occurred 0: No trigger event occurred
4	UI5	BTIM5 overflow flag 1: The counter has overflowed 0: The counter has not overflowed
3	TI4	BTIM4 trigger flag 1: A trigger event has occurred 0: No trigger event occurred
2	UI4	BTIM4 overflow flag 1: The counter has overflowed 0: The counter has not overflowed
1	TI3	BTIM3 trigger flag 1: A trigger event has occurred 0: No trigger event occurred
0	UI3	BTIM3 overflow flag 1: The counter has overflowed 0: The counter has not overflowed

9.4.8 Composite Interrupt Flag Clear Register (BTIM345_AICR)

Offset address: 0x04C

Reset Value: 0x0000 003F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										TI5	UI5	TI4	UI4	TI3	UI3
										R1W0	R1W0	R1W0	R1W0	R1W0	R1W0

Bit	Symbol	Description
31:6	Reserved	reserved bit
5	TI5	BTIM5 Trigger Flag Clear 1: no function 0: clear trigger flag
4	UI5	BTIM5 overflow flag cleared 1: no function 0: clear overflow flag
3	TI4	BTIM4 trigger flag clear 1: no function 0: clear trigger flag
2	UI4	BTIM4 overflow flag cleared 1: no function 0: clear overflow flag
1	TI3	BTIM3 trigger flag clear 1: no function 0: clear trigger flag
0	UI3	BTIM3 overflow flag cleared 1: no function 0: clear overflow flag

10 General Purpose Timer (GTIM)

10.1 Overview

GTIM is a 16 -bit timer with 4 -way compare capture function. BTIM3/4/5 and GTIM functions cannot be used at the same time, controlled by SysCtrl_CR1.GTIMCFG, GTIM is valid when this bit is 0.

10.2 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

10.3 Functional description

10.3.1 Functional block diagram

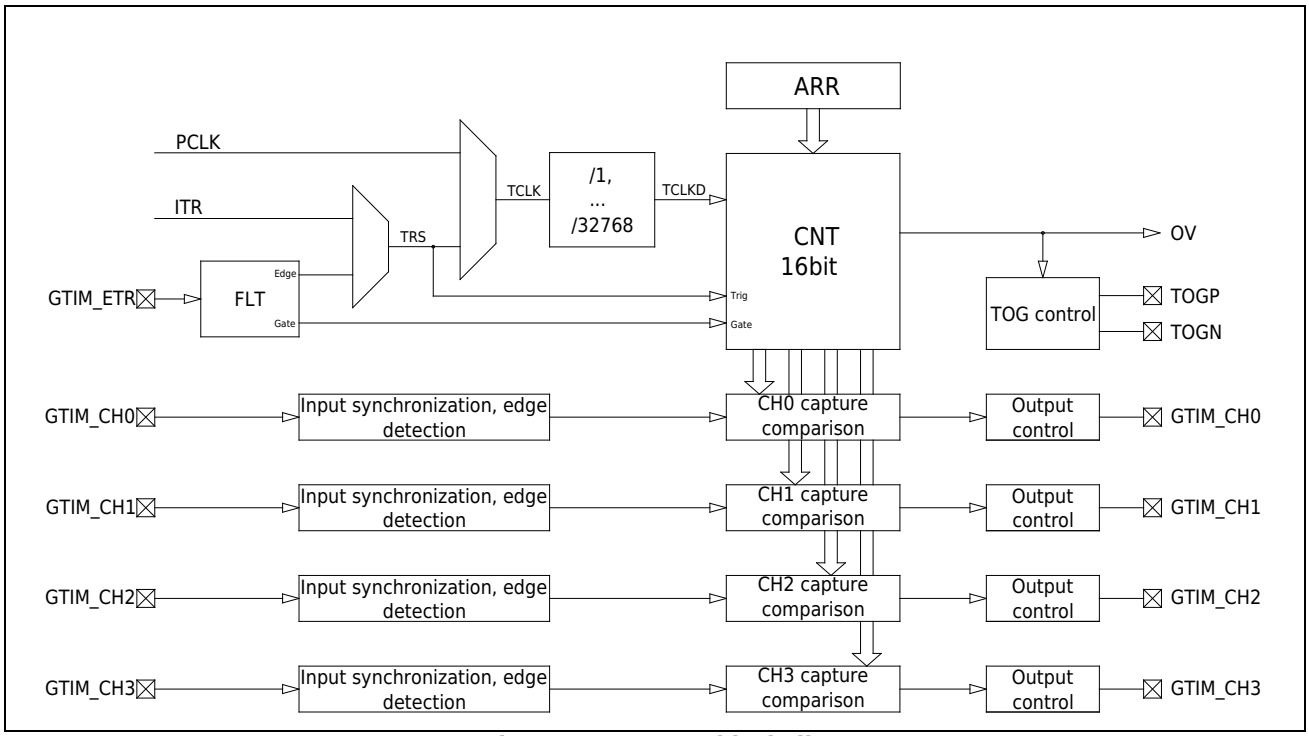
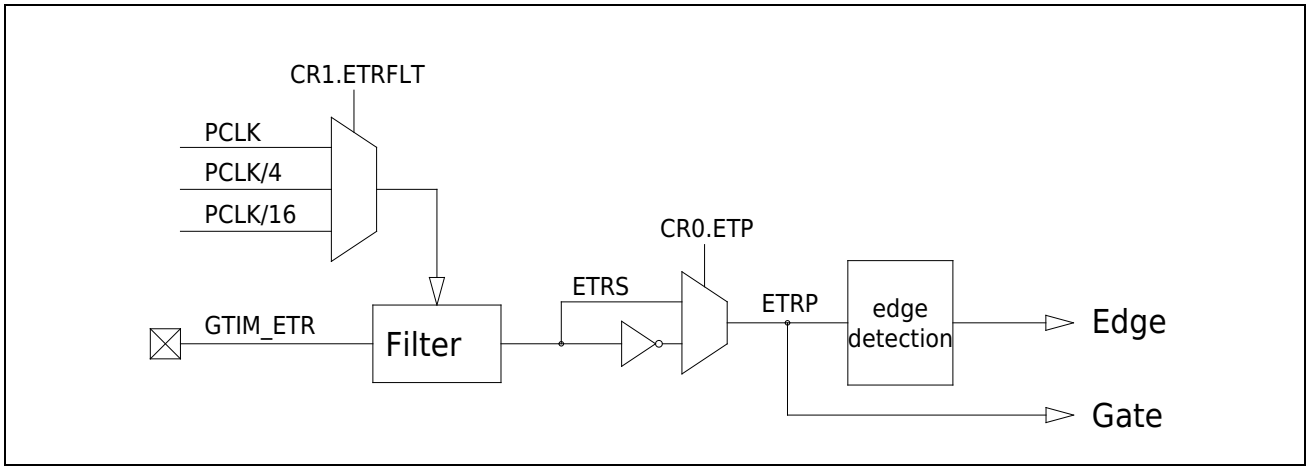


Figure 10-1 GTIM block diagram

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

10.3.2 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.



10.3.3 counting unit

On each rising edge of the TCLKD signal, the internal counter CNT is automatically incremented by 1. After 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 auxiliary function of GPIO to drive the external circuit.

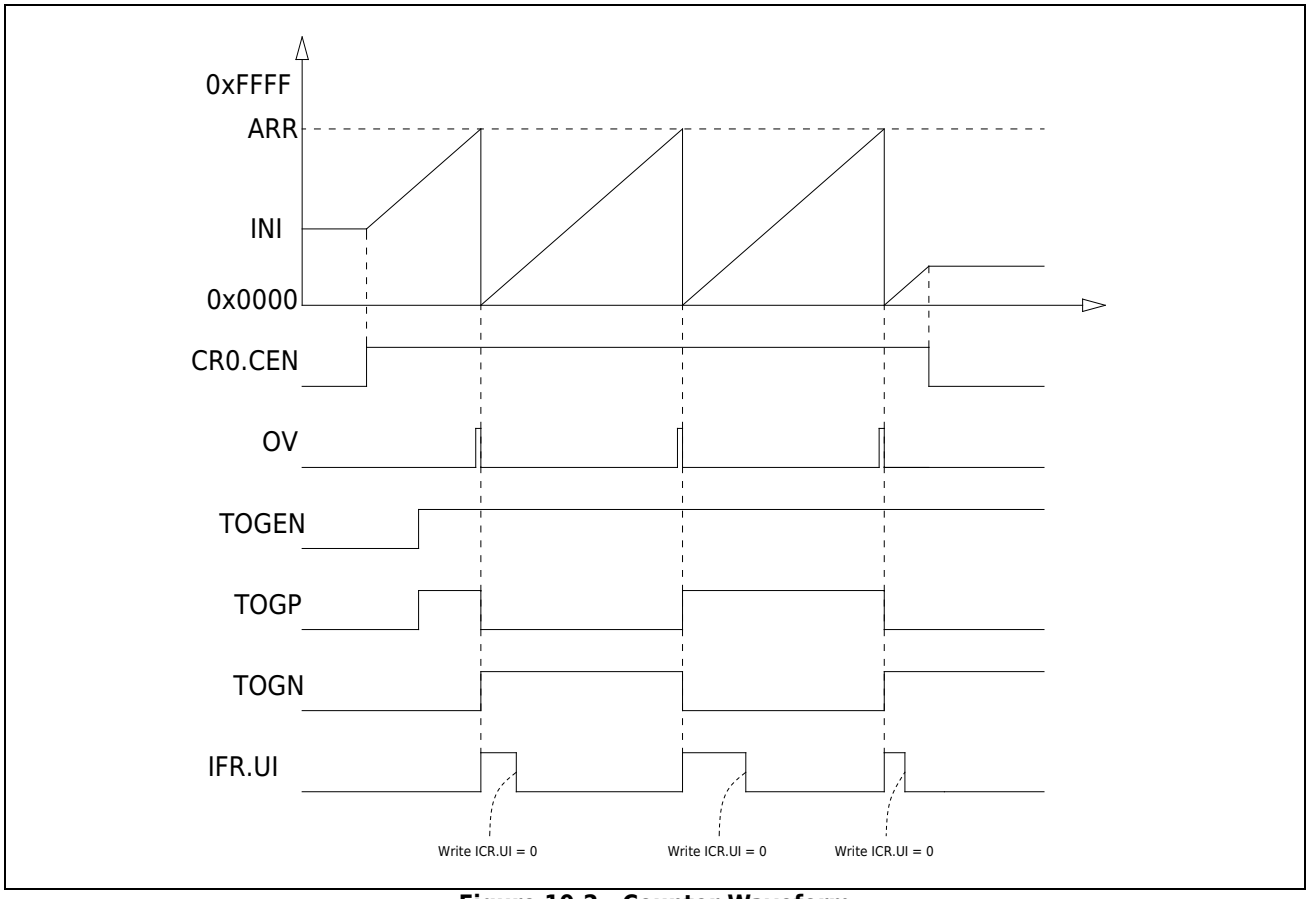


Figure 10-2 Counter Waveform

10.3.4 timer mode

When 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.

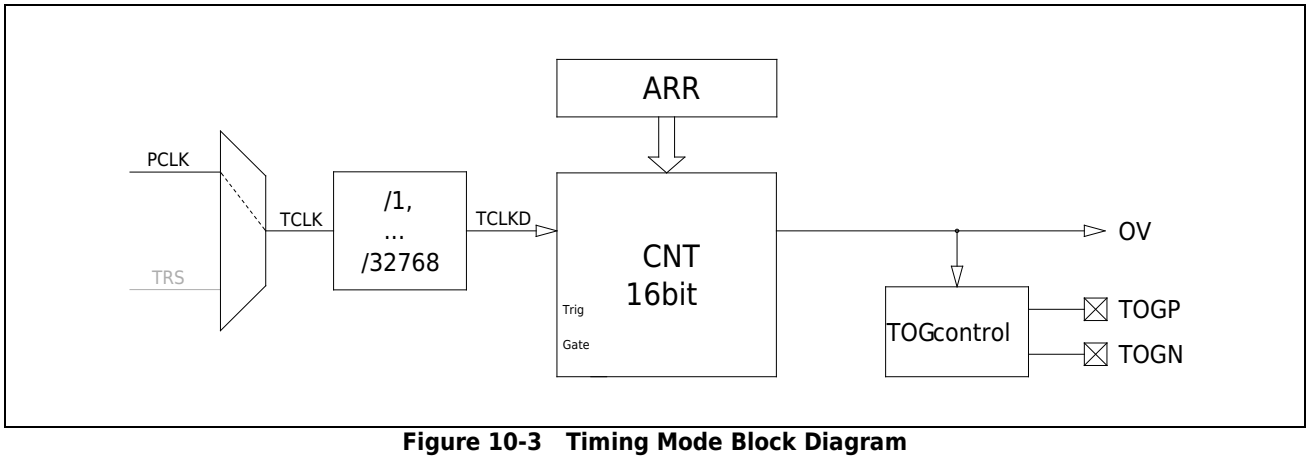


Figure 10-3 Timing Mode Block Diagram

10.3.5 counter mode

When GTIM_CR0.MD is set to 0x01, GTIM works in counter mode. In this mode, the count clock is 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.

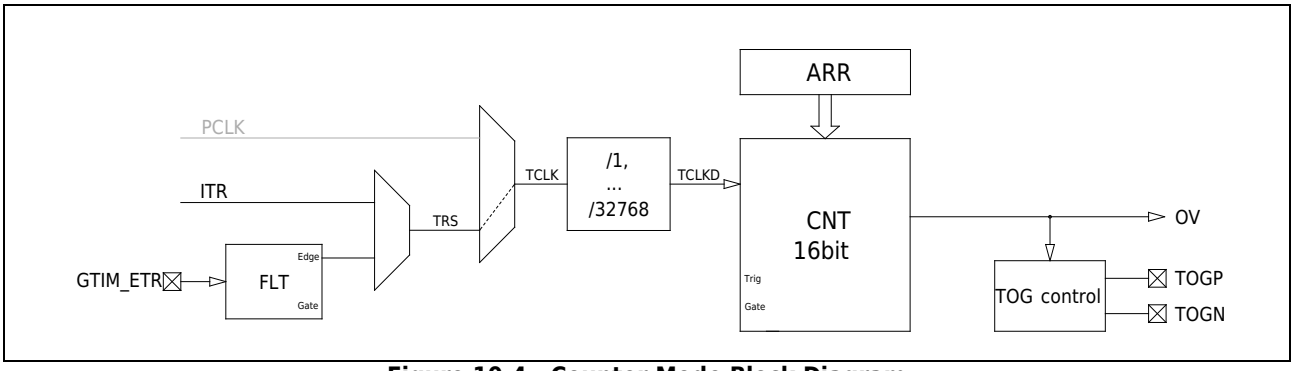


Figure 10-4 Counter Mode Block Diagram

10.3.6 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 TRS signal to be ETR or ITR by GTIM_CR0.TRS.

The rising edge or falling edge of 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.

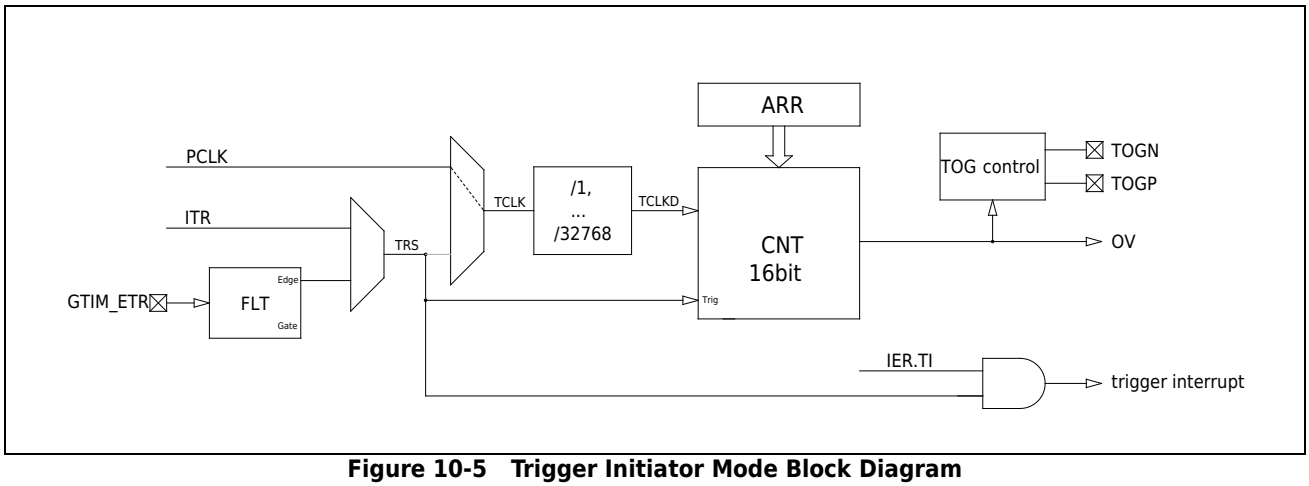


Figure 10-5 Trigger Initiator Mode Block Diagram

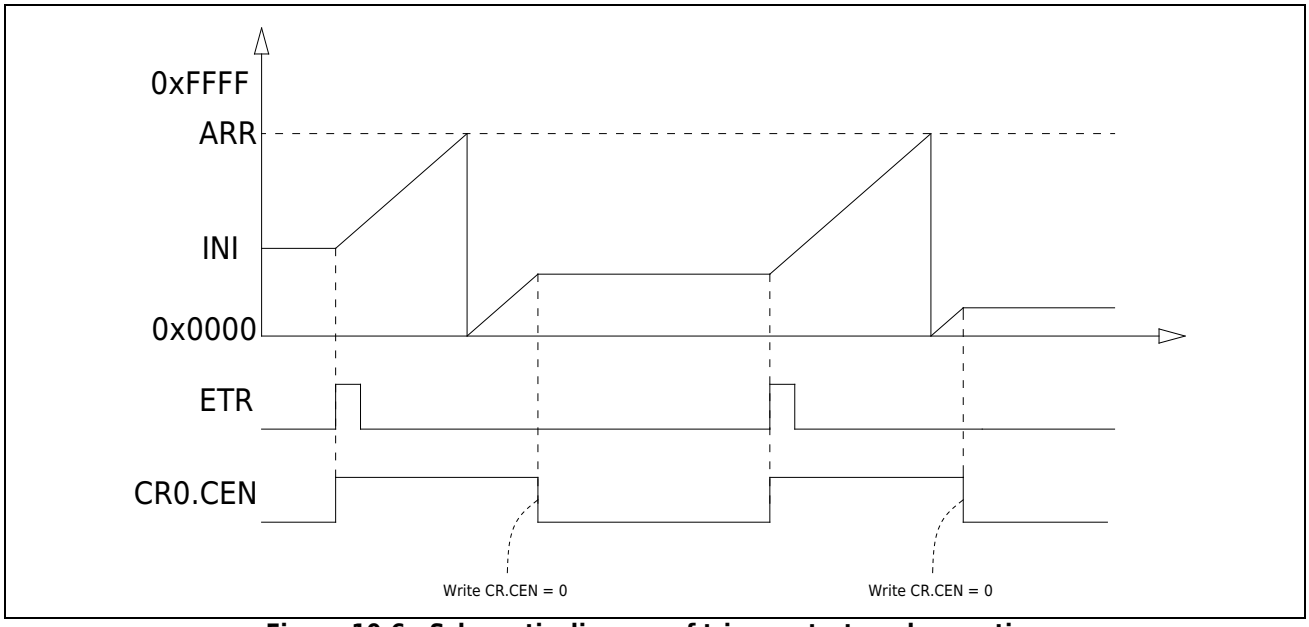


Figure 10-6 Schematic diagram of trigger start mode counting

10.3.7 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, and 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.

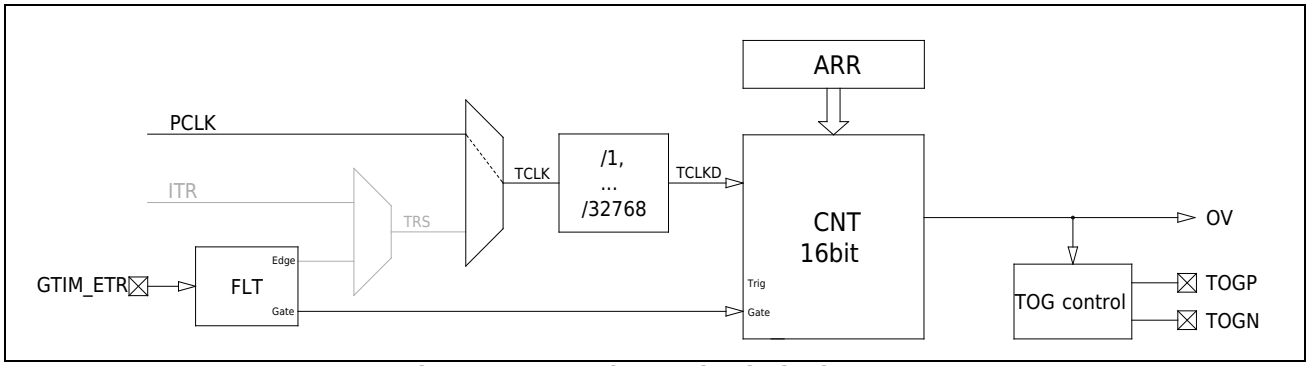


Figure 10-7 Gating Mode Block Diagram

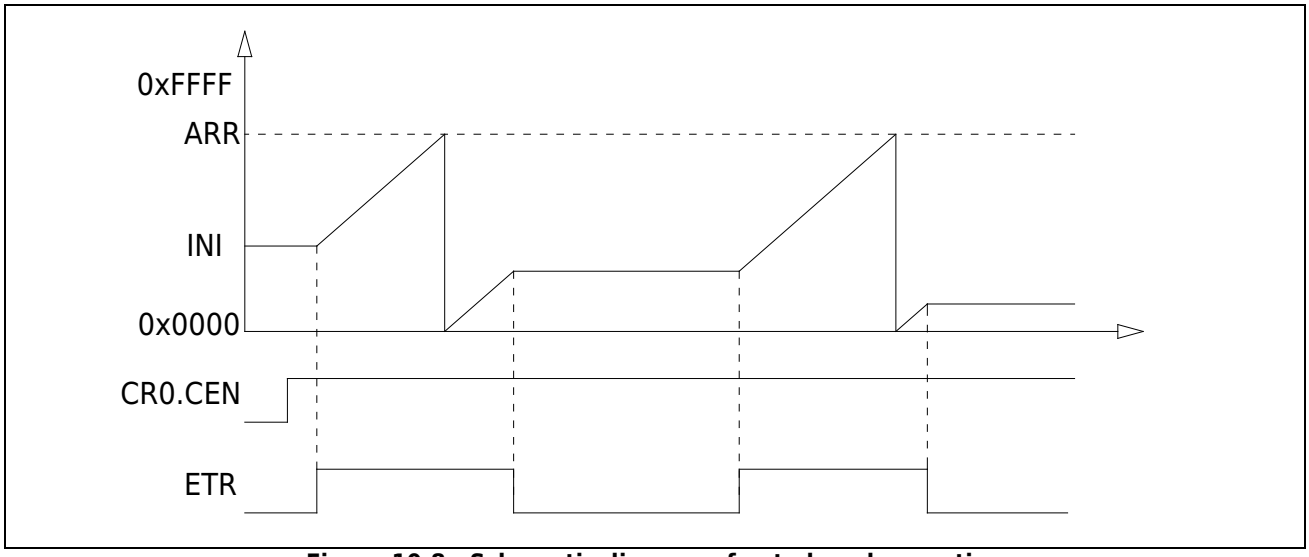


Figure 10-8 Schematic diagram of gated mode counting

10.3.8 compare capture function

GTIM has 4 comparison capture channels, CH0 ~ CH3.

10.3.8.1 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: Simultaneously capture the upper and lower 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.

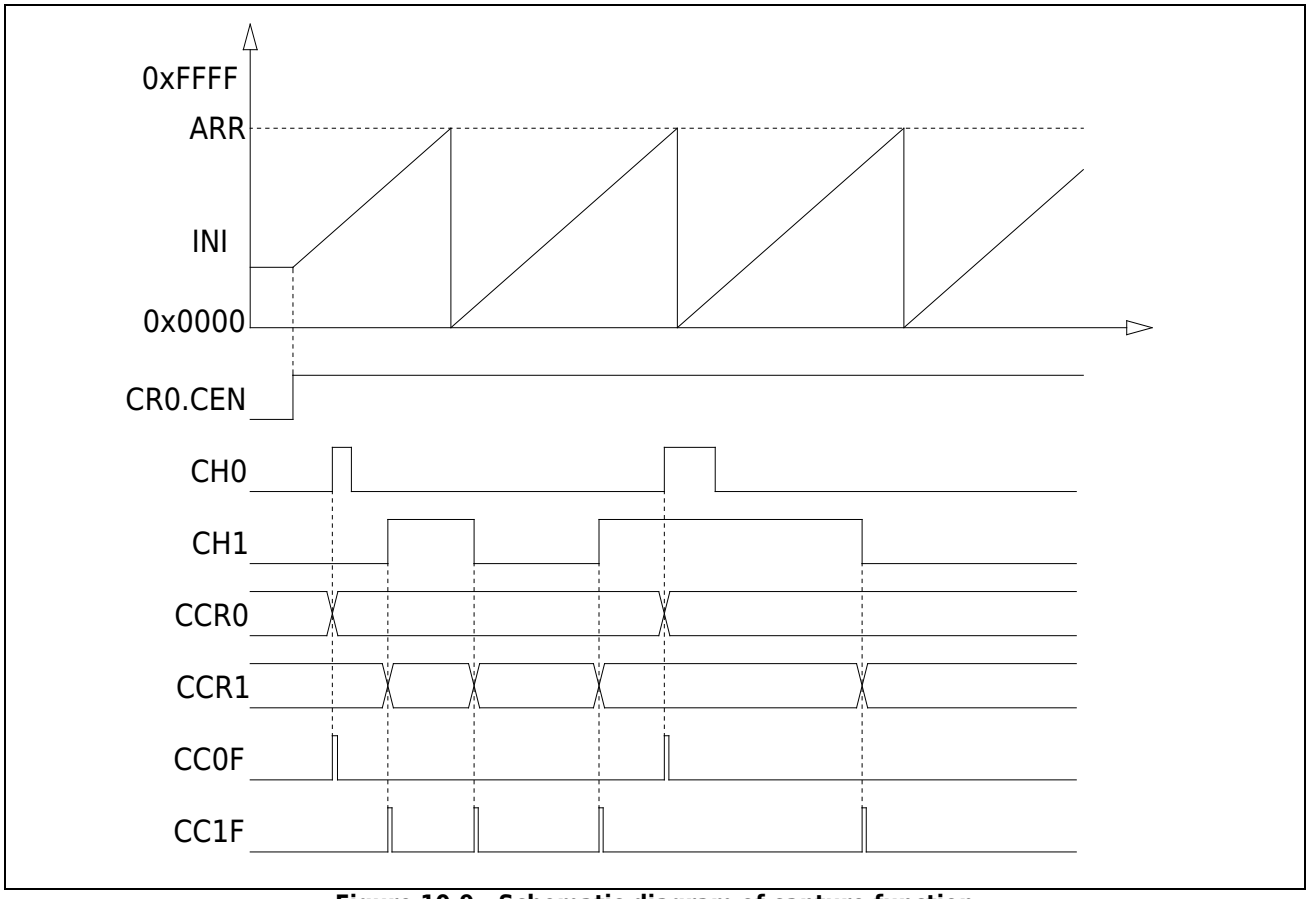


Figure 10-9 Schematic diagram of capture function

10.3.8.2 compare function

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

- 100: forced output low level
- 101: Mandatory output high level
- 110: PWM positive output (CNT >= 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: When the ARR is 99 in the forward output mode, the duty cycle corresponding to the CCR of 0~100 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.

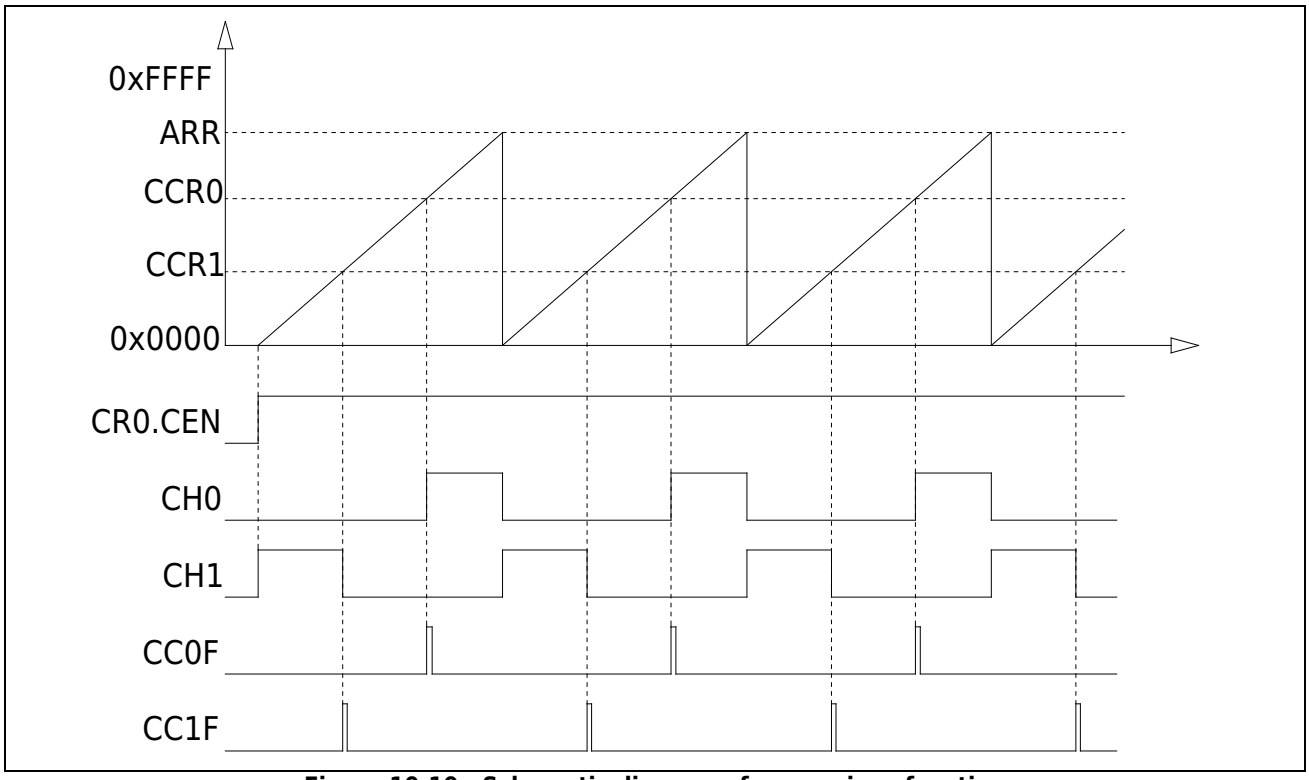


Figure 10-10 Schematic diagram of comparison function

10.3.9 Timer Cascading

Through register configuration, ATIM3 and GTIM can be combined as a cascaded 32 -bit timer. The ATIM3 timer uses the system clock PCLK as the timer clock, GTIM selects the previous overflow as the external pulse count and selects the external count function.

10.3.10 On-chip peripheral interconnection

When GPIOx_CR4.GTIMCH0 is 0, the source of GTIM_CHx signal is determined by GPIO_AFRL; when GPIOx_CR4.GTIMCH0 is 1~6, the source of GTIM_CHx signal is determined by the following table.

GPIOx_CR4.GTIMCH0	GTIM_CH0 source
001	LPUART0_RX
010	LPUART1_RX
011	LVD_OUT

10.4Register description

Table 10-1 Timer register list

Timer	base address	Description
GTIM	0x40007400	Timer base address

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

Register	Offset address	Description
GTIM_ARR	0x300	reload register
GTIM_CNT	0x304	count register
GTIM_CMMR	0x308	Compare Capture Control Register
GTIM_CR1	0x30C	control register 1
GTIM_CR0	0x310	control register 0
GTIM_IER	0x314	Interrupt Enable Control Register
GTIM_IFR	0x318	interrupt flag register
GTIM_ICR	0x31C	Interrupt Flag Clear Register
GTIM_CCR0	0x320	CH0 compare / capture register
GTIM_CCR1	0x324	CH1 compare / capture register
GTIM_CCR2	0x328	CH2 compare / capture register
GTIM_CCR3	0x32C	CH3 compare / capture register

10.4.1 Reload Register (GTIM_ARR)

Offset address: 0x300

Reset Value: 0x0000 FFFF

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

Bit	Symbol	Description
31:16	Reserved	reserved bit
15:0	ARR	timer reload value

10.4.2 Count register (GTIM_CNT)

Offset address: 0x304

Reset Value: 0x0000 0000

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

Bit	Symbol	Description
31:16	Reserved	reserved bit
15:0	CNT	Timer count value

10.4.3 Control Register 1 (GTIM_CR1)

Offset address: 0x30C

Reset Value: 0x0000 0000

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

Bit	Symbol	Description
31:9	Reserved	reserved bit
8:7	Reserved	Keep as 0, prohibit modification
6:4	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_{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
3:0	Reserved	reserved bit

10.4.4 Control Register 0 (GTIM_CR0)

Offset address: 0x310

Reset Value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				ETP	TRS		OST	PRS				TOGEN	MD		CEN
				RW	RW		RW	RW				RW	RW		RW

Bit	Symbol	Description
31:12	Reserved	reserved bit
11	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)
10:9	TRS	Trigger source selection 00: ETR pin input signal 01: ITR pin input signal (ATIM3_TRGO) 1x: reserved
8	OST	Single / continuous count mode control 0: continuous counting mode 1: Single count mode
7:4	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
3	TOGEN	TOG pin output enable control 1: TOGP, TOGN output signals with opposite levels 0: TOGP, TOGN output levels are both 0
2:1	MD	Timer working mode configuration 00: Timer mode, the count clock is PCLK 01: Counter mode, the counting clock comes from the TRS selection signal 10: Trigger start mode, the count clock is PCLK, TRS signal triggers the counter to start 11: Gating mode, counting clock is PCLK, TRS signal is used as gating
0	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

10.4.5 Interrupt Enable Control Register (GTIM_IER)

Offset address: 0x314

Reset Value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										CC3	CC2	CC1	CC0	TI	UI
										RW	RW	RW	RW	RW	RW

Bit	Symbol	Description
31:6	Reserved	reserved bit
5	CC3	CH3 capture compare interrupt enable control 1: Enable 0: Prohibited
4	CC2	CH2 capture compare interrupt enable control 1: Enable 0: Prohibited
3	CC1	CH1 capture compare interrupt enable control 1: Enable 0: Prohibited
2	CC0	CH0 capture compare interrupt enable control 1: Enable 0: Prohibited
1	TI	Trigger interrupt enable control 1: Enable 0: Prohibited
0	UI	Overflow interrupt enable control 1: Enable 0: Prohibited

10.4.6 Interrupt Flag Register (GTIM_IFR)

Offset address: 0x318

Reset Value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										CC3	CC2	CC1	CC0	TI	UI
										RW	RW	RW	RW	RW	RW

Bit	Symbol	Description
31:6	Reserved	reserved bit
5	CC3	CH3 comparison capture interrupt flag 1: A compare capture event has occurred 0: no compare capture event occurs
4	CC2	CH2 comparison capture interrupt flag 1: A compare capture event has occurred 0: no compare capture event occurs
3	CC1	CH1 comparison capture interrupt flag 1: A compare capture event has occurred 0: no compare capture event occurs
2	CC0	CH0 comparison capture interrupt flag 1: A compare capture event has occurred 0: no compare capture event occurs
1	TI	trigger interrupt flag 1: A trigger event has occurred 0: No trigger event occurred
0	UI	overflow interrupt flag 1: The counter has overflowed 0: The counter has not overflowed

10.4.7 Interrupt Flag Clear Register (GTIM_ICR)

Offset address: 0x31C

Reset Value: 0x0000 003F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										CC3	CC2	CC1	CC0	TI	UI
										RW	RW	RW	RW	RW	RW

Bit	Symbol	Description
31:6	Reserved	reserved bit
5	CC3	CH3 compare capture interrupt flag clear 1: no function 0: Clear compare capture interrupt flag
4	CC2	CH2 compare capture interrupt flag clear 1: no function 0: Clear compare capture interrupt flag
3	CC1	CH1 compare capture interrupt flag clear 1: no function 0: Clear compare capture interrupt flag
2	CC0	CH0 compare capture interrupt flag clear 1: no function 0: Clear compare capture interrupt flag
1	TI	trigger interrupt flag clear 1: no function 0: Clear trigger interrupt flag
0	UI	overflow interrupt flag clear 1: no function 0: Clear counter overflow flag

10.4.8 Compare Capture Control Register (GTIM_CMMR)

Offset address: 0x308

Reset Value: 0x0000_0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	CC3M			Res	CC2M			Res.	CC1M			Res	CC0M		
	RW				RW				RW				RW		

Bit	Symbol	Description
31:15	Reserved	reserved bit
14:12	CC3M	CH3 capture compare mode configuration See CC0M for function description
11	Reserved	reserved bit
10:8	CC2M	CH2 capture compare mode configuration See CC0M for function description
7	Reserved	reserved bit
6:4	CC1M	CH1 capture compare mode configuration See CC0M for function description
3	Reserved	reserved bit
2:0	CC0M	CH0 capture compare mode configuration 000: no function 001: rising edge capture 010: Falling edge capture 011: Simultaneously capture the upper and lower edges 100: forced output low level 101: Mandatory output high level 110: PWM positive output (CNT >= CCR0 output high level) 111: PWM reverse output (CNT < CCR output high level)

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

Offset address: 0x320 / 0x324 / 0x328 / 0x32C

Reset Value: 0xFFFF / 0x0000 / FFFF / 0x0000

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

Bit	Symbol	Description
31:16	Reserved	reserved bit
15:0	CCR	16 -bit compare capture register

11 Advanced Timer (ATIM)

11.1 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.

11.2 Main characteristics

- 6 independent PWM outputs CH0A, CH0B, CH1A, CH1B, CH2A, CH2B
- 3 complementary PWM outputs (CHxA, CHxB)
- 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

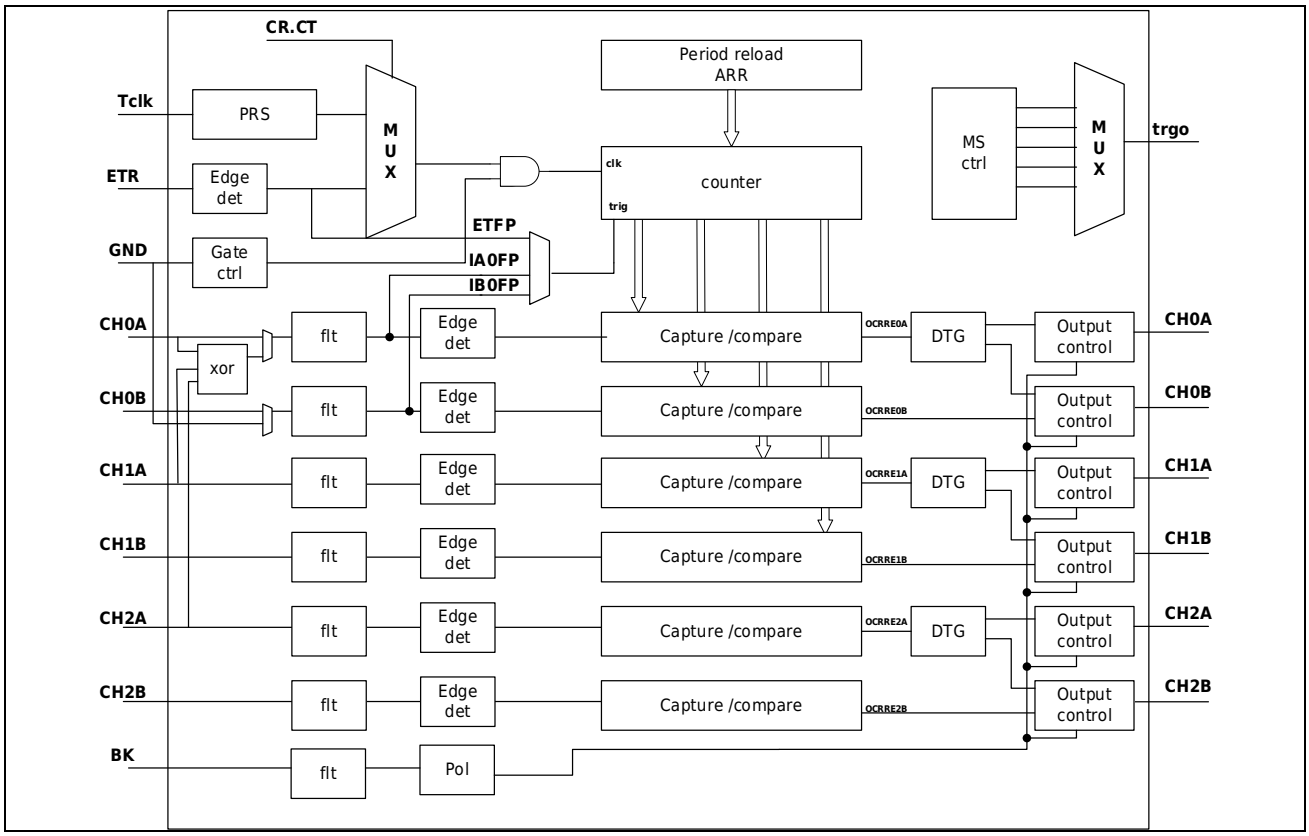


Figure 11-1 ATIM3 Block Diagram

11.3Functional description

11.3.1 timer clock

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

11.3.2 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.

11.3.3 Timer Prescaler

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

PRS	000	001	010	011	100	101	110	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.

11.3.4 Mode 0 count timer function

In this mode, the counter counts up. The counter supports two counting modes, reload mode and free counting mode, and you 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 counting range of the heavy load mode counts up from ARR to 0xFFFF overflow, and then starts counting from ARR, and the counting period is 0xFFFF-ARR+1; the free counting mode overflows after counting from the set count value to 0xFFFFFFFF, and the count value restarts from 0x0 after overflow count.

11.3.4.1 Functional block diagram

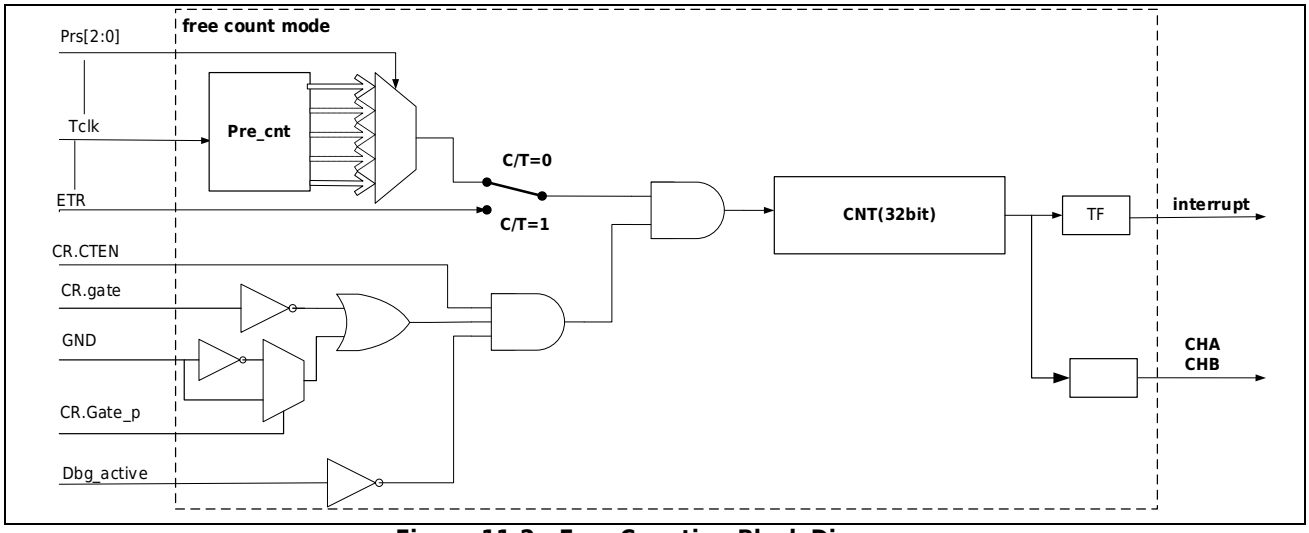


Figure 11-2 Free Counting Block Diagram

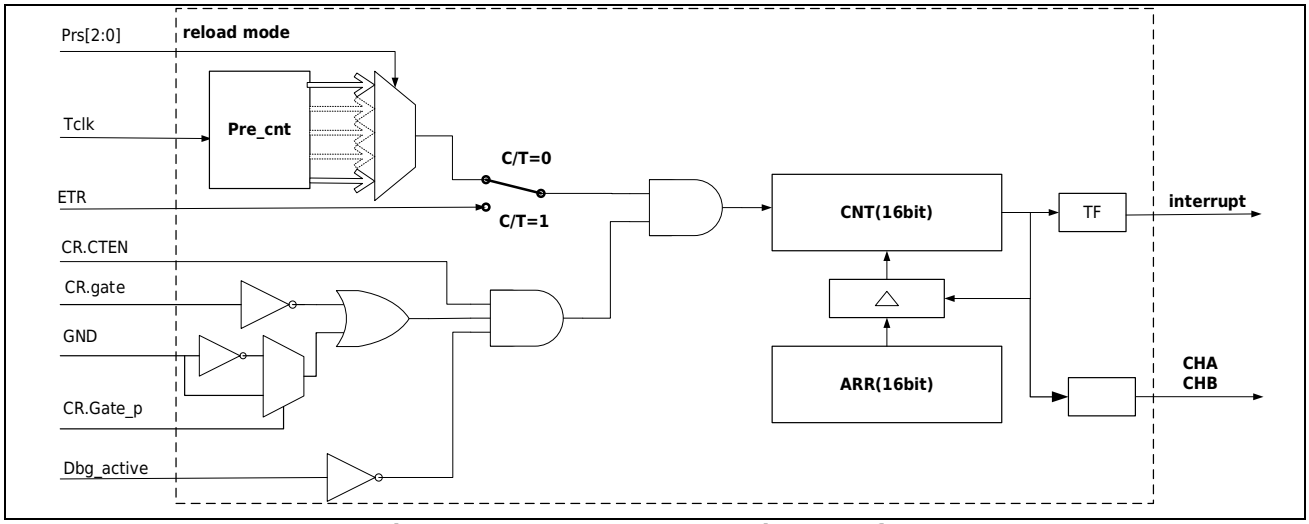


Figure 11-3 Heavy Duty Counting Waveform

11.3.4.2 count waveform

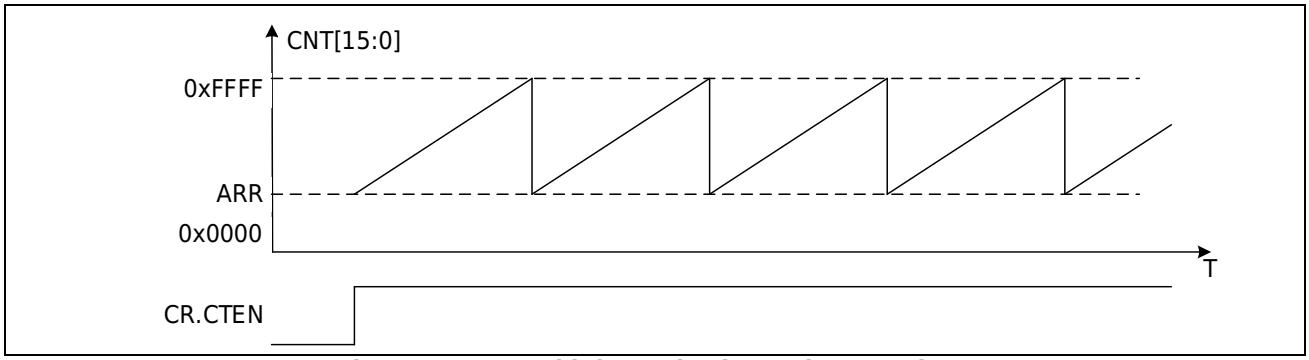


Figure 11-4 16 - bit heavy load counting waveform

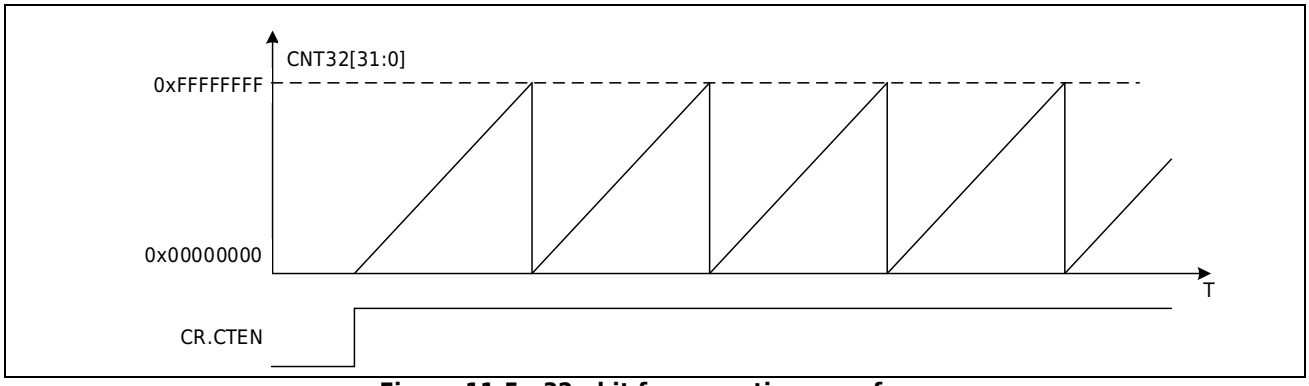


Figure 11-5 32 - bit free counting waveform

11.3.4.3 counting function

Counting functions are used to determine the number of times an event occurs. In the count function, the counter increments on 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.

