Linko Semiconductor Co., Ltd.

LKS32MC08x User Manual

CONTENT

List of Tables

List of Figures

1 Document Convention

1.1 Register Read/Write Permissions

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 4MHz RC oscillator

HSE: High Speed External Clock, the 4~8MHz external crystal clock

PLL: Phase Lock Loop Clock, the 96MHz PLL clock, which usually used as high-speed system clock

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

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

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

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

CW: Clockwise

CCW: Counterclockwise

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

2 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 2-1 System Address Space Allocation

Peripheral Modules	Clock/Soft Reset	Start Address	End Address	Size	Description
FLASH	PLL/None	0x0000_0000	0x0000_FFFF	64kB	FLASH memory
RAM	PLL /None	0x2000_0000	0x2000_1FFF	8kB	RAM
SYS	PLL /None	0x4000_0000	0x4000_03FF	1kB	SYSTEM control, Clock/Reset Management
FLSCR	PLL /None	0x4000_0400	0x4000_07FF	1kB	FLASH control registers
SPI	FCLK[8]/sft_rst[0]	0x4001_0000	0x4001_03FF	1kB	SPI interface
I2C	FCLK[0]/sft_rst[0]	0x4001_0400	0x4001_07FF	1kB	I2C interface
CMP	PLL /None	0x4001_0C00	0x4001_0FFF	1kB	Comparator
HALL	FCLK[1]/sft_rst[1]	0x4001_1000	0x4001_13FF	1kB	HALL interface
ADC	ACLK	0x4001_1400	0x4001_17FF	1kB	ADC interface
TIMER	FCLK[2]/sft_rst[2]	0x4001_1800	0x4001_1BFF	1kB	General Purpose Timer
MCPWM	FCLK[3]/sft_rst[3]	0x4001_1C00	0x4001_1FFF	1kB	Motor Control Pulse Width Modulation
GPIO	PLL/None	0x4001_2000	0x4001_23FF	1kB	General Purpose Input/Output
CRC	PLL/None	0x4001_2400	0x4001_27FF	1kB	Cyclic Redundancy Check
UART0	FCLK[4]/ sft_rst[4]	0x4001_2800	0x4001_2BFF	1kB	
UART1	FCLK[5]/ sft_rst[5]	0x4001_2C00	0x4001_2FFF	1kB	
DMA	PLL/None	0x4001_3000	0x4001_33FF	1kB	
CAN	FCLK[7]/None	0x4001_3400	0x4001_37FF	1kB	
SIF	PLL/None	0x4001_3800	0x4001_3BFF	1kB	
DSP	FCLK[6]/无	0x4001_4000	0x4001_5FFF	8kB	

3 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.

It can support up to 32 independent interrupt sources and interrupt vectors, of which 21 interrupt sources are used in the LKS32MC08X series chips, and the last 11 are reserved.

It supports up to four interrupt priority levels for programming.

Table 3-1 Interrupt Number List

Interrupt No.	Description	Interrupt No.	Description
-14	NMI		
-13	HardFault		
-12	Reserved		
-11	Reserved		
-10	Reserved		
-9	Reserved		
-8	Reserved		
-7	Reserved		
-6	Reserved		
-5	SVCall		
-4	Reserved		
-3	Reserved		
-2	PendSV		
-1	SysTick		
0	TIMER0	16	WAKEUP, system wake-up interrupt
1	TIMER1	17	Low voltage
2	TIMER2	18	DMA
3	TIMER3	19	CAN
4	ENCODER0	20	SIF
5	ENCODER1	21	Reserved
6	I2C	22	Reserved
7	GPIO	23	Reserved
8	UART0	24	Reserved
9	HALL	25	Reserved
10	SPI	26	Reserved
11	ADC	27	Reserved
12	DSP	28	Reserved
13	MCPWM	29	Reserved
14	UART1	30	Reserved
15	CMP	31	Reserved

4 Analog Circuit

4.1 Introduction

The analog circuit contains the following modules:

- Built-in 12bit SAR ADC, simultaneous double sampling, 3Msps sampling and conversion rate, up to 20 analog signal channels
- Built-in 4 operational amplifiers. Differential PGA mode is available.
- Built-in 2 comparators. Hysteresis mode is available.
- Built-in 12bit digital-to-analog converter (DAC)
- ± 2 °C built-in temperature sensor
- 1.2V 0.8% 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.

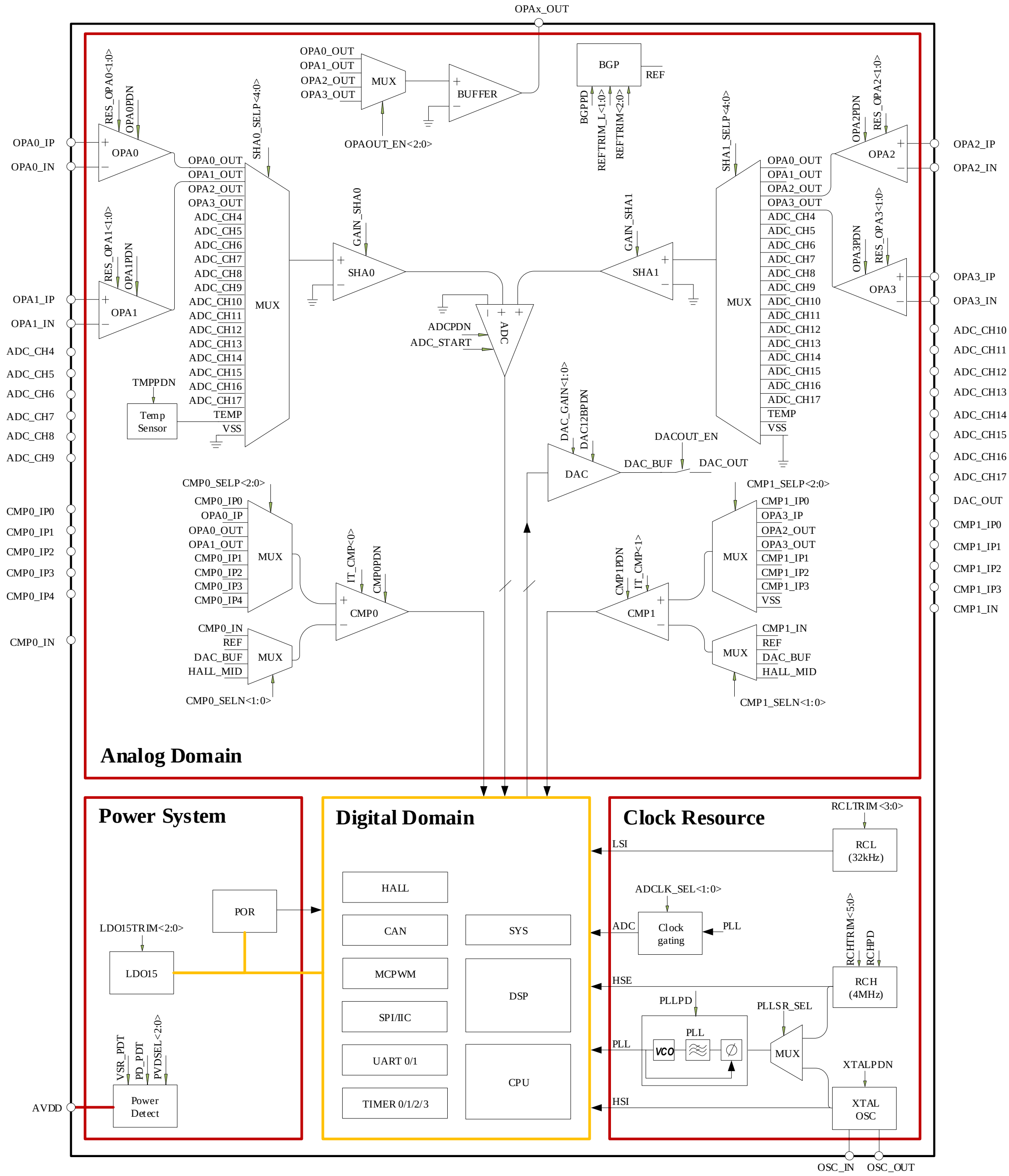


Fig. 4-1 Functional Block Diagram of Analog Circuit

4.2 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 POR module monitors the voltage of the AVDD and provides a reset signal for the digital circuit when the AVDD voltage is lower than 3.0V (for example, at the beginning of power on or when power off) to avoid abnormal operation of the digital circuit.

The PVD module monitors the 5V input power. If it is below a certain set threshold, it will remind the MCU by sending an alarm (interrupt) signal. Undervoltage only generates an undervoltage flag and does not generate a reset. The interrupt reminder threshold can be set to different voltages by the PVDSEL [1: 0] registers, and the PVD module can be turned off by setting PD_PDT ="1".

When the voltage is low, a low voltage interrupt will be triggered, and the interrupt number is 17. See Chapter 3 for details.

For the description of PVDSEL[1:0]/ PD_PDT, see the analog register SYS_AFE_REG6.

The low power supply voltage flag is shown in the analog front-end information register SYS_AFE_CMP.

4.3 Clock System (CLOCK)

The clock system consists of a 32KHz RC oscillator, a 4MHz RC oscillator, an external 4 to 8MHz crystal oscillator, and a PLL.

The 32kHz RC clock LSI is mainly used for watchdog module and reset/wakeup source filters in the system. The 4MHz RC clock can be used as the main clock of the MCU, and can provide a reference clock to PLL. PLL clock is up to 96MHz; the external 4 to 8MHz crystal oscillator is used as a backup clock.

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 ± 2.0%.

The 4MHz RC clock is turned on by setting RCHPD = ’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 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 96MHz, and the typical working frequency of ADC module is 48MHz. It can be set to different frequency by the register ADCLKSEL <1:0>.

PLL is turned on by setting PLLPDN = ’1’ (OFF by default, turn on when set to '1'). Before turning on the PLL module, the BGP (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.

The crystal oscillator circuit has a built-in amplifier but no oscillator capacitor. Connect a crystal between IO OSC_IN and OSC_OUT, and a 15pF capacitor to ground at each pin of OSC_IN and OSC_OUT. Set XTALPDN = ’1’ to start the oscillation.

For the description of ADCLKSEL<1:0>, see the analog register SYS_AFE_REG7.

For the description of BGPPD/RCHPD/XTALPDN/PLLPDN, see the analog register SYS_AFE_REG5.

4.4 Bandgap Voltage Reference (BGP)

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 2us 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_REG5

4.5 Analog-to-digital Converter (ADC)

Please refer to Chapter 10 ADC

4.6 Operational Amplifier (OPA)

The 4-channel of rail-to-rail OPAs is integrated, with a built-in feedback resistor, and an external resistor R_0 connected to the signal source on the pin. The resistance of feedback resistors R2: R1 can be adjusted by register RES_OPAx [1:0] to achieve different gains.

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

The schematic diagram of the amplifier is as follows:

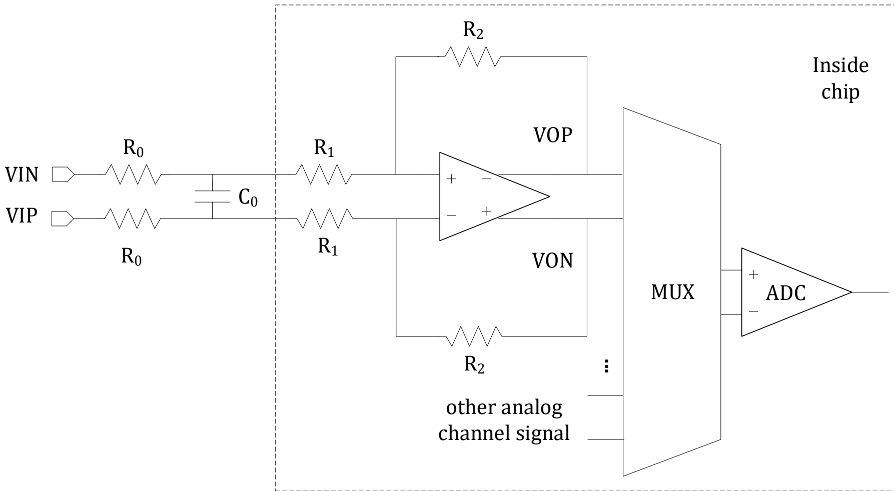


Fig. 4-2 Amplifier Block Diagram

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 $R_2/(R_1+R_0)$.

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

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

The OPA can select one of the output signals of the 4-channels amplifiers by setting OPAOUT_EN <2:0>, and send it to the P2.7 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<2:0>, see the analog register SYS_AFE_REG2.

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

For the description of OPAxPDN, see the analog register SYS_AFE_REG5.

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.

4.7 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 0.15uS/0.6uS through the register IT_CMP, 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. 4-1. 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.

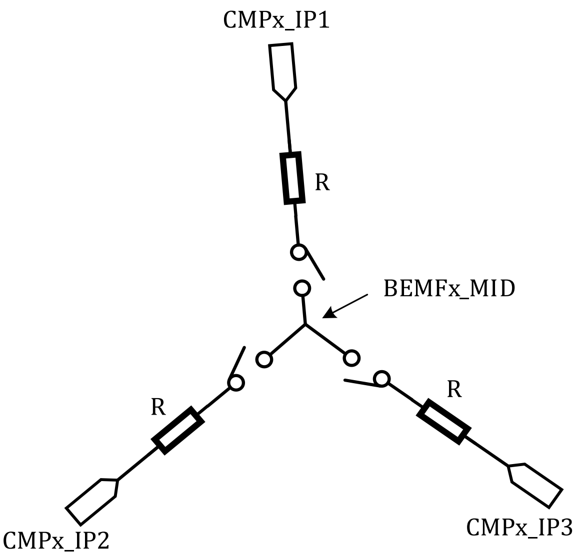


Fig. 4-2 BEMFx_MID Signal

The output of the comparator can be read through the register SYS_AFE_CMP.

For the description of IT_CMP<1:0>, 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_REG3

For the description of the output CMPx_RESULT of the comparator, see the Comparator output register SYS_AFE_CMP.

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_REG5.

4.8 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.

When measuring, select the internal reference and set the SYS_AFE_REG1.GAIN_REF = 0. Selecting an external reference causes a large deviation in the results.

The operating temperature of chips will be corrected before leaving the factory, and the corrected value is saved in the flash info area. The gain used for factory alignment is ADC_GAIN = 0. It is recommended that this gain also be selected for the application to make the alignment values more accurate.

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 18 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_REG5.

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

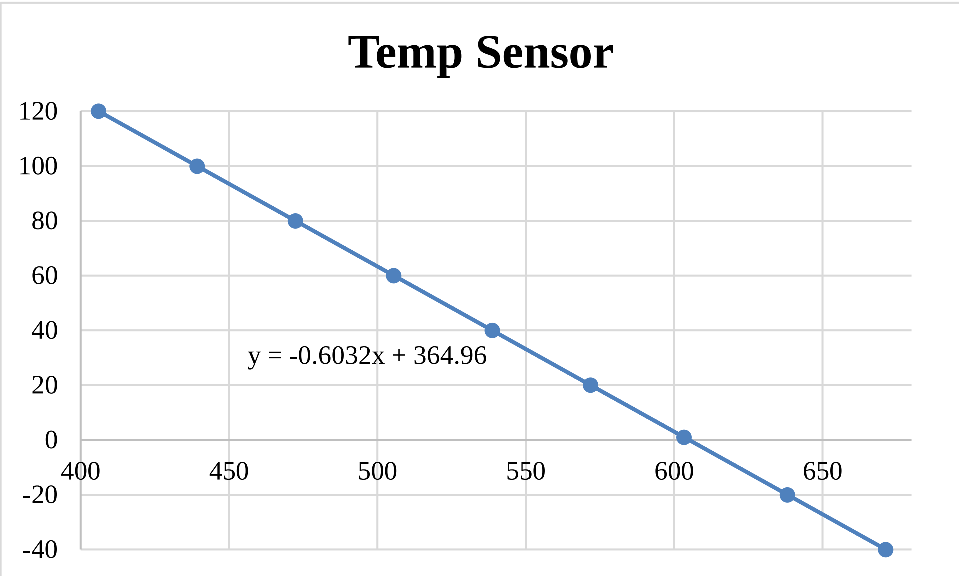


Fig. 4-4 Temperature Sensor Curve

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

$y=-0.6032x+364.96$

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, and the address is 0x0000039C. 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, and the address is 0x00000398. 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.

4.9 Digital-to-analog Converter (DAC)

The chip has a 1-channel 12bit DAC, the maximum range of the output signal can be set to 1.2V/3V/4.85V through the register DAC_GAIN <1:0>.

The 12bit 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 DAC12BPDN = 1. Turn on the BGP module before turning on the DAC module.

The input digital signal register of DAC is SYS_AFE_DAC, low 12-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 0xFFF is DAC_{fs}. As mentioned above, the value of DAC_{fs} can be set by the DAC12B_FS register. The analog signal amplitude corresponding to each gear signal (LSB) is $\frac{DAC_{fs}}{4096}$. If the digital value of SYS_AFE_DAC is Din, the analog signal of the DAC output corresponding to the digital signal is $\frac{DAC_{fs}}{4096} * Din$.

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

Address 0x00000330 is the parameter a of the 3V range, and address 0x00000340 is the parameter b of the 3V range.

The address 0x00000334 is the parameter a of the 1.2V range, and the address 0x00000344 is the parameter b of the 1.2V range.

Address 0x00000338 is the parameter a of the 4.85V range, and address 0x00000348 is the parameter b of the 4.85V range.

Except for external module usage via IO port, the analog signal output by the DAC can also be used as a reference signal for the comparator by connecting the configuration register to the negative side of the two-channel CMP inside the chip.. See the section 4.7 Comparator (CMP) for details.

For the description of DACOUT_EN, see the analog register SYS_AFE_REG3

For the description of DAC_GAIN<1:0>, see the analog register SYS_AFE_REG1

For the description of DAC12BPDN, see the analog register SYS_AFE_REG5

For the description of SYS_AFE_DAC, see the register SYS_AFE_DAC

5 System Control and Clock Reset

5.1 System Clock

5.1.1 Clock Source

As shown in the following table, the system includes 5 clock sources, of which the internal low-speed RC oscillator (LSI, Low Speed Internal Clock) and internal high-speed RC oscillator (HSI, High Speed Internal Clock) will not stop vibration. The external crystal oscillator (HSE, High Speed External Clock) may fail under extreme operating conditions, and only some applications will use HSE.

Table 5-1 System Clock Source

Clock Source	Frequency	Source	Error	Description
LSI	32kHz	Internal RC Oscillator	16KHz~48KHz	Internal system clock, used for watchdog module, and filtering and widening of reset signal.
HSI	4MHz	Internal RC Oscillator	Full temperature range error <2%	Can be used as PLL source clock
PLL	96MHz	PLL clock	0	Taking the HSI/HSE 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.
HSE	4~8MHz	External crystal oscillator	0	When the external crystal has strict requirements on clock accuracy (e.g, ppm level accuracy), a 4MHz HSE can be used as the PLL reference clock to generate the 96MHz main system clock. PLL reference clock cannot be other frequency than 4MHz.
SWD	1MHz*	Debugger	-	SWD's JTAG clock

* The typical value and actual size of the SWD clock rate are related to the hardware environment.

Besides, the crystal oscillator can support an external crystal frequency of 4-8MHz. If HSE is used as the reference clock of PLL, a 4MHz crystal is still needed for providing a reference clock of PLL with the same frequency as HSI. If using HSE as the main system clock, there is no such limitation.

As shown in

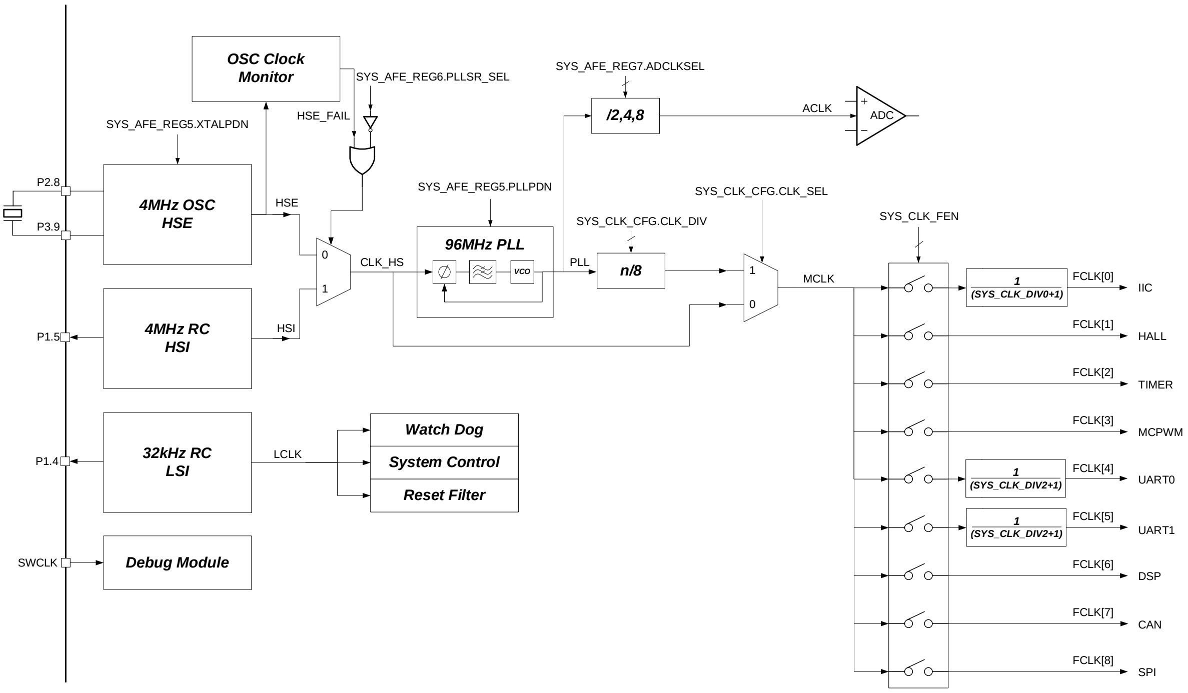


Fig. 5-1, the system can use the external crystal oscillator clock HSE or the internal high-speed RC oscillator HSI as the reference clock of PLL, and select by SYS_AFE_REG6.PLLSR_SEL. The chip can also detect whether the HSE clock is stopped. If HSE fails, it will automatically switch to HSI as the reference clock of PLL. The PLL multiplies the 4MHz reference clock CLK_HS by 24 times to 96MHz.

After the PLL is divided by n/8, the high-speed clock of 96MHz × n/8 can be obtained. SYS_CLK_CFG.CLK_SEL chooses one between the divided high-speed clock and the 4MHz CLK_HS 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, 36, 48, 60, 72, 84, 96 MHz and other frequency values. SYS_CLK_CFG.CLK_SEL means PLL or CLK_HS is selected as the main system clock. 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 0, SYS_CLK_CFG.CLK_DIV has no effect.

