Linko Semiconductor Co., Ltd.

LKS32MC03x User Manual

Content

18.2   REGISTER 220

   18.2.1   *Address Allocation* 220

   18.2.2   *DSP Division Registers* 221

      18.2.2.1   DSP_DID 221

      18.2.2.2   DSP_DIS 221

      18.2.2.3   DSP_QUO 221

      18.2.2.4   DSP_REM 222

   18.2.3   *DSP SQRT Registers* 222

      18.2.3.1   DSP_RAD 222

      18.2.3.2   DSP_SQRT 223

19   INDEPENDENT WATCHDOG (IWDG) **224**

   19.1   INTRODUCTION 224

   19.2   REGISTER 225

      19.2.1   *Address Allocation* 225

      19.2.2   *IWDG Password Register (IWDG_PSW)* 226

      19.2.3   *IWDG Configuration Register (IWDG_CFG)* 226

      19.2.4   *IWDG Clear Register (IWDG_CLR)* 226

      19.2.5   *IWDG Counter Timing Wake-up Threshold Register (IWDG_WTH)* 227

      19.2.6   *IWDG Counter Timeout Reset Threshold Register (IWDG_RTH)* 227

      19.2.7   *IWDG Current Count Register (IWDG_CNT)* 228

20   POWER MANAGEMENT MODULE (PMU) **229**

   20.1   PERIPHERAL CLOCK GATING 229

   20.2   PERIPHERAL CLOCK DIVIDER 229

   20.3   LOW POWER MODE 229

   20.4   SLEEP MODE 229

      20.4.1   *Enter Mode* 230

      20.4.2   *Exit Mode* 230

   20.5   DEEP SLEEP MODE 230

      20.5.1   *Enter Mode* 230

      20.5.2   *Exit Mode* 230

List of Tables

xxiv

List of Figures

iii

iv

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.

1.2 Abbreviations

Word: 32-bit data/instruction.

Halfword: 16-bit data/instruction.

Byte: 8-bit data.

Double word: 64-bit data.

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 4/8/12MHz RC oscillator

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

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

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

LKS32MC03x series MCUs are extremely cost-effective and compact, equipped with a 48MHz Cortex-M0 core, as well as necessary analog and peripheral resources. The module integration is more refined and widely suitable for lightweight applications.

2.2 Feature

- 48MHz 32-bit RISC kernel, 32bit hardware division coprocessor

- 4-channel DMA

- Low power sleep mode

- -40~105°C industrial operating temperature range

- 2.5V~5.5V single power supply, internally integrated digital power supply LDO

- Super antistatic and group pulse ability

2.2.1 Storage

- 16/32kB Flash, data anti-theft function

- 4kB RAM

2.2.2 Clock

- Built-in 4MHz high-precision RC clock, with full temperature range accuracy ±1%

- Built-in 32kHz low speed clock for low power mode

- Internal PLL can provide up to 48MHz clock

2.2.3 Peripheral Clock

- 1-channel UART

- 1-channel SPI

- 1-channel IIC

- Universal 16/32 bit Timer supports capturing and edge-aligned PWM

- Motor control dedicated PWM module supports 6 PWM outputs, and the dead zone can be configured

    o   Hall signal interface supports speed measurement and debouncing

    o   Hardware Watchdog

    o   26-channel GPIO

2.2.4 Analog Module

    o   Built-in 12bit SAR ADC, 1Msps sampling and conversion rate, up to 11 channels

    o   Built-in 1 OPA. Differential PGA mode is available

    o   Built-in 2 comparators

    o   Built-in 8bit DAC as internal comparator input

    o   1.2V 0.5% built-in linear regulator

    o   Built-in low power LDO and power monitoring circuit

    o   Built-in high-precision, low-temperature drift and high-frequency RC clock

2.3 **System Block Diagram**
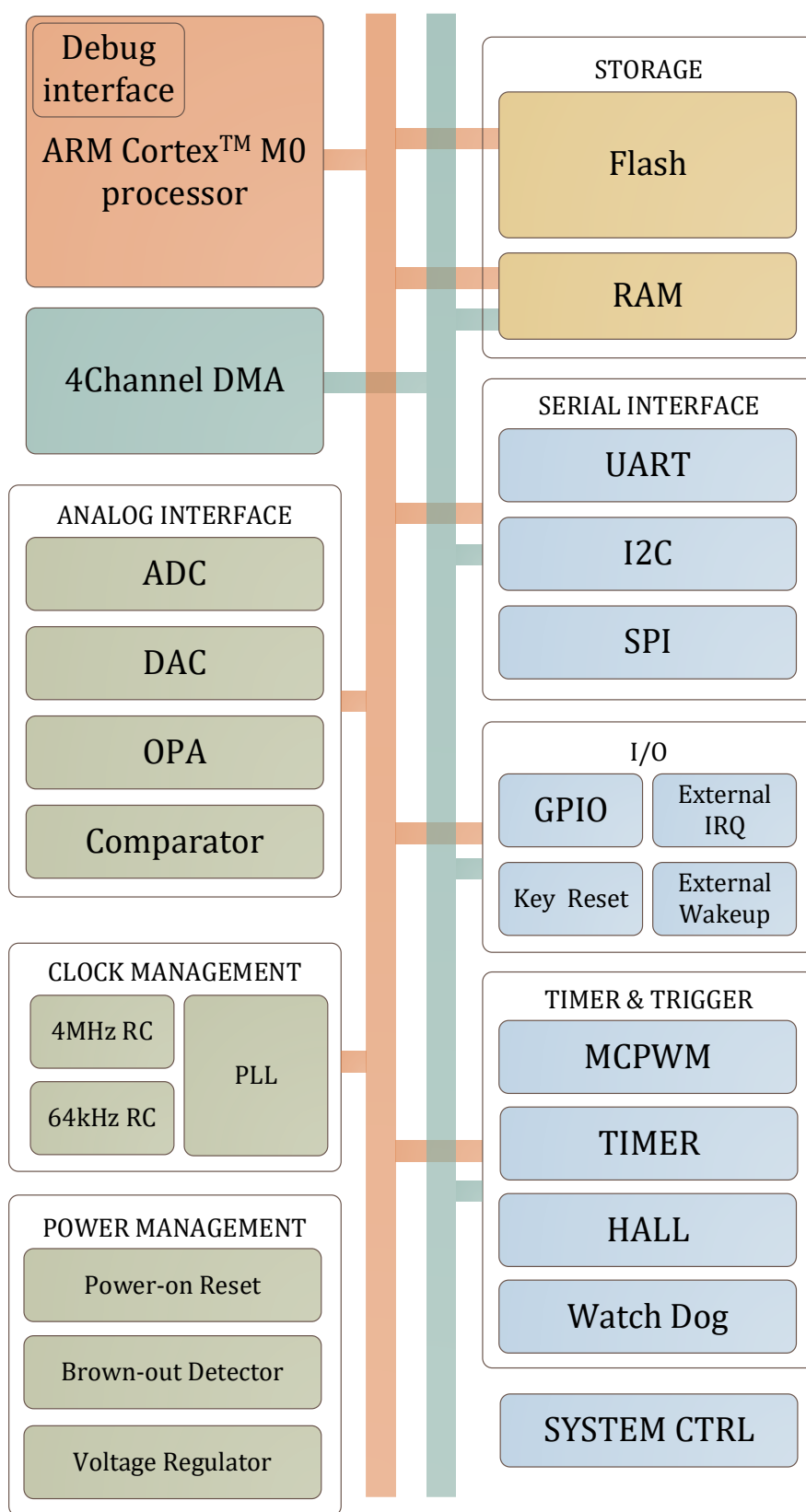
Fig. 2-1 LKS32MC03x system block diagram

4

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.

Table 3-1 System Address Space Allocation

Type	Device	Start Address	End Address	Size	Clock/Soft Reset
CODE	FLASH	0x0000_0000	0x0000_7FFF	32kB	Synchronous bus
	FLSCR	0x0001_0000	0x0001_00FF	256B	Synchronous bus
RAM	RAM	0x2000_0000	0x2000_0FFF	4kB	Synchronous bus
	SYS	0x4000_0000	0x4000_00FF	256B	Synchronous bus
	SPI	0x4001_0000	0x4001_00FF	256B	Peripheral clock gating [10] Soft reset [10]
	I2C	0x4001_0100	0x4001_01FF	256B	Peripheral clock gating [0] Soft reset [0]
	CMP	0x4001_0200	0x4001_02FF	256B	Peripheral clock gating [3] Soft reset [3]
	HALL	0x4001_0300	0x4001_03FF	256B	Peripheral clock gating [1] Soft reset [1]
	ADC	0x4001_0400	0x4001_04FF	256B	Peripheral clock gating [9] Soft reset [9]
	TIMER0	0x4001_0500	0x4001_05FF	256B	Peripheral clock gating [5] Soft reset [5]
	TIMER1	0x4001_0600	0x4001_06FF	256B	Peripheral clock gating [6] Soft reset [6]
	MCPWM	0x4001_0700	0x4001_07FF	256B	Peripheral clock gating [4] Soft reset [4]
	GPIO	0x4001_0800	0x4001_08FF	256B	Peripheral clock gating [7] Soft reset [7]
	UART	0x4001_0900	0x4001_09FF	256B	Peripheral clock gating [2] Soft reset [2]
	DMA	0x4001_0A00	0x4001_0AFF	256B	Peripheral clock gating [11] Soft reset [11]
	DIV	0x4001_0B00	0x4001_0BFF	256B	Peripheral clock gating [8] Soft reset [8]
	AON	0x4001_0C00	0x4001_0CFF	256B	Synchronous bus

Please refer to 6.3.7 SYS_CLK_FEN for the above peripheral clock gating control register, and 6.3.8 SYS_SFT_RST for soft reset control register.

4　Interrupt

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.

LKS32MC03x series chips support 14 external interrupt sources.

The interrupt controller supports up to four interrupt priority levels for programming.

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	TIMER0	16	Reserved
1	TIMER1	17	Reserved
2	MCPWM0	18	Reserved
3	MCPWM1	19	Reserved
4	I2C	20	Reserved
5	SPI	21	Reserved
6	GPIO	22	Reserved
7	HALL	23	Reserved
8	UART	24	Reserved
9	CMP	25	Reserved
10	ADC	26	Reserved
11	DMA	27	Reserved
12	WAKEUP, system wake-up interrupt	28	Reserved
13	Software interrupt	29	Reserved
14	Reserved	30	Reserved
15	Reserved	31	Reserved

5 Analog Circuit

5.1 Introduction

The analog circuit contains the following modules:

- Built-in 12bit SAR ADC, 1MHz sampling rate, up to 11 channel signals optional for each sampling circuit

- Built-in 1 operational amplifier. PGA mode is available

- Built-in 2 comparators. Hysteresis mode is available

- Built-in 8bit digital-to-analog converter (DAC)

- ± 2 ℃ 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.
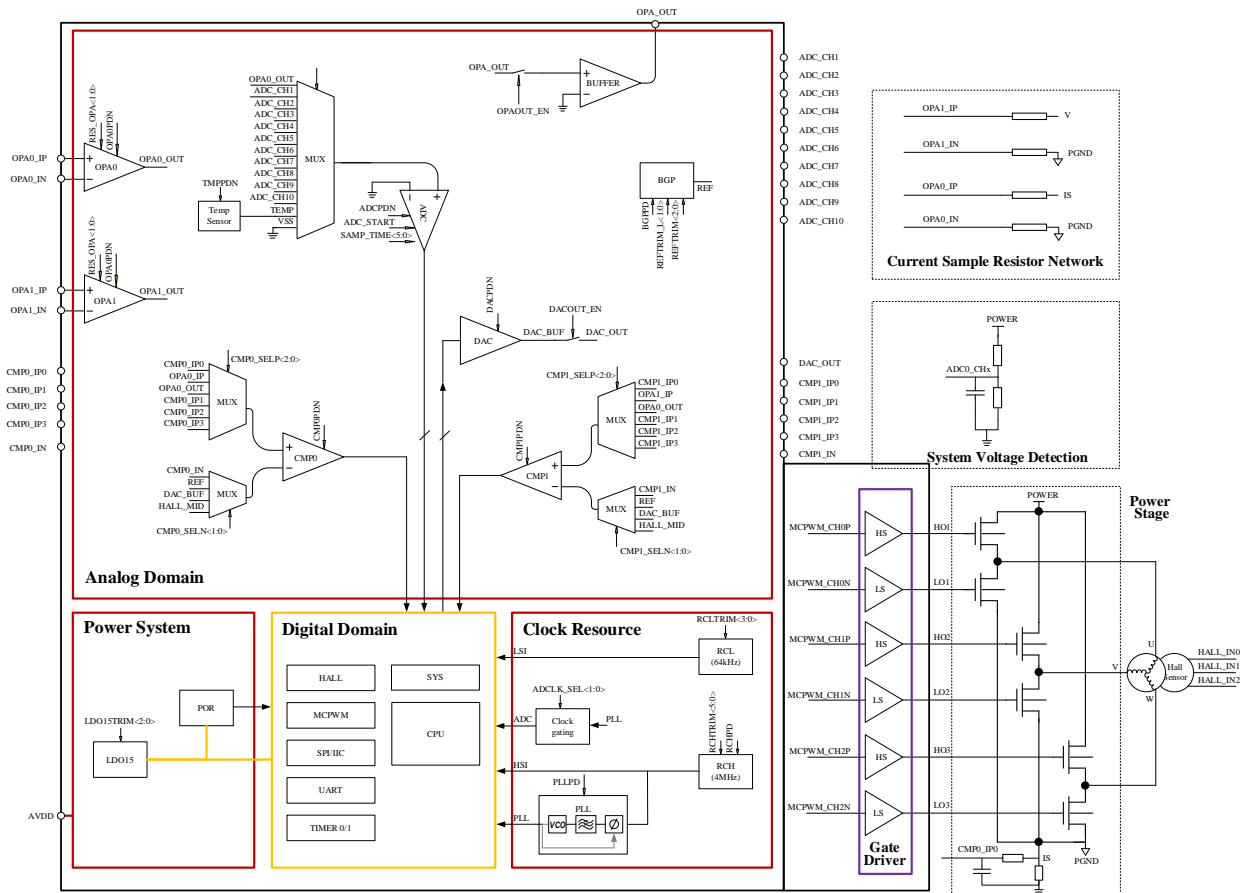
Fig. 5-1 Functional Block Diagram of Analog Circuit

5.1.1 Power Management System (POWER)

The power management system is composed of LDO15 module, power detection module (PVD), power-on/power-off reset module (POR).

The chip is powered by single 3.3 ~ 5V power supply to save off-chip power costs, and all internal digital circuits and PLL modules are powered by an internal LDO15.

The LDO automatically turns on after power-on, without software configuration.

The POR module monitors the voltage of the LDO15 and provides a reset signal for the digital circuit to avoid faults occurred when the voltage of LDO15 is lower than 1.25V (e.g.: power-on or power-off).

5.1.2 Clock System (CLOCK)

The clock system consists of an internal 32kHz RC oscillator, a 4MHz RC oscillator, and a PLL.

The 32kHz RC clock LSI is mainly used for watchdog module and reset/wakeup source filters in the system. It can also be used as the main clock of the MCU. The 4MHz RC clock can be used as the main clock of the MCU. PLL clock is up to 48MHz and is usually used as the main clock of MCU.

Both 32kHz and 4MHz RC clocks have been through the correct calibration procedure at the factory, in the range of -40 ~ 105°C, the accuracy of the 32kHz RC clock is ± 50%, and the accuracy of the 4MHz RC clock is ± 1%.

The calibration parameters for 4MHz RC clock at 5V differ slightly from 3.3V. For most 5V applications, the 4MHz RC clock has been calibrated. If it is applied at 3.3V, it is recommended to load the correction parameters to the corresponding register through software. For details, refer to the relevant application notes. If not loaded, the accuracy deviation of 4MHz RC clock increases by about 1.5%.

The 4MHz RC clock is turned on by setting BGPPD = '0' (ON by default, turn off when set to "1"). The RC clock needs a reference voltage and current provided by the Bandgap voltage reference module; thus, do remember to turn on the BGP module (BGPPD="0") before turning on the RC clock. When the chip is powered on, the 4MHz RC clock and BGP module are both turned on automatically. The 32kHz RC clock is always on and cannot be turned off.

The PLL multiplies the 4MHz RC clock to provide a higher frequency clock for modules like MCU and ADC. The highest frequency of MCU and PWM module is 48MHz, and the typical working frequency of ADC module is 24MHz.

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 (Bandgap) module should be turned on first. After the PLL is turned on, it needs a settling time of 6us 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 the description of BGPPD/PLLPDN, see the analog register SYS_AFE_REG0.

5.1.3 Bandgap Voltage Reference (BGP)

Bandgap Voltage Reference (BGP) provides reference voltage and current for ADC, DAC, RC clock, 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%

For the description of BGPPD, see the analog register SYS_AFE_REG0

5.1.4 ADC Module

Please refer to Chapter 12.

5.1.5 Operational Amplifier (OPA)

The 1-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_OPA [1:0] to achieve different gains.

For the description of RES_OPA<1:0>, see the analog register SYS_AFE_REG0

The schematic diagram of the amplifier is as follows:
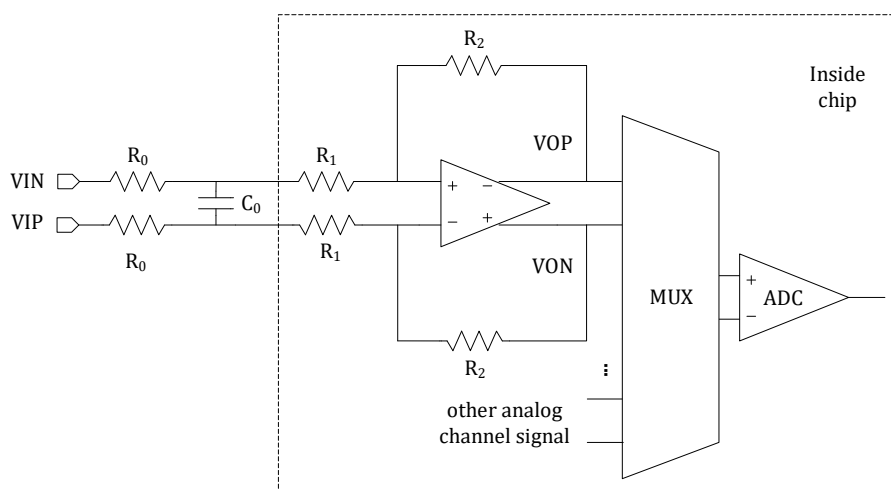
Fig. 5-2 Amplifier Block Diagram

The two R0 in the figure are the resistance of the external resistor, so the resistance must be equal. The close-loop gain of OPA is R2/(R1+R0).

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. C0 is the signal filter capacitor; it forms a first-order RC filter circuit with R0, and the specific resistance of R0 can be determined based on the filter constant of R0* C0. The filtering is not necessary if the Signal to Noise Ratio (SNR) is small, or minimum C0 is 15pF when the signal requires a large bandwidth (faster response speed).

The OPA can select the output signal of the amplifier by setting OPAOUT_EN, and send it to the P0.7 IO 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 the description of OPAOUT_EN, see the analog register SYS_AFE_REG0.

When the chip is powered on, the OPAMP module is OFF by default. It can be turned on by setting OPAPDN = 1, and turn on the BGP module before turning on the amplifier.

For the description of OPAPDN, see the analog register SYS_AFE_REG0

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.

The OPA has multiple sets of differential inputs on the chip port. Among them, OPA0_IP/OPA0_IN and OPA1_IP/OPA1_IN share the single internal operational amplifier at different time slot. When the ADC current sampling channel is ADC_CH8, that is, ADC0_CHN0[3:0]=8, it is actually sampling the output of OPA with the OPA1 differential input signal being amplified by the. When the ADC current sampling channel is configured as 0, it is ADC0_CHN0[3:0] =0, it is actually sampling the output of OPA with the OPA0 differential input signal being amplified by the OPA.

In addition, there are two sets of differential inputs of OPA0, OPA0_IP/OPA0_IN and OPA0_IP_B/OPA0_IN_B, which are selected by SYS_AFE_REG0.OPA0_B_EN. This setting is usually configured according to the pin function distribution of the specific chip model and remains unchanged during the application.

OPA's multiple input and selection control logic is show as figure below.
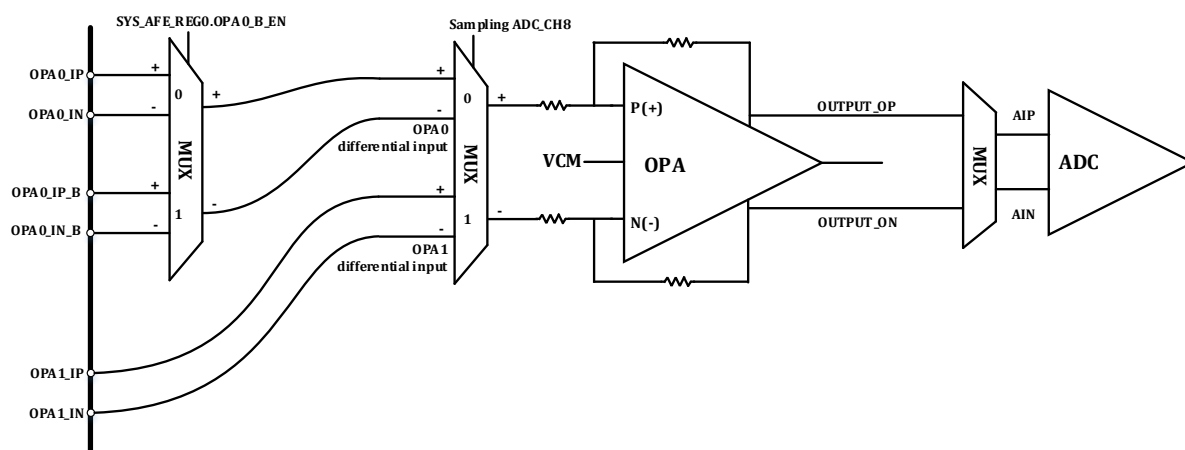
Fig. 5-3 OPA's multiple input and selection control logic

When ADC is sampling ADC_CH8, it's actually sampling OPA1 signal after amplified by OPA. The OPA input switch signal will always be in line with the next ADC channel to be sampled. It's recommended to set OPA1(i.e. ADC_CH8) as the first channel in the ADC sampling sequence which will allow enough settle time for OPA1 as OPA input.

At 1us settle time is required when switch OPA input between OPA0_IP/OPA0_IN and OPA1_IP/OPA1_IN. It's NOT recommended to sample OPA0/1 one after another right away.
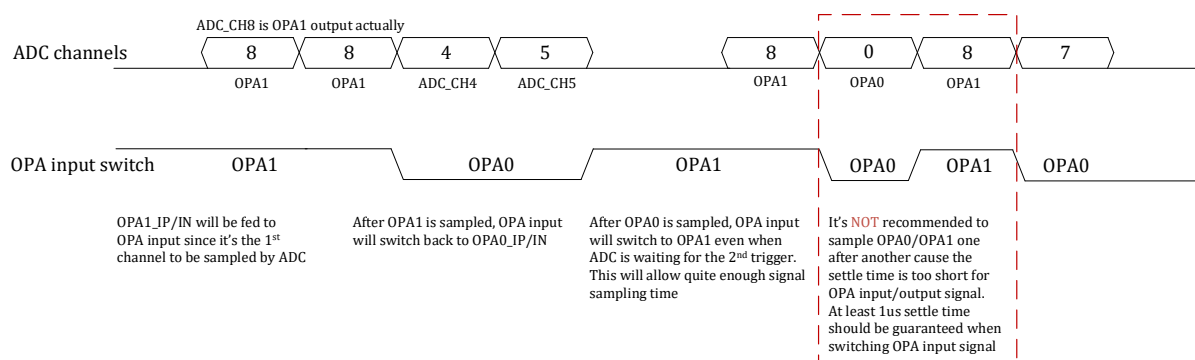
Fig. 5-4 OPA input switching and ADC sampling cooperation

5.1.6 Comparator (CMP)

Built-in 2-channel rail-to-rail comparators with programmable comparator speed, hysteresis voltage, and signal source.

The comparison delay can be set to <30nS/200nS through the register CMP_FT, and the hysteresis voltage can be set to 20mV/0mV by 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, which represents the two comparators, CMP0 and CMP1).

It should be noted that The BEMFx_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 Fig. 53. Among them, the resistance R=8.2k ohms, the switch in the picture will be turned on only after the negative input signal of the comparator is selected as BEMFx_MID, otherwise the switches are in the off state.

BEMFx_MID is mainly used for BLDC square wave mode control, the virtual motor phase line center point voltage, used for back-EMF zero-crossing detection. After the three phase lines are divided, connect to CMPx_IP1, CMPx_IP2, and CMPx_IP3 respectively. The MCU controls the negative end of the comparator to select BEMFx_MID, and the multiplexer at the positive end of the comparator selects CMPx_IP1, CMPx_IP2, and CMPx_IP3 in a time-division multiplexing manner. Compare the zero-crossing point of the back EMF.

CMPx_IP1

R

R   R

CMPx_IP2   CMPx_IP3
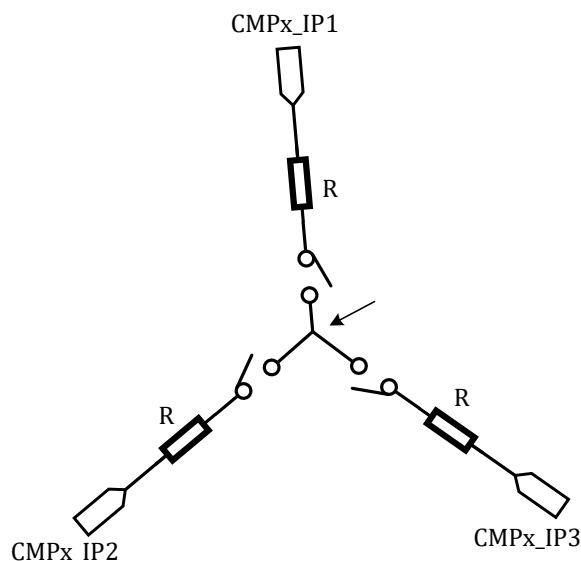
Fig. 5-5 BEMFx_MID Signal

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

For the description of CMP_FT, see the analog register SYS_AFE_REG1

For the description of CMPx_SELN<1:0>/ CMPx_SELP<2:0>/ CMP_HYS, see the analog register SYS_AFE_REG1

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), and turn on the BGP module before turning on the comparator.

For the description of CMPxPDN, see the analog register SYS_AFE_REG0

5.1.7 Temperature Sensor (TMP)

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 11 of the ADC.

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

For the description of TMPPDN, see the analog register SYS_AFE_REG0

The typical curve of the temperature sensor is shown in the figure below:
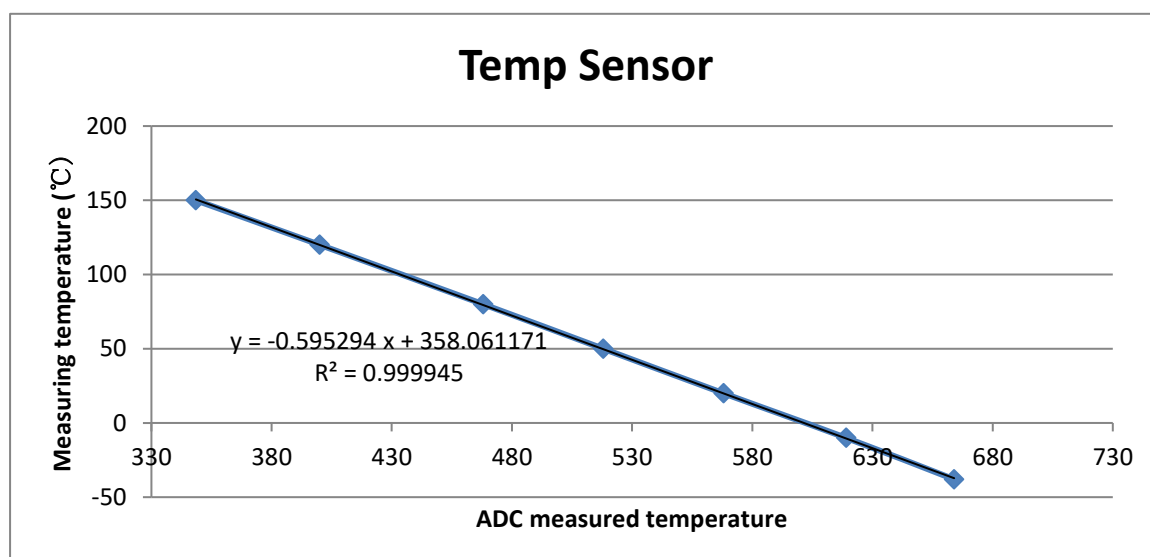
Fig. 5-6 Temperature Sensor Curve

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=-0.595x+358.1

The calculated value Y is the current temperature.

There are two coefficients in the formula, a = -0.6032, b = 364.96, 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 (read from library functions). 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 (read from library functions). 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 four bits before putting into the above formula.

5.1.8 Digital-to-analog Converter (DAC)

The chip has a 1-channel 8bit DAC, the maximum range of the output signal can be set to 3V. For C Version, the maximum range of the output signal can be set to 4.8V, You need to set SYS_AFE_REG2.BIT15=1 to use the DAC's 1.2V range.

The DAC can be output via IO port P0.0 by setting register DACOUT_EN = 1, which can drive a load resistance of over 5kΩ and a load capacitance of 50pF.

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 DACPDN = 1. Turn on the BGP module before turning on the DAC module.

The input digital signal register of DAC is SYS_AFE_DAC, low 8-bit is 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 0xFF is DAC_{fs}. The analog signal amplitude corresponding to each gear signal (LSB) is $\frac{DAC_{fs}}{256}$. If the digital value of SYS_AFE_DAC is Din, the analog signal of the DAC output corresponding to the digital signal is $\frac{DAC_{fs}}{256} * Din$

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.

The DAC AMC/DC correction values are stored in the NVR and can be used to calibrate the DAC output, please refer to the official library functions for specific usage.

For the description of DACOUT_EN, see SYS_AFE_REG1

For the description of DACPDN, see SYS_AFE_REG0

For the description of SYS_AFE_DAC, see SYS_AFE_DAC DAC

5.2   **Register**

5.2.1 Address Allocation

The analog register SYS_AFE_REG0 ~ SYS_AFE_REG2, among which the reserved registers (Res) 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 0x4000_0000.

Table 5-1 System Control Register

Name	Offset	Description
SYS_AFE_ADC	0x00	ADC data register
SYS_AFE_INFO	0x04	Chip version information register
	0x08	Reserved
SYS_OPA_SEL	0x0C	OPA switch control register
SYS_AFE_REG0	0x10	AFE 0
SYS_AFE_REG1	0x14	AFE 1
SYS_AFE_REG2	0x18	AFE 2
SYS_AFE_DAC	0x2C	DAC digital register

5.2.2 SYS_AFE_ADC ADC Original Data Register

Address: 0x4000_0000

Reset value: 0x0

Table 5-2 ADC Original Data Register SYS_AFE_ADC

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

Location	Bit Name	Description
[31:12]		Unused
[11:0]	ADC_RAW_D	Original data output by ADC

5.2.3 SYS_AFE_INFO Chip Version Information Register

Address: 0x4000_0004

Reset value: Different according to wafer version

Table 5-3 Chip Version Information Register SYS_AFE_INFO

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

Location	Bit Name	Description
[31:3]		Unused
[2:0]	Version	Chip version information 1: DAC range 0~3V 2: DAC range 0~3V/ 4.8V 3: DAC range 0~1.2V/ 3V/ 4.8V

This register is user-independent.

5.2.4 SYS_OPA_SEL OPA Switch Control Register

Address: 0x4000_000C

Reset value: 0x1

Table 5-4 OPA Switch Control Register SYS_OPA_SEL

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

	OPA_SEL_EN
	RW
	0

Location	Bit Name	Description
[31:1]		Unused
[0]	OPA_SEL_EN	OPA0/1 inputs the unique OPA in the time division multiplex chip, multiplexing enable switch, active high. After version B, the software can read and write this BIT.

SYS_ OPA_ SEL=0xCA can set this register to 1 to enable multiplexing function. SYS_ OPA_ SEL&=~BIT0 can set this register to 0 and turn off the multiplexing function.

According to whether OPA_SEL_EN is enabled or not, the 2-1 multi-selector in the AFE will control

The chip has 1 OPA module but 2 OPA input signals, so OPA0/OPA1 signals multiplex OPA module.

OPASELA is controlled by the ADC interface module hardware, which is the same as the analog

OPASELA	OPA A input channel selection	0: Select OPA0_IP/OPA0_IN pin as the OPA signal channel
		1: Select OPA1_IP/OPA1_IN pin as the OPA signal channel

register ADC_SELP<3:0>.

The corresponding relation of ADC physical channel selection register is as follows:

ADC_SELP<3:0>	ADC channel positive selection	0000:OPA0 output; 0001: ADC_CH1; 0010: ADC_CH2; 0011: ADC_CH3; 0100: ADC_CH4; 0101: ADC_CH5; 0110: ADC_CH6; 0111: ADC_CH7; 1000:ADC_CH8/OPA1 output; 1001:ADC_CH9; 1010:ADC_CH10; 1011: Connect to temperature sensor; 1100: Connect to internal ground; 1101: Connect to 2.4V reference;

Among which, CH8 is multiplexed.

Table 5-5 ADC Channel Multiplexing OPA List

OPA_SEL_EN	Current ADC sampling channel	Actual ADC sampling channel
0	8	ADC_CH8

1	8	OPA1_OUT

When OPA_SEL_EN=1, OPA input channel switch is automatically controlled by the hardware circuit; the switch control signal OPASELA input by OPA is 1 only in 8th ADC sampling channel, and 0 in all other channels.

It is recommended to sampling in OPA0 and OPA1 channels 1 at a certain interval. For example, in ADC single-segment sampling mode, 6 channels are sampled at one time. Sample in OPA1 for the first sampling, and OPA0 for the 6th sampling, to give OPA input signal sufficient setting time. If OPA1 is sampled for the first time in a certain round of sampling, the OPA input control signal is switched to OPA1 long before the sampling starts, so that the OPA1 sampling has enough setting time. After sampling OPA1, OPA input signal will switch back to OPA0. If the number of interval channels is enough, the OPA input signal will also give the OPA sufficient setting time.

5.2.5 SYS_AFE_REG0 AFE Register 0

Address: 0x4000_0010

Reset value: 0x0

Table 5-6 AFE Register 0 SYS_AFE_REG0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLLPDN	CMP1PDN	CMP0PDN	TMPPDN	DACPDN	BGPPD	OPAPDN	ADCPDN	REF2VDD	GA_AD	Res.	OPASEL	Res.	OPAOUT_EN	RES_OPA	
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	

Location	Bit Name	Description
[31:16]		Unused
[15]	PLLPDN	0: PLL off 1: PLL on
[14]	CMP1PDN	0: CMP 1 off 1: CMP 1 on
[13]	CMP0PDN	0: CMP 0 off 1: CMP 0 on
[12]	TMPPDN	0: TMP off 1: TMP on
[11]	DACPDN	0: DAC off 1: DAC on
[10]	BGPPD	0: Bandgap on 1: Bandgap off. Note that the polarity is opposite to other modules
[9]	OPAPDN	0: OPA off 1: OPA on

[8]	ADCPDN	0: ADC off 1: ADC on
[7]	REF2VDD	ADC reference source selection 0: Select internal 2.4V as ADC reference source 1: Select the AVDD power supply voltage as the ADC reference source. When REF2VDD=1, it is recommended to set the ADC range GA_AD=1, and the ADC full scale corresponds to AVDD.
[6]	GA_AD	ADC range selection 0: REF*3/2, when using the internal 2.4V reference, the ADC full scale is 2.4*3/2=3.6V, when using the AVDD power supply voltage as the reference, the ADC full scale is AVDD*3/2, if using 5V power supply, that is, the full scale is 7.5V; 1: REF. When using the internal 2.4V reference, the full scale of the ADC is 2.4V. When using the AVDD power supply voltage as the reference, the full scale of the ADC is AVDD. If 5V is used, the full scale is 5V.
[5]	Reserved bits	Must be factory setting value, cannot be changed
[4]	Reserved bits	Must be 0
[3]	Reserved bits	Must be 0
[2]	OPAOUT_EN	0: No output; 1: Output OPA output signal to IO port P0.7
[1:0]	RES_OPA	OPA feedback resistance 00: 200k:10k 01: 190k:20k 10: 180k:30k 11: 170k:40k

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.

5.2.6 SYS_AFE_REG1 AFE Register 1

Address: 0x4000_0014

Reset value: 0x0

Table 5-7 AFE Register 1 SYS_AFE_REG1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.		CMP1_SELP		CMP_FT		CMP0_SELP		CMP_HYS	Res.		CMP1_SELN		CMP0_SELN	DACOUT_EN	LDOOUT_EN
RW		RW		RW		RW		RW	RW		RW		RW	RW	RW
0		0		0		0		0	0		0		0	0	0

Location	Bit Name	Description
[31:16]		Unused
[15]	Reserved	Must be 0
[14:12]	CMP1_SELP	Positive selection of Comparator 1 signal 000: CMP1_IP0 001: OPA0_IP 010: OPA0_OUT 011: CMP1_IP1 100: CMP1_IP2 101: CMP1_IP3 110: NC 111: NC
[11]	CMP_FT	Enable fast comparison of comparator 1: The comparison speed of comparator is less than 30ns 0: Disabled, 200ns
[10:8]	CMP0_SELP	Positive selection of Comparator 0 signal 000: CMP0_IP0 001: OPA0_IP 010: OPA0_OUT 011: CMP0_IP1 100: CMP0_IP2 101: CMP0_IP3 110: NC 111: NC
[7]	CMP_HYS	Comparator hysteresis selection 0: 20mV 1: 0mv
[6]	Reserved	Must be 0
[5:4]	CMP1_SELN	Negative selection of Comparator 1 signal 00: CMP1_IN 01: REF 10: DAC output 11: BEMFx_MID
[3:2]	CMP0_SELN	Negative selection of Comparator 0 signal

		00: CMP0_IN
		01: REF
		10: DAC output
		11: BEMFx_MID
[1]	DACOUT_EN	DAC output to IO enable
		0: No output
		1: Output DAC to P0.0
[0]	LDOOUT_EN	LDO output to IO enable
		0: No output
		1: Output LDO15 to P0.0

5.2.7SYS_AFE_REG2 AFE Register 2

Address: 0x4000_0018

Reset value: 0x0

Table 5-8 AFE Register 2 SYS_AFE_REG2

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

Location	Bit Name	Description
[31:16]		Unused
[15]	DAC_GAIN for C Version	Only for C Version
		0: 3.0V/4.8V range
		1: 1.2V range
[14:13]	Reserved	Must be 0
[12:8]	SAMP_TIME	ADC sampling time selection
		00000->11111: The corresponding sampling time gradually increases from 5 to 36 ADC clock cycles
[7:0]	Reserved	Must be 0

5.2.8DAC Digital Register (SYS_AFE_DAC)

Address: 0x4000_002C

Reset value: 0x0

Table 5-9 DAC Digital Register (SYS_AFE_DAC)

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

	DAC_IN
	RW
	0

Location	Bit Name	Description
[31:8]		Unused
[7:0]	DAC_IN	DAC Digital Input to be Converted

6   System Control and Clock Reset

6.1   **Clock**

6.1.1 Clock Source

As shown in the following table, the system includes 4 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/LRC	32kHz	Internal RC Oscillator	Full temperature range error <50 %	Internal system clock, used for watchdog module, and filtering and widening of reset signal. It can also be used as the MCU main clock
HSI/HRC	4MHz	Internal RC Oscillator	Full temperature range error <1%	Can be used as PLL source clock
PLL	48MHz	PLL Clock	Depends on the PLL reference clock.If the internal HRC is used as the PLL reference clock, the PLL clock error is < 1%.	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	<10MHz	Debugger	-	JTAG/SWD clock, depending on the host settings

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 4MHz reference clock HSI by 12 times to 48MHz.

After the PLL is divided by n/8, the high-speed clock of 48MHz × n/8 can be obtained. SYS_CLK_CFG.CLK_SEL[0] chooses one between the divided high-speed clock and the 4MHz 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, 4MHz 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 12, 24, 48 MHz and other frequency values. SYS_CLK_CFG.CLK_SEL means PLL or HSI is selected as the main system clock. Besides, the system clock can be switch to LSI by setting SYS_CLK_CFG.CLK_SEL[1]=1. At this time, the system works at the 32kHz clock, and PLL/HRC/BGP analog modules can be turned off to reduce power consumption.

