Linko Semiconductor Co., Ltd.

LKS32MC45x User Manual

Contents

List of Tables

List of Figures

1 Document Convention

1.1 Register Read/Write Access

RW Read/write, available for software read and write.

RO Read-only, software can only read.

WO Write-only, software can only write. The default value will be returned when reading this bit.

RW1C Read and Write 1 to clear.

RS Read/write 1, 0 will be invalid.

1.2 Abbreviations

Word: 32-bit data/command.

Halfword: 16-bit data/command.

Byte: 8-bit data.

Double word: 64-bit data.

CRAM: Code RAM

ADC: Analog-Digital Converter

DAC: Digital-Analog Converter

BGP: Bandgap. Bandgap voltage reference

WDT: Watch dog

LSI: Low Speed Internal Clock, the 32kHz RC oscillator

HSI: High Speed Internal Clock, the 12MHz RC oscillator

HSE: High Speed External Clock, the 10-24MHz external crystal oscillator

PLL: Phase Lock Loop Clock, the 192MHz PLL clock, which usually used as high-speed system clock

POR: Power-On Reset. Reset signal generated when the chip system is powered on

NVR: Non-Volatile Register. A storage area in the flash that is different from the main area.

IAP (In-Application Programming): IAP means that the Flash of the microcontroller can be reprogrammed while the user program is running.

ICP (In-Circuit Programming): ICP means that you can use the JTAG protocol, SWD protocol or bootloader to program the Flash of the microcontroller when the device is installed on the Subscriber Circuit Board.

CW: Clockwise

CCW: Counterclockwise

Option bytes: Option byte, the MCU configuration byte saved in Flash.

2 System Overview

2.1 Introduction

LKS32MC45x series MCUs are high-end MCUs with dual motor drives, equipped with 192MHz Cortex-M4F CPU core and rich analog and peripheral resources, which are widely used in various motor control applications with high demand for arithmetic power.

2.2 Feature

- 192MHz 32-bit Cortex-M4F core
- 8-channel DMA
- 6uA low power sleep mode
- -40~105°C industrial operating temperature range
- 2.2V~3.6V single power supply, internally integrated digital power supply LDO. Some devices support 5V power input.
- Some IOs are 5V tolerant
- Super antistatic and group pulse ability

2.2.1 Storage

- 128/256kB Flash, data anti-theft function
- 40kB RAM

2.2.2 Clock

- Built-in 12MHz high-precision RC oscillator, with full temperature range accuracy ±1%
- Built-in 32kHz low speed clock for low power mode
- Internal PLL supporting up to 192MHz clock

2.2.3 Peripheral Clock

- 3 UART
- 2 SPI
- 2 IIC

- 3 universal 16-bit Timer and 2-channel universal 32bit Timer, supporting capturing and edge-aligned PWM
- 2 Motor control dedicated PWM module, each supporting 4/8 PWM outputs and independent dead zone control
- 2 hall signal interfaces, supporting speed measurement and debouncing
- Hardware independent watchdog + hardware window watchdog

2.2.4 Analog module

- Built-in 3-channel 14bit SAR ADC, with the sampling and conversion rate of 2 Msps. Each ADC has 16 signal channels, and the positive and negative signal channels can be freely configured.
- Built-in 6 OPA. Differential PGA mode is available
- Built-in 6 comparators
- Built-in 2 12bit DACs as internal comparator input
- 1.2V 0.5% built-in linear regulator
- Built-in low power LDO and power monitoring circuit
- Built-in high-precision, low-temperature drift and high-frequency RC oscillator

2.3 System Block Diagram

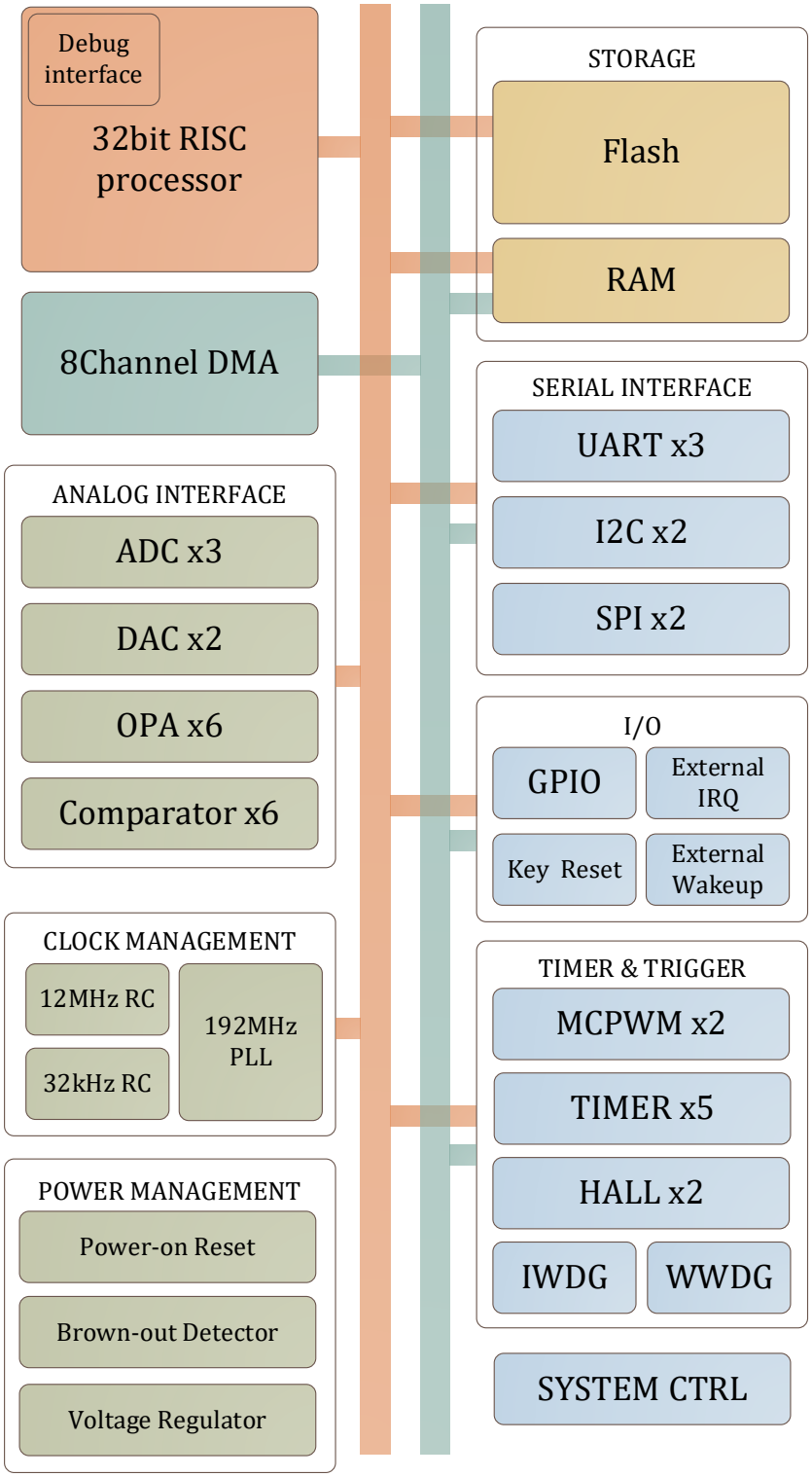


Figure 2-1 LKS32MC45x system block diagram

3 Address Space

The data bytes are stored in memory in little-endian format. The lowest address byte in a word is considered the least significant byte of the word, and the highest address byte is the most significant byte. All other unallocated on-chip memory and external memory are reserved address spaces.

Since the read speed of RAM is much higher than that of FLASH, copying programs from FLASH space to CRAM space can speed up the fetching and execution of programs. For some applications that require less Data RAM, SRAM can be used as CRAM for storing program fragments that need to run faster.

Four address spaces can be configured through the register SYS_REMAP, which is mainly used to map SRAM to 0x0000_0000-0x1FFF_FFFF to expand the Code RAM space so that programs can be loaded into RAM for execution to improve the operation efficiency of critical codes.

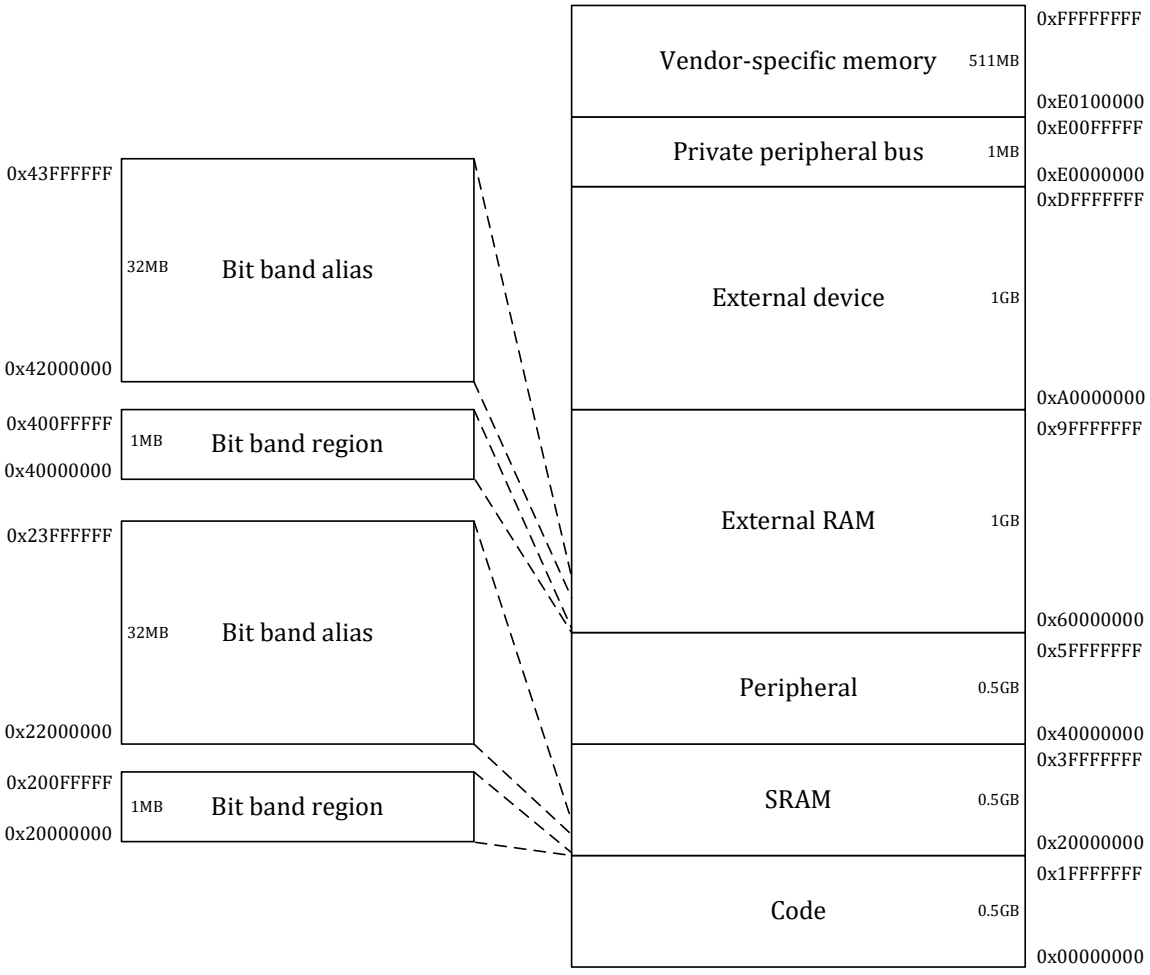


Figure 3-1 RISC CPU Memory Model

3.1 CODE Segment

Address space range: 0x0000_0000-0x1FFF_FFFF, accessed through the I/D Bus exit of the RISC CPU.

The CODE segment mainly contains On-chip Flash, Ext-Flash and Code RAM (CRAM).

Table 3-1 Storage Address Space

Start Address		Size
0x0000_0000	On-chip Flash	256kB
0x0003_FFFF		
0x0100_0000	Ext-Flash	Support up to 16MB
0x01FF_FFFF		
0x0200_0000	Reserved	
0x0FFF_FFFF		
0x1000_0000	CRAM	Support up to 24kB
0x1000_7FFF		

Ext-Flash supports up to 16MB, and the actual particle capacity can be less than 16MB.

However, the burning algorithms of On-chip Flash and Ext-Flash are different, so different burning methods should be selected according to the address in the burning algorithm.

CRAM supports up to 24kB. There is no CRAM in the chip by default, so you need to modify the register SYS_REMAP to switch the address space mapping and use SRAM as CRAM. In the load file, the functions that need to be put into CRAM for execution need to be specially specified, and this part of Code is copied from the load domain to the execution domain using the decentralized loading mechanism.

It should be noted that since the chip does not have CRAM by default, SYS_REMAP needs to be modified in the SystemInit() function to switch the address space mapping, after which __main() will complete the data initialization of the RW/ZI segment and copy the program from flash to CRAM before entering the main() function.

```
; Reset handler routine
Reset_Handler    PROC
                 EXPORT  Reset_Handler                 [WEAK]
        IMPORT   __main
        IMPORT   SystemInit


        LDR      R0, =__initial_sp          ; set stack pointer
        MSR      MSP, R0


ApplicationStart
                 LDR     R0, =SystemInit
                 BLX     R0
                 LDR     R0, =__main
                 BX      R0
                 ENDP
```

Figure 3-2 Reset_Handler implementation

3.2 RAM Segment

Address space range: 0x2000_0000-0x3FFF_FFFF, accessed through the S Bus exit of the RISC CPU.

SRAM is divided into 4 segments.

Table 3-2 SRAM Size and Use

	Size	Purpose
SRAM0	16kB	Data RAM
SRAM1	8kB	Data RAM/Code RAM
SRAM2	8kB	Data RAM/Code RAM
SRAM3	8kB	Data RAM/Code RAM

Table 3-3 Use of SRAM in different address space configurations

	REMAP==0	REMAP==1	REMAP==2	REMAP==3
SRAM0	Data RAM0	Data RAM0	Data RAM0	Data RAM0
SRAM1	Data RAM1	Data RAM1	Data RAM1	Code RAM2
SRAM2	Data RAM2	Data RAM2	Code RAM1	Code RAM1
SRAM3	Data RAM3	Code RAM0	Code RAM0	Code RAM0

3.3 PERIPHERAL Segment

Address space range: 0x4000_0000-0x5FFF_FFFF. It is the address space of all peripherals and accessed through the S Bus exit of the RISC CPU.

3.4 Address Space 0 (SYS_REMAP=0)

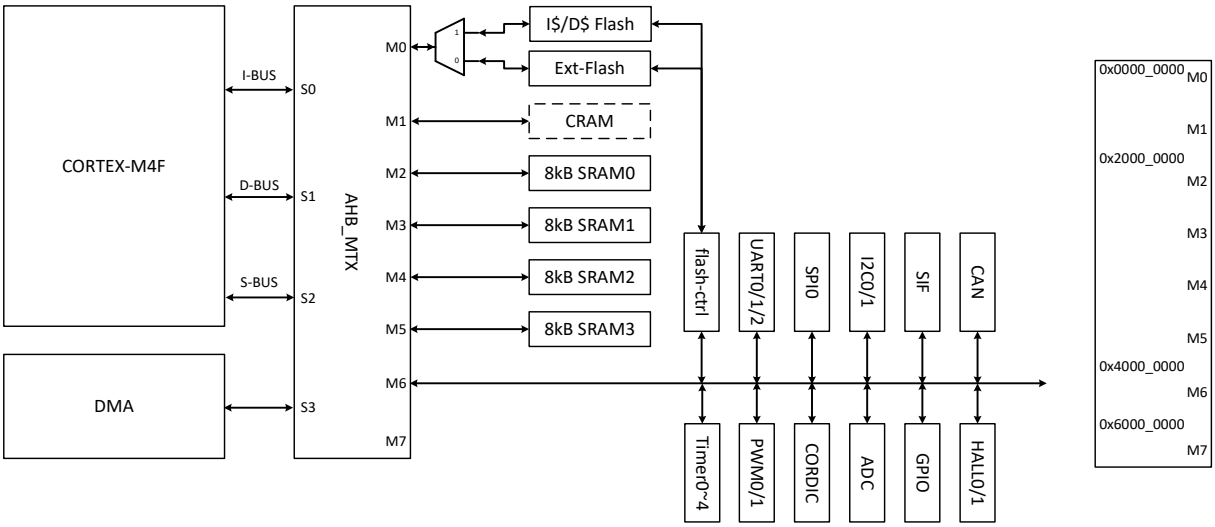


Figure 3-3 Bus architecture (REMAP=0)

This is the default bus architecture after reset. All SRAMs are Data RAM, without Code RAM for acceleration.

Table 3-4 System Address Space Allocation (REMAP=0)

Type	Peripheral Clock	Start Address	End Address	Operating Clock/Soft Reset	Size
CODE	FLASH	0x0000_0000	0x0003_FFFF	Same as bus / N/A	256kB
	EXT-FLASH	0x0100_0000	0x01FF_FFFF	Same as bus / N/A	16MB
RAM	SRAM0	0x2000_0000	0x2000_3FFF	Same as bus / N/A	16kB
	SRAM1	0x2000_4000	0x2000_5FFF	Same as bus / N/A	8kB
	SRAM2	0x2000_6000	0x2000_7FFF	Same as bus / N/A	8kB
	SRAM3	0x2000_8000	0x2000_9FFF	Same as bus / N/A	8kB
	Reserved	0x2000_A000	0x21FF_FFFF		
	Bit band alias	0x2200_0000	0x23FF_FFFF		32MB
	Reserved	0x2400_0000	0x3FFF_FFFF		
PERIP	Reserved	0x4000_0000	0x4000_03FF		1kB
	Reserved	0x4000_0400	0x4000_07FF		1kB
	Reserved	0x4000_0800	0x4000_0BFF		1kB
	Reserved	0x4000_0C00	0x4000_0FFF		1kB
	Reserved	0x4001_0000	0x4001_03FF		1kB
	Reserved	0x4001_0400	0x4001_07FF		1kB
	Reserved	0x4001_0800	0x4001_0BFF		1kB
	Reserved	0x4001_0C00	0x4001_0FFF		1kB

	I2C0	0x4001_1000	0x4001_13FF	Peripheral clock gating [0]/soft reset [0]	1kB
	Updated I2C	0x4001_1400	0x4001_17FF	Peripheral clock gating [1]/soft reset [1]	1kB
	HALL0	0x4001_1800	0x4001_1BFF	Peripheral clock gating [2]/soft reset [2]	1kB
	HALL1	0x4001_1C00	0x4001_1FFF	Peripheral clock gating [3]/soft reset [3]	1kB
	UART0	0x4001_2000	0x4001_23FF	Peripheral clock gating [4]/soft reset [4]	1kB
	UART1	0x4001_2400	0x4001_27FF	Peripheral clock gating [5]/soft reset [5]	1kB
	UART2	0x4001_2800	0x4001_2BFF	Peripheral clock gating [6]/soft reset [6]	1kB
	Reserved	0x4001_2C00	0x4001_2FFF		1kB
	CMP	0x4001_3000	0x4001_33FF	Peripheral clock gating [8]/soft reset [8]	1kB
	MCPWM0	0x4001_3400	0x4001_37FF	Peripheral clock gating [9]/soft reset [9]	1kB
	MCPWM1	0x4001_3800	0x4001_3BFF	Peripheral clock gating [10]/soft reset [10]	1kB
	Reserved	0x4001_3C00	0x4001_3FFF		1kB
	TIMER0	0x4001_4000	0x4001_40FF	Peripheral clock gating [12]/soft reset [12]	256B
	TIMER1	0x4001_4100	0x4001_41FF	Peripheral clock gating [13]/soft reset [13]	256B
	TIMER2	0x4001_4200	0x4001_42FF	Peripheral clock gating [14]/soft reset [14]	256B
	TIMER3	0x4001_4300	0x4001_43FF	Peripheral clock gating [15]/soft reset [15]	256B
	TIMER4	0x4001_4400	0x4001_44FF	Peripheral clock gating [16]/soft reset [16]	256B
	QEP0	0x4001_4500	0x4001_45FF	Peripheral clock gating [17]/soft reset [17]	256B
	QEP1	0x4001_4600	0x4001_46FF	Peripheral clock gating [18]/soft reset [18]	256B
	QEP2	0x4001_4700	0x4001_47FF	Peripheral clock gating [19]/soft reset [19]	256B
	QEP3	0x4001_4800	0x4001_48FF	Peripheral clock gating [20]/soft reset [20]	256B
	Reserved	0x4001_4900	0x4001_63FF		

	GPIO	0x4001_6400	0x4001_67FF	Peripheral clock gating [22]/soft reset [22]	1kB
	CRC	0x4001_6800	0x4001_6BFF	Peripheral clock gating [23]/soft reset [23]	1kB
	CORDIC	Address: 0x4001_6C00	0x4001_6FFF	Peripheral clock gating [24]/soft reset [24]	1kB
	FMAC	0x4001_7000	0x4001_77FF	Peripheral clock gating [25]/soft reset [25]	2kB
	ADC0	0x4001_7800	0x4001_7BFF	ACLK/soft reset [26]	1kB
	ADC1	0x4001_7C00	0x4001_7FFF	ACLK/soft reset [27]	1kB
	ADC2	0x4001_8000	0x4001_83FF	ACLK/soft reset [28]	1kB
	Reserved	0x4001_8400	0x4001_87FF		1kB
	CAN	0x4001_8800	0x4001_8BFF	Same as bus/soft reset [29]	1kB
	Reserved	0x4001_8C00	0x4001_8FFF		1kB
	AFE	0x4001_9000	0x4001_93FF		1kB
	Reserved	0x4001_9400	0x4001_97FF		1kB
	SYS	0x4001_9800	0x4001_9BFF	Same as bus / N/A	1kB
	DMA	0x4001_9C00	0x4001_9FFF	Same as bus / N/A	1kB
	FLSCR	0x4001_A000	0x4001_A3FF	Same as bus / N/A	1kB
	Reserved	0x4001_A400	0x4001_A7FF		1kB
	SPI0	0x4001_A800	0x4001_ABFF	Peripheral clock gating [30]/soft reset [30]	1kB
	SPI1	0x4001_AC00	0x4001_AFFF	Peripheral clock gating [31]/soft reset [31]	1kB
	EXT_FLSCR	0x4001_B000	0x4001_B3FF		1kB
	WWDG	0x4001_B400	0x4001_B7FF		1kB
always on	BRAM	0x4001_B800	0x4001_B87F		128B
	AONCR	0x4001_B880	0x4001_B8BF		64B
	IWDG	Address: 0x4001_B8C0	0x4001_B8FF		64B
	Reserved				
	Bit band alias	0x4200_0000	0x43FF_FFFF		32MB

3.5 Address Space 1 (SYS_REMAP=1)

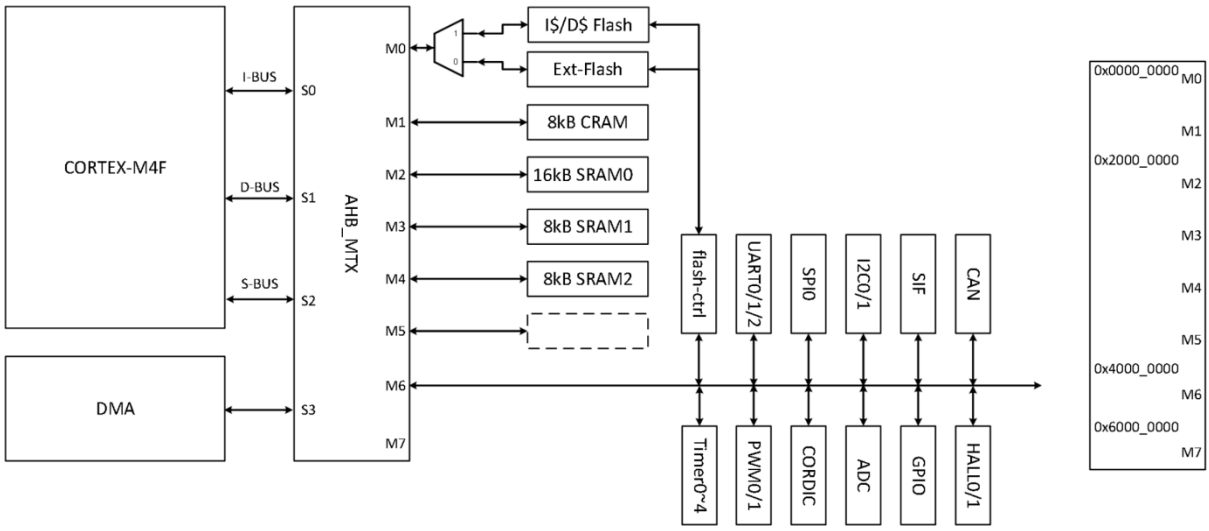


Figure 3-4 Bus architecture (REMAP=1)

The 8kB SRAM3 is mapped as CRAM, and SRAM0/1/2 are still used as Data RAM. CRAM is 8kB, and Data RAM is 32kB.

Table 3-5 System Address Space Allocation (REMAP=1)

Type	Peripheral Clock	Start Address	End Address	Operating Clock/Soft Reset	Size
CODE	FLASH	0x0000_0000	0x0003_FFFF	Same as bus / N/A	256kB
	EXT-FLASH	0x0100_0000	0x01FF_FFFF	Same as bus / N/A	16MB
RAM	SRAM0	0x2000_0000	0x2000_3FFF	Same as bus / N/A	16kB
	SRAM1	0x2000_4000	0x2000_5FFF	Same as bus / N/A	8kB
	SRAM2	0x2000_6000	0x2000_7FFF	Same as bus / N/A	8kB
	SRAM3	0x1000_8000	0x1000_9FFF	Same as bus / N/A	8kB
	Reserved	0x2000_A000	0x21FF_FFFF		
	Bit band alias	0x2200_0000	0x23FF_FFFF		32MB
	Reserved	0x2400_0000	0x3FFF_FFFF		
PERIP	Same as REMAP=0				

3.6 Address space 2 (SYS_REMAP=2)

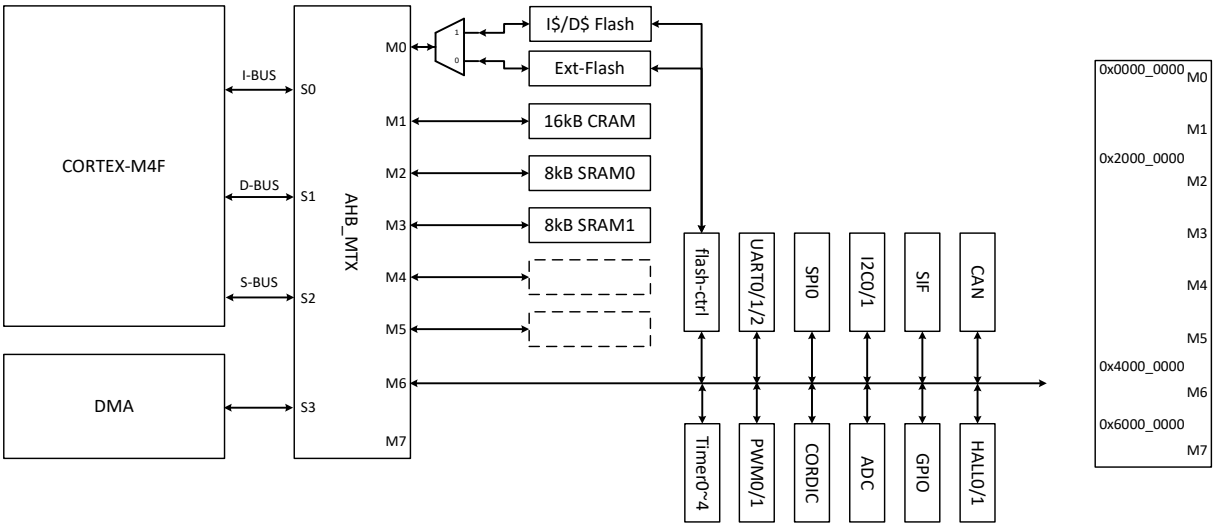


Figure 3-5 Bus architecture (REMAP=2)

The 8kB SRAM3/SRAM2 are mapped as CRAM0/1, respectively, and SRAM0/1 are still used as Data RAM. CRAM is 16kB, and Data RAM is 16kB.

Table 3-6 System Address Space Allocation (REMAP=2)

Type	Peripheral Clock	Start Address	End Address	Operating Clock/Soft Reset	Size
CODE	FLASH	0x0000_0000	0x0003_FFFF	Same as bus / N/A	256kB
	EXT-FLASH	0x0100_0000	0x01FF_FFFF	Same as bus / N/A	16MB
RAM	SRAM0	0x2000_0000	0x2000_3FFF	Same as bus / N/A	16kB
	SRAM1	0x2000_4000	0x2000_5FFF	Same as bus / N/A	8kB
	SRAM2	0x1000_6000	0x1000_7FFF	Same as bus / N/A	8kB
	SRAM3	0x1000_8000	0x1000_9FFF	Same as bus / N/A	8kB
	Reserved	0x2000_A000	0x21FF_FFFF		
	Bit band alias	0x2200_0000	0x23FF_FFFF		32MB
	Reserved	0x2400_0000	0x3FFF_FFFF		
PERIP	Same as REMAP=0				

3.7 Address Space 3 (SYS_REMAP=3)

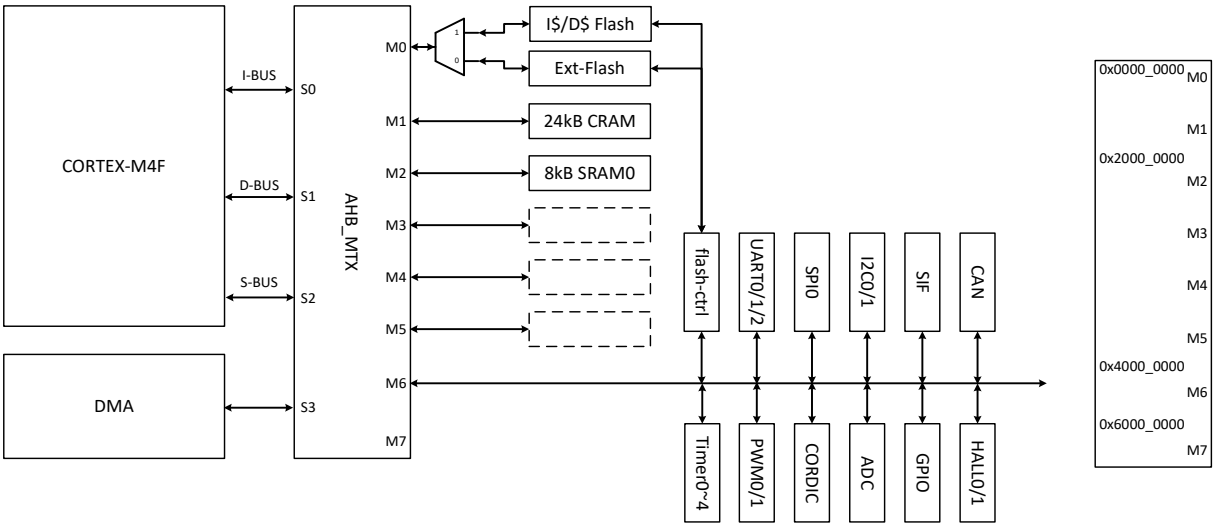


Figure 3-6 Bus architecture (REMAP=3)

The 8kB SRAM3/2/1 are mapped as CRAM0/1/2, respectively, and SRAM0 is still used as Data RAM. CRAM is 24kB, and Data RAM is 8kB.

Table 3-7 System Address Space Allocation (REMAP=3)

Type	Peripheral Clock	Start Address	End Address	Operating Clock/Soft Reset	Size
CODE	FLASH	0x0000_0000	0x0003_FFFF	Same as bus / N/A	256kB
	EXT-FLASH	0x0100_0000	0x01FF_FFFF	Same as bus / N/A	16MB
RAM	SRAM0	0x2000_0000	0x2000_3FFF	Same as bus / N/A	16kB
	SRAM1	0x1000_4000	0x1000_5FFF	Same as bus / N/A	8kB
	SRAM2	0x1000_6000	0x1000_7FFF	Same as bus / N/A	8kB
	SRAM3	0x1000_8000	0x1000_9FFF	Same as bus / N/A	8kB
	Reserved	0x2000_A000	0x21FF_FFFF		
	Bit band alias	0x2200_0000	0x23FF_FFFF		32MB
	Reserved	0x2400_0000	0x3FFF_FFFF		
PERIP	Same as REMAP=0				

4 Interruption

The nested vectored interrupt controller is inside the CPU core. When an interrupt event occurs, it will notify the CPU to suspend the execution of the main program, and enter the interrupt service function according to priority setting.

In addition to system interruption, the LKS32MC45x series chips have 38 independent interrupt sources and vectors. The interrupt controller supports up to 8bit 256 interrupt priority levels (0-255) for programming. A higher value of interrupt priority means lower priority. For non-system interrupts, 0 represents the highest priority. The non-maskable interrupt (NMI) corresponds to the SRAM check error event, and occurs when SYS_MEM_CFG.PCK_EN is 1 and SYS_MEM_CFG.PCK_ERR is 1.

Table 4-1 Interrupt Number List

Interrupt No.	Description	Interrupt No.	Description
-14	NMI		
-13	HardFault		
-12	Reserved		
-11			
-10			
-9			
-8			
-7			
-6			
-5	SVCall		
-4	Reserved		
-3			
-2	PendSV		
-1	SysTick		
0	I2C0	20	TIMER3
1	Updated I2C	21	TIMER4
2	HALL0	22	QEP0
3	HALL1	23	QEP1
4	UART0	24	QEP2
5	UART1	25	QEP3
6	UART2	26	GPIO(EXTI)
7	CMP0	27	ADC0
8	CMP1	28	ADC1
9	CMP2	29	ADC2
10	CMP3	30	CAN
11	CMP4	31	SPI0

12	CMP5	32	SPI1
13	MCPWM0_IRQ0	33	WWDG
14	MCPWM0_IRQ1	34	DMA
15	MCPWM1_IRQ0	35	WAKEUP
16	MCPWM1_IRQ1	36	PWRDN
17	TIMER0	37	FMAC
18	TIMER1	38	SW
19	TIMER2		

5 Analog Circuit

5.1 Introduction

The analog circuit contains the following modules:

- Built-in 3-channel 14BIT SAR ADC, with the sampling rate of 2MHz. Each ADC has up to 16 signal channels, and the positive and negative signal channels can be freely configured.
- Built-in 6 operational amplifier. PGA mode is available
- Built-in 6 comparators. Hysteresis mode is available
- Built-in 2-channel 12-bit digital-to-analog converter
- ± 2°C built-in temperature sensor
- Built-in linear regulator

The interrelationship between the modules and the control register of each module (see the "Analog Register Table" below for register description) are shown in the figure below.

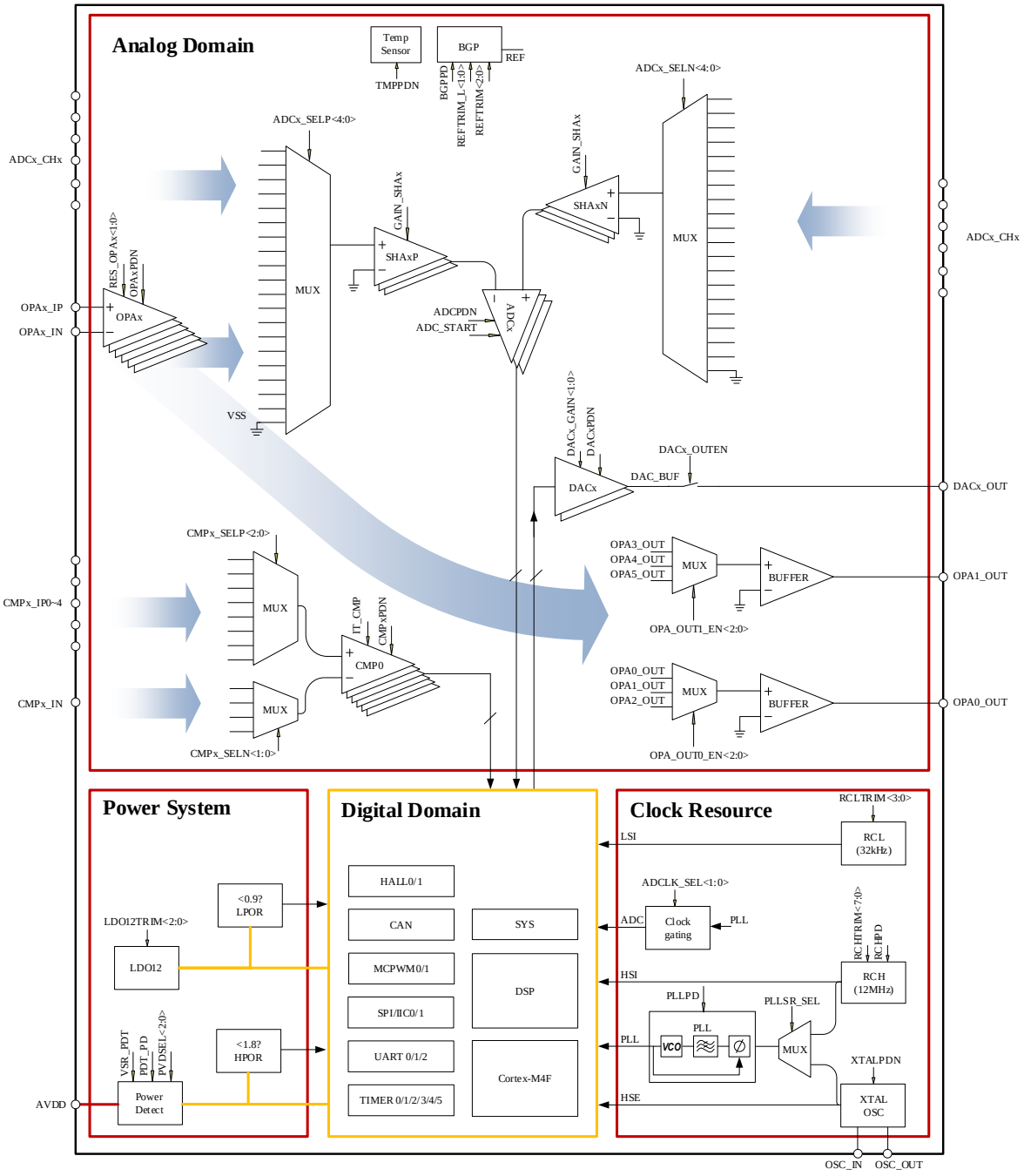


Figure 5-1 Functional Block Diagram of Analog Circuit

5.1.1 Power Management System (POWER)

The power management system is composed of LDO12 module, power detection module (PVD), and power-on/power-off reset module (POR).

The chip is powered by single 3.3V power supply to save off-chip power costs, and all internal digital circuits and PLL modules are powered by a built-in 1.2V LDO12.

The LDO is automatically turned on after power on, without requiring software configuration, but the LDO output voltage can be fine-tuned by software.

LDO12 has been calibrated before the chip shipment.

The LPOR module monitors the voltage of the LDO12 and provides a reset signal for the digital circuit to avoid faults occurred when the voltage of LDO12 is lower than 0.9V (e.g.: power-on or power-off).

The HPOR module monitors the voltage of the AVDD and provides a reset signal for the digital circuit to avoid faults occurred when the voltage of AVDD is lower than 1.8V (e.g.: power-on or power-off).

The PVD module detects the 3.3V input power supply and generates an alarm (interrupt) signal to alert the MCU if it falls below a set threshold value, which can be set to a different voltage by register PVDSEL [1:0]. The PVD module can be disabled by setting PDT_PD = '1'.

For description of PVDSEL [1:0]/PD_PDT, refer to Analog Register B (SYS_AFE_REGB)

5.1.2 Clock System (CLOCK)

The clock system consists of an internal 32KHz RC oscillator, a 12MHz RC oscillator, and a PLL.

The 32kHz RC oscillator LSI is mainly used for watchdog module and reset/wakeup source filters in the system; and the 12MHz RC oscillator is used as the master clock in MCU system, which can also use the PLL clock to provide up to 192MHz.

Both 32kHz and 12MHz RC oscillators have been correctly calibrated at the factory, and they can achieve ± 5% and ± 1% accuracy, respectively, at room temperature. The 12MHz RC oscillator also has an open user calibration register, which can further correct the accuracy to ±0.5%. In the range of -40-105°C, the accuracy of the 32KHz RC oscillator is ±20%, and the accuracy of the 12MHz RC oscillator is ±1%.

The clock has been calibrated before the chip shipment.

The 12MHz RC oscillator is turned on by setting RCHPD = '0'(ON by default, OFF when set to '1'). The RC oscillator needs a reference voltage and current provided by the BGP module. Therefore, BGP (BGPPD='0') must be enabled before the RC oscillator is enabled. When the chip is powered on, the 12MHz RC oscillator and BGP module are both turned on automatically. The 32kHz RC oscillator is always on and cannot be turned off.

The PLL multiplies the 12MHz RC oscillator to provide a higher frequency clock for modules like MCU and ADC. The maximum clock of MCU and PWM modules is 192MHz, and the typical operating clock of ADC module is 32MHz, which can be set to different ADC operating frequency by register ADCLKSEL [1:0].

PLL is turned on by setting PLLPDN = '1'(OFF by default, turn on when set to '1'). Before turning on the PLL module, the BGP module should be turned on first. After the PLL is turned on, it needs a

settling time of 8us to achieve a stable frequency output. When the chip is powered on, the RCH clock and BGP module are both turned on. PLL is OFF by default and enabled by software.

For description of ADCLKSEL<1:0>, refer to Analog Register 3 (SYS_AFE_REG3)

For description of BGPPD / RCHPD / PLLPDN, refer to Analog Register A (SYS_AFE_REGA)

5.1.3 Bandgap Voltage Reference (BGP)

Bandgap Voltage Reference (BGP) provides reference voltage and current for ADC, DAC, RC oscillator, PLL, temperature sensor, operational amplifier, comparator and FLASH. Turn on the Bandgap before using any of the above modules.

When the chip is powered on, the BGP module is turned on automatically. The voltage reference is turned on by setting BGPPD = '0'. From OFF to ON, BGP needs about 6us to stabilize. BGP output voltage is about 1.2V, and accuracy is ± 0.8%

The voltage of BGP has been calibrated before the chip shipment.

5.1.4 ADC Module

Please refer Chapter 13.

5.1.5 Operational Amplifier (OPA)

The 6-channel of rail-to-rail OPAs is integrated, with a built-in feedback resistor, and an external resistor R0 connected to the signal source on the pin. The resistance of feedback resistors R2:R1 can be adjusted by register RES_OPAx [1:0] to achieve different gains.

For description of RES_OPAx<1:0>, refer to Analog Register 0 (SYS_AFE_REG0) and Analog Register 1 (SYS_AFE_REG1)

The schematic diagram of the amplifier is as follows:

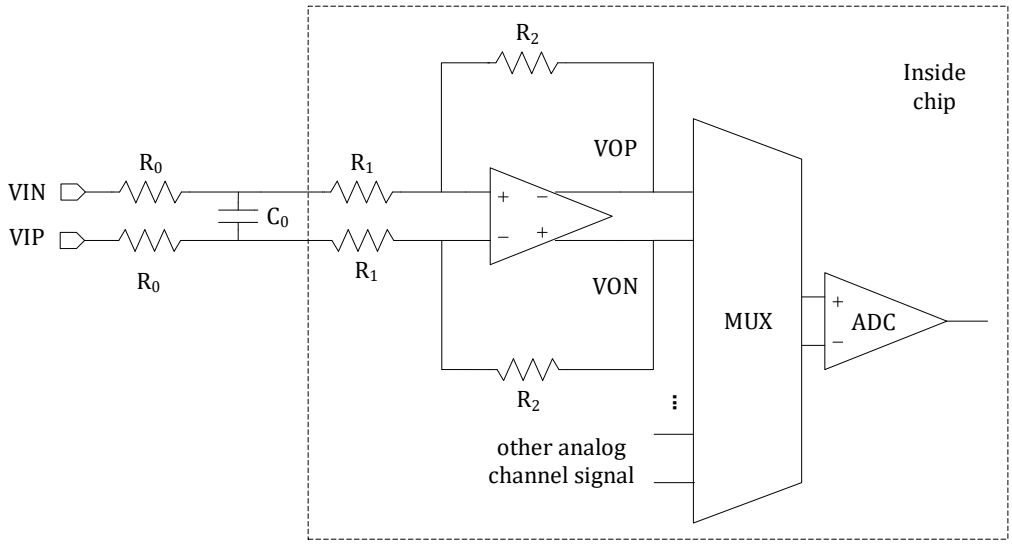


Figure 5-2 Amplifier Block Diagram

The two R_0 in the figure are the resistance of the external resistor, so the resistance must be equal. The close-loop gain of OPA is $R_2/(R_1+R_0)$.

For the application of MOS resistance direct sampling, it is recommended to connect an external resistance of >20kΩ to reduce the current flowing into the chip pin to avoid the voltage signal rises to tens of volts when lower MOS tube is turned off, and the upper tube is turned on.

For the application of shunt resistance sampling, it is recommended to connect an external resistor of 100-2KΩ. C_0 is the signal filter capacitor; it forms a first-order RC filter circuit with R_0, and the specific resistance of R_0 can be determined based on the filter constant of R_0* C_0. The filtering is not necessary if the Signal to Noise Ratio (SNR) is small, or C_0 is not necessary when the signal requires a large bandwidth (faster response speed).

OPA 0/1/2 can select one output signal of the 3-channel amplifier by setting OPAOUT0_EN<1:0>, and send it to P3.11 port through a buffer for measurement, and OPA3/4/5 can select one output signal of the 3-channel amplifier by setting OPAOUT1_EN<1:0>, and send it to P4.6 port through a buffer for measurement (see the corresponding relationship in the datasheet 'Pin Function Description"). Because of the BUFFER, the OPAMP is able to send one output signal in the normal working mode.

For description of OPAOUT0_EN<1:0>, refer to Analog Register 2 (SYS_AFE_REG2)

For description of OPAOUT1_EN<1:0>, refer to Analog Register 2 (SYS_AFE_REG2)

When the chip is powered on, the OPAMP module is OFF by default. It can be turned on by setting OPAxPDN = 1 (x=0,1,2,3,4,5), and turn on the BGP module before turning on the amplifier.

For description of OPAxPDN, refer to Analog

For built-in clamp diodes are integrated between the positive and negative OPA inputs, the motor phase line could be directly connected to the OPA input through a matching resistor, thereby simplifying the external circuit for MOSFET current sampling.

5.1.6 Comparator (CMP)

Built-in 6-channel rail-to-rail comparators with programmable comparator speed, hysteresis voltage, and signal source.

The comparison delay can be set to 150nS/600nS through the register IT_CMP, and the hysteresis voltage can be set to 20mV/40mV through CMP_HYS.

The signal sources of the positive and negative input of the comparator can be set by the registers CMPx_SELP[2:0] and CMPx_SELN[1:0] (x=0/1/2/3/4/5, which represents the comparators 0/1/2/3/4/5).

It should be noted that The HALLx_MID signals at the negative input terminals of the two comparators are the average of the CMPx_IP1/CMPx_IP2/CMPx_IP3 signals at the positive input terminals of the comparator. The specific connection method is shown in Figure 5-3. Among them, the resistance R=8.2KΩ, the switch in the picture will be turned on only after the negative input

signal of the comparator is selected as HALLx_MID, otherwise the switches are in the off state. When pins CMPx_IP1/CMPx_IP2/CMPx_IP3 connect to the HALL signal, the HALL signal can be compared with the HALLx_MID signal to quickly get the HALL signal status.

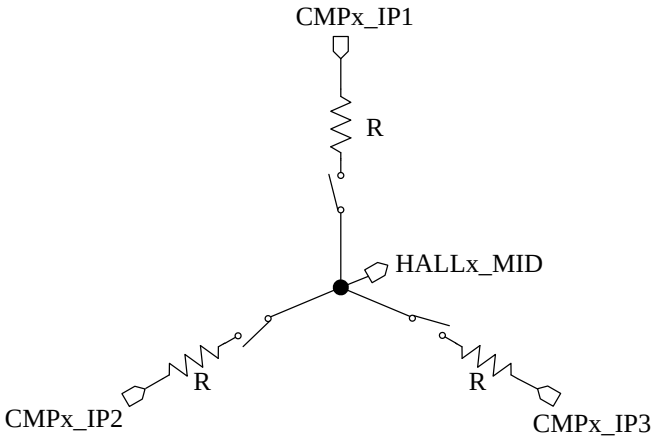


Figure 5-3 HALLx_MID signal

The output of the comparator can be read through the output data register CMP_DATA.

For description of IT_CMP<1:0>, refer to Analog Register 2 (SYS_AFE_REG2)

For description of CMP_HYS, refer to Analog Register 4 (SYS_AFE_REG4)

For description of CMPx_SELN<1:0>, refer to Analog Register 4 (SYS_AFE_REG4) and Analog Register 5 (SYS_AFE_REG5)

For description of CMPx_SELP<2:0>, refer to Analog Register 5 (SYS_AFE_REG5) and Analog Register 6 (SYS_AFE_REG6)

The original output status value of the comparator can be read through the Output Data Register (CMP_DATA)

When the chip is powered on, the comparator module is OFF by default. The comparator is turned on by setting CMPxPDN=1 (x=0, 1, 2, 3, 4, 5), and turn on the BGP module before turning on the comparator.

For description of CMPxPDN, refer to Analog Register 9 (SYS_AFE_REG9)

5.1.7 Temperature sensor

The chip has a built-in temperature sensor with an accuracy of 2°C in the range of -40°C to 85°C. The accuracy is 3°C in the range of 85°C to 105°C.

The operating temperature of chips will be corrected before leaving the factory, and the corrected value is saved in the flash info area.

When the chip is powered on, the temperature sensor module is OFF by default. Turn on the BGP module before turning on the temperature sensor.

The temperature sensor is turned on by setting TMPPDN = '1', and it takes about 2us to be stable after turning on. Thus, it should be turned on at least 2us ahead before the ADC measures the sensor output.

The temperature sensor signal is connected to channel 14 of the ADC2.

For the ADC settings, please refer to Chapter Analog to Digital Converter (ADC)

For description of TMPPDN, refer to Analog Register A (SYS_AFE_REGA)

The typical curve of the temperature sensor is shown in the figure below:

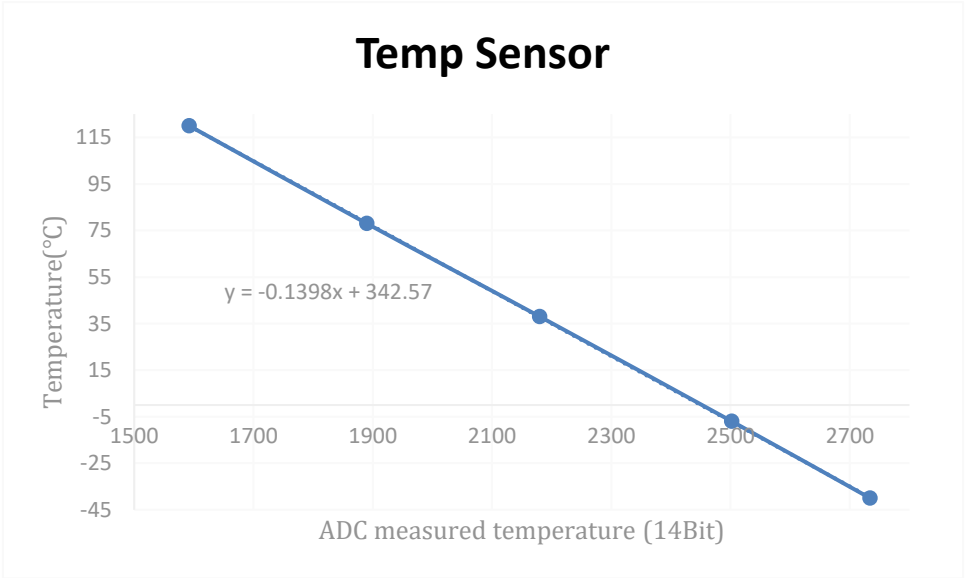


Figure 5-4 Temperature Sensor Curve

The X axis in the figure is the ADC value corresponding to the temperature signal of the temperature sensor, and the Y axis is the temperature of the sensor. When measuring temperature, configure the sensor-related registers according to the above requirements, and put the ADC value as X into the formula after the value obtained:

y=-0.1398x + 342.57

The calculated value Y is the current temperature.

There are two coefficients in the formula, a = -0.1398, b = 342.57, and the value of the coefficient b differs from different chips. The temperature sensor will be calibrated in the factory. Write the coefficient b into the Flash info area (address: 0x00000944). The decimal point of the coefficient b will be shifted to the right by one digit (multiplied by 10) and stored in the info area. The second digit after the decimal point will not be saved.

For the convenience of customers, the coefficient a will also be stored in the Flash info area (address: 0x00000940). The decimal point of the coefficient a will be shifted to the right by four digits (multiplied by 10000) and stored in the info area.

In actual use, the coefficient a and b should be read first from the address in different Flash info area, and then put the measured ADC value into the formula to calculate the current temperature. The unit is degrees Celsius. When calculating, pay attention to the digits of the decimal points of coefficient a and b, that is, the coefficient a should be divided by 10000, and the coefficient b should be divided by 10.

Please note that the above calculation formula is implemented based on ADC right-aligned mode. If the calculation adopts left-aligned mode, the ADC sampling value should be shifted right by two bits before putting into the above formula.

5.1.8 Digital-to-analog Converter (DAC)

The chip has 2-channel 12bit DACs, and the maximum range of the output signal can be set to 1.2V/3V through the register DACx_GAIN <1:0>, where x = 0, 1, representing DAC0/1.

DAC0 can be output via IO port P3.4 by setting register DAC0_OUTEN = 1, DAC1 can be output via IO port P4.7 by setting register DAC1_OUTEN = 1, which can drive a load resistance of over 5KΩ and a load capacitance of 50 pF. It can only output current, but cannot input sinking current.

The maximum output code rate of the DAC is 1MHz.

When the chip is powered on, the DAC module is OFF by default. DAC can be turned on by setting DACxPDN = 1. Turn on the BGP module before turning on the DAC module.

The input digital signal register of DAC is SYS_AFE_DACx, with low 12-bit being effective. The signal range is 0x000-0xFFF. The zero analog output corresponding to the signal range 0x000 is 0V, and the full-scale analog output corresponding to 0xFFF is DAC_{fs}. As mentioned above, the value of DAC_{fs} can be set by the register DAC12B_FS. The analog signal amplitude corresponding to each gear signal (LSB) is If the digital value of SYS_AFE_DAC is Din, the analog signal of the DAC output corresponding to the digital signal is $\frac{DAC_{fs}}{4096} * Din$

There are manufacturing deviations in the DAC by using different chips. Thus, the DAC has a calibration hardware module to offset the deviations. The DAC output formula is: y=ax+b. x is the value filled in SYS_AFE_DAC (ideal digital quantity). a is the value filled in register SYS_AFE_DAC_AMC and b is the value filled in register SYS_AFE_DAC_DC. The hardware multiplies and adds SYS_AFE_DACx, SYS_AFE_DACx_AMC and SYS_AFE_DACx_DC to obtain the corrected digital quantity, and sends it to the DACx input terminal, so that the final analog value of the DACx output is the ideal digital quantity. After the system is powered on, the calibration value of 3V is loaded by default. If it is changed to another range, the software will read the flash info area and reload it into the corresponding register.

DAC0:

Address 0x00000860 is the parameter a of the 3V range, and address 0x00000868 is the

parameter b of the 3V range.

The address 0x00000864 is the parameter a of the 1.2V range, and the address 0x0000086C is the parameter b of the 1.2V range.

DAC1:

Address 0x00000870 is the parameter a of the 3V range, and address 0x00000878 is the parameter b of the 3V range.

The address 0x00000874 is the parameter a of the 1.2V range, and the address 0x0000087C is the parameter b of the 1.2V range.

Except for external module usage via IO port, the analog signal output by the DAC can also be used as a reference signal for the comparator by connecting the configuration register to the negative side of the two-channel CMP inside the chip. See the section Comparator (CMP) for details.

For description of DACx_OUTEN and DACx_GAIN<1:0>, refer to Analog Register 4 (SYS_AFE_REG4)

For description of DACxPDN, refer to Analog Register A (SYS_AFE_REGA)

5.2 Register

5.2.1 Address Allocation

Analog register is SYS_AFE_REG0 - SYS_AFE_REGB, and the corresponding address space is 0x40019810 - 0x4001983C. Among them, the reserved registers (Resv.) must all be set as 0 (it will be reset to 0 after power on). Other registers will be configured according to the actual working situation.

The base address of the analog register is 0x40019800.

Table 5-1 System Control Register

Name	Offset	Description
SYS_AFE_INFO	0x04	Chip version information register
SYS_AFE_DBG0	0x08	Debug register
SYS_AFE_REG0	0x10	Analog register 0
SYS_AFE_REG1	0x14	Analog register 1
SYS_AFE_REG2	0x18	Analog register 2
SYS_AFE_REG3	0x1C	Analog register 3
SYS_AFE_REG4	0x20	Analog register 4
SYS_AFE_REG5	0x24	Analog register 5
SYS_AFE_REG6	0x28	Analog register 6
SYS_AFE_REG9	0x34	Analog register 9
SYS_AFE_REGA	0x38	Analog register A

SYS_AFE_REGB	0x3C	Analog register B
SYS_TMP_A	0x58	Temperature sensor coefficient A
SYS_TMP_B	0x5C	Temperature sensor coefficient B
SYS_AFE_DAC0	0x60	DAC0 digital register
SYS_AFE_DAC0_AMC	0x64	DAC0 gain calibration register
SYS_AFE_DAC0_DC	0x68	DAC0 DC bias register
SYS_AFE_DAC1	0x70	DAC1 digital register
SYS_AFE_DAC1_AMC	0x74	DAC1 gain calibration register
SYS_AFE_DAC1_DC	0x78	DAC1 DC bias register

5.2.2 Chip version information register(SYS_AFE_INFO)

Address: 0x4001_9804

Reset value: different according to wafer version

Table 5-2 Chip Version Information Register(SYS_AFE_INFO)

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

Location	Bit name	Description
[31:8]		Not used
[7:4]		Reserved
[3:0]	Version	Chip version information

This register is user independent.

5.2.3 Debug Register(SYS_AFE_DBG0)

Address: 0x4001_9808

Reset value:0x0

Table 5-3 Debug register(SYS_AFE_DBG0)

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

Location	Bit name	Explain
[31:14]		Not used
[13]	PWR_WEAK	The power supply is lower than the power failure monitoring threshold
[12:0]		Not used

The PWR_WEAK flag is set when the power supply voltage is lower than the threshold set by the power failure monitoring circuit, and a CPU power failure interrupt is generated. When the power supply is restored and the value is higher than the threshold, it is cleared. This flag is a read-only bit

and cannot be cleared by software.

For details about how to use power failure monitoring, see 5.1.1 Power Management System.

5.2.4 Analog Register 0 (SYS_AFE_REG0)

Address: 0x4001_9810

Reset value: 0x0

Table 5-4 Analog Register 0 (SYS_AFE_REG0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RES_OPA3			Res.	RES_OPA2			Res.	RES_OPA1			Res.	RES_OPA0		
RW	RW			RW	RW			RW	RW			RW	RW		
0	0			0	0			0	0			0	0		

Bit field	Bit Name	Description
[31:16]		Unused
[15]	Reserved	Reserved, must be 0
[14:12]	RES_OPA3	OPA3 feedback resistance 000: 20k:10k 001: 40k:10k 010: 80k:10k 011:160k:10k 100: 320k:10k 101: 320k:5k 110/111: 320k:10k;
[11]	Reserved	Reserved, must be 0
[10:8]	RES_OPA2	OPA2 feedback resistance 000: 20k:10k 001: 40k:10k 010: 80k:10k 011:160k:10k 100: 320k:10k 101: 320k:5k 110/111: 320k:10k;
[7]	Reserved	Reserved, must be 0
[6:4]	RES_OPA1	OPA1 feedback resistance 000: 20k:10k 001: 40k:10k 010: 80k:10k 011:160k:10k

		100: 320k:10k 101: 320k:5k 110/111: 320k:10k;
[3]	Reserved	Reserved, must be 0
[2:0]	RES_OPA0	OPA0 feedback resistance 000: 20k:10k 001: 40k:10k 010: 80k:10k 011:160k:10k 100: 320k:10k 101: 320k:5k 110/111: 320k:10k;

5.2.5 Analog Register 1 (SYS_AFE_REG1)

Address: 0x4001_9814

Reset value: 0x0

Table 5-5 Analog Register 1 (SYS_AFE_REG1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.					XDIV	XCSEL		Res.	RES_OPA5			Res.	RES_OPA4		
RW					RW	RW		RW	RW			RW	RW		
0					0	0		0	0			0	0		

Bit field	Bit Name	Description
[31:16]		Unused
[15:11]	Reserved	Reserved, must be 0
[10]	XDIV	Crystal oscillator frequency divided by 2 0: Use 12MHz external crystal oscillator 1: Use 24MHz external crystal oscillator, and set SYS_AFE_REG4.XTRSEL[0] to 1
[9:8]	XCSEL	Crystal oscillator capacitance adjustment
[7]	Reserved	Reserved, must be 0
[6:4]	RES_OPA5	OPA5 feedback resistance 000: 20k:10k 001: 40k:10k 010: 80k:10k 011:160k:10k 100: 320k:10k 101: 320k:5k 110/111: 320k:10k;

[3]	Reserved	Reserved, must be 0
[2:0]	RES_OPA4	OPA4 feedback resistance 000: 20k:10k 001: 40k:10k 010: 80k:10k 011:160k:10k 100: 320k:10k 101: 320k:5k 110/111: 320k:10k;

5.2.6 Analog Register 2 (SYS_AFE_REG2)

Address: 0x4001_9818

Reset value: 0x0

Table 5-6 Analog Register 2 (SYS_AFE_REG2)

15	14	13	12	11	10	9	8	7	6	5 4	3	2	1 0
LDOOUT_EN	Res.	Res.	REF2VDD	Res.	Res.	IT_CMP	CMP_FT	Res.	Res.	OPA_OUT1_EN	Res.	Res.	OPA_OUT0_EN
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:16]		Unused
[15]	LDOOUT_EN	LDO output to IO enable 1: Enable 0: Disable
[14]	Reserved	Reserved, must be 0
[13]	Reserved	Reserved, must be 0
[12]	REF2VDD	Use external input power as ADC REF 1: When using external input power as ADC REF, GAIN_REF configuration is invalid; 0: Take the default value as the internal ADC reference
[11]	Reserved	Reserved, must be 0
[10]	Reserved	Reserved, must be 0
[9]	IT_CMP	Comparison speed selection of comparators 1-5 0: Comparison speed 150ns 1: Comparison speed 600ns
[8]	CMP_FT	Enable comparator for quick comparison 1: When IT_CMP is the default '0', the comparison speed of the comparator is less than 30ns

		0: Disable
[7]	Reserved	Reserved, must be 0
[6]	Reserved	Reserved, must be 0
[5:4]	OPAOUT1_EN	Enable OPAx (x=3,4,5) output signal via IO port 00: No output 01: Output OPA3 signal via IO port P4.6; 10: Output OPA4 signal via IO port P4.6; 11: Output OPA5 signal via IO port P4.6;
[3]	Reserved	Reserved, must be 0
[2]	Reserved	Reserved, must be 0
[1:0]	OPAOUT0_EN	Enable OPAx (x= 0, 1, 2) output signal via IO port 00: No output 01: Output OPA0 signal via IO port P3.11; 10: Output OPA1 signal via IO port P3.11; 11: Output OPA2 signal via IO port P3.11;

5.2.7 Analog Register 3 (SYS_AFE_REG3)

Address: 0x4001_981C

Reset value: 0x0

Table 5-7 Analog Register 3 (SYS_AFE_REG3)

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

Bit field	Bit Name	Description
[31:16]		Unused
[15]	Reserved	Reserved, must be 0
[14]	Reserved	Reserved, must be 0
[13]	Reserved	Reserved, must be 0
[12]	Reserved	Reserved, must be 0
[11:10]	ADCCLKSEL	ADC clock frequency selection 01: Disable 00: 32MHz 11: 16MHz 10: 8MHz
[9]	Reserved	Reserved, must be 0

[8]	Reserved	Reserved, must be 0
[7:6]	Reserved	Reserved, must be 0
[5:4]	Reserved	Reserved, must be 0
[3:2]	Reserved	Reserved, must be 0
[1:0]	Reserved	Reserved, must be 0

5.2.8 Analog Register 4 (SYS_AFE_REG4)

Address: 0x4001_9820

Reset value: 0x0

Table5-8 Analog Register 4 (SYS_AFE_REG4)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP0_SELN		CMP1_SELN		Res.		Res.	Res.	XTRSEL		DAC1_OUTEN	DAC0_OUTEN	CMP_HYS	DAC1_GAIN	Res.	DAC0_GAIN
RW		RW		RW		RW	RW	RW		RW	RW	RW	RW	RW	RW
0		0		0		0	0	0		0	0	0	0	0	0

Bit field	Bit Name	Description
[31:16]		Unused
[15:14]	CMP0_SELN	Negative selection of Comparator 0 signal 00: CMP0_IN 01: REF 10: DAC0 output 11: Mean value of CMP0_IP1/2/3
[13:12]	CMP1_SELN	Negative selection of Comparator 1 signal 00: CMP1_IN 01: REF 11: DAC1 output 11: Mean value of CMP1_IP1/2/3
[11:10]	Reserved	Reserved, must be 0
[9]	Reserved	Reserved, must be 0
[8]	Reserved	Reserved, must be 0
[7:6]	XTRSEL	Crystal oscillator resistance adjustment XTRSEL<1>=1: N-terminal resistance increases XTRSEL<0>=1: P-terminal resistance is reduced by half
[5]	DAC1_OUTEN	DAC1 output to IO enable 1: Enable DAC1 output to IO P4.7 0: No output
[4]	DAC0_OUTEN	DAC0 output to IO enable

		1: Enable DAC0 output to IO P3.4 0: No output
[3]	CMP_HYS	Comparator hysteresis selection 0: 20mv 1: 40mv
[2]	DAC1_GAIN	DAC1 range selection 0: Full scale is 3V 1: Full scale is 1.2V
[1]	Reserved	Reserved, must be 0
[0]	DAC0_GAIN	DAC0 range selection 0: Full scale is 3V 1: Full scale is 1.2V

5.2.9 Analog Register 5 (SYS_AFE_REG5)

Address: 0x4001_9824

Reset value: 0x0

Table 5-9 Analog Register 5 (SYS_AFE_REG5)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CMP0_SELP				CMP1_SELP			CMP4_SELN		CMP5_SELN		CMP2_SELN		CMP3_SELN	
	RW				RW			RW		RW		RW		RW	
	0				0			0		0		0		0	

Bit field	Bit Name	Description
[31:16]		Unused
[15]	Reserved	Reserved, must be 0
[14:12]	CMP0_SELP	Positive selection of Comparator 0 signal 000: CMP0_IP0 001: OPA0_IP 010: OPA0_OUT_HF, half of OPA0 differential output. See the OPA application notes for details 011: OPA1_OUT_HF 100: CMP0_IP1 101: CMP0_IP2 110: CMP0_IP3 111: AVSS
[11]	Reserved	Reserved, must be 0

[10:8]	CMP1_SELP	Positive selection of Comparator 1 signal 000: CMP1_IP0 001: OPA1_IP 010: OPA1_OUT_HF 011: OPA2_OUT_HF 100: CMP1_IP1 101: CMP1_IP2 110: CMP1_IP3 111: AVSS
[7:6]	CMP4_SELN	Negative selection of Comparator 4 signal 00: CMP4_IN 01: REF 10: DAC0 output 11: DAC1 output
[5:4]	CMP5_SELN	Negative selection of Comparator 5 signal 00: CMP5_IN 01: REF 10: DAC0 output 11: DAC1 output
[3:2]	CMP2_SELN	Negative selection of Comparator 2 signal 00: CMP2_IN 01: REF 10: DAC0 output 11: DAC1 output
[1:0]	CMP3_SELN	Negative selection of Comparator 3 signal 00: CMP3_IN 01: REF 10: DAC0 output 11: DAC1 output

5.2.10 Analog Register 6 (SYS_AFE_REG6)

Address: 0x4001_9828

Reset value: 0x0

Table 5-10 Analog Register 6 (SYS_AFE_REG6)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CMP4_SELP				CMP5_SELP				CMP2_SELP				CMP3_SELP		
	RW				RW				RW				RW		
	0				0				0				0		

Bit field	Bit Name	Description

[31:16]		Unused
[15]	Reserved	Reserved, must be 0
[14:12]	CMP4_SELP	Positive selection of Comparator 4 signal 000: CMP4_IP0 001: OPA4_IP 010: OPA4_OUT_HF 011: OPA5_OUT_HF 100: CMP4_IP1 101: CMP4_IP2 110: CMP4_IP3 111: AVSS
[11]	Reserved	Reserved, must be 0
[10:8]	CMP5_SELP	Positive selection of Comparator 5 signal 000: CMP5_IP0 001: OPA5_IP 010: OPA5_OUT_HF 011: OPA0_OUT_HF 100: CMP5_IP1 101: CMP5_IP2 110: CMP5_IP3 111: AVSS
[7]	Reserved	Reserved, must be 0
[6:4]	CMP2_SELP	Positive selection of Comparator 2 signal 000: CMP2_IP0 001: OPA2_IP 010: OPA2_OUT_HF 011: OPA3_OUT_HF 100: CMP2_IP1 101: CMP2_IP2 110: CMP2_IP3 111: AVSS
[3]	Reserved	Reserved, must be 0
[2:0]	CMP3_SELP	Positive selection of Comparator 3 signal 000: CMP3_IP0 001: OPA3_IP 010: OPA3_OUT_HF 011: OPA4_OUT_HF 100: CMP3_IP1 101: CMP3_IP2 110: CMP3_IP3 111: AVSS

5.2.11 Analog Register 9 (SYS_AFE_REG9)

Address: 0x4001_9834

Reset value: 0x0

Table 5-11 Analog Register 9 (SYS_AFE_REG9)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCPDN	Res.	CMP5PDN	CMP4PDN	CMP3PDN	CMP2PDN	CMP1PDN	CMP0PDN	Res.		OPA5PDN	OPA4PDN	OPA3PDN	OPA2PDN	OPA1PDN	OPA0PDN
RW	RW	RW	RW	RW	RW	RW	RW	RW		RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0		0	0	0	0	0	0

Bit field	Bit Name	Description
[31:16]		Unused
[15]	ADCPDN	ADC enable 0: Disable 1: Enable
[14]	Reserved	Reserved, must be 0
[13]	CMP5PDN	CMP5 enable 0: Disable 1: Enable
[12]	CMP4PDN	CMP4 enable 0: Disable 1: Enable
[11]	CMP3PDN	CMP3 enable 0: Disable 1: Enable
[10]	CMP2PDN	CMP2 enable 0: Disable 1: Enable
[9]	CMP1PDN	CMP1 enable 0: Disable 1: Enable
[8]	CMP0PDN	CMP0 enable 0: Disable 1: Enable
[7:6]	Reserved	Reserved, must be 0
[5]	OPA5PDN	OPA5 enable 0: Disable 1: Enable
[4]	OPA4PDN	OPA4 enable

		0: Disable 1: Enable
[3]	OPA3PDN	OPA3 enable 0: Disable 1: Enable
[2]	OPA2PDN	OPA2 enable 0: Disable 1: Enable
[1]	OPA1PDN	OPA1 enable 0: Disable 1: Enable
[0]	OPA0PDN	OPA0 enable 0: Disable 1: Enable

5.2.12 Analog Register A (SYS_AFE_REGA)

Address: 0x4001_9838

Reset value: 0x0

Table 5-12 Analog Register A (SYS_AFE_REGA)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								PLLPDN	XTALPDN	TMPPDN	LDO2_CTL	DAC1PDN	DAC0PDN	RCHPD	BGPPD
RW								RW	RW	RW	RW	RW	RW	RW	RW
0								0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:16]		Unused
[15:8]	Reserved	Reserved, must be 0
[7]	PLLPDN	PLL enable 0: Disable 1: Enable
[6]	XTALPDN	Crystal oscillator circuit enable 0: Disable 1: Enable
[5]	TMPPDN	Temperature sensor enable 0: Disable 1: Enable
[4]	LDO2_CTL	Close the power failure LDO detection

		0: On 1: Off
[3]	DAC1PDN	DAC1 enable 0: Disable 1: Enable
[2]	DAC0PDN	DAC0 enable 0: Disable 1: Enable
[1]	RCHPD	RCH clock Off 0: On 1: Off
[0]	BGPPD	BGP Off 0: On 1: Off

If SYS_CLK_CFG selects PLL clock, PLLPDN is hardware-controlled, and software configuration of PLLPDN to disable PLL is invalid. Disabling PLL requires PLLPDN=0, and SYS_CLK_CFG does not select PLL as the chip master clock. Both conditions must be satisfied. Similarly, if SYS_CLK_CFG selects HRC clock, then RCHPD is controlled by hardware. It is invalid to configure RCHPD to disable RCH directly by software. Turning off PLL requires RCHPD=1 and the chip goes to sleep. If the chip master clock is a PLL clock and the HRC is a PLL reference clock, then the RCH is also controlled by the hardware. Since RCH and PLL depend on BGP, BGPPD is also hardware-controlled. When RCH or PLL is used on a chip, it is invalid to configure BGPPD in software to disable BGP. To disable BGP, disable the PLL and RCH in sequence and the chip goes to sleep. LDO2_CTL is also controlled by the hibernation state machine (see Section 24.4 for details).

5.2.13 Analog Register B (SYS_AFE_REGB)

Address: 0x4001_983C

Reset value: 0x0

Table 5-13 Analog Register B (SYS_AFE_REGB)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WK_EN								PLLSR_SEL	PORFIL_EN	WK_POL	Res.	PVDSEL		VSR_PDT	PDT_PD
RW								RW	RW	RW	RW	RW		RW	RW
0								0	0	0	0	0		0	0

Bit field	Bit Name	Description
[31:16]		Unused
[15:8]	WK_EN	Wakeup IO enable, 1: Enable, 0: Disable WK_EN[7]: Enable P1.12 as external wake-up signal

		WK_EN[6]: Enable P1.11 as external wake-up signal WK_EN[5]: Enable P0.14 as external wake-up signal WK_EN[4]: Enable P0.13 as external wake-up signal WK_EN[3]: Enable P0.5 as external wake-up signal WK_EN[2]: Enable P0.4 as external wake-up signal WK_EN[1]: Enable P0.3 as external wake-up signal WK_EN[0]: Enable P0.2 as external wake-up signal
[7]	PLLSR_SEL	PLL reference clock selection 0: Use RCH clock 1: Use crystal oscillator For products or applications that do not use crystal oscillator, the clock outputs 0; for products or applications that use crystal oscillator and when the oscillator does not stop, the clock outputs 1; and when the oscillator stops, the clock output 0
[6]	PORFIL_EN	POR filter enable 0: Disable 1: Enable
[5]	WK_POL	Wakeup IO polarity selection 0: Active low, add 100k pull-up resistance to the selected wakeup IO 1: Active high, add 100k pull-down resistance to the selected wakeup IO
[4]	Reserved	Reserved, all '0'
[3:2]	PVDSEL	Threshold selection for power failure detection 00: 3.0V 01: 2.7 10: 2.4V 11: 2.1V
[1]	VSR_PDT	BGP REF selection for power failure detection 0: Select low-power reference source 1: Select DAC0 output
[0]	PDT_PD	Close the power failure detection circuit 0: On 1: Off

5.2.14 Temperature Sensor Coefficient A Register (SYS_TMP_A)

Address: 0x4001_9858

Reset value: 0x0

Table 5-14 Temperature Sensor Coefficient A Register (SYS_TMP_A)

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

RW
0

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	TMP_GAIN_A	Gain calibration coefficient A of temperature sensor

5.2.15 Temperature Sensor Coefficient B Register (SYS_TMP_B)

Address: 0x4001_985C

Reset value: 0x0

Table 5-15 Temperature Sensor Coefficient B Register (SYS_TMP_B)

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

Bit field	Bit Name	Description
[31:16]		Unused
[15:12]	TMP_SIGN	Symbol extension bit of offset calibration coefficient B of temperature sensor
[11:0]	TMP_OFFSET_B	Offset calibration coefficient B of temperature sensor

5.2.16 DAC0 Digital Register (SYS_AFE_DAC0)

Address: 0x4001_9860

Reset value: 0x0

Table 5-16 DAC0 Digital Register (SYS_AFE_DAC0)

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

Bit field	Bit Name	Description
[31:12]		Unused
[11:0]	DAC_IN	DAC0 digital input to be converted

5.2.17 DAC0 Gain Calibration Register (SYS_AFE_DAC0_AMC)

Address: 0x4001_9864

Reset value: 0x0

Table 5-17 DAC0 Gain Calibration Register (SYS_AFE_DAC0_AMC)

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

Bit field	Bit Name	Description
[31:10]		Unused
[9:0]	DAC_AMC	DAC gain calibration value, which is a 10-bit unsigned fixed-point number. B [9] is the integer, and B [8:0] is the decimal

5.2.18 DAC0 DC Offset Register (SYS_AFE_DAC0_DC)

Address: 0x4001_9868

Reset value: 0x0

Table 5-18 DAC0 DC Offset Register (SYS_AFE_DAC0_DC)

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

Bit field	Bit Name	Description
[31:10]		Unused
[15:10]	SIGN_EXT	Signed bit extension, B[15:10]={6{B[9]}}, i.e., B[15:10] is all equal to B[9]
[9:0]	DAC_DC	DAC DC offset, which is a 10bit signed number, and B[9] is the signed bit

5.2.19 DAC1 Digital Register (SYS_AFE_DAC1)

Address: 0x4001_9870

Reset value: 0x0

Table 5-19 DAC1 Digital Register (SYS_AFE_DAC1)

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

	0

Bit field	Bit Name	Description
[31:12]		Unused
[11:0]	DAC_IN	DAC1 digital input to be converted

5.2.20 DAC1 Gain Calibration Register (SYS_AFE_DAC1_AMC)

Address: 0x4001_9874

Reset value: 0x0

Table 5-20 DAC1 Gain Calibration Register (SYS_AFE_DAC1_AMC)

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

Bit field	Bit Name	Description
[31:10]		Unused
[9:0]	DAC_AMC	DAC gain calibration value, which is a 10-bit unsigned fixed-point number. B [9] is the integer, and B [8:0] is the decimal

5.2.21 DAC1 DC Offset Register (SYS_AFE_DAC1_DC)

Address: 0x4001_9878

Reset value: 0x0

Table 5-21 DAC1 DC Offset Register (SYS_AFE_DAC1_DC)

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

Bit field	Bit Name	Description
[31:10]		Unused
[15:10]	SIGN_EXT	Signed bit extension, B[15:10]={6{B[9]}}, i.e., B[15:10] is all equal to B[9]
[9:0]	DAC_DC	DAC DC offset, which is a 10bit signed number, and B[9] is the signed bit

DAC gain calibration. The 12-bit DAC value of the digital output is DAC_raw, and the 12-bit DAC value after calibration is DAC_cali.

DAC_cali = DAC_raw * DAC_AMC – DAC_DC;

DAC_AMC is the DAC gain correction coefficient, which is a 10-bit unsigned fixed-point number, and B [9] is an integer, while B [8: 0] is a decimal about 1. For example, DAC_AMC = 10'b10_0001_0000 = 1+1/32, or

DAC_AMC = 10'b01_1110_1100 = 1-5/128

DAC_DC is DAC DC bias, which is a 10-bit signed integer, and signed extended to a 16-bit signed integer. The 16-bit width is more easy for software to read as a signed short variable.

After gain calibration, the calculation result is truncated and reserved, and the final DAC_cali is still a 12-bit integer.

Besides, the value after gain calibration and DC bias calibration will be saturated. The maximum value is 0xfff and the minimum value is 0x000.

Please note that the DAC has three output gears. After power on, the DAC calibration value will be loaded automatically by default. If switch to another gear, please use the library function provided by the manufacturer.

6 System Control and Clock Reset

6.1 Clock

6.1.1 Clock Source

As shown in the following table, the system includes 5 clock sources, of which the internal low-speed RC oscillator (LSI, Low Speed Internal Clock) and internal high-speed RC oscillator (HSI, High Speed Internal Clock) will not stop vibration.

Table 6-1 System Clock Source

Clock Source	Frequency	Source	Error	Description
LSI	32KHz	Internal RC oscillator	Full temperature range error <50%	Internal system clock, used for watchdog module, and filtering and widening of reset signal
HSI	12MHz	Internal RC oscillator	Full temperature range error <1%	Can be used as PLL source clock
PLL	192MHz	PLL clock	0	Taking the HSI as the reference clock, PLL outputs the clock that is 24 times the frequency of the HSI/HSE clock, which used as the main system clock
JTAG/SWD clock	Depend on the settings	Debugger		JTAG/SWD clock
XTAL	10~24MHz	Crystal		External crystal

The SWD clock is are related to the downloader settings.

The system can use the internal high-speed RC oscillator HSI as the reference clock of PLL. The PLL multiplies the 12MHz reference clock HSI by 16 times to 192MHz.

After the PLL is divided by n/8, the high-speed clock of 192MHz × n/8 can be obtained. SYS_CLK_CFG.CLK_SEL chooses one between the divided high-speed clock and the 12MHz HSI as the main system clock (MCLK). When the system is reset, the PLL is turned off and the HSI is turned on by default. The system selects the HSI clock, that is, 12MHz as the main system clock, so as to ensure that the power is low when the system is powered on.

MCLK is the main system clock. The n/8 frequency division can be controlled by the CLK_DIV bit field in the SYS_CLK_CFG register, which can generate 24, 48, 96MHz and other frequency values. SYS_CLK_CFG.CLK_SEL means PLL or 12MHz RC oscillator is selected as the main system clock. When the SYS_CLK_CFG. CLK_SEL is not 1, the system clock IHS/HSE or LSI is used as the operation clock,

and the SYS_CLK_CFG. CLK_DIV is no longer used for frequency division.

Table 6-2 Frequency Division Configuration When PLL is Used as MCLK Clock

SYS_CLK_CFG	Frequency Division Factor	Frequency/MHz	If Uniform
0x0101	1/8	24	Yes
0x0111	2/8	48	Yes
0x0155	4/8	96	Yes
0x01FF	8/8	192	Yes

The MCLK clock is supplied to the peripheral clock after the switch controlled by the SYS_CLK_FEN register. The I2C0/1 clock could be further divided by the SYS_CLK_DIV0 register, and the UART0/1/2 clock could be further divided by the SYS_CLK_DIV2 register.

The clock output by the PLL is used as the ADC clock (typical frequency is 96MHz) after being divided by 2/4/8 controlled by SYS_AFE_REG3.ADCCLKSEL, which is ACLK.

The 32kHz RC oscillator generates an LSI clock (LCLK), which is mainly used for the WDT working clock, as well as part of the system control, reset filtering and so on.

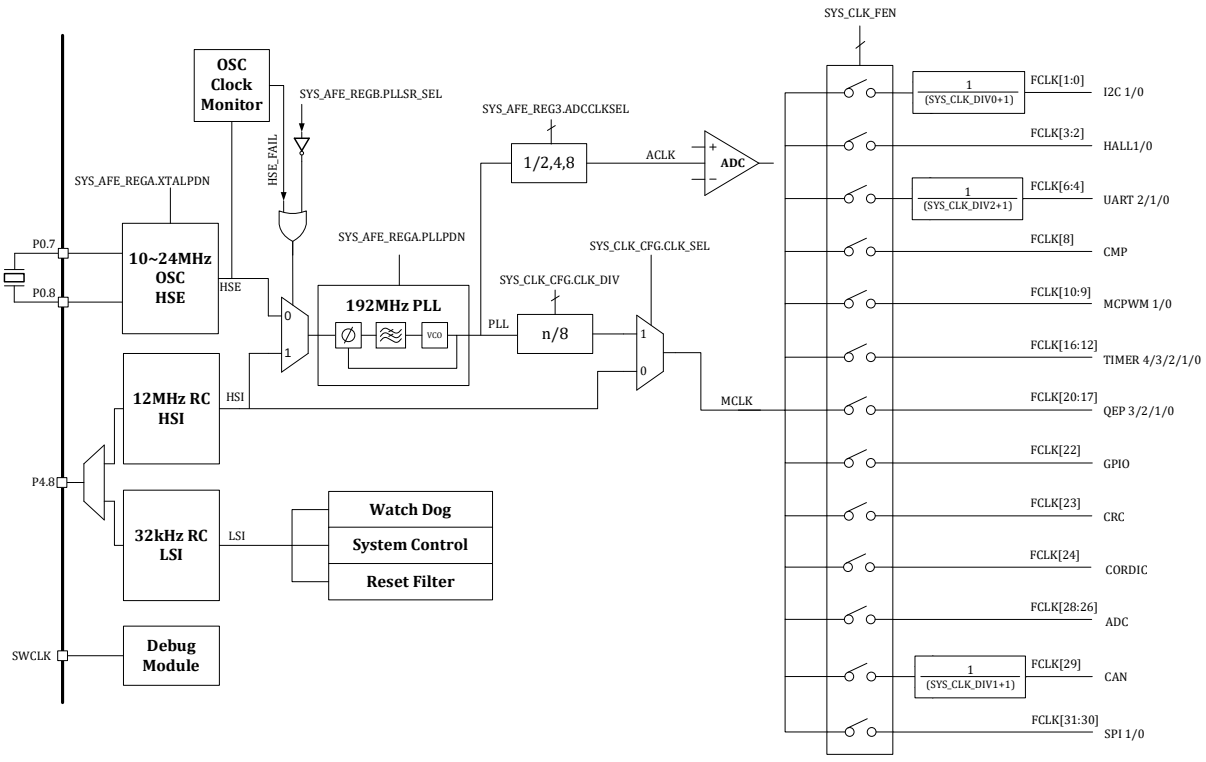


Figure 6-1 Clock Source

To ensure the system reliability, the clock system has a mechanism to prevent the clock from being shut down by mistake. For example, when the PLL is used as the main clock, the PLL cannot be turned off, and the HSI, which is the reference clock, cannot be turned off by software; once powered on, the 32kHz LSI clock start operating and cannot be turned off. SWCLK is provided by the debugger, and the clock frequency depends on the settings on the debug interface of the host computer.

To facilitate debugging and factory calibration, the high-speed RC oscillator HSI and low-speed

clock LSI can be output through chip pins by setting the second function of GPIO.

6.2 Reset

6.2.1 Reset Source

The reset source of the chip includes hardware reset and software reset.

6.2.1.1 Hardware reset

As shown in Table 6-3 , the system has four Hardware Reset Sources. The resets generated are all chip global resets. After the reset, the chip program counter returns to address 0, and all registers are restored to their default values. All four hardware reset sources are active low.

Table 6-3 System Reset Source

Name	Source	Description
LPORn	Internal 1.5V power management	Monitor 1.5V digital power supply, reset when it's below 1.25V
HPORn	Internal 3.3V power management	Monitor 3.3V digital power supply, reset when it's below 2.1V
RSTn	External Buttons	External RC reset circuit
IWDTn	Hardware watchdog	Count with LSI clock. If not feed the watchdog, then it will reset the CPU at a regular time, and the reset interval is configurable
WWDTn	Window watchdog	Count with system high-speed clock. If not feed the watchdog, then it will reset the CPU at a regular time, and the reset interval is configurable

6.2.1.1.1 Hardware Reset Architecture

LPORn/HPORn is an internal analog circuit, and RSTn is an external key. IWDTn is a reset signal of independent watchdog. It uses the 32kHz LSI clock and is enabled by default after the chip is powered on. The reset signal can be generated even when crystal oscillator stops or PLL is not working. WWDTn is the reset signal of window watchdog and counts with the system high-speed clock (usually the PLL master clock). The WWDTn is mainly used in application scenarios that require precise reset intervals. IWDTn and WWDTn signals will be disabled in debugging mode, which is configured by SYS_DBG_CFG.

After filtering and broadening preprocessing, all reset signals form a global reset signal to reset the whole chip. Flash and RAM memory content are not affected by reset.

Reset pulse with the P0.6 pin width of less than 32us will be filtered. It is required that the reliable reset width is greater than 200us.

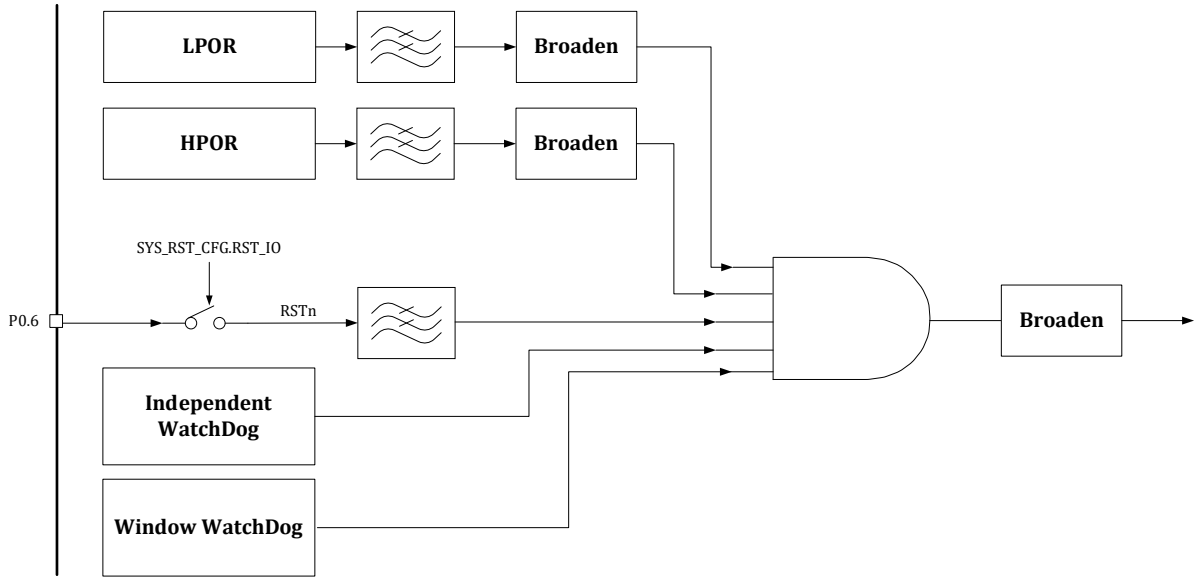


Figure 6-2 Reset Architecture

6.2.1.1.2 Hardware Reset Records

In the PMU module, AON_EVT_RCD register is used to save hardware reset and other events. When a hardware reset occurs, AON_EVT_RCD corresponds to the position bit. The AON_EVT_RCD register itself can only be reset by the LPORn reset signal; it can only clear the AON_EVT_RCD record by writing 0xCA40 to the AON_EVT_RCD register. With the reset record, we can easily understand whether and what kind of reset has occurred.

6.2.1.2 Software Reset

CPU soft reset can return the Program Counter (PC) to address 0, but it has no effect on the registers in all peripherals.

In the IDE (IDE: Integrated Development Environment) debug mode, clicking "Reset" has the same effect as "CPU Soft Reset", which will only return the PC to address 0, and do not affect the registers in all peripherals. However, if a soft reset of the peripheral module is performed in the bootloader, the peripheral register will be reset to the default value. For further details, please contact the chip vendor.

Some peripheral modules have a soft reset, which is performed by using the SYS_SFT_RST register. Write the corresponding bit to the register, restore the module state machine to its initial state, and restore the module register to the default value, see 6.3.9 for details.

6.2.2 Reset Scope

Table 6-4 Reset Scope

Reset Source	Scope
LPORn	Internal 1.5V power management, global reset
HPOR	Internal 3.3V power management, global reset, except very

	few registers
RSTn	External button, global reset, except very few registers
IWDTn	Independent watchdog, global reset, except the independent watchdog module and very few registers
WWDTn	Window watchdog, global reset, except the independent watchdog module and very few registers
SYS_SFT_RST.SPI1_ SFT_RST	SPI1
SYS_SFT_RST.SPI0_ SFT_RST	SPI0
SYS_SFT_RST.CAN_ SFT_RST	CAN
SYS_SFT_RST.ADC2_ SFT_RST	ADC2 digital interface module
SYS_SFT_RST.ADC1_ SFT_RST	ADC1 digital interface module
SYS_SFT_RST.ADC0_ SFT_RST	ADC0 digital interface module
SYS_SFT_RST.CORDIC_ SFT_RST	CORDIC
SYS_SFT_RST.CRC_ SFT_RST	CRC
SYS_SFT_RST.GPIO_ SFT_RST	GPIO
SYS_SFT_RST.QEP3_SFT_RST	QEP3
SYS_SFT_RST.QEP2_SFT_RST	QEP2
SYS_SFT_RST.QEP1_SFT_RST	QEP1
SYS_SFT_RST.QEP0_SFT_RST	QEP0
SYS_SFT_RST.TIMER4_ SFT_RST	TIMER4
SYS_SFT_RST.TIMER3_ SFT_RST	TIMER3
SYS_SFT_RST.TIMER2_ SFT_RST	TIMER2
SYS_SFT_RST.TIMER1_ SFT_RST	TIMER1
SYS_SFT_RST.TIMER0_ SFT_RST	TIMER0
SYS_SFT_RST.MCPWM1_ SFT_RST	MCPWM1
SYS_SFT_RST.MCPWM0_ SFT_RST	MCPWM0
SYS_SFT_RST.CMP_ SFT_RST	Comparator digital interface module
SYS_SFT_RST.UART2_ SFT_RST	UART2
SYS_SFT_RST.UART1_SFT_RST	UART1
SYS_SFT_RST.UART0_SFT_RST	UART0
SYS_SFT_RST.HALL1_SFT_RST	HALL1
SYS_SFT_RST.HALL0_SFT_RST	HALL0
SYS_SFT_RST.I2C1_SFT_RST	Updated I2C
SYS_SFT_RST.I2C0_SFT_RST	I2C0
NVIC_SystemReset();	"CPU Soft Reset" only resets CPU core registers and returns the PC to address 0, with all peripheral register values unchanged.

The SYS_IO_CFG.RST_IO controls whether P0[6] is used as GPIO or as an external reset pin, and the reset record register AON_EVT_RCD is reset only by LPOR.

A global reset resets all chip registers, including the CPU core registers and all peripheral registers except the very few registers described above.

As "CPU Soft Reset" only resets CPU core registers but not peripheral registers, it is recommended to reset the peripheral registers by way of powering off and powering on again or resetting externally after reburning the program.

Flash and SRAM memory content are not affected by reset.

6.3 Register

6.3.1 Address Allocation

The base address of the system module register is 0x40019800.

Table 6-5 System Control Register

Name	Offset	Description
SYS_CLK_CFG	0x80	Clock control register
SYS_IO_CFG	0x84	IO control register
SYS_DBG_CFG	0x88	Debug control register
SYS_CLK_DIV0	0x90	Peripheral clock divider register 0
SYS_CLK_DIV1	0x94	Peripheral clock divider register 1
SYS_CLK_DIV2	0x98	Peripheral clock divider register 2
SYS_CLK_FEN	0x9C	Peripheral clock gating register
SYS_SFT_RST	0xA4	Soft reset register
SYS_PROTECT	0xA8	Write protection register
SYS_CACHE_CFG	0xAC	Cache control register
SYS_MEM_CFG	0xC4	Memory control register

6.3.2 Clock Control Register (SYS_CLK_CFG)

Address: 0x4001_9880

Reset value: 0x0

Table 6-6 Clock Control Register (SYS_CLK_CFG)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		XTAL_FAILED		XTAL_FAIL		CLK_SEL		CLK_DIV							
		RW1C		RO		RW		RW							
		1		0		0		0							

Bit field	Bit Name	Description
[31:14]		Unused

[13]	HSE_FAILED	Whether the oscillator has stopped after being turned on If the oscillator has stopped, the position but is still recorded as 1 even if the oscillator continues.
[12]	HSE_FAIL	Whether the oscillator has stopped currently 0: The external oscillator works normally 1: The external oscillator stops
[11:10]		Unused
[9:8]	CLK_SEL	Source selection signal for system clock MCLK. HRC is selected by default. 0: HRC 1: PLL 2: LRC 3: XTAL Note that PLL/XTAL is turned off by default after power on, and should be enabled by software.
[7:0]	CLK_DIV	PLL output frequency division control. The default value is: 0x00: 1/8 frequency division 0x01: 1/8 frequency division 0x11: 1/4 frequency division 0x55: 1/2 frequency division 0xFF: 1/1 frequency division Other configurations are not recommended.

When CLK_SEL = 0, MCLK chooses the CLK_HS clock, and the division factor of SYS_CLK_CFG [7:0] is invalid. The final output clock frequency is CLK_HS clock frequency. According to SYS_AFE_REGB.PLLSR_SEL, when SYS_AFE_REGB.PLLSR_SEL=0, CLK_HS is the internal 12MHz RCH clock HSI of the chip; and when SYS_AFE_REGB.PLLSR_SEL=1, CLK_HS is the oscillator.

6.3.3 IO Control Register (SYS_IO_CFG)

Address: 0x4001_9884

Reset value: 0x02

Table 6-7 IO Control Register (SYS_IO_CFG)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								TRACEMUX	SWDMUX	RST_IO				IO_DS	
								RW	RW	RW				RW	
								0	0	0				2	

Bit field	Bit Name	Description
[31:8]		Unused

[7]	TRACEMUX	TRACE multiplexing control signal. The default value is TRACE 0: P1[3:0], P0[15:14] is used as the normal GPIO 1: P1[3] is multiplexed as SWV, {P0[15], P1[0:2]} is multiplexed as TRACEDATA[3:0], and P0[14] is multiplexed as TRACECLK
[6]	SWDMUX	JTAG/SWD multiplexing control signal. The default value is JTAG/ SWD 0: IO is used as JTAG/SWD P0.9--->nTRST P0.10--->SWDIOTMS P0.11 --->TDI P0.12--->SWCLKTCLK P0.13--->TDO 1: P0[13:9] is used as the normal GPIO
[5]	RST_IO	RSTn/ P 0.6 multiplexing control signal. The default value is RSTn 0:RSTn 1:P0.6 Note that the value is RSTn by default after powering on. The RSTn function may become invalid when the RSTn is enabled in subsequent software
[4]	POR_FIL	Not available to users The register uses SYS_AFE_REG6[11] as reset signal. Therefore, it is required to configure SYS_AFE_REG6[11]=1 first, and then enable POR filter, which is valid for both LPOR and HPOR
[3:2]		Unused
[1:0]	IO_DS	IO drive level selection 0: 4.5mA 1: 9mA 2: 13.5mA 3: 18mA The default drive level is 13.5mA after powering on

For the sake of safety, SWD/Jtag IO cannot be switched to GPIO within 30 ms after powering on. It can only be used as SWD/Jtag to ensure that the chip can be controlled by the debugger/downloader during the initial power-on phase. Even if the software rewrites the SYS_IO_CFG.SWDMUX, it will take effect after 30 ms. This means: Within 30 ms after powering on, the software can write 1'b0 to SYS_IO_CFG.SWDMUX, but the value returned is still 1'b1; if reading again after 30ms, the value returned will be 1'b0.

Note that if SWD/Jtag IO is switched to GPIO, you can only use the original downloader to erase the download unless the application is specifically designed.

6.3.4 Debug Control Register (SYS_DBG_CFG)

Address: 0x4001_9888

Reset value: 0x40

Table 6-8 Debug Control Register (SYS_DBG_CFG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SW_IRQ_TRIG															
W															
0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SW_IRQ	SFT_RST_PERI		DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP	DBG_TIM1_STOP	DBG_TIM0_STOP			DBG_IWDG_STOP	DBG_WWDG_STOP		DBG_STDBY	DBG_STOP	DBG_SLP
RW1C	RW		RW	RW	RW	RW	RW			RW	RW		RW	RW	RW
0	0		0	0	0	0	0			0	0		0	0	0

Bit field	Bit Name	Description
[31:16]	SW_IRQ_TRIG	Write 0x5AA5 to this bit segment to trigger a software interrupt, with SW_IRQ set to 1
[15]	SW_IRQ	Software interrupt flag. The interrupt number is 30, and write 1 to clear.
[14]	SFT_RST_PERI	Whether Debug Soft Reset resets all peripheral registers except Flash/SYS_AFE
[13]		Unused
[12]	DBG_TIM4_STOP	Timer4 stops in CPU halt state under debug mode 1: Timer4 stops counting in CPU halt state 0: Timer4 continues counting in CPU halt state
[11]	DBG_TIM3_STOP	Timer3 stops in CPU halt state under debug mode 1: Timer3 stops counting in CPU halt state 0: Timer3 continues counting in CPU halt state
[10]	DBG_TIM2_STOP	Timer2 stops in CPU halt state under debug mode 1: Timer2 stops counting in CPU halt state 0: Timer2 continues counting in CPU halt state
[9]	DBG_TIM1_STOP	Timer1 stops in CPU halt state under debug mode 1: Timer1 stops counting in CPU halt state 0: Timer1 continues counting in CPU halt state
[8]	DBG_TIM0_STOP	Timer0 stops in CPU halt state under debug mode 1: Timer0 stops counting in CPU halt state 0: Timer0 continues counting in CPU halt state

[7:6]		Unused
[5]	DBG_IWDG_STOP	Independent watchdog stops in CPU halt state under debug mode 1: Independent watchdog stops counting in CPU halt state 0: Independent watchdog continues counting in CPU halt state
[4]	DBG_WWDG_STOP	Window watchdog stops in CPU halt state under debug mode 1: Window watchdog stops counting in CPU halt state 0: Window watchdog continues counting in CPU halt state
[3]		Unused
[2]	DBG_STDBY	Debug Standby mode 0: (FCLK=Off, HCLK=Off) The whole digital circuit enters the power off state (except PMU). From a software perspective, exiting Standby is the same as power-on reset, except that AON_EVT_RCD indicates Sleep/Wake and other status bits 1: (FCLK=On, HCLK=On). If DBG_STDBY is set to 1, the digital part will not power off, and the internal high-speed clock will not close. However, PMU will still generate a reset signal when exiting the sleep state and refetch the command for implementation, making the implementation process same as when DBG_STDBY is not set
[1]	DBG_STOP	Debug STOP mode 0: (FCLK=Off, HCLK=Off). In STOP mode, the clock management mode turns off HCLK and FCLK, i.e. the processor clock and all peripheral clocks. When exiting the STOP mode, the clock management module does not reset and restores the clock settings before the STOP mode. In STOP mode, all peripheral registers are retained and can restore without reconfiguration. 1: (FCLK=On, HCLK=On). If DBG_STOP is set to 1, HCLK and FCLK continue to run in STOP mode.
[0]	DBG_SLP	Debug SLEEP mode 0: (FCLK=On, HCLK=Off). In SLEEP mode, FCLK is used as the clock of all peripherals and not turned off; HCLK is used as the CPU clock and turned off. In SLEEP mode, only the CPU clock is suspended, while all peripherals, including the system clock management module, remain in their original configuration. Therefore, after exiting the SLEEP mode, the software does not need to reconfigure the peripheral registers. 1: (FCLK=On, HCLK=On). If DBG_SLP is set to 1, HCLK will not be turned off after entering the SLEEP mode. The __WFI()/__WFE() command can set the processor to enter the SLEEP mode

6.3.5 Peripheral Clock Divider Register 0 (SYS_CLK_DIV0)

Address: 0x4001_9890

Reset value: 0x0

Table 6-9 Peripheral Clock Divider Register 0 (SYS_CLK_DIV0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV0															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	DIV0	I2C0/I2C1 clock=MCLK/(CLK_DIV0+1). MCLK is determined by the SYS_CLK_CFG division factor.

6.3.6 Peripheral Clock Divider Register 1 (SYS_CLK_DIV1)

Address: 0x4001_9894

Reset value: 0x0

Table 6-10 Peripheral Clock Divider Register 1 (SYS_CLK_DIV1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV0															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	DIV1	CAN clock=MCLK/(CLK_DIV1+1). MCLK is determined by the SYS_CLK_CFG division factor.

6.3.7 Peripheral Clock Divider Register 2 (SYS_CLK_DIV2)

Address: 0x4000_0098

Reset value: 0x0

Table 6-11 Peripheral Clock Divider Register 2 (SYS_CLK_DIV2)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV2															
RW															

0

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	DIV2	UART clock=MCLK/(CLK_DIV2+1). After UART0/UART1/UART2 sharing this frequency division configuration, the baud rate is further divided by the UART baud rate register, where MCLK is determined by the SYS_CLK_CFG division factor.

6.3.8 Peripheral Clock Gating Register (SYS_CLK_FEN)

Address: 0x4001_989C

Reset value: 0x0

Table 6-12 Peripheral Clock Gating Register (SYS_CLK_FEN)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPI1_CLK_EN	SPI0_CLK_EN	CAN_CLK_EN	ADC2_CLK_EN	ADC1_CLK_EN	ADC0_CLK_EN	FMAC_CLK_EN	CORDIC_CLK_EN	CRC_CLK_EN	GPIO_CLK_EN		QEP3_CLK_EN	QEP2_CLK_EN	QEP1_CLK_EN	QEP0_CLK_EN	TIMER4_CLK_EN
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW		RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0		0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMER3_CLK_EN	TIMER2_CLK_EN	TIMER1_CLK_EN	TIMER0_CLK_EN		MCPWM1_CLK_EN	MCPWM0_CLK_EN	CMP_CLK_EN		UART2_CLK_EN	UART1_CLK_EN	UART0_CLK_EN	HALL1_CLK_EN	HALL0_CLK_EN	I2C1_CLK_EN	I2C0_CLK_EN
RW	RW	RW	RW		RW	RW	RW		RW	RW	RW	RW	RW	RW	RW
0	0	0	0		0	0	0		0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31]	SPI1_CLK_EN	SPI1 clock control. It is turned off by default. 1: Enable SPI0 clock 0: Disable SPI0 clock
[30]	SPI0_CLK_EN	SPI0 clock control. It is turned off by default. 1: Enable SPI0 clock 0: Disable SPI0 clock
[29]	CAN_CLK_EN	CAN clock control. It is turned off by default. 1: Enable CAN clock 0: Disable CAN clock

[28]	ADC2_CLK_EN	ADC2 clock control. It is turned off by default. 1: Enable ADC2 clock 0: Disable ADC2 clock
[27]	ADC1_CLK_EN	ADC1 clock control. It is turned off by default. 1: Enable ADC1 clock 0: Disable ADC1 clock
[26]	ADC0_CLK_EN	ADC0 clock control. It is turned off by default. 1: Enable ADC0 clock 0: Disable ADC0 clock
[25]		Unused
[24]	CORDIC_CLK_EN	CORDIC clock control. It is turned off by default. 1: Enable CORDIC clock 0: Disable CORDIC clock
[23]	CRC_CLK_EN	CRC clock control. It is turned off by default. 1: Enable CRC clock 0: Disable CRC clock
[22]	GPIO_CLK_EN	GPIO clock control. It is turned off by default. 1: Enable GPIO clock 0: Disable GPIO clock
[21]		Unused
[20]	QEP3_CLK_EN	QEP3 clock control. It is turned off by default. 1: Enable QEP3 clock 0: Disable QEP3 clock
[19]	QEP2_CLK_EN	QEP2 clock control. It is turned off by default. 1: Enable QEP2 clock 0: Disable QEP2 clock
[18]	QEP1_CLK_EN	QEP1 clock control. It is turned off by default. 1: Enable QEP1 clock 0: Disable QEP1 clock
[17]	QEP0_CLK_EN	QEP0 clock control. It is turned off by default. 1: Enable QEP0 clock 0: Disable QEP0 clock
[16]	TIMER4_CKL_EN	TIMER4 clock control. It is turned off by default. 1: Enable TIMER4 clock 0: Disable TIMER4 clock
[15]	TIMER3_CKL_EN	TIMER3 clock control. It is turned off by default. 1: Enable TIMER3 clock 0: Disable TIMER3 clock
[14]	TIMER2_CKL_EN	TIMER2 clock control. It is turned off by default. 1: Enable TIMER2 clock 0: Disable TIMER2 clock
[13]	TIMER1_CKL_EN	TIMER1 clock control. It is turned off by default. 1: Enable TIMER1 clock

		0: Disable TIMER1 clock
[12]	TIMER0_CKL_EN	TIMER0 clock control. It is turned off by default. 1: Enable TIMER0 clock 0: Disable TIMER0 clock
[11]		Unused
[10]	MCPWM1_CKL_EN	MCPWM1 clock control. It is turned off by default. 1: Enable MCPWM1 clock 0: Disable MCPWM1 clock
[9]	MCPWM0_CKL_EN	MCPWM0 clock control. It is turned off by default. 1: Enable MCPWM0 clock 0: Disable MCPWM0 clock
[8]	CMP_CKL_EN	CMP clock control. It is turned off by default. 1: Enable CMP clock 0: Disable CMP clock
[7]		Unused
[6]	UART2_CLK_EN	UART2 clock control. It is turned off by default. 1: Enable UART2 clock 0: Disable UART2 clock
[5]	UART1_CLK_EN	UART1 clock control. It is turned off by default. 1: Enable UART1 clock 0: Disable UART1 clock
[4]	UART0_CLK_EN	UART0 clock control. It is turned off by default. 1: Enable UART0 clock 0: Disable UART0 clock
[3]	HALL1_CLK_EN	HALL1 clock control. It is turned off by default. 1: Enable HALL1 clock 0: Disable HALL1 clock
[2]	HALL0_CLK_EN	HALL0 clock control. It is turned off by default. 1: Enable HALL0 clock 0: Disable HALL0 clock
[0]	I2C1_CLK_EN	I2C1 clock control. It is turned off by default. 1: Enable I2C1 clock 0: Disable I2C1 clock
[0]	I2C0_CLK_EN	I2C0 clock control. It is turned off by default. 1: Enable I2C0 clock 0: Disable I2C0 clock

Note that the clock gating of each module only controls the working clock of each module. Even if the clock of each module is not turned on, it does not affect the software to access the configuration registers of each module.