11.3.4.4 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.

11.3.4.5 Timing diagram

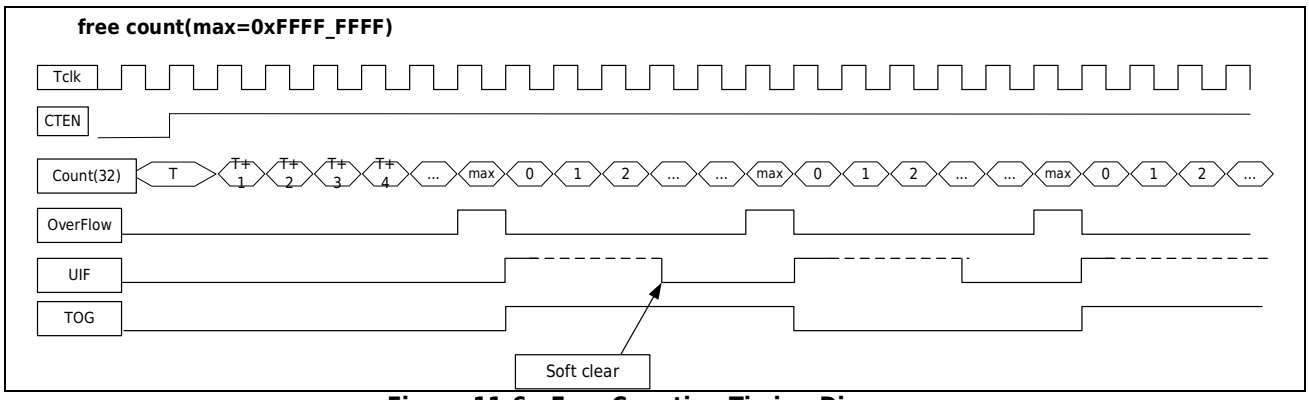


Figure 11-6 Free Counting Timing Diagram

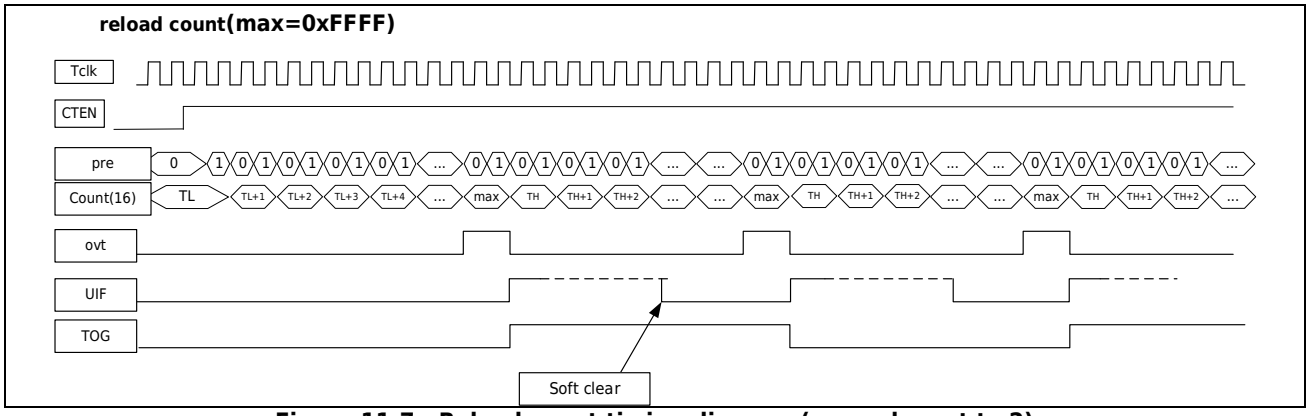


Figure 11-7 Reload count timing diagram (prescaler set to 2)

11.3.4.6 Buzzer function

The function of driving 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:

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

11.3.4.7 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.

11.3.5 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 CMAR, and a capture interrupt CAF will be generated. If the counter overflows, an overflow flag will be generated. Setting overflow interrupt enable will generate overflow interrupt.

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.

11.3.5.1 PWC functional block diagram

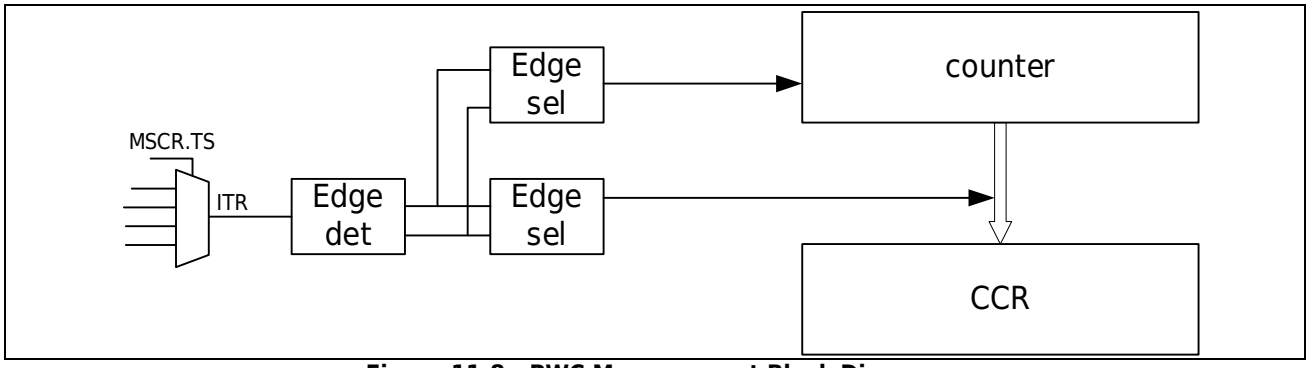


Figure 11-8 PWC Measurement Block Diagram

MSCR.TS	trigger selection 000: the signal ETFP 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)

11.3.5.2 PWC waveform measurement timing diagram

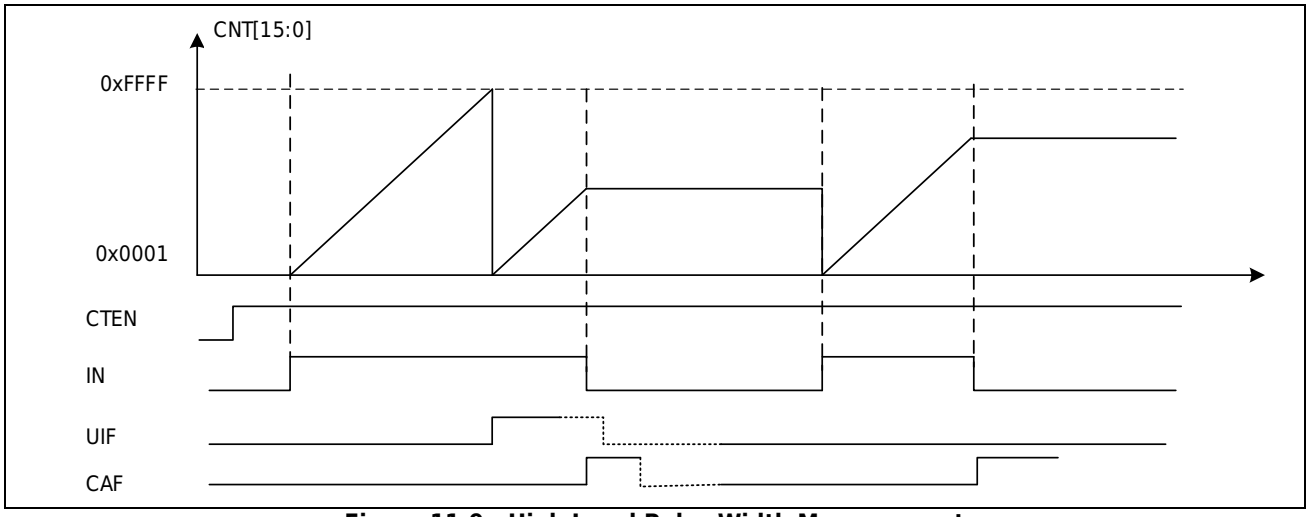


Figure 11-9 High Level Pulse Width Measurement

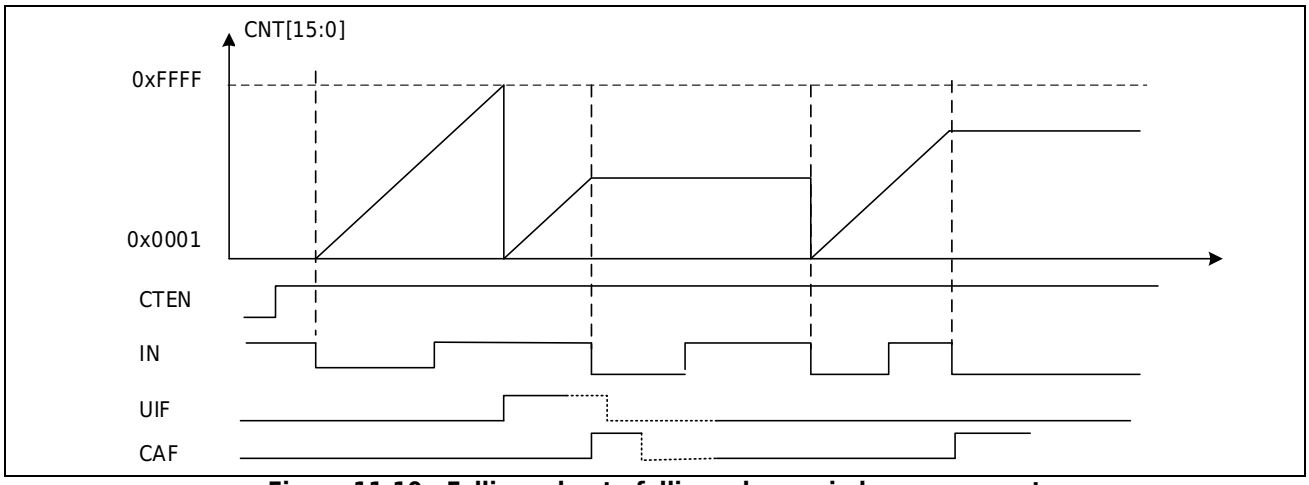


Figure 11-10 Falling edge to falling edge period measurement

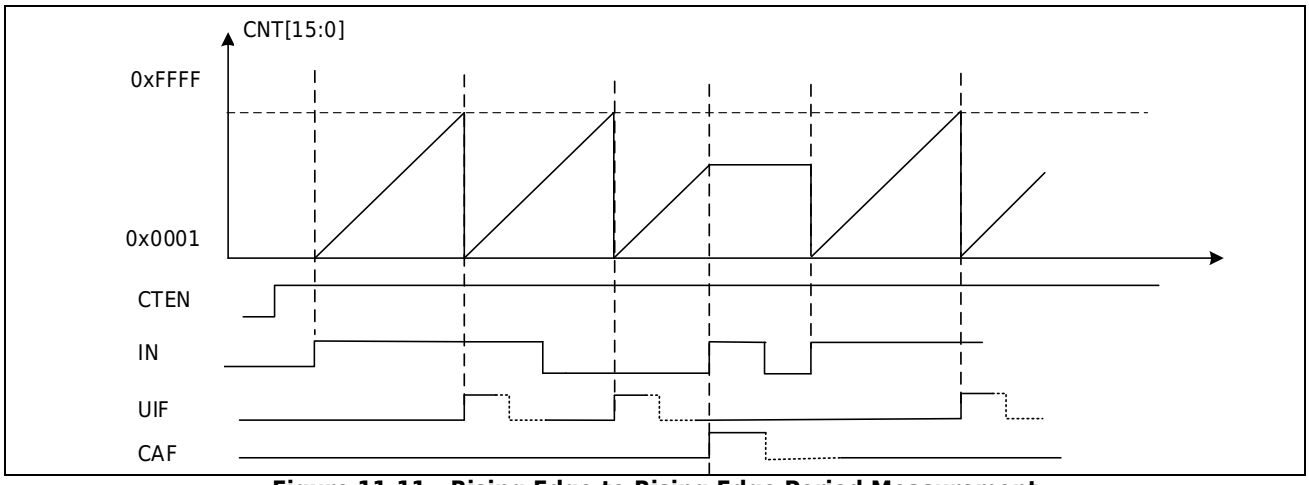


Figure 11-11 Rising Edge to Rising Edge Period Measurement

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 filtering and input reverse can be selected

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

	ITR0	ITR1	ITR2	ITR3
ATIM3	-	-	GTIM_OV	-

Note: Refer to the description of register MSCR.MMS for TRGO output.

11.3.5.3 PWC one-shot mode

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

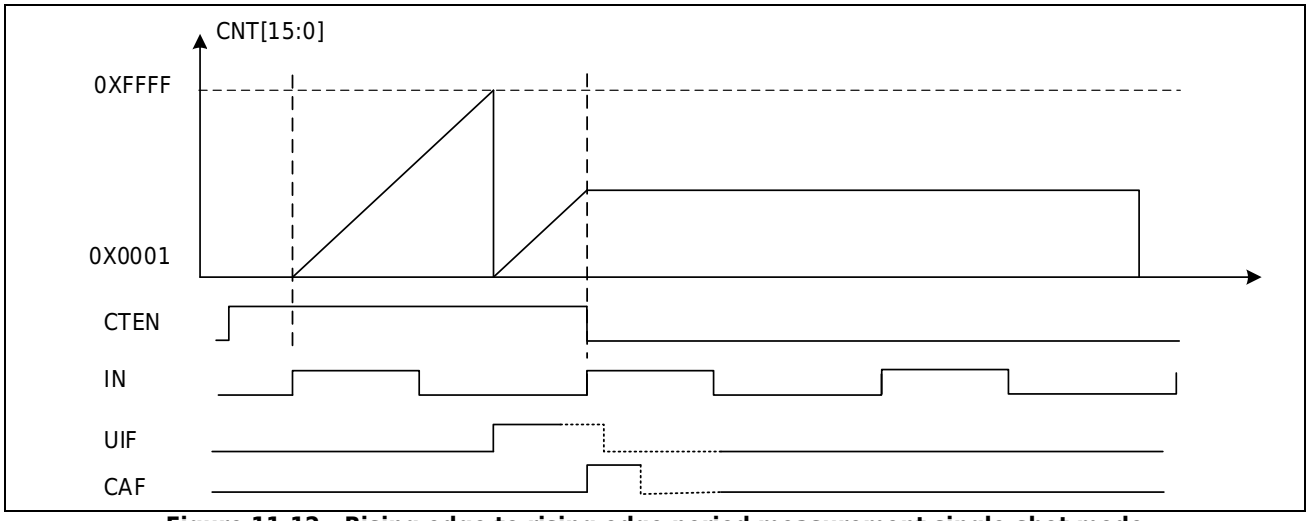


Figure 11-12 Rising edge to rising edge period measurement single-shot mode

11.3.5.4 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 low 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

11.3.6 Mode 2/3 compare capture mode

11.3.6.1 counter

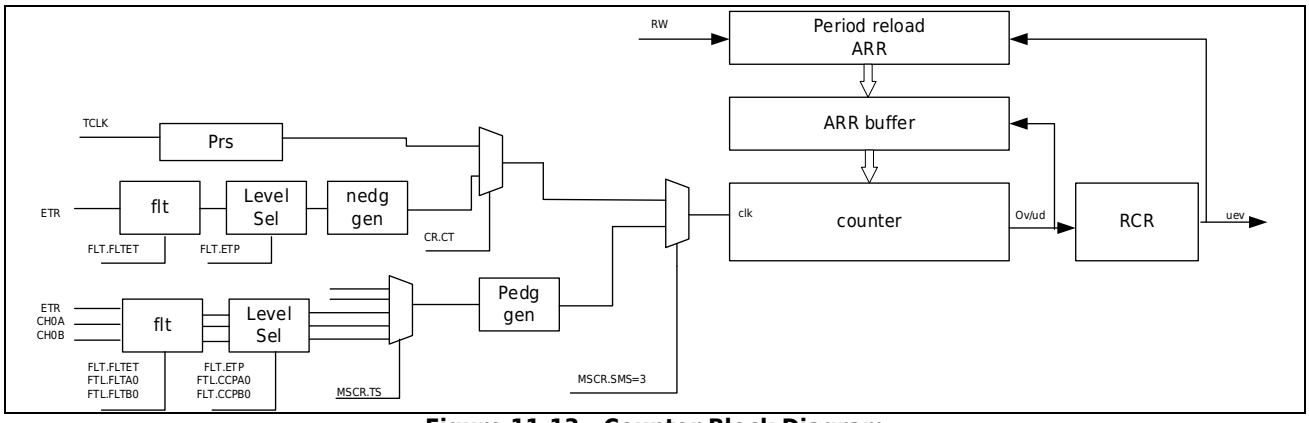


Figure 11-13 Counter Block Diagram

The main part of the counter is a 16 -bit counter with associated autoload 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
- Autoload Register ARR
- Repeat count register RCR
- Clock selection control registers FLT, CR0, MSCR, CR

The autoload register 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 autoload register is immediately updated to the cache register. When the timer is running and the cache function is valid, the value written to the autoload register will not be updated to the cache register immediately, and will be updated from the autoload register to the cache register only after the event is updated.

Clock selection and gating function, trigger function, reset function refer to the chapter of mode 2/3 slave mode.

11.3.6.2 counter waveform

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

When 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. If a repeat counter is used, an update event (UEV) is generated when the number of repetitions counted up reaches the number programmed in the repeat counter register plus one (ATIMx_RCR+1). Otherwise, an update event will be generated every time the counter overflows.

An update event will also be 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:

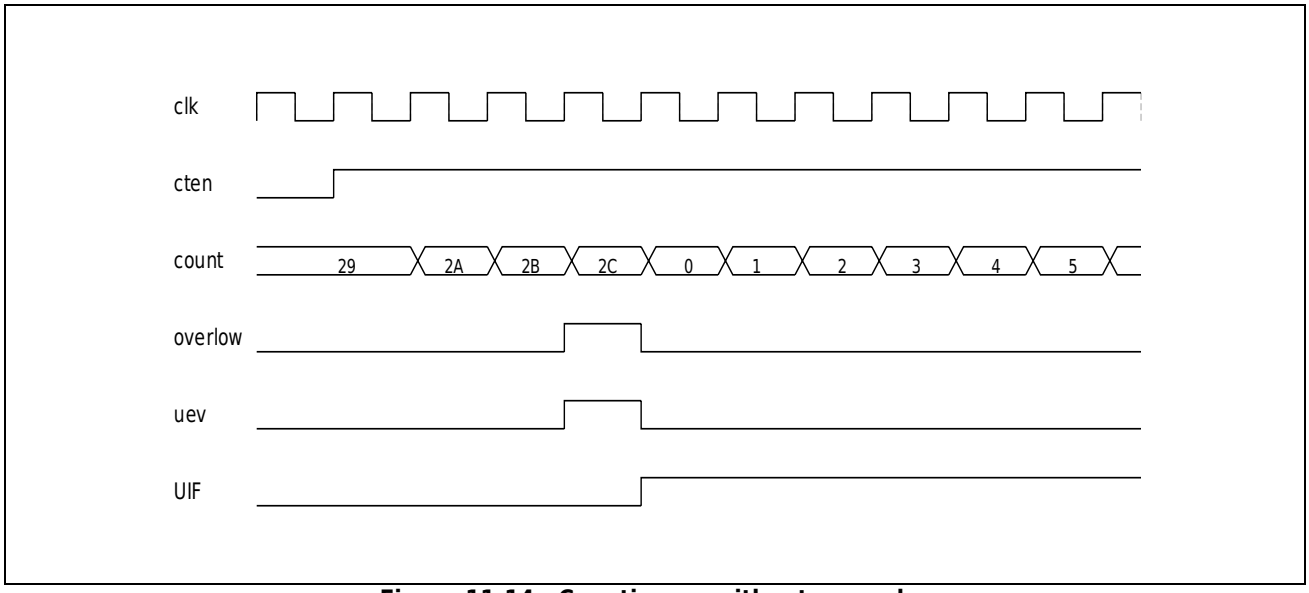


Figure 11-14 Counting up without prescaler

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

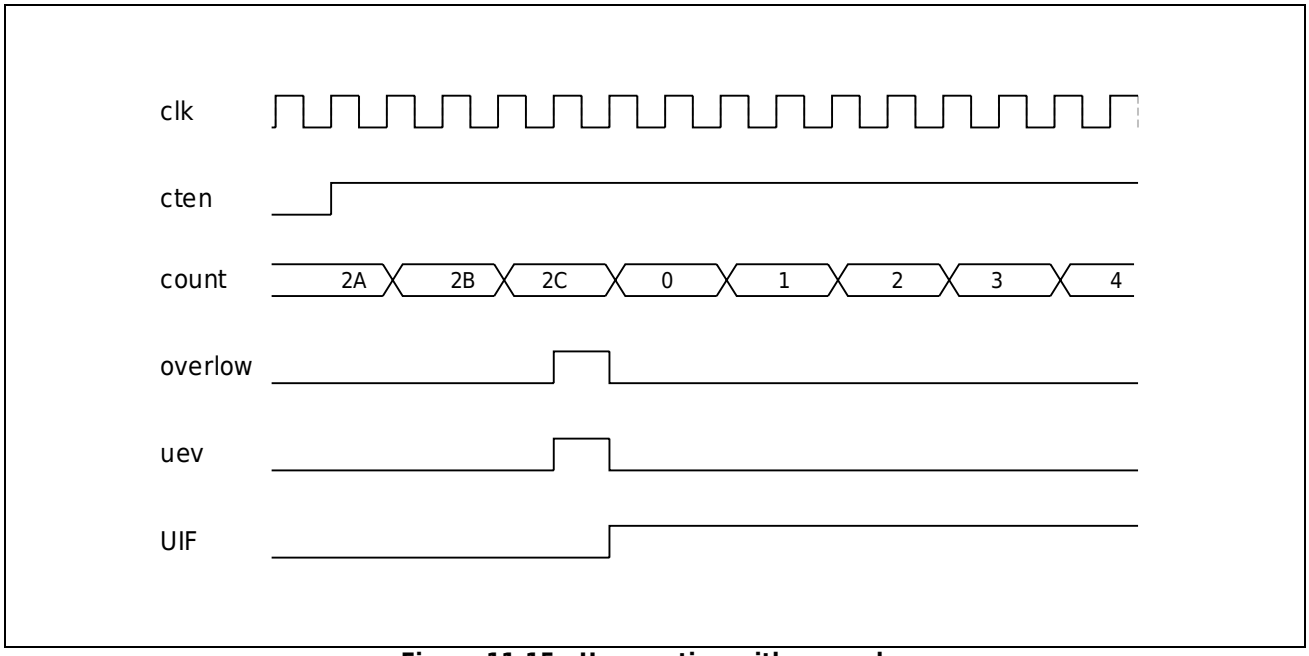


Figure 11-15 Up counting with prescaler

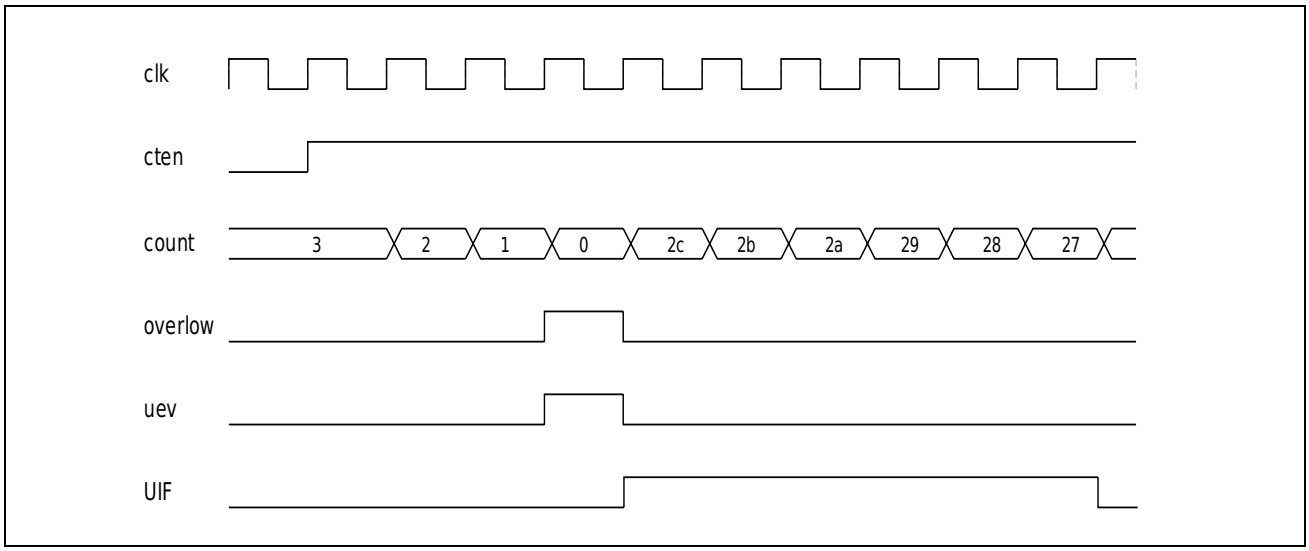


Figure 11-16 Down counting without prescaler

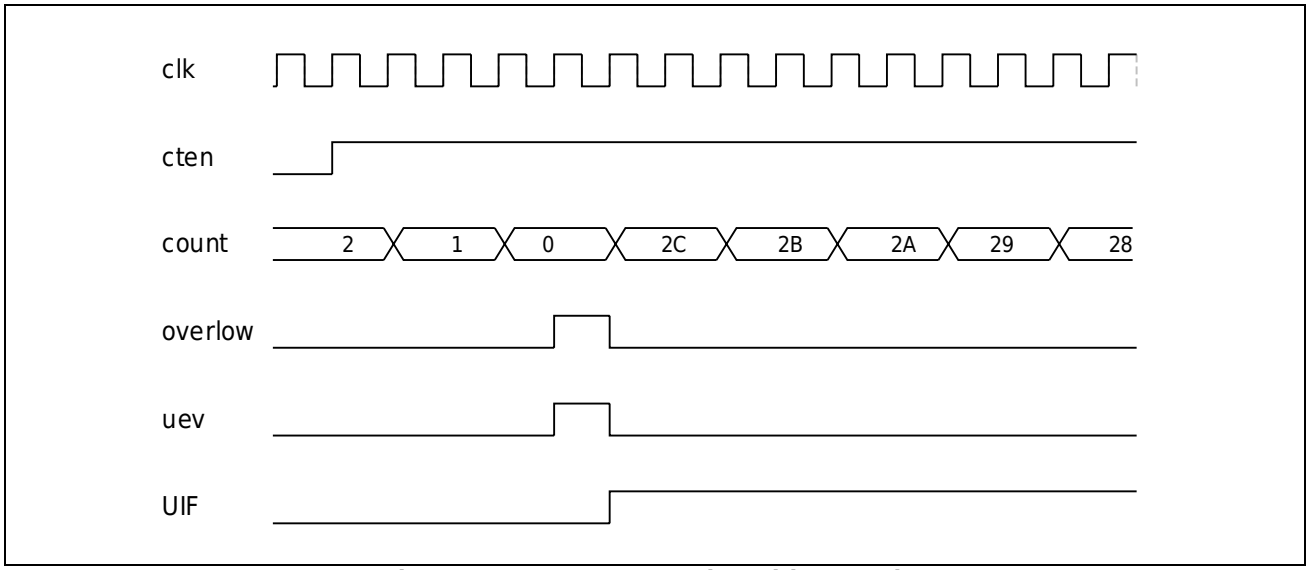


Figure 11-17 Down counting with prescaler

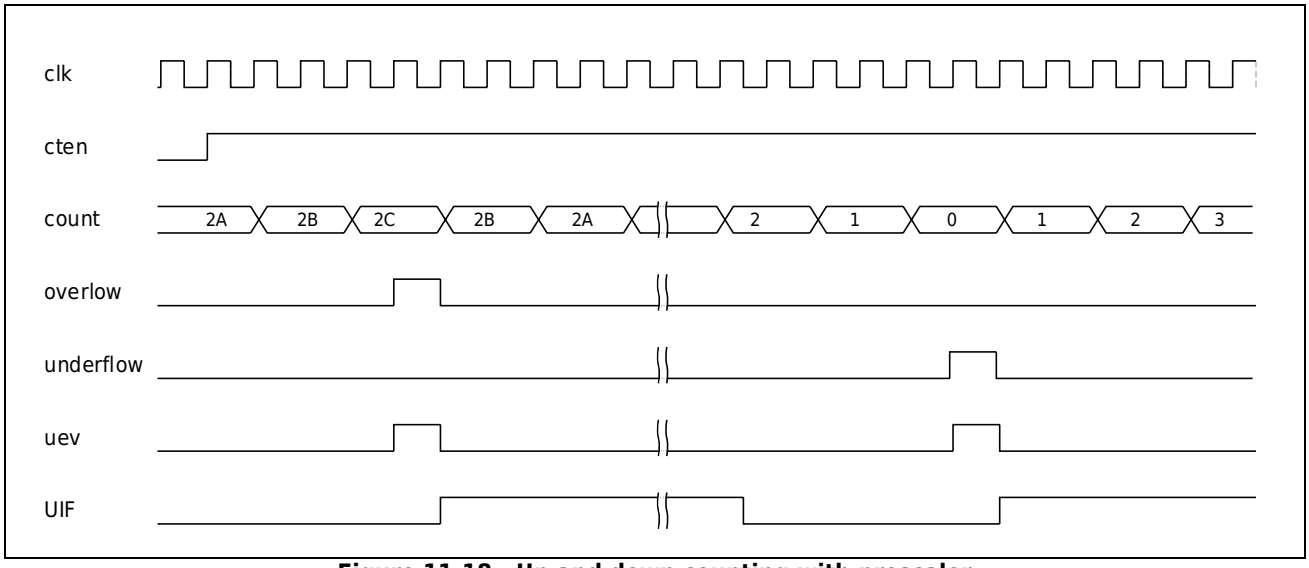


Figure 11-18 Up and down counting with prescaler

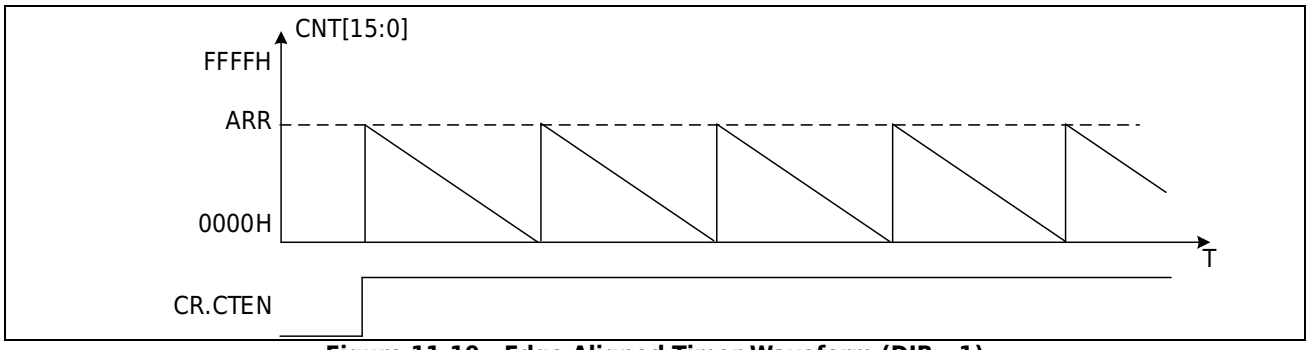


Figure 11-19 Edge Aligned Timer Waveform (DIR =1)

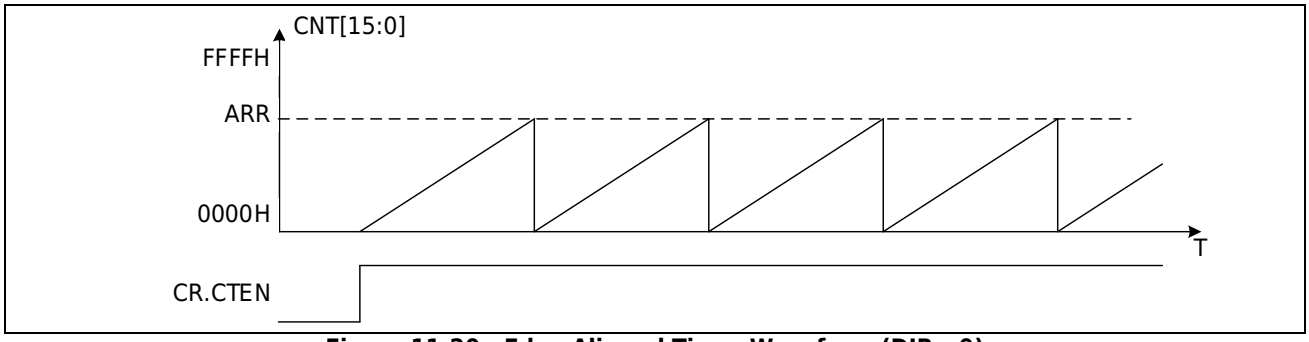


Figure 11-20 Edge Aligned Timer Waveform (DIR =0)

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. 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.

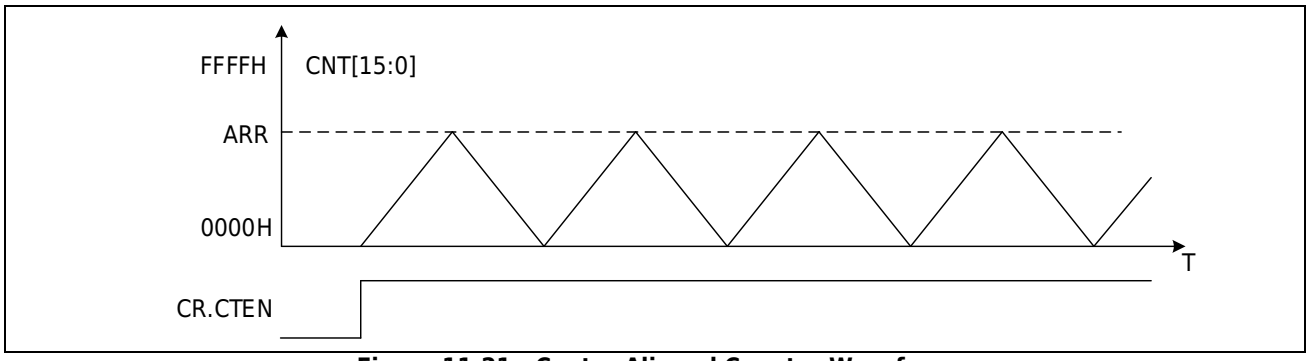


Figure 11-21 Center Aligned Counter Waveform

11.3.6.3 Repeat count

A repeating counter uses the counter's overflow to count down. When 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 cycle reload register is updated to the cycle cache register. In compare mode, the value of the compare register is updated to the compare cache register.

The repeat counter is decremented when the following conditions are true:

- 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

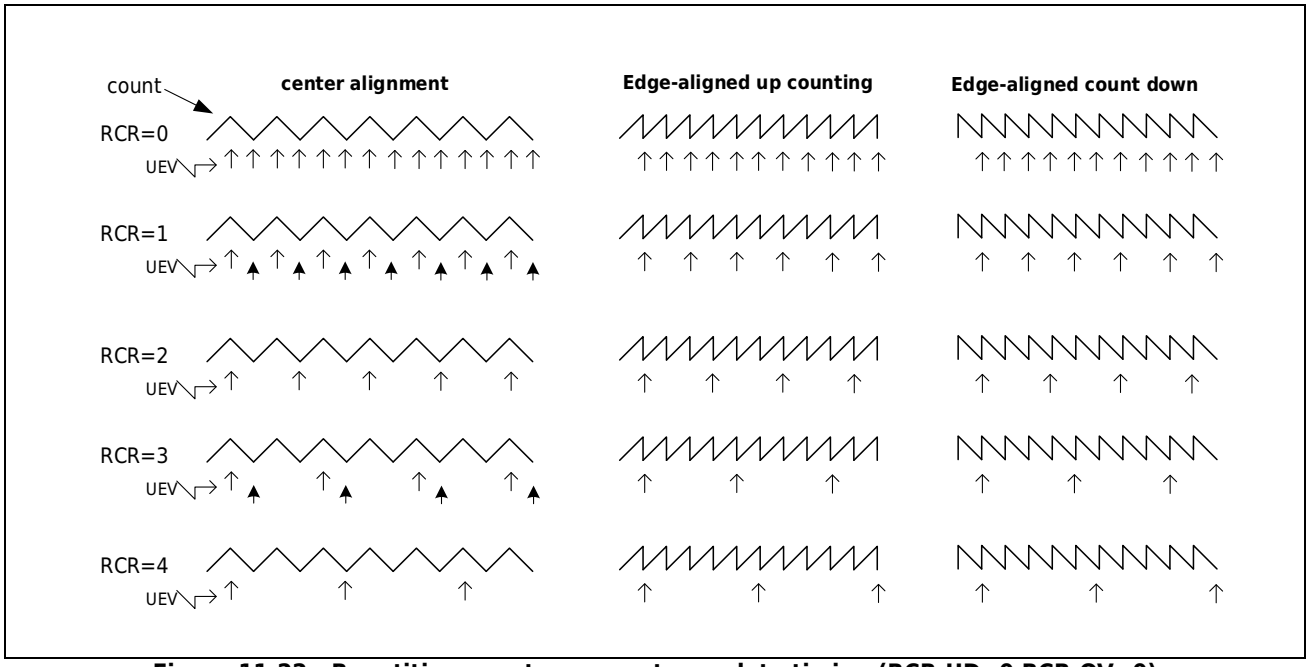


Figure 11-22 Repetition counter generates update timing (RCR.UD=0 RCR.OV=0)

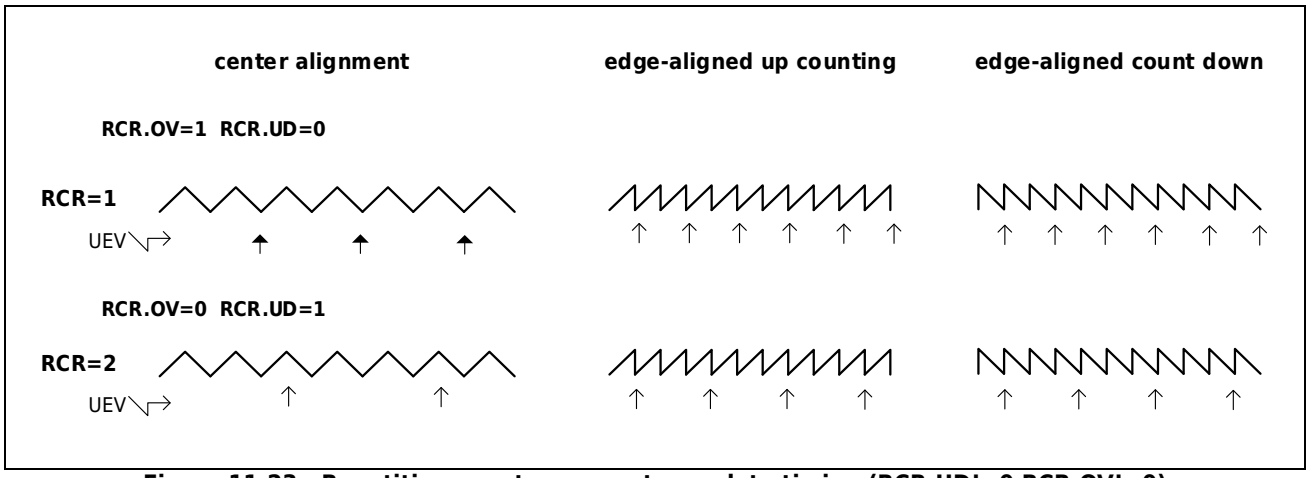


Figure 11-23 Repetition counter generates update timing (RCR.UD!=0 RCR.OV!=0)

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.

11.3.6.4 data cache

Both the auto-reload data ARR and the comparison register can be configured with a cache function. When the cache function is valid, the written period value ARR and comparison value CCR 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:

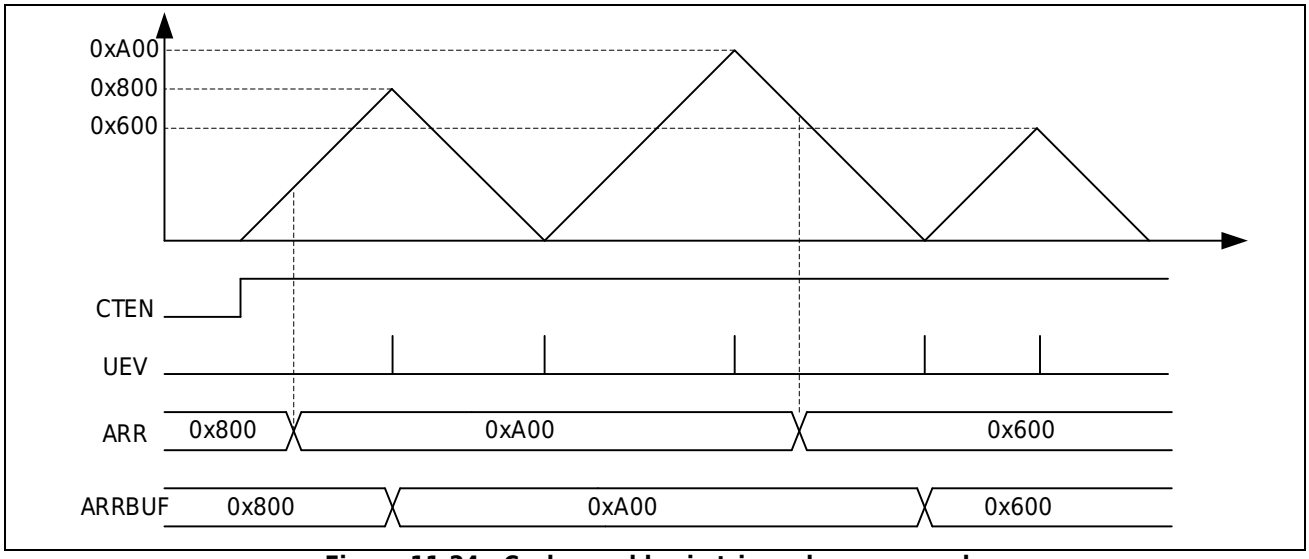


Figure 11-24 Cache enables in triangular wave mode

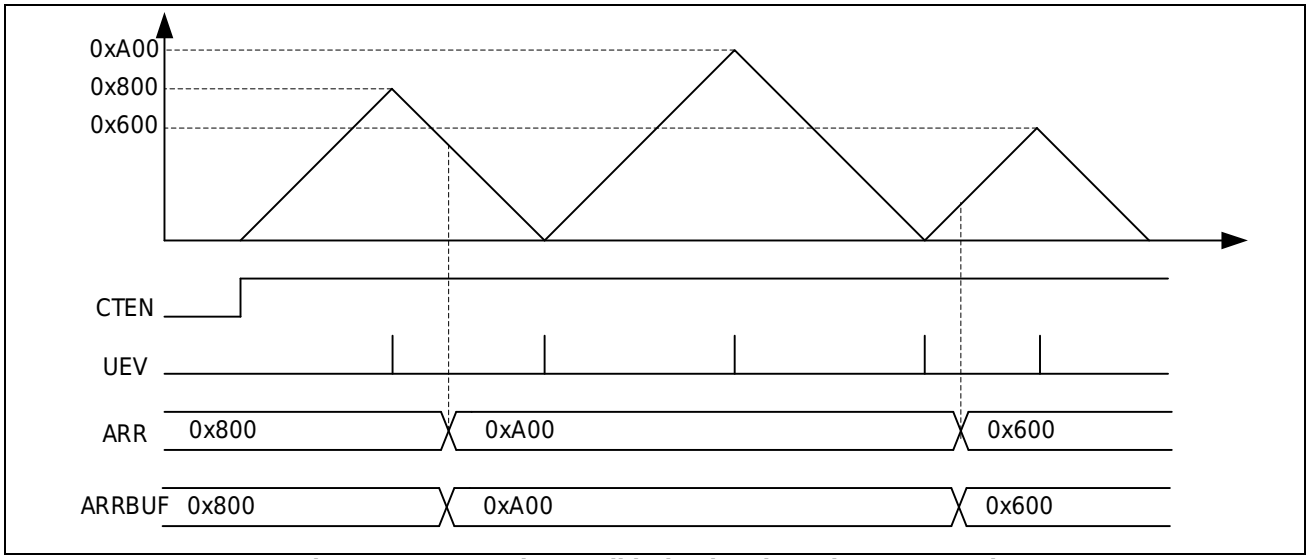


Figure 11-25 Cache Invalidation in Triangular Wave Mode

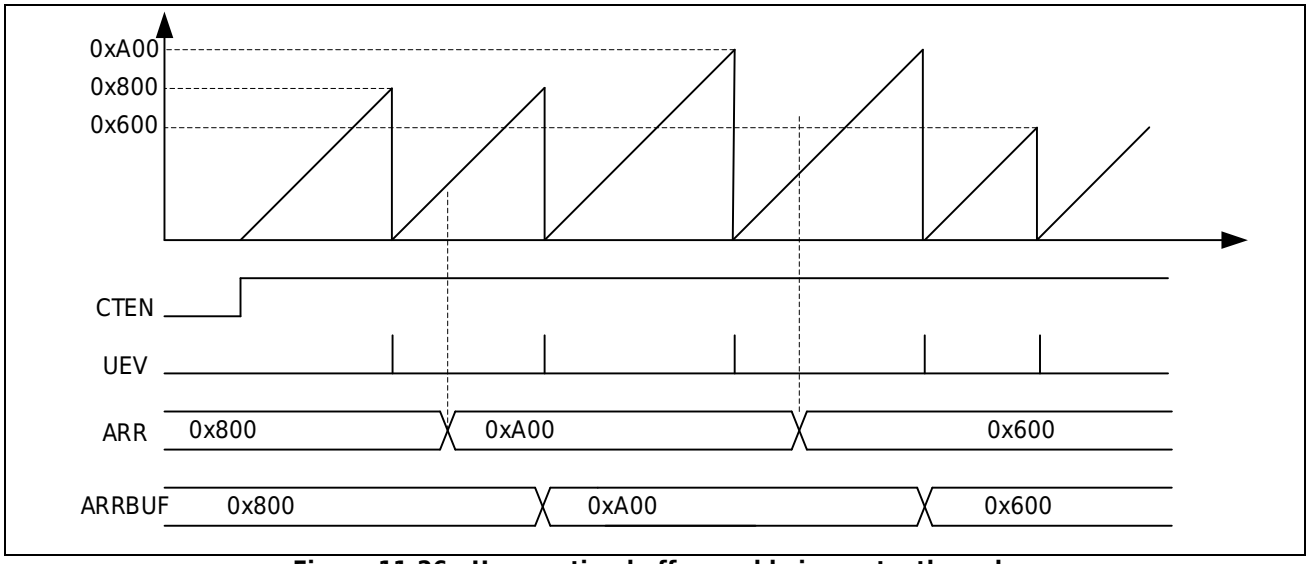


Figure 11-26 Up counting buffer enable in sawtooth mode

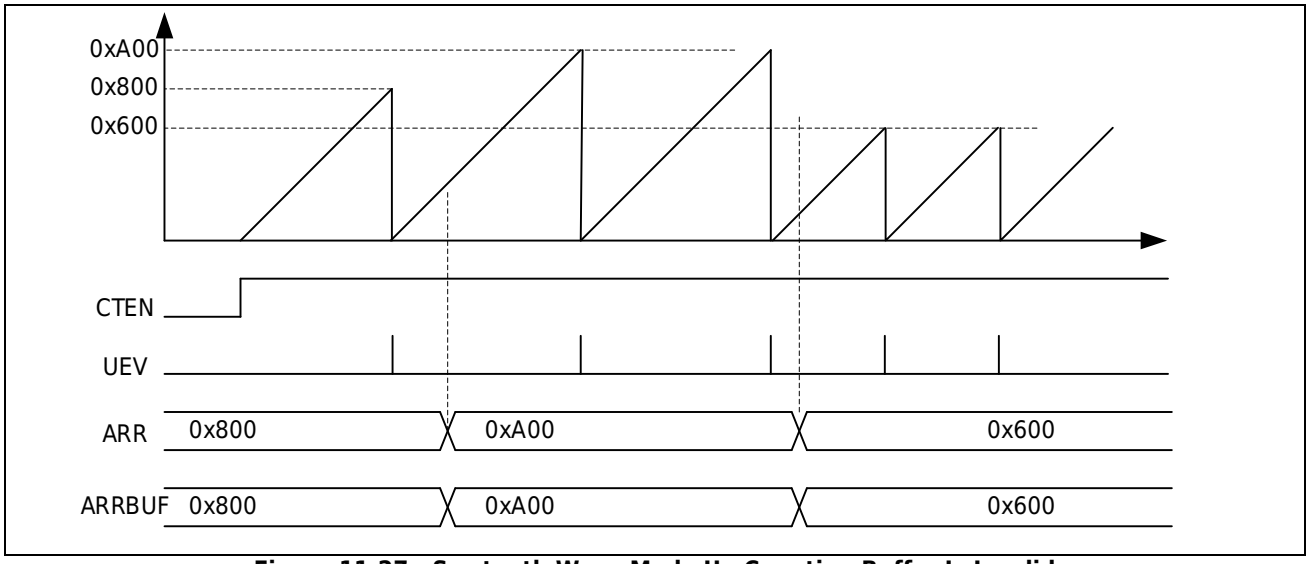


Figure 11-27 Sawtooth Wave Mode Up Counting Buffer Is Invalid

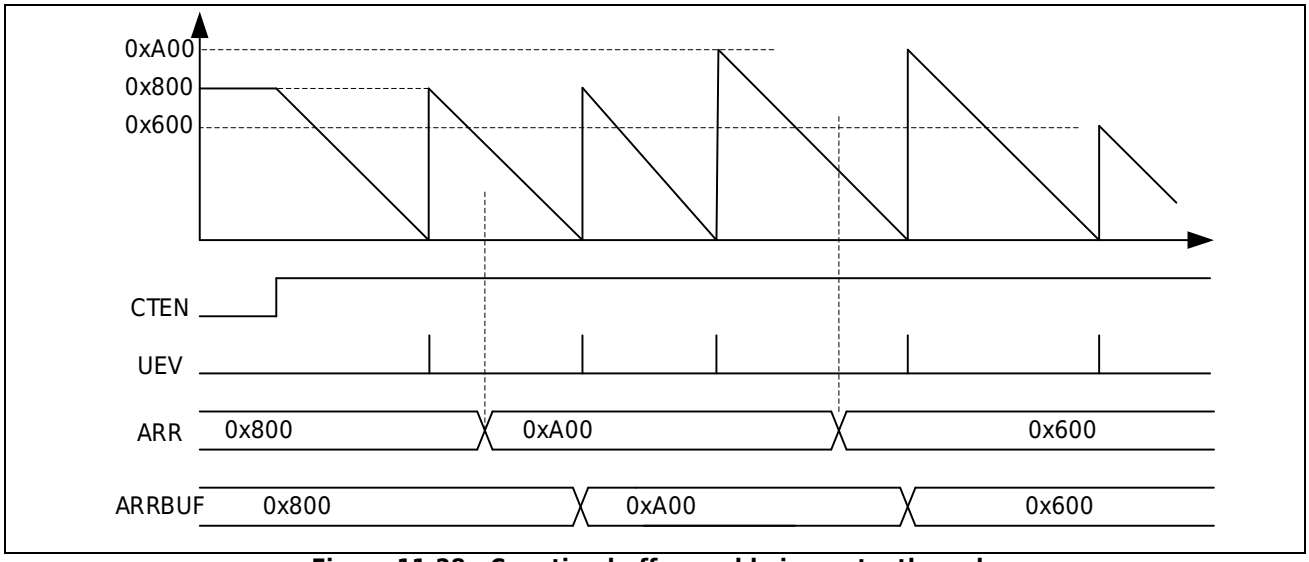


Figure 11-28 Counting buffer enable in sawtooth mode

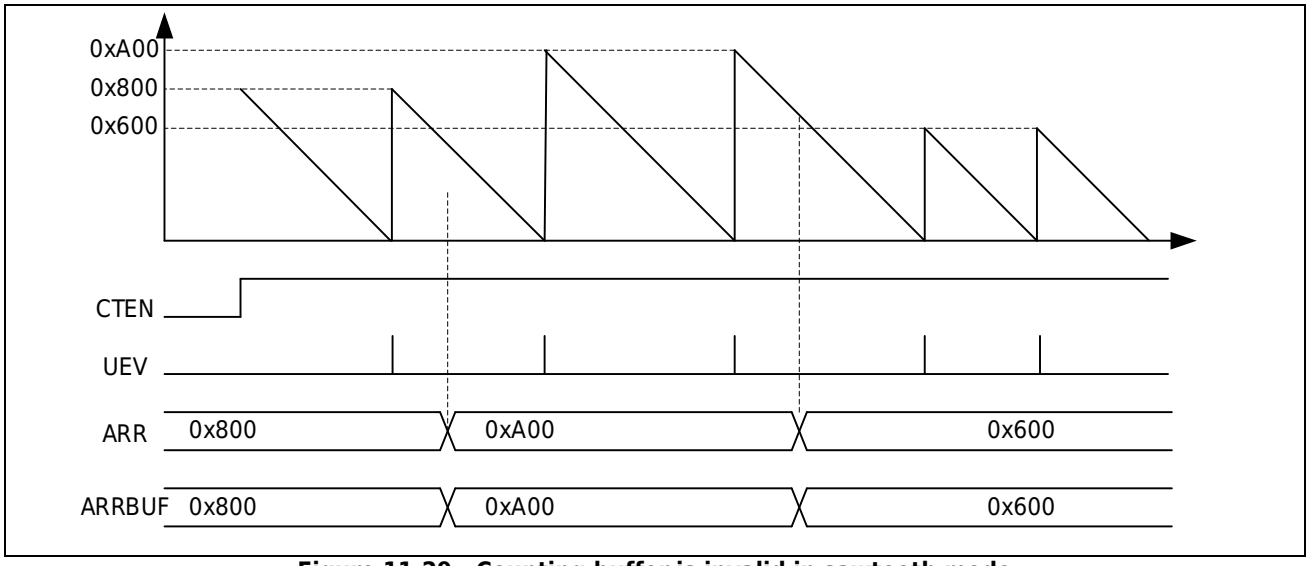


Figure 11-29 Counting buffer is invalid in sawtooth mode

In triangular wave mode and sawtooth counting mode, if the cache is not enabled, 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.