When SYS_CLK_CFG.CLK_SEL is 1, SYS_CLK_CFG.CLK_DIV is used as the PLL frequency division factor. When SYS_CLK_CFG.CLK_SEL is not 0, HSI or LSI is used as the working clock, and SYS_CLK_CFG.CLK_DIV has no effect.

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	6	Yes
0x0111	2/8	12	Yes
0x0155	4/8	24	Yes
0x01FF	8/8	48	Yes

The MCLK clock is supplied to the peripheral clock after the switch controlled by the SYS_CLK_FEN register. The I2C clock could be further divided by the SYS_CLK_DIV0 register, and the UART 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 48MHz) after being divided by 2, 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. When the system does not have many tasks and the power consumption is reduced, set SYS_CLK_CFG.CLK_SEL to 2 to switch the system clock to LSI.
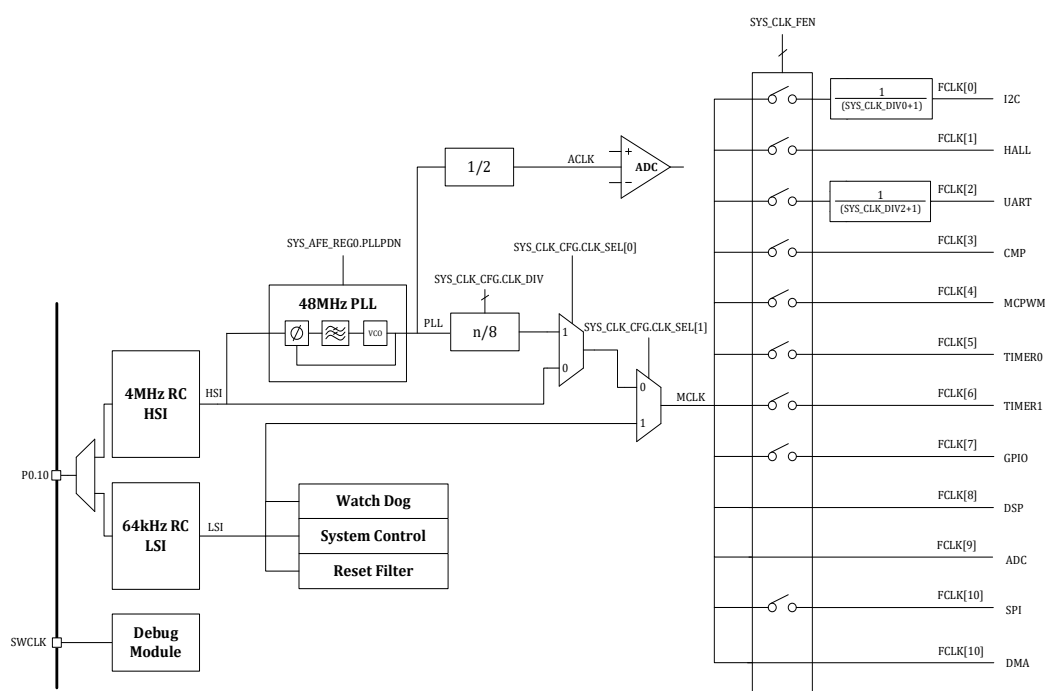
Fig. 6-1 Clock Architecture

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 frequency can be selected in the debug interface.

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**

6.2.1Reset 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 Hardware Reset Sources, the system has three 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.

Table 6-3 System Reset Source

Name	Source	Description
PORn	Internal Power Management	Monitor 1.5V digital power supply and 5V power supply, reset when the 1.5V is below 1.25V, or when the 5V is below 2.2V
RSTn	External Button	External reset circuit, active low
WDTn	Hardware Watchdog	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

As shown below, PORn is an internal analog circuit, and RSTn is an external key.

WDTn is a LSI clock cycle width signal, which is an internal digital signal. WDTn signal will be shielded in Debug mode, which is configured by SYS_DBG_CFG.

After pre-filtering and broadening the reset signal, a global reset signal is output through AND algorithm.

A reset signal short than 16us to P0.2 will be filtered. A reliable reset signal should be as long as 200us.

The three reset signals are global reset, so the reset levels and scopes are the same.
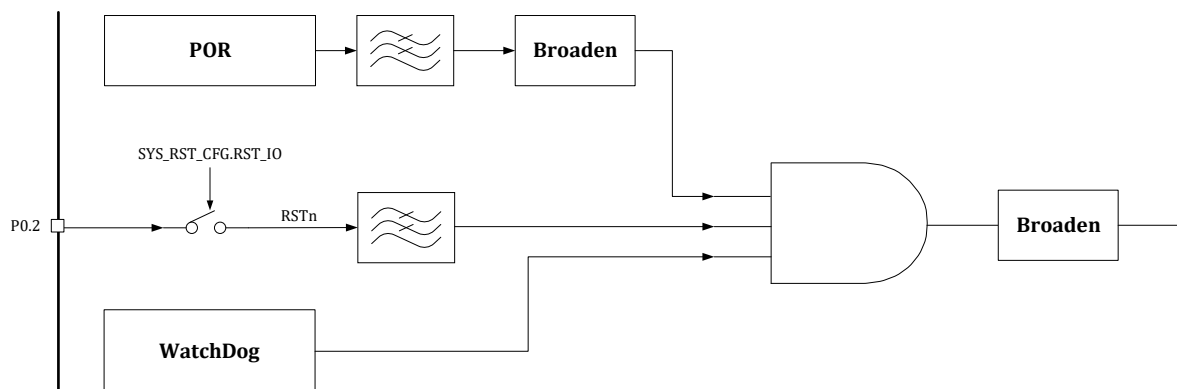
Fig. 6-2 Reset Architecture

6.2.1.1.2 Hardware Reset Records

The AON_EVT_RCD register is used to save hardware reset events. When a hardware reset occurs, the corresponding bit of AON_EVT_RCD is set to 1. The AON_EVT_RCD register itself cannot be reset by the reset signal; it can only clear the 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 PC (PC: Program Counter) 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" [RST] 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 6.3.8 for details.

6.2.2   Reset Scope

Table 6-4 Reset Scope

Reset Source	Scope
PORn	Internal power management, global reset
RSTn	External button, global reset, except very few registers
WDTn	Hardware watchdog, global reset, except very few registers
SYS_SFT_RST.SPI_ SFT_RST	SPI
SYS_SFT_RST.ADC_ SFT_RST	ADC digital interface module
SYS_SFT_RST.DIV_ SFT_RST	DIV
SYS_SFT_RST.GPIO_ SFT_RST	GPIO
SYS_SFT_RST.TIMER1_ SFT_RST	TIMER1
SYS_SFT_RST.TIMER0_ SFT_RST	TIMER0
SYS_SFT_RST.MCPWM_ SFT_RST	MCPWM
SYS_SFT_RST.CMP_ SFT_RST	Comparator digital interface module

SYS_SFT_RST.UART_SFT_RST	UART
SYS_SFT_RST.HALL_SFT_RST	HALL
SYS_SFT_RST.I2C_SFT_RST	I2C
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_RST_CFG.RST_IO controls whether P0[2] is used as GPIO or as an external reset pin, and the reset record register AON_EVT_RCD is reset only by POR.

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 0x4000_0000.

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_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

6.3.2 SYS_CLK_CFG Clock Control Register

Address: 0x4000_0080

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
						CLK_SEL		CLK_DIV							
						RW		RW							
						0		0							

Location	Bit Name	Description

[31:10]		Unused
[9:8]	CLK_SEL	Source selection signal for system clock MCLK. HRC is selected by default. 0: HRC 1: PLL 2: LRC 3: LRC Note that PLL 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 HRC clock (4MHz), and the division factor of SYS_CLK_CFG [7:0] is invalid. The final output clock frequency is 4MHz.

When CLK_SEL = 2, MCLK chooses the LRC clock (32kHz), and the division factor of SYS_CLK_CFG [7:0] is invalid. The final output clock frequency is 32kHz.

6.3.3SYS_IO_CFG IO Control Register

Address: 0x4000_0084

Reset value: 0x40

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
									SWDMUX	RST_IO					
									RW	RW					
									1	0					

Location	Bit Name	Description
[31:7]		Unused
[6]	SWDMUX	SWD multiplexing control signal. The default value is SWD 0:P1.8, P1.9 is used as the normal GPIO 1:P1.8 is multiplexed as SWCLK, and P1.9 is multiplexed as SWDIO
[5]	RST_IO	RSTn/ P0.2 multiplexing control signal. The default value is RSTn 0:RSTn 1:P0.2 Note that the value is RSTn by default after powering on. The RSTn function may

27

		become invalid when the RSTn is enabled in subsequent software
[4:0]		Unused

For the sake of safety, P1.8 and P1.9 can only be used as SWD but cannot be switched to GPIO function within 30ms after powering on Even if the software rewrites the SYS_MISC_CFG[6], it will take effect after 30ms. This means: Within 30ms, the software can write 1'b0 to SYS_IO_CFG[6], but the bit value returned is still 1'b1; if reading again after 30ms, the value returned will be 1'b0. That is, the software writing is valid, but it will not take effect immediately. This is to prevent the application from switching the SWD function to GPIO immediately after powering on, causing the chip to be unable to erase and program the flash through SWD later.

Note that after switch P1.8 and P1.9 to GPIO, the user can no longer debug the chip through SWD. Therefore, it is recommended to switch SWD to GPIO function after a certain delay. Moreover, it is recommended to design the software to allow P1.8 and P1.9 switch back to the SWD function.

6.3.4SYS_DBG_CFG Debug Control Register

Address: 0x4000_0088

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_TIM1_STOP	DBG_TIM0_STOP			DBG_IWDG_STOP				DBG_STOP	DBG_SLP
RW1C					RW	RW	RW			RW				RW	RW
0					0	0	0			0				0	0

Location	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.
[14:11]		Unused
[10]	SFT_RST_PERI	Whether Debug Soft Reset resets all peripheral registers except Flash/SYS_AFE Write 0x1F to BIT[14:10] to set the bit
[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
[4:2]		Unused
[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. Set SCB->SCR = (1UL<<2), and use __WFI()/___WFE() instruction to enter the STOP mode.
[0]	DBG_SL P	Debug LEEP 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() instruction can set the processor to enter the SLEEP mode

6.3.5 SYS_CLK_DIV0 Peripheral Clock Divider Register 0

Address: 0x4000_0090

Reset value: 0x0

Table 6-9 SYS_CLK_DIV0 Peripheral Clock Divider Register 0

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

Location	Bit Name	Description
[31:16]		Unused
[15:0]	DIV0	I2C clock=MCLK/(CLK_DIV0+1). MCLK is determined by the SYS_CLK_CFG division factor.

6.3.6 SYS_CLK_DIV2 Peripheral Clock Divider Register 2

Address: 0x4000_0098

Reset value: 0x0

Table 6-10 SYS_CLK_DIV2 Peripheral Clock Divider Register 2

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

Location	Bit Name	Description
[31:16]		Unused
[15:0]	DIV2	UART clock=MCLK/(CLK_DIV2+1). The baud rate is divided by the UART baud rate register, where MCLK is determined by the SYS_CLK_CFG division factor.

6.3.7SYS_CLK_FEN Peripheral Clock Gating Register

Address: 0x4000_009C

Reset value: 0x0

Table 6-11 SYS_CLK_FEN Peripheral Clock Gating Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				Res.	SPI_CLK_EN	Res.	DSP_CLK_EN	Res.	TIMER1_CLK_EN	TIMER0_CLK_EN	MCPWM_CLK_EN	CMP_CLK_EN	UART_CLK_EN	HALL_CLK_EN	I2C_CLK_EN
				RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
				0	0	0	0	0	0	0	0	0	0	0	0

Location	Bit Name	Description
[31:12]		Unused
[11]		Reserved
[10]	SPI_CLK_EN	SPI clock control. It is turned off by default. 1: Enable SPI clock 0: Disable SPI clock
[9]		Reserved
[8]	DSP_CLK_EN	DSP clock control. It is turned off by default. 1: Enable DSP clock 0: Disable DSP clock
[7]		Reserved
[6]	TIMER1_CLK_EN	TIMER1 clock control. It is turned off by default. 1: Enable TIMER1 clock 0: Disable TIMER1 clock
[5]	TIMER0_CLK_EN	TIMER0 clock control. It is turned off by default. 1: Enable TIMER0 clock 0: Disable TIMER0 clock
[4]	MCPWM_CLK_EN	MCPWM clock control. It is turned off by default. 1: Enable MCPWM clock 0: Disable MCPWM clock

[3]	CMP_CLK_EN	CMP clock control. It is turned off by default.
		1: Enable CMP clock
		0: Disable CMP clock
[2]	UART_CLK_EN	UART clock control. It is turned off by default.
		1: Enable UART clock
		0: Disable UART clock
[1]	HALL_CLK_EN	HALL clock control. It is turned off by default.
		1: Enable HALL clock
		0: Disable HALL clock
[0]	I2C_CLK_EN	I2C clock control. It is turned off by default.
		1: Enable I2C clock
		0: Disable I2C clock

Note that the clock of each module mentioned above is the working clock of its internal circuit. Even if the clock of the respective module is not turned on, it will not affect the software's access to the register of the module.

6.3.8 SYS_SFT_RST Soft Reset Register

Address: 0x4000_00A4

Reset value: 0x0

Table 6-12 Soft Reset Register SYS_SFT_RST

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				DMA_SFT_RST	SPI_SFT_RST	ADC_SFT_RST	DSP_SFT_RST	GPIO_SFT_RST	TIMER1_SFT_RST	TIMER0_SFT_RST	MCPWM_SFT_RST	CMP_SFT_RST	UART_SFT_RST	HALL_SFT_RST	I2C_SFT_RST
				WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
				0	0	0	0	0	0	0	0	0	0	0	0

Location	Bit Name	Description
[31:12]		Unused
[11]	DMA_ SFT_RST	DMA soft reset. It is not reset by default
		1: Reset DMA module
		0: Release DMA module
[10]	SPI_ SFT_RST	SPI soft reset. It is not reset by default
		1: Reset SPI module
		0: Release SPI module
[9]	ADC_ SFT_RST	ADC soft reset. It is not reset by default
		1: Reset ADC module
		0: Release ADC module
[8]	DSP_ SFT_RST	DSP soft reset. It is not reset by default

		1: Reset DSP module
		0: Release DSP module
[7]	GPIO_ SFT_RST	GPIO soft reset. It is not reset by default
		1: Reset GPIO module
		0: Release GPIO module
[6]	TIMER1_ SFT_RST	TIMER1 soft reset. It is not reset by default
		1: Reset TIMER1 module
		0: Release TIMER1 module
[5]	TIMER0_ SFT_RST	TIMER0 soft reset. It is not reset by default
		1: Reset TIMER0 module
		0: Release TIMER0 module
[4]	MCPWM_ SFT_RST	MCPWM soft reset. It is not reset by default
		1: Reset MCPWM module
		0: Release MCPWM module
[3]	CMP_ SFT_RST	CMP soft reset. It is not reset by default
		1: Reset CMP module
		0: Release CMP module
[2]	UART_SFT_RST	UART soft reset. It is not reset by default
		1: Reset UART module
		0: Release UART module
[1]	HALL_SFT_RST	HALL soft reset. It is not reset by default
		1: Reset HALL module
		0: Release HALL module
[0]	I2C_SFT_RST	I2C soft reset. It is not reset by default
		1: Reset I2C module
		0: Release I2C 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.

After resetting the ADC module, the register inside the ADC will be reset, and the ADC initialization function needs to be used again to read DC/AMC and other calibration parameters.

6.3.9 SYS_PROTECT/SYS_WR_PROTECT Write Protection Register

Address: 0x4000_00A8

Reset value: 0x0

Table 6-13 Write Protection Register SYS_PROTECT/SYS_WR_PROTECT

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

Location	Bit Name	Description

[31:16]		Unused
[15:0]	PSW	Except for SYS_AFE_DAC, SYS_AFE_DAC_AMC, and SYS_AFE_DAC_DC, other registers (registers starting with SYS_, including clock reset management and analog 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.

6.3.10   SYS_FLSE Erase Protection Register

Address: 0x4000_00D0

Reset value: 0x0

Table 6-1 Erase Protection Register SYS_FLSE

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

Location	Bit Name	Description
[31:16]		Unused
[15:0]	FLSE	Flash erase protection register. Write 0x8FCA to this register to disable flash erase protection, write other value to enable protection. Flash CANNOT be erased when erase protection is enabled.

6.3.11   SYS_FLSP Program Protection Register

Address: 0x4000_00D4

Reset value: 0x0

Table 6-2 Program Protection Register SYS_FLSP

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

Location	Bit Name	Description
[31:16]		Unused
[15:0]	FLSP	Flash program protection register. Write 0x8F35 to this register to disable program protection, write other value to enable protection. Flash CANNOT be programmed when program protection is enabled.

7 Non-volatile Memory

7.1 Introduction

The non-volatile memory includes two parts: FLASH and ROM.

The FLASH memory includes MAIN. The main flash memory area (MAIN) includes application programs and user data area. MAIN is 16kB or 32kB.

The ROM memory of 16KB is fixed before leaving factory.

For LKS32MC03x series chips, the non-volatile memory includes three models:

- Model 1: 32KB FLASH

- Model 2: 16KB FLASH, 16KB ROM

- Model 3: 16KB FLASH
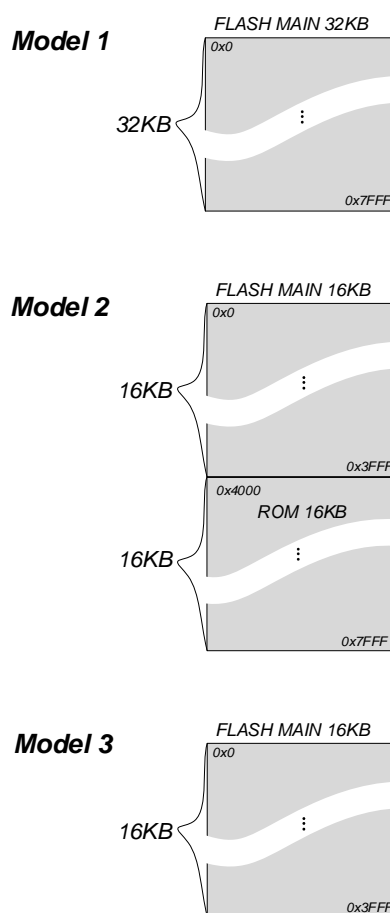
Fig. 7-1 Block Diagram of Non-volatile Memory Space Division and Models

7.2 **Features**

The non-volatile memory controller module mainly implements the related operations on the FLASH memory, and the ROM memory contains only execute operations, including:

➢ FLASH reading.

➢ FLASH writing.

➢ FLASH erase, including FULL erase and SECTOR erase.

➢ FLASH deep sleep, reducing the sleep power consumption of the chip.

➢ FLASH memory content encryption.

➢ FLASH reading acceleration, which improves the overall operation efficiency of the chip.

➢ FLASH control register access.

➢ ROM area cannot only execute operations but cannot be copied.

7.2.1Functional Description

The non-volatile memory controller implements operations such as reset/read/write/erase/sleep of FLASH memory. The following is the state transition diagram of the control module:
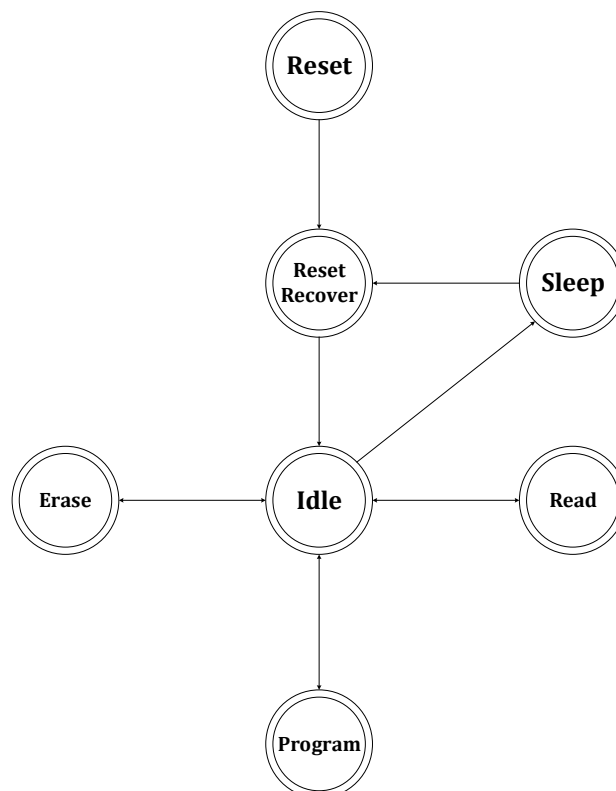
Fig. 7-2 FLASH Control State Transform Diagram

7.2.1.1   Reset

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

7.2.1.2   Sleep

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

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

7.2.1.3   Read

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

- The MCU fetches instructions and accesses data directly on the FLASH through the AHB bus. The fetch width is 32bit, and can only access data in the MAIN space. The hardware also provides an acceleration function to speed up the MCU to fetch instructions and access data.

- The MCU accesses the register of the controller through the AHB bus to read FLASH internal data indirectly. If a continuous reading is required, the hardware will accumulate the addresses automatically without updating the address register value each time.

- The MCU fetches instructions and accesses data directly on the ROM through the AHB bus. The fetch width is 32bit.

The process for reading the internal data indirectly on FLASH by accessing the register of the controller is as follows:
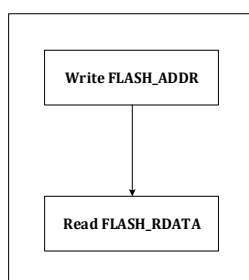
Fig. 7-3 Flow Chart of FLASH Indirect Read

7.2.1.4   FLASH Programming

FLASH Programming refers to programming operations on the FLASH memory bank. Generally, an erase operation should be performed before data programming. And, the programming can only be performed by accessing the registers of the FLASH controller. The specific process is:

- Write 0x8F35 to SYS_FLSP register, programming enable switch 1

- Write 1 to FLASH_CFG.PRG_EN, programming enable switch 2

- Write the programming address to FLASH_ADDR

- Write the programming data to FLASH_WDATA

The process for FLASH programming by accessing the register of this controller is as follows:
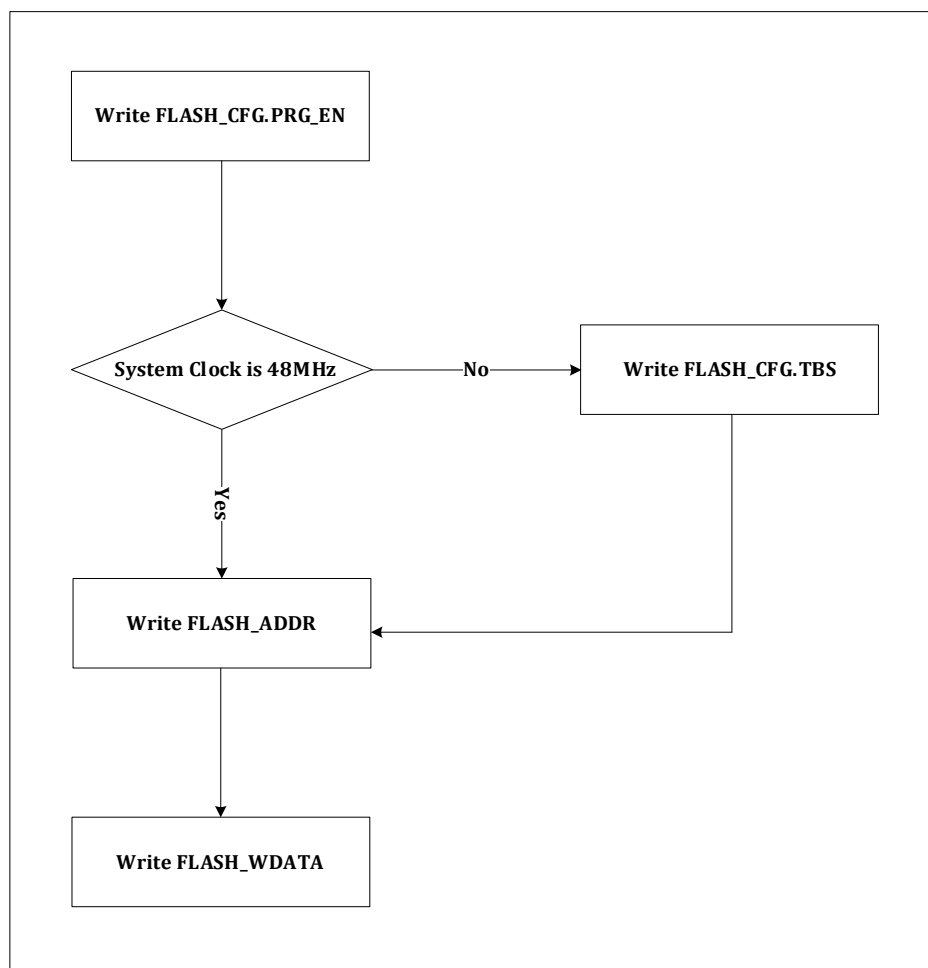
Fig. 7-4 Flow Chart of FLASH Programming

In order to prevent the FLASH zone from being programmed by mistake, an additional programming enable switch - SYS_FLSP is added. Write 0x8F35 to enable programming, and then configure FLASH_CFG.PRG_EN to enable FLASH programming.

For the selection of operating frequency, please refer to the configuration of SYS_CLK_CFG. The absolute time of the FLASH write/erase operation is fixed, and the count value corresponding to these absolute times should be saved in the FLASH controller. The default value of FLASH_CFG.TBS is the count value at 48MHz clock frequency; when the chip works at other frequencies, the value of FLASH_CFG.TBS should be set to achieve the count values of 24MHz and 12MHz (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 FLASH_CFG.TBS values at different frequencies, please refer to FLASH_CFG Register. Please note that only the values provided in the register description could be set to the FLASH_CFG.TBS, and other values are not available for writing into; otherwise, it may cause FLASH programming/erasing failure. It is recommended to read first on the FLASH_CFG, and then

38

perform other operations by follow the OR/AND method. Besides, the CPU will stop temporarily when the FLASH program/erase operation is performing until the operation is finished.

Fig. 7-4 only shows the flow of one programming. If continuous programming is required, set the FLASH_CFG.ADR_INC before writing to the FLASH_ADDR register to enable the address auto-increment mode. After then, repeatedly write into the FLASH_WDATA register, and the address will be added by 0x4 automatically each time when writing data into the FLASH_ADDR. The operation of continuous reading is similar to this. The continuous programming process is as follows:
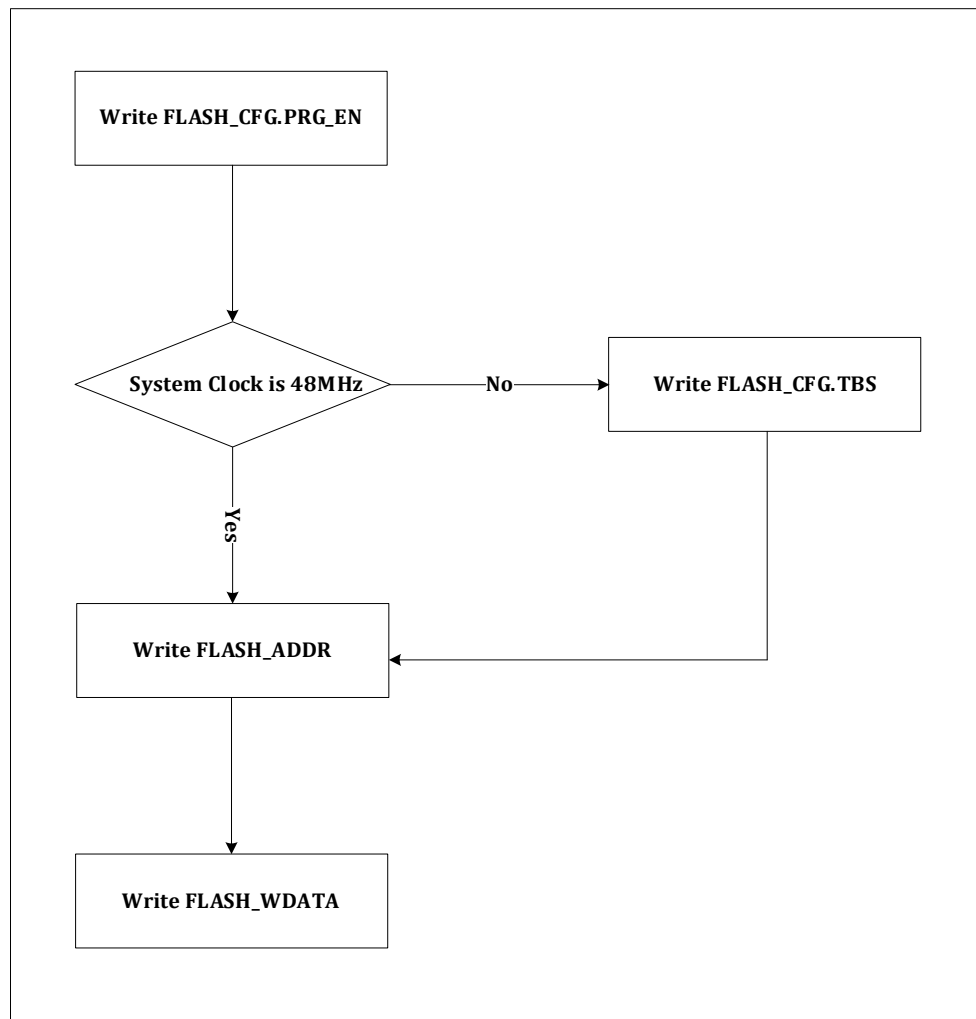
Fig. 7-5 Flow Chart of FLASH Programming

Note that ROM area only support one programming. But the MAIN area has no restriction. It is recommended to use the library function for programming in MAIN area.Off-line programmers designated by Linko are recommended for programming in ROM area.

7.2.1.5   FLASH Erase

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

● Write 0x8FCA to SYS_FLSE register, erase enable switch 1

● Write 1 to FLASH_CFG.ERS_EN, erase enable switch 2

● Write erase address to FLASH_ADDR

● Write FLASH_ERASE to trigger erase

In order to prevent the FLASH zone from being erased by mistake, an additional erase enable switch - SYS_FLSE is added. Write SYS_FLSE to protect FLASH from being erased by mistake, and write 0x8FCA to enable erase, and then configure FLASH_CFG.ERS_EN to enable FLASH erase.

Perform flash erase on the FLASH memory bank. The erase operation is divided into Sector erasure and FullChip erasure, corresponding to 512Byte erasure and 16KB/32KB erasure respectively. Which type of erase operation is performed can be determined by setting the FLASH control register.

The following table is the address allocation space of Sector (16KB version, no Sector32-Sector63).

Table 7-1 FLASH Sector Address Allocation

Name	Addresses	Size(Bytes)
Sector 0	0x0000 0000 - 0x0000 01FF	512
Sector1	0x0000 0200 - 0x0000 03FF	512
Sector2	0x0000 0400 - 0x0000 05FF	512
…	…	…
Sector63	0x0000 7E00 - 0x0000 7FFF	512

The flash erase operation flow is shown below.
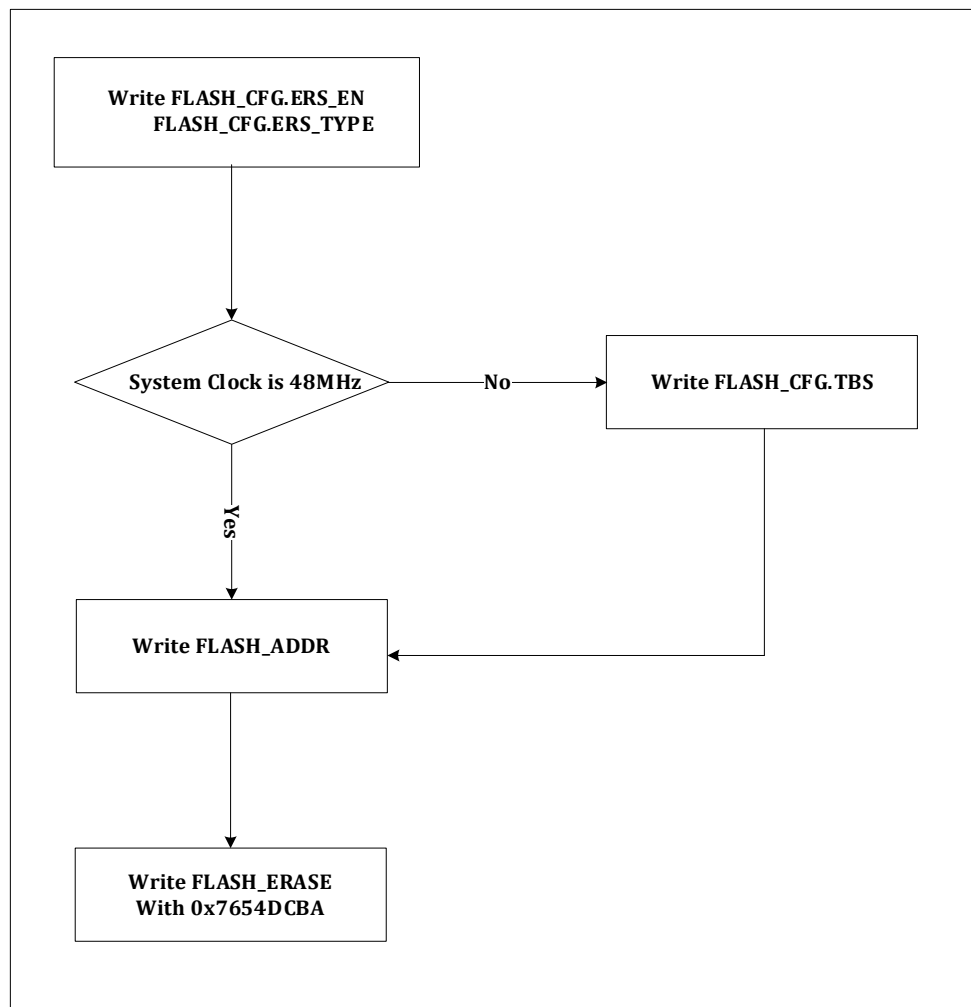
Fig. 7-6 Flow Chart of FLASH Erase

For Secotor erasure, determine which Secotor to be erased by FLASH_ADDR; For FullChip mode, the value of FLASH_ADDR will be invalid. Writing 0x7654DCBA to FLASH_ERASE, and triggers the erase operation.

Note that the ROM area do not support erasure operation. But the MAIN area has no restriction. It is recommended to use the library function for erasure in MAIN areas.

7.2.1.6   FLASH Prefetch

Due to the speed limitation of FLASH memory, this operation cannot reach the speed of 48MHz. When reading the FLASH, it takes more than 1 clock cycle (greater than 48MHz) to finish reading the data. In order to speed up the reading of data, the FLASH controller adds a prefetch function. After the FLASH controller finishes the current read operation, it will prefetch the data of the next WORD in turn without affecting the normal program execution. The prefetch operation can be turned on and off only by setting FLASH_CFG.PREF.

When the system frequency drops to 24MHz and below, the read operation of FLASH requires less than one clock cycle. At this time, you can configure FLASH_CFG.WAIT to 0 to save the extra wait. Remember to slow down the clock and change FLASH_CFG.WAIT to 0 when switching from high frequency to low frequency; change FLASH_CFG.WAIT to 1 and then adjust the clock faster when switching from low frequency to high frequency.

7.2.1.7   Non-volatile Memory Protection

The non-volatile memory encryption protection includes FLASH and ROM. As ROM is only executable and cannot be copied, it is under protection; this section mainly explains the protection of FLASH.

The purpose of FLASH protection is to prevent unauthorized access of FLASH content.

If the data in the FLASH memory is encrypted, users can decrypt the data in the FLASH memory. On the contrary, if the data in the FLASH memory is decrypted, users can encrypt the data in the FLASH memory. The data in the FLASH memory is encrypted by default. After the chip is powered on and reset, the hardware will perform an encryption status update automatically. Whether the data is encrypted or decrypted, it will remain unchanged after being updated.

FLASH memory has two specifications, 16kB and 32kB. Regardless of the specification, the last WORD in the corresponding specification is designed as an encrypted word (0x3FFC for 16KB, and 0x7FFC for 32KB). When the content of this WORD is written as all "1", it indicates that the FLASH content does not need encryption, read FLASH_PROTECT register to complete status update; when the content of this WORD is written as not-all "1", it indicates that the FLASH needs to be encrypted. If encryption is required, perform the programming of the last WORD, write a non-all "1" value, and read the FLASH_PROTECT register to trigger an encryption status update to finish encryption (reading the return value of FLASH_PROTECT has no reference significance).

There are two cases for the corresponding decryption process. If the last WORD has not been programmed to write a non-all "1" value, reading the FLASH_PROTECT register will finish the decryption update (regardless of the FLASH_PROTECT return value). If it has been programmed and written a non-all "1" value, decrypt by erasure operation. Firstly, perform an erase operation on FLASH, restore the last WORD to all "1" value, and then read the FLASH_PROTECT register to trigger an encryption status update and finish decryption (reading the return value of FLASH_PROTECT has no

reference significance).

7.2.1.8 FLASH Online Upgrade (IAP)

The IAP mode is used to implement remapping of the interrupt vector table. LKS32MC08X series chip contains register VTOR, which address is 0xE000_ED08, to remap the entry address of the interrupt vector table.

<p align="center">Table 7-2 Register Description of IAP VTOR</p>

Name	Reset Value	Offset	Location	Permission	Description
VTOR	0x0		[31:7]	RW	Perform write operation to write entry address of interrupt vector table

The default value is 0x0, and the entry address of the interrupt vector table is 0x0. When a non-zero value is written, the entry address of the interrupt vector table will be mapped to the address corresponding to the written value and take effect immediately.

Since the LKS32MC08X series chip has a VTOR register, users can update the entire FLASH content in situations. Besides, the interrupt operation can be turned on or off during online upgrades.

7.2.1.8.1 Start Interrupted Online Upgrade

Recommended software configuration process:

Turn off the interrupt controller of the CPU to stop receiving the new interrupt responses temporarily;

Place the interrupt processing function code at the new interrupt entry address;

Write the new interrupt entry address to the VTOR register;

Turn on the interrupt controller of the MCU to enable interrupts;

Jump to the online upgrade function to start the online upgrade;

Turn off the interrupt controller of the MCU after upgraded and set VTOR as the default value of 0;

Perform a MCU soft reset, and the PC restarts the upgraded program from address 0.

7.2.1.8.2 End Interrupted Online Upgrade

Turn off the interrupt controller of the MCU to stop receiving the new interrupt responses temporarily;

Jump to the online upgrade function to start the online upgrade; If the online upgrade uses UART-like peripheral communication, the MCU should poll the UART interrupt flag bit.

Perform a MCU soft reset, and the PC restarts the upgraded program from address 0.

7.2.1.8.3 Location of Online Upgrade Function

If the flash should be erased, place the online upgrade function in RAM; if an interrupt is required, place the new interrupt vector entry address in the RAM address space.

If only part of the flash area occupied by the application should be erased, place the online upgrade function in the free area of the high flash address, and then use the block to erase the old flash application and write the new application.

7.3   **Register**

7.3.1 Address Allocation

The base address of the FLASH controller module register is 0x0001_0000.

Table 7-3 List of FLASH Controller Register

Name	Offset	Description
FLASH_CFG	0x00	FLASH configuration register
FLASH_ADDR	0x04	Address register
FLASH_WDATA	0x08	Write data register
FLASH_RDATA	0x0C	Read data register
FLASH_ERASE	0x10	Erase enable register
FLASH_PROTECT	0x14	FLASH protection status register
FLASH_READY	0x18	FLASH free and busy status register

7.3.2 FLASH_CFG Configuration Register (Read back first, and then modify by the OR/AND form)

Address: 0x0001_0000

Reset value: 0xB0

Table 7-4 Configuration Register (FLASH_CFG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ERS_EN				PRG_EN				ADR_INC				PREF			
RW				RW				RW				RW			
0				0				0				0			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERS_TYPE				REGION				WAIT				TBS			
RW				RW				RW				RW			
0				0				1				30			

Location	Bit Name	Description
[31]	ERS_EN	FLASH erasure enable. The default value is 0. 0: Erasure off 1: Erasure on
[27]	PRG_EN	FLASH programming enable. The default value is 0. 0: Programming off 1: Programming on
[23]	ADR_INC	FLASH address increment enable. The default value is 0. 0: Address increment off 1: Address increment on

		When performing FLASH continuous read and write access, enable this function can reduce the operation of the address.
[19]	PREF	FLASH prefetch acceleration enable. The default value is 0. 0: Acceleration off 1: Acceleration on
[15]	ERS_TYPE	FLASH erasure type selection. The default value is 0. 0:Sector 1:FULL
[11]	REGION	Access FLASH area selection. The default value is 0. 0:MAIN
[7]	WAIT	Read FLASH data, WAIT switch. The default value is 1. 0: Read, without needing to wait for a cycle 1: Read, need to wait for a cycle
[5:0]	TBS	Program/erase time base register. The default value is 0x30. **Only the following values can be set:** 0x30: FLASH programming/erasing time base value at 48Mhz operating frequency. 0x17: FLASH programming/erasing time base value at 24Mhz operating frequency. 0x0B: FLASH programming/erasing time base value at 12Mhz operating frequency.