6.3.9 Soft Reset Register (SYS_SFT_RST)

Address: 0x4001_98A4

Reset value: 0x0

Table 6-13 Soft Reset Register (SYS_SFT_RST)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPI1_SFT_RST	SPI0_SFT_RST	CAN_SFT_RST	ADC2_SFT_RST	ADC1_SFT_RST	ADC0_SFT_RST	FMAC_SFT_RST	CORDIC_SFT_RST	CRC_SFT_RST	GPIO_SFT_RST	DMA_SFT_RST	QEP3_SFT_RST	QEP2_SFT_RST	QEP1_SFT_RST	QEP0_SFT_RST	TIMER4_SFT_RST
WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMER3_SFT_RST	TIMER2_SFT_RST	TIMER1_SFT_RST	TIMER0_SFT_RST		MCPWM1_SFT_RST	MCPWM0_SFT_RST	CMP_SFT_RST		UART2_SFT_RST	UART1_SFT_RST	UART0_SFT_RST	HALL1_SFT_RST	HALL0_SFT_RST	I2C1_SFT_RST	I2C0_SFT_RST
WO	WO	WO	WO		WO	WO	WO		WO	WO	WO	WO	WO	WO	WO
0	0	0	0		0	0	0		0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31]	SPI1_ SFT_RST	SPI1 soft reset. It is not reset by default 1: Reset SPI0 module 0: Release SPI0 module
[30]	SPI0_ SFT_RST	SPI0 soft reset. It is not reset by default 1: Reset SPI0 module 0: Release SPI0 module
[29]	CAN_ SFT_RST	CAN soft reset. It is not reset by default 1: Reset CAN module 0: Release CAN module
[28]	ADC2_ SFT_RST	ADC2 soft reset. It is not reset by default 1: Reset ADC2 module 0: Release ADC2 module
[27]	ADC1_ SFT_RST	ADC1 soft reset. It is not reset by default 1: Reset ADC1 module 0: Release ADC1 module
[26]	ADC0_ SFT_RST	ADC0 soft reset. It is not reset by default 1: Reset ADC0 module 0: Release ADC0 module
[25]		Unused

[24]	CORDIC_ SFT_RST	CORDIC soft reset. It is not reset by default 1: Reset CORDIC module 0: Release CORDIC module
[23]	CRC_ SFT_RST	CRC soft reset. It is not reset by default 1: Reset CRC module 0: Release CRC module
[22]	GPIO_ SFT_RST	GPIO soft reset. It is not reset by default 1: Reset GPIO module 0: Release GPIO module
[21]	DMA_SFT_RST	DMA soft reset. It is not reset by default 1: Reset DMA module 0: Release DMA module
[20]	QEP3_SFT_RST	QEP3 soft reset. It is not reset by default 1: Reset QEP3 module 0: Release QEP3 module
[19]	QEP2_SFT_RST	QEP2 soft reset. It is not reset by default 1: Reset QEP2 module 0: Release QEP2 module
[18]	QEP1_SFT_RST	QEP1 soft reset. It is not reset by default 1: Reset QEP1 module 0: Release QEP1 module
[17]	QEP0_SFT_RST	QEP0 soft reset. It is not reset by default 1: Reset QEP0 module 0: Release QEP0 module
[16]	TIMER4_ SFT_RST	TIMER4 soft reset. It is not reset by default 1: Reset TIMER4 module 0: Release TIMER4 module
[15]	TIMER3_ SFT_RST	TIMER3 soft reset. It is not reset by default 1: Reset TIMER3 module 0: Release TIMER3 module
[14]	TIMER2_ SFT_RST	TIMER2 soft reset. It is not reset by default 1: Reset TIMER2 module 0: Release TIMER2 module
[13]	TIMER1_ SFT_RST	TIMER1 soft reset. It is not reset by default 1: Reset TIMER1 module 0: Release TIMER1 module
[12]	TIMER0_ SFT_RST	TIMER0 soft reset. It is not reset by default 1: Reset TIMER0 module 0: Release TIMER0 module
[11]		Unused
[10]	MCPWM1_ SFT_RST	MCPWM1 soft reset. It is not reset by default 1: Reset MCPWM1 module 0: Release MCPWM1 module

[9]	MCPWM0_ SFT_RST	MCPWM0 soft reset. It is not reset by default 1: Reset MCPWM0 module 0: Release MCPWM0 module
[8]	CMP_ SFT_RST	CMP soft reset. It is not reset by default 1: Reset CMP module 0: Release CMP module
[7]		Unused
[6]	UART2_ SFT_RST	UART2 soft reset. It is not reset by default 1: Reset UART2 module 0: Release UART2 module
[5]	UART1_SFT_RST	UART1 soft reset. It is not reset by default 1: Reset UART1 module 0: Release UART1 module
[4]	UART0_SFT_RST	UART0 soft reset. It is not reset by default 1: Reset UART0 module 0: Release UART0 module
[3]	HALL1_SFT_RST	HALL1 soft reset. It is not reset by default 1: Reset HALL1 module 0: Release HALL1 module
[2]	HALL0_SFT_RST	HALL0 soft reset. It is not reset by default 1: Reset HALL0 module 0: Release HALL0 module
[1]	I2C1_SFT_RST	I2C1 soft reset. It is not reset by default 1: Reset I2C1 module 0: Release I2C1 module
[0]	I2C0_SFT_RST	I2C0 soft reset. It is not reset by default 1: Reset I2C0 module 0: Release I2C0 module

Please note that the module's soft reset will remain in the reset state after writing 1 to the corresponding bit of SYS_SFT_RST, and write 0 again to release the reset state. As the reset signal will be widened internally, wait at least 1 bus cycle after releasing soft reset before reading and writing the peripherals.

You can insert __asm("NOP"); to realize waiting.

6.3.10 Write Protection Register (SYS_PROTECT)

Address: 0x4001_98A8

Reset value: 0x0

Table 6-14 Write Protection Register (SYS_PROTECT)

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

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	PSW	Except for SYS_AFE_DACx, SYS_AFE_DACx_AMC, and SYS_AFE_DACx_DC (x=0,1), other registers are write-protected, and should write a password in advance to release the write-protection. Write 0x7A83 to enable the register's write operation. Write other values to prohibit registers'write operations.

6.3.11 Cache Control Register (SYS_CAHCE_CFG)

Address: 0x4001_98AC

Reset value: 0x0

Table 6-15 Cache Control Register (SYS_CACHE_CFG)

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

Bit field	Bit Name	Description
[31:1]		Unused
[0]	EN	Cache enabling, 1: Enable Cache, 0: Disable Cache

6.3.12 Memory Control Register (SYS_MEM_CFG)

Address: 0x4001_98C4

Reset value: 0x0

Table 6-16 Memory Control Register (SYS_MEM_CFG)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						REMAP		PCK_EN				PCK_ERR			
						RW		RW				RW1C			
						0		0				0			

Bit field	Bit Name	Description
[31:10]		Unused
[9:8]	REMAP	Remap address space. The address space before mapping the SRAM1/2/3 to 0x1FFFFFFF is used as Code RAM, which can be used to store critical code and accelerate the execution of critical code due to faster RAM access. 0: SRAM0/1/2/3 are used as Data RAM, without Code RAM 1: SRAM0/1/2 are used as Data RAM, and SRAM3 is used as Code RAM 2: SRAM0/1 are used as Data RAM, and SRAM2/3 are used as Code RAM 3: SRAM0 is used as Data RAM, and SRAM1/2/3 are used as Code RAM For detailed address space, please refer to Chapter 3 Address Space
[7:4]	PCK_EN	SRAM3/2/1/0 memory parity check enabling
[3:0]	PCK_ERR	SRAM3/2/1/0 memory parity check error, 0: No error occurs, 1: Error occurs, and write 1 to clear

Only when the parity function is enabled, the parity bit can be set when the parity fails. A non-maskable interrupt NMI is generated when a parity error occurs.

6.3.13 SYS_FLSE

地址：0x4001_98D0

复位值：0x0

表 6-17 擦除控制寄存器 SYS_FLSE

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

位置	位名称	说明
[31:16]		未使用
[15:0]	FLSE	**FLASH erase protection register. The erase function of FLASH is finally enabled by writing 0x8FCA to this register. The erase function of FLASH cannot take effect when other values are written.To prevent false erasure, it is recommended that this register is not activated when the erase function is not used.**

6.3.14 SYS_FLSP

地址：0x4001_98D4

复位值：0x0

表 6-18 编程控制寄存器 SYS_FLSP

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

位置	位名称	说明
[31:16]		未使用
[15:0]	FLSP	**FLASH program protection register. Write 0x8F35 to this register for the FLASH programming function to finally take effect. The programming function of FLASH cannot be effective when other values are written.To prevent misprogramming, it is recommended that this register is not activated when the programming function is not used.**

7 Non-Volatile Memory

7.1 Overview

For LKS32MC45x series chips, its non-volatile memory includes two parts: On-Chip FLASH memory and its controller and Ext-Chip FLASH controller.

On-Chip FLASH memory includes three zones: NVR, MAIN and ROM. NVR is 2KB, and MAIN and ROM depend on the models.

- The main flash memory zone (MAIN) of On-Chip FLASH includes application programs and user data area.
- The read-only memory zone (ROM) of On-Chip FLASH includes application programs and user data area.
- The information storage zone (NVR) of On-Chip FLASH is reserved for users.

The ROM zone of On-Chip FLASH reserves a programming opportunity for users, and users can write their own programs into the ROM zone and lock them. MCU can execute programs in ROM, but cannot transfer the content in the zone.

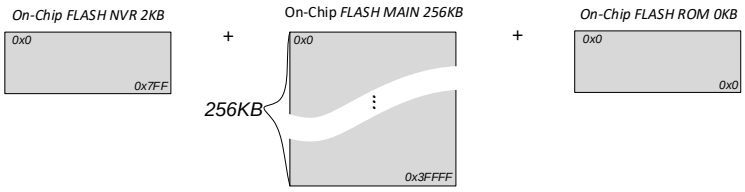
For LKS32MC45x series chips, On-Chip FLASH has seven combinations.

Ext-Chip FLASH controller supports a maximum of 16 MB of external SPI FLASH. The external SPI FLASH shall be purchased by customers, and the EXt-Flash controller of LKS32MC45x supports the mainstream external SPI FLASH on the market.

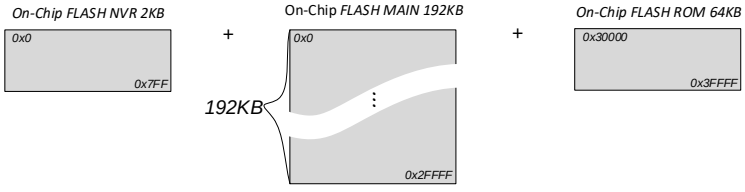
- The Ext-Chip FLASH controller only supports SPI FLASH and three data transmission modes, namely single-wire, dual-wire and four-wire.
- The Ext-Chip FLASH controller supports MCU to directly run the applications in the external SPI FLASH, and also supports to transfer the applications in the external SPI FLASH to the Chip for execution.

To improve the program execution efficiency of LKS32MC45x series chips and reduce the performance loss caused by the running speed of non-volatile memory lower than the system frequency, an additional Cache acceleration module is designed. The On-Chip Flash content can be prefetched and stored in Cache to accelerate its running efficiency; however, the Ext-Chip FLASH content cannot be stored in Cache and cannot be accelerated.

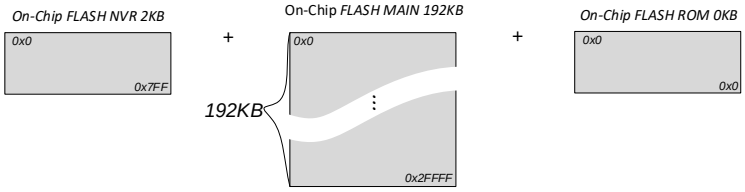
NVR 2KB/MAIN 256KB/ROM 0KB



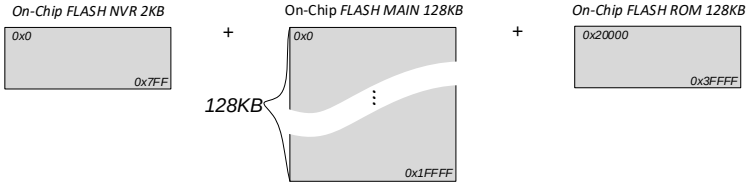
NVR 2KB/MAIN 192KB/ROM 64KB



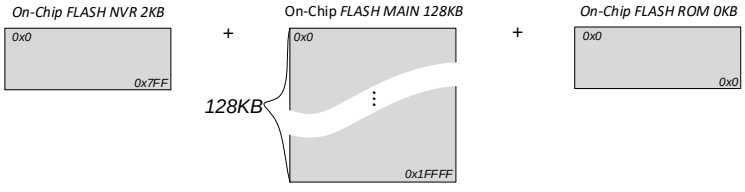
NVR 2KB/MAIN 192KB/ROM 0KB



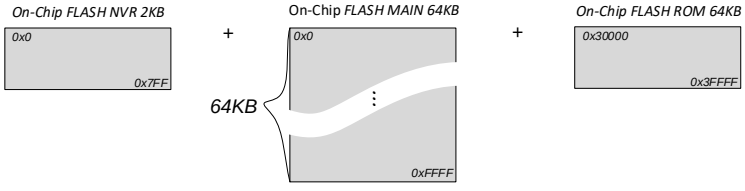
NVR 2KB/MAIN 128KB/ROM 128KB



NVR 2KB/MAIN 128KB/ROM 0KB



NVR 2KB/MAIN 64KB/ROM 64KB



NVR 2KB/MAIN 64KB/ROM 0KB

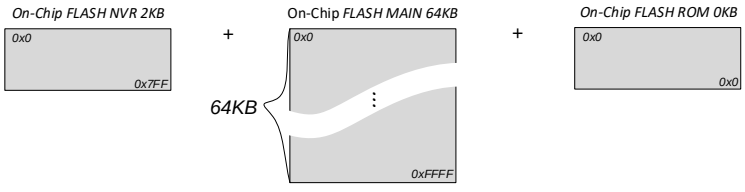


Figure 7-1 Space Division Block Diagram of On-Chip FLASH Memory

7.2 Features

The On-Chip Flash controller mainly performs relevant operations on the On-Chip FLASH memory, including:

- On-Chip FLASH reading, including reading operations of the NVR and MAIN. The locked ROM zone cannot be read.
- On-Chip FLASH programming, including programming operations of the NVR and MAIN. ROM does not support secondary programming.
- On-Chip FLASH erasing, supporting erase of Sector. NVR and MAIN support erase of Sector, but ROM does not support.
- On-Chip FLASH execution, that is, applications can directly run in MAIN and ROM, but cannot in NVR, which can only store data.
- On-Chip FLASH deep sleep, reducing the sleep power consumption of the chip.
- On-Chip FLASH memory content protection.
- On-Chip FLASH Cache acceleration, improving the overall operation efficiency of the chip.
- On-Chip FLASH control register access.

The Ext-Chip Flash controller mainly performs relevant operations on the Ext-Chip FLASH memory, including:

- Ext-Chip FLASH read.
- Ext-Chip FLASH programming.
- Ext-Chip FLASH erase.
- Ext-Chip FLASH control register access.

7.2.1 On-Chip FLASH Functional Description

The control module implements operations such as reset/read/write/erase/sleep of On-Chip FLASH memory. The following is the state transition diagram of the control module:

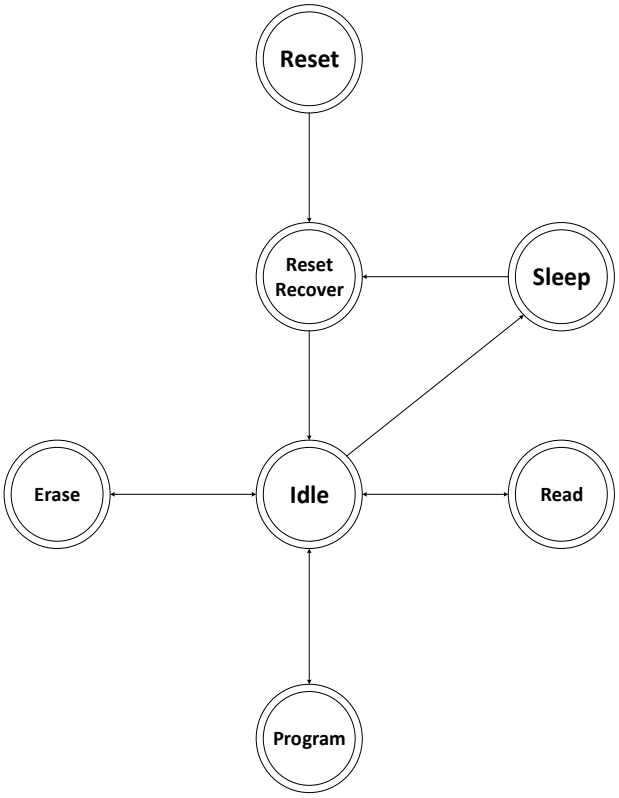


Figure 7-2 On-Chip FLASH Control State Transform Diagram

7.2.1.1 Reset

After the system is reset, it takes some times for the On-Chip FLASH recovery, thus to ensure the internal circuit stability of the On-Chip FLASH memory. After then, other operations could be performed on On-Chip FLASH. This recovery operation is automatically realized by hardware without software intervention.

7.2.1.2 Sleep

The sleep operation of On-Chip FLASH is divided into two parts: Standby and Deep Sleep. When no operations will be performed on On-Chip FLASH, On-Chip FLASH enters the StandBy state automatically (if Cache acceleration is turned on, this function is disabled). When the system executes Deep Sleep operation, it will trigger On-Chip FLASH to enter Deep Sleep, to further reduce power consumption. The operation of On-Chip FLASH entering Deep Sleep is completed by hardware automatically without software intervention.

When system is waked up by the outside world, the On-Chip FLASH will be waked up at the same time. After a period of recovery, On-Chip FLASH operations can be performed normally. This wake-up recovery operation is automatically completed by hardware without software intervention.

7.2.1.3 On-Chip FLASH Read

The read operation is the basic operation of On-Chip FLASH. The system can access the data in FLASH through two paths.

- MCU fetches instructions and numbers on On-Chip FLASH directly through I and D bus of AHB

bus. The bus width is 32 Bit and can only access (runs only, but cannot read) the data in MAIN and ROM. The hardware also provides a Cache acceleration function to improve the read efficiency of MCU.

- The MCU accesses the register of the controller through the S bus of AHB bus to read On-Chip FLASH internal data indirectly. Both the data in the MAIN and NVR can be accessed, while access to ROM is conditional; if a continuous reading is required, the hardware will accumulate the addresses automatically without updating the address register value each time. At this time, there is a situation of both I/D bus and S bus accessing the FLASH simultaneously. On-Chip FLASH gives priority to I/D bus access and S bus needs to wait for the I/D bus to be idle before completing the corresponding operations; once the S bus gains access to On-Chip FLASH, the I/D bus cannot seize it until the S bus access is complete.
- The register FSMC_REDY.READY indicates whether the current I/D bus completes the current read operation, 1 indicating completed and 0 uncompleted.
- The register FSMC_REDY.IREADY indicates whether the current S bus completes the current read operation, 1 indicating completed and 0 uncompleted.

The FLASH_CFG.REGION bit indicates which space is currently being accessed. See as follows:

Table 7-1 On-Chip FLASH Access Space Allocation

FSMC_ICFG.REGION	Zone of Access
0	MAIN zone and ROM zone*
1	NVR zone

*Note that ROM supports single programming by default. After programming, the zone is limited as XN (eXecute regioN) and cannot be read after being locked, nor it supports secondary programming and erasing.

The process for reading the internal data indirectly on On-Chip FLASH by accessing the register of the controller is as follows:

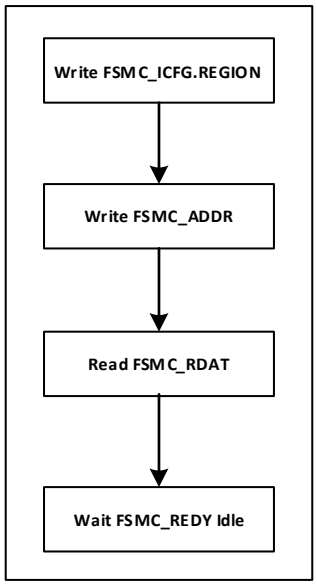


Figure 7-3 Flow Chart of On-Chip FLASH Indirect Read

7.2.1.4 On-Chip FLASH Programming

Programming refers to programming operations on the On-Chip FLASH memory bank. Generally, an erase operation (ensure that the corresponding programming background is the default value. Do not program the same address twice in a row) should be performed before data programming. And, the programming can only be performed by accessing the registers of the On-Chip FLASH controller. The programming width of LKS32MC45x chip is 128-bit, that is, writing 16 bytes at a time.

Programming is similar to reading. At this time, there is a situation of both I/D bus and S bus accessing the FLASH simultaneously, where the I/D bus reads the On-Chip FLASH data, and S bus performs programming. On-Chip FLASH gives priority to I/D bus access and S bus needs to wait for the I/D bus to be idle before completing the corresponding operations; once the S bus gains access to On-Chip FLASH, the I/D bus cannot seize it until the S bus access is complete

Different product models have different programming address space. The following table shows the address allocation for different combinations:

Table 7-2 On-Chip FLASH Programming Address Allocation

Combination	MAIN Area	ROM Area
Combination 1	0x0000 0000 - 0x0003 FFFF	0x0000 0000 - 0x0000 0000
Combination 2	0x0000 0000 - 0x0002 FFFF	0x0003 0000 - 0x0003 FFFF
Combination 3	0x0000 0000 - 0x0002 FFFF	0x0003 0000 - 0x0003 FFFF
Combination 4	0x0000 0000 - 0x0001 FFFF	0x0002 0000 - 0x0003 FFFF
Combination	0x0000 0000 - 0x0001 FFFF	0x0000 0000 - 0x0000 0000

5		
Combination 6	0x0000 0000 - 0x0000 FFFF	0x0003 0000 - 0x0003 FFFF
Combination 7	0x0000 0000 - 0x0000 FFFF	0x0000 0000 - 0x0000 0000

The specific programming process is:

- System register SYS_FLSP, write password to unlock programming protection
- On-Chip FLASH control register FSMC_ICFG, enable programming
- On-Chip FLASH address register FSMC_ADDR, write programming address
- On-Chip FLASH programming data register FSMC_WDAT0 to FSMC_WDAT3, write programming data
- On-Chip FLASH programming trigger register FSMC_WDAT, write trigger value to trigger programming
- Register FSMC_REDY.IREADY, indicate whether the current S bus completes the current read operation, 1 indicating completed and 0 uncompleted.

The process for On-chip FLASH programming by accessing the register of this controller is as follows:

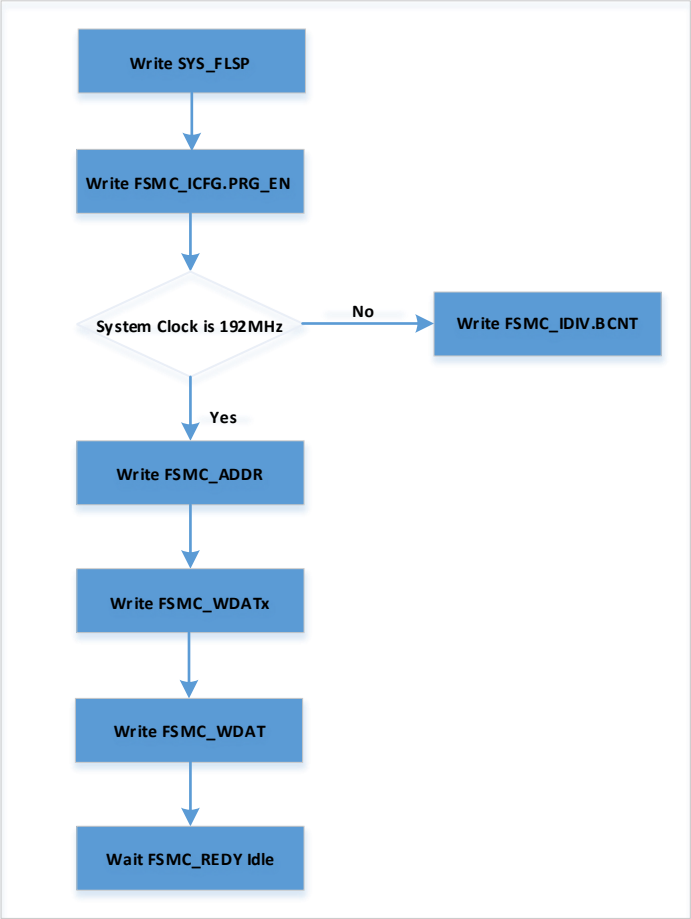


Figure 7-4 Flow Chart of On-Chip FLASH Programming

For the selection of operating frequency, please refer to the configuration of SYS_CLK_CFG. The absolute time of the On-Chip FLASH write/erase operation is fixed, and the count value corresponding to these absolute times should be saved in the On-Chip FLASH controller. The default value of FSMC_IDIV.BCNT0 and FSMC_IDIV.BCNT1 is the count value at 192MHz clock frequency; when the chip works at other frequencies, the value of FSMC_IDIV.BCNT0 and FSMC_IDIV.BCNT1 should be reset to achieve the count values of 96MHz/48MHz and 24MHz (other frequencies are not supported). In this way, the value obtained by multiplying the count value by the clock frequency is equal to a constant time. For the corresponding FSMC_IDIV.BCNT0 and FSMC_IDIV.BCNT1 values at different frequencies, please refer to the register description of this Chapter. Please note that only the values provided in the register description could be set to the FSMC_IDIV.BCNT0 and FSMC_IDIV.BCNT1, and other values are not available for writing into; otherwise, it may cause On-Chip FLASH programming/erasing failure. It is recommended to read first on the FSMC_IDIV, and then perform other operations by follow the OR/AND method.

When performing the On-Chip FLASH write/erase operation, FLASH is in busy state. If Cache is not hit, there will be no subsequent instruction/data to fetch MCU, and MCU will stop temporarily until the FLASH program/erase operation is finished. At this time, to ensure data consistency, the data stored in the Cache is invalid once performing the FLASH write/erase operation. You need to fetch the

command and data again.

Figure 7-4 only shows the flow of one programming. If continuous programming is required, set the FSMC_ICFG.ADR_INC before writing to the FSMC_ADDR register to enable the address auto-increment mode. After completing the current programming operation, the value of FSMC_ADDR will be added by 0x10 automatically. The operation of continuous reading is similar to this, but the auto-increment value is 0x04.

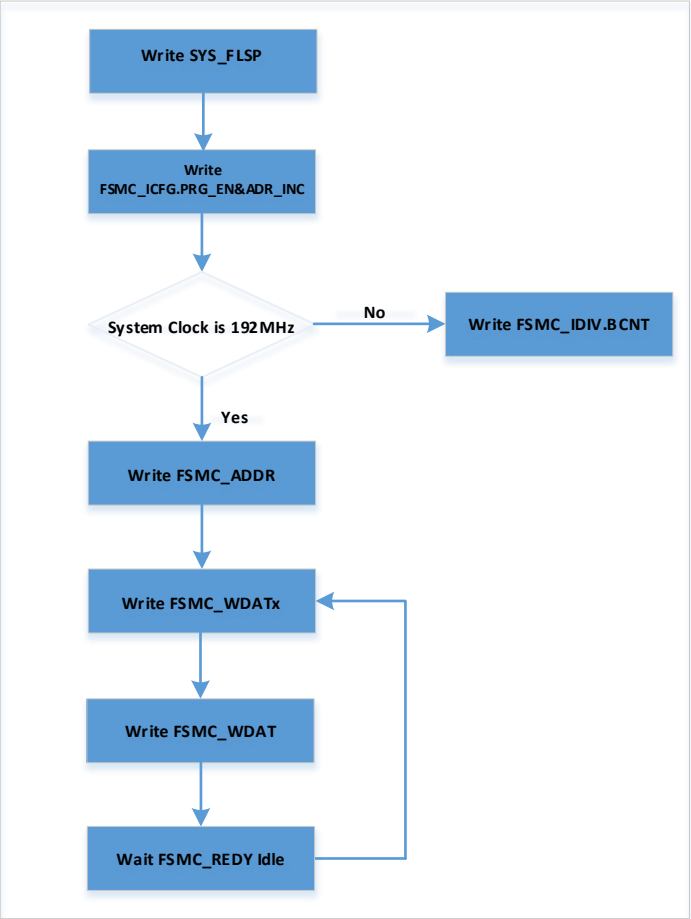


Figure 7-5 Flow Chart of On-Chip FLASH Programming

7.2.1.5 On-Chip FLASH Erase

The erase operation is the basic operation of On-Chip FLASH, which can only be achieved by accessing the registers of the On-Chip FLASH controller. The specific process is:

- System register SYS_FLSE, write password to unlock erase protection
- On-Chip FLASH, enable erase
- On-Chip FLASH address register FSMC_ADDR, write erase address
- On-Chip FLASH erase register FSMC_ERAS, trigger erase
- Register FSMC_REDY.IREADY, indicate whether the current S bus completes the current erase operation, 1 indicating completed and 0 uncompleted.

Programming is similar to reading. At this time, there is a situation of both I/D bus and S bus accessing the FLASH simultaneously, where the I/D bus reads the On-Chip FLASH data, and S bus performs erase. On-Chip FLASH gives priority to I/D bus access and S bus needs to wait for the I/D bus to be idle before completing the corresponding operations; once the S bus gains access to On-Chip FLASH, the I/D bus cannot seize it until the S bus access is complete

Erase refers to erase operations on the On-Chip FLASH memory bank. A Sector is 1024 bytes.

The following table is the address allocation space of Main (maximum) and Sector.

Table 7-3 MAIN Sector Address Allocation of On-Chip FLASH

Name	Address	Byte
Sector 0	0x0000 0000 - 0x0000 03FF	1024
Sector 1	0x0000 0400 - 0x0000 07FF	1024
Sector 2	0x0000 0800 - 0x0000 0BFF	1024
…	…	…
Sector256	0x0003 FC00 - 0x0003 FFFF	1024

The ROM area does not support erase;

and the NVR area can only realize Sector erasure.

Table 7-4 NVR Sector Address Allocation of On-Chip FLASH

Name	Address	Byte
Sector 0	0x0000 0000 - 0x0000 03FF	1024
Sector 1	0x0000 0400 - 0x0000 07FF	1024

The MAIN area can realize Sector erasure.

The flash erase operation flow is shown below.

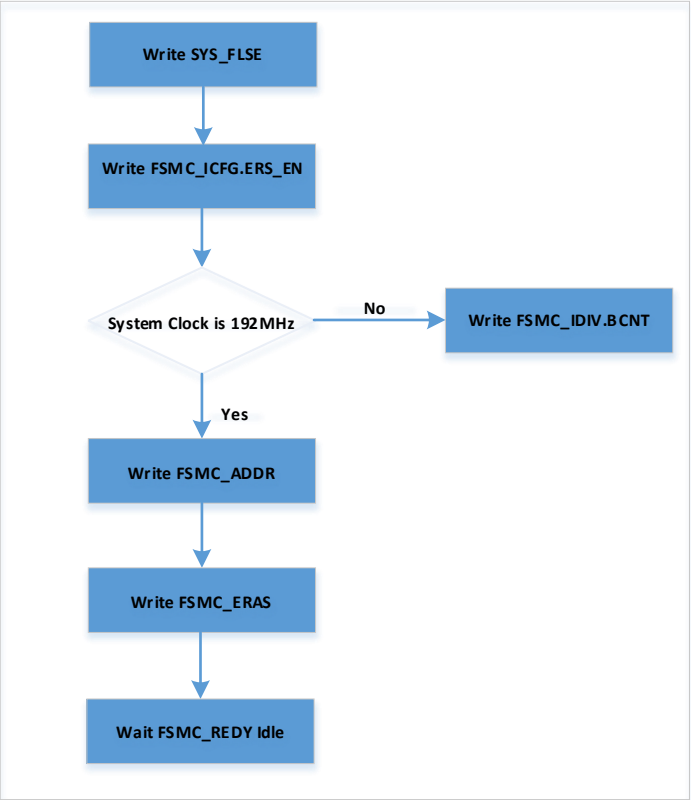


Figure 7-6 Flow Chart of On-Chip FLASH Erase

For Sector erasure, determine which Sector to be erased by FLASH_ADDR. Writing 0x7654DCBA to FSMC_ERAS triggers the erase operation.

For On-Chip FLASH memory bank, the default erase times is 100K. This module supports quick erase. Theoretically, the number of erases can be multiplied (erase with the minimum time). The quick erase mode supports Sector erasure. Compared with the normal erase operation, the quick erase process is as follows:

- System register SYS_FLSE, write password to unlock erase protection
- On-Chip FLASH erase, configure block erase and enable fast erase, and set the fast erase counter to 0
- On-Chip FLASH, enable erase
- On-Chip FLASH address register FSMC_ADDR, write erase address
- On-Chip FLASH erase register FSMC_ERAS, trigger erase
- Register FSMC_REDY.IREADY, indicate whether the current S bus completes the current erase operation, 1 indicating completed and 0 uncompleted.
- The user performs read all data corresponding to the current erase block address on On-Chip FLASH and determines whether all data are 1. If yes, it means the quick erase operation is successful; if not, it means the quick erase operation is failed, the value of the quick erase counter

will be increased by 1 and the erase operation will be triggered again. This operation will be repeated until successful erasure.

Note that the maximum value of the quick erase counter is 3. If the data corresponding to the current Sector address is still not all 1 after performing three times of erasure, it means that the On-Chip FLASH reaches the upper limit.

7.2.1.6 On-Chip FLASH Cache Acceleration

Due to the speed limitation of On-Chip FLASH memory bank, this operation cannot reach the maximum system speed. When reading the On-Chip FLASH, it takes more than 1 clock cycle to finish reading the data. In order to speed up the reading of data, the On-Chip FLASH controller adds a Cache function. The Cache operation can be turned on and off only by setting SYS_CACHE_CFG.EN.

Note that when performing On-Chip FLASH erase/programming operation, Cache will not refresh simultaneously, and the hardware will automatically configure the data stored in Cache to invalid, to ensure that there is no operation error caused by potential data mismatch.

7.2.1.7 On-Chip FLASH Protection

The protection policy for On-Chip FLASH memory is divided into two parts: overall On-Chip FLASH (including MAIN, NVR and ROM) protection policy and independent ROM protection policy. The enabling and disabling of overall On-Chip FLASH protection policy is realized by the protection word. If the protection word is enabled, the overall On-Chip FLASH protection policy is enabled; and if the field is disabled, the protection policy is disabled. The independent ROM protection policy is mainly for the read protection of ROM area. The enabling and disabling of the overall On-Chip FLASH protection policy does not affect the status of independent ROM protection policy. The specific logic of protection policy is as follows:

- If the ROM area does not perform programming operation and the protection word is not enabled, MCU and others can access and read the ROM area
- If the ROM area has performed programming operation but the protection word is not enabled, MCU and others can access and read the ROM area
- If the ROM area has performed programming operation and the protection word is enabled, others cannot access or read the ROM area, and MCU cannot read but can execute the ROM area
- If the protection word of overall On-Chip FLASH protection is not enabled, MCU and others can access and read the content in On-Chip FLASH (except in ROM area); whether the content of the ROM area can be accessed and read depends on the state of the protection word in the ROM area, but MCU can execute the applications in ROM area.
- If the protection word of overall On-Chip FLASH protection is enabled, others cannot access and read the content in On-Chip FLASH (including MAIN/ROM/NVR); MCU can access and read the content in On-Chip FLASH (except in ROM area); whether the content of the ROM area can be accessed and read depends on the state of the protection word in the ROM area, but MCU can

execute the applications in ROM area.

Depending on the size of the MAIN and ROM areas, the allocated addresses of the protection words corresponding to overall On-Chip FLASH (including MAIN, NVR and ROM) protection policy and independent ROM protection policy are different. The following table is the address allocation table of the protection word.

Table 7-5 Protection Word Address Allocation of On-Chip FLASH

Combination	Overall Protection Policy_MAIN			Independent ROM Protection Policy_ROM		
	Start	End	Protection Word Address	Start	End	Protection Word Address
Combination 1	0x0000 0000	0x0003 FFFF	0x0003 FFF0	0x0000 0000	0x0000 0000	N/A
Combination 2	0x0000 0000	0x0002 FFFF	0x0002 FFF0	0x0003 0000	0x0003 FFFF	0x0003 FFF0
Combination 3	0x0000 0000	0x0002 FFFF	0x0002 FFF0	0x0000 0000	0x0000 0000	N/A
Combination 4	0x0000 0000	0x0001 FFFF	0x0001 FFF0	0x0002 0000	0x0003 FFFF	0x0003 FFF0
Combination 5	0x0000 0000	0x0001 FFFF	0x0001 FFF0	0x0000 0000	0x0000 0000	N/A
Combination 6	0x0000 0000	0x0000 FFFF	0x0000 FFF0	0x0003 0000	0x0003 FFFF	0x0003 FFF0
Combination 7	0x0000 0000	0x0000 FFFF	0x0000 FFF0	0x0000 0000	0x0000 0000	N/A

If On-Chip FLASH is in overall protection state, the user can deprotect it to disable the protection of data in On-Chip FLASH (ROM area depends on its protection status). On the contrary, if On-Chip FLASH is in deprotected state, the user can protect it to enable the overall protection of data in On-Chip FLASH. The data in the On-Chip FLASH memory is in overall protection state by default. After the chip is powered on and reset, the hardware will update the protection status once automatically. Whether the data is protected or deprotected, it will remain unchanged after being updated.

As can be seen from the above table, we generally assign the last four Words in the current combination as protected words. If the protection word is all 1, it indicates that the corresponding policy is in deprotected state; if the protection word is not all 1, it indicates that the corresponding policy is in protected state. Therefore, to enable protection, perform programming of the corresponding protection word, write a value of not all 1, and read the FSMC_PROT register to trigger update of the protection stats once and complete protection (read the return value of FSMC_PROT to know the current protection status). Because two sets of protection policies have two protection words, we need to know which protection status is updated.

Table 7-6 Protection Word Address Allocation of On-Chip FLASH

FSMC_ICFG.SEL_PROT	Zone
0	Overall protection policy area
1	Independent ROM protection policy area

When FSMC_ICFG.PROT is set to 0, read the FSMC_PROT register and update the On-Chip FLASH in overall protection state; when FSMC_ICFG.PROT is set to 1, read the FSMC_PROT register and update the ROM area in independent protection state.

7.2.2 Functional Description of Ext-Chip FLASH Controller

The hardware LKS32MC45x executes the Ext-Chip FLASH controller, which does not include the external FLASH memory. The address mapping space of the external FLASH starts at 0x0100_0000 and ends at 0x01FF_FFFF, and the size is 16MB.

Ext-Chip FLASH controller has the following features:

- The address bus width is 24 bits, supporting the external FLASH capacity range of >64KB to ≤16MB.
- Only in SPI3 transmission mode, that is SCLK is in high level by default, the falling edge sends data and the rising edge collects data.
- Support single-line SPI transmission, double-line SPI transmission and four-line transmission.
- Access directly through MCU or indirectly through registers.

7.2.2.1 Ext-Chip FLASH Access Mode

Ext-Chip FLASH has two access modes: direct access and indirect access. Direct access means that the MCU addresses the corresponding Ext-Chip FLASH directly through the address. Indirect address means that the MCU access through the register of Ext-Chip FLASH controller. Generally, reading/running the content in Ext-Chip FLASH can use direct access mode; and reading/programming Ext-Chip FLASH can use indirect access mode.

7.2.2.2 Ext-Chip FLASH Parallel Transmission Number

Ext-Chip FLASH controller supports three parallel access numbers: single-line, double-line and four-line. The switch of different parallel numbers requires configuration instructions sent through the registers of the Ext-Chip FLASH controller. The common PIN multiplexing scheme of Ext-Chip FLASH is as follows:

Table 7-7 Description of Ext-Chip FLASH PIN and Its Interconnection with LKS32MC45x

Symbol	Extension	Remarks	LKS45x
CS#		Chip select	EFLS_CSN
SO	SIO1	Serial data output for 1 x I/O Serial data input and output for 4 x I/O read mode	EFLS_DAT[1]

WP#	SIO2	Write protection active low Serial data input and output for 4 x I/O read mode	EFLS_DAT[2]
SI	SIO0	Serial data input for 1x I/O Serial data input and output for 4 x I/O read mode	EFLS_DAT[0]
SCLK		Serial interface clock input	EFLS_CLK
HOLD#/RESET#	SIO3	Hardware Reset Pin Active low or to pause the device without deselecting the device Serial data input and output for 4 x I/O read mode	EFLS_DAT[3]

Single-line mode is used by default, that is, one line for sending data and the other for receiving data. WP# of Ext-Chip FLASH is equipped with pull-down resistor to avoid misprogramming and OLD#/RESET# is equipped with pull-up resistor. The signal corresponding to the LKS32MC45x is input by default, which does not affect the multiplexing function. To switch to the other two modes, Ext-Chip FLASH controller needs to send a command first, and the multiplexing will be enabled after Ext-Chip FLASH receives the command. At this time, Ext-Chip FLASH can switch to other modes and the Ext-Chip FLASH controller can enable multi-line transmission simultaneously.

7.2.2.3 Ext-Chip FLASH Command Format

Generally, the Ext-Chip FLASH command consists of the following parts, but some of which do not need to be sent depending to the specific command. Refer to the Ext-Chip FLASH document for details. Before Ext-Chip FLASH controller sends a command, all parts shall be configured. If some parts do not need to be sent, the data length of the corresponding parts can be configured to 0.

- Command
- Address
- Mode
- Dummy
- Data

Taking P25Q32H as an example, the following sets of configurations are provided for reference.

7.2.2.3.1 Ext-Chip FLASH Indirect Four-line Read

Indirect access is mainly to configure the FSMC_UCFG and other registers. The access waveform is as follows:

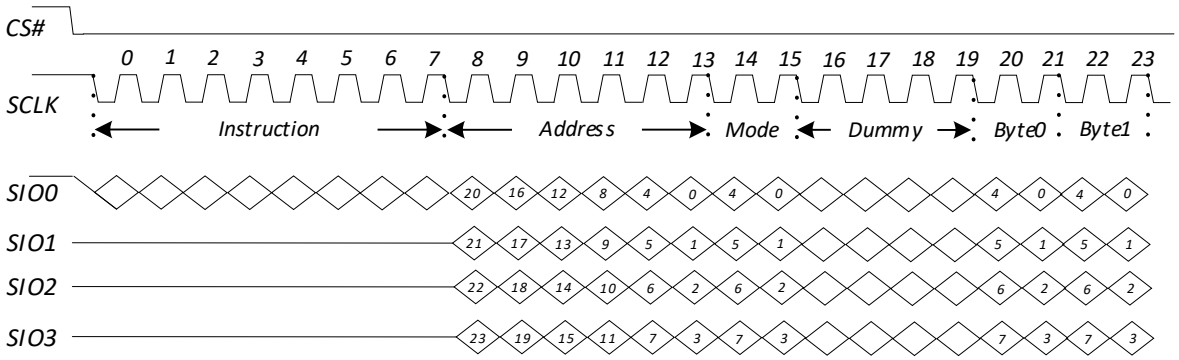


Figure 7-7 Data Waveform of Ext-Chip FLASH Indirect Four-line Read

All parts are included and in four-line mode (except the command part). FSMC_UCFG is configured as:

FSMC_UCFG = 0x3FC0_4123;

Here, FSMC_UCFG.DAT_LEN is set to 4, indicating that four bytes of data are read. It can also be set to other value.

7.2.2.3.2 Ext-Chip FLASH Direct Four-line Read

Direct access is mainly to configure the FSMC_FCFG and other registers. The access waveform is as follows:

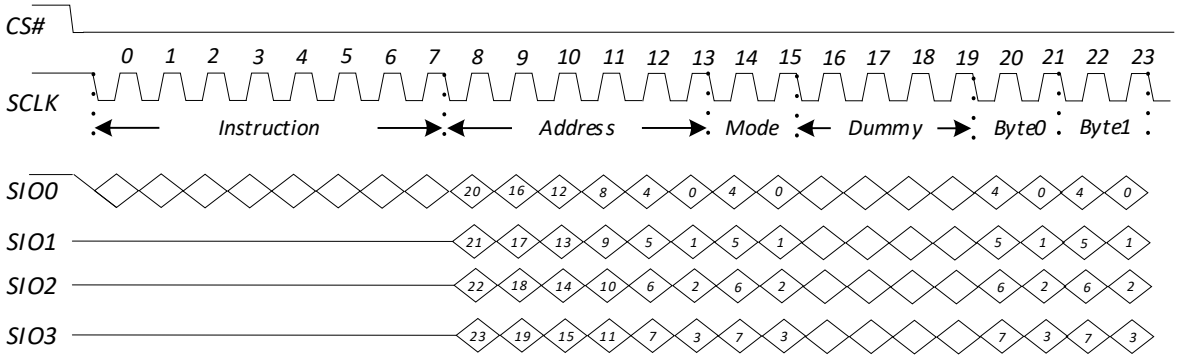


Figure 7-8 Data Waveform of Ext-Chip FLASH Direct Four-line Read

All parts are included and in four-line mode (except the command part). FSMC_FCFG is configured as:

FSMC_FCFG = 0x03FC_0421;

Here, FSMC_FCFG.DAT_LEN is set to 4, indicating that four bytes of data are read. It can also be set to other value. However, MCU only needs 32-bit data at a time, it is reasonable to set the data

length of direct access to 4.

7.2.2.4 Ext-Chip FLASH Storage Sequence

The minimum storage unit of Ext-Chip FLASH is byte. Bytes are stored in ascending order. When a read operation occurs, bytes are sent one by one according to the address increment; the receiver, in case of indirect access, decides the order of read according to the configuration of FSMC_UCFG.ORDER.

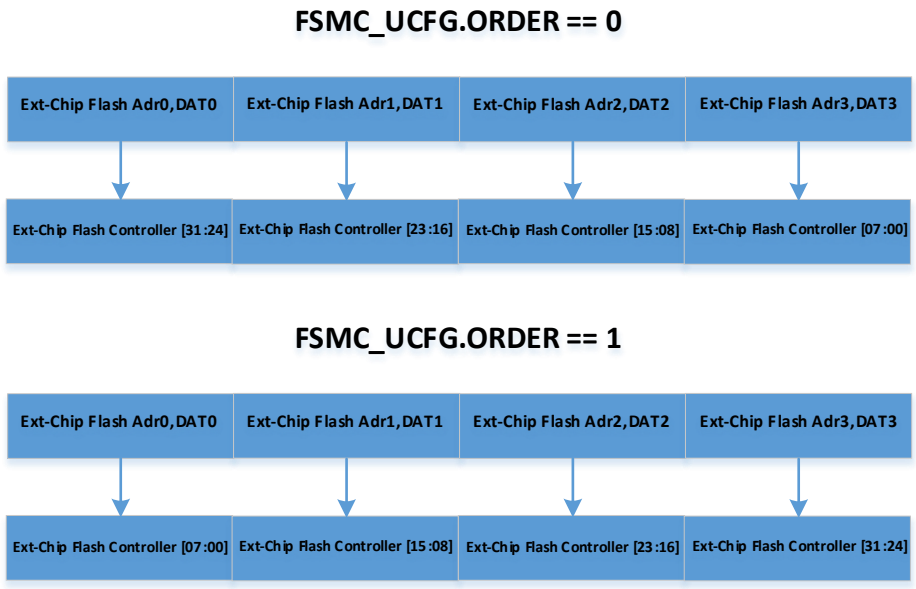


Figure 7-9 Data Read Sequence of Ext-Chip FLASH Indirect Access

In case of direct access, the condition is similar, and refer to the value of FSMC_FCFG.ORDER.

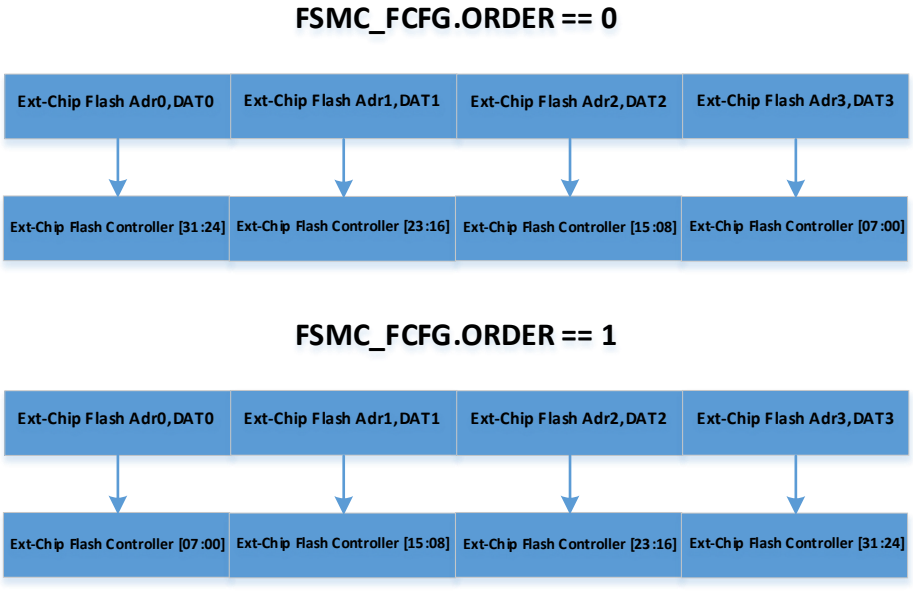


Figure 7-10 Data Read Sequence of Ext-Chip FLASH Direct Access

7.2.2.5 DMA Transmission

Ext-Chip FLASH controller supports DMA transmission. The DMA transmission mode is different for indirect access and direct access.

In case of indirect access, FSMC_UTRG.DMA_EN is set to 1 to enable DMA. The length of transmission data is an integer multiple of 4. That is, a DMA transmission request is generated after each transmission of 4 bytes. For example:

DMA_CTMS4 = 0x00010004;// 1 round 4-times

DMA_DADR4 = 0x20001000;// ram space

DMA_SADR4 = 0x4001A010;// ext-chip flash read data register

DMA_REN4 = 0x40000000;// ext-chip flash dma request enable

DMA_CCR4 = 0x00000A11;// word-alignment

DMA_CTRL = 0x00000001;// enable DMA

//wait DMA interrupt

In case of direct access, the FSMC_UTRG.DMA_EN control bit does not needs to be enabled. DMA software can be used to trigger the read access request to the Ext-Chip FLASH. For example:

DMA_CTMS4 = 0x00010004;// 1 round 4-times

DMA_DADR4 = 0x20001000;// ram space

DMA_SADR4 = 0x01000000;// ext-chip flash space

DMA_REN4 = 0x80000000;// soft trig

DMA_CCR4 = 0x00000AF1;// word-alignment

DMA_CTRL = 0x00000001;// enable DMA

//wait DMA interrupt

7.3 Register

7.3.1 Address Allocation

The base address of the non-volatile memory module register is 0x4001_A000.

Table 7-8 List of Non-volatile Memory Module Register

Name	Offset	Description

FSMC_UCMD	0x00	Ext-Chip FLASH indirect access command register
FSMC_UMOD	0x04	Ext-Chip FLASH indirect access mode register
FSMC_UADR	0x08	Ext-Chip FLASH indirect access address register
FSMC_UWDA	0x0C	Ext-Chip FLASH indirect access programming register
FSMC_URDA	0x10	Ext-Chip FLASH indirect access read register
FSMC_UCFG	0x20	Ext-Chip FLASH indirect access control register
FSMC_UTRG	0x24	Ext-Chip FLASH indirect access operation control register
FSMC_FCMD	0x30	Ext-Chip FLASH direct access command register
FSMC_FMOD	0x34	Ext-Chip FLASH direct access mode register
FSMC_FCFG	0x50	Ext-Chip FLASH direct access control register
FSMC_FTRG	0x54	Ext-Chip FLASH direct access operation register
FSMC_EDIV	0x60	Ext -Chip FLASH baud rate register
FSMC_WDAT0	0x70	On-Chip FLASH programming register, BIT[031:000]
FSMC_WDAT1	0x74	On-Chip FLASH programming register, BIT[063:032]
FSMC_WDAT2	0x78	On-Chip FLASH programming register, BIT[095:064]
FSMC_WDAT3	0x7C	On-Chip FLASH programming register, BIT[127:096]
FSMC_ICFG	0x80	On-Chip FLASH configuration register
FSMC_ADDR	0x84	On-Chip FLASH address register
FSMC_WDAT	0x88	On-Chip FLASH programming trigger register
FSMC_RDAT	0x8C	On-Chip FLASH read register
FSMC_ERAS	0x90	On-Chip FLASH erase trigger register
FSMC_PROT	0x94	On-Chip FLASH protection register
FSMC_REDY	0x98	On-Chip FLASH working status register
FSMC_IDIV	0x9C	On-Chip FLASH time base register

7.3.2 Ext-Chip FLASH Indirect Access Command Register (FSMC_UCMD)

Address: 0x4001_A000

Reset value: 0x0

Table 7-9 Ext-Chip FLASH Indirect Access Command Register (FSMC_UCMD)

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

Bit field	Bit Name	Description
[7:0]	UCMD	Ext-Chip FLASH indirect access command register

7.3.3 Ext-Chip FLASH Indirect Access Mode Register (FSMC_UMOD)

Address: 0x4001_A004

Reset value: 0x0

Table 7-10 Ext-Chip FLASH Indirect Access Mode Register (FSMC_UMOD)

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

Bit field	Bit Name	Description
[7:0]	UMOD	Ext-Chip FLASH indirect access mode register

7.3.4 Ext-Chip FLASH Indirect Access Address Register (FSMC_UADR)

Address: 0x4001_A008

Reset value: 0x0

Table 7-11 Ext-Chip FLASH Indirect Access Address Register (FSMC_UADR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								FSMC_UADR							
								RW							
								0							

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

Bit field	Bit Name	Description
[23:0]	FSMC_UADR	Ext-Chip FLASH indirect access address register

7.3.5 Ext-Chip FLASH Indirect Access Programming Register (FSMC_UWDA)

Address: 0x4001_A00C

Reset value: 0x0

Table 7-12 Ext-Chip FLASH Indirect Access Programming Register (FSMC_UWDA)

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

FSMC_UWDA
RW
0

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

Bit field	Bit Name	Description
[31:0]	FSMC_UWDA	Ext-Chip FLASH indirect access programming register

7.3.6 Ext-Chip FLASH Indirect Access Read Register (FSMC_URDA)

Address: 0x4001_A010

Reset value: 0x0

Table 7-13 Ext-Chip FLASH Indirect Access Read Register (FSMC_URDA)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FSMC_URDA															
RO															
0															

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

Bit field	Bit Name	Description
[31:0]	FSMC_URDA	Ext-Chip FLASH indirect access read register

7.3.7 Ext-Chip FLASH Indirect Access Control Register (FSMC_UCFG)

Address: 0x4001_A020

Reset value: 0x0

Table 7-14 Ext-Chip FLASH Indirect Access Control Register (FSMC_UCFG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ORDER		DAT_SPD		DMY_SPD		MOD_SPD		ADR_SPD		CMD_SPD		DAT_LEN			
RW		RW		RW		RW		RW		RW		RW			

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

Bit field	Bit Name	Description
31	ORDER	Direct access to read data size header settings 0: Ext-Chip FLASH stores data in big endian order 1: Ext-Chip FLASH stores data in little endian order
30	--	Reserved, set to 0
[29:28]	DAT_SPD	Data parallel sending configuration 00: One line for transmission 01: Two lines for transmission 11: Four lines for transmission 10: Illegal configuration
[27:26]	DMY_SPD	Dummy parallel sending configuration 00: One line for transmission 01: Two lines for transmission 11: Four lines for transmission 10: Illegal configuration
[25:24]	MOD_SPD	Mode parallel sending configuration 00: One line for transmission 01: Two lines for transmission 11: Four lines for transmission 10: Illegal configuration
[23:22]	ADR_SPD	Address parallel sending configuration 00: One line for transmission 01: Two lines for transmission 11: Four lines for transmission 10: Illegal configuration
[21:20]	CMD_SPD	Command parallel sending configuration 00: One line for transmission 01: Two lines for transmission 11: Four lines for transmission 10: Illegal configuration
[19:12]	DAT_LEN	For data length in data part, byte is the minimum unit, 0 indicating no data, and 1 indicating 1 byte, and so on. The maximum value is 15 bytes
[11:8]	DMY_LEN	For data length in dummy part, byte is the minimum unit, 0 indicating no data, and 1 indicating 1 byte, and so on. The maximum value is 15 bytes

[7:4]	MOD_LEN	For data length in mode part, byte is the minimum unit, 0 indicating no data, and 1 indicating 1 byte, and so on. The maximum value is 15 bytes
[3:0]	--	Reserved, set to 0

7.3.8 Ext-Chip FLASH Indirect Access Operation Register (FSMC_UTRG)

Address: 0x4001_A024

Reset value: 0x0

Table 7-15 Ext-Chip FLASH Indirect Access Operation Register (FSMC_UTRG)

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

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

Bit field	Bit Name	Description
[31:3]	--	Reserved, set to 0
[2]	DMA_EN	Ext-Chip FLASH controller enable DMA transfer operation
[1]	OP	Indirectly access Ext-Chip FLASH via Ext-Chip FLASH controller. Operation type 1: Programming/erase 0: Read
[0]	SEL	Indirectly access Ext-Chip FLASH via Ext-Chip FLASH controller. This bit is also the status bit. 1: Trigger operation, processing 0: Not trigger operation, idle

7.3.9 Ext-Chip FLASH Direct Access Command Register (FSMC_FCMD)

Address: 0x4001_A030

Reset value: 0x0

Table 7-16 Ext-Chip FLASH Direct Access Command Register (FSMC_FCMD)

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

	RW
	0

Bit field	Bit Name	Description
[7:0]	FCMD	Ext-Chip FLASH direct access command register

7.3.10 Ext-Chip FLASH Direct Access Mode Register (FSMC_FMOD)

Address: 0x4001_A034

Reset value: 0x0

Table 7-17 Ext-Chip FLASH Direct Access Mode Register (FSMC_FMOD)

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

Bit field	Bit Name	Description
[7:0]	FMOD	Ext-Chip FLASH direct access mode register

7.3.11 Ext-Chip FLASH Direct Access Control Register (FSMC_FCFG)

Address: 0x4001_A050

Reset value: 0x400

Table 7-18 Ext-Chip FLASH Direct Access Control Register (FSMC_FCFG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
				ORDER		DAT_SPD		DMY_SPD		MOD_SPD		ADR_SPD		CMD_SPD	
				RW		RW		RW		RW		RW		RW	
				0		0		0		0		0		0	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT_LEN								DMY_LEN				MOD_LEN			
RW								RW				RW			
0x4								0				0			

Bit field	Bit Name	Description
27	ORDER	Direct access to read data size header settings

		0: Ext-Chip FLASH stores data in big endian order 1: Ext-Chip FLASH stores data in little endian order
26	--	Reserved, set to 0
[25:24]	DAT_SPD	Data parallel sending configuration 00: One line for transmission 01: Two lines for transmission 11: Four lines for transmission 10: Illegal configuration
[23:22]	DMY_SPD	Dummy parallel sending configuration 00: One line for transmission 01: Two lines for transmission 11: Four lines for transmission 10: Illegal configuration
[21:20]	MOD_SPD	Mode parallel sending configuration 00: One line for transmission 01: Two lines for transmission 11: Four lines for transmission 10: Illegal configuration
[19:18]	ADR_SPD	Address parallel sending configuration 00: One line for transmission 01: Two lines for transmission 11: Four lines for transmission 10: Illegal configuration
[17:16]	CMD_SPD	Command parallel sending configuration 00: One line for transmission 01: Two lines for transmission 11: Four lines for transmission 10: Illegal configuration
[15:8]	DAT_LEN	For data length in data part, byte is the minimum unit, 0 indicating no data, and 1 indicating 1 byte, and so on. The maximum value is 15 bytes
[7:4]	DMY_LEN	For data length in dummy part, byte is the minimum unit, 0 indicating no data, and 1 indicating 1 byte, and so on. The maximum value is 15 bytes
[3:0]	MOD_LEN	For data length in mode part, byte is the minimum unit, 0 indicating no data, and 1 indicating 1 byte, and so on. The maximum value is 15 bytes

7.3.12 Ext-Chip FLASH Direct Access Operation Register (FSMC_FTRG)

Address: 0x4001_A054

Reset value: 0x0

Table 7-19 Ext-Chip FLASH Direct Access Operation Register (FSMC_FTRG)

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

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

Bit field	Bit Name	Description
[31:1]	--	Reserved, set to 0
[0]	UPDATE	Update the Ext-Chip FLASH controller direct access command register. Since there is a possibility that MCU has been accessing the Ext-Chip FLASH data directly, for the sake of safety, it is necessary to update the direct access register during idle time. This bit is also a status bit, indicating whether the update is completed. 1: Trigger update/updating 2: Not trigger update/updated

7.3.13 Ext -Chip FLASH Baud Rate Register (FSMC_EDIV)

Address: 0x4001_A060

Reset value: 0x44444

Table 7-20 Ext-Chip FLASH Baud Rate Register (FSMC_EDIV)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
												DAT_BAUDL		DAT_BAUDH	
												RW		RW	
												2		0	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMY_BAUDL		DMY_BAUDH		MOD_BAUDL		MOD_BAUDH		ADR_BAUDL		ADR_BAUDH		CMD_BAUDL		CMD_BAUDH	
RW		RW		RW		RW		RW		RW		RW		RW	
2		0		2		0		2		0		2		0	

Bit field	Bit Name	Description
[19:18]	DAT_BAUDL	Data baud rate configuration, number of low level duration cycles 0: One system clock cycle 1: Two system clock cycles 2: Three system clock cycles

		3: Four system clock cycles
[17:16]	DAT_BAUDH	Data baud rate configuration, number of high level duration cycles 0: One system clock cycle 1: Two system clock cycles 2: Three system clock cycles 3: Four system clock cycles
[15:14]	DMY_BAUDL	Dummy baud rate configuration, number of low level duration cycles 0: One system clock cycle 1: Two system clock cycles 2: Three system clock cycles 3: Four system clock cycles
[13:12]	DMY_BAUDH	Dummy baud rate configuration, number of high level duration cycles 0: One system clock cycle 1: Two system clock cycles 2: Three system clock cycles 3: Four system clock cycles
[11:10]	MOD_BAUDL	Mode baud rate configuration, number of low level duration cycles 0: One system clock cycle 1: Two system clock cycles 2: Three system clock cycles 3: Four system clock cycles
[09:08]	MOD_BAUDH	Mode baud rate configuration, number of high level duration cycles 0: One system clock cycle 1: Two system clock cycles 2: Three system clock cycles 3: Four system clock cycles
[07:06]	ADR_BAUDL	Address baud rate configuration, number of low level duration cycles 0: One system clock cycle 1: Two system clock cycles 2: Three system clock cycles 3: Four system clock cycles
[05:04]	ADR_BAUDH	Address baud rate configuration, number of high level duration cycles 0: One system clock cycle 1: Two system clock cycles 2: Three system clock cycles 3: Four system clock cycles
[03:02]	CMD_BAUDL	Command baud rate configuration, number of low level duration cycles 0: One system clock cycle 1: Two system clock cycles 2: Three system clock cycles 3: Four system clock cycles
[01:00]	CMD_BAUDH	Command baud rate configuration, number of high level duration

		cycles 0: One system clock cycle 1: Two system clock cycles 2: Three system clock cycles 3: Four system clock cycles

Note: Baud rate is the sum of L system clock cycle and H system clock cycle.

7.3.14 On-Chip FLASH Programming Register (FSMC_WDAT0)

Address: 0x4001_A070

Reset value: 0x0

Table 7-21 On-Chip FLASH Programming Register (FSMC_WDAT0)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WDATA															
RW															
0															

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

Bit field	Bit Name	Description
[31:0]	WDATA	On-Chip FLASH programming register, BIT[031:000]

7.3.15 On-Chip FLASH Programming Register (FSMC_WDAT1)

Address: 0x4001_A074

Reset value: 0x0

Table 7-22 On-Chip FLASH Programming Register (FSMC_WDAT1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WDATA															
RW															
0															

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

0

Bit field	Bit Name	Description
[31:0]	WDATA	On-Chip FLASH programming register, BIT[063:032]

7.3.16 On-Chip FLASH Programming Register (FSMC_WDAT2)

Address: 0x4001_A078

Reset value: 0x0

Table 7-23 On-Chip FLASH Programming Register (FSMC_WDAT2)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WDATA															
RW															
0															

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

Bit field	Bit Name	Description
[31:0]	WDATA	On-Chip FLASH programming register, BIT[095:064]

7.3.17 On-Chip FLASH Programming Register (FSMC_WDAT3)

Address: 0x4001_A07C

Reset value: 0x0

Table 7-24 On-Chip FLASH Programming Register (FSMC_WDAT3)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WDATA															
RW															
0															

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

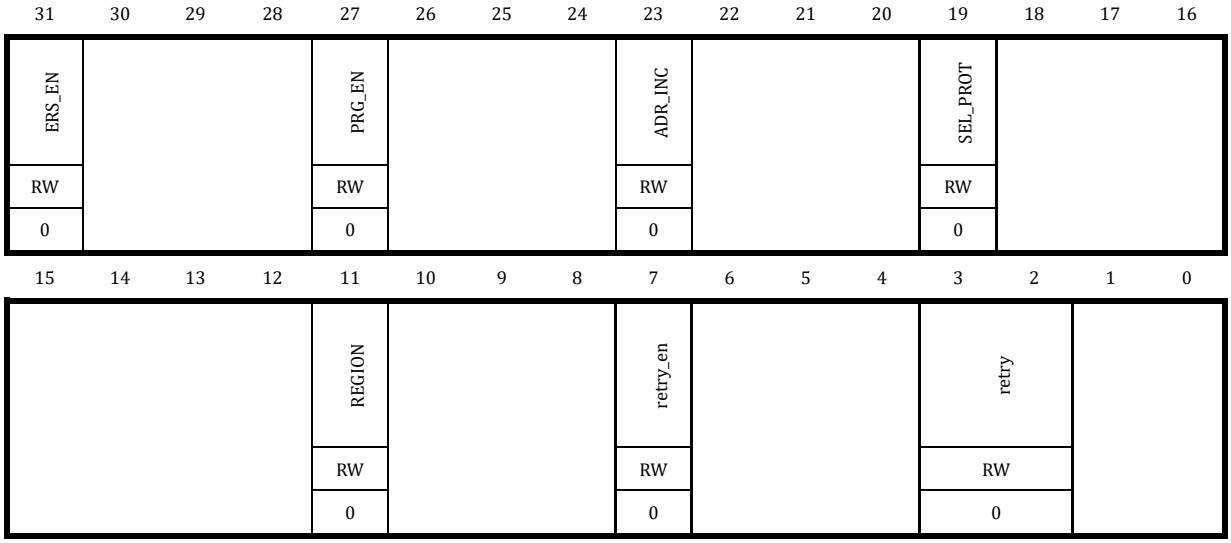
Bit field	Bit Name	Description
[31:0]	WDATA	On-Chip FLASH programming register, BIT[127:096]

7.3.18 On-Chip FLASH Configuration Register (FSMC_ICFG)

Address: 0x4001_A080

Reset value: 0x60

Table 7-25 On-Chip FLASH Configuration Register (FSMC_ICFG)



Bit field	Bit Name	Description
[31]	ERS_EN	On-Chip FLASH erase enable. The default value is 0. 0: Erasure off 1: Erasure on
[27]	PRG_EN	On-Chip FLASH programming enable. The default value is 0. 0: Programming off 1: Programming on
[23]	ADR_INC	On-Chip FLASH address increment enable. The default value is 0. 0: Address increment off 1: Address increment on When performing On-Chip FLASH continuous read and write access, enable this function can reduce the operation of the address.
[19]	SEL_PROT	On-Chip FLAS protection word selection 0: Select the protection word of overall On-Chip FLASH protection policy 1: Select the protection word of independent ROM protection policy
[11]	REGION	Access On-Chip FLASH area selection. The default value is 0. 0: MAIN/ROM (select the area according to the address) 1: NVR

[7]	RETRY_EN	Quick erase enable. 0: Disable 1: Enable
[3:2]	RETRY	Quick erase counter

7.3.19 On-Chip FLASH Address Register (FSMC_ADDR)

Address: 0x4001_A084

Reset value: 0x0

Table 7-26 On-Chip FLASH Address Register (FSMC_ADDR)

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR																	
RW																	
0																	

Bit field	Bit Name	Description
[31:16]		Unused
[17:0]	ADDR	Address register. Address register corresponding to read/write/erase operation. Read operation, read 32-BIT data once, and the last two bits of ADDR will be ignored by the On-Chip FLASH controller. Programming operation, program 128-BIT data once, and the last four bits of ADDR will be ignored by the On-Chip FLASH controller. Erase operation, a Sector is 1024-Byte, and the last ten bits of the ADDR will be ignored by the On-Chip FLASH controller when performing Sector erasure. When performing Full erasure, the value of ADDR will be ignored by the On-Chip FLASH controller.

7.3.20 On-Chip FLASH Programming Trigger Register (FSMC_WDAT)

Address: 0x4001_A088

Reset value: 0x0

Table 7-27 On-Chip FLASH Programming Trigger Register (FSMC_WDAT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WDATA															
WO															
0															

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

WO
0

Bit field	Bit Name	Description
[31:0]	WDATA	Write 0x7654DCBA to trigger the On-Chip FLASH programming

7.3.21 On-Chip FLASH Read Register (FSMC_RDAT)

Address: 0x4001_A08C

Reset value: 0x0

Table 7-28 On-Chip FLASH Read Register (FSMC_RDAT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDATA															
RO															
0															

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

Bit field	Bit Name	Description
[31:0]	RDATA	Perform read operation to read the corresponding address value of On-Chip FLASH

7.3.22 On-Chip FLASH Erase Trigger Register (FSMC_ERAS)

Address: 0x4001_A090

Reset value: 0x0

Table 7-29 On-Chip FLASH Erase Trigger Register (FSMC_ERAS)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ERASE															
WO															
0															

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

Bit field	Bit Name	Description
[31:0]	ERASE	Write 0x7654DCBA to trigger the On-Chip FLASH erasure

7.3.23 On-Chip FLASH Protection Register (FSMC_PROT)

Address: 0x4001_A094

Reset value: 0x0808

Table 7-30 On-Chip FLASH Protection Register (FSMC_PROT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
															RPROT
															RO
															1

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

Bit field	Bit Name	Description
[16]	RPROT	Protection Status of Independent ROM Protection Policy 0: Unprotected 1: Protecting
[0]	PROT	Protection Status of Overall On-Chip FLASH Protection Policy 0: Unprotected 1: Protecting

7.3.24 On-Chip FLASH Working Status Register (FSMC_REDY)

Address: 0x4001_A098

Reset value: 0x00000101

Table 7-31 On-Chip FLASH Working Status Register (FSMC_REDY)

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

Bit field	Bit Name	Description
[31:9]		Unused

[8]	IREADY	S bus access On-Chip FLASH status 0: S bus is accessing On-Chip FLASH, which cannot accept new access request from S bus 1: S bus is not accessing On-Chip FLASH, which can accept new access request from S bus
[7:1]		Unused
[0]	READY	I/D bus access On-Chip FLASH status 0: I/D bus is accessing On-Chip FLASH, which cannot accept new access request from I/D bus 1: I/D bus is not accessing On-Chip FLASH, which can accept new access request from I/D bus

Note: S bus is the external access bus, which directly accesses the On-Chip FLASH via the register and supports read/programming/erase; and I/D bus directly accesses the On-Chip FLASH via MCU and supports read operation only.

7.3.25 On-Chip FLASH Time Base Register (FSMC_IDIV)

Address: 0x4001_A09C

Reset value: 0x0

Table 7-32On-Chip FLASH Working Status Register (FSMC_IDIV)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	BCNT1														
	RW														
	0x5800														

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	BCNT0							CKL				CKH			
	RW							RW				RW			
	0x40							2				2			

Bit field	Bit Name	Description
[30:16]	BCNT1	On-Chip FLASH programming erase time base register 1. The value can be fine-tuned for different system frequencies. The default value is 0x5800, corresponding to the 192Msystem clock. 192M: 0x5800 096M: 0x2C00 048M: 0x1600
[14:08]	BCNT0	On-Chip FLASH programming erase time base register 0. The value can be fine-tuned for different system frequencies. The default value is 0x40,

		corresponding to the 192Msystem clock. It is generally adjusted depending on the frequency
[07:04]	CKL	On-Chip FLASH, clock signal low level duration 0x1: Only when the system frequency is lower than 96M, available value 0x2: Default value Other values are all invalid
[03:00]	CKH	On-Chip FLASH, clock signal high level duration 0x1: Only when the system frequency is lower than 96M, available value 0x2: Default value Other values are all invalid

8 DMA

8.1 Overview

Both DMA and CPU are the master devices of chip bus.

As shown in Figure 8-1. Some devices do not need to be accessed by DMA, and are only mounted on the but connected to the CPU. The devices including ADC, DAC, SPI, I2C, MCPWM, UART, Timer, GPIO, Hall and SRAM are shared and accessed by the CPU and DMA.

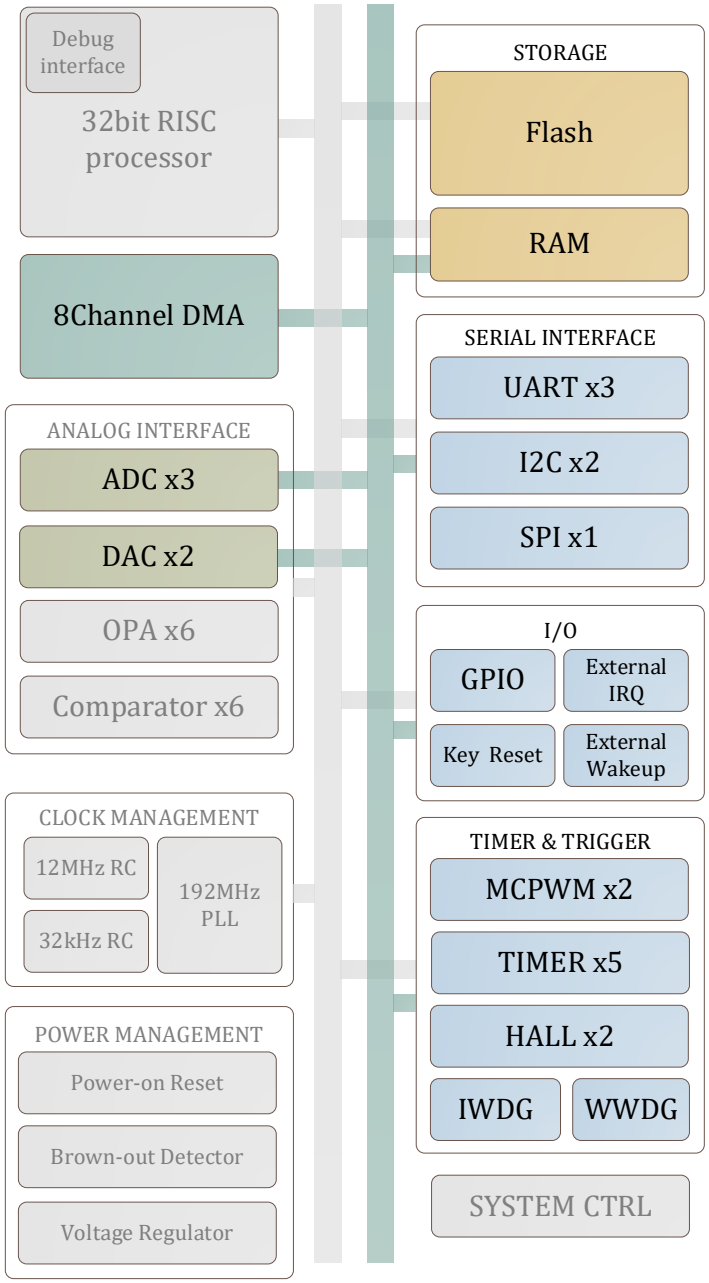


Figure 8-1 DMA AHB Bus Architecture

DMA has eight channels that share a handling engine. If one or more DMA requests happen when the DMA is in the idle state, it should be arbitrated according to the priority setting, but preemption

will not occur during the handling process, that is, only after a certain channel completes one round of handling and DMA is idle, the arbitration will be started to determine which channel will receive the request next.

Please note that DMA transmission is divided into multiple rounds, and DMA_CCRx.PINC can be used to control whether to choose one transmission for each round in multiple rounds or multiple transmissions in each round of transmission.

If it is configured to multiple transmissions in each round of transmission, that is DMA_CCRx.RMODE=0, DMA will continuously send DMA_CTMSx times data for a DMA request, and the hardware will clear the DMA_CCRx.EN and set an interrupt flag.

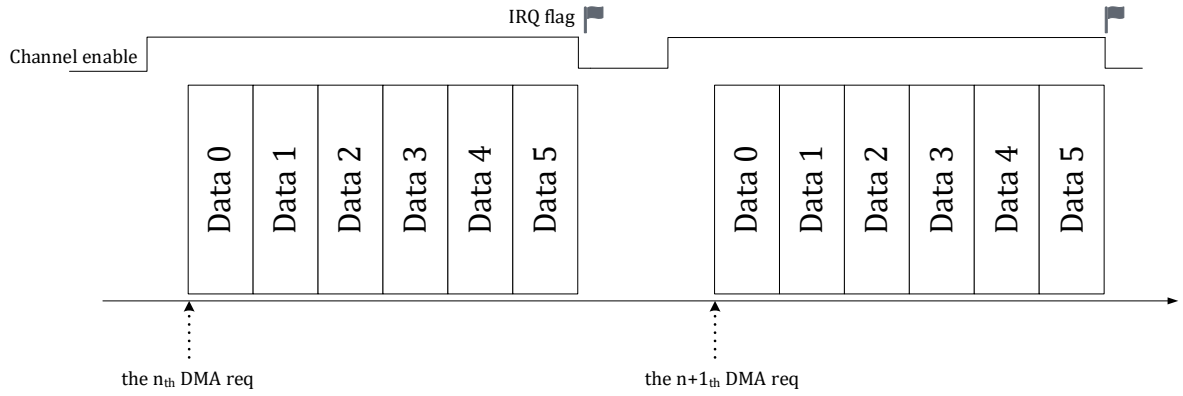


Figure 8-2 RMODE=0 DMA Transmission

If a channel is configured to multiple rounds, DMA may respond to the handling request of other channels between multiple rounds.

Multiple-round transmission only transmits data once in each round of transmission. After DMA_CTMSx rounds of transmission, the hardware will clear the DMA_CCRx.EN and set an interrupt flag.

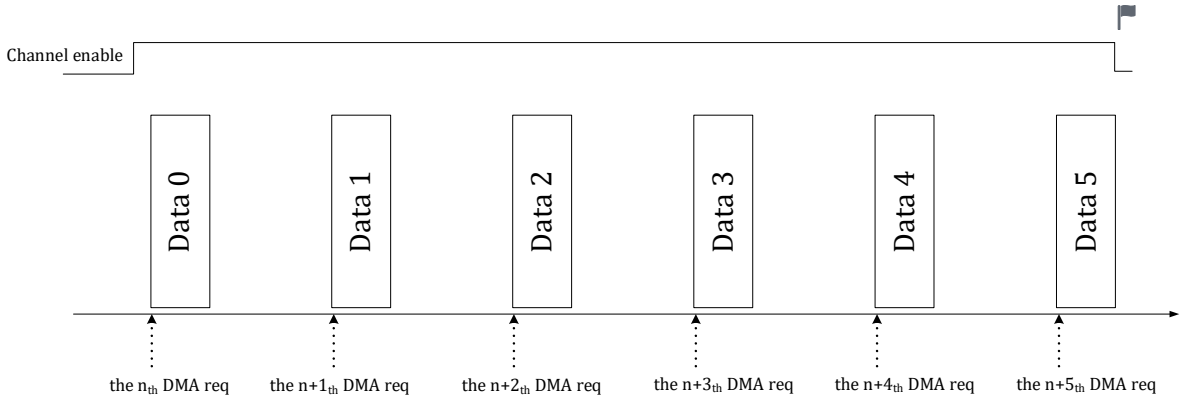


Figure 8-3 RMODE=1 DMA Transmission

Every time DMA completes a transfer, the address is incremented automatically according to

DMA_CCRx.SINC and DMA_CCRx.DINC. All peripheral register addresses are word aligned, so the peripheral address increment is always 0/4. For example, UART/SPI/I2C, who usually access the fixed address of UART_DATA or SPI/I2C FIFO interface each time, so the address does not need to be increased between rounds; when access the ADC data register, the address should be increased by 4 automatically, and set to increase between rounds. For memory, if increase between rounds is set, the value of each address increment is set according to the memory data bit width (DMA_CCRx.SBTW/DBTW); the address is automatically increased by 1 when the memory access bit width is byte, and increased by 2 for half-word while increased by 4 for word. The following figure illustrates how DMA access data source addresses change when SINC is configured differently, and the same applies to the destination addresses. It is configured to transmit 4 times in one round, and the data size of the source address and destination address are both words. In practical applications, if the configuration data size is byte or halfword, the address offset is incremented by 1/2.

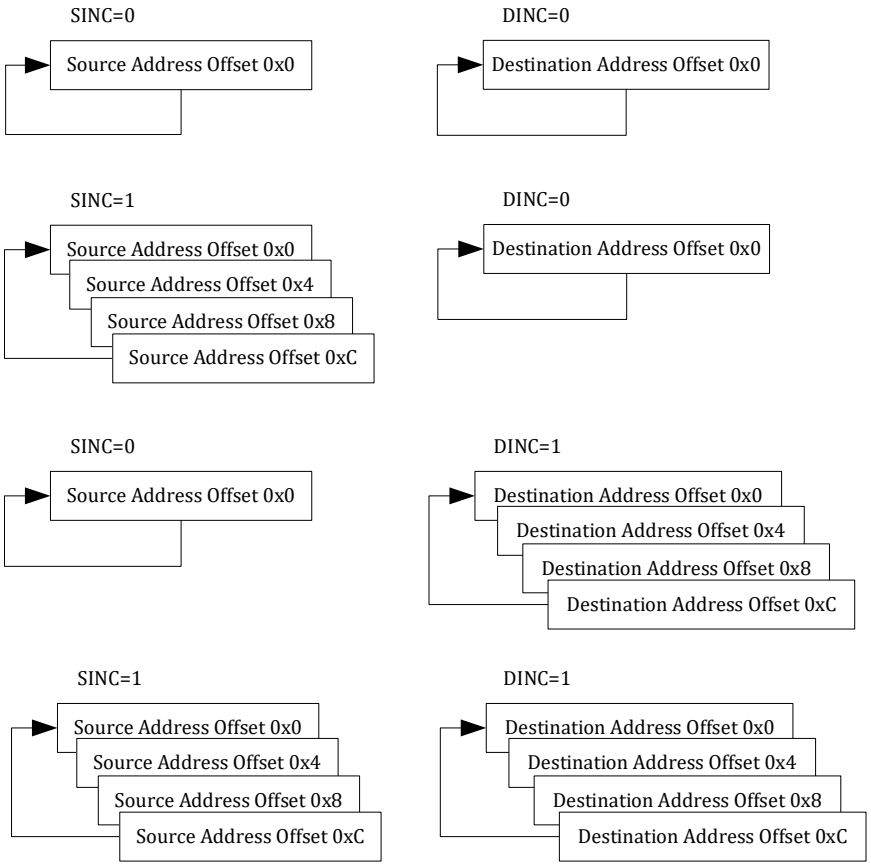


Figure 8-4 DMA Address Increment Control

Generally, if SINC=0 and DINC=0, both the source address and destination address are not incremental, which may be the transfer from the peripheral to the peripheral.

If SINC=1 and DINC=0, the source address is incremental but the destination address is not incremental, which may be the transfer of the memory array to the peripheral register interface.

If SINC=0 and DINC=1, the source address is not incremental but the destination address is incremental, which may be the transfer from the peripheral register interface to the memory array.

If SINC=1 and DINC=1, both the source address and the destination address are incremented, which is the transfer of multiple sets of data (such as ADC_DATx) to the memory array.

For power control considerations, the DMA module can be disabled by setting the DMA_CTRL.EN bit to 0 (Turn off the DMA_CCRx.EN enable corresponding to the eight channels before turning off the DMA enable), at which time the DMA clock is gated off. DMA contains the configuration register (can be configured by CMU) and data handling module (make various device access requests to the bus).

DMA supports three bit-wide transfer operations, including 8-bit, 16-bit or 32-bit (byte, half-word, or word). Select the bit width of peripheral and memory access by setting the DMA_CCRx.SBTW and DMA_CCRx.DBTW, and the bit width of peripheral access and the memory access can be different.

DMA supports cyclic mode, which is controlled by DMA_CCRx.CIRC. This mode is generally used together with DMA_CCRx.RMODE.

If DMA_CCRx.CIRC=1 and DMA_CCRx.RMODE=1, it will enter multiple-round cyclic mode. One DMA request signal only triggers one data transmission (e.g. transmit UART data to RAM), and the next transmission needs to waits for another request. In this mode, DMA_CTMSx can be configured to a large value and decreases by 1 after each transmission. Since DMA will not generate an interrupt flag for multiple round of transmission in cyclic mode, CPU can read the DMA_CTMSx to judge the data size currently transmitted, and will start processing if it is large enough. After completing DMA_CTMSx rounds of transmission, the address will return to the initial value. At this time, DMA_CTMSx will be reloaded to the preset value to start a new round of transmission. In cyclic mode, DMA channel enable DMA_CCRx.EN bit is controlled by the software, which is set to a high value and decreases to 0. To exit the transmission, DMA_CCRx.EN can be cleared by the software.

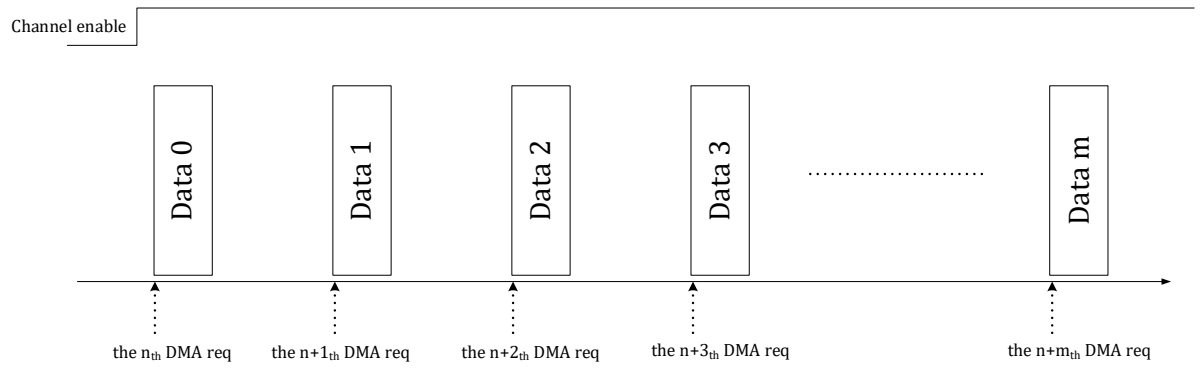


Figure 8-5 CIRC=1 and RMODE=1 DMA Transmission

If DMA_CCRx.CIRC=1 and DMA_CCRx.RMODE=0, it will enter single-shot mode, that is, DMA continuously transmits data for DMA_CTMSx times after receiving a request and generates an interrupt flag. In this mode, DMA_CCRx.EN does not need to be set by the software. It is automatically set after the DMA channel receives a request and cleared after completing one round of transmission. To exit the transmission, DMA_CCRx.CIRC and DMA_CCRx.EN can be cleared by the software.

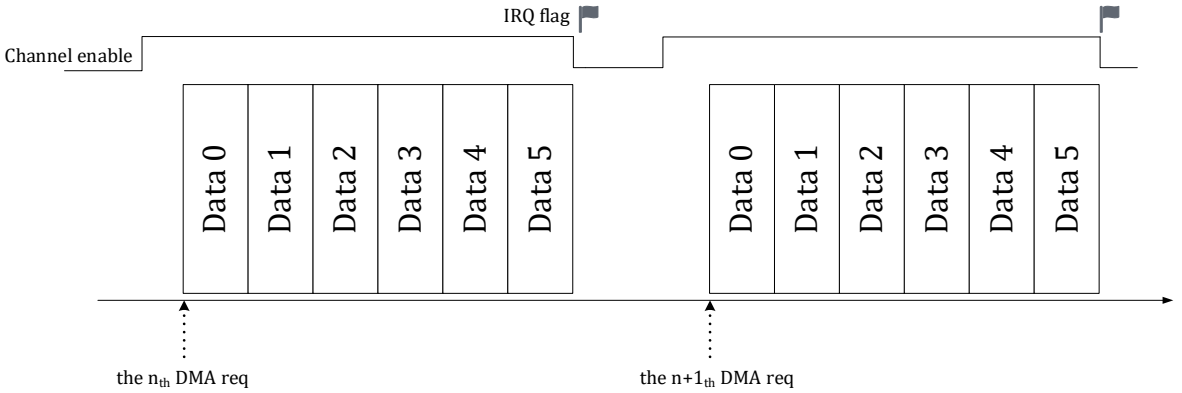


Figure 8-6 CIRC=1 and RMODE=0 DMA Transmission

In cyclic mode, DMA_CTMSx register is reloaded to the initial configuration value after decreasing to 0. If the data is transferred to the memory, the data previously transferred to the memory is overwritten; if it is transferred to a peripheral, another round of data transmission will be repeated. Taking the ADC data to the memory as an example with the data block size of 16bit×12channel×8 times, the DMA completes a round (96 half words) of transmission in the cyclic mode, and then restarts the next round of transmission. It will overwrite the previous memory address and will not set the DMA completion interrupt flag. In single-shot mode, DMA completes the DMA operation after completing the transfer of a certain size data block, and sets the DMA completion interrupt flag to the current channel, and the hardware will close the corresponding DMA channel automatically, that is, the hardware circuit sets DMA_CCRx.EN to 0 automatically after the channel transfer is completed.

8.2 Request

DMA requests have two types: software requests and hardware requests. Software requests are generated by setting DMA_RENx.SW_TRIG=1 of the corresponding DMA channel. A request is generated by writing "1", and should be cleared by software after the software trigger bit is set. The hardware request is usually an interrupt event of a peripheral. When a specific peripheral interrupt event is used as a DMA transfer request, the interrupt response of the corresponding event should be disabled, that is, CPU no longer responds to the interrupt, and the response is only a DMA transfer request. Besides, the hardware DMA request signal will be cleared by the DMA hardware after handling through the DMA channel, and the event flag does not need to be cleared by software.

Table 8-1 DMA Request

Trigger Source	Description
Software	The handling operation performed when the software is triggered is specified by the configuration register DMA_CCRx. When DMA_RENx.SW_TRIG is set, the DMA operation starts once the channel is enabled.
ADC	In the single-stage trigger mode of the ADC, an interrupt request is generated after sampling several channels at a time, and the converted value of the ADC is transferred to

	the SRAM by the DMA. The ADC single-sampling completion interrupt event is used as the DMA request signal. After the DMA response is received, the request signal is cleared by DMA, and do not clear it by software. Note that the software should disable the ADC sampling completion interrupt at the same time to prevent the CPU from responding.
Timer	Timer uses a zero-crossing/comparison event as DMA requests. The specific DMA operation is set by the configuration register, which is usually a timing event (such as triggering a DMA operation every 10ms).
SPI	The SPI module uses the full event of rx buffer as the DMA request signal. Since the SPI is transported and received at the same time, the "rx buffer full" is the event flag for both signal received and transported. Read the SPI FIFO auto-clear event flag.
MCPWM	The MCPWM module uses zero-crossing/end of counting cycle/4 ADC trigger signals as DMA requests. The specific DMA operation is set by the configuration register.
Updated I2C	The I2C module uses I2C0_SCR.BYTE_CMPLT, byte transmission completed, as a trigger DMA request. DMA clears the I2C request flag automatically. Other I2C interrupt events are still responded by the CPU.
UART	The UART module uses UART_IF to trigger DMA requests. If the transmission direction of the DMA configuration is from memory to UART, then generate a DMA request signal by UART transmission completion event; if the transmission direction is from UART to memory, then generate a DMA request signal by UART reception completion event. The event flag is cleared by DMA automatically. When the UART is operated by DMA, the corresponding interrupt should be disabled at the same time to prevent the CPU from responding.
CAN	CAN interrupt can be used as DMA request
HALL	HALL interrupt can be used as DMA request
CMP	CMP interrupt can be used as DMA request
FMAC	When setting FMAC_CR.DMAWEN=1, DMA will write data to be filtered to the FMAC when X1 buffer has enough idle space; when setting FMAC_CR.DMAREN=1, DMA will read filtering data from the FMAC when Y buffer has enough valid data. The data size to trigger DMA transmission depends on the FMAC settings
GPIO	GPIO interrupt can be used as DMA request

8.3 Priority

The priority of DMA adopts fixed priority, the priority is shown as Figure 8-7. To avoid situations where it is too late to respond to certain peripheral requests, the real-time response of the task should be considered when designing the application software, and each channel should not be configured to carry a great deal of data; otherwise, the response of other channels will be delayed.

As shown in Figure 8-7, the priority decreases from top to bottom. Among the 8 DMA channels, the priority relationship is: channel 0> channel 1> channel 2> ...> channel 8 (> sign means that the priority of the former is higher than latter). Usually, there are multiple hardware request events and one software request event within each channel of the DMA, and the hardware request priority is higher than the software request. The multiple hardware request events in the same channel have the

same priority. Usually, one of the DMA channels above is used to configure a hardware request event to be enabled. Multiple hardware requests should not occur simultaneously in one channel.

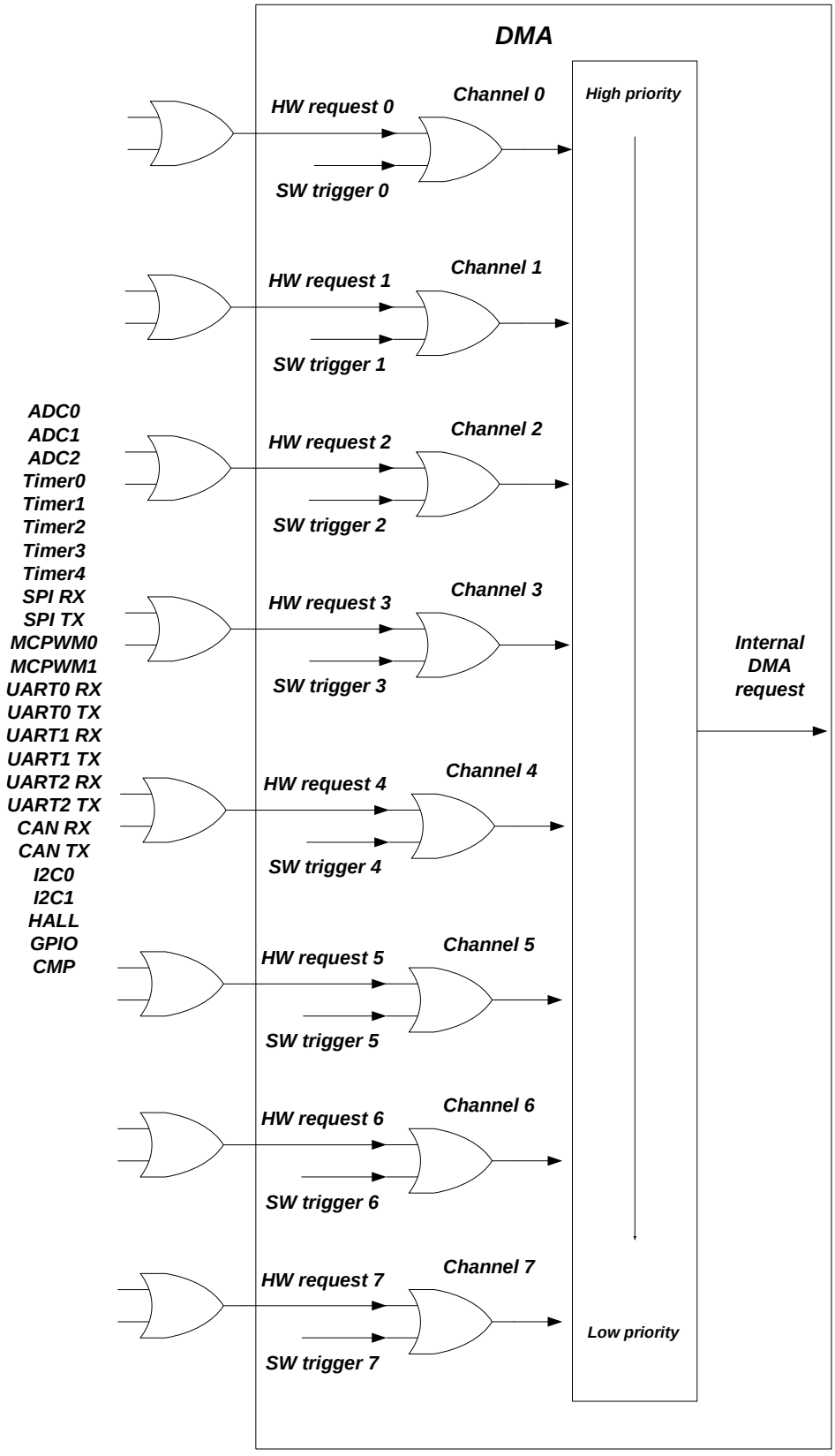


Figure 8-7 DMA Channel Priority

8.4 Arbitration

If one or more DMA requests happen when the DMA is in the idle state, or just completed the DMA transmission of a channel, it should be arbitrated according to the priority setting. Peripheral requests with higher priority will get the DMA service first. For example, every time a round of ADC data handling is completed in the ADC continuous mode, the completion event flag of ADC's sampling is cleared by the DMA, and the DMA returns to the idle state or turns to service other peripheral requests; for UART, it will re-arbitrate after transmit one byte; for SPI/I2C, it will re-arbitrate for each FIFO transmitted.

In order to avoid the long-term occupation of peripherals/SRAM by the CPU or DMA, a time slice mechanism has been added to the port arbitration module of peripherals/SRAM, that is, a master device releases access rights after a period of time. And then, the arbitration module will observe whether another master device is requesting access. If yes, it will allow another master device to access; otherwise, it continues the current unfinished access of the master device.

8.5 Interruption

After a channel of DMA completes the DMA operation or an error occurs, a DMA interrupt is generated. After a channel of DMA completes the DMA operation, it will automatically close the channel to enable DMA_CCRx.EN.

8.6 Register

8.6.1 Address Allocation

The base address of the DMA controller module register is 0x4001_9C00, and the register list is as follows:

Table 8-2 DMA Register List

Name	Offset Address	Description
DMA_CCR0	0x00	DMA channel 0 configuration register
DMA_REN0	0x04	DMA channel 0 request enable register
DMA_CTMS0	0x08	DMA channel 0 transfer count register
DMA_SADR0	0x0C	DMA channel 0 source address register
DMA_DADR0	0x10	DMA channel 0 destination address register
	0x14	
DMA_CCR1	0x18	DMA channel 1 configuration register
DMA_REN1	0x1C	DMA channel 1 request enable register
DMA_CTMS1	0x20	DMA channel 1 transfer count register
DMA_SADR1	0x24	DMA channel 1 source address register
DMA_DADR1	0x28	DMA channel 1 destination address register
	0x2C	
DMA_CCR2	0x30	DMA channel 2 configuration register
DMA_REN2	0x34	DMA channel 2 request enable register
DMA_CTMS2	0x38	DMA channel 2 transfer count register

DMA_SADR2	0x3C	DMA channel 2 source address register
DMA_DADR2	0x40	DMA channel 2 destination address register
	0x44	
DMA_CCR3	0x48	DMA channel 3 configuration register
DMA_REN3	0x4C	DMA channel 3 request enable register
DMA_CTMS3	0x50	DMA channel 3 transfer count register
DMA_SADR3	0x54	DMA channel 3 source address register
DMA_DADR3	0x58	DMA channel 3 destination address register
	0x5C	
DMA_CCR4	0x60	DMA channel 4 configuration register
DMA_REN4	0x64	DMA channel 4 request enable register
DMA_CTMS4	0x68	DMA channel 4 transfer count register
DMA_SADR4	0x6C	DMA channel 4 source address register
DMA_DADR4	0x70	DMA channel 4 destination address register
	0x74	
DMA_CCR5	0x78	DMA channel 5 configuration register
DMA_REN5	0x7C	DMA channel 5 request enable register
DMA_CTMS5	0x80	DMA channel 5 transfer count register
DMA_SADR5	0x84	DMA channel 5 source address register
DMA_DADR5	0x88	DMA channel 5 destination address register
	0x8C	
DMA_CCR6	0x90	DMA channel 6 configuration register
DMA_REN6	0x94	DMA channel 6 request enable register
DMA_CTMS6	0x98	DMA channel 6 transfer count register
DMA_SADR6	0x9C	DMA channel 6 source address register
DMA_DADR6	0xA0	DMA channel 6 destination address register
	0xA4	
DMA_CCR7	0xA8	DMA channel 7 configuration register
DMA_REN7	0xAC	DMA channel 7 request enable register
DMA_CTMS7	0xB0	DMA channel 7 transfer count register
DMA_SADR7	0xB4	DMA channel 7 source address register
DMA_DADR7	0xB8	DMA channel 7 destination address register
	0xBC	
DMA_CTRL	0xC0	DMA control register
DMA_IE	0xC4	DMA interrupt enable register
DMA_IF	0xC8	DMA interrupt flag register

8.6.2 DMA Control Register (DMA_CTRL)

Address: 0x4001_9CC0

Reset value: 0x0

Table 8-3 DMA Control Register (DMA_CTRL)

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

Bit field	Bit Name	Description
[31:1]		Unused
[0]	EN	DMA enable

8.6.3 DMA Interrupt Enable Register (DMA_IE)

Address: 0x4001_9CC4

Reset value: 0x0

Table 8-4 DMA Interrupt Enable Register (DMA_IE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								CH7_FIE	CH6_FIE	CH5_FIE	CH4_FIE	CH3_FIE	CH2_FIE	CH1_FIE	CH0_FIE
								RW	RW	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:8]		Unused
[7]	CH7_FIE	Channel 7 completion interrupt enable
[6]	CH6_FIE	Channel 6 completion interrupt enable
[5]	CH5_FIE	Channel 5 completion interrupt enable
[4]	CH4_FIE	Channel 4 completion interrupt enable
[3]	CH3_FIE	Channel 3 completion interrupt enable
[2]	CH2_FIE	Channel 2 completion interrupt enable
[1]	CH1_FIE	Channel 1 completion interrupt enable
[0]	CH0_FIE	Channel 0 completion interrupt enable

8.6.4 DMA Interrupt Flag Register (DMA_IF)

Address: 0x4001_9CC8

Reset value: 0x0

Table 8-5 DMA Interrupt Flag Register (DMA_IF)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								CH7_FIF	CH6_FIF	CH5_FIF	CH4_FIF	CH3_FIF	CH2_FIF	CH1_FIF	CH0_FIF
								RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C
								0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:8]		Unused
[7]	CH7_FIF	Channel 7 completion interrupt flag, active high, write 1 to clear.
[6]	CH6_FIF	Channel 6 completion interrupt flag, active high, write 1 to clear.
[5]	CH5_FIF	Channel 5 completion interrupt flag, active high, write 1 to clear.
[4]	CH4_FIF	Channel 4 completion interrupt flag, active high, write 1 to clear.
[3]	CH3_FIF	Channel 3 completion interrupt flag, active high, write 1 to clear.
[2]	CH2_FIF	Channel 2 completion interrupt flag, active high, write 1 to clear.
[1]	CH1_FIF	Channel 1 completion interrupt flag, active high, write 1 to clear.
[0]	CH0_FIF	Channel 0 completion interrupt flag, active high, write 1 to clear.

8.6.5 DMA Channel Configuration Register

8.6.5.1 DMA_CCRx (x =0,1,2,3,4,5,6,7)

The addresses are: 0x4001_9C00, 0x4001_9C18, 0x4001_9C30, 0x4001_9C48, 0x4001_9C60, 0x4001_9C78, 0x4001_9C90, and 0x4001_9CA8

Reset value: 0x0

Table 8-6 DMA Channel Configuration Register (DMA_CCRx)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				SBTW		DBTW			SINC		DINC	CIRC		RMODE	EN
				RW		RW			RW		RW	RW		RW	RW
				0		0			0		0	0		0	0

Bit field	Bit Name	Description
[31:12]		Unused
[11:10]	SBTW	Source address access bit width

		0: Byte 1: Halfword 2: Word 3: Reserved
[7]		Unused
[6]	SINC	Source address increment mode 0: No incremental 1: The address increases by 1/2/4 according to SBTW for each transmission
[5]		Unused
[4]	DINC	Destination address increment mode 0: No incremental 1: The address increases by 1/2/4 according to DBTW for each transmission
[3]	CIRC	Cycle mode, active high
[2]		Unused
[1]	RMODE	0: Single-round transmission, it means multiple transmissions in one round. When multiple DMA requests are received, one round of transmission is required. 1: Multi-round transmission, it means one transmission in one round. When multiple DMA requests are received, multiple rounds of transmission are required. Multiple round x multiple transmission is not supported
[0]	EN	Channel enable, active high. Set to 1 by software to enable the channel and start DMA handling. This bit is cleared by DMA after the handling operation is completed.

8.6.5.2 DMA_RENx (x = 0,1,2,3,4,5,6,7)

The addresses are: 0x4001_9C04, 0x4001_9C1C, 0x4001_9C34, 0x4001_9C4C, 0x4001_9C64, 0x4001_9C7C, 0x4001_9C94, and 0x4001_9CAC

Reset value: 0x0

Table 8-7 DMA Request Enable Register (DMA_RENx)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SW_TRIG		GPIO	FMAC_Y	FMAC_X1	CMP	HALL1	HALL0	I2C1 TX	I2C1 RX	UART2 TX	UART2 RX	UART1 TX	UART1 RX	UART0 TX	UART0 RX
RW		RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

I2C0 TX	I2C0 RX	MCPWM1	MCPWM0	SPI1 TX	SPI1 RX	SPI0 TX	SPI0 RX	TIMER4	TIMER3	TIMER2	TIMER1	TIMER0	ADC2	ADC1	ADC0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The memory and peripheral address set in the DMA channel should correspond to the enabled peripheral interrupt request, which should be guaranteed by the application software. The software request is always enabled, that is, the software writes 1 to the DMA_RENx.SW_TRIG bit to start a DMA transfer. Generally, each DMA channel can only enable one hardware DMA request at the same time. Software trigger flag DMA_RENx.SW_TRIG should be cleared by software after being handled by DMA.

Because CMP signal may be a slow level signal, it is recommended that the interrupt trigger type be edge trigger when using CMP to trigger DMA.

DMA_RENx register can be overwritten with DMA channel enabled.

8.6.5.3 DMA_CTMSx (x = 0,1,2,3,4,5,6,7)

The addresses are: 0x4001_9C08, 0x4001_9C20, 0x4001_9C38, 0x4001_9C50, 0x4001_9C68, 0x4001_9C80, 0x4001_9C98, and 0x4001_9CB0

Reset value: 0x0

Table 8-8 DMA Transfer Count Register (DMA_CTMSx)

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

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	TIMES	DMA channel x data transfer times per round. This register becomes read-only after the channel is enabled.

The DMA_CTMSx register can only write data after the channel is disabled, ie DMA_CCRx.EN = 0.

When DMA_CCRx.RMODE=0, it means transmit one round, and transmit DMA_CTMS times of data in each round;

When DMA_CCRx.RMODE=1, it means transmit DMA_CTMS round, and transmit data once in each round;

LKS32MC45x does not support multiple transmission in each round.

When peripheral data width is 16 and memory data width is 32, that is, CTMS.TIMES=16,

DMA_CCRx.RMODE=0, DMA should read peripheral data 16bit (2byte) ×16=32byte and write memory data 32bit (4byte) ×16=64byte each round, that is, reading 32 bytes of peripherals and entering 64 bytes of memory.

It is necessary to set CTMS.ROUND = 1 instead of setting it as 0, even if only one round is carried.

When DMA_CCRx.CIRC=1 (that is, cyclic mode), DMA_CTMSx no longer works, which is equivalent to an infinite round; when DMA_CTMSx is decreased to 0, it will be reloaded to the preset value to start a new round of transmission. In other cases, DMA_CTMSx should be set accordingly, such as MA_CTMSx=1 and DMA_CCRx.RMODE=1, which is used to carry one round of data.

When DMA_CCRx.RMODE=1, DMA_CTMSx reads the number of unhandled rounds, which will be reset to the configured value after completing DMA transmission. The number of times read is always the configured value. For example, if DMA_CTMSx=4 is configured, you may see that DMA_CTMSx is decremented from 4, and then 3, 2, 1, ... and then become 0. When the current channel of DMA needs to perform 4 rounds of transmission again, it is necessary to rewrite the round value to DMA_CTMSx. When DMA_CCRx.RMODE=0, DMA_CTMSx is read as the number of remaining unhandled times in the current round. However, since the data in a round is continuously handled by DMA, the DMA_CTMSx read by the software will change rapidly.

Note that before re-opening the DMA channel, the DMA_CTMSx register must be reconfigured. The DMA_CTMSx register has been decremented to 0 in the previous transfer. When setting the DMA_CTMSx register, ensure that DMA_CTRL=1, that is, DMA is enabled; otherwise, writing to DMA_CTMSx will be invalid.

8.6.5.4 DMA_SADRx (x = 0,1,2,3,4,5,6,7)

The addresses are: 0x4001_9C0C, 0x4001_9C24, 0x4001_9C3C, 0x4001_9C54, 0x4001_9C6C, 0x4001_9C84, 0x4001_9C9C, and 0x4001_9CB4

Reset value: 0x0

Table 8-9 DMA Source Address Register (DMA_SADRx)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDR															
RW															
0															

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

Bit field	Bit Name	Description
[31:0]	ADDR	DMA channel x source address

When DMA_CCRx.SBTW=2'b01, it is set to carry source data in units of 16-bit. If the value of DMA_SADRx.ADDR[0] is invalid, the source address will be incremented by 2.

When DMA_CCRx.SBTW=2'b10, it is set to carry source data in units of 32bit-bit. If the value of DMA_SADRx.ADDR [1:0] is invalid, the source address will be incremented by 4.

Note: The DMA_SADRx register can only write data after the channel is disabled, ie DMA_CCRx.EN=0!!!

8.6.5.5 DMA_DADRx (x = 0,1,2,3,4,5,6,7)

The addresses are: 0x4001_9C10, 0x4001_9C28, 0x4001_9C40, 0x4001_9C58, 0x4001_9C70, 0x4001_9C88, 0x4001_9CA0, and 0x4001_9CB8

Reset value: 0x0

Table 8-10 DMA Destination Address Register (DMA_DADRx)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDR															
RW															
0															

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

Bit field	Bit Name	Description
[31:0]	ADDR	DMA channel x destination address

When DMA_CCRx.DBTW=2'b01, it is set to carry data in units of 16-bit to the destination address. If the value of DMA_DADRx.ADDR[0] is invalid, the destination address will be incremented by 2.

When DMA_CCRx.DBTW=2'b10, it is set to carry memory data in units of 32-bit. If the value of DMA_DADRx.ADDR[1:0] is invalid, the destination address will be incremented by 4.

Note: The DMA_DADRx register can only write data after the channel is disabled, ie DMA_CCRx.EN=0!!!

9 GPIO

9.1 Overview

The LSK32MC45x series chip integrates 84 GPIOs. 8 GPIOs can be used as the wake-up source of the system, and 34 of which can be used as external interrupt source input.

9.1.1 Functional block diagram

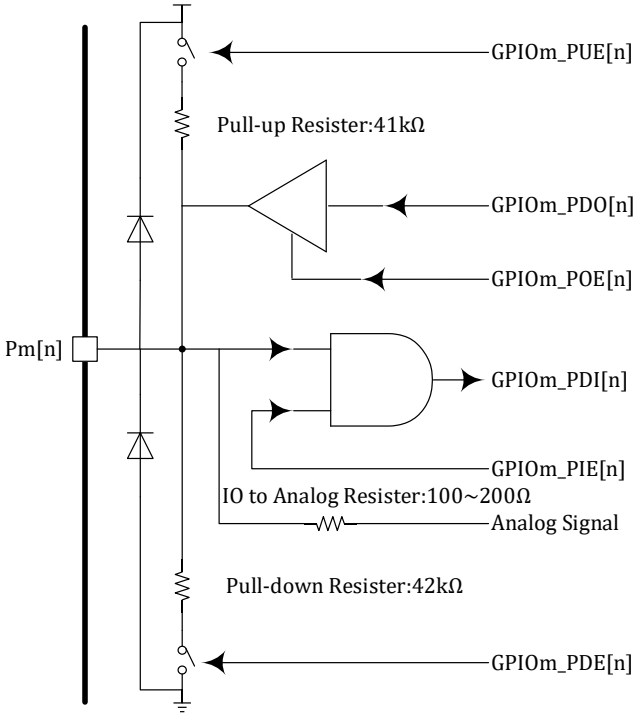


Figure 9-1 GPIO Functional Block Diagram

As shown in Figure 9-1, Pm [n] is the chip PAD, m can be 0-5, which means any group of six groups of GPIO, n can be 0-15, which means one IO in a group of 16-bit GPIO. The analog signal is directly connected to the PAD through a resistor in series,the resistance value is 100 to 200Ω. The digital signal is output through a three-state gate. When the output enables GPIOm_POE [n]=0, the buffer outputs a high-impedance state; otherwise, the buffer output is at the same level as GPIOm_PDO [n]. Digital signal input enters the chip through an AND gate. When GPIOm_PIE [n]=0, GPIOm_PDI [n] is always 0; when GPIOm_PIE [n]=1, that is, the input enable is turned on, the logic level of GPIOm_PDI [n] is at the same level as Pm [n]. The chip PAD can be configured with pull-up/pull-down resistors, with the typical value of pull-up resistance of 41kΩ and of pull-down resistance of 42kΩ. Due to the deviation of chip manufacturing process, the pull-up/down resistance value may have some deviation.