Table 5-2 Frequency Division Configuration When PLL is Used as MCLK Clock

SYS_CLK_CFG	Frequency Division Factor	Frequency/ MHz	If Uniform
0x0101	1/8	12	Yes
0x0111	2/8	24	Yes
0x0115	3/8	36	No
0x0155	4/8	48	Yes
0x0157	5/8	60	No
0x0177	6/8	72	No
0x017F	7/8	84	No
0x01FF	8/8	96	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 2/4/8 controlled by SYS_AFE_REG7.ADCLKSEL, 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.

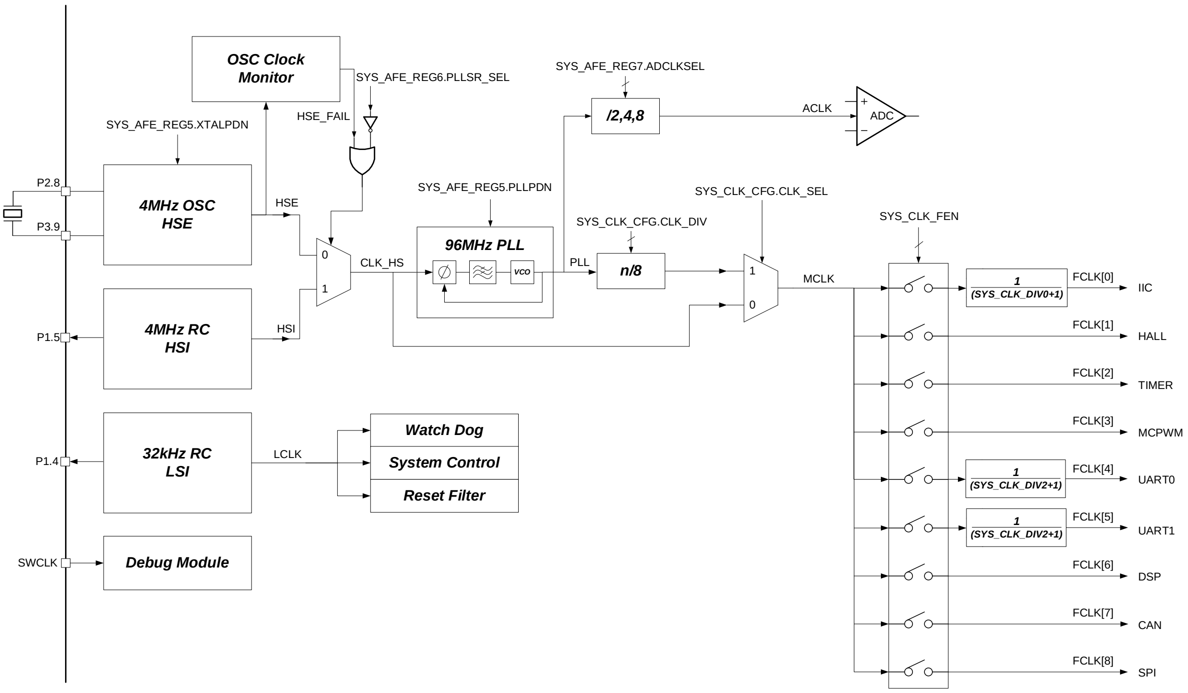


Fig. 5-1 Clock Architecture

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 or HSE, which is the reference clock, cannot be turned off by software; When CLK_HS is used as the main clock, HSI or HSE, which is the CLK_HS, cannot be turned off; If the system detects that the HSE has stopped vibration after being used, it will switch to the HSI clock automatically, instead of the HSE clock. 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.

5.1.2 Power Management and Sleep Wakeup

5.1.2.1 Sleep

MCLK can be gated by configuration, so that most digital circuits including the CPU and all peripherals are in a sleep state. During gating, the PMU state machine turns off analog modules such as PLL, HSI/HSE, and BGP in order to reduce power consumption.

Please note that only high-speed clocks such as PLL, HSI, and HSE are turned off when the system enters a sleep mode, and the LSI clock is still working. If the LSI-driven watchdog is enabled, the watchdog reset can be considered as a global reset and return the system to the initial state and start working again.

Write 0xDEAD to the SYS_CLK_SLP register to make the chip enter the sleep state, and then execute the __WFI () macro instruction to stop the CPU from fetching instructions.

Please configure the wake-up conditions when programming the application.

*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.

5.1.2.2 Wakeup

After sleep, external IO events and internal wake-up Timer can be used as wake-up sources.

The internal wake-up Timer is an independent Timer independent of the UTimer module. It uses an LSI clock, which is different from the general Timer and serves for the main system clock. The wake-up Timer can be set to a total of 8 wake-up time intervals of 0.25s, 0.5s, 1s, 2s, 4s, 8s, 16s, and 32s by the SYS_RST_CFG.WK_INTV.

Only the four IOs, P0[1:0] and P1[1:0], can be used as external wake-up IOs, and has independent enable and polarity. Please refer to the chapter 8.2.13.5 WAKE_POL and 8.2.13.6 WAKE_EN for specific register configuration. Please note that the external IO wake-up is a level trigger. If the external IO is at the wake-up level, it will wake up the chip immediately after it sleeps.

When programming the application, please try to avoid entering the sleep mode once the chip is powered on. If the internal wake-up Timer is the wake-up sources, the chip will sleep again after waking up, resulting in failure of ordinary downloaders to connect and debug, and an offline downloader provided by the chip vendor will be needed for application erasure and rewriting.

5.1.2.3 Peripheral Clock Gating (PCG)

The peripheral clock is divided by the system high-speed clock MCLK; it can be close turned off by setting the SYS_CLK_FEN register gating when the peripheral is not in use. There are eight peripheral clocks available for different peripheral modules that could be turned off when not being in use, and each peripheral clock has a clock gating. Please refer to

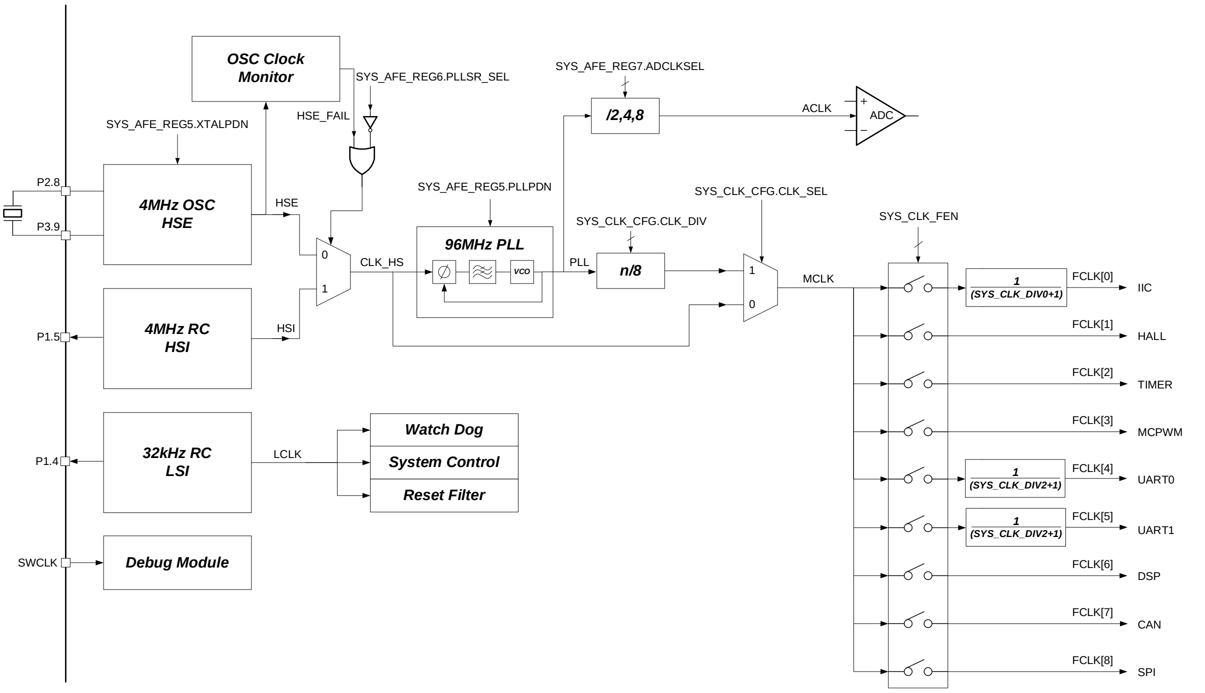


Fig. 5-1 Clock Architecture for details. The gated clock is turned off by default after power-on, and should be turned on by software before using the corresponding peripheral module.

I2C: FCLK[0]

Hall module: FCLK [1]

Timer module: FCLK [2]

MCPWM module: FCLK [3]

UART0/UART1: FCLK [4]/ FCLK [5]

DSP: FCLK [6]

CAN: FCLK [7]

SPI: FCLK [8]

5.1.2.4 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 UART0/1. Besides, the UART baud rate has an additional clock divider inside the UART module, as shown in

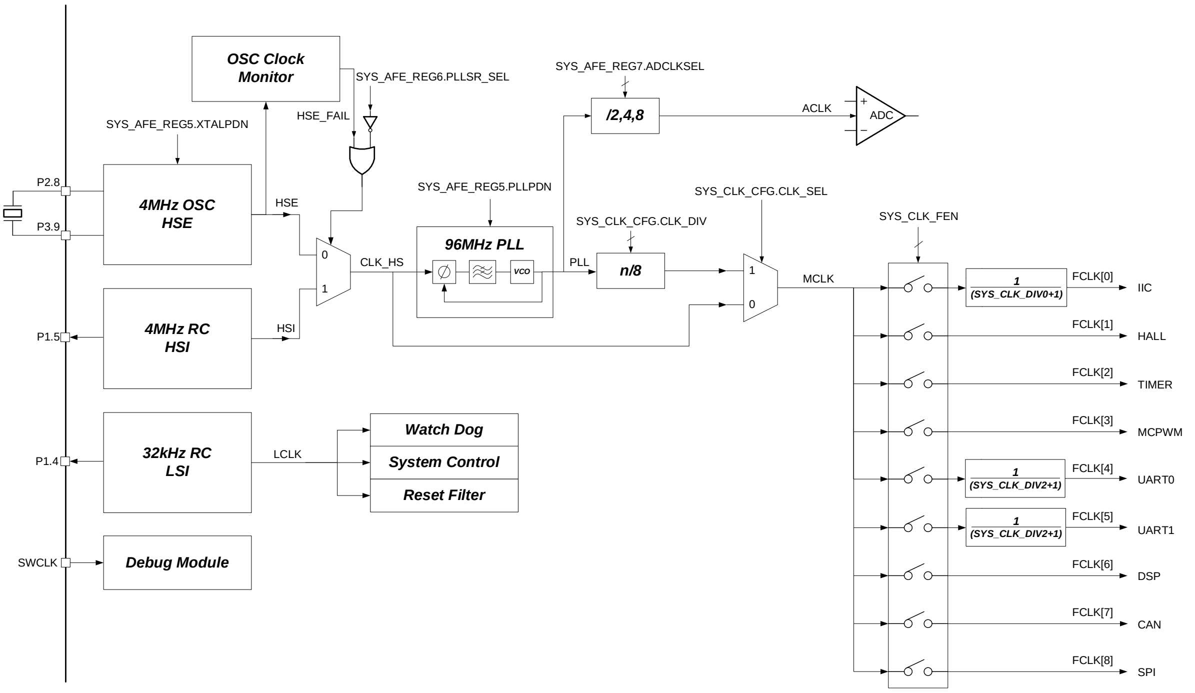


Fig. 5-1 Clock Architecture

5.2 Reset

5.2.1 Reset Source

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

5.2.1.1 Hardware Reset

As shown in Table 5-3 Hardware Reset Sources, the system has four Hardware Reset Sources. The resets generated are all chip global resets. After the reset, the chip program counter returns to address 0, and all registers are restored to their default values. The four hardware resets are all active low.

Table 5-3 Hardware Reset Source

Name	Source	Description
LPORn	Internal 1.5V Power Management	Monitor 1.5V digital power supply, reset when it's below 1.25V
HPORn	Internal 3.3V Power Management	Monitor 3.3V digital power supply, reset when it's below 2.5V
RSTn	External Buttons	External RC reset circuit
WDTn	Hardware watchdog	If not feed the watchdog, then it will reset the CPU at a regular time, and the reset interval is configurable.

5.2.1.1.1 Hardware Reset Architecture

As shown below, LPORn/HPORn is an internal analog circuit, and RSTn is an external key.

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

A reset signal short than 32us to P0.2 will be filtered. A reliable reset signal should be as long as 200us.

The four reset signals are global reset, so the reset levels and scopes are the same.

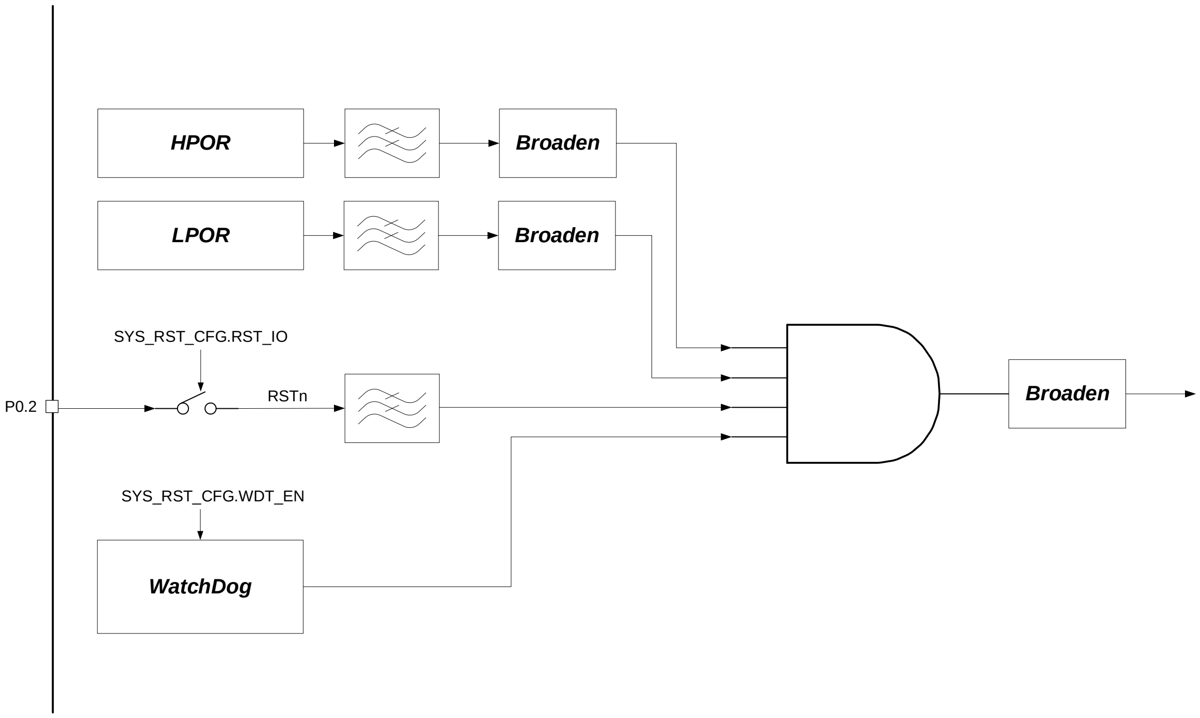


Fig. 5-2 Hardware Reset Architecture

5.2.1.1.2 Hardware Reset Records

The SYS_RST_SRC register is used to save hardware reset events. When a hardware reset occurs, the corresponding bit of SYS_RST_SRC is set. The SYS_RST_SRC register itself be reset by the reset signal; it can only clear the record by writing 0xCA40 to the SYS_CLR_RST register. With the reset record, we can easily understand whether and what kind of reset has occurred.

5.2.1.2 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" 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 5.3.22 for details.

5.3 Register

5.3.1 Address Allocation

The base address of the system module register is 0x4000_0000, and the register list is as follows:

Table 5-4 System Control Register

Name	Offset	Description
	0x00~0x08	Reserved
	0x10~0x14	Reserved
SYS_AFE_CMP	0x18	Comparator output register
	0x1C	Reserved
SYS_AFE_REG0	0x20	Analog register 0
SYS_AFE_REG1	0x24	Analog register 1
SYS_AFE_REG2	0x28	Analog register 2
SYS_AFE_REG3	0x2C	Analog register 3
SYS_AFE_REG4	0x30	Analog register 4
SYS_AFE_REG5	0x34	Analog register 5
SYS_AFE_REG6	0x38	Analog register 6
SYS_AFE_REG7	0x3C	Analog register 7
	0x54~0x78	Reserved
SYS_AFE_DAC	0x7C	DAC digital register
SYS_CLK_CFG	0x80	Clock control register
SYS_RST_CFG	0x84	Reset control register
SYS_RST_SRC	0x88	Register for recording reset source
SYS_CLR_RST	0x8C	Register for clearing reset source
SYS_CLK_DIV0	0x90	Peripheral clock divider registerc0
	0x94	Reserved
SYS_CLK_DIV2	0x98	Peripheral clock divider register 2
SYS_CLK_FEN	0x9C	Peripheral clock gating register
SYS_CLK_SLP	0xA0	Sleep register
	0xA4	Reserved
SYS_TRIM	0xA8	Correction mode register
SYS_SFT_RST	0xAC	Soft reset register
SYS_WR_PROTECT	0xB0	Write protection register
SYS_DAC_AMC	0xB4	DAC gain correction register
SYS_DAC_DC	0xB8	DAC DC offset register

5.3.2 SYS_AFE_CMP Comparator output register

Address: 0x4000_0018

Reset value: 0x0

Table 5-5 Comparator output register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP1	CMP0														
RO	RO														
0	0														

Location	Bit Name	Description
[31:16]		Unused
[15]	CMP1	CMP1 Output register
[14]	CMP0	CMP0 Output register
[13:0]		Unused

The CMP0/CMP1 in SYS_AFE_CMP is the original output of the comparator without filtering. The CMP0/CMP1 can also be output by setting the second function (AF1) of GPIO. For specific output pins, please check the DATASHEET.

5.3.3 Introduction to Analog Register

Address space of 0x40000020 ~ 0x4000003C is a register open to users, 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 following is a detailed description of each analog register.

5.3.4 AFE Register 0 (SYS_AFE_REG0)

Address: 0x4000_0020

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
IT_RBUF	IT_ADCMP	IT_AMP	IT_OPA	REF_OPA3	REF_OPA2	REF_OPA1	REF_OPA0
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]	IT_RBUF	Bias current adjustment for ADC reference buffer, default

		configuration 00: ×1 01: ×1.2 10: ×1.5 11: Prohibited
[13:12]	IT_ADCMP	Bias current adjustment for ADC comparator, default configuration 00: ×1; 01: Prohibited 10: ×0.66 11: ×1
[11:10]	IT_AMP	Bias current adjustment for ADC amplifier, default configuration 00: ×1 01: ×1.5 10: ×0.75 11: ×
[9:8]	IT_OPA	Bias current adjustment for OPA, default configuration 00: ×1 01: ×1.2 10: ×1.5 11: Prohibited
[7:6]	REF_OPA3	OPA3 feedback resistor 00: 200k:10.4k 01: 190k:20.4k 10: 180k:30.4k 11: 170k:40.4k
[5:4]	RES_OPA2	OPA2 feedback resistor 00: 200k:10.4k 01: 190k:20.4k 10: 180k:30.4k 11: 170k:40.4k
[3:2]	RES_OPA1	OPA1 feedback resistor 00: 200k:10.4k 01: 190k:20.4k 10: 180k:30.4k 11: 170k:40.4k
[1:0]	RES_OPA0	OPA0 feedback resistor 00: 200k:10.4k 01: 190k:20.4k 10: 180k:30.4k 11: 170k:40.4k

5.3.5 AFE Register 1 (SYS_AFE_REG1)

Address: 0x4000_0024

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
Reserved	DAC_GAIN	REFVDD5	REF2VDD	GAIN_REF	IT_CMP	CMP_FT
RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0

Location	Bit name	Description
[31:16]		Unused
[15]	Reserved	Reserved bits, all ‘0’
[14]	Reserved	Reserved bits, all ‘0’
[13]	Reserved	Reserved bits, all ‘0’
[12]	Reserved	Reserved bits, all ‘0’
[11:8]	Reserved	Reserved bits, all ‘0’
[7:6]	DAC_GAIN	DAC output configuration 00: DAC output configuration, full scale is 3V 01: DAC output configuration, full scale is 1.2V 10: DAC output configuration, full scale is 4.85V (Note: The power supply should to be 5V to avoid abnormal output of DAC.) 11: Misconfiguration
[5]	REFVDD5	When the power supply is ADC REF and the power supply is 5V, increase the signal range of the OPA output. 0: The maximum output range of the OPA is 3.3V. 1: Increase the maximum output range of OPA to 4.8V, and it can only be set to ‘1’ when REF2VDD is set as ‘1’.
[4]	REF2VDD	Use external input power as ADC REF 1: When using external input power as ADC REF, GAIN_REF configuration is invalid; 0: Take the default value as the internal ADC reference
[3]	GAIN_REF	ADC reference voltage adjustment, default configuration 0:×2; 1:×1;
[2:1]	IT_CMP	IT_CMP <1>: Comparison speed selection of comparator 1, 0: 150ns; 1: 600ns; IT_CMP <0>: Comparison speed selection of comparator 0, 0: 150ns; 1:600ns;
[0]	CMP_FT	Enable comparator for quick comparison 1: When IT_CMP <1: 0> is the default '00', the comparison speed of the comparator is less than 30ns

		0: Disabled, the comparison speed in the IT_CMP setting remains unchanged.