7.3.3 Address Register (FLASH_ADDR)

Address: 0x0001_0004

Reset value: 0x0

Table 7-5 Address Register (FLASH_ADDR)

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

Location	Bit Name	Description
[31:15]		Unused
[14:0]	ADDR	Address register. Address register corresponding to read/write/erase operation. The lowest two digits will be ignored by the FLASH controller because of the WORD operation. When performing the erase operation, the addresses should be aligned according to the erase type. One Sector is 512-Byte. If performing the Sector erase, the address should be an integer multiple of 512 (if offset, the offset will be ignored). If performing a full chip erase, the value of this register is not used for reference.

7.3.4 Write Register (FLASH_WDATA)

Address: 0x0001_0008

Reset value: 0x0

Table 7-6 Write Register (FLASH_WDATA)

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															

Location	Bit Name	Description
[31:0]	WDATA	Perform write operation to write the FLASH value

7.3.5 Read Register (FLASH_RDATA)

Address: 0x0001_000C

Reset value: 0x0

Table 7-7 Read Register (FLASH_RDATA)

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															

Location	Bit Name	Description
[31:0]	RDATA	Perform read operation to read the FLASH value

7.3.6 Erase Control Register (FLASH_ERASE)

Address: 0x0001_0010

Reset value: 0x0

Table 7-8 Erase Control Register (FLASH_ERASE)

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															

Location	Bit Name	Description
[31:0]	ERASE	Write 0x7654DCBA to trigger the erase operation

7.3.7 Encryption Status Register (FLASH_PROTECT)

Address: 0x0001_0014

Reset value: 0x0

Table 7-9 Encryption Status Register (FLASH_PROTECT)

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

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

Location	Bit Name	Description
[31:0]	PROTECT	Read this register to update the encryption/decryption status. Reading the return value has no reference significance.

7.3.8 Working Status Register (FLASH_READY)

Address: 0x0001_0018

Reset value: 0x0

Table 7-10 Working Status Register (FLASH_READY)

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

Location	Bit Name	Description

[31:1]		Unused
[0]	READY	1: FLASH is idle; 0: FLASH is busy

7.3.9NVR correction value address information

Calibration parameters of LKS32MC03x product memory chip:

Calibration parameters, each product is calibrated independently, and each product does not support the mixing of calibration parameters;

The calibration parameters shall be written before delivery. Programming and erasing are not supported after delivery. Only reading is supported;

Calibration parameters are read and accessed through library functions provided by LKS;

For calibration parameters, it is recommended to perform the access operation after the system interrupt is closed;

The lks32mc03x _ nvr. O file contains functions to read the calibration parameters:

Read function: uint32_t Read_Trim(uint32_t adr);

Table 7-11 Working Status Register (FLASH_READY)

地址	内容
0x0170	200K:10K Gear The high 16-bit stores the actual resistor value of R2 (note that the amplification is stored by 100 times). The low 16-bit stores the actual resistor value of R1 (note that the amplification is stored by 100 times).
0x0174	190K:20K Gear The high 16-bit stores the actual resistor value of R2 (note that the amplification is stored by 100 times). The low 16-bit stores the actual resistor value of R1 (note that the amplification is stored by 100 times).
0x0178	180K:30K Gear The high 16-bit stores the actual resistor value of R2 (note that the amplification is stored by 100 times). The low 16-bit stores the actual resistor value of R1 (note that the amplification is stored by 100 times).
0x017C	170K:40K Gear The high 16-bit stores the actual resistor value of R2 (note that the amplification is stored by 100 times). The low 16-bit stores the actual resistor value of R1 (note that the amplification is stored by 100 times).
0x0188	The AVDD voltage value of the chip with internal LDO power supply (note that it is stored

	after being amplified by 1000 times)
0x018C	The LKS03x chip with suffix C adds the 1.2 V range of the DAC, and the calibration parameters of the 1.2 V range are stored here. (Low 16-Bit is DC calibration value of 1.2 V gear, high 16-Bit is DAC GAIN calibration value of 1.2 V gear) Please use the library function to implement the exact call
0x0190	SYS_AFE_REG3 Calibration value
0x0194	SYS_AFE_REG4 Calibration value
0x0198	SYS_AFE_REG5 Calibration value(Low 16-Bit is 5.0 V calibration, High 16-Bit is 3.3 V calibration)
0x019C	SYS_AFE_REG6 Calibration value (Low 16-Bit is 5.0 V calibration, High 16-Bit is 3.3 V calibration)
0x01A0	ADC0_DC0 Calibration value，2/3 times gain
0x01A4	ADC0_AMC0 Calibration value，2/3 times gain
0x01A8	ADC0_DC1 Calibration value，1 times gain
0x01AC	ADC0_AMC1 Calibration value，1 times gain
0x01B0	OPA 200K Ohm VS 10K Ohm，GAIN Calibration value（Expand the results by 1000 times），R0 is 0 Ohm
0x01B4	OPA 190K Ohm VS 10K Ohm，GAIN Calibration value（Expand the results by 1000 times），R0 is 0 Ohm
0x01B8	OPA 180K Ohm VS 10K Ohm，GAIN Calibration value（Expand the results by 1000 times），R0 is 0 Ohm
0x01BC	OPA 170K Ohm VS 10K Ohm，GAIN Calibration value（Expand the results by 1000 times），R0 is 0 Ohm
0x01C0	DAC GAIN 校准值(A version chip, only 3.0 V gear, B version chip both) (The low 16-Bit is the 3.0 V gear calibration and the high 16-Bit is the 4.8 V gear calibration) Please use the library function to implement the exact call
0x01C4	DAC DC 校准值(A version chip, only 3.0 V gear, B version chip both) (The low 16-Bit is the 3.0 V gear calibration and the high 16-Bit is the 4.8 V gear calibration) Please use the library function to implement the exact call
0x01C8	Temperature sensor, slope calibration values
0x01CC	Temperature sensor, offset calibration values
0x01F4	High 16-bit storage OPA1 common-mode voltage value (Note that the magnification is 10000 times stored) Low 16-bit storage OPA0 common-mode voltage values (Note that the magnification is 10000 times stored)

8   SPI

8.1   **Introduction**

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.

8.2   **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, support DMA operation, including overflow detection and chip select signal anomaly detection

● The data length is adjustable from 8 bits to 16 bits

8.3   **Functional Description**

8.3.1 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.
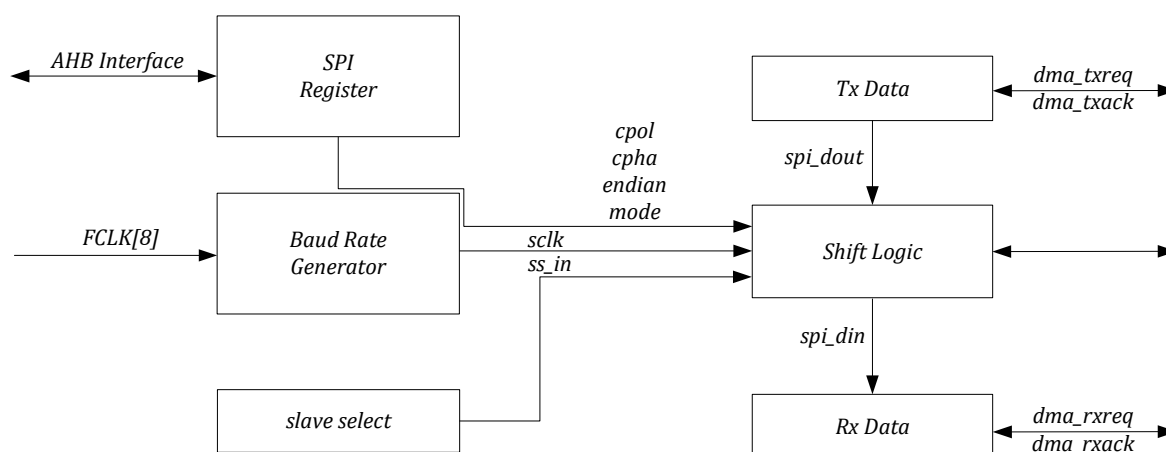
Fig. 8-1 SPI Module Structure Block Diagram

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.

and this signal input is invalid in non-Slave mode. 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.

8.3.2 Functional Description

8.3.2.1 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
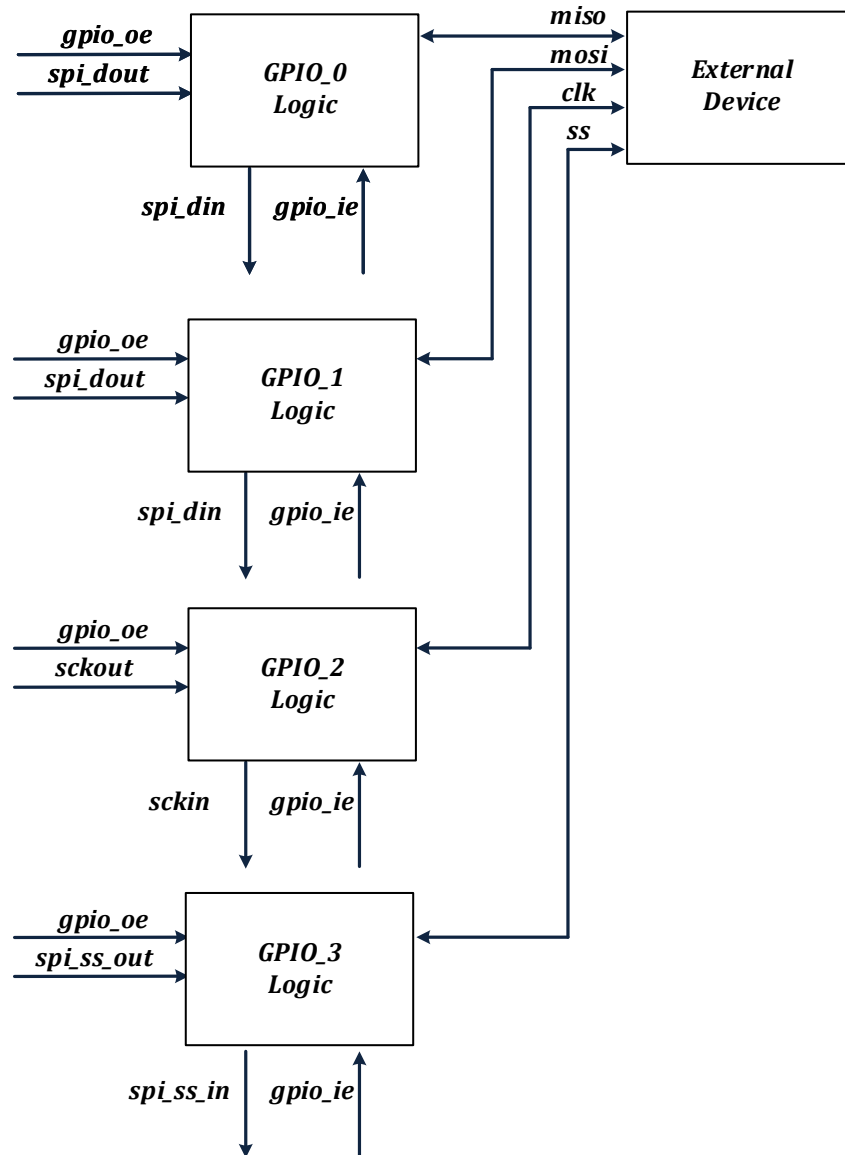
Fig. 8-2 SPI Interface Full Duplex Mode Interconnection Block Diagram

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.

8.3.2.2 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.
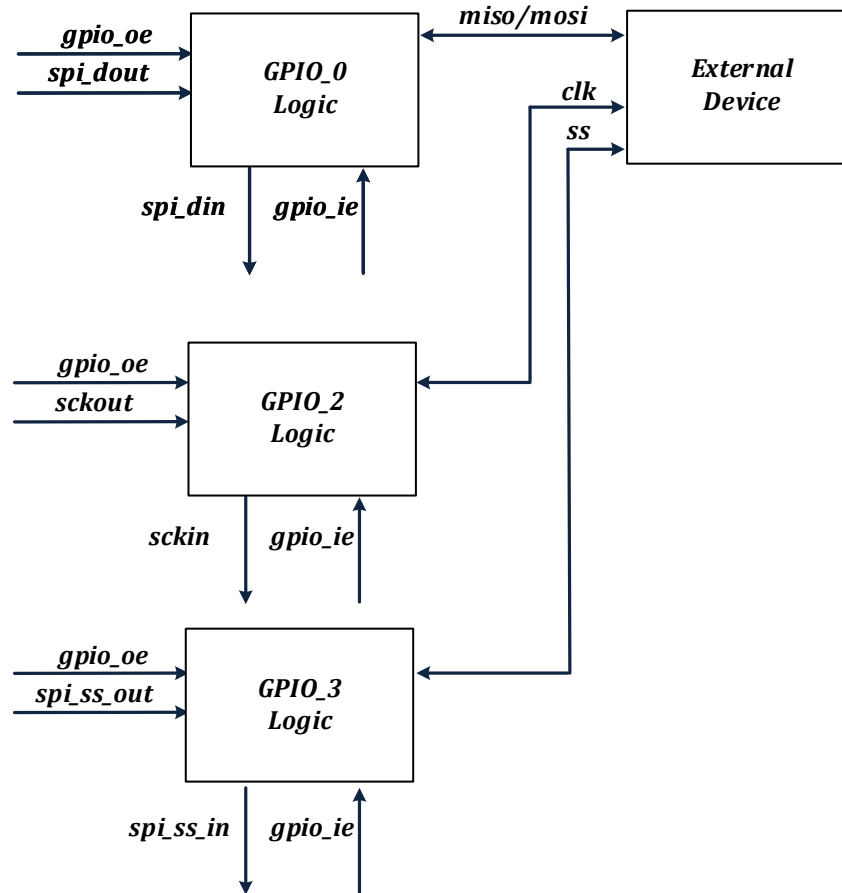
Fig. 8-3 SPI Interface Full Duplex Mode Interconnection Block Diagram

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.

8.3.2.3   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.
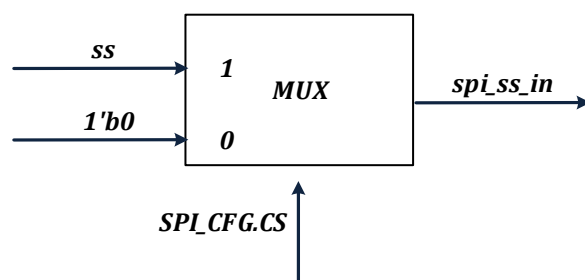
Fig. 8-4 SPI Module Chip Select Signal Selection in Slave Mode

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.
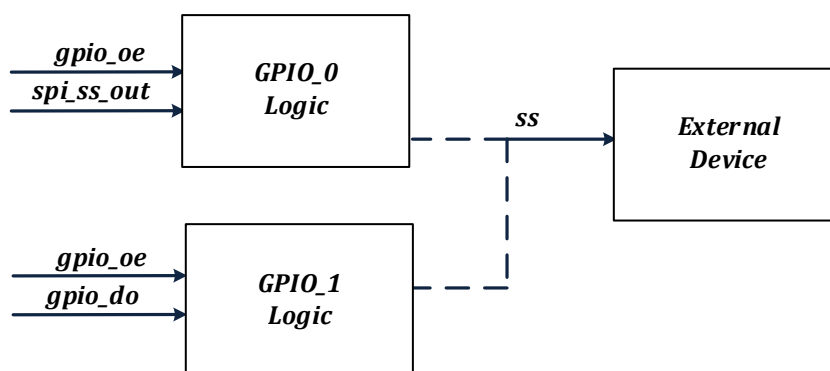
Fig. 8-5 SPI Module Chip Select Signal Selection in Master Mode

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.

8.3.2.4   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.

8.3.2.5   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.

8.3.2.6   DMA Transmission

In the application of large-capacity data transmission, the SPI interface supports DMA transmission, reducing the burden on the MCU. The maximum transmission volume of a transmission depends on the DMA module, and the minimum transmission volume is one byte. In full-duplex mode, DMA transmission can be achieved for both reception and transmission; in half-duplex mode, only DMA transmission can be achieved for reception or transmission.

After receiving the new data, the hardware generates a DMA request automatically and moves the data to RAM through the DMA module. Before sending new data, the hardware automatically generates a DMA request, and moves the data from the RAM to the SPI interface through the DMA module. Since SPI has no FIFO, SPI transmits a DMA request, and DMA can only move a byte of data. **To achieve multi-byte transmission, the DMA should be set for multiple rounds of transmission, with one byte transferred for each round.**

Corresponding register of the DMA module should be set for DMA transmission.

Since the SPI interface supports DMA transmission, it also supports MCU transmission. The difference between the two is that the data sent by DMA transmission comes from the movement of DMA; the data sent by MCU transmission comes from the movement of MCU.

The recommended software configuration process for DMA transmission is as follows:

- Initialize the DMA module, set the data source sent this time, the destination of the received data, and the transmission length.

- 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.

- Trigger the SPI interface and enter the send/receive state. The trigger condition is that the MCU performs a write operation to the TX_DATA register. Since the data finally sent is from the DMA, the data written by the MCU this time will not be mixed into the SPI transmission process.

8.3.2.7   MCU 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_TXDATA 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_TXDATA 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.

Note: If continuous transmission is required, SPI_TXDATA register should be set. The transmission completion information is obtained by interruption or polling.

8.3.2.8   External Event Transmission

Under the application of large-capacity data transmission, the SPI interface supports DMA transmission. During transmission, it may or may not be interrupted. The so-called interruption refers to the completion of the current byte transmission, and it needs to wait for an external event before starting the transmission of the next byte; the non-interruption refers to the completion of the current byte transmission and the transmission of the next byte. Interrupt mode should be used with other modules. For example, the timer module. The timer can trigger the transmission of the next byte. Interrupt mode is only valid in Master mode. SPI_IE.TRANS_TRIG controls whether to use interrupt mode.

8.3.2.9   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.
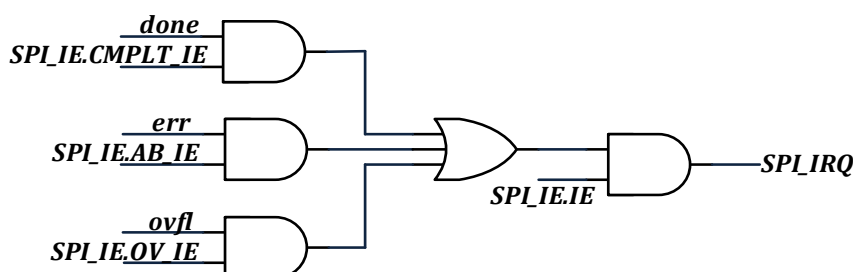
Fig. 8-6 Generation Diagram for SPI Module Interrupt Selection Signal

After the data transfer is completed, the end can be judged by DMA interrupt or SPI interrupt.

● In the transmission mode, the DMA first completes the moving operation, and the SPI transmits afterwards. After the SPI is sent, it can be used as the completion mark of this transmission.

● In the receiving mode, the SPI reception is completed, triggering the DMA move. The completion of DMA transfer can be used as the completion mark of this transfer.

Overflow event. In full-duplex mode, the DMAs sent and received are valid, and generally no overflow event will be sent; In half-duplex mode, there is only sending (or receiving), at this time the hardware blocks the receiving (or sending) overflow judgment.

8.3.2.10 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:

$$\text{SPI transmission baud rate = System clock}/(2*(BAUD + 1))$$

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.

8.4 **Register**

8.4.1 Address Allocation

The base address of the SPI module register is 0x4001_0000, and the register list is as follows:

Table 8-1 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_SIZE	0x14	SPI Transfer Data Length Register

8.4.2 System Control Register (SPI_CFG)

Address: 0x4001_0000

Reset value: 0x0

Table 8-2 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	1	0	0

Location	Bit Name	Description
[31:8]		Unused
[7:6]	DUPLEX	Half-duplex mode setting 00, 01: 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

8.4.3 Interrupt Register (SPI_IE SPI)

Address: 0x4001_0004

Reset value: 0x0

Table 8-3 Interrupt Register (SPI_IE SPI)

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

Location	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.

8.4.4 Baud Rate Setting Register (SPI_BAUD)

Address: 0x4001_0008

Reset value: 0x0

Table 8-4 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					

Location	Bit Name	Description
[31:9]		Unused
[7]	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.
[6]		Unused
[5:0]	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.

8.4.5 SPI Transmit Data Register (SPI_TXDATA)

Address: 0x4001_000C

Reset value: 0x0

Table 8-5 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								

Location	Bit Name	Description
[31:16]		Unused
[15:0]	TX_DATA	SPI Transmit Data Register

8.4.6 SPI Receive Data Register (SPI_RXDATA)

Address: 0x4001_0010

Reset value: 0x0

Table 8-6 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															

Location	Bit Name	Description
[31:16]		Unused
[15:0]	RX_DATA	SPI Receive Data Register

8.4.7 SPI Transfer Data Length Register (SPI_SIZE)

Address: 0x4001_0014

Reset value: 0x0

Table 8-7 SPI Transfer Data Length Register (SPI_SIZE)

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

Location	Bit Name	Description
[31:5]		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

9   I2C

9.1   **Introduction**

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.

9.2   **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.

● DMA with single byte buffer.

9.3   **Functional Description**

9.3.1 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.
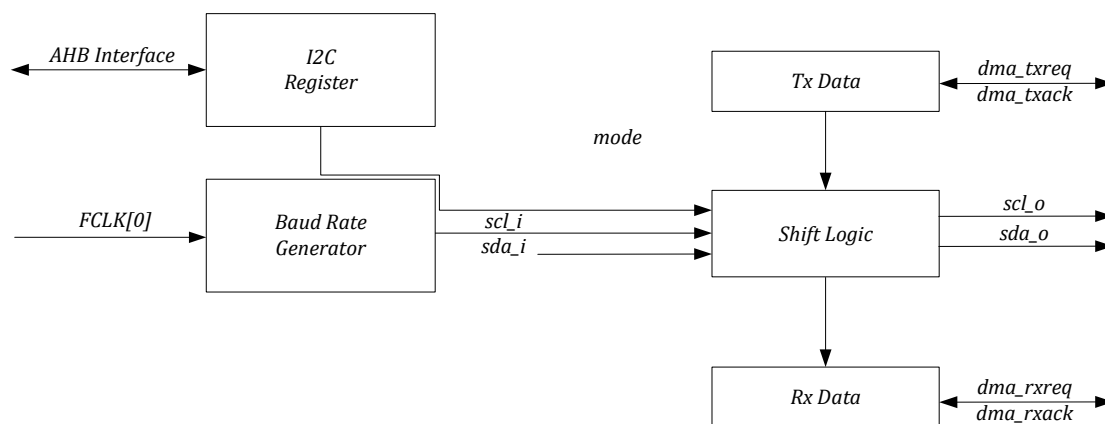
Fig. 9-1 I2C Module Top Functional Block Diagram

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.

9.3.2 Functional 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).

9.3.2.1   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.
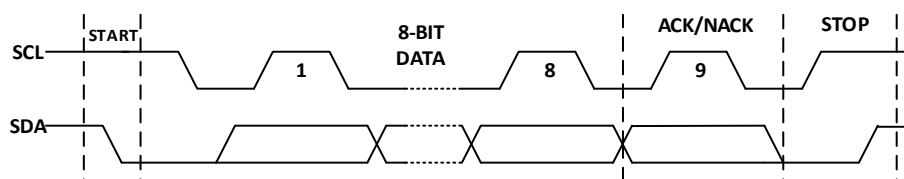
Fig. 9-2 Basic I2C Transmission Timing Diagram

The I2C interface has no FIFO. If a large amount of data is sent at once, DMA cooperation is required to reduce the burden on the MCU. The I2C interface supports DMA transfer (multi-byte transfer) and non-DMA transfer (single-byte transfer). The above four transmission modes are further expanded to:

- Single-byte transmission in slave mode, DMA transmission in slave mode

- Single-byte reception in slave mode, DMA reception in slave mode

- Single-byte transmission in main mode, DMA transmission in main mode

- Single-byte reception in master mode, DMA reception in master mode

In general, one byte at a time is transmitted in non-DMA mode (single transmission can be repeated, and the data should be provided by software). A continuous transmission can be multi-byte in DMA mode. The maximum is no more than 32 bytes. In extreme cases, one byte is transmitted at a time. Since there is no FIFO, only one byte is transmitted per DMA request, and the data transmission is completed in multiple rounds.

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.

- DMA transmission. Normally, an interrupt will be generated after the data is sent and the response is received (either ACK/NACK).

- DMA reception. Normally, an interrupt will be generated after the 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.

9.3.2.2   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 operating clock frequency of the I2C interface must be set by the register SYS_CLK_DIV0. The I2C interface clock is divided based on the system high-speed main clock, and SYS_CLK_DIV0 is the division factor of the I2C interface working clock.

- 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.

- Single-byte reception in slave mode. 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.

● Single-byte transmission in slave mode. 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.

● DMA reception in slave mode. Each time the data after the SIZE agreement is received, an interrupt is generated, and the I2C interface can pull the SCL low until the interrupt is completed.

● DMA transmission in slave mode. Each time the data after the SIZE agreement is sent and a response (ACK/NACK) is received, an interrupt is generated. The I2C interface can pull the SCL low until the interrupt is completed.

9.3.2.2.1 Slave Mode Transmission

Single-byte transmission does not mean that only one byte of data is transmitted. It means that after each byte of data is transmitted, an interrupt will be generated to determine whether to continue the transmission. The extreme case of single-byte transmission is to transmit only one byte of data. The following figure is a schematic diagram of the bus for single-byte transmission. As can be seen from the figure, the general single byte transmission process is as follows:

● Address match, generate address match interrupt, ready to start transmission.

● 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.

● Obtain the bus STOP event, this transmission is completed.
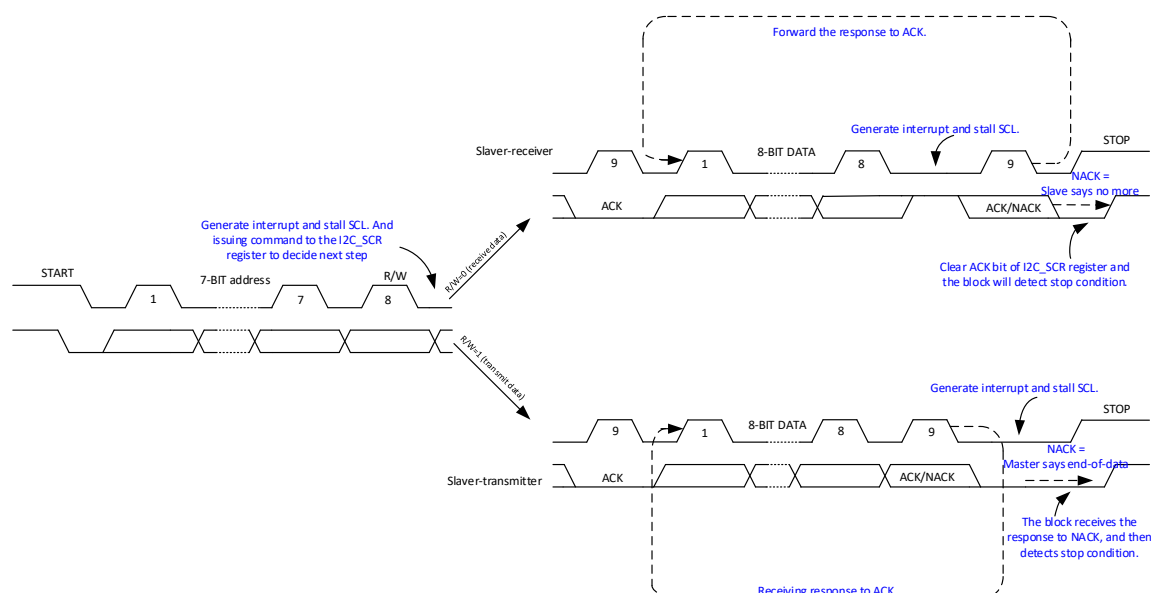
Fig. 9-3 Schematic Diagram of Transmission in Slave Mode

9.3.2.2.2 Slave Mode Transport

After the address matches, the slave sends the byte 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.

9.3.2.2.3 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.

9.3.2.3   Slave Mode DMA Transmission

Generally, only I2C clock, slave address and hardware address matching are enabled in slave mode. After waiting for an access request from the bus, determine whether to respond to this transmission request in situations. DMA transfer, which means that after each transfer of multiple bytes of data, an interrupt will be generated to determine whether to continue the transfer. The extreme case of DMA transfer is to transfer only one byte of data. Hardware address comparison function, NACK interrupt, and transfer completion interrupt are recommended to be enabled for DMA transmission. The general DMA transfer process is as follows:

● Set I2C slave address, enable I2C interrupt (enable hardware address comparison interrupt). Address matching, generate I2C address matching interrupt, set DMA in the interrupt processing function, ready to send data or ready to receive address. Then write I2C_SCR, ready to start transmission or end this transmission.

● In RX mode, an interrupt is generated after I2C_BCR.BURST_SIZE agreed byte is received, the software determines whether to continue receiving, and returns an ACK/NACK response.

● In TX mode, wait for the response (ACK/NACK) after sending the agreed byte of I2C_BCR.BURST_SIZE, an interrupt is generated after receiving the response, and the subsequent operation is judged based on the response.

● Obtain the bus completion flag, this transmission is completed.

9.3.2.3.1 Slave Mode DMA Transport

After the addresses match, the DMA is configured. Send a DMA request to move the byte from RAM to the I2C_DATA register, and then send it to the SDA line via the internal shift register. Before the I2C_DATA data is ready, the slave device can pull the SCL until the data to be sent has been written to the I2C_DATA register. Sending data in slave mode requires software assistance to trigger the first DMA move, and set I2C_BCR.BYTE_CMPLT to 1. The I2C interface performs the following operations after sending

the byte data agreed by I2C_BCR.BURST_SIZE:

● If the ACK bit is received, set the DMA, prepare for the next batch of data, and continue the transmission. SCL can be lowered during preparation.

● If the NACK bit is received, stop preparing the next batch of data and stop this transmission.
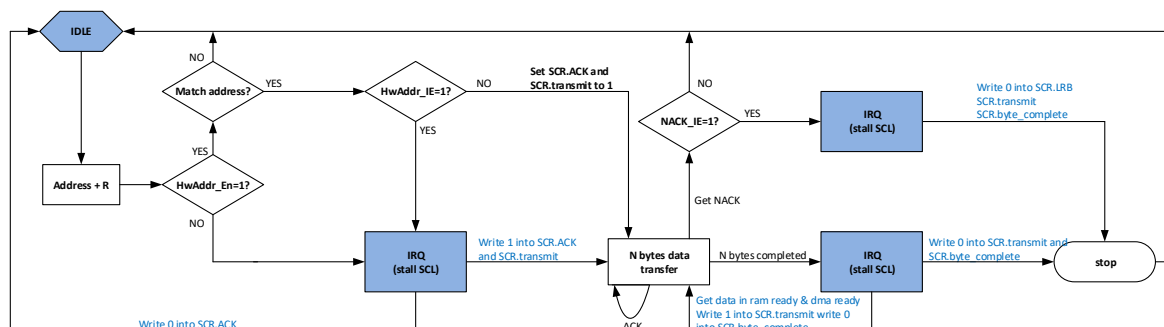
Fig. 9-4 Schematic Diagram of Multi-byte Transmission in Slave Mode

9.3.2.3.2 Slave Mode DMA Receive

After the addresses match, the DMA is configured, and the data received from the SDA line is stored in the I2C_DATA register, and then moved to the RAM through the DMA. The I2C interface will execute a DMA request after receiving a byte. After completing the data transfer agreed by I2C_BCR.BURST_SIZE, perform the following operations:

● If the ACK bit is set, an ACK response pulse is generated after I2C_BCR.BURST_SIZE agreed data is received.

● If the ACK bit is cleared, a NACK response pulse is generated after the I2C_BCR.BURST_SIZE agreed data is received.
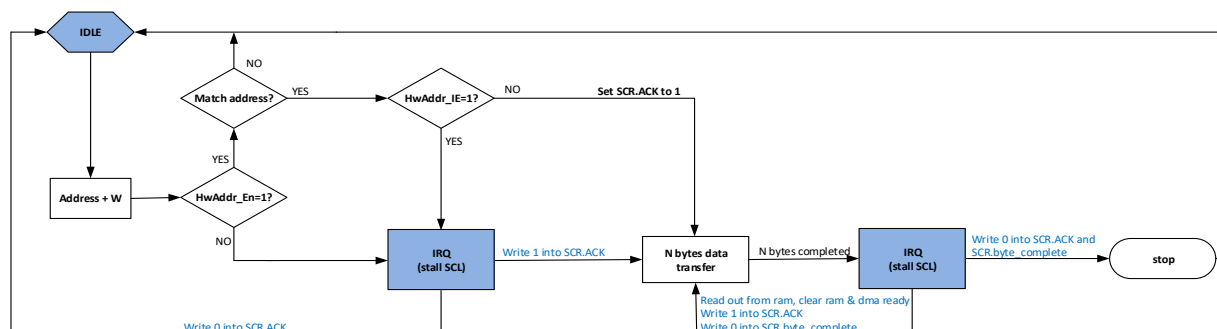
Fig. 9-5 Schematic Diagram of Multi-byte Reception in Slave Mode

9.3.2.4 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.

9.3.2.4.1 Master Mode Single-byte Transmission

Single-byte transmission does not mean that only one byte of data is transmitted. It means that after each byte of data is transmitted, an interrupt will be generated to determine whether to continue the transmission. The extreme case of single-byte transmission is to transmit only one byte of data. Fig. 6 is a schematic diagram of a bus for single-byte transmission. As can be seen from the figure, the general single byte transmission process is as follows:

● Determine whether the bus is idle, if it is idle, prepare to start transmission.

● 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.
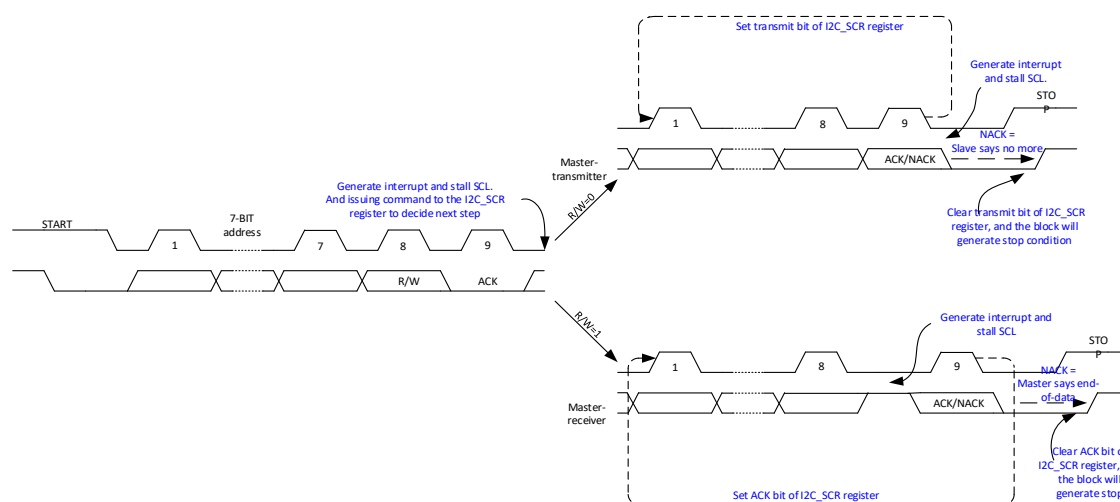
Fig. 9-6 Schematic Diagram of Single-byte Transmission in Master Mode

9.3.2.4.2 Master Mode Single-byte 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. 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.

● A STOP event is generated to end this transmission.

9.3.2.4.3 Master Mode Single-byte 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. 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.

- A STOP event is generated to end this transmission.

9.3.2.4.4 Master Mode DMA Transmission

DMA transfer, which means that after each transfer of multiple bytes of data, an interrupt will be generated to determine whether to continue the transfer. The extreme case of DMA transfer is to transfer only one byte of data. NACK interrupt and the transmission completion interrupt are recommended to be enabled for DMA transmission. The general DMA transfer process is as follows:

- The bus is idle and ready to start transmission.

- In RX mode, an interrupt is generated after I2C_BCR.BURST_SIZE agreed byte is received, the software determines whether to continue receiving, and returns an ACK/NACK response.

- In TX mode, wait for the response (ACK/NACK) after sending the agreed byte of I2C_BCR.BURST_SIZE, an interrupt is generated after receiving the response, and the subsequent operation is judged based on the response.

- Send STOP event, this transmission is completed.

9.3.2.4.5 Master Mode DMA Transport

The bus is idle and the DMA is configured. Send a DMA request to move the byte from RAM to the I2C_DATA register, and then send it to the SDA line via the internal shift register. Before the I2C_DATA data is ready, the master device may not generate the SCL clock until the data to be sent has been written to the I2C_DATA register. The I2C interface performs the following operations after sending the byte data agreed by SIZE:

- If the ACK bit is received, set the DMA, prepare for the next batch of data, and continue the transmission. SCL can be lowered during preparation.

- If the NACK bit is received, stop loading the next batch of data.

- If the data transmission is completed, stop the subsequent transmission.

- A STOP event is generated to end this transmission.

    The exception is:

- If the slave device address does not match or the slave device is not ready, the slave device will return NACK.

- The master device generates a STOP event to stop this transmission.

    After waiting for a period of time, reset the I2C register, turn off the channel enable signal

corresponding to the DMA, reset the DMA register, and send the transfer request again. The corresponding channel of DMA is closed since I2C has prefetch data, and DMA is not the initial state.
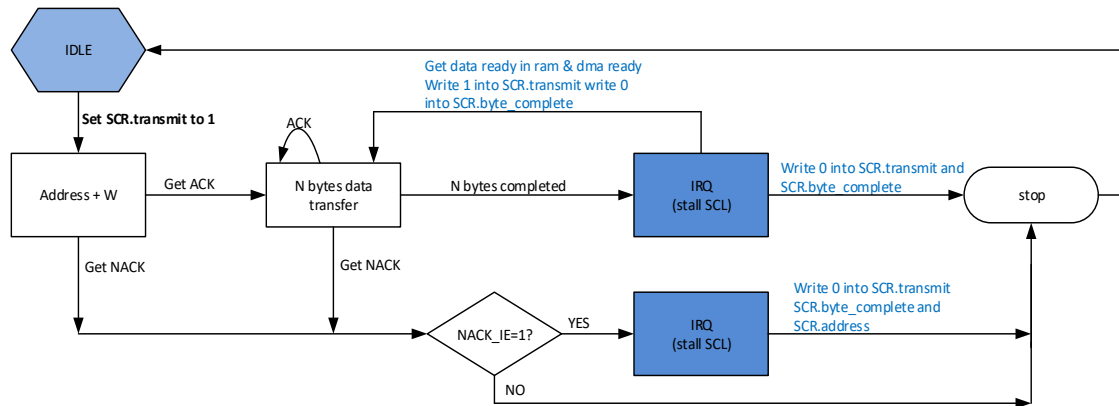
Fig. 9-7 Schematic Diagram of Multi-byte Transmission in Master Mode

9.3.2.4.6 Master Mode DMA Receive

The bus is idle, and the data received by the DMA from the SDA line is stored in the I2C_DATA register, and then moved to the RAM through DMA. The I2C interface will execute a DMA request after receiving a byte. After completing the data transfer agreed by I2C_BCR.BURST_SIZE, perform the following operations:

- If the ACK bit is set, an ACK response pulse is generated after I2C_BCR.BURST_SIZE agreed data is received.

- If the ACK bit is cleared, a NACK response pulse is generated after the I2C_BCR.BURST_SIZE agreed data is received.

- A STOP event is generated to end this transmission.