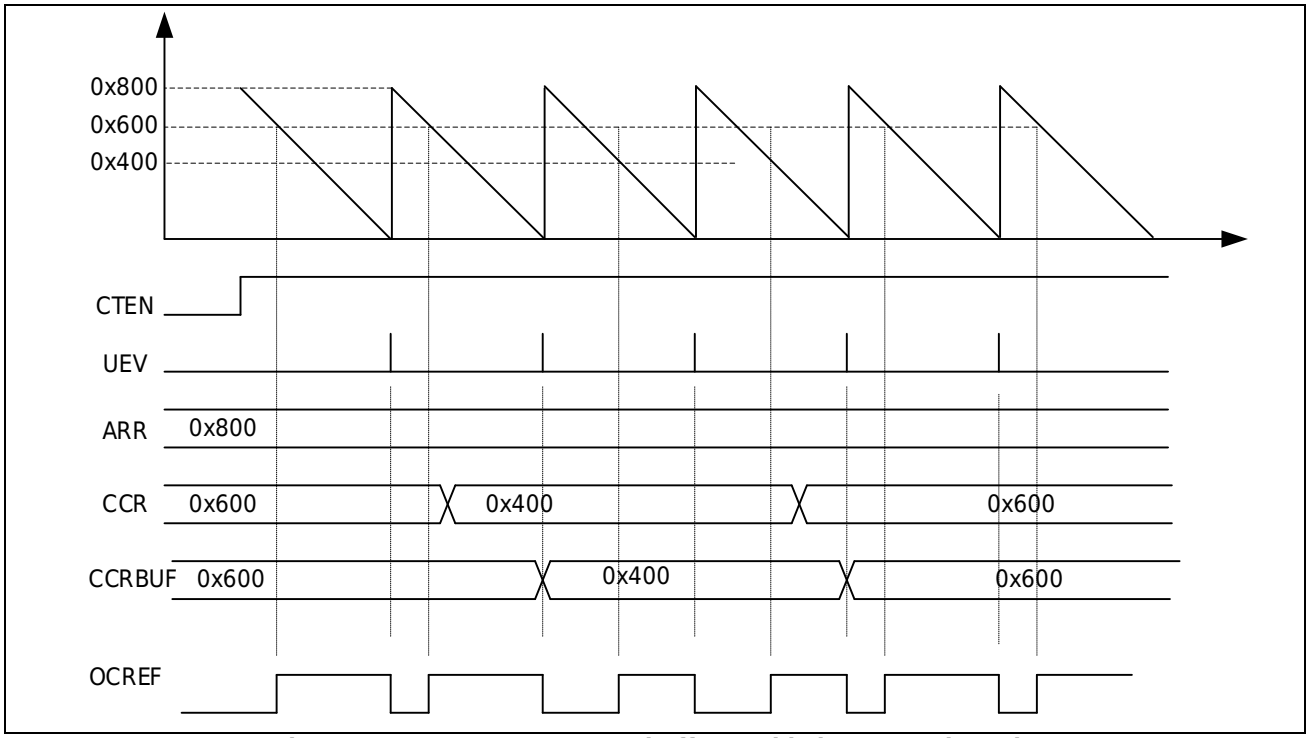


Figure 11-30 Count compare buffer enable in sawtooth mode

11.3.6.5 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.

The comparison output of OCREFB can only use single-point comparison, and the comparison register CCRB is used to control the comparison output of OCREFB.

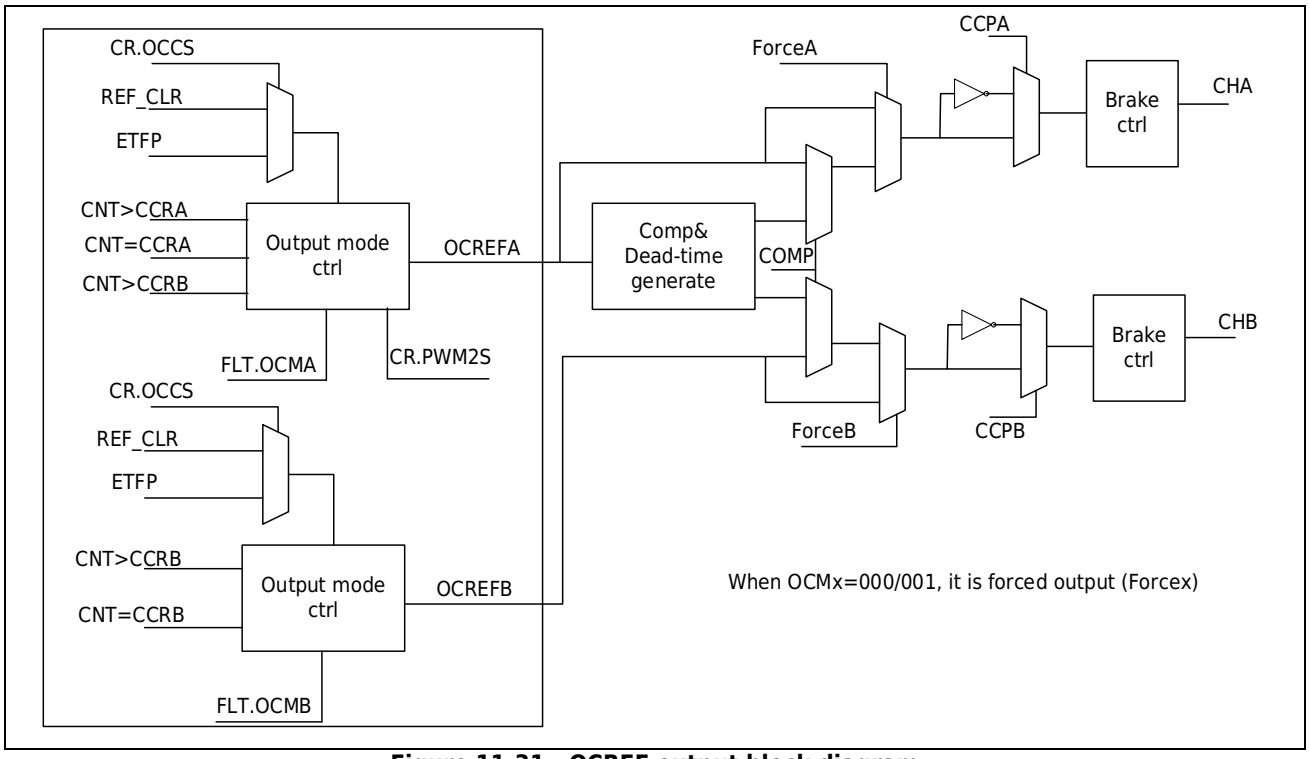


Figure 11-31 OCREF output block diagram

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<CCRxy outputs high, and when counting down, CNT>CCRxy outputs low level

Two-point comparison:

1) Counting CCRxA<CNT≤CCRxB output on the sawtooth wave is low level

2) Counting under sawtooth wave CCRxA<CNT≤CCRxB output is high level

3) Triangular wave up counting CNT<CCRxA output high, down counting 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≤CNT<CCRxB output on the sawtooth wave is high level

2) Counting under sawtooth wave CCRxA≤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.

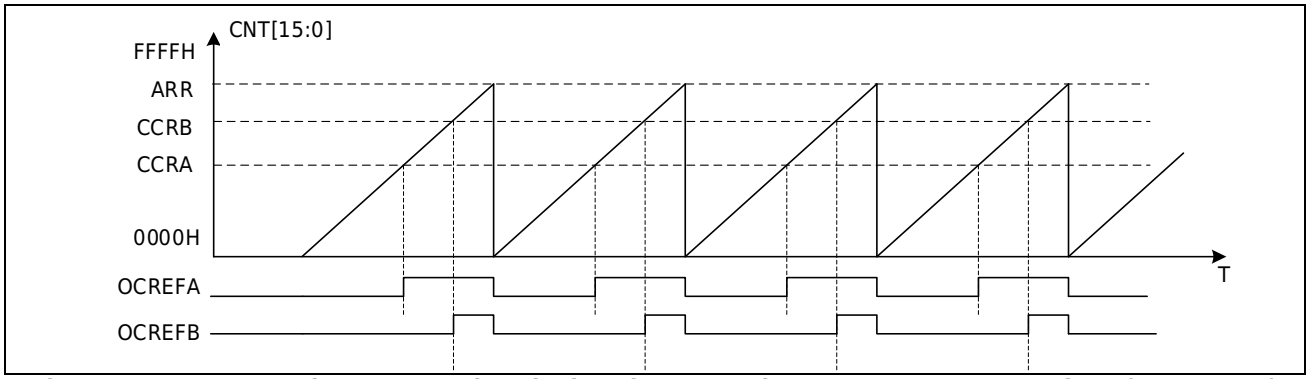


Figure 11-32 Sawtooth wave counting single point comparison OCREF output waveform (OCMx=111)

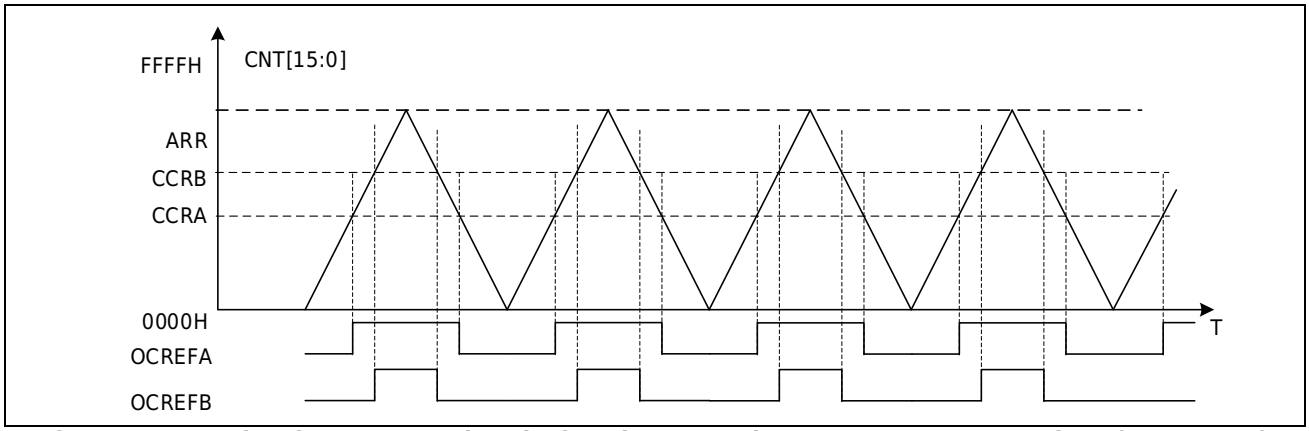


Figure 11-33 Triangle wave counting single point comparison OCREF output waveform (OCMx=111)

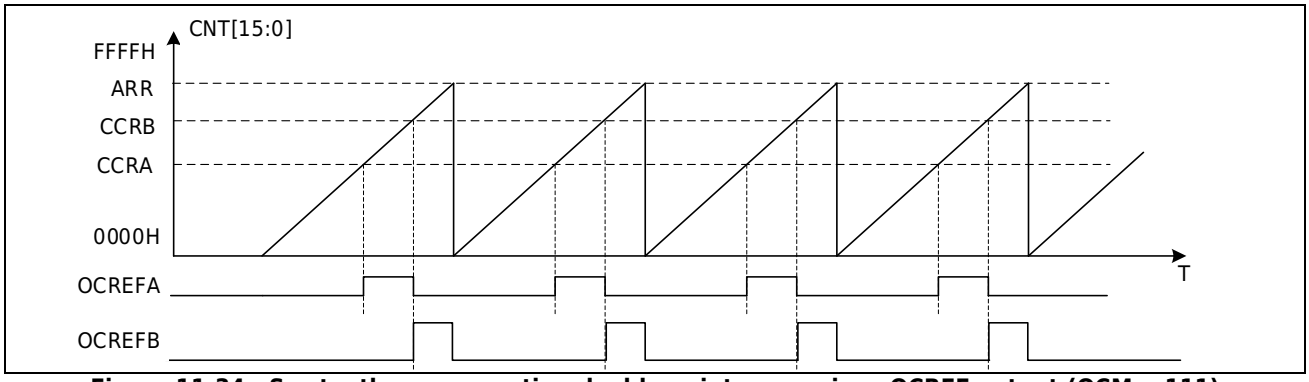


Figure 11-34 Sawtooth wave counting double-point comparison OCREF output (OCMx=111)

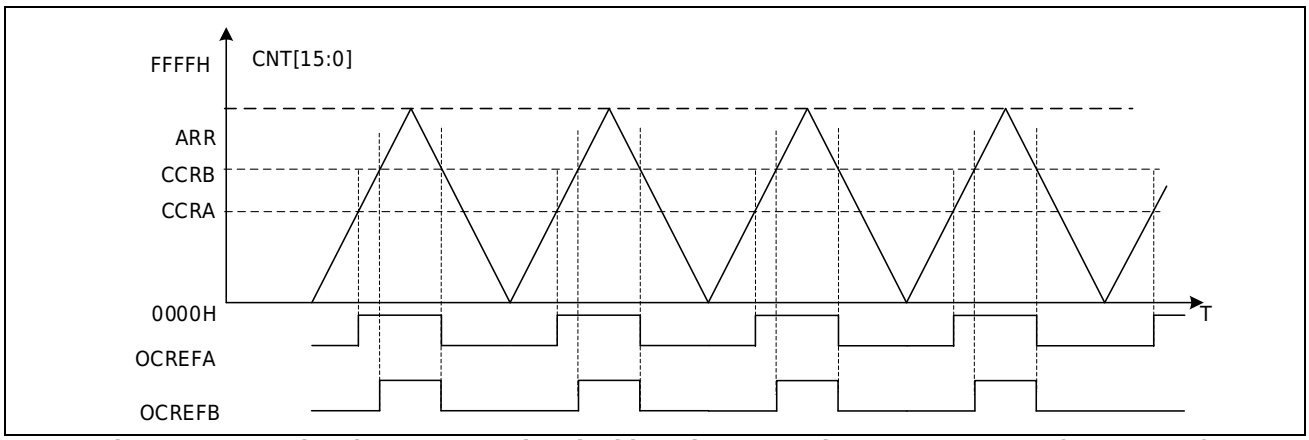


Figure 11-35 Triangle wave counting double-point comparison OCREF output (OCMx=111)

11.3.6.6 Independent PWM output

The output of 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.

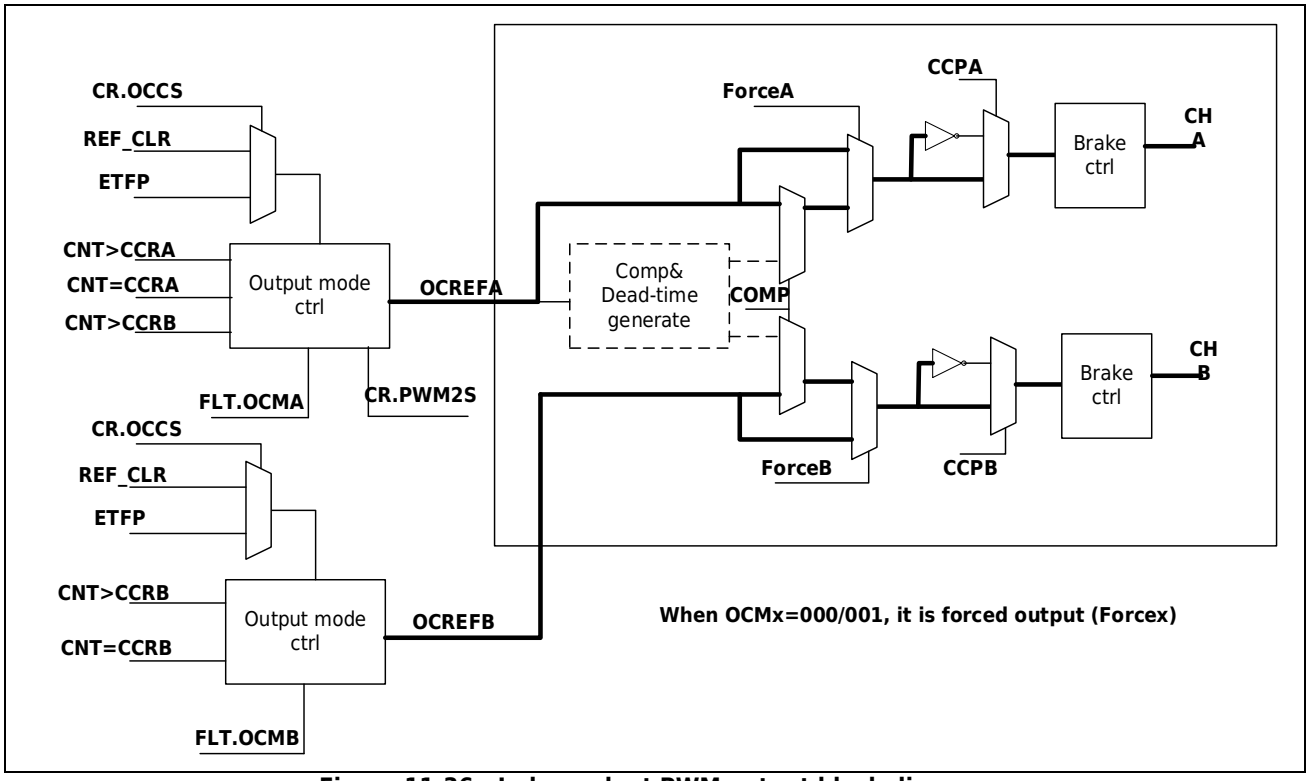


Figure 11-36 Independent PWM output block diagram

Relationship between PWM output and OCREF

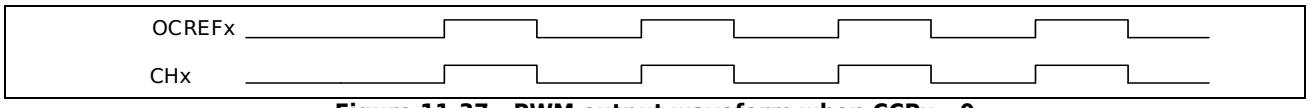


Figure 11-37 PWM output waveform when CCPx= 0

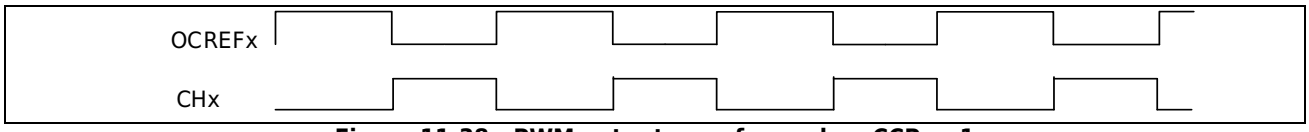


Figure 11-38 PWM output waveform when CCPx= 1

11.3.6.7 Complementary PWM output

The output of 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.

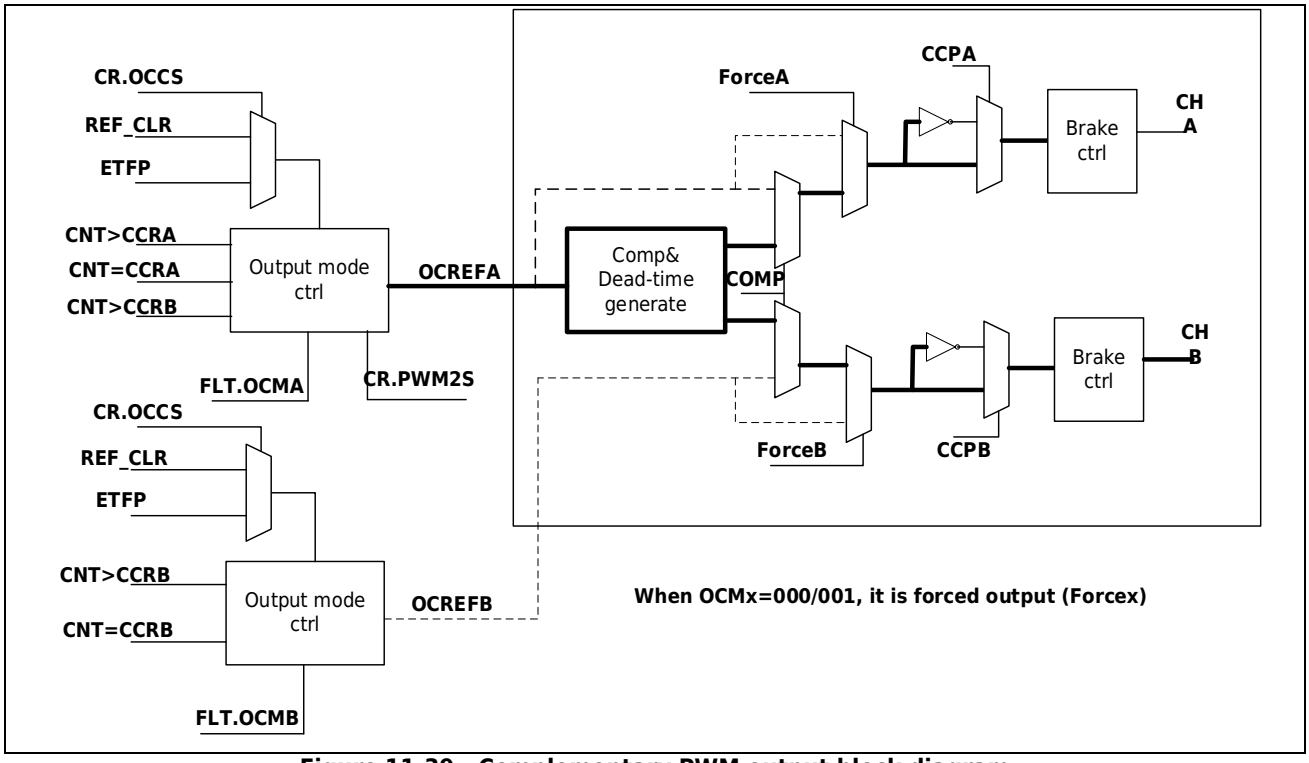


Figure 11-39 Complementary PWM output block diagram

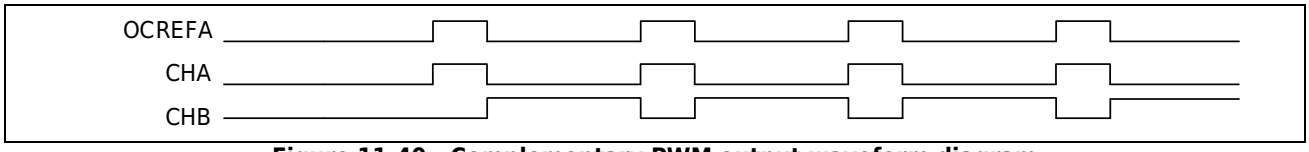


Figure 11-40 Complementary PWM output waveform diagram

11.3.6.8 PWM output with dead zone

The dead-time function can be set in complementary PWM output mode.

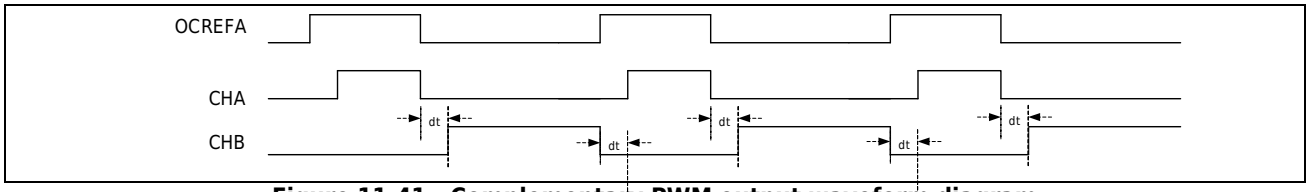


Figure 11-41 Complementary PWM output waveform diagram

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

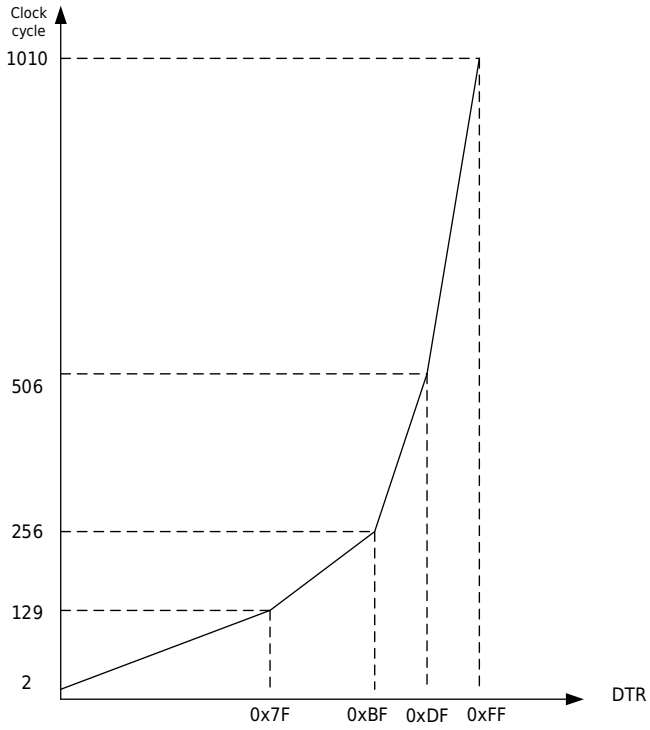


Figure 11-42 Dead Time

11.3.6.9 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 by an excitation signal to start, and can generate a programmable pulse width pulse after a programmable delay.

The counter can be started by the slave mode controller. Waveforms 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. In this way, the counter will automatically stop when the next update event UEV occurs.

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 ≦ ARR (special attention, 0<CCRxy)
- When counting down: CNT>CCRxy

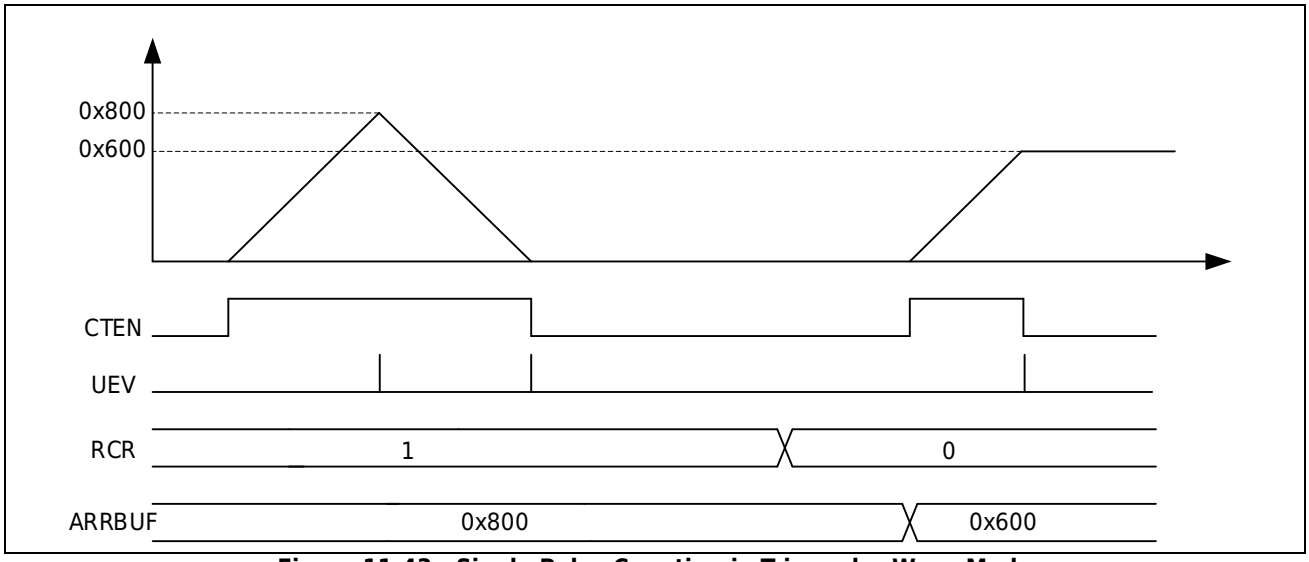


Figure 11-43 Single Pulse Counting in Triangular Wave Mode

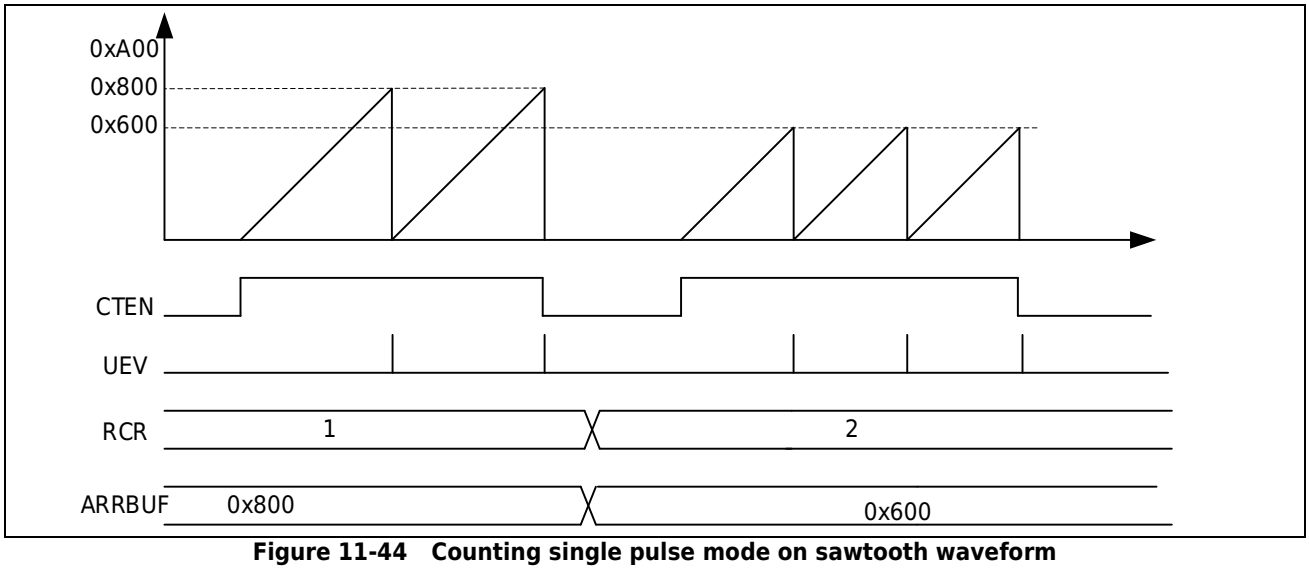


Figure 11-44 Counting single pulse mode on sawtooth waveform

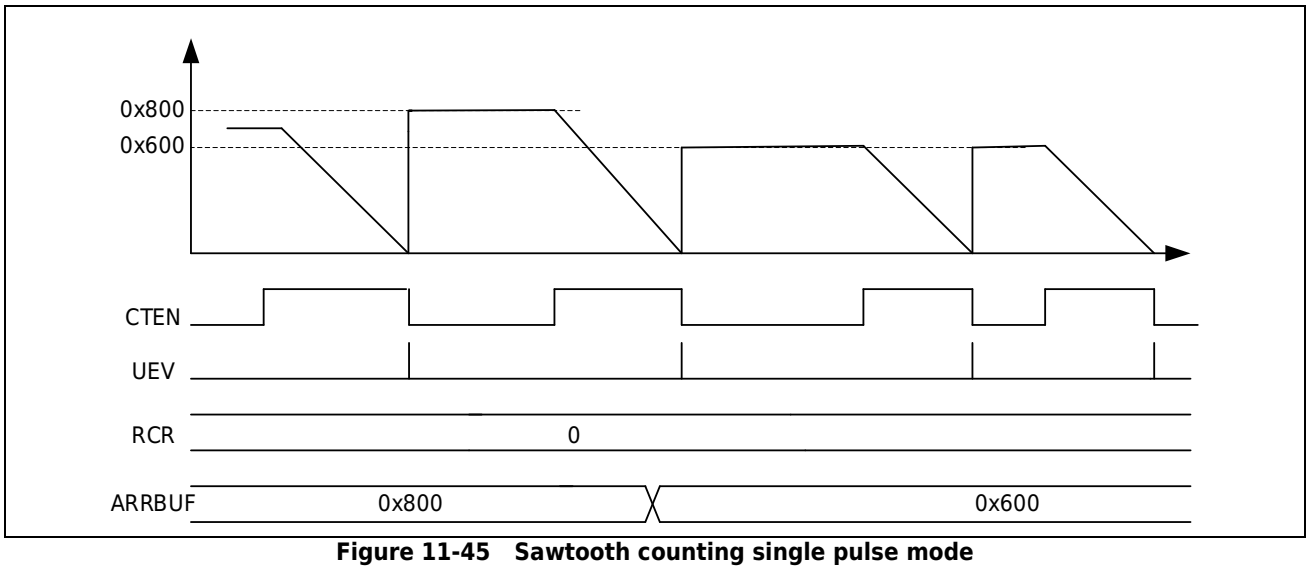


Figure 11-45 Sawtooth counting single pulse mode

11.3.6.10 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 and no output

When M23CR.CIS=2'b01, compare match when counting up

M23CR.CIS=2'b10 compare match when counting down

When M23CR.CIS=2'b11, the comparison 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 has no output

When CRCHx.CISB=2’b01, compare match when counting on channel x

When CRCHx.CISB=2’b10, compare match when channel x counts down

When CRCHx.CISB=2’b11, when channel x counts up and down, both comparisons match

The B channel comparison match is controlled separately for more flexible triggering of the ADC. For details, refer to the Timer Trigger ADC chapter.

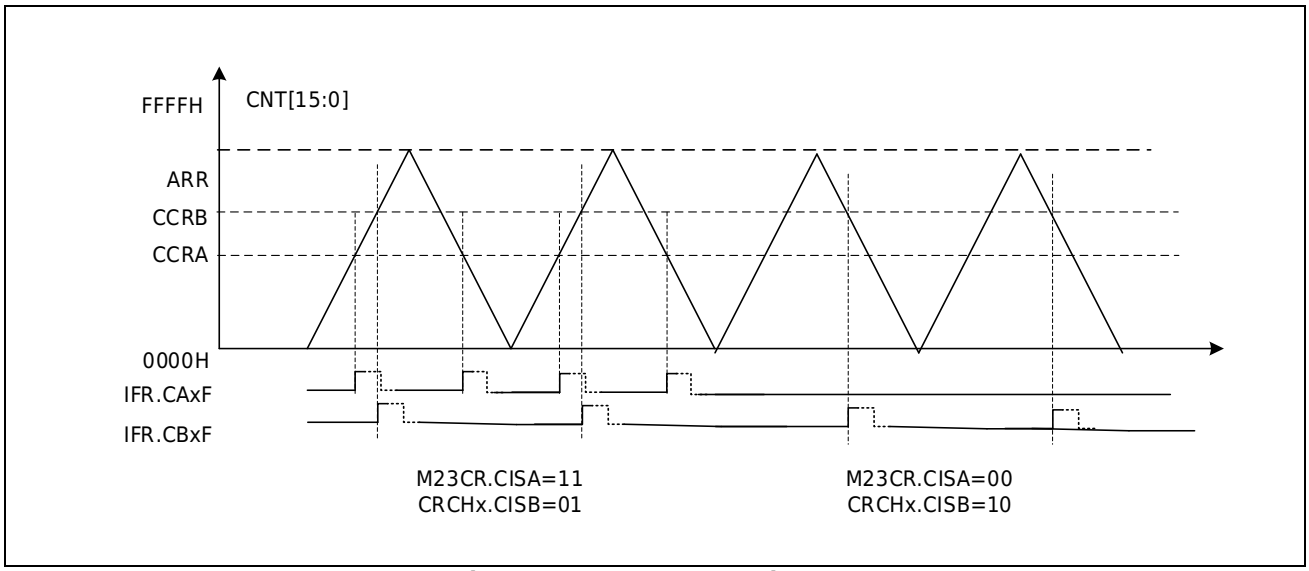


Figure 11-46 Interrupt Diagram

11.3.6.11 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.

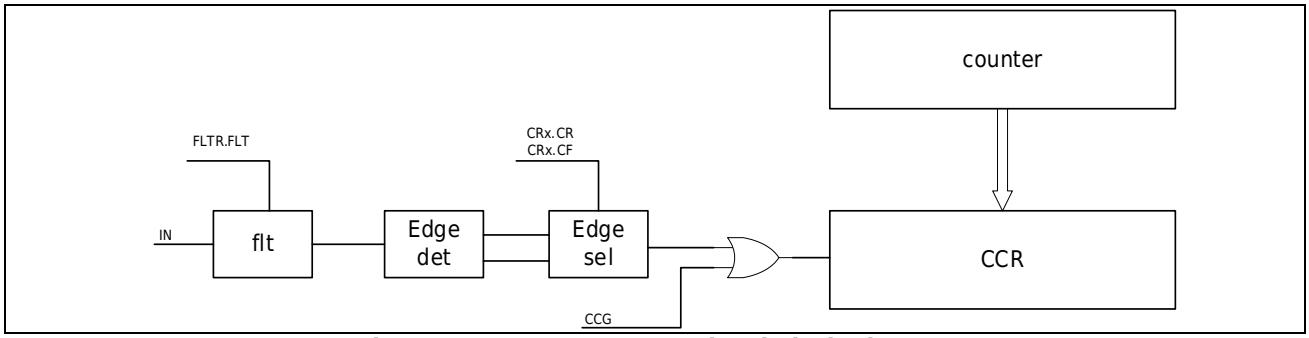


Figure 11-47 Capture Functional Block Diagram

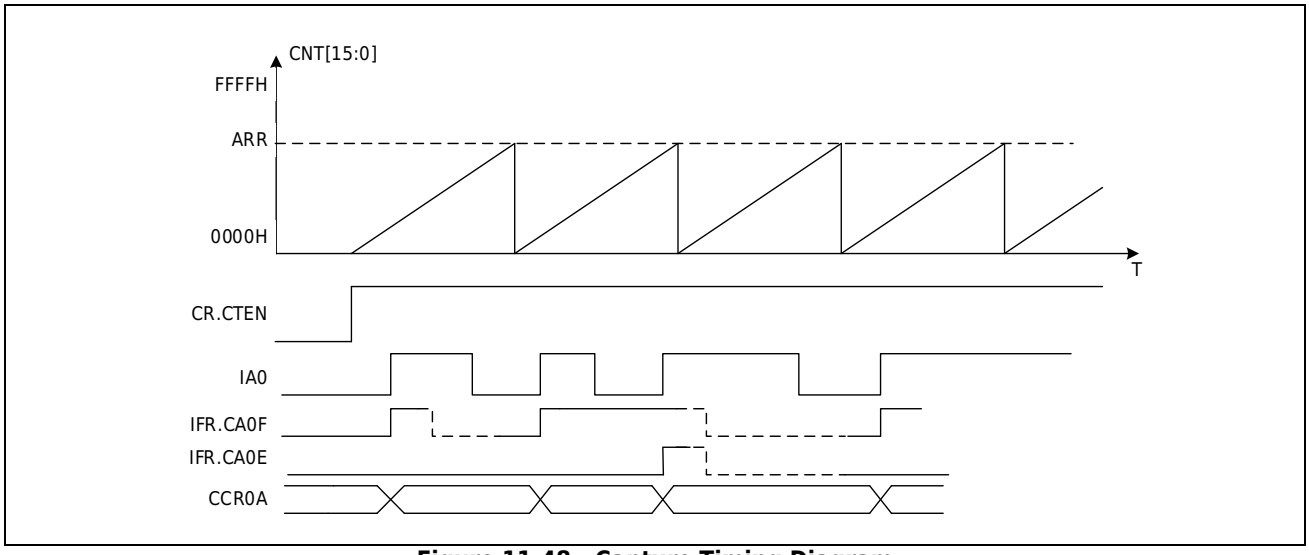


Figure 11-48 Capture Timing Diagram

The capture input of CH0A selects CH0A input or the XOR input of CH0A, CH1A, and CH2A through MSCR.IA0S.

IA0S	0	1
ATIM3	CHA0	CHA0 CHA1 CHA2 XOR input

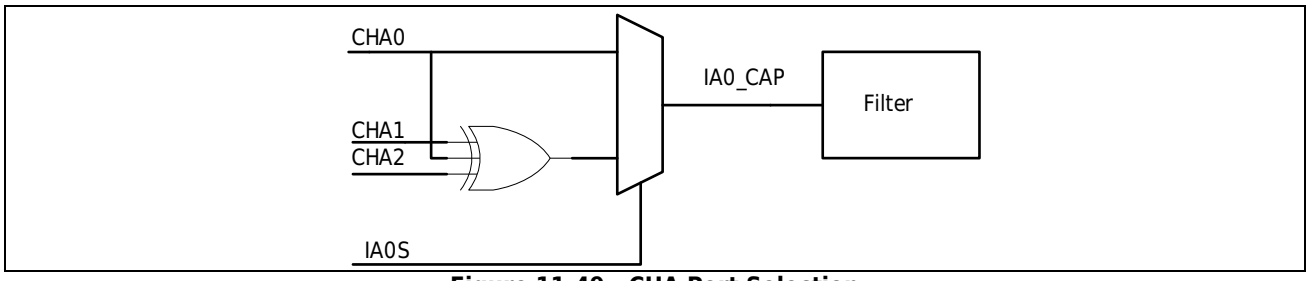


Figure 11-49 CHA Port Selection

The capture input of CH0B selects the CH0B input or the internal trigger signal selected by MSCR.TS through MSCR.IB0S.

<table>
<tr><td>IB0S</td><td>0</td><td>1</td></tr>
<tr><td>ATIM3</td><td>CHB0</td><td>Internal trigger MSCR.TS select signal
<table>
<tr><td>MCSCR.TS</td><td>Channel B capture input</td></tr>
<tr><td>000</td><td>ETR filter phase selection output signal, filter phase selection can be configured</td></tr>
<tr><td>001</td><td>Interconnect ITR0</td></tr>
<tr><td>010</td><td>Interconnect ITR1</td></tr>
<tr><td>011</td><td>Interconnect ITR2</td></tr>
<tr><td>100</td><td>Interconnect ITR3</td></tr>
<tr><td>101</td><td>CH0A edge</td></tr>
<tr><td>110</td><td>CH0A filter output signal, filter function can be configured</td></tr>
<tr><td>111</td><td>CH0B filter output signal, filter function can be configured</td></tr>
</table></td></tr>
</table>

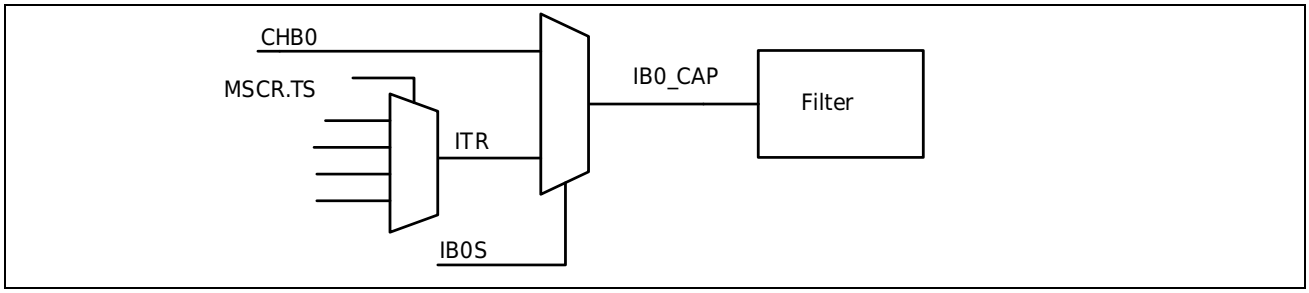


Figure 11-50 CHB Port Selection

11.3.6.12 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. Select input filter 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.