9.1.2 Product features

- 5 sets of 16bit GPIOs, 1 set of 4bit GPIO, totaling 84 GPIOs
- The push-pull open-drain mode can be configured
- Support configuration lock protection
- Support GPIO external interrupt
- Support GPIO wake-up

9.2 Register

9.2.1 Address Allocation

The base address of the GPIO 0 module in the chip is 0x4001_6400.

The base address of the GPIO 1 module in the chip is 0x4001_6440.

The base address of the GPIO 2 module in the chip is 0x4001_6480.

The base address of the GPIO 3 module in the chip is 0x4001_64C0.

The base address of the GPIO 4 module in the chip is 0x4001_6500.

The base address of the GPIO 5 module in the chip is 0x4001_6540.

The base address of the EXTI module in the chip is 0x4001_6580.

Except for the base address, the register definitions of GPIO 0/1/2/3/4/5 are the same. However, P5 only supports P5[3:0] and does not support P5[15:4].

Table 9-1 GPIOx Register List

Name	Offset Address	Description
GPIOx_PIE	0x00	GPIO x input enable
GPIOx_POE	0x04	GPIO x output enable
GPIOx_PDI	0x08	GPIO x input data
GPIOx_PDO	0x0C	GPIO x output data
GPIOx_PUE	0x10	GPIO x pull-up enable
GPIOx_PDE	0x14	GPIO x pull-down enable
GPIOx_PODE	0x18	GPIO x open-drain enable
GPIOx_PFLT	0x1C	GPIO x filter enable
GPIOx_F3210	0x20	GPIO x [3:0] function selection
GPIOx_F7654	0x24	GPIO x [7:4] function selection
GPIOx_FBA98	0x28	GPIO x [11:8] function selection
GPIOx_FFEDC	0x2C	GPIO x [15:12] function selection
GPIOx_BSRR	0x30	GPIO x bit operation register

GPIOx_BRR	0x34	GPIO x bit clear register

The base address of the GIPO interrupt/DMA module is 0x4001_6580.

Table 9-2 List of GPIO Interrupt/DMA Module Register

Name	Offset Address	Description
EXTI0_CR0	0x00	GPIO 0[7:0] interrupt/DMA trigger type
EXTI0_CR1	0x04	GPIO 0[15:8] interrupt/DMA trigger type
EXTI1_CR0	0x08	GPIO 1[7:0] interrupt/DMA trigger type
EXTI1_CR1	0x0C	GPIO 1[15:8] interrupt/DMA trigger type
EXTI2_CR0	0x10	GPIO 2[7:0] interrupt/DMA trigger type
EXTI2_CR1	0x14	GPIO 2[15:8] interrupt/DMA trigger type
EXTI3_CR0	0x18	GPIO 3[7:0] interrupt/DMA trigger type
EXTI3_CR1	0x1C	GPIO 3[15:8] interrupt/DMA trigger type
EXTI4_CR0	0x20	GPIO 4[7:0] interrupt/DMA trigger type
EXTI4_CR1	0x24	GPIO 4[15:8] interrupt/DMA trigger type
EXTI5_CR0	0x28	GPIO 5[7:0] interrupt/DMA trigger type
EXTI5_CR1	0x2C	GPIO 5[15:8] interrupt/DMA trigger type
EXTI0_IF	0x30	GPIO 0 interrupt flag
EXTI1_IF	0x34	GPIO 1 interrupt flag
EXTI2_IF	0x38	GPIO 2 interrupt flag
EXTI3_IF	0x3C	GPIO 3 interrupt flag
EXTI4_IF	0x40	GPIO 4 interrupt flag
EXTI5_IF	0x44	GPIO 5 interrupt flag
CLKO_SEL	0x50	Clock output selection
LCKR_PRT	0x54	GPIO protection
EXTI_REN	0x58	GPIO DMA request enable

9.2.2 GPIOx_PIE(x = 0,1,2,3,4,5)

The addresses are: 0x4001_6400, 0x4001_6440, 0x4001_6480, 0x4001_64C0, 0x4001_6500, and 0x4001_6540

Reset value: 0x0

Table 9-3 GPIOx Input Enable Register (GPIOx_PIE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIE15	PIE14	PIE13	PIE12	PIE11	PIE10	PIE9	PIE8	PIE7	PIE6	PIE5	PIE4	PIE3	PIE2	PIE1	PIE0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:16]		Unused
[15]	PIE15	GPIO x[15] / Px[15] input enable
[14]	PIE14	GPIO x[14] / Px[14] input enable
[13]	PIE13	GPIO x[13] / Px[13] input enable
[12]	PIE12	GPIO x[12] / Px[12] input enable
[11]	PIE11	GPIO x[11] / Px[11] input enable
[10]	PIE10	GPIO x[10] / Px[10] input enable
[9]	PIE9	GPIO x[9] / Px[9] input enable
[8]	PIE8	GPIO x[8] / Px[8] input enable
[7]	PIE7	GPIO x[7] / Px[7] input enable
[6]	PIE6	GPIO x[6] / Px[6] input enable
[5]	PIE5	GPIO x[5] / Px[5] input enable
[4]	PIE4	GPIO x[4] / Px[4] input enable
[3]	PIE3	GPIO x[3] / Px[3] input enable
[2]	PIE2	GPIO x[2] / Px[2] input enable
[1]	PIE1	GPIO x[1] / Px[1] input enable
[0]	PIE0	GPIO x[0] / Px[0] input enable

9.2.3 GPIOx_POE(x = 0,1,2,3,4,5)

The addresses are: 0x4001_6404, 0x4001_6444, 0x4001_6484, 0x4001_64C4, 0x4001_6504, and 0x4001_6544

Reset value: 0x0

Table 9-4 GPIOx Output Enable Register (GPIOx_POE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POE15	POE14	POE13	POE12	POE11	POE10	POE9	POE8	POE7	POE6	POE5	POE4	POE3	POE2	POE1	POE0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:16]		Unused
[15]	POE15	GPIO x[15] / Px[15] output enable
[14]	POE14	GPIO x[14] / Px[14] output enable
[13]	POE13	GPIO x[13] / Px[13] output enable
[12]	POE12	GPIO x[12] / Px[12] output enable

[11]	POE11	GPIO x[11] / Px[11] output enable
[10]	POE10	GPIO x[10] / Px[10] output enable
[9]	POE9	GPIO x[9] / Px[9] output enable
[8]	POE8	GPIO x[8] / Px[8] output enable
[7]	POE7	GPIO x[7] / Px[7] output enable
[6]	POE6	GPIO x[6] / Px[6] output enable
[5]	POE5	GPIO x[5] / Px[5] output enable
[4]	POE4	GPIO x[4] / Px[4] output enable
[3]	POE3	GPIO x[3] / Px[3] output enable
[2]	POE2	GPIO x[2] / Px[2] output enable
[1]	POE1	GPIO x[1] / Px[1] output enable
[0]	POE0	GPIO x[0] / Px[0] output enable

9.2.4 GPIOx_PDI(x = 0,1,2,3,4,5)

The addresses are: 0x4001_6408, 0x4001_6448, 0x4001_6488, 0x4001_64C8, 0x4001_6508, and 0x4001_6548

Reset value: 0x0

Table 9-5 GPIOx Input Data Register (GPIOx_PDI)

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

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	PDI	GPIO x input data

When GPIOx PIE=0, GPIOx PDI read-back is 0.

9.2.5 GPIOx_PDO(x = 0,1,2,3,4,5)

The addresses are: 0x4001_640C, 0x4001_644C, 0x4001_648C, 0x4001_64CC, 0x4001_650C, and 0x4001_654C

Reset value: 0x0

Table 9-6 GPIOx Output Data Register (GPIOx_PDO)

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

0

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	PDO	GPIO x output data

9.2.6 GPIOx_PUE(x = 0,1,2,3,4,5)

The addresses are: 0x4001_6410, 0x4001_6450, 0x4001_6490, 0x4001_64D0, 0x4001_6510, 0x4001_6550

Reset value: 0x0

Table 9-7 GPIOx pull-up Enable Register (GPIOx_PUE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUE15	PUE14	PUE13	PUE12	PUE11	PUE10	PUE9	PUE8	PUE7	PUE6	PUE5	PUE4	PUE3	PUE2	PUE1	PUE0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:16]		Unused
[15]	PUE15	GPIO x[15] / Px[15] pull-up enable
[14]	PUE14	GPIO x[14] / Px[14] pull-up enable
[13]	PUE13	GPIO x[13] / Px[13] pull-up enable
[12]	PUE12	GPIO x[12] / Px[12] pull-up enable
[11]	PUE11	GPIO x[11] / Px[11] pull-up enable
[10]	PUE10	GPIO x[10] / Px[10] pull-up enable
[9]	PUE9	GPIO x[9] / Px[9] pull-up enable
[8]	PUE8	GPIO x[8] / Px[8] pull-up enable
[7]	PUE7	GPIO x[7] / Px[7] pull-up enable
[6]	PUE6	GPIO x[6] / Px[6] pull-up enable
[5]	PUE5	GPIO x[5] / Px[5] pull-up enable
[4]	PUE4	GPIO x[4] / Px[4] pull-up enable
[3]	PUE3	GPIO x[3] / Px[3] pull-up enable
[2]	PUE2	GPIO x[2] / Px[2] pull-up enable
[1]	PUE1	GPIO x[1] / Px[1] pull-up enable
[0]	PUE0	GPIO x[0] / Px[0] pull-up enable

9.2.7 GPIOx_PDE(x = 0,1,2,3,4,5)

The addresses are: 0x4001_6414, 0x4001_6454, 0x4001_6494, 0x4001_64D4, 0x4001_6514, 0x4001_6554

Reset value: 0x0

Table 9-8 GPIOx pull-down Enable Register (GPIOx_PDE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDE15	PDE14	PDE13	PDE12	PDE11	PDE10	PDE9	PDE8	PDE7	PDE6	PDE5	PDE4	PDE3	PDE2	PDE1	PDE0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:16]		Unused
[15]	PDE15	GPIO x[15] / Px[15] Pull-down enable
[14]	PDE14	GPIO x[14] / Px[14] Pull-down enable
[13]	PDE13	GPIO x[13] / Px[13] Pull-down enable
[12]	PDE12	GPIO x[12] / Px[12] Pull-down enable
[11]	PDE11	GPIO x[11] / Px[11] Pull-down enable
[10]	PDE10	GPIO x[10] / Px[10] Pull-down enable
[9]	PDE9	GPIO x[9] / Px[9] Pull-down enable
[8]	PDE8	GPIO x[8] / Px[8] Pull-down enable
[7]	PDE7	GPIO x[7] / Px[7] Pull-down enable
[6]	PDE6	GPIO x[6] / Px[6] Pull-down enable
[5]	PDE5	GPIO x[5] / Px[5] Pull-down enable
[4]	PDE4	GPIO x[4] / Px[4] Pull-down enable
[3]	PDE3	GPIO x[3] / Px[3] Pull-down enable
[2]	PDE2	GPIO x[2] / Px[2] Pull-down enable
[1]	PDE1	GPIO x[1] / Px[1] Pull-down enable
[0]	PDE0	GPIO x[0] / Px[0] Pull-down enable

9.2.8 GPIOx_PODE(x = 0,1,2,3,4,5)

The addresses are: 0x4001_6418, 0x4001_6458, 0x4001_6498, 0x4001_64D8, 0x4001_6518, 0x4001_6558

Reset value: 0x0

Table 9-9 GPIOx Open-drain Enable Register (GPIOx_PODE)

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

PODE15	PODE14	PODE13	PODE12	PODE11	PODE10	PODE9	PODE8	PODE7	PODE6	PODE5	PODE4	PODE3	PODE2	PODE1	PODE0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:16]		Unused
[15]	PODE15	GPIO x[15] / Px[15] open-drain enable
[14]	PODE14	GPIO x[14] / Px[14] open-drain enable
[13]	PODE13	GPIO x[13] / Px[13] open-drain enable
[12]	PODE12	GPIO x[12] / Px[12] open-drain enable
[11]	PODE11	GPIO x[11] / Px[11] open-drain enable
[10]	PODE10	GPIO x[10] / Px[10] open-drain enable
[9]	PODE9	GPIO x[9] / Px[9] open-drain enable
[8]	PODE8	GPIO x[8] / Px[8] open-drain enable
[7]	PODE7	GPIO x[7] / Px[7] open-drain enable
[6]	PODE6	GPIO x[6] / Px[6] open-drain enable
[5]	PODE5	GPIO x[5] / Px[5] open-drain enable
[4]	PODE4	GPIO x[4] / Px[4] open-drain enable
[3]	PODE3	GPIO x[3] / Px[3] open-drain enable
[2]	PODE2	GPIO x[2] / Px[2] open-drain enable
[1]	PODE1	GPIO x[1] / Px[1] open-drain enable
[0]	PODE0	GPIO x[0] / Px[0] open-drain enable

9.2.9 GPIOx_PFLT(x = 0,1,2,3,4,5)

The addresses are: 0x4001_641C, 0x4001_645C, 0x4001_649C, 0x4001_64DC, 0x4001_651C, 0x4001_655C

Reset value: 0x0

Table 9-10 GPIOx Configuration Filter Register (GPIOx_PFLT)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PFLT 15	PFLT 14	PFLT 13	PFLT 12	PFLT 11	PFLT 10	PFLT 9	PFLT 8	PFLT 7	PFLT 6	PFLT 5	PFLT 4	PFLT 3	PFLT 2	PFLT1	PFLT0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:16]		Unused
[15]	PFLT15	GPIO x[15] / Px[15] configuration filtering
[14]	PFLT14	GPIO x[14] / Px[14] configuration filtering
[13]	PFLT13	GPIO x[13] / Px[13] configuration filtering
[12]	PFLT12	GPIO x[12] / Px[12] configuration filtering
[11]	PFLT11	GPIO x[11] / Px[11] configuration filtering
[10]	PFLT10	GPIO x[10] / Px[10] configuration filtering
[9]	PFLT9	GPIO x[9] / Px[9] configuration filtering
[8]	PFLT8	GPIO x[8] / Px[8] configuration filtering
[7]	PFLT7	GPIO x[7] / Px[7] configuration filtering
[6]	PFLT6	GPIO x[6] / Px[6] configuration filtering
[5]	PFLT 5	GPIO x[5] / Px[5] configuration filtering
[4]	PFLT 4	GPIO x[4] / Px[4] configuration filtering
[3]	PFLT 3	GPIO x[3] / Px[3] configuration filtering
[2]	PFLT 2	GPIO x[2] / Px[2] configuration filtering
[1]	PFLT1	GPIO x[1] / Px[1] configuration filtering
[0]	PFLT0	GPIO x[0] / Px[0] configuration filtering

9.2.10 GPIOx_F3210(x = 0,1,2,3,4,5)

The addresses are: 0x4001_6420, 0x4001_6460, 0x4001_64A0, 0x4001_64E0, 0x4001_6520, 0x4001_6560

Reset value: 0x0

Table 9-11 GPIOx Function Selection Register (GPIOx_F3210)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F3				F2				F1				F0			
RW				RW				RW				RW			
0				0				0				0			

Bit field	Bit Name	Description
[31:16]		Unused
[15:12]	F3	GPIO x[3] / Px[3] function selection
[11:8]	F2	GPIO x[2] / Px[2] function selection
[7:4]	F1	GPIO x[1] / Px[1] function selection
[3:0]	F0	GPIO x[0] / Px[0] function selection

GPIO multiplexing selection is as shown in Table 9-12

Table 9-12 GPIO Multiplexing

GPIOx_Fxxxx Configuration Value	2nd Function Code	Function
0x0	AF0	Analog function When using the analog function, turn off IE and OE corresponding to GPIO
0x1	AF1	Comparator output or clock output function
0x2	AF2	HALL
0x3	AF3	MCPWM
0x4	AF4	UART
0x5	AF5	SPI
0x6	AF6	IIC
0x7	AF7	Timer0(QEP0)/Time1(QEP1)/Time4
0x8	AF8	Timer2(QEP2)/Timer3(QEP3)
0x9	AF9	ADC trigger debug
0xA	AF10	CAN
0xB	AF11	External SPI flash

9.2.11 GPIOx_F7654(x = 0,1,2,3,4,5)

The addresses are: 0x4001_6424, 0x4001_6464, 0x4001_64A4, 0x4001_64E4, 0x4001_6524, 0x4001_6564

Reset value: 0x0

Table 9-13 GPIOx Function Selection Register (GPIOx_F7654)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F7				F6				F5				F4			
RW				RW				RW				RW			
0				0				0				0			

Bit field	Bit Name	Description
[31:16]		Unused
[15:12]	F7	GPIO x[7] / Px[7] function selection
[11:8]	F6	GPIO x[6] / Px[6] function selection
[7:4]	F5	GPIO x[5] / Px[5] function selection
[3:0]	F4	GPIO x[4] / Px[4] function selection

9.2.12 GPIOx_FBA98(x = 0,1,2,3,4,5)

The addresses are: 0x4001_6428, 0x4001_6468, 0x4001_64A8, 0x4001_64E8, 0x4001_6528, 0x4001_6568

Reset value: 0x0

Table 9-14 GPIOx Function Selection Register (GPIOx_FBA98)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F11				F10				F9				F8			
RW				RW				RW				RW			
0				0				0				0			

Bit field	Bit Name	Description
[31:16]		Unused
[15:12]	F11	GPIO x[11] / Px[11] function selection
[11:8]	F10	GPIO x[10] / Px[10] function selection
[7:4]	F9	GPIO x[9] / Px[9] function selection
[3:0]	F8	GPIO x[8] / Px[8] function selection

9.2.13 GPIOx_FFEDC(x = 0,1,2,3,4,5)

The addresses are: 0x4001_642C, 0x4001_646C, 0x4001_64AC, 0x4001_64EC, 0x4001_652C, 0x4001_656C

Reset value: 0x0

Table 9-15 GPIOx Function Selection Register (GPIOx_FFEDC)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F15				F14				F13				F12			
RW				RW				RW				RW			
0				0				0				0			

Bit field	Bit Name	Description
[31:16]		Unused
[15:12]	F15	GPIO x[15] / Px[15] function selection
[11:8]	F14	GPIO x[14] / Px[14] function selection
[7:4]	F13	GPIO x[13] / Px[13] function selection
[3:0]	F12	GPIO x[12] / Px[12] function selection

For a detailed list of GPIO function reuse, please refer to the corresponding device pin location in DATASHEET.

9.2.14 GPIOx_BSRR(x = 0,1,2,3,4,5)

The addresses are: 0x4001_6430, 0x4001_6470, 0x4001_64B0, 0x4001_64F0, 0x4001_6530,

0x4001_6570

Reset value: 0x0

Table 9-16 GPIOx Bit Operation Register (GPIOx_BSRR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLR15	CLR14	CLR13	CLR12	CLR11	CLR10	CLR9	CLR8	CLR7	CLR6	CLR5	CLR4	CLR3	CLR2	CLR1	CLR0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SET15	SET14	SET13	SET12	SET11	SET10	SET9	SET8	SET7	SET6	SET5	SET4	SET3	SET2	SET1	SET0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31]	CLR15	Write 1 to clear GPIO x[15], and writing 0 is invalid
[30]	CLR14	Write 1 to clear GPIO x[14], and writing 0 is invalid
[29]	CLR13	Write 1 to clear GPIO x[13], and writing 0 is invalid
[28]	CLR12	Write 1 to clear GPIO x[12], and writing 0 is invalid
[27]	CLR11	Write 1 to clear GPIO x[11], and writing 0 is invalid
[26]	CLR10	Write 1 to clear GPIO x[10], and writing 0 is invalid
[25]	CLR9	Write 1 to clear GPIO x[9], and writing 0 is invalid
[24]	CLR8	Write 1 to clear GPIO x[8], and writing 0 is invalid
[23]	CLR7	Write 1 to clear GPIO x[7], and writing 0 is invalid
[22]	CLR6	Write 1 to clear GPIO x[6], and writing 0 is invalid
[21]	CLR5	Write 1 to clear GPIO x[5], and writing 0 is invalid
[20]	CLR4	Write 1 to clear GPIO x[4], and writing 0 is invalid
[19]	CLR3	Write 1 to clear GPIO x[3], and writing 0 is invalid
[18]	CLR2	Write 1 to clear GPIO x[2], and writing 0 is invalid
[17]	CLR1	Write 1 to clear GPIO x[1], and writing 0 is invalid
[16]	CLR0	Write 1 to clear GPIO x[0], and writing 0 is invalid
[15]	SET15	Write 1 to set GPIO x[15] to 1, and writing 0 is invalid
[14]	SET14	Write 1 to set GPIO x[14] to 1, and writing 0 is invalid
[13]	SET13	Write 1 to set GPIO x[13] to 1, and writing 0 is invalid
[12]	SET12	Write 1 to set GPIO x[12] to 1, and writing 0 is invalid
[11]	SET11	Write 1 to set GPIO x[11] to 1, and writing 0 is invalid

[10]	SET10	Write 1 to set GPIO x[10] to 1, and writing 0 is invalid
[9]	SET9	Write 1 to set GPIO x[9] to 1, and writing 0 is invalid
[8]	SET8	Write 1 to set GPIO x[8] to 1, and writing 0 is invalid
[7]	SET7	Write 1 to set GPIO x[7] to 1, and writing 0 is invalid
[6]	SET6	Write 1 to set GPIO x[6] to 1, and writing 0 is invalid
[5]	SET5	Write 1 to set GPIO x[5] to 1, and writing 0 is invalid
[4]	SET4	Write 1 to set GPIO x[4] to 1, and writing 0 is invalid
[3]	SET3	Write 1 to set GPIO x[3] to 1, and writing 0 is invalid
[2]	SET2	Write 1 to set GPIO x[2] to 1, and writing 0 is invalid
[1]	SET1	Write 1 to set GPIO x[1] to 1, and writing 0 is invalid
[0]	SET0	Write 1 to set GPIO x[0] to 1, and writing 0 is invalid

If the high 16 bits and low 16 bits of BSRR are used to set a bit of GPIO to 1 and clear the bit at the same time, the bit will be cleared.

9.2.15 GPIOx_BRR(x = 0,1,2,3,4,5)

The addresses are: 0x4001_6434, 0x4001_6474, 0x4001_64B4, 0x4001_64F4, 0x4001_6534, 0x4001_6574

Reset value: 0x0

Table 9-17 GPIOx Bit Clear Register (GPIOx_BRR)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PFLT15	PFLT14	PFLT13	PFLT12	PFLT11	PFLT10	PFLT9	PFLT8	PFLT7	PFLT6	PFLT5	PFLT4	PFLT3	PFLT2	PFLT1	PFLT0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:16]		Unused
[15]	CLR15	Write 1 to clear GPIO x[15], and writing 0 is invalid
[14]	CLR14	Write 1 to clear GPIO x[14], and writing 0 is invalid
[13]	CLR13	Write 1 to clear GPIO x[13], and writing 0 is invalid
[12]	CLR12	Write 1 to clear GPIO x[12], and writing 0 is invalid
[11]	CLR11	Write 1 to clear GPIO x[11], and writing 0 is invalid
[10]	CLR10	Write 1 to clear GPIO x[10], and writing 0 is invalid
[9]	CLR9	Write 1 to clear GPIO x[9], and writing 0 is invalid
[8]	CLR8	Write 1 to clear GPIO x[8], and writing 0 is invalid
[7]	CLR7	Write 1 to clear GPIO x[7], and writing 0 is invalid

[6]	CLR6	Write 1 to clear GPIO x[6], and writing 0 is invalid
[5]	CLR5	Write 1 to clear GPIO x[5], and writing 0 is invalid
[4]	CLR4	Write 1 to clear GPIO x[4], and writing 0 is invalid
[3]	CLR3	Write 1 to clear GPIO x[3], and writing 0 is invalid
[2]	CLR2	Write 1 to clear GPIO x[2], and writing 0 is invalid
[1]	CLR1	Write 1 to clear GPIO x[1], and writing 0 is invalid
[0]	CLR0	Write 1 to clear GPIO x[0], and writing 0 is invalid

9.2.16 EXTIx_CR0(x = 0,1,2,3,4,5)

The addresses are: 0x4001_6580, 0x4001_6588, 0x4001_6590, 0x4001_6598, 0x4001_65A0, 0x4001_65A8

Reset value: 0x0

Table 9-18 GPIOx External Interrupt/DMA Configuration Register (EXTIx_CR0)

15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
T7	T6	T5	T4	T3	T2	T1	T0
RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:16]		Unused
[15:14]	T7	GPIO x[7]/Px[7] external interrupt/DMA trigger type selection 00: Not triggered 01: Falling edge trigger 10: Rising edge trigger 11: Trigger on both rising and falling edges
[13:12]	T6	GPIO x[6]/Px[6] external interrupt/DMA trigger type selection 00: Not triggered 01: Falling edge trigger 10: Rising edge trigger 11: Trigger on both rising and falling edges
[11:10]	T5	GPIO x[5]/Px[5] external interrupt/DMA trigger type selection 00: Not triggered 01: Falling edge trigger 10: Rising edge trigger 11: Trigger on both rising and falling edges
[9:8]	T4	GPIO x[4]/Px[4] external interrupt/DMA trigger type selection 00: Not triggered 01: Falling edge trigger 10: Rising edge trigger 11: Trigger on both rising and falling edges

[7:6]	T3	GPIO x[3]/Px[3] external interrupt trigger type selection 00: Not triggered 01: Falling edge trigger 10: Rising edge trigger 11: Trigger on both rising and falling edges
[5:4]	T2	GPIO x[2]/Px[2] external interrupt/DMA trigger type selection 00: Not triggered 01: Falling edge trigger 10: Rising edge trigger 11: Trigger on both rising and falling edges
[3:2]	T1	GPIO x[1]/Px[1] external interrupt/DMA trigger type selection 00: Not triggered 01: Falling edge trigger 10: Rising edge trigger 11: Trigger on both rising and falling edges
[1:0]	T0	GPIO x[0]/Px[0] external interrupt/DMA trigger type selection 00: Not triggered 01: Falling edge trigger 10: Rising edge trigger 11: Trigger on both rising and falling edges

9.2.17 EXTIx_CR1(x = 0,1,2,3,4,5)

The addresses are: 0x4001_6584, 0x4001_658C, 0x4001_6594, 0x4001_659C, 0x4001_65A4, 0x4001_65AC

Reset value: 0x0

Table 9-19 GPIOx External Interrupt Configuration Register (EXTIx_CR1)

15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
T15	T14	T13	T12	T11	T10	T9	T8
RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:16]		Unused
[15:14]	T15	GPIO x[15]/Px[15] external interrupt/DMA trigger type selection 00: Not triggered 01: Falling edge trigger 10: Rising edge trigger 11: Trigger on both rising and falling edges
[13:12]	T14	GPIO x[14]/Px[14] external interrupt/DMA trigger type selection 00: Not triggered

		01: Falling edge trigger 10: Rising edge trigger 11: Trigger on both rising and falling edges
[11:10]	T13	GPIO x[13]/Px[13] external interrupt/DMA trigger type selection 00: Not triggered 01: Falling edge trigger 10: Rising edge trigger 11: Trigger on both rising and falling edges
[9:8]	T12	GPIO x[12]/Px[12] external interrupt/DMA trigger type selection 00: Not triggered 01: Falling edge trigger 10: Rising edge trigger 11: Trigger on both rising and falling edges
[7:6]	T11	GPIO x[11]/Px[11] external interrupt/DMA trigger type selection 00: Not triggered 01: Falling edge trigger 10: Rising edge trigger 11: Trigger on both rising and falling edges
[5:4]	T10	GPIO x[10]/Px[10] external interrupt/DMA trigger type selection 00: Not triggered 01: Falling edge trigger 10: Rising edge trigger 11: Trigger on both rising and falling edges
[3:2]	T9	GPIO x[9]/Px[9] external interrupt/DMA trigger type selection 00: Not triggered 01: Falling edge trigger 10: Rising edge trigger 11: Trigger on both rising and falling edges
[1:0]	T8	GPIO x[8]/Px[8] external interrupt/DMA trigger type selection 00: Not triggered 01: Falling edge trigger 10: Rising edge trigger 11: Trigger on both rising and falling edges

9.2.18 EXTIx_IF(x = 0,1,2,3,4,5)

The addresses are: 0x4001_65B0, 0x4001_65B4, 0x4001_65B8, 0x4001_65BC, 0x4001_65C0, and 0x4001_65C4

Reset value: 0x0

Table 9-20 GPIOx External Interrupt Flag Register (EXTIx_IF)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IF15	IF14	IF13	IF12	IF11	IF10	IF9	IF8	IF7	IF6	IF5	IF4	IF3	IF2	IF1	IF0

RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:16]		Unused
[15]	IF15	GPIO x[15] / Px[15] external interrupt flag. Interrupt flag is active high, write 1 to clear
[14]	IF14	GPIO x[14] / Px[14] external interrupt flag. Interrupt flag is active high, write 1 to clear
[13]	IF13	GPIO x[13] / Px[13] external interrupt flag. Interrupt flag is active high, write 1 to clear
[12]	IF12	GPIO x[12] / Px[12] external interrupt flag. Interrupt flag is active high, write 1 to clear
[11]	IF11	GPIO x[11] / Px[11] external interrupt flag. Interrupt flag is active high, write 1 to clear
[10]	IF10	GPIO x[10] / Px[10] external interrupt flag. Interrupt flag is active high, write 1 to clear
[9]	IF9	GPIO x[9] / Px[9] external interrupt flag. Interrupt flag is active high, write 1 to clear
[8]	IF8	GPIO x[8] / Px[8] external interrupt flag. Interrupt flag is active high, write 1 to clear
[7]	IF7	GPIO x[7] / Px[7] external interrupt flag. Interrupt flag is active high, write 1 to clear
[6]	IF6	GPIO x[6] / Px[6] external interrupt flag. Interrupt flag is active high, write 1 to clear
[5]	IF5	GPIO x[5] / Px[5] external interrupt flag. Interrupt flag is active high, write 1 to clear
[4]	IF4	GPIO x[4] / Px[4] external interrupt flag. Interrupt flag is active high, write 1 to clear
[3]	IF3	GPIO x[3] / Px[3] external interrupt flag. Interrupt flag is active high, write 1 to clear
[2]	IF2	GPIO x[2] / Px[2] external interrupt flag. Interrupt flag is active high, write 1 to clear
[1]	IF1	GPIO x[1] / Px[1] external interrupt flag. Interrupt flag is active high, write 1 to clear
[0]	IF0	GPIO x[0] / Px[0] external interrupt flag. Interrupt flag is active high, write 1 to clear

Only some GPIOs can be used as external interrupt source input, which are as shown in the following table.

Table 9-21 GPIO Interrupt Resource Distribution

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P0	√	√	√								√	√	√	√	√	√
P1				√	√				√		√					
P2	√			√				√	√					√	√	√
P3	√	√	√	√	√			√	√				√	√		
P4														√	√	√
P5															√	√

9.2.19 CLKO_SEL

Address: 0x4001_65D0

Reset value: 0x0

Table 9-22 GPIO Output Clock Signal Selection Register (CLKO_SEL)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											HSE_OE	ADC_OE	PLL_OE	HSI_OE	LSI_OE
											RW	RW	RW	RW	RW
											0	0	0	0	0

Bit field	Bit Name	Description
[31:5]		Unused
[4]	HSE_OE	Crystal oscillator output enable. 1: Enable; 0: Disable.
[3]	ADC_OE	ADC clock output enable. 1: Enable; 0: Disable.
[2]	PLL_OE	PLL clock output enable. 1: Enable; 0: Disable.
[1]	HSI_OE	HRC clock output enable. 1: Enable; 0: Disable.
[0]	LSI_OE	LRC oscillator output enable. 1: Enable; 0: Disable.

CLKO_SEL is mainly used to output the internal clock of the chip, usually for debugging purposes.

9.2.20 LCKR_PRT

Address: 0x4001_65D4

Reset value: 0x0

Table 9-23 GPIO Protection Register (LCKR_PRT)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSW[15:1]															LOCK
WO															RW
0															0

Bit field	Bit Name	Description
[31:16]		Unused
[15:1]	PSW[15:1]	GPIO 保护密码高 15 位
[0]	PSW[0]/LOCK	Lowest bit of GPIO protection password. Write 0x5AC4 to LCKR_PRT[15:0] to release the GPIO configuration protection, and write other values to enable GPIO configuration protection; read LCKR_PRT[0] to get the current protection state, and read value of LCKR_PRT[15:1] is always 0. After enabling configuration protection, GPIOx_PIE/GPIOx_POE/GPIOx_PDO/GPIOx_PUE/GPIOx_PDE/GPIOx_PODE/GPIOx_F3210/GPIOx_F7654/GPIOx_FBA98/GPIOx_FFEDC/ GPIOx_BSRR/GPIOx_BRR cannot be written.

9.2.21 EXTI_REN

Address: 0x4001_65D8

Reset value: 0x0

Table 9-24 GPIO DMA Request Enable Register (EXTI_REN)

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

Bit field	Bit Name	Description
[31:6]		Unused
[5:0]	EXTI_REN	GPIO5~GPIO0 DMA request enable, and must be used with EXTIx_CR0/1

Each group of GPIO external request signals can be used uniformly for interrupt requests, or DMA requests, but the same group of signals is generally not used for both interrupt and DMA requests. DMA request signal is also generated from the interrupt flag setting. When the interrupt occurs, if corresponding bit of EXTI_REN is enabled, DMA will clear all the interrupt flags of the GPIO after responding to the request.

Only some GPIOs can be used as external interrupt and DMA request signals. For details, refer to the data manual.

9.3 Implementation Description

9.3.1 Pull-up

For LKS32MC45x series chips, all GPIOs are equipped with a pull-up resistor with the typical value of 41kΩ, which is controlled by GPIOx_PUE.

9.3.2 Pull-down

For LKS32MC45x series chips, all GPIOs are equipped with a pull-down resistor with the typical value of 42kΩ, which is controlled by GPIOx_PDE.

9.3.3 Filter

For LKS32MC45x series chips, some GPIOs support to filter the input signals, with the filter time width of 4 LSI clock cycles, about 120us, that is, the change of less than 120us will be filtered out. Note that because the filter uses the LSI clock and the RC has limited accuracy, the specific filter time constant may vary.

Table 9-25 GPIO Filter Resource Distribution

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P0			√	√	√	√	√				√	√	√	√		√
P1						√	√	√								
P2															√	√
P3									√				√			
P4	√	√												√	√	
P5															√	√

9.3.4 External interrupt

LKS32MC45x some GPIO can be used as external interrupt source, corresponding interrupt number 26 GPIO(EXTI). External interrupt can be triggered only after EXTIx_CR0 and EXTIx_CR1 are enabled.

Table 9-26 GPIO interrupt resource distribution

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P0	√	√	√								√	√	√	√	√	√
P1				√	√				√		√					
P2	√			√				√	√					√	√	√
P3	√	√	√	√	√			√	√				√	√		
P4														√	√	√
P5															√	√

9.3.5 External awaken

LKS32MC45x some GPIO can be used as external wake source.Need to configure the AON_IO_WAKE_EN enabled and set AON_IO_WAKE_POL to appropriate level.

Table 9-27 GPIO Wake-up resource distribution

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P0			√								√	√	√	√		
P1					√											
P2		√														

P3														√		
P4																
P5																

9.4 Application Guide

9.4.1 External Interrupt

For example:

```
GPIO0_PIE = 0x0080;                 // Enable P0[7] input
NVIC_EnableIRQ(GPIO_IRQn);          // Enable GPIO interrupt
__enable_irq();                     // Enable interrupt
i = 1000;
while(i--);
// P0 [7] External square wave signal on IO
EXTI0_CR0 = 0x8000;                 // Enable P0 [7] rising edge trigger and generate external interrupt
while(irq_flag != 2);               // External signal is flipped twice, two interrupts are generated, irq_flag increased twice in GPIO interrupt program
EXTI0_CR0 = 0x4000;                 // Enable p0 [7] falling edge trigger and generate external interrupt
while(irq_flag != 4);
EXTI0_CR0 = 0xC000;                 // Enable P0 [7] rising edge and falling edge trigger at the same time, and generate external interrupt
while(irq_flag != 8);
EXTI0_CR0 = 0x0000;                 // Disable P [7] rising and falling edge trigger at the same time, external interrupt cannot be generated
```

9.4.2 GPIO Analog Mode

Turn off the GPIO IE and OE to use the analog function. And then, the PAD is directly connected to the analog module through the internal resistance.

10 CRC

10.1 Overview

CRC, or Cyclic Redundancy Check Code: It is the most commonly used error check code in the field of data communication. Its characteristic is that the length of the information field and the check field can be selected at will. Cyclic Redundancy Check (CRC) is a data transmission error detection function that performs polynomial calculation on the data and appends the obtained result to the back of the frame. The receiving device also implements a similar algorithm to ensure the correctness and integrity of the data transmission.

The process of error detection using CRC can be described as to generate an R-bit CRC code for verification with certain rules at the transmit end according to the K-bit binary code sequence, and then append the CRC code to the original information to form a new binary code sequence number of K+R bits, and then transmit. After then, check at the receiving end according to the rules followed between the information code and the CRC code, to determine whether there is an error in transmission. This rule is called "generator polynomial" in error control theory.

10.2 Basic Principles

Append the R-bit check code to the K-bit information code, and the entire code length is N bits. Thus, such code is also called (N, K) code. For a given (N, K) code, it can be proved that there is a polynomial G (x) with the highest power of N-K=R. A K-bit information check code can be generated with a G (x), and G (x) is called the generator polynomial. The generation process of the check code is: Assuming that the information to be transmitted is polynomial C (X), and then shift C (x) to the left by R bits (C (x)*2R). Then, the R bit will be vacated to the right of C (x), which is the position of the check code. The remainder obtained by dividing C (x)*2R by the generator polynomial G (x) is the check code.

Any code composed of binary bit strings can correspond to a polynomial whose coefficients are only ‘0’or ‘1’. For example, the polynomialcorresponding to the code 1010111 is $x^6+x^4+x^2+x+1$, and the polynomial $x^5+x^3+x^2+x+1$ corresponds to the code 101111.

10.3 Basic Concepts

10.3.1 Correspondence

Direct correspondence between polynomials and binary numbers: The highest power of X corresponds to the highest bit of the binary number, and the following bits correspond to the powers of the polynomial. A term with the same power corresponds to 1, and a term without this corresponds to 0. It can be seen that the highest power of X is R, and the converted binary number has R+1 bits.

Polynomialsinclude generator polynomial G (X) and information polynomial C (X).

If the generator polynomial is $G(X)=X^4+X^3+X+1$, it can be converted into binary number 11011.

If the information transmitted is 101111, it can be converted into a data polynomial of $C(X)=X^5+X^3+X^2+X+1$.

10.3.2 Generator Polynomial

The generator polynomial is an agreement between the receiver and the sender, that is, a binary number. This number remains unchanged throughout the transmission process.

On the sender side, the generator polynomial is used to divide the information polynomial by 2 to generate a check code. On the receiving side, the generator polynomial is used to perform modulo-2 division detection on the received coding polynomial and determine the error location.

The following conditions should be met:

A. The highest and lowest bits of the generator polynomial must be 1.

B. When an error occurs in any bit of the transmitted information (CRC code), the remainder should not be "0" after being divided by the generator polynomial.

C. When errors occur in different bits, the remainder should be different.

D. If continue to divide the remainder, the remainder should be circulated.

10.3.3 CRC Digits

CRC check digits = generator polynomial digits - 1. Please note that some shorthands for generator polynomials have omitted the highest bit 1 of the generator polynomial.

10.3.4 Generation Steps

1. Convert the generator polynomial G (X) with the highest power of X to R into the corresponding R+1 binary number.

2. Shift the information code to the left by R bits, which is equivalent to the corresponding information polynomial C(X)*2R.

3. Divide the information code with a generator polynomial (binary number) to obtain the remainder of the R bits. Note: The remainder obtained by binary division is actually the remainder obtained by modulo 2 division, and it is not equal to the remainder obtained by dividing the corresponding decimal number.

4. Append the remainder to the information code, shift left and get the complete CRC code by vacating the position.

[Example] Suppose the generator polynomial used is G(X)=X3+X+1. The original 4-bit message is 1010. Find the encoded message.

Solution:

1. Convert the generator polynomial G(X)=X3+X+1 into the corresponding binary divisor 1011.

2. The generator polynomial in this question has 4 bits (R+1), and the original message C (X) is shifted to the left by 3 (R) bits to 1010 000. (Note: The check code calculated by the 4-bit generator polynomial is 3 bits, R is the number of check codes.)

3. Use the binary number corresponding to the generator polynomial to divide the original message shifted left by 3 bits by modulo 2 (high-bit alignment), which is equivalent to bitwise XOR:

1010000

1011

0001000

0001011

0000011

Obtained the remaining bits 011, so the final code is: 1010 011

POL=0x13, data=0x77

011101110000000

10010011

01111101000000

10010011

0110100100000

10010011

010000010000

10010011

00010001000

10010011

00011011

10.4 Register

10.4.1 Address Allocation

The base address of CRC is 0x4001_2400, and the register list is as follows:

Table 10-1 CRC Register List

Name	Offset Address	Description
CRC_DR	0x00	CRC data register (input information code/output code)
CRC_CR	0x04	CRC control register
CRC_INIT	0x08	CRC initial code register
CRC_POL	0x0C	Binary code register for CRC generator polynomial

10.4.2 Register Description

10.4.2.1CRC Data Register (CRC_DR)

Address: 0x4001_6800

Reset value: 0x0

Table 10-2 CRC Data Register (CRC_DR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DR															
RW															
0															

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

Bit field	Bit Name	Description
[31:0]	DR	Store the information code to be encoded and the code after CRC check

The CRC_DR register is used not only to put the data to be checked, but also to return the check result. Writing to the CRC_DR register triggers once CRC calculation. The data to be encoded should be written last after the configuration of CR and other registers is completed, thus to trigger the CRC calculation.

10.4.2.2 CRC Control Register (CRC_CR)

Address: 0x4001_6804

Reset value: 0x0

Table 10-3 CRC Control Register (CRC_CR)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			REV_OUT_TYPE			REV_IN_TYPE				POLY_SIZE					RESET
			RW			RW				RW					WO
			0			0				0					0

Bit field	Bit Name	Description
[31:13]		Unused
[12]	REV_OUT_TYPE	Whether to invert the code after CRC check and output, that is, b [31] = b [0], b [30] = b [1],... [b0] = b [31]
[11:10]		Unused
[9:8]	REV_IN_TYPE	Type of data inversion to be encoded 00: not reverse 01: Reverse by byte, that is, b [31] = b [24], b [30] = b [25],..., b [24] = b [31],..., b [7] = b [0], b [6] = b [1],..., b [0] = b [7] 10: Reverse by half word (16bit), that is b [31] = b [16], b [30] = b [17],..., b [16] = b [31],..., b [15] = b [0], b [14] = b [1],..., b [0] = b [15] 11: Reverse by words, that is, b [31] = b [0], b [30] = b [1],... [b0] = b [31]
[7:6]		Unused
[5:4]	POLY_SIZE	Output coding (polynomial) bit width 00: 32bits 01: 16bits 10: 8bits 11: 7bits
[3:1]		Unused
[0]	RESET	Data source for CRC calculation with input information code 0: from the last calculation result 1: from CRC_INIT Write 1 to reset the CRC data and automatically clear it. The value is always 0 after readback.

It should also be noted that writing 1 to CRC_CR.RESET will reset the CRC_INIT register to 0xFFFFFFFF.

If clearing the CRC calculation result is required, write 1 to CRC_CR.RESET; otherwise, the subsequent CRC calculation will take the previous calculation result as the initial value.

10.4.2.3CRC Initial Code Register (CRC_INIT)

Address: 0x4001_6808

Reset value: 0x0

Table 10-4 CRC Initial Code Register (CRC_INIT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INIT															
RW															
0xFFFFFFFF															

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

Bit field	Bit Name	Description
[31:0]	INIT	Store initial code

CRC_DR and CRC_INIT start to perform CRC check calculation after XOR.

10.4.2.4CRC Generation Code Register (CRC_POL)

Address: 0x4001_680C

Reset value: 0x0

Table 10-5 CRC Generation Code Register (CRC_POL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
POL															
RW															
0x04C11DB7															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POL															
RW															
0x04C11DB7															

Bit field	Bit Name	Description
[31:0]	POL	Store the generator code for the generator polynomial

11 CORDIC

11.1 Overview

The trigonometric CORDIC module has a bit width of 16 bits and a Q15 fixed-point format.

11.2 Register

11.2.1 Address Allocation

The base address of the DSP register in the chip is 0x4001_6C00.

Table 11-1 DSP Register List

Name	Offset	Description
DSP_SC	0x00	DSP status control register
DSP_THETA	0x04	DSP sin/cos input angle register
DSP_X	0x08	DSP arctan/module calculate input coordinates X register
DSP_Y	0x0C	DSP arctan/module calculate input coordinates Y register
DSP_SIN	0x10	DSP sin/cos calculation result sin register
DSP_COS	0x14	DSP sin/cos calculation result cos register
DSP_MOD	0x18	DSP arctan calculation result sqrt(X2+Y2) Register
DSP_ARCTAN	0x1C	DSP arctan calculation result arctan(Y/X) angle register

11.2.2 DSP Status Control Register (DSP_SC)

Address: 0x4001_6C00

Reset value: 0x2

Table 11-2 DSP Status Control Register (DSP_SC)

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

Bit field	Bit Name	Description
[31:3]		Reserved
[2]	MODE	CORDIC mode, 0: arctan, 1: sin/cos

[1:0]		Reserved

The selection of sin/cos mode and arctan mode, and the calculation of sin/cos or arctan by the CORDIC module share the same hardware circuit. Therefore, the appropriate mode selection should be done by setting the DSP_SC.MODE register before performing a certain calculation. When DSP_SC.MODE=1, the CORDIC module calculates sin/cos, the angle DSP_THETA is used as the input, and the sin/cos result is calculated and output to the register; when DSP_SC.MODE=0, the CORDIC module calculates arctan, the coordinates x/y are used as input, and the angles theta = arctan (y/x) and module = sqrt (x2+y2) are calculated and output to the register.

11.2.3 DSP sin/cos Register

Since the calculation of sin/cos and arctan in the CORDIC module calculation uses the same data path, it's necessary to write DSP_SC[2] to 1 before performing the calculation of sin/cos and arctan in the CORDIC module through the CPU, to make CORDIC enter sin/cos mode.

11.2.3.1 DSP_THETA

Address: 0x4001_6C04

Reset value: 0x0

Table 11-3 DSP sin/cos Angle Input Register

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

Bit field	Bit Name	Description
[31:16]		Reserved bit. Sign extension when reading, ie {16 {DSP_THETA [15]}}
[15:0]	THETA	DSP sin/cos input angle register

DSP_THETA is a 16-bit signed fixed-point number, representing the range (-32768 ~ 32767) corresponding to (–π ~ π). DSP_THETA is a write-only register, and writing will trigger the trigonometric function calculation once. DSP_SIN/DSP_COS uses DSP_THETA as the angle input and completes the calculation.

11.2.3.2 DSP_SIN

Address: 0x4001_6C10

Reset value: 0x0

Table 11-4 DSP sin/cos Sine Result Register

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

Bit field	Bit Name	Description
[31:16]		Reserved bit. Sign extension when reading, ie {16{DSP_SIN[15]}}
[15:0]	SIN	DSP sin/cos calculation result sin register

DSP_SIN is a 16-bit signed fixed-point number, including 1-bit sign bit, 1-bit integer bit, and 15-bit decimal bit; representing the range (–1 ~ 1).

11.2.3.3 DSP_COS

Address: 0x4001_6C14

Reset value: 0x0

Table 11-5 DSP sin/cos Cosine Result Register

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

Bit field	Bit Name	Description
[31:16]		Reserved bit. Sign extension when reading, ie {16{DSP_COS[15]}}
[15:0]	COS	DSP sin/cos calculation result cos register

DSP_COS is a 16-bit signed fixed-point number, including 1-bit sign bit, 1 -bit integer bit, and 15-bit decimal bit; representing the range (–1 ~ 1).

11.2.4 DSP arctan Register

11.2.4.1 DSP_X

Address: 0x4001_6C08

Reset value: 0x0

Table 11-6 DSP Arctan/Module Coordinate X Input Register

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

Bit field	Bit Name	Description
[15:0]	X	DSP arctan/module calculate input coordinates Y register

DSP_X is a 16-bit signed fixed-point number, including 1-bit sign bit, 15-bit integer bit; representing the range (–32768 ~ 32767).

DSP_X shall be written first when calculating arctan (DSP_Y/DSP_X).

11.2.4.2 DSP_Y

Address: 0x4001_6C0C

Reset value: 0x0

Table 11-7 DSP Arctan/Module Coordinate Y Input Register

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

Bit field	Bit Name	Description
[15:0]	Y	DSP arctan/module calculate input coordinates Y register

DSP_Y is a 16-bit signed fixed-point number, including 1-bit sign bit, 15-bit integer bit; representing the range (–32768 ~ 32767).

When calculating arctan(Y/X), first write DSP_X and then DSP_Y. Writing to DSP_Y triggers arctan calculation once.

11.2.4.3 DSP_MOD

Address: 0x4001_6C18

Reset value: 0x0

Table 11-8 DSP arctan Angle Result (Y/X) Register

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

MOD
RO
0

Bit field	Bit Name	Description
[31:16]		Reserved bit. Always read as 0.
[15:0]	MOD	DSP arctan calculation result sqrt(X2+Y2) Register

DSP_MOD is a 16-bit signed fixed-point number, representing the range (0 ~ 65535).

11.2.4.4 DSP_ARCTAN

Address: 0x4001_6C1C

Reset value: 0x0

Table 11-9 DSP arctan Angle Result (Y/X) Register

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

Bit field	Bit Name	Description
[31:16]		Reserved bit. Sign extension when reading, ie {16{DSP_ARCTAN[15]}}
[15:0]	ARCTAN	DSP arctan calculation result arctan(Y/X) angle register

DSP_is a 16-bit signed fixed-point number, representing the range (-32768 ~ 32767) corresponding to (–π ~ π).

12 Filter math accelerator (FMAC)

12.1 FMAC introduction

The filter math accelerator unit performs arithmetic operations on vectors. It comprises a multiplier/accumulator (MAC) unit, together with address generation logic which allows it to index vector elements held in local memory.

The unit includes support for circular buffers on input and output, which allows digital filters to be implemented. Both finite and infinite impulse response filters can be realized.

The unit allows frequent or lengthy filtering operations to be offloaded from the CPU, freeing up the processor for other tasks. In many cases it can accelerate such calculations compared to a software implementation, resulting in a speed-up of time critical tasks.

12.2 FMAC main features

- 16-bit x 16-bit multiplier
- 24+2-bit accumulator, supporting saturation process
- 16-bit data input/output
- 256×16-bit local data storage
- Local memory can define up to 3 data cache areas (two input caches and one output cache). The cache base address and cache size can be configured via registers
- Input and output caches can be used as circular buffer
- Filtering functions: FIR, direct Type 1 IIR
- Vector operations: Dot product, convolution, correlation
- AHB bus interface
- Supporting DMA read/write data

12.3 Function Description

The FMAC is shown in Figure 12-1

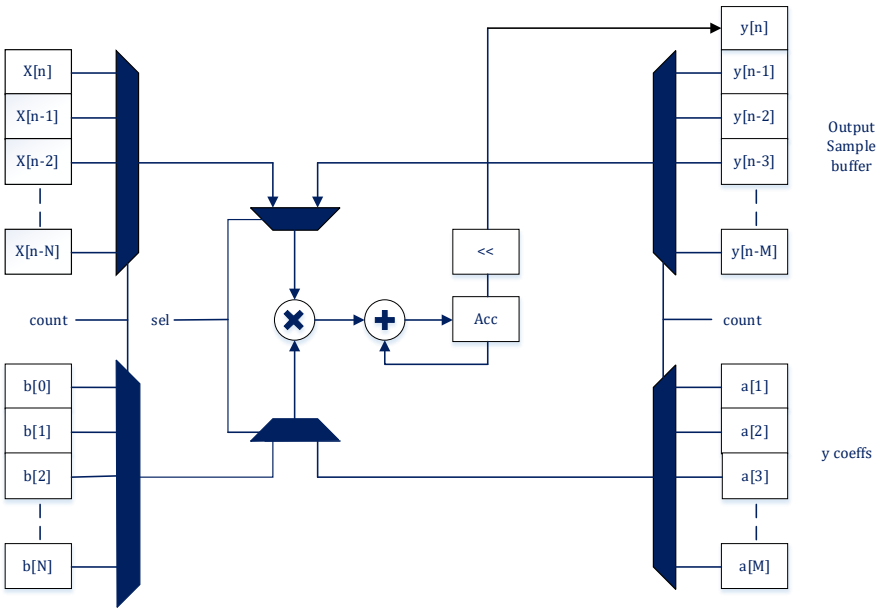


Figure 12-1 FMAC Block diagram

The unit is built around a fixed−point multiplier and accumulator (MAC). The MAC can take two 16-bit input signed values from memory, multiply them together and add them to the contents of the accumulator. The address of the input values in memory is determined using a set of pointers. These pointers can be loaded, incremented, decremented or reset by the internal hardware. The pointer and MAC operations are controlled by a built-in sequencer in order to execute the requested operation.

To calculate a dot product, the two input vectors are loaded into the local memory by the processor or DMA controller, and the requested operation is selected and started. Each pair of input vector elements is fetched from memory, multiplied together and accumulated.

When all the vector elements have been processed, the contents of the accumulator are stored in the local memory, from where they can be read out by the processor or DMA.

The finite impulse response (FIR) filter operation (also known as convolution) consists in repeatedly calculating the dot product of the coefficient vector and a vector of input samples, the latter being shifted by one sample delay, with the least recent sample being discarded and a new sample added, at each repetition.

The infinite impulse response (IIR) filter operation is the convolution of the feedback coefficients with the previous output samples, added to the result of the FIR convolution.

A more detailed description of the filter operations is given in 12.3.5 Filter functions

12.3.1 Local memory and buffers

The unit contains a 256 x 16-bit read/write memory which is used for local storage:

- Input values (the elements of the input vectors) are stored in two buffers, X1 andX2.
- Output values (the results of the operations) are stored in another buffer, Y.
- The locations and sizes of the buffers are designated as follows:
  - x1_base: the base address of the X1 buffer
  - x2_base: the base address of the X2 buffer
  - y_base: the base address of the Y buffer
  - x1_buf_size: the number of 16-bit addresses allocated to the X1 buffer
  - x2_buf_size: the number of 16-bit addresses allocated to the X2 buffer
  - y_buf_size: the number of 16-bit addresses allocated to the Y buffer.

These parameters are programmed in the corresponding registers when configuring the unit.

The CPU (or DMA controller) can initialize the contents of each buffer using the Initialization functions (12.3.4 Initialization functions) and writing to the write data register. The data is transferred to the location within the target buffer indicated by a write pointer. After each new write, the write pointer is incremented. When the write pointer reaches the end of the allocated buffer space, it wraps back to the base address. This feature is used to load the elements of a vector prior to an operation, or to initialize a filter and load filter coefficients.

Buffer configuration

The buffer sizes and base address offsets must be configured in the X1, X2 and Y buffer configuration registers. For each function, the required buffer size is specified in the function description in 12.3.5 Filter functions. The base addresses can be chosen anywhere in internal memory, provided that all buffers fit within the internal memory address range (0x00 to 0xFF), that is, base address + buffer size must be less than 256.

There is no constraint on the size and location of the buffers (they can overlap or even coincide exactly). For filter functions it is recommended not to overlap buffers as this can lead to erroneous behavior.

When circular buffer operation is required, an optional “headroom”, d, can be added to the buffer size. Furthermore, a watermark level can be set, to regulate the CPU or DMA activity. The value of d and the watermark level should be chosen according to the application performance requirements. For maximum throughput, the input buffer should never go empty, so d should be somewhat greater than the watermark level, allowing for any interrupt or DMA latency. On the other hand, if the input data can not be provided as fast as the unit can process them, the buffer can be allowed to empty waiting for the next data to be written, so d can be equal to the watermark level (to ensure that no overflow occurs on theinput).

12.3.2 Input buffers

The X1 and X2 buffers are used to store data for input to the MAC. Each multiplication takes a value from the X1 buffer and a value from the X2 buffer and multiplies them together. A pointer in the control unit generates the read address offset (relative to the buffer base address) for each value. The

pointers are managed by hardware according to the current function.

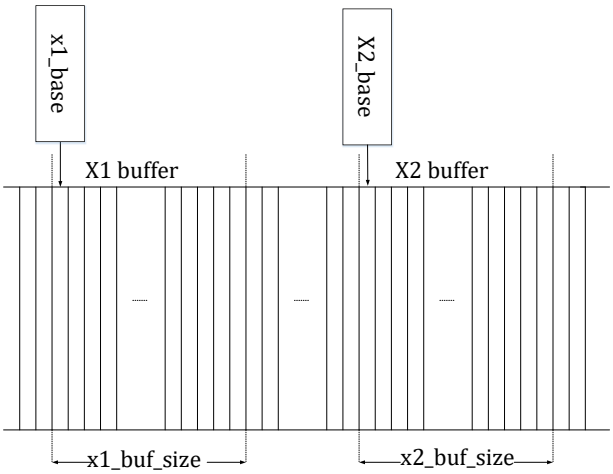


Figure 12-2 Input buffer areas

The X1 buffer can be used as a circular buffer, in which case new data are continually transferred into the input buffer whenever space is available. Pre-loading this buffer is optional for digital filters, since if no input samples have been written in the buffer when the operation is started, it is flagged as empty, which triggers the CPU or DMA to load new samples until there are enough to begin operation. Pre-loading is nevertheless useful in the case of a vector operation, that is, the input data is already available in system memory and circular operation is not required.

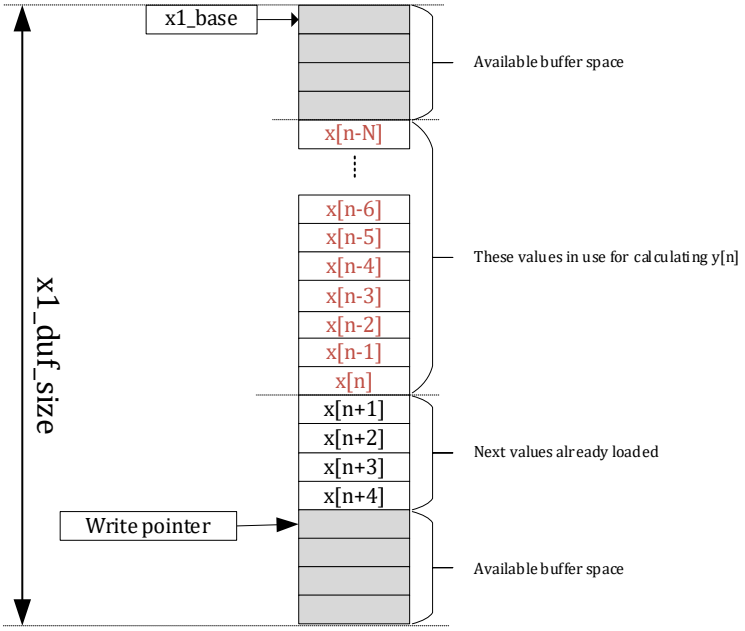


Figure 12-3 Circular input buffer

The X2 buffer can only be used in vector mode (that is not circular), and needs to be pre- loaded, except if the contents of the buffer do not change from one operation to the next. For filter functions, the X2 buffer is used to store the filter coefficients.

When operating as a circular buffer, the space allocated to the buffer (x1_buf_size) should generally be bigger than the number of elements in use for the current calculation, so that there are always new values available in the buffer. Figure 12-3 illustrates the layout of the buffer for a filter operation. While calculating an output sample y[n], the unit uses a set of N+1 input samples, x[n-N] to x[n]. When this is finished, the unit starts the calculation of y[n+1], using the set of input samples x[n-N+1] to x[n+1]. The least-recent input sample, x[n- N], drops out of the input set, and a new sample, x[n+1], is added to it.

The processor, or DMA controller, must ensure that the new sample x[n+1] is available in the buffer space when required. If not, the buffer is flagged as empty, which stalls the execution of the unit until a new sample is added. No underflow condition is signaled on the X1 buffer.

Note: If the flow of samples is controlled by a timer or other peripheral such as an ADC, the buffer regularly goes *empty, since the filter processes each new sample faster than the source can provide it. This is an essential feature of filter operation.*

If the number of free spaces in the buffer is less than the watermark threshold programmed in the FULL_WM bitfield of the FMAC_X1BUFCFG register, the buffer is flagged as full. As long as the full flag is not set, interrupts are generated, if enabled, to request more data for the buffer. The watermark allows several data to be transferred under one interrupt, without danger of overflow. Nevertheless, if an overflow does occur, the OVFL error flag is set and the write data is ignored. The write pointer is not incremented in the event of an overflow.

The operation of the X1 buffer during a filtering operation is illustrated in Figure 12-4. This example shows an 8-tap FIR filter with a watermark set to four.

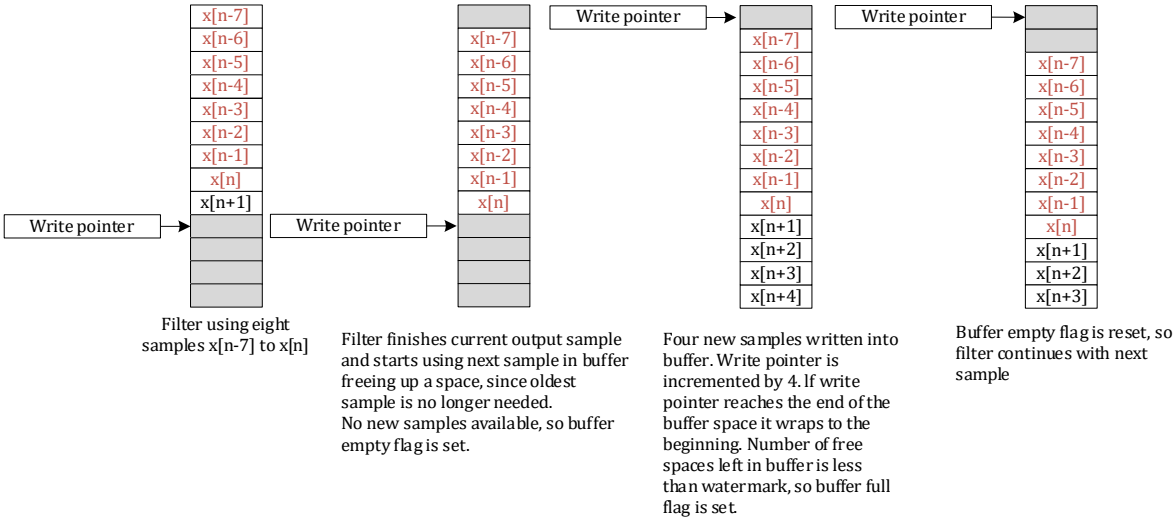


Figure 12-4 Circular input buffer operation

12.3.3 Output buffer

The Y (output) buffer is used to store the output of an accumulation. Each new output value is stored in the buffer until it is read by the processor or DMA controller. Each time a read access is made to the read data register, the read data is fetched from the address indicated by the read pointer. This pointer is incremented after each read, and wraps back to the base address when it reaches the end of the allocated Y buffer space.

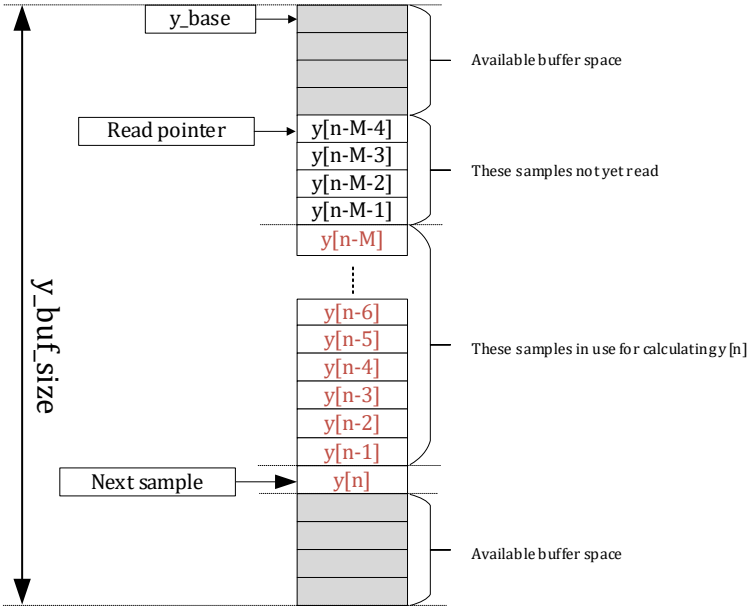


Figure 12-5 Circular output buffer

The Y buffer can also operate as a circular buffer. If the address for the next output value is the same as that indicated by the read pointer (ie. an unread sample), then the buffer is flagged as full and execution stalled until the sample is read.

In the case of IIR filters, the Y buffer is used to store the set of M previous output samples, y[n-M] to y[n-1], used for calculating the next output sample y[n]. Each time a new sample is added to the set, the least recent sample y[n-M] drops out.

If the number of unread data in the buffer is less than the watermark threshold programmed in the EMPTY_WM bitfield of the FMAC_YBUFCFG register, the buffer is flagged as empty. As long as the empty flag is not set, interrupts or DMA requests are generated, if enabled, to request reads from the buffer. The watermark allows several data to be transferred under one interrupt, without danger of underflow. Nevertheless, if an underflow does occur, the UNFL error flag is set. In this case, the read pointer is not incremented and the read operation returns the content of the memory at the read pointer address.

The operation of the Y buffer in circular mode is illustrated in Figure 12-6. This example shows a 7-tap IIR filter with a watermark set to four.

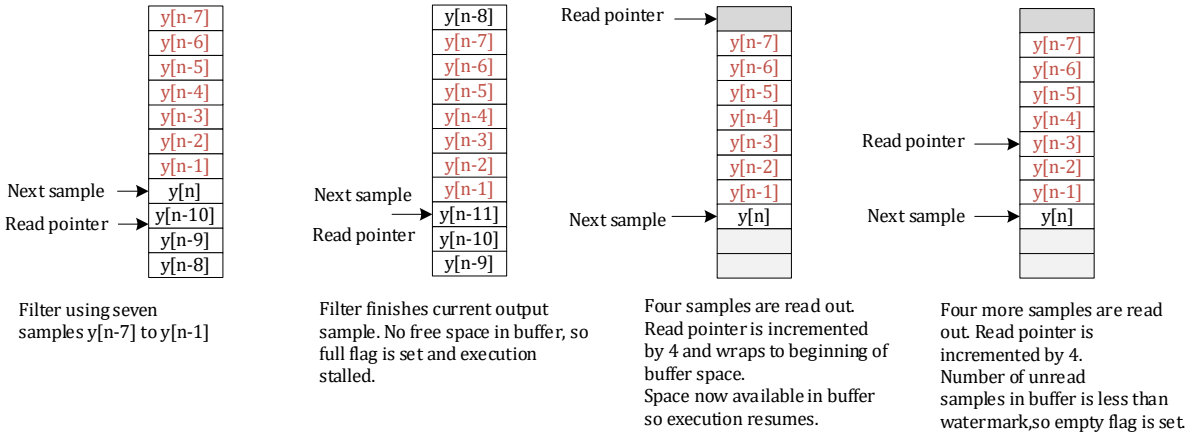


Figure 12-6 Circular output buffer operation

12.3.4 Initialization functions

The following functions initialize the FMAC unit. They are triggered by writing the appropriate value in the FUNC bitfield of the FMAC_PARAM register, with the START bit set. The P and Q bitfields must also contain the appropriate parameter values for each function as detailed below. The R bitfield is not used. When the function completes, the START bit is automatically reset by hardware.

During initialization, it is recommended that the DMA requests and interrupts be disabled. The transfer of data into the FMAC memory can be done by software or by memory-to- memory DMA transfers, since no flow control is required.

12.3.4.1 Load X1 buffer

This function pre-loads the X1 buffer with N+1 values, starting from the address in X1_BASE. Successive writes to the FMAC_WDATA register load the write data into the X1 buffer and increment the write address. The write pointer points to the address X1_BASE+N+1 when the function completes.

The function can be used to pre-load the buffer with the elements of a vector, or to initialize the input storage elements of a filter.

Parameters

- The parameter P contains the number of values, N+1, to be loaded into the X1 buffer.
- The parameters Q and R are not used.

The function completes when N+1 writes have been performed to the FMAC_WDATA register.

12.3.4.2 Load X2 buffer

This function pre-loads the X2 buffer with N+1+M values, starting from the address in X2_BASE. Successive writes to the FMAC_WDATA register load the write data into the X2 buffer and increment the write address.

The function can be used to pre-load the buffer with the elements of a vector, or the coefficients of a filter. In the case of an IIR, the N feed-forward and M feed-back coefficients are concatenated and loaded together into the X2 buffer. The total number of coefficients is equal to N + M. For an FIR, there are no feedback coefficients, so M = 0.

Parameters

- The parameter P contains the number of values, N+1, to be loaded into the X2 buffer starting from address X2_BASE.
- The parameter Q contains the number of values, M, to be loaded into the X2 buffer starting from address X2_BASE+N+1.
- The parameter R is not used.

The function completes whenN+1+M writes have been performed to the FMAC_WDATA register by CPU or DMA.

For FIR, x2 buffer is initialized with coefficient h(0), then h(1), h(2), …, h(N).

For IIR, x2 buffer is initialized with x coefficient b(0), b(1), b(2), …, b(N), and then y coefficient a(1), a(2), …, a(M). Y buffer initial sample number should be less than M.

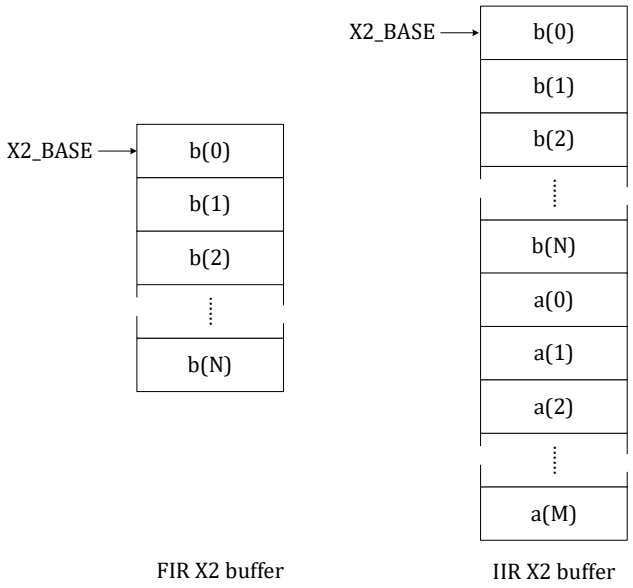


Figure 12-7 FIR and IIR filter x2 buffer coefficient

12.3.4.3 Load Y buffer

This function pre-loads the Y buffer with N+1 values, starting from the address in Y_BASE. Successive writes to the FMAC_WDATA register load the write data into the Y buffer and increment the write address. The read pointer points to the address Y_BASE+N+1 when the function completes.

The function can be used to pre-load the feedback storage elements of an IIR filter.

Parameters

- The parameter P contains the number of values N+1 to be loaded into the Y buffer.

- The parameters Q and R are not used.

The function completes when N writes have been performed to the FMAC_WDATA register.

12.3.5 Filter functions

The following filter functions are supported by the FMAC unit. These functions are triggered by writing the corresponding value in the FUNC bitfield of the FMAC_PARAM register with the START bit set. The P, Q and R bitfields must also contain the appropriate parameter values for each function as detailed below. The filter functions continue to run until the START bit is reset by software.

12.3.5.1 FIR filter (Convolution)

$$\underline{\boldsymbol{Y}} = \underline{\boldsymbol{B}} * \underline{\boldsymbol{X}}$$

$$y_n = 2^R \cdot \sum_{k=0}^{N} b_k x_{n-k}$$

This function performs a convolution of a vector **B** of length N+1 and a vector **X** of indefinite length. The elements of **Y** for incrementing values of n are calculated as the dot product,

This function corresponds to a finite impulse response (FIR) filter, where vector **B** contains the filter coefficients and vector **X** the sampled data, where $X_n = [x_{n-N}, \ldots, x_n]$, including N+1 samples of *X*, index is from *n*-*N* to *n*.

The structure of the filter (direct form) is shown in Figure 12-8.

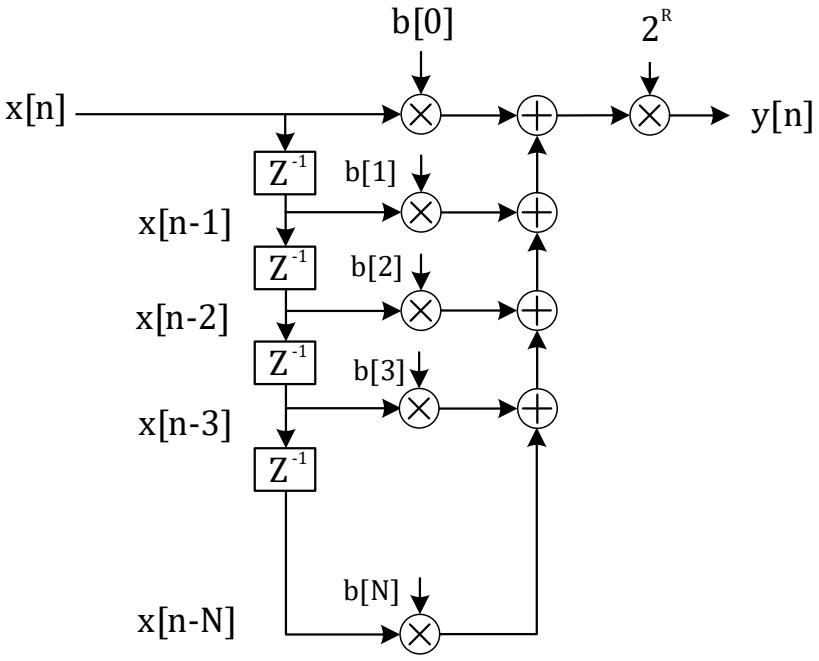


Figure 12-8 FIR filter structure

Note that the cross-correlation vector can be calculated by reversing the order of the coefficient vector **B**.

Input:

- X1 buffer contains the elements of vector X. It is a circular buffer of length N+1+d.
- X2 buffer contains the elements of vector B. It is a fixed buffer of length N+1.

Output:

- Y buffer contains the output values, y_n. It is a circular buffer of length d.

Parameters:

- The parameter P contains the length, N+1, of the coefficient vector B in the range 2~127.
- The parameter R contains the gain to be applied to the accumulator output. The value output to the Y buffer is multiplied by 2R, where R is in the range 0~7.
- The parameter Q is not used.

The function completes when the START bit in the FMAC_PARAM register is reset by software.

12.3.5.2 IIR filter

$$\boldsymbol{Y = B*X+A*Y'}$$

$$y_n = 2^R \cdot \left(\sum_{k=0}^{N} b_k x_{n-k} + \sum_{k=1}^{M} a_k y_{n-k} \right)$$

This function implements an infinite impulse response (IIR) filter. The filter output vector **Y** is the convolution of a coefficient vector **B** of length N+1 and a vector **X** of indefinite length, plus the convolution of the delayed output vector **Y'**with a second coefficient vector **A**, of length M. The elements of **Y** for incrementing values of n are calculated as $y_n = \mathbf{B} \cdot \mathbf{X_n} + \mathbf{A} \cdot \mathbf{Y_{n-1}}$, where $\mathbf{X_n} = [x_{n-N},...,x_n]$ comprises the N+1 elements of **X** at indexes n-N to n, while $\mathbf{Y_{n-1}} = [y_{n-M},...,y_{n-1}]$ comprises the M elements of **Y** at indexes n-M to n-1. The structure of the filter (direct form 1) is shown in Figure 12-9.

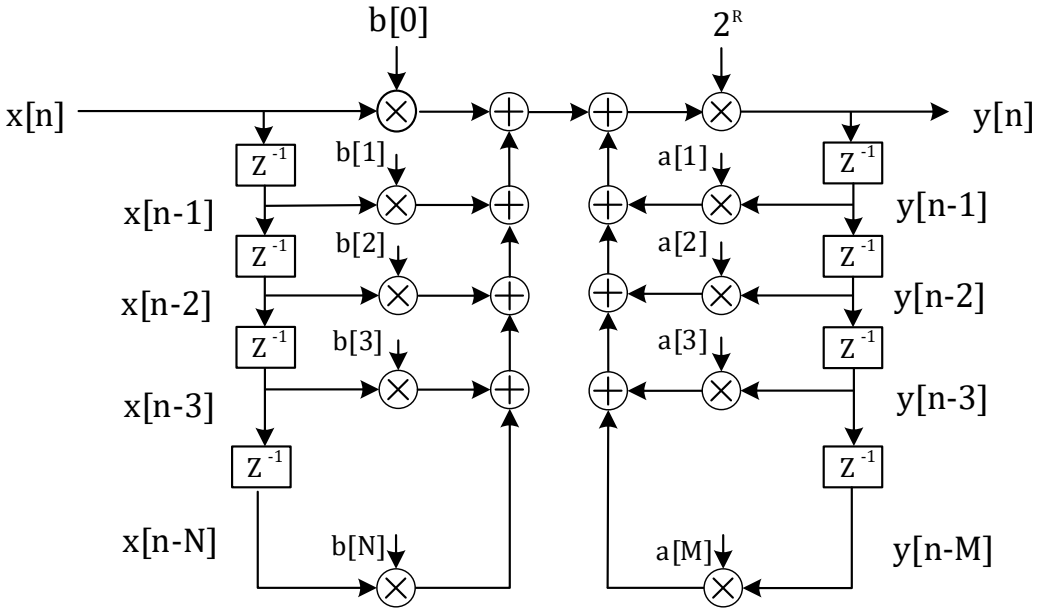


Figure 12-9 IIR filter structure (direct form 1)

Input:

- X1 buffer contains the elements of vector **X**. It is a circular buffer of length N + 1+ d.
- X2 buffer contains the elements of coefficient vectors **B** and **A**concatenated (b_0, b_1, b_2..., b_N, a_1, a_2, ..., a_M). It is a fixed buffer of length M+N+1.

Output:

- Y buffer contains the output values, y_n. It is a circular buffer of length M+d.

Parameters

- The parameter P contains the length, N + 1, of the coefficient vector **B** in therange 2~127.
- The parameter Q contains the length, M, of the coefficient vector **A** in the range1~63.
- The parameter R contains the gain to be applied to the accumulator output. The value output to the Y buffer is multiplied by 2^R, where R is in the range0~7.

The function completes when the START bit in the FMAC_PARAM register is reset by software.

12.3.6 Fixed point representation

The FMAC operates in fixed point signed integer format. Input and output values are Q1.15.

In Q1.15 format, numbers are represented by one sign bit and 15 fractional bits (binary decimal places). The numeric range is therefore -1 (0x8000) to $1-2^{-15}$ (0x7FFF).

The accumulator has 26 bits, of which 22 are fractional and 4 are integer/sign (Q4.22). This allows it to support partial accumulation sums in the range -8 (0x2000000) to +7.99999976 (0x1FFFFFF). A programmable gain from 0dB to 42dB in steps of 6dB can be applied at the output of the accumulator. The corresponding R is 0~7, i.e., Gain is 2^0~2^7. This gain is applied at the accumulator before Y buffer writing.

Note that the content of the accumulator is not saturated if the numeric range is exceeded. Partial sums whose value is greater than +7.99999976 or less than -8, wrap but this is harmless provided subsequent accumulations undo the wrapping. Nevertheless, the SAT flag in the FMAC_SR register is set if wrapping occurs, and generates an interrupt if the SATIEN bit is set in the FMAC_CR register. This helps in debugging the filter.

The data output by the accumulator can optionally be saturated, after application of the programmable gain, by setting the CLIPEN bit in the FMAC_CR register. If this bit is set, then any value which exceeds the numeric range of the Q1.15 output, is set to -1(0x8000) ~$1-2^{-15}$(0x7FFF), according to the sign. If clipping is not enabled, the unused accumulator bits after applying the gain is simply truncated.

12.3.7 Implementing FIR filters with the FMAC

The FMAC supports FIR filters of length N+1, where N+1 is the number of taps or coefficients. The minimum local memory requirement for a FIR filter of length N is 2N+3:

- N+1 coefficients
- N+1 input samples
- 1 output sample

Since the local memory size is 256, the maximum value for N is 127.

If maximum throughput is required, it may be necessary to allocate a small amount of extra space, d1 and d2, to the input and output sample buffers respectively, to ensure that the filter never stalls waiting for a new input sample, or waiting for the output sample to be read. In this case, the local memory requirement is 2N+2+d1+d2.

The buffers should be configured as follows:

- X1_BUF_SIZE = N+1+d1;
- X2_BUF_SIZE = N+1;
- Y_BUF_SIZE = d2 (or 1 if no extra space is required)

The buffer base addresses can be allocated anywhere, but the X2 buffer must not overlap with

the others, or else the coefficients are overwritten. An example configuration could be:

- X2_BASE = 0;
- X1_BASE = N+1;
- Y_BASE = 2N+2+d1

However, if the memory space is limited, the X1 and Y buffer areas can be overlapped, such that each output sample takes the place of the oldest input sample, which is no longer required:

- X2_BASE = 0;
- X1_BASE = N+1;
- Y_BASE = N+1

In this case, Y_BUF_SIZE = X1_BUF_SIZE = N+1+d1, so that the buffers remain in sync.

Note: The FULL_WM bitfield of X1 buffer configuration register must be programmed with a value less than or equal to log2(d1), otherwise the buffer is flagged full before N input samples have been written, and no more samples are requested. Similarly, the EMPTY_WM bitfield of the Y buffer configuration register must be less than or equal to log2(d2).

The filter coefficients **must** be pre-loaded into the X2 buffer, using the Load X2 Buffer function. The X1 buffer can optionally be pre-loaded with any number of samples up to a maximum of N+1. There is no point in pre-loading the Y buffer, since for the FIR filter there is no feedback path.

After configuring and initializing the buffers, the FMAC_CR register should be programmed according to the method used for writing and reading data to and from the FMAC memory.

Three methods are supported:

- Polling: No DMA request or Interrupt request is generated. Software must checkthat the X1_FULL flag is low before writing to WDATA, or that the Y_EMPTY flag is low before reading from RDATA.
- Interrupt: The interrupt request is asserted while the X1_FULL flag is low, for writes, or when the Y_EMPTY flag is low, for reads.
- DMA: DMA requests are asserted on the DMA write channel while the X1_FULL flag is low, and on the read channel while the Y_EMPTY flag is low.

Different methods can be used for read and for write. However, it is not recommended to use both interrupts and DMA requests for the same operation[(a)]. The valid combinations are listed in Table 12-1.

Table 12-1 Valid combinations for read and write methods

WIEN	RIEN	DMAWEN	DMAREN	Write	Read
0	0	0	0	Polling	Polling
0	1	0	0	Polling	Interrupt
1	0	0	0	Interrupt	Polling
1	1	0	0	Interrupt	Interrupt

0	0	0	1	Polling	DMA
0	0	1	0	DMA	Polling
0	0	1	1	DMA	DMA
0	1	1	0	DMA	Interrupt
1	0	0	1	Interrupt	DMA

a. If both interrupts and DMA requests are enabled then only DMA should perform the transfer.

The filter is started by writing to the FMAC_PARAM register with the following bitfield values:

- FUNC = 8 (FIR filter);
- P = N+1 (number of coefficients);
- Q = "Don't care";
- R = Gain;
- START = 1;

If less than N+1+d-2^{FULL_WM} values have been pre-loaded in the X1 buffer, the X1_FULL flag remains low. If the WIEN bit is set in the FMAC_CR register, then the interrupt request is asserted immediately to request the processor to write 2^{FULL_WM} additional samples into the buffer, via the FMAC_WDATA register. It remains asserted until the X1_FULL flag goes high in the FMAC_SR register. The interrupt service routine should check the X1_FULL flag after every 2^{FULL_WM} writes to the FMAC_WDATA register, and repeat the transfer until the flag goes high. Similarly, if the DMAWEN bit is set in the FMAC_CR register, DMA write channel requests are generated until the X1_FULL flag goes high.

The filter calculates the first output sample when at least N+1 samples have been written into the X1 buffer (including any pre-loaded samples).

When 2^{EMPTY_WM} output samples have been written into the Y buffer, the FMAC_SR.Y_EMPTY flag in the FMAC_SR register goes low. If the RIEN bit is set in the FMAC_CR register, the interrupt request is asserted to request the processor to read 2^{EMPTY_WM} samples from the buffer, via the FMAC_RDATA register. It remains asserted until the Y_EMPTY flag goes high. The interrupt service routine should check the Y_EMPTY flag after every 2^{EMPTY_WM} reads from the FMAC_RDATA register, and repeat the transfer until the flag goes high. If the DMAREN bit is set in the FMAC_CR, DMA read channel requests are generated until the Y_EMPTY flag goes high.

The filter continues to operate in this fashion until it is stopped by the software resetting the START bit.

12.3.8 Implementing IIR filters with the FMAC

The FMAC supports IIR filters of length N+1, where N+1 is the number of feed-forward taps or coefficients. The number of feedback coefficients, M, can be any value from 1 to N. Only direct form 1 implementations can be realized, so filters designed for other forms need to be converted.

The minimum memory requirement for an IIR filter with N+1 feed-forward coefficients and M

feed-back coefficients is 2N+2+2M:

- N+1+M coefficients
- N+1 input samples
- M output samples

If M = N, then the maximum filter length that can be implemented is N+1 = 64.

As for the FIR, for maximum throughput a small amount of additional space, d1 and d2, should be allowed in the input and output buffer size respectively, making the total memory requirement 2M +2N+2+d1+d2.

The buffers must be configured as follows:

- X1_BUF_SIZE = N+1+d1;
- X2_BUF_SIZE = N+1+M;
- Y_BUF_SIZE = M+d2;

The buffer base addresses can be allocated anywhere, but must not overlap. An example configuration is given below:

X2_BASE = 0;

X1_BASE = N+1+M;

Y_BASE = 2N+2+M+d1;

Note: The FULL_WM bitfield of X1 buffer configuration register must be programmed with a value less than *or equal to log_2(d1), otherwise the buffer is flagged full before N input samples have been written, and no more samples are requested. Similarly, the EMPTY_WM bitfield of the Y buffer configuration register must be less than or equal to log_2(d2).*

The filter coefficients (N+1 feed-forward followed by M feedback) **must** be pre-loaded into the X2 buffer, using the Load X2 Buffer function. The X1 buffer can optionally be pre-loaded with any number of samples up to a maximum of N+1. The Y buffer can optionally be pre-loaded with any number of values up to a maximum of M. This has the effect of initializing the feedback delay line.

After configuring the buffers, the FMAC_CR register should be programmed in the same way as for the FIR filter.

The filter is started by writing to the FMAC_PARAM register with the following bitfield values:

- FUNC = 9 (IIR filter);
- P = N+1 (number of feed-forward coefficients);
- Q = M (number of feed-back coefficients);
- R = Gain;
- START = 1;

If less than N+1+d-2^{FULL_WM} values have been pre-loaded in the X1 buffer, the X1_FULL flag remains low. If the WIEN bit is set in the FMAC_CR register, then the interrupt request is asserted

immediately to request the processor to write 2^{FULL_WM} additional samples into the buffer, via the FMAC_WDATA register. It remains asserted until the X1_FULL flag goes high in the FMAC_SR register. The interrupt service routine should check the X1_FULL flag after every 2^{FULL_WM} writes to the FMAC_WDATA register, and repeat the transfer until the flag goes high. Similarly, if the DMAWEN bit is set in the FMAC_CR register, DMA write channel requests are generated until the X1_FULL flag goes high.

The filter calculates the first output sample when at least N+1 samples have been written into the X1 buffer (including any pre-loaded samples). The first sample is calculated using the first N+1 samples in the X1 buffer, and the first M samples in the Y buffer (whether or not they are preloaded. The first output sample is written into the Y buffer at Y_BASE + M.

When 2^{EMPTY_WM} new output samples have been written into the Y buffer, the Y_EMPTY flag in the FMAC_SR register goes low. If the RIEN bit is set in the FMAC_CR register, the interrupt request is asserted to request the processor to read 2^{EMPTY_WM} samples from the buffer, via the FMAC_RDATA register. It remains asserted until the Y_EMPTY flag goes high. The interrupt service routine should check the Y_EMPTY flag after every 2^{EMPTY_WM} reads from the FMAC_RDATA register, and repeat the transfer until the flag goes high. If the DMAREN bit is set in the FMAC_CR, DMA read channel requests are generated until the Y_EMPTY flag goes high

The filter continues to operate in this fashion until it is stopped by the software resetting the START bit.

12.3.9 Examples of filter initialization

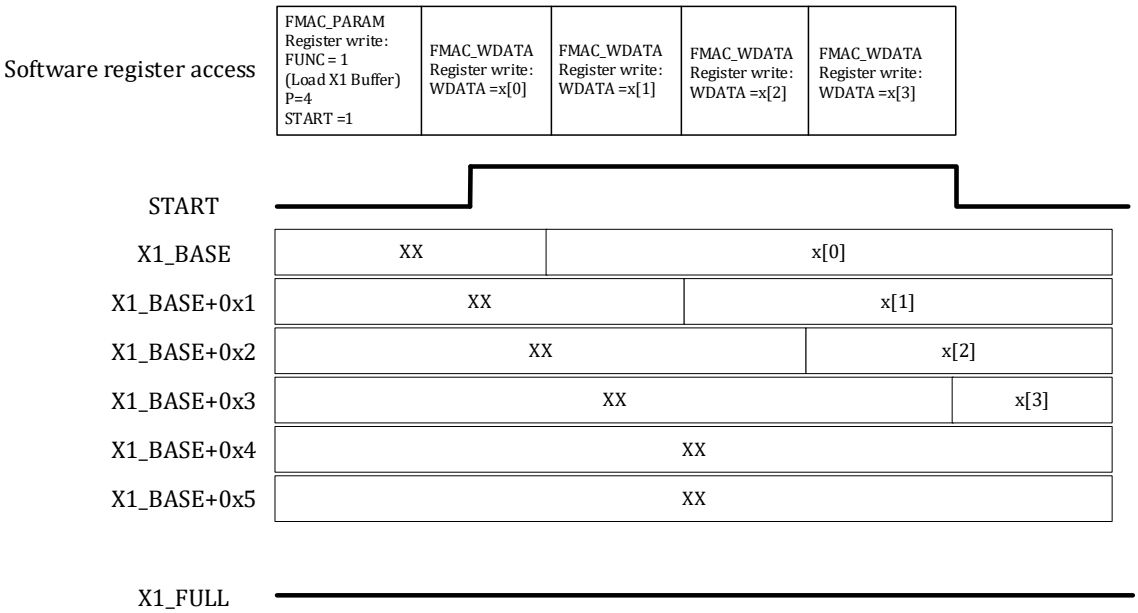


Figure 12-10 X1 buffer initialization

The example in Figure 12-10 X1 buffer initialization illustrates an X1 buffer pre-load with four

samples (P=4). The buffer size is six (X1_BUF_SIZE=6). The initialization is launched by programming the FMAC_PARAM register with the START bit set. The four samples are then written to FMAC_WDATA, and transferred into local memory from X1_BASE onwards. The START bit resets after the fourth sample has been written. At this point, the X1 buffer contains the four samples, in order of writing, and the write pointer (next empty space) is at X1_BASE + 0x4.

12.3.10 Examples of filter operation

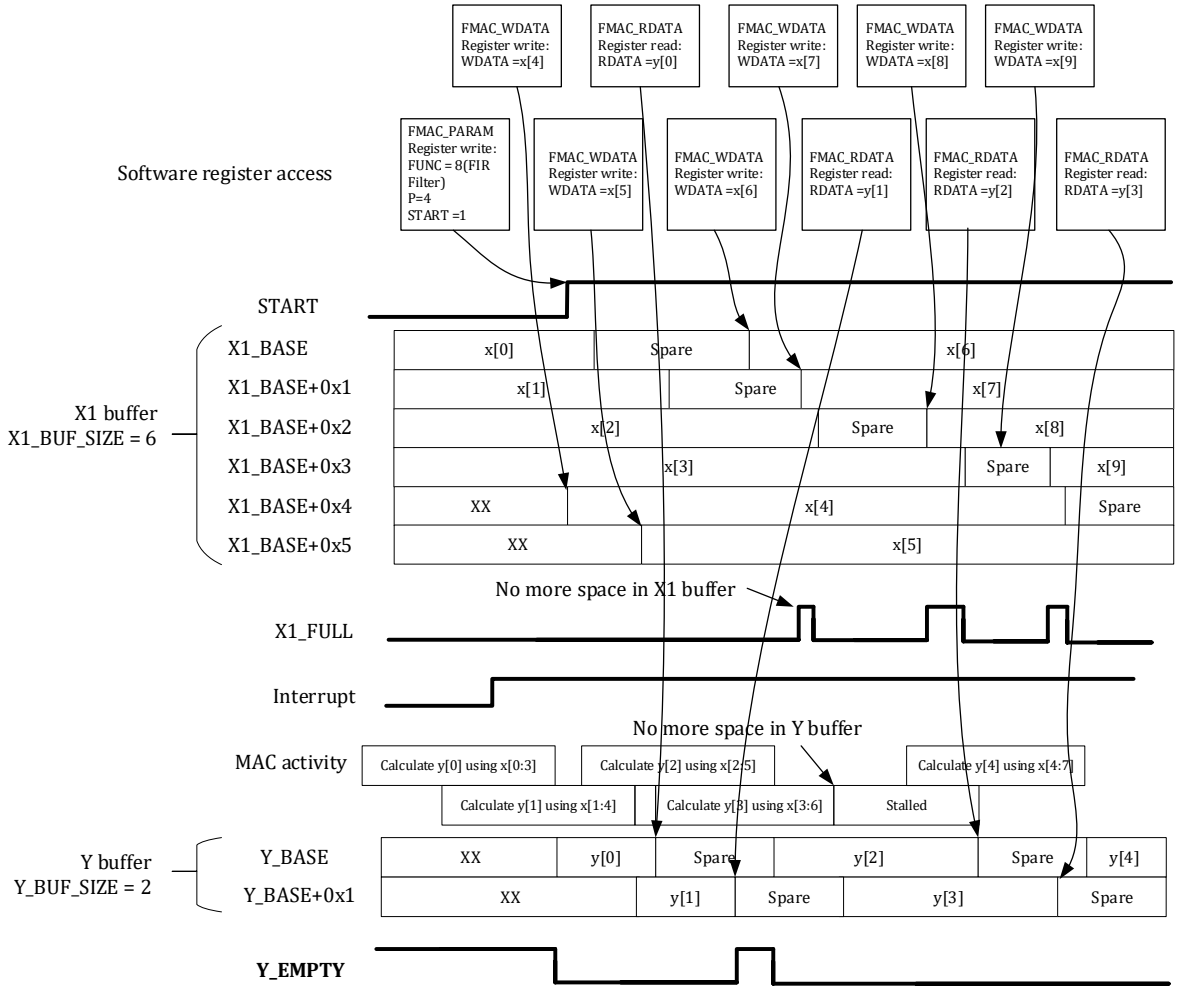


Figure 12-11 Filtering example 1

The example in Figure 12-11 Filtering example 1 illustrates the beginning of a filter operation. The filter has four taps (P=4). The X1 buffer size is six and the Y buffer size is two. The FULL_WM and EMPTY_WM bitfields are both set to 0. Prior to starting the filter, the X1 buffer has been pre-loaded with four samples, x[0:3]. So the filter starts calculating the first output sample, y[0], immediately after the START bit is set. Since the X1FULL flag is not set (due to two uninitialized spaces in the X1 buffer), the interrupt is asserted straight away, to request new data. The processor writes two new samples, x[4] and x[5], to the FMAC_WDATA register, which are transferred to the empty locations in the X1 buffer.

In the mean time, the FMAC finishes calculating the first output sample, y[0], and writes it into the Y buffer, causing the Y_EMPTY flag to go low. At the same time, the x[0] sample is discarded, as it is no longer required, freeing up its location in memory (at X1_BASE). The FMAC can immediately start work on the second output sample, y[1], since all the required input samples x[1:5] are present in the X1 buffer.

Since the Y_EMPTY flag is low, the interrupt remains active after the processor finishes writing

x[5]. The processor reads y[0] from the FMAC_RDATA register, freeing up its location in the Y buffer. There are now no samples in the output buffer since y[1] is still being calculated, so the Y_EMPTY flag goes high. Nevertheless, the interrupt remains active, because there is still free space in the X1 buffer, which the processor next fills with x[6], and so on.

Note: In this example, the processor can fill the input buffer more quickly than the FMAC can process them, so the X1_full flag regularly goes active. However, it struggles to read the Y buffer fast enough, so the FMAC stalls regularly waiting for space to be freed up in the Y buffer. This means the filter is not executing at maximum throughput. The reason is that the

filter length is small and the processor relatively slow, in this example. So, increasing the Y buffer size would not help.

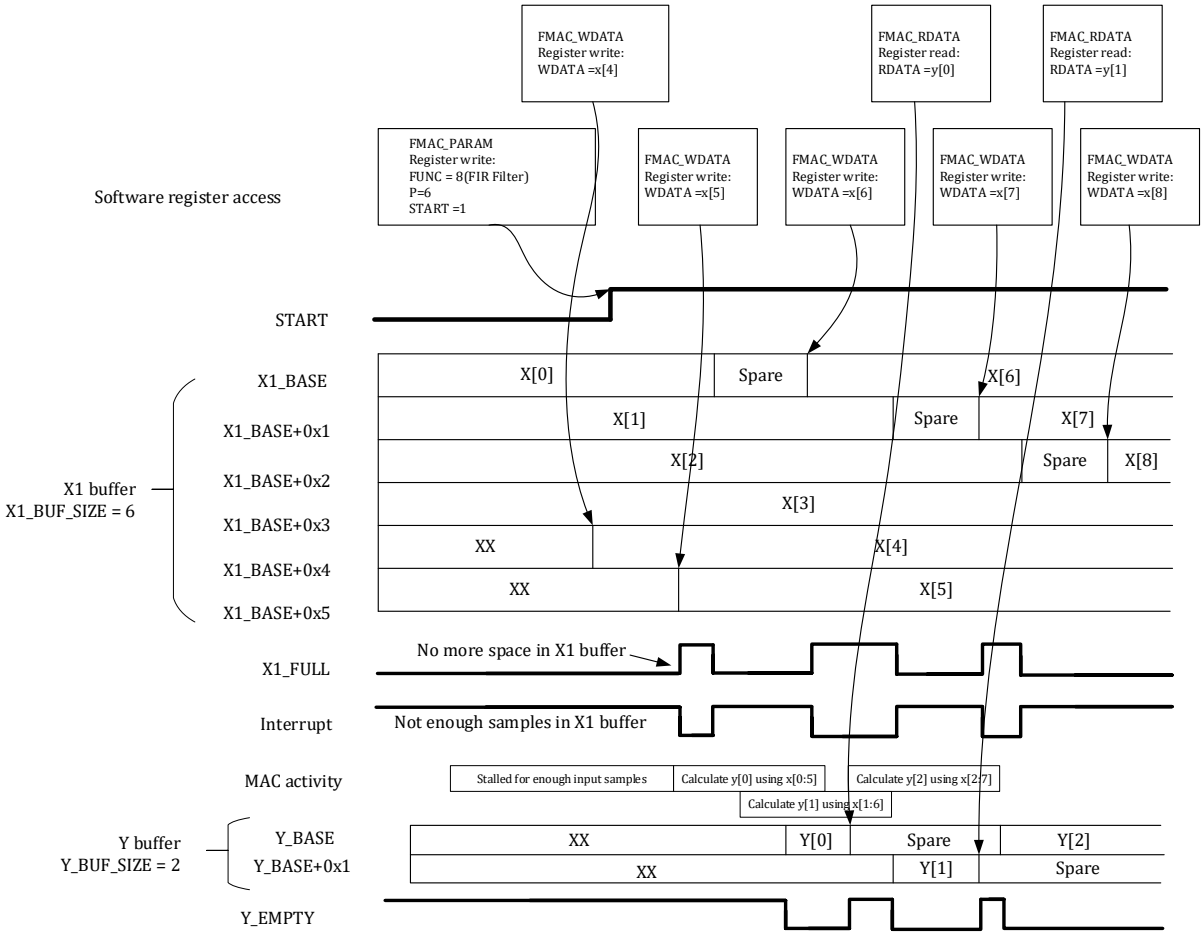


Figure 12-12 Filtering example 2

The example in Figure 12-12 Filtering example 2 illustrates the beginning of the same filter operation, but this time the filter has six taps (P=6). The X1 buffer size is six and the Y buffer size is two. The FULL_WM and EMPTY_WM bitfields are both set to 0. Prior to starting the filter, the X1 buffer has been pre-loaded with four samples, x[0:3]. Because there are not enough samples in the input buffer, the X1FULL flag is not set, so the interrupt is asserted straight away, to request new data. The FMAC is stalled.

The processor writes two new samples, x[4] and x[5], to the FMAC_WDATA register, which are transferred to the empty locations in the X1 buffer. As soon as there are six unused samples in the X1 buffer, the X1_FULL flag goes active (since the buffer size is six), causing the interrupt to go inactive. The FMAC starts calculating the first output sample, y[0]. Since this requires all six input samples, there are no free spaces in the X1 buffer and so the X1_FULL flag remains active. Only when the FMAC finishes calculating y[0] and writes it into the Y buffer, can x[0] be discarded, freeing up a space in the X1 buffer, and deasserting X1_FULL. At the same time, the Y_EMPTY flag goes inactive. Both these flag states cause the interrupt to be asserted, requesting the processor to write a new input sample, first of all, and then read the output sample just calculated. The FMAC remains stalled until a new input sample is written.

In this example, the processor has to wait for the FMAC to finish calculating the current output sample, before it can write a new input sample, and therefore the X1 buffer regularly goes empty, stalling the FMAC. This can be avoided by allowing some extra space in the input buffer.

12.3.11 Filter design tips

The FMAC architecture imposes some constraints detailed below, on the design of digital filters.

1. Implementation of direct form 2, or transposed forms, is not efficient. Filters which have been designed for such forms should be converted to direct form 1.

2. Cascaded filters must either be combined into a single stage, or implemented as separate filters. In the latter case, multiple sets of filter coefficients can be pre-loaded into the memory, one set per stage, and only the X2_BASE address changed to select which set is used. The most efficient method of implementing a multi-stage filter is to pre-load a large X1 buffer with input samples, run the IIR filter function on it using the first stage coefficients, and store the output samples back in memory. Then change the X2_BASE pointer to point to the 2nd stage coefficients, and reload the input buffer with the output of the first stage (with a gain if required), before running the IIR function again. The procedure is repeated for all stages. Once the final stage samples have been transferred back into system memory, the input buffer can be loaded with the next set of input samples, and a new round of calculations started. Note that the N+1 sample input buffer of each stage must be pre-loaded first of all with the N last inputs from the previous round, plus one new sample, in order to keep continuity between each round. Similarly, the output buffer of each stage must be loaded with the last M samples from the previous round, for the same reason.

3. The use of direct form 1 for IIR designs can lead to large positive or negative partial sums in the accumulator, if for example a large step occurs on the input, or some of the filter coefficients'absolute values are >1. Since the accumulator is limited to 26 bits, the biggest value that it can handle without wrapping (changing sign) is 0x1FFFFFF positive or 0x2000000 negative. This corresponds to 3.99999988 and -4 respectively in Q3.23 fixed point format. Wrapping does not represent a problem provided the wrapping is "undone" before the end of the accumulation. However, this is not always the case when a filter is starting up and can lead to unexpected results. Consider pre-loading the output buffer with suitable values to avoid this.

4. The IIR filter has feed-forward (numerator) coefficients [b_0, b_1, ..., b_N], and feed-back

(denominator) coefficients [1, a_1, ..., a_M]. Many IIR filters require some of the denominator coefficients to have an absolute value greater than 1 to achieve a steep roll-off in the frequency response. Given that the coefficients are coded in fixed point Q1.15 format, this is not possible. Nevertheless, by scaling the denominator coefficients by a factor 2^{-R}, such that 2^{-R}[1, a_1, ..., a_M] are all less than 1, such filters can be implemented. However, an inverse gain of 2^R must be applied at the output of the accumulator to compensate the scaling. This has an adverse effect on the signal-to- noise ratio.

12.4 Register

12.4.1 Address allocation

FMAC base address is 0x4001_7000

Table 12-1 FMAC register table

Name	Offset	Description
FMAC_X1BUFCFG	0x00	FMAC X1 buffer configuration register
FMAC_X2BUFCFG	0x04	FMAC X2 buffer configuration register
FMAC_YBUFCFG	0x08	FMAC Y buffer configuration register
FMAC_PARAM	0x0C	FMAC parameter register
FMAC_CR	0x10	FMAC control register
FMAC_SR	0x14	FMAC status register
FMAC_WDATA	0x18	FMAC write data register
FMAC_RDATA	0x1C	FMAC read data register

12.4.2 Register description

12.4.2.1FMAC_X1BUFCFG

Address：0x4001_7000

Reset value：0x0

Table 12-2 FMAC_X1BUFCFG FMAC X1 buffer configuration register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
						FULL_WM									
						RW									
						0									

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X1_BUF_SIZE								X1_BASE							
RW								RW							
0								0							

Bit field	Bit name	Description
[31:26]		NA
[25:24]	FULL_WM	Watermark for buffer full flag Defines the threshold for setting the X1 buffer full flag when operating in circular mode. The flag is set if the number of free spaces in the buffer is less than 2^{FULL_WM}. 0: Threshold = 1 1: Threshold = 2 2: Threshold = 4 3: Threshold = 8 Setting a threshold greater than 1 allows several data to be transferred into the buffer under one interrupt. Threshold should be set to 1 if DMA write requests are enabled (DMAWEN = 1 in FMAC_CR register).
[23:16]		NA
[15:8]	X1_BUF_SIZE	Allocated size of X1 buffer in 16-bit words The minimum buffer size is the number of feed-forward taps in the filter (+ the watermark threshold - 1).
[7:0]	X1_BASE	Base address of X1 buffer

12.4.2.2FMAC_X2BUFCFG

Address：0x4001_7004

Reset value：0x0

Table 12-3 FMAC_X2BUFCFG FMAC X2 buffer configuration register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X2_BUF_SIZE								X2_BASE							
RW								RW							
0								0							

Bit field	Bit name	Description
[31:16]		NA
[15:8]	X2_BUF_SIZE	Size of X2 buffer in 16-bit words This bitfield can not be modified when a function is ongoing (START = 1).

[7:0]	X2_BASE	Base address of X2 buffer This bitfield can not be modified when a function is ongoing (START = 1).

12.4.2.3FMAC_YBUFCFG

Address：0x4001_7008

Reset value：0x0

Table 12-4 FMAC_YBUFCFG FMAC Y buffer configuration register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
						EMPTY_WM									
						RW									
						0									

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

Bit field	Bit name	Description
[31:26]	未使用	NA
[25:24]	EMPTY_WM	Y buffer watermark for buffer empty flag Defines the threshold for setting the Y buffer empty flag when operating in circular mode. The flag is set if the number of unread values in the buffer is less than 2^{EMPTY_WM}. 0: Threshold = 1 1: Threshold = 2 2: Threshold = 4 3: Threshold = 8 Setting a threshold greater than 1 allows several data to be transferred from the buffer under one interrupt. Threshold should be set to 1 if DMA read requests are enabled (DMAREN = 1 in FMAC_CR register).
[23:16]		NA
[15:8]	Y_BUF_SIZE	Size of Y buffer in 16-bit words For FIR filters, the minimum buffer size is 1 (+ the watermark threshold). For IIR filters the minimum buffer size is the number of feedback taps (+ the watermark threshold).
[7:0]	Y_BASE	Base address of Y buffer

12.4.2.4FMAC_PARAM

Address：0x4001_700C

Reset value：0x0

Table 12-5 FMAC_PARAM FMAC parameter register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
START				FUNC				R							
RW				RW				RW							
0				0				0							

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

Bit field	Bit name	Description
[31]	START	Filter execution enable 0: Stop execution 1: Start execution Setting this bit triggers the execution of the function selected in the FUNC bitfield. Resetting it by software stops any ongoing function. For initialization functions, this bit is reset by hardware.
[30:28]		NA
[27:24]	FUNC	0: Reserved 1: Load X1 buffer 2: Load X2 buffer 3: Load Y buffer 4 to 7: Reserved 8: Convolution (FIR filter) 9: IIR filter (direct form 1) This bitfield can not be modified when a function is ongoing (START = 1)
[23:16]	R	Input parameter R. Gain factor of Y before Y buffer writing, i.e., Y is multiplied by 2^R then written in to Y buffer This bitfield can not be modified when a function is ongoing (START = 1)
[15:8]	Q	Input parameter Q. IIR feedback coefficient number. This bitfield can not be modified when a function is ongoing (START = 1)
[7:0]	P	Input parameter P.

		FIR/IIR feedforward coefficient number. This bitfield can not be modified when a function is ongoing (START = 1)

12.4.2.5FMAC_CR

Address：0x4001_7010

Reset value：0x0

Table 12-6 FMAC_CR FMAC control register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
															RESET
															WO
															0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLIPEN						DMAWEN	DMAREN				SATIEN	UNFLIEN	OVFLIEN	WIEN	RIEN
RW						RW	RW				RW	RW	RW	RW	RW
0						0	0				0	0	0	0	0

Bit field	Bit name	Description
[31:17]		NA
[16]	RESET	Reset FMAC unit This resets the write and read pointers, the internal control logic, the FMAC_SR register and the FMAC_PARAM register, including the START bit if active. Other register settings are not affected. This bit is clear by hardware automatically after writing. 0: Reset inactive 1: Reset active
[15]	CLIPEN	Enable clipping 0: Clipping disabled. Values at the output of the accumulator which exceed the Q1.15 range, wrap. 1: Clipping enabled. Values at the output of the accumulator which exceed the Q1.15 range are saturated to the maximum positive or negative value (+1 or -1) according to the sign.
[14:10]		NA
[9]	DMAWEN	DMA write channel request enable 0: Disable. No DMA requests are generated 1: Enable. DMA requests are generated while the X1 buffer is not full. This bit can only be modified when START= 0 in the FMAC_PARAM

		register. A read returns the current state of the bit.
[8]	DMAREN	DMA read channel request enable 0: Disable. No DMA requests are generated 1: Enable. DMA requests are generated while the Y buffer is not empty. This bit can only be modified when START= 0 in the FMAC_PARAM register. A read returns the current state of the bit.
[7:5]		NA
[4]	SATIEN	Saturation error interrupt enable 0: Disabled. No interrupts are generated upon saturation detection. 1: Enabled. An interrupt request is generated if the SAT flag is set This bit is set and cleared by software. A read returns the current state of the bit.
[3]	UNFLIEN	Y buffer underflow error interrupt enable 0: Disabled. No interrupts are generated upon underflow detection. 1: Enabled. An interrupt request is generated if the UNFL flag is set This bit is set and cleared by software. A read returns the current state of the bit.
[2]	OVFLIEN	X1 buffer overflow error interrupt enable 0: Disabled. No interrupts are generated upon overflow detection. 1: Enabled. An interrupt request is generated if the OVFL flag is set This bit is set and cleared by software. A read returns the current state of the bit.
[1]	WIEN	X1 buffer write interrupt enable 0: Disabled. No write interrupt requests are generated. 1: Enabled. An interrupt request is generated while the X1 buffer FULL flag is not set. This bit is set and cleared by software. A read returns the current state of the bit.
[0]	RIEN	Y buffer read interrupt enable 0: Disabled. No read interrupt requests are generated. 1: Enabled. An interrupt request is generated while the Y buffer EMPTY flag is not set. This bit is set and cleared by software. A read returns the current state of the bit.