    The exception is:

- If the slave device address does not match or the slave device is not ready, the slave device will return NACK.

- The master device generates a STOP event to stop this transmission.

After waiting for a period of time, reset the I2C register and send the transfer request again. Since no valid data was received last time, DMA had no operation, so there is no need to reset DMA.
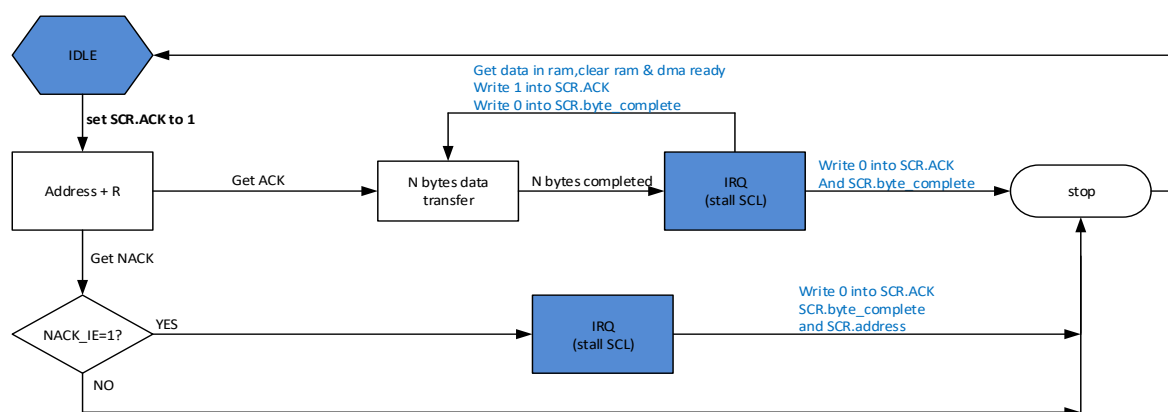
Fig. 9-8 Schematic Diagram of Multi-byte Reception in Master Mode

9.3.2.5   DMA Transmission

In the application of large-capacity data transmission, the I2C interface supports DMA transmission, reducing the burden on the MCU. The maximum transmission volume of a transmission is 16 bytes, and the minimum transmission volume is one byte. Since I2C has no FIFO, DMA can only move one byte of data after receiving the DMA request sent by I2C. To achieve multi-byte transmission, the DMA should be set for multiple rounds of transmission, with one byte transferred for each round.

After receiving the new data, the hardware generates a DMA request automatically and moves the data to RAM through the DMA module. Before sending new data, the hardware automatically generates a DMA request, and moves the data from the RAM to the I2C interface through the DMA module.

Corresponding register of the DMA module should be set for DMA transmission.

Since the I2C interface supports DMA transmission, it also supports MCU transmission. The difference between the two is that the data sent by DMA transmission comes from the movement of DMA; the data sent by MCU transmission comes from the movement of MCU.

The recommended software configuration process for DMA transmission is as follows:

● Initialize the DMA module, set the data source sent this time, the destination of the received data, and the transmission length.

● Initialize the GPIO module and set the GPIO multiplexed with I2C.

● Initialize the I2C interface, and set I2C_CFG/I2C_BCR/I2C_BSIZE and other registers.

● In master mode, trigger the I2C interface to enter the sending state; in slave mode, wait for the master to send a transmission request.

If the I2C interface is in slave mode for transmission, data prefetching should be considered, and I2C_BCR. SLV_DMA_PREF is the prefetching switch. Generally, the hardware address comparison (I2C_ADDR. ADDR_CMP) can be turned on during transmission in the slave mode, and choose whether to enable the hardware address comparison interrupt (I2C_BCR. BURST_ADDR_CMP).

● Turn on the hardware address comparison interrupt. After the address received from the device matches itself successfully, an interrupt will be generated. Inform the software that the master device requests the slave device data, and the software determines whether to receive it. If decide to receive, respond an ACK, and the software should set I2C_BCR. SLV_DMA_PREF to "1" to assist the

hardware in prefetching the first transmitted data; otherwise, it should respond a NACK.

Turn off the hardware address comparison interrupt. Once the START event occurs on the bus, the slave device hardware prefetches the first transmitted data, regardless of whether the slave device matches the address successfully. If matched successfully, the address match interrupt will not be generated and the data transmission will start immediately. If the match fails, the slave device will not transmit data. Since I2C has a prefetch operation, if the match fails, clear the DMA for the subsequent transmission if the next match is successful.

9.3.2.6 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.

9.3.2.7 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.

9.3.2.8 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. The clock frequency of the data and clock signals is: interface clock/17.

● I2C module working clock frequency = operating frequency/(CLK_DIV0+1)

● I2C baud rate = I2C module working clock frequency/17

9.4 **Register**

9.4.1 Address Allocation

The base address of the I2C module register is 0x4001_0100, and the register list is as follows:

Table 9-1 I2C Register Address Allocation List

Name	Offset	Description
I2C_ADDR	0x00	I2C address register
I2C_CFG	0x04	I2C configuration register
I2C_SCR	0x08	I2C status register
I2C_DATA	0x0C	I2C data register
I2C_MSCR	0x10	I2C master mode register
I2C_BCR	0x14	I2C DMA transfer control register
I2C_BSIZE	0x18	I2C transfer length register

9.4.2 Address Register (I2C_ADDR)

Address: 0x4001_0100

Reset value: 0x0

Table 9-2 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			

Location	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.

9.4.3 System Control Register (I2C_CFG)

Address: 0x4001_0104

Reset value: 0x0

Table 9-3 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

Location	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

9.4.4 Status Control Register (I2C_SCR)

Address: 0x4001_0108

Reset value: 0x0

Table 9-4 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_CM PLT	ADDR_DA TA	DATA_DIR	RX_ACK	Done
		RW	RW	RW	RW	RW	RW	RW	RW
		0	0	0	0	0	0	0	0

Location	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_AR B	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_CM PLT	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_DA TA	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.

9.4.5 Data Register (I2C_DATA)

Address: 0x4001_010C

Reset value: 0x0

Table 9-5 Data Register (I2C_DATA)

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

Location	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.

9.4.6 Main Mode Register (I2C_MSCR)

Address: 0x4001_0110

Reset value: 0x0

Table 9-6 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

Location	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.

9.4.7 I2C Transmission Control Register (I2C_BCR)

Address: 0x4001_0114

Reset value: 0x0

Table 9-7 I2C Transmission Control Register (I2C_BCR)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								BURST_NACK	BURST_ADDR_CMP	BUSRT_EN	SLV_DMA_PREF				
								RW	RW	RW	RW				
								0	0	0	0				

Location	Bit Name	Description
[31:8]		Unused
[7]	BURST_NACK	I2C transmission. NACK event interrupt enable signal. 1: enable this interrupt source 0: disable this interrupt source
[6]	BURST_ADDR_CMP	I2C transmission, hardware address matching interrupt enable signal. 1: enable this interrupt source 0: disable this interrupt source
[5]	BUSRT_EN	I2C multiple data transmission is enabled. DAM is required. 1: enable 0: disable
[4]	SLV_DMA_PREF	I2C multiple data transmission. Perform DMA transmission in slave mode, triggering the hardware to prefetch the first byte. The hardware is automatically cleared. 1: enable 0: disable

9.4.8 I2C Transfer Length Register (I2C_BSIZE)

Address: 0x4001_0118

Reset value: 0x0

Table 9-8 I2C Transfer Length Register (I2C_BSIZE)

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

Location	Bit Name	Description
[31:8]		Unused
[4:0]	SIZE	I2C data transmission length register. Used for multi-byte transmission. Actual bytes transferred = B [3: 0]+1

10 Comparator (CMP)

10.1 Introduction

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.

CMP can be used for the following functions:

1. Compare the zero-crossing point of the back EMF

2. Hardware overcurrent detection

3. The source of the fail signal of MCPWM

CMP main features：

1. Each comparator has configurable plus and minus inputs used for flexible voltage selection:

    - Multi-channel GPIO pins

    - OPA output signal

    - OPA positive terminal output signal

    - 1.2V BANDGAP reference source

    - DAC output signal

2. Programmable comparison speed, programmable hysteresis voltage

3. The output signal can be filtered, and the filtering depth can be selected

4. Per-channel can generate CMP interrupt

CMP0/1 can use MCPWM's 4 P channels for windowing 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.

Recommended configuration process:

      1. Open the CMP analog switch

Under the default state, the comparator module is turned off. By configuring the SYS_AFE_REG0.CMPxPDN(x=0/1, which represents two comparators of CMP0 / CMP1), the comparator can be opened. Turn on the Bandgap before using the comparator module.

2. Open the digital clock switch and signal input switch, select the clock filter coefficient

Turn on the digital clock switch of the CMP by configuring CMP_TCLK.CLK10_EN, 1 means opening, 0 means closed. Select the filter clock frequency division by configuring CMP_TCLK.FIL_CLK10_DIV16[7:4]. The range of numerical settings is 0 ~ 15, based on MCLK to divide by 1 to 16. By configuring CMP_CFG.CMPx_IN_EN to enable the signal input, 1 means enable, 0 means disable.

3. Configure the comparison speed and hysteresis of CMP

The comparison speed of the comparator is < 30ns/200ns by configuring SYS_AFE_REG1.CMP_FT. The comparator hysteresis can be set to 20mV/0mV by configuring SYS_AFE_REG1.CMP_HYS.

4. Select the positive and negative signal source of CMP

The positive signal of CMP has 8 signal sources to choose from. It can be set by configuring SYS_AFE_REG1.CMPx_SELP. There are 4 signal sources of the negative terminal to choose from, which can be set by configuring SYS_AFE_REG1.CMPx_SELN.

5. Configure the interrupt of CMP

By configuring CMP_IE.CMPx_IE to enable the CMP interrupt, 1 means enable, 0 means disable.

10.2 **Register**

10.2.1 Address Allocation

The base address of the CMP module register is 0x4001_0200, and the register list is as follows:

Table 10-1 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_BLCWIN	0x10	Comparator window control register 0
CMP_DATA	0x14	Comparator output value register

10.2.2 Comparator Interrupt Enable Register (CMP_IE)

Address: 0x4001_0200

Reset value: 0x0

Table 10-2 Comparator Interrupt Enable Register (CMP_IE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						CMP1_RE	CMP0_RE							CMP1_IE	CMP0_IE
						RW	RW							RW	RW
						0	0							0	0

Location	Bit Name	Description
[31:10]		Unused
[9]	CMP1_RE	Comparator 1 request enable, active high
[8]	CMP0_RE	Comparator 0 request enable, active high
[7:2]		Unused
[1]	CMP1_IE	Comparator 1 interrupt enable, active high
[0]	CMP0_IE	Comparator 0 interrupt enable, active high

10.2.3   Comparator Interrupt Flag Register (CMP_IF)

Address: 0x4001_0204

Reset value: 0x0

Table 10-3 Comparator Interrupt Flag Register (CMP_IF)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														CMP1_IF	CMP0_IF
														RW	RW
														0	0

Location	Bit Name	Description
[31:2]		Unused
[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

When CMP DMA request is enabled, i.e. CMP_IE.CMPx_RE=1, DMA automatically clears the corresponding CMPx_IF bits when the DMA receives the request and begins to move data. No software cleanup is required.

10.2.4   Comparator Divider Clock Control Register (CMP_TCLK)

Address: 0x4001_3008

Reset value: 0x0

Table 10-4 Comparator Divider Clock Control Register (CMP_TCLK)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										FIL_CLK10_DIV16		CLK10_EN		FIL_CLK10_DIV2	
										RW		RW		RW	
										0		0		0	

Location	Bit Name	Description
[31:8]		Unused
[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 0:1 frequency division 1:2 frequency division 2:4 frequency division 3:8 frequency division 4:16 frequency division 5:32 frequency division 6:64 frequency division 7:128 frequency division

CMP filter clock frequency is calculated according to the following formula

freq(CMP_Filter) = freq(MCLK)/2CMP_TCLK.FIL_CLK_DIV2/(CMP_TCLK.FIL_CLK_DIV16+1),

where FCLK[3] is the module clock controlled by SYS_CLK_FEN[3]. Note that the CMP_TCLK.CLK_EN bit should be enabled to generate the CMP filter clock.
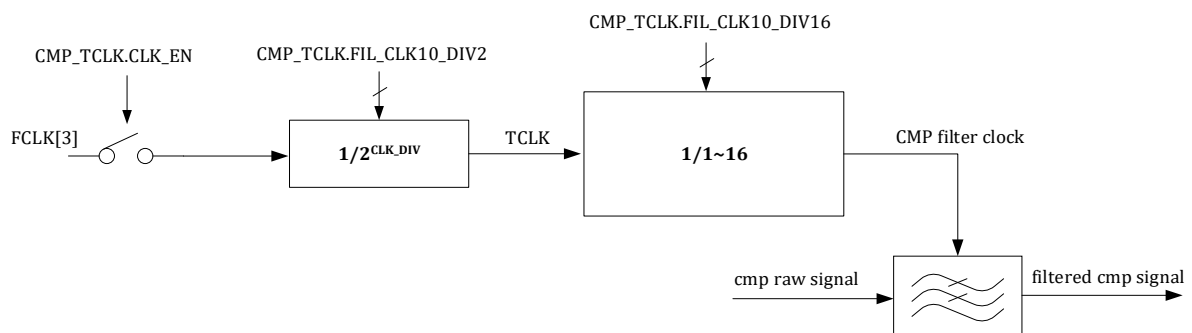
Fig. 10-1 Comparator Filter Clock Generation

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 system master clock cycles.

10.2.5   Comparator Control Register (CMP_CFG)

Address: 0x4001_020C

Reset value: 0x0

Table 10-5 Comparator Control Register (CMP_CFG)

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

Location	Bit Name	Description
[31:8]		Unused
[7]	CMP1_W_PWM_POL	Comparator 1 window PWM signal polarity selection, used when CMP_BLCWIN1 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 are as shown in Fig. 102.
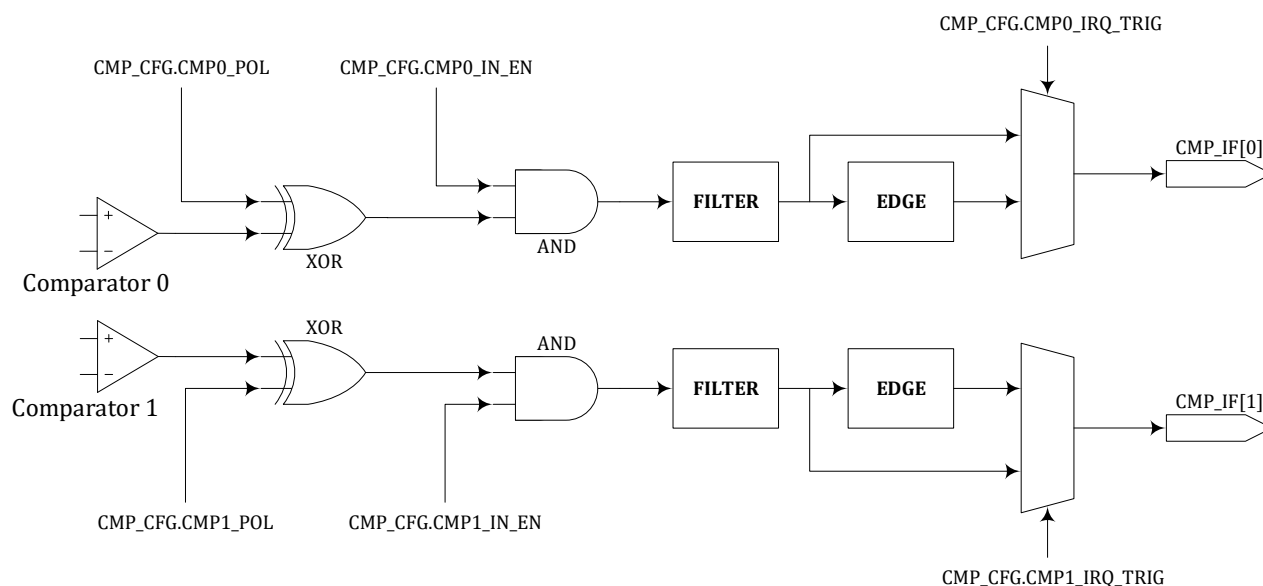
Fig. 10-2 Comparator Control and Interrupt Generation Logic

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 windowing. 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.
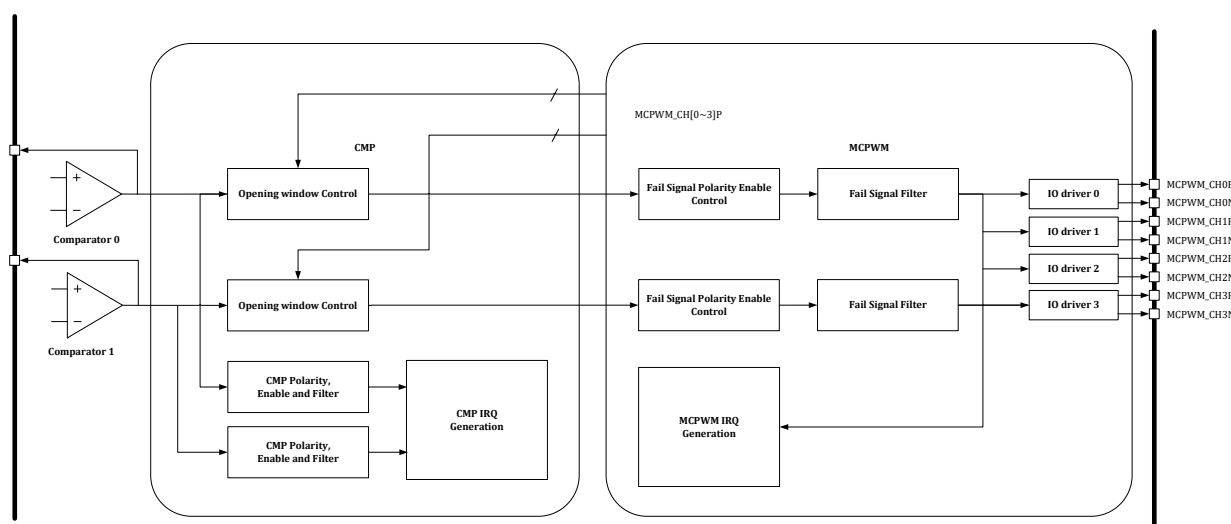
Fig. 10-3 CMP and MCPWM Linkage

For the windowing 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. The window control signal polarity of the comparator 1 is controlled by the CMP_CFG.CMP1_PWM_POL bit, and the logic is the same.

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 Fig. 104.
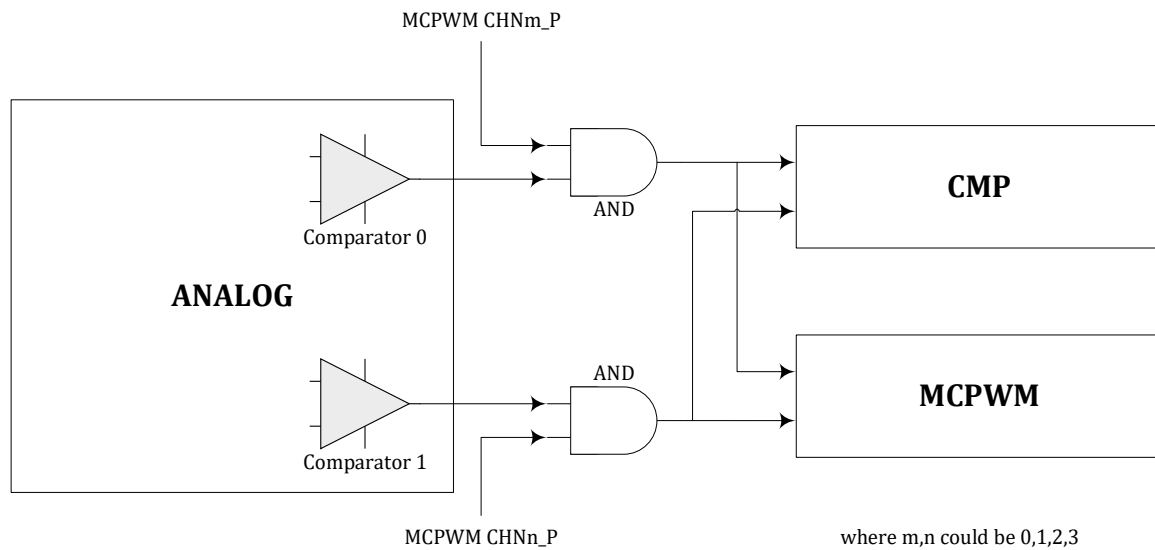
Fig. 10-4 Comparator Window Function Diagram

10.2.6   Comparator Window Control Register (CMP_BLCWIN)

Address: 0x4001_3010

Reset value: 0x0

Table 10-6 Comparator Window Control Register (CMP_BLCWIN)

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

Location	Bit Name	Description
[31:8]		Reserved
[7]	CMP1_CHN3P_WI	Use the P-tube switch control signal output from the CHN3_P channel

	N_EN	of the MCPWM 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 MCPWM 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 MCPWM 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 MCPWM 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 MCPWM 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 MCPWM 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 MCPWM 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 MCPWM module as the comparator 0 window enable

10.2.7   Comparator Output Value Register (CMP_DATA)

Address: 0x4001_3014

Reset value: 0x0

Table 10-7 Comparator Output Value Register (CMP_DATA)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						CMP1_FLT_DATA	CMP0_FLT_DATA							CMP1_RAW_DATA	CMP0_RAW_DATA
						R	R							R	R
						0	0							0	0

Location	Bit Name	Description
[31:10]		Unused
[9]	CMP1_FLT_DATA	Comparator 1 filtered signal
[8]	CMP0_FLT_DATA	Comparator 0 filtered signal
[2:7]		Unused
[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

83

11 HALL Signal Processing Module

11.1 Introduction

The chip supports three 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.

11.2 Implementation Description

11.2.1 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.

11.2.2 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.

11.2.3 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:
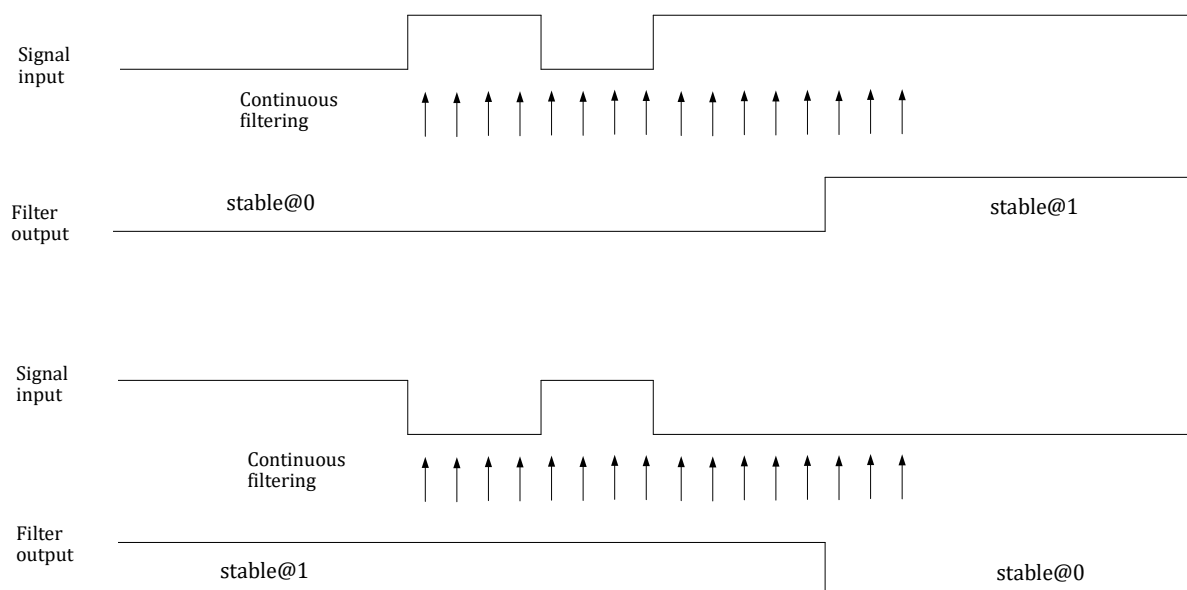
Fig. 11-1 7/5 Filter Module Block Diagram

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:

$$T_{filter} = T_{clk} * (HALL\_CFG.FIL\_LEN[14:0] + 1)$$

For example, at a 48MHz operating frequency, the period Tclk is 20.8.4ns, the maximum register configuration is 32767, and the longest filter width is about 20.8ns × 32768≈680us.

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, see 11.3.3 for details.

11.2.4   Capture

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 2.8 seconds at a 48MHz operating frequency, and achieve a time resolution of 20.8ns.

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.

11.2.5   Interrupt

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 terminal flag can be cleared by writing 1 to HALL_INFO.CHG_IF and HALL_INFO.OV_IF.

HALL interrupt event can be used as DMA request, but the CHG_RE, OV_RE or SW_RE needs to be enabled in HALL_CFG. After DMA receives a transfer request and begins moving data, the interrupt flag will be cleared. Software cleanup is not required.

11.2.6   Data Flow

The data flow of the HALL module is shown in the figure below. FCLK [1] is the HALL clock controlled by the SYS_CLK_FEN.HALL_CLK_EN gate control, which is usually a 48MHz PLL clock.
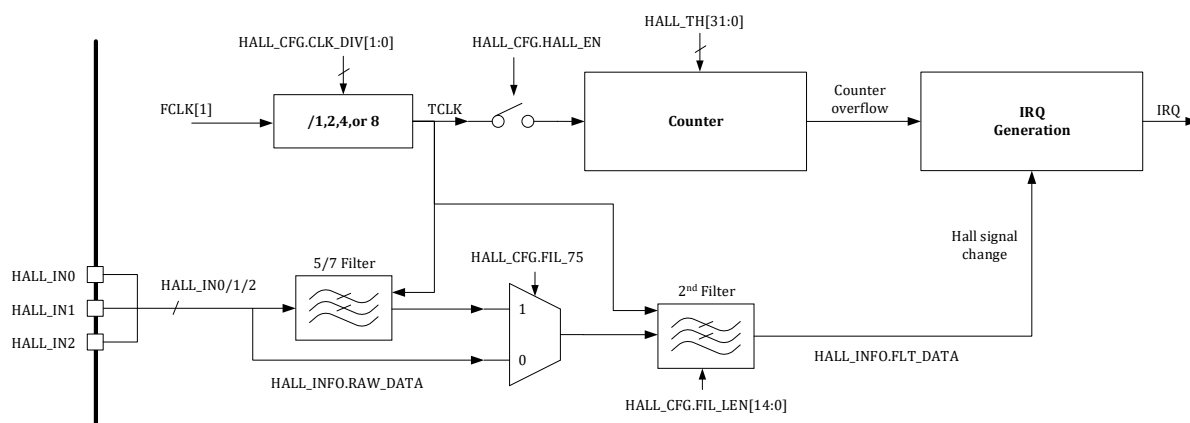
Fig. 11-2 Data Flow Diagram

11.3   **Register**

11.3.1   Address Allocation

The base address of the HALL module register is 0x4001_0300, and the register list is as follows:

Table 11-1 Module Register Address Allocation

Name	Offset	Description
HALL_CFG	0x00	HALL module configuration register
HALL_INFO	0x04	HALL module information register
HALL_WIDTH	0x08	HALL width count value register
HALL_TH	0x0C	HALL module counter threshold register
HALL_CNT	0x10	HALL count register

11.3.2   HALL Module Configuration Register (HALL_CFG)

Address: 0x4001_0300

Reset value: 0x0

Table 11-2 HALL Module Configuration Register (HALL_CFG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	SW_IE	OV_IE	CHG_IE		OV_RE	CHG_RE	HALL_EN				FIL_75			CLK_DIV	
	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
	FIL_LEN														
	RW														
	0														

Location	Bit Name	Description
[31]		Unused
[30]	SW_IE	Software-triggered HALL signal change interrupt enable. Active high. When SW_IE=1, writing 1 to HALL_INFO.SW_IF, a HALL signal change interrupt will be generated manually, and HALL_INFO.CHG_IF will be reset.
[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]		Reserved
[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.

11.3.3   HALL Module Information Register (HALL_INFO)

Address: 0x4001_0304

Reset value: 0x0

Table 11-3 HALL Module Information Register (HALL_INFO)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
													SW_IF	OV_IF	CHG_IF
													WO	RW1C	RW1C
													0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					RAW_DATA			Reserved					FLT_DATA		
RW					RO			RW					RO		
0					0			0					0		

Location	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

11.3.4   HALL Width Count Value Register (HALL_WIDTH)

Address: 0x4001_0308

Reset value: 0x0

Table 11-4 HALL Width Count Value Register (HALL_WIDTH)

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

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

CAP_CNT
RO
0

Location	Bit Name	Description
[31:24]		Unused
[23:0]	CAP_CNT	HALL width counter value

11.3.5　HALL Module Counter Threshold Register (HALL_TH)

Address: 0x4001_030C

Reset value: 0x0

Table 11-5 HALL Module Counter Threshold Register (HALL_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							

Location	Bit Name	Description
[31:24]		Unused
[23:0]	TH	HALL counter threshold

11.3.6　HALL Count Register (HALL_CNT)

Address: 0x4001_0310

Reset value: 0x0

Table 11-6 HALL Count Register (HALL_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

Location	Bit Name	Description
[31:24]		Unused
[23:0]	CNT	HALL count value. Write any value to clear

12 ADC

12.1 **Introduction**

The LKS32MC03X series chip integrates a 12-bit SARADC.

➢ 1MSPS sampling rate and 24MHz working frequency.

➢ Support 14 channels.

➢ Support software and hardware trigger

➢ Cooperate with MCPWM and UTimer unit to trigger ADC sample.

➢ Indication signal can be transmitted through GPIO for debugging.

➢ Support sampling with custom sampling sequences, such as single-round, double-round, and four-round, and the sequence number and channel number can be set flexibly.

➢ Support left and right alignment mode.

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 channels 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. Taking the four-round sampling triggered by MCPWM as an example, each round is sampled three times (complete three analog samples). TADC [0] triggers the ADC to start the first round of sampling. After the first round of sampling is completed, the ADC enters the waiting state and waits for the trigger event of TADC [1]; after TADC [1] occurs, trigger the ADC to start the second round of sampling; in the same way, TADC [2]/TADC [3] triggers the sampling of the third and fourth rounds respectively.

One round of sampling: may contain one, two or four rounds of sampling. Each round is triggered by a specific trigger signal; after completing one round of sampling, the ADC returns to the idle state and waits for the next trigger.

12.1.1 Functional Block Diagram

The ADC interface includes 10 data registers and several control registers.

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 12.2.4 for details). Taking ADC_CHN0 as an example. BIT[3:0] selects the analog channel of 0th sampling, and the channel number can be selected from CH0 to CH15. If ADC_CHN0 [3:0] = 0 and ADC_CHN0 [7:4] = 3, the analog value of the 0th sampling corresponds to channel CH0, the analog value of the 1st sampling corresponds to channel CH3, and so on.

The sampling times register ADC_CHNT controls the number of samples per round, and "1-15"

means "1-15" samples.

The control logic selects the trigger signal from MCPWM or the universal timer 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 TADC [0]/TADC[1]/TADC[2]/TADC[3], which can be selected as the trigger signal. The optional trigger signals are stored in the control register.

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. In the multi-round trigger mode, the conversion completion of each round can trigger one conversion done interruption.
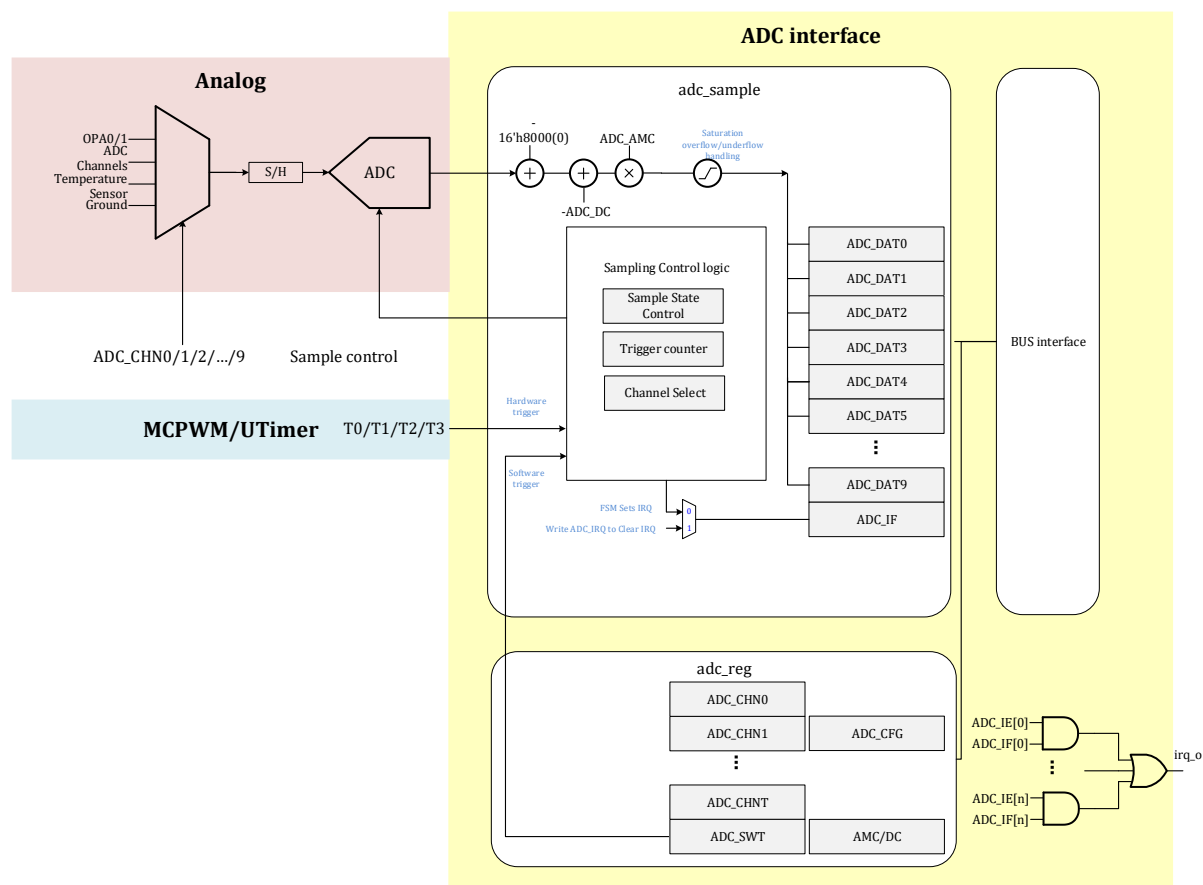
Fig. 12-1 Functional Block Diagram of ADC Sampling

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.

The maximum working clock of ADC is 24MHz. The sampling time can be configured as 5-37 ADC clock cycles. The data conversion time is fixed at 12 cycles. Taking the sampling time of 5 cycles as an example, a minimum of 17 cycles are required for sampling and conversion, which means that the maximum ADC conversion rate is 24/17=1.4Msps. However, considering the time of sampling establishment, it is recommended to set the sampling time to 8Cycle or above, with a corresponding conversion speed rate of 24/(8+12)=1.2Msps. By configuring register SYS_ AFE_ REG2 can set the sampling time.

12.1.2   ADC Trigger Mode

➢ Support single-round trigger, double-round trigger, four-round trigger to complete sampling

➢ Single-round trigger can set the number of trigger events

➢ The trigger source of the double-round trigger can only be the timing signal TADC[0]+TADC[1] of MCPWM/UTimer, or the twice software triggers.

➢ The trigger source of the four-round trigger can only be the timing signal TADC[0]+TADC[1]+TADC[2]+TADC[3] of MCPWM/UTimer which should happen in sequence, or four-round software triggers.

➢ Set to generate an interrupt after each trigger is completed.

➢ Trigger indication signal can be transmitted through GPIO for debugging

12.1.3   ADC Output Digital System

The ADC output data is 12-bit complement. The input signal 0 corresponds to 12h'0000_0000_0000. Taking the 2.4V range as an example, the input signal -2.4V corresponds to 12h'1000_0000_0000, and the input signal+2.4V corresponds to 12h'0111_1111_1111. The 12-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 12'h1000_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'h1000_0000_1101_0000; if the configuration is right-aligned, sign expansion is performed on the left, and the value stored in ADCx_DAT is 16'h1111_1000_0000_1101. Left alignment is recommended.

Table 12-1 Conversion of ADC Output Digital System

ADC 2.4V Range Analog Input Value/V	ADC 3.6V Range Analog Input Value/V	Converted Signed Number
2.4	3.6	12'h0111_1111_1111
0	0	12'h0000_0000_0000
-2.4	-3.6	12'h1000_0000_0000
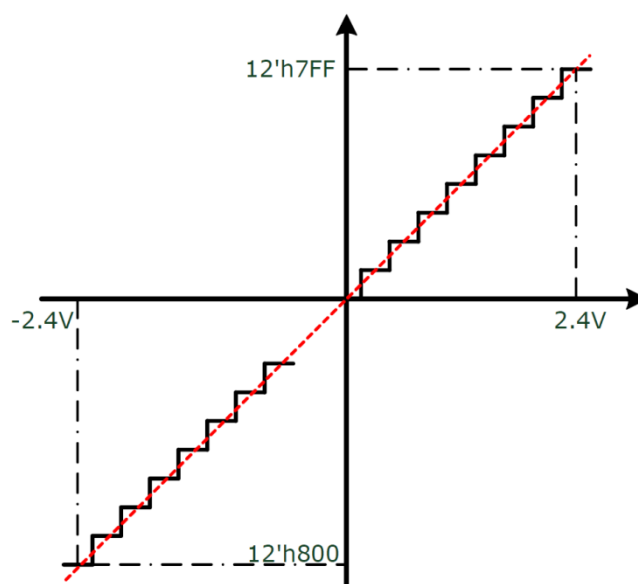
Fig. 12-2 ADC 2.4V range conversion curve

12.1.4   ADC Range

The ADC has two ranges: 2.4V and 3.6V. In 2.4V range mode, the maximum input signal amplitude is ±2.4V; in 3.6V range mode, the maximum input signal amplitude is ±3.6V.

When the ADC sampling channel is set as the output signal of the OPA (i.e. OPA0 ~ OPA1), 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 3.6V range. 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 (see 5.2.5 for gain selection method), 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 +/-2.4V due to objective reasons, the range of the ADC should be set to 2.4V.

When the ADC sampling channel is set as the input signal of the GPIO multiplex port, the ADC range 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.

The range selection is controlled by the SYS_AFE_REG0.GA_AD register, which is different from LKS32MC08x/05x, range can only be selected by software, cannot be automatically switched according to the channel configuration.

12.1.5   ADC Calibration

The ADC hardware interface module can perform DC offset calibration and gain calibration.

The AMP$_{correction}$, 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 = Saturation$((D_{ADC}-D_0)*AMP_{correction}-DC_{offset})$

Finally, the hardware will store the corrected D into the corresponding sampling data register.

12.1.6   ADC Threshold Monitoring (analog watchdog)

ADC interface module is equipped with 1 set 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. 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 inserted to the bit.

The threshold is a 12-bit signed number, so the threshold comparison is independent of the left and right alignment of the data. In left alignment, the ADCx_DATx[15:4] is compared with the threshold; and in right alignment, ADCx_DATx[11:0] is compared with the threshold.

12.1.7   ADC configuration process

Recommended configuration process:

1. Turn on the ADC analog switch and select the ADC operating frequency

    Turn on the ADC by setting SYS_AFE_REG0.ADCPDN to 1. Before turning on ADC, the BGP module, 4MHz RC oscillator and PLL should be turned on first.

2. Configure ADC data output format

    The output format of the ADC can be left or right alignment by setting the ADC_CFG.DATA_ALIGN. 0 means left alignment, 1 means right alignment.

3. Configure ADC sampling mode

    The sampling mode of the ADC can be selected by setting the ADC_CFG.TRG_MODE[9:8]. 00 means single-stage trigger, 01 means double-stage trigger, 11 means four-stage trigger.

4. Configure ADC trigger events

    The trigger event of ADC sampling can be selected by setting the ADC_CFG.SEL. In single-segment sampling mode, the ADC_CFG.SINGLE_TCNT[7:4] can be set to select the number of events required to trigger one sampling. The setting range is 0~15. 0 means that one event triggers one sampling, and 15 means that 16 events trigger one sampling.

5. Configure ADC Range

The gain mode can be selected by configuring the SYS_AFE_REG0.GA_AD. 0 means low gain (2/3 times), 1 means high gain (1 times).