11.3.7 Mode 2/3 Slave Mode

The timer can be synchronized to an external trigger in several modes: reset mode, gated mode and triggered mode.

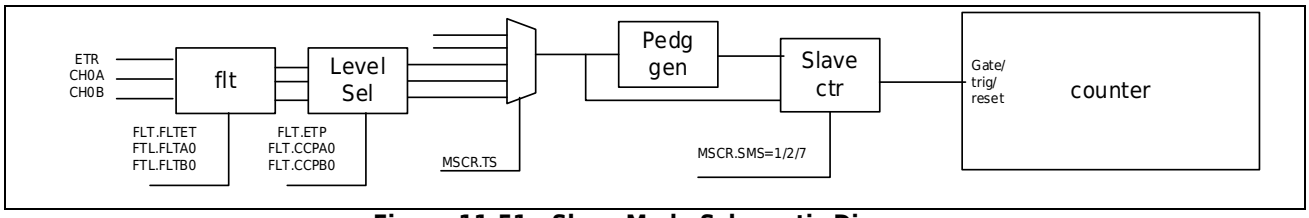


Figure 11-51 Slave Mode Schematic Diagram

Select from mode function by MSCR.SMS;

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

11.3.7.1 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. In gated mode, if CTEN=0, the counter cannot be started regardless of the trigger input level. As long as CH0A is low, the counter starts counting according to the internal clock, and stops counting once CH0A becomes high.

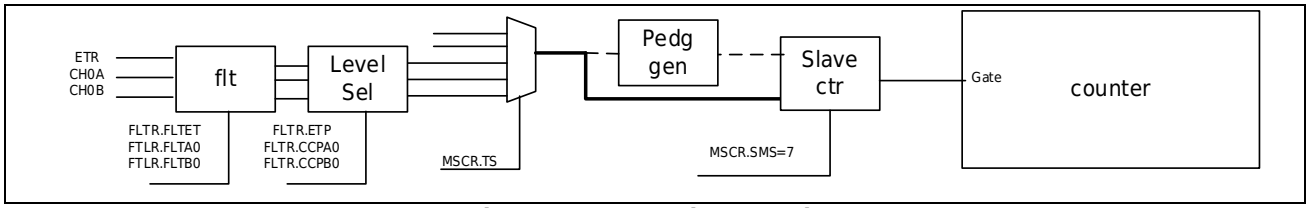


Figure 11-52 Gating Function

11.3.7.2 trigger function

The timer can be started synchronously by using external trigger (CH0A, CH0B, ETR), and the timer can also be started synchronously by using the internal interconnection signal of the timer combined with MSCR.MMS. 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 the 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.

11.3.7.3 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.

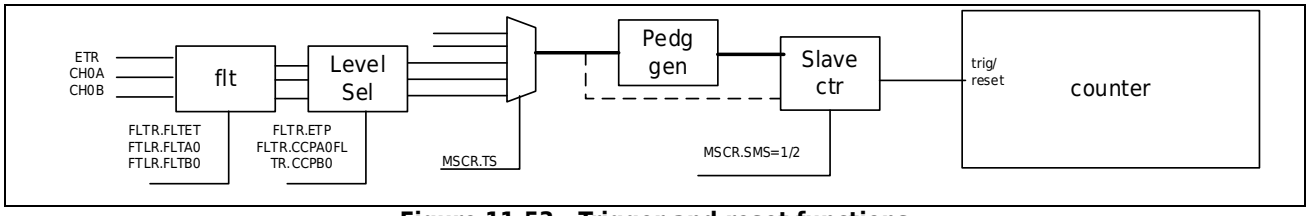


Figure 11-53 Trigger and reset functions

11.3.8 Quadrature code counting function

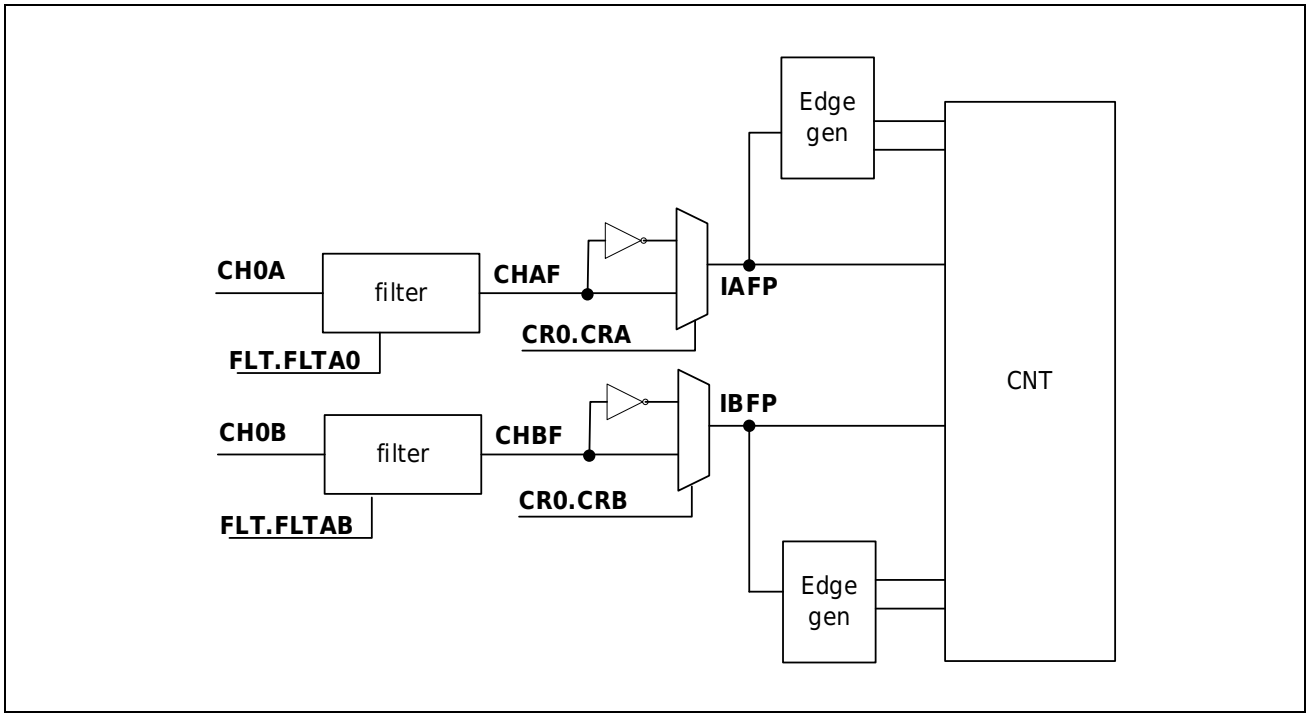
MSCR.SMS=4/5/6, corresponding to the mode 1/2/3 of the orthogonal coding mode. At this time, the counter encodes and counts according to the phase relationship of IAFP and IBFP. IAFP, IBFP is the port input CH0A, CH0B filter phase selection signal.

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.

			IAFP		IBFP	
	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

CHAF/CHBF is the signal filtered by port CH0A/CH0B, and IAFP/IBFP is the signal after port filter phase selection.



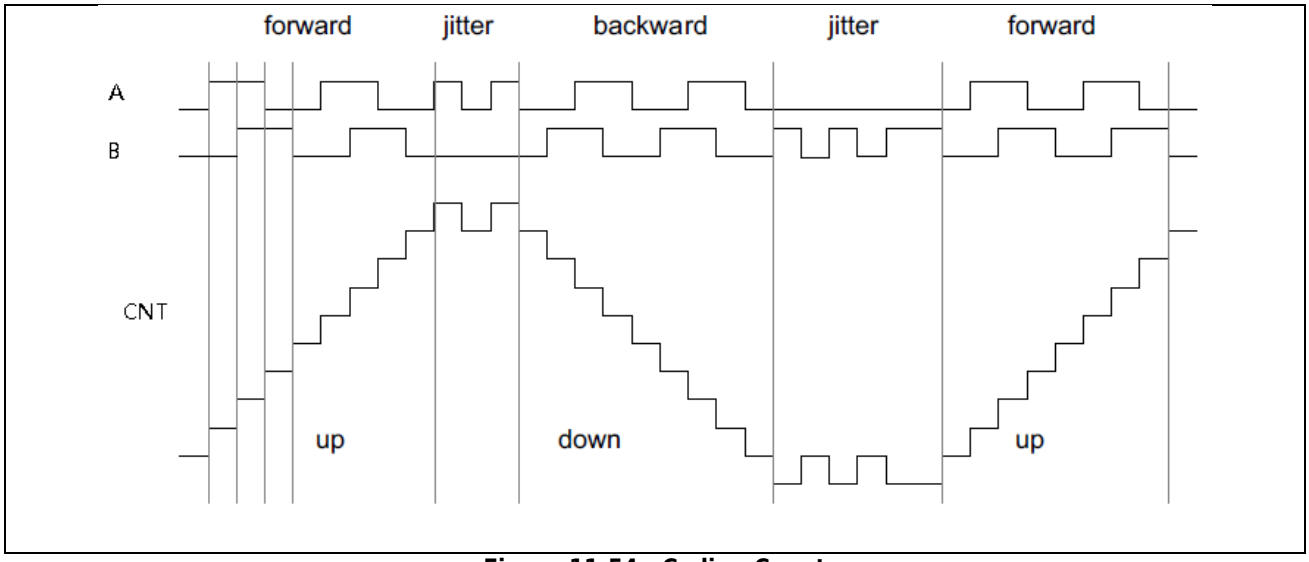


Figure 11-54 Coding Count

11.3.9 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.

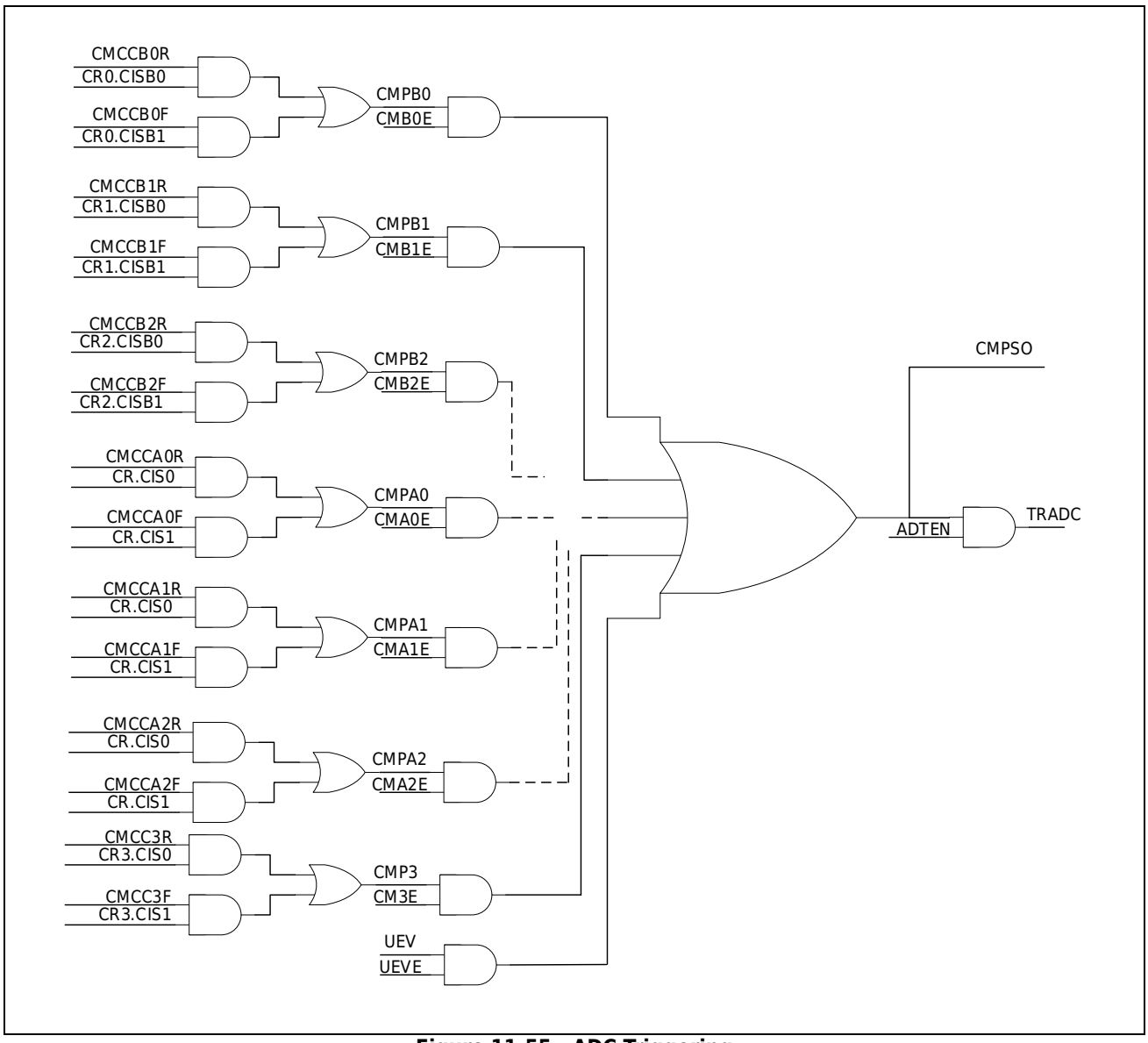


Figure 11-55 ADC Triggering

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

11.3.10 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.

11.3.11 Timer interconnection

The TRGO output signal can be connected to the ITR signal of other timers. The connection relationship is as follows:

	ITR0	ITR1	ITR2	ITR3
ATIM3	NC	NC	GTIM_OV	NC

11.3.12 CH0B Capture Input Interconnect

The CH0B input of ATIM3 can be directly input from the port, or can be connected to other modules or ports through the selection of the port function register GPIO_CR4.

When ATIM3_CHy=0x0, the capture input is the port input selected by GPIOx_AFRL. When ATIM3_CHy=0x1~0x3, it is connected to the input or output of other modules.

ATIM3	**CH0B**
000	GPIOx_AFRL
001	LPUART0_RXD
010	LPUART1_RXD
011	LVD_OUT

11.4Register description

Table 11-1 Timer register list

Timer	base address	Description
ATIM3	0x40005800	ATIM3 base address
ATIM0	0x40000C00	ATIM0 base address

(x=0,3)

Register	Offset address	Description
ATIMx_ARR	0x000	Timer reload register / cycle
ATIMx_CNT	0x004	Timer 16 -bit mode count register
ATIMx_M1CR	0x00C	Timer mode 1 control register (described in different modes)
ATIMx_M23CR	0x00C	Timer mode 23 control register (described in different modes)
ATIMx_IFR	0x010	Timer interrupt flag
ATIMx_ICLR	0x014	Timer interrupt clear register
ATIMx_MSCR	0x018	Master-slave mode control
ATIMx_FLTR	0x01C	filter control
ATIMx_ADTR	0x020	ADC trigger control
ATIMx_CRCH0	0x024	Compare Unit 0 Control Register
ATIM3_CRCH1	0x028	Compare Unit 1 Control Register
ATIM3_CRCH2	0x02C	Compare Unit 2 Control Register
ATIMx_DTR	0x030	dead zone register
ATIMx_RCR	0x034	Repeat Count Register
ATIMx_ARRDM	0x038	Timer reload register / period map address
ATIMx_CCR0A	0x03C	Compare 0A Register
ATIMx_CCR0B	0x040	Compare 0B register
ATIM3_CCR1A	0x044	Compare 1A Register
ATIM3_CCR1B	0x048	Compare 1B register
ATIM3_CCR2A	0x04C	Compare 2A Register
ATIM3_CCR2B	0x050	Compare 2B register
ATIM3_CCR3	0x054	Compare 3 registers
ATIM3_CRCH3	0x058	Compare Unit 3 Control Register

11.4.1 Mode 0 Register Description

11.4.1.1 16 -bit Mode Reload Register (ATIMx_ARR)

Offset address: 0x000

Reset Value: 0x0000 0000

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

Bit	Symbol	Description
31:16	Reserved	reserved bit
15:0	ARR	16 -bit reload timer reload value register

11.4.1.2 16 -bit Mode Count Register (ATIMx_CNT)

Offset address: 0x004

Reset Value: 0x0000 0000

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

Bit	Symbol	Description
31:16	Reserved	reserved bit
15:0	CNT	16 -bit reload timer count value register

11.4.1.3 32 -bit Mode Count Register (ATIMx_CNT32)

Offset address: 0x008

Reset Value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNT32[31:16]															
RW															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT32[15:0]															
RW															

Bit	Symbol	Description
31:0	CNT32	32 -bit timer count value register Note: Only valid in mode 0 32 -bit timer free counting mode, other modes prohibit writing this register

11.4.1.4 Control Register (ATIMx_M0CR)

Offset address: 0x00C

Reset Value: 0x0060 0008

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

15	14	13	12	11	10	9	8
Resvered		Mode		Resvered	UI	GATEP	GATE
		RW			RW	RW	RW

7	6	5	4	3	2	1	0
Resvered	PRS			TOG	CT	MD	CEN
	RW			RW	RW	RW	RW

Bit	Symbol	Description
31:14	Reserved	reserved bit
13:12	MODE	Operating Mode **00 Timer mode 0;** 01 PWC mode 10 sawtooth wave modes; 11 triangle wave modes
11	Reserved	reserved bit
10	UI	Interrupt enable control, enable interrupt after writing 1
9	GATEP	reserved bit
8	GATE	Reserved bit, **keep this bit at 0**
7	Reserved	reserved bit
6:4	PRS	Internal clock frequency division selection 000: 1; 001: 2; 010: 4; 011: 8; 100: 16; 101: 32; 110: 64; 111: 256;
3	TOG	Toggle output enable in mode 0 1: Toggle output enable 0: Inversion output turns off CHA, CHB output is low level
2	CT	Counting clock selection 0: Internal count clock PCLK 1: External count clock ETR;
1	MD	Mode selection 32 timing /16 timing mode selection 0: 32 -bit free count 1: 16 -bit reload count
0	CEN	Timer enable 0: timer stops; 1: Timer enable

11.4.1.5 Interrupt Flag Register (ATIMx_IFR)

Offset address: 0x010

Reset Value: 0x0000 0000

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

Bit	Symbol	Description
31:1	Reserved	reserved bit
0	UI	overflow interrupt

11.4.1.6 Interrupt Flag Clear Register (ATIMx_ICLR)

Offset address: 0x014

Reset Value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															UI
															R1W0

Bit	Symbol	Description
31:1	Reserved	reserved bit
0	UI	Overflow interrupt clear, write 0 to clear

11.4.1.7 Dead Time Register (ATIMx_DTR)

Offset address: 0x030

Reset Value: 0x0000 0000

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

Bit	Symbol	Description
31:13	Reserved	reserved bit
12	MO	Toggle output enable 0: Toggle output to input state 1: flip port to output state
11:0	Reserved	reserved bit

11.4.2 Mode 1 Register Description

11.4.2.1 16 -bit Mode Count Register (ATIMx_CNT)

Offset address: 0x004

Reset Value: 0x0000 0000

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

Bit	Symbol	Description
31:16	Reserved	reserved bit
15:0	CNT	Register count value

11.4.2.2 Control Register (ATIMx_M1CR)

Offset address: 0x00C

Reset Value: 0x0060 0008

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

15	14	13	12	11	10	9	8
Resvered	Oneshot	Mode		Resvered	UI	Edg2nd	Edg1st
	RW	RW			RW	RW	RW

7	6	5	4	3	2	1	0
Resvered	PRS			Resvered	CT	Resvered	CEN
	RW				RW		RW

Bit	Symbol	Description
31:15	Reserved	reserved bit
14	Oneshot	Single trigger mode selection 1: After completing a pulse measurement, it will end automatically, and you need to re-enable CTEN for another measurement. 0: Cyclic measurement
13:12	MODE	Operating Mode 00 Timer mode 0; **01 PWC mode** 10 sawtooth wave modes; 11 triangle wave modes
11	Reserved	reserved bit
10	UI	Interrupt enable control, enable interrupt after writing 1 Counting to 0xFFFF will overflow and generate an overflow flag
9	Edg2nd	Pulse width measurement end edge selection
8	Edg1st	Pulse width measurement start edge selection edg2nd Edg1st: 00 Measurement: Rising - upperiod; 01 Low level width; 10 High level width; 11 Falling edge - falling edge cycle
7	Reserved	reserved bit
6:4	PRS	Internal clock frequency division selection 000: 1; 001: 2; 010: 4; 011: 8; 100: 16; 101: 32; 110: 64; 111: 256;
3	Reserved	reserved bit
2	CT	Counting clock selection 0: Internal count clock TCLK 1: External count clock ETR;
1	Reserved	reserved bit
0	CEN	Pulse Width Measurement Enable 0: Prohibited; 1: Enable

11.4.2.3 Interrupt Flag Register (ATIMx_IFR)

Offset address: 0x010

Reset Value: 0x0000 0000

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

Bit	Symbol	Description
31:3	Reserved	reserved bit
2	PWC	Pulse Width Measurement Interrupt Flag
1	Reserved	Keep
0	UI	overflow interrupt flag

11.4.2.4 Interrupt Flag Clear Register (ATIMx_ICLR)

Offset address: 0x014

Reset Value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													CA0	Res.	UI
													R1W0		R1W0

Bit	Symbol	Description
31:3	Reserved	reserved bit
2	CA0	Pulse width measurement interrupt flag clear, write 0 to clear
1	Reserved	Keep
0	UI	Overflow interrupt clear, write 0 to clear

11.4.2.5 Master-Slave Mode Control Register (ATIMx_MSCR)

Offset address: 0x018

Reset Value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Resverved			IB0S	IA0S	Resvered			TS			Resvered				
			RW	RW				RW							

Bit	Symbol	Description
31:13	Reserved	reserved bit
12	IB0S	CH0B input selection 0： CH0B; 1: internal trigger TS selection signal;
11	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
10:8	Reserved	reserved bit
7:5	TS	trigger selection 000: the signal ETFP 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;
4:0	Reserved	reserved bit

11.4.2.6 Output Control Filtering (ATIMx_FLTR)

Offset address: 0x01C

Reset value: 0x00000000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Resvered									FLTB0			Res	FLTA0		
									RW				RW		

Bit	Symbol	Description
31	ETP	ETR input phase selection 0: same phase; 1: reverse input;
30:28	FLTET	ETR filter control filter settings 0xx: Filter invalid 100: pclk 3 continuous effective; 101: pclk/4 3 continuous effective 110: pclk/16 3 continuous effective; 111: pclk/64 3 continuous effective
27:7	Resvered	reserved bit
6:4	FLTB0	CHB input filter control; filter settings 0xx: Filter invalid 100: pclk 3 continuous effective; 101: pclk/4 3 continuous effective 110: pclk/16 3 continuous effective; 111: pclk/64 3 continuous effective
3	Resvered	reserved bit
2:0	FLTA0	CHA input filter control filter settings 0xx: Filter invalid 100: pclk 3 continuous effective; 101: pclk/4 3 continuous effective 110: pclk/16 3 continuous effective; 111: pclk/64 3 continuous effective

11.4.2.7 Control Register (ATIMx_CR0)

Offset address: 0x024;

Reset value: 0x0000 3000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							CIEA	Reserved		CSB	CSA	Reserved			
							RW			RW	RW				

Bit	Symbol	Description
31:9	Resvered	reserved bit
8	CIEA	CIEA pulse width measurement complete interrupt enable 0: Prohibited 1: Enable
7:6	Resvered	reserved bit
5	CSB	The B channel captures/compares the function selection, and the CSB needs to be set to 1 when the pulse width measurement uses channel B filtering 0: Comparison mode 1: Capture mode
4	CSA	The A channel captures/compares the function selection, and the CSA needs to be set to 1 when the pulse width measurement uses channel A filtering 0: Comparison mode 1: Capture mode
3:0	Resvered	reserved bit

11.4.2.8 Compare Capture Register (ATIMx_CCR0A)

Offset address: 0x03C

Reset Value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR0A															
R															

Bit	Symbol	Description
31:16	Reserved	reserved bit
15:0	CCR0A	Pulse Width Measurement Results

11.4.3 Mode 2.3 Register Description

11.4.3.1 16 -bit Mode Reload Register (ATIMx_ARR)

Offset address: 0x000

Reset Value: 0x0000 0000

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

Bit	Symbol	Description
31:16	Reserved	reserved bit
15:0	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

11.4.3.2 16 -bit Mode Count Register (ATIMx_CNT)

Offset address: 0x004

Reset Value: 0x0000 0000

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

Bit	Symbol	Description
31:16	Reserved	reserved bit
15:0	CNT	Reload Timer Count Value Register

Note: When using PWM to compare the output, the initial CNT value needs to be less than the value of ARR.

11.4.3.3 Control Register (ATIMx_M23CR)

Offset address: 0x00C

Reset Value: 0x0060 0008

31	30	29	28	27	26	25	24
Resvered		UND	OV	DIR	BG	UG	TG
		RW	RW	RW/RO	W1	W1	W1
23	22	21	20	19	18	17	16
OCC	CIS		BI	TI	TD	URS	OCCS
RW	RW		RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8
CSG	Oneshot	Mode		UD	UI	CFG	CRG
RW	RW	RW		RW	RW	RW	RW
7	6	5	4	3	2	1	0
BUFP	PRS			Pwm2s	CT	Comp	CTEN
RW	RW			RW	RW	RW	RW

Bit	Symbol	Description
31:30	Reserved	reserved bit
29	UND	Underflow interrupt enable
28	OV	Overflow interrupt enable
27	DIR	Counting direction, can only be written in sawtooth wave mode. Read-only in other modes, invalid for writing 0: count up 1: count down Note: DIR is automatically cleared to 0 when switching from other modes to center-aligned mode. DIR is automatically cleared in reset mode by software event update and external trigger in slave mode.
26	BG	Software brake, automatic reset Write 1 to generate software brake; Writing 0 is invalid
25	UG	Software update, automatic reset Write 1 to generate software update; Writing 0 is invalid Initialize the counter and update the cache register to the corresponding register (cache enable), the pre-divider counter will also be cleared.
24	TG	Triggered by software, automatically cleared, it needs to be triggered under the trigger mode SMS=2 and mode=2/3. Write 1 to generate software trigger; Writing 0 is invalid
23	OCC	OCREF clear enable 1: OCREF_CLR signal can clear OCREF output 0: OCREF output is not affected by OCREF_CLR
22:21	CIS	Center-aligned A compare interrupt mode (B compare interrupt is controlled separately in CRCHx register CISB) 00: no interrupt, 01: rising edge interrupt, 10: Falling edge interrupt, 11: Both the upper and lower edges are interrupted
20	BI	Brake Interrupt Enable 1: interrupt enable 0: interrupt disabled
19	TI	trigger interrupt enable 1: interrupt enable 0: interrupt disabled
18	TD	reserved bit
17	URS	update source 0: overflow/underflow/software update UG/slave mode reset; 1: overflow / underflow

16	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
15	CSG	Keep
14	Oneshot	Single trigger mode selection 1: Timer stops after an event update occurs. 0: loop count
13:12	MODE	Operating Mode 00 Timer mode 0; 01 PWC mode **10 sawtooth wave modes; 11 triangle wave modes**
11	UD	reserved bit
10	UI	UIE update interrupt enable 1: enable update interrupt 0: disable update interrupt
9	CFG	reserved bit
8	CRG	reserved bit
7	BUFP	reload cache enable 1: The period cache is enabled, and the period value will not be affected until the next event is updated after writing. 0: The period cache is invalid, and the period value will be affected immediately after writing
6:4	PRS	Internal clock frequency division selection 000：1； 001：2； 010：4； 011：8； 100：16； 101：32； 110：64； 111：256；
3	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
2	CT	Counting clock selection 0： Internal count clock TCLK 1： External count clock ETR;
1	Comp	PWM complementary output mode selection 0: independent PWM output 1: Complementary PWM output
0	CTEN	Timer enable 0: Prohibited; 1: Enable It can be enabled by external trigger, and this bit is automatically cleared at the end of oneshot mode

11.4.3.4 Interrupt Flag Register (ATIMx_IFR)

Offset address: 0x010

Reset Value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													CA3	Und	OV
													RO	RO	RO

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TI	BI	CB2E	CB1E	CB0E	CA2E	CA1E	CA0E	CB2	CB1	CB0	CA2	CA1	CA0	Res.	UI
RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO		RO

Bit	Symbol	Description
31:19	Reserved	reserved bit
18	CA3	Channel CH3 compare match flag
17	Und	underflow interrupt flag
16	OV	overflow interrupt flag
15	TI	trigger interrupt flag
14	BI	Brake Interrupt Flag
13	CB2E	Channel CH2B capture data loss flag 0: no data loss; 1: data loss
12	CB1E	Channel CH1B Capture Data Loss Flag 0: no data loss; 1: data loss
11	CB0E	Channel CH0B Capture Data Loss Flag 0: no data loss; 1: data loss
10	CA2E	Channel CH2A Capture Data Loss Flag 0: no data loss; 1: data loss
9	CA1E	Channel CH1A Capture Data Loss Flag 0: no data loss; 1: data loss
8	CA0E	Channel CH0A Capture Data Loss Flag 0: no data loss; 1: data loss
7	CB2	Channel CH2B capture / compare match flag 0: Does not occur 1: Occurs
6	CB1	Channel CH1B capture / compare match flag 0: Does not occur 1: Occurs
5	CB0	Channel CH0B capture / compare match flag 0: Does not occur 1: Occurs
4	CA2	Channel CH2A capture / compare match flag 0: Does not occur 1: Occurs
3	CA1	Channel CH1A capture / compare match flag 0: Does not occur 1: Occurs
2	CA0	Channel CH0A capture / compare match flag 0: Does not occur 1: Occurs
1	Reserved	reserved bit
0	UI	Event Update Interrupt Flag 0: Does not occur 1: Occurs

11.4.3.5 Interrupt Flag Clear Register (ATIMx_ICLR)

Offset address: 0x014

Reset Value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													CA3	Und	OV
													R1W0	R1W0	R1W0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TI	BI	CB2E	CB1E	CB0E	CA2E	CA1E	CA0E	CB2	CB1	CB0	CA2	CA1	CA0	Res.	UI
R1W0	R1W0	R1W0	R1W0	R1W0	R1W0	R1W0	R1W0	R1W0	R1W0	R1W0	R1W0	R1W0	R1W0		R1W0

Bit	Symbol	Description
31:19	Reserved	reserved bit
18	CA3	Channel CH3 comparison match flag is cleared, write 0 to clear
17	Und	Underflow interrupt flag cleared
16	OV	overflow interrupt flag clear
15	TI	Trigger interrupt flag to clear, write 0 to clear
14	BI	Brake interrupt flag is cleared, write 0 to clear
13	CB2E	Channel CH2B capture data loss flag clear, write 0 to clear
12	CB1E	Channel CH1B capture data loss flag clear, write 0 to clear
11	CB0E	Channel CH0B capture data loss flag clear, write 0 to clear
10	CA2E	Channel CH2A capture data loss flag clear, write 0 to clear
9	CA1E	Channel CH1A capture data loss flag clear, write 0 to clear
8	CA0E	Channel CH0A capture data loss flag clear, write 0 to clear
7	CB2	Channel CH2B capture / compare match flag clear, write 0 to clear
6	CB1	Channel CH1B capture / compare match flag clear, write 0 to clear
5	CB0	Channel CH0B capture / compare match flag clear, write 0 to clear
4	CA2	Channel CH2A capture / compare match flag clear, write 0 to clear
3	CA1	Channel CH1A capture / compare match flag clear, write 0 to clear
2	CA0	Channel CH0A capture / compare match flag clear, write 0 to clear
1	Reserved	Keep
0	UI	Event update interrupt clear, write 0 to clear

11.4.3.6 Master-Slave Mode Control Register (ATIMx_MSCR)

Offset address: 0x018

Reset Value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Resverved			IB0S	IA0S	SMS			TS			MSM	CCDS	MMS		
			RW	RW	RW			RW			RW	RW	RW		

Bit	Symbol	Description
31:13	Reserved	reserved bit
12	IB0S	CH0B input selection 0：CH0B; 1: internal trigger TS selection signal;
11	IA0S	IA0 input selection 0： CH0A 1： CH0A CH1A CH2A XOR Note: After setting to 1, any port change of the port will cause the input to change
10:8	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
7:5	TS	trigger selection 000: the signal ETFP 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;
4	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
3	CCDS	DMA comparison trigger selection in comparison mode; 0: compare match triggers DMA; 1： Compare match does not trigger DMA, event update instead of compare match triggers DMA
2:0	MMS	Master mode output select for internal interconnection 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 OCREF0A 101: Timer comparison parameter output OCREF1A 110: Timer comparison parameter output OCREF2A 111: Timer comparison parameter output OCREF0B

11.4.3.7 Output Control / Input Filtering (ATIMx_FLTR)

Offset address: 0x01C

Reset value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ETP	FLTET			BKP	FLTBK			CCPB2	OCMB2			CCPA2	OCMA2		
-	-			-	-			-	FLTB2			-	FLTA2		
RW	RW			RW	RW			RW	RW			RW	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCPB1	OCMB1			CCPA1	OCMA1			CCPB0	OCMB0			CCPA0	OCMA0		
-	FLTB1			-	FLTA1			CCPB0	FLTB0			CCPA0	FLTA0		
RW	RW			RW	RW			RW	RW			RW	RW		

Bit	Symbol	Description
31	ETP	ETR input phase selection 0: same phase; 1: reverse input;
30:28	FLTET	ETR filter control filter settings 0xx: Filter invalid 100: pclk 3 continuous effective; 101: pclk/4 3 continuous effective 110: pclk/16 3 continuous effective; 111: pclk/64 3 continuous effective
27	BKP	Brake BK input phase selection 0: same phase; 1: reverse input;
26:24	FLTBK	Brake input filter control filter settings 0xx: Filter invalid 100: pclk 3 continuous effective; 101: pclk/4 3 continuous effective 110: pclk/16 3 continuous effective; 111: pclk/64 3 continuous effective
23	CCPB2	Comparison function: CH2B channel comparison output phase control 0: normal output; 1: Reverse output
22:20	OCMB2 FLTB2	Comparison function: CH2B channel comparison control; refer to OCMB0 Capture function: CH2B input channel filter setting, refer to FLTBK
19	CCPA2	Comparison function: CH2A channel comparison output phase control 0: normal output; 1: Reverse output
18:16	OCMA2 FLTA2	Comparison function: CH2A channel comparison control; refer to OCMB0 Capture function: CH2A input channel filter setting, refer to FLTBK
15	CCPB1	Comparison function: CH1B channel comparison output phase control 0: normal output; 1: Reverse output
14:12	OCMB1 FLTB1	Comparison function: CH1B channel comparison control; refer to OCMB0 Capture function: CH1B input channel filter setting, refer to FLTBK
11	CCPA1	Comparison function: CH1A channel comparison output phase control 0: normal output; 1: Reverse output
10:8	OCMA1 FLTA1	Comparison function: CH1A channel comparison control; refer to OCMB0 Capture function: CH1A input channel filter setting, refer to FLTBK
7	CCPB0	Compare function: output compare mode CCPBx compare output CHBx port polarity control 0: normal output; 1: Reverse output Encode Count and Slave Mode Gating Function: Input Phase Control CCPB0 slave mode gating, reset, external trigger, external clock using CH0B port input polarity control 0: normal input; 1: reverse input
6:4	OCMB0 FLTB0	Comparison function: CH0B 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 101: Output a high level for one count period when comparing matches 110: PWM Mode 1 Single point comparison: When counting up, CNT<CCRxy outputs high, and when counting down, CNT>CCRxy outputs low level Two-point comparison: 1) Counting CCRxA<CNT≤CCRxB output on the sawtooth wave is low level 2) Counting under sawtooth wave CCRxA<CNT≤CCRxB output is high level 3) Triangular wave up counting CNT<CCRxA output high, down counting 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≤CNT<CCRxB output on the sawtooth wave is high level 2) Counting under sawtooth wave CCRxA≤CNT<CCRxB output is low level 3) Triangular wave up count CNT<CCRxA output low, down count CNT>CCRxB is high level Capture function: CH0B input channel filter setting, refer to FLTBK
3	CCPA0	Compare function: CCPAx compare output CHAx port polarity control 0: normal output; 1: Reverse output Encode Count and Slave Mode Gating Function: Input Phase Control CCPA0 slave mode gating, reset, external trigger, external clock using CH0A port input polarity control 0: normal input; 1: reverse input
2:0	OCMA0 FLTA0	Comparison function: A channel comparison control; refer to OCMB0 Capture function: CH0A input channel filter setting, refer to FLTBK

11.4.3.8 ADC Trigger Control Register (ATIMx_ADTR)

Offset address: 0x020

Reset Value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ADT	CMB2	CMB1	CMB0	CMA2	CMA1	CMA0	UEV
								RW	RW	RW	RW	RW	RW	RW	RW

Bit	Symbol	Description
31:8	Resvered	reserved bit
7	ADT	Enable ADC trigger global control 1: Enable 0: Prohibited
6	CMB2	Channel 2B compare match trigger ADC enable 1: Enable 0: Prohibited
5	CMB1	Channel 1B compare match trigger ADC enable 1: Enable 0: Prohibited
4	CMB0	Channel 0B compare match trigger ADC enable 1: Enable 0: Prohibited
3	CMA2	Channel 2A compare match trigger ADC enable 1: Enable 0: Prohibited
2	CMA1	Channel 1A compare match trigger ADC enable 1: Enable 0: Prohibited
1	CMA0	Channel 0A compare match trigger ADC enable 1: Enable 0: Prohibited
0	UEV	Event update triggers ADC enable 1: Enable 0: Prohibited

11.4.3.9 Channel 0 Control Register (ATIMx_CRCH0)

Offset address: 0x024

Reset Value: 0x0000 3000

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

Bit	Symbol	Description
31:16	Resvered	reserved bit
15	CCGB	Capture comparison B 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.
14	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.
13:12	CISB	B channel compare match setting 00 no match; 01 rise match; 10 fall match; 11 double match
11	CDEB	B capture compare trigger DMA enable 0: disable 1: enable
10	CDEA	A capture compare trigger DMA enable 0: Prohibited 1: Enable
9	CIEB	B capture compare trigger interrupt enable 0: Prohibited 1: Enable
8	CIEA	A capture compare trigger interrupt enable 0: Prohibited 1: Enable
7	BUFEB	Compare Function: B Compare Cache Enable Control 0: Prohibited 1: Enable
6	BUFEA	Compare Function: A Compare Cache Enable Control 0: Prohibited 1: Enable
5	CSB	B channel capture / compare function selection 0: compare mode 1: capture mode
4	CSA	A channel capture / compare function selection 0: compare mode 1: capture mode
3	CFB	B channel falling edge capture enable 0: Prohibited 1: Enable
2	CRB	B channel rising edge capture enable 0: Prohibited 1: Enable
3:2	BKSB	B channel comparison function output brake level control 00: High impedance output 01: no effect on output 10: forced output low level 11: forced output high level;
1	CFA	A channel falling edge capture enable 0: Prohibited 1: Enable
0	CRA	A channel rising edge capture enable 0: Prohibited 1: Enable
1:0	BKSA	A channel comparison function output brake level control 00: High impedance output 01: no effect on output 10: forced output low level

11: forced output high level;

11.4.3.10 Channel 1/2 Control Register (TIM3_CRCH1/2)

Offset address: TIM3_CRCH1: 0x028;

TIM3_CRCH2：0x02C

Reset Value: 0x0000 3000

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

Bit	Symbol	Description
31:16	Resvered	reserved bit
15	CCGB	Capture comparison B 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.
14	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.
13:12	CISB	B channel compare match setting 00 no match; 01 rise match; 10 fall match; 11 double match
11	CDEB	reserved bit
10	CDEA	reserved bit
9	CIEB	B capture compare trigger interrupt enable 0: Prohibited 1: Enable
8	CIEA	A capture compare trigger interrupt enable 0: Prohibited 1: Enable
7	BUFEB	Compare Function: B Compare Cache Enable Control 0: Prohibited 1: Enable
6	BUFEA	Compare Function: A Compare Cache Enable Control 0: Prohibited 1: Enable
5	CSB	B channel capture / compare function selection 0: compare mode 1: capture mode
4	CSA	A channel capture / compare function selection 0: compare mode 1: capture mode
3	CFB	B channel falling edge capture enable 0: Prohibited 1: Enable
2	CRB	B channel rising edge capture enable 0: Prohibited 1: Enable
3:2	BKSB	B channel comparison function output brake level control 00: High impedance output 01: keep the previous output l 10: forced output low level 11: forced output high level;
1	CFA	A channel falling edge capture enable 0: Prohibited 1: Enable
0	CRA	A channel rising edge capture enable 0: Prohibited 1: Enable

1:0	BKSA	A channel comparison function output brake level control 00: High impedance output 01: keep the previous output l 10: forced output low level 11: forced output high level;

11.4.3.11 Dead Time Register (ATIMx_DTR)

Offset address: 0x030

Reset Value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	VCE	Safe	MO	AO	BK	DT	Bksel	DTR							
RW	RW	RW	RW	RW	RW	RW	RW	RW							

Bit	Symbol	Description
31:15	Reserved	reserved bit
14	VCE	reserved bit
13	Safe	Safety brake enable (osc fail,brown down,lockup) 0: Prohibited 1: Enable
12	MO	PWM output enable 0: Prohibited 1: Enable
11	AO	PWM output is automatically enabled 0: Prohibited 1: Enable
10	BK	brake enable 0: Prohibited 1: Enable
9	DT	Dead zone control enable 0: Prohibited 1: Enable
8	bksel	reserved bit
7:0	DTR	Dead Time Register 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

11.4.3.12 Repeat Period Set Value Register (ATIMx_RCR)

Offset address: 0x034

Reset Value: 0x0000 0000

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

Bit	Symbol	Description
31:10	Reserved	reserved bit
9	UD	Overflow masking under repeat count control 0: undercount overflow enable 1: undercount overflow mask
8	OV	Overflow masking on repeat count control 0: undercount overflow enable 1: undercount overflow mask
7:0	RCR	recurrence count value Set RCR+1 period and an overflow controlled by [9:8] to generate an event update. When the counter overflows or underflows, the internal RCR_CNT is decremented by 1. When the count reaches zero, RCR_CNT reloads the value of RCR and generates an event to update the UEV signal

11.4.3.13 Channel 0 Compare Capture Register (ATIMx_CCR0A/B)

Offset address: ATIMx_CCR0A: 0x03C;

ATIMx_CCR0B: 0x040

Reset Value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR0y															
RW															

Bit	Symbol	Description
31:16	Reserved	reserved bit
15:0	CCR0y	Compare the capture register, the comparison has a cache function (y=A,B)

11.4.3.14 Channel 1/2 compare capture register (TIM3_CCR1/2 A/B)

Offset address: TIM3_CCR1A: 0x044

TIM3_CCR1B: 0x048

TIM3_CCR2A: 0x04C

TIM3_CCR2B: 0x050

Reset Value: 0x0000 0000

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

Bit	Symbol	Description
31:16	Reserved	reserved bit
15:0	CCRxy	Compare the capture register, the comparison has a cache function (x=1,2; y=A,B)

11.4.3.15 Channel 3 compare capture register (TIM3_CCR3)

Offset address: TIM3_CCR3: 0x054

Reset Value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3															
RW															

Bit	Symbol	Description
31:16	Reserved	reserved bit
15:0	CCR3	Compare capture register, compare with cache function

11.4.3.16 Channel 3 Control Register (ATIM3_CRCH3)

Offset address: 0x058

Reset Value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										C3	CIS		CD	CI	BUF
										RW	RW		RW	RW	RW

Bit	Symbol	Description
31:6	Resvered	reserved bit
5	C3	Channel Compare Enable
4:3	CIS	Channel Compare Match Settings 00 no match; 01 rise match; 10 fall match; 11 double match
2	CD	Compare Trigger DMA Enable 0: disable 1: enable
1	CI	Compare Trigger Interrupt Enable 0: Prohibited 1: Enable
0	BUF	Compare Function: Compare Cache Enable Control 0: Prohibited 1: Enable

12 Clock Trim Module (CTRIM)

12.1 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.

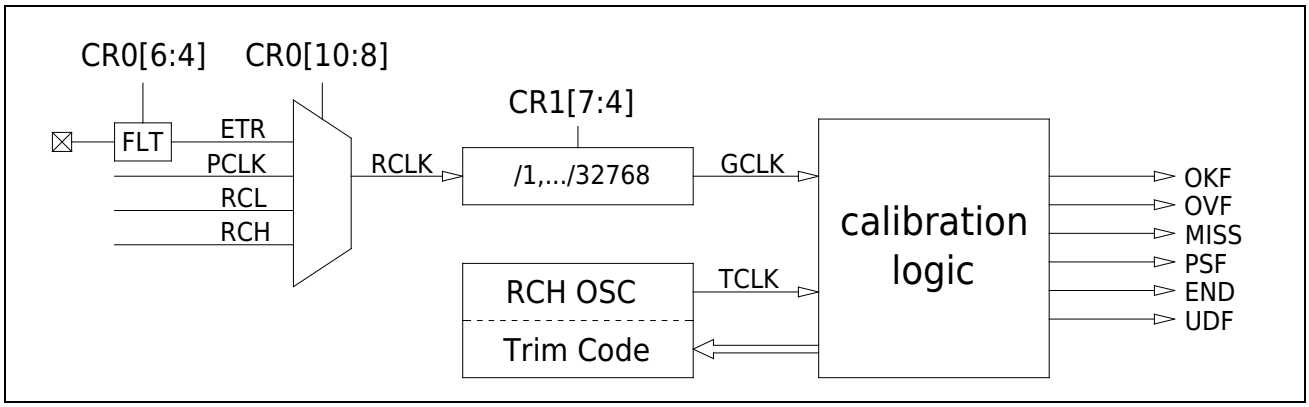
12.2 Main characteristics

- 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, cycle 30us ~ 65536s
- Timing mode: 4 clock sources to choose from
- Timing Mode: Toggle Output

12.3 Functional description

12.3.1 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. When calibrating RCH, it is necessary to provide a precise low-frequency clock signal to the module; the precise low-frequency clock signal can be the on-chip PCLK, or it can be input through the external ETR pin. Note that the RCLK frequency should be less than 1/2 of the RCH frequency. The working block diagram of RCH clock calibration is shown below.



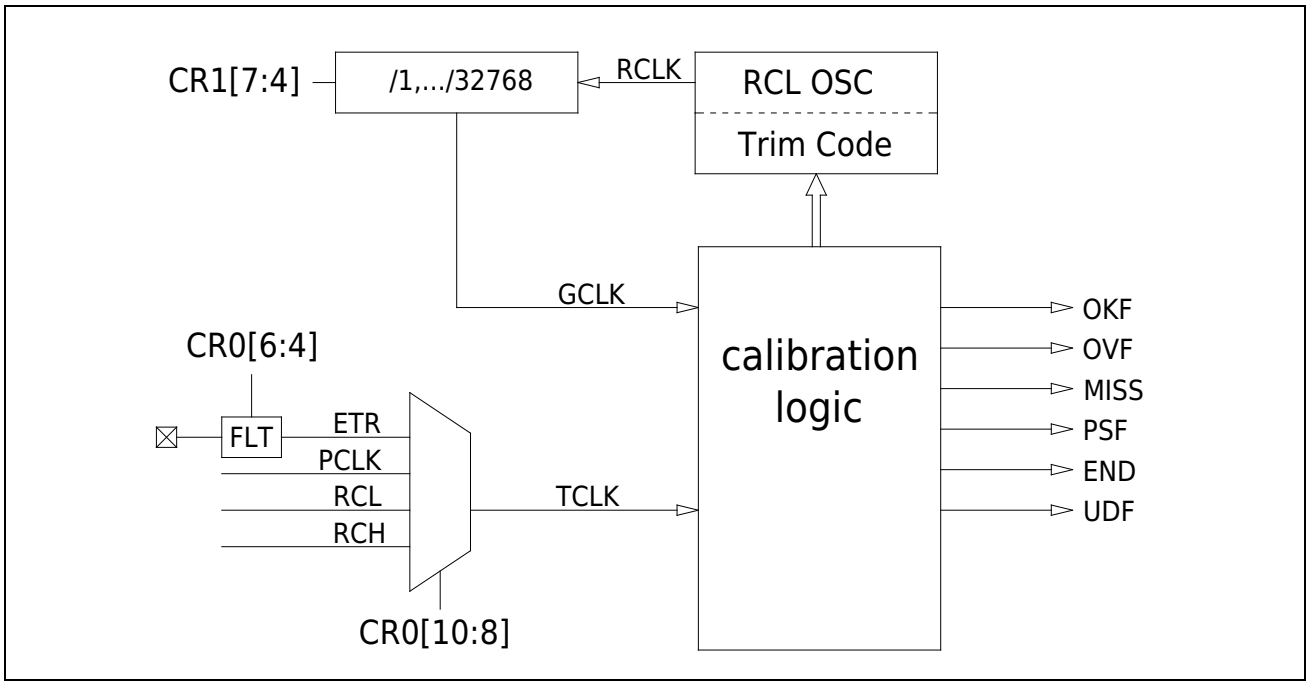
When the automatic calibration starts to work, the error counter counts down the TCLK signal (RCH oscillation frequency) from CTRIM.ARR in each GCLK period, and starts counting up after counting to 0. If the count value of the error counter in the GCLK cycle is less than the value of the CTRIM_FLIM register, set CTRIM_ISR.OKF to indicate that the RCH output frequency has reached the

expected; if the count value of the error counter in the GCLK cycle is greater than the value of the CTRIM_FLIM register, automatically adjust the RCH Trim Code and proceed to the next round of counting. The automatic adjustment rule is: if the counting direction is down counting, add CTRIM_CR0.STEP to the Trim Code of RCH; if the counting mode is up counting, then decrease CTRIM_CR0.STEP to the RCH Trim Code. The calibration logic will automatically reduce the value of CTRIM_CR0.STEP so that the frequency of the RCH output can approach the target value successively.