12.4.2.6FMAC_SR

Address：0x4001_7014

Reset value：0x1

Table 12-7 FMAC_SR FMAC status register

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					SAT	UNFL	OVFL							X1_FULL	Y_EMPTY
					RO	RO	RO							RO	RO
					0	0	0							0	1

Bit field	Bit name	Description
[31:11]		NA
[10]	SAT	Saturation error flag Saturation occurs when the result of an accumulation exceeds the numeric range of the accumulator. 0: No saturation detected 1: Saturation detected. If the SATIEN bit is set, an interrupt is generated. This flag is cleared by a reset of the unit.
[9]	UNFL	Y buffer underflow error flag An underflow occurs when a read is made from FMAC_RDATA when no valid data is available in the Y buffer. 0: No underflow detected 1: Underflow detected. If the UNFLIEN bit is set, an interrupt is generated. This flag is cleared by a reset of the unit.
[8]	OVFL	X1 buffer overflow error flag An overflow occurs when a write is made to FMAC_WDATA when no free space is available in the X1 buffer. 0: No overflow detected 1: Overflow detected. If the OVFLIEN bit is set, an interrupt is generated. This flag is cleared by a reset of the unit.
[7:2]		NA
[1]	X1_FULL	X1 buffer full flag The buffer is flagged as full if the number of available spaces is less than the FULL_WM threshold. The number of available spaces is the difference between the write pointer and the least recent sample currently in use. 0: X1 buffer not full. If the WIEN bit is set, the interrupt request is asserted until the flag is set. If DMAWEN is set, DMA write channel requests are generated until the flag is set. 1: X1 buffer full. This flag is set and cleared by hardware, or by a reset. *Note: after the last available space in the X1 buffer is filled there is a delay of 3 clock cycles before the X1FULL flag goes high. To avoid any risk of overflow it is recommended to insert a software delay after writing to the X1 buffer before reading the FMAC_SR. Alternatively, a FULL_WM threshold of 2 can be used*

[0]	Y_EMPTY	Y buffer empty flag The buffer is flagged as empty if the number of unread data is less than the EMPTY_WM threshold. The number of unread data is the difference between the read pointer and the current output destination address. 0: Y buffer not empty. If the RIEN bit is set, the interrupt request is asserted until the flag is set. If DMAREN is set, DMA read channel requests are generated until the flag is set. 1: Y buffer empty. This flag is set and cleared by hardware, or by a reset. *Note: after the last sample is read from the Y buffer there is a delay of 3 clock cycles before the Y_EMPTY flag goes high. To avoid any risk of underflow it is recommended to insert a software delay after reading from the Y buffer before reading the FMAC_SR. Alternatively, an EMPTY_WM threshold of 2 can be used.*

12.4.2.7FMAC_WDATA

Address：0x4001_7018

Reset value：0x0

Table 12-8 FMAC_WDATA FMAC write data register

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

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

Bit field	Bit name	Description
[31:16]		NA
[15:0]	WDATA	When a write access to this register occurs, the write data are transferred to the address offset indicated by the write pointer. The pointer address is automatically incremented after each write access.

12.4.2.8FMAC_RDATA

Address：0x4001_701C

Reset value：0x0

Table 12-9 FMAC_RDATA FMAC read data regiser

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

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

Bit field	Bit name	Description
[31:16]	未使用	NA
[15:0]	RDATA	When a read access to this register occurs, the read data are the contents of the Y output buffer at the address offset indicated by the READ pointer. The pointer address is automatically incremented after each read access.

13 ADC

13.1 Overview

The LKS32MC45x series chip integrates three 14-bit SARADCs, and 27 analog IO input signals/6 OPA outputs/temperature sensor output/analog ground can be used as ADC sampled signals. The main features are as follows:

- 3 ADC digital interfaces corresponding to three ADCs
- Support 2MSPS sampling rate and 32MHz working frequency
- Each ADC interface supports 16 channels
- Each ADC interface is equipped with 2 analog watchdogs to detect the upper and lower thresholds simultaneously
- Support oversampling
- Support software and hardware trigger
- Cooperate with MCPWM and Timer unit to trigger ADC sample. Indication signal can be transmitted through GPIO for debugging
- Support sampling with custom sampling sequences, and the sequence number and channel number can be set flexibly
- Support left and right alignment mode
- Support differential input of analog signal, and positive and negative input channels can be configured
- Support single-round sampling and double-round sampling
- Support hardware oversampling, with effective bits of over 18bit after oversampling

The terminology about ADC conventions are:

Single-time sampling: complete the sampling conversion of the corresponding analog signal quantity to the data signal quantity of one channel, and store the digital quantity to the ADC_DATx register;

Single-round sampling: may contain one or several samples. The analog signals for several samples can be the same or different. Sampling is usually triggered by MCPWM, UTimer, or software. One trigger signal could start one round sampling. After sampling round is completed, a corresponding round sampling completion interrupt is generated.

13.1.1 Functional block diagram

Each ADC interface includes 16 data registers (ADC samples the digital quantity corresponding to the analog quantity of each channel for 16 times) and several control registers.

It can sample three channels at the same time.

The data register ADC_DATx is used to store the digital value converted by the analog-to-digital converter (ADC) at the xth sampling, and the source of the analog signal is selected by a 4-bit setting within the register ADC_CHNx (see 13.2.3 for details). Taking ADC_PCHN0 as an example. Bit [3:0] selects the analog signal of 0th sampling, and the channel number can be selected from ADC_CH0 to ADC_CH14. If ADC_PCHN0 [3:0] = 0 and ADC_PCHN0 [7:4] = 3, the positive analog signal of the 0th sampling corresponds to IO ADC_CH0, the positive analog signal of the 1st sampling corresponds to IO ADC_CH3, and so on. Negative signal is designated by ADC_NCHN. If differential input is not required, the negative end can be specified as analog ground.

The sampling channel register ADC_CHNT controls the number of samples per round, and "1-15" means "1-15" samples, which cannot be set to "0". The sum of two sampling times should not exceed 16.

The control logic selects the trigger signal from MCPWM or the UTimer according to trigger configuration register ADC_TRIG to start sampling or initiates the sampling through the software trigger. MCPWM/UTimer will send timed trigger signal.

After a round conversion is completed (sampling and conversion of all channels within a round is completed), the ADC conversion done flag will be set.

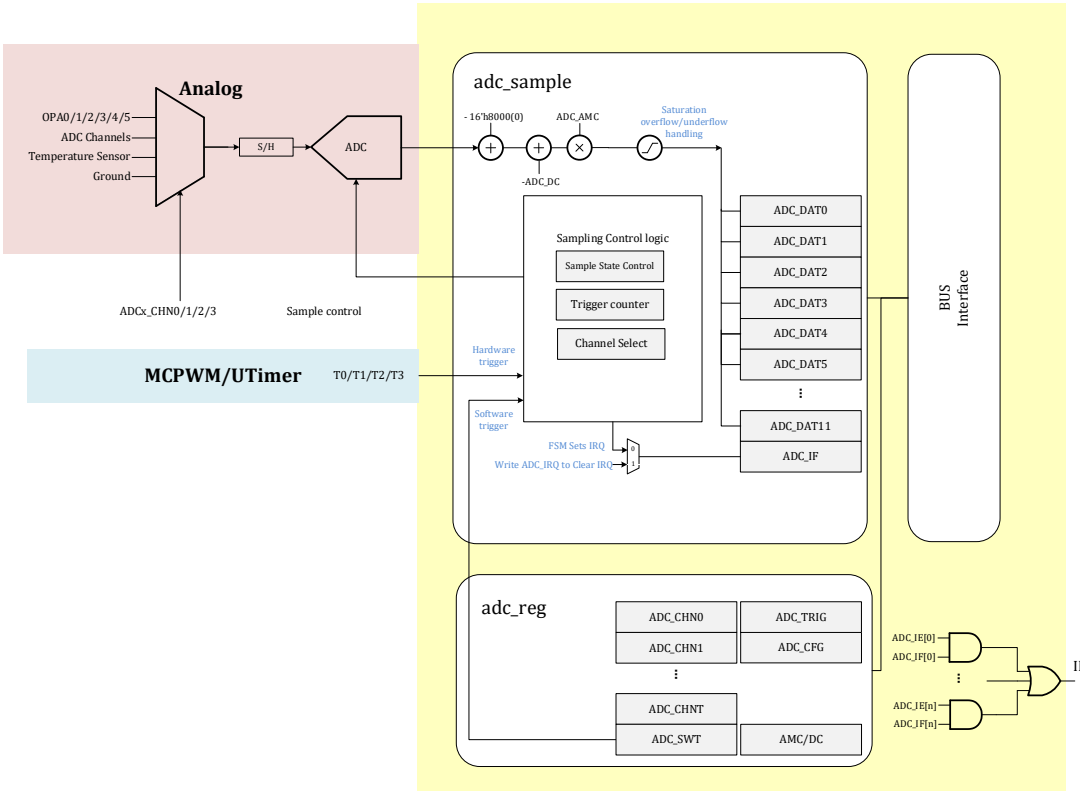


Figure 13-1 Functional Block Diagram of ADC Sampling

The user can set the sampling sequence and the source of the sampled signal flexibly, and even

sample one exact signal for several times. The control register allows the user to set the number of samples, improve the sampling frequency or reduce the sampling power consumption.

Both positive and negative signals can be configured, so users can flexibly select differential signal sources for each sampling and use IO resources.

Oversampling allows users to average signals through multiple samplings to get higher SNR in scenarios where sampling rate is not high but accuracy is required.

13.1.2 ADC Trigger Mode

- Support single-round trigger and double-round trigger
- Set the number of trigger events to realize interval trigger
- Support trigger by software
- Set to generate an interrupt after each trigger is completed
- Trigger indication signal can be transmitted through GPIO for debugging

13.1.3 ADC Output Digital System

The ADC output data is 14-bit complement. The input signal 0 corresponds to 14'b00_0000_0000_0000. Taking the gain configuration as an example, the input signal -2.2V corresponds to 14'b10_0000_0000 _0000, and the input signal +2.2V corresponds to 14'b01_1111_1111 _1111. The 14-bit complementary code after ADC conversion should be expanded to 16-bit and stored in the sampling data register of 16-bit width. Left or right alignment can be set by the configuration register. Taking 14'b10_0000_0000_1101 as an example, if the configuration is left-aligned, the right side is filled with four "0", and the value stored in ADCx_DAT is 16'b1000_0000_0011_0100; if the configuration is right-aligned, sign expansion is performed on the left, and the value stored in ADCx_DAT is 16'b1110_0000_0000_1101.

Table 13-1 Conversion of ADC Output Digital System

ADC Double Gain Input Analog Value/V	ADC 2/3 Times Gain Input/V	Converted Signed Number
2.2	3.3	14'b01_1111_1111_1111
0	0	14'b00_0000_0000_0000
-2.2	-3.3	14'b10_0000_0000_0000

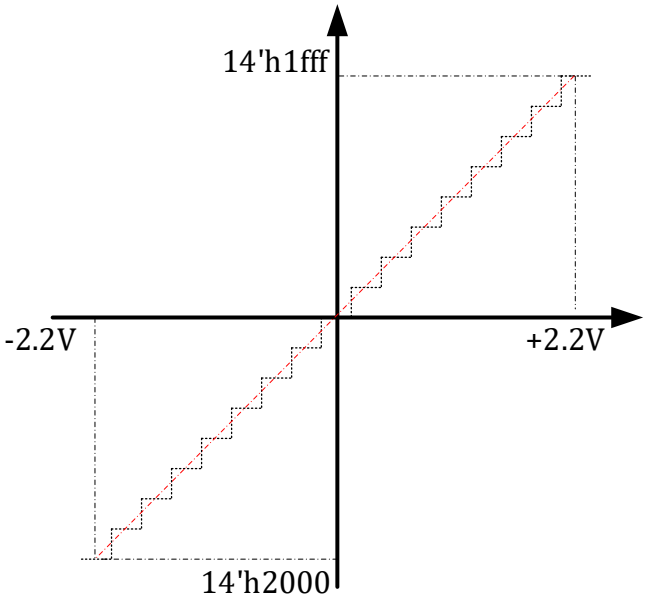


Figure 13-2 ADC Digital Range at Double Gain Setting

13.1.4 ADC Range

The ADC has two gain modes: high gain (1 times) and low gain (2/3 times). The ADC ranges of these two gains are also different. In 1x gain mode, the maximum input signal amplitude is ±2.2V; in 2/3 gain mode, the maximum input signal amplitude is ±3.3V.

When the ADC sampling channel is set as the output signal of the OPA (i.e. OPA0-OPA5), the appropriate OPA gain should be selected. This will allow the maximum signal in a specific application to be amplified to a level close to +/-3.3V, while setting the ADC to a gain of 2/3. For example, the maximum phase current is 100A (effective value of sine wave), and the MOS internal resistance (assuming as MOS internal resistance sampling) is 5mR, then the maximum input signal amplitude of the OPA is +/-707mV. Then, the OPA gain should be selected to be 4.5 times, and the amplified signal is about +/-3.18V.

If the output signal of the OPA is amplified and the maximum signal is still less than +/-2.2V due to objective reasons, the gain of the ADC should be set to 1 times.

When the ADC sampling channel is set as the input signal of the GPIO multiplex port, the ADC gain is also selected according to the maximum amplitude of the signal. Due to the limitation of the IO port, the signal range of the GPIO multiplex port can only be between -0.3V- AVDD+0.3V.

13.1.5 ADC Calibration

The ADC hardware interface module can perform DC offset calibration and gain calibration.

The AMPcorrection, which is a gain calibration factor, stored by ADC_AMC, is a 10-bit unsigned fixed-point number. ADC_AMC [9] is the integer, and ADC_AMC [8:0] is the decimal, which can represent a fixed-point number whose value is around "1".

ADC_DC stores the DC offset of ADC, which is usually obtained by measuring the AVSS (internal ground) of channel 15 and stored in flash at the calibration stage, and written to ADC_DC register by software at the system loading stage.

Record that the digital quantity output by the ADC is D_{ADC}, the true value corresponding to D_{ADC} is D, and D_0 is 0 in the coding system, then

$$D = (D_{ADC}\text{-}D_0\text{-}DC_{offset})*AMP_{correction}$$

Finally, the hardware will store the corrected D into the corresponding sampling data register. The ADC interface hardware circuit will automatically select $AMP_{correction}$ and DC_{offset} according to the gain configuration of each channel (ADC_GAIN0/1).

13.1.6 ADC Threshold Monitoring (analog watchdog)

Each ADC interface module is equipped with 2 sets of threshold monitoring circuit to monitor the upper and lower thresholds at the same time. ADC data register can be individually configured to enable or disable a set of threshold monitoring, and one data register can enable multiple sets of thresholds for monitoring simultaneously, but this is not usually configured. If the threshold monitoring is enabled, when the ADC completes a conversion and writes the corrected data to the ADCx_DAT register, a threshold comparison is performed. If the written data is beyond the threshold range, a threshold overlimit interrupt flag will be set to 1.

The threshold is a 14-bit signed number, so the threshold comparison is independent of the left and right alignment of the data. In left alignment, the ADCx_DATx[15:2] is compared with the threshold; and in right alignment, ADCx_DATx[13:0] is compared with the threshold.

13.1.7 Oversampling

ADC supports oversampling and averaging the signals before writing to the data register. The oversampling range is 1-128 times, which must be a power of 2 and is configured by ADCx_CFG.OVSR. By default, the signal is written to the data register after one sampling, that is, no oversampling is required. If oversampling times is configured, all signals specified by ADCx_CHNx shall be oversampled and averaged before written to the data register. The relationship between the value written to the data register and the value of the ADC multiple conversion is shown in the following formula.

$$ADCx\_DAT = \frac{1}{OVSR}\sum_{i=0}^{OVSR-1} ADCx\_DAT\_RAW_i$$

Oversampling can be used together with threshold monitoring. At this time, the threshold overlimit event is generated only when the average value exceeds the threshold range.

After configuring oversampling mode, the conversion time of each round of sampling will be increased by multiple and an interrupt flag will be generated after sampling conversion.

ADCx_CFG.TROVS can be set to one trigger to complete multiple ((ADCx_CFG.OVSR) samplings and averaging. As shown in Figure 13-3, when ADCx_CFG.TROVS=0 and ADCx_CFG.OVSR=1, that is the oversampling rate is 2, after triggering, ADC will sample each signal twice and average the value

before written to the corresponding register ADCx_DAT; if ADCx_CFG.TROVS=1, ADC can sample enough data for averaging after multiple (ADCx_CFG.OVSR) triggers. As shown in Figure 13-4, when ADCx_CFG.OVSR=1, that is the sampling rate is still 2, ADC can accumulate 2 sets of data for averaging after 2 triggers and write to the data register. If multiple triggers are required to accumulate data, each trigger can only sample one channel, that is ADCx_CHNT=1. If multiple signals shall be sampled, when switching channels, the data of different channels will be accumulated, resulting in wrong conversion data.

Oversampling can be used together with continuous sampling. In this case, ADC can continuously sample signals of a channel for oversampling times and save the averaged data to the data register.

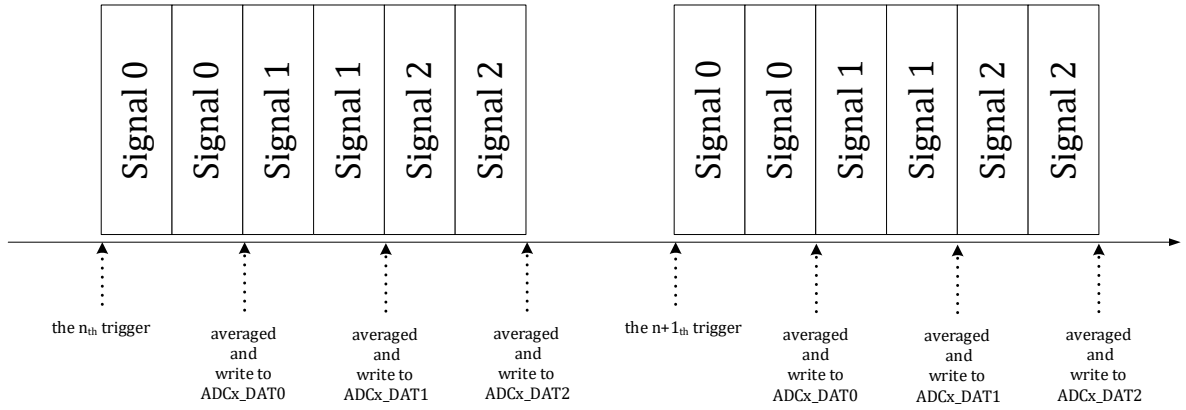


Figure 13-3 Oversampling Conversion Sequence Sample 1

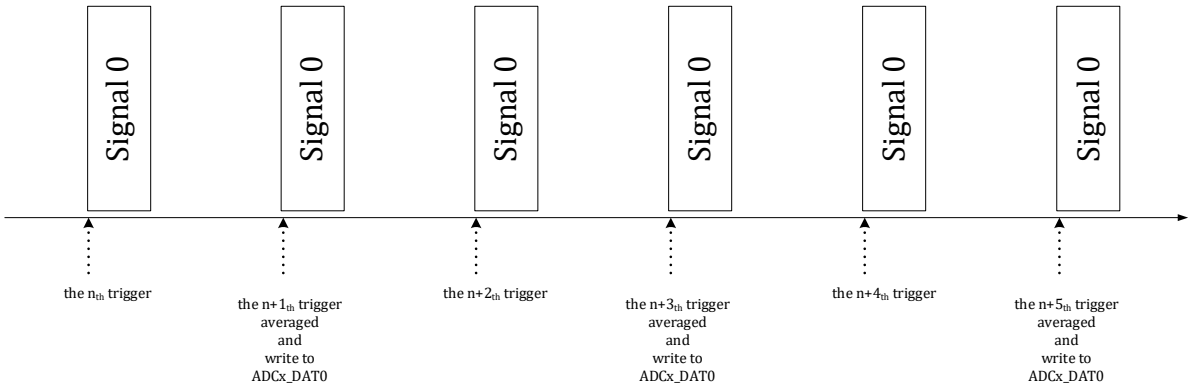


Figure 13-4 Oversampling Conversion Sequence Sample 2

13.2 Register

13.2.1 Address Allocation

The base address of ADC0 in chip is 0x4001_7800.

The base address of ADC1 in chip is 0x4001_7C00.

The base address of ADC2 in chip is 0x4001_8000.

Table 13-2 ADCx Register List

Name	Offset	Description
ADCx_DAT0	0x00	ADCx 0th sampling data
ADCx_DAT1	0x04	ADCx 1st sampling data
ADCx_DAT2	0x08	ADCx 2nd sampling data
ADCx_DAT3	0x0C	ADCx 3rd sampling data
ADCx_DAT4	0x10	ADCx 4th sampling data
ADCx_DAT5	0x14	ADCx 5th sampling data
ADCx_DAT6	0x18	ADCx 6th sampling data
ADCx_DAT7	0x1C	ADCx 7th sampling data
ADCx_DAT8	0x20	ADCx 8th sampling data
ADCx_DAT9	0x24	ADCx 9th sampling data
ADCx_DAT10	0x28	ADCx 10th sampling data
ADCx_DAT11	0x2C	ADCx 11th sampling data
ADCx_DAT12	0x30	ADCx 12th sampling data
ADCx_DAT13	0x34	ADCx 13th sampling data
ADCx_DAT14	0x38	ADCx 14th sampling data
ADCx_DAT15	0x3C	ADCx 15th sampling data
ADCx_PCHN0	0x40	ADCx 0th-3rd sampling positive input signal selection
ADCx_PCHN1	0x44	ADCx 4th-7th sampling positive input signal selection
ADCx_PCHN2	0x48	ADCx 8th-11th sampling positive input signal selection
ADCx_PCHN3	0x4C	ADCx 12th-15th sampling positive input signal selection
ADCx_NCHN0	0x50	ADCx 0th-3rd sampling negative input signal selection
ADCx_NCHN1	0x54	ADCx 4th-7th sampling negative input signal selection
ADCx_NCHN2	0x58	ADCx 8th-11th sampling negative input signal selection
ADCx_NCHN3	0x5C	ADCx 12th-15th sampling negative input signal selection
ADCx_CHNT	0x60	Number of ADCx sampling times in various trigger modes
ADCx_GAIN	0x70	ADCx gain selection
ADCx_CFG	0x74	ADCx mode configuration
ADCx_TRIG	0x78	ADCx sampling trigger configuration
ADCx_SWT	0x7C	ADCx software trigger
ADCx_DC0	0x80	DC offset when ADCx gain is 0
ADCx_AMC0	0x84	Gain calibration coefficient when ADCx gain is 0
ADCx_DC1	0x88	DC offset when ADCx gain is 1
ADCx_AMC1	0x8C	Gain calibration coefficient when ADCx gain is 1
ADCx_IE	0x90	ADCx interrupt enable
ADCx_IF	0x94	ADCx interrupt flag
ADCx_TH0	0x98	ADCx threshold 0
ADCx_GEN0	0x9C	ADCx threshold 0 monitoring enable
ADCx_TH1	0xA0	ADCx threshold 1

ADCx_GEN1	0xA4	ADCx threshold 1 monitoring enable

13.2.2 Sampling Data Register

13.2.2.1 ADCx_DAT0(x = 0,1,2)

The addresses are: 0x4001_7800, 0x4001_7C00, and 0x4001_8000.

Reset value: 0x0

Table 13-3 Sampling Data Register (ADCx_DAT0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT0															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	DAT0	ADCx 0th sampling data

13.2.2.2 ADCx_DAT1(x = 0,1,2)

The addresses are: 0x4001_7804, 0x4001_7C04, and 0x4001_8004.

Reset value: 0x0

Table 13-4 Sampling Data Register (ADCx_DAT1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT1															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	DAT1	ADCx 1st sampling data

13.2.2.3 ADCx_DAT2(x = 0,1,2)

The addresses are: 0x4001_7808, 0x4001_7C08, and 0x4001_8008.

Reset value: 0x0

Table 13-5 Sampling Data Register (ADCx_DAT2)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT2															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	DAT2	ADCx 2nd sampling data

13.2.2.4 ADCx_DAT3(x = 0,1,2)

The addresses are: 0x4001_780C, 0x4001_7C0C, and 0x4001_800C.

Reset value: 0x0

Table 13-6 Sampling Data Register (ADCx_DAT3)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT3															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	DAT3	ADCx 3rd sampling data

13.2.2.5 ADCx_DAT4(x = 0,1,2)

The addresses are: 0x4001_7810, 0x4001_7C10, and 0x4001_8010.

Reset value: 0x0

Table 13-7 Sampling Data Register (ADCx_DAT4)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT4															

RW
0

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	DAT4	ADCx 4th sampling data

13.2.2.6 ADCx_DAT5(x = 0,1,2)

The addresses are: 0x4001_7814, 0x4001_7C14, and 0x4001_8014.

Reset value: 0x0

Table 13-8 Sampling Data Register (ADCx_DAT5)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT5															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	DAT5	ADCx 5th sampling data

13.2.2.7 ADCx_DAT6(x = 0,1,2)

The addresses are: 0x4001_7818, 0x4001_7C18, and 0x4001_8018.

Reset value: 0x0

Table 13-9 Sampling Data Register (ADCx_DAT6)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT6															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	DAT6	ADCx 6th sampling data

13.2.2.8 ADCx_DAT7(x = 0,1,2)

The addresses are: 0x4001_781C, 0x4001_7C1C, and 0x4001_801C.

Reset value: 0x0

Table 13-10 Sampling Data Register (ADCx_DAT7)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT7															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	DAT7	ADCx 7th sampling data

13.2.2.9 ADCx_DAT8(x = 0,1,2)

The addresses are: 0x4001_7820, 0x4001_7C20, and 0x4001_8020.

Reset value: 0x0

Table 13-11 Sampling Data Register (ADCx_DAT8)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT8															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	DAT8	ADCx 8th sampling data

13.2.2.10 ADCx_DAT9(x = 0,1,2)

The addresses are: 0x4001_7824, 0x4001_7C24, and 0x4001_8024.

Reset value: 0x0

Table 13-12 Sampling Data Register (ADCx_DAT9)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT9															

RW
0

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	DAT9	ADCx 9th sampling data

13.2.2.11 ADCx_DAT10(x = 0,1,2)

The addresses are: 0x4001_7828, 0x4001_7C28, and 0x4001_8028.

Reset value: 0x0

Table 13-13 Sampling Data Register (ADCx_DAT10)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT10															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	DAT10	ADCx 10th sampling data

13.2.2.12 ADCx_DAT11(x = 0,1,2)

The addresses are: 0x4001_782C, 0x4001_7C2C, and 0x4001_802C

Reset value: 0x0

Table 13-14 Sampling Data Register (ADCx_DAT11)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT11															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	DAT11	ADCx 11th sampling data

13.2.2.13 ADCx_DAT12(x = 0,1,2)

The addresses are: 0x4001_7830, 0x4001_7C30, and 0x4001_8030.

Reset value: 0x0

Table 13-15 Sampling Data Register (ADCx_DAT12)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT12															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	DAT12	ADCx 12th sampling data

13.2.2.14 ADCx_DAT13(x = 0,1,2)

The addresses are: 0x4001_7834, 0x4001_7C34, and 0x4001_8034.

Reset value: 0x0

Table 13-16 Sampling Data Register (ADCx_DAT13)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT13															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	DAT13	ADCx 13th sampling data

13.2.2.15 ADCx_DAT14(x = 0,1,2)

The addresses are: 0x4001_7838, 0x4001_7C38, and 0x4001_8038.

Reset value: 0x0

Table 13-17 Sampling Data Register (ADCx_DAT14)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT14															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	DAT14	ADCx 14th sampling data

13.2.2.16 ADCx_DAT15(x = 0,1,2)

The addresses are: 0x4001_783C, 0x4001_7C3C, and 0x4001_803C.

Reset value: 0x0

Table 13-18 Sampling Data Register (ADCx_DAT15)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT15															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	DAT15	ADCx 15th sampling data

13.2.3 Signal Source Register

13.2.3.1 ADCx_PCHN0(x = 0,1,2)

The addresses are: 0x4001_7840, 0x4001_7C40, and 0x4001_8040.

Reset value: 0x0

Table 13-19 Positive Signal Source Register (ADCx_PCHN0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDS3				PDS2				PDS1				PDS0			
RW				RW				RW				RW			
0				0				0				0			

Bit field	Bit Name	Description
[31:16]		Unused
[15:12]	PDS3	ADCx 3rd sampling positive input signal selection
[11:8]	PDS2	ADCx 2nd sampling positive input signal selection
[7:4]	PDS1	ADCx 1st sampling positive input signal selection
[3:0]	PDS0	ADCx 0th sampling positive input signal selection

13.2.3.2 ADCx_PCHN1(x = 0,1,2)

The addresses are: 0x4001_7844, 0x4001_7C44, and 0x4001_8044.

Reset value: 0x0

Table 13-20 Positive Signal Source Register (ADCx_PCHN1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDS7				PDS6				PDS5				PDS4			
RW				RW				RW				RW			
0				0				0				0			

Bit field	Bit Name	Description
[31:16]		Unused
[15:12]	PDS7	ADCx 7th sampling positive input signal selection
[11:8]	PDS6	ADCx 6th sampling positive input signal selection
[7:4]	PDS5	ADCx 5th sampling positive input signal selection
[3:0]	PDS4	ADCx 4th sampling positive input signal selection

13.2.3.3 ADCx_PCHN2(x = 0,1,2)

The addresses are: 0x4001_7848, 0x4001_7C48, and 0x4001_8048.

Reset value: 0x0

Table 13-21 Positive Signal Source Register (ADCx_PCHN2)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDS11				PDS10				PDS9				PDS8			
RW				RW				RW				RW			
0				0				0				0			

Bit field	Bit Name	Description
[31:16]		Unused
[15:12]	PDS11	ADCx 11th sampling positive input signal selection
[11:8]	PDS10	ADCx 10th sampling positive input signal selection
[7:4]	PDS9	ADCx 9th sampling positive input signal selection
[3:0]	PDS8	ADCx 8th sampling positive input signal selection

13.2.3.4 ADCx_PCHN3(x = 0,1,2)

The addresses are: 0x4001_784C, 0x4001_7C4C, and 0x4001_804C.

Reset value: 0x0

Table 13-22 Positive Signal Source Register (ADCx_PCHN3)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDS15				PDS14				PDS13				PDS12			
RW				RW				RW				RW			
0				0				0				0			

Bit field	Bit Name	Description
[31:16]		Unused
[15:12]	PDS15	ADCx 15th sampling positive input signal selection
[11:8]	PDS14	ADCx 14th sampling positive input signal selection
[7:4]	PDS13	ADCx 13th sampling positive input signal selection
[3:0]	PDS12	ADCx 12th sampling positive input signal selection

13.2.3.5 ADCx_NCHN0(x = 0,1,2)

The addresses are: 0x4001_7850, 0x4001_7C50, and 0x4001_8050.

Reset value: 0x0

Table 13-23 Negative Signal Source Register (ADCx_NCHN0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDS3				NDS2				NDS1				NDS0			
RW				RW				RW				RW			
0				0				0				0			

Bit field	Bit Name	Description
[31:16]		Unused
[15:12]	NDS3	ADCx 3rd sampling negative input signal selection
[11:8]	NDS2	ADCx 2nd sampling negative input signal selection
[7:4]	NDS1	ADCx 1st sampling negative input signal selection
[3:0]	NDS0	ADCx 0th sampling negative input signal selection

13.2.3.6 ADCx_NCHN1(x = 0,1,2)

The addresses are: 0x4001_7854, 0x4001_7C54, and 0x4001_8054.

Reset value: 0x0

Table 13-24 Negative Signal Source Register (ADCx_NCHN1)

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

NDS7	NDS6	NDS5	NDS4
RW	RW	RW	RW
0	0	0	0

Bit field	Bit Name	Description
[31:16]		Unused
[15:12]	NDS7	ADCx 7th sampling negative input signal selection
[11:8]	NDS6	ADCx 6th sampling negative input signal selection
[7:4]	NDS5	ADCx 5th sampling negative input signal selection
[3:0]	NDS4	ADCx 4th sampling negative input signal selection

13.2.3.7 ADCx_NCHN2(x = 0,1,2)

The addresses are: 0x4001_7858, 0x4001_7C58, and 0x4001_8058.

Reset value: 0x0

Table 13-25 Negative Signal Source Register (ADCx_NCHN2)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDS11				NDS10				NDS9				NDS8			
RW				RW				RW				RW			
0				0				0				0			

Bit field	Bit Name	Description
[31:16]		Unused
[15:12]	NDS11	ADCx 11th sampling negative input signal selection
[11:8]	NDS10	ADCx 10th sampling negative input signal selection
[7:4]	NDS9	ADCx 9th sampling negative input signal selection
[3:0]	NDS8	ADCx 8th sampling negative input signal selection

13.2.3.8 ADCx_NCHN3(x = 0,1,2)

The addresses are: 0x4001_785C, 0x4001_7C5C, and 0x4001_805C.

Reset value: 0x0

Table 13-26 Negative Signal Source Register (ADCx_NCHN3)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDS15				NDS14				NDS13				NDS12			
RW				RW				RW				RW			
0				0				0				0			

Bit field	Bit Name	Description
[31:16]		Unused
[15:12]	NDS15	ADCx 15th sampling negative input signal selection
[11:8]	NDS14	ADCx 14th sampling negative input signal selection
[7:4]	NDS13	ADCx 13th sampling negative input signal selection
[3:0]	NDS12	ADCx 12th sampling negative input signal selection

13.2.3.9 ADC Analog Signal Selection

The three ADC interfaces can select different analog signals to be sampled, as shown in Table 13-27.

Table 13-27 ADC Analog Signal Source Distribution

Register	Analog Signal Corresponding to Different Configuration Value
ADC0_PCHNx	0000: OPA0 output 0001: OPA1 output 0010: OPA2 output 0011: OPA3 output 0100: OPA4 output 0101: OPA5 output 0110: ADC0_CH6 0111: ADC0_CH7 1000: ADC0_CH8 1001: ADC0_CH9 1010: ADC0_CH10 1011: ADC0_CH11 1100: ADC0_CH12 1101: ADC0_CH13 1110: ADC0_CH14 1111: Internal ground

ADC0_NCHNx	0000: Internal ground 0001: Internal ground 0010: Internal ground 0011: Internal ground 0100: Disable 0101: Disable 0110: ADC0_CH6 0111: ADC0_CH7 1000: ADC0_CH8 1001: ADC0_CH9 1010: ADC0_CH10 1011: ADC0_CH11 1100: ADC0_CH12 1101: ADC0_CH13 1110: ADC0_CH14 1111: Connect to internal ground
ADC1_PCHNx	0000: OPA0 output 0001: OPA1 output 0010: OPA2 output 0011: OPA3 output 0100: OPA4 output 0101: OPA5 output 0110: ADC1_CH6 0111: ADC1_CH7 1000: ADC1_CH8 1001: ADC1_CH9 1010: ADC1_CH10 1011: ADC1_CH11 1100: ADC1_CH12 1101: ADC1_CH13 1110: LDO12 1111: Internal ground

ADC1_NCHNx	0000: Internal ground 0001: Internal ground 0010: Internal ground 0011: Internal ground 0100: Disable 0101: Disable 0110: ADC1_CH6 0111: ADC1_CH7 1000: ADC1_CH8 1001: ADC1_CH9 1010: ADC1_CH10 1011: ADC1_CH11 1100: ADC1_CH12 1101: ADC1_CH13 1110: Internal ground 1111: Internal ground
ADC2_PCHNx	0000: OPA0 output 0001: OPA1 output 0010: OPA2 output 0011: OPA3 output 0100: ADC2_CH4 0101: ADC2_CH5 0110: ADC2_CH6 0111: ADC2_CH7 1000: ADC2_CH8 1001: ADC2_CH9 1010: ADC2_CH10 1011: ADC2_CH11 1100: ADC2_CH12 1101: ADC2_CH13 1110: Temperature sensor 1111: Internal ground

ADC2_NCHNx	0000: Internal ground 0001: Internal ground 0010: Internal ground 0011: Internal ground 0100: ADC2_CH4 0101: ADC2_CH5 0110: ADC2_CH6 0111: ADC2_CH7 1000: ADC2_CH8 1001: ADC2_CH9 1010: ADC2_CH10 1011: ADC2_CH11 1100: ADC2_CH12 1101: ADC2_CH13 1110: Internal ground 1111: Internal ground

When ADCx_PCHN selects the OPA output, ADCx_NCHN selects the negative end of the OPA output. Other values are invalid. That is, ADC samples the differential signal output by OPA. If the common-mode level Vcm of the op amp needs to be collected, it is recommended to output the amplified signal of the op amp to P3.11 or P4.6 through OPAx_OUT when the OPA input is short-circuited, and then sample it through ADC2_CH7 or ADC1_CH8.

13.2.4 Sampling Channel Number Register

13.2.4.1 ADCx_CHNT(x = 0,1,2)

The addresses are: 0x4001_7860, 0x4001_7C60, and 0x4001_8060.

Reset value: 0x0

Table 13-28 Sampling Channel Number Register (ADCx_CHNT)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								S2				S1			
								RW				RW			
								0				0			

Bit field	Bit Name	Description
[31:8]		Unused
[7:4]	S2	Second-round sampling channel number
[3:0]	S1	First-round sampling channel number

Note: The number of sampling channels cannot be set to 0. 1 means one channel, 2 means two

channels, ..., 15 means sixteen channels. If two-round sampling is adopted, the sum of two-round sampling channels should not exceed 16.

13.2.5 Configuration Register

13.2.5.1 ADCx_GAIN(x = 0,1,2)

The addresses are: 0x4001_7870, 0x4001_7C70, and 0x4001_8070.

Reset value: 0x0

Table 13-29 Gain Selection Register (ADCx_GAIN)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G15	G14	G13	G12	G11	G10	G9	G8	G7	G6	G5	G4	G3	G2	G1	G0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:12]		Unused
[15]	G15	DAT15 gain selection
[14]	G14	DAT14 gain selection
[13]	G13	DAT13 gain selection
[12]	G12	DAT12 gain selection
[11]	G11	DAT11 gain selection
[10]	G10	DAT10 gain selection
[9]	G9	DAT9 gain selection
[8]	G8	DAT8 gain selection
[7]	G7	DAT7 gain selection
[6]	G6	DAT6 gain selection
[5]	G5	DAT5 gain selection
[4]	G4	DAT4 gain selection
[3]	G3	DAT3 gain selection
[2]	G2	DAT2 gain selection
[1]	G1	DAT1 gain selection
[0]	G0	DAT0 gain selection

1: 1x gain mode, the maximum input signal amplitude is ±2.2V;

0: 2/3 times gain mode, the maximum input signal amplitude is ±3.3V;

Need to keep SYS_AFE_REG2.REF2VDD=0

13.2.5.2 ADCx_CFG(x = 0,1,2)

The addresses are: 0x4001_7874, 0x4001_7C74, and 0x4001_8074.

Reset value: 0x0

Table 13-30 Mode Configuration Register (ADCx_CFG)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			NSMP	FSM_RS	DATA_ALIGN		CSMP	TCNT				TROVS	OVSR		
			RW	RW	RW		RW	RW				RW	RW		
			0	0	0		0	0				0	0		

Bit field	Bit Name	Description
[31:13]		Unused
[12]	NSMP	0: Single-round sampling, 1: Two-round sampling
[11]	FSM_RS	State machine reset. After the software writes 1, the state machine returns to the idle state, and is automatically cleared upon completion. The reset control does not affect the configuration value of other ADC registers. Read the bit to return the current ADC status, 1 indicating that ADC is working on sampling conversion, and 0 indicating idle.
[10]	DATA_ALIGN	ADC_DAT alignment 0: Left aligned, with 2'h0 on the right, 1: right aligned, with 2bit sign bit on the left
[9]		Unused
[8]	CSMP	Continuous sampling mode 0: Not enable 1: Enable continuous sampling mode. After triggering, ADC will start sampling conversion to complete the conversion of all signals and start a new round of sampling conversion from 0th signal without waiting for the trigger signal
[7:4]	TCNT	Number of events required to trigger single-round sampling. 0: Indicating one event is required 1: Indicating two events are required …… 15: Indicating sixteen events are required
[3]	TROVS	Oversampling trigger mode 0: One trigger starts continuous oversampling for OVSR times, and the data is averaged to save in the register. Multiple

		channels can be configured for sampling at the same time, and each channel will be sampled for OVSR times before entering the sampling of the next channel. 1: One trigger starts one round of sampling conversion. ADC can sample enough data for averaging after OVSR trigger times and then save data to the register. In this configuration, ADCx_CHNT can only configure one channel for sampling due to the number of accumulators.
[2:0]	OVSR	Oversampling rate 0: 1, save data after 1 time of sampling by default 1: Save data after 2 times of sampling 2: Save data after 4 times of sampling 3: Save data after 8 times of sampling 4: Save data after 16 times of sampling 5: Save data after 32 times of sampling 6: Save data after 64 times of sampling 7: Save data after 128 times of sampling

13.2.5.3 ADCx_TRIG(x = 0,1,2)

The addresses are: 0x4001_7878, 0x4001_7C78, and 0x4001_8078.

Reset value: 0x0

Table 13-31 Sampling Trigger Configuration Register (ADCx_TRIG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
						UTIMER4_CMP1_EN	UTIMER4_CMP0_EN	UTIMER3_CMP1_EN	UTIMER3_CMP0_EN	UTIMER2_CMP1_EN	UTIMER2_CMP0_EN	UTIMER1_CMP1_EN	UTIMER1_CMP0_EN	UTIMER0_CMP1_EN	UTIMER0_CMP0_EN
						RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
						0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								MCPWM1_T3_EN	MCPWM1_T2_EN	MCPWM1_T1_EN	MCPWM1_T0_EN	MCPWM0_T3_EN	MCPWM0_T2_EN	MCPWM0_T1_EN	MCPWM0_T0_EN
								RW	RW	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

Bit field	Bit Name	Description

[31:26]		Unused
[25]	UTIMER4_CMP1_EN	UTIMER4 comparison event 1 trigger ADC sampling enable, active high
[24]	UTIMER4_CMP0_EN	UTIMER4 comparison event 0 trigger ADC sampling enable, active high
[23]	UTIMER3_CMP1_EN	UTIMER3 comparison event 1 trigger ADC sampling enable, active high
[22]	UTIMER3_CMP0_EN	UTIMER3 comparison event 0 trigger ADC sampling enable, active high
[21]	UTIMER2_CMP1_EN	UTIMER2 comparison event 1 trigger ADC sampling enable, active high
[20]	UTIMER2_CMP0_EN	UTIMER2 comparison event 0 trigger ADC sampling enable, active high
[19]	UTIMER1_CMP1_EN	UTIMER1 comparison event 1 trigger ADC sampling enable, active high
[18]	UTIMER1_CMP0_EN	UTIMER1 comparison event 0 trigger ADC sampling enable, active high
[17]	UTIMER0_CMP1_EN	UTIMER0 comparison event 1 trigger ADC sampling enable, active high
[16]	UTIMER0_CMP0_EN	UTIMER0 comparison event 0 trigger ADC sampling enable, active high
[15:8]		Unused
[7]	MCPWM1_T3_EN	MCPWM1 T3 event trigger ADC sampling enable, active high
[6]	MCPWM1_T2_EN	MCPWM1 T2 event trigger ADC sampling enable, active high
[5]	MCPWM1_T1_EN	MCPWM1 T1 event trigger ADC sampling enable, active high
[4]	MCPWM1_T0_EN	MCPWM1 T0 event trigger ADC sampling enable, active high
[3]	MCPWM0_T3_EN	MCPWM0 T3 event trigger ADC sampling enable, active high
[2]	MCPWM0_T2_EN	MCPWM0 T2 event trigger ADC sampling enable, active high
[1]	MCPWM0_T1_EN	MCPWM0 T1 event trigger ADC sampling enable, active high
[0]	MCPWM0_T0_EN	MCPWM0 T0 event trigger ADC sampling enable, active high

The trigger source of ADC is MCPWM or UTimer, and only one can be selected.

The trigger signal of UTimer to ADC can be sent by setting GPIO as the Timer (7th/8th) function, which is used to output Timer channel signal for observation.

The trigger signal of MCPWM to ADC can be sent by setting GPIO as the 9th function, that is, ADC_TRIGGER function, which is used to capture and debug. Every time an ADC trigger occurs, the ADC_TRIGGER signal flips over for capture output.

When two-round trigger is used, only the trigger signal of BIT0/2/4/6.../16/18/..., etc. can be used as the first-round trigger signal; and the trigger signal of BIT1/3/5/.../17/19/..., etc. can be used as the second-round trigger signal.

For example, when MCPWM0_T1_EN and MCPWM0_T0_EN are enabled, MCPWM0_T0_EN will trigger the first round of sampling but not the second round; and MCPWM0_T1_EN will trigger the second round of sampling, but not the first round.

13.2.6 Software Trigger Register

13.2.6.1 ADCx_SWT(x = 0,1,2)

The addresses are: 0x4001_787C, 0x4001_7C7C, and 0x4001_807C.

Reset value: 0x0

Table 13-32 Software Trigger Register (ADCx_SWT)

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

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	SWT	Trigger once when writing 0x5AA5.

Please note that the software trigger acquisition register is write-only, and the software trigger event is generated only when the write data is 0x5AA5. One write to the bus generates one software trigger. When a software trigger is generated after data is written, the register is automatically cleared, and 0x5AA5 will be written again after the next software trigger.

13.2.7 Calibration Register

One signal source of ADC optional analog channels is GND, and the DC offset of the system can be obtained by measuring this channel.

Generally, the system will carry out measurement of GND signal once after initialization, and store the measured value to DC offset register. Then, the sampling values of other channels'signals will automatically subtract DC offset each time, and then the converted data will be stored in the digital register.

Considering signal error, the signal subtracting the DC offset may be overflow. In this case, data will carry out saturation processing to make the signal in the range of -1.2V-+1.2V or -2.2V-+2.2V.

The ADC has two gain modes: high gain (1 times) and low gain (2/3 times). The ADC ranges of

these two gains are also different. In 1x gain mode, the maximum input signal amplitude is ±2.2V; in 2/3 gain mode, the maximum input signal amplitude is ±3.3V. Two ranges use different DC offsets and gain calibration coefficients. Therefore, there are two sets of calibration registers, i.e. ADCx_DC0, ADCx_AMC0 and ADCx_DC1, ADCx_AMC1.The chip has been calibrated by the factory before delivery, and the calibration data is stored in Flash info. When the chip is powered on, the calibration parameters are automatically loaded. When initializing the ADC module, you need to configure the DC offset according to the data alignment mode. See the library function provided by the chip vendor. After the ADC module is soft reset using SYS_SFT_RST, the registers inside the ADC are reset. The ADC initialization function needs to be reused to read the calibration parameters such as DC/AMC from the NVR.

13.2.7.1 ADCx_DC0(x = 0,1,2)

The addresses are: 0x4001_7880, 0x4001_7C80, and 0x4001_8080.

Reset value: 0x0

Table 13-33 0-level Gain DC Offset Register (ADCx_DC0)

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

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	DC_OFFSET	ADC sampling circuit DC offset

Considering that ADC's DC offset is a value close to 0, the register can only implement the low 10-bit. The high 8-bit is only the sign extension of ADCx_DC[9], and the ADCx_DC register will also perform sign extension when participating in ADC data correction.

If writing 0x12B0 to ADCx_DC, the actual value written is 0x2B0, and the read value is 0xFFB0.

The ADC_DC value stored should correspond to the right-aligned offset value. When the ADCx_CFG register is configured to be left-aligned, the hardware will automatically adjust ADCx_DC to participate in ADC correction according to the alignment setting. Specifically, when right-aligned, ADCx_DC[15:0] directly participates in the correction operation; when left-aligned, ADCx_DC[15:0] will first shift to the left by 2 bits and then participate in the ADC correction operation.

13.2.7.2 ADCx_AMC0(x = 0,1,2)

The addresses are: 0x4001_7884, 0x4001_7C84, and 0x4001_8084.

Reset value: 0x200

Table 13-34 0-level Gain Calibration Register (ADCx_AMC0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						AM_CALI									
						RW									
						0x200									

Bit field	Bit Name	Description
[31:10]		Unused
[9:0]	AM_CALI	ADC sampling circuit gain calibration coefficient

ADC gain calibration coefficient is a value close to 1, and 0x1F0 is calibrated to 1. For example, 0x1F0 is less than 1, and 0x205 is greater than 1.

13.2.7.3 ADCx_DC1(x = 0,1,2)

The addresses are: 0x4001_7888, 0x4001_7C88, and 0x4001_8088.

Reset value: 0x0

Table 13-35 1-level Gain DC Offset Register (ADCx_DC1)

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

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	DC_OFFSET	ADC sampling circuit DC offset

13.2.7.4 ADCx_AMC1(x = 0,1,2)

The addresses are: 0x4001_788C, 0x4001_7C8C, and 0x4001_808C.

Reset value: 0x200

Table 13-36 1-level Gain Calibration Register (ADCx_AMC1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						AM_CALI									
						RW									
						0x200									

Bit field	Bit Name	Description
[31:10]		Unused
[9:0]	AM_CALI	ADC sampling circuit gain calibration coefficient

13.2.8 Interrupt Enable Register

13.2.8.1 ADCx_IE(x = 0,1,2)

The addresses are: 0x4001_7890, 0x4001_7C90, and 0x4001_8090.

Reset value: 0x0

Table 13-37 Interrupt Enable Register (ADCx_IE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	HERR_RE	SERR_RE		AWD1_RE	AWD0_RE	SF2_RE	SF1_RE		HERR_IE	SERR_IE		AWD1_IE	AWD0_IE	SF2_IE	SF1_IE
	RW	RW		RW	RW	RW	RW		RW	RW		RW	RW	RW	RW
	0	0		0	0	0	0		0	0		0	0	0	0

Bit field	Bit Name	Description
[31:15]		Unused
[14]	HERR_RE	Hardware trigger DMA request enable that occurs in non-idle state
[13]	SERR_RE	Software trigger DMA request enable that occurs in non-idle state
[12]		Unused
[11]	AWD1_RE	DMA request enable triggered by threshold monitoring 1 overlimit
[10]	AWD0_RE	DMA request enable triggered by threshold monitoring 0 overlimit
[9]	SF2_RE	DMA request enable triggered by the completion of the second-round sampling
[8]	SF1_RE	DMA request enable triggered by the completion of the first-round sampling
[7]		Unused
[6]	HERR_IE	Hardware trigger interrupt enable that occurs in non-idle state
[5]	SERR_IE	Software trigger interrupt enable that occurs in non-idle state
[4]		Unused
[3]	AWD1_IE	Interrupt enable triggered by threshold monitoring 1 overlimit
[2]	AWD0_IE	Interrupt enable triggered by threshold monitoring 0 overlimit
[1]	SF2_IE	Interrupt enable triggered by the completion of the second-round sampling
[0]	SF1_IE	Interrupt enable triggered by the completion of the first-round

		sampling

13.2.8.2 ADCx_IF(x = 0,1,2)

The addresses are: 0x4001_7894, 0x4001_7C94, and 0x4001_8094.

Reset value: 0x0

Table 13-38 Interrupt Flag Register (ADCx_IF)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									HERR_IF	SERR_IF		AWD1_IF	AWD0_IF	SF2_IF	SF1_IF
									RW1C	RW1C		RW1C	RW1C	RW1C	RW1C
									0	0		0	0	0	0

Bit field	Bit Name	Description
[31:7]		Unused
[6]	HERR_IF	Hardware trigger interrupt flag that occurs in non-idle state
[5]	SERR_IF	Software trigger interrupt flag that occurs in non-idle state
[4]		Unused
[3]	AWD1_IF	Interrupt flag triggered by threshold monitoring 1 overlimit
[2]	AWD0_IF	Interrupt flag triggered by threshold monitoring 0 overlimit
[1]	SF2_IF	Interrupt flag triggered by the completion of the second-round sampling
[0]	SF1_IF	Interrupt flag triggered by the completion of the first-round sampling

In the above ADCx_IF flag, 0: indicates that no interrupt has occurred, 1: indicates that an interrupt has occurred, write 1 to clear.

13.2.9 Analog Watchdog

13.2.9.1 ADCx_TH0 (x = 0,1,2)

The addresses are: 0x4001_7898, 0x4001_7C98, and 0x4001_8098.

Reset value: 0x07FF_0000

Table 13-39 Threshold Register 0 (ADCx_TH0)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		HTH													
		RW													
		0x1FFF													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		LTH													
		RW													
		0x2000													

Bit field	Bit Name	Description
[31:30]		Unused
[29:16]	HTH	Higher threshold of ADC analog watchdog 0
[15:14]		Unused
[13:0]	LTH	Lower threshold of ADC analog watchdog 0

13.2.9.2 ADCx_GEN0 (x = 0,1,2)

The addresses are: 0x4001_789C, 0x4001_7C9C, and 0x4001_809C.

Reset value: 0x0

Table 13-40Monitoring Enable Register 0 (ADCx_GEN0)

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

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	GEN	ADC analog watchdog 0 enable bit BIT0: DAT0 watchdog monitoring enable BIT1: DAT1 watchdog monitoring enable …… BIT15: DAT15 watchdog monitoring enable

For example, if ADC_GEN[1]=1, the watchdog will monitor the value of ADCx_DAT1. If the value is greater than the ADCx_TH0.HTH or smaller than the ADCx_TH0.LTH, an interrupt flag will be generated.

13.2.9.3 ADCx_TH1(x = 0,1,2)

The addresses are: 0x4001_78A0, 0x4001_7CA0, and 0x4001_80A0.

Reset value: 0x07FF_0000

Table 13-41 Threshold Register 1 (ADCx_TH1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		HTH													
		RW													
		0x1FFF													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		LTH													
		RW													
		0x2000													

Bit field	Bit Name	Description
[31:30]		Unused
[29:16]	HTH	Higher threshold of ADC analog watchdog 1
[15:14]		Unused
[13:0]	LTH	Lower threshold of ADC analog watchdog 1

13.2.9.4 ADCx_GEN1 (x = 0,1,2)

The addresses are: 0x4001_78A4, 0x4001_7CA4, and 0x4001_80A4.

Reset value: 0x0

Table 13-42 Monitoring Enable Register 0 (ADCx_GEN1)

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

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	GEN	ADC analog watchdog 1 enable bit BIT0: DAT0 watchdog monitoring enable BIT1: DAT1 watchdog monitoring enable …… BIT15: DAT15 watchdog monitoring enable

13.3 Application Guide

13.3.1 ADC Sampling Trigger Mode

The ADC supports one-round and two-round sampling modes. The sampling start of each round

requires a specific external event to trigger, and each round of sampling supports setting different sampling times and sampling signal channels.

First Trigger

ADC sampling can be triggered by the timing event TADC[0]/TADC[1]/TADC[2]/TADC[3] from MCPWM/UTimer, any one or several trigger samples of four trigger sources are available for selection, and it can also be triggered by writing command words to ADCx_SWT using 16'h5AA5 software.

First Round of Sampling

Determine whether it is one sampling round.

Yes: When the sampling times reaches the preset value ADCx_CHNT[3:0], the ADC will return to the idle state 0; if sampling times has not reached the preset value, the sampling continues.

No: When the sampling times reaches the preset value ADCx_CHNT[3:0], the ADC will enter the idle state 1 (the first round of two-round or four-round sampling is completed, waiting for the second round to be triggered); if the sampling times has not reached the preset value, the first round of sampling continues.

Second Trigger

Second Round of Sampling

When the sampling times of the fourth round reaches the preset value ADC_CHNT[7: 4], the round of sampling will end, and the ADC will return to the idle state 0.

The trigger conditions of various hardware trigger modes are summarized in Table 13-43. The single-round sampling mode is different. It can determine by the ADC0_CFG register that whether one TADC event triggers sampling, or the sampling is triggered only when a certain number of TADC events had occurred, while the two-round and four-round sampling modes only support the sampling to be triggered by one TADC event.

Besides, the ADC module also supports the sampling to be triggered by writing special value through software. The software trigger also only supports trigger sampling by writing once.

Table 13-43 ADC Sampling Trigger Mode

<table>
<tr><th></th><th>Single-round Trigger</th><th>Two-round Trigger</th></tr>
<tr><td rowspan="7">TADC
Trigger</td><td>None (TADC trigger is not enabled)</td><td rowspan="7">The first round of TADC [0]
The second round of TADC [1]</td></tr>
<tr><td>C times of TADC [0]</td></tr>
<tr><td>C times of TADC [1]</td></tr>
<tr><td>C times of TADC [2]</td></tr>
<tr><td>C times of TADC [3]</td></tr>
<tr><td>C times
TADC[0]/TADC[1]/
TADC[2]/TADC[3]</td></tr>
<tr><td>Software
trigger</td><td>Write 16'h5aa5 to ADC_SWT</td><td>First round: Write 16'h5aa5 to
ADC_SWT</td></tr>
</table>

		Second round: Write 16'h5aa5 to ADC_SWT

Where, when multiple triggering events are configured to trigger ADC to start sampling, C times = ADCx_CFG.TCNT.

13.3.1.1 Single-round Trigger Mode

Single-round triggering mode refers to that ADC completes a sampling action when a trigger is received. One round of sampling may include multiple samplings of the analog signal, and the sampling times are set by the round register ADCx_CHNT; when the register value is 1-15, the corresponding sampling times are 1-15.

The trigger event can be triggered by the external MCPWM/UTimer timing signals TADC [0], TADC [1], TADC [2], TADC [3] to a preset number of times, or triggered by software.

Each sampled signal source is set and selected by the signal source register ADC_PCHN0/1/2/3 and ADC_NCHN0/1/2/3. The signal source should be selected before the trigger and shall remain unchanged before the sampling completed.

It will enter the idle state and generate a sampling completion interrupt upon the completion of one-round sampling.

Take the single-round sampling triggered by MCPWM as an example, set that the event is triggered when the ADCx_CFG.TCNT=4 and ADCx_TRIG.MCPWM0_T2_EN =1, that is MCPWM0 TADC [2] occurs four times. The state transition is shown in Figure 13-5 .

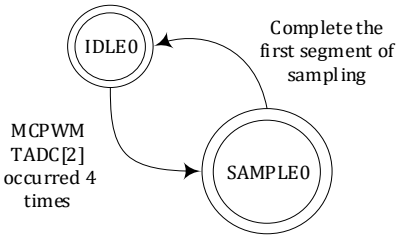


Figure 13-5 State Transition Diagram of ADC Single-round Sampling

13.3.1.2 Two-round Trigger Mode

Two-round trigger requires two triggers to complete one round of sampling. The first round is sampled when the first trigger arrives and the second round is sampled when the second trigger arrives.

The trigger event can be triggered by external MCPWM timing signals TADC [0] and TADC [1] or triggered twice by software.

When TADC [0] or software trigger occurs, the ADC_CHNT[3:0] sampling starts. And then, it will enter the idle state upon sampling completion and wait for the next trigger signal; when TADC [1] or software trigger occurs as the second trigger signal, the ADC_CHNT[7:4] sampling starts. The sampling times are set by the ADC_CHNT round register.

Each sampled signal source is set and selected by the register. The signal source should be selected before the trigger and shall remain unchanged before the sampling completed.

Software trigger has a lower priority than hardware trigger. If a software trigger occurs during the hardware-triggered sampling, the state machine will not process it, but generates an error interrupt. That is, the sampling request triggered by the software will be processed only when the state machine is in the idle state. If software trigger is required, the hardware trigger should be turned off in advance. Then write a 0x5AA5 to the ADC_SWT register to generate a software trigger.

Take the double-round sampling triggered by two software trigger as an example, the state transition is shown in Figure 13-6.

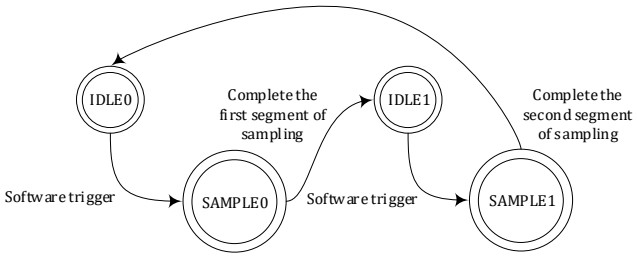


Figure 13-7 State Transition Diagram of ADC Two-round Sampling

13.3.2 Interruption

13.3.2.1 Done Interruption of Single Round Trigger Sampling

An interrupt flag is generated when sampling is completed.

13.3.2.2 Done Interruption of Two Round Trigger Sampling

An interrupt flag is generated when the first-round sampling is completed, and another interrupt flag is triggered when the second-round sampling is completed.

13.3.3 Configuration Modification

It is recommended to configure and modify ADC_CHNTx in the ADC interrupt. After entering the ADC interrupt, it means that the ADC has completed one round of sampling and is now in an idle state. Since the ADC operating status cannot be confirmed in the main program, the ADC trigger should be turned off in advance if the ADC_CHNx and ADC_CHNT registers should be modified in the main program, and then write "1" to ADC_CFG[11] to reset the ADC interface circuit state machine, thus ensuring that the ADC is not in a working state. If the ADC settings change during operation, the subsequent behavior will be unpredictable.

The sample program is as follows:

```
ADCx_CFG_temp = ADCx_CFG;        //Save ADCx_CFG
ADCx_CFG = 0x0000;               //Disable ADC trigger
ADCx_CFG = 0x0800;               //Reset ADC state machine
/*
```

Add your code below, like:

```
ADCx_CHNT = 0x0005
ADCx_CHN0 = 0x3210;
ADCx_CHN1 = 0x7654;
*/
ADCx_CFG = ADCx_CFG_temp;          //restore ADCx_CFG
```

13.3.4 Select the Corresponding Analog Channel

For the ADC sampling signal source, refer to Table 13-27, or refer to Table 2.2 Pin Function Selection in DATASHEET. Turn off the corresponding IO IE and OE to use the simulation function.

14 Universal Timer (UTimer)

14.1 Overview

14.1.1 Functional block diagram

As shown in Figure 14-1, the universal timer UTIMER mainly includes 5 independent Timers, and 4 quadrature encoder modules, which can be independently configured to run the count clock and filter constant. Each Timer can be used to output a waveform with a specific duty cycle, or it can capture an external waveform to detect the duty cycle.

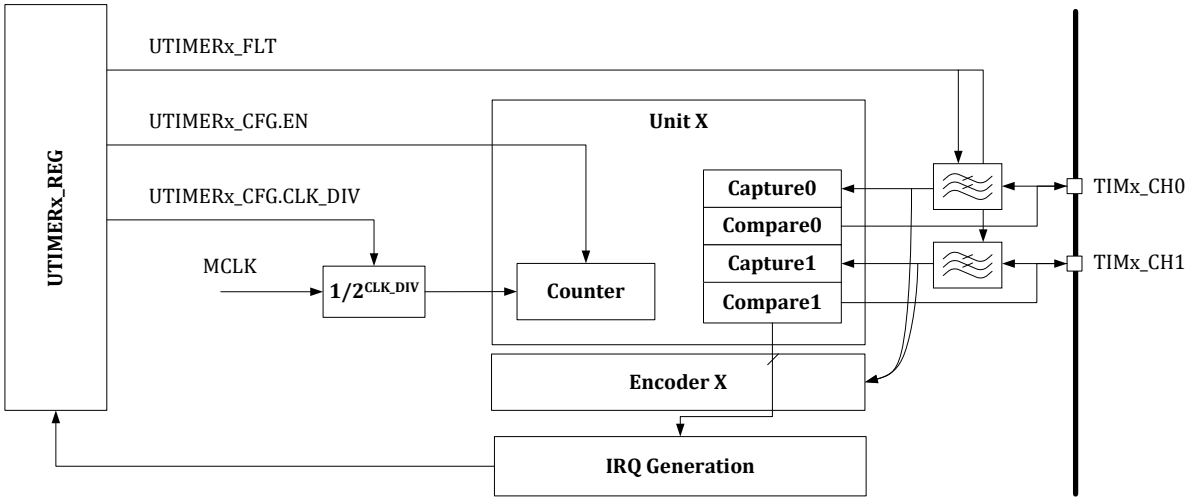


Figure 14-1 Block Top Functional Block Diagram

14.1.1.1 Bus Interface Module

The bus interface module includes:

Translate the access signals from the AHB bus into register read and write signals, control the clock of the register module, and initiate read and write to the register module.

CG clock gating module. When the AHB bus has no access, turn off the register module clock to reduce power consumption.

14.1.1.2 Register Module

UITMERx_REG module

Realize read and write control registers of each submodule.

Realize access to the status and result registers of each submodule.

Realize interrupt signal processing and interrupt generation for each submodule.

14.1.1.3 IO Filter Module

The IO filter module filters the input signal from outside the chip to reduce the effect of glitches

on the timer.

14.1.1.4 Universal Timer Module

The utimer_unt module implements general timer functions, including comparison and capture modes, which can process two external input signals or generate two pulse signals to be transmitted outside the chip.

utimer_unt module, support external events to start counting, and the source of external events can be configured. When an external event is triggered, the utimer_unt timer starts to increment. External signal can be used as the timer clock for counting.

14.1.1.5 Encoder module

The encoder module is used to count the encoder code signals sent from outside the chip. Clock Divider Module

The clock frequency dividing module is used to generate various signals of clock frequency dividing.

The timer module has a total of 5 independent general timers. The bit width of Timer0/Unit0 and Timer1/Unit1 is 16-bit, and the bit width of Timer2/Unit2 and Timer3/Unit3 is 32-bit. Each timer has two channels.

There are four encoder modules integrated on the timer module, where the encoder 0/1/2/3 inputs are from Timer0/1/2/3 channel 0/1, respectively.

14.1.2 Features

The timer module has the following characteristics:

- ✧ The independent counter is available for working in at different frequencies independently
- ✧ Timer0, Timer1 and Timer4 is a 16-bit general timer
- ✧ Timer2 and Timer3 is a 32-bit general timer
- ✧ Each general timer processes two external input signals (capture mode) or generates two output signals (comparison mode)
- ✧ Filter each input signal
- ✧ Timer can be triggered as DMA transmission

14.2 Implementation Description

14.2.1 Clock Divider

Each timer works at the main system frequency, and the frequency division counter is adopted to reduce the counting frequency, thus to realize the independent frequency division of each timer for

better write interrupt/count value.

14.2.2 Interrupt Flag Clear

The product adopts the design of writing 1 to each interrupt flag to clear it.

14.2.3 Filtering

The timer module has a total of 10/5 pairs of channel inputs, and the timer can filter each input to varying degrees.

The filter width can be adjusted by setting the filter register, 0-2040 Timer count clock widths are available.

Usually, Timer count clock is used for filtering input signals, that is, the clock after clock division of high-speed system clock according to UTIMER_UNTx_CFG.CLK_DIV.

As shown in Figure 14-2, the original input signal was reversed at several moments from t1 to t6, and the filter width was set as T. It can be seen that only the inversions that occur at times t3 and t6 are maintained for a time greater than T, and we can see from the filter output that the signal has only inverted twice.

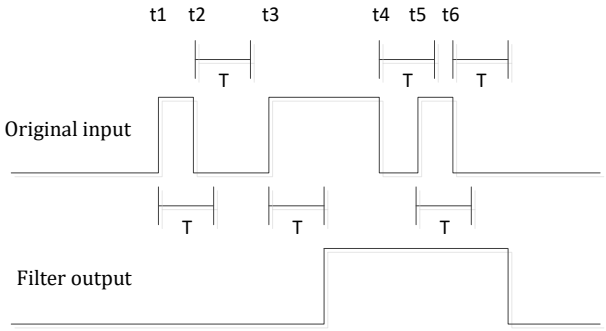


Figure 14-2 UTimer Filter Diagram

14.2.4 Mode

14.2.4.1 Counter

The counter in Timer counts in increasing direction.

The counter counts from 0 to the TH value, and then returns to 0 to restart counting. When the counter returns to 0, a zero return interrupt is generated. The actual counting period is clk_freq*(TH+1).

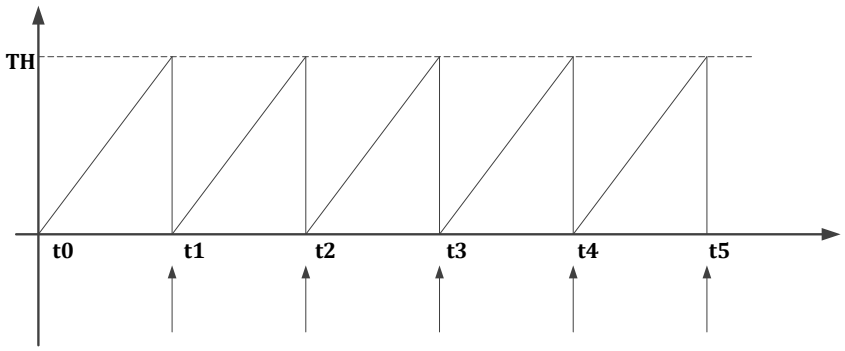


Figure 14-3 UTimer Universal Counter

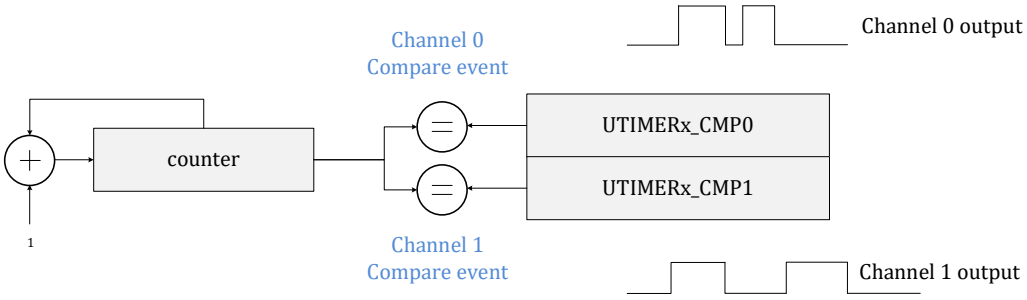
14.2.4.2 Comparison Mode

In comparison mode, a compare interrupt is generated when the counter counts to the UTIMERx_CMP0/ UTIMERx _value, and a compare pulse can be driven.

For Timer channel 0, when returning to zero, it will output a level (polarity can be configured) to the TIMERx_CH0; when the comparison event occurs, that is counting to UTIMERx_CMP0, the level is inverted, and another level is output to the IO port TIMERx_CH0.

For Timer channel 1, when returning to zero, it will output a level (polarity can be configured) to the TIMERx_CH1; when the comparison event occurs, that is counting to UTIMERx_CMP1, the level is inverted, and another level is output to the IO port TIMERx_CH1.

When the counter returns to zero, the zero return interrupt will still be generated. If setting UTIMERx_CMP0=0, Timer X channel 0 can always be 1; if setting UTIMERx_CMP0=UTIMERx_TH+1, Timer channel 0 can always be 0.



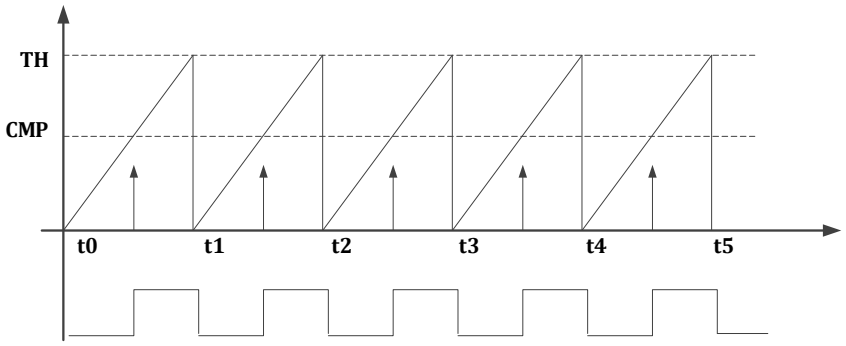


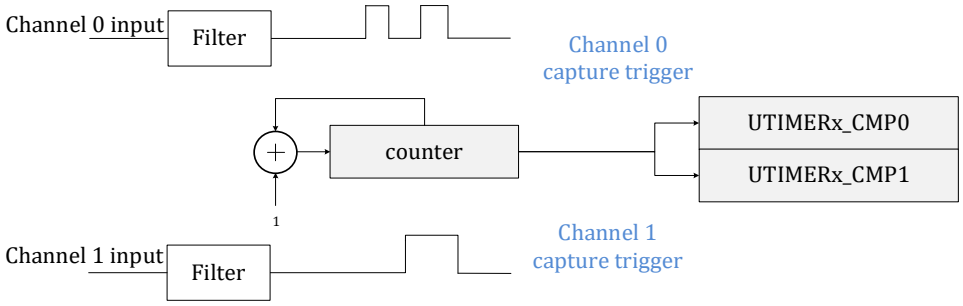
Figure 14-4 UTimer Comparison Mode

14.2.4.3 Capture Mode

In capture mode, Timer can detect the rising/falling or double edge of the input signal in the capture mode. When a capture event (that is, the input signal level changes) occurs, the timer count value is stored in the UTIMERx_CMP register and a capture interrupt is generated. When the counter returns to zero, the zero return interrupt will still be generated.

For Timer channel 0, it can be configured to detect the rising/falling or double edge. When the input signal of channel 0 is specially reversed, Timer will write the counter value to the UTIMERx_CMP0.

For Timer channel 1, it can be configured to detect the rising/falling or double edge. When the input signal of channel 1 is specially reversed, Timer will write the counter value to the UTIMERx_CMP1.



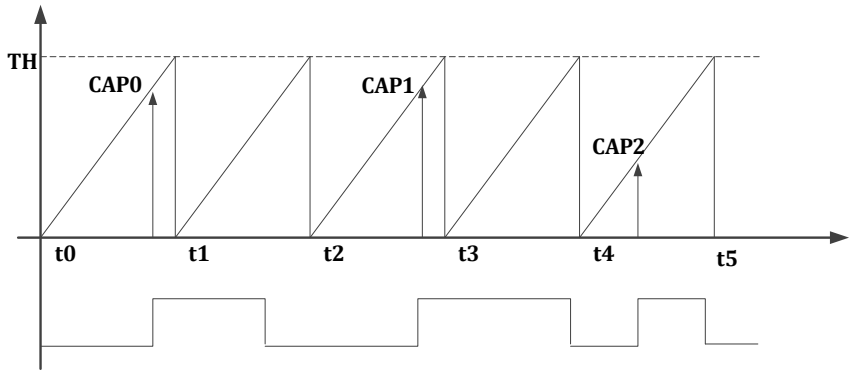


Figure 14-5 UTimer Capture Mode

As shown in Figure 14-5, the timer is set to capture on the rising edge. When changes in the rising edge of the input signal is captured at timing CAP0, CAP1, and CAP2, the timer count value at the corresponding time will be stored in the UTIMERx_CMP register.

14.2.5 ADC Trigger

The comparison event of Timer can be used as the ADC sampling trigger event. For details, refer to 13.2.5.3 ADCx_TRIG(x = 0,1,2).

14.2.6 Encoder

The encoder interface supports three modes: orthogonal coded signal, pulse signal with symbolic data, and CW/CCW double pulse signal.

T1 and T2 input signals of QEP0 come from GPIO input corresponding to Timer0 Channel0 and Channel1 respectively; T1 and T2 input signals of QEP1 come from GPIO input corresponding to Timer1 Channel0 and Channel1 respectively, and so on for QEP2/3. Enabling the encoder function does not affect the normal use of the Timer function.

The input signals T1/T2/Z of encoder are controlled by the corresponding Timer filtering coefficient. For example, Z signal of QEP0 is controlled by UTIMER0_FLT filtering time.

To use the encoder, the corresponding Timer clock in SYS_CLK_FEN shall be enabled. To use QEP1, the Time1 peripheral clock shall be enabled.

14.2.6.1 Orthogonal Coded Signal

The quadrature coded signal is mostly used to count the number of encoder turns. The input signals are T1 and T2, which support the two modes in the following table.

Generally speaking, the transition edge of T1 and T2 will cause the counter to increment or decrement. The counter counting direction (increasing or decreasing) is determined by the level of another steady-state signal other than the transition signal.

If T1 has a rising edge transition, then check whether T2 is high or low level. If it is a high level,

the counter decrements; if it is a low level, the counter increments; the change of T1 falling edge counter is just the opposite.

If T2 has a rising edge transition, then check whether T1 is high or low level. If it is a high level, the counter decrements; if it is a low level, the counter increments; the change of T2 falling edge counter is just the opposite.

The formula is as follows:

Counter Up = (T1 !=T2) @(T1 triggering edges) | (T1 == T2) @ (T2 triggering edges)

Counter Down = (T1 == T2) @ (T1 triggering edges) | (T1 != T2) @ (T2 triggering edges)

Table 14-1 Working Mode of Encoder Orthogonal Coding

Counting mode	T1/T2 level state (steady state signal)	T1 change edge state		T2 change edge state	
		Rising edge	Falling edge	Rising edge	Falling edge
T1 count only	T2 high level	Decrement	Increment	Not count	Not count
	T2 low level	Increment	Decrement	Not count	Not count
T1/T2 count	T2 high level	Decrement	Increment	Not count	Not count
	T2 low level	Increment	Decrement	Not count	Not count
	T1 high level	Not count	Not count	Increment	Decrement
	T1 low level	Not count	Not count	Decrement	Increment

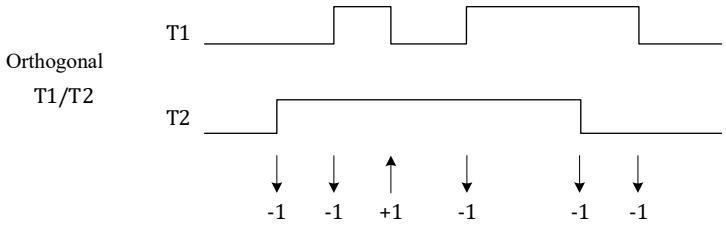


Figure 14-6 The Counts of Orthogonal Coded Signal when the Encoder Counts Only at Time T1

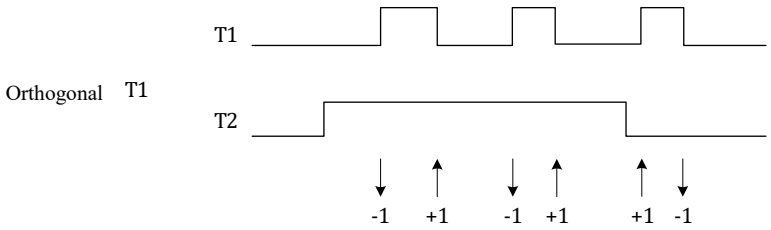


Figure 14-7 The Counts of Orthogonal Coded Signal when the Encoder Counts Only at Time T1 or T2

14.2.6.2 Pulse Signal with Symbolic Data Type

In this mode of operation, T1 is a pulse signal, and T2 is a signed signal. The edge of T1 triggers counting and the level of T2 controls the counting direction; if it is a high level, it increases; and if it is a low level, it decreases. It can also be set to count only T1 rising edges or both T1 rising and falling

edges.

Counter Up = (T2==1) @ (T1 triggering edges)

Counter Down = (T2==0) @ (T1 triggering edges)

Table 14-2 Working Mode of Pulse Signal with Symbolic Data Type

Counting mode	T2 level state (steady state signal)	T1 change edge state	
		Rising edge	Falling edge
Only T1 rising edge	Height	Increment	Not count
	Low	Decrement	Not count
T1 rising and falling edges	Height	Increment	Decrement
	Low	Decrement	Increment

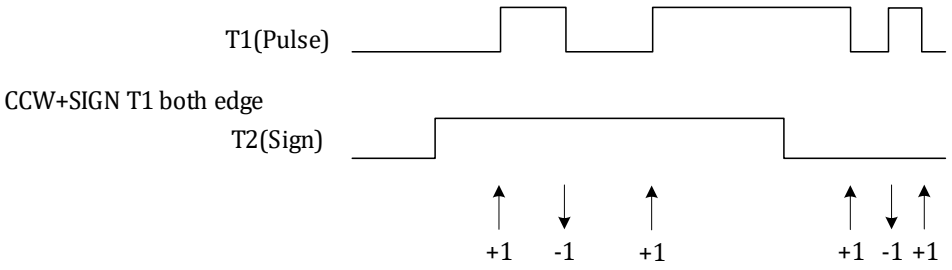


Figure 14-8 The Counts of Pulse Signal with Symbolic Data Type when the Encoder Counts both on T1 Rising and Falling Edges

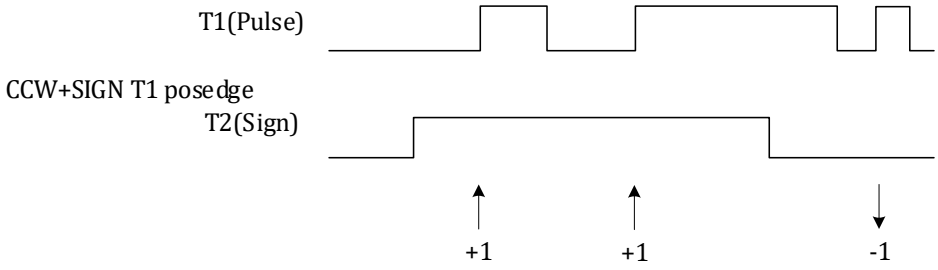


Figure 14-9 The Counts of Pulse Signal with Symbolic Data Type when the Encoder Counts only on T1 Rising Edges

14.2.6.3 CCW/CW Double Pulse Signal

The counter increments when T1 transitions, and decrements when T2 transitions. It can also be set to count only the rising edges or both the rising and falling edges. The formula is as follows:

Counter Up = 1 @ (T1 triggering edges)

Counter Down = 1 @ (T2 triggering edges)

Table 14-3 Encoder CCW/CW Double Pulse Working Mode

Counting mode	Changing edge state			
	T1 rising edge	T1 falling edge	T2 rising edge	T2 falling edge
T1/T2 rising edge	Increment	Not count	Decrement	Not count
T1/T2 rising and falling edges	Increment	Increment	Decrement	Decrement

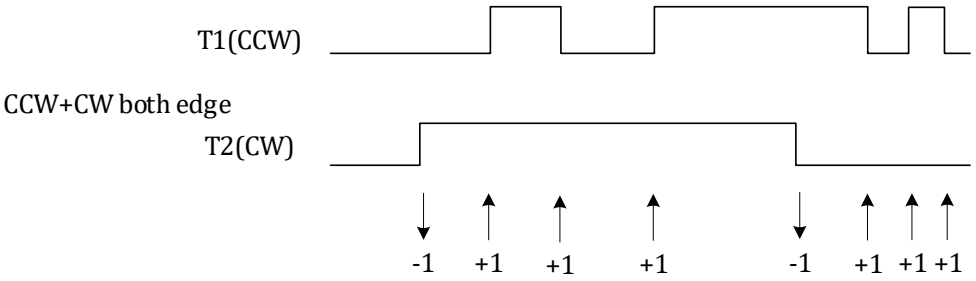


Figure 14-10 CCW/CW Double Pulse Signal Counting when Encoder Counts only on the T1 and T2 Rising Edge

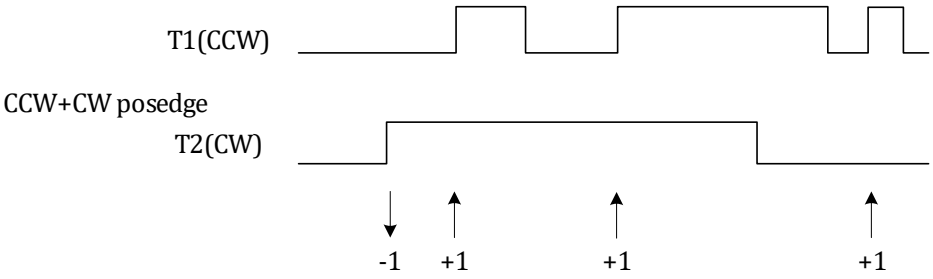


Figure 14-11 CCW/CW Double Pulse Signal Counting when Encoder Counts both on the Rising and Falling Edge T1 and T2

14.3 Register

14.3.1 Address Allocation

The base address of Timer0 in chip is 0x4001_4000.

Table 14-4Address Allocation of Timer0 Register

Name	Offset	Description
UTIMER0_CFG	0x00	Timer0 configuration register
UTIMER0_TH	0x04	Timer0 count threshold register
UTIMER0_CNT	0x08	Timer0 count value register
UTIMER0_CMP0	0x0C	Timer0 compare/capture register 0
UTIMER0_CMP1	0x10	Timer0 compare/capture register 1

UTIMER0_EVT	0x14	Timer0 external event selection register
UTIMER0_FLT	0x18	Timer0 filter control register
UTIMER0_IE	0x1C	Timer0 interrupt enable register
UTIMER0_IF	0x20	Timer0 interrupt flag register

The base address of Timer1 in chip is 0x4001_4100.

Table 14-5 Address Allocation of Timer1 Register

UTIMER1_CFG	0x00	Timer1 configuration register
UTIMER1_TH	0x04	Timer1 count threshold register
UTIMER1_CNT	0x08	Timer1 count value register
UTIMER1_CMP0	0x0C	Timer1 compare/capture register 0
UTIMER1_CMP1	0x10	Timer1 compare/capture register 1
UTIMER1_EVT	0x14	Timer1 external event selection register
UTIMER1_FLT	0x18	Timer1 filter control register
UTIMER1_IE	0x1C	Timer1 interrupt enable register
UTIMER1_IF	0x20	Timer1 interrupt flag register

The base address of Timer2 in chip is 0x4001_4200.

Table 14-6 Address Allocation of Timer2 Register

UTIMER2_CFG	0x00	Timer2 configuration register
UTIMER2_TH	0x04	Timer2 count threshold register
UTIMER2_CNT	0x08	Timer2 count value register
UTIMER2_CMP0	0x0C	Timer2 compare/capture register 0
UTIMER2_CMP1	0x10	Timer2 compare/capture register 1
UTIMER2_EVT	0x14	Timer2 external event selection register
UTIMER2_FLT	0x18	Timer2 filter control register
UTIMER2_IE	0x1C	Timer2 interrupt enable register
UTIMER2_IF	0x20	Timer2 interrupt flag register

The base address of Timer3 in chip is 0x4001_4300.

Table 14-7 Address Allocation of Timer3 Register

UTIMER3_CFG	0x00	Timer3 configuration register
UTIMER3_TH	0x04	Timer3 count threshold register
UTIMER3_CNT	0x08	Timer3 count value register
UTIMER3_CMP0	0x0C	Timer3 compare/capture register 0
UTIMER3_CMP1	0x10	Timer3 compare/capture register 1
UTIMER3_EVT	0x14	Timer3 external event selection register
UTIMER3_FLT	0x18	Timer3 filter control register
UTIMER3_IE	0x1C	Timer3 interrupt enable register

UTIMER3_IF	0x20	Timer3 interrupt flag register

The base address of Timer4 in chip is 0x4001_4400.

Table 14-8 Address Allocation of Timer4 Register

UTIMER4_CFG	0x00	Timer4 configuration register
UTIMER4_TH	0x04	Timer4 count threshold register
UTIMER4_CNT	0x08	Timer4 count value register
UTIMER4_CMP0	0x0C	Timer4 compare/capture register 0
UTIMER4_CMP1	0x10	Timer4 compare/capture register 1
UTIMER4_EVT	0x14	Timer4 external event selection register
UTIMER4_FLT	0x18	Timer4 filter control register
UTIMER4_IE	0x1C	Timer4 interrupt enable register
UTIMER4_IF	0x20	Timer4 interrupt flag register

Timer0/1/4 are the same.

Timer2/3 are the same, and the difference between Timer2/3 and Timer0/1/4 is that the related registers of Timer2/3 counter is 32 bits wide, while the related registers of Timer0/1 counter is 16 bits wide.

The base address of QEP0 in chip is 0x4001_4500.

Table 14-9 Address Allocation of QEP0 Register

QEP0_CFG	0x00	QEP0 configuration register
QEP0_TH	0x04	QEP0 count threshold register
QEP0_CNT	0x08	QEP0 count value register
QEP0_IE	0x0C	QEP0 interrupt enable register
QEP0_IF	0x10	QEP0 interrupt flag register

The base address of QEP1 in chip is 0x4001_4600.

Table 14-10 Address Allocation of QEP1 Register

QEP1_CFG	0x00	QEP1 configuration register
QEP1_TH	0x04	QEP1 count threshold register
QEP1_CNT	0x08	QEP1 count value register
QEP1_IE	0x0C	QEP1 interrupt enable register
QEP1_IF	0x10	QEP1 interrupt flag register

The base address of QEP2 in chip is 0x4001_4700.

Table 14-11 Address Allocation of QEP2 Register

QEP2_CFG	0x00	QEP2 configuration register
QEP2_TH	0x04	QEP2 count threshold register

QEP2_CNT	0x08	QEP2 count value register
QEP2_IE	0x0C	QEP2 interrupt enable register
QEP2_IF	0x10	QEP2 interrupt flag register

The base address of QEP3 in chip is 0x4001_4800.

Table 14-12 Address Allocation of QEP2 Register

QEP3_CFG	0x00	QEP3 configuration register
QEP3_TH	0x04	QEP3 count threshold register
QEP3_CNT	0x08	QEP3 count value register
QEP3_IE	0x0C	QEP3 interrupt enable register
QEP3_IF	0x10	QEP3 interrupt flag register

The implementation of encoder 0/1/ 2/3 is the same.

14.3.2 UTimer0 Register

14.3.2.1 Timer0 Configuration Register (UTIMER0_CFG)

Address: 0x4001_4000

Reset value: 0x0

Table 14-13 Timer0 Configuration Register (UTIMER0_CFG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN				CMP1_CLR_EN	CMP0_CLR_EN	ONE_TRIG	ETON		CLK_DIV			CLK_SRC			
RW				RW	RW	RW	RW		RW			RW			
0				0	0	0	0		0			0			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRC1				CH1_POL	CH1_MODE	CH1_FE_CAP_EN	CH1_RE_CAP_EN	SRC0				CH0_POL	CH0_MODE	CH0_FE_CAP_EN	CH0_RE_CAP_EN
RW				RW	RW	RW	RW	RW				RW	RW	RW	RW
0001				0	0	0	0	0				0	0	0	0

Bit field	Bit Name	Description
[31]	EN	Timer module enable, active high
[30:28]		Unused

[27]	CAP1_CLR_EN	When a CAP1 capture event occurs, Timer counter will be cleared, active high
[26]	CAP0_CLR_EN	When a CAP0 capture event occurs, Timer counter will be cleared, active high
[25]	ONE_TRIG	In comparison mode and when EN=0, writing 1 triggers the Timer to send a cycle pulse with a specific duty cycle, and this bit value is 1 during the pulse sending period and will be automatically cleared after one Timber cycle.
[24]	ETON	Timer counter enable configuration. The default value is 0. 0: Auto run. 1: Wait for external event to trigger counting, and stop after one cycle of counting The external event is configured by the register UTIMER0_EVT.
[23]		Unused
[22:20]	CLK_DIV	Timer counter frequency configuration. The counting frequency of the counter is the 2^{CLK_DIV} division of the main clock frequency. The default value is 0 and does not divide frequency. 3'h0: 1 frequency division 3'h1: 2 frequency division 3'h2: 4 frequency division 3 'h3: 8 frequency division 3'h4: 16 frequency division 3'h5: 32 frequency division 3'h6: 64 frequency division 3'h7: 128 frequency division
[19:16]	CLK_SRC	Timer clock source 4'h0: Internal clock of chip 4'h1: Reserved 4'h2: External clock signal of Timer0 channel 0 4'h3: External clock signal of Timer0 channel 1 4 'h4: External clock signal of Timer1 channel 0 4 'h5: External clock signal of Timer1 channel 1 4'h6: External clock signal of Timer2 channel 0 4'h7: External clock signal of Timer2 channel 1 4'h8: External clock signal of Timer3 channel 0 4'h9: External clock signal of Timer3 channel 1 4'hA: External clock signal of Timer4 channel 0 4'hB: External clock signal of Timer4 channel 1
[15:12]	SRC1	Channel 1 signal source in Timer capture mode. The default value is 3'h1. 4'h0: Timer channel 0, from chip GPIO (see Datasheet and application configuration) 4'h1: Timer channel 1, from chip GPIO (see Datasheet and

		application configuration) 4'h2: Output of comparator 0 4'h3: Output of comparator 1 4'h4: Output of comparator 2 4'h5: Output of comparator 3 4'h6: Output of comparator 4 4'h7: Output of comparator 5 4'h8: XOR of Timer channel 0 and 1
[11]	CH1_POL	Channel 1 output polarity control in compare mode: the output value when the counter count value returns to zero.
[10]	CH1_MODE	Timer channel 1 working mode selection. The default value is 0. 0: Comparison mode. Output square wave, and toggles when the channel 1 counter count value reaches 0 or the compare capture register value. 1: Capture mode. When a capture event occurs on the channel 1 input signal, the counter count value is stored in the channel 1 compare capture register.
[9]	CH1_FE_CAP_EN	Channel 1 falling edge capture event enable. 1: Enable; 0: Disable. A 1→0 transition on the Timer channel 1 input signal is considered a capture event. Falling edge event enable can coexist with rising edge event enable.
[8]	CH1_RE_CAP_EN	Channel 1 rising edge capture event enable. 1: Enable; 0: Disable. A 0→1 transition on the Timer channel 1 input signal is considered a capture event. Rising edge event enable can coexist with falling edge event enable.
[7:4]	SRC0	Channel 0 signal source in Timer capture mode. The default value is 3'h0. 4'h0: Timer channel 0, from chip GPIO (see Datasheet and application configuration) 4'h1: Timer channel 1, from chip GPIO (see Datasheet and application configuration) 4'h2: Output of comparator 0 4'h3: Output of comparator 1 4'h4: Output of comparator 2 4'h5: Output of comparator 3 4'h6: Output of comparator 4 4'h7: Output of comparator 5 4'h8: XOR of Timer channel 0 and 1
[3]	CH0_POL	Channel 0 output polarity control in compare mode: the output value when the counter count value returns to zero.
[2]	CH0_MODE	Timer channel 0 working mode selection. The default value is 0. 0: Comparison mode. Output square wave, and toggles when the channel 0 counter count value reaches 0 or the compare capture

		register value. 1: Capture mode. When a capture event occurs on the channel 0 input signal, the counter count value is stored in the channel 0 compare capture register.
[1]	CH0_FE_CAP_EN	Channel 0 falling edge capture event enable. 1: Enable; 0: Disable. A 1→0 transition on the Timer channel 0 input signal is considered a capture event. Falling edge event enable can coexist with rising edge event enable.
[0]	CH0_RE_CAP_EN	Channel 0 rising edge capture event enable. 1: Enable; 0: Disable. A 0→1 transition on the Timer channel 0 input signal is considered a capture event. Rising edge event enable can coexist with falling edge event enable.

When an external clock is used for counting, also set UTIMERx_CFG.TON=1 and the Timer clock in SYS_CLK_FEN must be enabled. UTIMERx_TH also needs to be set when using an external clock for counting.

If SRC0 is set to 4'h8 to capture the XOR value of two channels (usually two channels orthogonal to the capture encoder), and set CMP0_CLR_EN=1, CH0_FE_CAP_EN=1, CH0_RE_CAP_EN=1, and CH0_MODE=1, both rising edge and falling edge will be captured and the counter will be cleared each time with a signal edge. CMP0 is the calculated value between the two captured signal edges.

14.3.2.2 Timer0 Threshold Register (UTIMER0_TH)

Address: 0x4001_4004

Reset value: 0x0

Table 14-14 Timer0 Threshold Register (UTIMER0_TH)

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

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	TH	Timer0 counter count threshold. The counter counts from 0 to TH, and then returns to 0 to start counting.

14.3.2.3 Timer0 Count Register (UTIMER0_CNT)

Address: 0x4001_4008

Reset value: 0x0

Table 14-15 Timer0 Count Register (UTIMER0_CNT)

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

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	CNT	The current count value of the Timer0 counter. New count value can be written.

Note: The Timer clock needs to be started by SYS_CLK_FEN before writing to UTIMER0_CNT.

14.3.2.4 Timer0 Channel 0 Compare Capture Register (UTIMER0_CMP0)

Address: 0x4001_400C

Reset value: 0x0

Table 14-16 Timer0 Channel 0 Compare Capture Register (UTIMER0_CMP0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP0															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	CMP0	When Timer channel 0 works in compare mode and the counter count value is equal to CMP0, a comparison event occurs. When Timer channel 0 works in the capture mode, the counter count value when the capture event occurs is stored in the CMP0 register.

If setting CMP0=0, channel 0 can always be 1; if setting CMP0=TH+1, channel 0 can always be 0.

14.3.2.5 Timer0 Channel 1 Compare Capture Register (UTIMER0_CMP1)

Address: 0x4001_4010

Reset value: 0x0

Table 14-17 Timer0 Channel 1 Compare Capture Register (UTIMER0_CMP1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP1															
RW															

0

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	CMP1	When Timer channel 0 works in compare mode and the counter count value is equal to CMP1, a comparison event occurs. When Timer channel 0 works in the capture mode, the counter count value when the capture event occurs is stored in the CMP1 register.

If setting CMP1=0, channel 1 can always be 1; if setting CMP1=TH+1, channel 1 can always be 0.

14.3.2.6 Timer0 External Event Select Register (UTIMER0_EVT)

Address: 0x4001_4014

Reset value: 0x0

Table 14-18 Timer0 External Event Select Register (UTIMER0_EVT)

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

Bit field	Bit Name	Description
[31:5]		Unused
[4:0]	EVT_SRC	Timer external event selection register. This register should be used with UTIMER0_CFG.ETON. When ETON is high, select the event that triggers Timer0 count by this register. Note that the comparison event of Timer cannot trigger Timer for counting. 0: TIMER0 channel 0 comparison event 1: TIMER0 channel 1 comparison event 2: TIMER1 channel 0 comparison event 3: TIMER1 channel 1 comparison event 4: TIMER2 channel 0 comparison event 5: TIMER2 channel 1 comparison event 6: TIMER3 channel 0 comparison event 7: TIMER3 channel 1 comparison event 8: TIMER4 channel 0 comparison event 9: TIMER4 channel 1 comparison event 10: MCPWM0 TADC[0] comparison event 11: MCPWM0 TADC[1] comparison event 12: MCPWM0 TADC[2] comparison event 13: MCPWM0 TADC[3] comparison event

		14: MCPWM1 TADC[0] comparison event 15: MCPWM1 TADC[1] comparison event 16: MCPWM1 TADC[2] comparison event 17: MCPWM1 TADC[3] comparison event

14.3.2.7 Timer0 Filter Control Register (UTIMER0_FLT)

Address: 0x4001_4018

Reset value: 0x0

Table 14-19 Timer0 Filter Control Register (UTIMER0_FLT)

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

Bit field	Bit Name	Description
[31:8]		Unused
[7:0]	FLT	Filter width of channel 0/1, with value range of 0-255. If FLT is 0, the channel will not be filtered. If FLT is not 0, the channel will be filtered, with the filter width of FLTx8. When the channel signal level is stable beyond FLT×8 system clock cycle width, the filter output is updated; otherwise, the filter leaves the current output unchanged.

Note that the working clock of the above filter is same as that of the corresponding Timer after frequency division, which is controlled by the frequency division coefficient of UTIMERx_CFG.CLK_DIV.

If UTIMERx_FLT.FLT = 0x6, UTIMERx_CFG.CLK_DIV = 0x2, the running clock of the Timer is divided by 4 relative to the system clock. The channel input signal needs to be filtered by 8×6 times of Timer's operating clock, i.e., 8×6x4 system clock.

14.3.2.8 Timer0 Interrupt Enable Register (UTIMER0_IE)

Address: 0x4001_401C

Reset value: 0x0

Table 14-20 Timer0 Interrupt Enable Register (UTIMER0_IE)

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

	ZC_RE	CH1_RE	CH0_RE		ZC_IE	CH1_IE	CH0_IE
	RW	RW	RW		RW	RW	RW
	0	0	0		0	0	0

Bit field	Bit Name	Description
[31:11]		Unused
[10]	ZC_RE	Timer counter over-zero DMA request enable, active high.
[9]	CH1_RE	Timer channel 1 comparison/capture request enable, active high.
[8]	CH0_RE	Timer channel 0 comparison/capture request enable, active high.
[7:3]		Unused
[2]	ZC_IE	Timer counter over-zero interrupt enable, active high.
[1]	CH1_IE	Timer channel 1 comparison/capture interrupt enable, active high.
[0]	CH0_IE	Timer channel 0 comparison/capture interrupt enable, active high.

DMA request event is the same event flag as interrupt event. After the DMA request is processed, the interrupt flag is automatically cleared by the DMA hardware without CPU software intervention. Note that typically, if a DMA request is turned on, the corresponding interrupt will be disabled.

14.3.2.9 Timer0 Interrupt Flag Register (UTIMER0_IF)

Address: 0x4001_4020

Reset value: 0x0

Table 14-21 Timer0 Interrupt Flag Register (UTIMER0_IF)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													ZC_IF	CH1_IF	CH0_IF
													RW1C	RW1C	RW1C
													0	0	0

Bit field	Bit Name	Description
[31:3]		Unused
[2]	ZC_IF	Timer counter over-zero interrupt flag. Active high, write 1 to clear.
[1]	CH1_IF	Timer channel 1 comparison/capture interrupt flag. Active high,

		write 1 to clear.
[0]	CH0_IF	Timer channel 0 comparison/capture interrupt flag. Active high, write 1 to clear.

Write 1 to interrupt flag to clear. Generally, |= method is not recommended for clear, because |= will read the interrupt flag first and then change the bit to 1 to clear. If there are other interrupt flag bits, all will be cleared, which is not what the software expects. For example, the following expression it to clear ZC_IF, but if CH0_IF is set to 1 before writing, the software will first read the UTIMER_IF value of 0x24, and then execute 0x4|0x1=0x5 before writing, resulting in clear of both CH0_IF and ZC_IF, which may cause the Timer to enter one less interrupt due to capture.

```
UTIMER_IF|=0x4;
```

To clear ZC_IF bit, directly write 1 to BIT2, as follows.

```
UTIMER_IF=0x4;
```

14.3.3 UTimer1 Register

14.3.3.1 Timer1 Configuration Register (UTIMER1_CFG)

Address: 0x4001_4100

Reset value: 0x0

Table 14-22 Timer1 Configuration Register (UTIMER1_CFG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN				CAP1_CLR_EN	CAP0_CLR_EN	ONE_TRIG	ETON		CLK_DIV			CLK_SRC			
RW				RW	RW	RW	RW		RW			RW			
0				0	0	0	0		0			0			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRC1				CH1_POL	CH1_MODE	CH1_FE_CAP_EN	CH1_RE_CAP_EN	SRC0				CH0_POL	CH0_MODE	CH0_FE_CAP_EN	CH0_RE_CAP_EN
RW				RW	RW	RW	RW	RW				RW	RW	RW	RW
0001				0	0	0	0	0				0	0	0	0

Bit field	Bit Name	Description
[31]	EN	Timer module enable, active high
[30:28]		Unused
[27]	CAP1_CLR_EN	When a CAP1 capture event occurs, Timer counter will be cleared, active high

[26]	CAP0_CLR_EN	When a CAP0 capture event occurs, Timer counter will be cleared, active high
[25]	ONE_TRIG	In comparison mode and when EN=0, writing 1 triggers the Timer to send a cycle pulse with a specific duty cycle, and this bit value is 1 during the pulse sending period and will be automatically cleared after one Timber cycle.
[24]	ETON	Timer counter enable configuration. The default value is 0. 0: Auto run. 1: Wait for external event to trigger counting, and stop after one cycle of counting The external event is configured by the register UTIMER1_EVT.
[23]		Unused
[22:20]	CLK_DIV	Timer counter frequency configuration. The counting frequency of the counter is the 2^{CLK_DIV} division of the main clock frequency. The default value is 0 and does not divide frequency. 3'h0: 1 frequency division 3'h1: 2 frequency division 3'h2: 4 frequency division 3 'h3: 8 frequency division 3'h4: 16 frequency division 3'h5: 32 frequency division 3'h6: 64 frequency division 3'h7: 128 frequency division
[19:16]	CLK_SRC	Timer clock source 4'h0: Internal clock of chip 4'h1: Reserved 4'h2: External clock signal of Timer0 channel 0 4'h3: External clock signal of Timer0 channel 1 4 'h4: External clock signal of Timer1 channel 0 4 'h5: External clock signal of Timer1 channel 1 4'h6: External clock signal of Timer2 channel 0 4'h7: External clock signal of Timer2 channel 1 4'h8: External clock signal of Timer3 channel 0 4'h9: External clock signal of Timer3 channel 1 4'hA: External clock signal of Timer4 channel 0 4'hB: External clock signal of Timer4 channel 1
[15:12]	SRC1	Channel 1 signal source in Timer capture mode. The default value is 3'h1. 4'h0: Timer channel 0, from chip GPIO (see Datasheet and application configuration) 4'h1: Timer channel 1, from chip GPIO (see Datasheet and application configuration) 4'h2: Output of comparator 0

		4'h3: Output of comparator 1 4'h4: Output of comparator 2 4'h5: Output of comparator 3 4'h6: Output of comparator 4 4'h7: Output of comparator 5 4'h8: XOR of Timer channel 0 and 1
[11]	CH1_POL	Channel 1 output polarity control in compare mode: the output value when the counter count value returns to zero.
[10]	CH1_MODE	Timer channel 1 working mode selection. The default value is 0. 0: Comparison mode. Output square wave, and toggles when the channel 1 counter count value reaches 0 or the compare capture register value. 1: Capture mode. When a capture event occurs on the channel 1 input signal, the counter count value is stored in the channel 1 compare capture register.
[9]	CH1_FE_CAP_EN	Channel 1 falling edge capture event enable. 1: Enable; 0: Disable. A 1→0 transition on the Timer channel 1 input signal is considered a capture event. Falling edge event enable can coexist with rising edge event enable.
[8]	CH1_RE_CAP_EN	Channel 1 rising edge capture event enable. 1: Enable; 0: Disable. A 0→1 transition on the channel 1 input signal is considered a capture event. Rising edge event enable can coexist with falling edge event enable.
[7:4]	SRC0	Channel 0 signal source in Timer capture mode. The default value is 3'h0. 4'h0: Timer channel 0, from chip GPIO (see Datasheet and application configuration) 4'h1: Timer channel 1, from chip GPIO (see Datasheet and application configuration) 4'h2: Output of comparator 0 4'h3: Output of comparator 1 4'h4: Output of comparator 2 4'h5: Output of comparator 3 4'h6: Output of comparator 4 4'h7: Output of comparator 5 4'h8: XOR of Timer channel 0 and 1
[3]	CH0_POL	Channel 0 output polarity control in compare mode: the output value when the counter count value returns to zero.
[2]	CH0_MODE	Timer channel 0 working mode selection. The default value is 0. 0: Comparison mode. Output square wave, and toggles when the channel 0 counter count value reaches 0 or the compare capture register value. 1: Capture mode. When a capture event occurs on the channel 0

		input signal, the counter count value is stored in the channel 0 compare capture register.
[1]	CH0_FE_CAP_EN	Channel 0 falling edge capture event enable. 1: Enable; 0: Disable. A 1→0 transition on the channel 0 input signal is considered a capture event. Falling edge event enable can coexist with rising edge event enable.
[0]	CH0_RE_CAP_EN	Channel 0 rising edge capture event enable. 1: Enable; 0: Disable. A 0→1 transition on the channel 0 input signal is considered a capture event. Rising edge event enable can coexist with falling edge event enable.

14.3.3.2 Timer1 Threshold Register (UTIMER1_TH)

Address: 0x4001_4104

Reset value: 0x0

Table 14-23 Timer1 Threshold Register (UTIMER1_TH)

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

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	TH	Timer0 counter count threshold. The counter counts from 0 to TH, and then returns to 0 to start counting.

14.3.3.3 Timer1 Count Register (UTIMER1_CNT)

Address: 0x4001_4108

Reset value: 0x0

Table 14-24 Timer1 Count Register (UTIMER1_CNT)

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

Bit field	Bit Name	Description
[31:16]		Unused

[15:0]	CNT	The current count value of the Timer0 counter. New count value can be written.

Note: The Timer clock needs to be started by SYS_CLK_FEN before writing to UTIMER1_CNT.

14.3.3.4 Timer1 Channel 0 Compare Capture Register (UTIMER1_CMP0)

Address: 0x4001_410C

Reset value: 0x0

Table 14-25 Timer1 Channel 0 Compare Capture Register (UTIMER1_CMP0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP0															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	CMP0	When Timer channel 0 works in compare mode and the counter count value is equal to CMP0, a comparison event occurs. When Timer channel 0 works in the capture mode, the counter count value when the capture event occurs is stored in the CMP0 register.

14.3.3.5 Timer1 Channel 1 Compare Capture Register (UTIMER1_CMP1)

Address: 0x4001_4110

Reset value: 0x0

Table 14-26 Timer1 Channel 1 Compare Capture Register (UTIMER1_CMP1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP1															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	CMP1	When Timer channel 0 works in compare mode and the counter count value is equal to CMP1, a comparison event occurs. When Timer channel 0 works in the capture mode, the counter count value when the capture event occurs is stored in the CMP1 register.

14.3.3.6 Timer1 External Event Select Register (UTIMER1_EVT)

Address: 0x4001_4114

Reset value: 0x0

Table 14-27 Timer1 External Event Select Register (UTIMER1_EVT)

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

Bit field	Bit Name	Description
[31:4]		Unused
[3:0]	EVT_SRC	Timer1 external event selection register. This register should be used with UTIMER1_CFG.ETON. When ETON is high, select the event that triggers Timer0 count by this register. Note that the comparison event of Timer cannot trigger Timer for counting. 0: TIMER0 channel 0 comparison event 1: TIMER0 channel 1 comparison event 2: TIMER1 channel 0 comparison event 3: TIMER1 channel 1 comparison event 4: TIMER2 channel 0 comparison event 5: TIMER2 channel 1 comparison event 6: TIMER3 channel 0 comparison event 7: TIMER3 channel 1 comparison event 8: TIMER4 channel 0 comparison event 9: TIMER4 channel 1 comparison event 10: MCPWM0 TADC[0] comparison event 11: MCPWM0 TADC[1] comparison event 12: MCPWM0 TADC[2] comparison event 13: MCPWM0 TADC[3] comparison event 14: MCPWM1 TADC[0] comparison event 15: MCPWM1 TADC[1] comparison event 16: MCPWM1 TADC[2] comparison event 17: MCPWM1 TADC[3] comparison event

14.3.3.7 Timer1 Filter Control Register (UTIMER1_FLT)

Address: 0x4001_4118

Reset value: 0x0

Table 14-28 Timer1 Filter Control Register (UTIMER1_FLT)

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

Bit field	Bit Name	Description
[31:8]		Unused
[7:0]	FLT	Filter width of channel 0/1, with value range of 0-255. If FLT is 0, the channel will not be filtered. If FLT is not 0, the channel will be filtered, with the filter width of FLTx8. When the channel signal level is stable beyond FLT×8 system clock cycle width, the filter output is updated; otherwise, the filter leaves the current output unchanged.

Note that the working clock of the above filter is same as that of the corresponding Timer after frequency division, which is controlled by the frequency division coefficient of UTIMERx_CFG.CLK_DIV.

If UTIMERx_FLT.FLT = 0x6, UTIMERx_CFG.CLK_DIV = 0x2, the running clock of the Timer is divided by 4 relative to the system clock. The channel input signal needs to be filtered by 8×6 times of Timer's operating clock, i.e., 8×6x4 system clock.

14.3.3.8 Timer1 Interrupt Enable Register (UTIMER1_IE)

Address: 0x4001_411C

Reset value: 0x0

Table 14-29 Timer1 Interrupt Enable Register (UTIMER1_IE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					ZC_RE	CH1_RE	CH0_RE						ZC_IE	CH1_IE	CH0_IE
					RW	RW	RW						RW	RW	RW
					0	0	0						0	0	0

Bit field	Bit Name	Description
[31:11]		Unused
[10]	ZC_RE	Timer counter over-zero DMA request enable, active high.
[9]	CH1_RE	Timer channel 1 comparison/capture request enable, active high.
[8]	CH0_RE	Timer channel 0 comparison/capture request enable, active high.

[7:3]		Unused
[2]	ZC_IE	Timer counter over-zero interrupt enable, active high.
[1]	CH1_IE	Timer channel 1 comparison/capture interrupt enable, active high.
[0]	CH0_IE	Timer channel 0 comparison/capture interrupt enable, active high.