5.3.6 AFE Register 2 (SYS_AFE_REG2)

Address: 0x4000_0028

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

Reserved	Reserved	Reserved	Reserved	OPAOUT_EN
RW	RW	RW	RW	RW
0	0	0	0	0

Location	Bit name	Description
[31:16]		Unused
[15:14]	Reserved	Reserved bits, all '0'
[13]	Reserved	Reserved bits, all '0'
[12]	Reserved	Reserved bits, all '0'
[11:10]	Reserved	Reserved bits, all '0'
[9:8]	Reserved	Reserved bits, all '0'
[7]	Reserved	Reserved bits, all '0'
[6]	Reserved	Reserved bits, all '0'
[5:4]	Reserved	Reserved bits, all '0'
[3]	Reserved	Reserved bits, all '0'
[2:0]	OPAOUT_EN	Enable OPAx output signal via IO port p2_7 000: not output; 001: Output OPA0 signal via IO port; 010: Output OPA1 signal via IO port; 011: Output OPA2 signal via IO port; 100: output OPA3 signal via IO port; 101~111: Prohibited

5.3.7 AFE Register 3 (SYS_AFE_REG3)

Address: 0x4000_002C

Reset value: 0x0

Table 5-9 AFE Register 3 (SYS_AFE_REG3)

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

Res.	CMP1_SELP	DACOUT_EN	CMP0_SELP	CMP_HYS	REF_AD_EN	CMP1_SELN	CMP0_SELN	Res.	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	Reserved bits, all '0'
[14:12]	CMP1_SELP	Positive Selection of Comparator 1 Signal 000: Connect to CMP1_IP0 001: Connect to OPA3_IP 010: Connect to OPA2_OUT 011: Connect to OPA3_OUT 100: Connect to CMP1_IP1 101: Connect to CMP1_IP2 110: Connect to CMP1_IP3 111: Connect to AVSS Note: The above are pin names except AVSS/OPA2_OUT/OPA3_OUT. For the definition of the pins, please refer to the specific section in the DATASHEET.
[11]	DACOUT_EN	DAC Output and IO Enable 0: Disabled 1: Enable output via IO P0[0]
[10:8]	CMP0_SELP	Positive Selection of Comparator 0 Signal 000: Connect to CMP0_IP0 001: Connect to OPA0_IP 010: Connect to OPA0_OUT 011: Connect to OPA1_OUT 100: Connect to CMP0_IP1 101: Connect to CMP0_IP2 110: Connect to CMP0_IP3 111: Connect to CMP0_IP4 Note: The above are pin names except OPA0_OUT/OPA1_OUT. For the definition of the pins, please refer to the specific section in the DATASHEET.
[7]	CMP_HYS	Comparator Hysteresis Selection, default configuration 0: 20mv; 1: 0mv
[6]	REF_AD_EN	ADC REF Output Enable. For testing REF with the default configuration in normal conditions. 0: Not output 1: Enable REF BGP output via IO P2_3
[5:4]	CMP1_SELN	Negative Selection of Comparator 1 Signal 00: Connect to CMP1_IN 01: Connect to REF

		10: Connect to DAC output 11: Connect to HALL1_MID Note: The above CMP1_IN is the pin name, For the pin definition, please refer to the specific section in the DATASHEET; REF is the 1.2V BANDGAP reference inside the chip; The DAC output is the analog signal output by the DAC module inside the chip; HALL1_MID is the average value obtained by connecting the CMP1_IP1, CMP1_IP2, and CMP1_IP3 signal through a star connection.
[3:2]	CMP0_SELN	Negative Selection of Comparator 0 Signal 00: Connect to CMP0_IN 01: Connect to REF 10: Connect to DAC output 11: Connect to HALL0_MID Note: Note: The above CMP0_IN is the pin name, For the pin definition, please refer to the specific section in the DATASHEET; REF is the 1.2V BANDGAP reference inside the chip; The DAC output is the analog signal output by the DAC module inside the chip; HALL1_MID is the average value obtained by connecting the CMP1_IP1, CMP1_IP2, and CMP1_IP3 signal through a star connection.
[1]	Reserved	Reserved bits, all '0'
[0]	LDOOUT_EN	LDO Output and IO Enable 0: Not output 1:Enable output via IO P2.7

5.3.8 AFE Register 4 (SYS_AFE_REG4)

Address: 0x4000_0030

Reset value: 0x0

Table 5-10 AFE Register 4 (SYS_AFE_REG4)

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

Location	Bit name	Description
[31:16]		Unused
[15]	Reserved	Reserved bits, all '0'
[14]		Unused

[13]	Reserved	Reserved bits, all ‘0’
[12:6]		Unused
[5]	Reserved	Reserved bits, all ‘0’
[4:0]		Unused

5.3.9 AFE Register 5 (SYS_AFE_REG5)

Address: 0x4000_0034

Reset value: 0x0

Table 5-11 AFE Register 5 (SYS_AFE_REG5)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLLPDN	XTALPDN	TMPPDN	DAC12BPDN	Res.	RCHPD	Res.	BGPPD	CMP1PDN	CMP0PDN	OPA3PDN	OPA2PDN	OPA1PDN	OPA0PDN	Res.	ADCPDN
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]	PLLPDN	PLL Off Control 0: Turn off PLL (default) 1: Turn on PLL
[14]	XTALPDN	Crystal Oscillator Enable 0: Off (default) 1: On
[13]	TMPPDN	Temperature Sensor Enable 0: Off 1: On
[12]	DAC12BPDN	12-bit DAC Enable 0: Off 1: On
[11]	Reserved	Reserved bits, all ‘0’
[10]	RCHPD	RCH Clock Off 0: On 1: Off
[9]	Reserved	Reserved bits, all ‘0’
[8]	BGPPD	BGP Enable 0: On 1: Off
[7]	CMP1PDN	CMP1 Enable

		0: Off 1: On
[6]	CMP0PDN	CMP0 Enable 0: Off 1: On
[5]	OPA3PDN	OPA3 Enable 0: Off 1: On
[4]	OPA2PDN	OPA2 Enable 0: Off 1: On
[3]	OPA1PDN	OPA1 Enable 0: Off 1: On
[2]	OPA0PDN	OPA0 Enable 0: Off 1: On
[1]	Reserved	Reserved bits, all '0'
[0]	ADCPDN	ADC Enable 0: Off 1: On

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.

5.3.10 AFE Register 6 (SYS_AFE_REG6)

Address: 0x4000_0038

Reset value: 0x0

Table 5-12 AFE Register 6 (SYS_AFE_REG6)

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

RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0

Location	Bit name	Description
[31:16]		Unused
[15]	PLLSR_SEL	PLL Clock Source Selection 0: Use RCH as the input clock source; 1: Use XTAL OSC as input clock source
[14]	Reserved	Reserved bits, all '0'
[13]	Reserved	Reserved bits, all '0'
[12]	Reserved	Reserved bits, all '0'
[11]	Reserved	Reserved bits, all '0'
[10]	Reserved	Reserved bits, all '0'
[9:8]	PVDSEL	Threshold Selection for Power Failure Detection 00: 4.5V 01: 4.2V 10: 3.9V 11: 3.6V
[7]	Reserved	Reserved bits, all '0'
[6:5]	Reserved	Reserved bits, all '0'
[4:3]	Reserved	Reserved bits, all '0'
[2]	Reserved	Reserved bits, all '0'
[1]	Reserved	Reserved bits, all '0'
[0]	PD_PDT	Power Down Supply Voltage Detection Circuit 0: On 1: Off

5.3.11 AFE Register 7 (SYS_AFE_REG7)

Address: 0x4000_003C

Reset value: 0x0

Table 5-13 AFE Register 7 (SYS_AFE_REG7)

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

Location	Bit name	Description
[31:16]		Unused
[15]	Reserved	Reserved bits, all '0'

[14]	Reserved	Reserved bits, all '0'
[13:8]	Reserved	Reserved bits, all '0'
[7:6]	Reserved	Reserved bits, all '0'
[5:4]	ADCLKSEL	ADC Clock Frequency Selection 00: 48MHz 01: Prohibited 10: 12MHz 11: 24MHz
[3:0]	Reserved	Reserved bits, all '0'

5.3.12 DAC Digital Register (SYS_AFE_DAC)

Address: 0x4000_007C

Reset value: 0x0

Table 5-14 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
[31:12]		Unused
[11:0]	DAC_IN	DAC Digital Input to be Converted

5.3.13 Clock Control Register (SYS_CLK_CFG)

Address: 0x4000_0080

Reset value: 0x0

Table 5-15 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
[31:9]		Unused
[8]	CLK_SEL	CLK_HS/PLL selection, 0: CLK_HS, 1: PLL. CLK_HS is selected by default.

		CLK_HS can be HSI or HSE depending on whether an external crystal clock is used. PLL is turned off by default after power on, and should be enabled by software.
[7:0]	CLK_DIV	PLL output frequency division control. Choose which of the eight clock cycles to output the clock. For example, 8'b00000001 means 1/8 frequency division, 8'b00010001 means 1/4 frequency division, and 8'b00100101 means 1/3 frequency division, with uneven waveform.

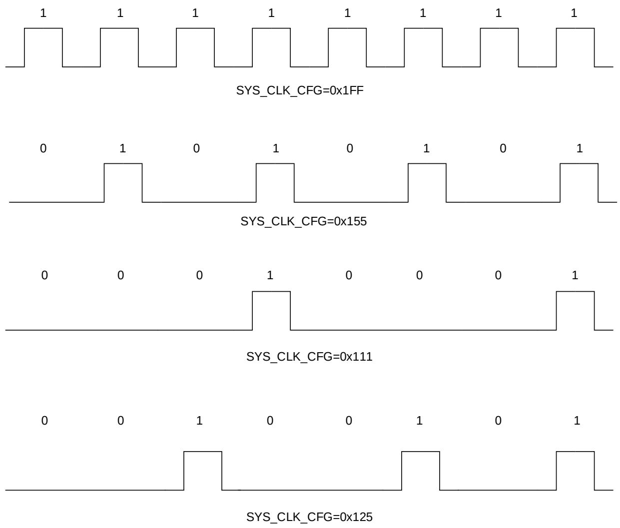


Fig. 5-3 The Waveform of Main Clock Frequency Division in different configurations of SYS_CLK_CFG

When using 4MHz HSI clock as the system master clock, the division factor of SYS_CLK_CFG [7: 0] is invalid. The final output clock frequency is 4MHz.

5.3.14 Reset Control Register (SYS_RST_CFG)

Address: 0x4000_0084

Reset value: 0x0

Table 5-16 Reset Control Register (SYS_RST_CFG)

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

Location	Bit name	Description
[31:6]		Unused
[5]	RST_IO	RSTn/P0 [2] multiplex selection, 0: RSTn, 1:P0[2]
[4:2]	WK_INTV	Sleep Wake-up Interval Setting 100: 4S 000: 0.25S 101: 8S

		001: 0.5S 010: 1S 011: 2S	110: 16S 111: 32S
[1]		Unused	
[0]	WDT_EN	Watchdog Enable. Highly effective.	

5.3.15 Reset Source Record Register (SYS_RST_SRC)

Address: 0x4000_0088

Reset value: 0x0

Table 5-17 Reset Source Record Register (SYS_RST_SRC)

15 14 13 12 11 10 9 8 7 6 5 4	3	2	1	0
	WDT_RST_RCD	KEY_RST_RCD	HPOR_RST_RCD	LPOR_RST_RCD
	RO	RO	RO	RO
	0	0	0	0

Location	Bit name	Description
[31:4]		Unused
[3]	WDT_RST_RCD	Watchdog reset occurred flag. Highly effective.
[2]	KEY_RST_RCD	Key reset occurred flag. Highly effective.
[1]	HPOR_RST_RCD	HPOR reset occurred flag. Highly effective.
[0]	LPOR_RST_RCD	LPOR reset occurred flag. Highly effective.

5.3.16 Reset Source Record Clear Register (SYS_CLR_RST)

Address: 0x4000_008C

Reset value: 0x0

Table 5-18 Reset Source Record Clear Register (SYS_CLR_RST)

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 0xCA40 to clear the reset flag record Please note that since the reset recording works in the low-speed clock domain, it may take a certain time to finish clearing, and the state of the record

		should not be read immediately after clearing.

5.3.17 Peripheral Clock Divider Register 0 (SYS_CLK_DIV0)

Address: 0x4000_0090

Reset value: 0x0

Table 5-19 Peripheral Clock Divider Register 0 (SYS_CLK_DIV0)

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

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.

5.3.18 Peripheral Clock Divider Register 2 (SYS_CLK_DIV2)

Address: 0x4000_0098

Reset value: 0x0

Table 5-20 Peripheral Clock Divider Register 2 (SYS_CLK_DIV2)

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

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

5.3.19 Peripheral Clock-Gating Register (SYS_CLK_FEN)

Address: 0x4000_009C

Reset value: 0x0

Table 5-21 Peripheral Clock-Gating Register (SYS_CLK_FEN)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							SPI_CLK_EN	CAN_CLK_EN	DSP_CLK_EN	UART1_CLK_EN	UART0_CLK_EN	MCPWM_CLK_EN	UTIMER_CLK_EN	HALL_CLK_EN	I2C_CLK_EN
							RW	RW	RW	RW	RW	RW	RW	RW	RW
							0	0	0	0	0	0	0	0	0

Location	Bit name	Description
[31:9]		Unused
[8]	SPI_CKL_EN	SPI clock gating. 1: enable clock; 0: disable clock
[7]	CAN_CLK_EN	CAN clock gating. 1: enable clock; 0: disable clock
[6]	DSP_CLK_EN	DSP clock gating. 1: enable clock; 0: disable clock
[5]	UART1_CLK_EN	UART1 clock gating. 1: enable clock; 0: disable clock
[4]	UART0_CLK_EN	UART0 clock gating. 1: enable clock; 0: disable clock
[3]	MCPWM_CLK_EN	MCPWM clock gating. 1: enable clock; 0: disable clock
[2]	UTIMER_CLK_EN	UTIMER clock gating. 1: enable clock; 0: disable clock
[1]	HALL_CLK_EN	HALL clock gating. 1: enable clock; 0: disable clock
[0]	I2C_CLK_EN	I2C clock gating. 1: enable clock; 0: disable clock

5.3.20 Sleep Register (SYS_CLK_SLP)

Address: 0x4000_00A0

Reset value: 0x0

Table 5-22 Sleep Register (SYS_CLK_SLP)

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 the password 0xDEAD, the system turns off the high-speed clock and

		enters the sleep state.

5.3.21 Correction Mode Register (SYS_TRIM)

Address: 0x4000_00A8

Reset value: 0x0

Table 5-23 Correction Mode Register (SYS_TRIM)

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

Location	Bit name	Description
[31:1]		Unused
[0]	TRIM	Enter TRIM mode after chip reset After TRIM ends, exit TRIM mode by soft reset.

5.3.22 Soft Reset Register (SYS_SFT_RST)

Address: 0x4000_00AC

Reset value: 0x0

Table 5-24 Soft Reset Register (SYS_SFT_RST)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							SPI_SFT_RST	CAN_SFT_RST	DSP_SFT_RST	UART1_SFT_RST	UART0_SFT_RST	MCPWM_SFT_RST	UIMTER_SFT_RST	HALL_SFT_RST	I2C_SFT_RST
							WO	WO	WO	WO	WO	WO	WO	WO	WO
							0	0	0	0	0	0	0	0	0

Location	Bit name	Description
[31:9]		Unused
[8]	SPI_SFT_RST	SPI soft reset. Write 1 to reset, then write 0 to release.
[7]	CAN_SFT_RST	CAN soft reset. Write 1 to reset, then write 0 to release.
[6]	DSP_SFT_RST	DSP soft reset. Write 1 to reset, then write 0 to release.
[5]	UART1_SFT_RST	UART1 soft reset. Write 1 to reset, then write 0 to release.
[4]	UART0_SFT_RST	UART0 soft reset. Write 1 to reset, then write 0 to release.

[3]	MCPWM_SFT_RST	MCPWM soft reset. Write 1 to reset, then write 0 to release.
[2]	UTIMER_SFT_RST	UTIMER soft reset. Write 1 to reset, then write 0 to release.
[1]	HALL_SFT_RST	HALL soft reset. Write 1 to reset, then write 0 to release.
[0]	I2C_SFT_RST	I2C soft reset. Write 1 to reset, then write 0 to release.

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.

5.3.23 Write Protection Register (SYS_WR_PROTECT)

Address: 0x4000_00B0

Reset value: 0x0

Table 5-25 Write Protection Register (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_REG3, SYS_AFE_DAC, SYS_AFE_DAC_AMC, SYS_AFE_DAC_DC, other registers are write-protected, and should write a password in advance to release the write-protection. Write 0x7A83 to enable the register's write operation. Write 0xCAFE to enable the WDT_CLR register's write operation. Write other values to prohibit registers' write operations.

5.3.24 DAC Gain Correction Register (SYS_AFE_DAC_AMC)

Address: 0x4000_00B4

Reset value: 0x0

Table 5-26 DAC Gain Correction Register (SYS_AFE_DAC_AMC)

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

Location	Bit name	Description

[31:10]		Unused
[9:0]	DAC_AMC	DAC gain calibration value. It's a 10-bit unsigned fixed-point number, of which B [9] is an integer, and B [8: 0] is a decimal.

5.3.25 DAC DC Offset Register (SYS_AFE_DAC_DC)

Address: 0x4000_00B8

Reset value: 0x0

Table 5-27 DAC DC Offset Register (SYS_AFE_DAC_DC)

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

Location	Bit name	Description
[31:8]		Unused
[7:0]	DAC_DC	DAC DC bias value. It's a 8bit signed number, of which B [7] is the sign bit.

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

DAC_cali = DAC_raw*DAC_AMC – DAC_DC;

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

DAC_AMC = 10'b01_1110_1100 = 1-5/128

DAC_DC Bit DAC DC bias, which is a 8-bit signed integer.

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

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

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

6 FLASH

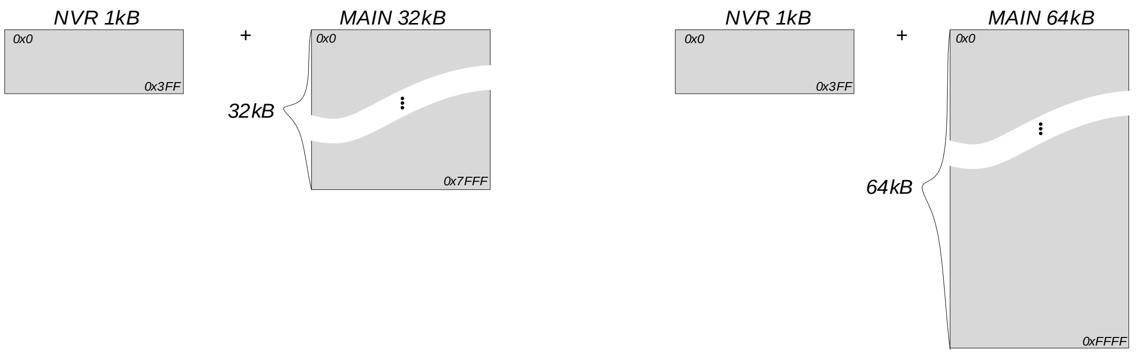
6.1 Introduction

The FLASH memory includes two parts: NVR and MAIN. NVR size is 1kB, MAIN is 32kB or 64kB (different models).

The main flash memory area (MAIN) includes application programs and user data area.

The information storage area (info area/NVR) includes three parts:

- Option bytes: includes user options for hardware and storage protection (does not occupy NVR space in the figure below)
- System memory: includes the boot loader code (does not occupy the NVR space in the figure below)
- User data area: 1KB reserved for users.



32kB flash空间划分 64kB flash空间划分

Fig. 6-1 Block Diagram of FLASH Memory Space Division

6.2 Features

The FLASH controller module mainly implements the related operations on the FLASH memory , including:

- FLASH reading, including reading operations of the NVR and MAIN.
- FLASH writing, including writing operations to the NVR and MAIN.
- FLASH erase, including CHIP erase and SECTOR erase. SECTOR erase only available in NVR, and the MAIN part supports both CHIP 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.

6.2.1 Functional Description

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

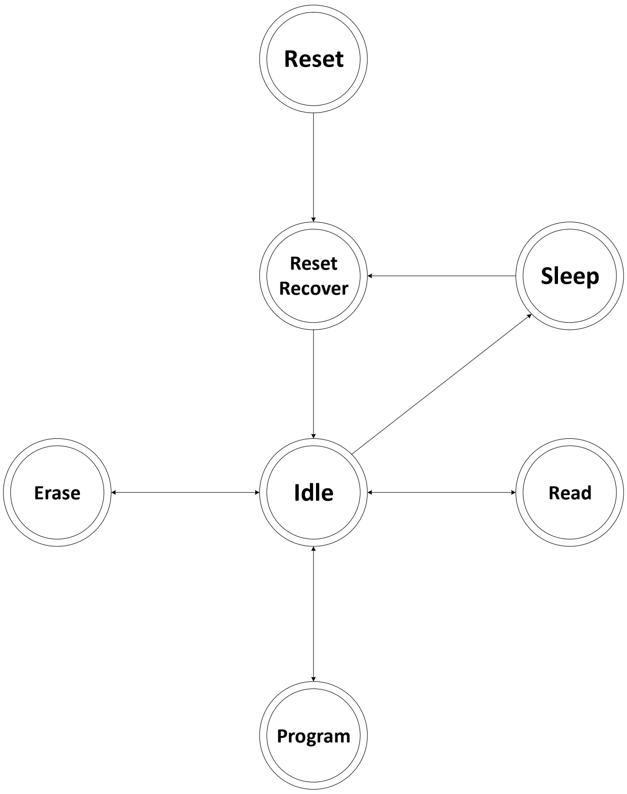


Fig. 6-2 FLASH Control State Transform Diagram

6.2.1.1 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.

6.2.1.2 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.

6.2.1.3 FLASH Read

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

- The CPU 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 CPU accesses the register of the controller through the AHB bus to read FLASH internal data indirectly. Both the data in the MAIN and NVR spaces can be accessed; if a continuous reading is required, the hardware will accumulate the addresses automatically without updating the address register value each time.

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

Table 6-1 LASH Access Space Allocation

NVR (FLASH_CFG.REGION)	Zone of Access
0	MAIN zone
1	NVR zone

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

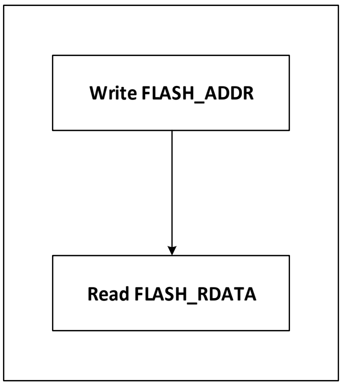


Fig. 6-3 Flow Chart of FLASH Indirect Read

6.2.1.4 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:

- Control configuration register (CFG), enable programming operation
- Address register (ADDR), write programming address
- Write data register (WDATA), write programming data

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

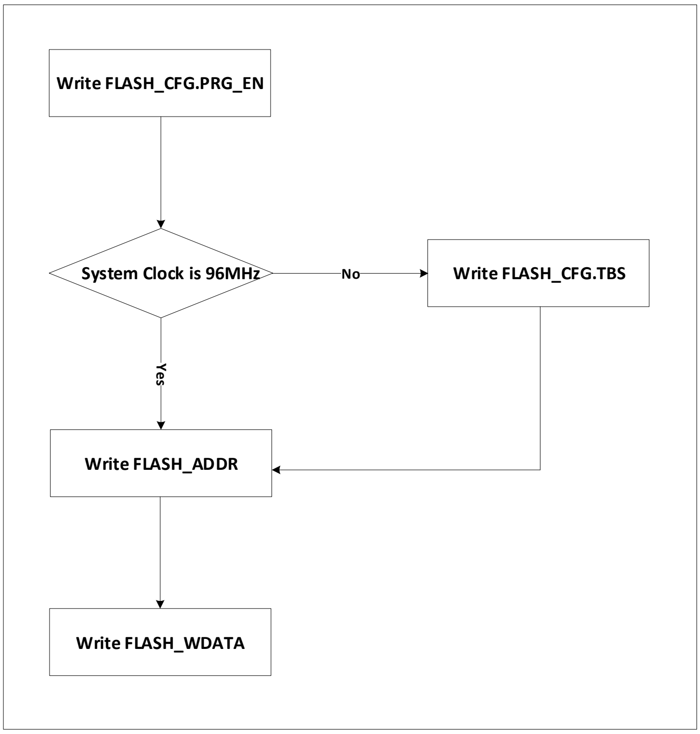


Fig. 6-4 Flow Chart of FLASH Programming

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 96MHz clock frequency; When the chip works at other frequencies, the value of FLASH_CFG.TBS should be set to achieve the count values of 48MHz/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 6.3.2. 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 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.