When the error counter counts down to zero, a UDF event is generated; when the error counter counts up to CTRIM_ARR, it stops counting and generates a MISS event. Whenever the rising edge of GCLK arrives, the error counter starts counting again.

12.3.2 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. When calibrating RCL, it is necessary to provide a precise high-frequency clock signal to the module; the precise high-frequency clock signal can be the on-chip PCLK, or it can be input through the external ETR pin. The working block diagram of RCL clock calibration is shown below.



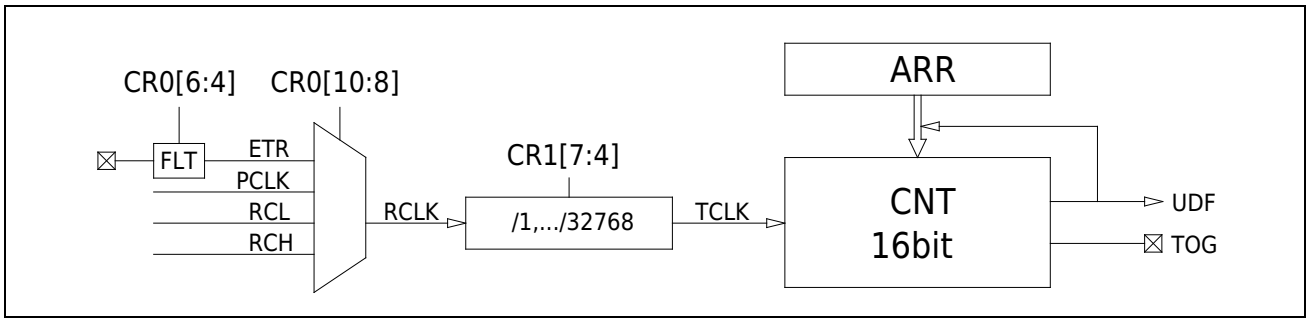
When the automatic calibration starts to work, the error counter counts down the TCLK clock signal from CTRIM.ARR in each cycle of GCLK (the RCL oscillation frequency after frequency division), and starts counting up after counting to 0. If the count value of the error counter in the GCLK cycle is less than the value of the CTRIM_FLIM register, set CTRIM_ISR.OKF to indicate that the RCL output frequency has reached the expected value; if the count value of the error counter in the GCLK cycle is greater than the value of the CTRIM_FLIM register, automatically adjust the Trim of the RCL Code

and proceed to the next round of counting. The automatic adjustment rule is: if the counting direction is down counting, then decrease CTRIM_CR0.STEP for the Trim Code of RCL; if the counting mode is up counting, then increase CTRIM_CR0.STEP for the RCL Trim Code. The calibration logic will automatically reduce the value of CTRIM_CR0.STEP so that the frequency of the RCL output can approach the target value step by step.

When the error counter counts down to zero, a UDF event is generated; when the error counter counts up to CTRIM_ARR, it stops counting and generates a MISS event. Whenever the rising edge of GCLK arrives, the error counter starts counting again.

12.3.3 Low Power Timer Mode

When configuring CTRIM.CR1.MD as 0x02, the module works ingeneral low power timermode. In this mode, CTIRM has the timing and interrupt functions of a general-purpose timer. If the clock of the timer is RCL, it can work normally in DeepSleep, supports automatic wake-up, and the cycle is 30us ~ 65536s. The operation block diagram of the general-purpose low-power timer is shown below.



The count value of the timer is counted down from ARR. When the count reaches 0, an underflow will occur. If the interrupt is enabled, an underflow interrupt will be generated. The counting cycle is ARR+1.

The TOG output level flips once every time the timer overflows; the user can output the TOG signal to the pin through the auxiliary function of GPIO to drive the external circuit.

12.4Software operation process

12.4.1 RCH Clock Automatic Calibration Example

Step1. Reset initialization of the CTRIM module is performed using a peripheral reset.

Step2. Configure the function of PA03 as CTRIM.ETR, and provide it with a 2KHz square wave.

Step3. Set CTRIM_CR1.MD to 0 to select RCH clock calibration mode.

Step4. Set CTRIM_CR0.SRC to 0, and the external ETR pin input signal is used as RCLK.

Step5. Set CTRIM_CR1.PRS to 1, divide RCLK by 2, and GCLK frequency is 1KHz.

Step6. Configure CTRIM_ARR as 47999, and the target count value in one GCLK count cycle is 48000.

Note: The ETR pin input signal is a 2KHz square wave, and the period of GCLK is 1ms. The theoretical count value of the RCH output frequency by the error counter within 1ms should be 48000.

Step7. Set CTRIM_CR0.FLIM to 240 and select RCH correction accuracy to 0.5%.

Note: 48000 * 0.005 = 240.

Step8. Set CTRIM_CR0.STEP to 4 and select the initial calibration step to be 16.

Step9. Set CTRIM_CR1.AUTO to 1 to enable automatic calibration.

Step10. Set CTRIM_CR1.EN to 1 to enable clock calibration.

Step11. Wait for CTRIM_ISR.END to be 1, automatic calibration is completed.

12.4.2 RCL Clock Automatic Calibration Example

Step1. Reset initialization of the CTRIM module is performed using a peripheral reset.

Step2. Configure the function of PA03 as CTRIM.ETR, and provide it with a 1MHz square wave.

Step3. Set CTRIM_CR1.MD to 1 to select RCL clock calibration mode.

Step4. Set CTRIM_CR0.SRC to 0, and the external ETR pin input signal is TCLK.

Step5. Set CTRIM_CR1.PRS to 0x0A, divide RCLK (RCL output frequency) by 1024, and the target frequency of GCLK is 32Hz.

Step6. Configure CTRIM_ARR as 31249, and the target count value in one GCLK count cycle is 31250.

Note: The target period of GCLK is 31.25ms, and the theoretical count value of the error counter to TCLK (1MHz) within this time should be 31250.

Step7. Set CTRIM_CR0.FLIM to 156 and select RCL correction accuracy to 0.5%.

Note: 31250 * 0.005 = 156.

Step8. Set CTRIM_CR0.STEP to 4 and select the initial calibration step to be 16.

Step9. Set CTRIM_CR1.AUTO to 1 to enable automatic calibration.

Step10. Set CTRIM_CR1.EN to 1 to enable clock calibration.

Step11. Wait for CTRIM_ISR.END to be 1, automatic calibration is completed.

12.5Register description

Table 12-1 CTRIM register

Base Address 0x40005000

Register	Offset address	Description
CTRIM_ARR	0x00	auto reload value
CTRIM_CNT	0x04	counter value
CTRIM_CR0	0x0C	control register 0
CTRIM_CR1	0x10	control register 1
CTRIM_IER	0x14	Interrupt Enable Control Register
CTRIM_ISR	0x18	Interrupt and Status Registers
CTRIM_ICR	0x1C	Interrupt Flag Clear Register
CTRIM_FCAP	0x20	The value of the counter at the end of the GCLK cycle
CTRIM_TVAL	0x24	TRIM value register, TRIM process software needs to read twice to be the same valid
CTRIM_FLIM	0x28	error tolerance

12.5.1 Autoload Register (CTRIM_ARR)

Offset address 0x00

Reset value 0x0000FFFF

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

Bit	Marking	Functional description
31:16	Reserved	Keep
15:0	ARR	Error counter reload value

12.5.2 Error Counter Register (CTRIM_CNT)

Offset address 0x04

Reset value 0x00000000

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

Bit	Marking	Functional description
31:16	Reserved	Keep
15:0	CNT	Error counter count value

12.5.3 Control Register 0 (CTRIM_CR0)

Offset address 0x0c

Reset value 0x0000

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

Bit	Marking	Functional description
31:11	Reserved	Keep
10:8	SRC	Clock source selection 000: ETR 001: reserved 010: reserved 011: PCLK 100: RCL 101: RCH 110: reserved 111: reserved
7	Reserved	internal reservation
6:4	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
3	Reserved	Keep
2:0	STEP	TRIM initial step configuration 000: 0x10 100: 0x10 001: 0x02 101: 0x20 010: 0x04 110: 0x40 011: 0x08 111: 0x80

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

12.5.4 Control Register 1 (CTRIM_CR1)

Offset address 0x010

Reset value 0x00000000

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

Bit	Marking	Functional description
31:9	Reserved	Keep
8	OST	Calibration Mode Configuration 0: real-time calibration mode, the TRIM process continues 1: One-shot calibration mode, end when STEP becomes 1
7:4	PRS	Reference Clock Prescaler Configuration 0000: Reserved, not configurable 0001: 2 frequency division 1001: 512 frequency division 0010: divide by 4 1010: divide by 1024 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
3	AUTO	RCH/RCL auto- TRIM enable configuration 1: Enable 0: Prohibited
2:1	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
0	EN	Timer Enable Control 1: Enable 0: Prohibited

12.5.5 Interrupt Enable Register (CTRIM_IER)

Offset address 0x14

Reset value 0x00000000

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

Bit	Marking	Functional description
31:5	Reserved	Keep
4	OV	TrimCode overflow interrupt enable configuration 1: Enable 0: Prohibited
3	MISS	Count failure interrupt enable configuration 1: Enable 0: Prohibited
2	PS	Error counter complete interrupt enable configuration 1: Enable 0: Prohibited
1	END	Auto Trim End Interrupt Enable Configuration 1: Enable 0: Prohibited
0	UD	Error Counter Underflow Interrupt Enable Configuration 1: Enable 0: Prohibited

12.5.6 Interrupt and Status Register (CTRIM_ISR)

Offset address 0x18

Reset value0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									OK	DIR	OV	MISS	PS	END	UD
									RO	RO	RO	RO	RO	RO	RO

Bit	Marking	Functional description
31:7	Reserved	Keep
6	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
5	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
4	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
3	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
2	PS	Error Counter Completion Flag 1: The error counter has completed one count and stopped 0: The error counter is counting
1	END	Auto Trim End Flag 1: Auto Trim is complete 0: Auto Trim is in progress
0	UD	Error Counter Underflow Flag 1: The error counter has counted to 0 0: The error counter has not counted to 0

12.5.7 Interrupt Status Clear Register (CTRIM_ICR)

Offset address 0x1C

Reset value 0x0000001F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										OK	OV	MISS	PS	END	UD
										R1W0	R1W0	R1W0	R1W0	R1W0	R1W0

Bit	Marking	Functional description
31:6	Reserved	Keep
5	OK	Write 0 to clear the calibration success OK flag
4	OV	Write 0 to clear the TrimCode overflow flag
3	MISS	Write 0 to clear count failure flag
2	PS	Write 0 to clear the error counter completion flag
1	END	Write 0 to clear auto- trim end flag
0	UD	Write 0 to clear the error counter underflow flag

12.5.8 Error Result Register (CTRIM_FCAP)

Offset address 0x20

Reset value 0x0000 FFFF

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

Bit	Marking	Functional description
31:16	Reserved	Keep
15:0	FCAP	Frequency error capture value

12.5.9 TrimCode register (CTRIM_TVAL)

Offset address 0x24

Reset value0x0000 0100

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

Bit	Marking	Functional description
31:16	Reserved	Keep
15:0	TVAL	Automatic TRIM result, the end of automatic TRIM needs to write the value of TVAL into the corresponding TRIM register

12.5.10 Error Upper Limit Register (CTRIM_FLIM)

Offset address 0x28

Reset value0x0000 0010

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

Bit	Marking	Functional description
31:8	Reserved	Keep
7:0	FLIM	error tolerance

13 Independent Watchdog (IWDT)

13.1 Overview

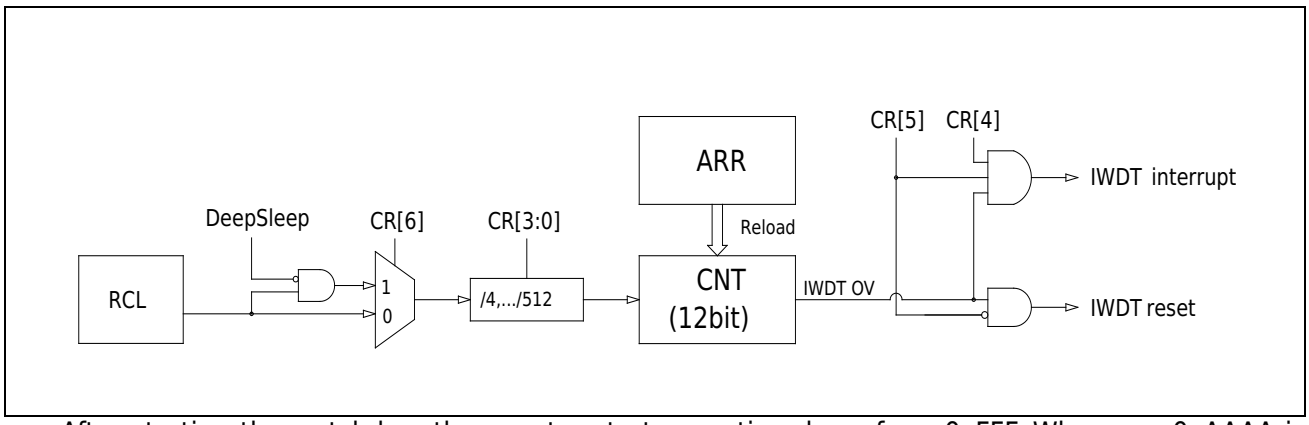
The device features an embedded stand-alone watchdog peripheral for high security and flexibility of use. This independent watchdog detects and resolves failures caused by software errors, triggering an interrupt or reset when its count reaches a programmed timeout value. The watchdog is driven by the internal low-speed clock RCL, even if the main clock fails, it can still work normally.

13.2 Main characteristics

- Overflow period 132us - 64s @ RCL32.8KHz
- Trigger interrupt or reset on overflow
- A running watchdog can be stopped by a specific write sequence
- Automatically pause counting in DeepSleep mode

13.3 Functional description

13.3.1 Functional block diagram



After starting the watchdog, the counter starts counting down from 0xFFF. Whenever 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.

13.3.2 Window watchdog mode

When the configured IWDT_WINR register value is less than IWDT_ARR, IWDG 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 changing the IWDT_WINR register value, the hardware will automatically

reload the watchdog and 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 value of the IWDT_WINR register and the count value is decremented to 0x000.

13.3.3 Independent Watchdog Mode

When the value of IWDT_WINR register is greater than or equal to IWDT_ARR, IWDG 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.

13.3.4 Watchdog overflow handling

The overflow signal of the watchdog can trigger an interrupt or a reset. When 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.

13.3.5 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.

13.3.6 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.

13.3.7 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.

13.4programming example

13.4.1 Configure IWDT as window watchdog

Step1. Write 0xCCCC to the IWDT_KR register to enable IWDG.

Step2. Write 0x5555 to the IWDT_KR register to release the write protection of the register.

Step3. Configure IWDT_CR, select the counting clock frequency of the watchdog, the action after overflow, and the DeepSleep feature.

Step4. Configure IWDT_ARR to select the overflow period of the watchdog.

Step5. Wait for IWDT_SR.ARRF to be 0.

Step6. Write the required value to IWDT_WINR to configure the window size.

Step7. Wait for IWDT_SR.WINRF to become 0, and the auto-refresh counter value is IWDT_ARR.

Step8. Before the count value is less than or equal to the value of the IWDT_WINR register and the count value is decremented to 0, send to IWDT_KR

Step9. Write 0xAAAA to the register to reload the IWDT.

13.4.2 Configure IWDT as an independent watchdog

Step1. Write 0xCCCC to the IWDT_KR register to enable IWDG.

Step2. Write 0x5555 to the IWDT_KR register to release the write protection of the register.

Step3. Configure IWDT_CR, select the counting clock frequency of the watchdog, the action after overflow, and the DeepSleep feature.

Step4. Configure IWDT_ARR to select the overflow period of the watchdog.

Step5. Wait for IWDT_SR to become 0x10, and the frequency division coefficient and reload value are all written.

Step6. The refresh counter value is the value of IWDG_ARR (IWDG_KR = 0x0000 AAAA)

Step7. Before the count value is decremented to 0, write 0xAAAA to the IWDT_KR register to reload the IWDT.

13.5Register description

Table 13-1 IWDT Register List

IWDT Base Address: 0x40000F80

Register	Offset address	Description
IWDT_KR	0x00	IWDT key-value register
IWDT_CR	0x04	IWDT control register
IWDT_ARR	0x08	IWDT reload register
IWDT_SR	0x0C	IWDT status register
IWDT_WINR	0x10	IWDT window register

13.5.1 Key-value register (IWDT_KR)

Offset address: 0x000

Reset Value: 0x0000 007F

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

Bit	Symbol	Description
31:16	Reserved	reserved bit
15:0	KR	Write 0xCCCC: start IWDT counter Write 0xAAAA: reload IWDT count value Write 0x1234 - 0x5678 in sequence: stop watchdog Write 0x5555: Release the write protection of IWDT_PR, IWDT_ARR, IWDT_WINR, IWDT_CR Write non- 0x5555: enable write protection of IWDT_PR, IWDT_ARR, IWDT_WINR, IWDT_CR

13.5.2 Control Register (IWDT_CR)

Offset address: 0x004

Reset Value: 0x0000 0000

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

Bit	Symbol	Description
31:7	Reserved	reserved bit
6	Pause	WDT pause control in Deepsleep mode 0: IWDT continues to run in DeepSleep mode 1: IWDT automatically pauses in DeepSleep mode
5	Action	Action configuration after IWDT overflow 0: A reset is generated after watchdog time-out 1: Generate interrupt after watchdog overflow
4	IE	IWDT interrupt enable control 0: Disable watchdog interrupt 1: Enable watchdog interrupt
3:0	PRS	Configure the frequency division ratio of watchdog count clock and RCL oscillator 111: 512 frequency division 110: 256 frequency division 101: 128 frequency division 100: 64 frequency division 011: 32 frequency division 010: 16 frequency division 001: 8 frequency division 000: 4 frequency division

Note:

- This register can only be modified after writing 0x5555 to the IWDT_KR register.
- IWDT_CR.PRS can only be modified when IWDT_SR.PRSF is 0.

13.5.3 Reload Register (IWDT_ARR)

Offset address: 0x008

Reset Value: 0x0000 0FFF

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

Bit	Symbol	Description
31:12	Reserved	reserved bit
11:0	ARR	IWDT reload value

Note:

- This register can only be modified after writing 0x5555 to the IWDT_KR register.
- This register can only be modified when IWDT_SR.ARRF is 0.

13.5.4 Window Register (IWDT_WINR)

Offset address: 0x010

Reset Value: 0x0000 0FFF

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

Bit	Symbol	Description
31:12	Reserved	reserved bit
11:0	WINR	window comparator comparison value

Note:

- This register can only be modified after writing 0x5555 to the IWDT_KR register.
- This register can only be modified when IWDT_SR.WINRF is 0.

13.5.5 Status Register (IWDT_SR)

Offset address: 0x00C

Reset Value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											RUN	OV	WINRF	ARRF	PRSF
											RO	R/W0	RO	RO	RO

Bit	Symbol	Description
31:5	Res.	reserved bit
4	RUN	WDT running flag 1: WDT is running 0: WDT is not running
3	OV	WDT overflow flag 1: WDT overflow, write 0 to clear 0: WDT has not overflowed
2	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.
1	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.
0	PRSF	CR.PRS register update flag 1: CR.PRS register is being updated 0: CR.PRS register update completed Note: The CR.PRS register can be updated only when the PRSF bit is 0.

14 Window Watchdog (WWDT)

14.1 Overview

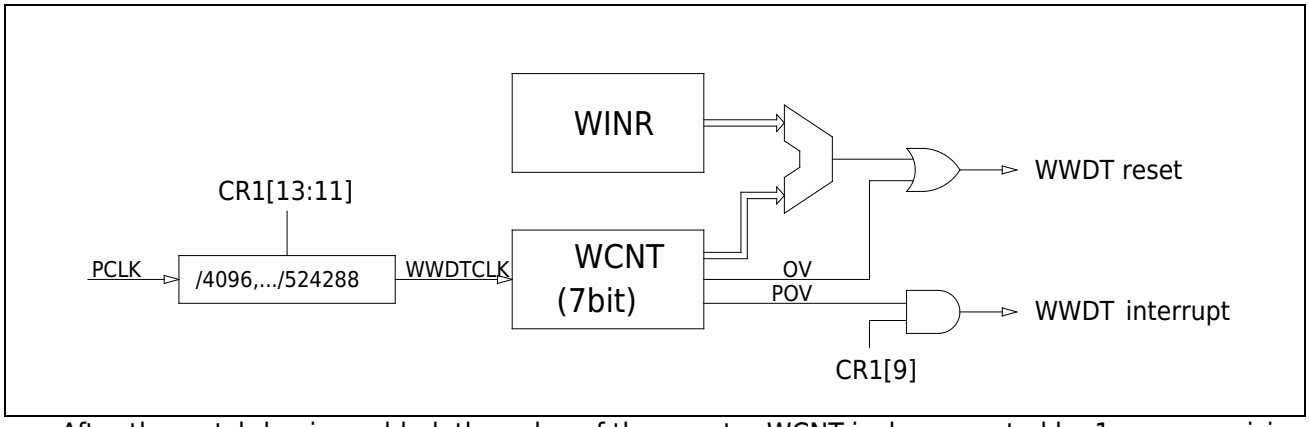
The window watchdog (WWDT) is often used to monitor software failures triggered by unexpected errors. An interrupt is generated when the counter value is decremented to 0x40, and a reset is generated when the counter value is decremented to 0x3F; if the user program updates the counter value when the counter value is greater than the window value, a reset is generated. Therefore, the user program can only update the value of the WWDT counter within a limited window (the counter value is less than the window value and greater than 0x3F).

14.2 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

14.3 Functional description

14.3.1 Functional block diagram



After the watchdog is enabled, the value of the counter WCNT is decremented by 1 on 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 WCNT is greater than the value of WINR when the user updates WCNT, the watchdog will also generate a reset signal.

14.3.2 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. It is recommended to enable the watchdog by writing 0xFF to WWDT_CR0 to ensure that the reset will not be triggered when the watchdog is enabled.

14.3.3 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:

$T_{WWDT} = T_{PCLK} \times 4096 \times 2^{CR1.PRS} \times (CR0.WCNT-0x3F)$

14.3.4 update down counter

Write data to WWDT_CR0.WCNT to update 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.

14.3.5 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.

14.4programming 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 as 2, WWDT count clock as 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, write 0x6F to WWDT_CR0.WCNT to reload the watchdog counter.

14.5Register description

Table 14-1 WWDT Register List

WWDT Base Address: 0x40006800

Register	Offset address	Description
WWDT_CR0	0x00	WWDT Control Register 0
WWDT_CR1	0x04	WWDT Control Register 1
WWDT_SR	0x08	WWDT status register

14.5.1 Control Register 0 (WWDT_CR0)

Offset address: 0x000

Reset Value: 0x0000 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								EN	WCNT						
								RW1	RW						

Bit	Symbol	Description
31:8	Reserved	reserved bit
7	EN	WWDT enable control, cannot be disabled after enabling 1: Enable WWDT 0: Disable WWDT
6:0	WCNT	Read: WWDT counter current count value Write: Update the current count value of the WWDT counter

14.5.2 Control Register 1 (WWDT_CR1)

Offset address: 0x004

Reset Value: 0x0000 007F

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		PRS			Res.	IE	Res.		WINR						
		RW				RW			RW						

Bit	Symbol	Description
31:14	Reserved	reserved bit
13:11	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
10	Res.	Reserved bit, read as 0
9	IE	WWDT pre-overflow interrupt enable control 1: Enable pre-overflow interrupt 0: Disable pre-overflow interrupt
8:7	Res.	reserved bit
6:0	WINR	Watchdog window value

14.5.3 Configuration Register (WWDT_SR)

Offset address: 0x008

Reset Value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															POV
															RW0

Bit	Symbol	Description
31:1	Res.	reserved bit
0	POV	WWDT pre-flow flag 1: WWDT has pre-flown, write 0 to clear this flag 0: WWDT pre-flow has not occurred

15 Low Power Synchronous Asynchronous Transceiver (LPUART)

15.1 Overview

The Low Power Synchronous Asynchronous Receiver (LPUART) is a UART that allows full-duplex UART communication in low power mode. Even when the microcontroller is in DeepSleep mode, the LPUART can normally receive UART data frames, and the receiving completion interrupt can wake up the microcontroller.

15.2 Main characteristics

- Configuration clock PCLK
- Transmission clock SCLK (SCLK can choose RCL and PCLK)
- Two character lengths: 8 bits, 9 bits
- Three checkout methods: no checkout, odd checkout, even checkout
- Three stop lengths: 1 bit, 2 bits, 1.5 bits
- Send and receive data in low power mode
- Two-wire full-duplex communication
- Single-wire half-duplex communication
- Hardware flow control (RTS, CTS)
- Multi-machine communication, automatic address recognition

15.3 Functional description

15.3.1 Functional block diagram

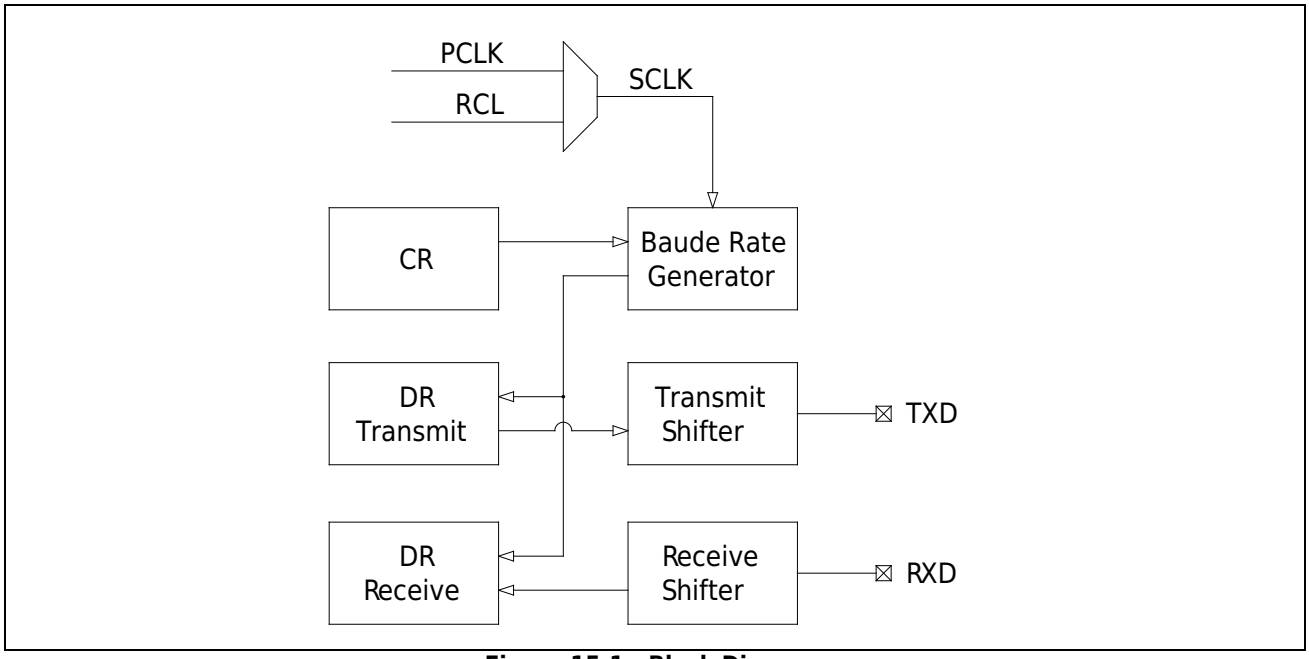


Figure 15-1 Block Diagram

15.3.2 Clock Description

The LPUART module has two clocks: configuration clock PCLK and transmission clock SCLK.

- configure clock

  The configuration clock (PCLK) is used for register configuration of the LPUART module by the system APB bus, and the configuration clock is fixed as the APB bus clock PCLK. When the system enters Deep Sleep (DeepSleep) mode, the PCLK clock will stop.

- Transmission clock

  The transmission clock (SCLK) is used for LPUART data transceiver logic work, and the internal low-speed clock (RCL) and PCLK clock can be selected. When the system enters Deep Sleep (DeepSleep) mode, if SCLK selects the internal low-speed clock (RCL), LPUART can still perform normal data transmission and reception without being affected by the system Deep Sleep (DeepSleep) mode.

15.3.3 Operating Mode

LPUART supports multiple working modes: synchronous half-duplex mode and asynchronous full-duplex mode. Through the combination of LPUARTx_CR.MODE and LPUARTx_CR.HDSEL, various working modes can be configured.

15.3.3.1 Comparison of four working modes

Configure LPUARTx_CR.MODE to select different transmission modes: Mode0~Mode3. The main function comparison of these four working modes is shown in the table below:

Table 15-1 Mode0/1/2/3 data structure

Operating Mode		Transmission bit width	data composition	Baud rate
Mode0	synchronous mode half duplex	8-bit	Data(8bit)	
Mode1	asynchronous mode full duplex	10-bit	Start(1bit) + Data(8bit) + Stop (1~2bit)	
Mode2	asynchronous mode full duplex	11-bit	Start (1bit) + Data(8bit) + B8(1bit) + Stop (1~2bit)	
Mode3	asynchronous mode full duplex	11-bit	Start (1bit) + Data(8bit) + B8(1bit) + Stop (1~2bit)	

Note:

- Mode0 can only be used as a host to send LPUART synchronous shift clock, and cannot be used as a slave to receive externally input LPUART synchronous shift clock;
- $f_{sclk}$is the clock frequency of SCLK;
- The definition of OVER is detailed in LPUARTx_CR;
- The definition of SCNT is detailed in LPUARTx_BRR;
- The B8 data bit is special and has different meanings in different applications, please refer to the following table:

Table 15-2 B8 Data Meaning

Application Scenario	LPUARTx_CR. ADRDET	LPUARTx_CR. B8CFG[1:0]	DR8 data meaning
Parity	-	01/10	When receiving, DR8 is the parity bit of the received 8-Bit data; When sending, DR8 is the parity bit of the sent 8-Bit data;
Multi-machine communication	1	-	DR8=1, it means that it is currently an address frame; DR8=0, which means the current data frame;
other	0	00/11	Received / sent DATA[8]

Note:

- When the multi-machine communication mode is turned on, the receiving data parity check is automatically closed; the sending data parity check is still controlled by B8CFG.

15.3.3.2 Mode0 data sending and receiving instructions

When setting LPUARTx_CR.RXEN to 0, LPUART works in sending state. Write the data to be sent into the LPUARTx_DR register, then the data will be output from RXD, and the synchronous shift clock will be output from TXD.

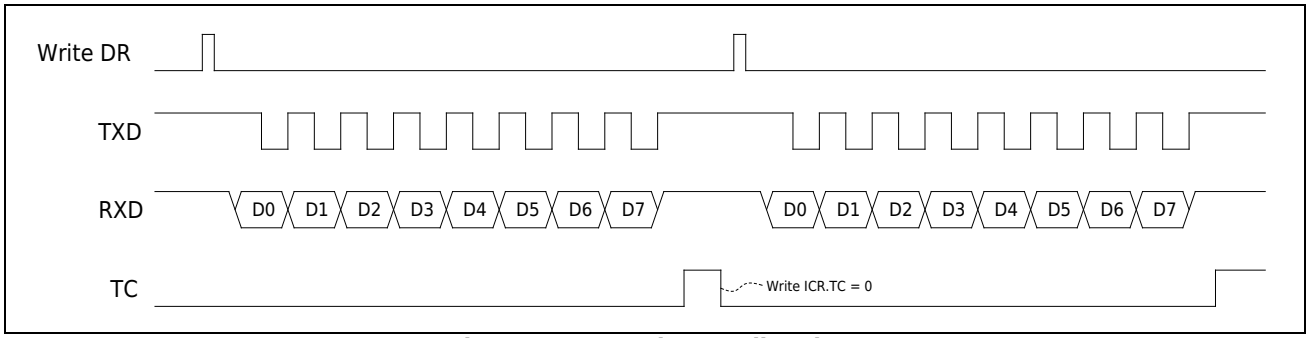


Figure 15-2 Mode0 sending data

When setting LPUARTx_CR.RXEN to 1, LPUART works in receiving state. Whenever the RC flag is 0, the TXD pin will output 8 synchronous shift clocks, and the data will be input from RXD. Clear the RC flag each time to receive the next byte.

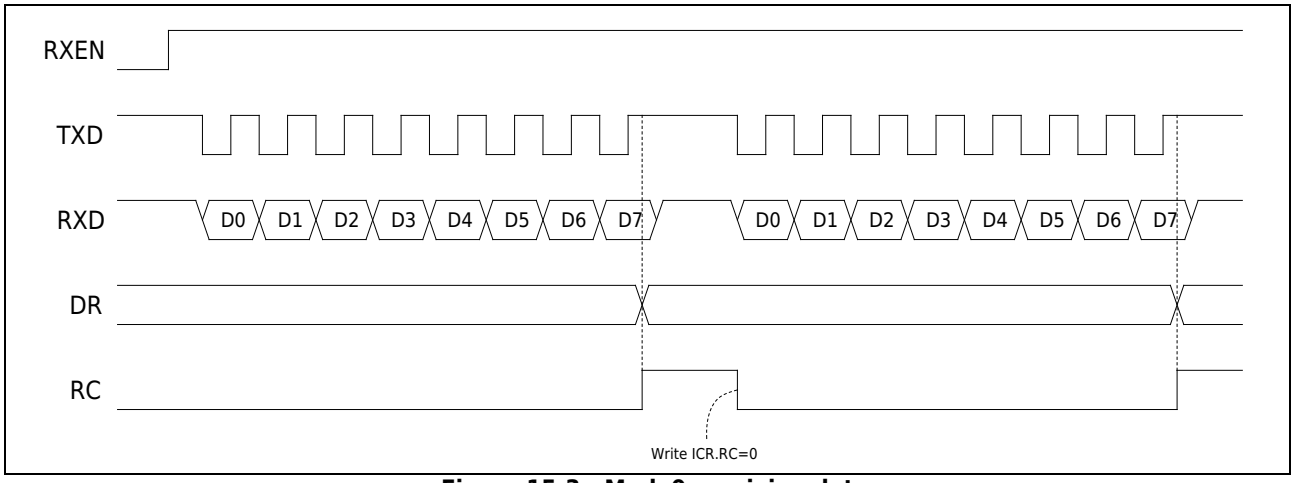


Figure 15-3 Mode0 receiving data

15.3.3.3 Mode1 data sending and receiving instructions

Whether LPUARTx_CR.RXEN is 0 or 1, data can be sent normally. When there is data in the LPUARTx_DR register, the hardware automatically loads the data into the transmit shift register and shifts it out from the TXD pin. The TXE flag is set to 1 by hardware whenever the data in the LPUARTx_DR register is loaded into the shift register. When the user finds that the TXE flag is 1, the user can write the next data to be sent to the LPUARTx_DR register. Whenever the TXD pin sends a frame of data, the TC flag will be set to 1 by hardware to indicate the current sending progress.

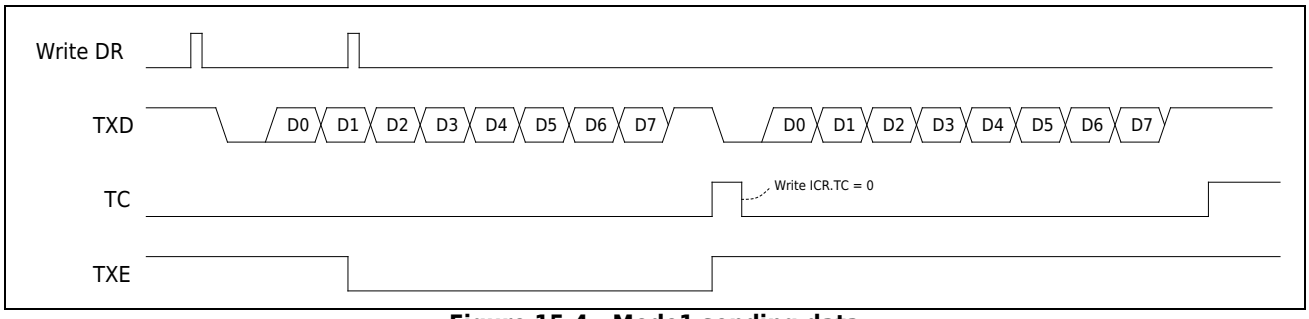


Figure 15-4 Mode1 sending data

When receiving data, you need to set LPUARTx_CR.RXEN to 1. Whenever a frame of data is received, the ISR.RC flag will be set to 1 by hardware, and the received data can be read from the LPUARTx_DR register.

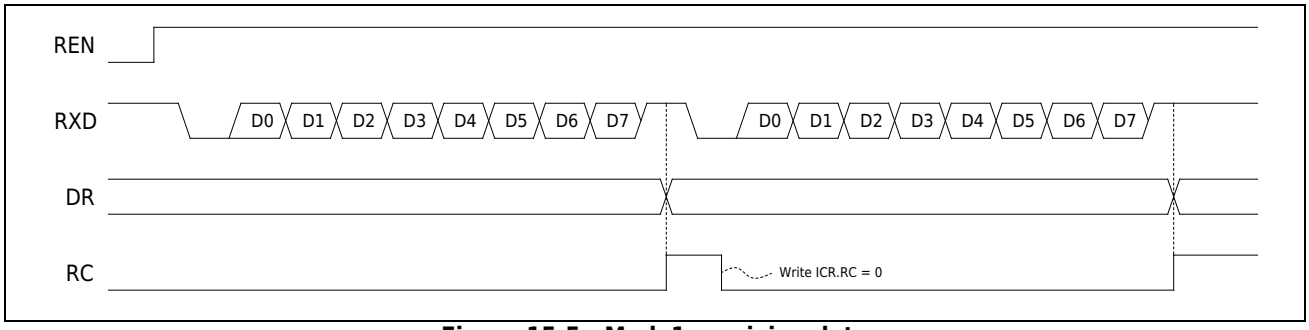


Figure 15-5 Mode1 receiving data

15.3.3.4 Mode2/Mode3 data sending and receiving instructions

Whether LPUARTx_CR.RXEN is 0 or 1, data can be sent normally. When there is data in the LPUARTx_DR register, the hardware automatically loads the data into the transmit shift register and shifts it out from the TXD pin. The TXE flag is set to 1 by hardware whenever the data in the LPUARTx_DR register is loaded into the shift register. When the user finds that the TXE flag is 1, the user can write the next data to be sent to the LPUARTx_DR register. Whenever the TXD pin sends a frame of data, the TC flag will be set to 1 by hardware to indicate the current sending progress.

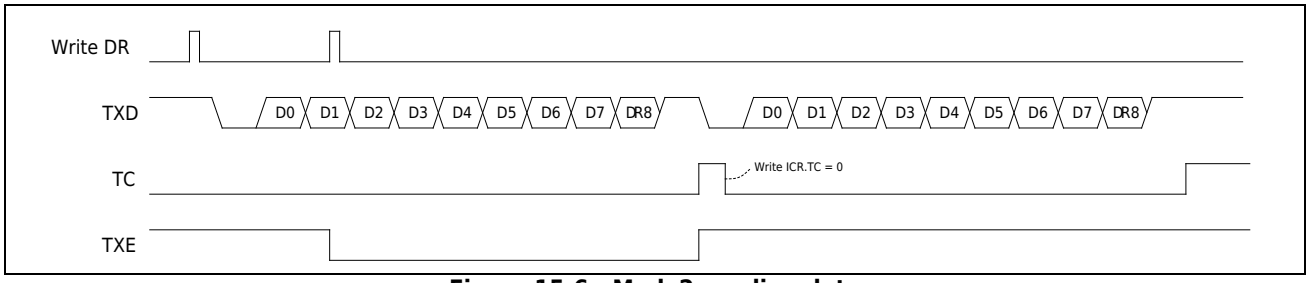


Figure 15-6 Mode2 sending data

When receiving data, you need to set LPUARTx_CR.RXEN to 1. Whenever a frame of data is received, the ISR.RC flag will be set to 1 by hardware, and the received data can be read from the LPUARTx_DR register.

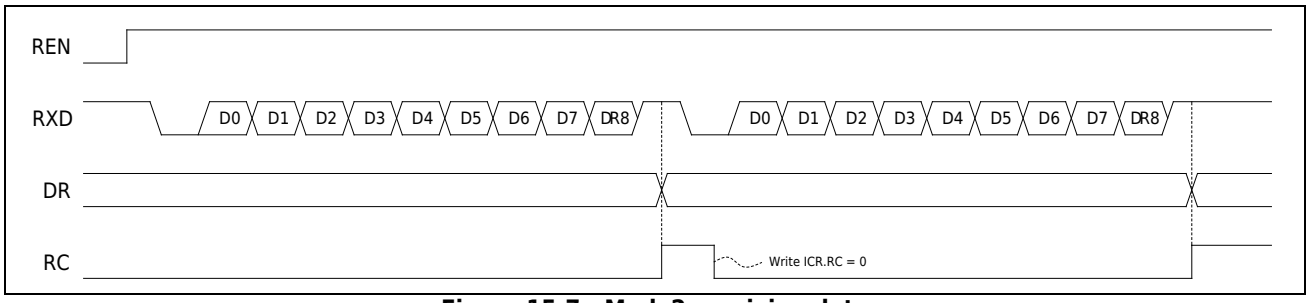


Figure 15-7 Mode2 receiving data

15.3.3.5 Single bus data transmission and reception instructions

When LPUARTx_CR.HDSEL is set to 1, LPUART only uses one TXD pin for data transmission and reception; users need to configure the TXD pin as an open-drain output and connect an external pull-up resistor.

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 LPUART_DR register, the TXD pin becomes an output state to send the data written by the user to the LPUART_DR 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.

15.3.4 Baud rate generation

The formulas for generating the baud rate of Mode0~Mode3 are different, as shown below for details.

Mode0 baud rate generation formula: $BaudRate=\frac{f_{SCLK}}{12}$

Mode1 baud rate generation formula: $BaudRate=\frac{f_{SCLK}}{OVER*SCNT}$

Mode2 baud rate generation formula: $BaudRate=\frac{f_{SCLK}}{OVER}$

Mode3 baud rate generation formula: $BaudRate=\frac{f_{SCLK}}{OVER*SCNT}$

Note:

- f_{SCLK}represents the frequency of the current SCLK;
- The definition of OVER is detailed in LPUARTx_CR;
- See LPUARTx_BRR for the definition of SCNT.

15.3.4.1 Transmission clock is an example of 4MHz baud rate setting (Mode1 / Mode3)

Table 15-3 SCL is 4MHz baud rate calculation table

Baud rate	SCLK = 4 MHz								
	OVER4			OVER8			OVER16		
	SCNT	actual baud rate	Error %	SCNT	actual baud rate	Error %	SCNT	actual baud rate	Error %
2400	417	2398.08	-0.08%	208	2403.85	0.16%	104	2403.85	0.16%
4800	208	4807.69	0.16%	104	4807.69	0.16%	52	4807.69	0.16%
9600	104	9615.38	0.16%	52	9615.38	0.16%	26	9615.38	0.16%
19200	52	19230.77	0.16%	26	19230.77	0.16%	13	19230.77	0.16%
38400	26	38461.54	0.16%	13	38461.54	0.16%	7	35714.29	-6.99%
57600	17	58823.53	2.12%	9	55555.56	-3.55%	4	62500.00	8.51%
76800	13	76923.08	0.16%	7	71428.57	-6.99%	3	83333.33	8.51%
115200	9	111111.11	-3.55%	4	125000.00	8.51%	2	125000.00	8.51%
128000	8	125000.00	-2.34%	4	125000.00	-2.34%	2	125000.00	-2.34%
256000	4	250000.00	-2.34%	2	250000.00	-2.34%	1	250000.00	-2.34%
1000000	1	1000000.00	0.00%	1	500000.00	-50.00%	0	/	/

15.3.4.2 Transmission clock is 8MHz baud rate setting example (Mode1 / Mode3)

Table 15-4 SCLK is 8MHz baud rate calculation table

Baud rate	SCLK = 8 MHz								
	OVER4			OVER8			OVER16		
	SCNT	actual baud rate	Error %	SCNT	actual baud rate	Error %	SCNT	actual baud rate	Error %
2400	833	2400.96	0.04%	417	2398.08	-0.08%	208	2403.85	0.16%
4800	417	4796.16	-0.08%	208	4807.69	0.16%	104	4807.69	0.16%
9600	208	9615.38	0.16%	104	9615.38	0.16%	52	9615.38	0.16%
19200	104	19230.77	0.16%	52	19230.77	0.16%	26	19230.77	0.16%
38400	52	38461.54	0.16%	26	38461.54	0.16%	13	38461.54	0.16%
57600	35	57142.86	-0.79%	17	58823.53	2.12%	9	55555.56	-3.55%
76800	26	76923.08	0.16%	13	76923.08	0.16%	7	71428.57	-6.99%
115200	17	117647.06	2.12%	9	111111.11	-3.55%	4	125000.00	8.51%
128000	16	125000.00	-2.34%	8	125000.00	-2.34%	4	125000.00	-2.34%
256000	8	250000.00	-2.34%	4	250000.00	-2.34%	2	250000.00	-2.34%
1000000	2	1000000.00	0.00%	1	1000000.00	0.00%	1	500000.00	-50.00%
2000000	1	2000000.00	0.00%	1	1000000.00	-50.00%	0	/	/

15.3.4.3 Transmission clock is 16MHz baud rate setting example (Mode1 / Mode3)

Table 15-5 SCLK is 16MHz baud rate calculation table

Baud rate	SCLK = 16 MHz								
	OVER4			OVER8			OVER16		
	SCNT	actual baud rate	Error %	SCNT	actual baud rate	Error %	SCNT	actual baud rate	Error %
2400	1667	2399.52	-0.02%	833	2400.96	0.04%	417	2398.08	-0.08%
4800	833	4801.92	0.04%	417	4796.16	-0.08%	208	4807.69	0.16%
9600	417	9592.33	-0.08%	208	9615.38	0.16%	104	9615.38	0.16%
19200	208	19230.77	0.16%	104	19230.77	0.16%	52	19230.77	0.16%
38400	104	38461.54	0.16%	52	38461.54	0.16%	26	38461.54	0.16%
57600	69	57971.01	0.64%	35	57142.86	-0.79%	17	58823.53	2.12%
76800	52	76923.08	0.16%	26	76923.08	0.16%	13	76923.08	0.16%
115200	35	114285.71	-0.79%	17	117647.06	2.12%	9	111111.11	-3.55%
128000	31	129032.26	0.81%	16	125000.00	-2.34%	8	125000.00	-2.34%
256000	16	250000.00	-2.34%	8	250000.00	-2.34%	4	250000.00	-2.34%
1000000	4	1000000.00	0.00%	2	1000000.00	0.00%	1	1000000.00	0.00%
2000000	2	2000000.00	0.00%	1	2000000.00	0.00%	1	1000000.00	-50.00%
4000000	1	4000000.00	0.00%	1	2000000.00	-50.00%	0	/	/

15.3.4.4 Transmission clock is 24MHz baud rate setting example (Mode1 / Mode3)

Table 15-6 SCLK is 24MHz baud rate calculation table

Baud rate	SCLK = 24 MHz								
	OVER4			OVER8			OVER16		
	SCNT	actual baud rate	Error %	SCNT	actual baud rate	Error %	SCNT	actual baud rate	Error %
2400	2500	2400.00	0.00%	1250	2400.00	0.00%	625	2400.00	0.00%
4800	1250	4800.00	0.00%	625	4800.00	0.00%	313	4792.33	-0.16%
9600	625	9600.00	0.00%	313	9584.66	-0.16%	156	9615.38	0.16%
19200	313	19169.33	-0.16%	156	19230.77	0.16%	78	19230.77	0.16%
38400	156	38461.54	0.16%	78	38461.54	0.16%	39	38461.54	0.16%
57600	104	57692.31	0.16%	52	57692.31	0.16%	26	57692.31	0.16%
76800	78	76923.08	0.16%	39	76923.08	0.16%	20	75000.00	-2.34%
115200	52	115384.62	0.16%	26	115384.62	0.16%	13	115384.62	0.16%
128000	47	127659.57	-0.27%	23	130434.78	1.90%	12	125000.00	-2.34%
256000	23	260869.57	1.90%	12	250000.00	-2.34%	6	250000.00	-2.34%
1000000	6	1000000.00	0.00%	3	1000000.00	0.00%	2	750000.00	-25.00%
2000000	3	2000000.00	0.00%	2	1500000.00	-25.00%	1	1500000.00	-25.00%
6000000	1	6000000.00	0.00%	1	3000000.00	-50.00%	0	/	/

15.3.4.5 Transmission clock is 32MHz baud rate setting example (Mode1 / Mode3)

Table 15-7 SCLK is 32MHz baud rate calculation table

Baud rate	SCLK = 32 MHz								
	OVER4			OVER8			OVER16		
	SCNT	actual baud rate	Error %	SCNT	actual baud rate	Error %	SCNT	actual baud rate	Error %
2400	3333	2400.24	0.01%	1667	2399.52	-0.02%	833	2400.96	0.04%
4800	1667	4799.04	-0.02%	833	4801.92	0.04%	417	4796.16	-0.08%
9600	833	9603.84	0.04%	417	9592.33	-0.08%	208	9615.38	0.16%
19200	417	19184.65	-0.08%	208	19230.77	0.16%	104	19230.77	0.16%
38400	208	38461.54	0.16%	104	38461.54	0.16%	52	38461.54	0.16%
57600	139	57553.96	-0.08%	69	57971.01	0.64%	35	57142.86	-0.79%
76800	104	76923.08	0.16%	52	76923.08	0.16%	26	76923.08	0.16%
115200	69	115942.03	0.64%	35	114285.71	-0.79%	17	117647.06	2.12%
128000	63	126984.13	-0.79%	31	129032.26	0.81%	16	125000.00	-2.34%
256000	31	258064.52	0.81%	16	250000.00	-2.34%	8	250000.00	-2.34%
1000000	8	1000000.00	0.00%	4	1000000.00	0.00%	2	1000000.00	0.00%
2000000	4	2000000.00	0.00%	2	2000000.00	0.00%	1	2000000.00	0.00%
4000000	2	4000000.00	0.00%	1	4000000.00	0.00%	1	2000000.00	-50.00%
8000000	1	8000000.00	0.00%	1	4000000.00	-50.00%	0	/	/

15.3.4.6 Transmission clock is 48MHz baud rate setting example (Mode1 / Mode3)

Table 15-8 SCLK is 48MHz baud rate calculation table

Baud rate	SCLK = 48 MHz								
	OVER4			OVER8			OVER16		
	SCNT	actual baud rate	Error %	SCNT	actual baud rate	Error %	SCNT	actual baud rate	Error %
2400	5000	2400.00	0.00%	2500	2400.00	0.00%	1250	2400.00	0.00%
4800	2500	4800.00	0.00%	1250	4800.00	0.00%	625	4800.00	0.00%
9600	1250	9600.00	0.00%	625	9600.00	0.00%	313	9584.66	-0.16%
19200	625	19200.00	0.00%	313	19169.33	-0.16%	156	19230.77	0.16%
38400	313	38338.66	-0.16%	156	38461.54	0.16%	78	38461.54	0.16%
57600	208	57692.31	0.16%	104	57692.31	0.16%	52	57692.31	0.16%
76800	156	76923.08	0.16%	78	76923.08	0.16%	39	76923.08	0.16%
115200	104	115384.62	0.16%	52	115384.62	0.16%	26	115384.62	0.16%
128000	94	127659.57	-0.27%	47	127659.57	-0.27%	23	130434.78	1.90%
256000	47	255319.15	-0.27%	23	260869.57	1.90%	12	250000.00	-2.34%
1000000	12	1000000.00	0.00%	6	1000000.00	0.00%	3	1000000.00	0.00%
2000000	6	2000000.00	0.00%	3	2000000.00	0.00%	2	1500000.00	-25.00%
4000000	4	3000000.00	0.00%	2	3000000.00	0.00%	1	3000000.00	0.00%
6000000	3	4000000.00	0.00%	2	3000000.00	/	1	3000000.00	/
12000000	1	12000000.00	0.00%	1	6000000.00	/	0	/	/

15.3.5 frame error detection

When LPUART works in Mode1/2/3, it has frame error detection function. When receiving data, if the stop bit is not recognized within the expected time, it is a framing error. When a frame error is detected, LPUARTx_ISR.FE will be set to 1 by hardware, and the user program should clear this flag in time.