6. Configure the number of ADC channels and select the sampling signal source

Setting ADC_CHNT can select the number of channels to be sampled in each sampling mode. The setting range is 1~16, and 1 represents one channel. The ADC_CHN0、ADC_CHN1、ADC_CHN2 are configured to select the sampling signal source of the ADC, and the setting range is 0~15.

7. Configuring ADC Interrupts

The ADC interrupts can be enabled by configuring the ADC_IE register. Even if interrupts are not enabled, interrupt events can still set ADC_IF, but no interrupt request will be raised.

12.2 Register

12.2.1   Address Allocation

The base address of ADC in chip is 0x4001_0400.

Table 12-2 ADC0 Register List

Name	Offset Address	Description
ADC_DAT0	0x00	ADC 0th sample data
ADC_DAT1	0x04	ADC 1st sample data
ADC_DAT2	0x08	ADC 2nd sample data
ADC_DAT3	0x0C	ADC 3rd sample data
ADC_DAT4	0x10	ADC 4th sample data
ADC_DAT5	0x14	ADC 5th sample data
ADC_DAT6	0x18	ADC 6th sample data
ADC_DAT7	0x1C	ADC 7th sample data
ADC_DAT8	0x20	ADC 8th sample data
ADC_DAT9	0x24	ADC 9th sample data
ADC_LTH	0x30	ADC analog watchdog lower threshold
ADC_HTH	0x34	ADC analog watchdog upper threshold
ADC_GEN	0x38	ADC analog watchdog monitoring enable
ADC_CHN0	0x40	ADC 0th-3rd sampling time selection
ADC_CHN1	0x44	ADC 4th-7th sampling time selection
ADC_CHN2	0x48	ADC 8th-11th sampling time selection
	0x4C	
ADC_CHNT	0x50	Number of ADC sampling channels in various trigger modes
ADC_CFG	0x54	ADC alignment mode configuration
ADC_SWT	0x58	ADC software trigger
ADC_DC	0x60	ADC DC offset
ADC_AMC	0x64	ADC gain correction
ADC_IE	0x68	ADC interrupt enable

ADC_IF	0x6C	ADC interrupt flag

12.2.2 Sampling Data Register

12.2.2.1 ADC_DAT0

地址:0x4001_0400

Reset value: 0x0

Table 12-3 Sampling Data Register (ADC_DAT0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT0															
RW															
0															

Location	Bit Name	Description
[31:16]		Unused
[15:0]	DAT0	0th sampling data

12.2.2.2 ADC_DAT1

Address: 0x4001_0404

Reset value: 0x0

Table 12-4 Sampling Data Register (ADC_DAT1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT1															
RW															
0															

Location	Bit Name	Description
[31:16]		Unused
[15:0]	DAT1	1st sampling data

12.2.2.3 ADC_DAT2

Address: 0x4001_0408

Reset value: 0x0

Table 12-5 Sampling Data Register (ADC_DAT2)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT2															
RW															
0															

Location	Bit Name	Description
[31:16]		Unused
[15:0]	DAT2	2nd sampling data

12.2.2.4 ADC_DAT3

Address: 0x4001_040C

Reset value: 0x0

Table 12-6 Sampling Data Register (ADC_DAT3)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT3															
RW															
0															

Location	Bit Name	Description
[31:16]		Unused
[15:0]	DAT3	3rd sampling data

12.2.2.5 ADC_DAT4

Address: 0x4001_0410

Reset value: 0x0

Table 12-7 Sampling Data Register (ADC_DAT4)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT4															
RW															
0															

Location	Bit Name	Description
[31:16]		Unused
[15:0]	DAT4	4th sampling data

12.2.2.6 ADC_DAT5

Address: 0x4001_0414

Reset value: 0x0

Table 12-8 Sampling Data Register (ADC_DAT5)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT5															
RW															

0	

Location	Bit Name	Description
[31:16]		Unused
[15:0]	DAT5	5th sampling data

12.2.2.7 ADC_DAT6

Address: 0x4001_0418

Reset value: 0x0

Table 12-9 Sampling Data Register (ADC_DAT6)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT6															
RW															
0															

Location	Bit Name	Description
[31:16]		Unused
[15:0]	DAT6	6th sampling data

12.2.2.8 ADC_DAT7

Address: 0x4001_041C

Reset value: 0x0

Table 12-10 Sampling Data Register (ADC_DAT7)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT7															
RW															
0															

Location	Bit Name	Description
[31:16]		Unused
[15:0]	DAT7	7th sampling data

12.2.2.9 ADC_DAT8

Address: 0x4001_0420

Reset value: 0x0

Table 12-11 Sampling Data Register (ADC_DAT8)

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

DAT8
RW
0

Location	Bit Name	Description
[31:16]		Unused
[15:0]	DAT8	8th sampling data

12.2.2.10 ADC_DAT9

Address: 0x4001_0424

Reset value: 0x0

Table 12-12 Sampling Data Register (ADC_DAT9)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT9															
RW															
0															

Location	Bit Name	Description
[31:16]		Unused
[15:0]	DAT9	9th sampling data

12.2.3 Analog Watchdog

12.2.3.1 ADC_LTH

Address: 0x4001_0430

Reset value: 0x0000_F800

Table 12-13 Lower Threshold Register (ADC_LTH)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Sign				LTH											
RO				RW											
0xF				0x800											

Location	Bit Name	Description
[31:16]		Unused
[15:12]		Sign extension
[11:0]	LTH	Lower threshold of ADC analog watchdog 0

ADC_LTH can only write to [11:0], [15:12] in read value is sign extension bit, and BIT[11] is sign bit.

12.2.3.2   ADC_HTH

Address: 0x4001_0434

Reset value: 0x0000_07FF

Table 12-14 Higher Threshold Register (ADC_HTH)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				HTH											
				RW											
				0x7FF											

Location	Bit Name	Description
[31:16]		Unused
[15:12]		Sign extension
[11:0]	HTH	Higher threshold of ADC analog watchdog 0

ADC_HTH can only write to [11:0], [15:12] in read value is sign extension bit, and BIT[11] is sign bit.

12.2.3.3   ADC_GEN

Address: 0x4001_0438

Reset value: 0x0

Table 12-15 Monitoring Enable Register (ADC_GEN)

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

Location	Bit Name	Description
[31:10]		Unused
[9:0]	GEN	ADC analog watchdog 0 enable bit BIT0: DAT0 watchdog monitoring enable BIT1: DAT1 watchdog monitoring enable …… BIT9: DAT9 watchdog monitoring enable

12.2.4   Signal Source Register

12.2.4.1   ADC_CHN0

Address: 0x4001_0440

Reset value: 0x0

Table 12-16 Signal Source Register (ADC_CHN0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DS3				DS2				DS1				DS0			
RW				RW				RW				RW			
0				0				0				0			

Location	Bit Name	Description
[31:16]		Unused
[15:12]	DS3	ADC 3rd sampling signal selection
[11:8]	DS2	ADC 2nd sampling signal selection
[7:4]	DS1	ADC 1st sampling signal selection
[3:0]	DS0	ADC 0th sampling signal selection

12.2.4.2 ADC_CHN1

Address: 0x4001_0444

Reset value: 0x0

Table 12-17 Signal Source Register (ADC_CHN1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DS7				DS6				DS5				DS4			
RW				RW				RW				RW			
0				0				0				0			

Location	Bit Name	Description
[31:16]		Unused
[15:12]	DS7	ADC 7th sampling signal selection
[11:8]	DS6	ADC 6th sampling signal selection
[7:4]	DS5	ADC 5th sampling signal selection
[3:0]	DS4	ADC 4th sampling signal selection

12.2.4.3 ADC_CHN2

Address: 0x4001_0448

Reset value: 0x0

Table 12-18 Signal Source Register (ADC_CHN2)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								DS9				DS8			
								RW				RW			
								0				0			

Location	Bit Name	Description
[31:8]		Unused
[7:4]	DS9	ADC 9th sampling signal selection
[3:0]	DS8	ADC 8th sampling signal selection

12.2.4.4 Sampling Signal Selection

Table 12-19 ADC Sampling Channel Signal Selection

ADC_CHNx.DSn Set Value	Sampling Analog Channel
0x0	OPA0_OUT
0x1	ADC_CH1
0x2	ADC_CH2
0x3	ADC_CH3
0x4	ADC_CH4
0x5	ADC_CH5
0x6	ADC_CH6
0x7	ADC_CH7
0x8	ADC_CH8/OPA1_OUT
0x9	ADC_CH9
0xA	ADC_CH10
0xB	Temperature Sensor (TMP)
0xC	Internal ground
0xD	2.4V reference source

12.2.5 Sampling Times Register

12.2.5.1 ADC_CHNT

Address: 0x4001_0450

Reset value: 0x0

Table 12-20 Sampling Times Register (ADC_CHNT)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S4				S3				S2				S1			
RW				RW				RW				RW			
0				0				0				0			

Location	Bit Name	Description
[31:16]		Unused
[15:12]	S4	Number of sampling times for the fourth round in four-round sampling mode
[11:8]	S3	Number of sampling times for the third round in four-round sampling mode
[7:4]	S2	Number of sampling times for the second round in four-round sampling mode
[3:0]	S1	Number of sampling channels for the first round in single-round, two-round or

		four-round sampling mode

Note that the number of sampling channels for each round cannot be set to 0. 1 means one channel, 2 means two channels, ..., 12 means twelve channels, ..., 16 means sixteen channels.

12.2.6   Configuration Register

12.2.6.1   ADC_CFG

Address: 0x4001_0454

Reset value: 0x0

Table 12-21 Configuration Register (ADC_CFG)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			SEL	FSM_RS	DATA_ALIGN	TRG_MODE		SINGLE_TCNT				TRG_EN			
			RW	RW	RW	RW		RW				RW			
			0	0	0	0		0				0			

Location	Bit Name	Description
[31:13]		Unused
[12]	SEL	TADC trigger source selection. 0:MCPWM, 1:UTimer
[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 If the read value is 1, it means that ADC is converting and is busy
[10]	DATA_ALIGN	ADC_DAT alignment 0: Left aligned, with 4'h0 on the right, 1: right aligned, with 4bit sign bit on the left
[9:8]	TRG_MODE	Trigger mode 0: single-round trigger; 1: double-round trigger; 2: reserved; 3: four-round trigger
[7:4]	SINGLE_TCNT	Number of events required to trigger a sample in single-round trigger mode. "0" means one event is required, "1" means two events are required, …… "15" means sixteen events are required
[3:0]	TRG_EN	TADC trigger ADC sampling enable signal. Active high. It is set to

		low by default.
		TRG_EN[0]:TADC[0] enable switch
		TRG_EN[1]:TADC[1] enable switch
		TRG_EN[2]:TADC[2] enable switch
		TRG_EN[3]:TADC[3] enable switch
		A high value indicates enable, and low value indicates disable.

The trigger source of ADC is MCPWM or UTimer, and only one can be selected. MCPWM/UTimer can provide four trigger events - TADC[3:0].

If the trigger source is UTimer, ADC sampling trigger events are UTimer_T0/ UTimer_T1/ UTimer_T2/ UTimer_T3, corresponding to the comparison event 0 and comparison event 1 of Timer0, and comparison event 0 and comparison event 1 of Timer1, respectively.

If the trigger source is MCPWM, ADC sampling trigger events are MCPWM_T0/ MCPWM_T1/ MCPWM_T2/ MCPWM_T3, corresponding to the events generated from MCPWM internal counter to MCPWM_TMR0/1/2/3, respectively.

The trigger signal of UTimer to ADC can be sent by setting GPIO as the 7th/8th function, that is, Timer0/1 function, which is used to capture and debug.

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.

12.2.7 Software Trigger Register

12.2.7.1 ADC_SWT

Address: 0x4001_0458

Reset value: 0x0

Table 12-22 Software Trigger Register (ADC_SWT)

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

Location	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 writes to the bus generates one software trigger. When a software trigger is generated after data is written, the register is automatically cleared, and wait for the next software trigger.

12.2.8 DC Bias Register

One signal source of ADC is GND, and the DC bias of ADC can be obtained by measuring this channel.

The chip has been calibrated in the factory, and the calibration value is saved in the Flash info area, which will be automatically loaded and stored in ADC_DC register when the chip is powered on. The future sampling value of other channels will subtract the DC bias value in ADC_DC, and then stored in the corresponding digital register (ADCx_DAT).

Considering signal error, the signal subtracting the DC bias may be overflow. In this case, ADC will carry out saturation processing to make the signal in the range of -2.4V ~ +2.4V or -3.6V ~ +3.6V.

12.2.8.1 ADC_DC

Address: 0x4001_0460

Reset value: 0x0

Table 12-23 DC Bias Register (ADC_DC)

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

Location	Bit Name	Description
[31:8]		Unused
[7:0]	DC_OFFSET	Gain ADC DC offset of sampling circuit

Considering that ADC's DC offset is a value close to 0, the high 8-bit is only the sign extension of ADC_DC[7], and the ADC_DC register will also perform sign extension when participating in ADC data correction. If writing 0x10B0 to ADC_DC, the actual value written is 0xB0, and the read value is 0xFFB0.

The ADC_DC value stored should correspond to the right-aligned offset value. When the ADC_CFG register is configured to be left-aligned, the hardware will automatically adjust ADC_DC to participate in ADC correction according to the alignment setting. Specifically, when right-aligned, ADC_DC[15:0] directly participates in the correction operation; when left-aligned, ADC_DC[15:0] will first shift to the left by 4 bits and then participate in the ADC correction operation.

12.2.9 Gain Calibration Register

12.2.9.1 ADC_AMC

Address: 0x4001_0464

Reset value: 0x200

Table 12-24 Gain Calibration Register (ADC_AMC)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						AMC									
						RW									
						0x200									

Location	Bit Name	Description
[31:10]		Unused
[9:0]	AMC	Gain calibration register for sampling circuit

12.2.10 Interrupt Enable Register

12.2.10.1 ADC_IE

Address: 0x4001_0468

Reset value: 0x0

Table 12-25 Interrupt Enable Register (ADC_IE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	AWD_RE	HERR_RE	SERR_RE	S4_RE	S3_RE	S2_RE	S1_RE		AWD_IE	HERR_IE	SERR_IE	S4_IE	S3_IE	S2_IE	S1_IE
	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

Location	Bit Name	Description
[31:15]		Unused
[14]	AWD_RE	Analog watchdog over threshold DMA request enable
[13]	HERR_RE	Hardware trigger DMA request enable that occurs in non-idle state
[12]	SERR_RE	Software trigger DMA request enable that occurs in non-idle state
[11]	S4_RE	DMA request enable triggered by the completion of the fourth-round

		sampling
[10]	S3_RE	DMA request enable triggered by the completion of the third-round sampling
[9]	S2_RE	DMA request enable triggered by the completion of the second-round sampling
[8]	S1_RE	DMA request enable triggered by the completion of the first-round sampling
[7]		Unused
[6]	AWD_IE	Analog watchdog over threshold interrupt enable
[5]	HERR_IE	Hardware trigger interrupt enable that occurs in non-idle state
[4]	SERR_IE	Software trigger interrupt enable that occurs in non-idle state
[3]	S4_IE	Interrupt enable triggered by the completion of the fourth-round sampling
[2]	S3_IE	Interrupt enable triggered by the completion of the third-round sampling
[1]	S2_IE	Interrupt enable triggered by the completion of the second-round sampling
[0]	S1_IE	Interrupt enable triggered by the completion of the first-round sampling

12.2.10.2 ADC_IF

Address: 0x4001_046C

Reset value: 0x0

Table 12-26 Interrupt Flag Register (ADC_IF)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									AWD_IF	HERR_IF	SERR_IF	S4_IF	S3_IF	S2_IF	S1_IF
									RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C
									0	0	0	0	0	0	0

Location	Bit Name	Description
[31:7]		Unused
[6]	AWD_IF	Analog watchdog over threshold interrupt flag
[5]	HERR_IF	Hardware trigger interrupt flag that occurs in non-idle state
[4]	SERR_IF	Software trigger interrupt flag that occurs in non-idle state
[3]	S4_IF	Interrupt flag triggered by the completion of the fourth-round sampling
[2]	S3_IF	Interrupt flag triggered by the completion of the third-round sampling
[1]	S2_IF	Interrupt flag triggered by the completion of the second-round sampling
[0]	S1_IF	Interrupt flag triggered by the completion of the first-round sampling

In the above ADC_IF flag, 0: indicates that no interrupt has occurred, 1: indicates that an interrupt has occurred, write 1 to clear.

12.3 **Application Guide**

12.3.1 ADC Sampling Trigger Mode

The ADC supports one-round, two-round, and four-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. The state transition inside the ADC is described as follows, including eight states: sampling state 0 ~ 3 and idle state 0 ~ 3.

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 ADC_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 ADC_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 ADC_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 second round reaches the preset value ADC_CHNT[7:4], it will determine whether it is a two-round sampling.

Yes: End this sampling and return to idle state 0.

No: Enter idle state 2 and wait for the third and fourth triggers to complete sampling.

Third Trigger

Third Round of Sampling

When the sampling times of the third round reaches the preset value ADC_CHNT[11:8], the ADC will enter the idle state 3.

Fourth Trigger

Fourth Round of Sampling

When the sampling times of the fourth round reaches the preset value ADC_CHNT[15:12], the ADC will return to the idle state 0.

The trigger conditions of various hardware trigger modes are summarized in Table 12-27. 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 12-27 Sampling Trigger Mode

		Single-round Trigger	Two-round Trigger	Four-round Trigger
TADC Trigger	None (TADC trigger is not enabled)		The first round of TADC [0] Two-round Trigger column: The first round of TADC [0] The second round of TADC [1]	The first round of TADC [0] The second round of TADC [1] The third round of TADC [2] The fourth round of TADC [3]
	C times of TADC [0]			
	C times of TADC [1]			
	C times of TADC [2]			
	C times of TADC [3]			
	C times TADC[0]/TADC[1]/ TADC[2]/TADC[3]			
Software trigger	Write 16'h5aa5 to ADC_SWT		First round: Write 16'h5aa5 to ADC_SWT Second round: Write 16'h5aa5 to ADC_SWT	First round: Write 16'h5aa5 to ADC_SWT Second round: Write 16'h5aa5 to ADC_SWT Third round: Write 16'h5aa5 to ADC_SWT Fourth round: Write 16'h5aa5 to ADC_SWT

12.3.1.1   Single-round Trigger Mode

Single-round triggering 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 ADC_CHNT; when the register value is 1 ~ 10, the corresponding sampling times are 1 ~ 10.

Assuming that the number of single-segment sampling configuration channels is 4, the sampled and converted data will be filled in ADC_DAT0, ADC_DAT1, ADC_DAT2, and ADC_DAT3 in sequence.

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_CHN0/1/2/3. 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 TADC [2] occurs four times. The state transition is shown in Fig. 123.
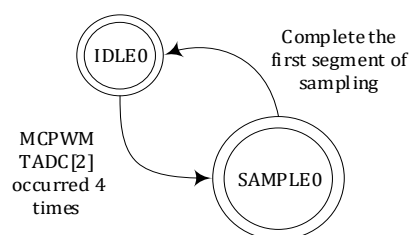
Fig. 12-3 State Transition Diagram of ADC Single-round Sampling

12.3.1.2   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.

Assuming that the number of two sampling configuration channels is 2 and 3, respectively, the data after the first sampling and conversion round will be filled into ADC0_DAT and ADC_DAT1 in turn, and the data after the second sampling and conversion round will be filled into ADC_DAT2, ADC_DAT3, and ADC_DAT4 in turn.

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 two-round sampling triggered by two software trigger as an example, the state transition is shown in Fig. 124.
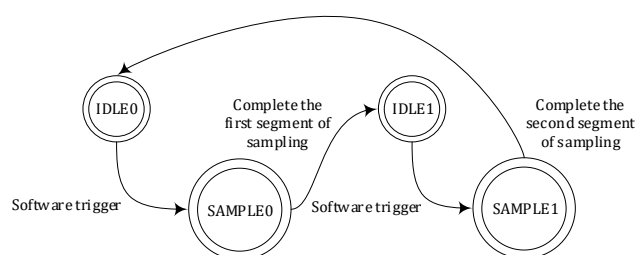
Fig. 12-4 State Transition Diagram of ADC Two-round Sampling

12.3.1.3 Four-round Trigger Mode

Similar to the two-round sampling trigger, the four trigger sources are TADC [0], TADC [1], TADC [2], and TADC [3], and they should be triggered sequentially by the timing signals of MCPWM, TADC [0], TADC [1], TADC [2], and TADC [3]; Or it can be triggered by four software trigger. The sampling times of the first, second, third and fourth sampling in the four-round sampling are ADC_CHNT[7:4], ADC_CHNT[3:0], ADC_CHNT[7:4], and ADC_CHNT[3:0], respectively. Taking the four-round sampling triggered by the MCPWM timing signal TADC [0], TADC [1], TADC [2], and TADC [3] as an example, the state transition is as shown in Fig. 125.

Assuming that the number of four-round sampling configuration channels is 2, 3, 1, and 5, respectively, the data after sampling and conversion of the first round will be filled into ADC0_DAT0, ADC0_DAT1 in turn, and the data after sampling and conversion of the second round will be filled into ADC0_DAT2, ADC0_DAT3, ADC0_DAT4, the data after sampling and conversion of the third round will be filled into ADC0_DAT5, and the data after sampling and conversion of the fourth round will be filled into ADC0_DAT6, ADC0_DAT7, ADC0_DAT8, ADC0_DAT9 in turn.
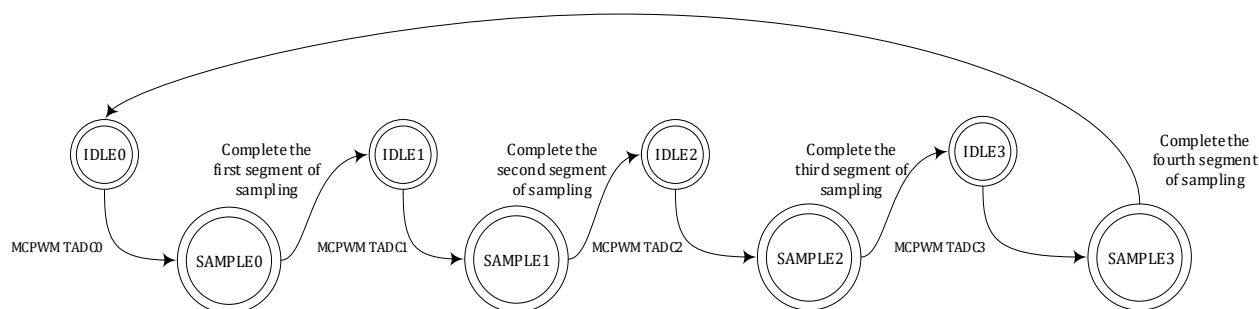
Fig. 12-5 State Transition Diagram of ADC Four-round Sampling

Before using the MCPWM timer to generate the ADC sampling trigger signals, MCPWM_TMR0, MCPWM_TMR1, MCPWM_TMR2, and MCPWM_TMR3 registers should be set in advance, which correspond to the MCPWM counter value when TADC0/1/2/3 occurs. The MCPWM_TH should also be set at the same time. Set the counter count range and MCPWM_TCLK, and then set the count clock frequency and enable the clock.

12.3.2 Interrupt

12.3.2.1 Interrupt of Single Round Trigger Sampling

An interrupt ADC_IF[0] is triggered when sampling is completed.

12.3.2.2 Interrupt of Two Round Trigger Sampling

An interrupt ADC_IF[0] is triggered when the first-round sampling is completed, and another done interruption ADC_IF[1] will be triggered when the second-round sampling is completed.

12.3.2.3 Interrupt of Four Round Trigger Sampling

One interrupt ADC_IF[0] / ADC_IF[1] / ADC_IF[2] / ADC_IF[3] is triggered when the first-round, second-round, third-round and the fourth-round sampling is completed accordingly.