DMA request event is the same event flag as interrupt event. After the DMA request is processed, the interrupt flag is automatically cleared by the DMA hardware without CPU software intervention. Note that typically, if a DMA request is turned on, the corresponding interrupt will be disabled.

14.3.3.9 Timer1 Interrupt Flag Register (UTIMER1_IF)

Address: 0x4001_4120

Reset value: 0x0

Table 14-30 Timer1 Interrupt Flag Register (UTIMER1_IF)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													ZC_IF	CH1_IF	CH0_IF
													RW1C	RW1C	RW1C
													0	0	0

Bit field	Bit Name	Description
[31:3]		Unused
[2]	ZC_IF	Timer counter over-zero interrupt flag. Active high, write 1 to clear.
[1]	CH1_IF	Timer channel 1 comparison/capture interrupt flag. Active high, write 1 to clear.
[0]	CH0_IF	Timer channel 0 comparison/capture interrupt flag. Active high, write 1 to clear.

14.3.4 UTimer2 Register

14.3.4.1 Timer2 Configuration Register (UTIMER2_CFG)

Address: 0x4001_4200

Reset value: 0x0

Table 14-31 Timer2 Configuration Register (UTIMER2_CFG)

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

EN		CAP1_CLR_EN	CAP0_CLR_EN	ONE_TRIG	ETON		CLK_DIV	CLK_SRC
RW		RW	RW	RW	RW		RW	RW
0		0	0	0	0		0	0

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

SRC1				CH1_POL	CH1_MODE	CH1_FE_CAP_EN	CH1_RE_CAP_EN	SRC0				CH0_POL	CH0_MODE	CH0_FE_CAP_EN	CH0_RE_CAP_EN
RW				RW	RW	RW	RW	RW				RW	RW	RW	RW
0001				0	0	0	0	0				0	0	0	0

Bit field	Bit Name	Description
[31]	EN	Timer module enable, active high
[30:28]		Unused
[27]	CAP1_CLR_EN	When a CAP1 capture event occurs, Timer counter will be cleared, active high
[26]	CAP0_CLR_EN	When a CAP0 capture event occurs, Timer counter will be cleared, active high
[25]	ONE_TRIG	In comparison mode and when EN=0, writing 1 triggers the Timer to send a cycle pulse with a specific duty cycle, and this bit value is 1 during the pulse sending period and will be automatically cleared after one Timber cycle.
[24]	ETON	Timer counter enable configuration. The default value is 0. 0: Auto run. 1: Wait for external event to trigger counting, and stop after one cycle of counting The external event is configured by the register UTIMER2_EVT.
[23]		Unused
[22:20]	CLK_DIV	Timer counter frequency configuration. The counting frequency of the counter is the 2^{CLK_DIV} division of the main clock frequency. The default value is 0 and does not divide frequency. 3'h0: 1 frequency division 3'h1: 2 frequency division 3'h2: 4 frequency division 3 'h3: 8 frequency division 3'h4: 16 frequency division 3'h5: 32 frequency division 3'h6: 64 frequency division 3'h7: 128 frequency division

[19:16]	CLK_SRC	Timer clock source 4'h0: Internal clock of chip 4'h1: Reserved 4'h2: External clock signal of Timer0 channel 0 4'h3: External clock signal of Timer0 channel 1 4 'h4: External clock signal of Timer1 channel 0 4'h5: External clock signal of Timer1 channel 1 4'h6: External clock signal of Timer2 channel 0 4'h7: External clock signal of Timer2 channel 1 4'h8: External clock signal of Timer3 channel 0 4'h9: External clock signal of Timer3 channel 1 4'hA: External clock signal of Timer4 channel 0 4'hB: External clock signal of Timer4 channel 1
[15:12]	SRC1	Channel 1 signal source in Timer capture mode. The default value is 3'h1. 4'h0: Timer channel 0, from chip GPIO (see Datasheet and application configuration) 4'h1: Timer channel 1, from chip GPIO (see Datasheet and application configuration) 4'h2: Output of comparator 0 4'h3: Output of comparator 1 4'h4: Output of comparator 2 4'h5: Output of comparator 3 4'h6: Output of comparator 4 4'h7: Output of comparator 5 4'h8: XOR of Timer channel 0 and 1
[11]	CH1_POL	Timer channel 1 output polarity control in compare mode: the output value when the counter count value returns to zero.
[10]	CH1_MODE	Timer channel 1 working mode selection. The default value is 0. 0: Comparison mode. Output square wave, and toggles when the channel 1 counter count value reaches 0 or the compare capture register value. 1: Capture mode. When a capture event occurs on the Timer channel 1 input signal, the counter count value is stored in the channel 1 compare capture register.
[9]	CH1_FE_CAP_EN	Timer channel 1 falling edge capture event enable. 1: Enable; 0: Disable. A 10 transition on the Timer channel 1 input signal is considered a capture event. Falling edge event enable can coexist with rising edge event enable.
[8]	CH1_RE_CAP_EN	Timer channel 1 rising edge capture event enable. 1: Enable; 0: Disable. A 0→1 transition on the Timer channel 1 input signal is considered

		a capture event. Rising edge event enable can coexist with falling edge event enable.
[7:4]	SRC0	Timer channel 0 signal source in Timer capture mode. The default value is 3'h0. 4'h0: Timer channel 0, from chip GPIO (see Datasheet and application configuration) 4'h1: Timer channel 1, from chip GPIO (see Datasheet and application configuration) 4'h2: Output of comparator 0 4'h3: Output of comparator 1 4'h4: Output of comparator 2 4'h5: Output of comparator 3 4'h6: Output of comparator 4 4'h7: Output of comparator 5 4'h8: XOR of Timer channel 0 and 1
[3]	CH0_POL	Channel 0 output polarity control in compare mode: the output value when the counter count value returns to zero.
[2]	CH0_MODE	Timer channel 0 working mode selection. The default value is 0. 0: Comparison mode. Output square wave, and toggles when the channel 0 counter count value reaches 0 or the compare capture register value. 1: Capture mode. When a capture event occurs on the channel 0 input signal, the counter count value is stored in the channel 0 compare capture register.
[1]	CH0_FE_CAP_EN	Channel 0 falling edge capture event enable. 1: Enable; 0: Disable. A 1→0 transition on the Timer channel 0 input signal is considered a capture event. Falling edge event enable can coexist with rising edge event enable.
[0]	CH0_RE_CAP_EN	Channel 0 rising edge capture event enable. 1: Enable; 0: Disable. A 0→1 transition on the Timer channel 0 input signal is considered a capture event. Rising edge event enable can coexist with falling edge event enable.

14.3.4.2 Timer2 Threshold Register (UTIMER2_TH)

Address: 0x4001_4204

Reset value: 0x0

Table 14-32 Timer2 Threshold Register (UTIMER2_TH)

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

TH
RW
0

Bit field	Bit Name	Description
[31:0]	TH	Timer0 counter count threshold. The counter counts from 0 to TH, and then returns to 0 to start counting.

14.3.4.3 Timer2 Count Register (UTIMER2_CNT)

Address: 0x4001_4208

Reset value: 0x0

Table 14-33 Timer2 Count Register (UTIMER2_CNT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNT															
RW															
0															

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

Bit field	Bit Name	Description
[31:0]	CNT	The current count value of the Timer2 counter. New count value can be written.

Note: The Timer clock needs to be started by SYS_CLK_FEN before writing to UTIMER2_CNT.

14.3.4.4 Timer2 Channel 0 Compare Capture Register (UTIMER2_CMP0)

Address: 0x4001_420C

Reset value: 0x0

Table 14-34 Timer2 Channel 0 Compare Capture Register (UTIMER2_CMP0)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CMP0															
RW															
0															

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

CMP0
RW
0

Bit field	Bit Name	Description
[31:0]	CMP0	When Timer channel 0 works in compare mode and the counter count value is equal to CMP0, a comparison event occurs. When Timer channel 0 works in the capture mode, the counter count value when the capture event occurs is stored in the CMP0 register.

14.3.4.5 Timer2 Channel 1 Compare Capture Register (UTIMER2_CMP1)

Address: 0x4001_4210

Reset value: 0x0

Table 14-35 Timer2 Channel 1 Compare Capture Register (UTIMER2_CMP1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CMP1															
RW															
0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP1															
RW															
0															

Bit field	Bit Name	Description
[31:0]	CMP1	When Timer channel 0 works in compare mode and the counter count value is equal to CMP1, a comparison event occurs. When Timer channel 0 works in the capture mode, the counter count value when the capture event occurs is stored in the CMP1 register.

14.3.4.6 Timer2 External Event Select Register (UTIMER2_EVT)

Address: 0x4001_4214

Reset value: 0x0

Table 14-36 Timer2 External Event Select Register (UTIMER2_EVT)

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

	0

Bit field	Bit Name	Description
[31:3]		Unused
[2:0]	EVT_SRC	Timer external event selection register. This register should be used with UTIMER2_CFG.ETON. When ETON is high, select the event that triggers Timer0 count by this register. Note that the comparison event of Timer cannot trigger Timer for counting. 0: TIMER0 channel 0 comparison event 1: TIMER0 channel 1 comparison event 2: TIMER1 channel 0 comparison event 3: TIMER1 channel 1 comparison event 4: TIMER2 channel 0 comparison event 5: TIMER2 channel 1 comparison event 6: TIMER3 channel 0 comparison event 7: TIMER3 channel 1 comparison event 8: TIMER4 channel 0 comparison event 9: TIMER4 channel 1 comparison event 10: MCPWM0 TADC[0] comparison event 11: MCPWM0 TADC[1] comparison event 12: MCPWM0 TADC[2] comparison event 13: MCPWM0 TADC[3] comparison event 14: MCPWM1 TADC[0] comparison event 15: MCPWM1 TADC[1] comparison event 16: MCPWM1 TADC[2] comparison event 17: MCPWM1 TADC[3] comparison event

14.3.4.7 Timer2 Filter Control Register (UTIMER2_FLT)

Address: 0x4001_4218

Reset value: 0x0

Table 14-37 Timer2 Filter Control Register (UTIMER2_FLT)

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

Bit field	Bit Name	Description
[31:8]		Unused

[7:0]	FLT	Filter width of channel 0/1, with value range of 0-255. If FLT is 0, the channel will not be filtered. If FLT is not 0, the channel will be filtered, with the filter width of FLTx8. When the channel signal level is stable beyond FLT×8 system clock cycle width, the filter output is updated; otherwise, the filter leaves the current output unchanged.

Note that the working clock of the above filter is same as that of the corresponding Timer after frequency division, which is controlled by the frequency division coefficient of UTIMERx_CFG.CLK_DIV.

If UTIMERx_FLT.FLT = 0x6, UTIMERx_CFG.CLK_DIV = 0x2, the running clock of the Timer is divided by 4 relative to the system clock. The channel input signal needs to be filtered by 8×6 times of Timer's operating clock, i.e., 8×6x4 system clock.

14.3.4.8 Timer2 Interrupt Enable Register (UTIMER2_IE)

Address: 0x4001_421C

Reset value: 0x0

Table 14-38 Timer2 Interrupt Enable Register (UTIMER2_IE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					ZC_RE	CH1_RE	CH0_RE						ZC_IE	CH1_IE	CH0_IE
					RW	RW	RW						RW	RW	RW
					0	0	0						0	0	0

Bit field	Bit Name	Description
[31:11]		Unused
[10]	ZC_RE	Timer counter over-zero DMA request enable, active high.
[9]	CH1_RE	Timer channel 1 comparison/capture request enable, active high.
[8]	CH0_RE	Timer channel 0 comparison/capture request enable, active high.
[7:3]		Unused
[2]	ZC_IE	Timer counter over-zero interrupt enable, active high.
[1]	CH1_IE	Timer channel 1 comparison/capture interrupt enable, active high.
[0]	CH0_IE	Timer channel 0 comparison/capture interrupt enable, active high.

DMA request event is the same event flag as interrupt event. After the DMA request is processed, the interrupt flag is automatically cleared by the DMA hardware without CPU software intervention. Note that typically, if a DMA request is turned on, the corresponding interrupt will be disabled.

14.3.4.9 Timer2 Interrupt Flag Register (UTIMER2_IF)

Address: 0x4001_4220

Reset value: 0x0

Table 14-39 Timer2 Interrupt Flag Register (UTIMER2_IF)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													ZC_IF	CH1_IF	CH0_IF
													RW1C	RW1C	RW1C
													0	0	0

Bit field	Bit Name	Description
[31:3]		Unused
[2]	ZC_IF	Timer counter over-zero interrupt flag. Active high, write 1 to clear.
[1]	CH1_IF	Timer channel 1 comparison/capture interrupt flag. Active high, write 1 to clear.
[0]	CH0_IF	Timer channel 0 comparison/capture interrupt flag. Active high, write 1 to clear.

14.3.5 UTimer3 Register

14.3.5.1 Timer3 Configuration Register (UTIMER3_CFG)

Address: 0x4001_4300

Reset value: 0x0

Table 14-40 Timer3 Configuration Register (UTIMER3_CFG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN				CAP1_CLR_EN	CAP0_CLR_EN	ONE_TRIG	ETON		CLK_DIV			CLK_SRC			
RW				RW	RW	RW	RW		RW			RW			
0				0	0	0	0		0			0			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SRC1	CH1_POL	CH1_MODE	CH1_FE_CAP_EN	CH1_RE_CAP_EN	SRC0	CH0_POL	CH0_MODE	CH0_FE_CAP_EN	CH0_RE_CAP_EN
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0001	0	0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31]	EN	Timer module enable, active high
[30:26]		Unused
[25]	CAP1_CLR_EN	When a CAP1 capture event occurs, Timer counter will be cleared, active high
[24]	CAP0_CLR_EN	When a CAP0 capture event occurs, Timer counter will be cleared, active high
[23]	ONE_TRIG	In comparison mode and when EN=0, writing 1 triggers the Timer to send a cycle pulse with a specific duty cycle, and this bit value is 1 during the pulse sending period and will be automatically cleared after one Timber cycle.
[22]	ETON	Timer counter enable configuration. The default value is 0. 0: Auto run. 1: Wait for external event to trigger counting, and stop after one cycle of counting The external event is configured by the register UTIMER3_EVT.
[21:18]	CLK_SRC	Timer clock source 4'h0: Internal clock of chip 4'h1: Reserved 4'h2: External clock signal of Timer0 channel 0 4'h3: External clock signal of Timer0 channel 1 4 'h4: External clock signal of Timer1 channel 0 4 'h5: External clock signal of Timer1 channel 1 4'h6: External clock signal of Timer2 channel 0 4'h7: External clock signal of Timer2 channel 1 4'h8: External clock signal of Timer3 channel 0 4'h9: External clock signal of Timer3 channel 1 4'hA: External clock signal of Timer4 channel 0 4'hB: External clock signal of Timer4 channel 1
[17:16]	CLK_DIV	Timer counter frequency configuration. The counting frequency of the counter is the 2^{CLK_DIV} division of the main clock frequency. The default value is 0 and does not divide frequency. 3'h0: 1 frequency division 3'h1: 2 frequency division 3'h2: 4 frequency division

		3 ’h3: 8 frequency division 3’h4: 16 frequency division 3’h5: 32 frequency division 3’h6: 64 frequency division 3’h7: 128 frequency division
[15:12]	SRC1	Channel 1 signal source in Timer capture mode. The default value is 3’h1. 4’h0: Timer channel 0, from chip GPIO (see Datasheet and application configuration) 4’h1: Timer channel 1, from chip GPIO (see Datasheet and application configuration) 4’h2: Output of comparator 0 4’h3: Output of comparator 1 4’h4: Output of comparator 2 4’h5: Output of comparator 3 4’h6: Output of comparator 4 4’h7: Output of comparator 5 4’h8: XOR of Timer channel 0 and 1
[11]	CH1_POL	Channel 1 output polarity control in compare mode: the output value when the counter count value returns to zero.
[10]	CH1_MODE	Timer channel 1 working mode selection. The default value is 0. 0: Comparison mode. Output square wave, and toggles when the channel 1 counter count value reaches 0 or the compare capture register value. 1: Capture mode. When a capture event occurs on the channel 1 input signal, the counter count value is stored in the channel 1 compare capture register.
[9]	CH1_FE_CAP_EN	Channel 1 falling edge capture event enable. 1: Enable; 0: Disable. A 1→0 transition on the Timer channel 1 input signal is considered a capture event. Falling edge event enable can coexist with rising edge event enable.
[8]	CH1_RE_CAP_EN	Channel 1 rising edge capture event enable. 1: Enable; 0: Disable. A 0→1 transition on the Timer channel 1 input signal is considered a capture event. Rising edge event enable can coexist with falling edge event enable.
[7:4]	SRC0	Channel 0 signal source in Timer capture mode. The default value is 3’h0. 4’h0: Timer channel 0, from chip GPIO (see Datasheet and application configuration) 4’h1: Timer channel 1, from chip GPIO (see Datasheet and application configuration) 4’h2: Output of comparator 0 4’h3: Output of comparator 1

		4'h4: Output of comparator 2 4'h5: Output of comparator 3 4'h6: Output of comparator 4 4'h7: Output of comparator 5 4'h8: XOR of Timer channel 0 and 1
[3]	CH0_POL	Channel 0 output polarity control in compare mode: the output value when the counter count value returns to zero.
[2]	CH0_MODE	Timer channel 0 working mode selection. The default value is 0. 0: Comparison mode. Output square wave, and toggles when the channel 0 counter count value reaches 0 or the compare capture register value. 1: Capture mode. When a capture event occurs on the channel 0 input signal, the counter count value is stored in the channel 0 compare capture register.
[1]	CH0_FE_CAP_EN	Channel 0 falling edge capture event enable. 1: Enable; 0: Disable. A 1→0 transition on the Timer channel 0 input signal is considered a capture event. Falling edge event enable can coexist with rising edge event enable.
[0]	CH0_RE_CAP_EN	Channel 0 rising edge capture event enable. 1: Enable; 0: Disable. A 0→1 transition on the Timer channel 0 input signal is considered a capture event. Rising edge event enable can coexist with falling edge event enable.

14.3.5.2 Timer3 Threshold Register (UTIMER3_TH)

Address: 0x4001_4304

Reset value: 0x0

Table 14-41 Timer3 Threshold Register (UTIMER3_TH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TH															
RW															
0															

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

Bit field	Bit Name	Description
[31:0]	TH	Timer0 counter count threshold. The counter counts from 0 to TH, and then returns to 0 to start counting.

14.3.5.3 Timer3 Count Register (UTIMER3_CNT)

Address: 0x4001_4308

Reset value: 0x0

Table 14-42 Timer3 Count Register (UTIMER3_CNT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNT															
RW															
0															

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

Bit field	Bit Name	Description
[31:0]	CNT	The current count value of the Timer0 counter. New count value can be written.

Note: The Timer clock needs to be started by SYS_CLK_FEN before writing to UTIMER3_CNT.

14.3.5.4 Timer3 Channel 0 Compare Capture Register (UTIMER3_CMP0)

Address: 0x4001_430C

Reset value: 0x0

Table 14-43 Timer3 Channel 0 Compare Capture Register (UTIMER3_CMP0)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CMP0															
RW															
0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP0															
RW															
0															

Bit field	Bit Name	Description
[31:0]	CMP0	When Timer channel 0 works in compare mode and the counter count value is equal to CMP0, a comparison event occurs. When Timer channel 0 works in the capture mode, the counter count value when the capture event occurs is stored in the CMP0 register.

14.3.5.5 Timer3 Channel 1 Compare Capture Register (UTIMER3_CMP1)

Address: 0x4001_4310

Reset value: 0x0

Table 14-44 Timer3 Channel 1 Compare Capture Register (UTIMER3_CMP1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CMP1															
RW															
0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP1															
RW															
0															

Bit field	Bit Name	Description
[31:0]	CMP1	When Timer channel 0 works in compare mode and the counter count value is equal to CMP1, a comparison event occurs. When Timer channel 0 works in the capture mode, the counter count value when the capture event occurs is stored in the CMP1 register.

14.3.5.6 Timer3 External Event Select Register (UTIMER3_EVT)

Address: 0x4001_4314

Reset value: 0x0

Table 14-45 Timer3 External Event Select Register (UTIMER3_EVT)

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

Bit field	Bit Name	Description
[31:3]		Unused
[2:0]	EVT_SRC	Timer external event selection register. This register should be used with UTIMER3_CFG.ETON. When ETON is high, select the event that triggers Timer0 count by this register. Note that the comparison event of Timer cannot trigger Timer for counting.

		0: TIMER0 channel 0 comparison event 1: TIMER0 channel 1 comparison event 2: TIMER1 channel 0 comparison event 3: TIMER1 channel 1 comparison event 4: TIMER2 channel 0 comparison event 5: TIMER2 channel 1 comparison event 6: TIMER3 channel 0 comparison event 7: TIMER3 channel 1 comparison event 8: TIMER4 channel 0 comparison event 9: TIMER4 channel 1 comparison event 10: MCPWM0 TADC[0] comparison event 11: MCPWM0 TADC[1] comparison event 12: MCPWM0 TADC[2] comparison event 13: MCPWM0 TADC[3] comparison event 14: MCPWM1 TADC[0] comparison event 15: MCPWM1 TADC[1] comparison event 16: MCPWM1 TADC[2] comparison event 17: MCPWM1 TADC[3] comparison event

14.3.5.7 Timer3 Filter Control Register (UTIMER3_FLT)

Address: 0x4001_4318

Reset value: 0x0

Table 14-46 Timer3 Filter Control Register (UTIMER3_FLT)

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

Bit field	Bit Name	Description
[31:8]		Unused
[7:0]	FLT	Filter width of channel 0/1, with value range of 0-255. If FLT is 0, the channel will not be filtered. If FLT is not 0, the channel will be filtered, with the filter width of FLTx8. When the channel signal level is stable beyond FLT×8 system clock cycle width, the filter output is updated; otherwise, the filter leaves the current output unchanged.

Note that the working clock of the above filter is same as that of the corresponding Timer after frequency division, which is controlled by the frequency division coefficient of UTIMERx_CFG.CLK_DIV.

If UTIMERx_FLT.FLT = 0x6, UTIMERx_CFG.CLK_DIV = 0x2, the running clock of the Timer is

divided by 4 relative to the system clock. The channel input signal needs to be filtered by 8×6 times of Timer's operating clock, i.e., 8×6x4 system clock.

14.3.5.8 Timer3 Interrupt Enable Register (UTIMER3_IE)

Address: 0x4001_431C

Reset value: 0x0

Table 14-47 Timer3 Interrupt Enable Register (UTIMER3_IE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					ZC_RE	CH1_RE	CH0_RE						ZC_IE	CH1_IE	CH0_IE
					RW	RW	RW						RW	RW	RW
					0	0	0						0	0	0

Bit field	Bit Name	Description
[31:11]		Unused
[10]	ZC_RE	Timer counter over-zero DMA request enable, active high.
[9]	CH1_RE	Timer channel 1 comparison/capture request enable, active high.
[8]	CH0_RE	Timer channel 0 comparison/capture request enable, active high.
[7:3]		Unused
[2]	ZC_IE	Timer counter over-zero interrupt enable, active high.
[1]	CH1_IE	Timer channel 1 comparison/capture interrupt enable, active high.
[0]	CH0_IE	Timer channel 0 comparison/capture interrupt enable, active high.

DMA request event is the same event flag as interrupt event. After the DMA request is processed, the interrupt flag is automatically cleared by the DMA hardware without CPU software intervention. Note that typically, if a DMA request is turned on, the corresponding interrupt will be disabled.

14.3.5.9 Timer3 Interrupt Flag Register (UTIMER3_IF)

Address: 0x4001_4320

Reset value: 0x0

Table 14-48 Timer3 Interrupt Flag Register (UTIMER3_IF)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													ZC_IF	CH1_IF	CH0_IF

	RW1C	RW1C	RW1C		
	0	0	0		

Bit field	Bit Name	Description
[31:3]		Unused
[2]	ZC_IF	Timer counter over-zero interrupt flag. Active high, write 1 to clear.
[1]	CH1_IF	Timer channel 1 comparison/capture interrupt flag. Active high, write 1 to clear.
[0]	CH0_IF	Timer channel 0 comparison/capture interrupt flag. Active high, write 1 to clear.

14.3.6 UTimer4 Register

14.3.6.1 Timer4 Configuration Register (UTIMER4_CFG)

Address: 0x4001_4400

Reset value: 0x0

Table 14-49 Timer4 Configuration Register (UTIMER4_CFG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN				CAP1_CLR_EN	CAP0_CLR_EN	ONE_TRIG	ETON		CLK_DIV			CLK_SRC			
RW				RW	RW	RW	RW		RW			RW			
0				0	0	0	0		0			0			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRC1				CH1_POL	CH1_MODE	CH1_FE_CAP_EN	CH1_RE_CAP_EN	SRC0				CH0_POL	CH0_MODE	CH0_FE_CAP_EN	CH0_RE_CAP_EN
RW				RW	RW	RW	RW	RW				RW	RW	RW	RW
0001				0	0	0	0	0				0	0	0	0

Bit field	Bit Name	Description
[31]	EN	Timer module enable, active high
[30:26]		Unused
[25]	CAP1_CLR_EN	When a CAP1 capture event occurs, Timer counter will be cleared, active high

[24]	CAP0_CLR_EN	When a CAP0 capture event occurs, Timer counter will be cleared, active high
[23]	ONE_TRIG	In comparison mode and when EN=0, writing 1 triggers the Timer to send a cycle pulse with a specific duty cycle, and this bit value is 1 during the pulse sending period and will be automatically cleared after one Timber cycle.
[22]	ETON	Timer counter enable configuration. The default value is 0. 0: Auto run. 1: Wait for external event to trigger counting, and stop after one cycle of counting The external event is configured by the register UTIMER4_EVT.
[21:20]	CLK_DIV	Timer counter frequency configuration. The counting frequency of the counter is the 2^{CLK_DIV} division of the main clock frequency. The default value is 0 and does not divide frequency. 3'h0: 1 frequency division 3'h1: 2 frequency division 3'h2: 4 frequency division 3 'h3: 8 frequency division 3'h4: 16 frequency division 3'h5: 32 frequency division 3'h6: 64 frequency division 3'h7: 128 frequency division
[19:16]	CLK_SRC	Timer clock source 4'h0: Internal clock of chip 4'h1: Reserved 4'h2: External clock signal of Timer0 channel 0 4'h3: External clock signal of Timer0 channel 1 4 'h4: External clock signal of Timer1 channel 0 4 'h5: External clock signal of Timer1 channel 1 4'h6: External clock signal of Timer2 channel 0 4'h7: External clock signal of Timer2 channel 1 4'h8: External clock signal of Timer3 channel 0 4'h9: External clock signal of Timer3 channel 1 4'hA: External clock signal of Timer4 channel 0 4'hB: External clock signal of Timer4 channel 1
[15:12]	SRC1	Channel 1 signal source in Timer capture mode. The default value is 3'h1. 4'h0: Timer channel 0, from chip GPIO (see Datasheet and application configuration) 4'h1: Timer channel 1, from chip GPIO (see Datasheet and application configuration) 4'h2: Output of comparator 0 4'h3: Output of comparator 1

		4'h4: Output of comparator 2 4'h5: Output of comparator 3 4'h6: Output of comparator 4 4'h7: Output of comparator 5 4'h8: XOR of Timer channel 0 and 1
[11]	CH1_POL	Channel 1 output polarity control in compare mode: the output value when the counter count value returns to zero.
[10]	CH1_MODE	Timer channel 1 working mode selection. The default value is 0. 0: Comparison mode. Output square wave, and toggles when the channel 1 counter count value reaches 0 or the compare capture register value. 1: Capture mode. When a capture event occurs on the channel 1 input signal, the counter count value is stored in the channel 1 compare capture register.
[9]	CH1_FE_CAP_EN	Channel 1 falling edge capture event enable. 1: Enable; 0: Disable. A 1→0 transition on the Timer channel 1 input signal is considered a capture event. Falling edge event enable can coexist with rising edge event enable.
[8]	CH1_RE_CAP_EN	Channel 1 rising edge capture event enable. 1: Enable; 0: Disable. A 0→1 transition on the Timer channel 1 input signal is considered a capture event. Rising edge event enable can coexist with falling edge event enable.
[7:4]	SRC0	Channel 0 signal source in Timer capture mode. The default value is 3'h0. 4'h0: Timer channel 0, from chip GPIO (see Datasheet and application configuration) 4'h1: Timer channel 1, from chip GPIO (see Datasheet and application configuration) 4'h2: Output of comparator 0 4'h3: Output of comparator 1 4'h4: Output of comparator 2 4'h5: Output of comparator 3 4'h6: Output of comparator 4 4'h7: Output of comparator 5 4'h8: XOR of Timer channel 0 and 1
[3]	CH0_POL	Channel 0 output polarity control in compare mode: the output value when the counter count value returns to zero.
[2]	CH0_MODE	Timer channel 0 working mode selection. The default value is 0. 0: Comparison mode. Output square wave, and toggles when the channel 0 counter count value reaches 0 or the compare capture register value. 1: Capture mode. When a capture event occurs on the channel 0 input signal, the counter count value is stored in the channel 0

		compare capture register.
[3]	CH0_MODE	Timer channel 0 working mode selection. The default value is 0. 0: Comparison mode. Output square wave, and toggles when the channel 0 counter count value reaches 0 or the compare capture register value. 1: Capture mode. When a capture event occurs on the channel 0 input signal, the counter count value is stored in the channel 0 compare capture register.
[2]	CH0_POL	Channel 0 output polarity control in compare mode: the output value when the counter count value returns to zero.
[1]	CH0_FE_CAP_EN	Channel 0 falling edge capture event enable. 1: Enable; 0: Disable. A 1→0 transition on the Timer channel 0 input signal is considered a capture event. Falling edge event enable can coexist with rising edge event enable.
[0]	CH0_RE_CAP_EN	Channel 0 rising edge capture event enable. 1: Enable; 0: Disable. A 0→1 transition on the Timer channel 0 input signal is considered a capture event. Rising edge event enable can coexist with falling edge event enable.

14.3.6.2 Timer4 Threshold Register (UTIMER4_TH)

Address: 0x4001_4404

Reset value: 0x0

Table 14-50 Timer4 Threshold Register (UTIMER4_TH)

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

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	TH	Timer0 counter count threshold. The counter counts from 0 to TH, and then returns to 0 to start counting.

14.3.6.3 Timer4 Count Register (UTIMER4_CNT)

Address: 0x4001_4408

Reset value: 0x0

Table 14-51 Timer4 Count Register (UTIMER4_CNT)

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

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	CNT	The current count value of the Timer0 counter. New count value can be written.

Note: The Timer clock needs to be started by SYS_CLK_FEN before writing to UTIMER4_CNT.

14.3.6.4 Timer4 Channel 0 Compare Capture Register (UTIMER4_CMP0)

Address: 0x4001_440C

Reset value: 0x0

Table 14-52 Timer4 Channel 0 Compare Capture Register (UTIMER4_CMP0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP0															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	CMP0	When Timer channel 0 works in compare mode and the counter count value is equal to CMP0, a comparison event occurs. When Timer channel 0 works in the capture mode, the counter count value when the capture event occurs is stored in the CMP0 register.

14.3.6.5 Timer4 Channel 1 Compare Capture Register (UTIMER4_CMP1)

Address: 0x4001_4410

Reset value: 0x0

Table 14-53Timer4 Channel 1 Compare Capture Register (UTIMER4_CMP1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP1															
RW															

0

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	CMP1	When Timer channel 0 works in compare mode and the counter count value is equal to CMP1, a comparison event occurs. When Timer channel 0 works in the capture mode, the counter count value when the capture event occurs is stored in the CMP1 register.

14.3.6.6 Timer4 External Event Select Register (UTIMER4_EVT)

Address: 0x4001_4414

Reset value: 0x0

Table 14-54Timer4 External Event Select Register (UTIMER4_EVT)

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

Bit field	Bit Name	Description
[31:3]		Unused
[2:0]	EVT_SRC	Timer external event selection register. This register should be used with UTIMER4_CFG.ETON. When ETON is high, select the event that triggers Timer0 count by this register. Note that the comparison event of Timer cannot trigger Timer for counting. 0: TIMER0 channel 0 comparison event 1: TIMER0 channel 1 comparison event 2: TIMER1 channel 0 comparison event 3: TIMER1 channel 1 comparison event 4: TIMER2 channel 0 comparison event 5: TIMER2 channel 1 comparison event 6: TIMER3 channel 0 comparison event 7: TIMER3 channel 1 comparison event 8: TIMER4 channel 0 comparison event 9: TIMER4 channel 1 comparison event 10: MCPWM0 TADC[0] comparison event 11: MCPWM0 TADC[1] comparison event 12: MCPWM0 TADC[2] comparison event 13: MCPWM0 TADC[3] comparison event

		14: MCPWM1 TADC[0] comparison event 15: MCPWM1 TADC[1] comparison event 16: MCPWM1 TADC[2] comparison event 17: MCPWM1 TADC[3] comparison event

14.3.6.7 Timer3 Filter Control Register (UTIMER3_FLT)

Address: 0x4001_4418

Reset value: 0x0

Table 14-55Timer4 Filter Control Register (UTIMER4_FLT)

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

Bit field	Bit Name	Description
[31:8]		Unused
[7:0]	FLT	Filter width of channel 0/1, with value range of 0-255. If FLT is 0, the channel will not be filtered. If FLT is not 0, the channel will be filtered, with the filter width of FLTx8. When the channel signal level is stable beyond FLT×8 system clock cycle width, the filter output is updated; otherwise, the filter leaves the current output unchanged.

Note that the working clock of the above filter is same as that of the corresponding Timer after frequency division, which is controlled by the frequency division coefficient of UTIMERx_CFG.CLK_DIV.

If UTIMERx_FLT.FLT = 0x6, UTIMERx_CFG.CLK_DIV = 0x2, the running clock of the Timer is divided by 4 relative to the system clock. The channel input signal needs to be filtered by 8×6 times of Timer's operating clock, i.e., 8×6x4 system clock.

14.3.6.8 Timer4 Interrupt Enable Register (UTIMER4_IE)

Address: 0x4001_441C

Reset value: 0x0

Table 14-56Timer4 Interrupt Enable Register (UTIMER4_IE)

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

	ZC_RE	CH1_RE	CH0_RE		ZC_IE	CH1_IE	CH0_IE
	RW	RW	RW		RW	RW	RW
	0	0	0		0	0	0

Bit field	Bit Name	Description
[31:11]		Unused
[10]	ZC_RE	Timer counter over-zero DMA request enable, active high.
[9]	CH1_RE	Timer channel 1 comparison/capture request enable, active high.
[8]	CH0_RE	Timer channel 0 comparison/capture request enable, active high.
[7:3]		Unused
[2]	ZC_IE	Timer counter over-zero interrupt enable, active high.
[1]	CH1_IE	Timer channel 1 comparison/capture interrupt enable, active high.
[0]	CH0_IE	Timer channel 0 comparison/capture interrupt enable, active high.

DMA request event is the same event flag as interrupt event. After the DMA request is processed, the interrupt flag is automatically cleared by the DMA hardware without CPU software intervention. Note that typically, if a DMA request is turned on, the corresponding interrupt will be disabled.

14.3.6.9 Timer4 Interrupt Flag Register (UTIMER4_IF)

Address: 0x4001_4420

Reset value: 0x0

Table 14-57Timer4 Interrupt Flag Register (UTIMER4_IF)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													ZC_IF	CH1_IF	CH0_IF
													RW1C	RW1C	RW1C
													0	0	0

Bit field	Bit Name	Description
[31:3]		Unused
[2]	ZC_IF	Timer counter over-zero interrupt flag. Active high, write 1 to clear.
[1]	CH1_IF	Timer channel 1 comparison/capture interrupt flag. Active high, write 1 to clear.

[0]	CH0_IF	Timer channel 0 comparison/capture interrupt flag. Active high, write 1 to clear.

14.3.7 QEP0 Register

14.3.7.1 QEP0 Configuration Register (QEP0_CFG)

QEP0_CFG address: 0x4001_4500

Reset value: 0x0

Table 14-58QEP0 Configuration Register (QEP0_CFG)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN					FE_CNT_EN	MODE						ZNC	ZPC	ZLC	ZEC
RW					RW	RW						RW	RW	RW	RW
0					0	0						0	0	0	0

Bit field	Bit Name	Description
[31:16]		Unused
[15]		QEP module enable
[14:11]		Unused
[10]	FE_CNT_EN	Whether to count on the falling edge (rising edge is always counted) in CCW+SIGN and CCW+CW mode
[9:8]	MODE	Encoder mode selection 00: counting on T1 01: counting on T1 & T2 The above are the counting modes of the orthogonal code signal 10: CCW+SIGN, pulse signal counting with symbolic data 11: CCW+CW, CCW+CW double pulse signal counting
[7:4]		Unused
[3]	ZNC	Z signal clear polarity selection: Low level/falling edge clear enable
[2]	ZPC	Z signal clear polarity selection: High level/rising edge clear enable
[1]	ZLC	Z signal level clear QEP counter enable
[0]	ZEC	Z signal edge clear QEP counter enable

When ZEC=1, the transition edge of Z signal can clear the QEP counting, and when ZNC=1, the falling edge of Z signal will clear the QEP counter, or when ZPC=1, the rising edge of Z signal will clear the QEP counter; if both ZNC=1 and ZPC=1, any change to the Z signal will clear the QEP counter, which will be immediately reset for future counting.

When ZLC=1, the effective level of Z signal can clear the QEP counting, and when ZNC=1, the low level of Z signal will clear the QEP counter, or when ZPC=1, the high level of Z signal will clear the QEP counter; generally, the effective level of Z signal is used to clear the QEP counter. ZNC and ZPC cannot be both 1 simultaneously, otherwise the counter will be always cleared. When effective level is used

for clearing, QEP counter will recover counting only after transition of Z signal.

14.3.7.2 QEP0 Count Threshold Register (QEP0_TH)

QEP0_TH address: 0x4001_4504

Reset value: 0x0

Table 14-59QEP0 Count Threshold Register (QEP0_TH)

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

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	TH	Count threshold TH. After the encoder counts up (increases) to the threshold value, counting up again will cause the counter to return to zero. After the encoder counts down (decreases) to zero, counting down again will cause the counter to return to the threshold value.

14.3.7.3 QEP0 Count Value Register (QEP0_CNT)

QEP0_CNT address: 0x4001_4508

Reset value: 0x0

Table 14-60QEP0 Count Value Register (QEP0_CNT)

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

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	CNT	QEP0 count value.

14.3.7.4 QEP0 Interrupt Enable Register (QEP0_IE)

Address: 0x4001_450C

Reset value: 0x0

Table 14-61QEP0 Interrupt Enable Register (QEP0_IE)

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

Bit field	Bit Name	Description
[31:2]		Unused
[1]	OF_IE	Overflow interrupt enable. Active high.
[0]	UF_IE	Underflow interrupt enable. Active high.

14.3.7.5 QEP0 Interrupt Flag Register (QEP0_IF)

Address: 0x4001_4510

Reset value: 0x0

Table 14-62QEP0 Interrupt Flag Register (QEP0_IF)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														OF_IF	UF_IF
														RW1C	RW1C
														0	0

Bit field	Bit Name	Description
[31:2]		Unused
[1]	OF_IF	Overflow interrupt flag. Active high, write 1 to clear. When the counter reaches the count threshold, the up-count event triggers an overflow interrupt.
[0]	UF_IF	Underflow interrupt flag. Active high, write 1 to clear. When the counter reaches zero, the down-count event triggers an overflow interrupt.

14.3.8 QEP1 Register

14.3.8.1 QEP1 Configuration Register (QEP1_CFG)

QEP1_CFG address: 0x4001_4600

Reset value: 0x0

Table 14-63QEP1 Configuration Register (QEP1_CFG)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN					FE_CNT_EN	MODE						ZNC	ZPC	ZLC	ZEC
RW					RW	RW						RW	RW	RW	RW
0					0	0						0	0	0	0

Bit field	Bit Name	Description
[31:11]		Unused
[10]	FE_CNT_EN	Whether to count on the falling edge (rising edge is always counted) in CCW+SIGN and CCW+CW mode
[9:8]	MODE	Encoder mode selection 00: counting on T1 01: counting on T1 & T2 The above are the counting modes of the orthogonal code signal 10: CCW+SIGN, pulse signal counting with symbolic data 11: CCW+CW, CCW+CW double pulse signal counting
[7:4]		Unused
[3]	ZNC	Z signal clear polarity selection: Low level/falling edge clear enable
[2]	ZPC	Z signal clear polarity selection: High level/rising edge clear enable
[1]	ZLC	Z signal level clear QEP counter enable
[0]	ZEC	Z signal edge clear QEP counter enable

When ZEC=1, the transition edge of Z signal can clear the QEP counting, and when ZNC=1, the falling edge of Z signal will clear the QEP counter, or when ZPC=1, the rising edge of Z signal will clear the QEP counter; if both ZNC=1 and ZPC=1, any change to the Z signal will clear the QEP counter, which will be immediately reset for future counting.

When ZLC=1, the effective level of Z signal can clear the QEP counting, and when ZNC=1, the low level of Z signal will clear the QEP counter, or when ZPC=1, the high level of Z signal will clear the QEP counter; generally, the effective level of Z signal is used to clear the QEP counter. ZNC and ZPC cannot be both 1 simultaneously, otherwise the counter will be always cleared. When effective level is used for clearing, QEP counter will recover counting only after transition of Z signal.

14.3.8.2 QEP1 Count Threshold Register (QEP1_TH)

QEP1_TH address: 0x4001_4604

Reset value: 0x0

Table 14-64QEP1 Count Threshold Register (QEP1_TH)

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

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	TH	QEP count threshold TH. After the encoder counts up (increases) to the threshold value, counting up again will cause the counter to return to zero. After the encoder counts down (decreases) to zero, counting down again will cause the counter to return to the threshold value.

14.3.8.3 QEP1 Count Value Register (QEP1_CNT)

QEP1_CNT address: 0x4001_4608

Reset value: 0x0

Table 14-65QEP1 Count Value Register (QEP1_CNT)

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

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	CNT	QEP count value.

14.3.8.4 QEP1 Interrupt Enable Register (QEP1_IE)

Address: 0x4001_460C

Reset value: 0x0

Table 14-66QEP1 Interrupt Enable Register (QEP1_IE)

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

	OF_IE	UF_IE
	RW	RW
	0	0

Bit field	Bit Name	Description
[31:2]		Unused
[1]	OF_IE	Overflow interrupt enable. Active high.
[0]	UF_IE	Underflow interrupt enable. Active high.

14.3.8.5 QEP1 Interrupt Flag Register (QEP1_IF)

Address: 0x4001_4610

Reset value: 0x0

Table 14-67 QEP1 Interrupt Flag Register (QEP1_IF)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														OF_IF	UF_IF
														RW1C	RW1C
														0	0

Bit field	Bit Name	Description
[31:2]		Unused
[1]	OF_IF	Overflow interrupt flag. Active high, write 1 to clear. When the counter reaches the count threshold, the up-count event triggers an overflow interrupt.
[0]	UF_IF	Underflow interrupt flag. Active high, write 1 to clear. When the counter reaches zero, the down-count event triggers an overflow interrupt.

14.3.9 QEP2 Register

14.3.9.1 QEP2 Configuration Register (QEP2_CFG)

QEP2_CFG address: 0x4001_4700

Reset value: 0x0

Table 14-68QEP2 Configuration Register (QEP2_CFG)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					FE_CNT_EN	MODE						ZNC	ZPC	ZLC	ZEC
					RW	RW						RW	RW	RW	RW
					0	0						0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN					FE_CNT_EN	MODE						ZNC	ZPC	ZLC	ZEC
RW					RW	RW						RW	RW	RW	RW
0					0	0						0	0	0	0

Bit field	Bit Name	Description
[31:11]		Unused
[10]	FE_CNT_EN	Whether to count on the falling edge (rising edge is always counted) in CCW+SIGN and CCW+CW mode
[9:8]	MODE	Encoder mode selection 00: counting on T1 01: counting on T1 & T2 The above are the counting modes of the orthogonal code signal 10: CCW+SIGN, pulse signal counting with symbolic data 11: CCW+CW, CCW+CW double pulse signal counting
[7:4]		Unused
[3]	ZNC	Z signal clear polarity selection: Low level/falling edge clear enable
[2]	ZPC	Z signal clear polarity selection: High level/rising edge clear enable
[1]	ZLC	Z signal level clear QEP counter enable
[0]	ZEC	Z signal edge clear QEP counter enable

When ZEC=1, the transition edge of Z signal can clear the QEP counting, and when ZNC=1, the falling edge of Z signal will clear the QEP counter, or when ZPC=1, the rising edge of Z signal will clear the QEP counter; if both ZNC=1 and ZPC=1, any change to the Z signal will clear the QEP counter, which will be immediately reset for future counting.

When ZLC=1, the effective level of Z signal can clear the QEP counting, and when ZNC=1, the low level of Z signal will clear the QEP counter, or when ZPC=1, the high level of Z signal will clear the QEP counter; generally, the effective level of Z signal is used to clear the QEP counter. ZNC and ZPC cannot be both 1 simultaneously, otherwise the counter will be always cleared. When effective level is used for clearing, QEP counter will recover counting only after transition of Z signal.

14.3.9.2 QEP2 Count Threshold Register (QEP2_TH)

QEP2_TH address: 0x4001_4704

Reset value: 0x0

Table 14-69QEP2 Count Threshold Register (QEP2_TH)

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

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	TH	QEP count threshold TH. After the encoder counts up (increases) to the threshold value, counting up again will cause the counter to return to zero. After the encoder counts down (decreases) to zero, counting down again will cause the counter to return to the threshold value.

14.3.9.3 QEP2 Count Value Register (QEP2_CNT)

QEP2_CNT address: 0x4001_4708

Reset value: 0x0

Table 14-70 QEP2 Count Value Register (QEP2_CNT)

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

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	CNT	QEP count value.

14.3.9.4 QEP2 Interrupt Enable Register (QEP2_IE)

Address: 0x4001_470C

Reset value: 0x0

Table 14-71QEP2 Interrupt Enable Register (QEP2_IE)

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

	OF_IE	UF_IE
	RW	RW
	0	0

Bit field	Bit Name	Description
[31:2]		Unused
[1]	OF_IE	Overflow interrupt enable. Active high.
[0]	UF_IE	Underflow interrupt enable. Active high.

14.3.9.5 QEP2 Interrupt Flag Register (QEP2_IF)

Address: 0x4001_4710

Reset value: 0x0

Table 14-72 QEP2 Interrupt Flag Register (QEP2_IF)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														OF_IF	UF_IF
														RW1C	RW1C
														0	0

Bit field	Bit Name	Description
[31:2]		Unused
[1]	OF_IF	Overflow interrupt flag. Active high, write 1 to clear. When the counter reaches the count threshold, the up-count event triggers an overflow interrupt.
[0]	UF_IF	Underflow interrupt flag. Active high, write 1 to clear. When the counter reaches zero, the down-count event triggers an overflow interrupt.

14.3.10 QEP3 Register

14.3.10.1 QEP3 Configuration Register (QEP3_CFG)

QEP3_CFG address: 0x4001_4800

Reset value: 0x0

Table 14-73QEP3 Configuration Register (QEP3_CFG)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					FE_CNT_EN	MODE						ZNC	ZPC	ZLC	ZEC
					RW	RW						RW	RW	RW	RW
					0	0						0	0	0	0

Bit field	Bit Name	Description
[31:11]		Unused
[10]	FE_CNT_EN	Whether to count on the falling edge (rising edge is always counted) in CCW+SIGN and CCW+CW mode
[9:8]	MODE	Encoder mode selection 00: counting on T1 01: counting on T1 & T2 The above are the counting modes of the orthogonal code signal 10: CCW+SIGN, pulse signal counting with symbolic data 11: CCW+CW, CCW+CW double pulse signal counting
[7:4]		Unused
[3]	ZNC	Z signal clear polarity selection: Low level/falling edge clear enable
[2]	ZPC	Z signal clear polarity selection: High level/rising edge clear enable
[1]	ZLC	Z signal level clear QEP counter enable
[0]	ZEC	Z signal edge clear QEP counter enable

When ZEC=1, the transition edge of Z signal can clear the QEP counting, and when ZNC=1, the falling edge of Z signal will clear the QEP counter, or when ZPC=1, the rising edge of Z signal will clear the QEP counter; if both ZNC=1 and ZPC=1, any change to the Z signal will clear the QEP counter, which will be immediately reset for future counting.

When ZLC=1, the effective level of Z signal can clear the QEP counting, and when ZNC=1, the low level of Z signal will clear the QEP counter, or when ZPC=1, the high level of Z signal will clear the QEP counter; generally, the effective level of Z signal is used to clear the QEP counter. ZNC and ZPC cannot be both 1 simultaneously, otherwise the counter will be always cleared. When effective level is used for clearing, QEP counter will recover counting only after transition of Z signal.

14.3.10.2QEP3 Count Threshold Register (QEP3_TH)

QEP3_TH address: 0x4001_4804

Reset value: 0x0

Table 14-74QEP3 Count Threshold Register (QEP3_TH)

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

TH
RW
0

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	TH	QEP count threshold TH. After the encoder counts up (increases) to the threshold value, counting up again will cause the counter to return to zero. After the encoder counts down (decreases) to zero, counting down again will cause the counter to return to the threshold value.

14.3.10.3 QEP3 Count Value Register (QEP3_CNT)

QEP3_CNT address: 0x4001_4808

Reset value: 0x0

Table 14-75 QEP3 Count Value Register (QEP3_CNT)

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

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	CNT	QEP count value.

14.3.10.4 QEP3 Interrupt Enable Register (QEP3_IE)

Address: 0x4001_480C

Reset value: 0x0

Table 14-76 QEP3 Interrupt Enable Register (QEP3_IE)

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

Bit field	Bit Name	Description
[31:2]		Unused
[1]	OF_IE	Overflow interrupt enable. Active high.
[0]	UF_IE	Underflow interrupt enable. Active high.

14.3.10.5 QEP3 Interrupt Flag Register (QEP3_IF)

Address: 0x4001_4810

Reset value: 0x0

Table 14-77 QEP3 Interrupt Flag Register (QEP3_IF)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														OF_IF	UF_IF
														RW1C	RW1C
														0	0

Bit field	Bit Name	Description
[31:2]		Unused
[1]	OF_IF	Overflow interrupt flag. Active high, write 1 to clear. When the counter reaches the count threshold, the up-count event triggers an overflow interrupt.
[0]	UF_IF	Underflow interrupt flag. Active high, write 1 to clear. When the counter reaches zero, the down-count event triggers an overflow interrupt.

15 HALL Signal Processing Module

15.1 Overview

The chip integrates two identical HALL signal processing modules, and the following instructions are for a HALL signal processing module.

Support 3 HALL signal inputs.

The processing of the input HALL sensor signal includes:

Filtering. Eliminate the effect of HALL signal glitches.

Capture. When the HALL input changes, record the current timer value and output an interrupt.

Overflow. When the HALL signal remains changed for a long time, resulting in the counter overflows, an interrupt is output.

15.2 Implementation Description

15.2.1 Signal Source

The HALL signal comes from GPIO, and the chip has two IOs that can be used as the source of each HALL signal. Users can choose to use one of the GPIO input signals as the HALL signal by setting the GPIO register.

Please see DATASHEET for detailed pin location.

15.2.2 System Clock

The working frequency of HALL module is adjustable. Users can select the 1/2/4/8 frequency division of the main clock as the operating frequency of the HALL module by setting the HALL_CFG.CLK_DIV register. Both filtering and counting work at this frequency.

15.2.3 Signal Filtering

The filter module is mainly used to remove glitches on the HALL signal.

The filtering includes two stages of filters, which can be turned on independently or simultaneously:

In the first stage, the "5 in 7" rule is used for filtering. That is, if five "1" is reached while filtering the seven consecutive sampling points, output as "1"; if reached or over five "0", output as "0"; otherwise, the output result would remain unchanged as the last filtering. Select whether to enable the first-stage filter by setting HALL_CFG.FIL_75. The details are shown below:

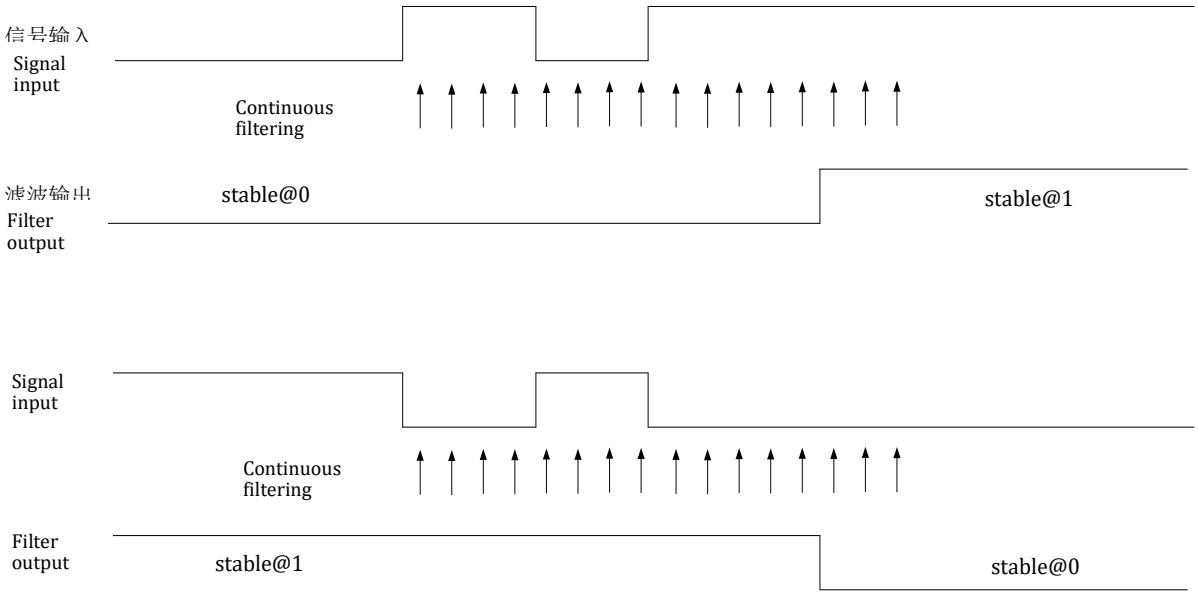


Figure 15-1 7/5 Filter Module Block Diagram

In the second stage, continuous filtering is adopted. If all results filtered are zero while filtering N consecutive sampling points, output as "0"; if all results filtered are "1", output as "1"; otherwise, the output result would remain unchanged as the last filtering.

Select the filtering depth of the second-stage filter by setting HALL_CFG.FIL_LEN, that is, the number of consecutive samples. The maximum number of consecutive samples is 2, and the calculation formula of the filter time constant is as follows:

$$T_{fit} = T_{clk} * (HALL\_CFG.FIL\_LEN[14:0] + 1)$$

For example, at a 192MHz operating frequency, the period Tclk is 5.2ns, the maximum register configuration is 32767, and the longest filter width is about 5.2ns × 32768 ≈ 170us.

Capture the filtered HALL signal by accessing HALL_INFO.FIL_DATA [2:0]; HALL_INFO.RAW_DATA [2:0] is the original HALL input signal before filtering.

15.2.4 Acquisition

The capture module is used to measure the time between two HALL signal changes, with a 24-bit counter as its core, which can record a maximum time width of about 0.87 seconds at a 192MHz operating frequency, and achieve a time resolution of 5.2ns.

HALL_CNT starts counting from 0. When the HALL signal changes, the current HALL_CNT value will be saved to the HALL_WIDTH register, and the current HALL signal is saved to HALL_INFO.FIL_DATA. Then, a HALL signal change interrupt is output, and HALL_CNT starts counting from 0 again.

When the counter count value reaches HALL_TH, the HALL counter overflow interrupt is output, and the counter starts counting from 0 again.

15.2.5 Interruption

Capture and overflow events trigger interrupts. The interrupt enable control bits are in HALL_CFG.CHG_IE and HALL_CFG.OV_IE, and the interrupt flag bits are in HALL_INFO.CHG_IF and HALL_INFO.OV_IF. The interrupt flag can be cleared by writing 1 to HALL_INFO.CHG_IF and HALL_INFO.OV_IF.

15.2.6 Data Flow

The data flow of the HALL module is shown in the figure below. FCLK is the system master clock controlled by the SYS_CLK_FEN gate control, which is usually a PLL clock.

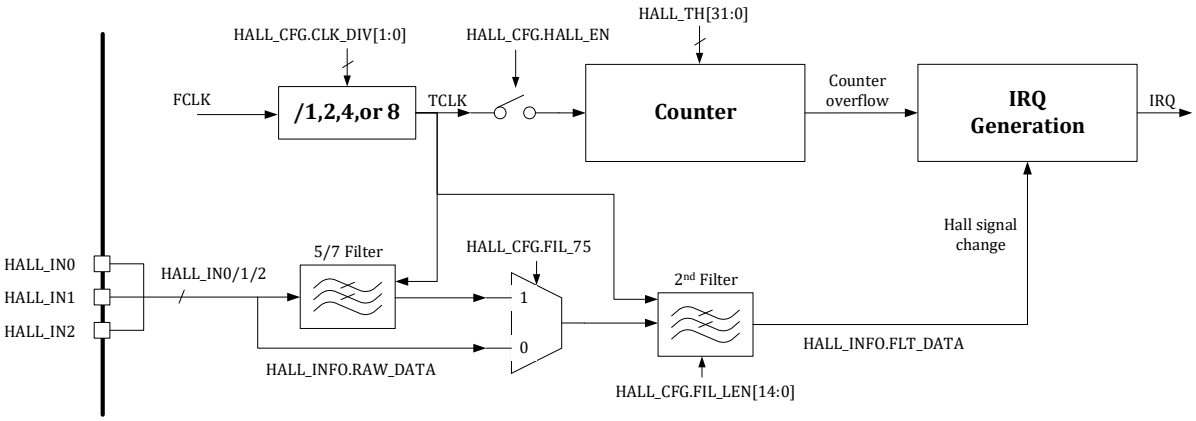


Figure 15-2 HALL Module Block Diagram

15.3 Register

15.3.1 Address Allocation

The base address of the HALL0 module register is 0x4001_1800.

The base address of the HALL1 module register is 0x4001_1C00.

The register list is as follows:

Table 15-1 HALL Module Register Address Allocation

Name	Offset	Description
HALLx_CFG	0x00	HALL module configuration register
HALLx_INFO	0x04	HALL module information register
HALLx_WIDTH	0x08	HALL width count value register
HALLx_TH	0x0C	HALL module counter threshold register
HALLx_CNT	0x10	HALL count register

15.3.2 Module Configuration Register (x = 0,1) (HALLx_CFG)

HALL0_CFG address: 0x4001_1800

HALL1_CFG address: 0x4001_1C00

Reset value: 0x0

Table 15-2HALL Module Configuration Register (HALLx_CFG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	SW_IE	OV_IE	CHG_IE	SW_RE	OV_RE	CHG_RE	HALL_EN				FIL_75			CLK_DIV	
	RW	RW	RW	RW	RW	RW	RW				RW			RW	
	0	0	0	0	0	0	0				0			0	

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

Bit field	Bit Name	Description
[31]		Unused
[30]	SW_IE	Software-triggered HALL signal change interrupt enable. Active high. After this bit is valid, writing 1 to INFO [18], a HALL signal change interrupt will be generated manually.
[29]	OV_IE	HALL counter overflow interrupt enable switch. Off by default. 0: Disable 1: Enable
[28]	CHG_IE	HALL signal change interrupt enable switch. Off by default. 0: Disable 1: Enable
[27]	SW_RE	DMA request enable for software-triggered HALL signal change. Active high. After this bit is valid, writing 1 to INFO [18], a HALL signal change DMA request will be generated manually.
[26]	OV_RE	HALL counter overflow DMA request enable switch. Off by default. 0: Disable 1: Enable
[25]	CHG_RE	HALL signal change DMA request enable switch. Off by default. 0: Disable 1: Enable
[24]	HALL_EN	HALL module enable signal. Off by default. 0: Disable 1: Enable
[23:21]		Unused
[20]	FIL_75	7/5 filter switch (sequential sampling for seven times, and five results

		should be the same). Off by default. 0: Disable 1: Enable
[19:18]		Unused
[17:16]	CLK_DIV	HALL clock division factor. No frequency division by default. 00: No frequency division 01: Two-divided frequency 10: Four-divided frequency 11: Eight-divided frequency
[15]		Unused
[14:0]	FIL_LEN	Filter width. Signals below the corresponding pulse width will be automatically filtered by the hardware. The calculation formula of the filter width is [14:0]+1.

15.3.3 Module Info Register (x = 0,1) (HALLx_INFO)

The addresses are: 0x4001_1804 and 0x4001_1C04.

Reset value: 0x0

Table 15-3HALL Module Information Register (HALLx_INFO)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
													SW_IF	OV_IF	CHG_IF
													RW	RW	RW
													0	0	0

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

Bit field	Bit Name	Description
[31:19]		Unused
[18]	SW_IF	Software-triggered HALL signal change interrupt. Trigger by writing 1, and clear automatically.
[17]	OV_IF	HALL counter overflow event flag. Write 1 to clear
[16]	CHG_IF	HALL signal change event flag. Write 1 to clear
[15:11]		Reserved bit. Write 0, and read 0
[10:8]	RAW_DATA	HALL value. Unfiltered result
[7:3]		Reserved bit. Write 0, and read 0
[2:0]	FLT_DATA	HALL value. Filter result

Generally, the software cannot detect the value of SW_IF which is automatically cleared, but CFG_IF will be set when the software triggers the HALL signal change. Therefore, HALL interrupt or DMA

request event can be generated after enabling CFG_IE or CFG_RE and writing software trigger.

15.3.4 HALL Width Count Value Register (x = 0,1) (HALL_WIDTH)

The addresses are: 0x4001_1808 and 0x4001_1C08.

Reset value: 0x0

Table 15-4HALL Width Count Value Register (HALLx_WIDTH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								CAP_CNT							
								RO							
								0							

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

Bit field	Bit Name	Description
[31:24]		Unused
[23:0]	CAP_CNT	HALL width counter value

15.3.5 HALL Module Counter Threshold Register (x = 0,1) (HALL_TH)

The addresses are: 0x4001_180C and 0x4001_1C0C.

Reset value: 0x0

Table 15-5HALL Module Counter Threshold Register (HALLx_TH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								TH							
								RW							
								0							

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

Bit field	Bit Name	Description
[31:24]		Unused
[23:0]	TH	HALL counter threshold

15.3.6 HALL Width Count Value Register (x = 0,1) (HALL_CNT)

The addresses are: 0x4001_1810 and 0x4001_1C10.

Reset value: 0x0

Table 15-6 HALL Count Register (HALLx_CNT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								CNT							
								RW							
								0							

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

Bit field	Bit Name	Description
[31:24]		Unused
[23:0]	CNT	HALL count value. Write any value to clear

16 MCPWM

16.1 Overview

The MCPWM module is a module that precisely controls the output of the motor drive waveform. The chip integrates two identical MCPWM modules, and the following instructions are for a MCPWM module.

A MCPWM module contains two 16-bit counter-up counters to provide two basic periods (hereinafter referred to as two time bases). The counter has four clock frequency divisions, 1-, 2-, 4-, and 8-divided frequency division, and the divided clock frequencies generated are 192, 96, 48 and 24MHz, respectively. Channel 0/1/2 must use time base 0, and channel 3 must uses time base 1.

It contains four groups of PWM generation modules.

-Able to produce four pairs (complementary signals) or eight independent (edge-aligned mode) non-overlapping PWM signals;

-Support edge-aligned PWM

-Center-aligned PWM

-Phase shift PWM

Besides, it can generate four channels of timing information at the same time as MCPWM, which is used to trigger the synchronous sampling of the ADC module for linkage with MCPWM.

It also contains a set of emergency stop protection modules for quickly shutting down the output of the MCPWM module without relying on CPU software processing. The MCPWM module can input eight emergency stop signals, four of which come from IO and four from the output of the on-chip comparator. When an emergency stop event occurs (supports effective level polarity selection), reset all MCPWM output signals to the specified state to avoid short circuit.

Moreover, there is an independent filter module for the emergency stop signal.

Each output IO of MCPWM supports two control modes: PWM hardware control or software direct control (for EABS soft brake, or BLDC square-wave commutation control).

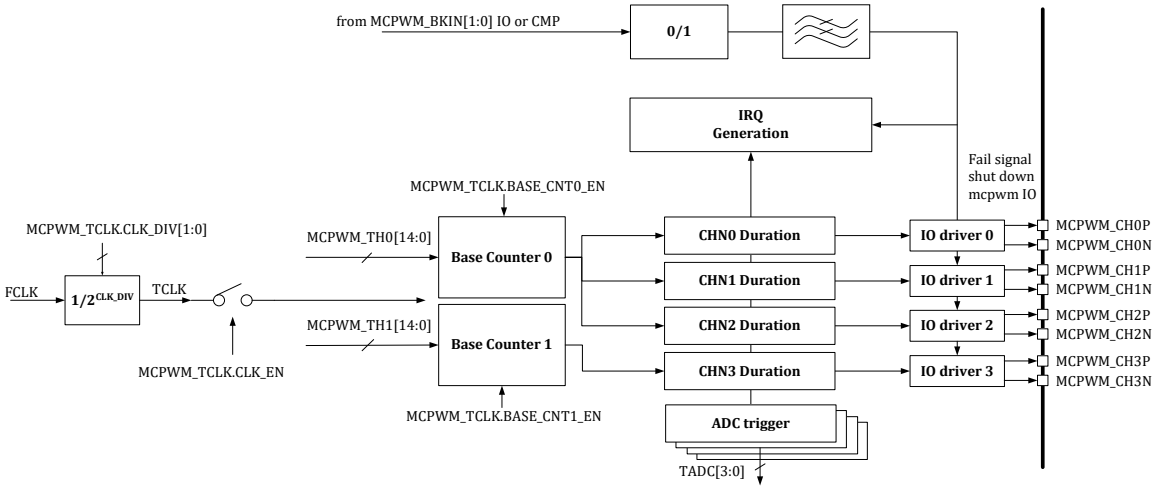


Figure 16-1 MCPWM Module Block Diagram

Usually, a 192MHz PLL clock is used as the operating clock of the MCPWM module to obtain higher timing accuracy.

16.1.1 Base Counter Module

The module is mainly composed of two count-up counters, with the count threshold of MCPWM_TH0/MCPWM_TH1. The counter starts at time t0, and counts up from -TH. It passes the time zero at time t1, counts at time t2 to TH to complete a counting cycle, and then returns to -TH to restart counting. The count period is (TH×2+1) times the count clock period.

Timed event interrupt can be generated at t0/t1 (current time t0 is the previous t2), MCPWM_IF.T0_IF and MCPWM_IF.T1_IF will be set.

The start and stop of the Base Counter can be controlled by register configuration MCPWM_TCLK.BASE_CNT_EN.

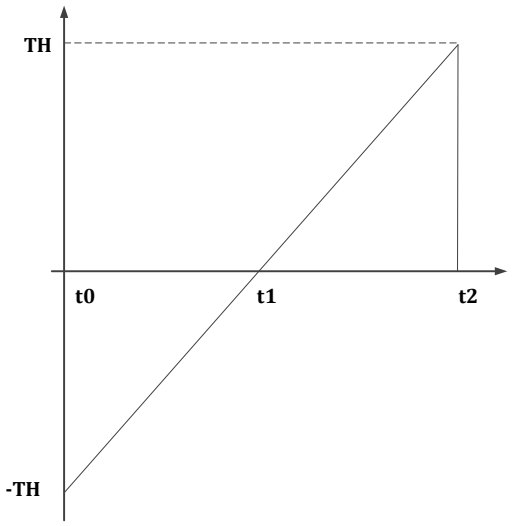


Figure 16-2 Base Counter t0/t1 Timing

Before running the MCPWM module, users should set the corresponding comparison thresholds (MCPWM_TH00 - MCWM_TH31) and dead-zone registers (MCPWM_DTH00 - MCWM_DTH31) in advance. In the actual operation process, the comparison threshold value and the PWM period register can also be changed. Update manually by writing to the MCPWM_UPDATE register, or complete hardware auto-update by setting MCPWM_SDCFG.T1_UPDATE_EN and MCPWM_SDCFG.T0_UPDATE_EN. The hardware update can only generate update events at time t0 and t1 (update t0 or t1 and update both t0 and t1 at all times), and the hardware loads the value of the load register into the running shadow register. The occurrence frequency of the update event can be set, that is, the update occurs every N time t0 and t1. Regardless of whether an update occurs, a corresponding interrupt can be generated at t0 and t1. If the hardware loads the value of the load register into the running register, a load interrupt is generated.

Select whether the update occurs at t0 or t1 or both by setting the MCPWM_SDCFG register, and set the update interval number as 1-16. The most frequent update configuration is that updates occur at t0 and t1, which occur continuously. The lowest speed update configuration is that the update occurs at t1, and updates every sixteen t1.

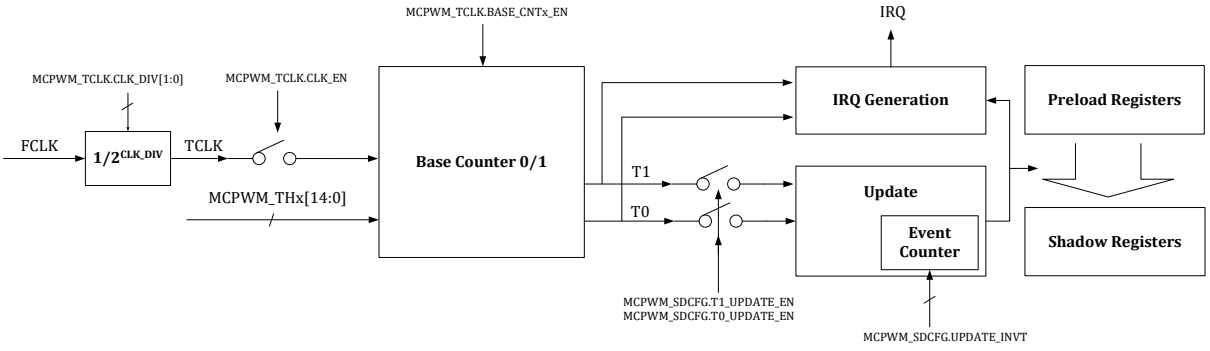


Figure 16-3 MCPWM Update Mechanism

The TH register MCPWM_THx and CNT register MCPWM_CNTx (x=0,1) correspond to Time base

0/1 have shadow registers and support manual update (update by writing to MCPWM_UPDATE through software) and auto update (update in case of specific hardware event). Shadow registers can be updated without enabling the MCPWM TCLK clock (i.e. MCPWM_TCLK.TCLK_EN = 0). MCPWM_TCLK.BASE_CNT0_EN and MCPWM_TCLK.BASE_CNT1_EN control the CNT counting enable of time base 0 and time base 1, respectively. If MCPWM_TCLK.BASE_CNT0_EN=0, time base 0 remains unchanged (no count is performed because the clock is disabled) after CNT completes updating, so does time base 1.

Besides, to accurately control the counting of the first MCPWM period, it is recommended to update MCPWM_THx and MCPWM_CNTx simultaneously. But MCPWM_THx and MCPWM_CNTx can also be updated separately. Then start counting by writing through software or triggering by external event to set MCWPM_TCLk.BASE_CNT0_EN/MCWPM_TCLk.BASE_CNT1_EN, so as to enable the CNT counting of time base 0 and time base 1, respectively.

16.1.2 FAIL Signal Processing

The FAIL signal is an emergency stop signal, which is mainly used to quickly turn off the power tube when abnormality occurs, so as to avoid irreversible hardware damage. The signal processing module mainly realizes the rapid shutdown of the PWM output by setting emergency stop event in situations. There are four fail signal inputs to MCPWM, namely FAIL0 and FAIL3, which come from the chip IO MCPWM_BKIN [3:0] or the four signals from the output CMP [5:0] of the on-chip comparator.

Channels 0/1/2 of MCPWM0 uses CMP[1:0], and Channel 3 of MCPWM0 uses CMP[5:4]. Channels 0/1/2 of MCPWM1 uses CMP[3:2], and Channel 3 of MCPWM1 uses CMP[5:4].

Table 16-1 MCPWM FAIL signal allocation

	MCPWM0 CH0/1/2		MCPWM0 CH3		MCPWM1 CH0/1/2		MCPWM1 CH3	
	FAIL0	FAIL1	FAIL2	FAIL3	FAIL0	FAIL1	FAIL2	FAIL3
MCPWM0_BKIN0	√							
MCPWM0_BKIN1		√						
MCPWM0_BKIN2			√					
MCPWM0_BKIN3				√				
MCPWM1_BKIN0					√			
MCPWM1_BKIN1						√		
MCPWM1_BKIN2							√	
MCPWM1_BKIN3								√
CMP0	√							
CMP1		√						
CMP2					√			
CMP3						√		
CMP4			√				√	
CMP5				√				√

Among them, MCPWMx_FAIL012 is used to control the FAIL signal selection, polarity and enable of CH012. MCPWMx_FAIL3 is used to control the FAIL signal selection, polarity and enable of CH3.

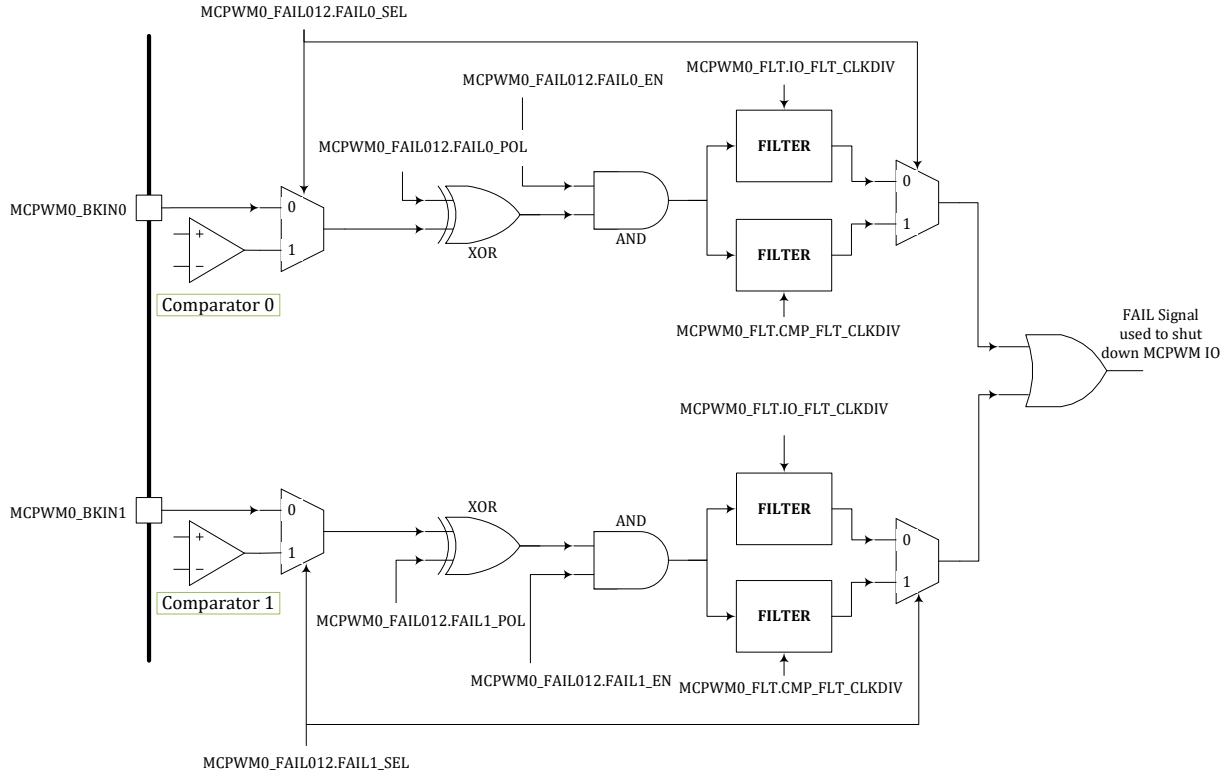


Figure 16-1 MCPWM0 FAIL0/1 Logic Diagram

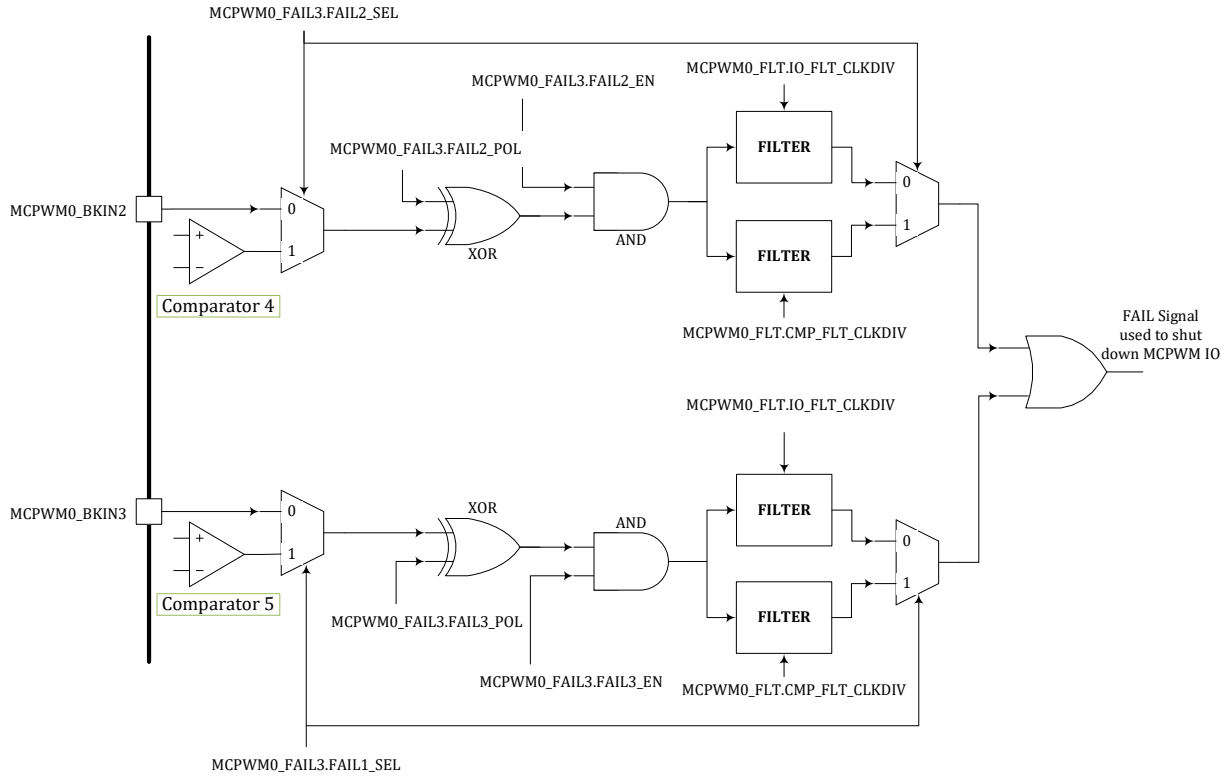


Figure 16-2 MCPWM0 FAIL2/3 Logic Diagram

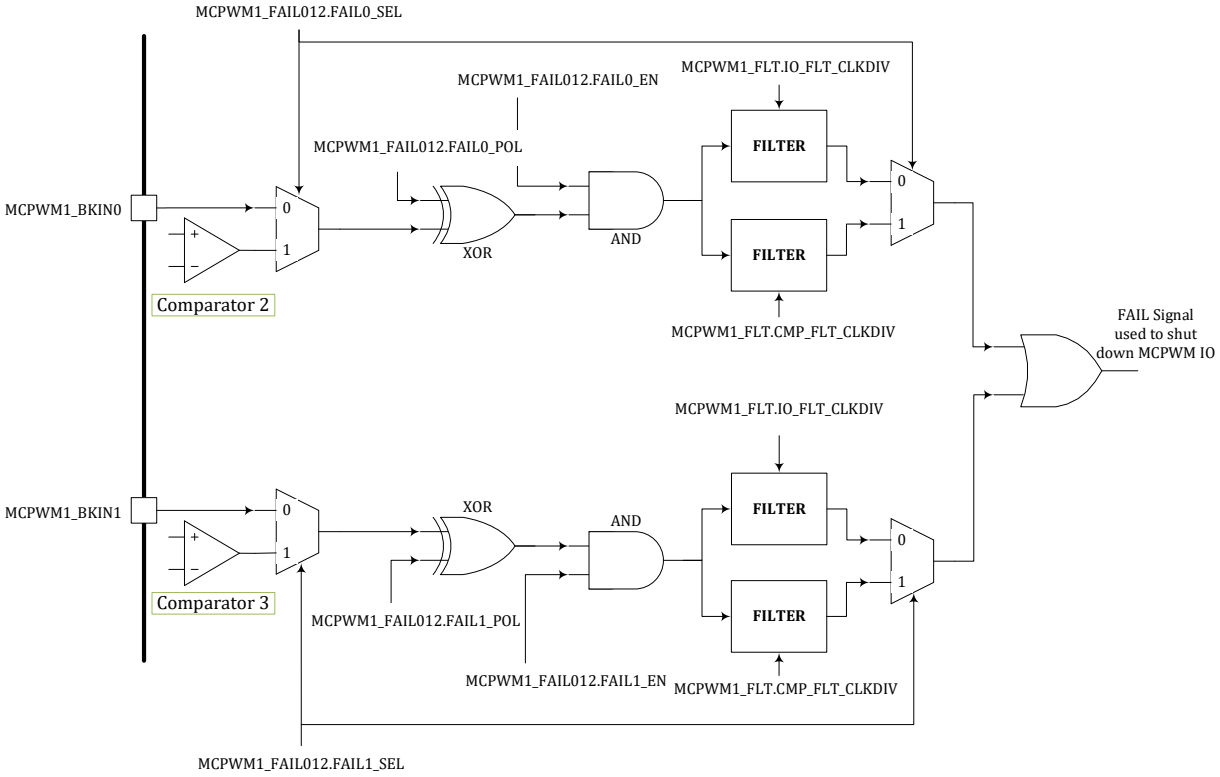


Figure 16-3 MCPWM1 FAIL0/1 Logic Diagram

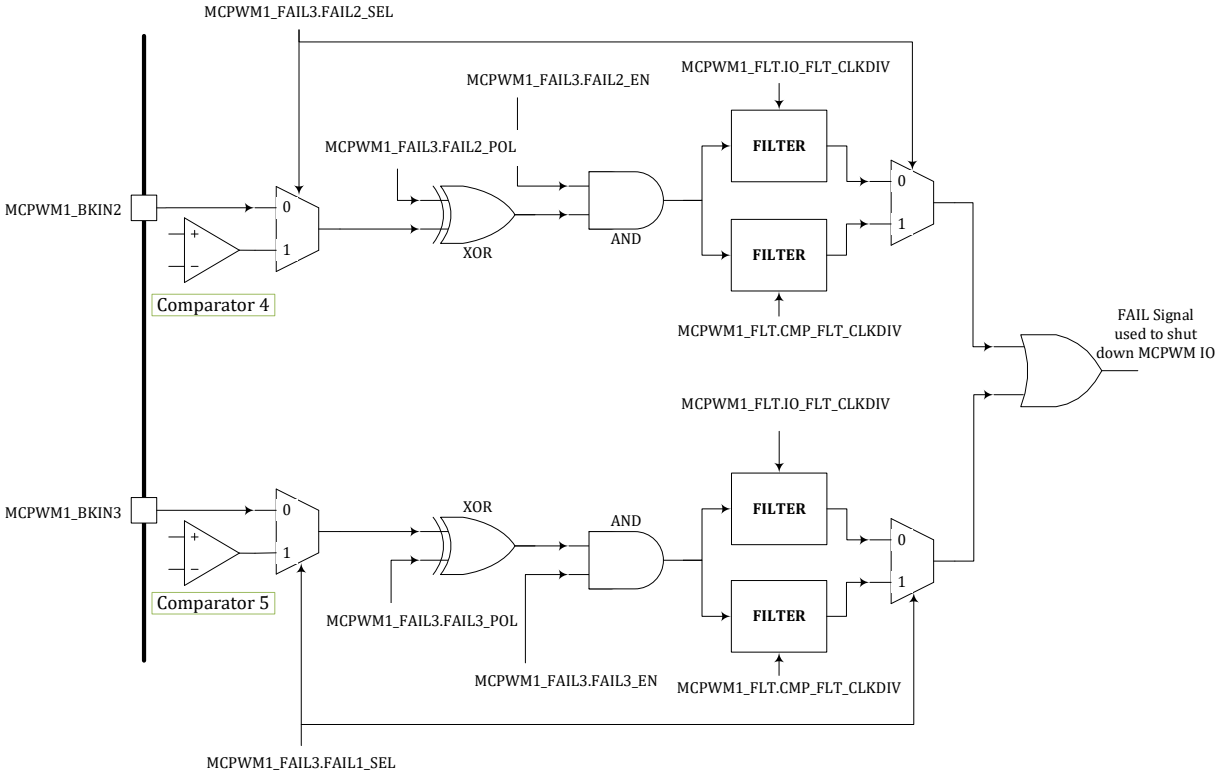


Figure 16-4 MCPWM1 FAIL2/3 Logic Diagram

The clock of the Filter module comes from the gated clock FCLK [10:9] of the system main clock MCLK, and is divided by two stages. The first-stage frequency division is controlled by MCPWM_TCLK.CLK_DIV, which divides by 1/2/4/8 times. The second-stage frequency division can achieve 1-16 times the frequency division. If the FAIL signal comes from MCPWM_BKIN [1:0], then use MCPWM_TCLK.IO_FLT_CLKDIV [3:0] as the second-stage frequency division coefficient; if the FAIL signal comes from the internal comparator output, then use MCPWM_TCLK.CMP_TCLK_CLKDIV [3:0] as the frequency division factor of the second stage, as shown in Figure 16-4.

The MCPWM module uses the frequency-divided clock to filter the Fail signal. The filter width is fixed at 16 cycles, that is, the input signal must be stable for at least 16 clock cycles (clock frequency divided by two) before the hardware determines it as a valid input signal. The formula for the filter time constant is as follows, where T_{MCLK} is the clock period of FCLK, 192MHz corresponds to 5.2ns. MCPWM_TCLK. FLT_CLKDIV may be MCPWM_TCLK.IO_FLT_CLKDIV or MCPWM_TCLK.CMP_FLT_CLKDIV depending on the configuration.

$$T = T_{FCLK} \times (MCPWM\_TCLK.CLK\_DIV) \times (MCPWM\_TCLK.FLT\_CLKDIV+1) \times 16$$

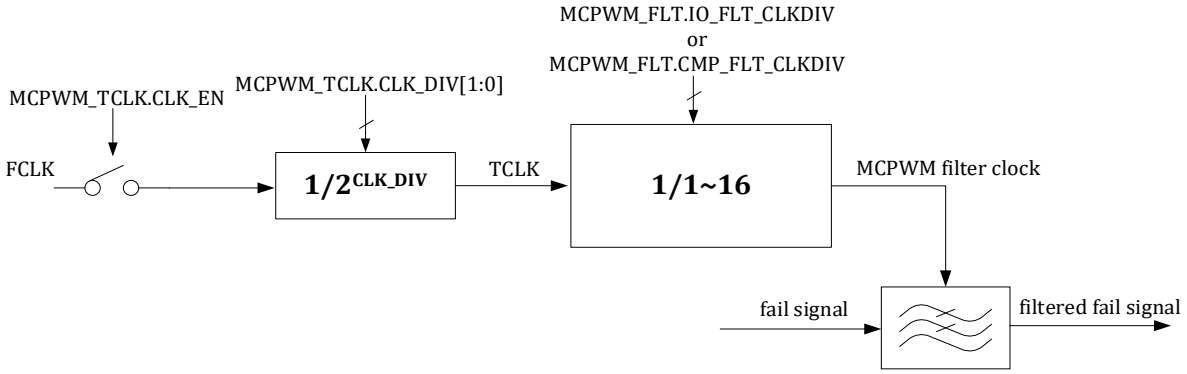


Figure 16-4 MCPWM Fail Signal Filtering Clock Generation Logic

Once a FAIL event occurs, the hardware forces the IO output to the default value of the fault specified in the MCPWM_FAIL.CHxN_DEFAULT and MCPWM_FAIL.CHxP_DEFAULT registers. After then, the values of MCPWM_FAIL.CHxN_DEFAULT and MCPWM_FAIL.CHxP_DEFAULT are directly output to the IO port, and are no longer affected by the polarity control such as MCPWM_FAIL.FAIL_POL. The default level of channels 0/1/2 is set by MCPWM_FAIL012, and of channel 3 set by MCPWM_FAIL3.

The MCPWM FAIL signal from the comparator is the original signal of analog comparator output, which is not filtered by the comparator digital interface module but can be window controlled by MCPWM channel signal. For settings of window control, refer to the comparator digital interface module. After the FAIL signal enters the MCPWM module, it can be filtered by MCPWM_TCLK.

16.1.3 MCPWM Special Output Status

All zero and all 1 output states are often used in motor control. The following complementary mode settings can get the desired output.

1. If THn0≥THn1, the chip is in a constant 0 state (CH<n>P off, CH<n>N on), no dead-zone
2. If THn0 = -TH, THn1 = TH, the chip is in a constant 1 state (CH<n>P is on, CH<n>P is off), no dead-zone

16.1.4 IO DRIVER Module

This module sets IO to the corresponding level according to the actual MCPWM register configuration. The overall data flow chart of the IO Driver module is as follows:

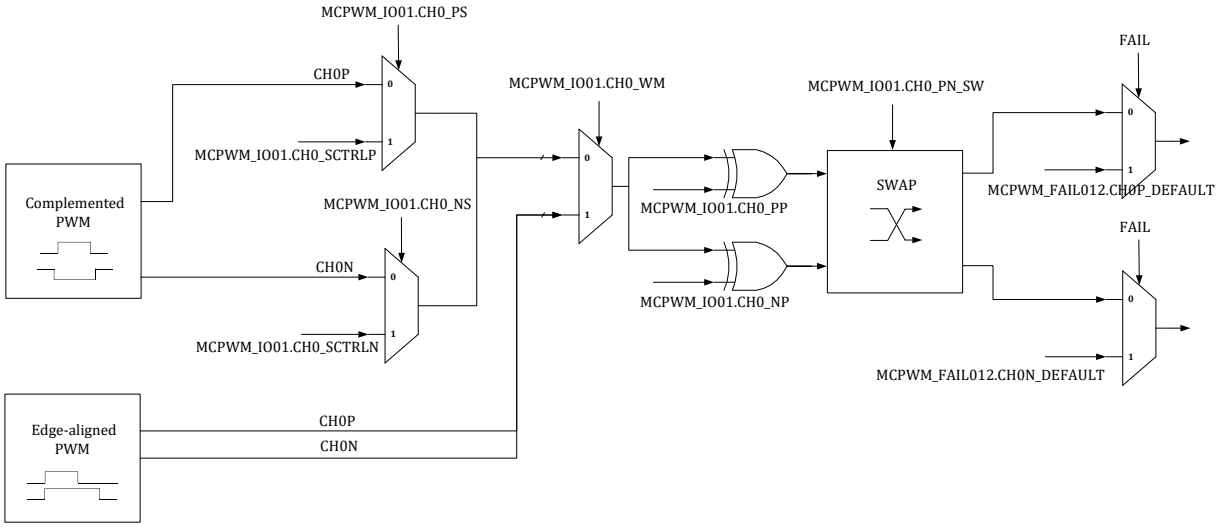


Figure 16-5 IO Driver Module Data Flow Chart

16.1.4.1 MCPWM Wave-form Output: Center-aligned Mode

The four MCPWM IO Drivers use independent control thresholds and independent dead-zone widths (each pair of complementary IO dead-zones need to be configured independently, that is, four dead-zone configuration registers) and share data update events.

TH<n>0 and TH<n>1 are used to control the start and shutdown of the<n>MCPWM channel IO, n is 1, 2, 3, and 4.

When the counter CNT counts up to TH<n>0, CH<n>N is turned off at time t3, and DTH<n>0 is delayed after dead-zone delay, and CH<n>P is turned on.

When the counter CNT counts up to TH<n>1, CH<n>P is turned off at time t4, and DTH<n>1 is delayed after dead-zone delay, and CH<n>N is turned on.

The independent startup and shutdown time control is adopted to provide phase control.

The dead-zone delay guarantees that CH<n>P/CH<n>N will not be high at the same time to avoid short circuit.

Both t3 and t4 will generate corresponding interrupts.

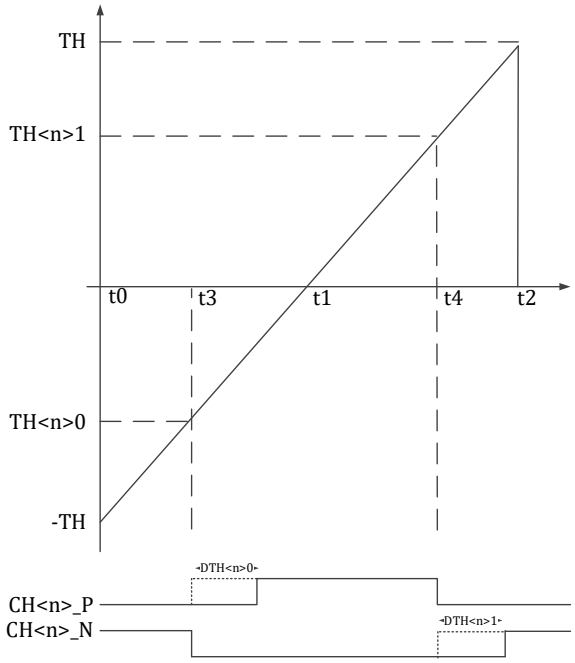


Figure 16-6 MCPWM Timing TH0 and TH1 - Complementary Mode

16.1.4.2 MCPWM Wave-form Control: Center-aligned Push-pull Mode

Complementary push-pull mode. In the first cycle, CH<n>P is turned on at time t3, and CH<n>P is turned off at time t4. In the second cycle, CH<n>N is turned on at time t3, and CH<n>N is turned off at time t4.

Both t3 and t4 will generate corresponding interrupts.

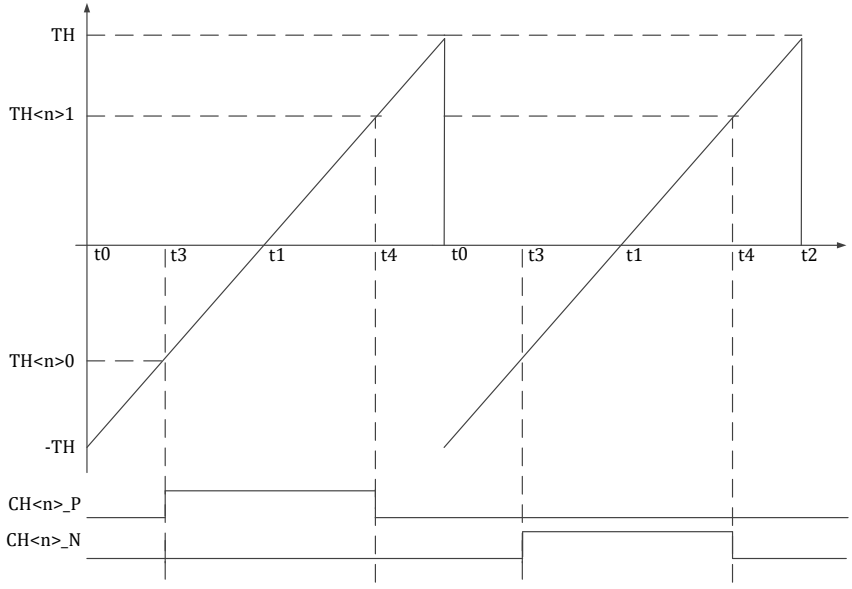


Figure 16-7 MCPWM Timing TH<n>0 and TH<n>1 - Center-aligned Push-pull Mode

16.1.4.3 MCPWM Wave-form Output: Edge-aligned Mode

In edge-aligned mode, CH<n>P and CH<n>N are turned on at time t0 at the same time, then CH<n>N is turned off at time t3, and CH<n>P is turned off at time t4.

Both t3 and t4 will generate corresponding interrupts.

In edge-aligned mode, CH<n>P and CH<n>N don't need dead-zone protection.

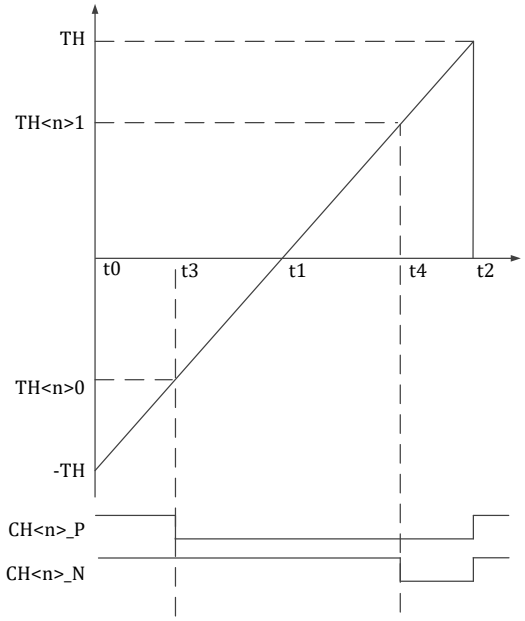


Figure 16-8 MCPWM Timing Edge-aligned Mode

16.1.4.4 MCPWM Wave-form Control: Edge-aligned Push-pull Mode

Edge-aligned push-pull mode. In the first cycle, CH<n>P is turned on at time t0, and CH<n>P is turned off at time t3. In the second cycle, CH<n>N is turned on 1 at time t0, and CH<n>N is turned off at time t3.

Both t0 and t3 will generate corresponding interrupts.

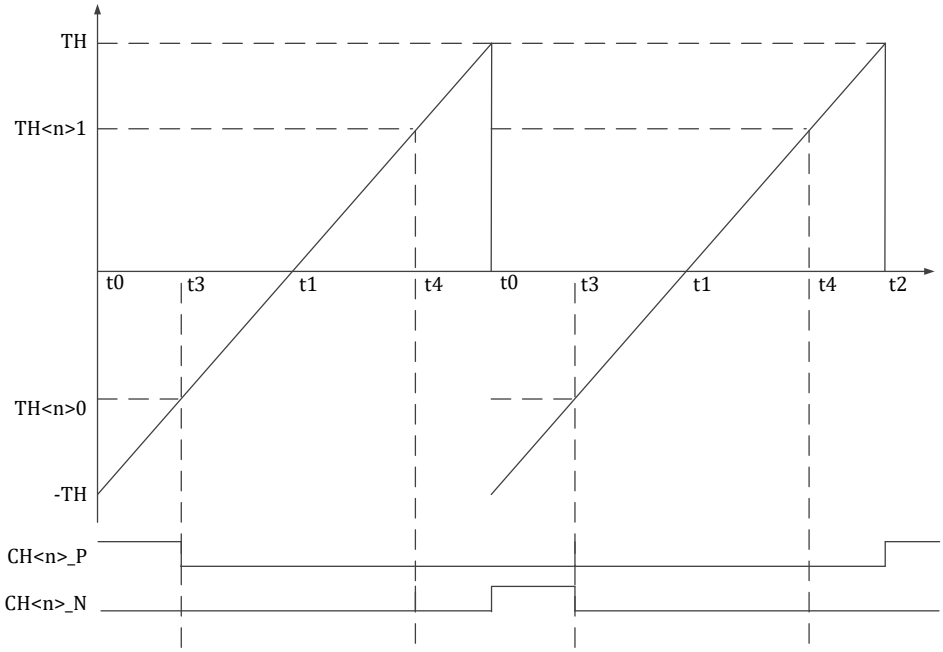


Figure 16-9 MCPWM Timing TH0 and TH1 Edge-aligned Push-pull Mode

16.1.4.5 MCPWM IO: Dead-zone Control

MCPWM IO is a pair of mutually exclusive control signals CH<n>P/CH<n>N, which controls the circuit shown in the figure below,

When CH<n>P is high and CH<n>N is low, Vout output is high (VDD);

When CH<n>P is low and CH<n>N is high, Vout output is low (VSS);

When CH<n>P is high and CH<n>N is high, Vout output is undefined, but a short circuit from VDD to VSS will occur accordingly;

When CH<n>P is low and CH<n>_is low, the Vout output is undefined.

It is necessary to avoid the situation where CH<n>P and CH<n>N are both high. The introduction of dead-zone can avoid the short circuit from VDD to VSS effectively.

Each channel deadband width of four groups of MCPWM IO supporting independent adjustment.

For complementary mode, MCPWM IO is automatically inserted into the dead-zone.

For edge-aligned mode, MCPWM IO has no dead-zone.

Added conflict detection for CH<n>P and CH<n>N in the IO Driver module. When a conflict occurs, it will pull IO low automatically and output an error interrupt (the error interrupt flag can be cleared by software or cleared automatically by hardware).

MCPWM IO can also be output by software configuration, that is, the dead-zone control is controlled by software. If the MCPWM is configured as center-aligned mode, the hardware short-circuit protection circuit is still valid to guarantee that P and N are not turned on at the same

time.

The position of CH<n>P and CH<n>N output to IO can be interchanged.

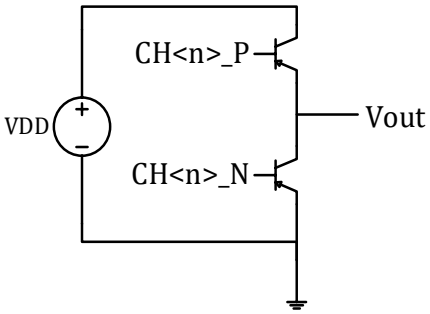


Figure 16-10 MCPWM IO Control Diagram

16.1.4.6 MCPWM IO: Polarity Setting

The effective levels of CH<n>P and CH<n>N can be set as high and low, and the effective level of each IO can be set individually. The position of CH<n>P and CH<n>N output to IO can be interchanged by software configuration.

16.1.4.7 MCPWM IO: Auto-protection

When an emergency stop event (Fail event) occurs, CH <n> P and CH <n> N should be switched to the off state automatically. Remember to turn off the level configuration (MCPWM_FAIL.CHxN_DEFAULT and MCPWM_FAIL.CHxP_DEFAULT control the default level). The default level of channels 0/1/2 is set by MCPWM_FAIL012, and of channel 3 set by MCPWM_FAIL3.

- After the chip works normally, the default output level of IO is the specified value of register MCPWM_FAIL.CHxN_DEFAULT and MCPWM_FAIL.CHxP_DEFAULT. When the user configuration is completed and MCPWM works normally, set MCPWM_FAIL.MCPWM_OE (ie MOE) to 1, and the IO output level is controlled by MCPWM IO drive module.
- When a FAIL short circuit condition occurs, the hardware switches to the IO default level immediately.
- When the chip is debugged, MCU Halt, it can be configured to stop MCPWM output, and output the default level value.

16.1.5 ADC Trigger Timer Module

MCPWM can provide ADC sampling sequence control. When the counter counts to MCPWM_TMR0/MCPWM_TMR1/MCPWM_TMR2/MCPWM_TMR3, a timing event can be generated to trigger ADC sampling. Besides, the trigger signal can be output to GPIO for debugging. For the specific GPIO output, please refer to the datasheet of the corresponding device. Among which, MCPWM_TMR0/ MCPWM_TMR1 must use the time base 0, and MCPWM_TMR2/ MCPWM_TMR3 can

use the time base 0/1, see 16.2. 33 MCPWM_TCLK.

Table 16-2 MCPWM Counter Threshold and Events

t0	-th
t1	0
tio0[0]	th00
tio0[1]	th01
TADC[0]	tmr0
TADC [1]	tmr1
TADC [2]	tmr2
TADC [3]	tmr3

16.1.6 Interruption

When MCPMW_IF0 and MCPWM_EIF [5:4] are set and enabled, MCPWMx_IRQ0 interrupt will be generated;

When MCPMW_IF1 and MCPWM_EIF [7:6] are set and enabled, MCPWMx_IRQ1 interrupt will be generated.

See 16.1.6 for details.

16.2 Register

16.2.1 Address Allocation

The base address of the MCPWM0 module register is 0x4001_3400.

The base address of the MCPWM1 module register is 0x4001_3800.

The implementation of MCPWM0 and MCPWM1 is identical, except for the FAIL signal source.

The register list is as follows:

Table 16-3 MCPWM Module Register List

Name	Offset Address	Description
MCPWMx_TH00	0x00	MCPWM CH0_P compare threshold register
MCPWMx_TH01	0x04	MCPWM CH0_N compare threshold register
MCPWMx_TH10	0x08	MCPWM CH1_P compare threshold register
MCPWMx_TH11	0x0C	MCPWM CH1_N compare threshold register
MCPWMx_TH20	0x10	MCPWM CH2_P compare threshold register
MCPWMx_TH21	0x14	MCPWM CH2_N compare threshold register
MCPWMx_TH30	0x18	MCPWM CH3_P compare threshold register
MCPWMx_TH31	0x1C	MCPWM CH3_N compare threshold register
MCPWMx_TMR0	0x20	Compare threshold 0 register for ADC sampling timer
MCPWMx_TMR1	0x24	Compare threshold 1 register for ADC sampling timer
MCPWMx_TMR2	0x28	Compare threshold 2 register for ADC sampling timer

MCPWMx_TMR3	0x2C	Compare threshold 3 register for ADC sampling timer
MCPWMx_TH0	0x30	MCPWM time base 0 threshold register
MCPWMx_TH1	0x34	MCPWM time base 1 threshold register
MCPWMx_CNT0	0x38	MCPWM time base 0 counter register
MCPWMx_CNT1	0x3C	MCPWM time base 1 counter register
MCPWMx_UPDATE	0x40	MCPWM load control register
MCPWMx_FCNT	0x44	CNT value at MCPWM FAIL
MCPWMx_EVT0	0x48	MCPWM time base 0 external trigger
MCPWMx_EVT1	0x4C	MCPWM time base 1 external trigger
MCPWMx_DTH00	0x50	MCPWM CH0 N channel dead-zone width control register
MCPWMx_DTH01	0x54	MCPWM CH0 P channel dead-zone width control register
MCPWMx_DTH10	0x58	MCPWM CH1 N channel dead-zone width control register
MCPWMx_DTH11	0x5C	MCPWM CH1 P channel dead-zone width control register
MCPWMx_DTH20	0x60	MCPWM CH2 N channel dead-zone width control register
MCPWMx_DTH21	0x64	MCPWM CH2 P channel dead-zone width control register
MCPWMx_DTH30	0x68	MCPWM CH3 N channel dead-zone width control register
MCPWMx_DTH31	0x6C	MCPWM CH3 P channel dead-zone width control register
MCPWMx_FLT	0x70	MCPWM filter clock divider register
MCPWMx_SDCFG	0x74	MCPWM load configuration register
MCPWMx_AUEN	0x78	MCPWM auto update enable register
MCPWMx_TCLK	0x7C	MCPWM clock divider control register
MCPWMx_IE0	0x80	Interrupt control register for MCPWM time base 0
MCPWMx_IF0	0x84	Interrupt flag register for MCPWM time base 0
MCPWMx_IE1	0x88	MCPWM interrupt control register
MCPWMx_IF1	0x8C	MCPWM interrupt flag register
MCPWMx_EIE	0x90	MCPWM abnormal interrupt control register
MCPWMx_EIF	0x94	MCPWM abnormal interrupt flag register
MCPWMx_RE	0x98	MCPWM DMA request enable register
MCPWMx_PP	0x9C	MCPWM push-pull mode enable register
MCPWMx_IO01	0xA0	MCPWM CH0 CH1 IO control register
MCPWMx_IO23	0xA4	MCPWM CH2 CH3 IO control register
MCPWMx_FAIL012	0xA8	MCPWM CH0 CH1 CH2 short circuit control register
MCPWMx_FAIL3	0xAC	MCPWM CH3 short circuit control register
MCPWMx_PRT	0xB0	MCPWM protection register

MCPWMx_CHMSK	0xB8	MCPWM channel masking bit register

Table 16-4 Registers Protected by MCPWM_PRT

Name	Offset Address	Description
MCPWMx_TH0	0x30	MCPWM threshold register
MCPWMx_TH1	0x34	MCPWM threshold register
MCPWMx_DTH00	0x50	MCPWM CH0 P channel dead-zone width control register
MCPWMx_DTH01	0x54	MCPWM CH0 N channel dead-zone width control register
MCPWMx_DTH10	0x58	MCPWM CH1 P channel dead-zone width control register
MCPWMx_DTH11	0x5C	MCPWM CH1 N channel dead-zone width control register
MCPWMx_DTH20	0x60	MCPWM CH2 P channel dead-zone width control register
MCPWMx_DTH21	0x64	MCPWM CH2 N channel dead-zone width control register
MCPWMx_DTH30	0x68	MCPWM CH3 P channel dead-zone width control register
MCPWMx_DTH31	0x6C	MCPWM CH3 N channel dead-zone width control register
MCPWMx_FLT	0x70	MCPWM filter clock divider register
MCPWMx_SDCFG	0x74	MCPWM load configuration register
MCPWMx_AUEN	0x78	MCPWM auto load enable register
MCPWMx_TCLK	0x7C	MCPWM clock divider control register
MCPWMx_IE0	0x80	Interrupt control register for MCPWM time base 0
MCPWMx_IE1	0x88	Interrupt control register for MCPWM time base 1
MCPWMx_EIE	0x90	MCPWM abnormal interrupt control register
MCPWMx_RE	0x98	MCPWM DMA request enable register
MCPWMx_PP	0x9C	MCPWM push-pull mode enable register
MCPWMx_IO01	0xA0	MCPWM CH0 CH1 IO control register
MCPWMx_IO23	0xA4	MCPWM CH2 CH3 IO control register
MCPWMx_FAIL012	0xA8	MCPWM CH0 CH1 CH2 short circuit control register
MCPWMx_FAIL3	0xAC	MCPWM CH3 short circuit control register
MCPWMx_CHMSK	0xB8	MCPWM channel masking bit register

Table 16-5 Registers with Shadow Registers

Name	Offset Address	Description
MCPWMx_TH00	0x00	MCPWM CH0_P compare threshold register

MCPWMx_TH01	0x04	MCPWM CH0_N compare threshold register
MCPWMx_TH10	0x08	MCPWM CH1_P compare threshold register
MCPWMx_TH11	0x0C	MCPWM CH1_N compare threshold register
MCPWMx_TH20	0x10	MCPWM CH2_P compare threshold register
MCPWMx_TH21	0x14	MCPWM CH2_N compare threshold register
MCPWMx_TH30	0x18	MCPWM CH3_P compare threshold register
MCPWMx_TH31	0x1C	MCPWM CH3_N compare threshold register
MCPWMx_TMR0	0x20	Compare threshold 0 register for ADC sampling timer
MCPWMx_TMR1	0x24	Compare threshold 1 register for ADC sampling timer
MCPWMx_TMR2	0x28	Compare threshold 2 register for ADC sampling timer
MCPWMx_TMR3	0x2C	Compare threshold 3 register for ADC sampling timer
MCPWMx_TH0	0x30	MCPWM time base 0 threshold register
MCPWMx_TH1	0x34	MCPWM time base 1 threshold register
MCPWMx_CNT0	0x38	MCPWM time base 0 counter register
MCPWMx_CNT1	0x3C	MCPWM time base 1 counter register

For all MCPWM configuration registers with shadow registers, the value is first written to the preload register, and then to the shadow register when an update event occurs, and the value of the shadow register is read when reading before or after loading.

16.2.2 MCPWMx_TH00(x = 0,1)

Unprotected register

The addresses are: 0x4001_3400 and 0x4001_3800.

Reset value: 0x0

Table 16-6 MCPWM_TH00 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TH00															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	TH00	MCPWM CH0_P comparison threshold, 16-bit signed number; after an update event occurs, this register is loaded into the MCPWM operating system.

16.2.3 MCPWMx_TH01(x = 0,1)

Unprotected register

The addresses are: 0x4001_3404 and 0x4001_3804.