15.3.6 Multi-machine communication

When working in Mode2/3, set LPUARTx_CR.ADRDET to 1 to enable multi-machine communication function. The master can send address frames and data frames, and the slave can receive data frames only after receiving matching address frames.

15.3.6.1 Host sends address frame and data frame

When LPUARTx_DR[8] is 1, the address frame is sent, and LPUARTx_DR[7:0] is the address of the target slave.

When LPUARTx_DR[8] is 0, the data frame is sent, and LPUARTx_DR[7:0] is the data to be sent.

15.3.6.2 Slave receives address frame and data frame

When LPUARTx_CR.ADRDET is set to 1, the slave only receives address frames that match its own address, and ignores data frames and unmatched address frames. When the slave device 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 LPUARTx_DR register and LPUARTx_ISR.RC will be set.

When setting LPUARTx_CR.ADRDET to 0, the slave only receives data frames and ignores address frames. The user program should set LPUARTx_CR.ADRDET to 0 after the address matches to receive subsequent data frames sent by the host.

15.3.6.3 Slave address configuration

The slave address consists of a broadcast address and a user-configured address:

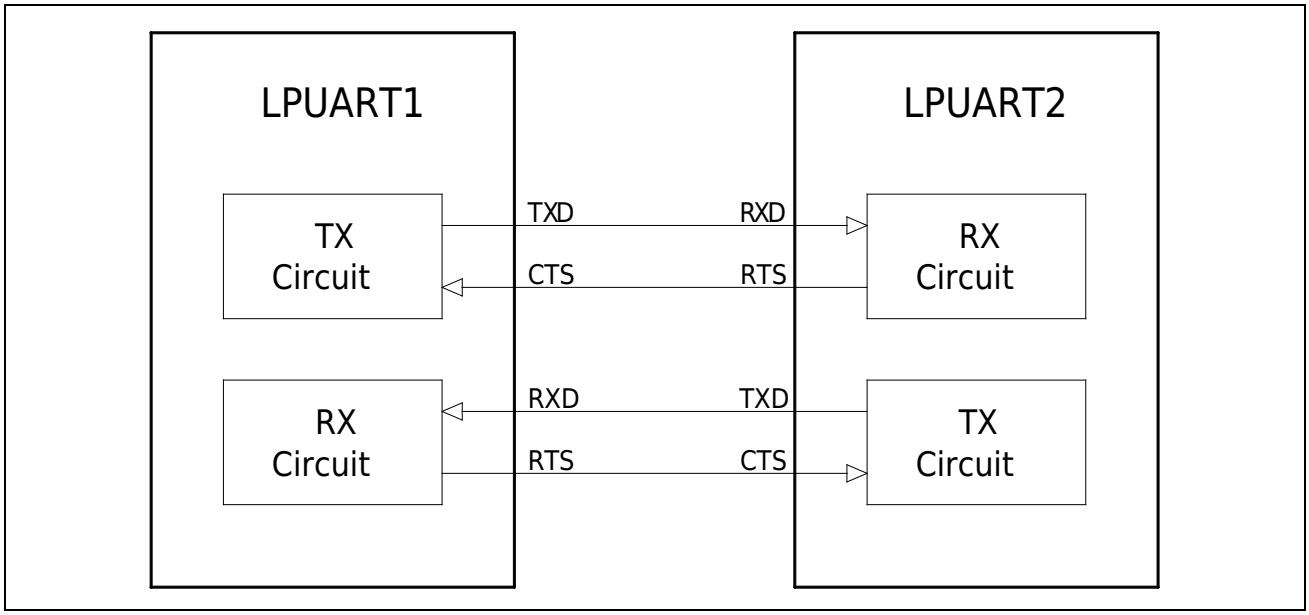
- Broadcast address: 0xFF
- User configuration address: jointly determined by LPUARTx_ADDR and LPUARTx_AddrMask

When an address frame is received, if (address received from the bus & LPUARTx_AddrMask) = (LPUARTx_ADDR & LPUARTx_AddrMask), it is considered that an address frame matching the slave address has been received.

Example: LPUARTx_ADDR = 0x55, LPUARTx_AddrMask = 0xF0, then the address that the slave can match is 0xFF (broadcast address), 0x50 ~ 0x5F (user configuration address).

15.3.7 Hardware flow control

By adding CTS and RTS signals, the function of LPUART hardware flow control can be realized, that is, the LPUART hardware module automatically controls the sending and receiving of data according to the high and low levels of CTS and RTS, without using software to judge. The hardware flow control diagram between two LPUART modules is as follows:



15.3.7.1 RTS flow control

When setting LPUARTx_CR.RTSEN to 1, the RTS pin indicates the receiving status of this module:

- When the receive buffer is empty, RTS becomes low level to notify the sender to send the next frame of data
- When the receiving buffer is full, RTS becomes high level, informing the sending end to suspend sending the next frame of data

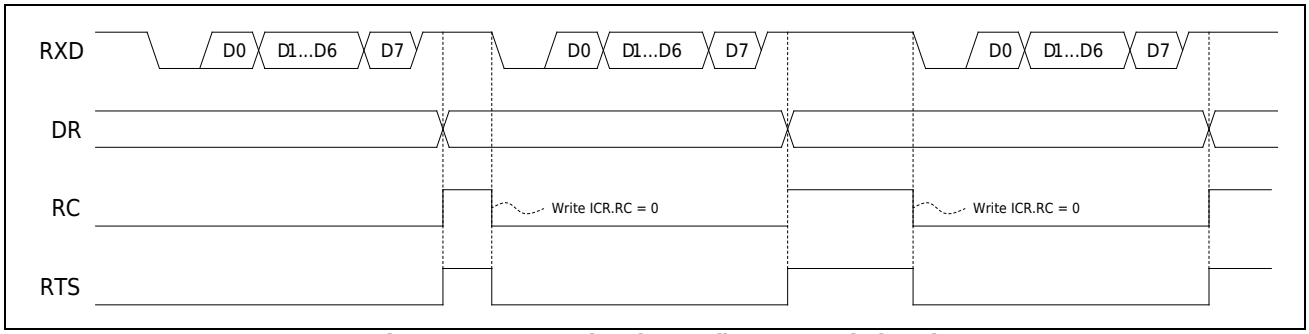


Figure 15-8 RTS hardware flow control signal

15.3.7.2 CTS flow control

When setting LPUARTx_CR.CTSEN to 1, the level of the CTS pin controls the sending rhythm of this module:

- If CTS is low, LPUART sends the next frame of data.
- If CTS is high level, after LPUART sends the current frame, wait for CTS to become low level.
- If the CTS level changes, LPUARTx_ISR.CTSIF is automatically set to 1.
- The current level value of the CTS pin can be obtained from LPUARTx_ISR.CTS.

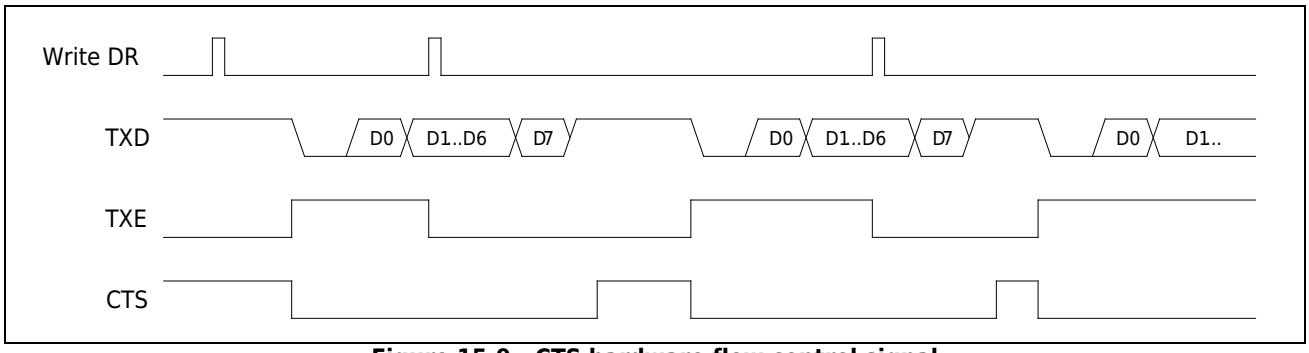


Figure 15-9 CTS hardware flow control signal

15.4programming example

15.4.1 send data example

Step1. Map TXD to the required pins according to the relevant description of the pin digital multiplexing function in the GPIO chapter; and configure TXD as a CMOS output.
Step2. Set SysCtrl_PeriClkEn0.LpUartx to 1 to enable LPUARTx configuration clock and working clock.
Step3. Configure CR.CLKSRC to select the transmission clock source of LPUART.
Step4. Configure CR.MODE to select the working mode of LPUART.
Step5. Configure the CR.OVER and BRR registers, and configure the communication baud rate.
Step6. Configure CR.STOP, CR.DR8CFG, configure the STOP length of LPUART, and the parity check mode.
Step7. Write 0x00 to ICR.TC to clear the sending completion flag.
Step8. Write a frame of data to be sent to the DR register.
Step9. Circularly query ISR.TC until it becomes 1, and the data frame written this time is sent.
Step10. If the subsequent data still needs to be sent, skip to step 7 Step7 to continue.

15.4.2 Receive data example

Step1. Map RXD to the required pin according to the relevant description of the pin digital multiplexing function in the GPIO chapter; and configure RXD as an input and enable the pull-up resistor.
Step2. Set Sysctrl_PeriClkEN0.LpUartx to 1 to enable LPUART configuration clock and working clock.
Step3. Configure CR.CLKSRC to select the transmission clock source of LPUART.
Step4. Configure CR.MODE to select the working mode of LPUART.
Step5. Configure the CR.OVER and BRR registers, and configure the communication baud rate.
Step6. Configure CR.STOP, CR.DR8CFG, configure the STOP length of LPUART, and the parity check mode.
Step7. Write 0x00 to the ICR register to clear all status flags.
Step8. Set CR.RXEN to 1 to enable RXD data reception.
Step9. Circularly query ISR.RC until it becomes 1, and one frame of data is received.
Step10. Query ISR.PE and FE to confirm whether the received data frame is valid. If it is invalid, it will handle the error; if it is valid, it will read the data from the DR register and save it.
Step11. Write 0x00 to ICR.RC to clear the receiving completion flag.
Step12. If the subsequent data still needs to be sent, skip to step 9 Step9 to continue.

15.5Register description

LPUART0 base address: 0x4000 0000

LPUART1 base address: 0x4000 4000

Register	Offset address	Description
LPUARTx_DR	0x00	Data register
LPUARTx_CR	0x04	control register
LPUARTx_ADDR	0x08	address register
LPUARTx_AddrMask	0x0C	Address Mask Register
LPUARTx_ISR	0x10	interrupt flag register
LPUARTx_ICR	0x14	Interrupt flag bit clear register
LPUARTx_BRR	0x18	baud rate register

15.5.1 Data Register (LPUARTx_DR)

Offset address: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							DR8	DR							
							RW	RW							

Bit	Marking	Functional description
31:9	Reserved	Keep
8	DR8	In Mode0/1, read this bit is 0, write this bit is invalid; In Mode2/3, this bit represents the Bit8 data bit, which can be divided into the following two situations: (1) When the hardware parity bit is turned on, this bit is the parity bit of the received data when receiving, and the verification is performed by the hardware. If the verification error occurs, the verification error flag bit PE is set to 1; this bit is invalid when sending, The parity bit of the sent data is calculated and sent by the hardware; (2) When the hardware parity bit is turned off, this bit is received data Bit8 when receiving; this bit is sent data Bit8 when sending; Note: When the multi-machine communication mode is turned on, the parity check of the received data is automatically closed; the parity check of the sent data is still controlled by B8CFG;
7:0	DR	When sending data, write the data to be sent into this register When receiving data, read the received data from this register

15.5.2 Control Register (LPUARTx_CR)

Offset address: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									HDSEL	FEIE	CTSIE	CTSEN	RTSEN	Reserved	
									RW	RW	RW	RW	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STOP		PEIE	CLKSRC		OVER		TXEIE	MODE		ADRDET	RXEN	DR8CFG		TCIE	RCIE
RW		RW	RW		RW		RW	RW		RW	RW	RW		RW	RW

Bit	Marking	Functional description
31:23	Reserved	reserved bit
22	HDSEL	Single bus function enable control 0: disable single bus function 1: enable single bus function
21	FEIE	Framing Error Interrupt Enable Control 0: disable frame error interrupt 1: Enable frame error interrupt
20	CTSIE	CTS signal toggle interrupt enable control 0: Disable CTS signal toggle interrupt 1: Enable CTS signal flip interrupt
19	CTSEN	CTS signal enable control 0: disable CTS signal 1: Enable CTS signal
18	RTSEN	RTS signal enable control 0: disable RTS signal 1: Enable RTS signal
17:16	Reserved	reserved bit
15:14	STOP	STOP bit duration configuration 00: 1 bit 01: 1.5 bit 10: 2 bit 11: reserved Note: In Mode0, STOP needs to be set to 00
13	PEIE	Parity error interrupt enable control 1: Enable parity error interrupt 0: Disable parity error interrupt
12:11	CLKSRC	Transmission clock SCLK source selection 11: RCL 10: RCL 01: PCLK 00: PCLK
10:9	OVER	Sampling frequency division configuration in each mode

OVER	Mode0	Mode1	Mode2	Mode3
11	Res.	Res.	Res.	Res.
10	Res.	4x sampling	8x sampling	4x sampling
01	Res.	8x sampling	16x sampling	8x sampling
00	Res.	16x sampling	32x sampling	16x sampling

8	TXEIE	TX Buf empty interrupt enable control 1: Enable TX Buf empty interrupt 0: disable TX Buf empty interrupt
7:6	MODE	Working mode configuration 00: Mode0, synchronous half-duplex, Data(8bit) 01: Mode1, asynchronous full duplex, Start(1bit) + Data(8bit) + Stop (1~2bit) 10: Mode2, asynchronous full duplex, Start (1bit) + Data(8bit) + B8(1bit) + Stop (1~2bit) 11: Mode3, asynchronous full duplex, Start (1bit) + Data(8bit) + B8(1bit) + Stop (1~2bit)
5	ADRDET	Multi-machine communication enable control 0: disabled, B8 function is determined by DR8CFG when sending and receiving 1: Enable, B8 is fixed at 1 when sending, and address matching detection will be performed when receiving

4	RXEN	Mode0: 0: send; 1: receive; Mode1/2/3: 0: send; 1: receive / send;
3:2	DR8CFG	DR8 data bit function configuration 00: Determined by software reading and writing DR[8] 01: hardware even parity 10: hardware odd parity 11: reserved
1	TCIE	Transmit complete interrupt enable control 1: Enable transmit complete interrupt 0: Disable transmit complete interrupt
0	RCIE	Receive complete interrupt enable control 1: Enable receive complete interrupt 0: Disable receive complete interrupt

15.5.3 Address Register (LPUARTx_ADDR)

Offset address: 0x08

Reset value: 0x0000 0000

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

Bit	Marking	Functional description
31:8	Reserved	Keep
7:0	ADDR	Slave device address

15.5.4 Address Mask Register (LPUARTx_AddrMask)

Offset address: 0x0C

Reset value: 0x0000 0000

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

Bit	Marking	Functional description
31:8	Reserved	Keep
7:0	MASK	Slave Device Address Mask Register

15.5.5 Flag Bit Register (LPUARTx_ISR)

Offset address: 0x10

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									CTS	CTSIF	PE	TXE	FE	TC	RC
									RO	RO	RO	RO	RO	RO	RO

Bit	Symbol	Functional description
31:7	Reserved	Keep
6	CTS	CTS pin level 0: CTS pin is low level 1: CTS pin is high level
5	CTSIF	CTS pin level change flag 0: CTS pin level does not change 1: CTS pin level has changed
4	PE	Parity error flag 0: No parity error occurred 1: A parity error has occurred
3	TXE	Tx Buffer free flag 0: Tx Buffer is not empty 1: Tx Buffer is idle
2	FE	frame error flag 0: No framing error occurred 1: A framing error has occurred
1	TC	TXD send completion flag bit 0: TXD is sending data 1: TXD transmission completed
0	RC	receive complete flag 0: no data received 1: A frame of data has been received

15.5.6 Flag Clear Register (LPUARTx_ICR)

Offset address: 0x14

Reset value: 0x0000 0037

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										CTSIF	PE	Res.	FE	TC	RC
										R1W0	R1W0		R1W0	R1W0	R1W0

Bit	Marking	Functional description
31:6	Reserved	Keep
5	CTSIF	ISR.CTSIF Flag Clear Control 1: no function 0: set the CTSIF flag to 0
4	PE	ISR.PE Flag Clear Control 1: no function 0: set the PE flag to 0
3	Res.	Keep
2	FE	ISR.FE Flag Clear Control 1: no function 0: set FE flag to 0
1	TC	ISR.TC Flag Clear Control 1: no function 0: set TC flag to 0
0	RC	ISR.RC Flag Clear Control 1: no function 0: set RC flag to 0

15.5.7 Baud Rate Register (LPUARTx_BRR)

Offset address: 0x18

Reset value: 0x0000 0000

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

Bit	Marking	Functional description
31:16	Reserved	Keep
15:0	SCNT	baud rate counter

16 Serial Peripheral Interface (SPI)

16.1 Overview

The serial peripheral interface (SPI) provides the 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.

16.2 Main characteristics

- Can be configured as master or slave, support multi-machine mode
- The maximum frequency division factor of the host mode is PCLK/2, and the maximum communication rate is 16M bps
- The maximum frequency division factor of slave mode is PCLK/4, and the maximum communication rate is 8M bps
- 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: 4bit ~ 16bit
- Two NSS methods: hardware control, software control
- Configurable serial clock polarity and phase

16.3Functional description

16.3.1 Functional block diagram

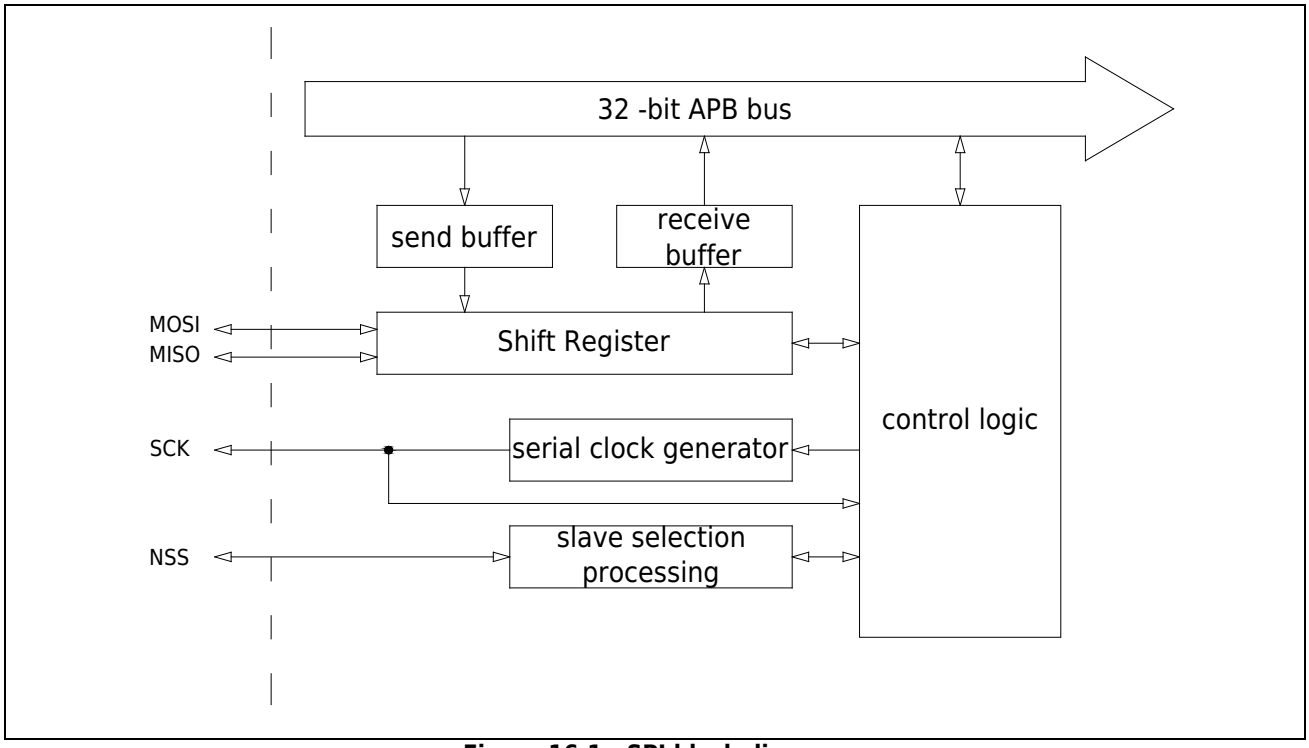


Figure 16-1 SPI block diagram

16.3.2 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 4bit to 16bit 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.

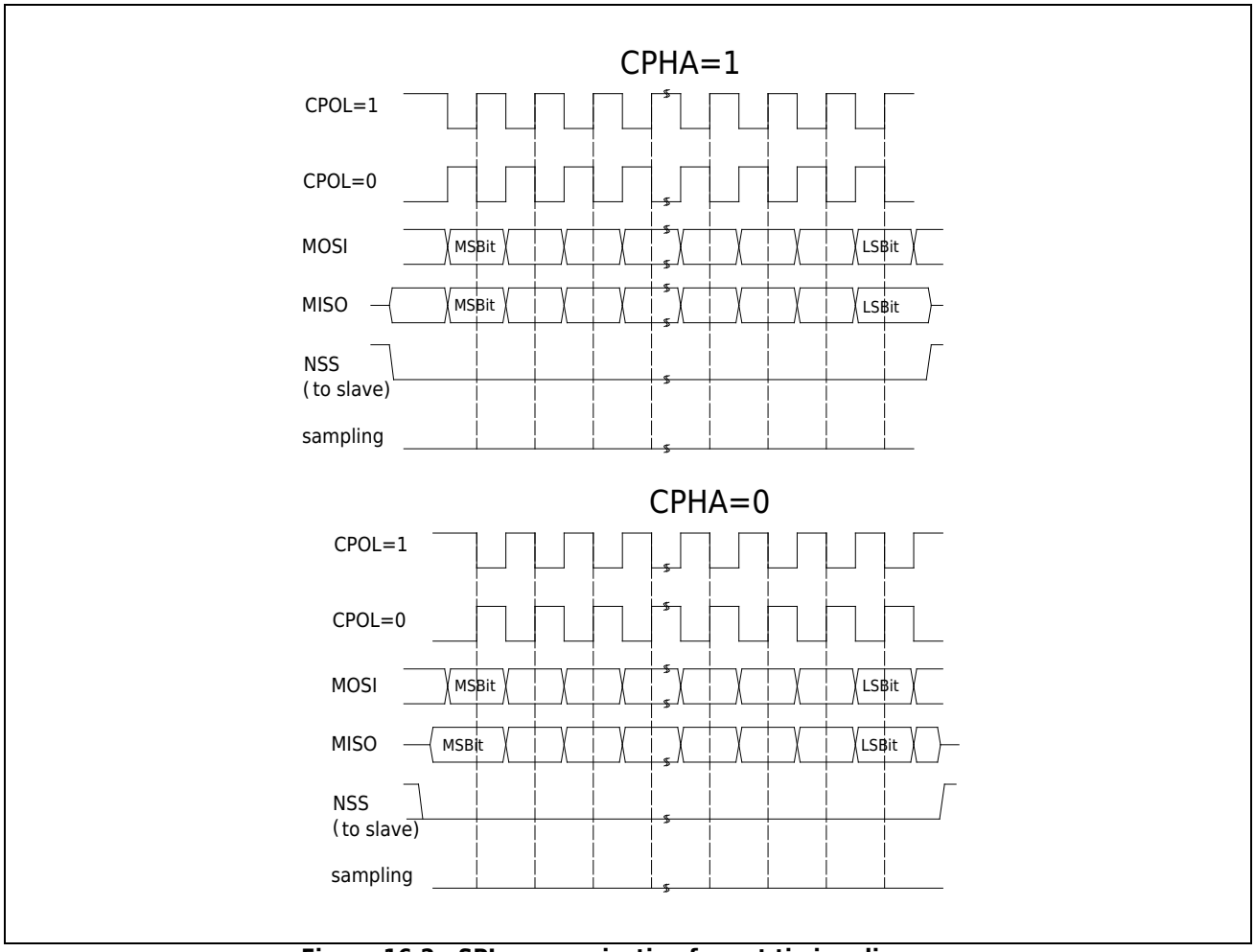


Figure 16-2 SPI communication format timing diagram

16.3.3 Slave Select Pin Configuration

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

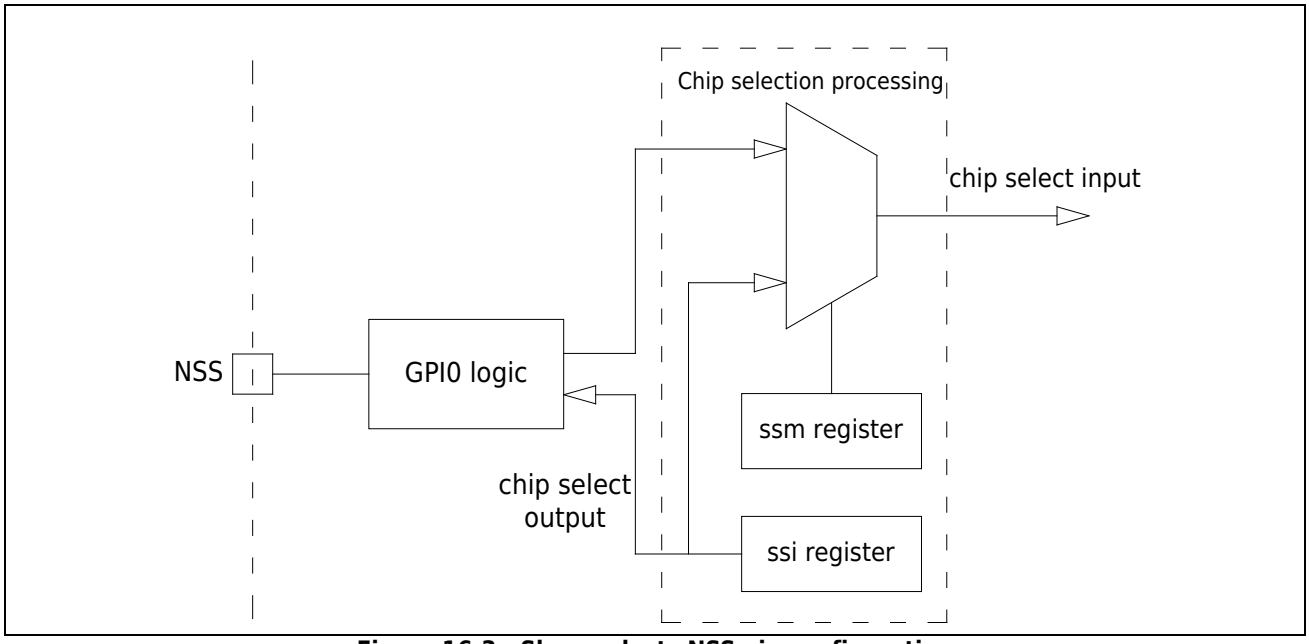


Figure 16-3 Slave selects NSS pin configuration

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 a chip selection function, and this module can be selected when the register SPI_SSI.SSI is 0.

16.3.4 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.

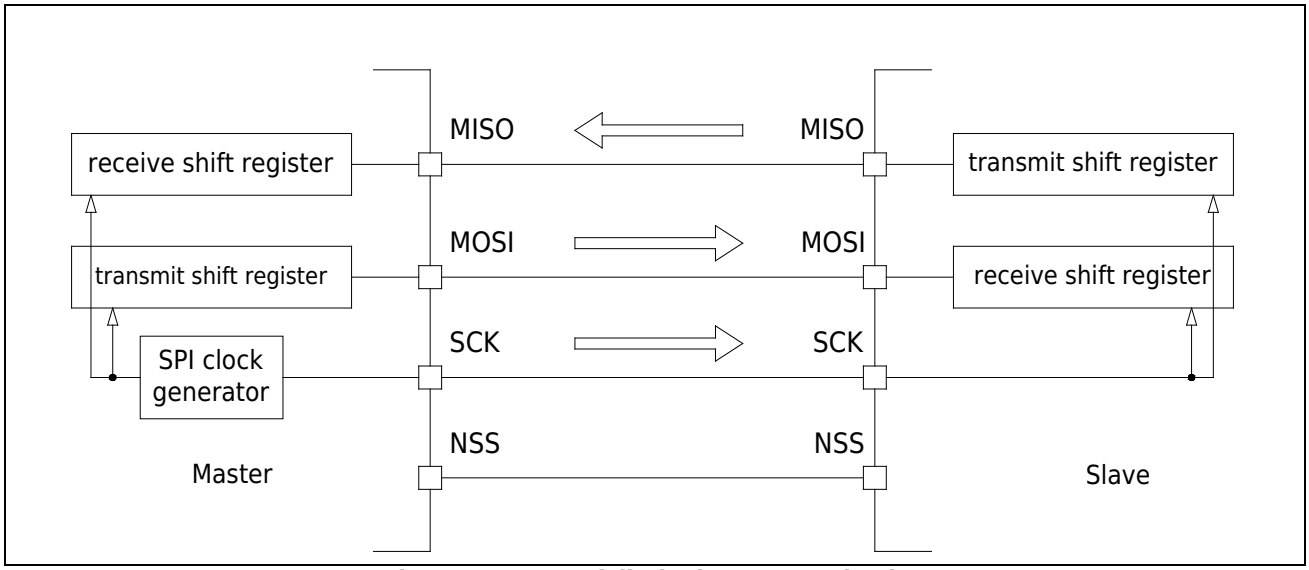


Figure 16-4 SPI full- duplex communication

16.3.5 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.HDOE. When HDOE.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 serially insert a current-limiting resistor on the data line to protect the chip pins.

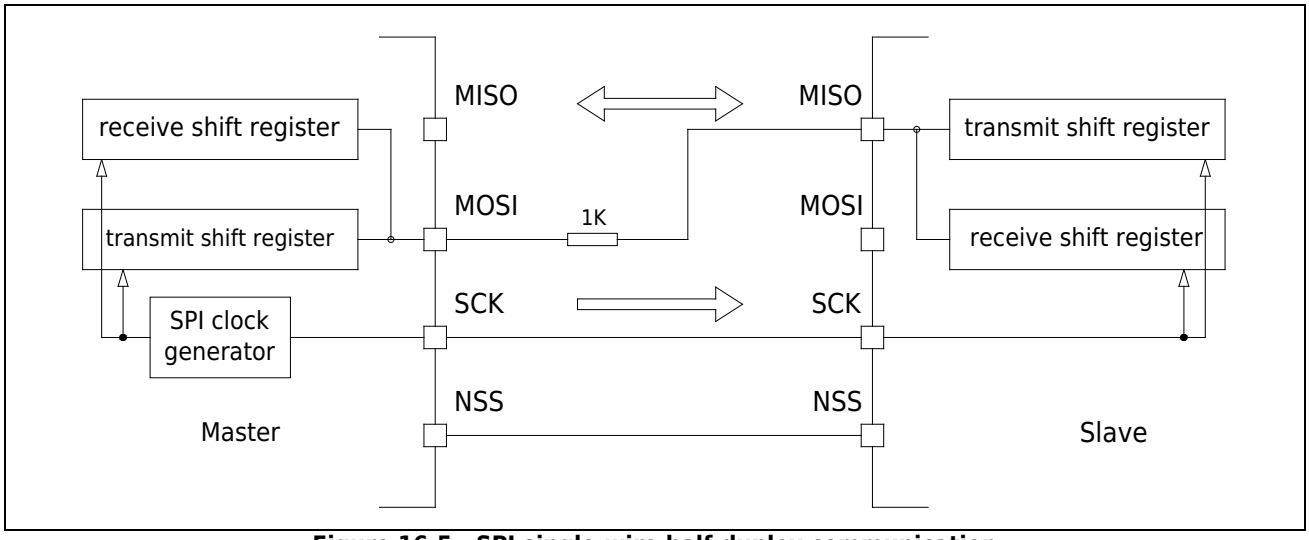


Figure 16-5 SPI single-wire half-duplex communication

16.3.6 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 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.

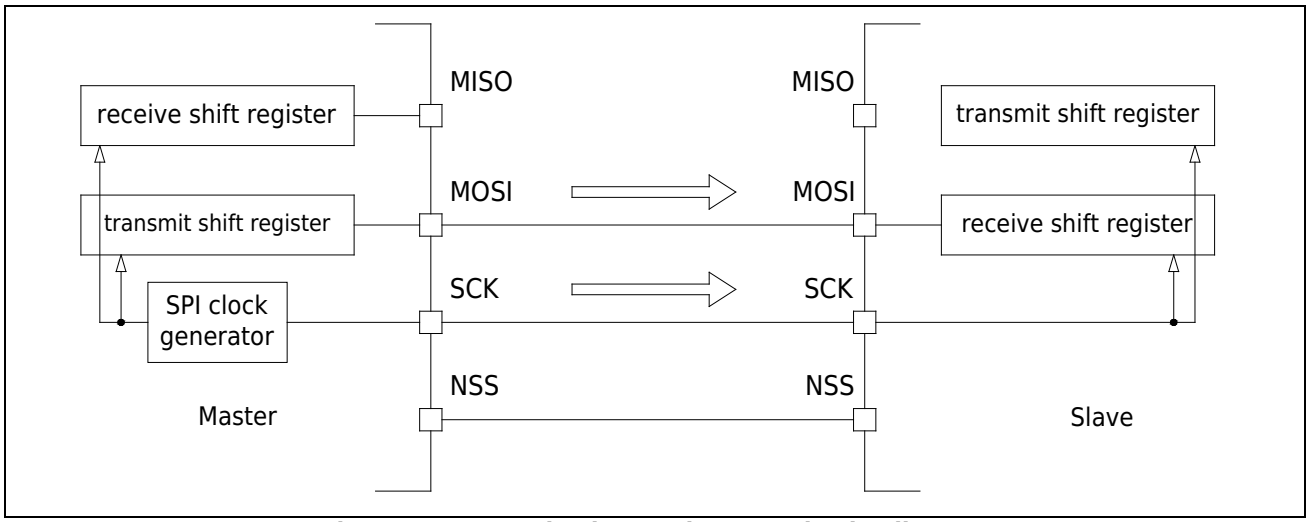


Figure 16-6 SPI simplex send communicationdiagram

Note:

- The 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.
- In this configuration, both MISO pins can be used as GPIO.

16.3.7 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. When the host tries to obtain the control right of the SPI bus, it can combine the mode error MODF to detect whether the bus conflicts. 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.

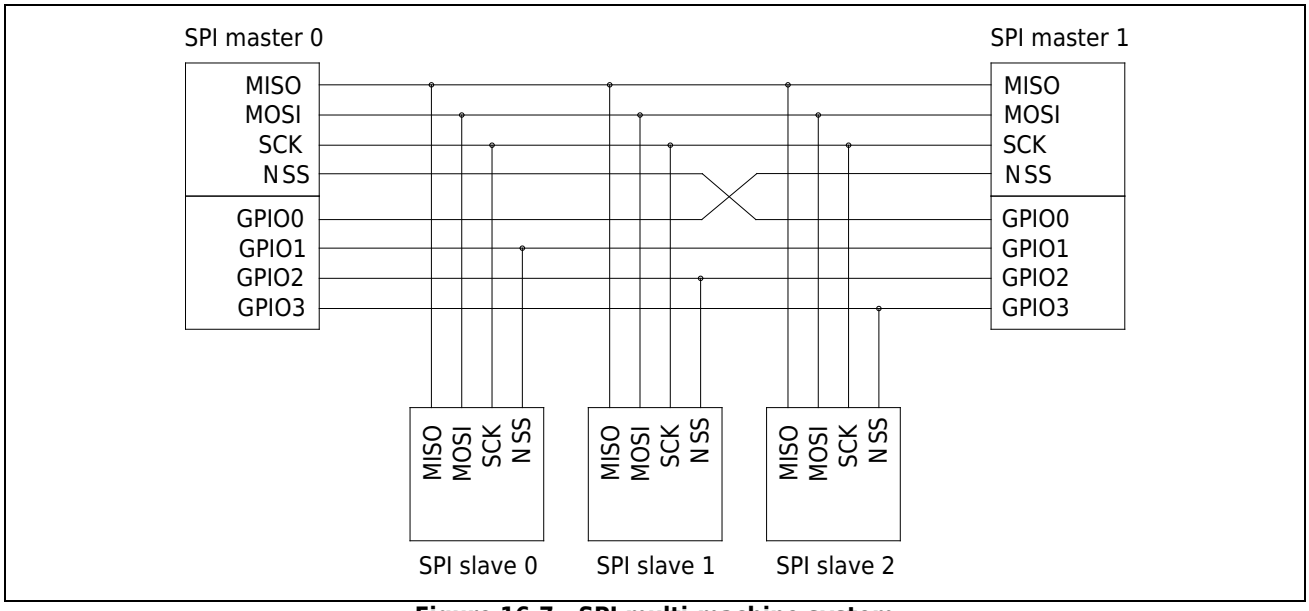


Figure 16-7 SPI multi-machine system

16.3.8 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 receive buffer non-empty flag indicates whether there is received data in the receive 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

The size of the 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.

If a mode error occurs, the SPI will 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.

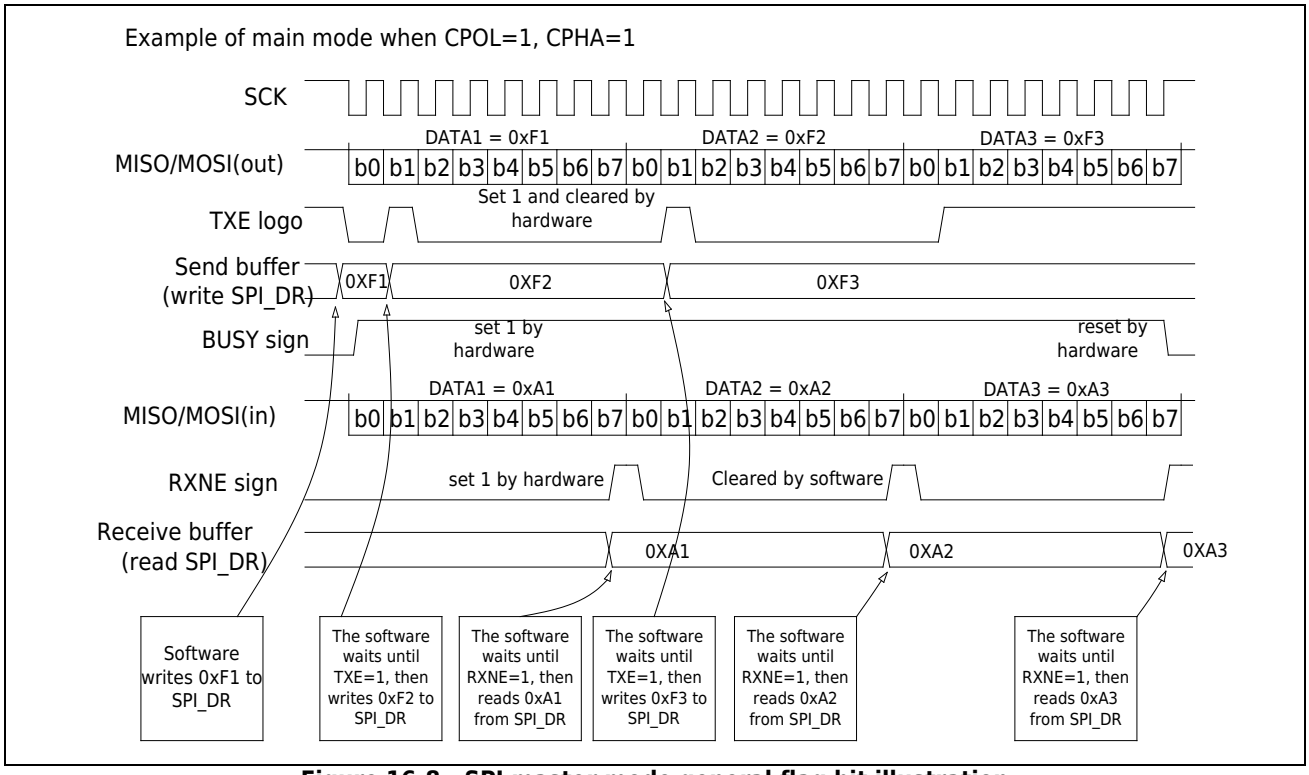


Figure 16-8 SPI master mode general flag bit illustration

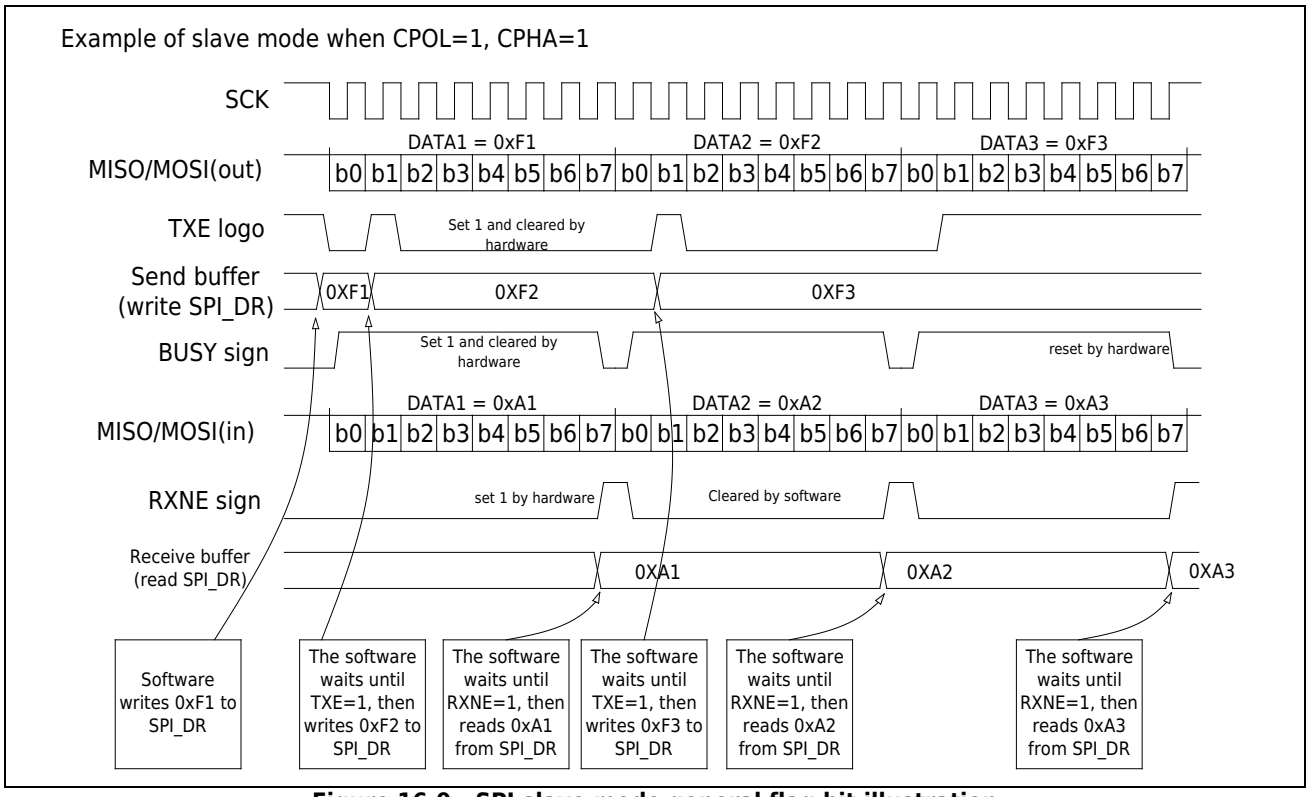


Figure 16-9 SPI slave mode general flag bit illustration

16.3.9 Interrupt generation and clearing

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

Table 16-1 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 interrupts, no matter whether the interrupt is enabled or not, the method described in the table can be used to clear the interrupt flag.

16.4programming example

16.4.1 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_PeriClkEN0.SPI to 1 to enable SPI configuration clock and working clock.

Step3. Set CR0.MSTR to 1 to select master 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 configure slave select mode.

Step7. Set CR0.CPOL, CR0.CPHA to 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 and select the corresponding slave.

Step12. Write the data to be sent into the DR register, and the SPI starts data transmission automatically.

Step13. The query waits for SR.RXNE to become 1, and completes 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.

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

Step1. Map SPI_MOSI and SPI_MISO to the required pins according to the relevant description of the pin digital multiplexing function in the GPIO chapter.

Step2. Set Sysctrl_PeriClkEN0.SPI to 1 to enable SPI configuration clock and working clock.

Step3. Set CR0.MSTR to 0 to select slave mode.

Step4. Set CR0.CM to 0, select full-duplex communication mode.

Step5. Set CR0.SSM to configure slave select mode.

Step6. Set CR0.CPOL, CR0.CPHA to 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 master selects this module.

Step12. Write the data to be sent into the DR register, and prepare the data to be sent.

Step13. The query waits for SR.RXNE to become 1, and completes 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.

16.4.3 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_PeriClkEN0.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 and CR0.CPHA to 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 master 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 machine 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 transmit 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. The master writes arbitrary data to the DR register, and the SPI master sends SCK for data transfer.

Step21. The host query waits for SR.RXNE to become 1 to complete 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.