Figure 6-4 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:

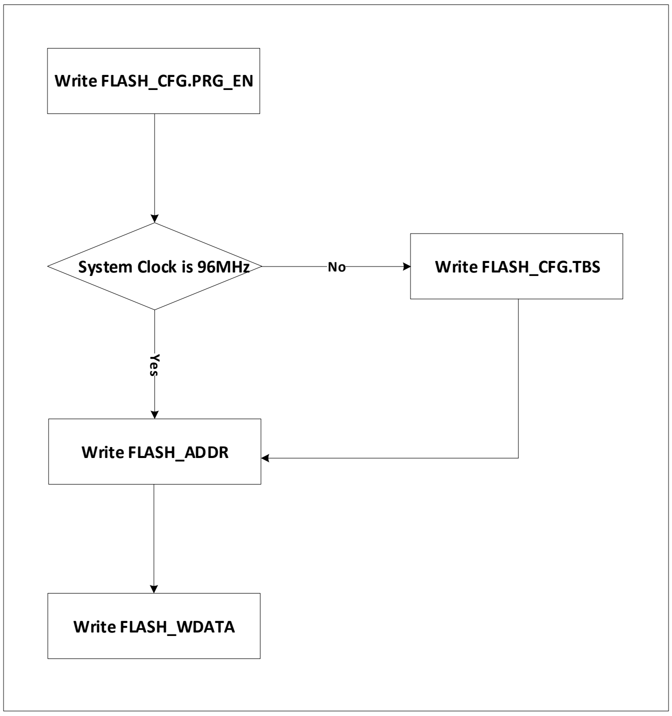


Fig. 6-5 Flow Chart of FLASH Programming

6.2.1.5 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:

- FLASH erase enable
- Address register (ADDR), write erase address
- Write erase register (ERASE), trigger 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 32KB/64kB 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 Block and Sector.

Table 6-2 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
...	...	...
Sector127	0x0000 FE00 - 0x0000 FFFF	512

The NVR area can only realize Sector erasure, and the MAIN area can realize both Sector erasure and FULL erasure. See as follows:

NVR (FLASH_CFG.REGION)	Sector Erase	FULL Erase
0	Main zone	Main zone
1	NVR zone	Main zone

The flash erase operation flow is shown below.

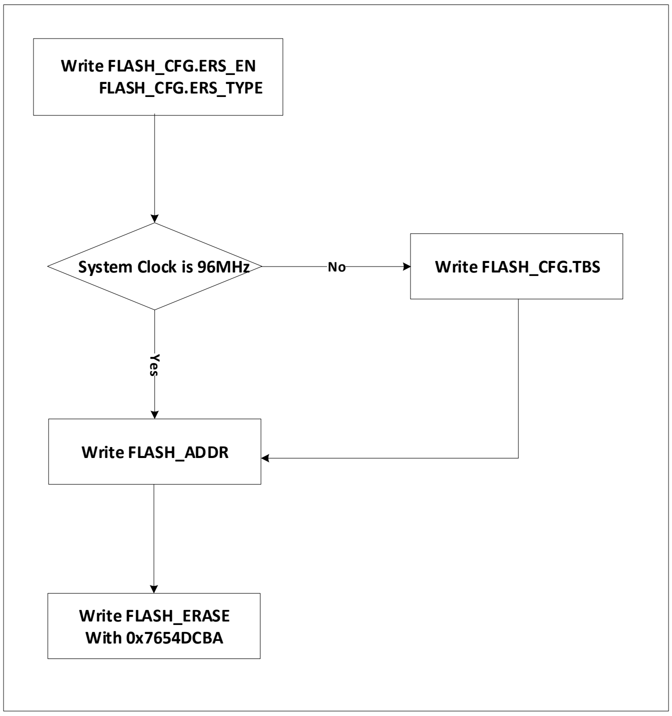


Fig. 6-6 Flow Chart of FLASH Erase

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.

6.2.1.6 FLASH Prefetch

Due to the speed limitation of FLASH memory, this operation cannot reach the speed of 96MHz. When reading the FLASH, it takes more than 1 clock cycle 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.

6.2.1.7 FLASH Encryption

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, 32kB and 64kB. Regardless of the specification, the last WORD in the corresponding specification is designed as an encrypted word. When the content of this WORD is written as all "1", it indicates that FLASH is in decryption state; When the content of this WORD is written as not-all "1", it indicates that the FLASH is in an encrypted state. 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 current 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).

6.2.1.8 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. The LKS32MC08X series chip contains the register VTOR, whose address is 0xE000_ED08, to remap the entry address of the interrupt vector table.

Table 6-3 Register Description of IAP VTOR

Name	Reset Value	Offset	Location	Permission	Description
VTOR	0x0		[31:7]	RW	Perform write operation to write entry address of interrupt vector table
			[6:0]	--	Write “0” by default

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.

6.2.1.8.1 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 CPU 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 CPU soft reset, and the PC restarts the upgraded program from address 0.

6.2.1.8.2 End Interrupted Online Upgrade

Turn off the interrupt controller of the CPU 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 CPU shoul poll the UART interrupt flag bit.

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

6.2.1.8.3 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.

6.3 Register

6.3.1 Address Allocation

The base address of the FLASH controller module register is 0x4000_0400, and the register list is as follows:

Table 6-4 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

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

Address: 0x4000_0400

Reset value: 0x00000060

Table 6-5 FLASH_CFG Configuration Register

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				TBS							
RW				RW				RW							
0				0				60							

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 1: NVR
[6:0]	TBS	Program/erase time base register. The default value is 0x60. **Only the following values can be set:** 0x60: FLASH programming/erasing time base value at 96Mhz operating frequency.

		0x2F: 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.

6.3.3 Address Register (FLASH_ADDR)

Address: 0x4000_0404

Reset value: 0x0

Table 6-6 Address Register (FLASH_ADDR)

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

Location	Bit name	Description
[31:16]		Unused
[15:0]	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.

6.3.4 Write Register (FLASH_WDATA)

Address: 0x4000_0408

Reset value: 0x0

Table 6-7 Write Register (FLASH_WDATA)

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

WDATA
RW
0

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

6.3.5 Read Register (FLASH_RDATA)

Address: 0x4000_040C

Reset value: 0x0

Table 6-8 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

6.3.6 Erase Control Register (FLASH_ERASE)

Address: 0x4000_0410

Reset value: 0x0

Table 6-9 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

6.3.7 Encryption Status Register (FLASH_PROTECT)

Address: 0x4000_0414

Reset value: 0x0

Table 6-10 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.

6.3.8 Working Status Register (FLASH_READY)

Address: 0x4000_0418

Reset value: 0x0

Table 6-11 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	Description

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

6.3.9 NVR correction value address information

Calibration parameters of LKS32MC08x 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 lks32mc08x_nvr.o file contains functions to read the calibration parameters:

Read function: uint32_t Read_Trim(uint32_t adr);

Table 6-122 List of calibration parameters

Address	Content
0x0310	ADC0_DC0 Calibration value
0x0314	ADC0_DC1 Calibration value
0x0318	ADC0_AMC0 Calibration value
0x031C	ADC0_AMC1 Calibration value
0x0320	ADC1_DC0 Calibration value
0x0324	ADC1_DC1 Calibration value
0x0328	ADC1_AMC0 Calibration value
0x032C	ADC1_AMC1 Calibration value
0x0330	DAC selects 3.00V range, SYS _ AFE _ DAC _ AMC calibration value (expands 512 times result)
0x0334	DAC selects 1.20V range,SYS_AFE_DAC_AMC calibration value（expands 512 times result）
0x0338	DAC selects 4.85V range,SYS_AFE_DAC_AMC calibration value（expands 512 times result）
0x0340	DAC selects 3.00V range,SYS_AFE_DAC_DC calibration value
0x0344	DAC selects 1.20V range,SYS_AFE_DAC_DC calibration value
0x0348	DAC selects 4.85V range,SYS_AFE_DAC_DC calibration value
0x0350	OPA0 ,200K Ohm VS 10.4K Ohm,GAIN calibration value（expands 1000 times result）,R0 is 0 Ohm
0x0354	OPA0 ,190K Ohm VS 20.4K Ohm,GAIN calibration value（expands 1000 times result）,R0 is 0 Ohm
0x0358	OPA0 ,180K Ohm VS 30.4K Ohm,GAIN calibration value（expands 1000 times result）,R0 is 0 Ohm
0x035C	OPA0 ,170K Ohm VS 40.4K Ohm,GAIN calibration value（expands 1000 times result）,R0 is

	0 Ohm
0x0360	OPA1 ,200K Ohm VS 10.4K Ohm,GAIN calibration value（expands 1000 times result）,R0 is 0 Ohm
0x0364	OPA1 ,190K Ohm VS 20.4K Ohm,GAIN calibration value（expands 1000 times result）,R0 is 0 Ohm
0x0368	OPA1 ,180K Ohm VS 30.4K Ohm,GAIN calibration value（expands 1000 times result）,R0 is 0 Ohm
0x036C	OPA1 ,170K Ohm VS 40.4K Ohm,GAIN calibration value（expands 1000 times result）,R0 is 0 Ohm
0x0370	OPA2 ,200K Ohm VS 10.4K Ohm,GAIN calibration value（expands 1000 times result）,R0 is 0 Ohm
0x0374	OPA2 ,190K Ohm VS 20.4K Ohm,GAIN calibration value（expands 1000 times result）,R0 is 0 Ohm
0x0378	OPA2 ,180K Ohm VS 30.4K Ohm,GAIN calibration value（expands 1000 times result）,R0 is 0 Ohm
0x037C	OPA2 ,170K Ohm VS 40.4K Ohm,GAIN calibration value（expands 1000 times result）,R0 is 0 Ohm
0x0380	OPA3 ,200K Ohm VS 10.4K Ohm,GAIN calibration value（expands 1000 times result）,R0 is 0 Ohm
0x0384	OPA3 ,190K Ohm VS 20.4K Ohm,GAIN calibration value（expands 1000 times result）,R0 is 0 Ohm
0x0388	OPA3 ,180K Ohm VS 30.4K Ohm,GAIN calibration value（expands 1000 times result）,R0 is 0 Ohm
0x038C	OPA3 ,170K Ohm VS 40.4K Ohm,GAIN calibration value（expands 1000 times result）,R0 is 0 Ohm
0x0398	Temperature sensor, slope calibration value
0x039C	Temperature sensor, slope calibration value
0x03B0	High 16-bit storage OPA1 common-mode voltage value,Low 16-bit storage OPA0 common-mode voltage value (Note that the magnification is 10000 times stored)
0x03B4	High 16-bit storage OPA3 common-mode voltage value,Low 16-bit storage OPA2 common-mode voltage value (Note that the magnification is 10000 times stored)
0x02C0	OPA0,200K Ohm:10.4K Ohm。High 16-bit is the actual resistor value of R2,Low 16-bit is the actual resistor value of R1（expands 100 times result）
0x02C4	OPA0,190K Ohm:20.4K Ohm。High 16-bit is the actual resistor value of R2,Low 16-bit is the actual resistor value of R1（expands 100 times result）
0x02C8	OPA0,180K Ohm:30.4K Ohm。High 16-bit is the actual resistor value of R2,Low 16-bit is the actual resistor value of R1（expands 100 times result）
0x02CC	OPA0,170K Ohm:40.4K Ohm。High 16-bit is the actual resistor value of R2,Low 16-bit is the actual resistor value of R1（expands 100 times result）
0x02D0	OPA1,200K Ohm:10.4K Ohm。High 16-bit is the actual resistor value of R2,Low 16-bit is the actual resistor value of R1（expands 100 times result）
0x02D4	OPA1,190K Ohm:20.4K Ohm。High 16-bit is the actual resistor value of R2,Low 16-bit is the actual resistor value of R1（expands 100 times result）

0x02D8	OPA1,180K Ohm:30.4K Ohm。High 16-bit is the actual resistor value of R2,Low 16-bit is the actual resistor value of R1（expands 100 times result）
0x02DC	OPA1,170K Ohm:40.4K Ohm。High 16-bit is the actual resistor value of R2,Low 16-bit is the actual resistor value of R1（expands 100 times result）
0x02E0	OPA2,200K Ohm:10.4K Ohm。High 16-bit is the actual resistor value of R2,Low 16-bit is the actual resistor value of R1（expands 100 times result）
0x02E4	OPA2,190K Ohm:20.4K Ohm。High 16-bit is the actual resistor value of R2,Low 16-bit is the actual resistor value of R1（expands 100 times result）
0x02E8	OPA2,180K Ohm:30.4K Ohm。High 16-bit is the actual resistor value of R2,Low 16-bit is the actual resistor value of R1（expands 100 times result）
0x02EC	OPA2,170K Ohm:40.4K Ohm。High 16-bit is the actual resistor value of R2,Low 16-bit is the actual resistor value of R1（expands 100 times result）
0x02F0	OPA3,200K Ohm:10.4K Ohm。High 16-bit is the actual resistor value of R2,Low 16-bit is the actual resistor value of R1（expands 100 times result）
0x02F4	OPA3,190K Ohm:20.4K Ohm。High 16-bit is the actual resistor value of R2,Low 16-bit is the actual resistor value of R1（expands 100 times result）
0x02F8	OPA3,180K Ohm:30.4K Ohm。High 16-bit is the actual resistor value of R2,Low 16-bit is the actual resistor value of R1（expands 100 times result）
0x02FC	OPA3,170K Ohm:40.4K Ohm。High 16-bit is the actual resistor value of R2,Low 16-bit is the actual resistor value of R1（expands 100 times result）

7 DMA

7.1 Introduction

After adding DMA, the main device on the bus is increased from CPU to CPU and DMA. The bus architecture needs to evolve from AHB lite to multi-layer AHB lite architecture. As shown in Fig. 7-1. Some devices do not need to be accessed by DMA, and are only mounted on the AHB bridge 0 connected to the CPU. The devices including ADC, DAC, SPI, I2C, MCPWM, UART, and SRAM are shared and accessed by the CPU and DMA, which are mounted on AHB bridge 1.

Basically, it is equivalent to a multiplexer with two choices according to the arbitration nature of the devices. Only the slave device with arbiter on the port can be accessed by DMA, otherwise it cannot be accessed by DMA, and all peripheral devices can be accessed by the CPU.

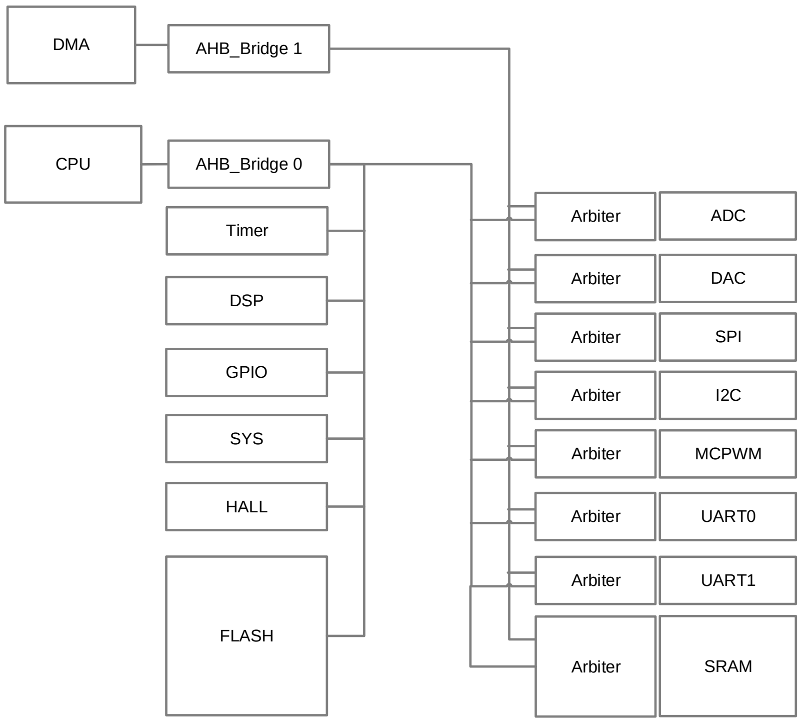


Fig. 7-1 Multi-layer AHB Lite Bus Architecture

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 four channels before turning off the DMA enable), at which time the DMA clock is gated off. DMA contains configuration registers (equivalent to a slave device configured by the CPU) and data handling modules (for the bus master device, initiate various device access requests to the bus).

DMA supports three bit-wide transfer operations, including 8-bit, 16-bit or 32-bit (byte, half-word, or word). Select the bit width of peripheral and memory access by setting the DMA_CCRx.PBTW and DMA_CCRx.MBTW, 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.PINC and DMA_CCRx.MINC. All peripheral register addresses are word aligned, so the peripheral address increment is always 0/4 (set based on DMA_CCRx.PINC=0/1). For example, UART/SPI/I2C, who usually access the fixed address of UART_DATA or SPI/I2C FIFO interface each time, DMA_CCRx.PINC=0; When access the ADC data register, the address should be increased by 4 automatically, and set DMA_CCRx.PINC=1. For memory, if DMA_CCRx.MINC=1 is set, the value of each address increment is set according to the memory data bit width (DMA_CCRx.MBTW); 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.

Please note that DMA transmission is divided into multiple rounds, and there can be multiple transmissions in each round of transmission. DMA_CCRx.PINC is used to control whether the peripheral address is incremented every time it is transferred. Each time the peripheral address is transmitted, the address of the previous round must be repeated, that is, it can be incremented within the round, but not incremented between the rounds; DMA_CCRx.MINC is used to control whether the memory address increments between rounds, and ensure that each round of transmission will repeat the address of the previous round..

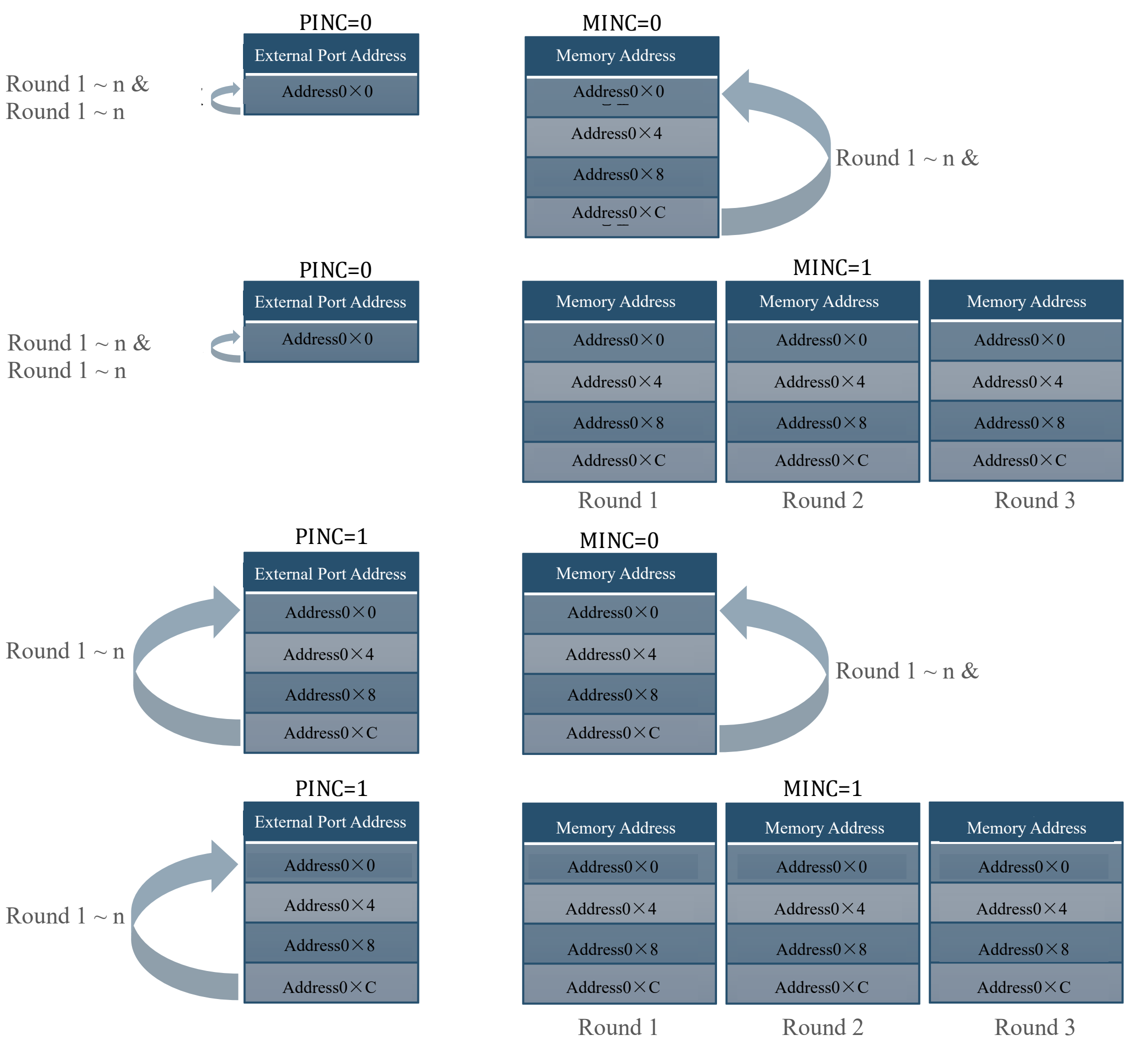


Fig. 7-2 DMA Address Increment Control

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 ADC data to the memory as an example, set the data block size to 16bit for 12channels eight times. The DMA completes a round of 96 half-word transfers in the cyclic mode, and then restarts the next round of transfer, and overwrites the previous memory address, without setting the DMA completion interrupt flag. 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.

7.2 Request

DMA requests have two types: software requests and hardware requests. Software requests are generated by setting DMA_CCRx.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. Besides, the hardware DMA request signal will be cleared by the DMA hardware after accepting the handshake through the DMA channel, and the event flag cannot be cleared by software.