12.3.3   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.FSM_RESET 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;
ADCx_CFG = 0x0000;
ADCx_CFG = 0x0800;
/*
        Add your code below, like:
ADCx_CHNT = 0x0005
ADCx_CHN0 = 0x3210;
ADCx_CHN1 = 0x7654;
*/
ADCx_CFG = ADCx_CFG_temp;
```

12.3.4   Select the Corresponding Analog Channel

For the channel corresponding to the signal sampled by the ADC, please refer to Pin Function Selection in DATASHEET. Turn off the corresponding IO IE and OE to use the simulation function.

13 Universal Timer (UTimer)

13.1 Introduction

13.1.1 Functional block diagram

As shown in Fig. 131, the universal timer UTIMER mainly includes two independent Timers, 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.
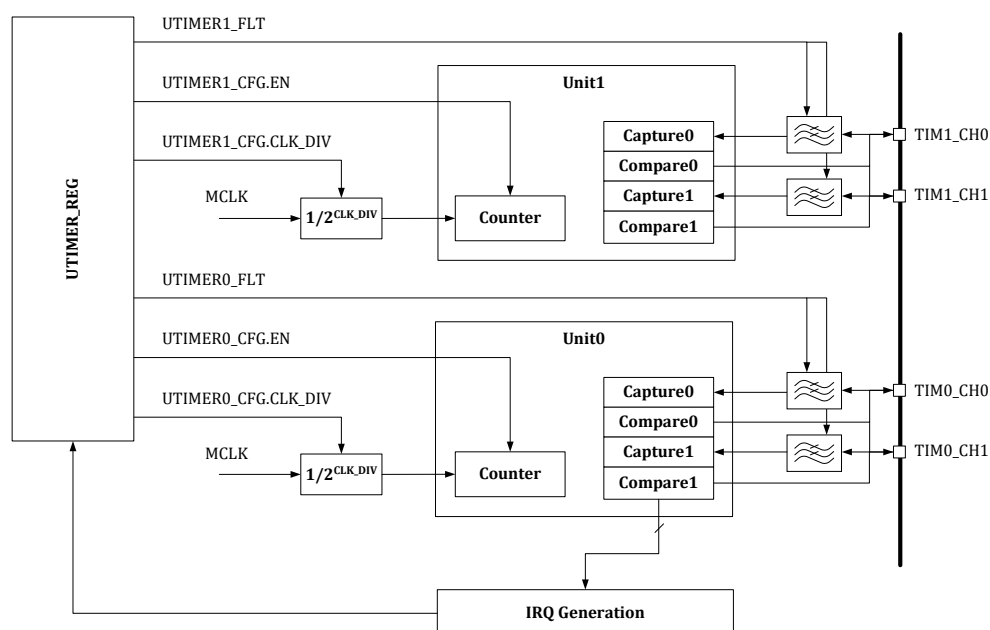
Fig. 13-1 Block Top Functional Block Diagram

13.1.1.1 Register Module

Read and write control registers of each submodule.

Access to the status and result registers of each submodule.

Interrupt signal processing and interrupt generation for each submodule.

13.1.1.2 IO Filter Module

The IO filter module filters the input signal from outside the chip to reduce the effect of glitches on the timer.

13.1.1.3 Universal Timer Module

The utimer_unt module implements general timer functions, including comparison and capture modes. Each timer includes two channels, 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.

13.1.1.4 Clock Divider Module

The clock frequency dividing module is used to generate various signals of clock frequency dividing.

13.1.2   Features

The timer module has the following characteristics:

● Available for working in at different frequencies independently

● Timer0 is a 16-bit general timer

● Timer1 is a 32bit general timer

● Each general timer processes two external input signals (capture mode) or generates two output signals (comparison mode)

● Available for filtering each input signal of up to 2040 main clocks, that is, when the chip works at a clock frequency of 48MHz, it can filter out glitches below 42uS width.

13.2 **Implementation Description**

13.2.1   Clock Divider

Timer uses UTIMERx_CFG.CLK_DIV for frequency division, to reduce the counting frequency, and the frequency division coefficient can be set to 1/2/4/8/16/32/64/128.

13.2.2   Interrupt Flag Clear

The product adopts the design of writing 1 to each interrupt flag to clear it.

13.2.3   Filtering

Each timer module has two channels, which share one filtering coefficient when working in input capture mode.

The filter width can be adjusted by setting the filter register UTIMERx_FLT, 0 ~ 2040 system clock widths are available.

Timer count clock is used for filtering input signals. UTIMERx_CFG.CLK_DIV has an impact on the division of the timer count clock and the filtered clock.

As shown in Fig. 132, 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.
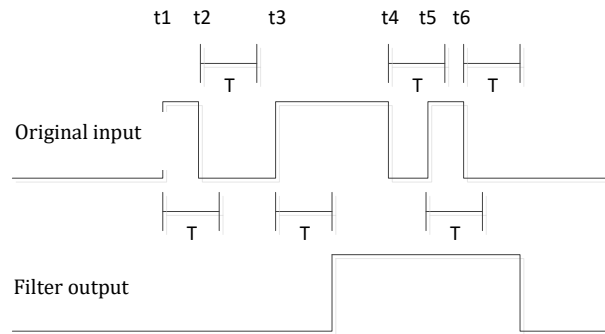
Fig. 13-2 UTimer Filter Diagram

13.2.4   Mode

13.2.4.1 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. Timer period will be clk_freq*(TH+1).
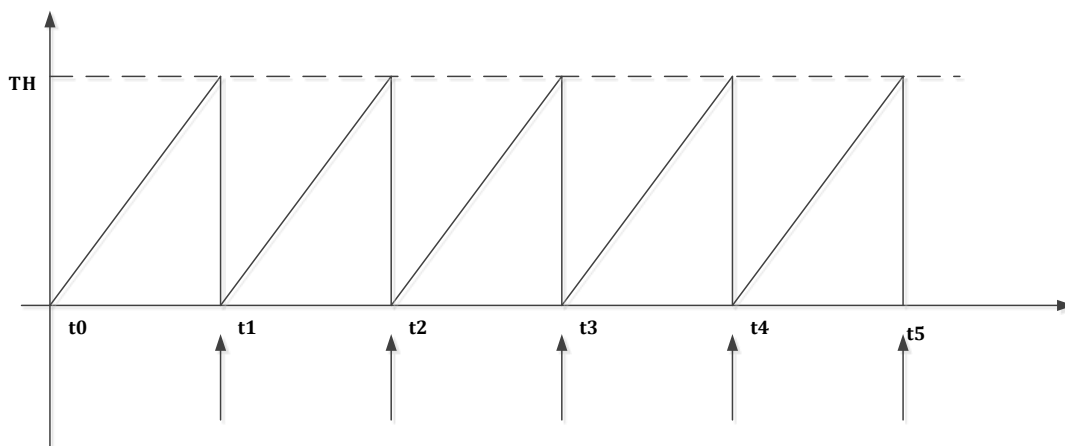
Fig. 13-3 UTimer

The counter clock can be configured by UTIMERx_CFG.CLK_SRC, and the system clock (with channel of 48MHz PLL clock) inside the chip or channel 0/1 of Timer0/1 can be used as the external clock for counting.

The counter clock can divide the frequency by UTIMERx_CFG.CLK_DIV to reduce the counting frequency.

13.2.4.2 Comparison Mode

In compare mode, a compare interrupt is generated when the counter counts to the UTIMERx_CMP value, and a compare pulse can be driven. When returning to zero, it will output a level to the IO port (polarity can be configured); when the comparison event occurs, the level is inverted, and another level is output to the IO port. 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.
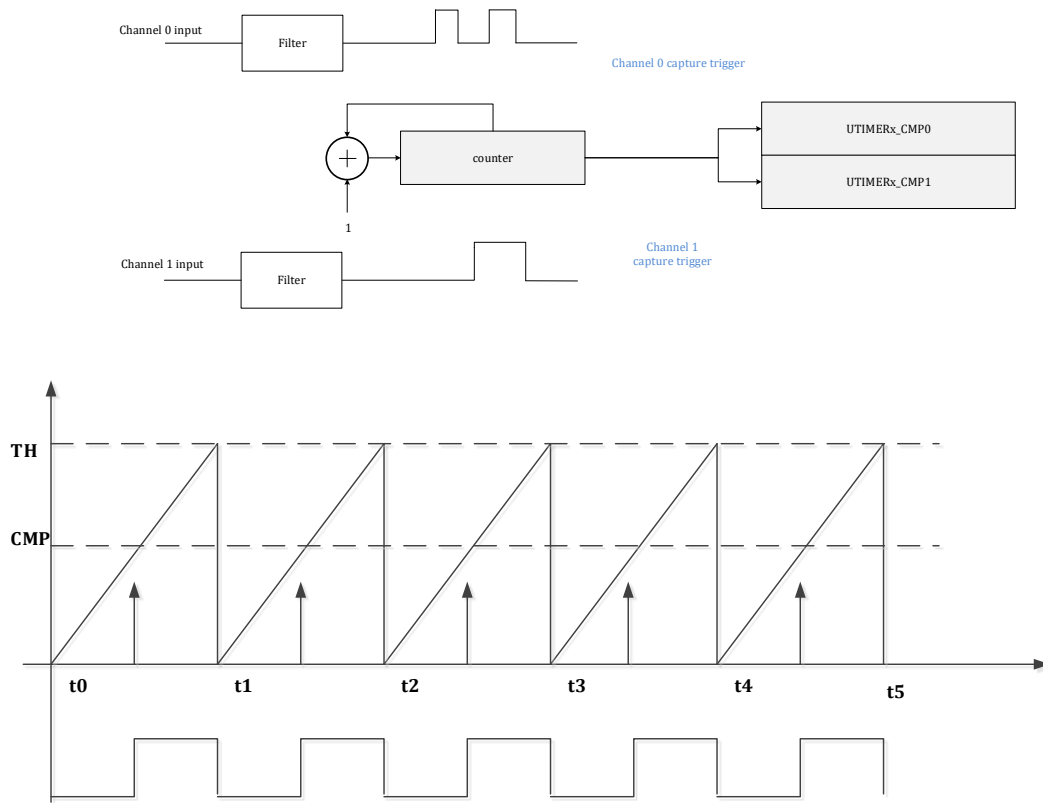
Fig. 13-4 UTimer Comparison Mode

In compare mode, when UTIMERx_CFG.EN=0, that is Timer does not start counting, write 1 to UTIMERx_CFG.ONE_TRIG to trigger the Timer to send a cycle pulse with a specific duty cycle. After sending, Timer will return to idle state and does not act anymore.

13.2.4.3 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 UTIMER_UNTx_CMP register and a capture interrupt is generated. When the counter returns to zero, the zero return interrupt will still be generated.
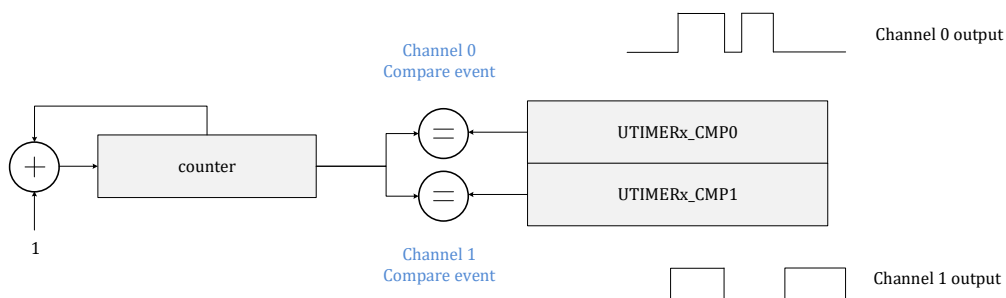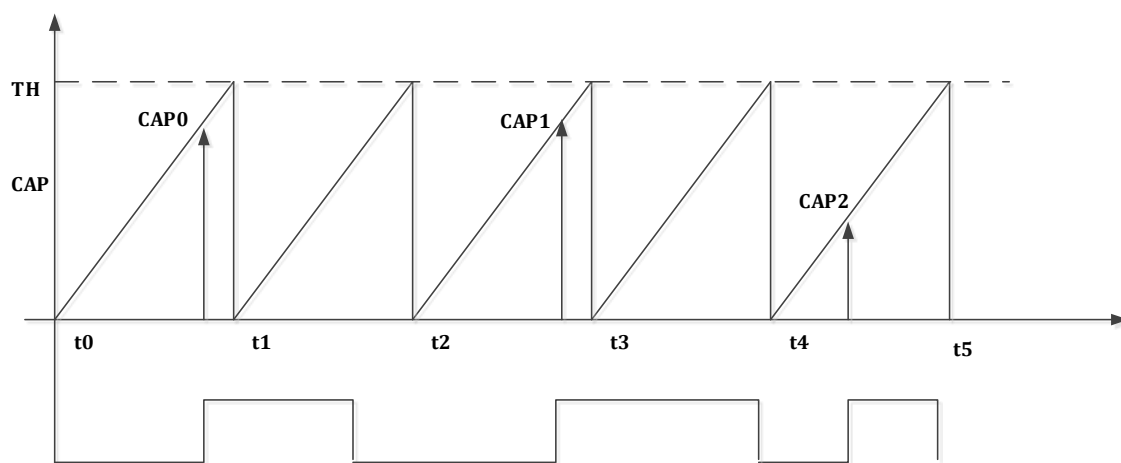
Fig. 13-5 UTimer Capture Mode

As shown in Figure 135, 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 UTIMER_UNTx_CMP register.

In capture mode, the signal source captured by channel 0/1 can be set by UTIMERx_CFG.SRC0 and UTIMERx_SRC1. It can be set to the channel signal from IO, or from analog comparator, and XOR of two channel signals from IO.

You can set UTIMERx_CFG.CAP0_CLR_EN or UTIMERx_CFG.CAP1_CLR_EN to clear Timer automatically, that is, when a capture event occurs in channel 0/1, UTIMER_CNT will be automatically cleared to restart counting. This enables the Timer to calculate the period and duty cycle of the captured signal.

For example, if channel 0/1 of Timer0 is set to capture the same signal at the same time, channel 0 captures the rising edge and channel 1 captures the falling edge. Besides, CAP0_CLR_EN is enabled, that is UTIMER_CNT will be automatically cleared when channel 0 capture event occurs. When channel 0 capture event (CAP0) occurs, UTIMERx_CMP0 records the period of last signal cycle, and UTIMERx_CMP1 records the high-level duty cycle of last signal cycle.

Similarly, if channel 0/1 of Timer0 is set to capture the same signal at the same time, channel 0 captures the rising edge and channel 1 captures the falling edge. Besides, CAP1_CLR_EN is enabled, that is UTIMER_CNT will be automatically cleared when channel 1 capture event occurs. When channel 1 capture event (CAP1) occurs, UTIMERx_CMP1 records the period of last signal cycle, and UTIMERx_CMP0 records the low-level duty cycle of last signal cycle.

13.2.4.4 One-time trigger

In the comparison mode, when UTIMERx_CFG.EN=0, that is, when the Timer does not start counting, by writing 1 to UTIMERx_CFG.ONE_TRIG, the Timer can be triggered to issue a waveform with a specific duty cycle of a period. After the end, the Timer returns to the idle state, and the channel no longer operates. If the TIMER single trigger occurs again within the counting period, the TIMER resets the CNT and starts counting for a period.

When UTIMERx_CFG.EN=1, by writing 1 to UTIMERx_CFG.ONE_TRIG, the Timer can be triggered

to reset the CNT and emit a waveform with a specific duty cycle of a period. After one cycle, normal counting continues.
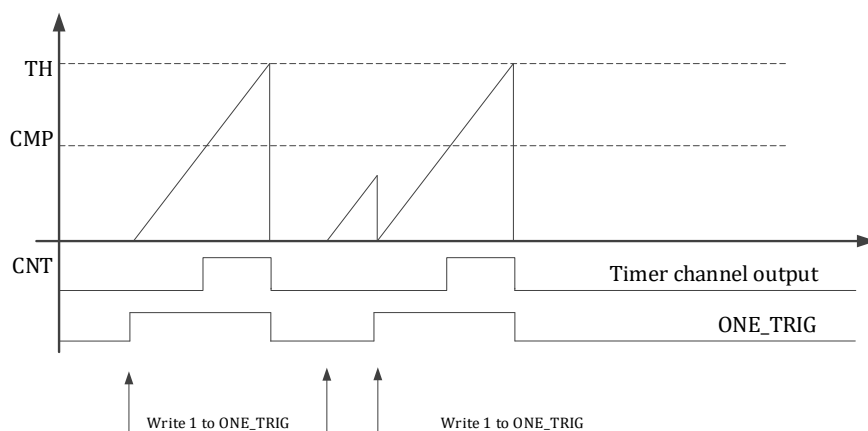
Fig.13-1 UTIMER One-time trigger

13.2.5   ADC Trigger

The comparison event of Timer0/1 (CMP0/1) can be used as the ADC sampling trigger event.

13.3 **Register**

13.3.1   Address Allocation

The base address of Timer0 in chip is 0x4001_0500.

Table 13-1 Address 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_0600.

Table 13-2 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 difference between Timer0 and Timer1 is that the related registers of Timer0 counter is 16 bits wide, while the related registers of Timer1 counter is 32 bits wide.

13.3.2 UTimer0 Register

13.3.2.1 Timer0 Configuration Register (UTIMER0_CFG)

Address: 0x4001_0500

Reset value: 0x0

Table 13-3 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

Location	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
[23]		Unused
[22:20]	CLK_DIV	Timer0 counter frequency configuration. The counting frequency of the counter is the 2CLK_DIV division of the main clock frequency. The default value is 0 and does not divide frequency. 0:1 frequency division 1:2 frequency division 2:4 frequency division 3:8 frequency division 4:16 frequency division 5:32 frequency division 6:64 frequency division 7:128 frequency division
[19:16]	CLK_SRC	Timer clock source 0: Internal clock of chip 1: Reserved 2: External clock signal of Timer0 channel 0 3: External clock signal of Timer0 channel 1 4: External clock signal of Timer1 channel 0 5: External clock signal of Timer1 channel 1
[15:12]	SRC1	Channel 1 signal source in Timer capture mode. The default value is 4'h1. 0: Timer channel 0, from chip GPIO (see Datasheet and application configuration) 1: Timer channel 1, from chip GPIO (see Datasheet and application configuration) 2: Output of CMP0 3: Output of CMP1 8: 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 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. 0: Timer channel 0, from chip GPIO (see Datasheet and application configuration) 1: Timer channel 1, from chip GPIO (see Datasheet and application configuration) 2: Output of CMP0 3: Output of CMP1

Location	Bit Name	Description
		8: 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.

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.

SRC0 and SRC1 can be set to sample the same TIMER channel IO, capture the rising edge and falling edge respectively, and clear the TIMER CNT when the rising edge or falling edge, which is convenient to calculate the period and duty cycle of square wave. In this way, two TIMER channels will be occupied. However, only one TIMER channel I/O is used, and the other TIMER channel I/O can be used for other functions.

13.3.2.2 Timer0 Threshold Register (UTIMER0_TH)

Address: 0x4001_0504

Reset value: 0x0

Table 13-4 Timer0 Threshold Register (UTIMER0_TH)

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

Location	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.

13.3.2.3 Timer0 Count Register (UTIMER0_CNT)

Address: 0x4001_0508

Reset value: 0x0

Table 13-5 Timer0 Count Register (UTIMER0_CNT)

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

Location	Bit Name	Description
[31:16]		Unused
[15:0]	CNT	The current count value of the Timer0 counter. New count value can be written.

Note that the Timer0 clock needs to be started by SYS_CLK_FEN.TIMER0_CLK_EN before writing to UTIMER0_CNT.

13.3.2.4 Timer0 Channel 0 Compare Capture Register (UTIMER0_CMP0)

Address: 0x4001_050C

Reset value: 0x0

Table 13-6 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																

Location	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, the channel 0 value can always be 1. If setting CMP0 = TH+1 in CH0_POL, the channel 0 value can always be CH0_POL.

13.3.2.5 Timer0 Channel 1 Compare Capture Register (UTIMER0_CMP1)

Address: 0x4001_0510

Reset value: 0x0

Table 13-7 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

Location	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, the channel value can always be 1. If setting CMP1 = TH+1 in CH1_POL, the channel value can always be CH1_POL.

13.3.2.6 Timer0 External Event Select Register (UTIMER0_EVT)

Address: 0x4001_0514

Reset value: 0x0

Table 13-8 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				

Location	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 10: MCPWM TADC[0] comparison event 11: MCPWM TADC[1] comparison event 12: MCPWM TADC[2] comparison event 13: MCPWM TADC[3] comparison event

13.3.2.7 Timer0 Filter Control Register (UTIMER0_FLT)

Address: 0x4001_0518

Reset value: 0x0

Table 13-9 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							

Location	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 = 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.

13.3.2.8 Timer0 Interrupt Enable Register (UTIMER0_IE)

Address: 0x4001_051C

Reset value: 0x0

Table 13-10 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

Location	Bit Name	Description
[31:11]		Unused
[10]	ZC_RE	Timer counter over-zero 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.

13.3.2.9 Timer0 Interrupt Flag Register (UTIMER0_IF)

Address: 0x4001_0520

Reset value: 0x0

Table 13-11 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

Location	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;

13.3.3 UTimer1 Register

13.3.3.1 Timer1 Configuration Register (UTIMER1_CFG)

Address: 0x4001_0600

Reset value: 0x0

Table 13-12 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

Location	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: Run automatically 1: Wait for external event to trigger counting, and stop after counting one cycle
[23]		Unused
[22:20]	CLK_DIV	Timer0 counter frequency configuration. The counting frequency of the counter is the 2CLK_DIV division of the main clock frequency. The default value is 0 and does not divide frequency. 00:1 frequency division 1:2 frequency division 2:4 frequency division 3:8 frequency division 4:16 frequency division 5:32 frequency division 6:64 frequency division 7:128 frequency division

[19:16]	CLK_SRC	Timer clock source 0: Internal clock of chip 1: Reserved 2: External clock signal of Timer0 channel 0 3: External clock signal of Timer0 channel 1 4: External clock signal of Timer1 channel 0 5: External clock signal of Timer1 channel 1
[15:12]	SRC1	Channel 1 signal source in Timer capture mode. The default value is 3'h1. 0: Timer channel 0, from chip GPIO (see Datasheet and application configuration) 1: Timer channel 1, from chip GPIO (see Datasheet and application configuration) 2: Output of CMP0 3: Output of CMP1 8: 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 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 01 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. 0: Timer channel 0, from chip GPIO (see Datasheet and application configuration) 1: Timer channel 1, from chip GPIO (see Datasheet and application configuration) 2: Output of CMP0 3: Output of CMP1 8: 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_CA	Channel 0 falling edge capture event enable. 1: enable; 0: disable.

	P_EN	A 10 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_CA P_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.

13.3.3.2 Timer1 Threshold Register (UTIMER1_TH)

Address: 0x4001_0604

Reset value: 0x0

Table 13-13 Timer1 Threshold Register (UTIMER1_TH)

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

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

Location	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.

13.3.3.3 Timer1 Count Register (UTIMER1_CNT)

Address: 0x4001_0608

Reset value: 0x0

Table 13-14 Timer1 Count Register (UTIMER1_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								

Location	Bit Name	Description
[31:0]	CNT	The current count value of the Timer1 counter. New count value can be written.

Note that the Timer1 clock needs to be started by SYS_CLK_FEN. TIMER1_CLK_EN before writing to UTIMER1_CNT.

13.3.3.4 Timer1 Channel 0 Compare Capture Register (UTIMER1_CMP0)

Address: 0x4001_060C

Reset value: 0x0

Table 13-15 Timer1 Channel 0 Compare Capture Register (UTIMER1_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															

Location	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.

If setting CMP0 = 0, the channel 0 value can always be 1. If setting CMP0 = TH+1 in CH0_POL, the channel 0 value can always be CH0_POL.

13.3.3.5 Timer1 Channel 1 Compare Capture Register (UTIMER1_CMP1)

Address: 0x4001_0610

Reset value: 0x0

Table 13-16 Timer1 Channel 1 Compare Capture Register (UTIMER1_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															

Location	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.

If setting CMP1 = 0, the channel value can always be 1. If setting CMP1 = TH+1 in CH1_POL, the channel value can always be CH1_POL.

13.3.3.6 Timer1 External Event Select Register (UTIMER1_EVT)

Address: 0x4001_0614

Reset value: 0x0

Table 13-17 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				

Location	Bit Name	Description
[31:3]		Unused
[2:0]	EVT_SRC	Timer 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 10: MCPWM TADC[0] comparison event 11: MCPWM TADC[1] comparison event 12: MCPWM TADC[2] comparison event 13: MCPWM TADC[3] comparison event

13.3.3.7 Timer1 Filter Control Register (UTIMER1_FLT)

Address: 0x4001_0618

Reset value: 0x0

Table 13-18 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							

Location	Bit Name	Description
[31:8]		Unused
[7:0]	FLT	Filter width of channel 0/1. with value range of 0-255. When 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.

13.3.3.8 Timer1 Interrupt Enable Register (UTIMER1_IE)

Address: 0x4001_061C

Reset value: 0x0

Table 13-19 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

Location	Bit Name	Description
[31:11]		Unused
[10]	ZC_RE	Timer counter over-zero 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.

13.3.3.9 Timer1 Interrupt Flag Register (UTIMER1_IF)

Address: 0x4001_0620

Reset value: 0x0

Table 13-20 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

Location	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 MCPWM

14.1 Introduction

The MCPWM module is a module that precisely controls the output of the motor drive waveform, which contains a 16-bit counter-up counter to provide two counting 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 48MHz, 24MHz, 12MHz and 6MHz, 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 four emergency stop signals, two of which come from IO and two 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).
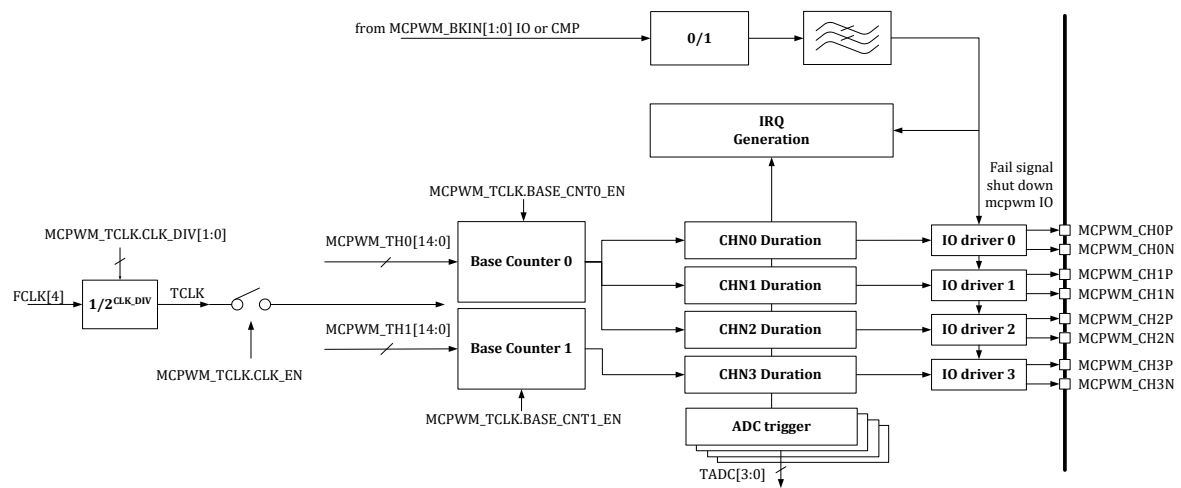
Fig. 14-1 MCPWM Module Block Diagram

Usually, a 48MHz clock is used as the operating frequency of the MCPWM module to ensure the timing accuracy.

14.1.1   Base Counter Module

The module is mainly composed of a count-up counter, and its count threshold is 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 0/1 can be controlled by register configuration MCPWM_TCLK.BASE_CNT0_EN/MCPWM_TCLK.BASE_CNT1_EN.
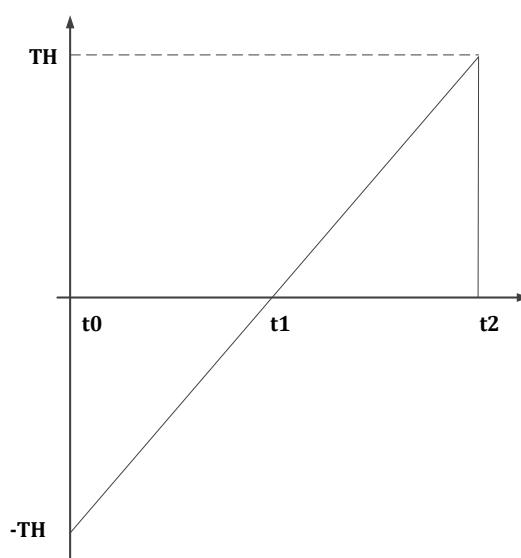
Fig. 14-2 Base Counter t0/t1 Timing

Before running the MCPWM module, users should set the corresponding comparison thresholds (MCPWM_TH00 ~ MCWM_TH31), dead-zone registers (MCPWM_DTH00 ~ MCWM_DTH31) and the PWM period (MCPWM_TH0/MCPWM_DTH1) 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 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.
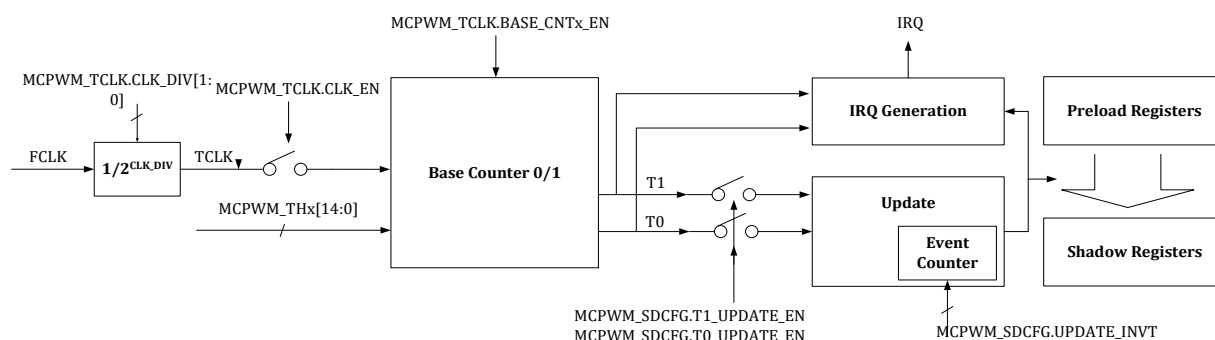
Fig. 14-3 MCPWM Update Mechanism

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 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.

14.1.2   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 two fail signal inputs to MCPWM, namely FAIL0 and FAIL1, which come from the chip IO MCPWM_BKIN [1: 0] or the output CMP [1: 0] of the on-chip comparator.

Among them, the P/N channel 0/1/2 of MCPWM can choose to use CMP[0] or MCPWM_BKIN[0], and the P/N channel 3 of MCPWM can choose to use CMP[1] or MCPWM_BKIN[1].
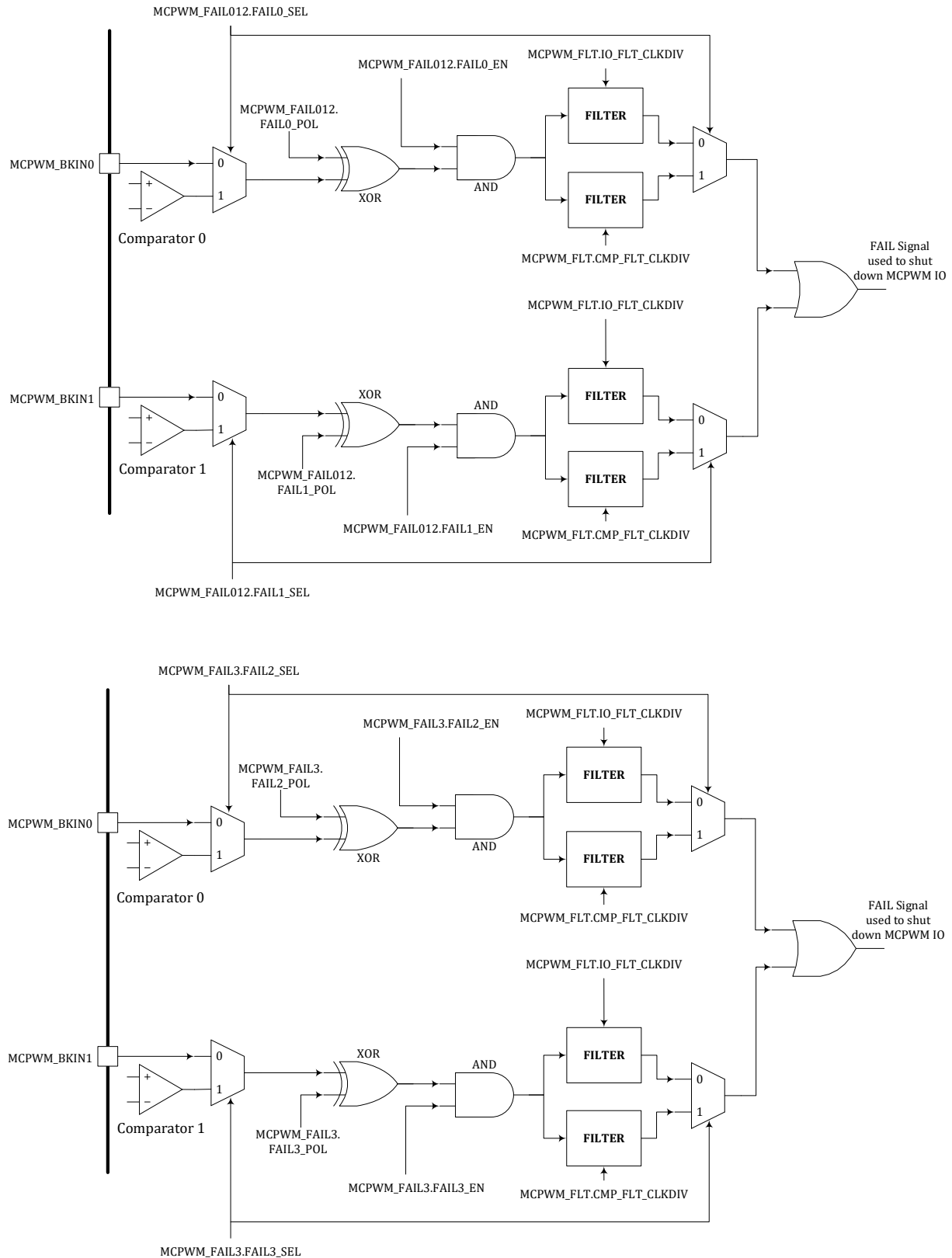
Fig. 14-4 MCPWM FAIL Logic Diagram

The clock of the Filter module comes from the gated clock FCLK [4] of the system main clock MCLK (see SYS_CLK_FEN), and is divided by two stages. The first-stage frequency division is

controlled by MCPWM_FLT.CLK_DIV, which divides by 1/2/4/8 times. The second-stage frequency division can achieve 1 ~ 256 times the frequency division. If the Fail signal comes from MCPWM_BKIN [1: 0], then use MCPWM_TCLK.IO_FLT_CLKDIV [7:0] as the second-stage frequency division coefficient; if the Fail signal comes from the internal comparator output, then use MCPWM_TCLK.CMP_FLT_CLKDIV [7:0] as the frequency division factor of the second stage, as shown in Fig. 145.

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 TMCLK is the clock period of MCLK/FCLK [4], 48MHz corresponds to 20.8ns. FLT_CLKDIV may be MCPWM_FLT.IO_FLT_CLKDIV or MCPWM_FLT.CMP_FLT_CLKDIV depending on the configuration.

$$T = T_{MCLK} \times (MCPWM\_TCLK.CLK\_DIV) \times (FLT\_CLKDIV+1) \times 16$$
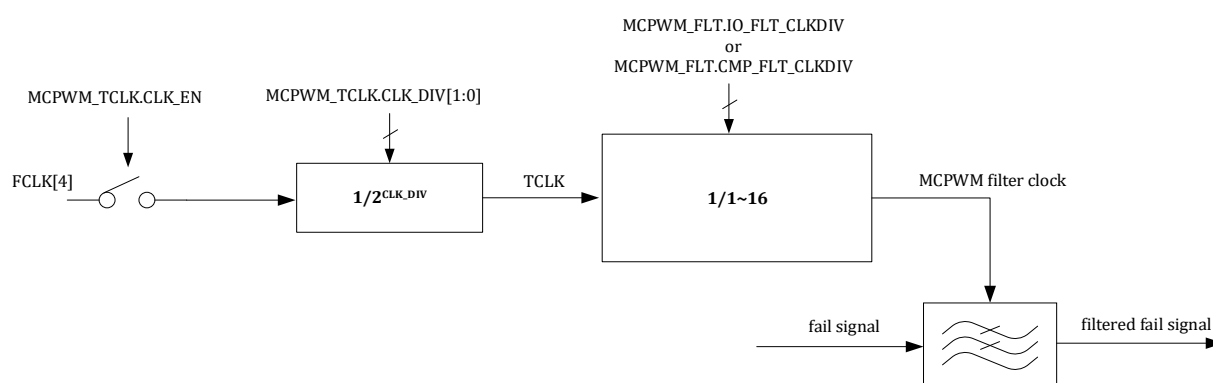
Fig. 14-5 MCPWM Fail Signal Filtering Clock Generation Logic

Once a Fail event occurs, the hardware forces the IO output to the default value of the fault specified in the MCPWM_FAILx.CHxN_DEFAULT and MCPWM_FAILx.CHxP_DEFAULT registers. After then, the values of MCPWM_FAILx.CHxN_DEFAULT and MCPWM_FAILx.CHxP_DEFAULT are directly output to the IO port, and are no longer affected by the polarity control such as MCPWM_FAILx.FAIL_POL. Among which, MCPWM_FAIL012 is used to control the default value of the fault in channel 0/1/2, and MCPWM_FAIL3 is used to control the default value of the fault in channel 3.

The two Fail signals from the comparator are controlled by the comparator windowing, but are not controlled by the comparator filter. After the Fail signal enters MCPWM, it can be filtered in the MCPWM module.

14.1.3   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

14.1.4 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:
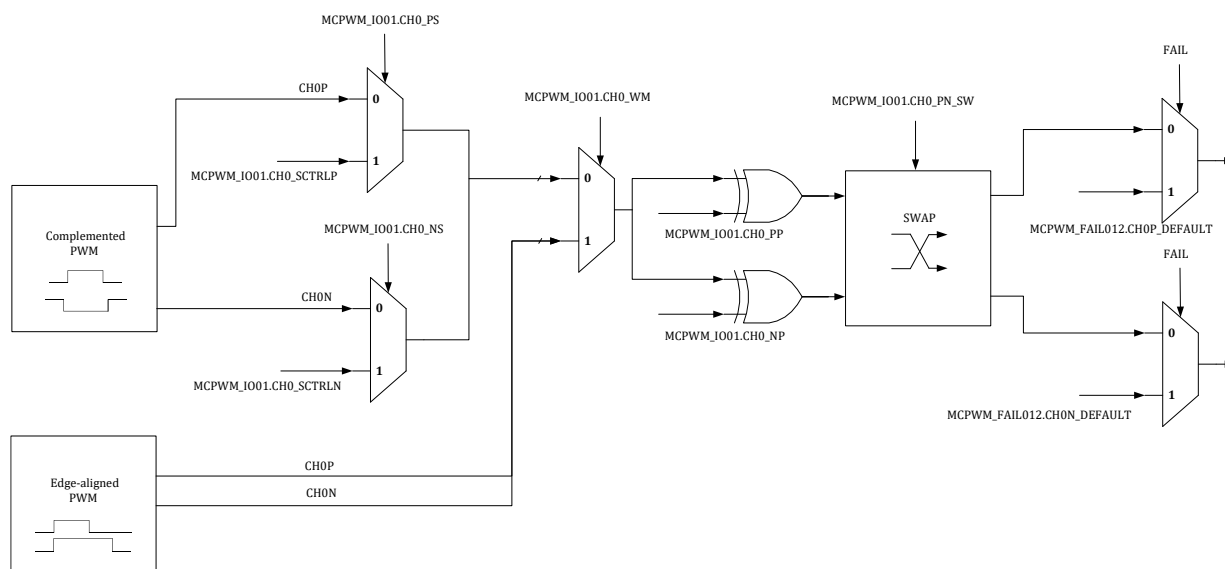
Fig. 14-6 IO Driver Module Data Flow Chart

14.1.4.1 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 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 DTH0 is delayed after dead-zone delay, and CH <n> P is turned on.

When the counter CNT value counts up to reach TH <n> 1, CH <n> P is turned off at time t4, after dead-zone delay DTH1, 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.
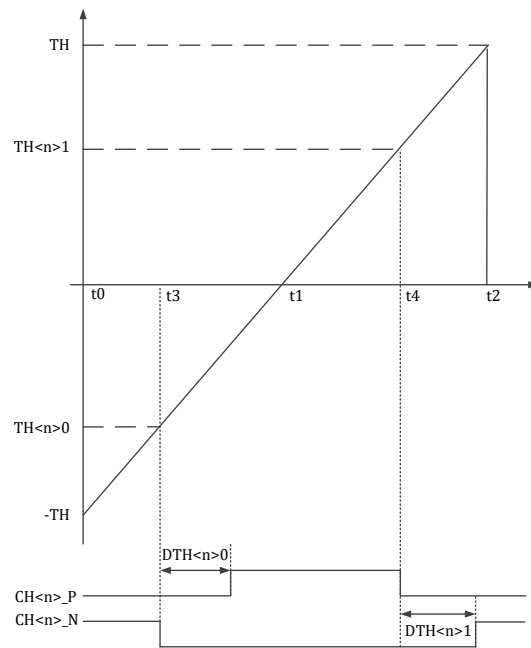
Fig. 14-7 MCPWM Timing TH <n> 0 and TH <n> 1 - Complementary Mode

14.1.4.2   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.
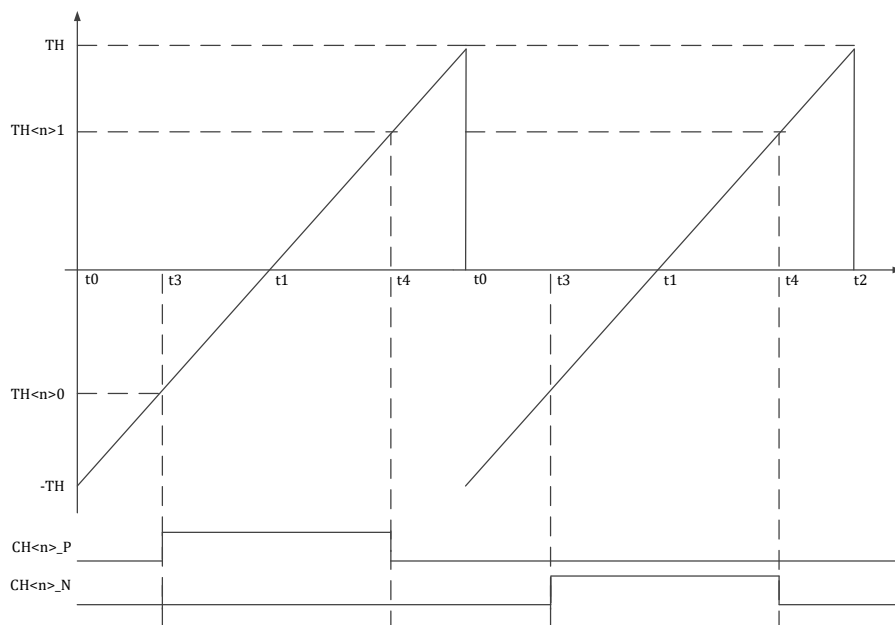
Fig. 14-8 MCPWM Timing TH<n>0 and TH<n>1 - Center-aligned Push-pull Mode

14.1.4.3   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.
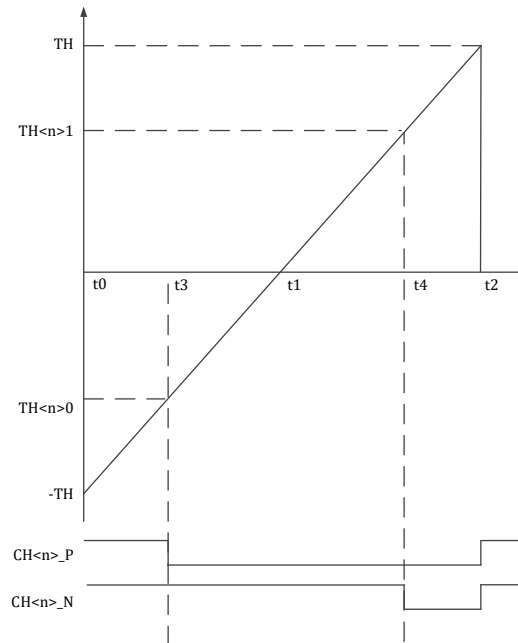
Fig. 14-9 MCPWM Timing Edge-aligned Mode

14.1.4.4   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.
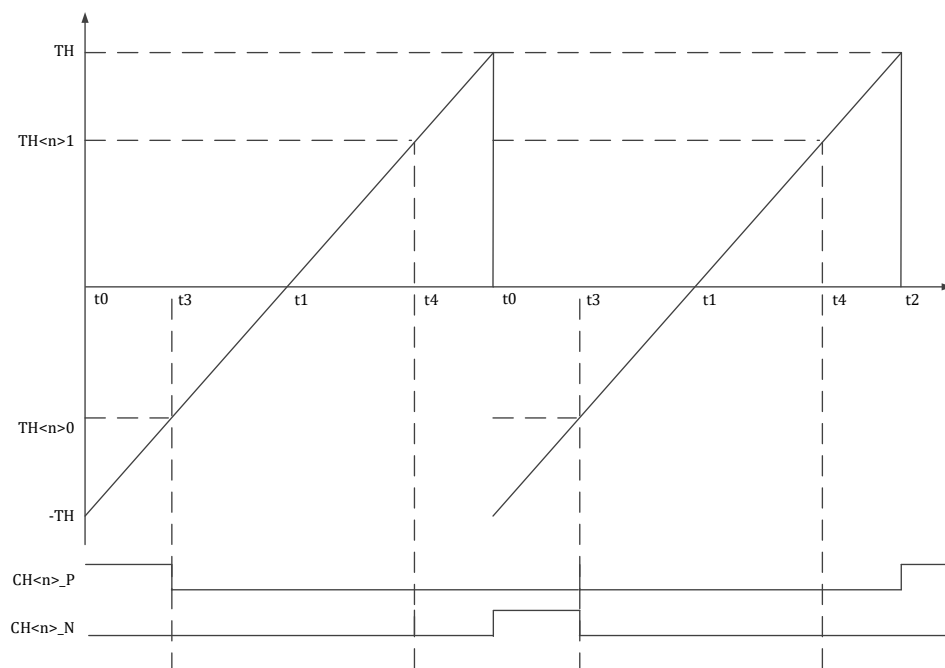
Fig. 14-10 MCPWM Timing TH <n> 0 and TH <n> 1 Edge-aligned Push-pull Mode

14.1.4.5   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.

The dead-zone width of the four groups of MCPWM IO can share the same dead-zone width setting MCPWM_DTH0/1.

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.
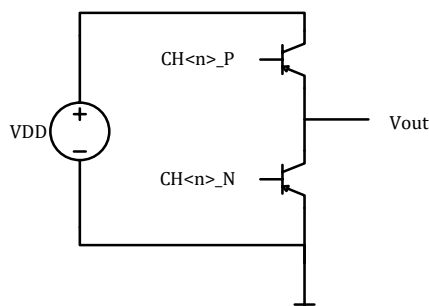
Fig. 14-11 MCPWM IO Control Diagram

14.1.4.6 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.

14.1.4.7 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_FAILx.CHxN_DEFAULT and MCPWM_FAILx.CHxP_DEFAULT control the default level).

> ➤ After the chip works normally, the default output level of IO is the specified value of register MCPWM_FAILx.CHxN_DEFAULT and MCPWM_FAILx.CHxP_DEFAULT. When the user configuration is completed and MCPWM works normally, set MCPWM_FAILx.MCPWM_OE (ie MOE) to 1, and the IO output level is controlled by MCPWM IO module.

> ➤ When a Fail short circuit condition occurs, the hardware switches to the IO default output level immediately.

> ➤ When the chip is debugged, MCU Halt, MCPWM stops output, and output the default level value.

14.1.5 ADC Trigger Timer Module

MCPWM can provide ADC sampling 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 14.2.27 MCPWM_TCLK.

Table 14-1 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

14.1.6   Interrupt

When MCPMW_IF0 and MCPWM_EIF [5:4] are set and enabled, MCPWM_IRQ0 interrupt will be generated;

When MCPMW_IF1 and MCPWM_EIF [7:6] are set and enabled, MCPWM_IRQ1 interrupt will be generated.

14.2 **Register**

14.2.1   Address Allocation

The base address of the MCPWM module register is 0x4001_0700,

and the register list is as follows:

Table 14-2 MCPWM Module Register List

Name	Offset Address	Description
MCPWM_TH00	0x00	MCPWM CH0_P compare threshold register
MCPWM_TH01	0x04	MCPWM CH0_N compare threshold register
MCPWM_TH10	0x08	MCPWM CH1_P compare threshold register
MCPWM_TH11	0x0C	MCPWM CH1_N compare threshold register
MCPWM_TH20	0x10	MCPWM CH2_P compare threshold register
MCPWM_TH21	0x14	MCPWM CH2_N compare threshold register
MCPWM_TH30	0x18	MCPWM CH3_P compare threshold register
MCPWM_TH31	0x1C	MCPWM CH3_N compare threshold register
MCPWM_TMR0	0x20	Compare threshold 0 register for ADC sampling timer
MCPWM_TMR1	0x24	Compare threshold 1 register for ADC sampling timer
MCPWM_TMR2	0x28	Compare threshold 2 register for ADC sampling timer
MCPWM_TMR3	0x2C	Compare threshold 3 register for ADC sampling timer
MCPWM_TH0	0x30	MCPWM time base 0 threshold register
MCPWM_TH1	0x34	MCPWM time base 1 threshold register
MCPWM_CNT0	0x38	MCPWM time base 0 counter register
MCPWM_CNT1	0x3C	MCPWM time base 1 counter register
MCPWM_UPDATE	0x40	MCPWM load control register
MCPWM_FCNT	0x44	CNT value at MCPWM FAIL
MCPWM_EVT0	0x48	MCPWM time base 0 external trigger
MCPWM_EVT1	0x4C	MCPWM time base 1 external trigger
MCPWM_DTH0	0x50	MCPWM CH0/1/2/3 N channel dead-zone width control register
MCPWM_DTH1	0x54	MCPWM CH0/1/2/3 P channel dead-zone width control register
MCPWM_FLT	0x70	MCPWM filter clock divider register
MCPWM_SDCFG	0x74	MCPWM load configuration register
MCPWM_AUEN	0x78	MCPWM auto update enable register
MCPWM_TCLK	0x7C	MCPWM clock divider control register

MCPWM_IE0	0x80	Interrupt control register for MCPWM time base 0
MCPWM_IF0	0x84	Interrupt flag register for MCPWM time base 0
MCPWM_IE1	0x88	MCPWM interrupt control register
MCPWM_IF1	0x8C	MCPWM interrupt flag register
MCPWM_EIE	0x90	MCPWM abnormal interrupt control register
MCPWM_EIF	0x94	MCPWM abnormal interrupt flag register
MCPWM_RE	0x98	MCPWM DMA request enable register
MCPWM_PP	0x9C	MCPWM push-pull mode enable register
MCPWM_IO01	0xA0	MCPWM CH0 CH1 IO control register
MCPWM_IO23	0xA4	MCPWM CH2 CH3 IO control register
MCPWM_FAIL012	0xA8	MCPWM CH0 CH1 CH2 short circuit control register
MCPWM_FAIL3	0xAC	MCPWM CH3 short circuit control register
MCPWM_PRT	0xB0	MCPWM protection register
MCPWM_SWAP	0xB4	MCPWM channel remapping register
MCPWM_CHMSK	0xB8	MCPWM channel masking register

Table 14-3 Registers Protected by MCPWM_PRT

Name	Offset Address	Description
MCPWM_TH0	0x30	MCPWM threshold register
MCPWM_TH1	0x34	MCPWM threshold register
MCPWM_DTH0	0x50	MCPWM CH0/1/2/3 N channel dead-zone width control register
MCPWM_DTH1	0x54	MCPWM CH0/1/2/3 P channel dead-zone width control register
MCPWM_FLT	0x70	MCPWM filter clock divider register
MCPWM_SDCFG	0x74	MCPWM load configuration register
MCPWM_AUEN	0x78	MCPWM auto load enable register
MCPWM_TCLK	0x7C	MCPWM clock divider control register
MCPWM_IE0	0x80	Interrupt control register for MCPWM time base 0
MCPWM_IE1	0x88	Interrupt control register for MCPWM time base 1
MCPWM_EIE	0x90	MCPWM abnormal interrupt control register
MCPWM_RE	0x98	MCPWM DMA request enable register
MCPWM_PP	0x9C	MCPWM push-pull mode enable register
MCPWM_IO01	0xA0	MCPWM CH0 CH1 IO control register
MCPWM_IO23	0xA4	MCPWM CH2 CH3 IO control register
MCPWM_FAIL012	0xA8	MCPWM CH0 CH1 CH2 short circuit control register
MCPWM_FAIL3	0xAC	MCPWM CH3 short circuit control register
MCPWM_CHMSK	0xB8	MCPWM channel masking register

Table 14-4 Registers with Shadow Registers

Name	Offset Address	Description
MCPWM_TH00	0x00	MCPWM CH0_P compare threshold register
MCPWM_TH01	0x04	MCPWM CH0_N compare threshold register
MCPWM_TH10	0x08	MCPWM CH1_P compare threshold register

MCPWM_TH11	0x0C	MCPWM CH1_N compare threshold register
MCPWM_TH20	0x10	MCPWM CH2_P compare threshold register
MCPWM_TH21	0x14	MCPWM CH2_N compare threshold register
MCPWM_TH30	0x18	MCPWM CH3_P compare threshold register
MCPWM_TH31	0x1C	MCPWM CH3_N compare threshold register
MCPWM_TMR0	0x20	Compare threshold 0 register for ADC sampling timer
MCPWM_TMR1	0x24	Compare threshold 1 register for ADC sampling timer
MCPWM_TMR2	0x28	Compare threshold 2 register for ADC sampling timer
MCPWM_TMR3	0x2C	Compare threshold 3 register for ADC sampling timer
MCPWM_TH0	0x30	MCPWM time base 0 threshold register
MCPWM_TH1	0x34	MCPWM time base 1 threshold register
MCPWM_CNT0	0x38	MCPWM time base 0 counter register
MCPWM_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.

14.2.2   MCPWM_TH00

Unprotected register

Address: 0x4001_0700

Reset value: 0x0

Table 14-5 MCPWM_TH00 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TH00															
RW															
0															

Location	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.

14.2.3   MCPWM_TH01

Unprotected register

Address: 0x4001_0704

Reset value: 0x0

Table 14-6 MCPWM_TH00 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TH01															

RW
0

Location	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.

14.2.4   MCPWM_TH10

Unprotected register

Address: 0x4001_0708

Reset value: 0x0

Table 14-7 MCPWM_TH10 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TH10															
RW															
0															

Location	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.

14.2.5   MCPWM_TH11

Unprotected register

Address: 0x4001_070C

Reset value: 0x0

Table 14-8 MCPWM_TH11 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TH11															
RW															
0															

Location	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.

14.2.6   MCPWM_TH20

Unprotected register

Address: 0x4001_0710

Reset value: 0x0

<div align="center">Table 14-9 MCPWM_TH20 Configuration Register</div>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TH20															
RW															
0															

Location	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.

14.2.7   MCPWM_TH21

Unprotected register

Address: 0x4001_0714

Reset value: 0x0

<div align="center">Table 14-10 MCPWM_TH21 Configuration Register</div>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TH21															
RW															
0															

Location	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.

14.2.8   MCPWM_TH30

Unprotected register

Address: 0x4001_0718

Reset value: 0x0

Table 14-11 MCPWM_TH30 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TH30															
RW															
0															

Location	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.

14.2.9   MCPWM_TH31

Unprotected register

   Address: 0x4001_071C

   Reset value: 0x0

Table 14-12 MCPWM_TH31 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TH31															
RW															
0															

Location	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.

14.2.10   MCPWM_TMR0

Unprotected register

   Address: 0x4001_0720

   Reset value: 0x0

Table 14-13 MCPWM_TMR0 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TMR0															
RW															
0x7FFF															

Location	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.

14.2.11 MCPWM_TMR1

Unprotected register

Address: 0x4001_0724

Reset value: 0x0

Table 14-14 MCPWM_TMR1 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TMR1															
RW															
0x7FFF															

Location	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.

14.2.12 MCPWM_TMR2

Unprotected register

Address: 0x4001_0728

Reset value: 0x0

Table 14-15 MCPWM_TMR2 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TMR2															
RW															
0x7FFF															

Location	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.

14.2.13 MCPWM_TMR3

Unprotected register

Address: 0x4001_072C

Reset value: 0x0

Table 14-16 MCPWM_TMR3 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TMR3															
RW															
0x7FFF															

Location	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.

14.2.14 MCPWM_TH0

Write-protected register

Address: 0x4001_0730

Reset value: 0x0

Table 14-17 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														

Location	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.

14.2.15 MCPWM_TH1

Write-protected register

Address: 0x4001_0734

Reset value: 0x0