Reset value: 0x0

Table 16-7 MCPWM_TH01 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TH01															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	TH01	MCPWM CH0_N comparison threshold, 16-bit signed number; after an update event occurs, this register is loaded into the MCPWM operating system.

16.2.4 MCPWMx_TH10(x = 0,1)

Unprotected register

The addresses are: 0x4001_3408 and 0x4001_3808.

Reset value: 0x0

Table 16-8 MCPWM_TH10 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TH10															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
15:0]	TH10	MCPWM CH1_P comparison threshold, 16-bit signed number; after an update event occurs, this register is loaded into the MCPWM operating system.

16.2.5 MCPWMx_TH11(x = 0,1)

Unprotected register

The addresses are: 0x4001_340C and 0x4001_380C.

Reset value: 0x0

Table 16-9 MCPWM_TH11 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TH11															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	TH11	MCPWM CH1_N comparison threshold, 16-bit signed number; after an update event occurs, this register is loaded into the MCPWM operating system.

16.2.6 MCPWMx_TH20(x = 0,1)

Unprotected register

The addresses are: 0x4001_3410 and 0x4001_3810.

Reset value: 0x0

Table 16-10MCPWM_TH20 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TH20															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	TH20	MCPWM CH2_P comparison threshold, 16-bit signed number; after an update event occurs, this register is loaded into the MCPWM operating system.

16.2.7 MCPWMx_TH21(x = 0,1)

Unprotected register

The addresses are: 0x4001_3414 and 0x4001_3814.

Reset value: 0x0

Table 16-11MCPWM_TH21 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TH21															

RW
0

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	TH21	MCPWM CH2_N comparison threshold, 16-bit signed number; after an update event occurs, this register is loaded into the MCPWM operating system.

16.2.8 MCPWMx_TH30(x = 0,1)

Unprotected register

The addresses are: 0x4001_3418 and 0x4001_3818.

Reset value: 0x0

Table 16-12MCPWM_TH30 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TH30															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	TH30	MCPWM CH3_P comparison threshold, 16-bit signed number; after an update event occurs, this register is loaded into the MCPWM operating system.

16.2.9 MCPWMx_TH31(x = 0,1)

Unprotected register

The addresses are: 0x4001_341C and 0x4001_381C.

Reset value: 0x0

Table 16-13MCPWM_TH31 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TH31															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	TH31	MCPWM CH3_N comparison threshold, 16-bit signed number; after an update event occurs, this register is loaded into the MCPWM operating system.

16.2.10 MCPWMx_TMR0(x = 0,1)

Unprotected register

The addresses are: 0x4001_3420 and 0x4001_3820.

Reset value: 0x0

Table 16-14MCPWM_TMR0 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TMR0															
RW															
0x7FFF															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	TMR0	Compare threshold 0 register for ADC sampling timer, 16-bit signed number; when MCPWM_CNT0=TMR0, a TADC[0] event occurs to trigger ADC for sampling. After an update event occurs, this register is loaded into the MCPWM operating system.

16.2.11 MCPWMx_TMR1(x = 0,1)

Unprotected register

The addresses are: 0x4001_3424 and 0x4001_3824.

Reset value: 0x0

Table 16-15 MCPWM_TMR1 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TMR1															
RW															
0x7FFF															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	TMR1	Compare threshold 1 register for ADC sampling timer, 16-bit signed

		number; when MCPWM_CNT=TMR1, a TADC[1] event occurs to trigger ADC for sampling. After an update event occurs, this register is loaded into the MCPWM operating system.

16.2.12 MCPWMx_TMR2(x = 0,1)

Unprotected register

The addresses are: 0x4001_3428 and 0x4001_3828.

Reset value: 0x0

Table 16-16 MCPWM_TMR2 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TMR2															
RW															
0x7FFF															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	TMR2	Compare threshold 2 register for ADC sampling timer, 16-bit signed number; after an update event occurs, this register is loaded into the MCPWM operating system.

16.2.13 MCPWMx_TMR3(x = 0,1)

Unprotected register

The addresses are: 0x4001_342C and 0x4001_382C.

Reset value: 0x0

Table 16-17MCPWM_TMR3 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TMR3															
RW															
0x7FFF															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	TMR3	Compare threshold 3 register for ADC sampling timer, 16-bit signed number; after an update event occurs, this register is loaded into the MCPWM operating system.

16.2.14 MCPWMx_TH0(x = 0,1)

Write-protected register

The addresses are: 0x4001_3430 and 0x4001_3830.

Reset value: 0x0

Table 16-18 Time Base 0 Register (MCPWM_TH0)

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

Bit field	Bit Name	Description
[31:15]		Unused
[14:0]	TH	MCPWM time base 0 counter threshold, 15-bit unsigned number, counting from -TH to TH by time base 0 counter in MCPWM operating system; after an update event occurs, this register is loaded into the MCPWM operating system.

16.2.15 MCPWMx_TH1(x = 0,1)

Write-protected register

The addresses are: 0x4001_3434 and 0x4001_3834.

Reset value: 0x0

Table 16-19 Time Base 1 Register (MCPWM_TH1)

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

Bit field	Bit Name	Description
[31:15]		Unused
[14:0]	TH	MCPWM time base 1 counter threshold, 15-bit unsigned number, counting from -TH to TH by time base 1 counter in MCPWM operating system; after an update event occurs, this register is loaded into the MCPWM operating system.

16.2.16 MCPWMx_CNT0(x = 0,1)

Unprotected register

The addresses are: 0x4001_3438 and 0x4001_3838.

Reset value: 0x0

Table 16-20 MCPWM_CNT0 Register

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

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	CNT	Write to the register, and update the set value of time base 0 counter; after an update event occurs, this register is loaded into the time base 0 CNT of MCPWM operating system. The read value is the value of time base 0 counter in MCPWM operating system, and the value range is -TH to +TH.

16.2.17 MCPWMx_CNT1(x = 0,1)

Unprotected register

The addresses are: 0x4001_343C and 0x4001_383C.

Reset value: 0x0

Table 16-21 MCPWM_CNT1 Register

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

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	CNT	Write to the register, and update the set value of time base 1 counter; after an update event occurs, this register is loaded into the time base 1 CNT of MCPWM operating system. The read value is the value of time base 1 counter in MCPWM operating system, and the value range is -TH to +TH.

16.2.18 MCPWMx_UPDATE(x = 0,1)

Unprotected register

The addresses are: 0x4001_3440 and 0x4001_3840.

Reset value: 0x0

Table 16-22 MCPWM Manual Update Register (MCPWM_UPDATE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT1_UPDATE	CNT0_UPDATE	TH1_UPDATE	TH0_UPDATE	TMR3_UPDATE	TMR2_UPDATE	TMR1_UPDATE	TMR0_UPDATE	TH31_UPDATE	TH30_UPDATE	TH21_UPDATE	TH20_UPDATE	TH11_UPDATE	TH10_UPDATE	TH01_UPDATE	TH00_UPDATE
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:16]		Unused
[15]	CNT1_UPDATE	Manually load the contents of the MCPWM_CNT1 register into the MCPWM operating system. 1: Load; 0: Not load.
[14]	CNT0_UPDATE	Manually load the contents of the MCPWM_CNT0 register into the MCPWM operating system. 1: Load; 0: Not load.
[13]	TH1_UPDATE	Manually load the contents of the MCPWM_TH1 register into the MCPWM operating system. 1: Load; 0: Not load.
[12]	TH0_UPDATE	Manually load the contents of the MCPWM_TH0 register into the MCPWM operating system. 1: Load; 0: Not load.
[11]	TMR3_UPDATE	Manually load the contents of the MCPWM_TMR3 register into the MCPWM operating system. 1: Load; 0: Not load.
[10]	TMR2_UPDATE	Manually load the contents of the MCPWM_TMR2 register into the MCPWM operating system. 1: Load; 0: Not load.
[9]	TMR1_UPDATE	Manually load the contents of the MCPWM_TMR1 register into the MCPWM operating system. 1: Load; 0: Not load.
[8]	TMR0_UPDATE	Manually load the contents of the MCPWM_TMR0 register into the MCPWM operating system. 1: Load; 0: Not load.
[7]	TH31_UPDATE	Manually load the contents of the MCPWM_TH31 register into the

		MCPWM operating system. 1: Load; 0: Not load.
[6]	TH30_UPDATE	Manually load the contents of the MCPWM_TH30 register into the MCPWM operating system. 1: Load; 0: Not load.
[5]	TH21_UPDATE	Manually load the contents of the MCPWM_TH21 register into the MCPWM operating system. 1: Load; 0: Not load.
[4]	TH20_UPDATE	Manually load the contents of the MCPWM_TH20 register into the MCPWM operating system. 1: Load; 0: Not load.
[3]	TH11_UPDATE	Manually load the contents of the MCPWM_TH11 register into the MCPWM operating system. 1: Load; 0: Not load.
[2]	TH10_UPDATE	Manually load the contents of the MCPWM_TH10 register into the MCPWM operating system. 1: Load; 0: Not load.
[1]	TH01_UPDATE	Manually load the contents of the MCPWM_TH01 register into the MCPWM operating system. 1: Load; 0: Not load.
[0]	TH00_UPDATE	Manually load the contents of the MCPWM_TH00 register into the MCPWM operating system. 1: Load; 0: Not load.

Write 1 to MCPWM_UPDATE bit can trigger to write the value from the preload register to the shadow register, and MCPWM_UPDATE is automatically cleared. Every time 1 is written, a software/manual trigger is performed. Writing 1 to MCPWM_UPDATE[15:14] will cause MCPWM_CNT1/MCPWM_CNT0 to be updated to the preload value. Please note whether CNT value needs to be updated in application.

Writing 0 to MCPWM_UPDATE[15:14] has no effect, but writing 0 immediately after writing 1 may cause the shadow register to fail to update. Therefore, writing 0 to MCPWM_UPDATE is not recommended.

16.2.19 MCPWMx_FCNT(x = 0,1)

Unprotected register

The addresses are: 0x4001_3444 and 0x4001_3844.

Reset value: 0x0

Table 16-23 MCPWM_FCNT Register

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

RW
0

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	FCNT	If MCPWM_FAIL012[15]=1, when a fail0/1 event occurs, record the MCPWM_CNT0 value and store it to MCPWM_FCNT; if MCPWM_FAIL3[15]=1, when a fail2/3 event occurs, record the MCPWM_CNT0 value and store it to MCPWM_FCNT.

16.2.20 MCPWMx_EVT0(x = 0,1)

Unprotected register

The addresses are: 0x4001_3448 and 0x4001_3848.

Reset value: 0x0

Table 16-24 MCPWM Time Base 0 External Trigger Register (MCPWM_EVT0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMER3_CMP1	TIMER3_CMP0	TIMER2_CMP1	TIMER2_CMP0	TIMER1_CMP1	TIMER1_CMP0	TIMER0_CMP1	TIMER0_CMP0	MCPWM1_TMR3	MCPWM1_TMR2	MCPWM1_TMR1	MCPWM1_TMR0	MCPWM0_TMR3	MCPWM0_TMR2	MCPWM0_TMR1	MCPWM0_TMR0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:16]		Unused
[15]	TIMER3_CMP1	Time base 0 starts counting when a TIMER3 CMP1 event occurs
[14]	TIMER3_CMP0	Time base 0 starts counting when a TIMER3 CMP0 event occurs
[13]	TIMER2_CMP1	Time base 0 starts counting when a TIMER2 CMP1 event occurs
[12]	TIMER2_CMP0	Time base 0 starts counting when a TIMER2 CMP0 event occurs
[11]	TIMER1_CMP1	Time base 0 starts counting when a TIMER1 CMP1 event occurs
[10]	TIMER1_CMP0	Time base 0 starts counting when a TIMER1 CMP0 event occurs
[9]	TIMER0_CMP1	Time base 0 starts counting when a TIMER0 CMP1 event occurs
[8]	TIMER0_CMP0	Time base 0 starts counting when a TIMER0 CMP0 event occurs
[7]	MCPWM1_TMR3	Time base 0 starts counting when a MCPWM1 TMR3 event occurs
[6]	MCPWM1_TMR2	Time base 0 starts counting when a MCPWM1 TMR2 event occurs
[5]	MCPWM1_TMR1	Time base 0 starts counting when a MCPWM1 TMR1 event occurs
[4]	MCPWM1_TMR0	Time base 0 starts counting when a MCPWM1 TMR0 event occurs
[3]	MCPWM0_TMR3	Time base 0 starts counting when a MCPWM0 TMR3 event occurs

[2]	MCPWM0_TMR2	Time base 0 starts counting when a MCPWM0 TMR2 event occurs
[1]	MCPWM0_TMR1	Time base 0 starts counting when a MCPWM0 TMR1 event occurs
[0]	MCPWM0_TMR0	Time base 0 starts counting when a MCPWM0 TMR0 event occurs

It is not recommended to use Timer4 comparison event as an external trigger source for MCPWM base 0.

16.2.21 MCPWMx_EVT1(x = 0,1)

Unprotected register

The addresses are: 0x4001_344C and 0x4001_384C.

Reset value: 0x0

Table 16-25 MCPWM Time Base 1 External Trigger Register (MCPWM_EVT1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMER3_CMP1	TIMER3_CMP0	TIMER2_CMP1	TIMER2_CMP0	TIMER1_CMP1	TIMER1_CMP0	TIMER0_CMP1	TIMER0_CMP0	MCPWM1_TMR3	MCPWM1_TMR2	MCPWM1_TMR1	MCPWM1_TMR0	MCPWM0_TMR3	MCPWM0_TMR2	MCPWM0_TMR1	MCPWM0_TMR0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:16]		Unused
[15]	TIMER3_CMP1	Time base 1 starts counting when a TIMER3 CMP1 event occurs
[14]	TIMER3_CMP0	Time base 1 starts counting when a TIMER3 CMP0 event occurs
[13]	TIMER2_CMP1	Time base 1 starts counting when a TIMER2 CMP1 event occurs
[12]	TIMER2_CMP0	Time base 1 starts counting when a TIMER2 CMP0 event occurs
[11]	TIMER1_CMP1	Time base 1 starts counting when a TIMER1 CMP1 event occurs
[10]	TIMER1_CMP0	Time base 1 starts counting when a TIMER1 CMP0 event occurs
[9]	TIMER0_CMP1	Time base 1 starts counting when a TIMER0 CMP1 event occurs
[8]	TIMER0_CMP0	Time base 1 starts counting when a TIMER0 CMP0 event occurs
[7]	MCPWM1_TMR3	Time base 1 starts counting when a MCPWM1 TMR3 event occurs
[6]	MCPWM1_TMR2	Time base 1 starts counting when a MCPWM1 TMR2 event occurs
[5]	MCPWM1_TMR1	Time base 1 starts counting when a MCPWM1 TMR1 event occurs
[4]	MCPWM1_TMR0	Time base 1 starts counting when a MCPWM1 TMR0 event occurs
[3]	MCPWM0_TMR3	Time base 1 starts counting when a MCPWM0 TMR3 event occurs
[2]	MCPWM0_TMR2	Time base 1 starts counting when a MCPWM0 TMR2 event occurs
[1]	MCPWM0_TMR1	Time base 1 starts counting when a MCPWM0 TMR1 event occurs
[0]	MCPWM0_TMR0	Time base 1 starts counting when a MCPWM0 TMR0 event occurs

It is not recommended to use Timer4 comparison event as an external trigger source for MCPWM base 1.

16.2.22 MCPWMx_DTH00(x = 0,1)

Write-protected register

The addresses are: 0x4001_3450 and 0x4001_3850.

Reset value: 0x0

Table 16-26 MCPWM_DTH00 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTH00															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	DTH00	MCPWM CH0 P channel dead-zone width control register, 10-bit unsigned number

16.2.23 MCPWMx_DTH01(x = 0,1)

Write-protected register

The addresses are: 0x4001_3454 and 0x4001_3854.

Reset value: 0x0

Table 16-27 MCPWM_DTH01 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTH01															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	DTH01	MCPWM CH0 N channel dead-zone width control register, 10-bit unsigned number

16.2.24 MCPWMx_DTH10(x = 0,1)

Write-protected register

The addresses are: 0x4001_3458 and 0x4001_3858.

Reset value: 0x0

Table 16-28 MCPWM_DTH10 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTH10															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	DTH10	MCPWM CH1 P channel dead-zone width control register, 10-bit unsigned number

16.2.25 MCPWMx_DTH11(x = 0,1)

Write-protected register

The addresses are: 0x4001_345C and 0x4001_385C.

Reset value: 0x0

Table 16-29 MCPWM_DTH11 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTH11															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	DTH11	MCPWM CH1 N channel dead-zone width control register, 10-bit unsigned number

16.2.26 MCPWMx_DTH20(x = 0,1)

Write-protected register

The addresses are: 0x4001_3460 and 0x4001_3860.

Reset value: 0x0

Table 16-30MCPWM_DTH20 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTH20															

RW
0

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	DTH20	MCPWM CH2 P channel dead-zone width control register, 10-bit unsigned number

16.2.27 MCPWMx_DTH21(x = 0,1)

Write-protected register

The addresses are: 0x4001_3464 and 0x4001_3864.

Reset value: 0x0

Table 16-31 MCPWM_DTH21 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTH21															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	DTH21	MCPWM CH2 N channel dead-zone width control register, 10-bit unsigned number

16.2.28 MCPWMx_DTH30(x = 0,1)

Write-protected register

The addresses are: 0x4001_3468 and 0x4001_3868.

Reset value: 0x0

Table 16-32 MCPWM_DTH30 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTH30															
RW															
0															

Bit field	Bit Name	Description

[31:16]		Unused
[15:0]	DTH30	MCPWM CH3 P channel dead-zone width control register, 10-bit unsigned number

16.2.29 MCPWMx_DTH31(x = 0,1)

Write-protected register

The addresses are: 0x4001_346C and 0x4001_386C.

Reset value: 0x0

Table 16-33 MCPWM_DTH31 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTH31															
RW															
0															

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	DTH31	MCPWM CH3 N channel dead-zone width control register, 10-bit unsigned number

16.2.30 MCPWMx_FLT(x = 0,1)

Write-protected register

The addresses are: 0x4001_3470 and 0x4001_3870.

Reset value: 0x0

Table 16-34 MCPWM Filter Clock Divider Register (MCPWM_FLT)

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

Bit field	Bit Name	Description
[31:16]		Unused
[15:8]	CMP_FLT_CLKDIV	The filter clock divider register output by the comparator is divided based on the system clock and affects MCPWM_FAIL [3:0]. The formula is as follows: System clock / (CMP_FLT_CLKDIV + 1). The frequency division

		range is 1 to 256.
[7:0]	IO_FLT_CLKDIV	The filter clock divider register of the GPIO input is divided based on the system clock, and affects MCPWM_FAIL [3: 0]. The formula is as follows: System clock / (IO_FLT_CLKDIV + 1). The frequency division range is 1 to 256.

MCPWM uses the clock after frequency division to filter the FAIL signal fixedly for 16 cycles. When 256 frequency division is used, the filter width is 4096 FCLK clock widths. FCLK is the system high-speed clock, which is generally the 192MHz PLL clock.

16.2.31 MCPWMx_SDCFG(x = 0,1)

Write-protected register

The addresses are: 0x4001_3474 and 0x4001_3874.

Reset value: 0x0

Table 16-35 MCPWM_SDCFG Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TR1_AEC	TR1_T1_UEN	TR1_T0_UEN	TR1_UP_INTV					TR0_AEC	TR0_T1_UEN	TR0_T0_UEN	TR0_UP_INTV			
	RW	RW	RW	RW					RW	RW	RW	RW			
	0	0	0	0					0	0	0	0			

Bit field	Bit Name	Description
[31:15]		Unused
[14]	TR1_AEC	Whether the update event automatically clears MCPWM_EIF [7:6] and sets MCPWM_FAIL3.MOE to restore MCPWM channel 3 signal output. 1: Enable automatic fault clearing function; 0: Disable automatic fault clearing function.
[13]	TR1_T1_UEN	Time base 1 t1 (over-zero) event update is enabled. 1: Enable; 0, disable.
[12]	TR1_T0_UEN	Time base 1 t0 (starting point) event update enable. 1: Enable; 0, disable.
[11:8]	TR1_UP_INTV	Time base 1 update interval. Once the number of t0 and t1 events is equal to TR1_UP_INTV, the MCPWM system triggers the operation of the MCPWM_TH1, TH30, TH31 and MCPWM_TMR registers automatically, and loaded into the MCPWM operating system. If both TR1_T1_UEN and TR1_T0_UEN are closed, this type of loading will not be triggered, and the loading can only be triggered manually.
[7]		Unused

[6]	TR0_AEC	Whether the update event automatically clears MCPWM_EIF [5:4] and sets MCPWM_FAIL012.MOE to restore MCPWM channel 0/1/2 signal output. 1: Enable automatic fault clearing function; 0: Disable automatic fault clearing function.
[5]	TR0_T1_UEN	Time base 0 t1 (over-zero) event update is enabled. 1: Enable; 0, disable.
[4]	TR0_T0_UEN	Time base 0 t0 (starting point) event update enable. 1: Enable; 0, disable.
[3:0]	TR0_UP_INTV	Time base 0 update interval. Once the number of t0 and t1 events is equal to TR0_UP_INTV+1, the MCPWM system triggers the operation of the MCPWM_TH0, MCPWM_TH00-TH21 and MCPWM_TMR registers automatically, and loaded into the MCPWM operating system. If both TR0_T1_UEN and TR0_T0_UEN are closed, this type of loading will not be triggered, and the loading can only be triggered manually.

16.2.32 MCPWMx_AUEN(x = 0,1)

Write-protected register

The addresses are: 0x4001_3478 and 0x4001_3878.

Reset value: 0x0

Table 16-36 MCPWM Auto Update Enable Register (MCPWM_AUEN)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT1_AUPDATE	CNT0_AUPDATE	TH1_AUPDATE	TH0_AUPDATE	TMR3_AUPDATE	TMR2_AUPDATE	TMR1_AUPDATE	TMR0_AUPDATE	TH31_AUPDATE	TH30_AUPDATE	TH21_AUPDATE	TH20_AUPDATE	TH11_AUPDATE	TH10_AUPDATE	TH01_AUPDATE	TH00_AUPDATE
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:16]		Unused
[15]	CNT1_AUPDATE	MCPWM_CNT1 auto load enable. 1: Load; 0: Not load.
[14]	CNT0_AUPDATE	MCPWM_CNT0 auto load enable. 1: Load; 0: Not load.
[13]	TH1_AUPDATE	MCPWM_TH1 auto load enable. 1: Load; 0: Not load.
[12]	TH0_AUPDATE	MCPWM_TH0 auto load enable. 1: Load; 0: Not load.
[11]	TMR3_AUPDATE	MCPWM_TMR3 auto load enable. 1: Load; 0: Not load.
[10]	TMR2_AUPDATE	MCPWM_TMR2 auto load enable. 1: Load; 0: Not load.
[9]	TMR1_AUPDATE	MCPWM_TMR1 auto load enable. 1: Load; 0: Not load.

[8]	TMR0_AUPDATE	MCPWM_TMR0 auto load enable. 1: Load; 0: Not load.
[7]	TH31_AUPDATE	MCPWM_TH31 auto load enable. 1: Load; 0: Not load.
[6]	TH30_AUPDATE	MCPWM_TH30 auto load enable. 1: Load; 0: Not load.
[5]	TH21_AUPDATE	MCPWM_TH21 auto load enable. 1: Load; 0: Not load.
[4]	TH20_AUPDATE	MCPWM_TH20 auto load enable. 1: Load; 0: Not load.
[3]	TH11_AUPDATE	MCPWM_TH11 auto load enable. 1: Load; 0: Not load.
[2]	TH10_AUPDATE	MCPWM_TH10 auto load enable. 1: Load; 0: Not load.
[1]	TH01_AUPDATE	MCPWM_TH01 auto load enable. 1: Load; 0: Not load.
[0]	TH00_AUPDATE	MCPWM_TH00 auto load enable. 1: Load; 0: Not load.

16.2.33 MCPWMx_TCLK(x = 0,1)

Write-protected register

The addresses are: 0x4001_349C and 0x4001_389C.

Reset value: 0x0

Table 16-37 MCPWM_TCLK Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						EVT_CNT1_EN	EVT_CNT0_EN	BASE_CNT1_EN	BASE_CNT0_EN	TMR3_TB	TMR2_TB		CLK_EN	CLK_DIV	
						RW	RW	RW	RW	RW	RW		RW	RW	
						0	0	0	0	0	0		0	0	

Bit field	Bit Name	Description
[31:10]		Unused
[9]	EVT_CNT1_EN	Time base 1 external trigger enable
[8]	EVT_CNT0_EN	Time base 0 external trigger enable
[7]	BASE_CNT1_EN	MCPWM time base 1 counter enable switch. 1: Enable; 0: Disable.
[6]	BASE_CNT0_EN	MCPWM time base 0 counter enable switch. 1: Enable; 0: Disable.
[5]	TMR3_TB	TMR3 time base selection, 0: time base 0, 1: time base 1
[4]	TMR2_TB	TMR2 time base selection, 0: time base 0, 1: time base 1
[3]		Unused
[2]	CLK_EN	MCPWM working clock enable. 1: Enable; 0: Disable.
[1:0]	CLK_DIV	MCPWM working clock divider register. 0: system clock 1: system clock/2 2: system clock/4 3: system clock/8

The shadow register can be updated only after enabling MCPWM_TCLK.CLK_EN. After

configuring the shadow register, start MCPWM_TCLK.BASE_CNT0/1_EN to make the time base 0 and time base 1 start counting synchronously. If TH0 and TH1 are set to the same value, the time base 0 and time base 1 must be at the same frequency.

When using external trigger MCPWM to start counting, you need to configure BASE_CNTx_EN to 0, EVT_CNTx_EN to 1, and set MCPWM_EVTx to select the appropriate external trigger source. After a trigger event occurs, BASE_CNTx_EN will be set to 1 by hardware.

16.2.34 MCPWMx_IE0(x = 0,1)

Write-protected register

The addresses are: 0x4001_3480 and 0x4001_3880.

Reset value: 0x0

Table 16-38 MCPWM Time Base 0 Interrupt Control Register (MCPWM_IE0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	UP_IE	TMR3_IE	TMR2_IE	TMR1_IE	TMR0_IE			TH21_IE	TH20_IE	TH11_IE	TH10_IE	TH01_IE	TH00_IE	T1_IE	T0_IE
	RW	RW	RW	RW	RW			RW	RW	RW	RW	RW	RW	RW	RW
	0	0	0	0	0			0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:15]		Unused
[14]	UP_IE	MCPWM_TH/MCPWM_TH00 - MCPWM_TH31/MCPWM_TMR0 - MCPWM_TMR3 and other registers are updated to enable interrupt source. 1: Enable; 0: Disable.
[13]	TMR3_IE	The count value of the counter in the MCPWM operating system is equal to MCPWM_TMR3 interrupt source enable. 1: Enable; 0: Disable.
[12]	TMR2_IE	The count value of the counter in the MCPWM operating system is equal to MCPWM_TMR2 interrupt source enable. 1: Enable; 0: Disable.
[11]	TMR1_IE	The count value of the counter in the MCPWM operating system is equal to MCPWM_TMR1 interrupt source enable. 1: Enable; 0: Disable.
[10]	TMR0_IE	The count value of the counter in the MCPWM operating system is equal to MCPWM_TMR0 interrupt source enable. 1: Enable; 0: Disable.
[9]	TH31_IE	The count value of the counter in the MCPWM operating system is equal to MCPWM_TH31 interrupt source enable. 1: Enable; 0: Disable.
[8]	TH30_IE	The count value of the counter in the MCPWM operating system is equal to MCPWM_TH30 interrupt source enable. 1: Enable; 0: Disable.
[7]	TH21_IE	The count value of the counter in the MCPWM operating system is equal to MCPWM_TH21 interrupt source enable. 1: Enable; 0: Disable.
[6]	TH20_IE	The count value of the counter in the MCPWM operating system is equal to MCPWM_TH20 interrupt source enable. 1: Enable; 0: Disable.

[5]	TH11_IE	The count value of the counter in the MCPWM operating system is equal to MCPWM_TH11 interrupt source enable. 1: Enable; 0: Disable.
[4]	TH10_IE	The count value of the counter in the MCPWM operating system is equal to MCPWM_TH10 interrupt source enable. 1: Enable; 0: Disable.
[3]	TH01_IE	The count value of the counter in the MCPWM operating system is equal to MCPWM_TH01 interrupt source enable. 1: Enable; 0: Disable.
[2]	TH00_IE	The count value of the counter in the MCPWM operating system is equal to MCPWM_TH00 interrupt source enable. 1: Enable; 0: Disable.
[1]	T1_IE	t1 event. The count value of the counter reaches 0 and the interrupt source is enabled. 1: Enable; 0: Disable.
[0]	T0_IE	t0 event. The count value of the counter returns to MCPWM_TH, and the interrupt source is enabled. 1: Enable; 0: Disable.

16.2.35 MCPWMx_IF0(x = 0,1)

Unprotected register

The addresses are: 0x4001_3484 and 0x4001_3884.

Reset value: 0x0

Table 16-39 MCPWM Time Base 0 Interrupt Control Register (MCPWM_IF0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	UP_IF	TMR3_IF	TMR2_IF	TMR1_IF	TMR0_IF			TH21_IF	TH20_IF	TH11_IF	TH10_IF	TH01_IF	TH00_IF	T1_IF	T0_IF
	RW1C	RW1C	RW1C	RW1C	RW1C			RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C
	0	0	0	0	0			0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:15]		Unused
[14]	UP_IF	Time base 0 update event MCPWM_TH0/MCPWM_TH00- MCPWM_TH21/MCPWM_TMR and other registers are updated to the interrupt source event of MCPWM operating system. 1: occurred; 0: did not occurred. Write 1 to clear.
[13]	TMR3_IF	The count value of the counter in the MCPWM operating system is equal to MCPWM_TMR3 interrupt source event. 1: occurred; 0: did not occurred. Write 1 to clear.
[12]	TMR2_IF	The count value of the counter in the MCPWM operating system is equal to MCPWM_TMR2 interrupt source event. 1: occurred; 0: did not occurred.

		Write 1 to clear.
[11]	TMR1_IF	The count value of the counter in the MCPWM operating system is equal to MCPWM_TMR1 interrupt source event. 1: occurred; 0: did not occurred. Write 1 to clear.
[10]	TMR0_IF	The count value of the counter in the MCPWM operating system is equal to MCPWM_TMR0 interrupt source event. 1: occurred; 0: did not occurred. Write 1 to clear.
[9]	TH31_IF	The count value of the counter in the MCPWM operating system is equal to MCPWM_TH31 interrupt source event. 1: occurred; 0: did not occurred. Write 1 to clear.
[8]	TH30_IF	The count value of the counter in the MCPWM operating system is equal to MCPWM_TH30 interrupt source event. 1: occurred; 0: did not occurred. Write 1 to clear.
[7]	TH21_IF	The count value of the counter in the MCPWM operating system is equal to MCPWM_TH21 interrupt source event. 1: occurred; 0: did not occurred. Write 1 to clear.
[6]	TH20_IF	The count value of the counter in the MCPWM operating system is equal to MCPWM_TH20 interrupt source event. 1: occurred; 0: did not occurred. Write 1 to clear.
[5]	TH11_IF	The count value of the counter in the MCPWM operating system is equal to MCPWM_TH11 interrupt source event. 1: occurred; 0: did not occurred. Write 1 to clear.
[4]	TH10_IF	The count value of the counter in the MCPWM operating system is equal to MCPWM_TH10 interrupt source event. 1: occurred; 0: did not occurred. Write 1 to clear.
[3]	TH01_IF	The count value of the counter in the MCPWM operating system is equal to MCPWM_TH01 interrupt source event. 1: occurred; 0: did not occurred. Write 1 to clear.
[2]	TH00_IF	The count value of the counter in the MCPWM operating system is equal to MCPWM_TH00 interrupt source event. 1: occurred; 0: did not occurred. Write 1 to clear.
[1]	T1_IF	t1 event. Interrupt source event where the count value of the counter reaches 0. 1: occurred; 0: did not occurred. Write 1 to clear.
[0]	T0_IF	t0 event. Interrupt source event where the count value of the counter returns to MCPWM_TH. 1: occurred; 0: did not occurred. Write 1 to clear.

16.2.36 MCPWMx_IE1(x = 0,1)

Write-protected register

The addresses are: 0x4001_3488 and 0x4001_3888.

Reset value: 0x0

Table 16-40 MCPWM Time Base 1 Interrupt Control Register (MCPWM_IE1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	UP_IE	TMR3_IE	TMR2_IE			TH31_IE	TH30_IE							T1_IE	T0_IE
	RW	RW	RW			RW	RW							RW	RW
	0	0	0			0	0							0	0

Bit field	Bit Name	Description
[31:15]		Unused
[14]	UP_IE	MCPWM_TH/MCPWM_TH00- MCPWM_TH31/MCPWM_TMR0-MCPWM_TMR3 and other registers are updated to enable interrupt source of MCPWM operating system. 1: Enable; 0: Disable.
[13]	TMR3_IE	The count value of the counter in the MCPWM operating system is equal to MCPWM_TMR3 interrupt source enable. 1: Enable; 0: Disable.
[12]	TMR2_IE	The count value of the counter in the MCPWM operating system is equal to MCPWM_TMR2 interrupt source enable. 1: Enable; 0: Disable.
[11]	TMR1_IE	The count value of the counter in the MCPWM operating system is equal to MCPWM_TMR1 interrupt source enable. 1: Enable; 0: Disable.
[10]	TMR0_IE	The count value of the counter in the MCPWM operating system is equal to MCPWM_TMR0 interrupt source enable. 1: Enable; 0: Disable.
[9]	TH31_IE	The count value of the counter in the MCPWM operating system is equal to MCPWM_TH31 interrupt source enable. 1: Enable; 0: Disable.
[8]	TH30_IE	The count value of the counter in the MCPWM operating system is equal to MCPWM_TH30 interrupt source enable. 1: Enable; 0: Disable.
[7]	TH21_IE	The count value of the counter in the MCPWM operating system is equal to MCPWM_TH21 interrupt source enable. 1: Enable; 0: Disable.
[6]	TH20_IE	The count value of the counter in the MCPWM operating system is equal to MCPWM_TH20 interrupt source enable. 1: Enable; 0: Disable.
[5]	TH11_IE	The count value of the counter in the MCPWM operating system is equal to MCPWM_TH11 interrupt source enable. 1: Enable; 0: Disable.
[4]	TH10_IE	The count value of the counter in the MCPWM operating system is equal to MCPWM_TH10 interrupt source enable. 1: Enable; 0: Disable.
[3]	TH01_IE	The count value of the counter in the MCPWM operating system is equal to MCPWM_TH01 interrupt source enable. 1: Enable; 0: Disable.
[2]	TH00_IE	The count value of the counter in the MCPWM operating system is equal to MCPWM_TH00 interrupt source enable. 1: Enable; 0: Disable.
[1]	T1_IE	t1 event. The count value of the counter reaches 0 and the interrupt source is enabled. 1: Enable; 0: Disable.
[0]	T0_IE	t0 event. The count value of the counter returns to MCPWM_TH, and the

		interrupt source is enabled. 1: Enable; 0: Disable.

16.2.37 MCPWMx_IF1(x = 0,1)

Unprotected register

The addresses are: 0x4001_349C and 0x4001_389C.

Reset value: 0x0

Table 16-41 MCPWM Time Base 1 Interrupt Flag Register (MCPWM_IF1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	UP_IF	TMR3_IF	TMR2_IF			TH31_IF	TH30_IF							T1_IF	T0_IF
	RW1C	RW1C	RW1C			RW1C	RW1C							RW1C	RW1C
	0	0	0			0	0							0	0

Bit field	Bit Name	Description
[31:15]		Unused
[14]	UP_IF	Time base 0 update event MCPWM_TH1/MCPWM_TH30- MCPWM_TH31/MCPWM_TMR and other registers are updated to the interrupt source event of MCPWM operating system. 1: occurred; 0: did not occurred. Write 1 to clear.
[13]	TMR3_IF	The count value of the counter in the MCPWM operating system is equal to MCPWM_TMR3 interrupt source event. 1: occurred; 0: did not occurred. Write 1 to clear.
[12]	TMR2_IF	The count value of the counter in the MCPWM operating system is equal to MCPWM_TMR2 interrupt source event. 1: occurred; 0: did not occurred. Write 1 to clear.
[11:10]		Unused
[9]	TH31_IF	The count value of the counter in the MCPWM operating system is equal to MCPWM_TH31 interrupt source event. 1: occurred; 0: did not occurred. Write 1 to clear.
[8]	TH30_IF	The count value of the counter in the MCPWM operating system is equal to MCPWM_TH30 interrupt source event. 1: occurred; 0: did not occurred. Write 1 to clear.
[7:2]		Unused
[1]	T1_IF	t1 event. Interrupt source event where the count value of the counter reaches 0. 1: occurred; 0: did not occurred. Write 1 to clear.

[0]	T0_IF	t0 event. Interrupt source event where the count value of the counter returns to MCPWM_TH. 1: occurred; 0: did not occurred. Write 1 to clear.

16.2.38 MCPWMx_EIE(x = 0,1)

Write-protected register

The addresses are: 0x4001_3490 and 0x4001_3890.

Reset value: 0x0

Table 16-42 MCPWM_EIE Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								FAIL3_IE	FAIL2_IE	FAIL1_IE	FAIL0_IE				
								RW	RW	RW	RW				
								0	0	0	0				

Bit field	Bit Name	Description
[31:8]		Unused
[7]	FAIL3_IE	FAIL3 interrupt source enable. 1: Enable; 0: Disable.
[6]	FAIL2_IE	FAIL2 interrupt source enable. 1: Enable; 0: Disable.
[5]	FAIL1_IE	FAIL1 interrupt source enable. 1: Enable; 0: Disable.
[4]	FAIL0_IE	FAIL0 interrupt source enable. 1: Enable; 0: Disable.
[3:0]		Unused

16.2.39 MCPWMx_RE(x = 0,1)

Write-protected register

The addresses are: 0x4001_3498 and 0x4001_3898.

Reset value: 0x0

Table 16-43 MCPWM_RE Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								TR1_T1_RE	TR1_T0_RE	TR0_T1_RE	TR0_T0_RE	TMR3_RE	TMR2_RE	TMR1_RE	TMR0_RE
								RW	RW	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:8]		Unused
[7]	TR1_T1_RE	Time base 1 T1 event DMA request enable. 1: Enable; 0: Disable.
[6]	TR1_T0_RE	Time base 1 T0 event DMA request enable. 1: Enable; 0: Disable.
[5]	TR0_T1_RE	Time base 0 T1 event DMA request enable. 1: Enable; 0: Disable.
[4]	TR0_T0_RE	Time base 0 T0 event DMA request enable. 1: Enable; 0: Disable.
[3]	TMR3_RE	MCPWM counter hits TMR3, DMA request enable. 1: Enable; 0: Disable.
[2]	TMR2_RE	MCPWM counter hits TMR2, DMA request enable. 1: Enable; 0: Disable.
[1]	TMR1_RE	MCPWM counter hits TMR1, DMA request enable. 1: Enable; 0: Disable.
[0]	TMR0_RE	MCPWM counter hits TMR0, DMA request enable. 1: Enable; 0: Disable.

16.2.40 MCPWMx_EIF(x = 0,1)

Unprotected register

The addresses are: 0x4001_3494 and 0x4001_3894.

Reset value: 0x0

Table 16-44 MCPWM_EIF Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								FAIL3_IF	FAIL2_IF	FAIL1_IF	FAIL0_IF				
								RW1C	RW1C	RW1C	RW1C				
								0	0	0	0				

Bit field	Bit Name	Description
[31:8]		Unused
[7]	FAIL3_IF	FAIL3 interrupt source event. 1: occurred; 0: did not occurred. Write 1 to clear.
[6]	FAIL2_IF	FAIL2 interrupt source event. 1: occurred; 0: did not occurred. Write 1 to clear.
[5]	FAIL1_IF	FAIL1 interrupt source event. 1: occurred; 0: did not occurred. Write 1 to clear.
[4]	FAIL0_IF	FAIL0 interrupt source event. 1: occurred; 0: did not occurred. Write 1 to clear.
[3:0]		Unused

16.2.41 MCPWMx_PP(x = 0,1)

Write-protected register

The addresses are: 0x4001_349C and 0x4001_389C.

Reset value: 0x0

Table 16-45 MCPWM_PP Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												IO3_PPE	IO2_PPE	IO1_PPE	IO0_PPE
												RW	RW	RW	RW
												0	0	0	0

Bit field	Bit Name	Description
[31:4]		Unused
[3]	IO3_PPE	IO3 push-pull mode enable signal. 1: Enable; 0: Disable.
[2]	IO2_PPE	IO2 push-pull mode enable signal. 1: Enable; 0: Disable.
[1]	IO1_PPE	IO1 push-pull mode enable signal. 1: Enable; 0: Disable.
[0]	IO0_PPE	IO0 push-pull mode enable signal. 1: Enable; 0: Disable.

Push-pull mode enable signal varies according to different operating modes. Edge mode: turn on the edge-aligned push-pull mode; center alignment: turn on the central-aligned push-pull mode.

16.2.42 MCPWMx_IO01(x = 0,1)

Write-protected register

The addresses are: 0x4001_34A0 and 0x4001_38A0.

Reset value: 0x0

Table 16-46 MCPWM_IO01 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH1_WM	CH1_PN_SW	CH1_SCTRLP	CH1_SCTRLN	CH1_PS	CH1_NS	CH1_PP	CH1_NP	CH0_WM	CH0_PN_SW	CH0_SCTRLP	CH0_SCTRLN	CH0_PS	CH0_NS	CH0_PP	CH0_NP
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:16]		Unused
[15]	CH1_WM	CH1 working mode selection. 1: Edge mode; 0: complementary mode.
[14]	CH1_PN_SW	CH1 P and N channel output interchange selection. That is, the P channel signal is finally output from the N channel, and the N channel signal is finally output from the P channel. 0: interchanged; 1: not interchanged.
[13]	CH1_SCTRLP	When CH1_PS = 1, the value output to CH1 P channel.
[12]	CH1_SCTRLN	When CH1_NS = 1, the value output to CH1 N channel.
[11]	CH1_PS	CH1 P source. 1: From CH1_SCTRLP; 0: MCPWM internal counter is generated.
[10]	CH1_NS	CH1 N source. 1: From CH1_SCTRLN; 0: MCPWM internal counter is generated.
[9]	CH1_PP	CH1 P polarity selection. 1: CH1 P signal is inverted and output; 0: CH1 P signal is output normally.
[8]	CH1_NP	CH1 N polarity selection. 1: CH1 N signal is inverted and output; 0: CH1 N signal is output normally.
[7]	CH0_WM	CH0 working mode selection. 1: Edge mode; 0: complementary mode.
[6]	CH0_PN_SW	CH0 P and N channel output interchange selection. That is, the P channel signal is finally output from the N channel, and the N channel signal is finally output from the P channel. 0: interchanged; 1: not interchanged.
[5]	CH0_SCTRLP	When CH0_PS = 1, the value output to CH0 P channel.
[4]	CH0_SCTRLN	When CH0_NS = 1, the value output to CH0 N channel.
[3]	CH0_PS	CH0 P source. 1: From CH0_SCTRLP; 0: The counter is generated in the MCPWM operating system.
[2]	CH0_NS	CH0 N source. 1: From CH0_SCTRLN; 0: The counter is generated in the MCPWM operating system.
[1]	CH0_PP	CH0 P polarity selection. 1: CH0 P signal is inverted and output; 0: CH0 P signal is output normally.
[0]	CH0_NP	CH0 N polarity selection. 1: CH0 N signal is inverted and output; 0: CH0 N signal is output normally. **Polarity selection follows channel switching. For example, CH0 N selects the inverted output, and at the same time selects channel switching, the CH0 N after the exchange is still the inverted output.**

16.2.43 MCPWMx_IO23(x = 0,1)

Write-protected register

The addresses are: 0x4001_34A4 and 0x4001_38A4.

Reset value: 0x0

Table 16-47 MCPWM_IO23 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH3_WM	CH3_PN_SW	CH3_SCTRLP	CH3_SCTRLN	CH3_PS	CH3_NS	CH3_PP	CH3_NP	CH2_WM	CH2_PN_SW	CH2_SCTRLP	CH2_SCTRLN	CH2_PS	CH2_NS	CH2_PP	CH2_NP
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:16]		Unused
[15]	CH3_WM	CH3 working mode selection. 1: Edge mode; 0: complementary mode.
[14]	CH3_PN_SW	CH3 P and N channel output interchange selection. That is, the P channel signal is finally output from the N channel, and the N channel signal is finally output from the P channel. 0: interchanged; 1: not interchanged.
[13]	CH3_SCTRLP	When CH3_PS = 1, the value output to CH3 P channel.
[12]	CH3_SCTRLN	When CH3_NS = 1, the value output to CH3 N channel.
[11]	CH3_PS	CH3 N source. 1: From CH3_SCTRLP; 0: The counter is generated in the MCPWM operating system.
[10]	CH3_NS	CH3 N source. 1: From CH3_SCTRLN; 0: The counter is generated in the MCPWM operating system.
[9]	CH3_PP	CH3 P polarity selection. 1: CH3 P signal is inverted and output; 0: CH3 P signal is output normally.
[8]	CH3_NP	CH3 N polarity selection. 1: CH3 N signal is inverted and output; 0: CH3 N signal is output normally.
[7]	CH2_WM	CH2 working mode selection. 1: Edge mode; 0: complementary mode.
[6]	CH2_PN_SW	CH2 P and N channel output interchange selection. That is, the P channel signal is finally output from the N channel, and the N channel signal is finally output from the P channel. 0: interchanged; 1: not interchanged.
[5]	CH2_SCTRLP	When CH2_PS = 1, the value output to CH2 P channel.
[4]	CH2_SCTRLN	When CH2_NS = 1, the value output to CH2 N channel.
[3]	CH2_PS	CH2 P source. 1: From CH2_SCTRLP; 0: The counter is generated in the MCPWM operating system.
[2]	CH2_NS	CH2 N source. 1: From CH2_SCTRLN; 0: The counter is generated in the MCPWM operating system.
[1]	CH20_PP	CH2 P polarity selection. 1: CH2 P signal is inverted and output; 0: CH2 P signal is output normally.

[0]	CH2_NP	CH2 N polarity selection. 1: CH2 N signal is inverted and output; 0: CH2 N signal is output normally. **Polarity selection follows channel switching. For example, CH0 N selects the inverted output, and at the same time selects channel switching, the CH0 N after the exchange is still the inverted output.**

16.2.44 MCPWMx_FAIL012(x = 0,1)

Write-protected register

The addresses are: 0x4001_34A8 and 0x4001_38A8.

Reset value: 0x0

Table 16-48 MCPWM_FAIL012 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FAIL_0CAP		CH2N_DEFAULT	CH2P_DEFAULT	CH1N_DEFAULT	CH1P_DEFAULT	CH0N_DEFAULT	CH0P_DEFAULT	HALT_PRT	MCPWM_OE	FAIL1_EN	FAIL0_EN	FAIL1_POL	FAIL0_POL	FAIL1_SEL	FAIL0_SEL
RW		RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0		0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:16]		Unused
[15]	FAIL_0CAP	When a fail0/1 event occurs, the MCPWM_CNT0 value is recorded and stored to MCPWM_FCNT
[14]		Unused
[13]	CH2N_DEFAULT	CH2 N channel default value
[12]	CH2P_DEFAULT	CH2 P channel default value
[11]	CH1N_DEFAULT	CH1 N channel default value
[10]	CH1P_DEFAULT	CH1 P channel default value
[9]	CH0N_DEFAULT	CH0 N channel default value
[8]	CH0P_DEFAULT	CH0 P channel default value When a FAIL event occurs or MOE is 0, the corresponding channel outputs the default level. **The default level output is not affected by the exchange and polarity control of BIT0, BIT1, BIT8, BIT9, BIT6, BIT14 of MCPWM_IO01 and MCPWM_IO23.**
[7]	HALT_PRT	The MCU enters the HALT state, and the MCPWM output value is selected. 1: Normal output; 0: Force MCPWM to output protection value.

[6]	MOE	MOE controls MCPWM CH P and N output values. 1: Output the normal signal generated by MCPWM 0: Output CHxN_DEFAULT and CHxP_DEFAULT default values. This default value is not controlled by polarity, channel selection, etc. Any change of MCPWM_EIF.FAIL1_IF and MCPWM_EIF.FAIL0_IF to "1" will trigger MCPWM_OE to become "0", and output the default value.
[5]	FAIL1_EN	FAIL1 input enable. 1: Enable; 0: Disable.
[4]	FAIL0_EN	FAIL0 input enable. 1: Enable; 0: Disable.
[3]	FAIL1_POL	FAIL1 polarity selection. 1: Invert signal polarity input. The input signal is active low; 0: Normal signal polarity input. The input signal is active high.
[2]	FAIL0_POL	FAIL0 polarity selection. 1: Invert signal polarity input. The input signal is active low; 0: Normal signal polarity input. The input signal is active high.
[1]	FAIL1_SEL	FAIL1 source selection. 1: Comparator result; 0: From GPIO.
[0]	FAIL0_SEL	FAIL0 source selection. 1: Comparator result; 0: From GPIO.

For MCPWM0, FAIL0 signal can come from comparator 0 or MCPWM0_BKIN0, and FAIL1 signal can come from comparator 1 or MCPWM0_BKIN1.

For MCPWM1, FAIL0 signal can come from comparator 2 or MCPWM1_BKIN0, and FAIL1 signal can come from comparator 3 or MCPWM1_BKIN1.

The FAIL0/1 signal is used for MCPWM channel 0/1/2 working at time base 0.

16.2.45 MCPWMx_FAIL3(x = 0,1)

Write-protected register

The addresses are: 0x4001_34AC and 0x4001_38AC.

Reset value: 0x0

Table 16-49 MCPWM_FAIL3 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FAIL_1CAP						CH3N_DEFAULT	CH3P_DEFAULT	HALT_PRT	MOE	FAIL3_EN	FAIL2_EN	FAIL3_POL	FAIL2_POL	FAIL3_SEL	FAIL2_SEL
RW						RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0						0	0	0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:16]		Unused
[15]	FAIL_1CAP	When a fail2/3 event occurs, the MCPWM_CNT1 value is stored to MCPWM_FCNT
[14:10]		Unused
[9]	CH3N_DEFAULT	CH3 N channel default value
[8]	CH3P_DEFAULT	CH3 P channel default value When a FAIL event occurs or MOE is 0, the corresponding channel outputs the default level. **The default level output is not affected by the exchange and polarity control of BIT0, BIT1, BIT8, BIT9, BIT6, BIT14 of MCPWM_and IO23.**
[7]	HALT_PRT	The MCU enters the HALT state, and the MCPWM output value is selected. 1: Normal output; 0: Force MCPWM to output protection value.
[6]	MOE	MOE controls MCPWM CH3 P and N output values. 1: Output the normal signal generated by MCPWM 0: Output CH3N_DEFAULT and CH3P_DEFAULT default values. This default value is not controlled by polarity, channel selection, etc. Any change of MCPWM_EIF.FAIL2_IF and MCPWM_EIF.FAIL3_IF to "1" will trigger MCPWM_OE to become "0", and output the default value.
[5]	FAIL3_EN	FAIL3 input enable. 1: Enable; 0: Disable.
[4]	FAIL2_EN	FAIL2 input enable. 1: Enable; 0: Disable.
[3]	FAIL3_POL	FAIL3 polarity selection. 1: Invert signal polarity input. The input signal is active low; 0: Normal signal polarity input. The input signal is active high.
[2]	FAIL2_POL	FAIL2 polarity selection. 1: Invert signal polarity input. The input signal is active low; 0: Normal signal polarity input. The input signal is active high.
[1]	FAIL3_SEL	FAIL3 source selection. 1: Comparator result; 0: From GPIO.
[0]	FAIL2_SEL	FAIL2 source selection. 1: Comparator result; 0: From GPIO.

For MCPWM0, FAIL2 signal can come from comparator 4 or MCPWM0_BKIN2, and FAIL3 signal can come from comparator 5 or MCPWM0_BKIN3.

For MCPWM1, FAIL2 signal can come from comparator 4 or MCPWM1_BKIN2, and FAIL3 signal can come from comparator 5 or MCPWM1_BKIN3.

MCPWM_FAIL can be used to set emergency stop events and block MCPWM signal output. There are two emergency stop events for channel 0/1/2, FAIL0 and FAIL1; and two emergency stop events for channel 3, FAIL2 and FAIL3. Each FAIL signal has two optional signal sources, Comparator output or MCPWM_BKIN.

The FAIL2/3 signal is used for MCPWM channel 1 3 working at time base.

The input signal of FAIL can be processed by digital filtering, and the first frequency division of the filtered clock is set by the MCPWM_TCLK.CLK_DIV register. The filter clock frequency division output by signal source Comparator is set by MCPWM_TCLK.CMP_FLT_CLKDIV; and the filter clock frequency division of the signal source MCPWM_BKIN is set by MCPWM_TCLK.IO_FLT_CLKDIV.

Finally, filter circuit will filter the FAIL signal with 16 filter clocks, that is, the signal can only pass through the filter if the signal stabilization time exceeds 16 filter cycles. **Filter width = filter clock period*16.**

See more in Fail Signal Processing.

16.2.46 MCPWMx_PRT(x = 0,1)

Unprotected register

The addresses are: 0x4001_34B0 and 0x4001_38B0.

Reset value: 0x0

Table 16-50 MCPWM_PRT Register

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

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	PRT	Write 0xDEAD to release the write protection of the MCPWM register; write other values, the MCPWM register enters write protection state. The read value of the register is always 0.

16.2.47 MCPWMx_CHMSK(x = 0,1)

Write-protected register

The addresses are: 0x4001_34B8 and 0x4001_38B8.

Table 16-51 MCPWM_CHMSK Channel Masking Bit Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										CH2P_FAIL_EN	CH2N_FAIL_EN	CH1P_FAIL_EN	CH1N_FAIL_EN	CH0P_FAIL_EN	CH0N_FAIL_EN
										RW	RW	RW	RW	RW	RW
										1	1	1	1	1	1

Bit field	Bit Name	Description
[31:6]		Unused
[5]	CH2P_FAIL_EN	CH2_P FAIL event channel masking enable, active high. It is turned on by default
[4]	CH2N_FAIL_EN	CH2_N FAIL event channel masking enable, active high. It is turned on by default
[3]	CH1P_FAIL_EN	CH1_P FAIL event channel masking enable, active high. It is turned on by default
[2]	CH1N_FAIL_EN	CH1_N FAIL event channel masking enable, active high. It is turned on by default
[1]	CH0P_FAIL_EN	CH0_P FAIL event channel masking enable, active high. It is turned on by default
[0]	CH0N_FAIL_EN	CH0_N FAIL event channel masking enable. 1: When a FAIL event occurs, the CH0_N channel level output is the default value; 0: When a FAIL event occurs, CH0_N channel level is not affected and is still controlled by MCPWM internal hardware. The FAIL mechanism is enabled by default

17 UART

17.1 Overview

Universal Asynchronous Receiver/Transmitter (UART) is a kind of asynchronous receiver/transmitter.

UART features are as follows:

- Support full-duplex operation
- Support single-line half-duplex operation
- Support 8/9 data bit
- Support 1/2 stop bit
- Support odd/even/no parity mode
- 1 byte tx buffer
- 1 byte rx buffer
- Support LIN break character transceiving
- Support detection of idle frame
- Support Multi-drop Slave/Master mode

17.2 Function Description

17.2.1 Send

The UART includes a byte tx buffer. When the tx buffer has data, the UART loads the data of the tx buffer to the serial shift register and sends it out via UART_TXD port. As long as there is data in the UART tx buffer, the UART will automatically send it.

After the loading is completed, the tx buffer empty interrupt is generated. Then, user can fill the tx buffer with the next byte to be sent. After the transmission is completed, the UART will load this byte for transmission.

After the transmission is completed, a transmission completion interrupt will be generated.

TX flow:

Set SYS_CLK_FEN. UART_CLK_EN = 1 to enable the UART clock

Set UART_CTRL.BYTE_LEN and select 8/9-bit data byte length

Set UART_CTRL.CK_EN and select whether to enable the data byte check

Set UART_CTRL.CK_TYPE and select odd parity check or even parity check

Set UART_CTRL.BIT_ORDER and select LSB first or MSB first when transmitting

Set UART_CTRL.STOP_LEN and select the stop bit of 1bit or 2bit

If DMA is used to transfer data, set UART_RE. TX_BUF_EMPTY_RE=1, that is, when the TX buffer is empty, trigger DMA to transfer new data to UART; if software polling or interrupt mode is used, set UART_IE. TX_BUF_EMPTY_IE=1.

If DMA is used to transfer data to UART for transmission, set DMA accordingly.

If software is used to transfer data to UART for transmission, use the software to write data to UART_BUFF to trigger the UART sent data from the UART_TXD port.

17.2.2 Receiving

The UART includes a byte of rx buffer. When the byte is received, a receiving interrupt will be generated and the received byte will be stored in the rx buffer; the user should finish reading this byte before receiving the next byte in the UART; otherwise, the buffer will be written to the newly received byte. RX uses a high-speed clock internally to oversample the UART_RX signal to determine the steady-state data signal, preventing noise interference from causing erroneous reception.

17.2.3 UART Frame Format

The format of data sent and received on UART signals is usually as follows:

Signal line is idle;

1 bit Start bit: 1 bit Zero

Data word, 8bits or 9bits, LSB first or MSB first

1/2 bit Stop bit: 1/2bit Ones

The data byte length can be set by UART_CTRL.BYTE_LEN to 8bit (UART_CTRL.BYTE_LEN=0) or 9bit (UART_CTRL.BYTE_LEN=1). The start bit is 1-bit zero, and TX signal line is low; the TX signal line is high when stopping the bit.

The stop bit length is set by UART_CTRL.

Note than when the UART_CTR.CK_EN bit is enabled, the parity bit will replace the highest bit of the data byte, i.e. MSB. At this time, an 8-bit data is actually 7-bit data + 1-bit check word, and a 9-bit data is actually 8-bit data + 1-bit check word.

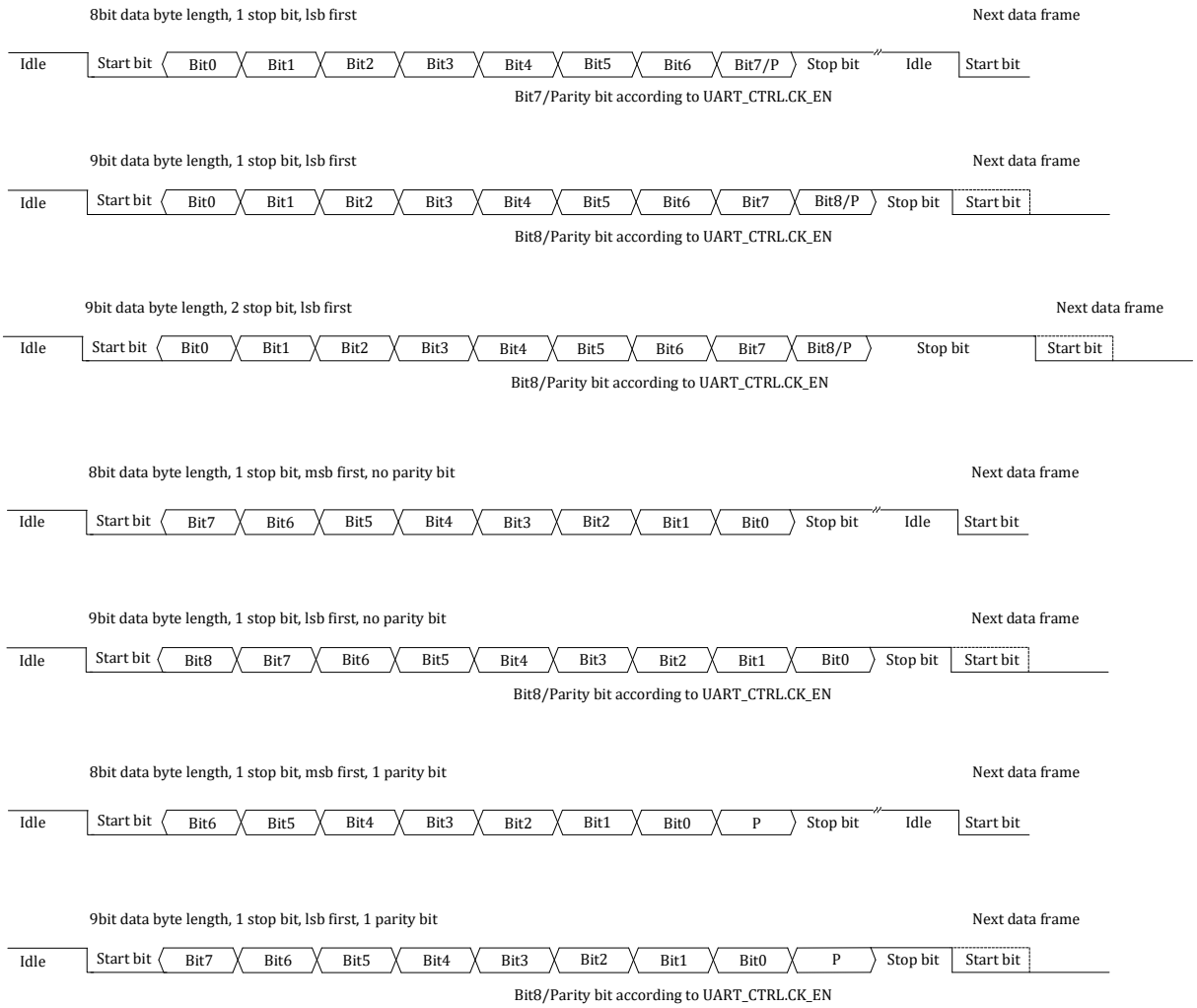


Figure 17-1 Example of UART Baud Rate Configuration

17.2.4 Baud Rate Configuration

The UART input clock is the main clock, and the baud rate is realized by two-stage frequency division.

Baud rate = UART module clock/(256*DIVH+DIVL+1)

The UART module clock can be divided by SYS_CLK_DIV2.

UART module clock = main clock/(1+SYS_CLK_DIV2)

Table 17-1 Example of UART Baud Rate Configuration

UART Baud Rate	SYS_CLK_DIV2	UART_DIVH	UART_DIVL
600	0x0007	0x9C	0x3F
1200	0x0003	0x9C	0x3F
2400	0x0001	0x9C	0x3F
4800	0x0000	0x9C	0x3F
9600	0x0000	0x4E	0x1F

19200	0x0000	0x27	0x0F
38400	0x0000	0x13	0x87
43000	0x0000	0x11	0x71
56000	0x0000	0x0D	0x65
57600	0x0000	0x0D	0x05
115200	0x0000	0x06	0x83

Note: The baud rate configuration factor is only an example and may not be unique.

Note: The baud rate configuration factor is only an example and may not be unique.

17.2.5 TX/RX Interchange

UART module supports TX/RX interchange. By configuring the GPIO corresponding to TX as input enable and the GPIO corresponding to RX as output enable, the TX and RX ports can be interchanged. At this time, GPIO second function is still UART, and the configuration of UART does not need to be modified.

Besides, if a GPIO is to be used as TX and RX at the same time, the IO needs to be used as the input or output port in a time-division multiplexing manner, corresponding to RX or TX, to realize single-port half-duplex logic.

17.2.6 Multi-computer Communication

In multi-computer communication, one computer is used as the master device and several other computers are used as the slave devices. The UARTm_TXD port of the master connects to the UARTs_RXD port of all slaves, and the UARTs_TXD of all slaves is connected to the UARTm_RXD port of the master. As shown in Figure 17-2, it shows the interconnection of one master and three slaves (the addresses are 0x51,0x52, and 0x53 respectively).

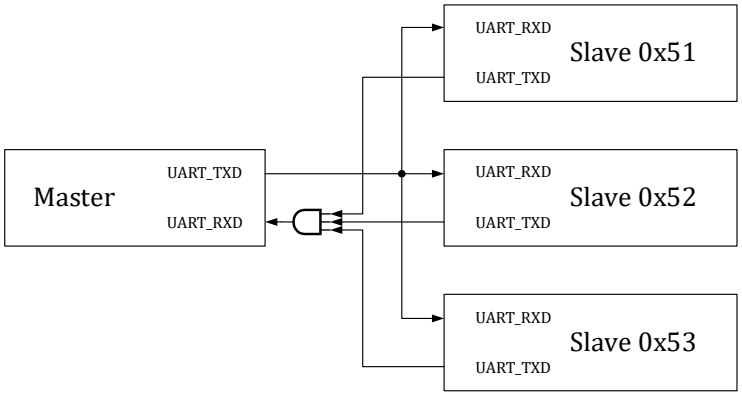


Figure 17-2 UART Multi-computer Communication Connection Topology

To reduce the packet processing load of the slave, the slave should only process the UART data sent to it and ignore the data sent to other slaves. In multi-computer communication scenario, each slave has an 8-bit device address, i.e. UART_ADR. After the slave sets UART_CTRL.MD_EN=1, the multi-computer communication filter mode is enabled and the slave does not need to set MD_EN.

When the MSB (BIT8) of the data byte sent by the master is 1, the lower 8-bit (BIT7:BIT0) is the slave address sent by the master, that is, the **address byte**; when the MSB (BIT8) of the sending data byte is 0, the lower 8-bit (BIT7:BIT0) is the data sent by the master to the specific slave, that is, the **data byte**.

After all the slaves receive the data byte with MSB (BIT8)=1, they judge whether the address of the lower 8-bit (BIT7:BIT0) is equal to their own UART_ADR configuration value. If yes, it means that the subsequent data is sent to this slave, otherwise the slave enters the silent state, ignoring all subsequent data sent by the master. When the slave receives the data byte with MSB (BIT8)=1 again and the lower 8-bit (BIT7:BIT0) address is equal to its own UART_ADR configuration value, it means that the slave is selected and it will exit the silent state. After the slave sets UART_CTRL.MD_EN=1 to enable the multi-computer communication filter mode, it immediately enters the silent state. If the master does not send the address byte with MSB=1, and directly sends the data byte, all the slaves are in silent state and will not receive any data.

In multi-computer communication mode, the master does not need to set UART_CTRL.MD_EN=1, but the slave needs to set UART_CTRL. MD_EN=1. If address filtering is not required, the slave can also not set UART_CTRL.MD_EN=1, and use software to read the received data. At this time, the slave will receive and process all message data.

In the multi-computer communication mode, if the slave sets UART_CTRL.MD_EN=1, the address byte will not set the UART_IF.RX_DONE_IF flag regardless of whether the address byte hits UART_ADR or not; if the address does not match, the slave is in a silent state. At this time, even if the master sends data, UART_IF.RX_DONE_IF is not set. If the address matches, when the data byte sent by the master is received, the UART_IF.RX_DONE_IF flag of the slave is set to 1, indicating that the slave has received a data byte.

Since the parity bit occupies 1 data bit and the multi-computer communication needs 9 bits, the multi-computer communication does not currently support parity bit. When setting UART_CTRLMD_EN=1, please do not set UART_CTRL.CK_EN=1. Both the master and slave need to set UART_CTRL.BYTE_LEN=1 to use 9-bit mode, and set UART_CTRL.MD_EN. The multi-computer communication supports MSB first, that is UART_CTRL.BIT_ORDER=1.

As shown in Figure 17-3 , the master first sends the data 0x03, but no slave device address is hit at this time, so none of the 3 slaves receive 0x03. After that, the master does not send data, but directly sends the address byte 0x153, and the slave device 0x53 is selected. The master continuously sends 3 data bytes, i.e. 0x03, 0x53 and 0x04, all of which are received by the slave 0x53. Then the master device sends the address byte 0x152, and then the data byte 0x07, which is received by the slave device 0x52.

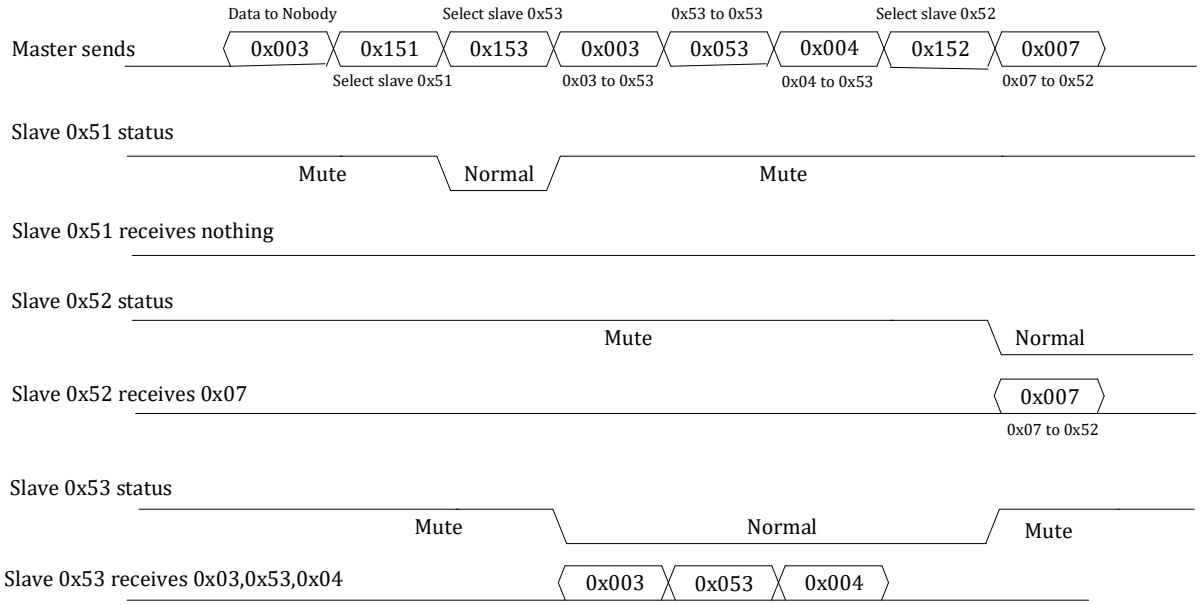


Figure 17-3 Example of UART Multi-computer Communication

17.2.7 Check bit

You can set UART_CTRL.CK_EN = 1 to enable the parity bit. There are four UART frame formats, depending on the length of the byte uart_ctrl. BYTE_LEN.

Table 17-2 UART Frame Format

BYTE_LEN	CK_EN	UART frame
0	0	| StartBit | 8bit data | StopBit |
0	1	| StartBit | 7bit data | Parity | StopBit |
1	0	| StartBit | 9bit data | StopBit |
1	1	| StartBit | 8bit data | Parity | StopBit |

17.3 Register

17.3.1 Address Allocation

The implementation of UART0, UART1 and UART2 are the same.

The base address of UART0 is 0x4001_2000.

The base address of UART1 is 0x4001_2400.

The base address of UART2 is 0x4001_2800.

Table 17-3 UART Address Distribution List

Name	Offset Address	Description
UARTx_CTRL	0x00	UART control register

UARTx_DIVH	0x04	High byte register with UART baud rate setting
UARTx_DIVL	0x08	Low byte register with UART baud rate setting
UARTx_BUFF	0x0C	UART transceiver buffer register
UARTx_ADR	0x10	485 communication address matching register
UARTx_STT	0x14	UART status register
UARTx_RE	0x18	UART DMA request enable register
UARTx_IE	0x1C	UART interrupt enable register
UARTx_IF	0x20	UART interrupt flag register
UARTx_IOC	0x24	UART IO control

17.3.2 UARTx Control Register (x = 0,1,2) (UARTx_CTRL)

The addresses are: 0x4001_2000, 0x4001_2400, and 0x4001_2800.

Reset value: 0x0

Table 17-4 UART Control Register (UARTx_CTRL)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								DUPLEX	LIN_EN	MD_EN	CK_EN	CK_TYPE	BIT_ORDER	STOP_LEN	BYTE_LEN
								RW	RW	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:8]		Unused
[7]	DUPLEX	Duplex mode. The default value is 0. 0: Full duplex; 1: Half duplex
[6]	LIN_EN	LIN mode enable. The default value is 0. 0: Disable; 1: Enable
[5]	MD_EN	Enable Multi-drop. The default value is 0. 0: Disable; 1: Enable
[4]	CK_EN	Data check switch. The default value is 0. 0: Disable; 1: Enable
[3]	CK_TYPE	Parity check. The default value is 0. 0: EVEN; 1: ODD
[2]	BIT_ORDER	Bits transmission order. The default value is 0. 0: LSB, 1: MSB
[1]	STOP_LEN	Stop bit length. The default value is 0. 0:1-Bit; 1:2-Bit
[0]	BYTE_LEN	Data length. The default value is 0.

		0: 8-Bit; 1: 9-Bit

17.3.3 UARTx Baud Rate High-byte Register (x = 0,1,2) (UARTx_DIVH)

The addresses are: 0x4001_2004, 0x4001_2404, and 0x4001_2804.

Reset value: 0x0

Table 17-5 UART Baud Rate High-byte Register (UARTx_DIVH)

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

Bit field	Bit Name	Description
[31:8]		Unused
[7:0]	DIVH	Baud rate setting high byte BAUDRATE =main clock/(1+256* UART_DIVH+UART_DIVL)

17.3.4 UARTx Baud Rate High-byte Register (x = 0,1,2) (UARTx_DIVL)

The addresses are: 0x4001_2008, 0x4001_2408, and 0x4001_2808.

Reset value: 0x0

Table 17-6 UART Baud Rate Low-byte Register (UART_DIVL)

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

Bit field	Bit Name	Description
[31:8]		Unused
[7:0]	DIVL	Baud rate setting low byte BAUDRATE =main clock/(1+256* UART_DIVH+UART_DIVL)

17.3.5 UARTx Transceiver Buffer Register (x = 0,1,2) (UARTx_BUFF)

The addresses are: 0x4001_200C, 0x4001_240C, and 0x4001_280C.

Table 17-7 UART Transceiver Buffer Register (UART_BUFF)

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

	BUFF
	RW
	0

Bit field	Bit Name	Description
[31:9]		Unused
[8:0]	BUFF	Write: data transmit buffer; read: data receive register

The Tx_buffer and Rx_buffer of the UART share the address UART_BUFF. Among them, Tx_buffer is write-only, Rx_buffer is read-only. Therefore, read access to UARTx_BUFF is to access UART_RX_BUFF, and write access to UARTx_BUFF is to access UART_TX_BUFF.

When UARTx_CTRL.BYTE_LEN=0, UART only uses the lower 8-bit (UARTx_BUFF[7:0]) of buffer;

When UARTx_CTRL.BYTE_LEN=1, UART only uses all the 9-bit (UARTx_BUFF[8:0]) of buffer.

If UARTx_CTRL.CK_EN is enabled, that is, parity bit is enabled, the highest bit of UARTx_BUFF (BIT8 when UARTx_CTRL.BYTE_LEN=1 or BIT7 UARTx_CTRL.BYTE_LEN=0) is invalid, and is replaced with a parity bit when sent and received as a parity bit without being written to the UARTx_BUFF.

17.3.6 UARTx Address Match Register (x = 0,1,2) (UARTx_ADR)

The addresses are: 0x4001_2010, 0x4001_2410, and 0x4001_2810.

Reset value: 0x0

Table 17-8 UART Address Match Register (UART_ADR)

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

Bit field	Bit Name	Description
[31:8]		Unused
[7:0]	ADR	Slave address during multi-computer communication

17.3.7 UARTx Status Register (x = 0,1,2) (UARTx_STT)

The addresses are: 0x4001_2014, 0x4001_2414, and 0x4001_2814.

Reset value: 0x0

Table 17-9 UART Status Register (UART_STT)

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

	RX_BUSY	ADR_MATCH	TX_DONE	TX_BUF_EMPTY
	R	R	R	R
	0	0	1	1

Bit field	Bit Name	Description
[31:4]		Unused
[3]	RX_BUSY	1: UART has detected the start character and is receiving 0: UART receiver is idle This status bit is read-only, and its setting and clear are completed by the hardware
[2]	ADR_MATCH	Address match flag in Multi-drop mode. 1: match; 0: not match.
[1]	TX_DONE	Transmitted flag bit. 1: done; 0: undone.
[0]	TX_BUF_EMPTY	Tx buffer status bit. 1: empty; 0: not empty.

17.3.8 UARTx DMA Request Enable Register (x = 0,1,2) (UARTx_RE)

The addresses are: 0x4001_2018, 0x4001_2418, and 0x4001_2818.

Reset value: 0x0

Table 17-10 UART DMA Request Enable Register (UART_RE)

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

Bit field	Bit Name	Description
[31:3]		Unused
[2]	TX_BUF_EMPTY_RE	DMA request switch when TX buffer is empty. The default value is 0. 0: Disable; 1: Enable.

[1]	RX_DONE_RE	DMA request switch when RX is done. The default value is 0. 0: Disable; 1: Enable.
[0]	TX_DONE_RE	DMA request switch when TX is done. The default value is 0. 0: Disable; 1: Enable.

17.3.9 UARTx Interrupt Enable Register (x = 0,1,2) (UARTx_IE)

The addresses are: 0x4001_201C, 0x4001_241C, and 0x4001_281C.

Reset value: 0x0

Table 17-11UART Interrupt Enable Register (UART_IE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							IDLE_IE	LBD_IE	RX_OV_IE	TX_OV_IE	CK_ERR_IE	STOP_ERR_IE	TX_BUF_EMPTY_IE	RX_DONE_IE	TX_DONE_IE
							RW	RW	RW	RW	RW	RW	RW	RW	RW
							0	0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:9]		Unused
[8]	IDLE_IE	Idle frame interrupt enable 0: Disable; 1: Enable
[7]	LBD_IE	LIN break character detection interrupt enable 0: Disable; 1: Enable
[6]	RX_OV_IE	Receive buffer overflow interrupt enable 0: Disable; 1: Enable
[5]	TX_OV_IE	Send buffer overflow interrupt enable 0: Disable; 1: Enable
[4]	CK_ERR_IE	Check error interrupt switch. The default value is 0. 0: Disable; 1: Enable
[3]	STOP_ERR_IE	Stop bit error interrupt switch. The default value is 0. 0: Disable; 1: Enable
[2]	TX_BUF_EMPTY_IE	Tx buffer empty interrupt switch. The default value is 0. 0: Disable; 1: Enable
[1]	RX_DONE_IE	Rx completion interrupt switch. The default value is 0. 0: Disable; 1: Enable
[0]	TX_DONE_IE	Tx completion interrupt switch. The default value is 0. 0: Disable; 1: Enable

17.3.10 UARTx Interrupt Enable Register (x = 0,1,2) (UARTx_IF)

The addresses are: 0x4001_2020, 0x4001_2420, and 0x4001_2820.

Reset value: 0x0

Table 17-12 UART Interrupt Flag Register (UART_IF)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							IDLE_IF	LBD_IF	RX_OV_IF	TX_OV_IF	CK_ERR_IF	STOP_ERR_IF	TX_BUF_EMPTY_IF	RX_DONE_IF	TX_DONE_IF
							RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C
							0	0	0	0	0	0	1	0	1

Bit field	Bit Name	Description
[31:9]		Unused
[8]	IDLE_IF	Idle frame interrupt flag, active high, write 1 to clear When UART detects an idle frame, the hardware will set to the flag. If UARTx_IE.IDLE_IE=1, an interrupt request will be generated; after writing 1 to clear, IDLE_IF flag will be set only after RX_DONE_IF flag is reset. In multi-drop scenario, only when the slave address is hit, that is UARTx_STT.ADR_MATCH=1, which indicates that the master computer intends to communicate with the slave, IDLE_IF will be set; otherwise, even if UARTx_RX signal is high, IDLE_IF will not be set.
[7]	LBD_IF	LIN break character detection interrupt flag, active high, write 1 to clear
[6]	RX_OV_IF	Receive buffer overflow interrupt flag, active high, write 1 to clear.
[5]	TX_OV_IF	Send buffer overflow interrupt flag, active high, write 1 to clear.
[4]	CK_ERR_IF	Check error interrupt flag, active high, write 1 to clear.
[3]	STOP_ERR_IF	Stop bit error interrupt flag, active high, write 1 to clear.
[2]	TX_BUF_EMPTY_IF	Tx buffer empty interrupt flag, active high, write 1 to clear.
[1]	RX_DONE_IF	Rx completion interrupt flag, active high, write 1 to clear.
[0]	TX_DONE_IF	Tx completion interrupt flag, active high, write 1 to clear.

Write 1 to interrupt flag to clear. Generally, |= method is not recommended for clear, because |= will read the interrupt flag first and then change the bit to 1 to clear. If there are other interrupt flag bits, all will be cleared, which is not what the software expects. For example, the following expression it to clear TX_DONE_IF, but if RX_DONE_I is set to 1 before writing, the software will first read the UART_IF value of 0x2, and then execute 0x2|0x1= 0x3 before writing, resulting in clear of both RX_DONE_IF and TX_DONE_IF, which may cause the UART receive one less byte of data due to one less

interrupt caused by receiving data.

```
UART_IF|=0x1;
```

To clear TX_DONE_IF bit, directly write 1 to BIT0, as follows.

```
UART_IF=0x1;
```

17.3.11 UARTx IO Control Register (x = 0,1,2) (UARTx IO_IOC)

The addresses are: 0x4001_2024, 0x4001_2424, and 0x4001_2824.

Reset value: 0x0

Table 17-13 UARTIO Control Register (UART_IOC)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								SBK	LBDL		AUTO			TXD_INV	RXD_INV
								RW	RW		RW			RW	RW
								0	0		0			0	0

Bit field	Bit Name	Description
[31:8]		Unused
[7]	SBK	In LIN mode, writing 1 to send a break character, that is 13 0s in a row to automatically clear after completion. If not completed, it will read 1. UARTx_CTRL.LIN_EN=1 shall be configured before writing 1 to the bit; otherwise, break character cannot be sent.
[6]	LBDL	LIN break character detection length, 10/11 0s. 0:10bits, 1:11bits
[5]		Unused
[4]	AUTO	Baud rate adaptive IO port enable switch. 0: Disable; 1: Enable
[3:2]		Unused
[1]	TXD_INV	TXD output polarity enable switch. The default value is 0. 0: Normal output; 1: Inverted output. Normal output polarity means that when the software sends "1", the hardware sends "1"; invert output polarity means that the when the software sends "1", the hardware sends "0".
[0]	RXD_INV	RXD input polarity enable switch. The default value is 0. 0: Normal input; 1: Inverted input. Normal input polarity means that when the hardware receives "1", the software receives "1"; invert input polarity means that when the

		hardware receives "1", the software receives "0".

18 I2C

18.1 Overview

The I2C bus interface connects the microcontroller and the serial I2C bus. It provides multi-master functions to control the specific timing, protocol, arbitration and timing of all I2C buses. Besides, it supports standard and fast modes.

18.2 Main Features

- Multi-master function: this module can be used as both master and slave.

  I2C master device function: generate clock, START and STOP events.

  I2C slave device functions: programmable I2C hardware address comparison (only supports 7-bit hardware address), stop bit detection.

- Provide different communication speeds depending on the frequency division.
- Status flags: transmitter/receiver mode flag, byte transmission end flag, I2C bus busy flag.
- Error flags: Loss of arbitration in master mode, acknowledgment (ACK) error after address/data transmission, start or stop condition where misalignment was detected.
- An interrupt vector contains five interrupt sources: bus error interrupt source, completion interrupt source, NACK interrupt source, hardware address matching interrupt source, and transfer completion interrupt source.

18.3 Function Description

18.3.1 Functional block diagram

This interface adopts a synchronous serial design to achieve I2C transmission between the MCU and external devices, and supports polling and interrupt mode to obtain transmission status information. The main functional modules of this interface are shown in the figure below.

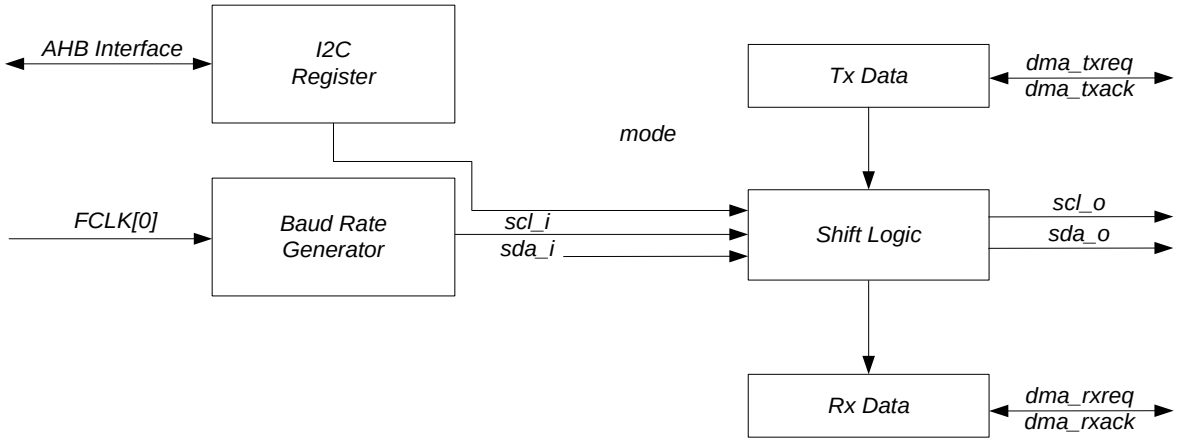


Figure 18-1 I2C Module Top Functional Block Diagram

The I2C interface communicates with the outside world only with two signal lines, SCL and SDA. SDA is a bidirectional multiplexed signal line, controlled by sda_oe. Module-level I2C interface signals include scl_i, sda_i, scl_o, sda_o, and sda_oe.

scl_i: clock signal. When the I2C interface is set as slave mode, this is the clock input signal for the I2C bus.

sda_i: data signal. When the I2C interface receives data (regardless of master mode or slave mode), this is the data input signal of the I2C bus.

scl_o: clock signal. When the I2C interface is set as the main mode, this is the clock output signal of the I2C bus.

sda_o: data signal. When the I2C interface sends data (regardless of master mode or slave mode), this is the data output signal of the I2C bus.

sda_oe: data enable signal. When sda_o is output, sda_oe is valid; when sda_i is input, sda_oe is invalid.

18.3.2 Function Description

The I2C module receives and sends data, and converts the data from serial to parallel, or parallel to serial, and can enable or disable interrupts. The interface is connected to the I2C bus via data pins (SDA) and clock pins (SCL).

18.3.2.1 Mode selection

The interface can operate in one of the following four modes:

- Slave TX mode
- Slave RX mode
- Master TX mode
- Master RX mode

The I2C interface is not enabled by default. The interface enters master mode or slave mode

according to the configuration. When arbitration is lost or a stop signal is generated, the master mode releases the bus automatically and generates an abnormal interrupt. Multiple host functions are available.

In master mode, the I2C interface starts data transmission and generates a clock signal. Serial data transmission always starts with a start condition and ends with a stop condition. The start condition and stop condition are generated by software control in the master mode.

In slave mode, the I2C interface can recognize its own address (7 bits). The software can control to enable or disable the hardware address comparison function, which can reduce the burden on the MCU. Only when the addresses match, the MCU is notified to perform relevant processing.

The data and address are transmitted in 8 bits/byte, with the high order first. The one byte following the start condition is the address, and the address is only sent in master mode.

During the ninth clock after the eight clocks for one byte transmission, the receiver must send back an acknowledge bit (ACK) to the transmitter.

The software can enable or disable acknowledgement (ACK), and can set the address of the I2C interface.

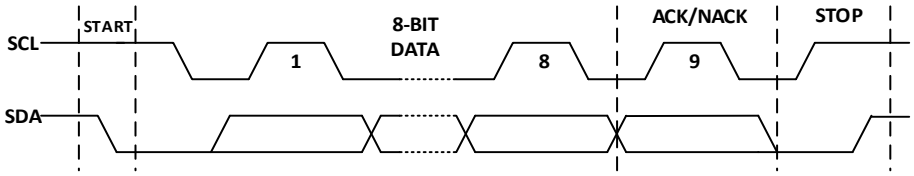


Figure 18-2 Basic I2C Transmission Timing Diagram

In general, one byte at a time is transmitted (single transmission can be repeated, and the data should be provided by software). All the above modes follow the basic principles below:

- Single byte transmission. An interrupt will be generated after the 8-bit data is sent and the response is received (either ACK/NACK).
- Single byte reception. An interrupt will be generated after the 8-bit data is received.
- When the I2C interface is set as the master mode, the I2C interface will release the bus after detecting an error, restore to the initial state and generate an interrupt signal.

18.3.2.2 Slave Mode

Both the master mode and slave mode of the I2C interface are turned off by default. If operating in slave mode, the slave mode should be enabled. In order to generate the correct timing, the working clock of the I2C interface must be set in the register CLK_DIV0.

- In slave mode, the I2C interface monitors the signals on the bus at all times. Once the start condition is detected, it will save the address bit data and read-write bit data.
- In slave mode, if the hardware address matching function is enabled, an interrupt will be

generated and the MCU will be notified for subsequent processing only when the addresses match. If the function is not enabled, an interrupt will be generated each time the address and read/write bit data is received.

- Slave mode receive. Each time a byte of data is received, an interrupt is generated. The I2C interface can pull SCL low until the interrupt is completed and continue the subsequent operations.
- Slave mode transport. After each byte is sent and a response (ACK/NACK) is received, an interrupt is generated. The I2C interface can pull SCL low until the interrupt is completed and continue the subsequent operations.

18.3.2.2.1 Slave Mode Transmission

After each transfer of one byte of data, the hardware will generate an interrupt and the software will determine whether to continue the transfer. Figure 15-3 is a schematic diagram of a bus for slave mode transmission. As can be seen from the figure, the transmission process is as follows:

- Address match, generate address match interrupt, ready to start transmission.
- In slave RX mode, an interrupt is generated after a byte is received, the software determines whether to continue receiving, and returns an ACK/NACK response.
- In slave TX mode, an interrupt is generated when receiving the response (ACK/NACK) after a byte is sent, and subsequent operations are judged based on the response.
- Obtain the bus STOP event, this transmission is completed.

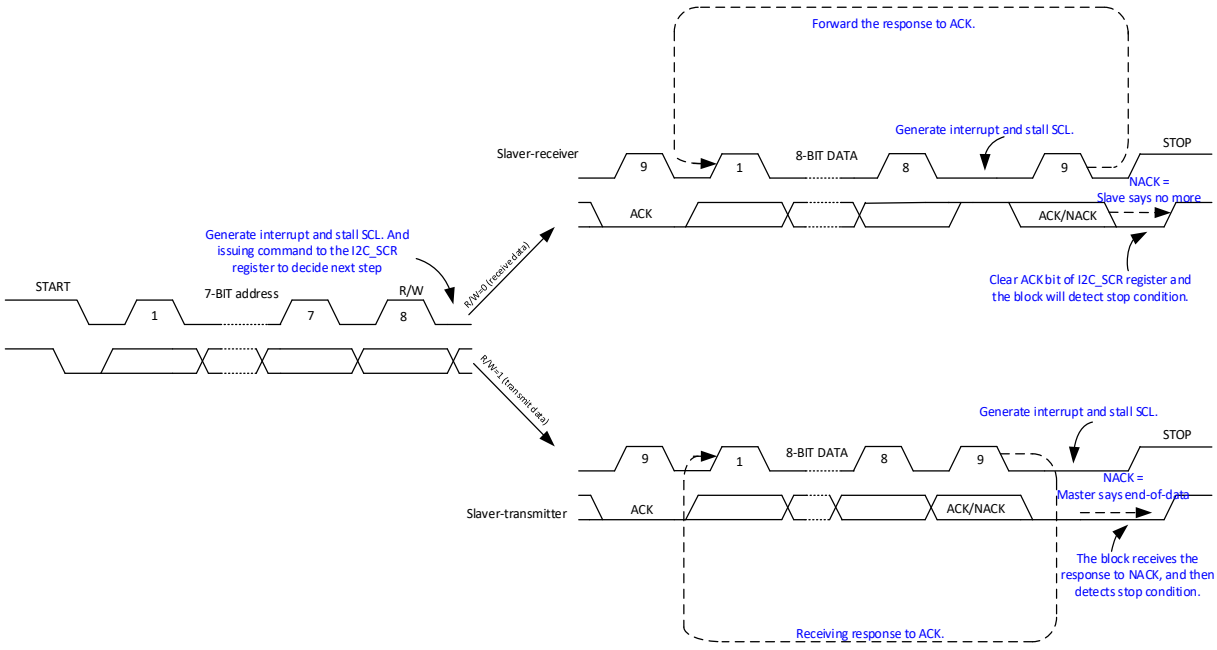


Figure 18-3 Schematic Diagram of Transmission in Slave Mode

18.3.2.2.2 Slave Mode Transport

After the address matches, the slave sends the data from the I2C_DATA register to the SDA line via the internal shift register. Before the I2C_DATA data is ready, the slave device can lower the SCL until the data to be sent has been written to the I2C_DATA register. The I2C interface performs the following operations after sending each byte:

- If the ACK bit is received, the next byte of data is loaded and the transmission continues. The SCL can be lowered during the loading process.
- If the NACK bit is received, stop loading the next byte.
- Wait for the STOP event to end this transmission.

18.3.2.2.3 Slave Mode Single-byte Receive

After the address matches, the slave receiver stores the data received from the SDA line through the internal shift register into the I2C_DATA register. The I2C interface performs the following operations after receiving each byte:

- If the ACK bit is set, an ACK response pulse is generated after a byte is received.
- If the ACK bit is cleared, a NACK response pulse is generated after a byte is received.
- Wait for the STOP event to end this transmission.

18.3.2.3 Master Mode

Both the master mode and slave mode of the I2C interface are turned off by default. If operating in the master mode, the master mode should be enabled. In order to generate the correct timing, the working clock of the I2C interface must be set in the register CLK_DIV0.

Judge whether the bus is idle before performing the transmission via I2C interface in master mode. Read BIT3 of the I2C_MSCR register to query the current bus status. If the bus is busy, turn on the I2C interrupt and determine whether the bus is idle by receiving the STOP interrupt event. Only in the idle state can the START state and subsequent data be sent normally.

18.3.2.3.1 Master Mode Transmission

After each transfer of one byte of data, an interrupt will be generated to determine whether to continue the transfer. The following figure is a schematic diagram of the bus for slave mode transmission. As can be seen from the figure, the transmission process is as follows:

- Determine whether the bus is idle, if it is idle, prepare to start transmission.
- First, send the slave address. If the address is matched, the transmission can be continued; otherwise the transmission will stop.

- In the RX mode, an interrupt is generated after a byte is received, the software determines whether to continue receiving, and returns an ACK/NACK response.
- In the TX mode, an interrupt is generated when receiving the response (ACK/NACK) after a byte is sent, and subsequent operations are judged based on the response.
- Send the bus STOP event, this transmission is completed.

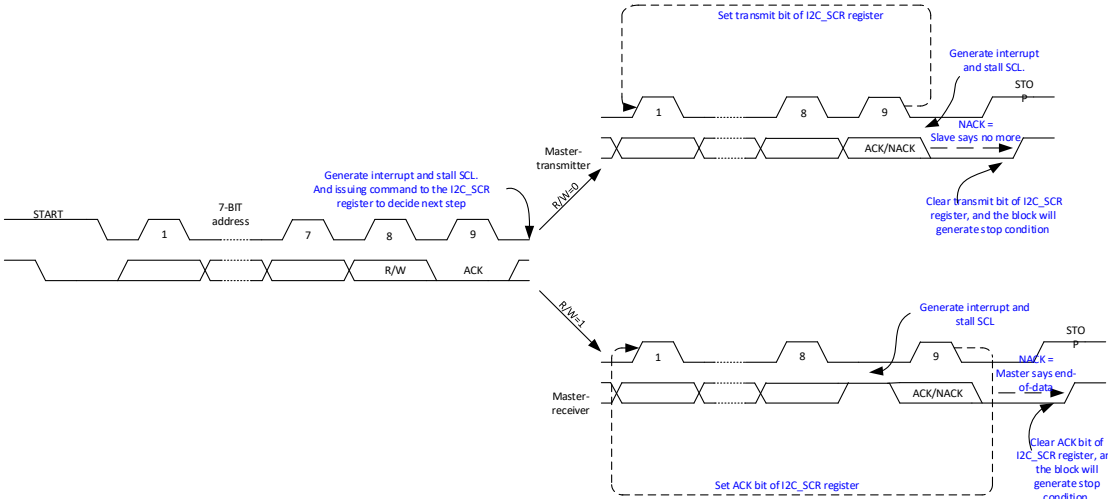


Figure 18-4 Schematic Diagram of Single-byte Transmission in Master Mode

18.3.2.3.2 Master Mode Transport

After the transmission starts, the I2C interface sends the byte from the I2C_DATA register to the SDA line via the internal shift register, and the slave address is sent via the I2C_DATA. Before the I2C_DATA data is ready, the master device may not generate the SCL clock signal until the data to be sent has been written to the I2C_DATA register. The I2C interface performs the following operations after sending each byte:

- If the ACK bit is received, the next byte of data is loaded and the transmission continues. The SCL can be lowered during the loading process.
- If the NACK bit is received, stop loading the next byte.
- The transmission is completed if it is a master computer. Stop the subsequent transmission whether the received bit is ACK/NACK.
- A STOP event is generated to end this transmission.

18.3.2.3.3 Master Mode Receive

After the transmission starts, the I2C interface stores the data received from the SDA line through the internal shift register in the I2C_DATA register, and the slave address is sent via I2C_DATA. The I2C interface performs the following operations after receiving each byte:

- If the ACK bit is set, an ACK response pulse is generated after a byte is received.
- If the ACK bit is cleared, a NACK response pulse is generated after a byte is received.
- The transmission is completed if it is a master computer. Stop the subsequent transmission whether the sent bit is ACK/NACK.
- A STOP event is generated to end this transmission.

18.3.2.4 I2C Bus Exception Handling

During an address or data byte transmission, a bus error is generated when the I2C interface detects an external stop or start condition. Generally speaking, the bus error is caused by interference on the bus, and some I2C devices are not synchronized with the I2C network, resulting in a START event/STOP event sent automatically. According to the I2C protocol, when a bus error occurs, the interface logic of this I2C device must be reset after receiving a START event/STOP event. For the slave device, this operation is OK; for the master device, a bus error will force it to release the bus and reset its I2C interface logic. Since the master device does not respond to external START and STOP events, an interrupt handler function is required to handle this exception after a bus error occurred, and instruct the master device to continue to monitor the bus situation, so as to perform subsequent I2C bus transmission.

For I2C interface: In master mode, bus errors can be detected and bus error interrupts can also be generated; in slave mode, bus errors will trigger address data to be received, while allowing the I2C interface to return to an idle state and generate interrupts.

18.3.2.5 Interrupt Handling

The I2C interface contains three types of interrupt events, including data transmission completion event, bus error event, STOP event, NACK event, and hardware address matching event.

- Data transmission completion event. The current data transmission is completed. Active high, write 0 to clear BIT0 of I2C_SCR.Done.
- Bus error event. During the transmission process, the bus generates an erroneous START event/STOP event. Active high, write 0 to clear I2C_SCR.STT_ERR.
- STOP event. When the current data transmission is completed, the master device sends a STOP event. The slave device receives a STOP event and generates a corresponding interrupt. Active high, write 0 to clear I2C_SCR.STOP_EVT.
- NACK event. The sender receives a NACK response, indicating that the receiver cannot continue subsequent transmissions. Active high, write 0 to clear I2C_SCR.RX_ACK
- Hardware address matching event. The address received in slave mode matches the address of this device, and a corresponding interrupt is generated. Active high, write 0 to clear I2C_SCR.ADDR_DATA.

18.3.2.6 Communication Speed Setting

The working clock of the I2C interface comes from the frequency division of the system clock. The frequency division register is CLK_DIV0 of the SYS module.

The I2C interface adopts a synchronous design, and the signals of external devices should be synchronously sampled. The synchronous clock is the working clock of the I2C interface.

- I2C module operating clock frequency = system frequency/ (CLK_DIV0 + 1)
- The Serial data(SDA) and Serial clock line(SCL) clock frequency = I2C module operating clock frequency/17
- I2C baud rate = I2C module operating clock frequency/17

18.4 Register

18.4.1 Address Allocation

The base address of the I2Cx module register is Yes 0x4001_1000, and the 0x4001_1400. The register list is as follows:

Table 18-1 I2C Register Address Allocation List

Name	Offset	Description
I2Cx_ADDR	0x00	I2C address register
I2Cx_CFG	0x04	I2C configuration register
I2Cx_SCR	0x08	I2C status register
I2Cx_DATA	0x0C	I2C data register
I2Cx_MSCR	0x10	I2C master mode register
I2Cx_BCR	0x14	I2C DMA transfer control register

18.4.2 Address Register (x = 0,1) (I2Cx_ADDR)

The addresses are: 0x4001_1000 and 0x4001_1400.

Reset value: 0x0

Table 18-2 Address Register (I2C_ADDR)

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

Bit field	Bit Name	Description
[31:8]		Unused

[7]	ADDR_CMP	I2C hardware address comparison enable switch. Only effective in DMA mode. The default value is 0. 0: Disable 1: Enable
[6:0]	ADDR	I2C device hardware address in slave mode. The slave device address only needs to be filled in the DMA mode in master mode.

18.4.3 System Control Register (x = 0,1) (I2Cx_CFG)

The addresses are: 0x4001_1004 and 0x4001_1404.

Reset value: 0x0

Table 18-3 System Control Register (I2C_CFG)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								IE	TC_IE	BUS_ERR_IE	STOP_IE			MST_MODE	SLV_MODE
								RW	RW	RW	RW			RW	RW
								0	0	0	0			0	0

Bit field	Bit Name	Description
[31:8]		Unused
[7]	IE	I2C interrupt enable signal. The default value is 0. 1: Enable I2C interrupt 0: Disable I2C interrupt
[6]	TC_IE	I2C data transfer completion interrupt enable signal. The default value is 0. 1: Enable this interrupt source 0: Disable this interrupt source
[5]	BUS_ERR_IE	I2C bus error event interrupt enable signal. The default value is 0. 1: Enable this interrupt source 0: Disable this interrupt source
[4]	STOP_IE	I2C STOP event interrupt enable signal. The default value is 0. 1: Enable this interrupt source 0: Disable this interrupt source
[3:2]		NA
[1]	MST_MODE	I2C master mode enable signal. The default value is 0. 1: Enable master mode 0: Disable master mode
[0]	SLV_MODE	I2C slave mode enable signal. The default value is 0. 1: Enable slave mode 0: Disable slave mode

18.4.4 Status Control Register (x = 0,1) (I2Cx_SCR)

The addresses are: 0x4001_1008 and 0x4001_1408.

Reset value: 0x0

Table 18-4 Status Control Register (I2C_SCR)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								STT_ERR	LOST_ARB	STOP_EVT	BYTE_CMPLT	ADDR_DATA	DATA_DIR	RX_ACK	Done
								RW	RW	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:8]		Unused
[7]	STT_ERR	Bus error status flag. Both master TX/RX modes can be used, write 0 to clear. 0: no START/STOP bus error 1: START/STOP bus error
[6]	LOST_ARB	Bus arbitration lost status flag bit. Both master TX/RX modes can be used. This bit is set when a bus arbitration lost event occurs, no interrupt event is generated, and this bit should be checked in the byte completion interrupt. Any START event on the bus will cause the hardware to clear this bit. 0: No bus arbitration lost error occurred 1: A bus arbitration lost error occurred
[5]	STOP_EVT	STOP event status flag. Both master and slave TX/RX modes can be used, write 0 to clear. 0: no STOP event 1: STOP event occurred
[4]	BYTE_CMPLT	ACK generation control bit. Both master/slave RX modes can be used. The sender sends the current byte and the receiver responds to it. This bit retains to be 0 for sender, and is configured according to the actual condition for receiver. 0: byte transmission is complete, return NACK response 1: byte transmission is completed, return ACK response
[3]	ADDR_DATA	Address data flag. Both master and slave TX/RX modes can be used. After START, the first byte is the address data. This is a prompt bit. write 0 to clear. 0: The data sent or received is not Address data 1: The data sent or received is Address data

[2]	DATA_DIR	Send or receive control bit. This bit is 1 for master/slave TX modes. Trigger transmission and the hardware will reset automatically; this bit is 0 for master/slave RX modes, wait to receive. 0: Receive 1: Triggered sent
[1]	RX_ACK	Receive response flag bit. Both master and slave TX modes can be used. It is used to inform the sender of the receiver's feedback. After receiving the feedback, this bit will be cleared. 0: This I2C interface sends data and receives an ACK response 1: This I2C interface sends data and receives a NACK response
[0]	Done	Transfer completion status flag bit. Both master and slave TX/RX modes can be used. write 0 to clear. 0: transmission undone 1: transmission done

Generally, after entering the interrupt, read the I2C_SCR register to get the current I2C bus status and what stage the current transmission is in; then, write different values to the I2C_SCR, and the software informs the hardware what to do next.

18.4.5 Data Register (x = 0,1) (I2C_DATA)

The addresses are: 0x4001_100C and 0x4001_140C.

Reset value: 0x0

Table 18-5 Data Register (I2C_DATA)

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

Bit field	Bit Name	Description
[31:8]		Unused
[7:0]	DATA	Data register. Master-slave TX/RX mode can be used. The sender writes the sent data; and the receiver reads the received data. Note that the address data also belongs to data, and the master mode can only write the address data to be sent to the register.

18.4.6 Slave Mode Register (x = 0,1) (I2Cx_MSCR)

The addresses are: 0x4001_1010 and 0x4001_1410.

Reset value: 0x0

Table 18-6 Main Mode Register (I2C_MSCR)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												BUSY	MST_CHECK	RESTART	START
												RW	RW	RW	RW
												0	0	0	0

Bit field	Bit Name	Description
[31:4]		Unused
[3]	BUSY	I2C bus, idle and busy state. 0: Detected STOP event, idle. 1: Detected START event, busy.
[2]	MST_CHECK	Master mode scrambles for the bus flag. If the bus is scrambled, set to 1; if the STOP event or bus collision occurs, the module releases the bus and sets to 0.
[1]	RESTART	Trigger the START event again and write 1 is valid. After sending START, the hardware is cleared to 0. Set I2C_CFG [1] to 1 to achieve write "1" operation.
[0]	START	Trigger START event and send address data to the bus, write 1 is valid. Set I2C_CFG [1] to 1 to achieve write "1" operation.

18.4.7 I2C Transmission Control Register (x = 0,1) (I2Cx_BCR)

The addresses are: 0x4001_1004 and 0x4001_1404.

Reset value: 0x0

Table 18-7 DMA Transmission Control Register (I2C_BCR)

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

Bit field	Bit Name	Description
[31:8]		Unused
[7]	NACK_IE	I2C transmission. NACK event interrupt enable signal. 0: Disable this interrupt source 1: Enable this interrupt source
[6]	ADDR_CMP_IE	I2C transmission, hardware address matching interrupt enable signal.

		0: Disable this interrupt source 1: Enable this interrupt source
[5]	BUSRT_EN	I2C multiple data transmission is enabled. DAM is required. 1：使能 0: Disable
[4:0]	--	Reserved

19 SPI

19.1 Overview

The SPI interface is mainly used in application scenarios where the external design uses the SPI protocol. SPI working mode software is optional, the default is SPI Motorola mode. The SPI interface supports full-duplex transmission and half-duplex transmission. When the interface is set in Master mode, it can send clock signals for use by external Slave devices.

19.2 Main Features

- Support Master and Slave mode
- Support full-duplex transmission. Three or four signal lines can be used according to the application.
- Support half-duplex transmission. Twi signal lines can be used according to the application.
- Programmable clock polarity and phase
- Programmable data sequence: MSB or LSB
- The fastest transmission speed is 1/8 of the system's highest clock frequency
- Chip select signals are optional. In the Master mode, the chip select signal can be controlled by software or generated by hardware; in the Slave mode, the chip select signal can be constant and effective, or it can come from an external device
- No local FIFO, including overflow detection and chip select signal anomaly detection

19.3 Function Description

19.3.1 Functional block diagram

This interface uses a synchronous serial design to achieve SPI transmission between the MCU and external devices, and supports polling and interrupt mode to obtain transmission status information. The main functional modules of this interface are shown in the figure below.

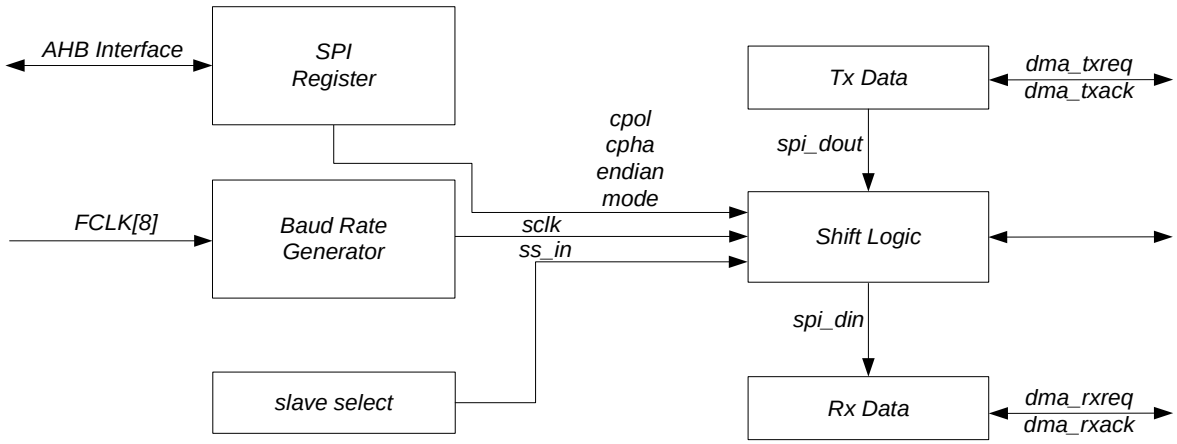


Figure 19-1 List of SPI Module Control Register

Interface signals include spi_din, spi_dout, sclk_in, sclk_out, ss_in and ss_out.

Spi_din: data signal received by the interface. Compared with the SPI protocol, when the interface is configured in Master mode, it is equivalent to MISO; when the interface is configured in Slave mode, it is equivalent to MOSI.

spi_dout: data signal sent by the interface. Compared with the SPI protocol, when the interface is configured in Master mode, it is equivalent to MOSI; when the interface is configured in Slave mode, it is equivalent to MISO.

sclk_in: clock signal received by the interface. The working mode of the interface is Slave, and this signal input is invalid in non-Slave mode.

sclk_out: clock signal sent by the interface. The working mode of the interface is Master, and in non-Master mode, the signal output is always 0.

ss_in: chip select signal received by the interface. The working mode of the interface is Slave, and this signal input is invalid in non-Slave mode.

ss_out: chip select signal sent by the interface. The working mode of the interface is Master, and this signal output is always 1 in non-Master mode.

19.3.2 Function Description

19.3.2.1 Full-duplex Mode

The SPI interface is configured in full-duplex mode by default. Thus, two data lines are required for data transmission. The change of the data signal occurs on the edge of the clock signal, which is synchronized with the clock signal.