16.5Register description

register list

Table 16-2 SPI register list

SPI Base Address: 0x40000800

Offset	register name	access	Register description
0x00	SPI_CR0	RW	SPI Control Register 0
0x04	SPI_CR1	RW	SPI Control Register 1
0x08	SPI_HDOE	RW	SPI single-wire half-duplex output enable register
0x0C	SPI_SSI	RW	SPI Internal Slave Select Register
0x10	SPI_SR	RO	SPI status register
0x14	SPI_ICR	WO	SPI Interrupt Clear Register
0x18	SPI_DR	RW	SPI data register

16.5.1 SPI Control Register 0 (SPI_CR0)

Address offset: 0x00

Reset value: 0x00000700

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CM		SSM	WIDTH				LSBF	MSTR	CPOL	CPHA	BRR			EN
	RW		RW	RW				RW	RW	RW	RW	RW			RW

Bit	Marking	Functional description
31:15	Reserved	
14:13	CM	Communication mode configuration 00: full duplex bidirectional 01: simplex only send 10: simplex only 11: Single-wire half-duplex
12	SSM	Slave Select Configuration (Master Mode) 1: SSI register determines slave select output value 0: NSS pin determines slave select input value Slave Select Configuration (Slave Mode) 1: SSI register determines slave select input value 0: NSS pin determines slave select input value
11:8	WIDTH	Frame data width configuration 0011: 4bit 0100: 5bit ... 1110: 15bit 1111: 16bit
7	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
6	MSTR	Working mode configuration 1: Host mode 0: Slave mode
5	CPOL	Serial Clock Polarity Configuration 1: High level during standby 0: low level in standby
4	CPHA	Serial Clock Phase Configuration 1: Front edge shift / back edge sampling 0: sampling on the leading edge / shifting on the trailing edge
3:1	BRR	Host mode baud rate configuration 000: PCLK/2 001: PCLK/4 010: PCLK/8 011: PCLK/16 ... 110: PCLK/128 111: reserved
0	EN	enable control 1: Enable SPI module 0: disable SPI module

16.5.2 SPI Control Register 1 (SPI_CR1)

Address offset: 0x04

Reset value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								MODFIE	SSERRIE	OVFIE	UDFIE	SSRIE	SSFIE	RXNEIE	TXEIE
								RW	RW	RW	RW	RW	RW	RW	RW

Bit	Marking	Functional description
31:8	Reserved	
7	MODFIE	Mode Error Interrupt Enable Control 1: Enable 0: disabled
6	SSERRIE	Slave Select Error Interrupt Enable Control in Slave Mode 1: Enable 0: disabled
5	OVFIE	Receive buffer overflow error interrupt enable control 1: Enable 0: disabled
4	UDFIE	Transmit buffer underflow error interrupt enable control in slave mode 1: Enable 0: disabled
3	SSRIE	Slave select input rising edge interrupt enable control 1: Enable 0: disabled
2	SSFIE	Slave select input falling edge interrupt enable control 1: Enable 0: disabled
1	RXNEIE	Receive buffer non-empty interrupt enable control 1: Enable 0: disabled
0	TXEIE	Transmit buffer empty interrupt enable control 1: Enable 0: disabled

16.5.3 SPI single-wire half-duplex output enable register (SPI_HDOE)

Address offset: 0x08

Reset value: 0x00000000

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

Bit	Marking	Functional description
31:1	Reserved	reserved bit
0	HDOE	MOSI/MISO transceiver switching control 0: receive only 1: send only Note: only valid in single-line half-duplex communication

16.5.4 SPI Internal Slave Select Register (SPI_SSI)

Address offset: 0x0C

Reset value: 0x00000001

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

Bit	Marking	Functional description
31:1	Reserved	reserved bit
0	SSI	Internal slave selection, valid when the SPI_CR0.SSM register is 1 (the register determines the slave selection output value in master mode / the register determines the slave selection input value in slave mode)

16.5.5 SPI status register (SPI _ SR)

Address offset: 0x10

Reset value: 0x00000001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						SSLVL	BUSY	MODF	SSERR	OVF	UDF	SSR	SSF	RXNE	TXE
						RO	RO	RO	RO	RO	RO	RO	RO	RO	RO

Bit	Marking	Functional description
31:10	Reserved	
9	SSLVL	State of slave select input 1: Slave select input is high level 0: Slave select input is low level Note: When the master mode and SPI_CR0.SSM=1, this bit is meaningless.
8	BUSY	Bus busy flag bit 1: Bus busy 0: Bus idle
7	MODF	Mode error flag (MODF is triggered when the slave selection input is low in master mode) 1: error 0: normal
6	SSERR	Slave selection error flag in slave mode 1: error 0: normal
5	OVF	Receive buffer overflow error flag 1: error 0: normal
4	UDF	Transmit buffer underflow error flag in slave mode 1: error 0: normal
3	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.
2	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.
1	RXNE	Receive buffer not empty flag 1: Receive buffer is not empty 0: receive buffer is empty
0	TXE	send buffer empty flag 1: send buffer empty 0: send buffer is not empty

16.5.6 SPI Interrupt Clear Register (SPI_ICR)

Address offset: 0x14

Reset value: 0x000000FF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								MODF	SSERR	OVF	UDF	SSR	SSF	RXNE	TXE
								R1W0	R1W0	R1W0	R1W0	R1W0	R1W0	R1W0	R1W0

Bit	Marking	Functional description
31:8	Reserved	reserved bit
7	MODF	Write 0 to clear the ISR.MODF flag, read as 1.
6	SSERR	Write 0 to clear the ISR.SSERR flag, read as 1.
5	OVF	Write 0 to clear the ISR.OVF flag, read as 1.
4	UDF	Write 0 to clear the ISR.UDF flag, read as 1.
3	SSR	Write 0 to clear the ISR.SSR flag, read as 1.
2	SSF	Write 0 to clear the ISR.SSF flag, read as 1.
1	RXNE	Write 0 to clear ISR.RXNE flag, read as 1.
0	TXE	Writing 0 sets the ISR.TXE flag to 1 and reading it to 1.

16.5.7 SPI data register (SPI _ DR)

Address offset: 0x18

Reset value: 0x0000 0000

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

Bit	Marking	Functional description
31:16	Reserved	
15:0	DR	When sending data, write the data to be sent into this register When receiving data, read the received data from this register

17 I2C bus (I2C)

17.1 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. The 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 hosts request control of the bus at the same time.

The 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.

17.2 Main characteristics

The 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
- Host supports 10 -bit addressing function
- Support noise filtering function
- Supports three slave addresses
- Support broadcast address
- Support interrupt status query function

17.3protocol description

The I2C bus transfers information using the SCL (serial clock bus) and SDA (serial data bus) that connect devices. The host 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.

17.3.1 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).

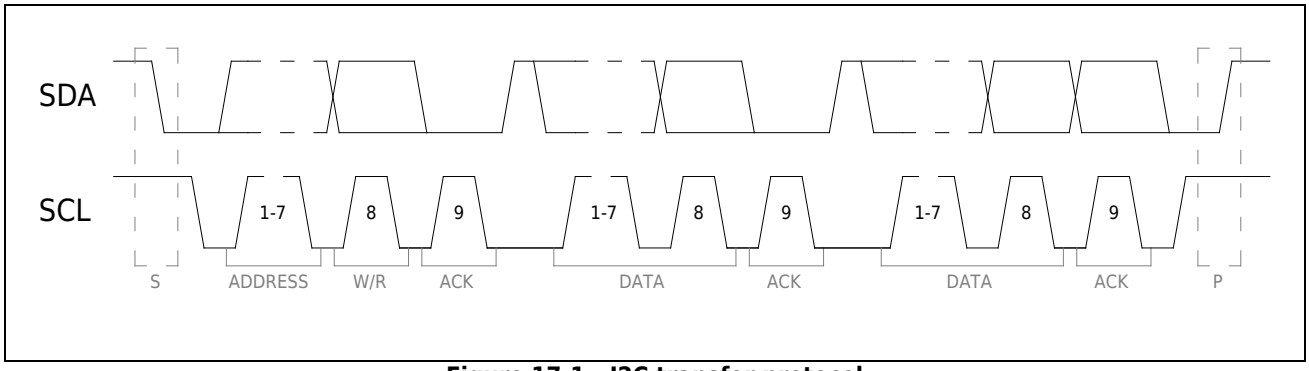


Figure 17-1 I2C transfer protocol

- 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.

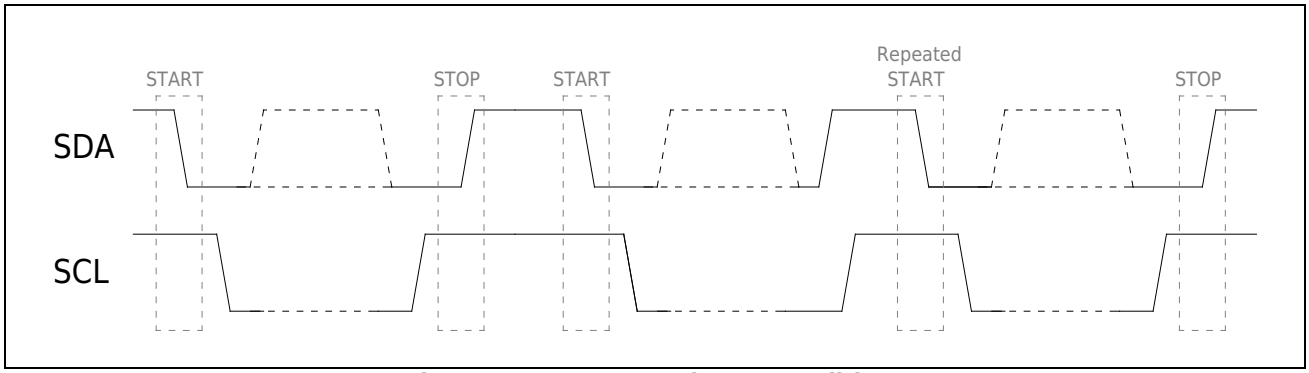


Figure 17-2 START and STOP conditions

- 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.

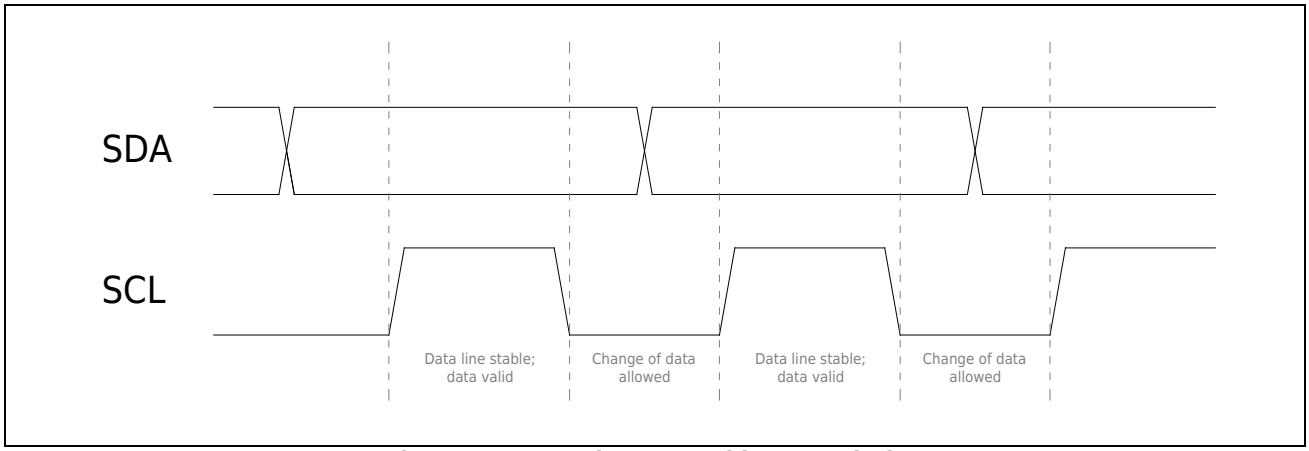


Figure 17-3 I2C bus upper bit transmission

17.3.2 Acknowledgment on the I2C bus

Each byte transferred is followed by an acknowledge bit. By pulling the SDA line low, the receiver is allowed to echo the transmitter. 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.

When the host is used as a sending device, if a non-response signal (NACK) is generated on the slave, the host 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 host is used as a receiving device, a non-response signal (NACK) occurs, and the slave releases the SDA line, causing the host to generate a stop signal or a repeated start signal.

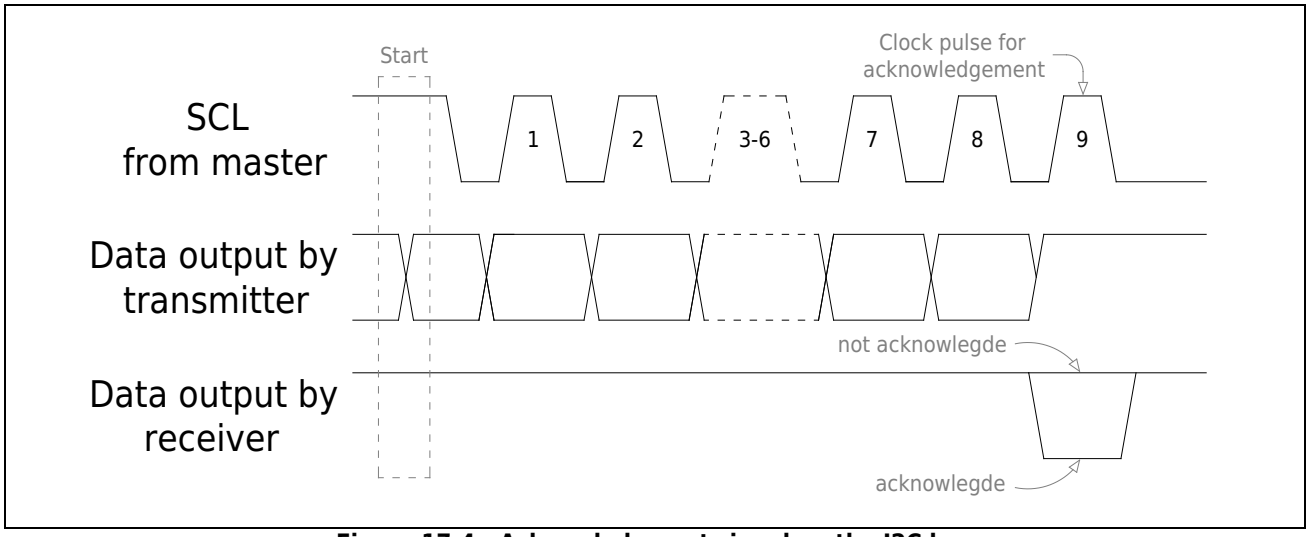


Figure 17-4 Acknowledgment signal on the I2C bus

17.3.3 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 hosts 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 waits until the slave completes the operation. And release 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 hosts attempt to control the bus.

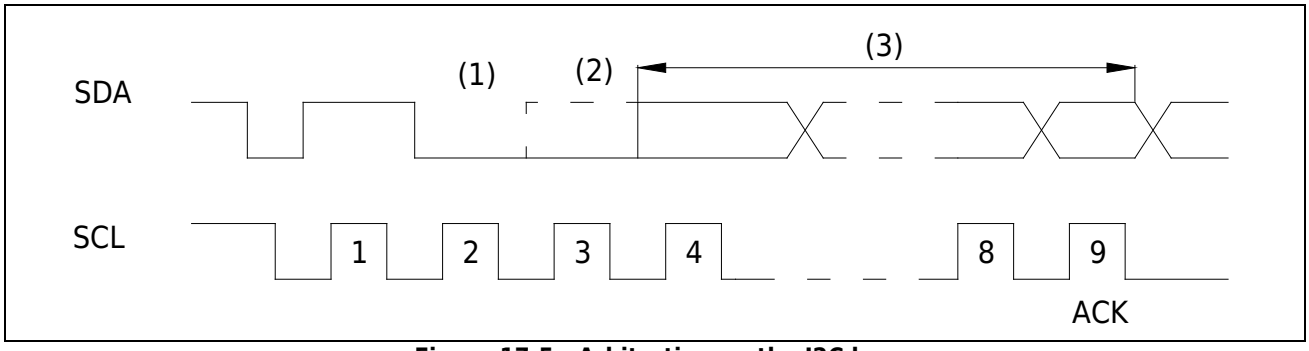


Figure 17-5 Arbitration on the I2C bus

1) Another device sends serial data;
2) Another device first negates a logic 1 sent by the I2C master by pulling SDA low (dotted line). Arbitration is lost, I2C enters slave receive mode;
3) At this time, I2C is in the slave receiving mode, but still generates clock pulses until the current byte is sent. I2C will not generate a clock pulse for the next byte transfer. Once arbitration is won, the data transfer on SDA is initiated by the new master.

17.4 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 hosts start transmitting data at the same time. The status of the I2C bus can be queried in the status register.

17.4.1 Functional block diagram

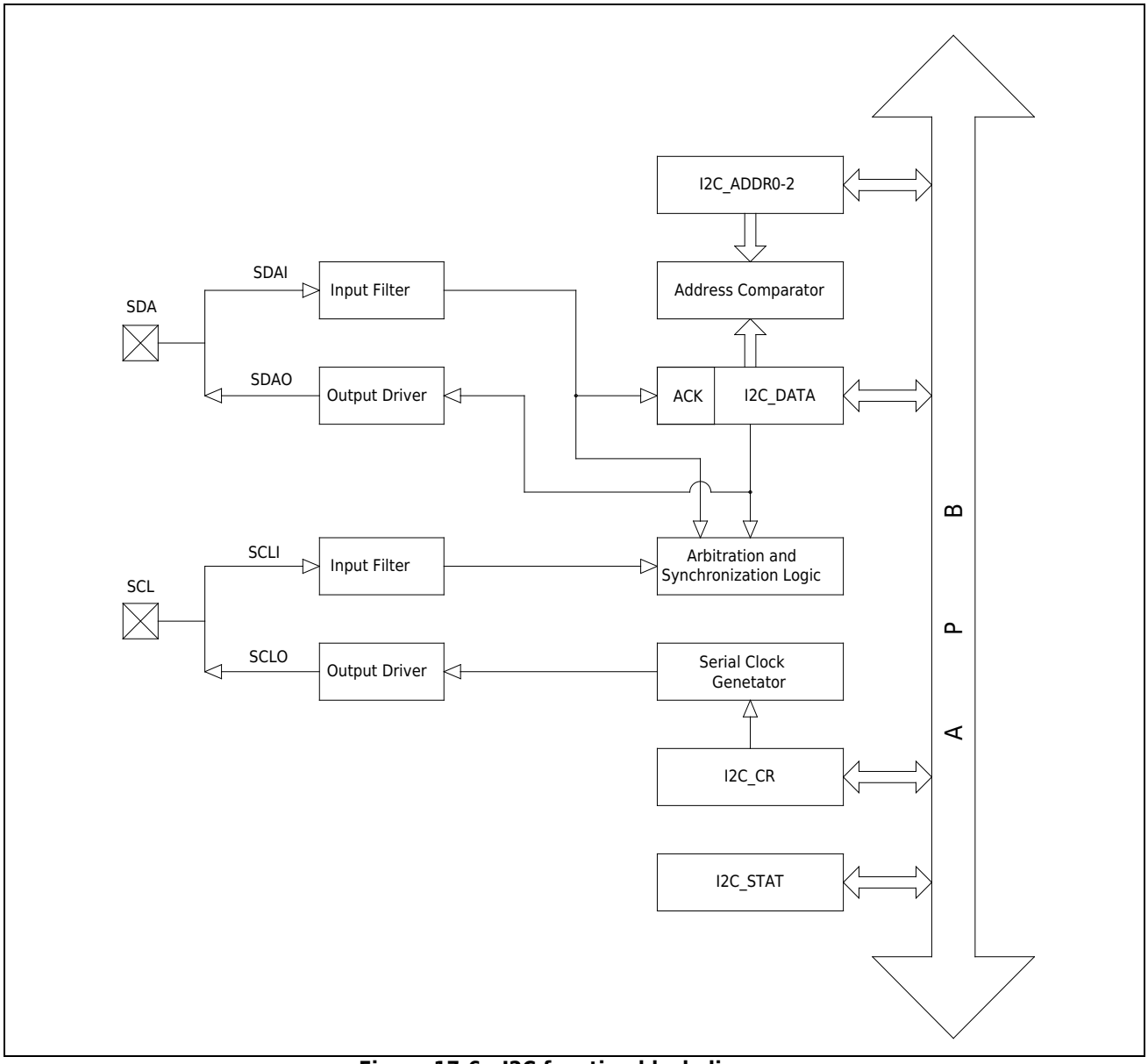


Figure 17-6 I2C function block diagram

17.4.2 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 17-1 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

17.4.3 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.

17.4.4 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.

17.4.5 response flag

The 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.

17.4.6 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.

17.4.7 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: Host send mode, Host 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.

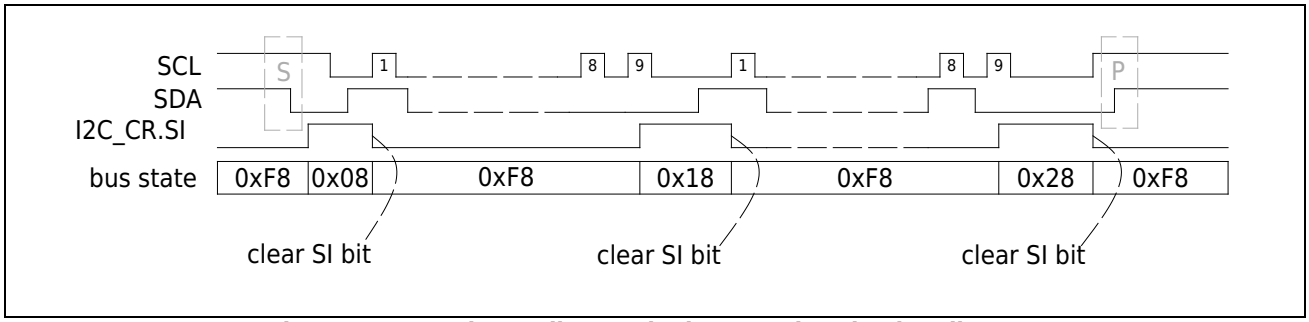


Figure 17-7 Main sending mode data synchronization diagram

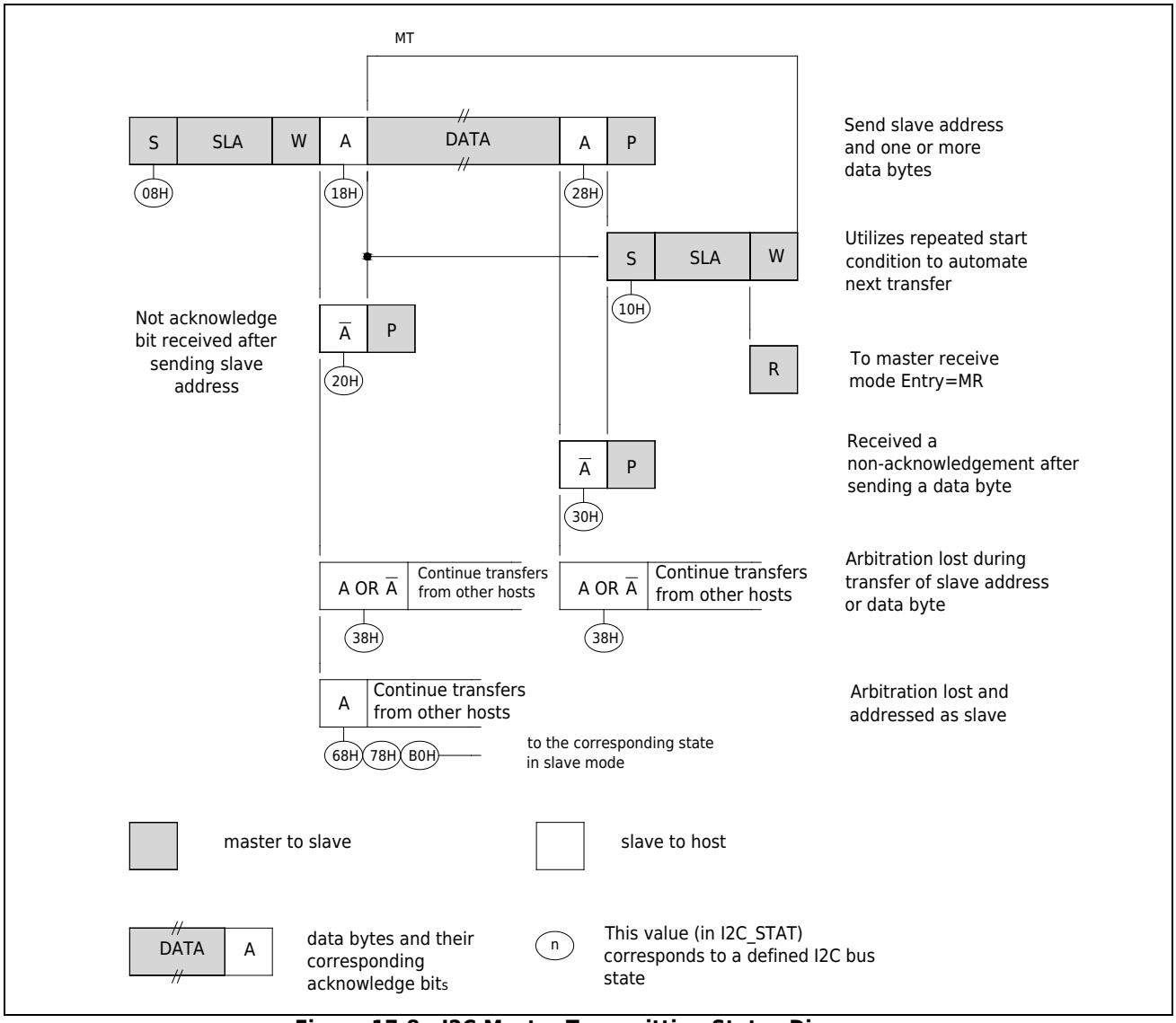


Figure 17-8 I2C Master Transmitting Status Diagram

- Host receiving mode

  The master receives the mode, and the data is transmitted by the slave. The initialization setting is the same as the master sending mode, after the master 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 an acknowledgment bit after receiving the data; if it is 0, the master does not respond with a NACK after receiving the data. Then the host can send a stop signal or repeat the start signal to start the next round of data transmission.

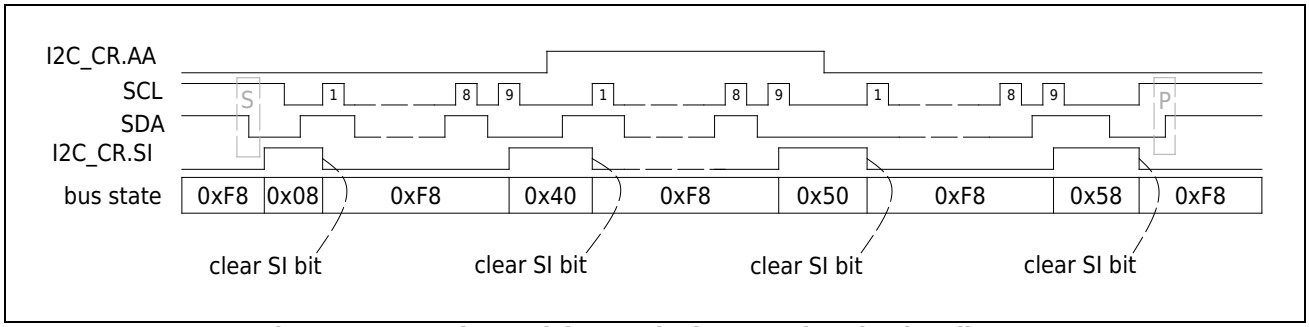


Figure 17-9 Main receiving mode data synchronization diagram

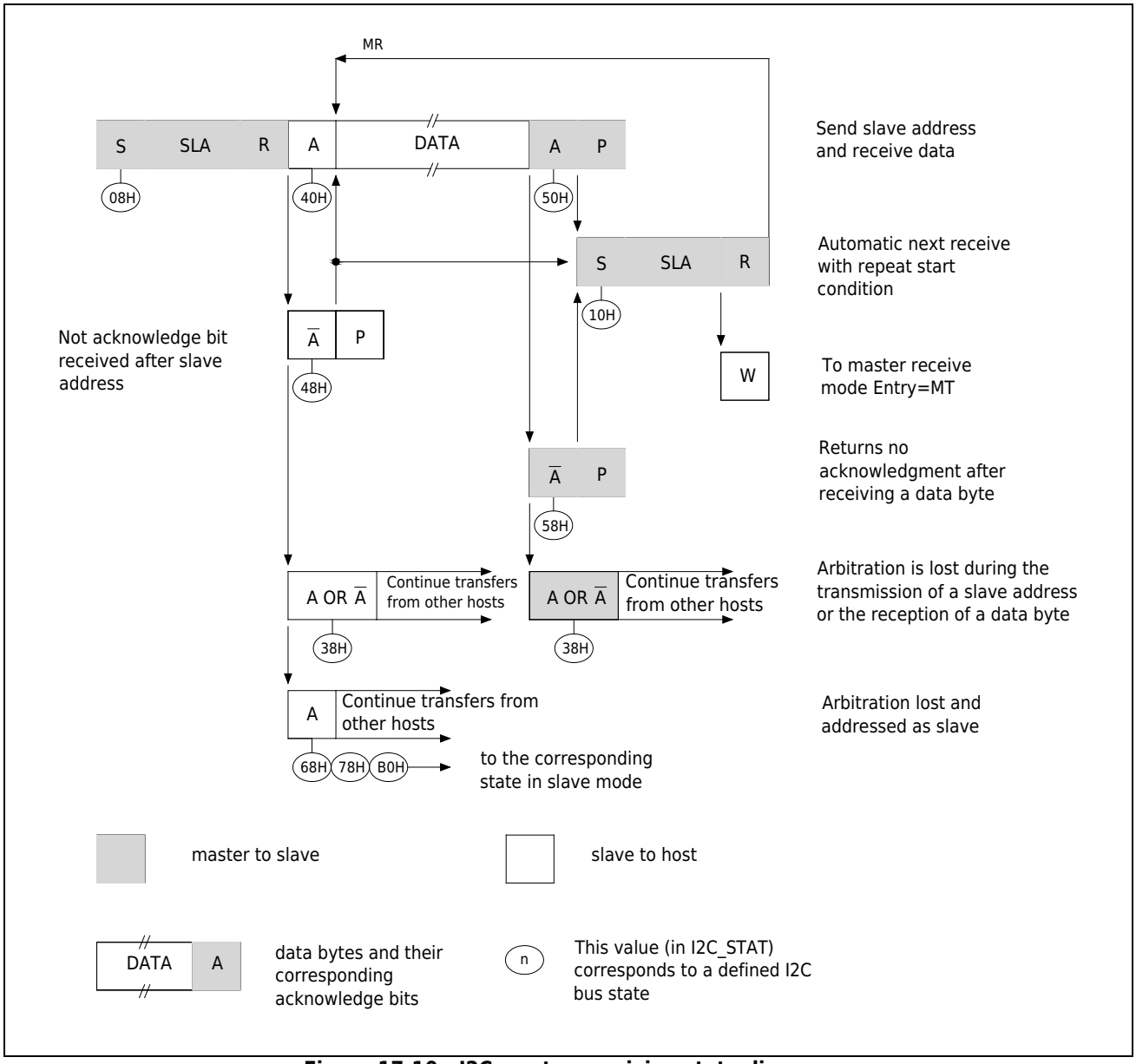


Figure 17-10 I2C master receiving state diagram

- 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 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.

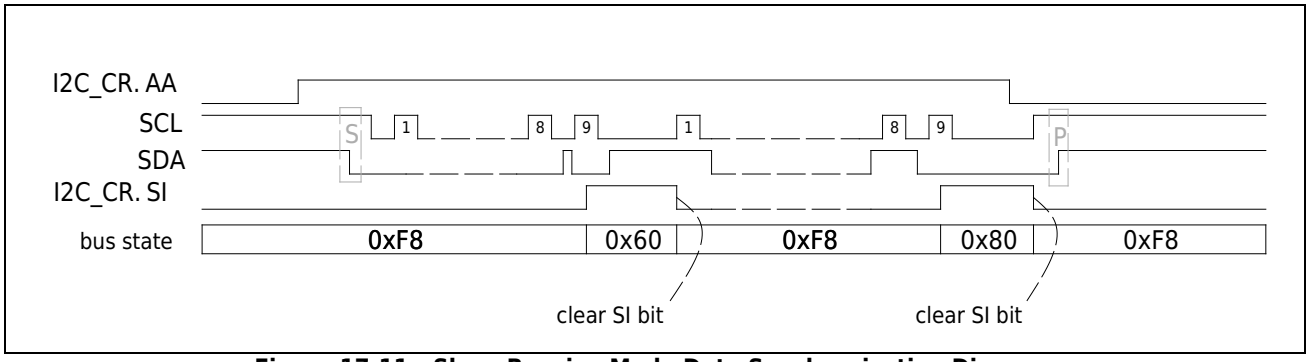


Figure 17-11 Slave Receive Mode Data Synchronization Diagram

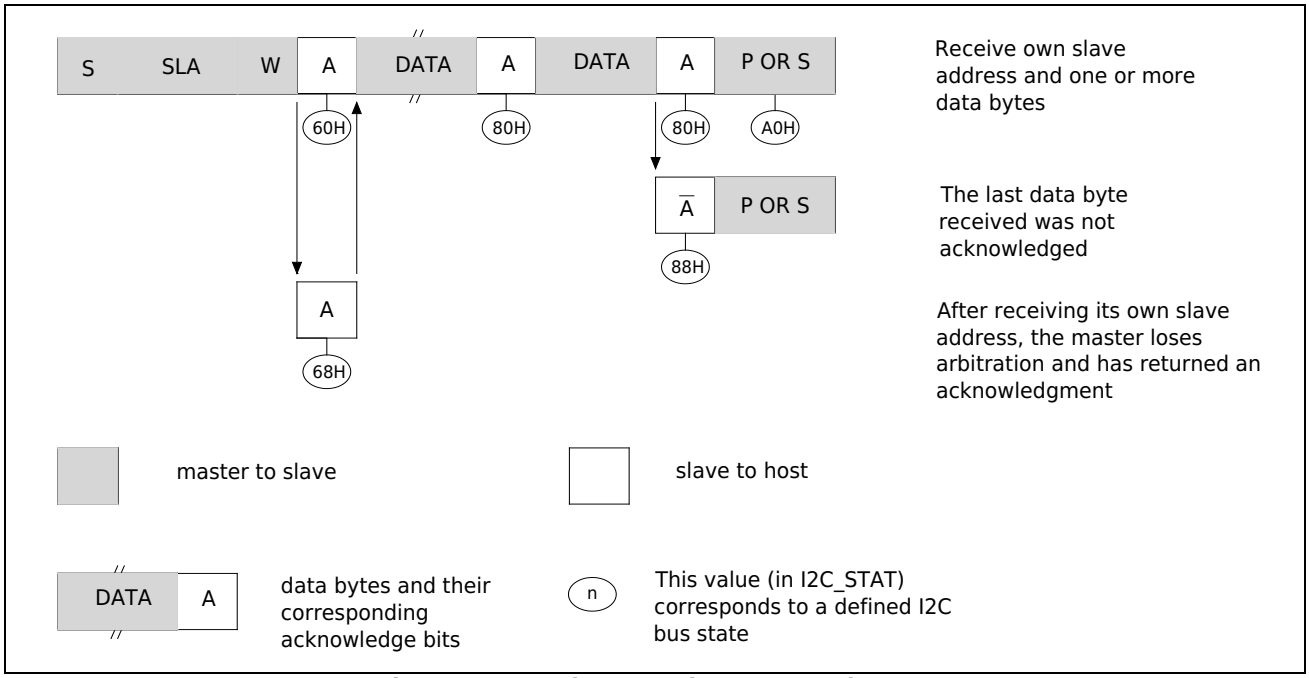


Figure 17-12 Slave Receiver Status Diagram

- 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 host 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.

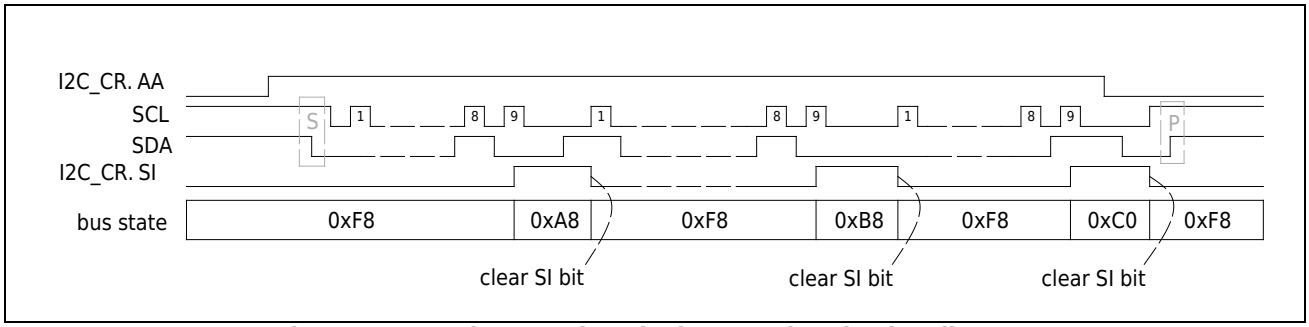


Figure 17-13 Slave send mode data synchronization diagram

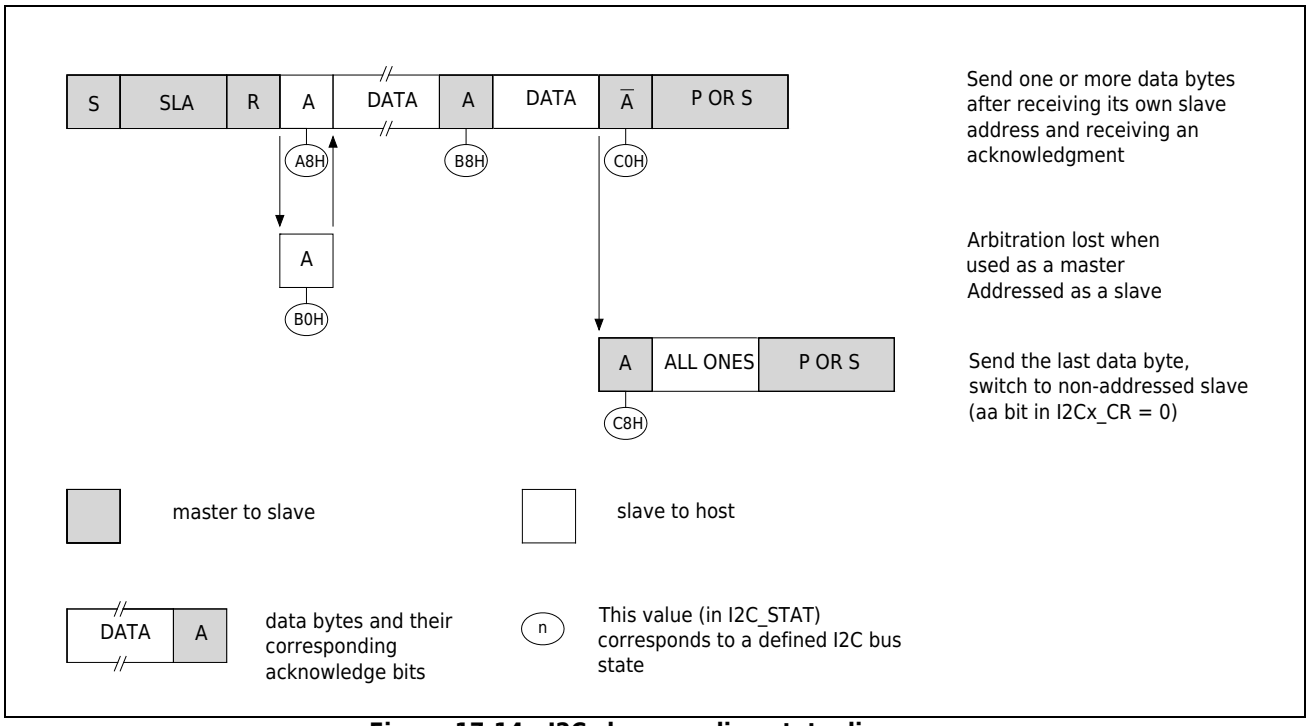


Figure 17-14 I2C slave sending state diagram

- general call mode

  General call mode is a special slave receiving mode, the addressing mode is 0x00, the slave address and read and write are both 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.

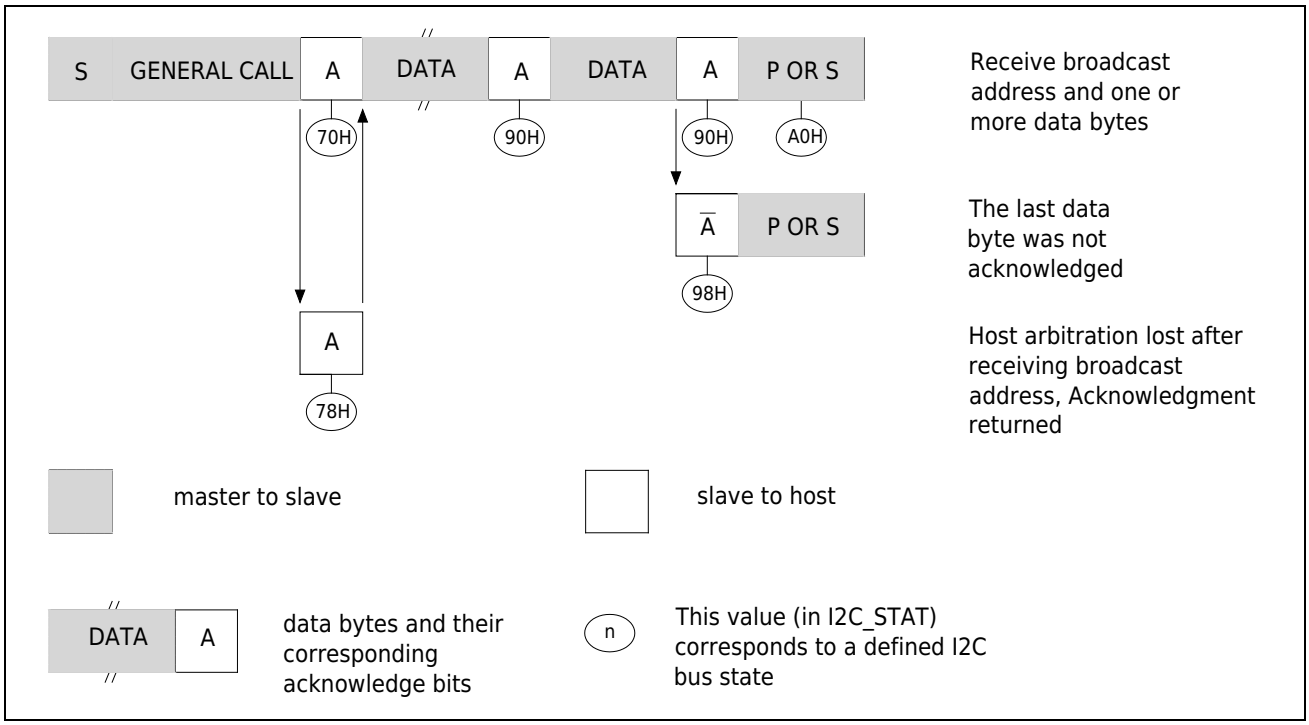


Figure 17-15 I2C General Call State Diagram

17.4.8 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. Bus errors can also occur when external disturbances affect the internal I2C block signals. SI is set when a bus error occurs.

Table 17-2 I2C Status Code Description

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
master receive mode	
08H	Start condition sent
10H	Repeated start condition sent
38H	Arbitration lost in non- 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	When statically addressed, a STOP condition or a repeated START condition is received
Slave send mode	
A8H	Received its own SLA+R and returned ACK

status code	Description
B0H	When the host loses arbitration, has received its own SLA+R, and has returned ACK
B8H	Data sent, ACK received
C0H	Data bytes 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	When statically addressed, a STOP condition or a repeated START condition is received
rest of miscellaneous status	
F8H	No relevant status information available, SI=0
*	A bus error occurs during transmission, or external interference causes I2C to enter an undefined state

17.5programming example

17.5.1 Host sending example

Step1. Map 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 to 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, Start signal has been sent on the bus, set I2C_CR.STA to 0.

Step9. Query I2C_STAT, if the value of this register is 0x08 or 0x10, proceed 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 on 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, data has been sent on 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, 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.

17.5.2 Host receiveexample

Step1. Map 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 to 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, Start signal has been sent on 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 on the bus.

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

Step13. Set I2C_CR.AA to 1 to enable the acknowledge flag.

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

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

Step16. Query I2C_STAT, if the value of this register 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.

17.5.3 Slave receiving example

Step1. Map 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 to 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 acknowledge flag.
Step7. Wait for I2C_CR.SI to become 1, addressed by SLA+W.
Step8. Query I2C_STAT, if the value of this register is 0x60, proceed to the next step, otherwise, perform error handling.
Step9. Set I2C_CR.SI to 0, the master 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 value of the register 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.

17.5.4 Slave sending example

Step1. Map 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 to 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 acknowledge flag.
Step7. Wait for I2C_CR.SI to become 1, addressed by SLA+R.
Step8. Query I2C_STAT, if the value of this register is 0xA8, continue to execute 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, data has been sent on the bus.
Step11. Query I2C_STAT, if the value of this register is 0xB8 or 0xC0, proceed to the next step, otherwise, perform error handling.
Step12. If the data to be sent is not completed, jump to Step9 to continue execution.
Step13. Set I2C_CR.AA to 0 and set I2C_CR.SI to 0.

17.6Register description

17.6.1 register list

Base address: 0x40000400

Table 17-3 Register List

Offset	register name	access	Register description
0x00	I2C_BRREN	RW	I2C Baud Rate Generator Enable Register
0x04	I2C_BRR	RW	I2C Baud Rate Counter Configuration Register
0x08	I2C_CR	RW	I2C configuration register
0x0c	I2C_DATA	RW	I2C data register
0x14	I2C_STAT	RO	I2C state register
0x10	I2C_ADDR0	RW	I2C Slave Address 0 Register
0x20	I2C_ADDR1	RW	I2C Slave Address 1 Register
0x24	I2C_ADDR2	RW	I2C Slave Address 2 Register
0x28	I2C_MATCH	RW	I2C Slave Slave Address Match Register

17.6.2 Baud Rate Generator Enable Register (I2C_BRREN)

Address offset: 0x00

Reset value: 0x0000 0000

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

Bit	Marking	Functional description
31：1	Reserved	reserved bit
0	EN	Baud Rate Generator Enable Control 1: Enable 0: Prohibited

17.6.3 Baud Rate Counter Configuration Register (I2C_BRR)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bit	Marking	Functional description
31:8	Reserved	reserved bit
7:0	BRR	I2C bus SCL baud rate configuration $f_{SCL} = f_{PCLK} / 8 / (BRR+1)$, where BRR >0

17.6.4 Configuration Register (I2C_CR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									EN	STA	STO	SI	AA	Res	FLT
									RW	RW	RW	RW	RW		RW

Bit	Marking	Functional description
31:7	Reserved	reserved bit
6	EN	Module Enable Control 1: Enable 0: Prohibited
5	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.
4	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.
3	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
2	AA	response control 1: Send ACK during the answer phase 0: Send NAK during the answer phase
1	Reserved	
0	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.

17.6.5 Data register (I2C_DATA)

Address offset: 0x0c

Reset value: 0x00000000

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

Bit	Marking	Functional description
31:8	Reserved	reserved bit
7:0	DR	Data register In send mode, write the data to be sent In receive mode, read out received data

17.6.6 Status Register (I2C_STAT)

Address offset: 0x14

Reset value: 0x00000000

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

Bit	Marking	Functional description
31:8	Reserved	reserved bit
7:0	STAT	I2C status register, the specific definition of the status value is detailed in the [Status code expression] chapter

17.6.7 Slave Address 0 Register (I2C_ADDR0)

Address offset: 0x10

Reset value: 0x00000000

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

Bit	Marking	Functional description
31:8	Reserved	reserved bit
7:1	Addr	I2C slave mode address 0
0	GC	Broadcast Address Acknowledgment Enable 1: Enable 0: Prohibited

17.6.8 Slave Address 1 Register (I2C_ADDR1)

Address offset: 0x20

Reset value: 0x00000000

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

Bit	Marking	Functional description
31:8	Reserved	reserved bit
7:1	Addr	I2C slave mode address 1.
0	Reserved	reserved bit

17.6.9 Slave Address 2 Register (I2C_ADDR2)

Address offset: 0x24

Reset value: 0x00000000

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

Bit	Marking	Functional description
31:8	Reserved	reserved bit
7:1	Addr	I2C slave mode address 2.
0	Reserved	reserved bit

17.6.10 Slave Address Match Register (I2C_MATCH)

Address offset: 0x28

Reset value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													AD2F	AD1F	AD0F
													RO	RO	RO

Bit	Marking	Functional description
31:3	Reserved	reserved bit
2	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
1	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
0	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

Note: The address match flag will be cleared in the following three situations:

- When the module is reset
- When START/STOP is sent
- When the internal STOP of the slave

18 Analog-to-digital converter (ADC)

18.1 Module Introduction

External analog signals need to be converted into digital signals to be further processed by the MCU. A 10 -bit successive approximation analog-to-digital converter (SAR ADC) module with high precision and high conversion rate is integrated inside. Has the following properties:

- 10-bit conversion accuracy;
- 1M SPS conversion speed;
- Support single conversion and continuous conversion;
- 2 reference sources: AVCC voltage, ExRef pin;
- 15 input channels, including 14 external pin inputs and 1 built- in BGR 0.9V voltage;

Note: Select AVCC as the reference source, measure BGR 0.9V; AVCC voltage can be calculated.