Table 7-1 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_CCRx.SW_TRIG is set, the DMA operation starts once the channel is enabled. **This version of DMA does not support memory to memory handling!**
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. There are two options: Option 1: If the UARTx_IE.TX_BUF_EMPTY_RE is 1. The UARTx module will

	prefetch the first byte for transmission; Once the data enters the transmit queue, UARTx_IE.UARTx_BUFF is empty, and the hardware will automatically request DMA to move the next byte until the data is moved. After the DMA is moved, the DMA completion interrupt will be generated; However, UARTx may not have sent the last byte, and an exception may occur if to operate UARTx immediately. It is recommended to enable the UARTx_IE.TX_DONE_IE interrupt in the DMA interrupt handler. When UARTx finished sending the last byte, and generated a transmission completion interrupt, turned off UARTx_IE.TX_DONE_IE in the UARTx interrupt processing function. Option 2: If the UARTx_IE.TX_DONE_RE is 1. The UART module does prefetch the first byte. Compared with option 1, if the data length of the current transmission is Len, the number of bytes transmitted by the DMA configuration is Len; Turn on the DMA interrupt, after the DMA transfer is completed, the UART is also sent, and the soft reset UARTx module reinitializes UARTx to start the next UARTx transmission.
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 signals 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 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.
CAN	CAN RXFIFO not empty or TX done could generate DMA request.

7.3 Priority

The priority of DMA adopts fixed priority, the priority is shown as Fig. 7-3. 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. 7-3, 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 three 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 three 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.

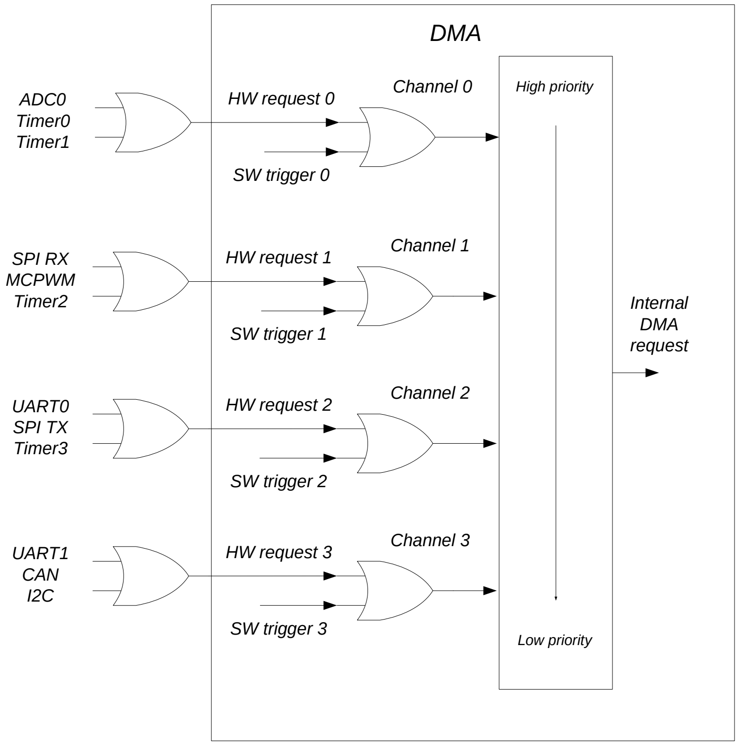


Fig. 7-3 DMA Channel Priority

7.4 Arbitration

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

The priority of CPU accessing RAM is always higher than DMA.

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.

7.5 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.

7.6 Register

7.6.1 Address Allocation

The base address of the DMA controller module register is 0x4001_3000, and the register list is as follows:

Table 7-2 DMA Register List

Name	Offset Address	Description
DMA_CCR0	0x00	DMA channel 0 configuration register
DMA_CDTSZ0	0x04	DMA channel 0 data block size register
DMA_CPAR0	0x08	DMA channel 0 peripheral address register
DMA_CMAR0	0x0C	DMA channel 0 memory address register
DMA_CCR1	0x10	DMA channel 1 configuration register
DMA_CDTSZ1	0x14	DMA channel 1 data block size register
DMA_CPAR1	0x18	DMA channel 1 peripheral address register
DMA_CMAR1	0x1C	DMA channel 1 memory address register
DMA_CCR2	0x20	DMA channel 2 configuration register
DMA_CDTSZ2	0x24	DMA channel 2 data block size register
DMA_CPAR2	0x28	DMA channel 2 peripheral address register
DMA_CMAR2	0x2C	DMA channel 2 memory address register
DMA_CCR3	0x30	DMA channel 3 configuration register
DMA_CDTSZ3	0x34	DMA channel 3 data block size register
DMA_CPAR3	0x38	DMA channel 3 peripheral address register
DMA_CMAR3	0x3C	DMA channel 3 memory address register
DMA_CTRL	0x40	DMA control register
DMA_IF	0x44	DMA interrupt flag register

7.6.2 DMA Controller Register (DMA_CTRL)

Address: 0x4001_3040

Reset value: 0x0

Table 7-3 Controller Register (DMA_CTRL)

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

Location	Bit name	Description
[31:2]		Unused
[1]	PRIORITY	0: CPU priority is higher; 1: DMA priority is higher. The CPU priority must

		be higher for this version.
[0]	EN	DMA Enable

7.6.3 DMA Interrupt Flag Register (DMA_IF)

Address: 0x4001_3044

Reset value: 0x0

Table 7-4 DMA Interrupt Flag Register (DMA_IF)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								CH3_EIF	CH2_EIF	CH1_EIF	CH0_EIF	CH3_FIF	CH2_FIF	CH1_FIF	CH0_FIF
								RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C
								0	0	0	0	0	0	0	0

Location	Bit name	Description
[31:8]		Unused
[7]	CH3_EIF	Channel 3 error interrupt flag
[6]	CH2_EIF	Channel 2 error interrupt flag
[5]	CH1_EIF	Channel 1 error interrupt flag
[4]	CH0_EIF	Channel 0 error interrupt flag
[3]	CH3_FIF	Channel 3 interrupt completion flag
[2]	CH2_FIF	Channel 2 interrupt completion flag
[1]	CH1_FIF	Channel 1 interrupt completion flag
[0]	CH0_FIF	Channel 0 interrupt completion flag

7.6.4 DMA Channel Configuration Register

7.6.4.1 DMA_CCRx (where x = 0,1,2,3)

The addresses are: 0x4001_3000, 0x4001_3010, 0x4001_3020, 0x4001_3030

Reset value: 0x0

Table 7-5 DMA Channel Configuration Register (DMA_CCRx)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SW_TRIG	REQ_EN			MBTW		PBTW		MINC	PINC	CIRC	DIR		TEIE	TCIE	EN
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]	SW_TRIG	Software triggered, active high*2

[14:12]	REQ_EN	Channel x three hardware DMA request enable *1, active high
[11:10 AM]	MBTW	Memory access bit width, 0: byte, 1: half-word, 2: word, 3: reserved
[9:8]	PBTW	Peripheral access bit width, 0: byte, 1: half-word, 2: word, 3: reserved
[7]	MINC	Whether the second round of memory address increases on the basis of the first round of address (increased within the round), active high
[6]	PINC	Indicate whether the peripheral address increases in each round (the peripheral address must repeat the first round of address in the second round), active high
[5]	CIRC	Cycle mode, active high
[4]	DIR	Transfer direction, 0: peripheral to memory, 1: memory to peripheral
[3]		Reserved
[2]	TEIE	Error interrupt enable, active high
[1]	TCIE	Transfer completion interrupt enable, active high
[0]	EN	Channel x enable, active high. This bit is cleared by DMA after all channel operations are completed.

* 1 Taking channel 0 as an example, DMA_CCR0.REQ_EN [2: 0] is the DMA request enable of Timer1, Timer0, and ADC0, respectively. Besides, 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 to the DMA_CCRx.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.*2 Software trigger flag DMA_CCRx.SW_TRIG should be cleared by software after being written to 1.

Table 7-6 DMA Channel Request Signal

DMA Channels	Number of Device Request Signal	Peripheral Modules
Channel 0	0	ADC0
	1	Timer0
	2	Timer1
Channel 1	0	SPI_RX
	1	MCPWM
	2	Timer2
Channel 2	0	UART0
	1	SPI_TX
	2	Timer3
Channel 3	0	UART1
	1	CAN
	2	I2C

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

The addresses are: 0x4001_3004, 0x4001_3014, 0x4001_3024, 0x4001_3034

Reset value: 0x0

Table 7-7 DMA Transfer Count Register (DMA_CTMSx)

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

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

Location	Bit Name	Description
[31:24]		Unused
[11:16]	ROUND	DMA channel x sampling rounds
[15:9]		Unused
[8:0]	TIMES	DMA channel x data transfer times per round, 0 ~ 512. 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 peripheral data width is 16 and memory data width is 32, that is, CTMS.TIMES = 16, CTMS.ROUND = 2, DMA should read peripheral data 16bit×16=32byte and write memory data 32bit×16=64byte each round. A total of two rounds of data need to be carried, that is, reading 64 bytes of peripherals and entering 128 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), CTMS.ROUND no longer works, which is equivalent to an infinite round; In other cases, CTMS.ROUND should be set accordingly, such as CTMS.ROUND=1, which is used to carry one round of data.

7.6.4.3 DMA_CPARx (where x = 0,1,2,3)

The addresses are: 0x4001_3008, 0x4001_3018, 0x4001_3028, 0x4001_3038

Reset value: 0x0

Table 7-8 DMA Peripheral Address Register (DMA_ CPARx)

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

	PERI_ADDR
	RW
	0

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

Location	Bit name	Description
[31:17]		Unused
[16:0]	PERI_ADDR	DMA channel x peripheral address

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

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

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

Since it only carries data between memory and peripherals, DMA_CPAR only stores the lower 17 bits of the peripheral address, the upper 15-bit is always 0x2000, and the upper 20-bit may be 0x40000 (SYS register) or 0x4001*(peripheral registers).

7.6.4.4 DMA_CMARx (where x = 0,1,2,3)

The addresses are: 0x4001_300C, 0x4001_301C, 0x4001_302C, 0x4001_303C

Reset value: 0x0

Table 7-9 DMA Memory Address Register (DMA_CMARx)

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

Location	Bit name	Description
[31:13]		Unused
[12:0]	MEM_ADDR	DMA channel x memory address

When DMA_CCRx.MBTW=2'b01, it is set to carry peripheral data in units of 16-bit. If the value of

CMARx.MEM_ADDR [0] is invalid, the peripheral address will be incremented by 2.

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

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

Since it only carries data between memory and peripherals, DMA_CPAR only stores the lower 13-bit of the peripheral address, corresponding to the SRAM 8kB address space. The upper 19-bit is always 0x10000.

8 GPIO

8.1 Introduction

LSK32MC08X series chips integrate a total of 4 groups of 16-bit GPIO. Four GPIOs, P0.0, P0.1, P1.0, and P1.1 can be used as system wake-up sources. Sixteen GPIOs from P0.15 to P0.0 can be used as external interrupt source input.

P0.2 can be used as external reset pin or GPIO. It can be switched by software setting SYS_RST_CFG.RST_IO. After power on, P0.2 is used as an external reset pin by default. Therefore, it should be noted that the external signal of P0.2 shouldn't keep low after power on; otherwise, the chip will always be in reset state. After the reset is released, the software can set SYS_RST_CFG.RST_IO to 1 and switch P0.2 to GPIO function. SYS_RST_CFG is protected by SYS_WR_PROTECT.

In some low pin count packages, such as SSOP24, P2.15 and SWDIO are multiplexed into the same pin. This multiplexing is achieved by directly bonding together in package. After the chip is powered on, the input and output of P2.15 are disabled by default, but it doesn't affect SWDIO's participation in SWD communication. **If the application should reuse this pin as GPIO, that is, use P2.15. Please retain some means to turn off the output enable of P2.15 again to switch the pin to SWDIO; otherwise, it may cause the chip SWD unusable.**

8.1.1 Functional Block Diagram

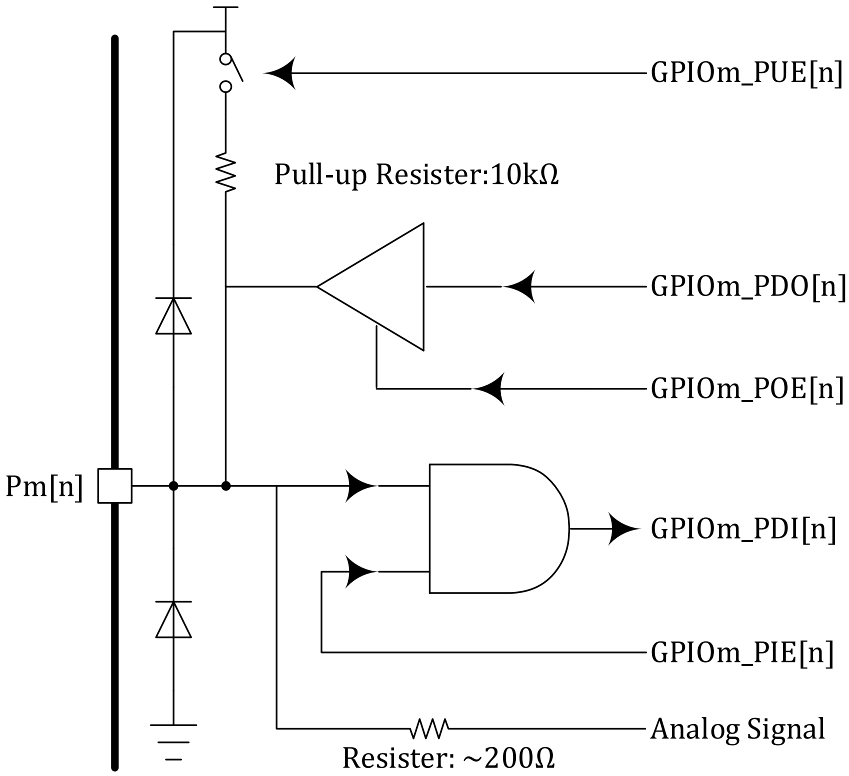


Fig. 8-1 GPIO Functional Block Diagram

As shown in Fig. 8-1, Pm [n] is the chip PAD, m can be 0 ~ 3, which means any group of four 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 Chapter 8.3.2 Pull-up. The PAD without a pull-up resistor can also be set by the GPIOm_PUE [n] register, but it has no practical effect.

8.1.2 Features

- Four groups of 16bit GPIO
- Support open drain
- Some IOs provide internal pull-up resistor
- Support configuration lock protection
- Support external interrupt
- Support GPIO wake-up

8.2 Register

8.2.1 Address Allocation

The base address of the GPIO 0 module in the chip is 0x40012000

The base address of the GPIO 1 module in the chip is 0x40012040.

The base address of the GPIO 2 module in the chip is 0x40012080.

The base address of the GPIO 3 module in the chip is 0x400120C0.

Except for the base address, the register definitions of GPIO 0, 1, 2, and 3 are the same. The register list is as follows:

Table 8-1 GPIOx Register List

Register 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_LCKR	0x1C	GPIO x configuration lock
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

The base address of the GIPO interrupt/wake-up/configuration lock module is 0x40012100, and the register list is as follows:

Table 8-2 Register List of GPIO Interrupt/Wake-up/Configuration Lock Module

Name	Offset Address	Description
EXTI_CR0	0x00	GPIO 0 [7: 0] interrupt trigger type
EXTI_CR1	0x04	GPIO 0 [15:8] interrupt trigger type
EXTI_IF	0x08	GPIO interrupt flag
LCKR_PRT	0x0C	GPIO protection lock configuration
WAKE_POL	0x10	GPIO wake signal polarity
WAKE_EN	0x14	GPIO wake-up enable

8.2.2 GPIOx_PIE

The addresses are: 0x4001_2000, 0x4001_2040, 0x4001_2080, 0x4001_20C0

Reset value: 0x0

Table 8-3 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

8.2.3 GPIOx_POE

The addresses are: 0x4001_2004, 0x4001_2044, 0x4001_2084, 0x4001_20C4

Reset value: 0x0

Table 8-4 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

8.2.4 GPIOx_PDI

The addresses are: 0x4001_2008, 0x4001_2048, 0x4001_2088, 0x4001_20C8

Reset value: 0x0

Table 8-5 GPIOx Data Input 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 data input

8.2.5 GPIOx_PDO

The addresses are: 0x4001_200C, 0x4001_204C, 0x4001_208C, 0x4001_20CC

Reset value: 0x0

Table 8-6 GPIOx Data Output 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 data output

8.2.6 GPIOx_PUE

The addresses are: 0x4001_2010, 0x4001_2050, 0x4001-2090, 0x4001_20D0

Reset value: 0x0

Table 8-7 GPIOxPull-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

8.2.7 GPIOx_PODE

The addresses are: 0x4001_2018, 0x4001_2058, 0x4001_2098, 0x4001_20D8

Reset value: 0x0

Table 8-8 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	Description

	name	
[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

8.2.8 GPIOx_LCKR

The addresses are: 0x4001_201C, 0x4001_205C, 0x4001_209C, 0x4001_20DC_

Reset value: 0x0

Table 8-9 GPIOx Configuration Lock Register (GPIOx_LCKR)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLCKR15	PLCKR14	PLCKR13	PLCKR12	PLCKR11	PLCKR10	PLCKR9	PLCKR8	PLCKR7	PLCKR6	PLCKR5	PLCKR4	PLCKR3	PLCKR2	PLCKR1	PLCKR0
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]	PLCKR15	GPIO x[15]/Px[15] configuration lock
[14]	PLCKR14	GPIO x[14]/Px[14] configuration lock
[13]	PLCKR13	GPIO x[13]/Px[13] configuration lock
[12]	PLCKR12	GPIO x[12]/Px[12] configuration lock
[11]	PLCKR11	GPIO x[11]/Px[11] configuration lock
[10]	PLCKR10	GPIO x[10]/Px[10] configuration lock
[9]	PLCKR9	GPIO x[9]/Px[9] configuration lock

[8]	PLCKR8	GPIO x[8]/Px[8] configuration lock
[7]	PLCKR7	GPIO x[7]/Px[7] configuration lock
[6]	PLCKR6	GPIO x[6]/Px[6] configuration lock
[5]	PLCKR5	GPIO x[5]/Px[5] configuration lock
[4]	PLCKR4	GPIO x[4]/Px[4] configuration lock
[3]	PLCKR3	GPIO x[3]/Px[3] configuration lock
[2]	PLCKR2	GPIO x[2]/Px[2] configuration lock
[1]	PLCKR1	GPIO x[1]/Px[1] configuration lock
[0]	PLCKR0	GPIO x[0]/Px[0] configuration lock

Configuration protection, active high; when the configuration is valid, GPIO input/output/pull-down/open drain/function selection cannot be modified; Please note that LCKR can only be rewritten when LCKR_PRT write protection is turned on.

8.2.9 GPIOx_F3210

The addresses are: 0x4001_2020, 0x4001_2060, 0x4001_20A0, 0x4001_20E0

Reset value: 0x0

Table 8-10 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
[3:12 PM]	F3	GPIO x[3]/Px[3] function selection
[11:8]	F2	GPIO x[2]/Px[2] function selection
[7:4]	F1	GPIO x[1]/Px[1] function selection
[3:0]	F0	GPIO x[0]/Px[0] function selection

GPIO pin function multiplexing is shown in Table 8-11.

Table 8-11 GPIO Pin Multiplex Function

GPIOx_Fxxxx Set Value	Second Function Code	Description
0x0	AF0	Analog function
0x1	AF1	SYS_AF, digital signal output function such as comparator and clock
0x2	AF2	HALL
0x3	AF3	MCPWM

0x4	AF4	UART
0x5	AF5	SPI
0x6	AF6	IIC
0x7	AF7	Timer0/Time1
0x8	AF8	Timer2/ Timer3/QEP0/QEP1
0x9	AF9	ADC trigger debug
0xA	AF10	CAN
0xB	AF11	SIF

8.2.10 GPIOx_F7654

The addresses are: 0x4001_2024, 0x4001_2064, 0x4001_20A4, 0x4001_20E4

Reset value: 0x0

Table 8-12 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
[3:12 PM]	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

8.2.11 GPIOx_FBA98

The addresses are: 0x4001_2028, 0x4001_2068, 0x4001_20A8, 0x4001_20E8

Reset value: 0x0

Table 8-13 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	Description

	name	
[31:16]		Unused
[3:12 PM]	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

8.2.12 GPIOx_FFEDC

The addresses are: 0x4001_202C, 0x4001_206C, 0x4001_20AC, 0x4001_20EC

Reset value: 0x0

Table 8-14 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
[3:12 PM]	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.

8.2.13 External Interrupt, Wake-up, Lock Protection

P0.0/P0.1/P1.0/P1.1 these 4 GPIOs can be used as the wake-up source of the system. A total of 16 GPIOs from P0.15 to P0.0 can be used as external interrupt source input. The wake-up function and external interrupt function use the GPIO function of IO, and the second function of GPIO can be configured as 0.

8.2.13.1 EXTI_CR0

Address: 0x4001_2100

Reset value: 0x0

Table 8-15 External Interrupt Configuration Register (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
[3:14 PM]	T7	GPIO 0 [7]/P0 [7] External interrupt trigger type selection 00: not triggered 01: falling edge trigger 10: Rising edge trigger 11: Trigger on both rising and falling edges
[1:12 PM]	T6	GPIO 0 [6]/P0 [6] External interrupt trigger type selection 00: not triggered 01: falling edge trigger 10: Rising edge trigger 11: Trigger on both rising and falling edges
[11:10 AM]	T5	GPIO 0 [5]/P0 [5] External interrupt 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 [4]/P0 [4] External interrupt 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 [3]/P0 [3] External interrupt trigger type selection 00: not triggered 01: falling edge trigger 10: Rising edge trigger 11: Trigger on both rising and falling edges
[5:4]	T2	GPIO 0 [2]/P0 [2] External interrupt 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 [1]/P0 [1] External interrupt 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 interrupt trigger type selection 00: not triggered

		01: falling edge trigger 10: Rising edge trigger 11: Trigger on both rising and falling edges

8.2.13.2 EXTI_CR1

Address: 0x4001_2104

Reset value: 0x0

Table 8-16 External Interrupt Configuration Register (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
[3:14 PM]	T15	GPIO 0 [15]/P0 [15] External interrupt trigger type selection 00: not triggered 01: falling edge trigger 10: Rising edge trigger 11: Trigger on both rising and falling edges
[1:12 PM]	T14	GPIO 0 [14]/P0 [14] External interrupt trigger type selection 00: not triggered 01: falling edge trigger 10: Rising edge trigger 11: Trigger on both rising and falling edges
[11:10 AM]	T13	GPIO 0 [13]/P0 [13] External interrupt 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 0 [12]/P0 [12] External interrupt 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 0 [11]/P0 [11] External interrupt trigger type selection 00: not triggered 01: falling edge trigger 10: Rising edge trigger 11: Trigger on both rising and falling edges

[5:4]	T10	GPIO 0 [10]/P0 [10] External interrupt 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 [9]/P0 [9] External interrupt 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 [8]/P0 [8] External interrupt trigger type selection 00: not triggered 01: falling edge trigger 10: Rising edge trigger 11: Trigger on both rising and falling edges

8.2.13.3 EXTI_IF

Address: 0x4001_2108

Reset value: 0x0

Table 8-17 External Interrupt Configuration 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 0[15]/P0[15] External interrupt flag Interrupt flag is active high, write 1 to clear
[14]	IF14	GPIO 0[14]/P0[14] External interrupt flag Interrupt flag is active high, write 1 to clear
[13]	IF13	GPIO 0[13]/P0[13] External interrupt flag Interrupt flag is active high, write 1 to clear
[12]	IF12	GPIO 0[12]/P0[12] External interrupt flag Interrupt flag is active high, write 1 to clear
[11]	IF11	GPIO 0[11]/P0[11] External interrupt flag Interrupt flag is active high, write 1 to clear
[10]	IF10	GPIO 0[10]/P0[10] External interrupt flag

		Interrupt flag is active high, write 1 to clear
[9]	IF9	GPIO 0[9]/P0[9] External interrupt flag Interrupt flag is active high, write 1 to clear
[8]	IF8	GPIO 0[8]/P0[8] External interrupt flag Interrupt flag is active high, write 1 to clear
[7]	IF7	GPIO 0[7]/P0[7] External interrupt flag Interrupt flag is active high, write 1 to clear
[6]	IF6	GPIO 0[6]/P0[6] External interrupt flag Interrupt flag is active high, write 1 to clear
[5]	IF5	GPIO 0[5]/P0[5] External interrupt flag Interrupt flag is active high, write 1 to clear
[4]	IF4	GPIO 0[4]/P0[4] External interrupt flag Interrupt flag is active high, write 1 to clear
[3]	IF3	GPIO 0[3]/P0[3] External interrupt flag Interrupt flag is active high, write 1 to clear
[2]	IF2	GPIO 0[2]/P0[2] External interrupt flag Interrupt flag is active high, write 1 to clear
[1]	IF1	GPIO 0[1]/P0[1] 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

8.2.13.4 LCKR_PRT

Address: 0x4001_210C

Reset value: 0x0

Table 8-18 Lock Protection Register (LCKR_PRT)

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

Location	Bit name	Description
[31:16]		Unused
[15:0]	PRT	Configure lock write protection; Write 0x5AC4 to turn off the write protection, and then modify GPIO_LCKR; Write any other data to enable write protection; The B [0] state indicates whether the current write protection is enabled, high level indicates that it is in write protection state, and low level indicates that it is in non-write protection state.