When the interface is in Master mode:

- spi_din is the data input, connected to the MISO of the external Slave device
- spi_dout is the data output, connected to the MOSI of the external Slave device

- spi_ss_out is a chip select signal, choose whether to use this signal or software to control other GPIO implementation according to the application

When the interface is in Slave mode:

- spi_din is the data input, connected to the MOSI of the external master device
- spi_dout is the data output, connected to the MISO of the external master device
- spi_ss_in is the chip select signal, depending on whether the signal is used or the chip select is always valid

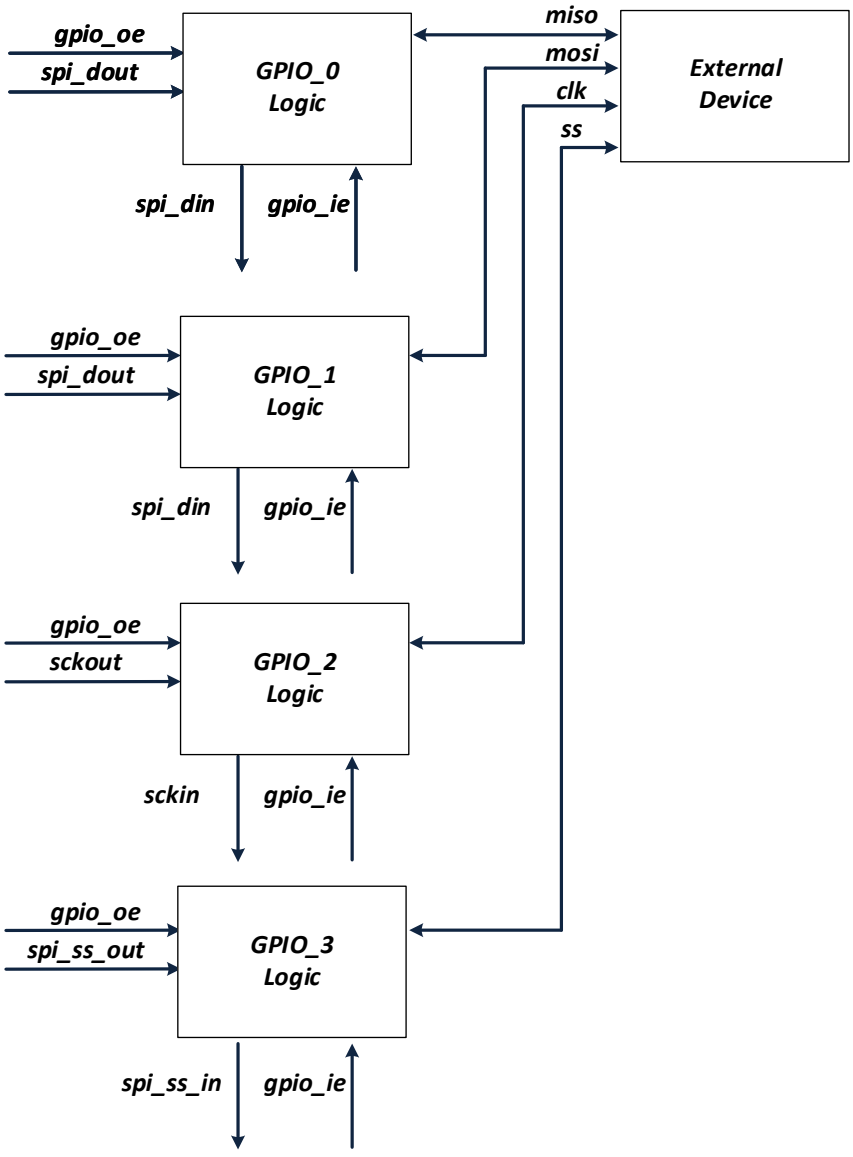


Figure 19-2 SPI Interface Full Duplex Mode Interconnection Block Diagram

As can be seen from the above figure, if GPIO is configured as an output, the SPI interface can send data; if GPIO is configured as an input, the SPI interface can receive data.

19.3.2.2 Half-duplex Mode

The SPI interface can be set in half-duplex mode. Thus, only one data line is needed for data transmission. The change of the data signal occurs on the edge of the clock signal, which is synchronized with the clock signal. A transmission can only be in one direction, either transmitting or receiving.

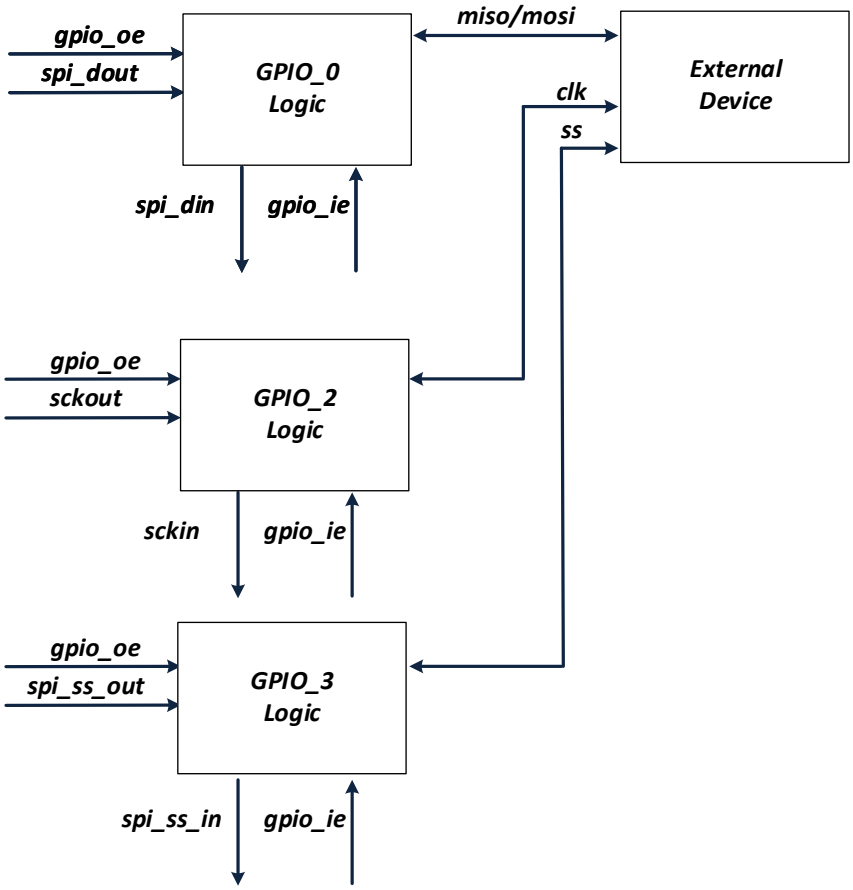


Figure 19-3 SPI Interface Full Duplex Mode Interconnection Block Diagram

Note that if the interface is a Master in the above figure, CLK is the output signal of the interface; if the interface is the Slave, CLK is the input signal of the interface.

Transmit only

SPI_CFG. DUPLEX is set to 2, and half-duplex transmission mode is valid. This interface can only transmit data. GPIO_0's oe is enabled, sending spi_dout data to the outside world; GPIO_0's ie is off, and the spi_din constant input is 0. It supports sending in Master/Slave mode.

Receive only

SPI_CFG. DUPLEX is set to 3, and half-duplex reception mode is valid. This interface can only receive data. GPIO_0's oe is off, spi_dout cannot send data to the outside world; GPIO_0's ie is on, and spi_din receives data from the outside. In this mode, it supports reception in Master/Slave mode.

Note that in full-duplex mode, two GPIOs are used for data transmission. In half-duplex mode, one GPIO can be selected for data transmission.

19.3.2.3 Chip Select Signal

When this interface is in Slave mode, the chip select signal is optional, and CFG [5] determines the chip select source. ss is the strobe enable signal sent by the master device. Active low.

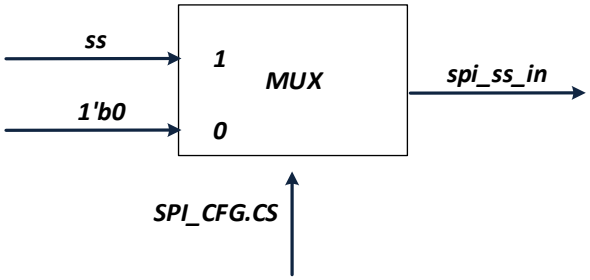


Figure 19-4 SPI Module Chip Select Signal Selection in Slave Mode

When this interface is in Master mode, the chip select signal is also selectable. The module hardware generates a standard chip-select signal, which can be shielded by actual application by software operating additional GPIO.

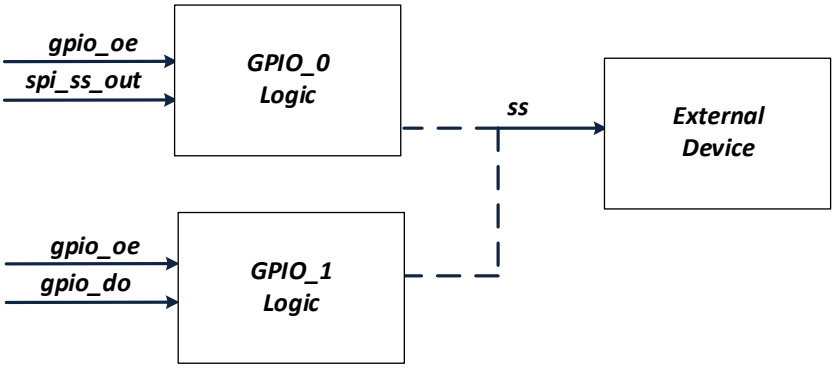


Figure 19-5 SPI Module Chip Select Signal Selection in Master Mode

Note that the dotted line in Figure 16-5 only indicates uncertainty. If spi_ss_out is used as the source of ss, then GPIO_0 is interconnected with external devices; if software is used to operate GPIO, GPIO_1 can be interconnected with external devices.

19.3.2.4 Communication Format

In the SPI communication process, the sending or receiving operation is based on the SPI clock. The communication format is controlled by SPI_CFG. SAMPLE and SPI_CLK_POL. SPI_CFG. SAMPLE is Phase control bit and SPI_CLK_POL is Polarity control bit.

Polarity controls the level status of the SPI clock signal by default. When Polarity is 0, the default clock level is low; when Polarity is 1, the default level is high.

Phase controls the transmission/reception time of SPI data. When Phase is 0, the clock transitions from the default level to the first transition edge is the time to sample data, and when Phase is 1, the clock transitions from the default level to the first transition edge is the time to transmit data.

19.3.2.5 Data Format and Length

SPI data transmission format is divided into two types: MSB and LSB. The data transmission format is controlled by SPI_CFG.ENDIAN. Note that the hardware automatically converts the transmission format during data transmission without software conversion.

SPI data length is configurable from 8 bits to 16 bits. SPI_SIZE.BITSIZE controls the length.

19.3.2.6 Transmission

MCU transmission can only send/receive one byte at a time, and should judge whether the transfer is completed by interruption or polling after each completion. Whether in master mode or slave mode, only writing SPI_TX_DATA register can trigger the transmission. The master mode is active transmission, and the slave mode is to load data to the sending queue and wait for the clock signal from the master mode to start transmission. The recommended software configuration process for MCU transmission is as follows:

- Initialize the GPIO module and set the GPIO multiplexed with SPI.
- Initialize the SPI interface, and set SPI_IE/SPI_CFG/SPI_BAUD/SPI_SIZE and other registers.
- The MCU performs a write operation on the SPI_TX_DATA register, triggering the SPI interface to enter the transmission process. In slave mode, the data is loaded to the internal status machine and wait for the master device to initiate a read operation. In master mode, trigger to send.

Support continuous transmission, which is controlled by SPI_BAUD.TRANS_MODE. This is mainly applicable to master mode.

In discontinuous mode, a complete output transmission is: the chip select signal is valid, and SPI_TX_DATA writes the send value to trigger the data transmission with the length of SPI_SIZE.BITSIZE; after completing transmission, the chip select signal will become invalid.

In continuous mode, a complete output transmission is: the chip select signal is valid, and SPI_TX_DATA writes the send value to trigger the data transmission; after completing the transmission of data with the length of SPI_SIZE.BITSIZE, SPI_TX_DATA writes a new value to trigger the next data transmission with the length of SPI_SIZE.BITSIZE; the chip select signal remains valid until the batch of data is transmitted.

19.3.2.7 Interrupt Handling

The SPI interface contains three types of interrupt events, including data transmission completion event, abnormal event and overflow event.

- Data transmission completion event. Current data has been transferred. Active high, write 1 to SPI_IE. CMPLT_IF to clear.
- For abnormal events, the SPI interface is in Slave mode. If the chip select signal is disturbed during transmission and is pulled high, a chip select abnormal event will occur. Active high, write 1 to SPI_IE. AB_IF to clear.

- Overflow event. If the data is not returned to RAM through DMA in time, or data is obtained from RAM, an overflow event will occur. Active high, write 1 to SPI_IE. OV_IF to clear.

The above events do not trigger the SPI interrupt by default, but can set SPI_IE[7:4] to enable the event to generate an interrupt.

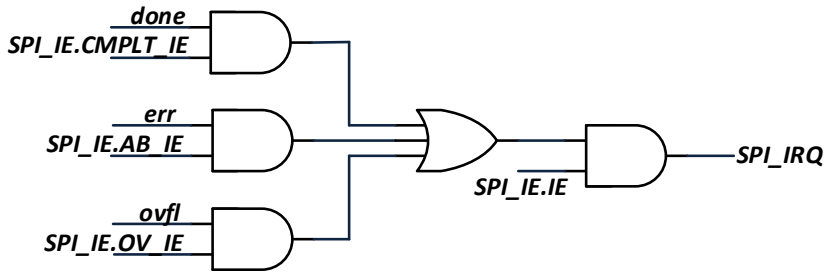


Figure 19-6 Generation Diagram for SPI Module Interrupt Selection Signal

19.3.2.8 Baud rate setting

The SPI interface clock is obtained by dividing the system clock by the frequency division factor from SPI_BAUD. BAUD. The calculation formula for SPI transmission baud rate is:

SPI transmission baud rate = System clock/(2*(BAUD + 1))

The SPI protocol is a half-shot protocol. The rising edge sends data and the falling edge collects data; or the falling edge sends data and the rising edge uses data.

The SPI interface adopts a synchronous design, and the signals of external devices need to be synchronously sampled. The synchronous clock is the system clock. Synchronization of data and clock signals (Slave mode) requires two beats of the system clock. Considering the clock phase shift, the redundancy of the system clock is required at this time. It is deduced from this that the fastest BAUD rate is 1/8 of the system clock, the high-level period is four-beat system clock, and the low-level period is four-beat system clock. Therefore, the set value of SPI_BAUD. BAUD cannot be less than 3.

19.4 Register

19.4.1 Address Allocation

The base address of the SPI0 module register is 0x4001_A800. The base address of the SPI1 module register is 0x4001_AC00, and the register list is as follows:

Table 19-1 List of SPI Module Control Register

Name	Offset	Description
SPI_CFG	0x00	SPI Configuration Register
SPI_IE	0x04	SPI Interrupt Register
SPI_BAUD	0x08	SPI Baud Rate Setting Register
SPI_TXDATA	0x0C	SPI Transmit Data Register
SPI_RXDATA	0x10	SPI Receive Data Register

SPI_BITSIZE	0x14	SPI transfer data byte length register

19.4.2 System Control Register (SPI_CFG)

Address: 0x4001_A800, 0x4001_AC00

Reset value: 0x0

Table 19-2 System Control Register (SPI_CFG)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								DUPLEX		CS	MS	CPHA	CPOL	ENDIAN	EN
								RW		RW	RW	RW	RW	RW	RW
								0		1	0	0	0	0	0

Bit field	Bit Name	Description
[31:8]		Unused
[7:6]	DUPLEX	Half-duplex mode setting 0X: Turn off half-duplex mode 10: Turn on half-duplex mode, transmit only 11: Turn on half-duplex mode, receive only
[5]	CS	Source of chip select signal under SPI slave device. The default value is 1. 0: The chip select signal in Slave mode is always a valid value --0 1: The chip select signal in Slave mode comes from the Master device
[4]	MS	SPI master-slave mode selection. The default value is 0. 0: Slave mode 1: Master mode
[3]	CPHA	SPI phase selection. The default value is 0. 0: Phase is 0 1: Phase is 1
[2]	CPOL	SPI polarity selection. The default value is 0. 0: Polarity is 0 1: Polarity is 1
[1]	ENDIAN	SPI module transmission sequence. The default value is 0. 0: MSB, high bit is transmitted first 1: LSB, low bit is transmitted first
[0]	EN	SPI module enable signal. The default value is 0. 0: turn off the SPI module 1: turn on the SPI module

19.4.3 SPI Interrupt Register (SPI_IE)

Address: 0x4001_A804, 0x4001_AC04

Reset value: 0x0

Table 19-3 SPI Interrupt Register (SPI_IE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								IE	CMPLT_IE	AB_IE	OV_IE	TRANS_TRIG	CMPLT_IF	AB_IF	OV_IF
								RW	RW	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:8]		Unused
[7]	IE	SPI interrupt enable switch. The default value is 0. 0: Disable SPI interrupt 1: Enable SPI interrupt
[6]	CMPLT_IE	SPI transmission, complete event interrupt enable signal. 0: Disable this interrupt source 1: Enable this interrupt source
[5]	AB_IE	SPI transmission, abnormal event interrupt enable signal. 0: Disable this interrupt source 1: Enable this interrupt source
[4]	OV_IE	SPI transmission, interrupt enable signal for overflow event. The default value is 0. 0: Disable this interrupt source 1: Enable this interrupt source
[3]	TRANS_TRIG	Transmission trigger selection. 1: external trigger 0: internally executed automatically. Only the master mode is valid
[2]	CMPLT_IF	SPI transmission, complete event. Active high, write 1 to clear.
[1]	AB_IF	SPI transmission, abnormal events. In Slave mode, the transmission is not completed, and the chip select signal invalid event occurs. Active high, write 1 to clear.
[0]	OV_IF	SPI transmission, overflow event. The old data received last time has not been taken away, the new data received this time has arrived. Active high, write 1 to clear.

19.4.4 Baud Rate Setting Register (SPI_BAUD)

Address: 0x4001_A808, 0x4001_AC08

Reset value: 0x0

Table 19-4 Baud Rate Setting Register (SPI_BAUD)

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

Bit field	Bit Name	Description
[15]	TRANS_MODE	SPI data transfer method. The default is 0, DMA mode. 0: The SPI interface supports DMA to move data to the SPI interface to complete data transfer and receiving. 1: The SPI interface supports the MCU to move data to the SPI interface to complete data transfer and receiving.
[14:12]	--	--
[11:0]	BAUD	SPI transmission baud rate configuration. The calculation formula of SPI actual transmission speed is: SPI transmission speed = system clock/(2*(BAUD+1)) Remember, the set value of BAUD cannot be less than 3.

19.4.5 SPI Transmit Data Register (SPI_TXDATA)

Address: 0x4001_A80C, 0x4001_AC0C

Reset value: 0x0

Table 19-5 SPI Transmit Data Register (SPI_TXDATA)

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

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	TX_DATA	SPI Transmit Data Register

19.4.6 SPI Receive Data Register (SPI_RXDATA)

Address: 0x4001_A810, 0x4001_AC10

Reset value: 0x0

Table 19-6 SPI Receive Data Register (SPI_RXDATA)

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

RW
0

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	RX_DATA	SPI Receive Data Register

19.4.7 SPI Data Byte Length Register (SPI_SIZE)

Address: 0x4001_A814, 0x4001_AC14

Reset value: 0x0

Table 19-7 SPI Data Byte Length Register (SPI_BITSIZE)

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

Bit field	Bit Name	Description
[31:8]		Unused
[4:0]	BITSIZE	Byte length register. 0x00: Invalid 0x07: Invalid 0x08：8-Bit 0x09：9-Bit … 0x0E：14-Bit 0x0F：15-Bit 0x10：16-Bit

20 Comparator (CMP)

20.1 Overview

The comparator signal processing module (Hereinafter referred to as CMP module. For better distinguish, the analog comparator in the following figure is represented by Comparator, and the digital CMP module is represented by CMP) is used to process the output signals generated by the two analog rail-to-rail comparators and consists of a series of digital circuits such as enable, polarity control, and filtering. The signal processing clock is obtained by dividing the main clock. This module is also used to generate a comparator interrupt to the CPU.

CMP0/1 can use the 4 P channels of MCPWM0 for window control, and CMP2/3 can use the 4 P channels of MCPWM1 for window control. CMP4/5 does not support window control.

The unfiltered original output value of the analog comparator can be obtained by reading the CMP_DATA value. The unfiltered original output value of the analog comparator can also be sent by configuring the second function of GPIO. For specific GPIO second function configuration and introduction location, please refer to the device datasheet.

For more information about the analog comparator, including the selection of its input signal and the configuration of hysteresis, please refer to Chapter 5.1.6.

20.2 Register

20.2.1 Address Allocation

The base address of the CMP module register is 0x4001_3000, and the register list is as follows:

Table 20-1 Comparator Register List

Name	Offset Address	Description
CMP_IE	0x00	Comparator interrupt enable register
CMP_IF	0x04	Comparator interrupt flag register
CMP_TCLK	0x08	Comparator divider clock control register
CMP_CFG	0x0C	Comparator control register
CMP_BLCWIN0	0x10	Comparator window control register 0
CMP_BLCWIN1	0x14	Comparator window control register 1
CMP_DATA	0x18	Comparator output value register

20.2.2 Comparator Interrupt Enable Register (CMP_IE)

Address: 0x4001_3000

Reset value: 0x0

Table 20-2 Comparator Interrupt Enable Register (CMP_IE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		CMP5_RE	CMP4_RE	CMP3_RE	CMP2_RE	CMP1_RE	CMP0_RE			CMP5_IE	CMP4_IE	CMP3_IE	CMP2_IE	CMP1_IE	CMP0_IE
		RW	RW	RW	RW	RW	RW			RW	RW	RW	RW	RW	RW
		0	0	0	0	0	0			0	0	0	0	0	0

Bit field	Bit Name	Description
[31:14]		Unused
[13]	CMP5_RE	Comparator 5 request enable, active high
[12]	CMP4_RE	Comparator 4 request enable, active high
[11]	CMP3_RE	Comparator 3 request enable, active high
[10]	CMP2_RE	Comparator 2 request enable, active high
[9]	CMP1_RE	Comparator 1 request enable, active high
[8]	CMP0_RE	Comparator 0 request enable, active high
[7:6]		Unused
[5]	CMP5_IE	Comparator 5 interrupt enable, active high
[4]	CMP4_IE	Comparator 4 interrupt enable, active high
[3]	CMP3_IE	Comparator 3 interrupt enable, active high
[2]	CMP2_IE	Comparator 2 interrupt enable, active high
[1]	CMP1_IE	Comparator 1 interrupt enable, active high
[0]	CMP0_IE	Comparator 0 interrupt enable, active high

Because CMP signal may be a slow level signal, it is recommended that the interrupt trigger type be edge trigger when using CMP to trigger DMA.

20.2.3 Comparator Interrupt Flag Register (CMP_IF)

Address: 0x4001_3004

Reset value: 0x0

Table 20-3 Comparator Interrupt Flag Register (CMP_IF)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										CMP5_IF	CMP4_IF	CMP3_IF	CMP2_IF	CMP1_IF	CMP0_IF
										RW	RW	RW	RW	RW	RW
										0	0	0	0	0	0

Bit field	Bit Name	Description

[31:6]		Unused
[5]	CMP5_IF	Comparator 5 interrupt flag, active high, write 1 to clear
[4]	CMP4_IF	Comparator 4 interrupt flag, active high, write 1 to clear
[3]	CMP3_IF	Comparator 3 interrupt flag, active high, write 1 to clear
[2]	CMP2_IF	Comparator 2 interrupt flag, active high, write 1 to clear
[1]	CMP1_IF	Comparator 1 interrupt flag, active high, write 1 to clear
[0]	CMP0_IF	Comparator 0 interrupt flag, active high, write 1 to clear

20.2.4 Comparator Divider Clock Control Register (CMP_TCLK)

Address: 0x4001_3008

Reset value: 0x0

Table 20-4 Comparator Divider Clock Control Register (CMP_TCLK)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								FIL_CLK54_DIV16				CLK54_EN	FIL_CLK54_DIV2		
								RW				RW	RW		
								0				0	0		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIL_CLK32_DIV16				CLK32_EN	FIL_CLK32_DIV2			FIL_CLK10_DIV16				CLK10_EN	FIL_CLK10_DIV2		
RW				RW	RW			RW				RW	RW		
0				0	0			0				0	0		

Bit field	Bit Name	Description
[31:24]		Unused
[23:20]	FIL_CLK54_DIV16	Comparator 5/4 filter clock frequency division, based on MCLK to divide by 1 to 16, affecting the time to enter the comparator interrupt
[19]	CLK54_EN	Comparator 5/4 filter clock enable, active high
[18:16]	FIL_CLK54_DIV2	Comparator 5/4 filter clock frequency division: 3'h0:1 frequency division, 3'h1:2 frequency division, 3'h2:4 frequency division, 3'h3:8 frequency division, 3'h4:16 frequency division, 3'h5:32

		frequency division, 3'h6:64 frequency division, and 3'h7:128 frequency division
[15:12]	FIL_CLK32_DIV16	Comparator 3/2 filter clock frequency division, based on MCLK to divide by 1 to 16, affecting the time to enter the comparator interrupt
[11]	CLK32_EN	Comparator 3/2 filter clock enable, active high
[10:8]	FIL_CLK32_DIV2	Comparator 3/2 filter clock frequency division: 3'h0:1 frequency division, 3'h1:2 frequency division, 3'h2:4 frequency division, 3'h3:8 frequency division, 3'h4:16 frequency division, 3'h5:32 frequency division, 3'h6:64 frequency division, and 3'h7:128 frequency division
[7:4]	FIL_CLK10_DIV16	Comparator filter clock frequency division, based on MCLK to divide by 1 to 16, affecting the time to enter the comparator interrupt
[3]	CLK10_EN	Comparator 1/0 filter clock enable, active high
[2:0]	FIL_CLK10_DIV2	Comparator 1/0 filter clock frequency division: 3'h0:1 frequency division, 3'h1:2 frequency division, 3'h2:4 frequency division, 3'h3:8 frequency division, 3'h4:16 frequency division, 3'h5:32 frequency division, 3'h6:64 frequency division, and 3'h7:128 frequency division

CMP filter clock frequency

Freq(CMP_Filter) = Freq(MCLK)/2CMP_TCLK.FIL_CLK_DIV2/(CMP_TCLK.FIL_CLK_DIV16+1), where MCLK is the main clock. Note that the CMP_TCLK.CLK_EN bit should be enabled to generate the CMP filter clock.

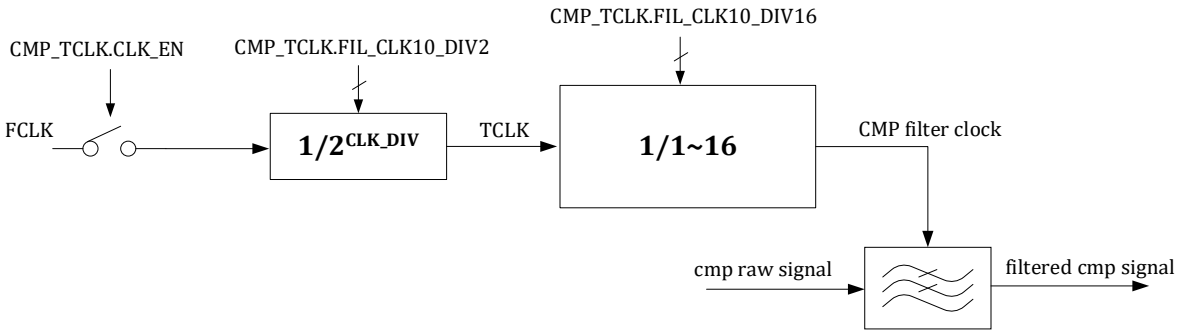


Figure 20-1 Comparator Register List

The CMP module uses this filter clock to filter the output signal of the analog comparator for sixteen clock cycles, that is, only the signal stabilization time exceeds sixteen filter clock cycles to pass the filter. The filtered signal output by the CMP module will change. If the input signal is stable for less than sixteen filter clock cycles, the filtered signal output by the CMP module will remain unchanged. **Filter width = filter clock period*16.**

Because the filter clock can divide frequency, the filter width range of CMP signal is

1*1*16~128*16*16 bus cycles, that is, 16~32768 bus cycles.

20.2.5 Comparator Control Register (CMP_CFG)

Address: 0x4001_300C

Reset value: 0x0

Table 20-5 Comparator Control Register (CMP_CFG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								CMP5_W_PWM_POL	CMP5_IRQ_TRIG	CMP5_IN_EN	CMP5_POL	CMP4_W_PWM_POL	CMP4_IRQ_TRIG	CMP4_IN_EN	CMP4_POL
								RW	RW	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP3_W_PWM_POL	CMP3_IRQ_TRIG	CMP3_IN_EN	CMP3_POL	CMP2_W_PWM_POL	CMP2_IRQ_TRIG	CMP2_IN_EN	CMP2_POL	CMP1_W_PWM_POL	CMP1_IRQ_TRIG	CMP1_IN_EN	CMP1_POL	CMP0_W_PWM_POL	CMP0_IRQ_TRIG	CMP0_IN_EN	CMP0_POL
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:24]		Unused
[23]	CMP5_W_PWM_POL	Comparator 5 window PWM signal polarity selection, used when CMP_BLCWIN1 is enabled
[22]	CMP5_IRQ_TRIG	Comparator 5 interrupt trigger type, 0: level trigger, 1: edge trigger
[21]	CMP5_IN_EN	Comparator 5 signal input enable
[20]	CMP5_POL	Comparator 5 polarity selection, 0: active high; 1: active low
[19]	CMP4_W_PWM_POL	Comparator 4 window PWM signal polarity selection, used when CMP_BLCWIN1 is enabled
[18]	CMP4_IRQ_TRIG	Comparator 4 interrupt trigger type, 0: level trigger, 1: edge trigger
[17]	CMP4_IN_EN	Comparator 4 signal input enable
[16]	CMP4_POL	Comparator 4 polarity selection, 0: active high; 1: active low
[15]	CMP3_W_PWM_POL	Comparator 3 window PWM signal polarity selection, used when CMP_BLCWIN0 is enabled
[14]	CMP3_IRQ_TRIG	Comparator 3 interrupt trigger type, 0: level trigger, 1: edge trigger
[13]	CMP3_IN_EN	Comparator 3 signal input enable
[12]	CMP3_POL	Comparator 3 polarity selection, 0: active high; 1: active low
[11]	CMP2_W_PWM_POL	Comparator 2 window PWM signal polarity selection, used when

		CMP_BLCWIN0 is enabled
[10]	CMP2_IRQ_TRIG	Comparator 2 interrupt trigger type, 0: level trigger, 1: edge trigger
[9]	CMP2_IN_EN	Comparator 2 signal input enable
[8]	CMP2_POL	Comparator 2 polarity selection, 0: active high; 1: active low
[7]	CMP1_W_PWM_POL	Comparator 1 window PWM signal polarity selection, used when CMP_BLCWIN0 is enabled
[6]	CMP1_IRQ_TRIG	Comparator 1 interrupt trigger type, 0: level trigger, 1: edge trigger
[5]	CMP1_IN_EN	Comparator 1 signal input enable
[4]	CMP1_POL	Comparator 1 polarity selection, 0: active high; 1: active low
[3]	CMP0_W_PWM_POL	Comparator 0 window PWM signal polarity selection, used when CMP_BLCWIN0 is enabled
[2]	CMP0_IRQ_TRIG	Comparator 0 interrupt trigger type, 0: level trigger, 1: edge trigger
[1]	CMP0_IN_EN	Comparator 0 signal input enable
[0]	CMP0_POL	Comparator 0 polarity selection, 0: active high; 1: active low

The polarity and enable control of the comparator 0/1 are as shown in Figure 20-2

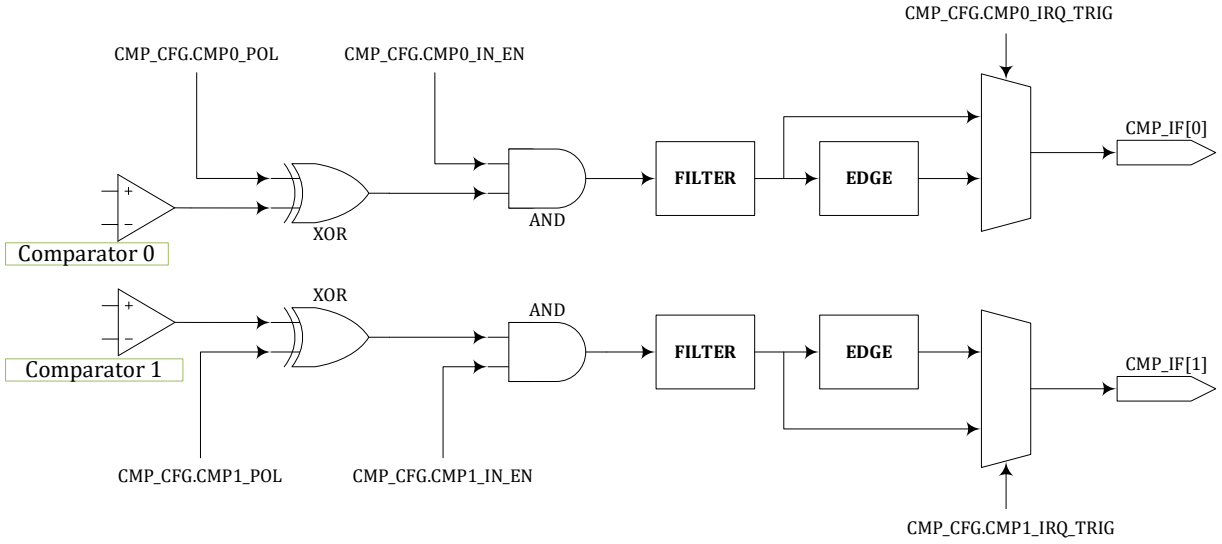


Figure 20-2 Comparator Control and Interrupt Generation Logic (taking CMP0/1 as an example)

The comparator module and the MCPWM module can work together, and the P-tube control signal of the MCPWM module can be used as the control signal for comparator window. However, the interrupt signal of the comparator itself is generated regardless of the window control and is only affected by the CMP_CFG register.

The fail signal of MCPWM can come from GPIO or from the comparator module, and is controlled by the MCPWM_FAIL register. If the fail signal of MCPWM comes from the comparator, it is controlled by the window inside the comparator module. After the fail signal enters MCPWM, it will also be processed with polarity enable and filtering. It is similar to the comparator module but completely independent, and is controlled by the register inside MCPWM. The error interrupt signal related to fail in MCPWM is affected by the polarity-enable filter control register in MCPWM. For details, please refer to the MCPWM chapter.

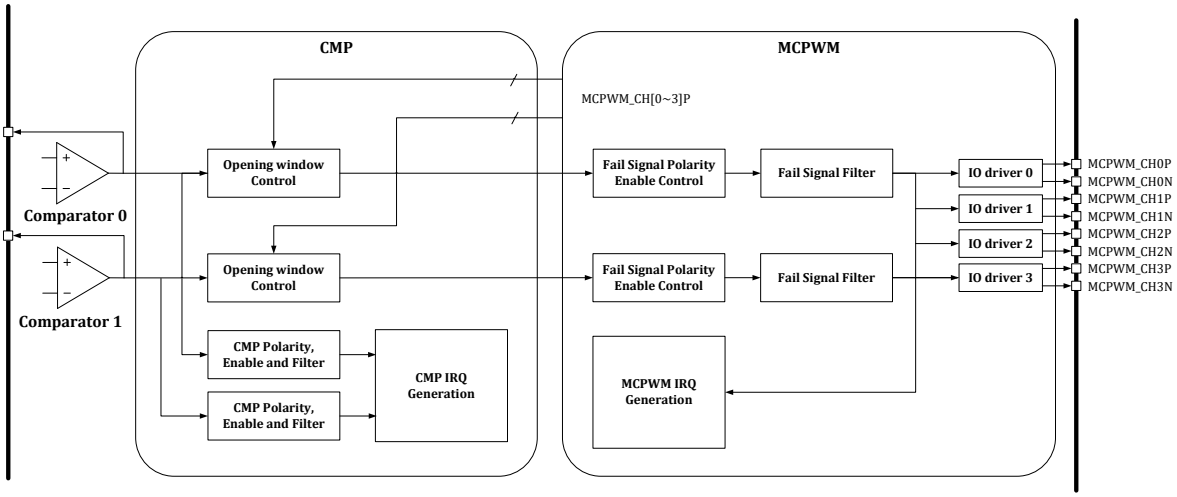


Figure 20-3 CMP and MCPWM Linkage

For the window function of the comparator, if CMP_CFG.CMP0_PWM_POL = 1, then when the corresponding MCPWM CHNx_P signal is 1, the comparator 0 can generate a comparison signal output, and the comparison signal is 0 at other times. Conversely, if CMP_CFG.CMP0_PWM_POL = 0, then when the corresponding MCPWM CHNx_P signal is 0, the comparator 0 can generate a comparison signal output, and the comparison signal is 0 at other times.

Note : CMP_CFG.CMP0_PWM_POL and CMP_CFG.CMP1_PWM_POL will also affect the comparator signal sent to the MCPWM module as a FAIL signal, as shown in Figure 20-4 . The MCPWM FAIL signal from the comparator is the original signal of analog comparator output, which is not filtered by the comparator digital interface module but can be window controlled by MCPWM channel signal. For settings of window control, refer to the comparator digital interface module. After the FAIL signal enters the MCPWM module, it can be filtered by MCPWM_TCLK.

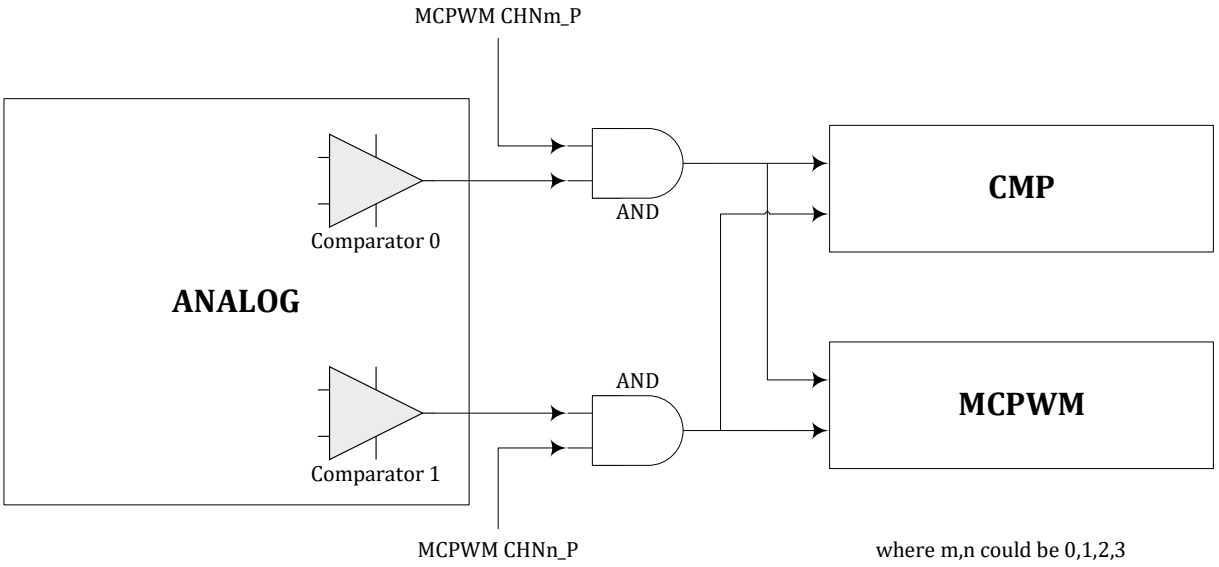


Figure 20-4 Comparator Window Function Diagram

20.2.6 Comparator Window Control Register 0 (CMP_BLCWIN0)

Address: 0x4001_3010

Reset value: 0x0

Table 20-6 Comparator Window Control Register 0 (CMP_BLCWIN0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP3_CHN3P_WIN_EN	CMP3_CHN2P_WIN_EN	CMP3_CHN1P_WIN_EN	CMP3_CHN0P_WIN_EN	CMP2_CHN3P_WIN_EN	CMP2_CHN2P_WIN_EN	CMP2_CHN1P_WIN_EN	CMP2_CHN0P_WIN_EN	CMP1_CHN3P_WIN_EN	CMP1_CHN2P_WIN_EN	CMP1_CHN1P_WIN_EN	CMP1_CHN0P_WIN_EN	CMP0_CHN3P_WIN_EN	CMP0_CHN2P_WIN_EN	CMP0_CHN1P_WIN_EN	CMP0_CHN0P_WIN_EN
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:16]		Reserved
[15]	CMP3_CHN3P_WIN_EN	Use the P-tube switch control signal output from the CHN3_P channel of the MCPWM1 module as the comparator 3 window enable
[14]	CMP3_CHN2P_WIN_EN	Use the P-tube switch control signal output from the CHN2_P channel of the MCPWM1 module as the comparator 3 window enable
[13]	CMP3_CHN1P_WIN_EN	Use the P-tube switch control signal output from the CHN1_P channel of the MCPWM1 module as the comparator 3 window enable
[12]	CMP3_CHN0P_WIN_EN	Use the P-tube switch control signal output from the CHN0_P channel of the MCPWM1 module as the comparator 3 window enable
[11]	CMP2_CHN3P_WIN_EN	Use the P-tube switch control signal output from the CHN3_P channel of the MCPWM1 module as the comparator 2 window enable
[10]	CMP2_CHN2P_WIN_EN	Use the P-tube switch control signal output from the CHN2_P channel of the MCPWM1 module as the comparator 2 window enable
[9]	CMP2_CHN1P_WIN_EN	Use the P-tube switch control signal output from the CHN1_P channel of the MCPWM1 module as the comparator 2 window enable
[8]	CMP2_CHN0P_WIN_EN	Use the P-tube switch control signal output from the CHN0_P channel of the MCPWM1 module as the comparator 2 window enable

[7]	CMP1_CHN3P_WIN_EN	Use the P-tube switch control signal output from the CHN3_P channel of the MCPWM0 module as the comparator 1 window enable
[6]	CMP1_CHN2P_WIN_EN	Use the P-tube switch control signal output from the CHN2_P channel of the MCPWM0 module as the comparator 1 window enable
[5]	CMP1_CHN1P_WIN_EN	Use the P-tube switch control signal output from the CHN1_P channel of the MCPWM0 module as the comparator 1 window enable
[4]	CMP1_CHN0P_WIN_EN	Use the P-tube switch control signal output from the CHN0_P channel of the MCPWM0 module as the comparator 1 window enable
[3]	CMP0_CHN3P_WIN_EN	Use the P-tube switch control signal output from the CHN3_P channel of the MCPWM0 module as the comparator 0 window enable
[2]	CMP0_CHN2P_WIN_EN	Use the P-tube switch control signal output from the CHN2_P channel of the MCPWM0 module as the comparator 0 window enable
[1]	CMP0_CHN1P_WIN_EN	Use the P-tube switch control signal output from the CHN1_P channel of the MCPWM0 module as the comparator 0 window enable
[0]	CMP0_CHN0P_WIN_EN	Use the P-tube switch control signal output from the CHN0_P channel of the MCPWM0 module as the comparator 0 window enable

20.2.7 Comparator Window Control Register 1 (CMP_BLCWIN1)

Address: 0x4001_3014

Reset value: 0x0

Table 20-7 Comparator Window Control Register 1 (CMP_BLCWIN0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP5_1CHN3P_WIN_EN	CMP5_1CHN2P_WIN_EN	CMP5_1CHN1P_WIN_EN	CMP5_1CHN0P_WIN_EN	CMP5_0CHN3P_WIN_EN	CMP5_0CHN2P_WIN_EN	CMP5_0CHN1P_WIN_EN	CMP5_0CHN0P_WIN_EN	CMP4_1CHN3P_WIN_EN	CMP4_1CHN2P_WIN_EN	CMP4_1CHN1P_WIN_EN	CMP4_1CHN0P_WIN_EN	CMP4_0CHN3P_WIN_EN	CMP4_0CHN2P_WIN_EN	CMP4_0CHN1P_WIN_EN	CMP4_0CHN0P_WIN_EN
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:16]		Reserved

[15]	CMP5_1CHN3P_WIN_EN	Use the P-tube switch control signal output from the CHN3_P channel of the MCPWM1 module as the comparator 5 window enable
[14]	CMP5_1CHN2P_WIN_EN	Use the P-tube switch control signal output from the CHN2_P channel of the MCPWM1 module as the comparator 5 window enable
[13]	CMP5_1CHN1P_WIN_EN	Use the P-tube switch control signal output from the CHN1_P channel of the MCPWM1 module as the comparator 5 window enable
[12]	CMP5_1CHN0P_WIN_EN	Use the P-tube switch control signal output from the CHN0_P channel of the MCPWM1 module as the comparator 5 window enable
[11]	CMP5_0CHN3P_WIN_EN	Use the P-tube switch control signal output from the CHN3_P channel of the MCPWM0 module as the comparator 5 window enable
[10]	CMP5_0CHN2P_WIN_EN	Use the P-tube switch control signal output from the CHN2_P channel of the MCPWM0 module as the comparator 5 window enable
[9]	CMP5_0CHN1P_WIN_EN	Use the P-tube switch control signal output from the CHN1_P channel of the MCPWM0 module as the comparator 5 window enable
[8]	CMP5_0CHN0P_WIN_EN	Use the P-tube switch control signal output from the CHN0_P channel of the MCPWM0 module as the comparator 5 window enable
[7]	CMP4_1CHN3P_WIN_EN	Use the P-tube switch control signal output from the CHN3_P channel of the MCPWM1 module as the comparator 4 window enable
[6]	CMP4_1CHN2P_WIN_EN	Use the P-tube switch control signal output from the CHN2_P channel of the MCPWM1 module as the comparator 4 window enable
[5]	CMP4_1CHN1P_WIN_EN	Use the P-tube switch control signal output from the CHN1_P channel of the MCPWM1 module as the comparator 4 window enable
[4]	CMP4_1CHN0P_WIN_EN	Use the P-tube switch control signal output from the CHN0_P channel of the MCPWM1 module as the comparator 4 window enable
[3]	CMP4_0CHN3P_WIN_EN	Use the P-tube switch control signal output from the CHN3_P channel of the MCPWM0 module as the comparator 4 window enable
[2]	CMP4_0CHN2P_WIN_EN	Use the P-tube switch control signal output from the CHN2_P channel of the MCPWM0 module as the comparator 4 window enable
[1]	CMP4_0CHN1P_WIN_EN	Use the P-tube switch control signal output from the CHN1_P

		channel of the MCPWM0 module as the comparator 4 window enable
[0]	CMP4_0CHN0P_WIN_EN	Use the P-tube switch control signal output from the CHN0_P channel of the MCPWM0 module as the comparator 4 window enable

Usually, there is 1bit in CMP_ BLCWIN[3:0] is 1, indicating that the corresponding CHNx_P is used to control the signal generation of the comparator. If CMP_ BLCWIN[3:0] is 4 'b0000, it indicates that the comparator comparison signal is generated independently of the MCPWM signal.

CMP0/1 can use the 4 P channels of MCPWM0 for window control, CMP2/3 can use the 4 P channels of MCPWM1 for window control, and CMP4/5 can use the 4 P channels of both MCPWM0 and MCPWM1 for window control.

20.2.8 Comparator Output Value Register (CMP_DATA)

Address: 0x4001_3018

Reset value: 0x0

Table 20-8 Comparator Output Value Register (CMP_DATA)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		CMP5_FLT_DATA	CMP4_FLT_DATA	CMP3_FLT_DATA	CMP2_FLT_DATA	CMP1_FLT_DATA	CMP0_FLT_DATA			CMP5_RAW_DATA	CMP4_RAW_DATA	CMP3_RAW_DATA	CMP2_RAW_DATA	CMP1_RAW_DATA	CMP0_RAW_DATA
		RO	RO	RO	RO	RO	RO			RO	RO	RO	RO	RO	RO
		0	0	0	0	0	0			0	0	0	0	0	0

Bit field	Bit Name	Description
[31:14]		Unused
[13]	CMP5_FLT_DATA	Comparator 5 filtered signal
[12]	CMP4_FLT_DATA	Comparator 4 filtered signal
[11]	CMP3_FLT_DATA	Comparator 3 filtered signal
[10]	CMP2_FLT_DATA	Comparator 2 filtered signal
[9]	CMP1_FLT_DATA	Comparator 1 filtered signal
[8]	CMP0_FLT_DATA	Comparator 0 filtered signal
[7:6]		Unused
[5]	CMP5_RAW_DATA	Comparator 5 original output signal, directly from analog comparator 5
[4]	CMP4_RAW_DATA	Comparator 4 original output signal, directly from analog comparator 4

[3]	CMP3_RAW_DATA	Comparator 3 original output signal, directly from analog comparator 3
[2]	CMP2_RAW_DATA	Comparator 2 original output signal, directly from analog comparator 2
[1]	CMP1_RAW_DATA	Comparator 1 original output signal, directly from analog comparator 1
[0]	CMP0_RAW_DATA	Comparator 0 original output signal, directly from analog comparator 0

21 CAN-FD

21.1 Introduction

The CAN-CTRL core is a serial communications controller that performs serial communication according to the CAN protocol. This CAN bus interface uses the basic CAN principle and meets all constraints of the CAN-specification 2.0B active. Furthermore, this CAN core can be configured to meet the specification of CAN with flexible data rate CAN FD.

The CAN protocol uses a multi-master bus configuration for the transfer of frames (communication objects) between nodes of the network and manages the error handling without any burden on the CPU. The CAN-CTRL bus controller enables the user to set up economic and reliable links between various components. The CAN-CTRL core appears to a microcontroller as a memory-mapped I/O device. A CPU accesses the CAN-CTRL core to control transmission or reception of frames through a two wire CAN bus system. The connection to a CAN bus is illustrated below.

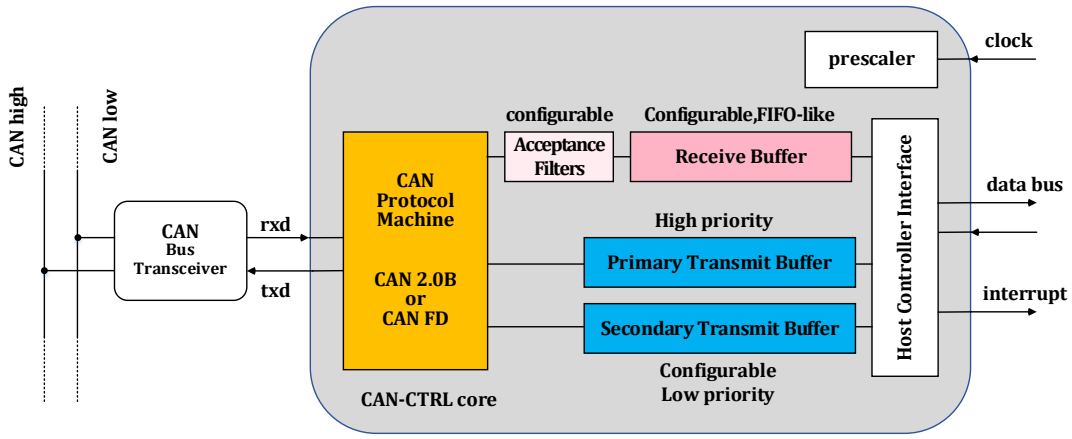


Figure 21-1 Connection to CAN Bus and Main Features of the CAN-CTRL Core

21.1.1 The CAN Protocol

CAN communication is organized in frames. Two types of frames exist: standard and extended frames. For CAN 2.0 the maximum data payload is up to 8 bytes while for CAN FD up to 64 bytes can be transmitted using one frame.

All CAN nodes are equal in terms of bus access. There is no super-node because the CAN is a multi- master bus.

Data addressing is done using message identifiers. In a CAN network only one node shall transmit messages with a certain identifier. All nodes receive all messages and the node host controller has to decide if it was addressed by the appropriate message identifier. To reduce the load of a host controller a CAN core may use acceptance filters. These filters compare all received message

identifiers to user- selectable bit patterns. Only if a message passes an acceptance filter, it will be stored in the receive buffer and signaled to the host controller.

The identifiers of CAN frames are also used for bus arbitration. The CAN protocol machine stops transmission of a message with a low-priority identifier when a message with a higher priority identifier is transmitted by another CAN node. The CAN protocol machine automatically attempts to re-transmit the stopped message at the next possible transmit position. A smaller identifier means a higher priority.

CAN 2.0B defines data bit rates up to 1Mbit/s. For CAN FD there is no fixed limitation. For CAN FD the standard defines a bit rate switching. If enabled the transmission of the payload of frames can be done at higher speed while the frame header is transmitted at lower speed.

21.1.2 Classic CAN 2.0B and How to Enable CAN FD

CAN FD is a downward-compatible extension of the classic CAN 2.0B protocol. Every CAN FD nodeis able to receive and transmit classic CAN frames.

The host controller firmware can select at run-time for every frame if the frame will be transmitted as classic CAN or CAN FD frame. This is done using the configuration bits in the transmit buffer. For received frames the status bits in the receive buffer signal if a received frame was a classic CAN or CAN FD frame.

21.1.3 Time-Stamping

21.1.3.1 Time-Triggered CAN

Optionally CAN-CTRL can be used for Time-Triggered CAN communication (TTCAN) according to ISO 11898-4. CAN-CTRL offers partial hardware support and requires host software interactions in real time in this mode.

The basic concept of TTCAN is to have a timer for time-stamping received messages and for triggering messages for transmission. One node in the CAN network is the time master. A time master transmits a reference message. With the reference message the cycle time starts. The time between two reference messages is a basic cycle. Inside the basic cycle messages can be transmitted in time windows. The TTCAN system administrator defines the start and duration of each time window during an offline setup.

There are three time-window types:

- Exclusive time window (only one node is allowed to transmit one frame with a defined ID)
- Free time window (unused time window for further extension of the network)
- Arbitrating time window (several nodes may transmit a frame and arbitration takes place)

If a frame is received, it gets the actual cycle time as time-stamp. For transmissions CAN-CTRL offers a hardware trigger that starts transmission of a predefined message at a predefined cycle time.

CAN-CTRL automatically detects a reference message upon reception and starts the cycle time. The hardware timer is a 16bit timer running at the CAN bit time as defined for ISO 11898-4 level 1. CAN- CTRL can be used as a time master.

Beside a hardware trigger for message transmissions CAN-CTRL offers a watch trigger to detect a missing reference message.

Partial hardware support means that the host controller needs to prepare the actions of the node for each time window. e.g., the host needs to define the message for the next transmission and define the trigger time for it. In other words: the host needs to take care of the transmission columns and the system matrix.

21.1.3.2 CiA 603 Time-Stamping

CAN in Automation (CiA) defines in specification 603 a method for time-stamping with at least 16 bits, which is supported by CAN-CTRL. It can be used in addition to TTCAN.

The basic concept of CiA 603 is to have a free-running timer which counts clock cycles and not CAN bit times. The precision shall be least 10µs (16 bit) or 1µs (32 bit or more). Time-stamps are acquired at the SOF or EOF of a CAN / CAN FD frame.

CiA 603 is supposed to support time-stamping and time-synchronization of AUTOSAR. For AUTOSAR one node in the CAN network is the time master. A time master transmits a synchronization message (SYNC message). The time-stamp of the SYNC message is acquired by the time master and all time slaves. The difference in time between the event of commanding a SYNC message until the time when the SYNC message actually gets transmitted will be transmitted in a follow-up (FUP) message by the time master.

CiA defines rules to read out the timer as well as to modify the timer. CAN-CTRL does not include the timer, but uses an external timer. CAN-CTRL only includes the mechanism of time-stamping, the register to store one transmission time stamp (TTS) and the memory to store reception time stamps (RTS) for all received messages.

21.2 Features

21.2.1 Feature List

- Supports CAN specification
  - CAN 2.0B (up to 8 bytes payload, verified by Bosch reference model)
  - CAN FD (up to 64 bytes payload, ISO 11898-1:2015 or non-ISO Bosch)
- Free programmable data rate:
  - CAN 2.0B defines data rates up to 1Mbit/s
  - CAN FD is limited by the transceiver
- Programmable baud rate prescaler (1 to 1/256)

- Receive buffer (RB)
  - FIFO-like behavior
  - Received messages which are “not accepted” or “incorrect” don’t overwrite already stored messages.
- Two Transmit Buffers
  - one Primary Transmit Buffer (PTB)
  - one Secondary Transmit Buffer (STB)
- Independent and programmable internal 29bit acceptance filters
  - 16 acceptance filters
- Extended features
  - Single Shot Transmission Mode (for PTB and / or for STB)
  - Listen Only Mode
  - Loop Back Mode (internal and external)
  - Transceiver Standby Mode
- Extended status and error report
  - Capturing of last occurred kind of error and of arbitration lost position
  - Programmable Error Warning Limit
- Configurable interrupt sources
- Time-stamping:
  - ISO 11898-4 Time-Triggered CAN with partial hardware support
  - CiA 603 time-stamping
- Compatibility to AUTOSAR
- Optimized for SAE J1939

21.2.2 Upward Compatibility

The CAN specification includes reserved bits for protocol extension. This has been used to build the CAN FD specification on top of the CAN 2.0B specification. Reserved bits are transmitted low (dominant) if not used. Unfortunately, the CAN 2.0B specification defines the behavior for the case that a reserved bit is high (recessive) to accept this and proceed with the frame. Therefore, if a CAN FD frame (which has a different and unexpected form compared to a CAN 2.0B frame) is received by a CAN 2.0B node that uses this behavior then this will result in an error frame by the CAN 2.0B node which destroys the frame. This behavior is called “CAN FD intolerant”.

To be upward compatible to new protocol specifications a so-called protocol exception event shall take place if a node detects a reserved bit high. This holds for CAN 2.0B as well as for CAN FD nodes. A protocol exception event results in no action for a receiver. The receiver just ignores this frame, does not generate an ACK, waits for bus idle and may transmit or receive the next frame. For a CAN 2.0B node this is called “CAN FD tolerant” and enables coexistence of CAN 2.0B and FD frames within one network.

Older CAN 2.0B conformance tests check for the “CAN FD intolerant” behavior but it is recommended to use the new “CAN FD tolerant” behavior or in general the protocol exception event

to be upward compatible regardless if the node is a CAN 2.0B or CAN FD node.

21.2.3 Message Buffers

21.2.3.1 Message Buffers Concept

The concept of the message buffers is illustrated in Figure below. This schematic focuses on the buffers and hides other details of the CAN-CTRL core. All buffer slots are big enough to store frames with the maximum length

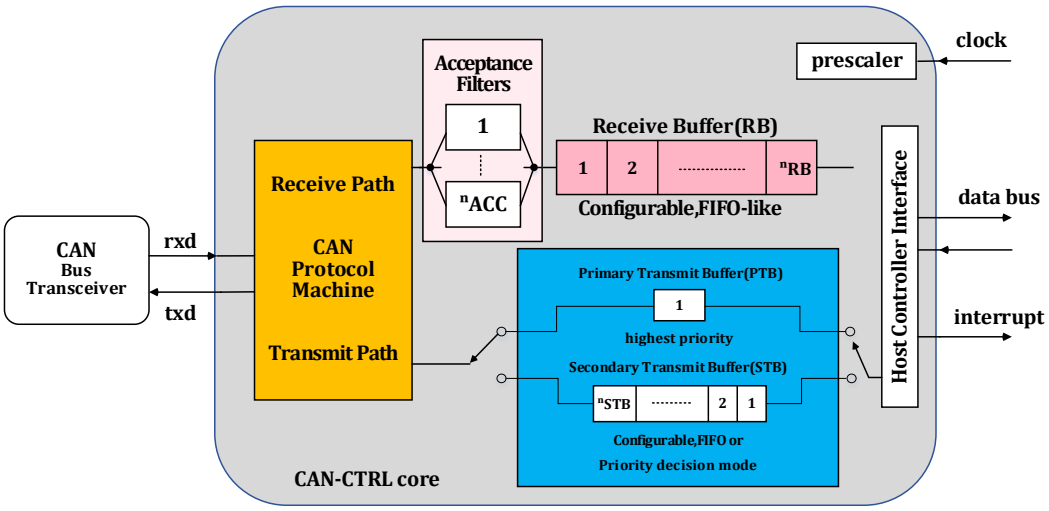


Figure 21-2 Message Buffers Concept

21.2.3.2 Receive Buffer

To reduce the load of received frames for the host controller, the core uses acceptance filters. The CAN- CTRL core checks the message identifier during acceptance filtering. If the received frame matches the filter criteria of one of the acceptance filters then it will be stored in the Receive Buffer (RB), which has FIFO-like behavior.

Depending on the number of available message slots, the host controller does not need to read incoming messages immediately. The CAN-CTRL core is able to generate interrupts upon every received message, when the RB is full or filled to a user-selectable “almost full” limit. Because of the FIFO-like behavior, the host controller always reads the oldest message from the RB.

21.2.3.3 Transmit Buffer

For frame transmission purposes, two Transmit Buffers (TB) are offered. The Primary TB (PTB) has a higher priority, but is able to buffer only one frame. The Secondary TB (STB) has a lower priority. It can act in FIFO or in priority mode. The priority decision between PTB and STB is fixed and fully independent from the CAN bus arbitration. Bus arbitration is a priority decision based on the frame identifiers.

The STB can be commanded to transmit one or all stored frames. In FIFO mode with every transmission the oldest frame inside this buffer is transmitted first. In priority mode the frame with the highest priority inside this buffer (based on the frame identifier) is transmitted first.

A frame located in the PTB has always a higher priority for the CAN protocol machine than the frames in the STB regardless of the frame identifiers. A PTB transmission stops and delays an STB transmission. The STB transmission is automatically restarted after the PTB frame has been successfully transmitted.

A PTB transmission starts at the next transmit position that is possible by the CAN protocol (after the next interframe space). Because of this, an STB transmission that has won the arbitration and is actually transmitted, will be completed before.

Interrupting STB transmissions using a PTB transmission may happen in the following cases:

1. The STB is commanded to output all stored frames and the host controller decides to command a PTB transmission before all STB transmissions are completed.
2. The STB is commanded to output a single frame and the host controller decides to command a PTB transmission before the STB transmission is completed.

If the host controller waits until each commanded transmission is completed, then it can easily decide which buffer shall transmit the next frame. As a drawback a message with a low-priority identifiermay block more important messages. Then the host could abort the message.

21.2.3.4 Transmit Buffer Application Example in FIFO Mode

In the example a CAN node is used for sensor measurements. The CAN system engineer has decided to handle these sensor measurements with low priority. Therefore, frame identifiers with low priority are used for CAN frames carrying the sensor data.

The host controller automatically acquires measurement results and places them as CAN frames in the STB. Because of high traffic at the CAN bus, it may happen that the sensor data frames cannot be transmitted immediately and several frames remain in the STB for some time. Later, if there is less traffic at the CAN bus, they will be transmitted.

In the situation where several frames remain in the STB, an event may happen that forces the host controller to output an important high-priority frame. In such a situation, the host can use the PTB. A frame in the PTB will be always transmitted before all frames in the STB.

The advantage of having two transmit buffers is the option to keep all messages. No message has to be aborted (discarded) in case of a high-priority event.

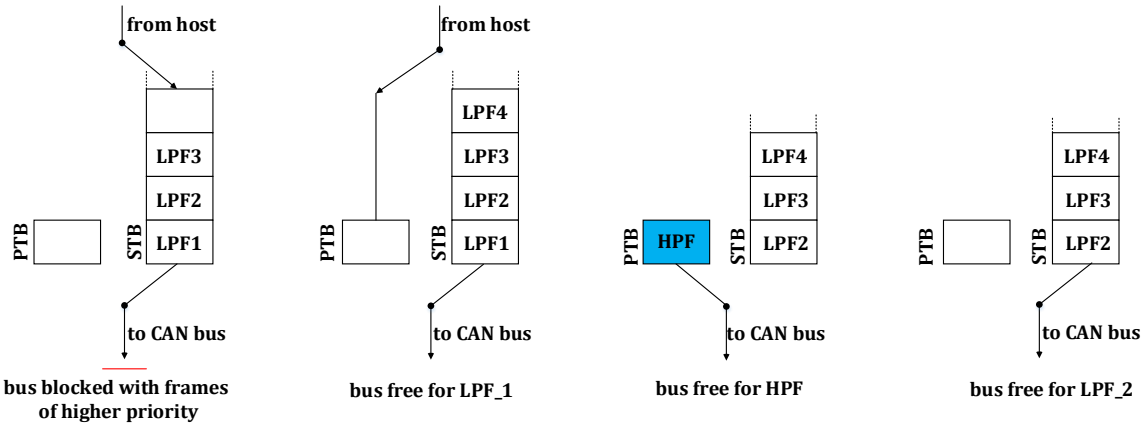


Figure 21-3 Transmit Buffer Application Example (TSALL=1)

The given application example is explained step by step. LPF_x is the abbreviation of "low priority frame" and HPF means "high priority frame". As can be seen, LPF_1 is blocked by other higher priority frames from other CAN nodes in the first part of the figure. The host controller puts a fourth frame into the STB (LPF_4) and then decides to output the HPF. Meanwhile, the CAN bus was free to transmit LPF_1 which is followed by HPF, LPF_2 and so on.

The priority decision between the PTB and STB is done by the CAN protocol machine, but the CAN protocol itself uses another independent priority decision mechanism which is called bus arbitration. For bus arbitration the frame identifier defines the priority level.

In the example given above, it would be possible to place a frame with a very low-priority identifier in the PTB while higher priority frames remain in the STB. For the decision between PTB and STB always the PTB wins regardless of the frame identifier. It is the task of the host controller to place frames with meaningful identifier priorities in the appropriate buffers.

21.2.3.5 Transmit Buffer Application Example in Priority Mode

In priority mode for the STB CAN-CTRL automatically changes the order of the frames inside the STB. As a result, the frame with the highest priority is transmitted first. **The priority decision is the same as for bus arbitration and therefore a frame identifier with a lower value has a higher priority.** Regardless of that the PTB has always the highest priority.

Reordering the frames inside the STB is done automatically as soon as new frame is prepared for transmission and written to the STB. As a result, the frame with the highest priority is selected for transmission. For the host controller it is like fire-and-forget: it just needs to write a new frame into the STB while priority mode is active.

In general, a frame with a higher priority will never interrupt an active transmission. To give an example in Figure 21-3, in the leftmost part the frame LPF_1 is currently under transmission while LPF_4 is written to the STB. Even if LPF_4 has higher priority than LPF_1, still LPF_1 will be transmitted until it loses arbitration on the bus. Then after the node has lost arbitration, LPF_4 will be selected for transmission because of its priority.

Priority reordering is automatically done in background, but takes some time. During heavy load when a lot of new frames with different priorities are written to the STB and the node loses arbitration or there are errors on the bus, it may be that not the frame with the highest priority is selected. This may happen if this frame has been written too shortly before the deadline for the decision which frame will be next. But this is a very rare condition and in general the host should not care about this.

If two frames with the same priority are written to the STB, then the oldest frame will be transmitted first regardless of any reordering with other lower- or higher-prioritized frames.

21.2.3.6 Aborting a Transmission

If the situation arises, where a message in a transmit buffer cannot be sent due to its low priority, this would block the buffer for a long time. In order to avoid this, the host controller can withdraw the transmission request and abort the message. Aborting is possible for PTB and STB as well as for single or all frame transmission.

21.2.3.7 The Transmit Buffer in TTCAN Mode

In TTCAN mode each slot in the STB can be addressed by the host. Each slot can be defined as filled or empty. Therefore, each slot can be dedicated to one specific message.

Transmission in TTCAN mode is started with triggers. Each trigger includes a pointer to a message slot. The PTB has no special priority in TTCAN mode and therefore is handled just like an STB slot.

In is convenient for a host application to dedicate one message to one buffer slot. Then a task in the host can update the message at any time and then if the trigger for this message is active, it can be transmitted. But having enough slots for all messages requires a lot of hardware. Because of the partial hardware support of ISO 11898-4 it is possible to use a buffer slot for more than one message in a basic cycle. Therefore ISO 11898-4 is possible even if the STB is disabled and the core includes only the single PTB message slot.

21.2.4 CAN 2.0 and CAN FD Frames

CAN FD is a protocol extension of CAN 2.0. The main differences are:

- Data payload: Up to 8 bytes for CAN 2.0 and up to 64 bytes for CAN FD
- One configurable bit rate for CAN 2.0, but 2 for CAN FD: slow for arbitration and fast for data phase

All types of frames for CAN 2.0 and CAN FD are shown in Figure 21-4. The abbreviations are explained in the CAN specification and in short in Table 21-1. For Classic CAN frames (CAN 2.0) some bit names are renamed with the CAN FD ISO specification, but here the older names are still used for easier backward-reference.

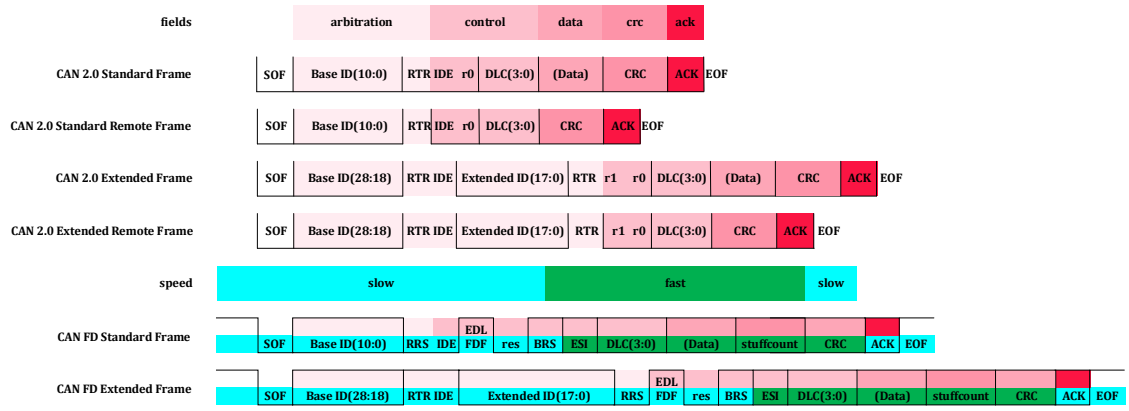


Figure 21-4 CAN 2.0 and CAN FD Frame Types

Table 21-1 CAN Bit Abbreviations

Abbreviation	Description	Comment
ID	IDentifier	
RTR	Remote Transmission Request	Remote or Data frame
SRR	Substitute Remote Request	
RRS	Remote Request Substitution	
IDE	IDentifier Extension	Standard or Extended frame
DLC	Data Length Code	Number of payload bytes
EDL	Extended Data Length	CAN 2.0 or CAN FD frame
FDF	FD Format indicator (=EDL)	CAN 2.0 or CAN FD frame
BRS	Bit Rate Switch	
ESI	Error State Indicator	
r1, r0, res	Reserved bits	

The CAN FD specification by Bosch (non-ISO) uses the name EDL while the CAN FD ISO specification uses the name FDF for the same bit. Both names are synonyms. This document uses the name FDF.

For CAN FD ISO frames the stuff count is transmitted as a part of the CRC field. For CAN FD non-ISO frames, the stuff count is not part of the frame. Furthermore, the CRC checkers have a different initialization for non-ISO and ISO frames. Therefore, ISO and non-ISO frames are incompatible.

The CAN protocol machine in the CAN-CTRL core automatically transmits and receives the frames and embeds the appropriate control and status bits. The host controller is required to select the desired frame type (IDE, RTR, FDF), chose the identifier and set the data payload.

21.2.5 AUTOSAR and SAE J1939

Both AUTOSAR and SAE J1939 are software stacks which can be used with CAN-CTRL.

For AUTOSAR the number of transmission buffers should be set to at least 3 slots (generic parameter STB_SLOTS) and transmissions can be aborted (bits TPA and TSA in register TCMD). Time-stamping and time-synchronization is supported using hardware time-stamping according to CiA 603.

SAE J1939 uses extended IDs (29 bits). For easy handling the acceptance filters can be configured to accept only extended IDs (bits AIDE and AIDEE in register ACF_3).

Table 21-2 Name Definitions

Abbreviation	Description
ACF	Acceptance Filter
PTB	Primary Transmit Buffer (high priority)
RB	Receive Buffer
RDC	Receiver Delay Compensation (related to TDC)
RTS	Reception Time Stamp
SP	Sample Point
SSP	Secondary Sample Point (CAN FD)
STB	Secondary Transmit Buffer (low priority)
TDC	Transmitter Delay Compensation (CAN FD specification)
TTCAN	Time-Triggered CAN (ISO 11898-4)
TTS	Transmission Time Stamp
TQ	Time Quanta (CAN specification)

All register definitions are given including an access definition. Access definitions are given using an abbreviation in the form <access>-<reset_value>. Possible abbreviations for the <access> attribute are “r” for read, “w” for write and “rw” for read/write access. The <reset_value > attribute can be “0”, “1” and “u” for uninitialized registers. Example: “rw-0” means “readable and writeable and reset to 0” while “r-u” means “readable and uninitialized”. Unused bits are always r-0.

21.3 Registers (Software Interface)

CAN-FD base address is 0x40018800.

Table 21-3 Register Map

Offset	Bit position								Register name
	7	6	5	4	3	2	1	0	
0x00 ~ 0x4f	Receive Buffer Registers and Reception Time Stamp								RBUF (and RTS)
0x50 ~ 0x97	Transmit Buffer Registers								TBUF
0x98 ~ 0x9f	Transmission Time Stamp								TTS

Offset	Bit position								Register name
0xa0	RESET	LBME	LBMI	TPSS	TSSS	RACTIVE	TACTIVE	BUSOFF	CFG_STAT
0xa1	TBSEL	LOM	STBY	TPE	TPA	TSONE	TSALL	TSA	TCMD
0xa2	FD_ISO *	TSNEXT	TSMODE	TTTBM	-		TSSTAT(1:0)		TCTRL
0xa3	SACK	ROM	ROV	RREL	RBALL	-	RSTAT(1:0)		RCTRL
0xa4	RIE	ROIE	RFIE	RAFIE	TPIE	TSIE	EIE	TSFF	RTIE
0xa5	RIF	ROIF	RFIF	RAFIF	TPIF	TSIF	EIF	AIF	RTIF
0xa6	EWARN	EPASS	EPIE	EPIF	ALIE	ALIF	BEIE	BEIF	ERRINT
0xa7	AFWL(3:0)				EWL(3:0)				LIMIT
0xa8	S_Seg_1(7:0)								S_Seg_1 *
0xa9	-	S_Seg_2(6:0)							S_Seg_2 *
0xaa	-	S_SJW(6:0)							S_SJW *
0xab	S_PRESC(7:0)								S_PRESC *
0xac	-			F_Seg_1(4:0)					F_Seg_1 *
0xad	-				F_Seg_2(3:0)				F_Seg_2 *
0xae	-				F_SJW(3:0)				F_SJW *
0xaf	F_PRESC(7:0)								F_PRESC *
0xb0	KOER(2:0)			ALC(4:0)					EALCAP
0xb1	TDCEN	SSPOFF(6:0)							TDC *
0xb2	RECNT								RECNT
0xb3	TECNT								TECNT
0xb4	-		SELMASK	-	ACFADR				ACFCTRL
0xb5	-						TIMEPOS	TIMEEN	TIMECFG
0xb6	AE_7	AE_6	AE_5	AE_4	AE_3	AE_2	AE_1	AE_0	ACF_EN_0
0xb7	AE_15	AE_14	AE_13	AE_12	AE_11	AE_10	AE_9	AE_8	ACF_EN_1
0xb8	ACODE_x or AMASK_x (7:0)								ACF_0 *
0xb9	ACODE_x or AMASK_x (15:8)								ACF_1 *
0xba	ACODE_x or AMASK_x (23:16)								ACF_2 *
0xbb	-	AIDEE	AIDE	ACODE_x or AMASK_x (28:24)					ACF_3 *
0xbc	VERSION(7:0)								VER_0
0xbd	VERSION(15:8)								VER_1
0xbe	TBE	TBF	TBPTR(5:0)						TBSLOT
0xbf	WTIE	WTIF	TEIF	TTIE	TTIF	T_PRESC(1:0)		TTEN	TTCFG
0xc0	REF_ID(7:0)								REF_MSG_0
0xc1	REF_ID(15:8)								REF_MSG_1
0xc2	REF_ID(23:16)								REF_MSG_2
0xc3	REF_IDE	-		REF_ID(28:24)					REF_MSG_3
0xc4	-		TTPTR(5:0)						TRIG_CFG_0
0xc5	TEW(3:0)				-	TTYPE(2:0)			TRIG_CFG_1
0xc6	TT_TRIG(7:0)								TT_TRIG_0
0xc7	TT_TRIG(15:8)								TT_TRIG_1
0xc8	TT_WTRIG(7:0)								TT_WTRIG_0
0xc9	TT_WTRIG(15:8)								TT_WTRIG_1

Offset	Bit position	Register name
0xca	-	-
0xcb	-	-

* registers can only be written if bit RESET in register CFG_STAT is set.

A write access to an addressable location not shown in the register map results in no action and a read access will result in the value 0x00.

Table 21-4 Transmission Time Stamp TTS (0x98 to 0x9f)

Bits	Name	Access	Function
63:0	TTS	r-0	64bit Transmission Time Stamp TTS holds the time stamp of the last transmitted frame for CiA 603 time stamping. Every new frame overwrites TTS if TTSEN=1. The TTS is intended to be used by the time master to acquire the time-stamp of the SYNC message.

Table 21-5 Configuration and Status Register CFG_STAT (0xa0)

Bits	Name	Access	Function
7	RESET	rw-1	RESET request bit 1 - The host controller performs a local reset of CAN-CTRL. 0 - no local reset of CAN-CTRL Some register (e.g for node configuration) can only be modified if RESET=1. Bit RESET forces several components to a reset state. A detailed definition is given in chap. 21.4.11 RESET is automatically set if the node enters “bus off” state. Note that a CAN node will participate in CAN communication after RESET is switched to 0 after 11 CAN bit times. This delay is required by the CAN standard (bus idle time). If RESET is set to 1 and immediately set to 0, then it takes some time until RESET can be read as 0 and becomes inactive. The reason is clock domain crossing from host to CAN clock domain. RESET is held active as long as needed depending on the relation between host and CAN clock.
6	LBME	rw-0	Loop Back Mode, External (chap. 21.4.10.4) 0 - Disabled 1 - Enabled LBME should not be enabled while a transmission is active.
5	LBMI	rw-0	Loop Back Mode, Internal (chap. 21.4.10.4) 0 - Disabled 1 - Enabled LBMI should not be enabled while a transmission is active.
4	TPSS	rw-0	Transmission Primary Single Shot mode for PTB (chap. 21.4.10.1) 0 - Disabled 1 - Enabled
3	TSSS	rw-0	Transmission Secondary Single Shot mode for STB (chap. 21.4.10.1) 0 - Disabled 1 - Enabled
2	RACTIVE	r-0	Reception ACTIVE (Receive Status bit) 1 - The controller is currently receiving a frame. 0 - No receive activity.
1	TACTIVE	r-0	Transmission ACTIVE (Transmit Status bit) 1 - The controller is currently transmitting a frame. 0 - No transmit activity.

0	BUSOFF	rw-0	Bus Off (Bus Status bit, chap. 21.4.7) 1 - The controller status is “bus off”. 0 - The controller status is “bus on”. Writing a 1 to BUSOFF will reset TECNT and RECNT. This should be done only for debugging.

Table 21-6 Command Register TCMD (0xa1)

Bits	Name	Access	Function
7	TBSEL	rw-0	Transmit Buffer Select Selects the transmit buffer to be loaded with a message. Use the TBUF registers for access. TBSEL needs to be stable all the time the TBUF registers are written and when TSNEXT is set. 0 - PTB (high-priority buffer) 1 - STB The bit will be reset to the hardware reset value if (TTEN=1 and TTTBM=1).
6	LOM	rw-0	Listen Only Mode (chap. 21.4.10.2) 0 - Disabled 1 - Enabled LOM cannot be set if TPE, TSONE or TSALL is set. No transmission can be started if LOM is enabled and LBME is disabled. LOM=1 and LBME=0 disables all transmissions. LOM=1 and LBME=1 disables the ACK for received frames and error frames, but enables the transmission of own frames.
5	STBY	rw-0	Transceiver Standby Mode 0 - Disabled 1 - Enabled This register bit is connected to the output signal stby which can be used to control a standby mode of a transceiver. **In this implementation, this bit should always be 0.** STBY cannot be set to 1 if TPE=1, TSONE=1 or TSALL=1. If the host sets STBY to 0 then the host needs to wait for the time required by the transceiver to start up before the host requests a new transmission.
4	TPE	rw-0	Transmit Primary Enable 1 - Transmission enable for the message in the high-priority PTB 0 - No transmission for the PTB If TPE is set, the message from the PTB will be transmitted at the next possible transmit position. A started transmission from the STB will be completed before, but pending new messages are delayed until the PTB message has been transmitted. TPE stays set until the message has been transmitted successfully or it is aborted using TPA. The host controller can set TPE to 1 but cannot reset it to 0. This would only be possible using TPA and aborting the message. The bit will be reset to the hardware reset value if RESET=1, STBY=1, (LOM=1 and LBME=0) or (TTEN=1 and TTTBM=1).
			Transmit Primary Abort

Bits	Name	Access	Function
3	TPA	rw-0	1 – Aborts a transmission from PTB which has been requested by TPE=1 but not started yet. (The data bytes of the message remain in the PTB.) 0 – no abort The bit has to be set by the host controller and will be reset by CAN-CTRL. Setting TPA automatically de-asserts TPE. The host controller can set TPA to 1 but cannot reset it to 0. During the short time while the CAN-CTRL core resets the bit, it cannot be set by the host. The bit will be reset to the hardware reset value if RESET=1 or (TTEN=1 and TTTBM=1). TPA should not be set simultaneously with TPE.
2	TSONE	rw-0	Transmit Secondary ONE frame 1 – Transmission enables one message in the STB. In FIFO mode this is the oldest message and in priority mode this is the one with the highest priority. TSONE in priority mode is difficult to handle, because it is not always clear which message will be transmitted if new messages are written to the STB meanwhile. The controller starts the transmission as soon as the bus becomes vacant and no request of the PTB (bit TPE) is pending. 0 – No transmission for the STB. TSONE stays set until the message has been transmitted successfully or it is aborted using TSA. The host controller can set TSONE to 1 but cannot reset it to 0. This would only be possible using TSA and aborting the message. The bit will be reset to the hardware reset value if RESET=1, STBY=1, (LOM=1 and LBME=0) or (TTEN=1 and TTTBM=1).
1	TSALL	rw-0	Transmit Secondary ALL frames 1 – Transmission enables all messages in the STB. The controller starts the transmission as soon as the bus becomes vacant and no request of the PTB (bit TPE) is pending. 0 – No transmission for the STB. TSALL stays set until all messages have been transmitted successfully or they are aborted using TSA. The host controller can set TSALL to 1 but cannot reset it to 0. This would only be possible using TSA and aborting the messages. The bit will be reset to the hardware reset value if RESET=1, STBY=1, (LOM=1 and LBME=0) or (TTEN=1 and TTTBM=1). If during a transmission the STB is loaded with a new frame then the new frame will be transmitted too. In other words: a transmission initiated by TSALL is finished when the STB becomes empty.
			Transmit Secondary Abort 1 – Aborts a transmission from STB which has been requested but not started yet. For a TSONE transmission, only one frame is aborted while for a TSALL Transmission,

Bits	Name	Access	Function
0	TSA	rw-0	all frames are aborted. One or all message slots will be released which updates TSSTAT. All aborted messages are lost because they are not accessible any more. If in priority mode a TSONE transmission is aborted, then it is not clear which frame will be aborted if new frames are written to the STB meanwhile. 0 – no abort The bit has to be set by the host controller and will be reset by CAN-CTRL. Setting TSA, automatically de-asserts TSONE or TSALL respectively. The host controller can set TSA to 1 but cannot reset it to 0. The bit will be reset to the hardware reset value if RESET=1. TSA should not be set simultaneously with TSONE or TSALL.

Setting both TSONE and TSALL is meaningless. While TSALL is already set, it is impossible to set TSONE and vice versa. If both TSONE and TSALL are set simultaneously then TSALL wins and TSONE is cleared by the CAN-CTRL core.

Table 21-7 Transmit Control Register TCTRL (0xa2)

Bits	Name	Access	Function
7	FD_ISO	rw-1	CAN FD ISO mode 0 – Bosch CAN FD (non-ISO) mode 1 – ISO CAN FD mode (ISO 11898-1:2015) ISO CAN FD mode has a different CRC initialization value and an additional stuff bit count. Both modes are incompatible and must not be mixed in one CAN network. This bit has no impact to CAN 2.0B. This bit is only writeable if RESET=1.
6	TSNEXT	rw-0	Transmit buffer Secondary NEXT 0 - no action 1 - STB slot filled, select next slot. After all frame bytes are written to the TBUF registers, the host controller has to set TSNEXT to signal that this slot has been filled. Then the CAN-CTRL core connects the TBUF registers to the next slot. Once a slot is marked as filled a transmission can be started using TSONE or TSALL. It is possible to set TSNEXT and TSONE or TSALL together in one write access. TSNEXT has to be set by the host controller and is automatically reset by the CAN-CTRL core immediately after it was set. Setting TSNEXT is meaningless if TBSEL=0. In this case TSNEXT is ignored and automatically cleared. It does not do any harm. If all slots of the STB are filled, TSNEXT stays set until a slot becomes free (chap.21.4.6). TSNEXT has no meaning in TTCAN mode and is fixed to 0.

5	TSMODE	rw-0	Transmit buffer Secondary operation MODE 0 - FIFO mode 1 - priority decision mode In FIFO mode frames are transmitted in the order in that they are written into the STB. In priority decision mode the frame with the highest priority in the STB is automatically transmitted first. The ID of a frame is used for the priority decision. A lower ID means a higher priority of a frame. A frame in the PTB has always the highest priority regardless of the ID. TSMODE shall be switched only if the STB if empty.
4	TTTBM	rw-1	TTCAN Transmit Buffer Mode If TTEN=0 then TTTBM is ignored, otherwise the following is valid: 0 - separate PTB and STB, behavior defined by TSMODE 1 - full TTCAN support: buffer slots selectable by TBPTR and TTPTR For event-driven CAN communication (TTEN=0), the system provides PTB and STB and the behavior of the STB is defined by TSMODE. Then TTTBM is ignored. For time-triggered CAN communication (TTEN=1) with full support of all features including time-triggered transmissions, TTTBM=1 needs to be chosen. Then the all TB slots are addressable using TTPTR and TBPTR. For time-triggered CAN communication (TTEN=1) with only support of reception time-stamps, TTTBM=0 can be chosen. Then the transmit buffer acts as in event-driven mode and the behavior can be selected by TSMODE. TTTBM shall be switched only if the TBUF is empty.
3:2	-	r-0	reserved
1:0	TSSTAT	r-0	Transmission Secondary STATus bits If TTEN=0 or TTTBM=0: 00 – STB is empty 01 – STB is less than or equal to half full 10 – STB is more than half full 11 – STB is full If the STB is disabled using STB_DISABLE, then TSSTAT=00. If TTEN=1 and TTTBM=1: 00 – PTB and STB are empty 01 – PTB and STB are not empty and not full 11 – PTB and STB are full

Table 21-8 Receive Control Register RCTRL (0xa3)

Bits	Name	Access	Function
7	SACK	rw-0	Self-ACKnowledge 0 – no self-ACK 1 – self-ACK when LBME=1
6	ROM	rw-0	Receive buffer Overflow Mode In case of a full RBUF when a new message is received, then ROM selects the following: 1 – The new message will not be stored. 0 – The oldest message will be overwritten.

5	ROV	r-0	Receive buffer OVerflow 1 – Overflow. At least one message is lost. 0 – No Overflow. ROV is cleared by setting RREL=1.
4	RREL	rw-0	Receive buffer RELease The host controller has read the actual RB slot and releases it. Afterwards the CAN-CTRL core points to the next RB slot. RSTAT gets updated. 1 – Release: The host has read the RB. 0 – No release
3	RBALL	rw-0	Receive Buffer stores ALL data frames 0 – normal operation 1 – RB stores correct data frames as well as data frames with error (chap. 21.4.9.4)
2	-	r-0	Reserved
1:0	RSTAT	r-0	Receive buffer STATus 00 - empty 01 - ＞ empty and ＜ almost full (AFWL) 10 - ≥ almost full (programmable threshold by AFWL) but not full and no overflow 11 - full (stays set in case of overflow – for overflow signaling see ROV)

Table 21-9 Receive and Transmit Interrupt Enable Register RTIE (0xa4)

Bits	Name	Access	Function
7	RIE	rw-1	Receive Interrupt Enable 0 – Disabled, 1 – Enabled
6	ROIE	rw-1	RB Overrun Interrupt Enable 0 – Disabled, 1 – Enabled
5	RFIE	rw-1	RB Full Interrupt Enable 0 – Disabled, 1 – Enabled
4	RAFIE	rw-1	RB Almost Full Interrupt Enable 0 – Disabled, 1 – Enabled
3	TPIE	rw-1	Transmission Primary Interrupt Enable 0 – Disabled, 1 – Enabled
2	TSIE	rw-1	Transmission Secondary Interrupt Enable 0 – Disabled, 1 – Enabled
1	EIE	rw-1	Error Interrupt Enable 0 – Disabled, 1 – Enabled

0	TSFF	r-0	If TTEN=0 or TTTBM=0: Transmit Secondary buffer Full Flag 1 - The STB is filled with the maximal number of messages. 0 - The STB is not filled with the maximal number of messages If the STB is disabled using STB_DISABLE, then TSFF=0. If TTEN=1 and TTTBM=1: Transmit buffer Slot Full Flag 1 - The buffer slot selected by TBPTR is filled. 0 - The buffer slot selected by TBPTR is empty.

Table 21-10 Receive and Transmit Interrupt Flag Register RTIF (0xa5)

Bits	Name	Access	Function
7	RIF	rw-0	Receive Interrupt Flag 1 - Data or a remote frame has been received and is available in the receivebuffer. 0 - No frame has been received.
6	ROIF	rw-0	RB Overrun Interrupt Flag 1 - At least one received message has been overwritten in the RB. 0 - No RB overwritten. In case of an overrun both ROIF and RFIF will be set.
5	RFIF	rw-0	RB Full Interrupt Flag 1 - All RBs are full. If no RB will be released until the next valid message isreceived, the oldest message will be lost. 0 - The RB FIFO is not full.
4	RAFIF	rw-0	RB Almost Full Interrupt Flag 1 - number of filled RB slots ≥ AFWL_i 0 - number of filled RB slots < AFWL_i
3	TPIF	rw-0	Transmission Primary Interrupt Flag 1 - The requested transmission of the PTB has been successfully completed. 0 - No transmission of the PTB has been completed. In TTCAN mode, TPIF will never be set. Then only TSIF is valid.
2	TSIF	rw-0	Transmission Secondary Interrupt Flag 1 - The requested transmission of the STB has been successfully completed. 0 - No transmission of the STB has been completed successfully. In TTCAN mode TSIF will signal all successful transmissions, regardless of storage location of the message.
1	EIF	rw-0	Error Interrupt Flag 1 - The border of the error warning limit has been crossed in either direction, or the BUSOFF bit has been changed in either direction. 0 - There has been no change.
0	AIF	rw-0	Abort Interrupt Flag 1 - After setting TPA or TSA the appropriated message(s) have been aborted. It is recommended to not set both TPA and TSA simultaneously because both source AIF. 0 - No abort has been executed. The AIF does not have an associated enable register. See also chap. 3.9.5 for further information.

To reset an interrupt flag, the host controller needs to write an 1 to the flag. Writing a 0 has no effect. If a new interrupt event occurs while the write access is active then this event will set the flag and override the reset. This ensures that no interrupt event is lost.

Interrupt flags will only be set if the associated interrupt enable bit is set.

Table 21-11 **ERRor INTerrupt Enable and Flag Register ERRINT (0xa6)**

Bits	Name	Access	Function
7	EWARN	r-0	Error WARNing limit reached 1 - One of the error counters RECNT or TECNT is equal or bigger than EWL 0 - The values in both counters are less than EWL.
6	EPASS	r-0	Error Passive mode active 0 - not active (node is error active) 1 - active (node is error passive)
5	EPIE	rw-0	Error Passive Interrupt Enable
4	EPIF	rw-0	Error Passive Interrupt Flag. EPIF will be activated if the error status changes from error active to error passive or vice versa and if this interrupt is enabled.
3	ALIE	rw-0	Arbitration Lost Interrupt Enable
2	ALIF	rw-0	Arbitration Lost Interrupt Flag
1	BEIE	rw-0	Bus Error Interrupt Enable
0	BEIF	rw-0	Bus Error Interrupt Flag

To reset an interrupt flag, the host controller needs to write a 1 to the flag. Writing a 0 has no effect. If a new interrupt event occurs while the write access is active then this event will set the flag and override the reset. This ensures that no interrupt event is lost.

Interrupt flags will only be set if the associated interrupt enable bit is set.

Table 21-12 Warning Limits Register LIMIT (0xa7)

Bits	Name	Access	Function

7:4	AFWL(3:0)	rw-0x1	receive buffer Almost Full Warning Limit AFWL defines the internal warning limit AFWL_i with n_{RB} being the number of available RB slots. $AFWL\_i = \begin{cases} AFWL & n_{RB} < 16 \\ 2 \cdot AFWL & 16 \le n_{RB} < 32 \\ 4 \cdot AFWL & 32 \le n_{RB} < 64 \\ \vdots & \vdots \end{cases}$ AFWL_i is compared to the number of filled RB slots and triggers RAFIF if equal. The valid range of AFWL_i=[1…n_{RB}]. AFWL = 0 is meaningless and automatically treated as 0x1. (Note that AFWL is meant in this rule and not AFWL_i.) AFWL_i ＞n_{RB} is meaningless and automatically treated as n_{RB} . AFWL_i = n_{RB} is a valid value, but note that RFIF also exists.
3:0	EWL(3:0)	rw-0xB	Programmable Error Warning Limit = (EWL+1)×8. Possible Limit values: 8, 16, … 128. The value of EWL controls EIF. EWL needs to be transferred using CDC from host to CAN clock domain. During transfer EWL register bits are write-locked for the host for a few clocks until CDC is complete.

Table 21-13 Bit Timing Register S_Seg_1 (0xa8)

Bits	Name	Access	Function
7:0	S_Seg_1(7:0)	rw-0x3	Bit Timing Segment 1 (slow speed) The sample point will be set to t_{Seg_1} =(Seq_1+2)・*TQ* after start of bit time.

Table 21-14 Bit Timing Register S_Seg_2 (0xa9)

Bits	Name	Access	Function
7	-	r-0	reserved
6:0	S_Seg_2(6:0)	rw-0x2	Bit Timing Segment 2 (slow speed) Time t_{Seg_2}=(Seq_2+1)・TQ after the sample point.

Table 21-15 Bit Timing Register S_SJW (0xaa)

Bits	Name	Access	Function
7	-	r-0	reserved
6:0	S_SJW(6:0)	rw-0x2	Synchronization Jump Width (slow speed) The Synchronization Jump Width t_{SJW}=(SJW_1+1)・TQ is the maximum time for shortening or lengthening the Bit Time for resynchronization, where TQ is a time quanta.

Table 21-16 Bit Timing Register F_Seg_1 (0xac)

Bits	Name	Access	Function
7:5	-	r-0	reserved
4:0	F_Seg_1(4:0)	rw-0x3	Bit Timing Segment 1 (fast speed) The sample point will be set to t_{Seg_1}=(Seq_1+2)・TQ after start of bit time.

Table 21-17 Bit Timing Register F_Seg_2 (0xad)

Bits	Name	Access	Function
7:4	-	r-0	reserved
3:0	F_Seg_2(3:0)	rw-0x2	Bit Timing Segment 2 (fast speed) Time t_{Seg_2}=(Seq_2+1)・TQ after the sample point.

Table 21-18 Bit Timing Register F_SJW (0xae)

Bits	Name	Access	Function
7:4	-	r-0	reserved
3:0	F_SJW(3:0)	rw-02	Synchronization Jump Width (fast speed) The Synchronization Jump Width t_{SJW}=(SJW_1+1)・TQ is the maximum time for shortening or lengthening the Bit Time for resynchronization, where TQ is a time quanta.

Table 21-19 Prescaler Registers S_PRESC (0xab) and F_PRESC (0xaf)

Bits	Name	Access	Function
7:0	S_PRESC F_PRESC	rw-0x01	Prescaler (slow and fast speed) The prescaler divides the system clock to get the time quanta clock tq_clk. Valid range PRESC=[0x00, 0xff] results in divider values 1 to 256.

Table 21-20 Transmitter Delay Compensation Register TDC (0xb1)

Bits	Name	Access	Function
7	TDCEN	rw-0	Transmitter Delay Compensation ENable TDC will be activated during the data phase of a CAN FD frame if BRS is active if TDCEN=1. For more details about TDC see chap. 21.7.4.
6:0	SSPOFF	rw-0x00	Secondary Sample Point OFFset The transmitter delay plus SSPOFF defines the time of the secondary sample point for TDC. SSPOFF is given as a number of TQ.

Writing to S_Seg_1, S_Seg_2, S_SJW, S_PRESC, F_Seg_1, F_Seg_2, F_SJW, F_PRESC and TDC is only possible if RESET=1. A detailed description of the CAN bus bit timing is given in chap. 3.10. The reset value sets the bit timing which is described in the example in chap. 21.7.3.

All timing parameters in are given for slow (prefix “S_”) and fast speed (prefix “F_”). Slow speed is used for CAN 2.0 and the CAN FD arbitration phase. Fast speed is used for the CAN FD data phase.

Table 21-21 Error and Arbitration Lost Capture Register EALCAP (0xb0)

Bits	Name	Reset Value	Function
7:5	KOER(2:0)	r-0x0	Kind Of ERror (Error code) 000 - NO ERROR 001 - BIT ERROR 010 - FORM ERROR 011 - STUFF ERROR 100 - ACKNOWLEDGEMENT ERROR 101 - CRC ERROR 110 - OTHER ERROR (Dominant bits after own error flag, received active Error Flag too long, dominant bit during Passive-Error-Flag after ACK error) 111 - not used KOER is updated with each new error. Therefore, it stays untouched when frames are successfully transmitted or received.
4:0	ALC(4:0)	r-0x0	Arbitration Lost Capture (bit position in the frame where the arbitration has been lost)

Table 21-22 Error Counter Registers RECNT (0xb2) and TECNT (0xb3)

Bits	Name	Access	Function
7:0	RECNT	r-0x00	Receive Error CouNT (number of errors during reception) RECNT is incremented and decremented as defined in the CAN specification. RECNT does not overflow. See chap. 21.4.7 for more details about RECNT and the “bus off” state.

7:0	TECNT	r-0x00	Transmit Error CouNT (number of errors during transmission) TECNT is incremented and decremented as defined in the CAN specification. In case of the “bus off state” TECNT may overflow. See chap. 21.4.7 for more details about TECNT and the “bus off” state.

The acceptance filter registers ACF_x provide access to the acceptance filter codes ACODE_x and acceptance filter masks AMASK_x depending on the setting of SELMASK. (See Figure 21-5 and also Table 21-23). Write access to ACF_x is only possible if RESET=1.

The acceptance filters are build using a true 32bit wide memory and therefore a write access needs to be performed as 32bit write.

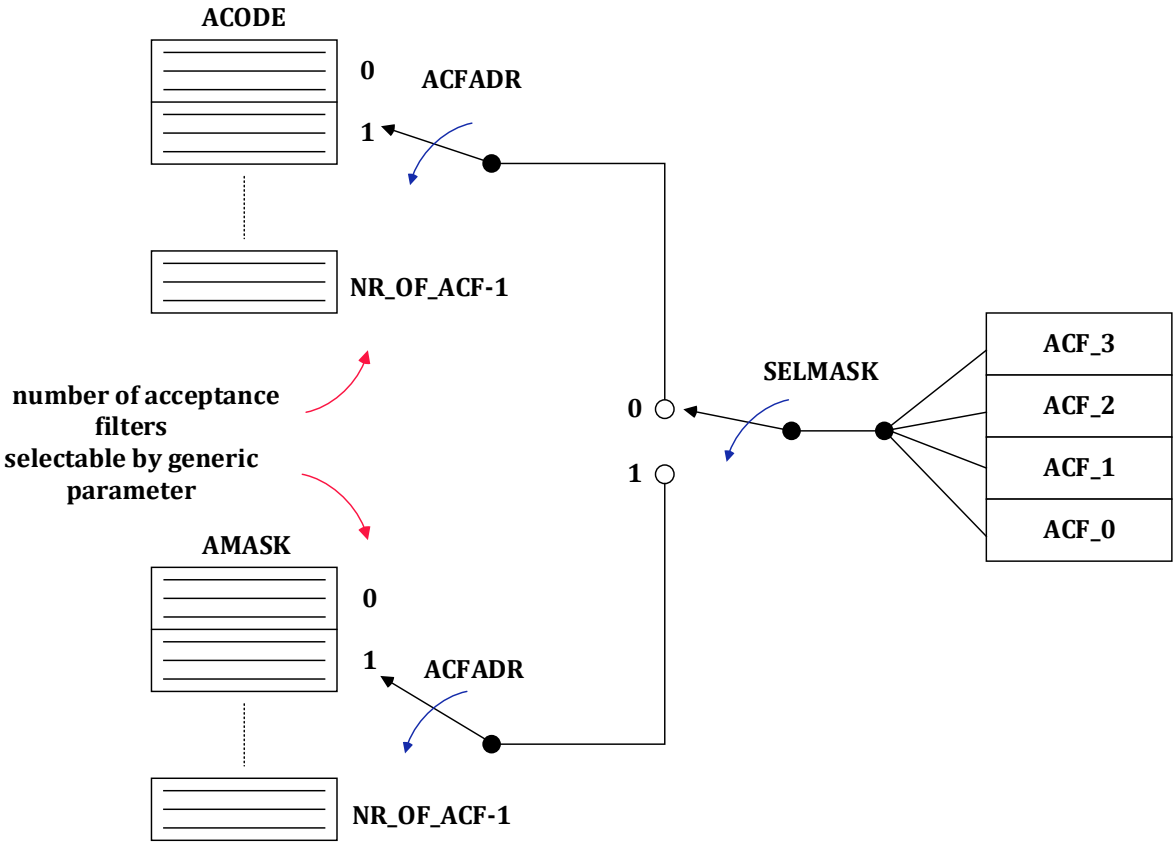


Figure 21-5 Access to the Acceptance Filters

Table 21-23 Acceptance CODE ACODE_x (register ACF_x (0xb8 to 0xbb))

Bits	Name	Access	Function
7:0	ACODE_0 ACODE_x	rw-0x00 rw-u	Acceptance CODE 1 - ACC bit value to compare with ID bit of the received message 0 - ACC bit value to compare with ID bit of the received message ACODE_x(10:0) will be used for standard frames. ACODE_x(28:0) will be used for extended frames. Only filter 0 is affected by the power-on reset. All other filters stay uninitialized.

Table 21-24 Acceptance MASK AMASK_x (register ACF_x (0xb8 to 0xbb))

Bits	Name	Access	Function
7:0	AMASK_0 AMASK_x	rw-0xFF rw-u	Acceptance MASK 1 - acceptance check for these bits of receive identifier disabled 0 - acceptance check for these bits of receive identifier enable AMASK_x(10:0) will be used for standard frames. AMASK_x(28:0) will be used for extended frames. Disabled bits result in accepting the message. Therefore, the default configuration after reset for filter 0 accepts all messages. Only filter 0 is affected by the power-on reset. All other filters stay uninitialized.

The AMASK_x includes additional bits in register ACF_3 (Table 21-25) which can be only accessed if SELMASK=1. These bits can be used to accept only either standard or extended frames with the selected ACODE / AMASK setting or to accept both frame types. Only acceptance filter 0 is affected by the power-on reset and it is configured to accept both frame types after power-up.

Table 21-25 Bits in Register ACF_3, if SELMASK=1

Bits	Name	Access	Function
6	AIDEE	rw-0 rw-u	Acceptance mask IDE bit check enable 1 - acceptance filter accepts either standard or extended as defined by AIDE 0 - acceptance filter accepts both standard or extended frames Only filter 0 is affected by the power-on reset. All other filters stay uninitialized.
5	AIDE	rw-0 rw-u	Acceptance mask IDE bit value If AIDEE=1 then: 1 - acceptance filter accepts only extended frames 0 - acceptance filter accepts only standard frames Only filter 0 is affected by the power-on reset. All other filters stay uninitialized.

Table 21-26 **Acceptance Filter Enable ACF_EN_0 (0xb6)**

Bits	Name	Access	Function
7:0	AE_x	rw-0x01	Acceptance filter Enable 1 - acceptance filter enabled 0 - acceptance filter disable Each acceptance filter (AMASK / ACODE) can be individually enabled or disabled. Only filter number 0 is enabled by default after hardware reset. Disabled filters reject a message. Only enabled filters can accept a message if the appropriate AMASK / ACODE configuration matches. To accept all messages one filter x has to be enabled by setting AE_x=1, AMASK_x=0xff and ACODE_x=0x00. This is the default configuration after hardware reset for filter x=0 while all other filters are disabled.

Table 21-27 Acceptance Filter Enable ACF_EN_1 (0xb7)

Bits	Name	Access	Function
7:0	AE_x	rw-0x00	Acceptance filter Enable 1 - acceptance filter enabled 0 - acceptance filter disable Each acceptance filter (AMASK / ACODE) can be individually enabled or disabled. Disabled filters reject a message. Only enabled filters can accept a message if the appropriate AMASK / ACODE configuration matches.

Table 21-28 Version Information VER_0 (0xbc) and VER_1 (0xbd)

Bits	Name	Access	Function
15:0	VER_0 VER_1	r	Version of CAN-CTRL, given as decimal value. VER_1 holds the major version and VER_0 the minor version. Example: version 5x16N00S00 is represented by VER_1=5 and VER_0=16.

Table 21-29 CiA 603 Time-Stamping TIMECFG (0xb5)

Bits	Name	Access	Function
7:2	-	r-0	Reserved
1	TIMEPOS	rw-1	TIME-stamping Position 0 – SOF 1 – EOF (see chap. 21.6) TIMEPOS can only be changed if TIMEEN=0, but it is possible to modify TIMPOS with the same write access that sets TIMEEN=1.
0	TIMEEN	rw-0	TIME-stamping ENable 0 – disabled 1 – enabled

Table 21-30 TTCAN: TB Slot Pointer TBSLOT (0xbe)

Bits	Name	Access	Function
7	TBE	rw-0	set TB slot to “Empty” 1 - slot selected by TBPTR shall be marked as “empty” 0 - no action TBE is automatically reset to 0 as soon as the slot is marked as empty and TSFF=0. If a transmission from this slot is active, then TBE stays set as long as either the transmission completes or after a transmission error or arbitration loss the transmission is not active any more. If both TBF and TBE are set, then TBE wins.
6	TBF	rw-0	set TB slot to “Filled” 1 - slot selected by TBPTR shall be marked as “filled” 0 - no action TBF is automatically reset to 0 as soon as the slot is marked as filled and TSFF=1. If both TBF and TBE are set, then TBE wins.
5:0	TBPTR	rw-0x00	Pointer to a TB message slot. 0x00 - Pointer to the PTB others - Pointer to a slot in the STB The message slot pointed to by TBPTR is readable / writable using the TBUF registers. Write access is only possible if TSFF=0. Setting TBF to 1 marks the selected slot as filled and setting TBE to 1 marks the selected slot as empty. TBSEL and TSNEXT are unused in TTCAN mode and have no meaning. TBPTR can only point to buffer slots, that exist in the hardware. Unusable bits of TBPTR are fixed to 0. TBPTR is limited to the PTB and 63 STB slots. More slots cannot be used in TTCAN mode. If TBPTR is too big and points to a slot that is not available, then TBF and TBE are reset automatically and no action takes place.

Table 21-31 TTCAN: Time Trigger Configuration TTCFG (0xbf)

Bits	Name	Access	Function
0	TTEN	rw-0	Time Trigger Enable 1 - TTCAN enabled, timer is running 0 - disabled
2:1	T_PRESC	rw-0x0	TTCAN Timer PRESCaler 00 - 1 01 - 2 10 - 4 11 - 8 The TTCAN time base is a CAN bit time defined by S_PRES, S_SEG_1 and S_SEG_2. With T_PRESC an additional prescaling factor of 1, 2, 4 or 8 is defined. T_PRESC can only be modified if TTEN=0, but it is possible to modify T_PRESC and set TTEN simultaneously with one write access.
			Time Trigger Interrupt Flag

3	TTIF	rw-0	TTIF will be set if TTIE is set and the cycle time is equal to the trigger time TT_TRIG. Writing a one to TTIF resets it. Writing a zero has no impact. TTIF will be set only once. If TT_TRIG gets not updated, then TTIF will be not set again in the next basic cycle.
4	TTIE	rw-1	Time Trigger Interrupt Enable If TTIE is set, then TTIF will be set if the cycle time is equal to the trigger time TT_TRIG.
5	TEIF	rw-0	Trigger Error Interrupt Flag The conditions when TEIF will be set, are defined in chap. 21.5.4. There is no bit to enable or disable the handling of TEIF.
6	WTIF	rw-0	Watch Trigger Interrupt Flag WTIF will be set if the cycle count reaches the limited defined by TT_WTRIG and WTIE is set.
7	WTIE	rw-1	Watch Trigger Interrupt Enable

To reset an interrupt flag, the host controller needs to write a 1 to the flag. Writing a 0 has no effect. If a new interrupt event occurs while the write access is active then this event will set the flag and override the reset. This ensures that no interrupt event is lost.

Interrupt flags will only be set if the associated interrupt enable bit is set.

Table 21-32 TTCAN: Reference Message REF_MSG_0 to REF_MSG_3 (0xc0 to 0xc3)

Bits	Name	Access	Function
28:0	REF_ID	rw- 0x0000	REFerence message IDentifier. If REF_IDE is 1 - REF_ID(28:0) is valid (extended ID) 0 - REF_ID(10:0) is valid (standard ID) REF_ID is used in TTCAN mode to detect a reference message. This holds for time slaves (reception) as well as for the time master (transmission). If the reference message is detected and there are no errors, then the Sync_Mark of this frame will become the Ref_Mark. REF_ID(2:0) is not tested and therefore the appropriate register bits are forced to 0. These bits are used for up to 8 potential time masters. CAN-CTRL recognizes the reference message only by ID. The payload is not tested. Additional note: A time master will transmit a reference message in the same way as a normal frame. REF_ID is intended for detection of a successful transmission of a reference message.
31	REF_IDE	rw-0	REFerence message IDE bit.

Table 21-3 Register Map gives a definition of the bit positions of REF_ID and REF_IDE inside REF_MSG_0 to REF_MSG_2.

A write access to REF_MSG_3 starts a data transfer of the reference message to the CAN clock

domain (clock domain crossing). During this automatic transfer a write lock for all registers REF_MSG_0 to

REF_MSG_2 is active. The transfer takes up to 6 clocks in the CAN clock domain and up to 6 clocks in the host clock domain. The write access to REF_MSG_2 is necessary to make a new reference message active.

Table 21-33 TTCAN: Trigger Configuration TRIG_CFG_0 (0xc4)

Bits	Name	Access	Function
5:0	TTPTR	rw-0x00	Transmit Trigger TB slot Pointer If TTPTR is too big and points to a slot that is not available, then TEIF is set and no new trigger can be activated after a write access to TT_TRIG_1. If TTPTR points to an empty slot, then TEIF will be set at the moment, when the trigger time is reached.
7:6	-	r-00	reserved

Table 21-34 TTCAN: Trigger Configuration TRIG_CFG_1 (0xc5)

Bits	Name	Access	Function
2:0	TTYPE(2:0)	rw-0x0	Trigger Type 000 - Immediate Trigger for immediate transmission 001 - Time Trigger for receive triggers 010 - Single Shot Transmit Trigger for exclusive time windows 011 - TransmitStart Trigger for merged arbitrating time windows 100 - TransmitStop Trigger for merged arbitrating time windows others - no action The time of the trigger is defined by TT_TRIG. TTPTR selects the TB slot for the transmit triggers.
3	-	r-0	reserved
7:4	TEW(3:0)	rw-0x0	Transmit Enable Window For a single shot transmit trigger there is a time of up to 16 ticks of the cycle time where the frame is allowed to start. TWE+1 defines the number of ticks. See chap. 21.5.4.3 for further details. TEW=0 is a valid setting and shortens the transmit enable window to 1 tick.

Table 21-35 TTCAN: Trigger Time TT_TRIG_0 and TT_TRIG_1 (0xc6 and 0xc7)

Bits	Name	Access	Function
7:0	TT_TRIG	rw-0x00	Trigger Time TT_TRIG(15:0) defines the cycle time for a trigger. For a transmission trigger the earliest point of transmission of the SOF of the appropriate frame will be TT_TRIG+1.