- ADC voltage input range: 0~Vref;
- Support on-chip peripherals to automatically trigger ADC conversion, effectively reducing chip power consumption and improving real-time conversion.

18.2 ADC block diagram

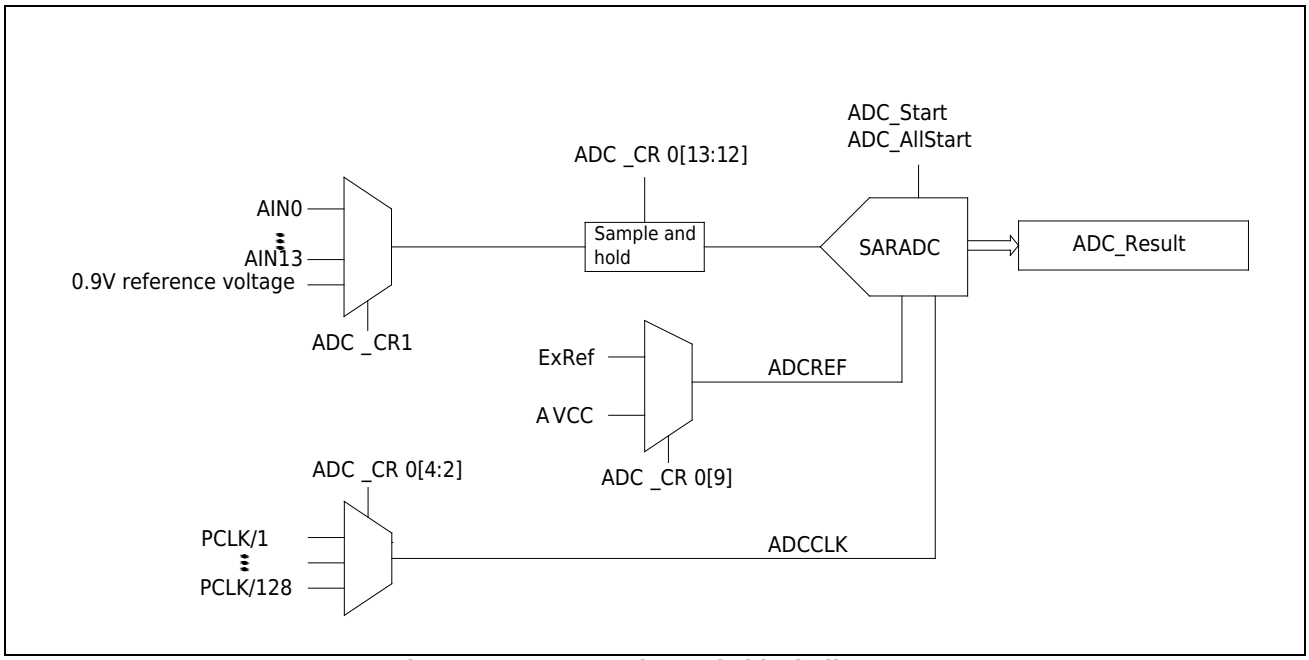


Figure 18-1 ADC schematic block diagram

18.3Conversion Timing and Conversion Speed

The ADC conversion timing is shown in the figure below: a complete ADC conversion consists of a sampling process and a successive comparison process. Among them, the sampling process requires 6~12 ADCCLKs, which are configured by ADC_CR0.SAM; the successive comparison process requires 13 ADCCLKs. Therefore, an ADC conversion requires a total of 19 ~25 ADCCLKs.

The unit of ADC conversion speed is SPS, that is, how many ADC conversions are performed per second. The calculation method of the ADC conversion speed is: the frequency of ADCCLK / the number of ADCCLKs required for one ADC conversion.

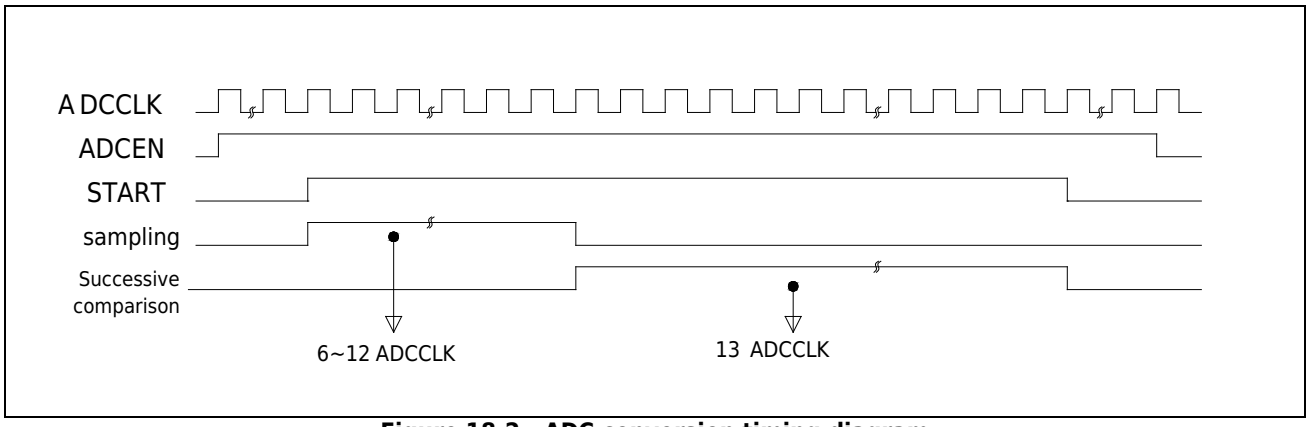


Figure 18-2 ADC conversion timing diagram

The ADC conversion speed is related to the ADC reference voltage and AVCC voltage. The maximum conversion speed is shown in the following table:

ADC reference voltage	AVCC voltage	Maximum conversion speed	Maximum ADCCLK frequency
AVCC / ExRef	1.8V~2.4V	200K SPS	4MHz
AVCC / ExRef	2.4V~2.7V	500K SPS	16MHz
AVCC / ExRef	2.7V~5.5V	1M SPS	24MHz

18.4single conversion mode

In single-conversion mode, only one conversion of the set channel is performed after the ADC starts. This mode can be started by setting the ADC_start.Start bit or by setting the external trigger of ADC_ExtTrigger. After the ADC conversion of each channel is completed, the ADC_IFR.EOC bit will be automatically set to 1, and the result will be saved in the ADC_Result register.

ADC single conversion operation process:

Step1. Configure the corresponding bit of GPIOx_ADS to configure the ADC channel to be converted as an analog port.

Step2. Set GPIOC_ADS.bit3 to 1 to configure the ADC external reference voltage pin 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_CR0.En to 1 to enable the ADC module.

Step4. Configure ADC_CR0.Ref to select the reference voltage of ADC.

Step5. Configure ADC_CR0.SAM and ADC_CR0.CkDiv to set the conversion speed of ADC.

Step6. Delay 5us, wait for the ADC module to start up.

Step7. Configure ADC_CR1.CHSEL to select the channel to be converted.

Step8. Set ADC_ICR to 0, clear ADC_IFR flag.

Step9. Set ADC_Start.Start to 1 to start ADC single conversion.

Step10. Wait for ADC_IFR.EOC to become 1, read the ADC_Result register to get the ADC conversion result.

Step11. To convert other channels, repeat Step7~Step10.

Step12. Set ADC_CR0.En to turn off the ADC module.

18.5External triggerconversion mode

ADC conversions can be initiated either by software configuration or by an external trigger.

Configure the ADC_ExtTrigger register to set the external trigger source for ADC conversion.

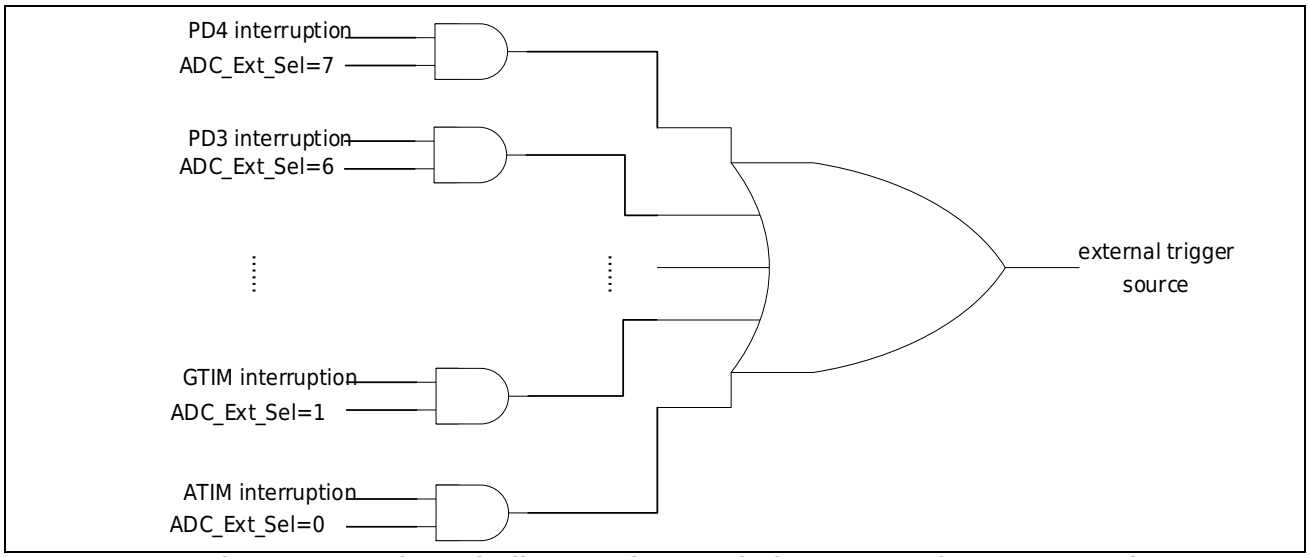


Figure 18-3 Schematic diagram of external trigger source for ADC conversion

18.6Continuous conversion mode

When ADC_AllStart.Start is set to 1, the ADC module works in the continuous conversion mode; when ADC_AllStart.Start is set to 0, the ADC module exits the continuous conversion mode.

In the continuous conversion mode, the ADC module will continue to perform ADC conversion, and the ADC_IFR.EOC bit will be automatically set to 1 after each ADC conversion is completed, and the result will be saved in the ADC_Result register. After the first conversion is completed, the user

program can read the ADC_Result register at any time to obtain the result of the ADC conversion at the current moment; every time ADC_Result is read, the ADC_IFR.EOC flag will be cleared.

ADC continuous conversion operation process:

Step1. Configure the corresponding bit of GPIOx_ADS to configure the ADC channel to be converted as an analog port.

Step2. Set GPIOC_ADS.bit3 to 1 to configure the ADC external reference voltage pin 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_CR0.En to 1 to enable the ADC module.

Step4. Configure ADC_CR0.Ref to select the reference voltage of ADC.

Step5. Configure ADC_CR0.SAM and ADC_CR0.CkDiv to set the conversion speed of ADC.

Step6. Delay 5us, wait for the ADC module to start up.

Step7. Configure ADC_CR1.CHSEL to select the channel to be converted.

Step8. Set ADC_AllStart.Start to 1 to start ADcontinuous conversion.

Step9. After the first ADC conversion is completed, the ADC_Result register can be read at any time to obtain the real-time conversion result of the ADC.

Step10. Set ADC_AllStart.Start to 0 to exit continuous conversion mode.

Step11. Set ADC_CR0.En to turn off the ADC module.

18.7Measure AVCC voltage

Set the reference source of the ADC to AVCC, set the input channel of the ADC to the internal BGR 0.9V, and calculate the voltage of AVCC through the ADC conversion result. Note that the conversion speed of the ADC under this condition should not exceed 200K SPS. Under this condition, the calculation formula of AVCC voltage is: $V_{AVCC} = \frac{V_{BGR} \times 1023}{ADC\_Result}$.

Where V_{BGR} is the internal reference voltage, its typical value is 0.9V.

Calculation example:

When ADC_Result is 279, the voltage of AVCC is $\frac{0.9 \times 1023}{279} = 3.3V$.

When ADC_Result is 460, the voltage of AVCC is $\frac{0.9 \times 1023}{460} = 2.0V$.

18.8ADC Interrupt

The ADC interrupt request is shown in the table below:

Interrupt source	Interrupt logo	interrupt enable
ADC single conversion completed	ADC_IFR.EOC	ADC_IER.EOC

18.9ADC module registers

Base Address 0x40002400

Table 18-1 ADC register

Register	Offset address	Description
ADC_CR0	0x004	ADC Configuration Register 0
ADC_CR1	0x008	ADC Configuration Register 1
ADC_Result	0x0A0	ADC conversion result register
ADC_IFR	0x0B0	ADC Interrupt Flag Register
ADC_IER	0x0B4	ADC Interrupt Enable Register
ADC_ICR	0x0B8	ADC Interrupt Clear Register
ADC_ExtTrigger	0x0BC	ADC conversion external interrupt trigger source configuration register
ADC_Start	0x0C0	ADC conversion start control register
ADC_AllStart	0x0CC	ADC continuous conversion start control register

18.9.1 ADC Configuration Register 0 (ADC_CR0)

Offset address 0x004

Reset value 0x00002600

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		SAM		Res.	Ref		Res.				CkDiv			Res.	En
		RW			RW						RW				RW

Bit	Marking	Functional description
31:14	Reserved	Keep
13:12	SAM	ADC sampling cycle selection (an additional 6~12 ADC clock cycles are required for sampling) 00: 6 ADC clock cycles 01: 8 ADC clock cycles 10: 11 ADC clock cycles 11: 12 ADC clock cycles
11	Reserved	Keep
10:9	Ref	ADC reference voltage selection x0: external reference voltage ExRef (PC03) x1: AVCC voltage
8:5	Reserved	Keep
4:2	CkDiv	ADC clock selection 000: PCLK clock 001: PCLK clock divided by 2 010: PCLK clock divided by 4 011: PCLK clock divided by 8 100: PCLK clock divided by 16 101: PCLK clock divided by 32 110: PCLK clock divided by 64 111: PCLK clock divided by 128
1	Reserved	Keep
0	En	ADC enable control 1: enable ADC 0: disable ADC

18.9.2 ADC Configuration Register 1 (ADC_CR1)

Offset address 0x008

Reset value 0x00000000

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

Bit	Marking	Functional description
31:4	Reserved	Keep
3:0	CHSEL	Channel selection 0000: channel 0 pin PC04 0001: Channel 1 pin PC06 0010: Channel 2 pin PD02 0011: Channel 3 pin PD03 0100: Channel 4 pin PD04 0101: Channel 5 pin PD05 0110: Channel 6 pin PD06 0111: Channel 7 pin PA01 1000: Channel 8 pin PA02 1001: Channel 9 pin PB05 1010: Channel 10 pin PB04 1011: Channel 11 pin PB02 1100: Channel 12 pin PB01 1101: Channel 13 pin PB00 1110: Channel 14 internal 0.9V reference voltage

18.9.3 ADC conversion result register (ADC_Result)

Offset address 0x0A0

Reset value 0x00000000

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

Bit	Marking	Functional description
31:16	Reserved	Keep
15:0	Result	ADC conversion result Note: Reading ADC_Result can clear ADC_IFR.EOC

18.9.4 ADC Interrupt Flag Register (ADC_IFR)

Offset address 0x0B0

Reset value 0x00000000

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

Bit	Marking	Functional description
31:1	Reserved	Keep
0	EOC	ADC conversion complete flag 1: ADC completes one conversion 0: ADC has not completed one conversion

18.9.5 ADC Interrupt Enable Register (ADC_IER)

Offset address 0x0B4

Reset value 0x00000000

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

Bit	Marking	Functional description
31:1	Reserved	Keep
0	EOC	ADC one conversion complete interrupt enable 1: Enable 0: Prohibited

18.9.6 ADC Flag Clear Register (ADC_ICR)

Offset address 0x0B8

Reset value 0x00000001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															EOC
															R1W0

Bit	Marking	Functional description
31:1	Reserved	Keep
0	EOC	Write 0 to clear the ADC single conversion complete flag Write 1 has no effect

18.9.7 ADC External Trigger Configuration Register (ADC_ExtTrigger)

Offset address 0x0BC

Reset value 0x00000000

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

Bit	Marking	Functional description
31:4	Reserved	Keep
3:0	TRIGSEL	ADC trigger source selection 0000: ATIM3 output trigger signal, see ATIM3_ADTR for details 0001: GTIM overflow signal 0010: PB03 interrupt flag changes from 0 to 1 signal 0011: Signal when the PB04 interrupt flag changes from 0 to 1 0100: Signal when PC03 interrupt flag changes from 0 to 1 0101: Signal when PC04 interrupt flag changes from 0 to 1 0110: Signal when PD03 interrupt flag changes from 0 to 1 0111: Signal when PD04 interrupt flag changes from 0 to 1 Note: When the selected trigger source is GPIO, the user program should clear the interrupt flag of the trigger source in time.

18.9.8 ADC Conversion Start Control Register (ADC_Start)

Offset address 0x0C0

Reset value 0x00000000

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

Bit	Marking	Functional description
31:1	Reserved	Keep
0	Start	ADC conversion control 1: Start ADC conversion, and automatically clear to zero after a conversion is completed. 0: no action

18.9.9 ADC Continuous Conversion Control Register (ADC_AllStart)

Offset address 0x0CC

Reset value 0x00000000

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

Bit	Marking	Functional description
31:1	Reserved	Keep
0	Start	ADC continuous conversion control 1: start ADC continuous conversion 0: Stop ADC continuous conversion, need to wait for the end of the current conversion

19 Low voltage detector (LVD)

19.1 Introduction to LVD

LVD can be used to monitor the voltage of VCC and 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. The module can work normally in Active, Sleep and DeepSleep modes.

LVD has the following characteristics:

- 4 monitoring sources, AVCC, PA03, PC03, PD04
- 16 -step threshold voltage, 1.8V~3.3V 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.

19.2 LVD block diagram

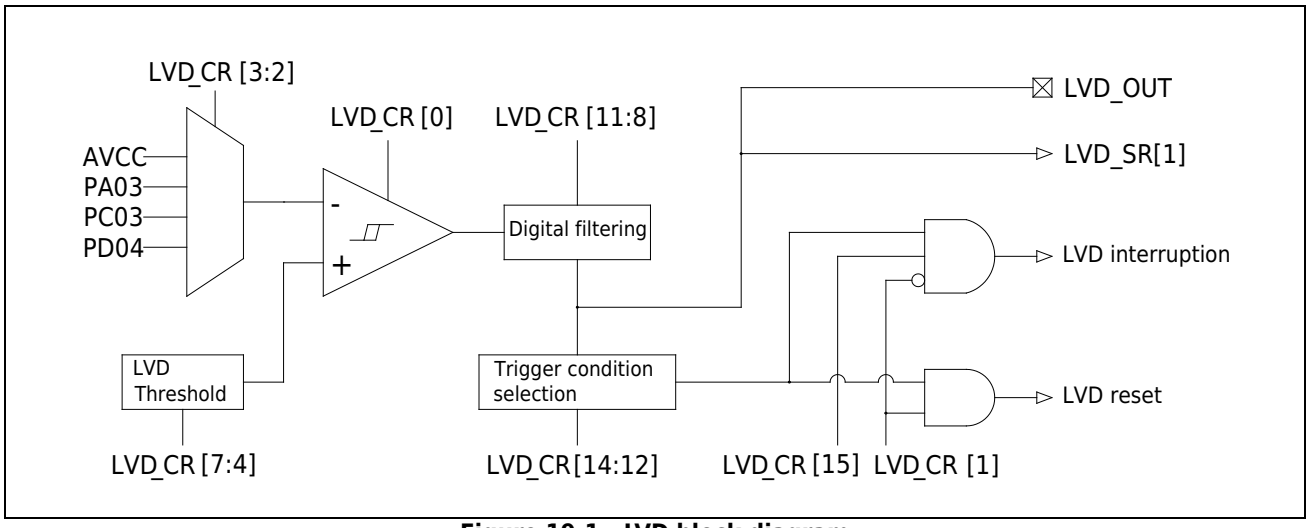


Figure 19-1 LVD block diagram

19.3hysteresis function

The built-in voltage comparator of LVD has a hysteresis function, which can enhance the anti-interference ability of the chip. When LVD is disabled, the output signal of LVD is at low level; when LVD is enabled, the output signal of LVD will not flip until the input signal is higher or lower than the threshold voltage 20mV, and the input and output signals are as follows:

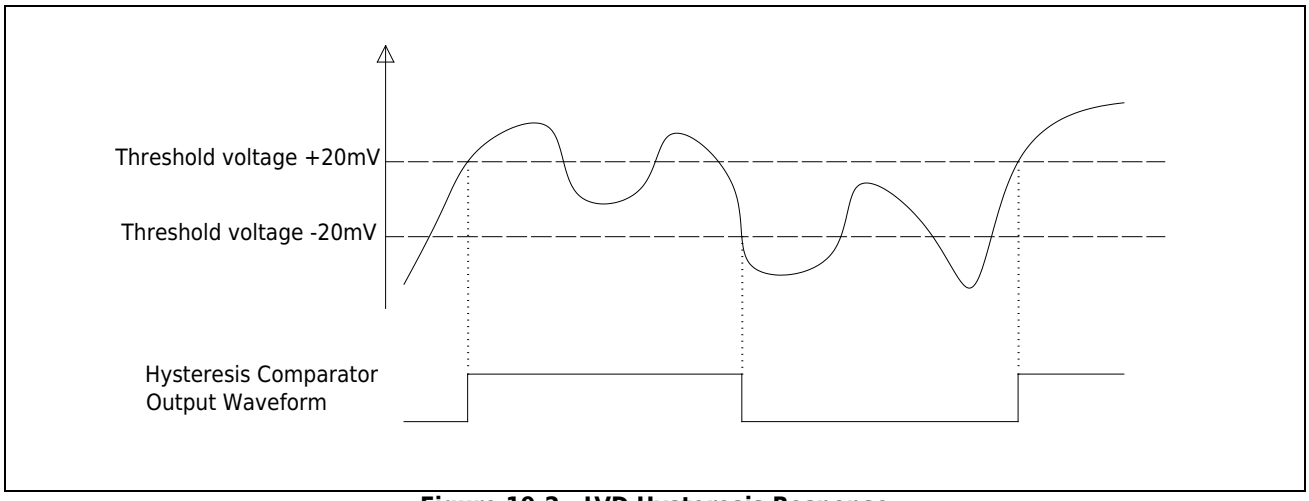


Figure 19-2 LVD Hysteresis Response

19.4Digital filtering

If the working environment of the chip is bad, the output of the hysteresis comparator will appear noise signal. When the digital filtering module is enabled, 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. Enable the digital filtering module, and the filtering diagram is as follows:

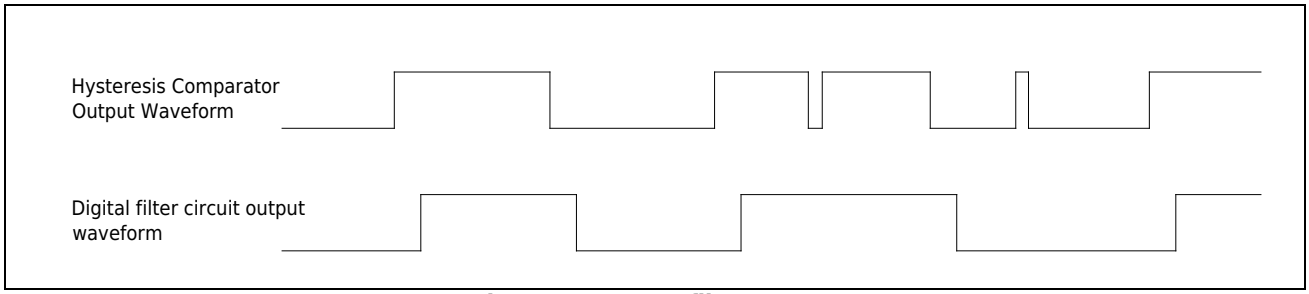


Figure 19-3 LVD filter output

19.5Configuration example

19.5.1 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. As described in the GPIO chapter, configure the pins of the voltage sources to be monitored as analog functions.

Note: If the voltage source to be monitored is AVCC, ignore this step.

Step2. Configure LVD_CR.Source to select the voltage source to be monitored.

Step3. Configure LVD_CR.VTDS to select the threshold voltage of LVD.

Step4. Configure LVD_CR.FltTime to select the LVD filter time.

Step5. Configure LVD_CR.FltEN to enable LVD filtering.

Step6. Set LVD_CR.HTEN to 1, select high level to trigger LVD action.

Step7. Set LVD_CR.ACT to 1, select LVD action as reset.

Step8. Set LVD_CR.EN to 1 to enable LVD.

Note: When the user program is initialized, read the reset flag register to know whether the reset source is LVD.

19.5.2 LVD configured as voltage change interrupt

In this mode, an interrupt is generated when a change in voltage is detected relative to a threshold point. That is, an interrupt is generated when the voltage rises from a point below the threshold to a point above the threshold or falls from a point above the threshold to a point below the threshold.

The configuration method is as follows:

Step1. As described in the GPIO chapter, configure the pins of the voltage sources to be monitored as analog functions.
Note: If the voltage source to be monitored is AVCC, ignore this step.

Step2. Configure LVD_CR.Source to select the voltage source to be monitored.

Step3. Configure LVD_CR.VTDS to select the threshold voltage of LVD.

Step4. Configure LVD_CR.FiltTime to select the LVD filter time.

Step5. Configure LVD_CR.FltEn to enable LVD filtering.

Step6. Set LVD_CR.RTEN and LVD_CR.FTEN to 1 to select level change to trigger LVD action.

Step7. Set LVD_CR.ACT to 0, select LVD action as interrupt.

Step8. Set LVD_CR.IE to 1 to enable LVD Interrupt.

Step9. Enable LVD interrupt in NVIC interrupt vector table.

Step10. Set LVD_CR.EN to 1 to enable LVD.

Step11. Execute the operations required by the user in the LVD interrupt service routine; write 0x00 to LVD_IFR before exiting the interrupt service routine to clear the interrupt flag.

19.6LVD register

Base Address 0x40002680

Table 19-1 LVD Register

Register	Offset address	Description
LVD_CR	0x000	LVD configuration register
LVD_SR	0x004	LVD Interrupt Flag Register

19.6.1 LVD Configuration Register (LVD_CR)

Offset address 0x000

Reset value 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IE	HTEN	RTEN	FTEN	FltTime			FltEn	VTDS				Source		ACT	EN
RW	RW	RW	RW	RW			RW	RW				RW		RW	RW

Bit	Marking	Functional description
31:16	Reserved	Keep
15	IE	LVD interrupt enable 1: Enable 0: Prohibited
14	HTEN	High level trigger enable (the monitored voltage is lower than the threshold voltage) 1: Enable 0: Prohibited
13	RTEN	Rising edge trigger enable (the monitored voltage changes from higher than the threshold voltage to lower than the threshold voltage) 1: Enable 0: Prohibited
12	FTEN	Falling edge trigger enable (the monitored voltage changes from lower than the threshold voltage to higher than the threshold voltage) 1: Enable 0: Prohibited
11:9	FltTime	Digital filter time configuration (filter clock is RCL clock) 111: filter width is less than 4095 clock cycles 110: filter width is less than 1023 clock cycles 101: filter width is less than 255 clock cycles 100: filter width is less than 63 clock cycles 011: filter width is less than 15 clock cycles 010: filter width is less than 7 clock cycles 001: filter width is less than 3 clock cycles 000: filter width is less than 1 clock cycles Note: The filter time is only valid when FLTEN is 1.
8	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.
7:4	VTDS	LVD threshold voltage selection 1111: 3.3v 1110: 3.2v 1101: 3.1v 1100: 3.0v 1011: 2.9v 1010: 2.8v 1001: 2.7v 1000: 2.6v 0111: 2.5v 0110: 2.4v 0101: 2.3v 0100: 2.2v 0011: 2.1v 0010: 2.0v 0001: 1.9v 0000: 1.8v
3:2	Source	LVD monitoring source selection 11: PD04 port input voltage 10: PC03 port input voltage 01: PA03 port input voltage 00: AVCC supply voltage
1	ACT	LVD trigger action selection 1: System reset 0: NVIC interrupt

0	EN	LVD enable control 1: enable LVD 0: disable LVD

19.6.2 LVD Interrupt Register (LVD_SR)

Offset address 0x004

Reset value 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														FLTV	INTF
														RO	RW0

Bit	Marking	Functional description
31:2	Reserved	Keep
1	FLTV	Level value of digital filter output 1: digital filter output high level 0: Digital filter output low level
0	INTF	LVD interrupt flag 1: LVD interrupt flag has been generated, write 0 to clear 0: LVD interrupt flag not generated

20 Device Electronic Signature

The electronic signature is stored in the system storage area of the flash memory (Flash), and contains the chip identification information. The electronic signature can be read by user firmware or external devices to match different configurations of HC32Fxxx / HC32Lxxx microcontrollers.

20.1 Product Unique Identifier (UID) Register

Typical application scenarios for unique ID codes:

- used as a serial number
- Use it as a key or cryptographic primitive to increase code security when programming internal flash memory
- Activation of the secure bootstrap process, etc.

The 80 -bit unique device identifier provides a reference number that is unique to any device and any context. The user can never change these bits. The 80 -bit unique device identifier can also be read in different ways such as single byte / halfword / word, and then concatenated using a custom algorithm.

Base address: 0x0010 1080

Offset address	Functional description	bit field	Remark
0	Y coordinate	UID[07:00]	0 – 127
1	X coordinate	UID[15:08]	0 – 127
2	Wafer serial number	UID[23:16]	1 - 25
3	LOT number 0	UID[31:24]	
4	LOT number 1	UID[39:32]	
5	LOT number 2	UID[47:40]	
6	RFU	UID[55:48]	
7	RFU	UID[63:56]	
8	RFU	UID[71:64]	
9	RFU	UID[79:72]	

20.2 Product Model Register

0x0010 0D90 ~ 0x0010 0DAF stores the ASCII code of the product model. If the product model is less than 32 bytes, fill it with 0x00.

Example: 484333324630303243345042 2D5453534F50323000000000000000000000000000 represents the product model HC32F002C4PB-TSSOP20.

20.3FLASH capacity register

Base address: 0x0010 0DB0

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

Bit	Marking	Functional description
31:0	FlashSize	The capacity of the product's built-in Flash, in bytes Example: 0x00004800 means the Flash capacity is 18K Byte

20.4RAM capacity register

Base address: 0x0010 0DB4

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

Bit	Marking	Functional description
31:0	RamSize	The capacity of the product's built-in RAM, in bytes Example: 0x00000800 means the RAM capacity is 2K Byte

20.5Pin Count Register

Base address: 0x0010 0DBA

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

Bit	Marking	Functional description
15:0	Pin Count	The number of product pins, in unit Example: 0x0014 means the number of product pins is 20

20.6One-time programming area (OTP area)

There is a 128 -byte OTP storage area in the chip, and its address range is 0x00100F00～0x00100F7F. This data area can only write data through the ISP command, and it can only be written once. Neither chip erase nor page erase can erase the data in this area. The user program can read the data in this area through the read operation.

Based on the characteristics of the OTP area, the user program can store unchangeable information in the OTP area.

21 SWD debug interface

The HC32F002 series uses the ARM Cortex-M0+ core, which has a hardware debug module SWD that supports complex debugging operations. The hardware debug module allows the core to stop when fetching instructions (instruction breakpoints) or accessing data (data breakpoints). When the kernel is stopped, both the internal state of the kernel and the external state of the system can be queried in the IDE. After the query is complete, the core and peripherals can be restored and program execution continues. When the HC32F002 microcontroller is connected to the debugger and starts debugging, the debugger will use the kernel's hardware debugging module for debugging.

Note:

– SWD can't work in DeepSleep mode, please debug in Active and Sleep mode.

21.1 SWD debugging additional functions

This product uses ARM Cortex-M0+ CPU, the core contains hardware extensions for advanced debugging functions, so the debugging functions of this product are consistent with Cortex-M0+. Debug extensions allow the kernel to halt the kernel when fetching instructions (instruction breakpoints) or fetching 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.

When the debug host is connected to the MCU and debugged, the debug function will be used.

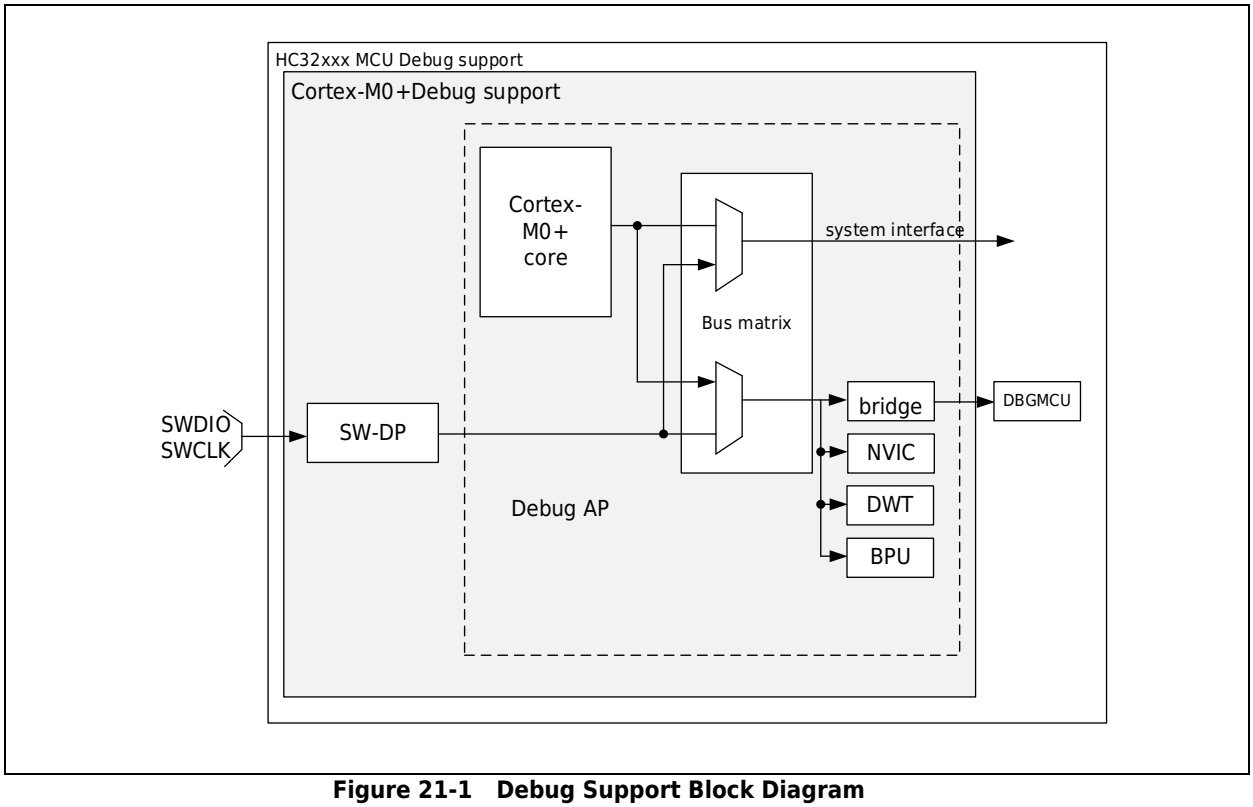


Figure 21-1 Debug Support Block Diagram

Debug features built into the Cortex®-M0+ core are part of the ARM® CoreSight Design Suite.

The ARM® Cortex®-M0+ core provides integrated on-chip debugging support. It includes:

- SW-DP: serial line
- BPU: Breakpoint unit
- DWT: data watchpoint trigger

Note:

– Details on the debug features supported by the ARM® Cortex®-M0+ core can be found in the Cortex®-M0+ Technical Reference Manual.

21.2ARM® Reference Documentation

- Cortex®-M0+ Technical Reference Manual (TRM)
  Available from developer.arm.com.
- ARM® Debug Interface V5
- ARM® CoreSight Design Suite Version r1p1 Technical Reference Manual

21.3Debug port pins

21.3.1 SWD port pins

HC32F002The SWD series requires 2 pins, as shown in the table below.

SWD port name	Debug function	pin assignment
SWCLK	Serial clock	PD01
SWDIO	Serial data input/output	PC07

21.3.2 SW-DP pin assignment

If it is enabled when burning the code [2-3 level encryption chip] option SWD debugging function will be disabled after power-on. If there is no enable or [level 1 encryption chip] option when burning the program, after power on PC07/PD01 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 debug function of the SWD pin, and the SWD pin will be released for normal use GPIO. The configuration and functions of SWD pins are summarized in the following table:

[Encryption chip] option	SWDIO configuration	PC07/PD01 function
no encryption	0	SWD
no encryption	1	GPIO
1 encryption	0	SWD (for decryption only)
1 encryption	1	GPIO
2-3 Encryption	0	NA
2-3 Encryption	1	GPIO

21.3.3 SWD pin

After user software releases SWDIO, GPIO controller takes control of these pins. The reset state of the GPIO control register will set the I/O into the equivalent state:

- SWDIO: input pull-up
- SWCLK: input pull-up

No external resistors need to be added due to built-in pull-up resistors.

21.4SWD port

21.4.1 Introduction to SWD Agreement

This synchronous serial protocol uses two pins:

- SWCLK: Clock from host to target
- SWDIO: Bidirectional

Using this protocol, two sets of register sets (DPACC register set and APACC register set) can be read and written simultaneously. LSB is ahead when transferring data.

For SWDIO bidirectional management, the line must be pulled up on the board (ARM® recommends 100 K). These pull-up resistors are internally configurable. No external pull-up resistors are required.

Every time the SWDIO direction is changed in the protocol, the conversion time is inserted, and the line is not driven by the host or the target. By default, this conversion time is one bit, but can be adjusted by configuring the SWCLK frequency.

21.4.2 SWD protocol sequence

Each sequence consists of three phases:

1. The host sends a packet request (8 bits)

2. Acknowledgment response sent by target (3 digits)

3. The host or target sends the data transfer phase (33 bits)

Bit	Name	Note
0	start up	must be 1
1	APnDP	0: DP access; 1: AP access
2	RnW	0: write request; 1: read request
4:3	A[3:2]	Address field of DP or AP register
5	Parity	Unit parity for the first few bits
6	Stop	0
7	reside	Not driven by the host. Must be read as 1 by target due to pull-up

For a detailed description of the DPACC and APACC registers, refer to the Cortex®-M0+ TRM.

The packet request is always followed by a conversion time (1 bit by default), at which point neither the host nor the target will drive.

Bit	Name	Note
2:0	ACK	001: FAULT 010: WAIT 100: OK

The ACK response must be followed by transition time only when a READ transaction occurs or a WAIT or FAULT acknowledgment is received.

Bit	Name	Note
0:31	WDATA or RDATA	write or read data
32	Parity	Single parity for 32 data bits

The transfer time must be after the DATA transfer only when a READ transaction occurs.

21.4.3 SW-DP state machine (reset, idle state, ID code)

The state machine of SW -DP has an internal ID code for identifying SW-DP. The code is JEP-106 compliant. This ID code is the default ARM® code, set to **0x0BB11477** (equivalent to Cortex®-M0+ equivalent).

Note:

- The SW-DP state machine is inactive until the target reads the ID code.

- The SW-DP state machine is in reset state after power-on reset or after the line has been at high level for more than 50 cycles.
- If the line is low for at least two cycles after the reset state, the SW-DP state machine is in the idle state.
- After the reset state, the state machine must first enter the idle state and then perform a read access to the DP-SW ID CODE register. Otherwise, the target will issue a FAULT acknowledgment response on another transaction.

See Cortex®-M0+ TRM and CoreSight Design Suite r1p0TRM for more details on the SW-DP state machine.

21.4.4 DP and AP read / write access

- Read access to DP is not delayed: target response can be sent immediately (if ACK=OK) or delayed (if ACK=WAIT).
- Delay read access to AP. This means that access results will be returned on the next transfer. If the next access to be performed is not an AP access, the DP-RDBUFF register must be read to obtain the result.
- Every time an AP read access or RDBUFF read request is made, the READOK flag of the DP-CTRL/STAT register will be updated to know whether the AP read access is successful.

- SW-DP has a write buffer (for DP or AP writes) so that write operations can be accepted even while other operations are still pending. If the write buffer is full, the target acknowledgment response is WAIT. Except for IDCODE read, CTRL /STAT read or ABORT write, these operations will also be accepted when the write buffer is full.
- Since there are asynchronous clock domains SWCLK and HCLK, two additional SWCLKs are required after the write operation (after the parity bit) cycles These cycles should be applied while the line is being driven low (idle state).

This is especially important when writing to the CTRL/STAT register to initiate a power-on request. Otherwise the next operation (one that is only valid after the core is powered on) will be executed immediately, which will cause a failure.

21.4.5 SW-DP register

These registers can be accessed when APnDP=0:

A[3:2]	RW	CTRLSEL bit of the SELECT register	Register	note
00	read		IDCODE	Manufacturer code set to default ARM® for Cortex®-M0+ code. **0x0BB11477** (identifies SW-DP)
00	Write		ABORT	
01	read / write	0	DP-CTRL/STAT	Purpose: – request system or debug power on – configures the transmission operation for AP access – controls compare and verify operations – to read some status flags (overflow and power-on acknowledgment)
01	read / write	1	WIRE CONTROL	Used to configure the physical serial port protocol (such as the duration of the transition time)
10	read		READ RESEND	Allows recovery of read data without repeating the original AP transfer.
10	Write		SELECT	4 -word register window for selecting the currently accessed port and activity
11	read / write		READ BUFFER	This read buffer is useful since the AP access has already been issued (providing the result of the read AP request when the next AP transaction is executed). This read buffer captures the data in the AP and presents it as the result of a previous read without initiating a new operation.

21.4.6 SW-AP Register

These registers can be accessed when APnDP=1:

There are multiple AP registers, which are addressed in the following combinations:

- Shift value A[3:2]
- Current value of DP SELECT register

Address	A[3:2] value	Note
0x0	00	Reserved, must remain at reset value.
0x4	01	DP CTRL/STAT register. Used for: – request system or debug power on – configures the transmission operation for AP access – controls compare and verify operations – to read some status flags (overflow and power-on acknowledgment)
0x8	10	DP SELECT Register: Used to select the currently accessed port and active 4 -word register window. – bits 31:24: APSEL: select the current AP (select the current AP) – bits 23:8: Reserved – bits 7:4: APBANKSEL: Select the active 4 -word register window on the current AP – bits 3:0: Reserved
0xC	11	DP RDBUFF register: used to obtain the final result after performing a series of operations through the debugger (No need to request new JTAG-DP operations)

21.5kernel debugging

Debug the kernel through the kernel debug registers. Debug access to these registers is through the debug access port. It consists of four registers:

Register	Note
DHCSR	32 -bit debug stop control and status register This register provides information about the state of the processor, enables the core to enter a debug stop state, and provides processor stepping capabilities.
DCRSR	17 -bit debug core register selector registers: This register selects the processor register to be read or written.
DCRDR	32 -bit debug core register data register: This register holds the data read and written between the register and the processor selected by the DCRSR (selector) register. data.
DEMCR	32 -bit Debug Exception and Watchdog Control Registers: This register provides vector capture and debug monitor control.

These registers are not reset on system reset. They can only be reset by a power-on reset. See Cortex®-M0+ TRM for more details.

In order to put the core into the debug stop state immediately after reset, it is necessary to:

- Enable Debug and Exception Monitoring Control Register Bit 0 (VC_CORRESET)
- Enable Debug Halt Control and Status Register Bit 0 (C_DEBUGEN)

21.6BPU (Breakpoint unit)

The Cortex®-M0+ BPU implementation provides four breakpoint registers.

21.6.1 BPU function

Processor breakpoint implements PC -based breakpoint functionality.

See the ARMv6-M ARM® and ARM® CoreSight Components Technical Reference Manual for more information on the BPU CoreSight Identification Registers, their addresses and access types.

21.7DWT (Data Watchpoint)

The Cortex®-M0+DWT implementation provides a watchpoint register set.

21.7.1 DWT function

Processor watchpoints implement data address and PC -based watchpoint functionality (i.e. PC sampling registers) and support for comparator address masks as described in ARMv6-M ARM®.

21.7.2 DWT Program Counter Sample Register

Processors implementing the Data Watchpoint Unit also implement the ARMv6-M optional DWT Program Counter Sampling Register (DWT_PCSR). This register allows the debugger to periodically sample the PC without stopping the processor. This provides a rough analysis. See ARMv6-M ARM® for more information.

Cortex®-M0+ DWT_PCSR records passing condition codes and instructions and instructions failing condition codes.

21.8Debugging component (DBG)

The MCU Debug Component helps the debugger provide support for:

- low power mode
- Timer during breakpoint, clock control of watchdog

21.8.1 Debug support for low power modes

To enter a low-power mode (Sleep, DeepSleep), the instruction WFI or WFE must be executed.

The MCU supports several low-power modes that disable the CPU clock or reduce CPU power consumption.

The kernel does not allow turning off FCLK or HCLK during a debug session. They must remain active as they are required for debug connections during debugging. MCUs integrate special methods that allow users to debug software in low-power modes.

21.8.2 Debug support for timers, watchdog

During a breakpoint, the behavior of the counters of the timer and watchdog must be selected:

- The counter continues to count while the breakpoint is generated. For example, this is often required when PWM is controlling a motor.
- When a breakpoint is generated, the counter stops counting. This is required when used with a watchdog.

22 Appendix A SysTick Timer

22.1 Introduction to SysTick Timer

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-M0 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 (abnormal 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.

22.2 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 as 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 as 1 to enable SysTick interrupt.
Step6: Configure SysTick->CTRL.ENABLE as 1 to enable SysTick.
Step7: Read SysTick->CTRL in the interrupt service routine to clear the overflow flag.

Note: The SysTick overflow period is SysTick->LOAD+1, the configuration example is as follows:

Timer	SysTick->LOAD	overflow cycle
HCLK 4MHz	3999	1ms

22.3SysTick register

Address	Name	CMSIS symbol	full name
0xE000 E010	SYS_CSR	SysTick->CTRL	SysTick Control and Status Register
0xE000 E014	SYS_RVR	SysTick->LOAD	SysTick reload register
0xE000 E018	SYS_CVR	SysTick->VAL	SysTick current value register
0xE000 E01C	SYS_CALIR	SysTick->CALIB	SysTick Calibration Value Register

22.3.1 SysTick Control and Status Register (CTRL)

Bit	Symbol	Functional description	Types of	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 No reference clock, always use system clock as SysTick clock	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

22.3.2 SysTick reload register (LOAD)

Bit	Symbol	Functional description	Types of	Reset value
31:24	Reserved	-	-	-
23:0	RELOAD	SysTick timer reload value	RW	undefined

22.3.3 SysTick Current Value Register (VAL)

Bit	Symbol	Functional description	Types of	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

22.3.4 SysTick Calibration Value Register (CALIB)

Bit	Symbol	Functional description	Types of	Reset value
31	NOREF	SysTick current count clock flag 1: The current count clock is the core clock 0: no function	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	10ms calibration value	RO	-

23 Appendix B Document Conventions

23.1 List of register-related abbreviations

The following abbreviations are used in register descriptions:

RW read and write, software can read and write these bits.

RO is read-only, software can only read these bits.

WO is write only, software can only write to this bit. Reading this bit will return invalid data.

W1 only writes 1, the hardware automatically clears 0, writing 0 is invalid

R0W1 software reads this bit as 0, writes 1 to clear this bit. Writing a 0 has no effect on the value of this bit.

RW0 software can read and write this bit, write 1 is invalid, write 0 to clear

R1W0 software reads this bit as 1, and writes 0 to clear this bit. Writing a 1 has no effect on the value of this bit.

RC software can read this bit. When this bit is read, it is automatically cleared. Writing "0" has no effect on the value of this bit.

Res, Reserverd Reserved bit, must keep the reset value.

23.2 Glossary

This section provides brief definitions of acronyms and abbreviations used in this document:

Word: 32 -bit data.

Half Word: 16 -bit data.

Byte: 8 -bit data.

IAP (In-Application Programming): IAP means that the microcontroller's Flash can be reprogrammed while the user program is running.

ICP (In Circuit Programming): ICP means that the Flash of the microcontroller can be programmed using the JTAG protocol, SWD protocol or bootloader when the device is installed on the user application circuit board.

AHB: Advanced High Performance Bus.

APB: Advanced Peripheral Bus.

DMA: Direct Memory Access.

TIM: timer

Version revision history

version number	Revision Date	modify the content
Rev1.0	2021/12/31	The first draft is released.
Rev1.01	2022/03/09	The company logo is updated.
Rev1.02	2023/06/21	1) In "General Purpose Timer " chapter, filter information of Control Register 1 (GTIM_CR1) is modified. 2) In the "Mode 1 Register Description" chapter, Control Register (ATIMx_CR0) adds a function description that is not yet open. 3) "Device Electronic Signature" chapter, Product Unique Identifier (UID) Register, base address: 0x0010 0980 is changed to 0x0010 1080; Pin Count Register, base address: 0x0010 08AA is changed to 0x0010 0DBA.