153

Table 14-18 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								

Location	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.

14.2.16 MCPWM_CNT0

Unprotected register

Address: 0x4001_0738

Reset value: 0x0

Table 14-19 MCPWM_CNT0 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							CNT								
							RW								
							0x8000								

Location	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.

14.2.17 MCPWM_CNT1

Unprotected register

Address: 0x4001_073C

Reset value: 0x0

Table 14-20 MCPWM_CNT1 Register

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

CNT
RW
0x8000

Location	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.

14.2.18   MCPWM_UPDATE

Unprotected register

Address: 0x4001_0740

Reset value: 0x0

Table 14-21 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
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

Location	Bit Name	Description
[31:16]		Unused
[15]	CNT1_UPDATE	Write 1 to trigger by software (manually) to load the MCPWM_CNT1 from preload register to the shadow register used by MCPWM
[14]	CNT0_UPDATE	Write 1 to trigger by software (manually) to load the MCPWM_CNT0 from preload register to the shadow register used by MCPWM
[13]	TH1_UPDATE	Write 1 to trigger by software (manually) to load the MCPWM_TH1 from preload register to the shadow register used by MCPWM
[12]	TH0_UPDATE	Write 1 to trigger by software (manually) to load the MCPWM_TH0 from preload register to the shadow register used by MCPWM
[11]	TMR3_UPDATE	Write 1 to trigger by software (manually) to load the MCPWM_TMR3 from preload register to the shadow register used by MCPWM
[10]	TMR2_UPDATE	Write 1 to trigger by software (manually) to load the MCPWM_TMR2 from preload register to the shadow register used by MCPWM

[9]	TMR1_UPDATE	Write 1 to trigger by software (manually) to load the MCPWM_TMR1 from preload register to the shadow register used by MCPWM
[8]	TMR0_UPDATE	Write 1 to trigger by software (manually) to load the MCPWM_TMR0 from preload register to the shadow register used by MCPWM
[7]	TH31_UPDATE	Write 1 to trigger by software (manually) to load the MCPWM_TH31 from preload register to the shadow register used by MCPWM
[6]	TH30_UPDATE	Write 1 to trigger by software (manually) to load the MCPWM_TH30 from preload register to the shadow register used by MCPWM
[5]	TH21_UPDATE	Write 1 to trigger by software (manually) to load the MCPWM_TH21 from preload register to the shadow register used by MCPWM
[4]	TH20_UPDATE	Write 1 to trigger by software (manually) to load the MCPWM_TH20 from preload register to the shadow register used by MCPWM
[3]	TH11_UPDATE	Write 1 to trigger by software (manually) to load the MCPWM_TH11 from preload register to the shadow register used by MCPWM
[2]	TH10_UPDATE	Write 1 to trigger by software (manually) to load the MCPWM_TH10 from preload register to the shadow register used by MCPWM
[1]	TH01_UPDATE	Write 1 to trigger by software (manually) to load the MCPWM_TH01 from preload register to the shadow register used by MCPWM
[0]	TH00_UPDATE	Write 1 to trigger by software (manually) to load the MCPWM_TH00 from preload register to the shadow register used by MCPWM

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. No software cleanup is required. Every time 1 is written, a software/manual trigger is performed. Writing MCPWM_UPDATE[15:14] will cause MCPWM_CNT0/1 to be updated to the preload value. Please note whether CNT needs to be updated.

Writing 0 to MCPWM_UPDATE has no effect, but writing 0 immediately after writing 1 may cause the shadow register to fail to load. Therefore, writing 0 is not recommended.

14.2.19   MCPWM_FCNT

Unprotected register

Address: 0x4001_0744

Reset value: 0x0

Table 14-22 MCPWM_FCNT Register

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

Location	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.

14.2.20   MCPWM_EVT0

Unprotected register

Address: 0x4001_0748

Reset value: 0x0

Table 14-23 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
				TIMER1_CMP1	TIMER1_CMP0	TIMER0_CMP1	TIMER0_CMP0					MCPWM0_TMR3	MCPWM0_TMR2	MCPWM0_TMR1	MCPWM0_TMR0
				RW	RW	RW	RW					RW	RW	RW	RW
				0	0	0	0					0	0	0	0

Location	Bit Name	Description
[31:12]		Unused
[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
[7:4]		Unused
[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

14.2.21   MCPWM_EVT1

Unprotected register

Address: 0x4001_074C, 0x4001_384C

Reset value: 0x0

Table 14-24 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

	TIMER1_CMP1	TIMER1_CMP0	TIMER0_CMP1	TIMER0_CMP0		MCPWM0_TMR3	MCPWM0_TMR2	MCPWM0_TMR1	MCPWM0_TMR0
	RW	RW	RW	RW		RW	RW	RW	RW
	0	0	0	0		0	0	0	0

Location	Bit Name	Description
[31:12]		Unused
[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
[7:4]		Unused
[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

14.2.22 MCPWM_DTH0

Write-protected register

Address: 0x4001_0750

Reset value: 0x0

Table 14-25 MCPWM_DTH0 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTH0															
RW															
0															

Location	Bit Name	Description
[31:16]		Unused
[15:0]	DTH0	MCPWM CH0/1/2/3 P channel dead-zone width control register, 10-bit unsigned number

14.2.23 MCPWM_DTH1

Write-protected register

Address: 0x4001_0754

Reset value: 0x0

Table 14-26 MCPWM_DTH1 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTH1															
RW															
0															

Location	Bit Name	Description
[31:16]		Unused
[15:0]	DTH1	MCPWM CH0/1/2/3 N channel dead-zone width control register, 10-bit unsigned number

14.2.24　MCPWM_FLT

Write-protected register

Address: 0x4001_0770

Reset value: 0x0

Table 14-27 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							

Location	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 [1:0]. 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 [1: 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 TCLK clock widths.

14.2.25　MCPWM_SDCFG

Write-protected register

Address: 0x4001_0774

Reset value: 0x0

Table 14-28 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		

Location	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, 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.

14.2.26   MCPWM_AUEN

Write-protected register

Address: 0x4001_0778

Reset value: 0x0

Table 14-29 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

Location	Bit Name	Description
[31:16]		Unused
[15]	CNT1_AUPDATE	MCPWM_CNT1 auto load enable. 1: load; 0: no load.
[14]	CNT0_AUPDATE	MCPWM_CNT0 auto load enable. 1: load; 0: no load.
[13]	TH1_AUPDATE	MCPWM_TH1 auto load enable. 1: load; 0: no load.
[12]	TH0_AUPDATE	MCPWM_TH0 auto load enable. 1: load; 0: no load.
[11]	TMR3_AUPDATE	MCPWM_TMR3 auto load enable. 1: load; 0: no load.
[10]	TMR2_AUPDATE	MCPWM_TMR2 auto load enable. 1: load; 0: no load.
[9]	TMR1_AUPDATE	MCPWM_TMR1 auto load enable. 1: load; 0: no load.
[8]	TMR0_AUPDATE	MCPWM_TMR0 auto load enable. 1: load; 0: no load.
[7]	TH31_AUPDATE	MCPWM_TH31 auto load enable. 1: load; 0: no load.
[6]	TH30_AUPDATE	MCPWM_TH30 auto load enable. 1: load; 0: no load.
[5]	TH21_AUPDATE	MCPWM_TH21 auto load enable. 1: load; 0: no load.
[4]	TH20_AUPDATE	MCPWM_TH20 auto load enable. 1: load; 0: no load.
[3]	TH11_AUPDATE	MCPWM_TH11 auto load enable. 1: load; 0: no load.
[2]	TH10_AUPDATE	MCPWM_TH10 auto load enable. 1: load; 0: no load.
[1]	TH01_AUPDATE	MCPWM_TH01 auto load enable. 1: load; 0: no load.
[0]	TH00_AUPDATE	MCPWM_TH00 auto load enable. 1: load; 0: no load.

14.2.27   MCPWM_TCLK

Write-protected register

Address: 0x4001_077C

Reset value: 0x0

Table 14-30 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	

Location	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, and the MCPWM corresponding counter will start counting.

14.2.28 MCPWM_IE0

Write-protected register

Address: 0x4001_0780

Reset value: 0x0

Table 14-31 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

Location	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.

14.2.29 MCPWM_IF0

Unprotected register

    Address: 0x4001_0784

    Reset value: 0x0

Table 14-32 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

Location	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.

14.2.30 MCPWM_IE1

Write-protected register

Address: 0x4001_0788

Reset value: 0x0

Table 14-33 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

Location	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.

14.2.31 MCPWM_IF1

Unprotected register

     Address: 0x4001_078C

     Reset value: 0x0

Table 14-34 MCPWM Time Base 1 Interrupt Flag Register (MCPWM_IF1)

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

Location	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.

14.2.32 MCPWM_EIE

Write-protected register

Address: 0x4001_0790

Reset value: 0x0

Table 14-35 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				

Location	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

14.2.33 MCPWM_EIF

Unprotected register

Address: 0x4001_0794

Reset value: 0x0

Table 14-36 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				

Location	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

14.2.34 MCPWM_RE

Write-protected register

Address: 0x4001_0798

Reset value: 0x0

Table 14-37 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

Location	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.

14.2.35 MCPWM_PP

Write-protected register

Address: 0x4001_079C

Reset value: 0x0

Table 14-38 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

Location	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.

14.2.36   MCPWM_IO01

Write-protected register

Address: 0x4001_07A0

Reset value: 0x0

Table 14-39 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

Location	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. 1: interchangeable; 0: not interchangeable.
[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. 1: interchangeable; 0: not interchangeable.
[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.**

14.2.37　MCPWM_IO23

Write-protected register

　　Address: 0x4001_07A4

　　Reset value: 0x0

Table 14-40 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

Location	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. 1: interchangeable; 0: not interchangeable.
[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 P 来源。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. 1: interchangeable; 0: not interchangeable.
[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.**

14.2.38　MCPWM_FAIL012

Write-protected register

　　Address: 0x4001_07A8

　　Reset value: 0x0

Table 14-41 MCPWM_FAIL012 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FAIL_0CAP		CH2P_DEFAULT	CH2N_DEFAULT	CH1P_DEFAULT	CH1N_DEFAULT	CH0P_DEFAULT	CH0N_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

Location	Bit Name	Description
[31:16]		Unused
[15]	FAIL_0CAP	When a fail0/1 event occurs, the MCPWM_CNT0 value is stored to MCPWM_FCNT
[14]		Unused
[13]	CH2P_DEFAULT	CH2_P channel default value
[12]	CH2N_DEFAULT	CH2_N channel default value
[11]	CH1P_DEFAULT	CH1_ P channel default value
[10]	CH1N_DEFAULT	CH1_N channel default value
[9]	CH0P_DEFAULT	CH0_P channel default value
[8]	CH0N_DEFAULT	CH0_N 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 1 result; 0: From GPIO.
[0]	FAIL0_SEL	FAIL0 source selection. 1: Comparator 1 result; 0: From GPIO.

For MCPWM channel 0/1/2, FAIL0 signal can come from comparator 0 or MCPWM0_BKIN0, and FAIL1 signal can come from comparator 1 or MCPWM0_BKIN1.

The FAIL0/1 signal is used for MCPWM channel 0/1/2 working at time base 0.

14.2.39   MCPWM_FAIL3

Write-protected register

Address: 0x4001_07AC

Reset value: 0x0

Table 14-42 MCPWM_FAIL3 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FAIL_1CAP						CH3P_DEFAULT	CH3N_DEFAULT	HALT_PRT	MOE	FAIL1_EN	FAIL0_EN	FAIL1_POL	FAIL0_POL	FAIL1_SEL	FAIL0_SEL
RW						RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0						0	0	0	0	0	0	0	0	0	0

Location	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 1 result; 0: From GPIO channel 1.
[0]	FAIL2_SEL	FAIL2 source selection. 1: Comparator 0 result; 0: From GPIO channel 0.

For MCPWM channel 3, FAIL2 signal can come from comparator 0 or MCPWM0_BKIN0, and FAIL3 signal can come from comparator 1 or MCPWM0_BKIN1.

The FAIL2/3 signal is used for MCPWM channel 1 3 working at time base.

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 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.

14.2.40   MCPWM_PRT

Unprotected register

Address: 0x4001_07B0

Reset value: 0x0

Table 14-43 MCPWM_PRT Register

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

Location	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.

14.2.41   MCPWM_SWAP

Write-protected register, not open register

Address: 0x4001_07B4

Reset value: 0x0

Table 14-44 MCPWM_SWAP Register

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

Location	Bit Name	Description
[31:1]		Unused
[0]	MCPWM_SWAP	Writing 0x67 to this register will write 1 to BIT[0], and other values will write 0 to BIT[0].

If MCPWM_SWAP value is 0, the relationship between MCPWM channel output and GPIO is as follows:

Table 14-45 MCPWM Default Output IO

MCPWM Output Sequence	Corresponding GPIO Sequence
MCPWM_CH0P	P0.10
MCPWM_CH0N	P0.11
MCPWM_CH1P	P0.12
MCPWM_CH1N	P0.13
MCPWM_CH2P	P0.14
MCPWM_CH2N	P0.15
MCPWM_CH3P	P1.8
MCPWM_CH3N	P1.9

If MCPWM_SWAP value is 1, it covers the application environment of pre-driver chip. The sequence is changed within the logic to facilitate chip packaging. Generally, only three sets of MCPWM channels are needed in applications, so only these three sets of sequences are converted. The relationship is as follows:

Table 14-46 Output IO after MCPWM Channel Mapping

MCPWM Output Sequence	Corresponding GPIO Sequence
MCPWM_CH0N	P0.10
MCPWM_CH1N	P0.11
MCPWM_CH2N	P0.12
MCPWM_CH0P	P0.13
MCPWM_CH1P	P0.14
MCPWM_CH2P	P0.15
MCPWM_CH3P	P1.8
MCPWM_CH3N	P1.9

14.2.42   MCPWM_CHMSK

Write-protected register

Address: 0x4001_07B8

Reset value: 0x0

Table 14-47 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

Location	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

15 GPIO

15.1 Introduction

LSK32MC03x series chips integrate a total of 2 groups of GPIO, in which P0 includes 16 GPIOs and P1 includes 10 GPIOs. Some GPIOs can be used as system wake-up sources, and some GPIOs can be used as external interrupt source input.

15.1.1 Functional Block Diagram
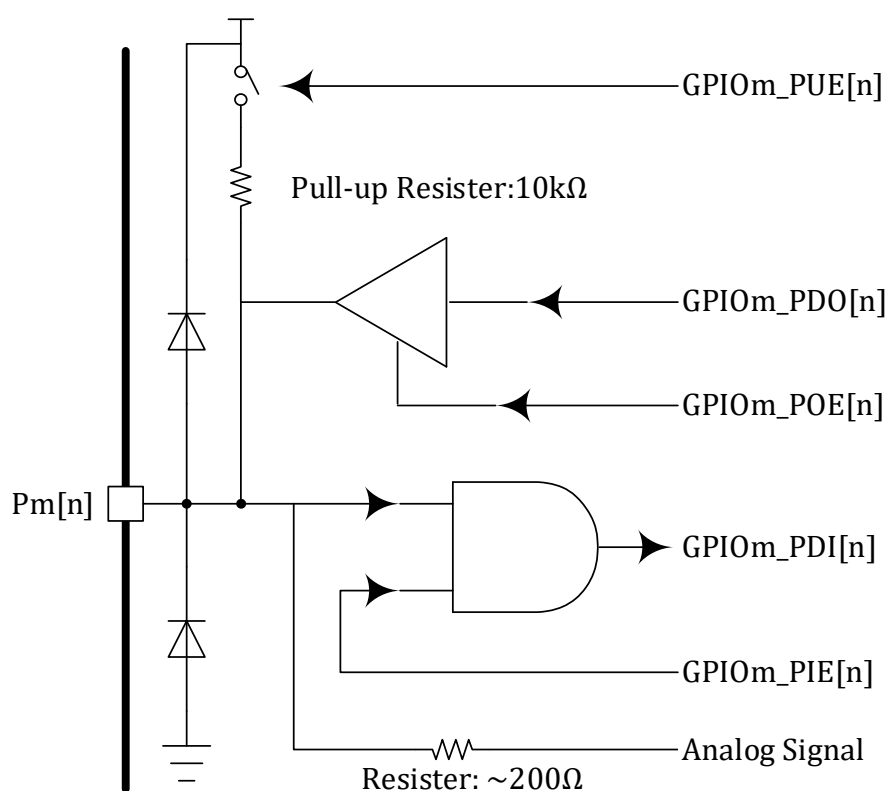
Fig. 15-1 GPIO Functional Block Diagram

As shown in Fig. 15-1, Pm [n] is the chip PAD, m can be 0 ~ 1, which means any group of two 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 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 level of GPIOm_PDI [n] is at the same level as Pm [n]. The chip PAD can be configured with pull-ups. The P0 [2] pin is multiplexed as the external reset pin RSTN. The pull-up resistor is 100kΩ and the remaining pull-up resistors are 10kΩ. Please note that not all PADs are equipped with pull-up resistors. For specific PADs with pull-up resistor resources, please refer to Section 15.2.6. The PAD without a pull-up resistor can also be set by the GPIOm_PUE [n] register, but it has no practical effect.

15.1.2   Features

➢   26-channel GPIO

➢   The push-pull open-drain mode can be configured

➢   Some GPIOs support external interrupt

➢   Some GPIOs can be used as wake-up sources

➢   Some GPIOs support input signal filtration

Please refer to 15.2.15.4 for which GPIOs support external interrupt

Please refer to 20.6.5 for which GPIOs can be used as wake-up sources

Please refer to 15.3.3 for which GPIOs can filter input signals.

15.2   **Register**

15.2.1   Address Allocation

The base address of the GPIO 0 module in the chip is 0x4001_0800.

The base address of the GPIO 1 module in the chip is 0x4001_0840.

Except for the base address, the register definitions of GPIO 0 and 1 are the same.

Table 15-1 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_PODE	0x18	GPIO x open-drain 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/wake-up/configuration lock module is 0x4001_0880.

Table 15-2 Register List of GPIO Interrupt/Wake-up/Configuration Lock Module

Name	Offset Address	Description
EXTI_CR0	0x00	External interrupt configuration register 0
EXTI_CR1	0x04	External interrupt configuration register 1
EXTI_IE	0x08	GPIO interrupt/DMA enable

EXTI_IF	0x0C	GPIO interrupt flag
CLKO_SEL	0x10	Output clock selection signal

15.2.2   GPIOx_PIE

The addresses are: 0x4001_0800 and 0x4001_0840.

Reset value: 0x0

Table 15-3 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

Location	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

15.2.3   GPIOx_POE

The addresses are: 0x4001_0804 and 0x4001_0844.

Reset value: 0x0

Table 15-4 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

Location	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

15.2.4 GPIOx_PDI

The addresses are: 0x4001_0808 and 0x4001_0848.

Reset value: 0x0

Table 15-5 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															

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

15.2.5 GPIOx_PDO

The addresses are: 0x4001_080C and 0x4001_084C.

Reset value: 0x0

Table 15-6 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								

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

15.2.6   GPIOx_PUE

The addresses are: 0x4001_0810 and 0x4001_0850.

Reset value: 0x0

Table 15-7 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

Location	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 |

Note that not all IOs have pull-up function. For details, see 15.3.2. The PUE registers corresponding to IOs without pull-up function are not implemented. Therefore, writing 1 to these registers are invalid, and the read value is always 0.

15.2.7   GPIOx_PODE

The addresses are: 0x4001_0818 and 0x4001_0858.

Reset value: 0x0

Table 15-8 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

Location	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

15.2.8   GPIOx_PFLT

The addresses are: 0x4001_081C and 0x4001_085C.

Reset value: 0x0

Table 15-9 GPIOx Configuration Lock 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

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

Only some GPIOs support input signal filtration. For details, see 15.3.3.

15.2.9   GPIOx_F3210

The addresses are: 0x4001_0820 and 0x4001_0860.

Reset value: 0x0

Table 15-10 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			

Location	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

The GPIO second function can be set for F3/F2/F1/F0 is as shown in Table 15-11. GPIO function is sparse mapping, that is, each GPIO can only be configured with part of the second function. For GPIO function distribution, please refer to the corresponding datasheet. The following GPIOx_F7654, GPIOx_FBA98, GPIOx_FFEDC are the same as GPIOx_F3210.

Table 15-11 GPIO Second Function

GPIO Second Function Value	Functional Description
0	Analog function. When GPIO is used as analog function, the corresponding IO output and pull-up functions shall be disabled, to prevent the IO output signal or in-chip pull-up resistance from interfering the analog signal.
1	CMP_OUT, analog comparator direct output/clock output
2	HALL, HALL signal input
3	MCPWM, MCPWM channel output or stop signal input
4	UART, serial port RXD or TXD
5	SPI, SPI clock, chip selection, data input/output
6	I2C, I2C clock, I2C data
7	TIMER0, TIMER0 channel 0/1, as input for capture mode or external clock sources and as output for comparison mode
8	TIMER1, TIMER1 channel 0/1, as input for capture mode or external clock sources and as output for comparison mode
9	ADC_TRIGGER, ADC sampling trigger signal output. Every time an ADC trigger occurs, the ADC_TRIGGER signal flips over once.

15.2.10   GPIOx_F7654

The addresses are: 0x4001_0824 and 0x4001_0864.

Reset value: 0x0

Table 15-12 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			

Location	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

15.2.11 GPIOx_FBA98

The addresses are: 0x4001_0828 and 0x4001_0868.

Reset value: 0x0

Table 15-13 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			

Location	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

15.2.12 GPIOx_FFEDC

The addresses are: 0x4001_082C and 0x4001_086C.

Reset value: 0x0

Table 15-14 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			

Location	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.

15.2.13 GPIOx_BSRR

The addresses are: 0x4001_0830 and 0x4001_0870.

Reset value: 0x0

Table 15-15 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

Location	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.

15.2.14   GPIOx_BRR

The addresses are: 0x4001_0834 and 0x4001_0874.

Reset value: 0x0

Table 15-16 GPIOx Bit Clear Register (GPIOx_BRR)

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

Location	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

15.2.15   External Event

EXTI_CR0 and EXTI_CR1 are used to select the level type externally triggered by the GPIO signal.

15.2.15.1 EXTI_CR0

Address: 0x4001_0880

Reset value: 0x0

Table 15-17 External Trigger Configuration Register 0 (EXTI_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	

Location	Bit Name	Description
[31:16]		Unused
[15:14]	T7	GPIO 0[9]/ P0[9] external 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 0[8]/P0[8] external 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 0[7]/ P0[7] external 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 0[6]/ P0[6] external 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 0[5]/ P0[5] external 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 0[4]/ P0[4] external 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 0[2]/ P0[2] external 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 0[0]/ P0[0] external trigger type selection 00: Not triggered 01: Falling edge trigger 10: Rising edge trigger 11: Trigger on both rising and falling edges

15.2.15.2 EXTI_CR1

Address: 0x4001_0884

Reset value: 0x0

Table 15-18 External Trigger Configuration Register 1 (EXTI_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	

Location	Bit Name	Description
[31:16]		Unused
[15:14]	T15	GPIO 1[9]/ P1[9] external 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 1[8]/ P1[8] external 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 1[7]/ P1[7] external 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 1[6]/ P1[6] external 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 1[5]/ P1[5] external 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 1[4]/ P1[4] external 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 0[15]/ P0[15] external 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 0[14]/ P0[14] external trigger type selection 00: Not triggered 01: Falling edge trigger 10: Rising edge trigger 11: Trigger on both rising and falling edges

Table 15-19 GPIO Interrupt Resource Distribution

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

15.2.15.3 EXTI_IE

Address: 0x4001_0888

Reset value: 0x0

Table 15-20 GPIO Interrupt Enable Register (EXTI_IE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						EXTI1_RE	EXTI0_RE							EXTI1_IE	EXTI0_IE
						RW	RW							RW	RW
						0	0							0	0

Location	Bit Name	Description
[31:10]		Unused
[9]	EXTI1_RE	GPIO1 DMA request enable, and must be used with EXTI_CR1
[8]	EXTI0_RE	GPIO0 DMA request enable, and must be used with EXTI_CR0
[7:2]		Unused
[1]	EXTI1_IE	GPIO1 interrupt enable, and must be used with EXTI_CR1
[0]	EXTI0_IE	GPIO0 interrupt enable, and must be used with EXTI_CR0

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 EXTIx_RE is enabled, DMA will clear all the interrupt flags of the GPIO after responding to the request.

15.2.15.4 EXTI_IF

Address: 0x4001_088C

Reset value: 0x0

Table 15-21 External Interrupt Flag Register (EXTI_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

Location	Bit Name	Description
[31:16]		Unused
[15]	IF15	GPIO 1[9] / P1[9] external interrupt flag. Interrupt flag is active high, write 1 to clear
[14]	IF14	GPIO 1[8] / P1[8] external interrupt flag. Interrupt flag is active high, write 1 to clear
[13]	IF13	GPIO 1[7] / P1[7] external interrupt flag. Interrupt flag is active high, write 1 to clear

[12]	IF12	GPIO 1[6] / P1[6] external interrupt flag. Interrupt flag is active high, write 1 to clear
[11]	IF11	GPIO 1[5] / P1[5] external interrupt flag. Interrupt flag is active high, write 1 to clear
[10]	IF10	GPIO 1[4] / P1[4] external interrupt flag. Interrupt flag is active high, write 1 to clear
[9]	IF9	GPIO 0[15] / P0[15] external interrupt flag. Interrupt flag is active high, write 1 to clear
[8]	IF8	GPIO 0[14] / P0[14] external interrupt flag. Interrupt flag is active high, write 1 to clear
[7]	IF7	GPIO 0[9] / P0[9] external interrupt flag. Interrupt flag is active high, write 1 to clear
[6]	IF6	GPIO 0[8] / P0[8] external interrupt flag. Interrupt flag is active high, write 1 to clear
[5]	IF5	GPIO 0[7] / P0[7] external interrupt flag. Interrupt flag is active high, write 1 to clear
[4]	IF4	GPIO 0[6] / P0[6] external interrupt flag. Interrupt flag is active high, write 1 to clear
[3]	IF3	GPIO 0[5] / P0[5] external interrupt flag. Interrupt flag is active high, write 1 to clear
[2]	IF2	GPIO 0[4] / P0[4] external interrupt flag. Interrupt flag is active high, write 1 to clear
[1]	IF1	GPIO 0[2] / P0[2] external interrupt flag. Interrupt flag is active high, write 1 to clear
[0]	IF0	GPIO 0[0] / P0[0] external interrupt flag. Interrupt flag is active high, write 1 to clear

15.2.15.5 CLKO_SEL

Address: 0x4001_0890

Reset value: 0x0

Table 15-22 GPIO Output Clock Signal Selection Register (CLKO_SEL)

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

Location	Bit Name	Description
[31:4]		Unused
[3]	ADC_OE	ADC output enable. 1: enable; 0: disable.
[2]	PLL_OE	PLL output enable. 1: enable; 0: disable.
[1]	HSI_OE	HSI output enable. 1: enable; 0: disable.
[0]	LSI_OE	LSI output enable. 1: enable; 0: disable.

The clock signals of the chip can be output and observed from P0.10. Note that because the ADC/PLL is a high-frequency clock signal, it may not be able to drive a large load outside the chip.

If you need to output the clock, you need to configure the second function of P0.10 to 1, that is, GPIO0_FBA98 = 0x0100, and enable the P0.10 output enable GPIO0_POE=0x0400, and configure CLKO_SEL to select a clock for output through P0.10.

15.3 **Implementation Description**

15.3.1   Pull-up

For LKS32MC03x series chips, some GPIOs have 10kΩ pull-up resistance. The GPIOs equipped with the pull-up function are as follows:

Table 15-23 GPIO Pull-up Resource Distribution

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

15.3.2   Filter

For LKS32MC03x 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 15-24 GPIO Filter Resource Distribution

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

15.4 **Application Guide**

15.4.1   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

EXTI_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*

EXTI_CR0 = 0x4000;   *// Enable p0 [7] falling edge trigger and generate external interrupt*

while(irq_flag != 4);

EXTI_CR0 = 0xC000;   *// Enable P0 [7] rising edge and falling edge trigger at the same*

                                                       time, and generate external interrupt

while(irq_flag != 8);

EXTI_CR0 = 0x0000   *// Disable P [7] rising and falling edge trigger at the same time,*

                                                       external interrupt cannot be generated

15.4.2   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.

16 UART

16.1 **Introduction**

Universal Asynchronous Receiver/Transmitter (UART) is a kind of asynchronous receiver/transmitter.

UART features are as follows:

- o   Support full-duplex operation

- o   Support single-line half-duplex operation

- o   Support 8/9 data bit

- o   Support 1/2 stop bit

- o   Support odd/even/no parity mode

- o   1 byte tx buffer

- o   1 byte rx buffer

- o   Support Multi-drop Slave/Master mode

16.2 **Function Description**

16.2.1   Transport (TX)

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.

16.2.2   Receive (RX)

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.

16.2.3   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.

8bit data byte length, 1 stop bit, lsb first

9bit data byte length, 1 stop bit, lsb first

9bit data byte length, 2 stop bit, lsb first

8bit data byte length, 1 stop bit, msb first, no parity bit

9bit data byte length, 1 stop bit, lsb first, no parity bit

8bit data byte length, 1 stop bit, msb first, 1 parity bit

9bit data byte length, 1 stop bit, lsb first, 1 parity bit
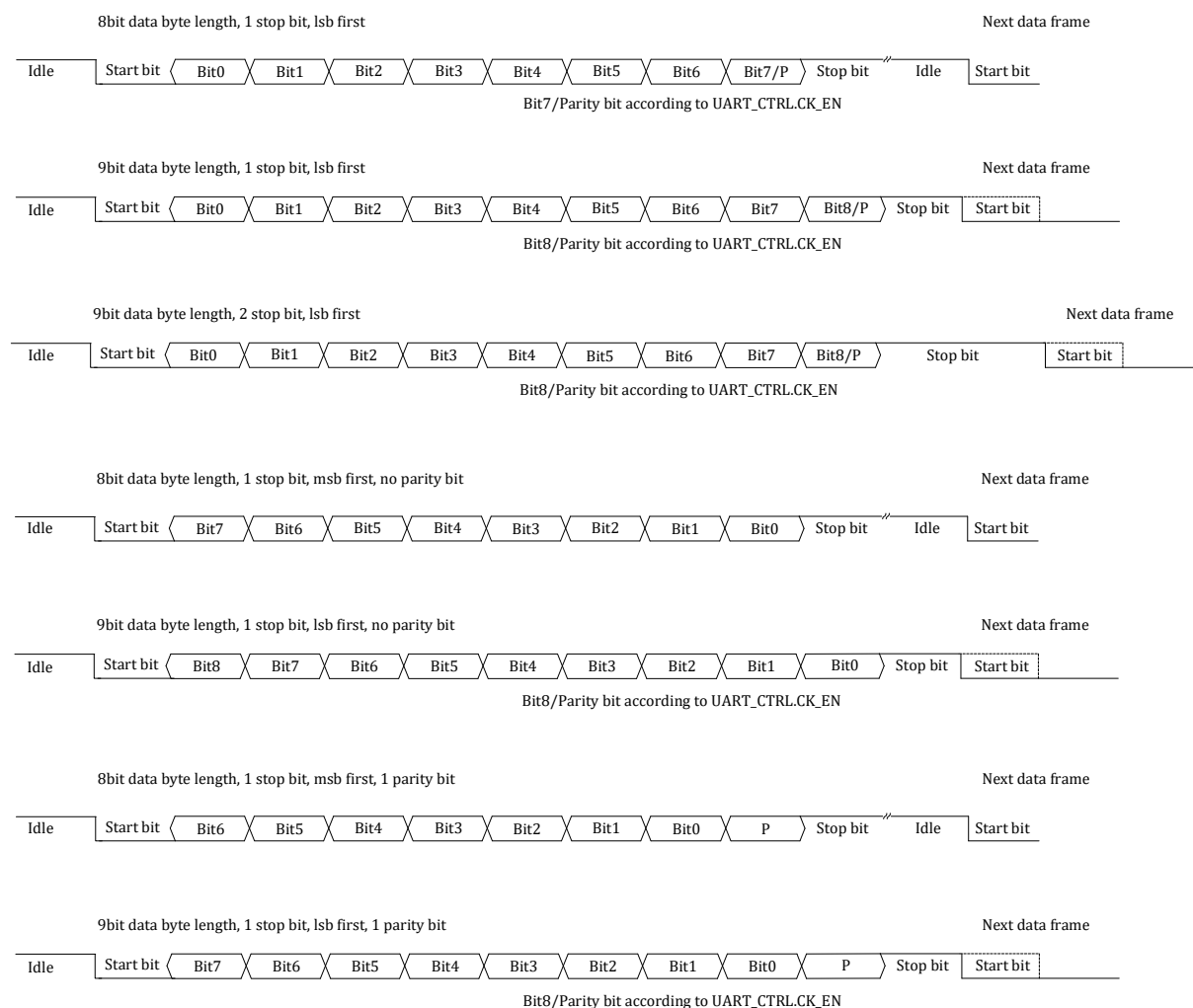
Fig. 16-1 UART Frame Format

16.2.4   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 16-1 Example of UART Baud Rate Configuration

UART Baud Rate	SYS_CLK_DIV2	UART_DIVH	UART_DIVL
150	0x0007	0x9C	0x3F
300	0x0003	0x9C	0x3F
600	0x0001	0x9C	0x3F
1200	0x0000	0x9C	0x3F
2400	0x0000	0x4E	0x1F

4800	0x0000	0x27	0x0F
9600	0x0000	0x13	0x87
19200	0x0000	0x09	0xC3
38400	0x0000	0x04	0xE1
57600	0x0000	0x03	0x40
115200	0x0000	0x01	0x9F

Note: The baud rate configuration factor is only an example and may not be unique.

16.2.5   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.

16.2.6   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 Fig. 162, it shows the interconnection of one master and three slaves (the addresses are 0x51,0x52, and 0x53 respectively).
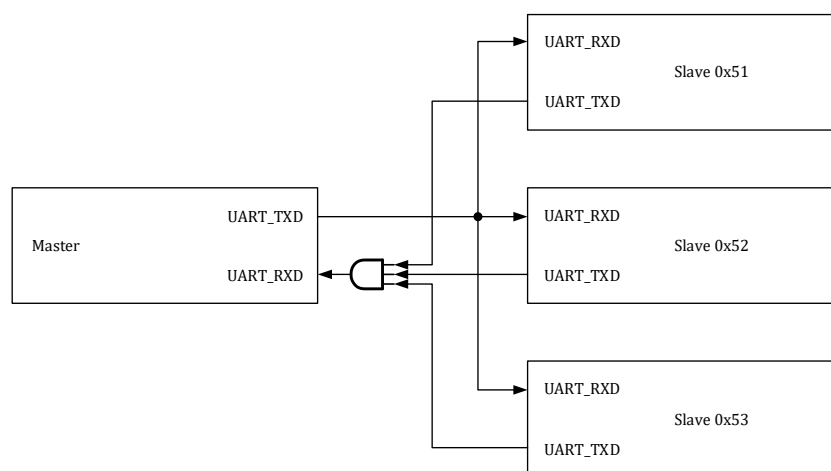
Fig. 16-2 UART Multi-computer Communication Connection Topology

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. Each slave has an 8-bit device address, i.e. UART_ADR. Both the master and slave need to set UART_CTRL.MD_EN=1, then enable the multi-computer communication

filter mode, and set UART_CTRL.BYTE_LEN=1 to use 9-bit data length. 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 also needs to set UART_CTRL.MD_EN = 1. When the slave sends data to the master, it shall first send a 9-bit data (1 + master UART_ADR) to hit the master address, and send all subsequent data with highest bit of 0 directly, without needing to send the address byte with the highest bit of 1 again. Other slaves can also send a 9-bit data (1 + master UART_ADR) first to hit the master address, and then send data byte with highest bit of 0, or send the data byte directly when the master address is hit.

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.

Note: The multi-computer communication does not currently support parity bit and MSB first mode. When setting UART_CTRL.MD_EN=1, please do not set UART_CTRL.CK_EN=1, or set UART_CTRL.BIT_ORDER=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.

As shown in Fig. 163, 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. The master device sends the address byte 0x151, and the slave device 0x51 is selected. 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.
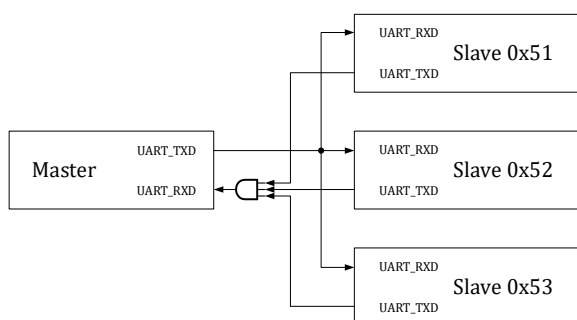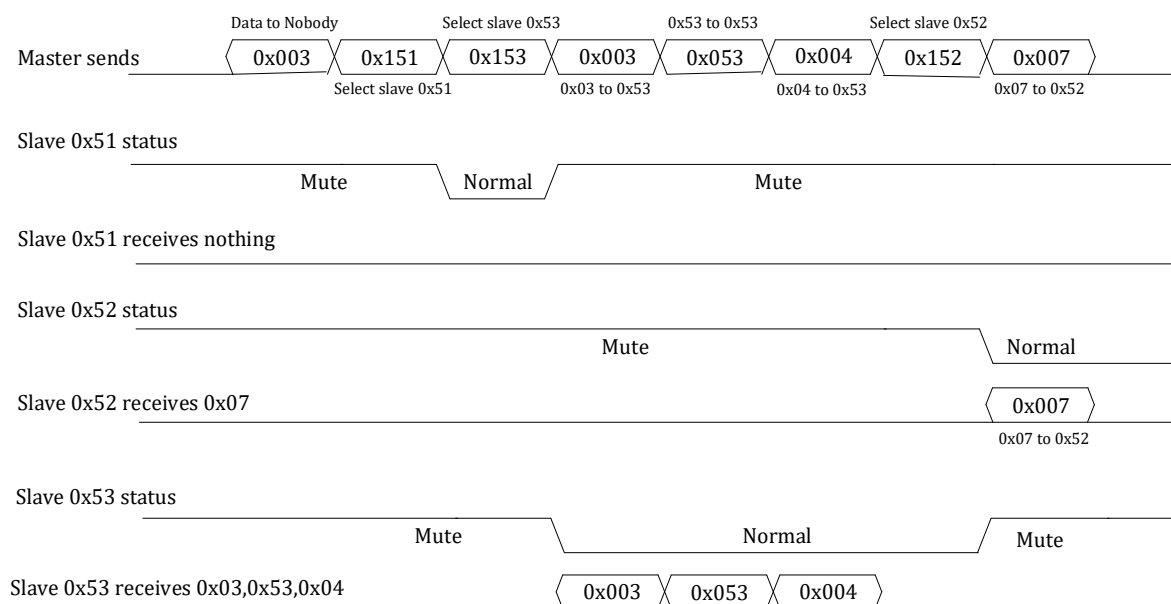
Fig. 16-3 Example of UART Multi-computer Communication

16.2.7   Parity 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. The setting of UART_CTRL.BIT_ORDER, i.e. LSB first or MSB first, does not affect the frame format.

Table 16-2 UART Frame Format

BYTE_LEN	CK_EN	UART frame
0	0	| StartBit | 8bit data BIT[7:0] | StopBit |
0	1	| StartBit | 7bit data BIT[6:0] | Parity | StopBit |
1	0	| StartBit | 9bit data BIT[8:0] | StopBit |
1	1	| StartBit | 8bit data BIT[7:0] | Parity | StopBit |

16.3 **Register**