A write access to TT_TRIG_1 starts a data transfer of the trigger definition to the CAN clock domain (clock domain crossing) and activates the trigger. If the trigger is active, then a write lock for all registers TRIG_CFG_0, TRIG_CFG_1, TT_TRIG_0 and TT_TRIG_1 is active. The write lock becomes

inactive when the trigger time is reached (TTIF gets set is TTIE is enabled) or an error is detected (TEIF gets set).

Table 21-36 TTCAN: Watch Trigger Time TT_WTRIG_0 and TT_WTRIG_1 (0xc8 and 0xc9)

Bits	Name	Access	Function
7:0	TT_WTRIG	rw-0xff	Watch Trigger Time TT_WTRIG(15:0) defines the cycle time for a watch trigger. The initial watch trigger is the maximum cycle time 0xffff. See chap. 21.5.5 for more details.

A write access to TT_WTRIG_1 starts a data transfer of the trigger definition to the CAN clock domain (clock domain crossing). During this automatic transfer a write lock for the registers TT_WTRIG_0 and TT_WTRIG_1 is active. The transfer takes up to 6 clocks in the CAN clock domain and up to 6 clocks in the host clock domain. The write access to TT_WTRIG_1 is necessary to make a new trigger active.

Table 21-37 Receive Buffer Registers RBUF – Standard Format (r-0)

Address	Bit position								Function
	7	6	5	4	3	2	1	0	
RBUF	ID(7:0)								Identifier
RBUF+1	-					ID(10:8)			Identifier
RBUF+2	-								Identifier
RBUF+3	ESI	-							Identifier
RBUF+4	IDE=0	RTR	FDF	BRS	DLC(3:0)				Control
RBUF+5	KOER			TX	-				Status
RBUF+6	CYCLE_TIME(7:0)								TTCAN
RBUF+7	CYCLE_TIME(15:8)								TTCAN
RBUF+8	d1(7:0)								Data byte 1
RBUF+9	d2(7:0)								Data byte 2
.	…								.
RBUF+71	d64(7:0)								Data byte 64
RBUF+72	RTS(7:0)								CiA 603
.	…								.
RBUF+79	RTS(63:56)								CiA 603

Table 21-38 Receive Buffer Registers RBUF – Extended Format (r-0)

Address	Bit position								Function
	7	6	5	4	3	2	1	0	
RBUF	ID(7:0)								Identifier
RBUF+1	ID(15:8)								Identifier
RBUF+2	ID(23:16)								Identifier
RBUF+3	ESI	-		ID(28:24)					Identifier
RBUF+4	IDE=1	RTR	FDF	BRS	DLC(3:0)				Control
RBUF+5	KOER			TX	-				Status

RBUF+6	CYCLE_TIME(7:0)	TTCAN
RBUF+7	CYCLE_TIME(15:8)	TTCAN
RBUF+8	d1(7:0)	Data byte 1
RBUF+9	d2(7:0)	Data byte 2
.	…	.
RBUF+71	d64(7:0)	Data byte 64
RBUF+72	RTS(7:0)	CiA 603
.	…	.
RBUF+79	RTS(63:56)	CiA 603

The RBUF registers (0x00 to 0x4f) point the message slot with the oldest received message in the RB. All RBUF registers can be read in any order.

KOER in RBUF has the same meaning as the bits KOER in register EALCAP. KOER in RBUF becomes meaningful if RBALL=1.

Status bit TX in RBUF is set to 1 if the loop back mode (chap. 21.4.10.4) is activated and the core has received its own transmitted frame. This can be useful if LBME=1 and other nodes in the network do also transmissions.

The time-stamp CYCLE_TIME will be stored in RBUF only in TTCAN mode. This is the cycle time at the SOF of this frame. The cycle time of a reference message is always 0.

The Reception Time Stamps (RTS) for CiA 603 time-stamping are stored for each received message at the end of the RBUF address range. Therefore, in contrast to TTS, RTS is related to the actual selected RBUF slot.

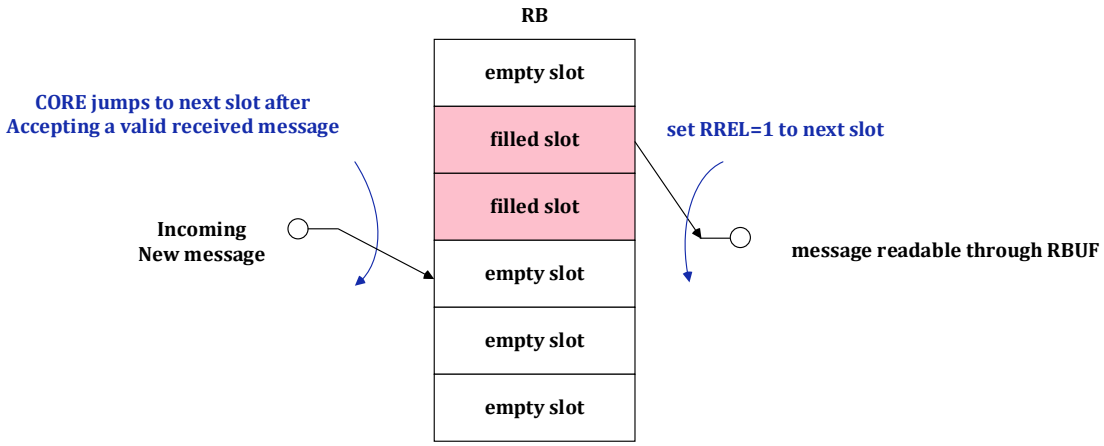


Figure 21-6 Schematic of the FIFO-like RB (example with 6 slots)

Table 21-39 Transmit Buffer Registers TBUF – Standard Format (rw-u)

Address	Bit position								Function
	7	6	5	4	3	2	1	0	
TBUF	ID(7:0)								Identifier

TBUF+1	-					ID(10:8)	Identifier
TBUF+2	-						Identifier
TBUF+3	TTSEN	-					Identifier
TBUF+4	IDE=0	RTR	FDF	BRS	DLC(3:0)		Control
TBUF+8	d1(7:0)						Data byte 1
TBUF+9	d2(7:0)						Data byte 2
.	...						.
TBUF+71	d64(7:0)						Data byte 64

Table 21-40 Transmit Buffer Registers TBUF – Extended Format (rw-u)

Address	**Bit position**								**Function**
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
TBUF	ID(7:0)								Identifier
TBUF+1	ID(15:8)								Identifier
TBUF+2	ID(23:16)								Identifier
TBUF+3	TTSEN	-		ID(28:24)					Identifier
TBUF+4	IDE=1	RTR	FDF	BRS	DLC(3:0)				Control
TBUF+8	d1(7:0)								Data byte 1
TBUF+9	d2(7:0)								Data byte 2
.	...								.
TBUF+71	d64(7:0)								Data byte 64

The TBUF registers (0x50 to 0x97) point the next empty message slot in the STB if TBSEL=1 or to the PTB otherwise. All TBUF registers can be written in any order. For the STB it is necessary to set TSNEXT to mark a slot filled and to jump to the next message slot.

Please mind the gap inside the addressing range of TBUF from TBUF+5 to TBUF+7. This is for better address segment alignment. The memory cells in the gap can be read and written, but have no meaning for the CAN protocol.

TBUF is build using a true 32bit wide memory and therefore a write access needs to be performed as 32bit write.

Both RBUF and TBUF include some frame-individual control bits (Table 21-41). For RBUF these bits signal the status of the appropriate CAN control field bits of the received CAN frame while for TBUF these bits select the appropriate CAN control field bit for the frame that has to be transmitted.

In contrast to RTS, which is stored for every received frame, TTS is stored only for the last transmitted frame if TTSEN=1. TTS is not related to the actual selected TBUF slot.

Table 21-41 Control bits in RBUF and TBUF

Bit	Description
IDE	IDentifier Extension 0 – Standard Format: ID(10:0) 1 – Extended Format: ID(28:0)
RTR	Remote Transmission Request 0 – data frame 1 – remote frame Only CAN 2.0 frames can be remote frames. There is no remote frame for CAN FD. Therefore, RTR is forced to 0 if FDF =1 in TBUF and RBUF. If a CAN FD frame is received with bit RRS=1, then this is ignored, a data payload is expected for reception instead and RTR in RBUF is overridden but the CRC of the frame is calculated with RRS=1.
FDF	CAN FD frame 0 – CAN 2.0 frame (up to 8 bytes payload) 1 – CAN FD frame (up to 64 bytes payload)
BRS	Bit Rate Switch 0– nominal / slow bit rate for the complete frame 1– switch to data / fast bit rate for the data payload and the CRC Only CAN FD frames can switch the bitrate. Therefore, BRS is forced to 0 if FDF =0.
ESI	Error State Indicator This is a read-only status bit for RBUF and is not available in TBUF. The protocol machine automatically embeds the correct value of ESI into transmitted frames. ESI is only included in CAN FD frames and does not exist in CAN 2.0 frames. 0 – CAN node is error active 1 – CAN node is error passive ESI in RBUF is always low for CAN 2.0 frames. The error state for transmission is shown with bit EPASS in register ERRINT.
TTSEN	Transmit Time-Stamp ENable For CiA 603 time-stamping the acquisition of a transmit time stamp TTS can be selected in the TBUF: 0 – no acquisition of a transmit time stamp for this frame 1 – TTS update enabled

The Data Length Code (DLC) in RBUF and TBUF defines the length of the payload – the number of payload bytes in a frame. See Table 21-42 for further details.

Remote frames (only for CAN 2.0 frames where FDF =0) are always transmitted with 0 payload bytes, but the content of the DLC is transmitted in the frame header. Therefore, it is possible to code some information into the DLC bits for remote frames. But then care needs to be taken if different CAN nodes are allowed to transmit a remote frame with the same ID. In this case all transmitters need to use the same DLC because otherwise this would result in an unresolvable collision.

Table 21-42 Definition of the DLC (according to the CAN 2.0 / FD specification)

DLC (binary)	Frame Type	Payload in Bytes
0000 to 1000	CAN 2.0 and CAN FD	0 to 8
1001 to 1111	CAN 2.0	8
1001	CAN FD	12
1010	CAN FD	16
1011	CAN FD	20
1100	CAN FD	24
1101	CAN FD	32
1110	CAN FD	48
1111	CAN FD	64

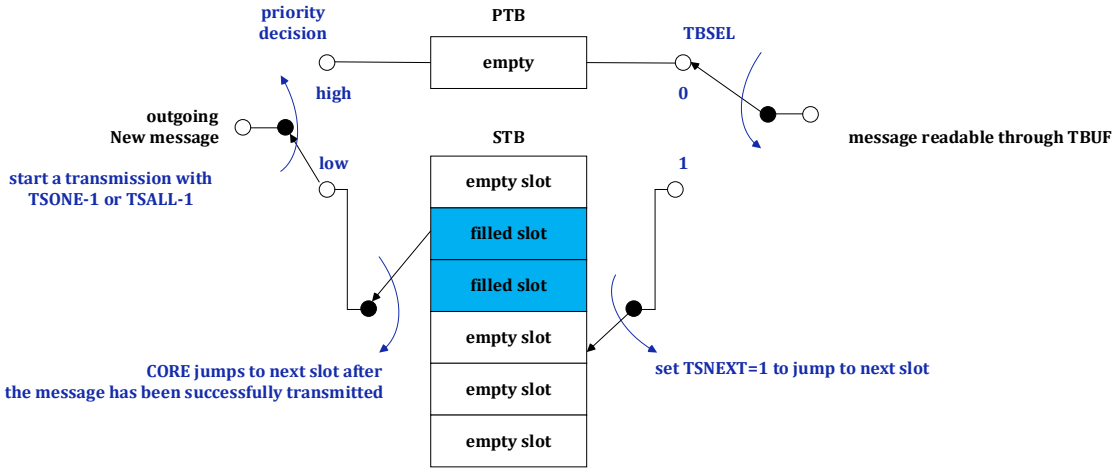


Figure 21-7 Schematic of PTB and STB in FIFO mode (empty PTB and 6 STB slots)

21.4 General Operation

This chapter describes handling of CAN communication. Before communication is possible, the CAN- CTRL core has to be configured to match the CAN bus timings.

21.4.1 Acceptance Filters

To reduce the load of received frames for the host controller, the core uses acceptance filters. The CAN- CTRL core checks the message identifier during acceptance filtering. Therefore, the length of each acceptance filter is 29 bits.

If a message passes one of the filters, then it will be accepted. If accepted, the message will be stored into the RB and finally RIF is set if RIE is enabled. If the message is not accepted, RIF is not set and the RB FIFO pointer is not increased. Messages that are not accepted will be discarded and overwritten by the next message. No stored valid message will be overwritten by any not accepted message.

Independently of the result of acceptance filtering, the CAN-CTRL core checks every message on the bus and sends an acknowledge or an error frame to the bus.

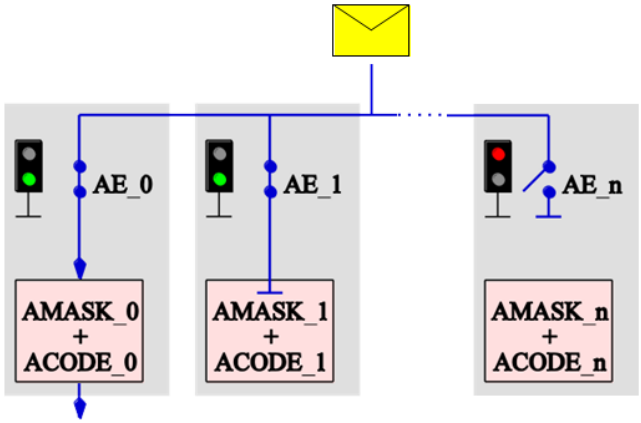


Figure 21-8 Example of acceptance filtering

The acceptance mask defines which bits shall be compared while the acceptance code defines the appropriate values. Setting mask bits to 0 enables the comparison of the selected acceptance code bits with the corresponding message identifier bits. Mask bits that are set to 1 are disabled for the acceptance check and this results in accepting the message.

The identifier bits will be compared with the corresponding acceptance code bits ACODE as follows:

- standard: ID(10:0) with ACODE(10:0)
- extended: ID(28:0) with ACODE(28:0)

Example: If AMASK_x(0)=0 and all other AMASK_x bits are 1 then the value of the last ID bit has to be equal to ACODE(0) for an accepted message. All other ID bits are ignored by the filter.

Figure 21-8 gives an example of acceptance filtering using several filters. In this example the filter 0 and 1 are enabled by the appropriate AE_x bits in the ACF_EN_x registers. All other filters are disabled and therefore do not accept any message. For the enabled filters the combination of AMASX_x and ACODE_x defines if a message is accepted (like in the example for filter 0) or not accepted (like in the example for filter 1).

Note: Disabling a filter by setting AE_x=0 blocks messages. In contrast to this disabling a mask bit in AMASK_x disables the check for this bit which results in accepting messages.

The definitions of AMASK and ACODE alone do not distinguish between standard or extended frames. If bit AIDEE=1 then the value of AIDE defines with frame type is accepted. Otherwise, if AIDE=0 both types are accepted.

After power-on reset the CAN-CTRL core is configured to accept all messages. (Filter 0 is enabled by AE_0=1, all bits in AMASK_0 are set to 1 and AIDEE=0. All other filters are disabled. Filter 0 is the only filter that has defined reset values for AMASK / ACODE while all other filters have undefined reset values.)

21.4.2 Message Reception

The received data will be stored in the RB as shown in Figure 21-9. The RB has FIFO-like behavior. Every received message that is valid and accepted sets RIF=1 if RIE is enabled. RSTAT is set depending of the fill state. When the number of filled buffers is equal to the programmable value AFWL then RAFIF is set if RAFIE is enabled. In case, when all buffers are full, the RFIF is set if RFIE is enabled.

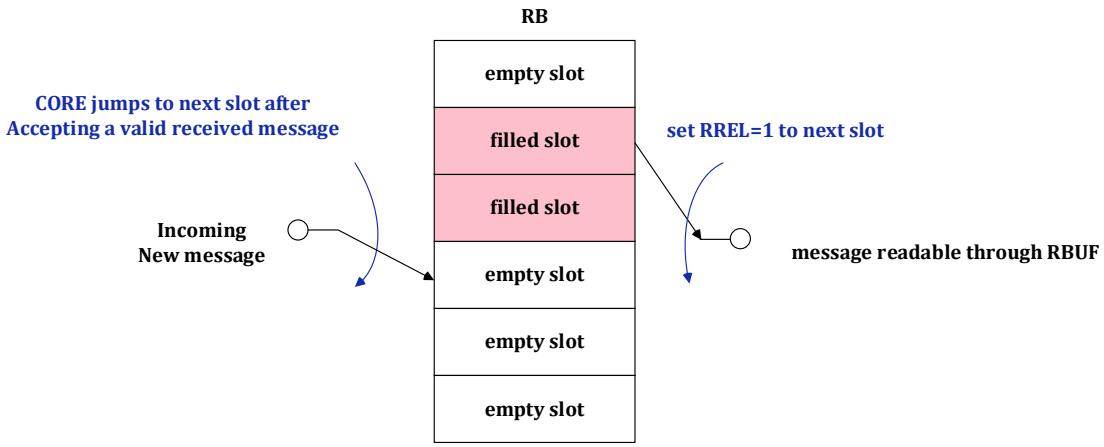


Figure 21-9 Schematic of the FIFO-like RB (example with 6 slots)

The RB always maps the message slot containing the oldest message to the RBUF registers.

The maximum payload length for CAN 2.0 messages is 8 bytes and for CAN FD messages 64 bytes. The individual length of each message is defined by the DLC. Because of this the RB provides slots for each message and the host controller is required to set RREL to jump to the next RB slot. All RBUF bytes of the actual slot can be read in any order.

If the RB is full, the next incoming message will be stored temporarily until it passes for valid (6^{th} EOF bit). Then if ROM=0 the oldest message will be overwritten by the newest or if ROM=1 the newest message will be discarded. In both cases ROIF is set if ROIE is enabled. If the host controller reads the oldest message and sets RREL before a new incoming message becomes valid then no message will be lost.

21.4.3 Handling message receptions

Without acceptance filtering the CAN-CTRL core would signal the reception of every frame and the host would be required to decide if it was addressed. This would result in quite a big load on the host controller.

It is possible to disable interrupts and use the acceptance filters to reduce the load for the host controller. For a basic operation RIF is set to 1 if RIE is enabled and the CAN-CTRL core has received a

valid message. To reduce the number of reception interrupts it is possible to use RAIE / RAIF (RB Almost full Interrupt) or RFIE / RFIF (RB Full Interrupt) instead of RIE / RIF (Reception Interrupt). The “almost full limit” is programmable using AFWL.

1. Read the oldest message from the RB FIFO using the RBUF registers.
2. Release the RB slot with RREL=1. This selects the next message (the next FIFO slot). RBUF will be updated automatically.
3. Repeat these actions until RSTAT signals an empty RB.

If the RB FIFO is full and a new received message is recognized as valid (6th EOF bit) then one message will be lost (see bit ROM). Before this event, no message is lost. This should give enough time for the host controller to read at least one frame from the RB after the RB FIFO has been filled and the selected interrupt has occurred. To enable this behavior the RB includes one more (hidden) slot. This hidden slot is used to receive a message, validate it and check it if it matches the acceptance filters before an overflow occurs.

21.4.4 Message Transmission

Before starting any transmission, at least one of the transmit buffers (PTB or STB) has to be loaded with a message (Figure 21-10). TPE signals if the PTB is locked and TSSTAT signals the fill state of the STB. The TBUF registers provide access to both the PTB as well as to the STB. Below is the recommended programming flow:

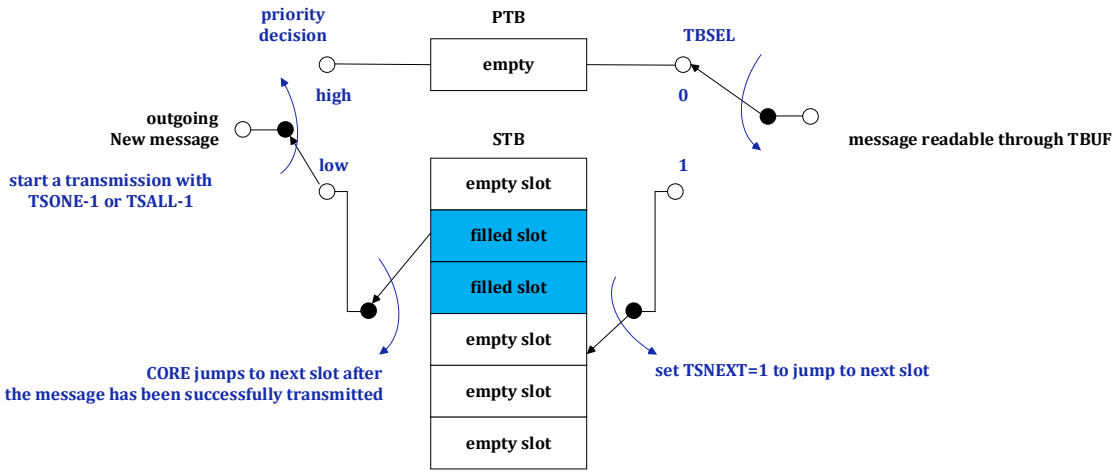


Figure 21-10 Schematic of PTB and STB in FIFO mode (empty PTB and 6 STB slots)

The maximum payload length for CAN 2.0 messages is 8 bytes and for CAN FD messages 64 bytes. The individual length of each message is defined by the DLC. For remote frames (bit RTR) the DLC becomes meaningless, because remote frames always have a data length of 0 bytes. The host controller is required to set TSNEXT to jump to the next STB slot. All TBUF bytes can be written in any order.

Setting TSNEXT=1 is meaningless if TBSEL=0 selects the PTB. In this case TSNEXT is

automatically cleared and does no harm.

Bit TPE should be set to start a transmission when using the PTB. To use the STB, TSONE has to be set to start a transmission of a single message or TSALL to transmit all messages.

The PTB has always a higher priority than the STB. If both transmit buffers have the order to transmit, the PTB message will be always sent first regardless of the frame identifiers. If a transmission from the STB is already active, it will be completed before the message from the PTB is sent at the next possible transmit position (the next interframe slot). After the PTB transmission is completed or aborted, the CAN- CTRL core returns to process other pending messages from the STB.

When the transmission is completed, the following transmission interrupts are set:

- For the PTB, TPIF is set if TPIE is enabled
- For the STB using TSONE, TSIF is set if one message has been completed and TSIE is enabled.
- For the STB using TSALL, TSIF is set if all messages have been completed and if TSIE is enabled. In other words: TSIE is set if the STB is empty. Therefore, if the host controller writes an additional message to the STB after a TSALL transmission has been started then the additional message will be also transmitted before TSIF will be set.

21.4.5 Message transmission abort

If the situation arises, where a message in a transmit buffer cannot be sent due to its low priority, this would block the buffer for a long time. In order to avoid this, the host controller can withdraw the transmission request by setting TPA or TSA respectively, if the transmission has not yet been started.

Both TPA and TSA source a single interrupt flag: AIF. The CAN protocol machine executes an abort only if it does not transmit anything to the CAN bus. Therefore, the following is valid:

- There is no abort during bus arbitration.
    - If the node loses arbitration, the abort will be executed afterwards.
    - If the node wins arbitration, the frame will be transmitted.
- There is no abort while a frame is transmitted.
    - If a frame is transmitted successfully then a successful transmission is signaled to the host controller. In this case no abort is signaled. This is done by the appropriate interrupt and status bits.
    - After an unsuccessful transmission where the CAN node does not receive an acknowledge, the error counter is incremented and the abort will be executed.
    - If there is at least one frame left in the STB, while the host has commanded all frames to be transmitted (TSALL=1), then both the completed frame as well as the abort is signaled to the host.

Because of these facts aborting a transmission may take some time depending on the CAN communication speed and frame length. If an abort is executed, this results in the following actions:

- TPA releases the PTB which results in TPE=0.

The frame data is still stored in the PTB after releasing the PTB.

- TSA releases one single message slot or all message slots of the STB. This depends on whether TSONE or TSALL was used to start the transmission. TSSTAT will be updated accordingly. Releasing a frame in the STB results in discarding the frame data because the host cannot access it.

Setting both TPA and TSA simultaneously is not recommended. If a host controller decides to do it anyway, then AIF will be set and both transmissions from PTB and STB will be aborted if possible. As already stated, if one transmission will be completed before the abort can be executed this will result in signaling a successful transmission. Therefore, the following interrupt flags may be set if enabled:

- AIF (once for both PTB and STB transmission abort)
- TPIF + AIF
- TSIF + AIF
- TPIF + TSIF (very seldom, will only happen if the host does not handle TPIF immediately)
- TPIF + TSIF + AIF (very seldom, will only happen if the host does not handle TPIF and TSIF immediately)

To clear the entire STB both TSALL and TSA need to be set. In order to detect if a message cannot be sent for a long time because it loses arbitration the host may use the ALIF / ALIE.

21.4.6 A Full STB

After writing a message to the STB, TSNEXT=1 marks a buffer slot filled and jumps to the next free message slot. TSNEXT is automatically reset to 0 by the CAN-CTRL core after this operation.

If the last message slot has been filled and therefore all message slots are occupied then TSNEXT stays set until a new message slot becomes free. While TSNEXT=1, then writing to TBUF is blocked by the CAN-CTRL core.

When a slot becomes free, then the CAN-CTRL core automatically resets TSNEXT to 0. A slot becomes free if a frame form the STB is transmitted successfully or if the host requests an abort (TSA=1). If a TSALL transmission is aborted, then TSNEXT is also reset, but additionally the complete STB is marked as empty.

21.4.7 Error Handling

On one hand CAN-CTRL does automatic error handling which means that in most cases the host controller does not care about errors. This includes automatic message retransmission and automatic deletion of received messages with errors. On the other hand, if required by the host, CAN-CTRL may optionally give detailed information about errors and signal every error to the host by interrupt. This enables to run an application like a CAN bus monitor at the host.

Every CAN node has 3 states of error handling:

1. Error-Active: The node automatically transmits active error frames upon detection of an error.
2. Error-Passive: The node transmits passive error frame upon detection of an error. This means that it does not transmit a dominant value to the bus, but expects an error frame.
3. Bus Off: After too many errors a node goes “bus off” where it stops touching the bus.

To handle the 3 error states every CAN node has two error counters: the transmit and the receive error counter. Both are incremented and decremented as defined by the CAN specification and the node enters the appropriate error state upon reaching a counter level defined by the CAN specification.

The error counters are incremented if there are errors. Dangerous errors that are probably caused by this node will result in incrementing the counter by 8, errors that are probably caused by other nodes will result in incrementing the counter by 1. Valid frame transmission or valid receptions result in decrementing the counters. All this is defined by the CAN specification and handled automatically by CAN-CTRL.

An error frame (better: an error flag) is different to a data frame. It is a dominant pulse for at least 6 consecutive dominant bits, which is a stuff bit violation for other nodes. A host controller cannot command to transmit an error frame. It is done fully automatically by CAN-CTRL.

If CAN-CTRL is commanded to transmit a frame then it automatically tries retransmissions as fast as possible until the frame gets transmitted without error or the node goes bus off. If a frame is received and CAN-CTRL detects an error, then the received data is discarded. Because of the automatically transmitted error frame the sender of the frame will retransmit the frame. Frames in RBUF are never overwritten by frames with errors. Only valid received frames may result in a RBUF overflow.

21.4.8 The Bus Off State

The “bus off” state is signaled using the status bit BUSOFF in register CFG_STAT (Table 21-5). A CAN node enters the “bus off” state automatically if its transmit error counter becomes >255. Then it will not take part in further communications until it returns into the error-active state again. Setting BUSOFF to 1 also sets the EIF interrupt if EIE is enabled. A CAN node returns to error-active state if it is reset by a power-on reset or if it receives 128 sequences of 11 recessive bits (recovery sequences). Please note that between each recovery sequence the bus may have dominant state.

In the “bus off” state, RECNT and TECNT stay unchanged. Please note that while entering bus off state TECNT rolls over and therefore may hold a small value. Therefore, it is recommended to use TECNT before the node enters bus off state and status bit BUSOFF afterwards.

If the node recovers from “bus off” state then RECNT and TECNT are automatically reset to 0.

If a frame is pending for transmission but the CAN node has entered bus off state, then this frame is kept pending. If a node returns to error-active state after bus off, then the transmission of the pending frame will be tried. If this is not desired, then the frame should be aborted by the host controller.

21.4.9 Extended Status and Error Report

During CAN bus communication transmission errors may occur. The following features support detection and analysis of them. This can be used for extended bus monitoring.

21.4.9.1 Programmable Error Warning Limit

Errors during reception / transmission are counted by RECNT and TECNT. A programmable error warning limit EWL, located in register LIMIT, can be used by the host controller for flexible reaction on those events. The limit values can be chosen in steps of 8 errors from 8 to 128:

Limit of count of errors = (EWL+1)×8

The interrupt EIF will be set if enabled by EIE under the following conditions:

- the border of the error warning limit has been crossed in either direction by RECNT or TECNT or
- the BUSOFF bit has been changed in either direction.

21.4.9.2 Arbitration Lost Capture (ALC)

The core is able to detect the exact bit position in the Arbitration Field where the arbitration has been lost. This event can be signaled by the ALIF interrupt if it is enabled. The value of ALC stays unchanged if the node is able to win the arbitration. Then ALC holds the old value of the last loss of arbitration.

The value of ALC is defined as follows: A frame starts with the SOF bit and then the first bit of the ID is transmitted. This first ID bit has ALC value 0, the second ID bit ALC value 1 and so on. See Figure 21-4 for the bit order in all types of CAN 2.0 and CAN FD frames.

Arbitration is only allowed in the arbitration field (Figure 21-4). Therefore, the maximum value of ALC is 31 which is the RTR bit in extended frames.

Additional hint: If a standard remote frame arbitrates with an extended frame, then the extended frame loses arbitration at the IDE bit and ALC will be 12. The node transmitting the standard remote frame will not notice that an arbitration has been taken place, because this node has won.

It is impossible to get an arbitration loss outside of the arbitration field. Such an event is a bit error.

21.4.9.3 Kind Of Error (KOER)

The core recognizes errors on the CAN bus and stores the last error event in the KOER bits. A CAN bus error can be signaled by the BEIF interrupt if it is enabled. Every new error event overwrites the previous stored value of KOER. Therefore, the host controller has to react quickly on the error event. The error codes are described in Table 21-21.

KOER is updated with each new error. Therefore, it stays untouched when frames are successfully transmitted or received. This opens the opportunity for a delayed error investigation.

21.4.9.4 Reception of All Data Frames (RBALL)

If RBALL=1 then all received data frames are stored even those with error. This holds also for loop back mode (chap.21.4.10.4). Only data frames are stored in RBUF. Error frames or overload frames are not stored.

If CiA 603 time-stamping is enabled (TIMEEN=1) and the timestamp is configured for the EOF (TIMEPOS=1) then in the case of an error the time-stamp is acquired at the start of the error frame.

Most of the possible errors can only occur if the node is the transmitter of the frame. In this case a frame would only be stored in RBUF if a loop back mode is activated. Depending on the type of error parts of the frame stored in an RBUF slot may be valid while other parts are unknown. Table 21-43 lists the possibilities.

Table 21-43 RBALL and KOER

KOER	Condition	Description
no error	All	Successful reception.
BIT ERROR	Receiver	Can only occur in the ACK slot. All stored data are valid.
	Transmitter	The payload is always invalid. The header bits including the ID may be valid. In the arbitration phase detection of a wrong bit is part of the arbitration and therefore not a BIT ERROR. But if a stuff bit in the arbitration phase is detected wrong by the transmitter of the frame, then this is a BIT ERROR and in this case the header bits are invalid. Therefore, the header bits cannot be trusted, but if the header matches to an expected header of a frame; it can be used for further investigation.
FORM ERROR	All	Only FORM ERRORs in a data frame are covered. This includes errors in the CRC delimiter, the ACK delimiter or the EOF bits. All stored data are valid.
STUFF ERROR	Receiver	The position of a STUFF ERROR is unknown. All stored data are invalid.
	Transmitter	Can only happen during arbitration. All stored data are invalid.
ACK ERROR	Receiver	Can only occur if the node is in LOM. All stored data are valid.
	Transmitter	Can only occur in loop back mode without self-ACK. All stored data are valid.
CRC ERROR	Receiver	All stored data are invalid.

Please note that even if (parts of) the data are valid under certain conditions, then care must be taken. There is no guarantee for this and it depends on the reason for the error if this statement is correct. Example: if the node has a bad bit timing configuration, then there is no correct

synchronization and therefore no valid data. Therefore, the statement that (parts of) the data is valid signals only the possibility that it is so.

21.4.10 Extended Features

21.4.10.1 Single Shot Transmission

Sometimes an automatic re-transmission is not desired. Therefore, the order to transmit a message only once can be set separately for the transmit buffers PTB by the bit TPSS and for the transmit buffer STB by the bit TSSS. In this case no re-transmission will be performed in the event of an error or arbitration lost if the selected transmission is active.

In the case of an immediate successful transmission there is no difference to normal transmission. But in the case of an unsuccessful transmission the following will happen:

- TPIF gets set if enabled, the appropriate transmit buffer slot gets cleared
- In case of an error, KOER and the error counters get updated. BEIF gets set if BEIE is enabled and the other error interrupt flags will act accordingly.
- In case of a lost arbitration, ALIF gets set if ALIE is enable.

Therefore, for single shot transmission TPIF alone does not indicate if the frame has been successfully transmitted. Therefore, single shot transmission should only be used together with BEIF and ALIF.

If single shot transmission is used with TSALL and there is more than one frame in the STB then for each frame a single shot transmission is done. Regardless if any of the frames is not successfully transmitted (e.g., because of an ACK error) the CAN-CTRL advances to the next frame and stops if the STB is empty. Therefore, in this scenario only the error counters indicate what has happened. This can be quite complex to evaluate because if one of two frames got errors the host cannot detect which one was the successful one.

If the bus is occupied by another frame, if a single shot transmission is started, then CAN-CTRL waits till the bus is idle and then tries to transmit the single shot frame.

21.4.10.2 Listen Only Mode (LOM)

LOM provides the ability of monitoring the CAN bus without any influence to the bus. LOM by CAN-CTRL is compatible to the bus monitoring feature defined in the CAN FD specification.

Another application is the automatic bit rate detection where the host controller tries different timing settings until no errors occur.

Errors will be monitored (KOER, BEIF) in LOM.

In LOM the core is not able to write dominant bits onto the bus (no active error flags or overload flags, no acknowledge). This is done using the following rules:

- In LOM the protocol machine acts as if in error passive mode where only recessive error flags are generated. Only the protocol machine acts as if in error passive mode. All other

components including the status registers are not touched.

- In LOM the protocol machine does not generate a dominant ACK.
- The error counters stay unchanged regardless of any error condition. Important facts regarding ACKs for LOM:
- If a frame is transmitted by a node, then an ACK visible at the bus will be only generated if at least one additional node is attached to the bus that is not in LOM. Then there will be no error and all nodes (also those in LOM) will receive the frame.
- If there is an active or passive error flag after an ACK error, then the node in LOM is able to detect this as ACK error.

Activation of LOM should not be done while a transmission is active. The host controller has to take care of this. There is not additional protection from CAN-CTRL.

If LOM is enabled then no transmission can be started.

The loop back mode (external) LBME (chap. 21.4.10.4) plays an important role for the behavior of LOM. If LBME is disabled, then LOM acts as described above and the node cannot write any dominant bit to the bus. But if LBME is activated, then the node is allowed to transmit a frame. LBME allows only the transmission of a frame including the optional self-ACK (SACK), but the node will not respond with an ACK to frames from other nodes and will not generate error or overload frames. In Summary the combination of LOM and LBME is “a silent receiver, that is able to transmit if necessary”.

21.4.10.3 Bus Connection Test

To check if a node is connected to the bus the following steps shall be done:

- Transmit a frame. If the node is connected to the bus, then its TX bits are visible on its RX input.
- If there are other nodes connected to the CAN bus, then a successful transmission (including an acknowledge from the other nodes) is expected. No error will be signaled.
- If the node is the only node that is connected to the CAN bus (but the connection between the bus, the transceiver and the CAN-CTRL core is fine) then the first regular error situation occurs in the ACK slot because of no acknowledge from other nodes. Then a BEIF error interrupt will be generated if enabled and KOER= “100” (ACK error).
- If the connection to the transceiver or the bus is broken then immediately after the SOF bit the BEIF error interrupt will be set and KOER= “001” (BIT error).

21.4.10.4 Loop Back Mode (LBMI and LBME)

CAN-CTRL supports two Loop Back Modes: internal (LBMI) and external (LBME). Both modes result in reception of the own transmitted frame which can be useful for self-tests. See Figure 21-11 for details.

In LBMI CAN-CTRL is disconnected from the CAN bus and the txd output is set to recessive. The output data stream is internally fed back to the input. In LBMI the node generates a self-ACK to avoid an ACK error.

In LBME CAN-CTRL stays connected to the transceiver and a transmitted frame will be visible on the bus. With the help of the transceiver CAN-CTRL receives its own frame. In LBME the node does not generate a self-ACK when SACK=0, but generates a self-ACK when SACK=1. Therefore, in LBME

with SACK=0 there are two possible results upon a frame transmission:

1. Another node receives the frame too and generates an ACK. This will result in a successful transmission and reception.
2. No other node is connected to the bus and this results in an ACK error. To avoid retransmissions and incrementing the error counters it is recommended to use TPSS or TSSS if it is unknown if other nodes are connected.

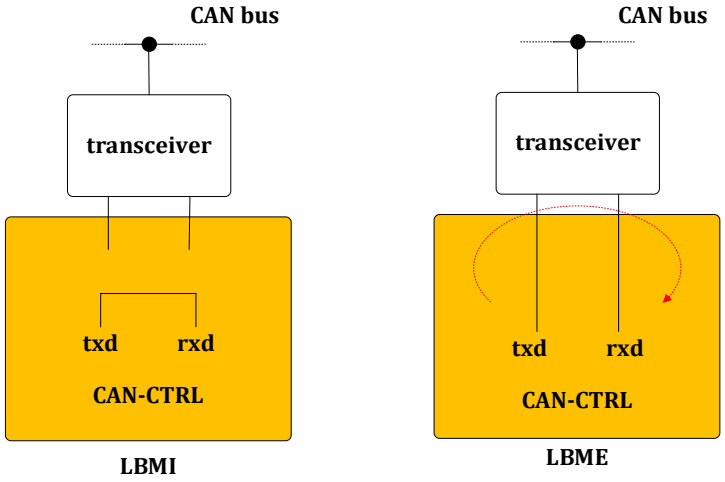


Figure 21-11 Loop Back Mode: Internal and External

In Loop Back Mode the core receives its own message, stores it in the RBUF and sets the appropriate receive and transmit interrupt flags if enabled.

LBMI can be useful for chip-internal and software tests while LBME can test the transceiver and the connections to it.

Activation of LBMI or LBME should not be done while a transmission is active. The host controller has to take care of this. There is not additional protection from CAN-CTRL.

If the node is connected to a CAN bus then switching back from LBMI to normal operation must not be done by simply setting LBMI to 0, because then it may be that this occurs just at the moment while another CAN node is transmitting. In this case switching back to normal operation shall be done by setting bit RESET to 1. This automatically clears LBMI to 0. Finally RESET can be disabled and the core returns back to normal operation. In contrast to this LBME can be disabled every time.

LBME can be used in combination with LOM.

21.4.10.5 Error Counter Reset

According to the CAN / CAN FD standard RECNT counts receive errors and TECNT counts transmit errors. After too many transmit errors a CAN node has to go to bus off state (chap. 21.4.8). Bit RESET does not modify the errors counters or the bus off state. The CAN / CAN FD specification defines rules how to disable bus off state and to decrease the error counters. A good node will recover from all of this automatically if only a temporary error has caused the problems. The classic CAN 2.0B specification requires this automatic behavior without host controller interaction to avoid the

babbling idiot problem.

The CAN FD specification relaxes this and allows the host controller to override the automatic error counter handling. But this should be used with extra care and is suggested to use only for debugging purposes.

Writing a 1 to the bit BUSOFF resets the error counters and therefore forces the node to leave the bus off state. This is done without setting EIF.

21.4.11Software Reset

If bit RESET in CFG_STAT is set to 1 then the software reset is active. Several components are forced to a reset state if RESET=1 but no all components are touched by RESET. Some components are only sensitive to a hardware reset. The reset values of all bits are always the same for software and hardware reset.

Table 21-44 Software Reset

Register	RESET	Comment
ACFADR	No	-
ACODE_x	No	Register is writeable if RESET=1 and write-locked if 0.
AE_x	No	-
AFWL	No	-
AIF	Yes	-
ALC	Yes	-
ALIE	No	-
ALIF	Yes	-
AMASK_x	No	Register is writeable if RESET=1 and write-locked if 0.
BEIE	No	-
BEIF	Yes	-
BUSOFF	(No)	An error counter reset by setting BUSOFF=1 also resets BUSOFF.
EIE	No	-
EIF	No	-
EPASS	No	-
EPIE	No	-
EPIF	Yes	-
EWARN	No	-
EWL	Yes	-
FD_ISO	No	Register is writeable if RESET=1 and write-locked if 0.
F_PRESC	No	Register is writeable if RESET=1 and write-locked if 0.
F_Seg_1	No	Register is writeable if RESET=1 and write-locked if 0.
F_Seg_2	No	Register is writeable if RESET=1 and write-locked if 0.
F_SJW	No	Register is writeable if RESET=1 and write-locked if 0.
KOER	Yes	-

Register	RESET	Comment
LBME	Yes	-
LBMI	Yes	-
LOM	No	-
RACTIVE	Yes	Reception is immediately cancelled even if a reception is active. No ACK will be generated.
RAFIE	No	-
RAFIF	Yes	-
RBALL	Yes	-
RBUF	(Yes)	All RB slots are marked as empty. RBUF contains unknown data.
RECNT	No	An error counter reset is possible by setting BUSOFF=1.
REF_ID	No	-
REF_IDE	No	-
RFIE	No	-
RFIF	Yes	-
RIE	No	-
RIF	Yes	-
ROIE	No	-
ROIF	Yes	-
ROM	No	-
ROV	Yes	All RB slots are marked as empty.
RREL	Yes	-
RSTAT	Yes	All RB slots are marked as empty.
SACK	Yes	-
SELMASK	No	-
STBY	No	-
S_PRESC	No	Register is writeable if RESET=1 and write-locked if 0.
S_Seg_1	No	Register is writeable if RESET=1 and write-locked if 0.
S_Seg_2	No	Register is writeable if RESET=1 and write-locked if 0.
S_SJW	No	Register is writeable if RESET=1 and write-locked if 0.
TACTIVE	Yes	All transmissions are immediately terminated with RESET. If a transmission is active this will result in an erogenous frame. Other nodes will generate error frames.
TBE	Yes	-
TBF	Yes	-
TBPTR	No	-
TBSEL	Yes	TBUF fixed to point to PTB.
TBUF	(Yes)	All STB slots are marked as empty. Because of TBSEL TBUF points to the PTB.
TECNT	No	An error counter reset is possible by setting BUSOFF=1.
TEIF	Yes	-
TIMEEN	No	-
TIMEPOS	No	-
TPA	Yes	-
TPE	Yes	-

Register	RESET	Comment
TSA	Yes	-
TSALL	Yes	-
TSMODE	No	-
TSNEXT	Yes	-
TSONE	Yes	-
TPIE	No	-
TPIF	Yes	-
TPSS	Yes	-
TSFF	Yes	All STB slots are marked as empty.
TSIE	No	-
TSIF	Yes	-
TSSS	Yes	-
TSSTAT	Yes	All STB slots are marked as empty.
TTEN	Yes	-
TTIF	Yes	-
TTIE	No	-
TTPTR	No	-
TTS	No	-
TTTBM	No	-
TTYPE	No	-
TT_TRIG	No	-
TT_WTRIG	No	-
T_PRESC	No	-
WTIE	No	-
WTIF	Yes	-

21.5 Time-Triggered CAN

Time-Triggered CAN (TTCAN) is an operation mode according to ISO 11898-4 where frames will be transmitted only at predefined time windows. There are three time window types:

- Exclusive time window (only one node is allowed to transmit one frame with a defined ID)
- Free time window (unused time window for further extension of the network)
- Arbitrating time window (several nodes may transmit a frame and arbitration takes place)

The timing is defined offline by a TTCAN system administrator and organized in a system matrix as shown in Figure 21-12. A row is called a basic cycle and it starts with a reference message. The reference message is transmitted by the time master. Other messages can be transmitted by any node including the time master.

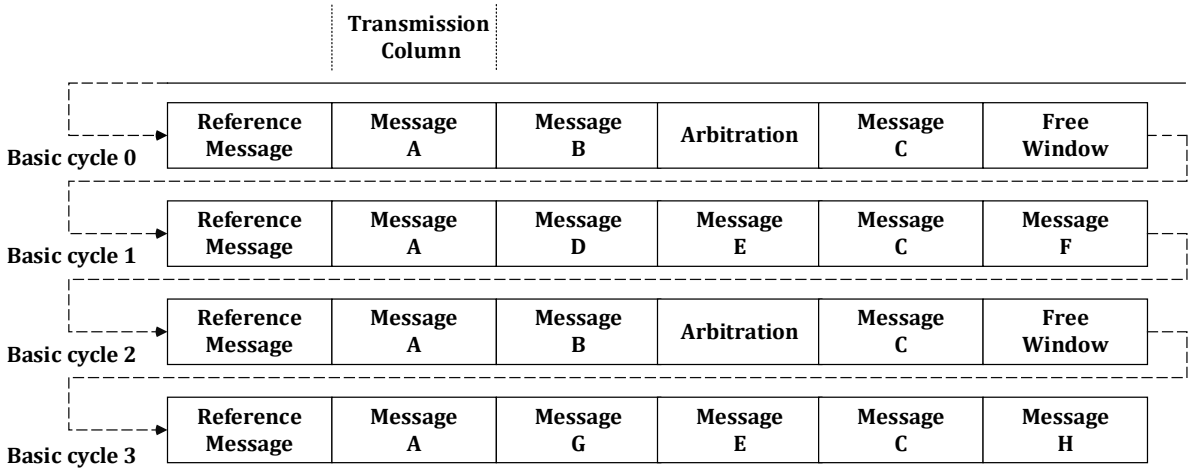


Figure 21-12 Example System Matrix

The time of the SOF of a message is a Sync_Mark. The Sync_Mark of a reference message is the Ref_Mark. Timing is done by a free-running 16bit timer. The difference between the timer and the value of the Ref_Mark is the cycle time. In other words: each basic cycle starts the cycle time at the Ref_Mark. The cycle time is stored in the RBUF as a time stamp.

The length of a time window is defined by the TTCAN administrator and it is long enough to carry one message. Therefore, messages must be transmitted in single shot mode. There is only one exception from this rule: if several arbitrating time windows are merged then it is possible to enable retransmission. But this needs to be disabled early enough to not influence the following time window.

To transmit a frame in a time window, CAN-CTRL offers a hardware trigger. This trigger needs to be configured by the host. The desired frame needs to be stored in a TBUF slot and the trigger needs to point to this slot. Then if the cycle time comes to the time that is defined for the trigger, CAN-CTRL automatically transmits the frame in single shot mode.

If the trigger gets active this will be signaled to the host by an interrupt. Then the host needs to configure the next trigger for the next time window. This requires a real-time response by the host. The duration of one time window is the time that the host is allowed to use.

If CAN-CTRL needs to operate as time master, then the reference message needs to be placed in a TBUF slot like all other messages and will be transmitted when the hardware trigger gets active.

Detection of a reference message is done automatically by all of the time slaves and the time master. This is done by setting the correct REF_ID and REF_IDE bit of the reference message in the REF_MSG_x registers. Upon detection CAN-CTRL automatically updates the Ref_Mark which starts the cycle time.

Additionally, to the trigger for messages, CAN-CTRL includes a watch trigger. This should be used to check whether the time since the last reference message has been too long. The host shall configure the watch trigger for the periodic case where each basic cycle is followed by the next basic cycle or for the event-triggered case where there is a time gap between the basic cycles and the next

cycle is started upon an event. If the watch trigger gets active, this results in the watch interrupt.

21.5.1 The TBUF in TTCAN Mode

The behavior if the TBUF depends on the register bit TTTBM. If TTTBM=1, then each TBUF slot can be addressed by the host controller which is required to use transmission triggers. Otherwise, if TTTBM=0, then only time-stamping and time triggers are supported.

21.5.1.1 TBUF in TTCAN Mode if TTTBM=1

In TTCAN mode with TTTBM=1 the STB is used as an array of message slots. Each slot can be addressed by TBPTR. The host can mark a slot as filled or as empty using TBF and TBE. Filled slots are write-locked. TBSEL and TSNEXT have no meaning in TTCAN mode and are ignored.

The PTB can be addressed by TBPTR=0. This makes the PTB useable as any other STB slot. In fact, the PTB has no special properties in this mode and is associated with the STB. This furthermore results in the fact that a successful transmission is always signaled using TSIF.

There is no FIFO operation and no priority decision for the TBUF in TTCAN mode. Furthermore, onlyone frame can be selected for transmission.

A message trigger defines the time when a message needs to be transmitted (the beginning of a time window) and it selects the message using the pointer TTPTR. If the trigger event takes place, then selected message transmission is started. Finally, the trigger interrupt is set to signal the host that the next action needs to be prepared.

All transmissions can only be started using a trigger. See chap. 4.4 for details. TPE, TSONE, TSALL, TPSS and TPA are fixed to 0 and ignored in TTCAN mode.

21.5.1.2 TBUF in TTCAN Mode if TTTBM=0

TTTBM=0 offers the combination of event-driven CAN communication and time-stamping for received messages.

In this mode the PTB and the STB provide the same behavior as if TTEN=0. Then the PTB always has higher priority than the STB and the STB can operated in FIFO mode (TSMODE=0) or in priority mode (TSMODE=1).

21.5.2 TTCAN Operation

After power-up a time master needs to do the initialization as defined in ISO 11898-4. There may be up to 8 potential time masters in a CAN network. Each one has its own reference message ID (the last 3 bits of the ID). Potential time masters may try to transmit their reference message according to their priority. Lower-prioritized time masters shall try to transmit their reference messages later.

If TTEN is set, then the 16bit timer is running. If a reference message is detected or if a time master successfully transmits its reference message, then CAN-CTRL copies the Sync_Mark of this message to the Ref_Mark which sets the cycle time to 0. The reception of the reference message will set RIF respectively the successful transmission will set TPIF or TSIF. Then the host needs to prepare the trigger for the next action.

A trigger for an action can be a reception trigger. This just triggers the interrupt and it can be used to detect if an expected message is not received. Such a trigger can also be used for other actions, but this depends on the host application.

A different trigger is a transmission trigger. This starts the frame in the TBUF slot, where the trigger points to with TTPTR. If the TBUF slot is marked as empty, then no transmission is started, but the trigger interrupt is set.

It is possible that one host task updates a message slot with a new message if it is necessary by the host application. This could be e.g., a new sensor value. Later then if TTCAN requires this message to be transmitted, it will be activated by the trigger. In other words: one host task needs to be responsible for TTCAN and a different host task may update the message slots. This requires that one message slot is exclusively used for one message (e.g., slot 1 for a temperature sensor). If not enough TBUF slots are available then slots need to be shared by different messages.

The TTCAN host application needs to keep track of the system matrix. If a trigger gets active, then the host application needs to prepare the next transmission column. The host needs to handle also the basic cycles. The reference message includes the cycle count.

Most operations in ISO 11989-4 require single shot transmission. See chap. 21.4.10.1 for further details about handling successful and unsuccessful transmissions.

21.5.3 TTCAN Timing

CAN-CTRL supports ISO 11898-4 level 1. This includes a 16bit timer running at the speed of a CAN bit time (defined by S_PRESC, S_SEG_1, S_SEG_2). An additional prescaler is defined by T_PRESC. If TTEN=1 then the timer is counting continuously.

At the SOF of the message, the timer value is the Sync_Mark. If the message was a reference message, then this value is copied to the Ref_Mark. The cycle time is the timer value minus the Ref_Mark. It is the time that is used as time-stamp for received messages or as trigger time for messages, that need to be transmitted. An overflow protection is included and therefore the cycle time is always strictly monotonic inside a basic cycle.

ISO 11898-4 does not include CAN FD baud rate switching. CAN-CTRL therefore always operates the timer with slow nominal bit rate. The timer is running freely and it is not influenced by synchronization or baud rate switching. Therefore, a timer tick is not synchronous to the start of a CAN bit.

The timer is running in the CAN clock domain while all control and status bits are in the host clock domain. Therefore, the pure timer value is not readable for the host as this would require clock

domain crossing. Trigger events need to be used to synchronize actions required by the host to the timer.

Interrupts because of trigger events, reception or transmission require clock domain crossing and therefore include some clocks delay. This only becomes relevant if after a trigger the host application decides to start a transmission. (Therefore, it is recommended to use a transmission trigger instead.) In all other cases the host application has enough time to prepare all actions for the next trigger event (which is the next time window). Almost all hosts are fast enough to do a lot of actions during the duration of a CAN frame (which is the duration of a time window).

Timing according ISO 11898-4 level 1 is not perfect. The cycle time counts not synchronously with the CAN bits, because a CAN bit can be shortened or lengthened because of CAN synchronization.

Therefore, the cycle time of one node may differ compared to the cycle time of other nodes. In many cases this will be +/- 1 tick but it is not limited to this. The begin of a new basic cycle with the transmission of a reference message will resynchronize the nodes.

If a transmission trigger becomes active, then the appropriate transmission can only be started with the next CAN bit. Therefore, the earliest point of transmission of the SOF of the frame will be at TT_TRIG+1.

21.5.4 TTCAN Trigger Types

The trigger type is defined by TTYPE. TTPTR is a pointer to a TB message slot and TT_TRIG defines the cycle time of the trigger.

Triggers and the related actions shall finish before the maximum cycle time 0xffff is reached as this is the maximum length of a basic cycle.

Except for the immediate trigger, all triggers set TTIF if TTIE is enabled.

If TTTBM=1, only time triggers are supported. Using other triggers in this mode will result in setting TEIF.

If a trigger is activated after a write access to TT_TRIG_1, then write access to TT_CFG_0, TT_CFG_1, TT_TRIG_0 and TT_TRIG_1 are locked until the trigger time is reached (TTIF is set if TTIE is set) or an error is detected (TEIF is set). Therefore, no new trigger can override an active trigger. The write lock is also removed if TTEN=0.

21.5.4.1 Immediate Trigger

An immediate trigger starts the immediate transmission of a frame that is pointed to by TTPTR. TTIF is not set. To start a trigger, TT_TRIG_1 needs to be written. The value that is written, does not care for an immediate trigger.

In TTCAN mode TPE, TSONE, TSALL cannot be used. The immediate trigger is the replacement.

Only the transmission of one frame can be started with an immediate trigger. The host must not command a second immediate trigger before the first transmission has been completed successfully or unsuccessfully.

For an immediate trigger the single shot mode can be selected by TSSS. A transmission can be aborted using TSA. (TPSS and TPA have no meaning.)

If TTPTR of an immediate trigger points to an empty slot, then TEIF is set.

21.5.4.2 Time Trigger

A time trigger just generates an event by setting TTIF and therefore generates an interrupt. No other actions are done.

The time trigger can be used as a reception trigger. If the node expects a message to be received in a time window, then if the message is missing and RIF is not set, a reception trigger can be used to signal this. A reception trigger shall be set after the latest moment when the message is expected to be successfully received.

If TT_TRIG is lower than the actual cycle time, then TEIF is set.

A time trigger can be used if TTTBM=1. This is the only trigger type, that is available in this mode.

21.5.4.3 Single Shot Transmit Trigger

A single shot transmit trigger is intended to be used for exclusive time windows, where a message needs to be transmitted in single shot mode. The selected message is defined by TTPTR.

Single shot mode is automatically used regardless of the state of TSSS. The register bit TSSS is ignored and stays unchanged.

Single shot transmit triggers are intended to be used for exclusive time windows. For this ISO 11898-4 defines a transmit enable window of up to 16 ticks of the cycle time. The register bits TEW(3:0)+1 define the number of ticks. A frame cannot be started if the bus is occupied by another frame. This should not happen in an exclusive time window, but the transmit enable window ensures that there is no delayed start which would result in a violation of the next time window. If the transmit enable window closes and the frame could not be started, then it is aborted. As a result, the TB slot of the frame will be marked as empty and AIF will be set if AIE is set. The data of the frame in the TB slot will not be touched and therefore the slot only needs to be marked as filled again if the same data shall be transmitted in a next attempt.

If TT_TRIG is lower than the actual cycle time, then TEIF is set and no action is done.

21.5.4.4 Transmit Start Trigger

A transmit start trigger is intended to be used for merged arbitrating time windows, where several nodes may transmit messages and CAN arbitration takes place. The selected message is defined by TTPTR.

TSSS defines if retransmission or single shot mode is used.

If the selected frame cannot be transmitted (arbitration loss, several transmissions after an error), then the transmission can be stopped using the transmit stop trigger.

If TT_TRIG is lower than the actual cycle time, then TEIF is set and no action is done.

21.5.4.5 Transmit Stop Trigger

A transmit stop trigger is intended to be used to stop a transmission, that is started with the transmit start trigger.

If a transmission is stopped, then the frame is aborted. Therefore, AIF is set is AIE is set and the TB slot of the frame is marked as empty. The data of the frame in the TB slot will not be touched and therefore the slot only needs to marked as filled again if the same data shall be transmitted in a next attempt.

Chap. 21.4.5 gives further details.

The behavior of a transmit stop trigger is similar to the end of the transmit enable window used by the single shot transmit trigger. If a transmit stop trigger points to an empty slot (which means, that the message has already been transmitted), then not action is done an TEIF is not set.

If TT_TRIG is lower than the actual cycle time, then TEIF is set but the stop is executed.

21.5.5 TTCAN Watch Trigger

In contrast to the generic trigger types explained in chap. 21.5.4 the watch trigger has a dedicated interrupt flag WTIF. If the cycle count equal to the value defined by TT_WTRIG, then WTIF is set is WTIE is set.

The watch trigger is intended to be used if the time since the last valid reference message has been too long. Reference messages can be received in a periodic cycle or after an event. The host application needs to take care of this and has to adjust the watch trigger accordingly.

The default value of WTIE is 1 and therefore the default watch trigger 0xffff is automatically valid. To turn off the watch trigger, WTIE needs to be set to 0.

If TT_WTRIG is updated and it is lower than the actual cycle time, then TEIF is set. The watch trigger can be used if TTTBM=1.

21.6 CiA 603 Time-Stamping

21.6.1 Time-Stamping

CAN in Automation (CiA) defines in specification 603 a method for time-stamping with at least 16 bits, which is supported by CAN-CTRL. It can be used in addition to TTCAN or stand-alone.

The basic concept of CiA 603 is to have a free-running timer which counts clock cycles and not CAN bit times. The precision is 1μs (64bit timer). CAN-CTRL is equipped with an 64bit free-running timer.

After an event a copy of the timer is taken. This is the time-stamp. This time- stamp is stored into RTS or TTS respectively where it can be read by the host. After it is stored, a handshake signal signals the completion of the action. This describes a CDC mechanism with 3 clock domains.

Time-stamps are acquired at the sample point of SOF or the EOF bit where the frame is taken to be valid. This can be selected by the configuration bit TIMEPOS. Seven recessive bits after the ACK delimiter form the EOF of a CAN / CAN FD frame. A frame gets valid for a receiver at the last but one bit of EOF, but for a transmitter at the last bit of EOF.

Software-based time-stamping, which is widely used in many systems, relies on the reception and transmission interrupt. Therefore time-stamping at EOF is recommended.

CiA 603 is supposed to support time-stamping and time-synchronization of AUTOSAR. For AUTOSAR one node in the CAN network is the time master. A time master transmits a synchronization message (SYNC message). The time-stamp of the SYNC message is acquired by the time master and all time slaves. The difference in time between the event of commanding a SYNC message until the time when the SYNC message actually gets transmitted will be transmitted in a follow-up (FUP) message by the time master. Therefore CAN-CTRL supports only one time-stamp for transmitted frames (TTS) but individual time-stamps for all received frames (RTS). Generation of a time-stamp for transmitted frames can be enabled or disabled for each frame individually using bit TTSEN inside a TBUF slot.

CiA defines rules to read out the timer as well as to modify the timer. CAN-CTRL does not include the timer, but uses an external timer. CAN-CTRL only includes the mechanism of time-stamping, the register to store TTS and the memory to store RTS for each frame.

Register bit TIMEEN enables or disables time-stamping. If disabled TTS and RTS are invalid.

21.6.2 The External Timer

An external timer needs to be used for CiA 603 time-stamping. This timer is not included in CAN-CTRL because it may be that such a timer already exists in a system which can be re-used for CiA 603 time- stamping or that a CAN multicore instance (chap. 9) uses one timer for all multicores.

The external timer shall be at least a free running counter, that increments with timer_clk. Such a very simple timer is also included in the testbench. The bit width of the timer is free to chose. CAN-CTRL stores either 32 or 64 bits when a time-stamp is acquired. If the timer has less bits, the

unused bits shall be filled with zeros.

A timer for CiA 603 time-stamping shall have a precision of at least 10µs (16 bit) or 1µs (32 bit or more) and at most 1ns. It is based on a physical second and not on the CAN bit time. The timer shall start from zero and shall counter upward continuously.

CiA 603 defines a prescaler for the timer to compensate different levels of precision from different nodes in a network. As an alternative and replacement for the prescaler for CAN-CTRL is suggested to use a 64 bit timer and do the precision adaption using software if necessary.

CiA 603 defines the option to modify the timer value. If a modification is possible, then it shall take place as fast as possible or immediately in the ideal case. If the timer is not in the same clock domain as the host controller, then a modification may take some time, but such a delay shall be minimized.

CiA 603 requires the ability to read the timer at any time.

In summary such a timer is a free-running counter, which can be read and written by the host while such an access may include some delays because of clock domain crossing. Such a timer together with the CIiA 603 time-stamping provided by CAN-CTRL support time synchronization as defined by AUTOSAR.

21.6.3 Timer Bit Width

CiA 603 time-stamping focuses on support for AUTOSAR. AUTOSAR handles time synchronization and therefore time-stamps have to be converted into real time. With the recommendation, that the precision shall be at least 10µs (16 bit) or 1µs (32 bit or more) a bit width of 32 bits is sufficient for good AUTOSAR support. It has to be noted, that the conversion into real time results in some computational effort, that has to be done by the host processor, but this is part of the AUTOSAR concept.

Using a bit width of 64 bits can be preferable in custom applications where the time-stamp can be used immediately as real time without conversion. 64 bits are enough for many thousand years without overflow. Such custom applications need to take care of the fact that all nodes in a network need to count with the same <timer_clk> frequency or otherwise the clock rate ratios need to be considered if time-stamps of other nodes are used for calculations.

21.7 CAN Bit Time

21.7.1 Data Bit Rates

CAN 2.0B defines data bit rates up to 1Mbit/s. For CAN FD there is no fixed limitation. For real life systems the data rates are limited by the used transceiver and the achievable clock frequency of the CAN-CTRL core.

The CAN-CTRL core can be programmed to arbitrarily chosen data rates only limited by the

range of the bit settings in the appropriate bit timing and prescaler registers

21.7.2 Definitions

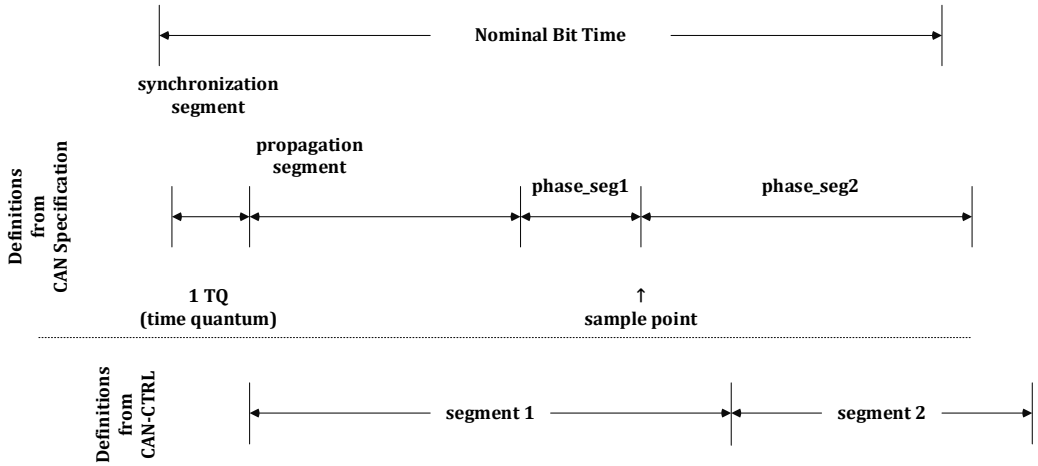


Figure 21-13 CAN Bit Timing Specifications

The CAN bit time *BT* consists of several segments as shown in Figure 21-13. Each segment consists of a number of time quanta units n_{TQ}. The duration of a time quanta TQ is:

$$TQ = \frac{n_{prescaler}}{f_{clock}}$$

The values of n_{TQ} and $n_{prescaler}$ have to be chosen depending on the system clock frequency match BT_{real} as close as possible to $BT_{ideal} = 1/BR$ where *BR* is the CAN bus baud rate:

$$BT_{ideal} \approx BT_{real} = \frac{n_{prescaler} \cdot n_{TQ}}{f_{clock}} = t_{Seq\_1} + t_{Seg\_2}$$

The CAN specification requires several relationships between the segment lengths (Table 21-45) which results in relationships between t_{Seg_1}, t_{Seg_2} and the maximum synchronization jump width t_{SJW}. Please note that lists the minimum configuration ranges defined by the CAN/CAN FD specification.

Table 21-45 CAN Timing Segments (Minimum Configuration Ranges)

Segment	Description		
SYNC_SEG	Synchronization Segment = 1 TQ		
PROP_SEG	Propagation Segment		
	[1...8] TQ	CAN 2.0 bit rate	CAN FD not enabled
	[1...48] TQ	CAN 2.0 bit rate	CAN FD enabled
	[1…48] TQ	CAN FD nominal bit rate	
	[0...8] TQ	CAN FD data bit rate	
PHASE_SEG1	Phase Buffer Segment 1		
	[1...8] TQ	CAN 2.0 bit rate	CAN FD not enabled

	[1...16] TQ	CAN 2.0 bit rate	CAN FD enabled
	[1...16] TQ	CAN FD nominal bit rate	
	[1...8] TQ	CAN FD data bit rate	
PHASE_SEG2	Phase Buffer Segment 2		
	[2...8] TQ	CAN 2.0 bit rate	CAN FD not enabled
	[2...16] TQ	CAN 2.0 bit rate	CAN FD not enabled
	[2...16] TQ CAN FD nominal bit rate		
	[2...8] TQ	CAN FD data bit rate	
SJW	Synchronization Jump Width		
	[1...4] TQ	CAN 2.0 bit rate	CAN FD not enabled
	[1...16] TQ	CAN 2.0 bit rate	CAN FD not enabled
	[1...16] TQ CAN FD nominal bit rate		
	[1...8] TQ	CAN FD data bit rate	
IPT	Information Processing Time = [0...2] TQ PHASE_SEG2 ≥ IPT		

As can be seen in Figure 21-13 the CAN-CTRL core collects SYNC_SEG, PROP_SEG and PHASE_SEG1 into one group and the length of the group is configurable with t_{Seg_1}. Table 21-46 lists the available configuration ranges. Please note that the CAN-CTRL core does not check if all rules are met and offers a wider configuration range than defined by the CAN / CAN FD specification.

Table 21-46 CAN-CTRL Timing Settings (Available Configuration ranges)

Setting	Requirements				
t_{Seg_1}	[2...65] TQ		CAN	2.0 bit rate	(slow)
	[2...65] TQ		CAN	FD nominal bit rate rate	(slow)
	[2...17] TQ		CAN	FD data bit	(fast)
t_{Seg_2}	[1...8] TQ	$t_{Seg\_1} \geq t_{Seg\_2}+2$	CAN	2.0 bit rate	(slow)
	[1...32] TQ	$t_{Seg\_1} \geq t_{Seg\_2}+2$	CAN	FD nominal bit rate rate	(slow)
	[1...8] TQ	$t_{Seg\_1} \geq t_{Seg\_2}+1$	CAN	FD data bit	(fast)
t_{SWJ}	[1...16] TQ	$t_{Seg\_2} \geq t_{SWJ}$	CAN	2.0 bit rate	(slow)
	[1...16] TQ	$t_{Seg\_2} \geq t_{SWJ}$	CAN	FD nominal bit rate rate	(slow)
	[1...8] TQ	$t_{Seg\_2} \geq t_{SWJ}$	CAN	FD data bit	(fast)

For CAN 2.0 bit rate and CAN FD (slow) nominal bit rate the register settings S_Seg_1, S_Seg_2, S_SJW and S_PRESC define the appropriate segment lengths. For CAN FD (fast) data bit rate the register F_Seg_1, F_Seg_2, F_SJW and F_PRESC are valid.

$$t_{Seg_1} = (S_{Seg_1} + 2) \cdot TQ \qquad t_{Seg_1} = (F_{Seg_1} + 2) \cdot TQ$$
$$t_{Seg_2} = (S_{Seg_2} + 1) \cdot TQ \qquad t_{Seg_2} = (F_{Seg_2} + 2) \cdot TQ$$
$$t_{SJW} = (S_{SWJ} + 1) \cdot TQ \qquad t_{SJW} = (F_{SJW} + 2) \cdot TQ$$
$$n_{prescaler} = (S\_PRESC + 1) \qquad n_{prescaler} = (F\_PRESC + 1)$$

A CAN FD core switches from slow nominal bit rate to fast data bit rate at the sample point in the BRS bit and switches back at the sample point in the CRC delimiter bit.

An illustration a suitable bit rate configuration is given in Figure 21-14 where $f_{tq\_clk} = \frac{f_{clock}}{n_{prescaler}}$

he bit time BT_{real} , the sample point and the synchronization jump width SJW will be derived

from tq_clk.

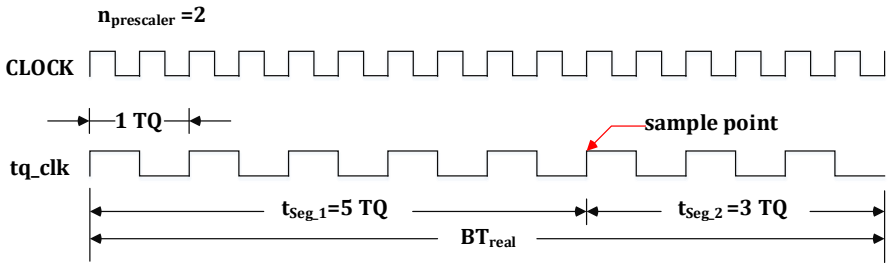


Figure 21-14 Clock Division for Bit Sampling

Having the requirements from Table 21-46 in mind the host controller has to define the length of segment 1, segment 2 and the synchronization jump width for the slow bit rate and if required for CAN FD for thefast bit rate.

Some suggestions, given as rules of the thumb:

- Segment 1 must be slightly larger than segment 2. Then the sample point is in a little bitlater than in the middle of the bit time.
- The synchronization jump width must not be bigger than segment 2. If SJW is too small the CAN node may be too slow to resynchronize, if SJW is too big then the CAN node may resynchronize too often. SJW being half as long as segment 2 seems to be a suitable value.
- All CAN nodes connected to a CAN bus should choose similar settings if possible.

The fastest CAN bus speed can be configured using the minimum timing values. But there are some things that need to be considered:

If the prescalers are bigger than one then all other timing parameters could be set to zero, but this violates the rule $t_{Seg\_1} \geq t_{Seg\_2}+2$ (slow speed) resp. $t_{Seg\_1} \geq t_{Seg\_2}+1$ (fast speed) and therefore S_Seg_1 and F_Seg_1 should be at least set to 1. But such a selecting of timing parameters works in theory, but may not be robust enough for real nets.

If the prescalers are set to one then synchronization becomes difficult. In general, the CAN specification requires one bit time to be at least 8 TQ long for CAN 2.0 and CAN FD slow nominal bit rate. For CAN-CTRL one bit time of fast data bit rate needs to be also at least 8TQ long if the fast prescaler is set to 1.

- In summary the fastest CAN bus frequency is the can_clk frequency divided by 8: prescalersset to 1 and 8 TQ bit time.

21.7.3 Example Configuration

This example refers to Figure 21-14. It is an example for CAN 2.0 / slow bit rate configuration but all statements can be easily translated to CAN FD (fast) data bit rate. The following steps need to be carried for the configuration of the CAN-CTRL core:

1. Set bit RESET=1.
2. Set registers S_Seg_1 and S_Seg_2:

In the example the data rate on the bus f_{bus}=1Mbaud and the system clock is 16MHz.

The values of n_{TQ} and $n_{prescaler}$ have to be selected to fit BT_{real} as close as possible to BT_{ideal}

In this example $n_{prescaler}$ =2 and n_{TQ} =8 are chosen which results in a perfect match:

BT_{ideal} = BT_{real} = 8TQ.

With BT_{real} =t_{Seg_1} + t_{Segt_2} and the time segment definitions given in chapter 21.7.1 t_{Seg_1} = 5TQ

and t_{Seg_2} = 5TQ can be chosen as suitable values which finally results in the register settings S_Seg_1=3 and S_Seg_2=2.

3. Load the acceptance code and mask registers (optional).

4. Set register S_SJW:
   With $t_{Seg\_2} \geq t_{SJW}$ one is free to chose t_{SJW} = 3which finally results in 2.

5. Load the clock prescaler register S_PRESC: $n_{prescaler}$ = PRESC+1 results in S_PRESC=1.

6. Set bit RESET=0.

The given order is not mandatory. It is merely necessary to set bit RESET=1 at the beginning, as it is otherwise not possible to load the bit timing, ACODE and AMASK registers. RESET=0 is required upon completion of the configuration. The controller then waits for 8 recessive bits (frame end) and resumes its normal operation.

7. Continue with configuration of interrupts other configuration bits and execute commands.

21.7.4 Bit Rate Switching and the Sample Point

In a CAN network when CAN FD frames with bit rate switching are used the exact position of the sample point is important and needs to be for all nodes as similar as possible.

The bit rate is switched after the sample point of the BRS bit and the sample point of the CRC delimiter. Therefore, the lengths of these bits are intermediate. As can be seen in Figure 6-3 the position of the sample point makes a big difference for the absolute length of the BRS bit. (1 TQ of the slow nominal bit rate may be much bigger than several TQs of the fast data rate.) If the data rate is much higher than the nominal bit rate then an earlier or later sample point may lead to false samples at fast data bit rate.

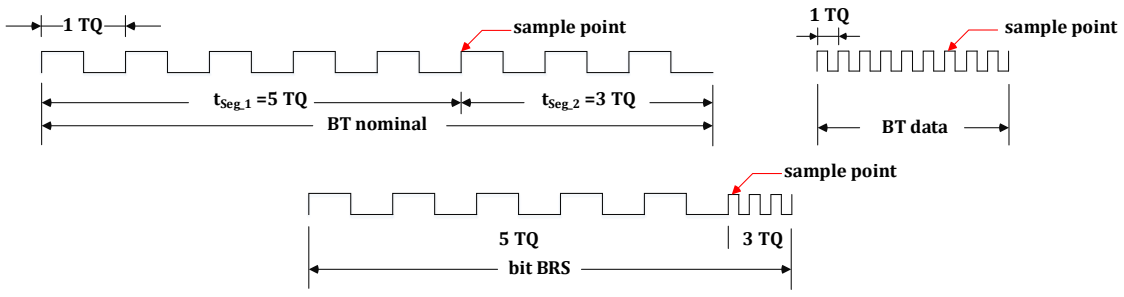


Figure 21-15 Bit Rate Switching at bit BRS *S_PRESC≠F_PRESC*

21.7.5 Bit Timing Configuration for CAN FD Nodes

As stated in chap. 21.7.4 the position of the sample point is very important for bit rate switching. Therefore, there is the general recommendation for all CAN FD nodes to use exactly the same timing parameters in a CAN network. In contrast to classic CAN where only the data rate needs to be the same for all nodes, CAN FD requires the sample point at the same position. In others words segment 1 and segment 2 need to be configured to the same length for all nodes. Furthermore, it is recommended to use also the same length of one TQ to make synchronization for all nodes equal.

This requirement can only be achieved if all nodes run with the same base clock frequency (called can_clk for CAN-CTRL) or at least with compatible clock frequencies. The recommended settings are 20MHz, 40MHz or 80MHz.

21.7.6 TDC and RDC

For CAN FD nodes transmitter delay compensation (TDC) can be optionally enabled while receiver delay compensation (RDC) is automatically active. These features have the following background: For communication with CAN FD data bit rate, it may be that the delay of the transmitting transceiver or the delay of the bus is bigger than a bit time. This needs to be compensated. Without TDC, the bit rate in the data-phase of FD frames is limited by the fact that the transmitter detects a bit error if it cannot receive its own transmitted bit latest at the sample point of that bit.

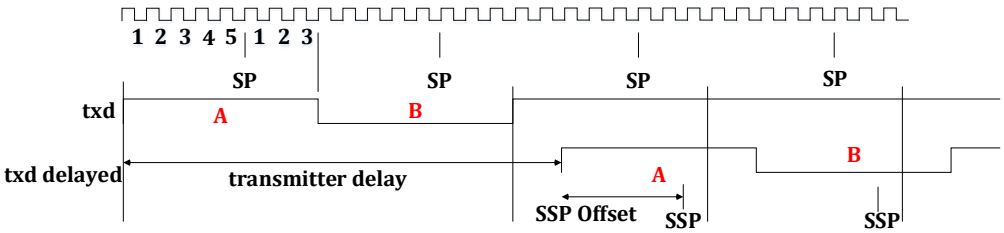


Figure 21-16 Transmitter Delay

Figure 21-16 gives an example of the effect of a big transmitter delay. In this figure a txd data stream is shown starting with bits A and B. One bit time consists of t_{Seg_1} =5 TQ before the Sample Point (SP) and t_{Segt_2} = 3 TQ behind. In this example the transmitter delay is longer than 2bit times. Therefore, the original SP cannot be used to sample the correct bit value and the CAN FD specification defines an additional Secondary Sample Point (SSP). If TDC is enabled with bit TDCEN=1 then CAN-CTRL automatically determines the transmitter delay. The position of the SSP is the transmitter delay plus the SSP Offset which is defined by the configuration bits SSPOFF. SSPOFF must be given as a number of TQ and it is suggested to set t_{Seg_1} equal to SSPOFF. (Please remember that F_SEG_1 defines t_{Seg_1} during data bit rate. Therefore, in the example given in Figure 21-16 SSPOFF=5 is chosen, because F_SEG_1=3.)

ISO 11898-1:2015 requires to use only F_PRESC=0 or 1 if that TDC is used. With this requirement CAN- CTRL is capable of automatically determining a transmitter delay of up to 3bit times of the fast data bit rate.

A delay bigger than a bit time needs to be taken into account also for receiving nodes. CAN FD defines an additional hard synchronization at the falling edge between bit FDF =1 and the following r0 bit. (At this time the slow nominal bit rate is active.) Synchronization for CAN and CAN FD is in general defined to operate in time steps of one TQ. But steps of one TQ at nominal bit rate may be too coarse for a synchronization for the fast data bit rate. Therefore, the additional hard synchronization at the FDF bit needs to violate this rule and needs to synchronize as exact as possible and only limited by the clock frequency of the system. CAN-CTRL uses this special synchronization and this is called RDC. RDC is automatically done during reception if FDF =1, no matter if TDC is enabled or not.

21.7.7 Bit Timing Recommendations

CAN FD timing configuration requires to use the same data rate and the same sample point for all nodes in a CAN network. Therefore, it is recommended to use 20MHz, 40MHz or 80MHz as source for the CAN protocol machine (can_clk).

Please note that the bit time should be at least 8TQ if the prescaler is set to 1 for a stable communication. Lower settings are possible, but may cause problems.

The decision if TDC is used or not depends on the CAN network. The tables only give a suggestion and TDC can be enabled or disabled as needed.

Table 21-47 Abbreviations about CAN bit timing

Abbreviation	Description
PSP	Primary Sample Point
SSP	Secondary Sample Point
Seg 1	Segment 1
Seg 2	Segment 2
TDC	Transmitter Delay Compensation (see SSPOFF)

Table 21-48 Recommendations for 20MHz can_clk

Bit Rate [Mbit/s]	PSP [%]	SSP [%]	Prescaler	Bit Time [TQ]	Seg 1 [TQ]	Seg 2 [TQ]	SJW [TQ]	TDC[clk]
0.25(arbitration)	80	-	1	80	64	16	16	-
0.5(arbitration)	80	-	1	40	32	8	8	-
0.5	80	(disable)	1	40	32	8	8	-
0.833	79	(disable)	1	24	19	5	5	-
1	80	80	1	20	16	4	4	16
1.538	77	77	1	13	10	3	3	10

2	80	80	1	10	8	2	2	8
4	80	80	1	5	4	1	1	4
5	75	75	1	4	3	1	1	3

Table 21-49 Recommendations for 40MHz can_clk

Bit Rate [Mbit/s]	PSP [%]	SSP [%]	Prescaler	Bit Time [TQ]	Seg 1 [TQ]	Seg 2 [TQ]	SJW [TQ]	TDC[clk]
0.25(arbitration)	80	-	2	80	64	16	16	-
0.5(arbitration)	80	-	1	80	64	16	16	-
0.5	80	(disable)	2	40	32	8	8	-
0.833	79	(disable)	2	24	19	5	5	-
1	80	80	1	40	32	8	8	32
1.538	77	77	1	26	20	6	6	20
2	80	80	1	20	16	4	4	16
3.077	77	77	1	13	10	3	3	10
4	80	80	1	10	8	2	2	8
5	75	75	1	8	6	2	2	6
6.667	83	83	1	6	5	1	1	5
8	80	80	1	5	4	1	1	4
10	75	75	1	4	3	1	1	3

Table 21-50 Recommendations for 80MHz can_clk

Bit Rate [Mbit/s]	PSP [%]	SSP [%]	Prescaler	Bit Time [TQ]	Seg 1 [TQ]	Seg 2 [TQ]	SJW [TQ]	TDC[clk]
0.25(arbitration)	80	-	4	80	64	16	16	-
0.5(arbitration)	80	-	2	80	64	16	16	-
0.5	80	(disable)	4	40	32	8	8	-
0.833	79	(disable)	4	24	19	5	5	-
1	80	80	2	40	32	8	8	64
1.538	77	77	2	26	20	6	6	40
2	80	80	2	20	16	4	4	32
3.077	77	77	2	13	10	3	3	20
4	80	80	1	20	16	4	4	16
5	75	75	1	16	12	4	4	12
6.667	83	83	1	12	10	2	2	10
8	80	80	1	10	8	2	2	8
10	75	75	1	8	6	2	2	6

22 Window Watchdog (WWDG)

22.1 Overview

Window watchdog (WWDG) is used to detect software anomalies due to external interference or unforeseen logic. When an anomaly occurs, the software does not normally execute in sequence.

WWDG uses the frequency division clock of high-speed main system clock for working, generally the frequency division clock of PLL clock, which has high clock accuracy and is applicable to the scenarios that have accurate requirements on the watchdog timeout period.

WWDG contains a 7bit count-down counter WWDG_CR.T[6:0]. When the highest bit T[6] of the counter is cleared, the watchdog will generate full-chip reset; if the watchdog counter WWDG_CFG.W[6:0] is refreshed by the software before counting to the window value WWDG_CFG.W[6:0], it will also generate a reset. That is, the watchdog must be refreshed and reset within the valid window.

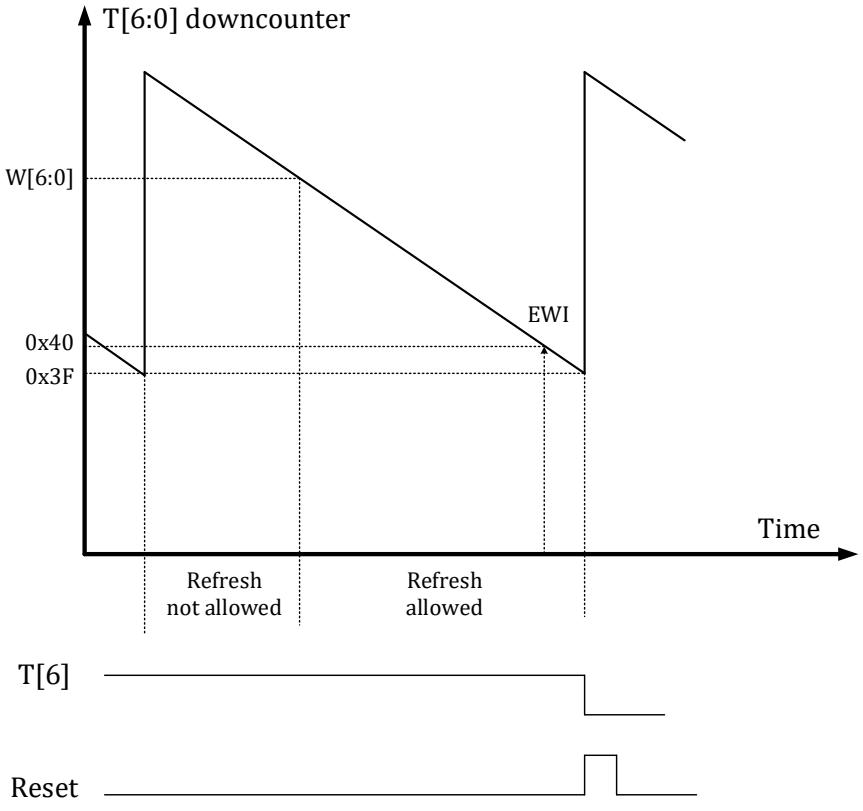


Figure 22-1 WWDG Counting and Refresh Mechanism

Programmable independent count-down counter

Controllable reset

A reset is generated when the watchdog count value is less than 0x40 (if the watchdog can be enabled)

A reset is generated when the watchdog count value is refreshed before counting to the window value (if the watchdog can be enabled)

Early Wakeup Interrupt: EWI. An early wakeup interrupt will be generated when EWI bit and the watchdog are enabled and the counter counts to 0x40.

In some applications, early wakeup interrupt can be used for software system inspection or system recovery without generating a reset. In these applications, the watchdog count value can be refreshed and reset in the interrupt handler function of the early wakeup interrupt.

If early wakeup interrupt cannot get a timely response, the watchdog counter will decrease counting to 0x3F to generate a watchdog reset/

The watchdog timeout reset time is calculated according to the following formula

$$Timeout_{WWDG}=t_{MCLK}\times 32768\times 2^{WDGTB}\times(T[5:0]+1)$$

Where the $Timeout_{WWDG}$ is the watchdog timeout period,

and t_{MCLK} is the main system clock

32768 is the frequency division coefficient of internal fixed clock of watchdog

WDGTB is the pre-frequency division coefficient of watchdog count clock

T[5:0] is the lower 6 bits of watchdog counter load value.

As the watchdog generally uses the frequency division clock of the high-speed clock for counting, there are 32768×2WDGTB high-speed clock cycles counting from 0x40 to 0x3F, which is enough for the software to reset the watchdog.

Table 22-1 WWDG Timeout Period (related to the main system clock)

WDGTB	Prescaler(2^{WDGTB})	Minimum Timeout Period	Maximum Timeout Period
0	1	0.17ms	10.92ms
1	2	0.34ms	21.84ms
2	4	0.68ms	43.69ms
3	8	1.36ms	87.38ms

In debugging mode, when the CPU pauses, the system can control whether to continue WWDG counting according to the SYS_DBG_CFG.DBG_WWDG_STOP bit.

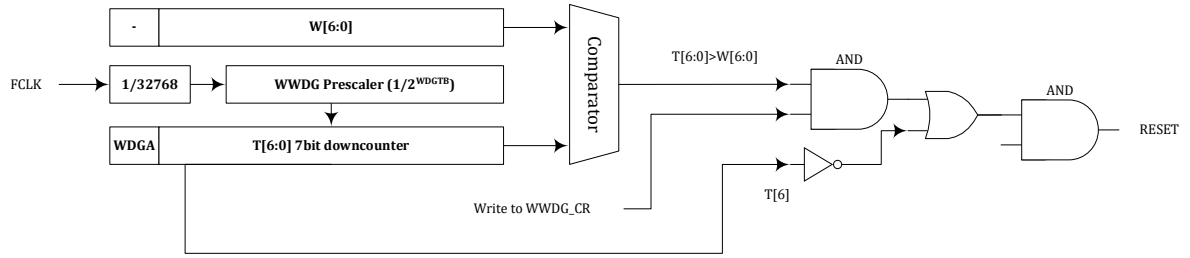


Figure 22-2 WWDG Register List

The application software needs to write WWDG_CR at the right time to refresh the watchdog counter to prevent MCU reset. This refresh operation needs to be performed when the count value T[6:0] < window register value W[6:0]; otherwise writing WWDG_CR outside the window will also generate a reset action. T[6] is forced to write 1 each time, so the range of data that can be written by WWDG_CR.T[6:0] is 0xFF~0xC.

22.2 Register

22.2.1 Address Allocation

The base address of WWDG is 0x4001B400.

Table 22-2 WWDG Register List

Name	Offset Address	Description
WWDG_CR	0x00	WWDG control register
WWDG_CFG	0x04	WWDG configuration register
WWDG_IF	0x08	WWDG interrupt flag register

22.2.2 WWDG Control Register (WWDG_CR)

Address: 0x4001_B400

Reset value: 0x0

Table 22-3WWDG Control Register (WWDG_CR)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								WDG_EN	T						
								RW	RW						
								0	0x7F						

Bit field	Bit Name	Description

[31:8]		Unused
[7]	WDG_EN	WWDG enable, 1: Enable; 0: Disable. It is disabled by default when powering on. The software can write 1 to enable the watchdog, and write 0 and 0x3C to [15:8] to disable the watchdog. After being disabled, no reset signal will be generated, but it can still count and generate an early interrupt
[6:0]	T	WWDG counter. Use the internal frequency division clock of the watchdog for count-down counting, and a reset signal is generated when the count value is less than 0x40(T[6] is 0). To avoid immediate reset when writing 0 to T[6], T[6] is forced to set 1 when writing, and the software can only write T[5:0].

22.2.3 WWDG Configuration Register (WWDG_CFG)

Address: 0x4001_B404

Reset value: 0x0

Table 22-4 WWDG Configuration Register (WWDG_CFG)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						WDGTB		EWI	W						
						RW		RS	RW						
						0		0	0x7F						

Bit field	Bit Name	Description
[31:10]		Unused
[9:8]	WDGTB	Pre-frequency division coefficient of watchdog count clock 2'h0: Main system clock/32768/1 2'h1: Main system clock/32768/2 2'h2: Main system clock/32768/4 2'h3: Main system clock/32768/8
[7]	EWI	Early interrupt enable, 1: Enable; 0: Disable. It is disabled by default when powering on. Writing 1 can enable the interrupt, and writing 0 is invalid. This bit can only be reset by hardware or cleared by power off after setting to 1. An early interrupt will be generated when the watchdog counts to 0x40.
[6:0]	W	WWDG feed window. A reset signal is generated when T>W, that is, the valid reset window is W>=T>0x40

22.2.4 WWDG Interrupt Flag Register (WWDG_IF)

Address: 0x4001_B408

Reset value: 0x0

Table 22-5 WWDG Interrupt Flag Register (WWDG_IF)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															EWIF
															RW1C
															0

Bit field	Bit Name	Description
[31:2]		Unused
[0]	EWIF	WWDG early interrupt flag. EWIF is set to 1 when the counter counts to 0x40, writing 1 will clear, and 0 will be invalid. Even when the watchdog is not enabled, EWIF still can be set, because the watchdog counter T[6:0] still counts when the watchdog is disabled. Only when both WDGA and EWI are set to 1, the watchdog will raise an interrupt request to the CPU.

23 Independent Watchdog (IWDG)

23.1 Overview

Independent watchdog (IWDG) uses 32kHz low-speed RC (LRC) clock for working, which can ensure normal operation even when the main clock of the system fails. Since the LRC oscillator is not as accurate as the system high-speed clock, the independent watchdog can be used for application scenarios that do not require high accuracy of the timeout period.

IWDG contains a 21-bit count-down counter internally. IWDG can be configured to generate a wake source for system deep sleep (clock off or power off) or a full-chip reset signal in the event of a timeout. When MCU enters deep sleep mode, the system clocks, including PLL/HRC, are turned off, while the LRC oscillator is always operating, so the IWDG can continue to work normally; LRC and IWDG work normally even when the system enters Standby mode and most circuits are in power-off state.

Generally, the timer wake-up threshold of the watchdog is less than the timeout reset threshold, but greater than 0x8, that is, IWDG starts to count down from IWDG_RTH after reloading, and a wake-up signal is generated when the count reaches IWDG_WTH. If IWDG is used for timing wake-up of deep sleep, refresh and reset the IWDG before entering sleep; IWDG starts counting down from IWDG_RTH, and generates an IWDG wake-up signal after a relatively accurate time of dormancy. If IWDG timing wake-up of deep sleep is not enabled, the chip can also be released from deep sleep when IWDG generates a full chip reset, but all MCU register configurations will be reset at the same time.

IWDG timeout reset time is calculated according to the following formula

$Timeout_{IWDG}=t_{LRC}\times(IWDG\_RTH-7)$

IWDG wake-up time is calculated according to the following formula

$Wakeup_{IWDG}=t_{LRC}\times(IWDG\_RTH-IWDG\_WTH)$

$Timeout_{IWDG}$ is the timeout reset time of IWDG, and $Wakeup_{IWDG}$ is the wake-up time of IWDG, t_{LRC} is the LRC oscillator cycle, 1/32kHz=31.25uS, IWDG_RTH is the time-out reset value of IWDG, and IWDG_WTH is the wake-up threshold of IWDG.

IWDG_RTH=0x001000, the corresponding minimum reset interval of IWDG is 4096/32kHz = 0.128s.

IWDG_RTH=0x1FF000, the corresponding maximum reset interval of IWDG is 511×4096/32kHz = 65.408s.

About IWDG feed dog cross-clock synchronization issues:

Since IWDG counters work in the LRC clock domain, the software runs in the system master clock domain, which is usually a PLL clock. When the software is fed to the dog by writing to the IWDG RTH or writing to the IWDG CLR, the watchdog counter is reset from the time the software writes to the watchdog, requiring up to 3 LRC clock cycles.

Therefore, if the software needs to read the IWDG CNT after feeding the dog, it needs to delay at least 3 LRC clock cycles to read the IWDG CNT and find that the counter has been reset.

Software write modify IWDG RTH/IWDG WTH/IWDG CFG/IWDG PSW and other registers, write immediately take effect without waiting. Only the reset of the IWDG CNT takes effect after a delay.

23.2 Register

23.2.1 Address Allocation

The base address of IWDG is 0x4001B8C0.

Table 23-1 IWDG Register

Name	Offset	Description
IWDG_PSW	0x00	IWDG password register
IWDG_CFG	0x04	IWDG configuration register
IWDG_CLR	0x08	IWDG clear register
IWDG_WTH	0x0C	IWDG counter timing wake-up threshold register
IWDG_RTH	0x10	IWDG counter timeout reset threshold register
IWDG_CNT	0x14	IWDG current count register

23.2.2 IWDG Password Register (IWDG_PSW)

Address: 0x4001_B8C0

Reset value: 0x0

Table 23-2 IWDG Password Register (IWDG_PSW)

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

Bit field	Bit Name	Description
[31:16]		Unused
[15:0]	PSW	Write 0xA6B4 to write data to IWDG_CLR/IWDG_RTH. Writing data to IWDG_CLR or IWDG_RTH will clear the password. Therefore, write the password again before writing data to the of the watchdog.

23.2.3 IWDG Configuration Register (IWDG_CFG)

Address: 0x4001_B8C4

Reset value: 0x0

Table 23-3 IWDG Configuration Register (IWDG_CFG)

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

Bit field	Bit Name	Description
[31:16]		Unused
[4]	DWK_EN	Deep sleep timing wake-up enable. 0: Disable; 1: Enable
[3:1]		Unused
[0]	WDG_EN	IWDG enable. 0: Disable; 1: Enable This function is enabled by default. Write 1 to set it, and write 0 and 0x3C to [15:8] at the same time to clear it. When the IWDG is disabled, no reset signal will be generated, but it can still count and generate a timing wake-up signal.

Writing to IWDG_CFG does not need to write a password to IWDG_PSW.

23.2.4 IWDG Clear Register (IWDG_CLR)

Address: 0x4001_B8C8

Reset value: 0x0

Table 23-4 IWDG Clear Register (IWDG_CLR)

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

Bit field	Permission	Description
[31:16]	NA	Unused
[15:0]	WO	Write byte 16'b0111_1001_1000_110B_0, and the upper 15 bits are the password. When the password is correct, B[0] can be written Write 1 to B[0] to reset the WDT counter to TH, and the bit is automatically cleared after writing. Writing 0 is invalid

Writing to IWDG_CLR needs to write a password to IWDG_PSW first.

23.2.5 IWDG Counter Timing Wake-up Threshold Register (IWDG_WTH)

Address: 0x4001_B8CC

Reset value: 0x001F_F000

Table 23-5 IWDG Counter Timing Wake-up Threshold Register (IWDG_WTH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
											IWDG_WTH				
											RW				
											0x1F				

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

Bit field	Bit Name	Description
[31:21]		Unused
[20:12]	WTH	Timing wake-up threshold of IWDG. The watchdog uses 32kHz LRC oscillator to count down from IWDG_RTH and generate a wake-up signal when the value reaches IWDG_WTH.
[11:0]		The value is always 0

Writing to IWDG_WTH does not need to write a password to IWDG_PSW.

23.2.6 IWDG Counter Timeout Reset Threshold Register (IWDG_RTH)

Address: 0x4001_B8D0

Reset value: 0x001F_F000

Table 23-6 IWDG Counter Timeout Reset Threshold Register (IWDG_RTH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
											IWDG_RTH				
											RW				
											0x1F				

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

Bit field	Bit Name	Description
[31:21]		Unused
[20:12]	RTH	The IWDG timeout reset threshold is also the reload value. The watchdog uses 32kHz LRC oscillator to count from IWDG_RTH and generate a reset when the value reaches IWDG 0x7. Writing 0x0 to the register will be forcibly overwritten by the hardware to 0x1000. The IWDG_RTH register can be overwritten after writing the correct password to the IWDG_PSW. Overwriting IWDG_RTH also resets the watchdog counter, which starts

		with the new IWDG_RTH.
[11:0]		The value is always 0

To prevent IWDG_RTH from being written to 0, when the software writes all zeros, the hardware will forcibly rewrite it to 0x1000, and the lower 12 bits of the register are always 0.

23.2.7 IWDG Current Count Register (IWDG_CNT)

Address: 0x4001_B8D4

Reset value: 0x0000_0000

Table 23-7 IWDG Current Count Register (IWDG_CNT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
											IWDG_CNT				
											RO				
											0x00				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IWDG_CNT															
RO															
0x0000															

Bit field	Bit Name	Description
[31:21]		Unused
[20:0]	CNT	Current count of the watchdog, which is ≤IWDG_RTH.

24 Power Management Module (PMU)

The chip can reduce power consumption by reducing the running clock frequency and turning off part of the circuit clock or circuit power.

24.1 Peripheral Clock Gating

The peripheral clock is divided by the system high-speed clock MCLK; it can be close turned off by setting the SYS_CLK_FEN register gating when the peripheral is not in use. Each peripheral clock has a clock gating. Please refer to 6.3.8 for details. **The gated clock is turned off by default after power-on, and should be turned on by software before using the corresponding peripheral module.**

24.2 Peripheral Clock Divider

Some peripherals have independent clock dividers, which enables it to work with an appropriate frequency.

Among them, SYS_CLK_DIV0 is the division factor of I2C, while SYS_CLK_DIV2 is the division factor of UART0/1. Please refer to 6.3.3 and 6.3.6 for details. Besides, the UART baud rate has an additional clock divider inside the UART module.

24.3 Low Power Mode

Table 24-1 Summary of Low Power Modes

<table>
<tr><th>Mode</th><th>Enter Mode</th><th>Exit Mode</th><th>PLL/HSI/HSE</th><th>LSI</th><th>Core power on</th></tr>
<tr><td>Sleep
Sleep</td><td>WFI/WFE</td><td>Any interrupt
Debug
External reset
IWDG reset</td><td>PLL/HSI/HSE On, CPU clock Off, NVIC/peripheral clock On</td><td rowspan="3">On</td><td rowspan="2">On</td></tr>
<tr><td>Deep Sleep
Deep Sleep</td><td>SLEEPDEEP + WFI/WFE</td><td>IWDG timing wake-up
IO wake-up
Debug
External reset
IWDG reset</td><td rowspan="2">PLL/HSI/HSE Off, CPU/peripheral clock Off</td></tr>
<tr><td>Stop (core power off)
Standby</td><td>PWR_CFG.PG + SLEEPDEEP + WFI/WFE</td><td>IWDG timing wake-up
IO wake-up
External reset
IWDG reset</td><td>Most circuits are powered off, except power management, reset management,</td></tr>
</table>

					backing memory and IWDG.

24.4 Sleep Mode

In sleep mode, CPU clock is turned off, but NVIC module continues to work, and all peripherals and IOs work normally.

The sleep mode has no effect on the power supply of the circuit, and all register states and memory data maintain normal power supply during sleep.

You can refer to the sleep routine to turn off the clock of each module in the digital part and turn off the analog ADC/OPA/CMP/DAC module before entering sleep.

24.4.1 Enter Mode

The sleep mode can be entered by executing the WFI/WFE command. There are two different ways to enter sleep mode, depending on the SLEEPONEXIT bit setting of the CPU core.

Sleep-now: If SLEEPONEXIT is 0, CPU enters the sleep mode immediately after executing the WFI/WFE command.

Sleep-on-exit: If SLEEPONEXIT is 1, CPU enters the sleep mode after completing all interruptions.

24.4.2 Exit Mode

If sleep mode is entered by using WFI, any interrupt can wake up CPU.

If sleep mode is entered by using WFE, the peripheral interrupt flag or IO wake-up event can be used as a wake-up event. When using the peripheral interrupt flag as a wake-up event, the peripheral raises an interrupt signal to the CPU, but the corresponding interrupt is not enabled in the NVIC, and SEVONPEND needs to be set to 1. After waking up, the peripheral interrupt flag and NVIC interrupt pending bit need to be cleared by software.

Waking up from sleep in this manner results in the shortest wake time because there is no interrupted entry and exit involved.

Debug operation can wake up the chip from the sleep mode.

24.5 Deep Sleep Mode

The deep sleep mode turns off all system high-speed clocks, including PLL/HSI/HSE, but the 32kHz RC oscillator LSI still works normally. At the same time, LDO enters the low power mode, and BGP is turned off.

The deep sleep mode can further reduce power consumption compared with the sleep mode.

The deep sleep mode has no effect on the power supply of the circuit, and all register states and memory data maintain normal power supply during deep sleep.

If a peripheral needs to use a high-speed clock to complete an operation in progress, such as a flash erase write, deep sleep delays entry until the operation is complete.

24.5.1 Enter Mode

Set the System Control Register SLEEPDEEP of CPU core to 1, and execute the WFI/WFE command to enter deep sleep.

24.5.2 Exit Mode

IO wake-up or IWDG timing wake-up signal can wake the chip from deep sleep.

External reset or IWDG reset can reset all chips and release the deep sleep state.

Debug operation can wake up the chip from the deep sleep mode.

24.6 Standby Mode

Standby mode can minimize the chip power consumption. This mode further powers off most of the digital circuits in addition to the deep sleep mode. However, after entering standby mode, the LSI oscillator, PMU, RMU and IWDG can still work normally, and the BRAM will not power off, which can still provide 128Bytes storage space to keep the key system data before entering the standby mode.

Besides, IO wake-up enable polarity selection can ensure that the system reset, sleep and wake-up event record register is not powered off.

All IOs are in high resistance state in the standby mode, and the IO wake-up signal is sent directly from the PAD to the chip's internal wake-up logic.

24.6.1 Enter Mode

Set PWR_CFG.PG to 1, the System Control Register SLEEPDEEP of CPU core to 1, and execute the WFI/WFE command to enter standby mode.

If only PWR_CFG.PG is set to 1, and SLEEPDEEP is set to 0, the chip will enter the sleep mode rather than the standby mode after executing the WFI/WFE command.

24.6.2 Exit Mode

IO wake-up or IWDG timing wake-up signal can wake the chip from deep sleep.

External reset or IWDG reset can reset all chips and release the deep sleep state.

Note: At this time, the debug operation cannot wake up the chip from the standby mode.

When programming applications, please try to avoid entering the sleep state immediately after power on, and set a reasonable wake-up source before entering the sleep mode.

24.7 Register

24.7.1 Address Allocation

The base address of PMU is 0x4001B800.

Table 24-2 PMU Address Space

Name	Offset	Description
BRAM	0x00~0x7F	Backing memory
PWR_CFG	0x80	PMU configuration register
EVT_RCD	0x84	Event record register
IO_WAKE_POL	0x88	IO wake-up polarity register
IO_WAKE_EN	0x8C	IO wake-up enable register

24.7.2 BRAM Backing Memory

The backing memory has 128bytes and is not powered off in standby mode. It is used to keep the key data in memory or register before entering the standby mode to avoid data loss after power off. BRAM only allows writing and reading by word, it can write continuously or read continuously, but avoid reading immediately after writing data. BRAM needs at least one bus cycle for writing internally.

24.7.3 PMU Configuration Register (AON_PWR_CFG)

Address: 0x4001_B880

Reset value: 0x0

Table 24-3 PMU Configuration Register AON_PWR_CFG

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

Bit field	Bit Name	Description
[31:2]		Unused
[1]	IOWK_FLT	IO wake-up signal filter enable. The default value is enable.
[0]	PG	Standby mode configuration bit

24.7.4 Event Record Register (AON_EVT_RCD)

Address: 0x4001_B884

Reset value: 0x0

Table 24-4 Event Record Register (AON_EVT_RCD)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	STANDBY	DEEPSLEEP	SLEEP			IWDG_WK	IO_WK			CPOR_RST	WWDG_RST	IWDG_RST	KEY_RST	HPOR_RST	LPOR_RST
	RO	RO	RO			RO	RO			RO	RO	RO	RO	RO	RO
	0	0	0			0	0			0	0	0	0	1	1

Bit field	Bit Name	Description
[31:15]		Unused
[14]	STANDBY	Standby (power-off) mode record, high value indicates occurred
[13]	DEEPSLEEP	Deep sleep record, high value indicates occurred
[12]	SLEEP	Sleep record, high value indicates occurred
[11:10]		Unused
[9]	IWDG_WK	IWDG timing wake-up record, high value means sleep is periodically woken up by IWDG, including wake-up of Deep Sleep and Standby modes
[8]	IO_WK	IO wake-up record, high value means sleep is periodically woken up by IO, including wake-up of Deep Sleep and Standby modes
[7:6]		Unused
[5]	CPOR_RST	CPU power-on reset occurrence record, high value indicates occurred
[4]	WWDG_RST	WWDG reset occurrence record, high value indicates occurred
[3]	IWDG_RST	IWDG reset occurrence record, high value indicates occurred
[2]	KEY_RST_RCD	Key reset occurrence record, high value indicates occurred
[1]	HPOR_RST_RCD	HPOR reset occurrence record, high value indicates occurred
[0]	LPOR_RST_RCD	LPOR reset occurrence record, high value indicates occurred

Write 0xCA40 to EVT_RCD register to clear all event records. Because EVT_RCD works in the LSI clock domain, it takes a certain time for the software to write the password to clear it.

24.7.5 IO Wake-up Polarity Register (AON_IO_WAKE_POL)

Address: 0x4001_B888

Reset value: 0x0

Table 24-5 IO Wake-up Polarity Register (AON_IO_WAKE_POL)

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

	WK_POL
	RW
	0

Bit field	Bit Name	Description
[31:1]		Unused
[0]	WK_POL	IO external wake-up trigger level selection. 1: high level; 0: low level

24.7.6 IO Wake-up Enable Register (AON_IO_WAKE_EN)

Address: 0x4001_B88C

Reset value: 0x0

Table 24-6 IO Wake-up Enable Register (AON_IO_WAKE_EN)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								P3_2_EN	P2_14_EN	P1_11_EN	P0_13_EN	P0_5_EN	P0_4_EN	P0_3_EN	P0_2_EN
								RW	RW	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

Bit field	Bit Name	Description
[31:8]		Unused
[7]	P3_2_EN	P3[2] external wake-up trigger enable
[6]	P2_14_EN	P2[14] external wake-up trigger enable
[5]	P1_11_EN	P1[11] external wake-up trigger enable
[4]	P0_13_EN	P0[13] external wake-up trigger enable
[3]	P0_5_EN	P0[5] external wake-up trigger enable
[2]	P0_4_EN	P0[4] external wake-up trigger enable
[1]	P0_3_EN	P0[3] external wake-up trigger enable
[0]	P0_2_EN	P0[2] external wake-up trigger enable

25 Version history

Table 25-1 Document's Version History

Time	Version No.	Description
2024.01.19	1.47	Add a note about sleep, Modify IIC clock frequency splitting
2023.10.22	1.46	Add the chip version information register
2023.09.25	1.45	Added notes on IWDG feeding dogs clock issues and GPIO PDI
2023.08.10	1.44	Modify errors in the DAC calibration address and GPIO sections
2023.07.09	1.43	Modify the IWDG address and Figure 3-4 SRAM0 error
2023.05.07	1.42	Updated the description for SYS_FLSE and SYS_FLSP
2023.04.07	1.41	Add the description of the acquisition Vcm
2023.02.22	1.40	Modified the LSI precision range
2023.02.18	1.39	Modified the description of MCPWM_SDCFG
2023.02.10	1.38	Added descriptions of PLLPDN, RCHPD, BGPPD, LDO2_CTL
2023.02.07	1.37	Update SYS_AFE_REG3.ADCCLKSEL description
2023.01.29	1.36	Modify the incorrect part of MCPWMx_EIF and MCPWMx_RE register address
2022.11.23	1.35	Added soft reset ADC module description
2022.11.10	1.34	Add connection resistance between IO and internal analog circuit, Updated ADC
2022.09.05	1.33	Update MCPWMx_IO01/IO23, P/N are interchanged by default.
2022.08.21	1.32	Update CAN-FD
2022.07.29	1.31	Simplify SYS_AFE_REG description
2022.06.24	1.3	Updated CMPx_SELN/CMPx_SELP
2021.09.15	1.2	Minor revision
2021.07.04	1.1	Update CAN
2021.05.24	1.0	Release
2021.04.29	0.9	Modified FMAC figures
2021.04.05	0.8	Updated FSMC, that is chapter on-chip flash
2021.04.02	0.7	Updated SPI
2021.04.02	0.6	Updated I2C
2021.03.12	0.5	Updated DMA
2021.03.11	0.4	Updated ADC
2021.03.09	0.3	Updated SYS_AFE_REGx, checked all text expression, and updated block diagram of module description
2021.06.16	0.2	Added FMAC
2020.02.08	0.1	Initial version, an internal version that contains test-related instructions

Disclaimer

LKS and LKO are registered trademarks of Linko.

Linko tries its best to ensure the accuracy and reliability of this document, but reserves the right to change, correct, enhance, modify the product and/or document at any time without prior notice. Users can obtain the latest information before placing an order.

Customers should select the appropriate Linko product for their application needs and design, validate and test your application in detail to ensure that it meets the appropriate standards and any safety, security or other requirements. The customer is solely responsible for this.

Linko hereby acknowledges that no intellectual property licenses, express or implied, are granted to Linko or to third parties.

Resale of Linko products on terms other than those set forth herein shall void any warranty warranties made by Linko for such products.

For earlier versions, please refer to this document.