8.2.13.5 WAKE_POL

Address: 0x4001_2110

Reset value: 0x0

Table 8-19 External Wake-up Source Polarity Configuration Register (WAKE_POL)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												GPIO1_1_POL	GPIO1_0_POL	GPIO0_1_POL	GPIO0_0_POL
												RW	RW	RW	RW
												0	0	0	0

Location	Bit name	Description
[31:4]		Unused
[3]	GPIO1_1_POL	GPIO 1 [1]/P1 [1] External wakeup trigger level selection 1: high level; 0: low level
[2]	GPIO1_0_POL	GPIO 1 [0]/P1 [0] External wakeup trigger level selection 1: high level; 0: low level
[1]	GPIO0_1_POL	GPIO 0 [1]/P0 [1] External wakeup trigger level selection 1: high level; 0: low level
[0]	GPIO0_0_POL	GPIO 0 [0]/P0 [0] External wakeup trigger level selection 1: high level; 0: low level

8.2.13.6 WAKE_EN

Address: 0x4001_2114

Reset value: 0x0

Table 8-20 External Wake-up Source Enable Configuration Register (WAKE_EN)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												GPIO1_1_WKEN	GPIO1_0_WKEN	GPIO0_1_WKEN	GPIO0_0_WKEN
												RW	RW	RW	RW
												0	0	0	0

Location	Bit name	Description
[31:4]		Unused
[3]	GPIO1_1_WKEN	GPIO 1 [1]/P1 [1] External wake-up enable 1: enable; 0: disable.
[2]	GPIO1_0_WKEN	GPIO 1 [0]/P1 [0] External wake-up enable 1: enable; 0: disable.
[1]	GPIO0_1_WKEN	GPIO 0 [1]/P0 [1] External wake-up enable 1: enable; 0: disable.

[0]	GPIO0_0_WKEN	GPIO 0 [0]/P0 [0] External wake-up enable 1: enable; 0: disable.

8.3 Function Implementation

8.3.1 Pull-up

LKS32MC08X series chips are implemented by internal analog circuits with pull-up function. All GPIOs have a pull-up control register PUE, but not all GPIOs have a pull-up circuit. The GPIOs equipped with the pull-up function are as follows:

Table 8-21 GPIO Pull-up Resource Distribution

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

8.4 Application Guide

8.4.1 Configuration Lock

The chip provides protection for GPIO configuration. When the write protection of LCKR_PRT is enabled, the GPIO_LCKR of three groups of GPIO cannot be modified, GPIO_PIE/GPIO_POE/GPIO_PUE/GPIO_PDE/GPIO_ODE/GPIO_F3210/GPIO_F7654/GPIO_FBA98/GPIO_FFEDC cannot be modified;

If the GPIO configuration should be modified, the LCKR_PRT write protection should be removed first , then write 0 to the corresponding GPIO GPIO_LCKR, unlock the configuration lock, and then modify the GPIO configuration.

Examples are as follows:

```
    GPIO0_PIE    = 0x1234;
    GPIO1_PIE    = 0x7777;
    GPIO2_PIE    = 0xF000;
//----------------------------------------------------
//   lock specific gpio
    GPIO0_LCKR   = 0x0100;          //lock gpio0 here
    GPIO1_LCKR   = 0xFFFF;          //lock gpio1 here
```

```
    GPIO2_LCKR    = 0x8000;            //lock gpio2 here
//---------------------------------------------------
//    modify to test if gpio config is locked
    GPIO0_PIE     = 0x3333;
    GPIO1_PIE     = 0x0000;
    GPIO2_PIE     = 0x0000;
//---------------------------------------------------
//    read gpio config to flag PASS or FAIL
if(GPIO0_PIE     != 0x3233)FAIL;
if(GPIO1_PIE     != 0x7777)FAIL;
if(GPIO2_PIE     != 0x8000)FAIL;

     LCKR_PRT       = 0x0000;      // write any value other than 0x5AC4 to enable lock protect
    GPIO0_LCKR    = 0x0000;
    GPIO1_LCKR    = 0x0000;
    GPIO2_LCKR    = 0x0000;

if(GPIO0_LCKR    != 0x0100)FAIL;
if(GPIO1_LCKR    != 0xFFFF)FAIL;
if(GPIO2_LCKR    != 0x8000)FAIL;

     LCKR_PRT       = 0x5AC4;      // disable protect
    GPIO0_LCKR    = 0x0000;
    GPIO1_LCKR    = 0x0000;
    GPIO2_LCKR    = 0x0000;

if(GPIO0_LCKR    != 0x0000)FAIL;
if(GPIO1_LCKR    != 0x0000)FAIL;
if(GPIO2_LCKR    != 0x0000)FAIL;
```

i=1000;

while(i--);

PASS;

8.4.2 External Interrupt

Examples are as follows:

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 is handled in GPIO interrupt

Increment twice in the 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

i = 1000;

while(i--);

if(irq_flag != 8)FAIL;

i = 1000;

while(i--);

PASS;

}

8.4.3 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.

9 CRC

9.1 Introduction

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

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

9.2 Basic Principles

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

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

9.3 Basic Concepts

9.3.1 Correspondence

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

Polynomials include generator polynomial G (X) and information polynomial C (X).

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

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

9.3.2 Generator Polynomial

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

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

The following conditions should be met:

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

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

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

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

9.3.3 CRC Digits

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

9.3.4 Generation Steps

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

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

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

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

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

Solution:

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

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

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

```
1010000
1011
------------------
0001000
0001011
------------------
0000011
```

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

```
POL=0x13, data=0x77
011101110000000
 10010011
 01111101000000
  10010011
  0110100100000
   10010011
   010000010000
    10010011
```

00010001000

10010011

    00011011

9.4 Register

9.4.1 Address Allocation

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

Table 9-1 CRC Register List

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

9.4.2 Register Description

9.4.2.1 CRC Data Register (CRC_DR)

Address: 0x4001_2400

Reset value: 0x0

Table 9-2 CRC Data Register (CRC_DR)

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">DR</td></tr>
<tr><td colspan="16">RW</td></tr>
<tr><td colspan="16">0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">DR</td></tr>
<tr><td colspan="16">RW</td></tr>
<tr><td colspan="16">0</td></tr>
</table>

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

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

9.4.2.2 CRC Control Register (CRC_CR)

Address: 0x4001_2404

Reset value: 0x0

Table 9-3 CRC Control Register (CRC_CR)

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

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

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

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

9.4.2.3 CRC Initial Code Register (CRC_INIT)

Address: 0x4001_2408

Reset value: 0x0

Table 9-4 CRC Initial Code Register (CRC_INIT)

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

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

Location	Bit name	Description
[31:0]	INIT	Store initial code

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

9.4.2.4 CRC Generation Code Register (CRC_POL)

Address: 0x4001_240C

Reset value: 0x0

Table 9-5 CRC Generation Code Register (CRC_POL)

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

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

Location	Bit name	Description
[31:0]	POL	Store the generator code for the generator polynomial

10 ADC

10.1 Introduction

The LKS32MC08X series chip integrates a 12-bit SARADC, with dual-channel simultaneous sampling, which can sample two channels at the same time. Each sampling circuit corresponds to 10 input channels, a total of 20 channels. The main features are as follows:

- 3Msps sampling and conversion rate, system frequency is 48MHz.
- Support up to 20 analog signal channels
- Support software and hardware trigger
- Cooperate with MCPWM and UTimer unit to trigger ADC sample. Indication signal ADC_TRIGGER can be transmitted through GPIO for debugging.
- Support sampling with custom sampling sequences, such as single-stage, double-stage, and four-stage, and the sequence number and channel number can be set flexibly.
- Support continuous sampling mode.
- Support DMA.
- Support ADC conversion value comparison, set the upper or lower threshold, and the ADC comparison interrupt will be triggered when the threshold is reached.
- 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-stage 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 stage sampling. After sampling is completed, a corresponding stage sampling completion interrupt is generated; Taking the four-stage sampling triggered by MCPWM as an example, each stage is sampled three times (complete three analog samples). TADC [0] triggers the ADC to start the first stage of sampling. After the first stage 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 stage of sampling; In the same way, TADC [2]/TADC [3] triggers the sampling of the third and fourth stages respectively.

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

10.1.1 Functional Block Diagram

The ADC interface includes 20 data registers and several control registers.

The ADC is equipped with a double sampling circuit, which samples two channels at the same time, and then completes the conversion in turn.

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 5-bit setting within the register ADC_CHNx (see 10.2.3 Signal Source and Register for details) Taking ADC_CHN0 as an example. If ADC_CHN0 [4:0] = 0 and ADC_CHN0 [12:8] = 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. Note: Since the double sampling circuit works synchronously, the 0th sampling and the 1st sampling are completed at the same time, that is, the sampling circuit consumes two analog channels selected by ADC_CHNx at a time.

The sampling times register ADC_CHNT0/1 controls the number of samples per stage, and "1" means one sample, "2" means two samples, ..., "12" means twelve samples, ..., "20" means twenty samples. The double sampling circuit will complete two samples at a time; if the configured number of samples is odd, only one sampled analog signal is converted at the last sampling.

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.

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-stage trigger mode, the conversion completion of each stage can trigger one conversion done interruption.

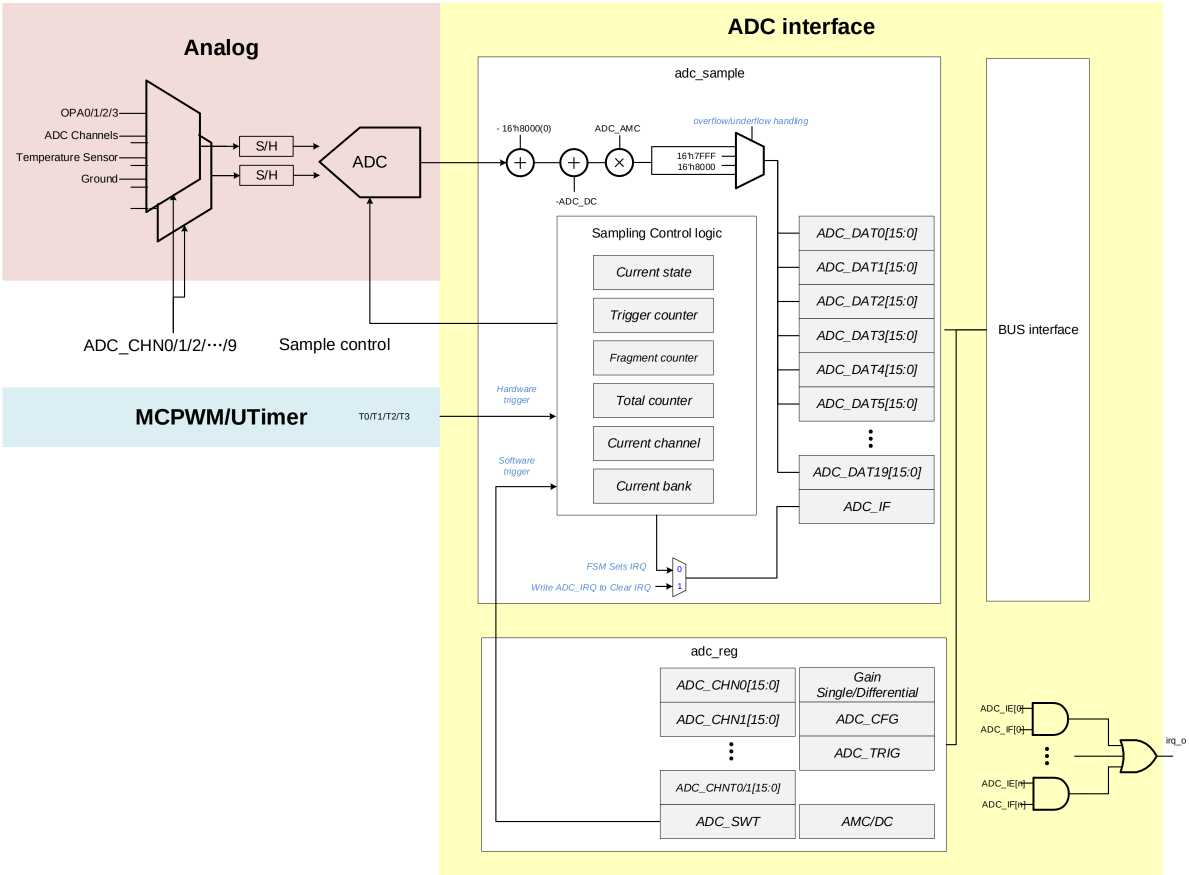


Fig. 10-1 Functional Block Diagram of ADC Sampling

The user can set the sampling sequence and the source of the sampled signal flexibly, and even sample one exact signal for several times. Besides, the ADC gain of each sample can also be configured through the register (two gain level selectable). The control register allows the user to set the number of samples, improve the sampling frequency or reduce the sampling power consumption.

The ADC module is shut down by default when the chip is powered up. Turn on the ADC by setting SYS_AFE_REG5.ADCPDN to 1. Before turning on ADC, the BGP module, 4MHz RC oscillator and PLL should be turned on first, and then select the ADC frequency by setting the SYS_AFE_REG7.ADCCLKSEL.

For the description of ADCPDN, see the analog register 5.3.9 SYS_AFE_REG5.

For the description of ADCCLKSEL<1:0>, see the analog register 5.3.11 SYS_AFE_REG7.

An ADC requires a cycle of 16 ADC clocks to complete one conversion, of which 13 are conversion cycles and 3 are sampling cycles. When the ADC clock is set to 48MHz, the conversion rate is 3Msps.

When the ADC is working at a lower frequency, the power consumption can be reduced by setting register SYS_AFE_REG2.CURRIT.

For the description of CURRIT<1:0>, see the analog register 5.3.6 SYS_AFE_REG2.

10.1.2 ADC Trigger Mode

- Support single-stage trigger, double-stage trigger, four-stage trigger to complete sampling

- Single-stage trigger can set the number of trigger events, and the sampling will start when the trigger event occurs a certain number of times.
- The trigger source of the double-stage trigger can only be the timing signal T0+T1 of MCPWM/UTimer, or the twice software triggers.
- The trigger source of the four-stage trigger can only be the timing signal T0+T1+T2+T3 of MCPWM/UTimer, or quartic software triggers.
- Set to generate an interrupt after each trigger is completed.
- Trigger indication signal can be transmitted through GPIO for debugging

Table 10-1 Trigger Source Configuration of different ADC Trigger Mode

Trigger Mode	Current Stage	Possible Trigger Source		
		MCPWM	UTIMER	Software trigger
Single-stage Trigger	-	T0 for N times or T1 for N times or T2 for N times or T3 for N times or T0/T1/T2/T3 for N times	N times T0 or T1 for N times or T2 for N times for N times or T0/T1/T2/T3 for N times	One-time software trigger
Two-stage Trigger	Stage 1	T0	T0	Software trigger
	Stage 2	T1	T1	Software trigger
Four-stage Trigger	Stage 1	T0	T0	Software trigger
	Stage 2	T1	T1	Software trigger
	Stage 3	T2	T2	Software trigger
	Stage 4	T3	T3	Software trigger

Where N = ADC_TRIG.SINGLE_TNCT.

MCPWM and UTIMER trigger could be enabled at the same time. However, due to sampling timing control, such a configuration will not be used usually.

The number of times sampled in each stage is controlled by ADC_CHNT0/1. ("1" means "1" time, "2" means "2" times, ..., "12" means "12" times).

Table 10-2 Example of the Number of Channels Sampled in Four-stage Trigger Mode

Stage n	ADC_CHNT0/1	Register Value	Number of Sampling Channels in the Current stage
1	ADC_CHNT[3:0]	4	4
2	ADC_CHNT[7:4]	1	1

3	ADC_CHNT[11:8]	6	6
4	ADC_CHNT[15:12]	1	1

10.1.3 ADC Analog Channel

Each ADC module has ten channel signal source registers to control the signal selection of sampling sequence 0 to 19. ADC_CHN0 controls the sampling sequence 0 to 1, ADC_CHN1 controls the sampling sequence 2 to 3, ..., ADC_CHN9 controls the sampling sequence 18 to 19. Each sequence selection range is 0 ~ 19, corresponding to channels 0 ~ 19, that is, a certain channel can be sampled multiple times. Each sample corresponds to a result register. After the conversion is completed, the ADC sampling result can be obtained in the corresponding result register.

Table 10-3 Correspondence between ADC Channel Selection and Register Configuration

ADC Sampling Sequence	Sampling Data Register	Signal Source Register
0th sampling	ADC_DAT0	ADC_CHN0[4:0]
1st sampling	ADC_DAT1	ADC_CHN0[12:8]
2nd sampling	ADC_DAT2	ADC_CHN1[4:0]
3rd sampling	ADC_DAT3	ADC_CHN1[12:8]
......		
18th sampling	ADC_DAT18	ADC_CHN9[4:0]
19th sampling	ADC_DAT19	ADC_CHN9[12:8]

10.1.4 ADC Interrupt

The ADC interrupt signal is set high after each stage is sampled.

If a software or hardware trigger event occurs while the ADC is working, an abnormal trigger interrupt will be generated.

ADC_DAT0 has a threshold interrupt, and upper threshold/lower threshold settings are available. Set the threshold value by ADC_CFG [1], and an interrupt occurs when ADC_DAT0 exceeds ADC_DAT0_TH.

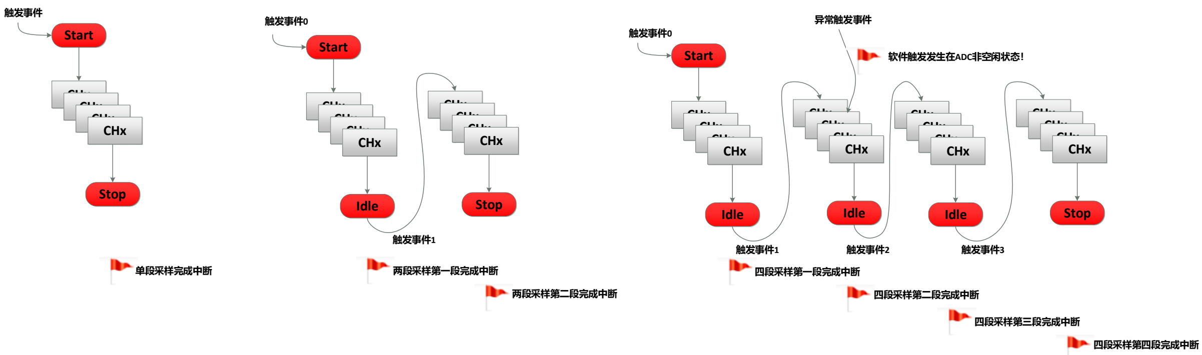


Fig. 10-2 ADC Interrupt Generation

10.1.5 ADC Output Digital System

The ADC output data is 12-bit complement. The input signal 0 corresponds to 12h'0000_0000_0000. Taking the gain configuration 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.

Please note that the final ADC data may exceed the 12-bit signed number range for the sake of gain calibration and DC offset calibration. For example, in the right-aligned mode, the number of ADC conversions may be 0xF745, intercepting the lower 12 bits, and taking out 0x745 at this time may cause negative numbers to be treated as positive numbers, that is, an overflow error may occur. Or the digital quantity of a certain conversion of ADC may be 0x0810. Then, intercept the lower 12bit and take out 0x810 may cause positive numbers to be treated as negative numbers by mistake. Therefore, the ADC data should be processed as a 16-bit signed number.

Table 10-4 Conversion of ADC Output Digital System

ADC Double Gain Input Analog Value/V	ADC 2/3 Times Gain Input Analog 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. 10-3 ADC Digital Range at Double Gain Setting

10.1.6 ADC Reference Voltage and Range

The ADC has two reference voltage configurations 2.4V/1.2V, which can be selected by setting the SYS_AFE_REG1.GAIN_REF bit. See 5.3.5 for details.

10.1.6.1 2.4V Reference Voltage Mode

When the chip operating voltage is 5V, it is recommended to use the default reference voltage of 2.4V. At this time, the minimum power supply voltage for ADC normal operation is 3.2V.

The ADC has two gain modes: high gain (1 times) and low gain (2/3 times). The ADC ranges of these two gains are also different. In 1x gain mode, the maximum input signal amplitude is ± 2.4V; in 2/3 gain 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 ~ OPA3), the appropriate OPA gain should be selected. This will allow the maximum signal in a specific application to be amplified to a level close to +/- 3.3V, while setting the ADC to a gain of 2/3. For example, the maximum phase current is 100A (effective value of sine wave), and the MOS internal resistance (assuming as MOS internal resistance sampling) is 5mR, then the maximum input signal amplitude of the OPA is +/- 707mV. Then, the OPA gain should be selected to be 4.5 times (see 4.6Operational Amplifier (OPA) 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 gain of the ADC should be set to 1 times.