16.3.1   Address Allocation

The base address of UART is 0x40010900.

<p align="center">Table 16-3 UART Address Distribution List</p>

Name	Offset Address	Description
UART_CTRL	0x00	UART control register
UART_DIVH	0x04	High byte register with UART baud rate setting
UART_DIVL	0x08	Low byte register with UART baud rate setting
UART_BUFF	0x0C	UART transceiver buffer register
UART_ADR	0x10	485 communication address matching register
UART_STT	0x14	UART status register
UART_RE	0x18	UART DMA request enable register
UART_IE	0x1C	UART interrupt enable register
UART_IF	0x20	UART interrupt flag register
UART_IOC	0x24	UART IO control

16.3.2   UART Control Register (UART_CTRL)

Address: 0x4001_ 0900

Reset value: 0x0

<p align="center">Table 16-4 UART Control Register (UART_CTRL)</p>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									Resv.	MD_EN	CK_EN	CK_TYPE	BIT_ORDER	STOP_LEN	BYTE_LEN
									RW	RW	RW	RW	RW	RW	RW
									0	0	0	0	0	0	0

Location	Bit Name	Description
[31:7]		Unused
[6]		Reserved
[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

16.3.3   UART Baud Rate High-byte Register (UART_DIVH)

Address: 0x4001_0904

Reset value: 0x0

Table 16-5 UART Baud Rate High-byte Register (UART_DIVH)

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

Location	Bit Name	Description
[31:8]		Unused
[7:0]	DIVH	Baud rate setting high byte BAUDRATE =main clock/(1+256* UART_DIVH+UART_DIVL)

16.3.4   UART Baud Rate Low-byte Register (UART_DIVL)

Address: 0x4001_0908

Reset value: 0x0

Table 16-6 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							

Location	Bit Name	Description
[31:8]		Unused
[7:0]	DIVL	Baud rate setting low byte BAUDRATE =main clock/(1+256* UART_DIVH+UART_DIVL)

201

16.3.5   UART Transceiver Buffer Register (UART_BUFF)

Table 16-7 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								

Location	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.

16.3.6   UART Address Match Register (UART_ADR)

Address: 0x4001_0910

Reset value: 0x0

Table 16-8 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							

Location	Bit Name	Description
[31:8]		Unused
[7:0]	ADR	Slave address during multi-computer communication

16.3.7   UART Status Register (UART_STT)

Address: 0x4001_0914

Reset value: 0x0

Table 16-9 UART Status Register (UART_STT)

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

Location	Bit Name	Description
[31:3]		Unused
[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.

16.3.8   UART DMA Request Enable Register (UART_RE)

Address: 0x4001_0918.

Reset value: 0x0

Table 16-10 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

Location	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.

16.3.9   UART Interrupt Enable Register (UART_IE)

Address: 0x4001_091C

Reset value: 0x0

Table 16-11 UART Interrupt Enable Register (UART_IE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									Resv.	Resv.	CK_ERR_IE	Stop bit error interrupt switch. The	TX_BUF_EMPTY_IE	RX_DONE_IE	TX_DONE_IE
									RW	RW	RW	RW	RW	RW	RW
									0	0	0	0	0	0	0

Location	Bit Name	Description
[31:7]		Unused
[6:5]		Reserved, must be 0
[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

16.3.10   UART Interrupt Flag Register (UART_IF)

Address: 0x4001_0920

Reset value: 0x0

Table 16-12 UART Interrupt Flag Register (UART_IF)

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

					CK_ERR_IF	STOP_ERR_IF	TX_BUF_EMPTY_IF	RX_DONE_IF	TX_DONE_IF
					RW1C	RW1C	RW1C	RW1C	RW1C
					0	0	1	0	1

Location	Bit Name	Description
[31:7]		Unused
[6:5]		Reserved
[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;

16.3.11 UART IO Control Register (UART_IOC)

Address: 0x4001_0924

Reset value: 0x0

Table 16-13 UART IO Control Register (UART_IOC)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											Resv.			TXD_INV	RXD_INV
											RW			RW	RW
											0			0	0

Location	Bit Name	Description
[31:5]		Unused
[4]		Reserved
[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".

17 DMA

17.1 Introduction

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

As shown in Fig. 17-1. 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.
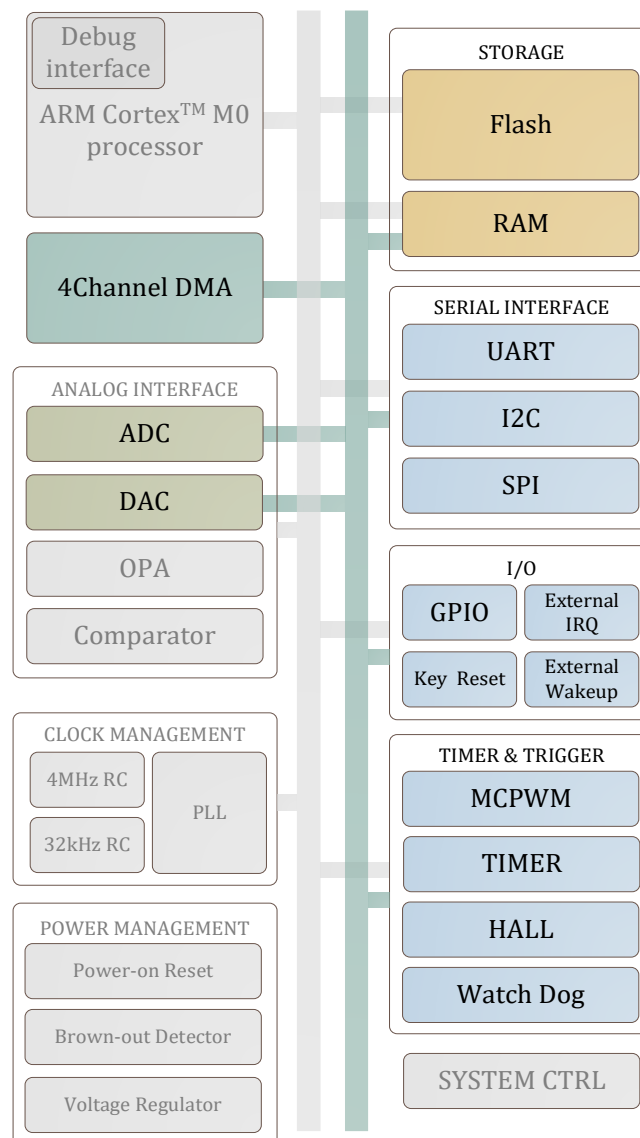
Fig. 17-1 Multi-layer AHB Lite Bus Architecture

DMA has four 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 the handling and DMA is idle, the arbitration will be started to determine which channel will receive the request next.

There are two ways for DMA transmission: DMA_CCRx.RMODE=1, multiple rounds, one transmission in each round of transmission; DMA_CCRx.RMODE=0, one round, continuous transmission

multiple times.

If it is configured to transmit data multiple times in one round, that is, DMA_CCRx.RMODE=0, then a DMA request, DMA continuously sends DMA_CTMSx data, and then the hardware clears the DMA_CCRx.EN bit and sets the interrupt flag.
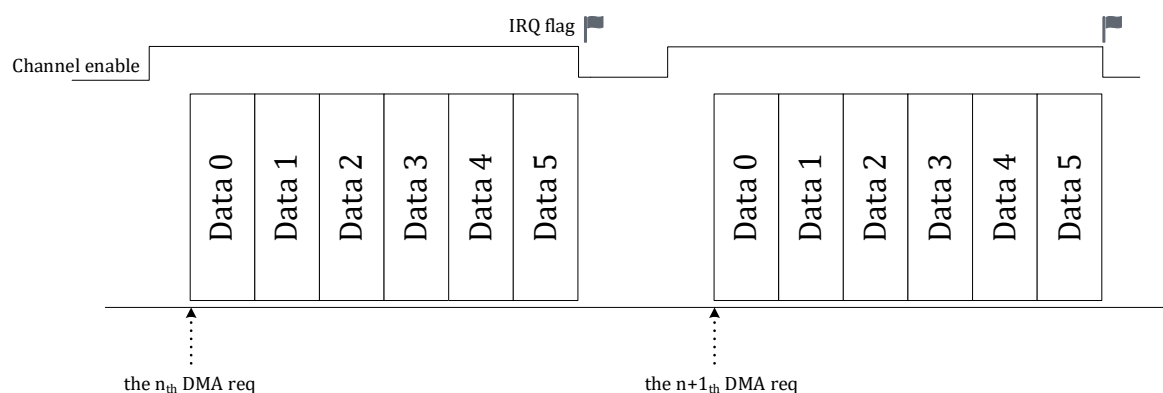
Fig.17-1 RMODE=0 DMA transmission

If a channel is configured for multi-round transfers, the DMA may respond to transfer requests from other channels between rounds of transfers.

Multi-round transmission only transfers data once per round. After transferring the DMA_CTMSx round, the hardware clears the DMA_CCRx.EN bit and sets the interrupt flag.
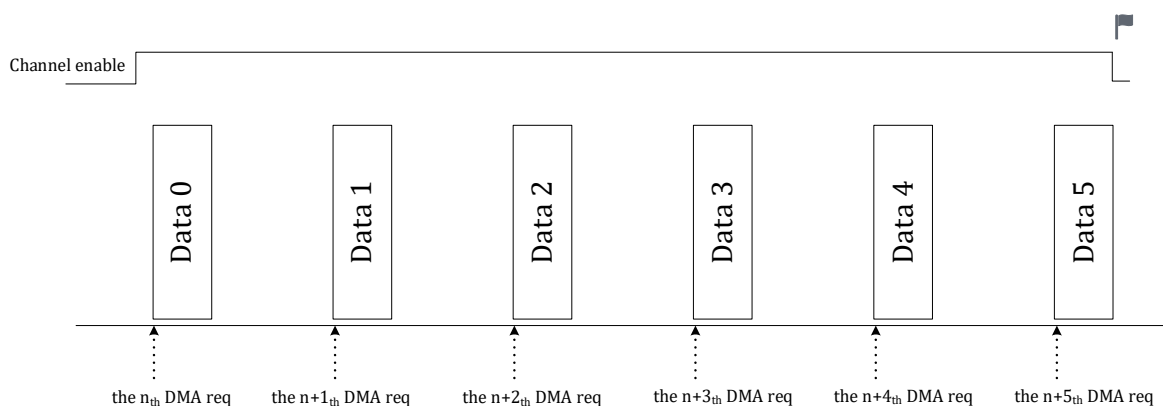
Fig.17-2 RMODE=1 DMA transmission

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

Every time a DMA request occurs, DMA starts a round of transmission. After completing one (multiple) rounds of transmission, it responds to the next request according to the current DMA channel request. The next request can still be the current channel, it can also be other channels. If without request, the channel will enter the idle state, and the rounds to be transmitted for current DMA channel will reduce one at the same time. If the number of rounds to be transmitted is 0, DMA will generate an interrupt after completing the transmission of current channel, and the DMA channel enable DMA_CCRx.EN is automatically cleared by the hardware. During the transmission, the number of rounds to be transmitted can be read through the DMA_CTMSx register. The number of rounds to be transmitted cannot be read during the transmission, because DMA transmits continuously and the value changes quickly.

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 channel before turning off the DMA

enable), at which time the DMA clock is gated off.

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.

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.
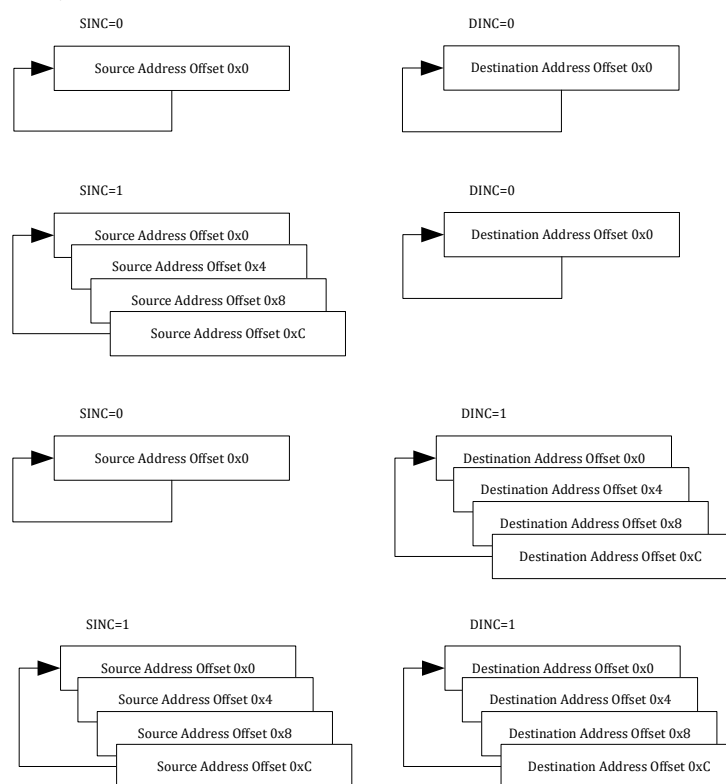
Fig. 17-2 DMA Address Increment Control

Fig. 17-2 only shows the address incremental example. It is configured to transmit 4 times in one round, or transmit 4 rounds, 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.

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.

DMA has two modes: cyclic mode and single mode, which are controlled by DMA_CCRx.CIRC. In cyclic mode, DMA completes the transfer of a certain size of data blocks and restarts the next round of transfer. 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 UART data to the memory as an example, the DMA completes a round (one byte) of transfers in the cyclic mode, and then restarts the next round of transfer, without setting the DMA completion interrupt flag.

Note: If address incremental is set in cyclic mode, and DMA_CCRx.SINC or DMA_CCRx.DINC=1, DMA uses a fixed address space of 256 bytes, and returns to 0 when the value reaches 256 bytes. Generally, the cyclic mode is used to transfer UART/SPI/I2C data to the memory. After querying DMA_CTMSx that the transmission data has accumulated enough length, for example, the length of a data frame is reached, the CPU performs data processing. Therefore, it is necessary to create 256-byte char arrays in memory.

In single-shot mode, DMA completes the DMA operation after completing the transfer of a certain size data block, sets the DMA completion interrupt flag, 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. The specific rounds are controlled by the DMA_CTMS register.

17.2 **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 peripheral interrupt flag does not need to be cleared by software.

Table 17-1 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.
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.
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.
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.
Timer	Timer uses a zero-crossing/comparison/capture 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).
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.
GPIO	GPIO interrupt can be used as DMA request
HALL	HALL interrupt can be used as DMA request

17.3 **Priority**

The priority of DMA adopts fixed priority, the priority is shown as Fig. 173. 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 Fig. 173, the priority decreases from top to bottom. Among the 4 DMA channels, the priority relationship is: channel 0> channel 1> channel 2> channel 3 (> 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 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.
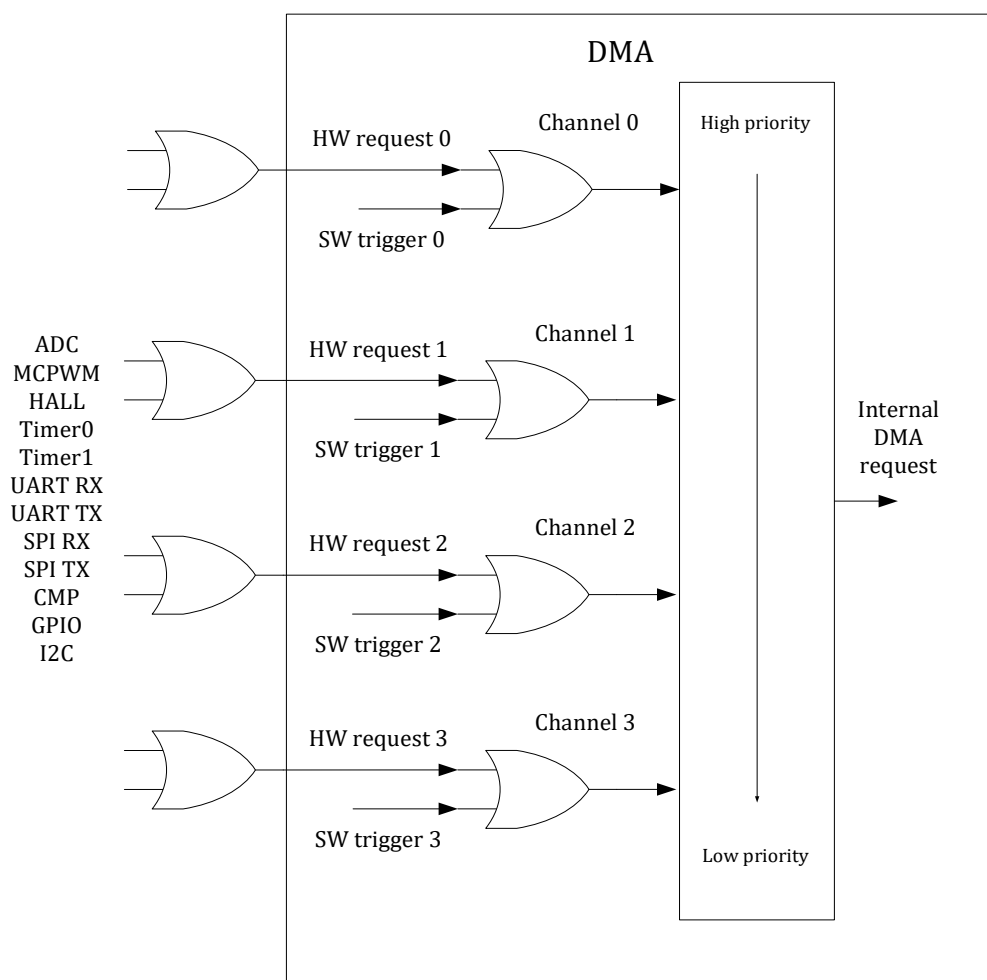
Fig. 17-3 DMA Channel Priority

17.4 **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/SPI/I2C, it will re-arbitrate for each byte transferred.

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.

17.5 **Interrupt**

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.

17.6 **Register**

17.6.1   Address Allocation

The base address of the DMA controller module register is 0x4001_0A00, and the register list is as follows:

Table 17-2 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
DMA_CCR1	0x20	DMA channel 1 configuration register
DMA_REN1	0x24	DMA channel 1 request enable register
DMA_CTMS1	0x28	DMA channel 1 transfer count register
DMA_SADR1	0x2C	DMA channel 1 source address register
DMA_DADR1	0x30	DMA channel 1 destination address register
DMA_CCR2	0x40	DMA channel 2 configuration register
DMA_REN2	0x44	DMA channel 2 request enable register
DMA_CTMS2	0x48	DMA channel 2 transfer count register
DMA_SADR2	0x4C	DMA channel 2 source address register
DMA_DADR2	0x50	DMA channel 2 destination address register
DMA_CCR3	0x60	DMA channel 3 configuration register
DMA_REN3	0x64	DMA channel 3 request enable register
DMA_CTMS3	0x68	DMA channel 3 transfer count register
DMA_SADR3	0x6C	DMA channel 3 source address register
DMA_DADR3	0x70	DMA channel 3 destination address register
DMA_CTRL	0x80	DMA control register
DMA_IE	0x84	DMA interrupt enable register
DMA_IF	0x88	DMA interrupt flag register

17.6.2   DMA Control Register (DMA_CTRL)

Address: 0x4001_0A80

Reset value: 0x0

Table 17-3 DMA Control Register (DMA_CTRL)

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

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

17.6.3 DMA Interrupt Enable Register (DMA_IE)

Address: 0x4001_0A84

Reset value: 0x0

Table 17-4 DMA Interrupt Enable Register (DMA_IE)

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

Location	Bit Name	Description
[31:4]		Unused
[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

17.6.4 DMA Interrupt Flag Register (DMA_IF)

Address: 0x4001_0A88

Reset value: 0x0

Table 17-5 DMA Interrupt Flag Register (DMA_IF)

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

				RW1C	RW1C	RW1C	RW1C
				0	0	0	0

Location	Bit Name	Description
[31:4]		Unused
[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.

17.6.5　DMA Channel Configuration Register

17.6.5.1　DMA_CCRx (where x =0,1,2,3)

The addresses are: 0x4001_0A00, 0x4001_0A18, 0x4001_0A30, and 0x4001_0A48.

Reset value: 0x0

Table 17-6 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

Location	Bit Name	Description
[31:12]		Unused
[11:10]	SBTW	Source address access bit width 0: Byte 1: Halfword 2: Word 3: Reserved
[9:8]	DBTW	Destination 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.

17.6.5.2 DMA_RENx (where x = 0,1,2,3)

The addresses are: 0x4001_0A04, 0x4001_0A1C, 0x4001_0A34, and 0x4001_0A4C.

Reset value: 0x0

Table 17-7 DMA Request Enable Register (DMA_ RENx)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SW_TRIG	I2C	GPIO	CMP	SPI TX	SPI RX			UART TX	UART RX	TIMER1	TIMER0		HALL	MCPWM	ADC
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

Generally, 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. The hardware request from the peripheral device enters the DMA and forms a request signal through the OR logic. Each DMA channel should 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.

17.6.5.3 DMA_CTMSx (where x = 0,1,2,3)

The addresses are: 0x4001_0A08, 0x4001_0A20, 0x4001_0A38, and 0x4001_0A50.

Reset value: 0x0

216

Table 17-8 DMA Transfer Count Register (DMA_CTMSx)

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

Location	Bit Name	Description
[7:0]	TMS	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_CTRL=1 and DMA_CCRx.EN=0, refilling the CTMSx value could clear the number of rounds which is already been sent by DMA.

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;

LKSMC03x does not support multiple transmission in each round.

When peripheral data width is 16 and memory data width is 32, that is, DMA_CTMSx=16, DMA_CCRx.RMODE=0, DMA should read peripheral data 16bit=2byte and write memory data 32bit=4byte each round. A total of two rounds of data need to be carried, 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, loop mode), DMA_CTMSx no longer works, which is equivalent to an infinite round; 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.

17.6.5.4   DMA_SADRx (where x = 0,1,2,3)

The addresses are: 0x4001_0A0C, 0x4001_0A24, 0x4001_ 0A3C, and 0x4001_0A54.

Reset value: 0x0

Table 17-9 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															

Location	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 peripheral address will be incremented by 2.

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

17.6.5.5   DMA_DADRx (where x = 0,1,2,3)

The addresses are: 0x4001_0A10, 0x4001_0A28, 0x4001_0A40, and 0x4001_0A58.

Reset value: 0x0

Table 17-10 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															

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

When DMA_CCRx.DBTW=2'b01, it is set to carry memory data in units of 16-bit. If the value of DADRx. ADDR[0] is invalid, the memory 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 DADRx. ADDR[1:0] is invalid, the memory 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!!!

18 DSP Coprocessor

18.1 Introduction

Both the dividend and the divisor of the divider are 32-bit signed numbers, and the calculation is completed in 32 bus cycles (up to 48MHz).

When the divisor is 0, the division result is uncertain.

Operands of the divider: the dividend and the divisor should be limited to $-(2^{31}-1) \sim (2^{31}-1)$, and assignment to -2^{31} should be avoided.

When divider is used, the software should first write the dividend DSP_DID and then the divisor DSP_DIS. The division operation is triggered when the divisor DSP_DIS is written. During the division operation, the software can execute other instructions. However, if the read result instruction (DSP_QUP or DSP_REM) is given before the division calculation is completed, CPU will wait to execute other instructions until the divider obtains the result. Writing a new operand (DSP_DID or DSP_DIS) to the divider before the division calculation is completed will also cause the CPU to enter the waiting state, and the new operand is not allowed to be written until the divider completes the current calculation. The above two cases have no other impact than causing the CPU to wait several bus cycles.

The radicand in SQRT module is a 32-bit unsigned number, the result is a 16-bit unsigned number, and the calculation is completed in 16 bus cycles (up to 48MHz).

When using the SQRT module, writing the radicand DSP_RAD will trigger the square root extractor to start operation, and the square root result can be read from the DSP_SQRT register after operation. During the square root operation, the software can execute other instructions. However, if the read result instruction (DSP_SQRT) is given before the calculation is completed, CPU will wait to execute other instructions until the SQRT obtains the result. Writing a new operand (DSP_RAD) before the square root operation is completed will also cause the CPU to enter the waiting state, and the new operand is not allowed to be written until the square root operation is completed. The above two cases have no other impact than causing the CPU to wait several bus cycles.

18.2 Register

18.2.1 Address Allocation

The base address of the divider register in the chip is 0x4001_0B00.

Table 18-1 DSP Register List

Name	Offset	Description
DSP_DID	0x20	DSP dividend
DSP_DIS	0x24	DSP divisor
DSP_QUO	0x28	DSP division quotient
DSP_REM	0x2C	DSP division remainder
DSP_RAD	0x30	DSP radicand
DSP_SQRT	0x34	DSP square root

18.2.2 DSP Division Registers

18.2.2.1 DSP_DID

Address: 0x4001_0B20

Reset value: 0x0

Table 18-2 Divider Register for Dividend

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

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

Location	Bit Name	Description
[31:0]	DID	Divider Register for Dividend

18.2.2.2 DSP_DIS

Address: 0x4001_0B24

Reset value: 0x0

Table 18-3 Divider Register for Divisor

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

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

Location	Bit Name	Description
[31:0]	DIS	Divider Register for Divisor

18.2.2.3 DSP_QUO

Address: 0x4001_0B28

Reset value: 0x0

<div align="center">Table 18-4 Division Quotient Register</div>

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

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

Location	Bit Name	Description
[31:0]	QUO	Division Quotient Register

18.2.2.4 DSP_REM

Address: 0x4001_0B2C

Reset value: 0x0

<div align="center">Table 18-5 Division Remainder Register</div>

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

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

Location	Permission	Description
[31:0]	REM	Division Remainder Register

18.2.3 DSP SQRT Registers

18.2.3.1 DSP_RAD

Address: 0x4001_0B30

Reset value: 0x0

Table 18-6 DSP SQRT Register for Radicand

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

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

Location	Bit Name	Description
[31:0]	RAD	DSP SQRT Register for Radicand

18.2.3.2 DSP_SQRT

Address: 0x4001_0B34

Reset value: 0x0

Table 18-7 DSP SQRT Register

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

Location	Bit Name	Description
[31:16]		Reserved bit. Always read as 0
[15:0]	SQRT	DSP SQRT Register

Write radicand to DSP to trigger an SQRT calculation; the 32-bit SQRT calculation could finish in several cycles. While reading the SQRT calculation result, DSP_SQRT, the CPU will enter a wait state. The calculation result will be returned through the bus after the SQRT calculation is done. Software could read immediately after writing DSP_RAD.

19 Independent Watchdog (IWDG)

19.1 Introduction

Independent watchdog (IWDG) uses 32kHz low-speed RC (LRC/LSI) clock for working, which can ensure normal operation even when the main clock of the system stops.

IWDG contains a 21-bit count-down counter internally, which starts to count down from the configured value after startup. IWDG can be configured to generate a system wake-up source as a scheduled wake-up source. A full chip reset signal can also be generated in case of timeout. When MCU enters deep sleep mode, the system clocks, including PLL/HRC, are turned off, while the LRC clock is always operating, so the IWDG can continue to work normally.

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, refresh and reset the IWDG before entering sleep; IWDG starts counting down from IWDG_RTH, and generates an IWDG wake-up signal after a period of time. If IWDG timing wake-up is not enabled, the chip can also be released from sleep when IWDG generates a full chip reset, but all MCU register configurations will be reset at the same time.
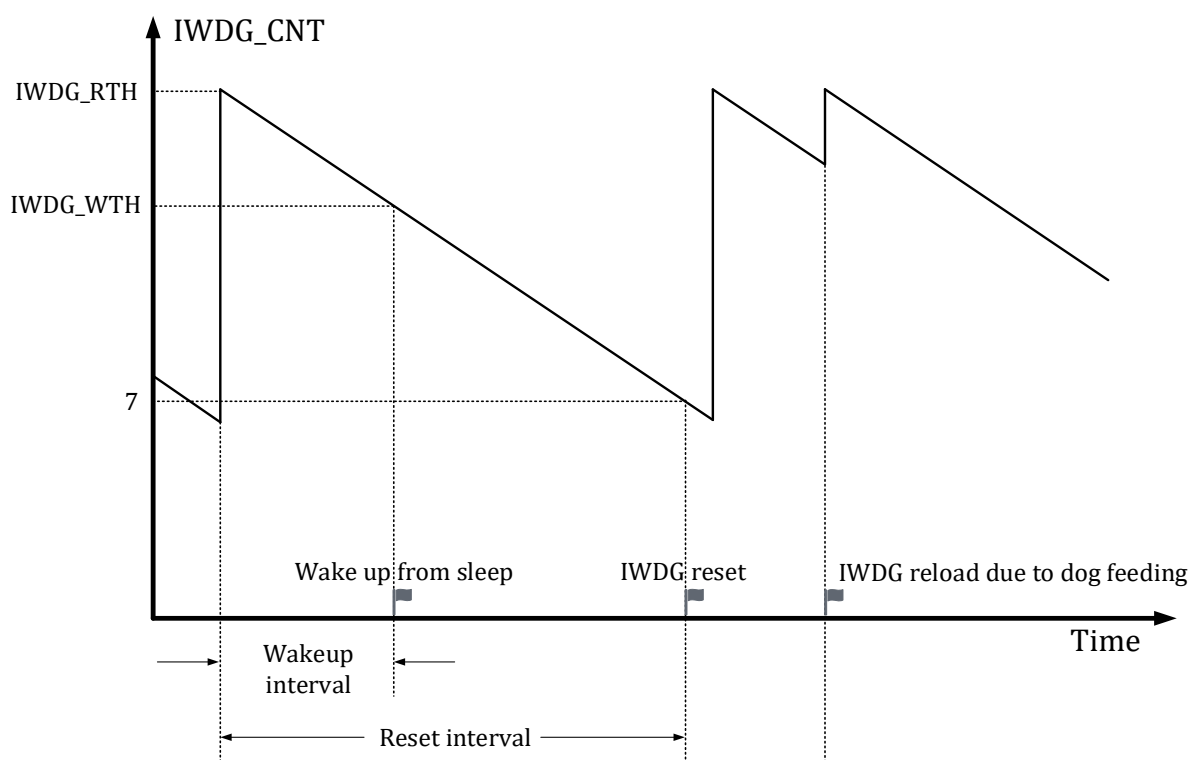

Fig. 19-1 IWDG counter timing

IWDG timeout reset time is calculated according to the following formula

Timeout$_{IWDG}$=t$_{LRC}$×(IWDG_RTH-7)

IWDG wake-up time is calculated according to the following formula

Wakeup$_{IWDG}$=t$_{LRC}$×(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 clock 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≈128ms.

IWDG_RTH=0x1FF000, the corresponding maximum reset interval of IWDG is 511×4096/32kHz≈64s.

Note that different chips have different values due to the limited accuracy of low-speed RC clock, and the deviation of the low-speed RC clock does not exceed ±50% in the full temperature range.

Regarding IWDG cross-clock feeding dogs, it should be noted that due to the limited accuracy of the low-speed RC clock, there is a certain deviation between different chips, and usually the low-speed RC clock deviation does not exceed ±50% over the whole temperature range.

Since IWDG counters work in the LRC clock domain, the software runs in the system primary clock domain, 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.

19.2　**Register**

19.2.1　Address Allocation

The base address of IWDG is 0x40010C00.

Table 19-1 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

　225

19.2.2   IWDG Password Register (IWDG_PSW)

Address: 0x4001_0C00

Reset value: 0x0

<p align="center">Table 19-2 IWDG Password Register (IWDG_PSW)</p>

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

Location	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 IWDG_CLR/ IWDG_RTH register of the watchdog.

19.2.3   IWDG Configuration Register (IWDG_CFG)

Address: 0x4001_0C04

Reset value: 0x1

<p align="center">Table 19-3 IWDG Configuration Register (IWDG_CFG)</p>

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

Location	Bit Name	Description
[31:5]		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.

19.2.4   IWDG Clear Register (IWDG_CLR)

Address: 0x4001_0C08

Reset value: 0x0

Table 19-4 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															

Location	Name	Description
[31:16]		Unused
[15:0]	IWDG_CLR	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.

19.2.5   IWDG Counter Timing Wake-up Threshold Register (IWDG_WTH)

Address: 0x4001_0C0C

Reset value: 0x001F_F000

Table 19-5 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															

Location	Name	Description
[31:21]	NA	Unused
[20:12]	IWDG_WTH	Timing wake-up threshold of IWDG. The watchdog uses 32kHz LRC clock to count down from IWDG_RTH and generate a wake-up signal when the value reaches IWDG_WTH.
[11:0]	R	The value is always 0

Writing to IWDG_WTH does not need to write a password to IWDG_PSW.

19.2.6   IWDG Counter Timeout Reset Threshold Register (IWDG_RTH)

Address: 0x4001_0C10

Reset value: 0x001F_F000

Table 19-6 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															

Location	Name	Description
[31:21]	NA	Unused
[20:12]	IWDG_RTH	The IWDG timeout reset threshold is also the reload value. The watchdog uses 32kHz LRC clock 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]	R	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. Writing to IWDG_RTG needs to write a password to IWDG_PSW first.

19.2.7   IWDG Current Count Register (IWDG_CNT)

Address: 0x4001_0C14

Reset value: 0x0000_0000

Table 19-7 IWDG Current Count Register (IWDG_CNT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
											IWDG_CNT				
											R				
											0x00				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IWDG_CNT															
R															
0x0000															

Location	Name	Description
[31:21]	NA	Unused
[20:0]	IWDG_CNT	Current count of the watchdog, which is ≤IWDG_TH.

20 Power Management Module (PMU)

The chip can reduce power consumption by reducing the running clock frequency and turning off part of the circuit clock.

20.1 Peripheral Clock Gating

The peripheral clock is gated or 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 6.3.7 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.**

20.2 Peripheral Clock Divider

Some peripherals have independent clock dividers, which enables it to work with an appropriate frequency.

Among them, SYS_CLK_DIV[0] is the division factor of I2C, while SYS_CLK_DIV[2] is the division factor of UART. 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.

20.3 Low Power Mode

Table 20-1 Summary of Low Power Modes

Mode	Enter Mode	Exit Mode	PLL/HSI	LSI
Sleep	WFI/WFE	Any interrupt Debug External reset IWDG reset	PLL/HSI On, CPU clock Off, NVIC/peripheral clock On	On
Deep Sleep	SLEEPDEEP + WFI/WFE	IWDG timing wake-up IO wake-up Debug External reset IWDG reset	PLL/HSI Off, CPU/peripheral clock Off	

20.4 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.

* It is recommended to refer to the official sleep routine configuration. Before sleep, you need to turn off the clock of all digital modules, and turn off analog ADC/OPA/CMP/DAC and other modules.

20.4.1   Enter Mode

The sleep mode can be entered by executing the WFI/WFE instruction. 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 instruction.

Sleep-on-exit: If SLEEPONEXIT is 1, CPU enters the sleep mode after completing all interruptions.

20.4.2   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.

20.5 Deep Sleep Mode

The deep sleep mode turns off all system high-speed clocks, including PLL/HSI, but the 32kHz RC clock 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.

20.5.1   Enter Mode

Set the System Control Register SLEEPDEEP of CPU core to 1, and execute the WFI/WFE instruction to enter deep sleep.

20.5.2   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.

20.6 **Register**

20.6.1   Address Allocation

The base address of PMU is 0x40010C20.

Table 20-2 PMU Address Space

Name	Offset	Description
AON_PWR_CFG	0x0	
AON_EVT_RCD	0x4	Event record register
AON_IO_WAKE_POL	0x8	IO wake-up polarity register
AON_IO_WAKE_EN	0xC	IO wake-up enable register

20.6.2   PMU Configuration Register (AON_PWR_CFG)

Address: 0x4001_0C20

Reset value: 0x0

Table 20-3 PMU Configuration Register AON_PWR_CFG

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

Location	Bit Name	Description
[31:2]		Unused
[1]	IOWK_FLT	IO wake-up signal filter enable. The default value is enable.
[0]		Unused

The software can choose whether to filter the IO wake-up signal. If the on-chip filter is enabled, the IO signal is sent to the PMU after being filtered with a 120us time constant. On-chip IO signal filtering can replace board-level filtering in certain application scenarios, but it will introduce a wake up delay of about 120us. If on-chip IO wake-up signal filtering is not enabled, the application scenario needs to guarantee that the IO wake-up input signal is stable without interference, and board-level filtering can be used to avoid unnecessary waste of power consumption due to the chip being woken up by mistake.

20.6.3   Event Record Register (AON_EVT_RCD)

Address: 0x4001_0C24

Reset value: 0x0

Table 20-4 Event Record Register (AON_EVT_RCD)

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

	DEEPSLEEP	SLEEP		IWDG_WK	IO_WK			IWDG_RST	KEY_RST		POR_RST
	RO	RO		RO	RO			RO	RO		RO
	0	0		0	0			0	0		1

Location	Bit Name	Description
[31:14]		Unused
[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 Deep Sleep is periodically woken up by IWDG
[8]	IO_WK	IO wake-up record, high value means Deep Sleep is periodically woken up by IO
[7:4]		Unused
[3]	IWDG_RST	IWDG reset occurrence record, high value indicates occurred
[2]	KEY_RST_RCD	Key reset occurrence record, high value indicates occurred
[1]		Unused
[0]	POR_RST_RCD	POR reset occurrence record, high value indicates occurred

Write 0xCA40 to AON_EVT_RCD register to clear all event records. Because EVT_RCD works in the LSI clock domain, it takes 32 us for the software to write the password to clear it.

20.6.4 IO Wake-up Polarity Register (AON_ IO_WAKE_POL)

Address: 0x4001_0C28

Reset value: 0x0

Table 20-5 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

Location	Bit Name	Description
[31:1]		Unused
[0]	WK_POL	IO external wake-up trigger level selection. 1: high level; 0: low level

All IO wake-up signals use the same polarity selection.

20.6.5   IO Wake-up Enable Register (AON_IO_WAKE_EN)

Address: 0x4001_0C2C

Reset value: 0x0

Table 20-6 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
								P1_9_EN	P1_8_EN	P1_5_EN	P0_10_EN	P0_9_EN	P0_8_EN	P0_2_EN	P0_0_EN
								RW	RW	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

Location	Bit Name	Description
[31:8]		Unused
[7]	P1_9_EN	P1[9] external wake-up trigger enable
[6]	P1_8_EN	P1[8] external wake-up trigger enable
[5]	P1_5_EN	P1[5] external wake-up trigger enable
[4]	P0_10_EN	P1[2] external wake-up trigger enable
[3]	P0_9_EN	P0[9] external wake-up trigger enable
[2]	P0_8_EN	P0[8] external wake-up trigger enable
[1]	P0_2_EN	P0[2] external wake-up trigger enable
[0]	P0_0_EN	P0[0] external wake-up trigger enable

21 Version History

Table 21-1 Document's Version History

Date	Version No.	Description
June 12, 2024	1.55	NVR calibration parameter information added in FLAH chapter
May 29, 2024	1.54	Revised description of PLL clock accuracy
April 3, 2024	1.53	DAC adds description of software correction
Mar 25,2024	1.52	Added the description of DAC for version C
Mar 13,2024	1.51	Optimizes the SYS_AFE_INFO register description
Jan 26,2024	1.50	Revised the description of the filter coefficient section for the MCPWM FAIL part
Jan 15,2024	1.49	Added a description about sleep mode
Jan 1,2024	1.48	UTIMER_CFG register CR1 bit information correction
September 25, 2023	1.47	Added a note on IWDG feed dog cross-clock synchronization issues Fixed MCPWM FAIL3 register description
August 30, 2023	1.46	Add description of the version number
August 15, 2023	1.45	Modified the GPIO functional block diagram
April 12, 2023	1.44	Modified the description of modifying OPA Added the description of ADC sampling time
February 11, 2023	1.43	Added instructions on closing PLL and other operations
December 18, 2022	1.42	Remove NVR related descriptions
November 28, 2022	1.41	Update the LRC clock frequency
November 24, 2022	1.4	Add instructions for resetting ADC module with SYS_SFT_RST
November 12, 2022	1.39	Update the LRC clock frequency and full temperature error range Modified the description of SYS_AFE_INFO.Version
November 7, 2022	1.38	Added connection resistance between IO and internal analog circuit
October 28, 2022	1.37	Added the description of SYS_AFE_INFO.Version Modified the description of MCPWM_DTH0/MCPWM_DTH1
October 25, 2022	1.36	Modified the description of DSP_DIS/DSP_RAD Modified the power supply
October 24, 2022	1.35	Added the description of TIMER SRC0/1 capture the same signal
October 14, 2022	1.34	Modified the LRC clock frequency and full temperature error range
September 29, 2022	1.33	Modified the description of SYS_OPA_SEL.OPA_SEL_EN and added write function
September 15, 2022	1.32	Added ADC/OPA/CMP configuration flow
July 29, 2022	1.31	Simplify SYS_AFE_REG description
December 29, 2021	1.3	SYS_FLSE, SYS_FLSP
June 23, 2021	1.0	Official version
May 8, 2021	0.4	Modified the description of ADC_CFG.FSM_RS and removed the

		read function Modified SYS_DBG_CFG
March 18, 2021	0.3	Modified LRC clock frequency
February 2, 2021	0.2	Completed the modification of all chapters except flash/i2c/spi
December 1m 2020	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.