When the ADC sampling channel is set as the input signal of the GPIO multiplex port, the ADC gain is also selected according to the maximum amplitude of the signal. Due to the limitation of the IO port, the signal range of the GPIO multiplex port can only be between -0.3V ~ AVDD+0.3V.

The high and low gain selection is controlled by the ADC_GAIN0/1 gain register.

10.1.6.2 1.2V Reference Voltage Mode

When the chip operating voltage is 3.3V, it is needed to use the default reference voltage of 1.2V. At this time, the minimum power supply voltage for ADC normal operation is 2.8V. Set the ADC reference voltage to 1.2V by setting SYS_AFE_REG1.GAIN_REF = ’1’. Compared with the 2.4V mode, the effective accuracy (ENOB) of the ADC remains unchanged.

The difference between the two modes is:

1) It has no effect on the most commonly used ADC 2/3 times gain setting (+/- 3.6V range). The chip will automatically adjust the gain to 1/3, and the ADC sampling value of the same signal will not change, so as the range.

2) For one time gain setting of ADC, after setting SYS_AFE_REG1.GAIN_REF = '1', the range of the ADC channel using this gain setting is changed to +/- 1.2V, and the ADC value of the same signal will also be doubled.

3) For the internal temperature sensor, the ADC_DAT should be divided by two when calculating the temperature using the a/b coefficient. For the description of the temperature sensor a/b coefficient, please refer to4.8Temperature Sensor (TMP). Since a/b coefficient is calibrated in factory using 2.4V reference. So 2.4V reference is recommended when using internal temperature sensor.

10.1.7 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".

The chip has been calibrated in the factory, and the calibration value is stored in the NVR, which will be automatically loaded when the chip is powered on. When the ADC module is initialized, the DC offset should be set according to the data left-right alignment mode. Please refer to the library functions provided by the chip supplier for details.

Please note that the ADC has high gain and low gain configurations. The two configurations correspond to two sets of correction parameters. Each set of correction data includes a DC offset (hereinafter referred to as DC_{offset}) and a $AMP_{correction}$. Besides, each set of correction parameters includes two sets of sampling circuits a/b corresponding to DC/AMC respectively. The correction coefficient corresponding to high gain is ADC_DC_A1/ADC_AMC_A1 and ADC_DC_B1/ADC_AMC_B1, and the correction coefficient corresponding to low gain is ADC_DC_A0/ADC_AMC_A0 and ADC_DC_B0/ADC_AMC_B0.

Record that the digital quantity output by the ADC is D_{ADC}, the true value corresponding to D_{ADC} is D, and D_0 is 0 in the coding system, then

$D = (D_{ADC}-D_0-DC_{offset})*AMP_{correction}$

Finally, the hardware will store the corrected D into the corresponding sampling data register. The ADC interface hardware circuit will automatically select $AMP_{correction}$ and DC_{offset} according to the gain configuration of each channel (ADC_GAIN0/1).

10.1.8 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_REG5.ADCPDN to 1. Before turning on ADC, the BGP module, 4MHz RC oscillator and PLL should be turned on first, and then select the ADC frequency by setting the SYS_AFE_REG7.ADCCLKSEL. 00 means 48MHz, 01 means 12MHz, 11 means 24MHz.

2. Configure ADC data output format

   The output format of the ADC can be left or right alignment by setting the ADC0_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 ADC0_TRIG.TRG_MODE[13:12]. 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 ADC0_TRIG, and there are a total of 8 sampling events to choose from. In single-segment sampling mode, the ADC0_TRIG.SINGLE_TCNT[11:8] 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 reference voltage of the ADC can be selected by configuring SYS_AFE_REG1.GAIN_REF. 0 means 2.4V, 1 means 1.2V. The gain mode can be selected by configuring the ADC_GAIN0/1. 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 the ADC0_CHNT0 and ADC0_CHNT1 registers can select the number of channels to be sampled in each sampling mode. The setting range is 1~20, and 1 represents one channel. The ADC0_CHN0, ADC0_CHN1 and other registers are configured to select the sampling signal source of the ADC, and the setting range is 0~15.

7. Configuring ADC Interrupts

   The ADC has a total of seven interrupts: the first to fourth segment sampling completion interrupts, software triggered interrupts that occur in non-idle states, hardware triggered interrupts that occur in non-idle states, and ADC0_DTA0 over threshold interrupts. The above interrupts can be enabled by configuring the ADC0_IE register. Even if interrupts are not enabled, interrupt events can still set ADC0_IF, but no interrupt request will be raised.

10.2 Register

10.2.1 Address Allocation

The base address of the ADC in the chip is 0x4001_1400, and the register list is as follows:

Table 10-5 ADC0 Register List

Name	Offset Address	Description
ADC0_DAT0	0x00	ADC 0^{th} sample data
ADC0_DAT1	0x04	ADC 1^{st} sample data
ADC0_DAT2	0x08	ADC 2^{nd} sample data
ADC0_DAT3	0x0C	ADC 3^{rd} sample data
ADC0_DAT4	0x10	ADC 4^{th} sample data
ADC0_DAT5	0x14	ADC 5^{th} sample data
ADC0_DAT6	0x18	ADC 6^{th} sample data
ADC0_DAT7	0x1C	ADC 7^{th} sample data
ADC0_DAT8	0x20	ADC 8^{th} sample data
ADC0_DAT9	0x24	ADC 9^{th} sample data
ADC0_DAT10	0x28	ADC 10^{th} sample data
ADC0_DAT11	0x2C	ADC 11^{th} sample data
ADC0_DAT12	0x30	ADC 12^{th} sample data
ADC0_DAT13	0x34	ADC 13^{th} sample data
ADC0_DAT14	0x38	ADC 14^{th} sample data
ADC0_DAT15	0x3C	ADC 15^{th} sample data
ADC0_DAT16	0x40	ADC 16^{th} sample data
ADC0_DAT17	0x44	ADC 17^{th} sample data
ADC0_DAT18	0x48	ADC 18^{th} sample data

ADC0_DAT19	0x4C	ADC 19th sample data
ADC0_CHN0	0x50	0th/1st Sampling signal selection
ADC0_CHN1	0x54	2nd/3rd Sampling signal selection
ADC0_CHN2	0x58	4th/5h^{t} Sampling signal selection
ADC0_CHN3	0x5C	6th/7th Sampling signal selection
ADC0_CHN4	0x60	8th/9th Sampling signal selection
ADC0_CHN5	0x64	10th/11th Sampling signal selection
ADC0_CHN6	0x68	12th/13th Sampling signal selection
ADC0_CHN7	0x6C	14th/15th Sampling signal selection
ADC0_CHN8	0x70	16th/17th Sampling signal selection
ADC0_CHN9	0x74	18th/19th Sampling signal selection
ADC0_CHNT0	0x78	Number of the first two round ADC sampling times in various trigger modes
ADC0_CHNT1	0x7C	Number of the last two round ADC sampling times in various trigger modes
	0x80	Reserved
	0x84	Reserved
ADC0_GAIN0	0x88	Sample-and-hold circuit gain control of ADC 0th ~ 9th sampling
ADC0_GAIN1	0x8C	Sample-and-hold circuit gain control of ADC channel 10th ~ 19th sampling
ADC0_DC_A0	0x90	Non-one-fold gain DC offset of ADC sample-and-hold circuit A
ADC0_DC_A1	0x94	1x gain DC offset of ADC sample-and-hold circuit A
ADC0_AMC_A0	0x98	Non-one-fold gain correction of ADC sample-and-hold circuit A
ADC0_AMC_A1	0x9C	1x gain correction of ADC sample-and-hold circuit A
ADC0_DC_B0	0xA0	Non-one-fold gain DC offset of ADC sample-and-hold circuit B
ADC0_DC_B1	0xA4	1x gain DC offset of ADC sample-and-hold circuit B
ADC0_AMC_B0	0xA8	Non-one-fold gain correction of ADC sample-and-hold circuit B
ADC0_AMC_B1	0xAC	1x gain correction of ADC sample-and-hold circuit B
ADC0_IE	0xB0	ADC interrupt enable
ADC0_IF	0xB4	ADC interrupt flag
ADC0_CFG	0xB8	ADC alignment mode configuration
ADC0_TRIG	0xBC	ADC sampling mode configuration
ADC0_SWT	0xC0	ADC software trigger
ADC0_DAT0_TH	0xC4	ADC channel 0 threshold register

10.2.2 Sampling Data Register

10.2.2.1 ADC0_DAT0

Address: 0x4001_1400

Reset value: 0x0

Table 10-6 Sampling Data Register (ADC0_DAT0)

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">DAT0</td></tr>
<tr><td colspan="16">RW</td></tr>
<tr><td colspan="16">0</td></tr>
</table>

Location	Bit name	Description
[31:16]		Unused
[15:0]	DAT0	0th Sampling data

10.2.2.2 ADC0_DAT1

Address: 0x4001_1404

Reset value: 0x0

Table 10-7 Sampling Data Register (ADC0_DAT1)

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">DAT1</td></tr>
<tr><td colspan="16">RW</td></tr>
<tr><td colspan="16">0</td></tr>
</table>

Location	Bit name	Description
[31:16]		Unused
[15:0]	DAT1	1st Sampling data

10.2.2.3 ADC0_DAT2

Address: 0x4001_1408

Reset value: 0x0

Table 10-8 Sampling Data Register (ADC0_DAT2)

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">DAT2</td></tr>
<tr><td colspan="16">RW</td></tr>
<tr><td colspan="16">0</td></tr>
</table>

Location	Bit name	Description
[31:16]		Unused

[15:0]	DAT2	2nd Sampling data

10.2.2.4 ADC0_DAT3

Address: 0x4001_140C

Reset value: 0x0

Table 10-9 Sampling Data Register (ADC0_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

10.2.2.5 ADC0_DAT4

Address: 0x4001_1410

Reset value: 0x0

Table 10-10 Sampling Data Register (ADC0_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

10.2.2.6 ADC0_DAT5

Address: 0x4001_1414

Reset value: 0x0

Table 10-11 Sampling Data Register (ADC0_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

10.2.2.7 ADC0_DAT6

Address: 0x4001_1418

Reset value: 0x0

Table 10-12 Sampling Data Register (ADC0_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

10.2.2.8 ADC0_DAT7

Address: 0x4001_141C

Reset value: 0x0

Table 10-13 Sampling Data Register (ADC0_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

10.2.2.9 ADC0_DAT8

Address: 0x4001_1420

Reset value: 0x0

Table 10-14 Sampling Data Register (ADC0_DAT8)

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">DAT8</td></tr>
<tr><td colspan="16">RW</td></tr>
<tr><td colspan="16">0</td></tr>
</table>

Location	Bit name	Description
[31:16]		Unused
[15:0]	DAT8	8th Sampling data

10.2.2.10 ADC0_DAT9

Address: 0x4001_1424

Reset value: 0x0

Table 10-15 Sampling Data Register (ADC0_DAT9)

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">DAT9</td></tr>
<tr><td colspan="16">RW</td></tr>
<tr><td colspan="16">0</td></tr>
</table>

Location	Bit name	Description
[31:16]		Unused
[15:0]	DAT9	9th Sampling data

10.2.2.11 ADC0_DAT10

Address: 0x4001_1428

Reset value: 0x0

Table 10-16 Sampling Data Register (ADC0_DAT10)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT10															
RW															
0															

Location	Bit name	Description
[31:16]		Unused
[15:0]	DAT10	10th Sampling data

10.2.2.12 ADC0_DAT11

Address: 0x4001_142C

Reset value: 0x0

Table 10-17 Sampling Data Register (ADC0_DAT11)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT11															
RW															
0															

Location	Bit name	Description
[31:16]		Unused
[15:0]	DAT11	11th Sampling data

10.2.2.13 ADC0_DAT12

Address: 0x4001_1430

Reset value: 0x0

Table 10-18 Sampling Data Register (ADC0_DAT12)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT12															
RW															
0															

Location	Bit name	Description
[31:16]		Unused

[15:0]	DAT12	12th Sampling data

10.2.2.14 ADC_DAT13

Address: 0x4001_1434

Reset value: 0x0

Table 10-19 Sampling Data Register (ADC0_DAT13)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT13															
RW															
0															

Location	Bit name	Description
[31:16]		Unused
[15:0]	DAT13	13th Sampling data

10.2.2.15 ADC0_DAT14

Address: 0x4001_1438

Reset value: 0x0

Table 10-20 Sampling Data Register (ADC0_DAT14)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT14															
RW															
0															

Location	Bit name	Description
[31:16]		Unused
[15:0]	DAT14	14th Sampling data

10.2.2.16 ADC0_DAT15

Address: 0x4001_143C

Reset value: 0x0

Table 10-21 Sampling Data Register (ADC0_DAT15)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT15															
RW															
0															

Location	Bit name	Description
[31:16]		Unused
[15:0]	DAT15	15th Sampling data

10.2.2.17 ADC0_DAT16

Address: 0x4001_1440

Reset value: 0x0

Table 10-22 Sampling Data Register (ADC0_DAT16)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT16															
RW															
0															

Location	Bit name	Description
[31:16]		Unused
[15:0]	DAT16	16th Sampling data

10.2.2.18 ADC0_DAT17

Address: 0x4001_1444

Reset value: 0x0

Table 10-23 Sampling Data Register (ADC0_DAT17)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT17															
RW															
0															

Location	Bit name	Description
[31:16]		Unused

Location	Bit name	Description
[15:0]	DAT17	17th Sampling data

10.2.2.19 ADC0_DAT18

Address: 0x4001_1448

Reset value: 0x0

Table 10-24 Sampling Data Register (ADC0_DAT18)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT18															
RW															
0															

Location	Bit name	Description
[31:16]		Unused
[15:0]	DAT18	18th Sampling data

10.2.2.20 ADC0_DAT19

Address: 0x4001_144C

Reset value: 0x0

Table 10-25 Sampling Data Register (ADC0_DAT19)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT19															
RW															
0															

Location	Bit name	Description
[31:16]		Unused
[15:0]	DAT19	19th Sampling data

10.2.3 Signal Source Register

10.2.3.1 ADC0_CHN0

Address: 0x4001_1450

Reset value: 0x0

Table 10-26 Signal Source Register (ADC0_CHN0)

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

Location	Bit name	Description
[31:13]		Unused
[12:8]	DS1	ADC sampling signal selection 1
[7:5]		Unused
[4:0]	DS0	ADC sampling signal selection 0

The 0th and 1st ADC sampling is synchronous sampling.

10.2.3.2 ADC0_CHN1

Address: 0x4001_1454

Reset value: 0x0

Table 10-27 Signal Source Register (ADC0_CHN1)

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

Location	Bit name	Description
[31:13]		Unused
[12:8]	DS3	ADC sampling signal selection 3
[7:5]		Unused
[4:0]	DS2	ADC sampling signal selection 2

The 2nd and 3rd ADC sampling is synchronous sampling.

10.2.3.3 ADC0_CHN2

Address: 0x4001_1458

Reset value: 0x0

Table 10-28 Signal Source Register (ADC0_CHN2)

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

15–13	12–8	7–5	4–0
	DS5		DS4
	RW		RW
	0		0

Location	Bit name	Description
[31:13]		Unused
[12:8]	DS5	ADC sampling signal selection 5
[7:5]		Unused
[4:0]	DS4	ADC sampling signal selection 4

The 4th and 5th ADC sampling is synchronous sampling.

10.2.3.4 ADC0_CHN3

Address: 0x4001_145C

Reset value: 0x0

Table 10-29 Signal Source Register (ADC0_CHN3)

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

Location	Bit name	Description
[31:13]		Unused
[12:8]	DS7	ADC sampling signal selection 7
[7:5]		Unused
[4:0]	DS6	ADC sampling signal selection 6

The 6th and 7th ADC sampling is synchronous sampling.

10.2.3.5 ADC0_CHN4

Address: 0x4001_1460

Reset value: 0x0

Table 10-30 Signal Source Register (ADC0_CHN4)

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
[31:13]		Unused
[12:8]	DS9	ADC sampling signal selection 9
[7:5]		Unused
[4:0]	DS8	ADC sampling signal selection 8

The 8th and 9th ADC sampling is synchronous sampling.

10.2.3.6 ADC0_CHN5

Address: 0x4001_1464

Reset value: 0x0

Table 10-31 Signal Source Register (ADC0_CHN5)

15 14 13	12 11 10 9 8	7 6 5	4 3 2 1 0
	DS11		DS10
	RW		RW
	0		0

Location	Bit name	Description
[31:13]		Unused
[12:8]	DS11	ADC sampling signal selection 11
[7:5]		Unused
[4:0]	DS10	ADC sampling signal selection 10

The 10th and 11th ADC sampling is synchronous sampling.

10.2.3.7 ADC0_CHN6

Address: 0x4001_1468

Reset value: 0x0

Table 10-32 Signal Source Register (ADC0_CHN6)

15 14 13	12 11 10 9 8	7 6 5	4 3 2 1 0
	DS13		DS12
	RW		RW
	0		0

Location	Bit name	Description
[31:13]		Unused
[12:8]	DS13	ADC sampling signal selection 13
[7:5]		Unused
[4:0]	DS12	ADC sampling signal selection 12

The 12th and 13th ADC sampling is synchronous sampling.

10.2.3.8 ADC0_CHN7

Address: 0x4001_146C

Reset value: 0x0

Table 10-33 Signal Source Register (ADC0_CHN7)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			DS15								DS14				
			RW								RW				
			0								0				

Location	Bit name	Description
[31:13]		Unused
[12:8]	DS15	ADC sampling signal selection 15
[7:5]		Unused
[4:0]	DS14	ADC sampling signal selection 14

The 14th and 15th ADC sampling is synchronous sampling.

10.2.3.9 ADC0_CHN8

Address: 0x4001_1470

Reset value: 0x0

Table 10-34 Signal Source Register (ADC0_CHN8)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			DS17								DS16				
			RW								RW				
			0								0				

Location	Bit name	Description
[31:13]		Unused
[12:8]	DS17	ADC sampling signal selection 17
[7:5]		Unused
[4:0]	DS16	ADC sampling signal selection 16

The 16th and 17th ADC sampling is synchronous sampling.

10.2.3.10 ADC0_CHN9

Address: 0x4001_1474

Reset value: 0x0

Table 10-35 Signal Source Register (ADC0_CHN9)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			DS19								DS18				
			RW								RW				
			0								0				

Location	Bit name	Description
[31:13]		Unused
[12:8]	DS19	ADC sampling signal selection 19
[7:5]		Unused
[4:0]	DS18	ADC sampling signal selection 18

The 18th and 19th ADC sampling is synchronous sampling.

Table 10-36 Channel Selection of ADC Sampling Signal

Positive selection of input signal of ADC sampling	5’h00 OPA0_OUT 5’h01 OPA1_OUT 5’h02 OPA2_OUT 5’h03 OPA3_OUT 5’h04 ADC_CH4 5’h05 ADC_CH5 5’h06 ADC_CH6 5’h07 ADC_CH7 5’h08 ADC_CH8 5’h09 ADC_CH9 5’h0A ADC_CH10 5’h0B ADC_CH11 5’h0C ADC_CH12

	5'h0D ADC_CH13 5'h0E ADC_CH14 5'h0F ADC_CH15 5'h10 ADC_CH16 5'h11 ADC_CH17 5'h12 Temp 5'h13 VSS

The negative terminal of the input signal of the ADC sampling circuit is grounded uniformly.

Taking the eight samples of single-round trigger sampling as an example, the 0/1, 2/3, 4/5, and 6/7 samples set in the ADC0_CHNx register are synchronous sampling. If the set sampling times are odd, only the two channels will be sampled synchronously at the last time, but only the sampling value of one channel will be converted.

10.2.4 Sampling times Register

10.2.4.1 ADC0_CHNT0

Address: 0x4001_1478

Reset value: 0x0

Table 10-37 Sampling Times Register (ADC0_CHNT0)

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

Location	Bit name	Description
[31:13]		Unused
[12:8]	S2	Number of sampling times for the second round in two-round or four-round sampling mode
[7:5]		Unused
[4:0]	S1	Number of sampling channels for the first round in single-round, two-round or four-round sampling mode

10.2.4.2 ADC0_CHNT1

Address: 0x40011400_ 0x7C

Reset value: 0x0

Table 10-38 Sampling Times Register (ADC0_CHNT1)

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

Location	Bit name	Description
[31:13]		Unused
[12:8]	S4	Number of sampling channels for the fourth round in four- round sampling mode
[7:5]		Unused
[4:0]	S3	Number of sampling channels for the third round in four- round sampling mode

1 means one channel, 2 means two channels, ..., 12 means twelve channels, ..., 20 means twenty channels

10.2.5 Configuration Register

10.2.5.1 ADC0_CFG

Address: 0x4001_14B8

Reset value: 0x0

Table 10-39 Configuration Register (ADC0_CFG)

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

Location	Bit name	Description
[31:3]		Unused
[2]	FSM_RESET	State machine reset. After the software is written, the state machine returns to the idle state, and is automatically cleared upon completion.
[1]	TH_TYPE	ADC0_DAT0_TH as upper threshold or lower threshold 1: Upper threshold 0: Lower threshold
[0]	DATA_ALIGN	ADC0_DAT alignment 0: Left aligned, with 4'h0 at the right, 1: right-aligned, with 4bit sign bit on the left

10.2.5.2 ADC0_TRIG

Address: 0x4001_14BC

Reset value: 0x0

Table 10-40 Trigger Control Register (ADC0_TRIG)

15	14	13 12	11 10 9 8	7	6	5	4	3	2	1	0
	CON_TRIG	TRG_MODE	SINGLE_TCNT	UTIMER3_CMP1_E	UTIMER3_CMP0_E	UTIMER2_CMP1_E	UTIMER2_CMP0_E	MCPWM_TRG3_E	MCPWM_TRG2_E	MCPWM_TRG1_E	MCPWM_TRG0_E
	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
[31:15]		Unused
[14]	CON_TRIG	Continuous trigger enable, active high Write 1, ADC enters continuous trigger mode for continuous sampling Write 0, ADC stops sampling
[1:12 PM]	TRG_MODE	Trigger mode 0: single-round trigger; 1: double-round trigger; 2: reserved; 3: four-round trigger
[11:8]	SINGLE_TCNT	Number of events required to trigger a sample in single-round trigger mode ("0" means one event is triggered, "15" means sixteen events are triggered)
[7]	UTIMER3_CMP1_E	4’b1xxx: UTimer3 CMP1 is enabled
[6]	UTIMER3_CMP0_E	4’bx1xx: UTimer3 CMP0 is enabled
[5]	UTIMER2_CMP1_E	4’bxx1x: UTimer2 CMP1 is enabled
[4]	UTIMER2_CMP0_E	4’bxxx1: UTimer2 CMP0 is enabled
[3]	MCPWM_TRG3_E	4’b1xxx: MCPWM T3 is enabled
[2]	MCPWM_TRG2_E	4’bx1xx: MCPWM T2 is enabled
[1]	MCPWM_TRG1_E	4’bxx1x: MCPWM T1 is enabled
[0]	MCPWM_TRG0_E	4’bxxx1: MCPWM T0 is enabled

TADC [3] = MCPWM_T3 | UTimer3_CMP1

TADC [2] = MCPWM_T2 | UTimer3_CMP0

TADC [1] = MCPWM_T1 | UTimer2_CMP1

TADC [0] = MCPWM_T0 | UTimer2_CMP0

Four trigger sources from two different devices are sent to the ADC sampling module as a trigger event TADC [3: 0].

Before entering the continuous sampling mode, the ADC sampling module must be in an idle state. It is recommended to turn off all hardware trigger enable of ADC, and write ADC0_CFG.FSM_RESET=1 to make ADC sampling state machine return to idle state.

The trigger signal of MCPWM to ADC can be sent by setting GPIO as the ninth function, that is, ADC_TRIGGER function, which is used to capture and debug. The ADC trigger signal is a narrow pulse of one ADC clock cycle inside the chip. Every time an ADC trigger occurs, the ADC_TRIGGER signal flips over for capture output.

The trigger signal of UTimer to ADC can be sent by setting GPIO as the eighth function, that is, Timer2/3 function, which is used to capture and debug.

10.2.6 Gain Selection Register

10.2.6.1 ADC0_GAIN0

Address: 0x4001_1488

Reset value: 0x0

Table 10-41 Gain Selection Register (ADC0_GAIN0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						DG9	DG8	DG7	DG6	DG5	DG4	DG3	DG2	DG1	DG0
						RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
						0	0	0	0	0	0	0	0	0	0

Location	Bit name	Description
[31:10]		Unused
[9]	DG9	ADC0_DAT9 sample-and-hold circuit gain selection
[8]	DG8	ADC0_DAT8 sample-and-hold circuit gain selection
[7]	DG7	ADC0_DAT7 sample-and-hold circuit gain selection
[6]	DG6	ADC0_DAT6 sample-and-hold circuit gain selection
[5]	DG5	ADC0_DAT5 sample-and-hold circuit gain selection
[4]	DG4	ADC0_DAT4 sample-and-hold circuit gain selection
[3]	DG3	ADC0_DAT3 sample-and-hold circuit gain selection
[2]	DG2	ADC0_DAT2 sample-and-hold circuit gain selection
[1]	DG1	ADC0_DAT1 sample-and-hold circuit gain selection
[0]	DG0	ADC0_DAT0 sample-and-hold circuit gain selection

0: 2/3 gain, 1: 1x gain.

10.2.6.2 ADC0_GAIN1

Address: 0x4001_148C

Reset value: 0x0

Table 10-42 Gain Selection Register (ADC0_GAIN1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						DG19	DG18	DG17	DG16	DG15	DG14	DG13	DG12	DG11	DG10
						RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
						0	0	0	0	0	0	0	0	0	0

Location	Bit name	Description
[31:10]		Unused
[9]	DG19	ADC0_DAT19 sample-and-hold circuit gain selection
[8]	DG18	ADC0_DAT18 sample-and-hold circuit gain selection
[7]	DG17	ADC0_DAT17 sample-and-hold circuit gain selection
[6]	DG16	ADC0_DAT16 sample-and-hold circuit gain selection
[5]	DG15	ADC0_DAT15 sample-and-hold circuit gain selection
[4]	DG14	ADC0_DAT14 sample-and-hold circuit gain selection
[3]	DG13	ADC0_DAT13 sample-and-hold circuit gain selection
[2]	DG12	ADC0_DAT12 sample-and-hold circuit gain selection
[1]	DG11	ADC0_DAT11 sample-and-hold circuit gain selection
[0]	DG10	ADC0_DAT10 sample-and-hold circuit gain selection

0: 2/3 gain, 1: 1x gain.

10.2.7 Interrupt Enable Register

10.2.7.1 ADC0_IE

Address: 0x4001_14B0

Reset value: 0x0

Table 10-43 Interrupt Enable Register (ADC0_IE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									TH_IE	HERR_IE	SERR_IE	S4FINISH_IE	S3FINISH_IE	S2FINISH_IE	S1FINISH_IE
									RW	RW	RW	RW	RW	RW	RW

	0	0	0	0	0	0	0

Location	Bit name	Description
[31:7]		Unused
[6]	TH_IE	ADC0_DAT0 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]	S4FINISH_IE	Interrupt enable triggered by the completion of the fourth-round sampling
[2]	S3FINISH_IE	Interrupt enable triggered by the completion of the third-round sampling
[1]	S2FINISH_IE	Interrupt enable triggered by the completion of the second-round sampling
[0]	S1FINISH_IE	Interrupt enable triggered by the completion of the first-round sampling

10.2.7.2 ADC0_IF

Address: 0x4001_14B4

Reset value: 0x0

Table 10-44 Interrupt Flag Register (ADC0_IF)

15 14 13 12 11 10 9 8 7	6	5	4	3	2	1	0
	TH_IF	HERR_IF	SERR_IF	S4FINISH_IF	S3FINISH_IF	S2FINISH_IF	S1FINISH_IF
	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C
	0	0	0	0	0	0	0

Location	Bit name	Description
[31:7]		Unused
[6]	TH_IF	ADC0_DAT0 over threshold interrupt flag
[5]	HERR_IF	Interrupt flag for hardware trigger occurs in non-idle state
[4]	SERR_IF	Interrupt flag for software trigger occurs in non-idle state
[3]	S4FINISH_IF	Interrupt flag for the fourth-round sampling completion
[2]	S3FINISH_IF	Interrupt flag for the third-round sampling completion
[1]	S2FINISH_IF	Interrupt flag for the second-round sampling completion
[0]	S1FINISH_IF	Interrupt flag for the first-round sampling completion

In the above ADC0_IF flag, 0: indicates that no interrupt has occurred, 1: indicates that an interrupt has occurred, write 1 to clear.

10.2.8 Software Trigger Register

10.2.8.1 ADC0_SWT

Address: 0x4001_14C0

Reset value: 0x0

Table 10-45 Software Trigger Register (ADC0_SWT)

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">SWT</td></tr>
<tr><td colspan="16">WO</td></tr>
<tr><td colspan="16">0</td></tr>
</table>

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 write 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.

10.2.9 DC Bias Register

The chip has been calibrated in the factory, and the calibration value is saved in the Flash info area, which will be automatically loaded when the chip is powered on. When the ADC module is initialized, the DC offset should be set according to the left-right alignment mode. For details, please refer to the library functions provided by the chip supplier.

10.2.9.1 ADC0_DC_A0

Address: 0x4001_1490

Reset value: 0x0

Table 10-46 DC Bias Register (ADC0_DC_A0)

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">DC_OFFSET</td></tr>
<tr><td colspan="16">RW</td></tr>
<tr><td colspan="16">0</td></tr>
</table>

Location	Bit name	Description

[31:16]		Unused
[15:0]	DC_OFFSET	Sampling circuit A: Non-double gain ADC DC offset

10.2.9.2 ADC0_DC_A1

Address: 0x4001_1494

Reset value: 0x0

Table 10-47 DC Bias Register (ADC0_DC_A1)

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

Location	Bit name	Description
[31:16]		Unused
[15:0]	DC_OFFSET	Sampling circuit A: Double gain ADC DC offset

10.2.9.3 ADC0_DC_B0

Address: 0x4001_14A0

Reset value: 0x0

Table 10-48 DC Bias Register (ADC0_DC_B0)

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

Location	Bit name	Description
[31:16]		Unused
[15:0]	DC_OFFSET	Sampling circuit B: Non-double gain ADC DC offset

10.2.9.4 ADC0_DC_B1

Address: 0x4001_14A4

Reset value: 0x0

Table 10-49 DC Bias Register (ADC0_DC_B1)

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

Location	Bit name	Description
[31:16]		Unused
[15:0]	DC_OFFSET	Sampling circuit B: Double gain ADC DC offset

10.2.10 Gain Calibration Register

10.2.10.1 ADC0_AMC_A0

Address: 0x4001_1498

Reset value: 0x0

Table 10-50 Gain Calibration Register (ADC0_AMC_A0)

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

Location	Bit name	Description
[31:10]		Unused
[9:0]	AM_CALI	Non-double gain ADC gain calibration register for sampling circuit A

10.2.10.2 ADC0_AMC_A1

Address: 0x4001_149C

Reset value: 0x0

Table 10-51 Gain Calibration Register (ADC0_AMC_A1)

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

Location	Bit name	Description
[31:10]		Unused
[9:0]	AM_CALI	Double gain ADC gain calibration register for sampling circuit A

10.2.10.3 ADC0_AMC_B0

Address: 0x4001_14A8

Reset value: 0x0

Table 10-52 Gain Calibration Register (ADC0_AMC_B0)

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

Location	Bit name	Description
[31:10]		Unused
[9:0]	AM_CALI	Non-double gain ADC gain calibration register for sampling circuit B

10.2.10.4 ADC0_AMC_B1

Address: 0x4001_14AC

Reset value: 0x0

Table 10-53 Gain Calibration Register (ADC0_AMC_B1)

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

Location	Bit name	Description
[31:10]		Unused
[9:0]	AM_CALI	Double gain ADC gain calibration register for sampling circuit B

The $AMP_{correction}$, which is a gain calibration factor, stored by ADC0_AMC is a 10-bit unsigned fixed-point number. ADC_AMC [9] is the integer, and ADC_AMC [8: 0] is the decimal. The stored value is around "1".

The ADC has 1x gain and non-1x gain two configurations. The two configurations correspond to two sets of correction parameters. Each set of correction data includes a DC offset (hereinafter referred to as DC_{offset}) and a $AMP_{correction}$.

Record that the digital quantity output by the ADC is D_{ADC}, the true value corresponding to D_{ADC} is D, and D_0 is 0 in the coding system, then

$D = (D_{ADC}-D_0)*AMP_{correction}-DC_{offset}$

The hardware will select the DC and AMC value corresponding to A and B circuits automatically when correcting data on different channels.

10.2.11 Threshold Register for Channel 0

10.2.11.1 ADC0_DAT0_TH

Address: 0x4001_14C4

Reset value: 0x0

Table 10-54 Threshold Register for Channel 0 (ADC0_DAT0_TH)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT0_TH															
RW															
0															

Location	Bit name	Description
[31:16]		Unused
[15:0]	DAT0_TH	Channel 0 threshold register

This register is used for setting the ADC value comparison interrupt.

When ADC0_DAT0 is greater than or less than the value set by ADC0_DAT0_TH, ADC value comparison interrupt will be triggered. According to the configuration of ADC0_CFG [1], ADC0_DAT0_TH can be used as the upper or lower threshold of ADC0_DAT0, **which can replace the hardware comparator in some applications.** Specifically, when ADC0_CFG.TH_TYPE = 1, that is, ADC0_DAT0_TH is the upper threshold, and if ADC0_DAT0> ADC0_DAT0_TH, a threshold interrupt is generated; When ADC0_CFG.TH_TYPE = 0, that is, ADC0_DAT0_TH is the lower threshold, and if ADC0_DAT0 <ADC0_DAT0_TH, a threshold interrupt is generated.

10.3 Implementation Description

10.3.1 DMA Request

Only ADC_IF.S1FINISHI_IF, the first sample completion event, can be used as a DMA request event. Therefore, the DMA request will be initiated only when the first-round trigger sampling is completed or when the first-round of the multi-round trigger sampling is completed, and this request has no relationship with the ADC_IE setting. After the DMA responds to the request, it will clear ADC_IF.S1FINISHI_IF immediately through the clear-on-write.

10.3.2 Continuous Sampling

When the continuous sampling mode is set, that is, ADC_TRIG.CON_TRIG = 1, the ADC will repeatedly sample each channel in the single-round mode, and immediately start the next round of single-round sampling upon completion; If the single-sampling interrupt is turned on, it will interrupt frequently. In most cases, the CPU has no time to respond to such frequent interrupts. Software writes ADC_TRIG.CON_TRIG = 0 to disable continuous sampling.

A threshold interrupt will be generated and notify the CPU only when a signal is detected that exceeds a threshold.

Continuous sampling is available for multi-channel sampling, and the setting of specific sampling channel is the same as the single-sampling mode.

10.4 Application Guide

10.4.1 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_CHNT0.S1, that is, reaching the sampling times for the first sampling, the ADC will return to the idle state 0 after the first round sampling; if sampling times has not reached the preset value, the sampling continues.

No: When the sampling times reaches the preset value ADC_CHNT.S1, 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

Enter the second round of sampling

Second Round of Sampling

When the sampling times of the second round reaches the preset value ADC_CHNT0.S2, that is, reaching the required sampling times of the second sampling, 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

Enter the third round of sampling. If this state is reached, it must be the four-round sampling.

Third Round of Sampling

When the sampling times of the third round reaches the preset value ADC_CHNT1.S3, that is, reaching the required sampling times of the third sampling, 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_CHNT1.S4, that is, reaching the required sampling times of the third sampling, the ADC will return to the idle state 0.

The trigger conditions of various hardware trigger modes are summarized in Table 10-55. The single-round sampling mode is different. It can determine by the ADC0_CFG register that whether one MCPWM/UTimer timing event triggers sampling, or the sampling is triggered only when a certain number of MCPWM/UTimer timing events had occurred, while the two-round and four-round sampling modes only support the sampling to be triggered by one MCPWM/UTimer timing event.

Besides, the ADC module also supports the sampling to be triggered by writing 0x5AA5 to ADC0_SWT through software. The software trigger also only supports trigger sampling by writing once.

Table 10-55 ADC Sampling Trigger Mode

	Single-round Trigger	Two-round Trigger	Four-round Trigger
Timer Trigger	None (Timer trigger is not enabled)	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
	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]		[2] The fourth round of TADC [3]
Software trigger	Write 16‘h5aa5 to ADC_SWT	First round: Write 0x5AA5 to ADC0_SWT Second round: Write 0x5AA5 to ADC0_SWT	First round: Write 0x5AA5 to ADC0_SWT Second round: Write 0x5AA5 to ADC0_SWT Third round : Write 0x5AA5 to ADC0_SWT Fourth round: Write 0x5AA5 to ADC0_SWT

10.4.1.1 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_CHNT0 [4: 0]; When the register value is 1 ~ 20, the corresponding sampling times are 1 ~ 20.

Assuming that the number of sampling channels for the single-round sampling is "4", the converted data will be filled into ADC0_DAT0, ADC0_DAT1, ADC0_DAT2, and ADC0_DAT3 in turn.

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.../19. 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. 10-4.

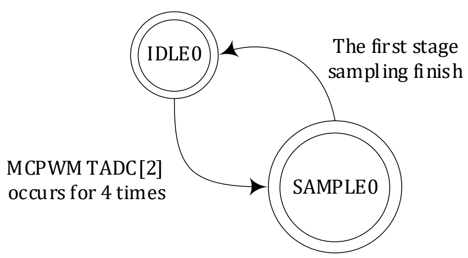


Fig. 10-4 State Transition Diagram of ADC Single-round Sampling

10.4.1.2 Two-round Trigger Mode

Two-round trigger requires two triggers to complete one round of sampling. The first round is sampled when the first trigger arrives and the second round is sampled when the second trigger arrives.

The trigger event can be triggered by external MCPWM/UTimer timing signals TADC [0] and TADC [1] or triggered twice by software.

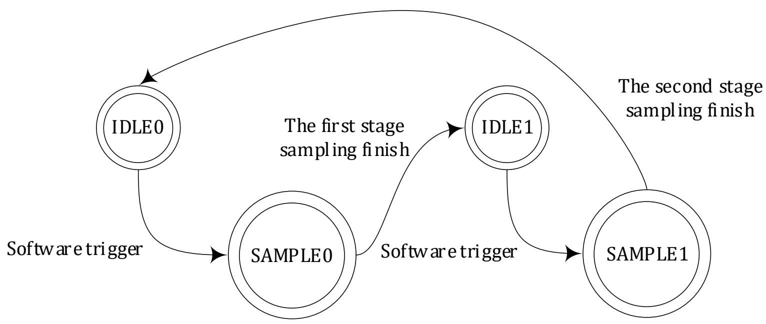
When TADC [0] or software trigger occurs, the 0th to 4th ADC_CHNT0 sampling starts. And then, it will enter the idle state upon sampling completion and wait for the next trigger signal; When TADC [1] or software trigger occurs as the second trigger signal, the 8th to 12th ADC_CHNT0 sampling starts. The sampling times are set by the ADC_CHNT0 round register.

Assuming that the number of channels in the two-round sampling is set as two and three, respectively, the converted data in the first round of sampling will be filled into ADC0_DAT0 and ADC0_DAT1 in turn, and the converted data in the second round of sampling will be filled in ADC0_DAT2, ADC0_DAT3, and ADC0_DAT4 in turn.

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. 10-5.

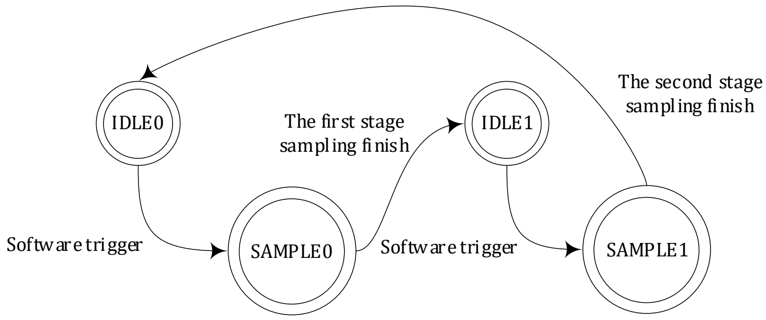


Fig. 10-5 State Transition Diagram of ADC Two-round Sampling

10.4.1.3 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/UTimer, 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_CHNT0 [4: 0], ADC_CHNT0 [12: 8], ADC_CHNT1 [4: 0], and ADC_CHNT1 [12: 8], respectively.

Assuming that the number of channels in the four-round sampling is two, three, one, and five, respectively, the converted data of the first round of sampling will be filled to ADC0_DAT0 and ADC0_DAT1 in turn, the converted data of the first round of sampling will be filled to ADC0_DAT2, ADC0_DAT3 and ADC0_DAT4 in turn, the converted data of the first round of sampling will be filled to ADC0_DAT5, and the converted data of the first round of sampling will be filled to ADC0_DAT62, ADC0_DAT72, ADC0_DAT82, ADC0_DAT9 and ADC0_DAT10 in turn.

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

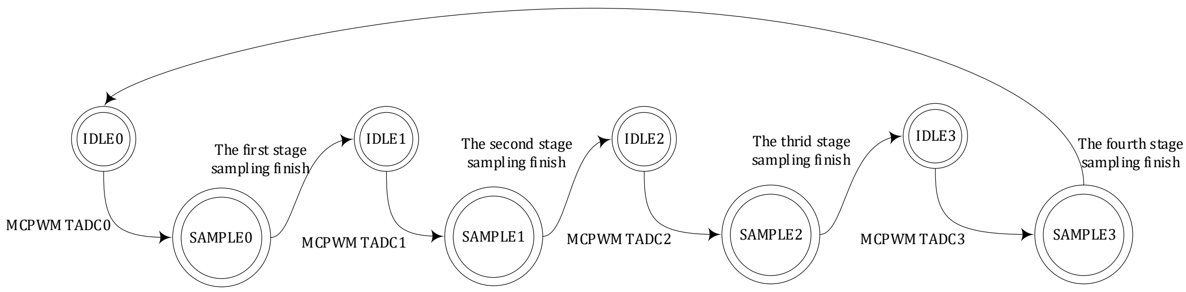


Fig. 10-6.

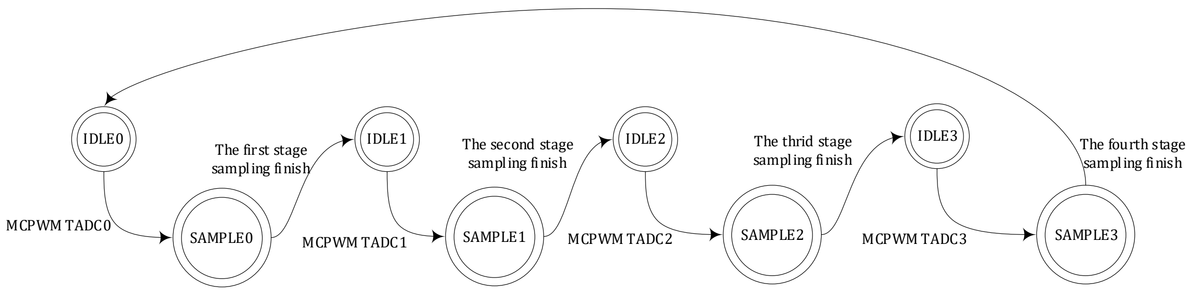


Fig. 10-6 State Transition Diagram of ADC Four-round Sampling

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.

Similarly, if using UTimer to trigger the ADC regularly, UTIMER_UNT2_CMP0/ UTIMER_UNT2_CMP1/ UTIMER_UNT3_CMP0/UTIMER_UNT3_CMP1 should also be set.

10.4.2 Interrupt

10.4.2.1 Done Interruption of Single Round Trigger Sampling

A done interruption is triggered when sampling is completed.

10.4.2.2 Done Interruption of Two Round Trigger Sampling

A done interruption is triggered when the first-round sampling is completed, and another done interruption will be triggered when the second-round sampling is completed.

10.4.2.3 Done Interruption of Four Round Trigger Sampling

One done interruption is triggered when the first-round, second-round, third-round and the fourth-round sampling is completed accordingly.

10.4.3 Configuration Modification

It is recommended to configure and modify ADC0_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 ADC0_CHNx and ADC0_CHNT registers should be modified in the main program, and then write "1" to ADC0_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:

ADC0_TRIG_temp = ADC0_TRIG; // Save ADC sampling trigger configuration

ADC0_TRIG = 0x0000; // Disable ADC sampling trigger

ADC0_CFG | = 0x0004; // Reset the state machine of the ADC interface circuit

//Here are only examples for the modification of the ADC sampling channels and the number of channels.

ADC0_CHNT0 = 0x0005 // Modify the number of ADC single sampling channels to 5

ADC0_CHN0 = 0x0305; // Modify the 0th and 1st ADC sampling channels to analog channels 5 and 3

ADC0_CHN1 = 0x0604; // Modify the 2th and 3th ADC sampling channels to analog channels 4 and 6

ADC0_TRIG = ADC0_TRIG_temp; // Restore ADC sampling trigger configuration

10.4.4 Select the Corresponding Analog Channel

For the channel corresponding to the signal sampled by the ADC, please refer to Table 2.2 Pin Function Selection in Table 10-36 and DATASHEET. Turn off the corresponding IO IE and OE to us function.

11 General Timer

11.1 Introduction

11.1.1 Functional Block Diagram

As shown in Fig. 11-1, The universal timer UTIMER mainly includes four 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.

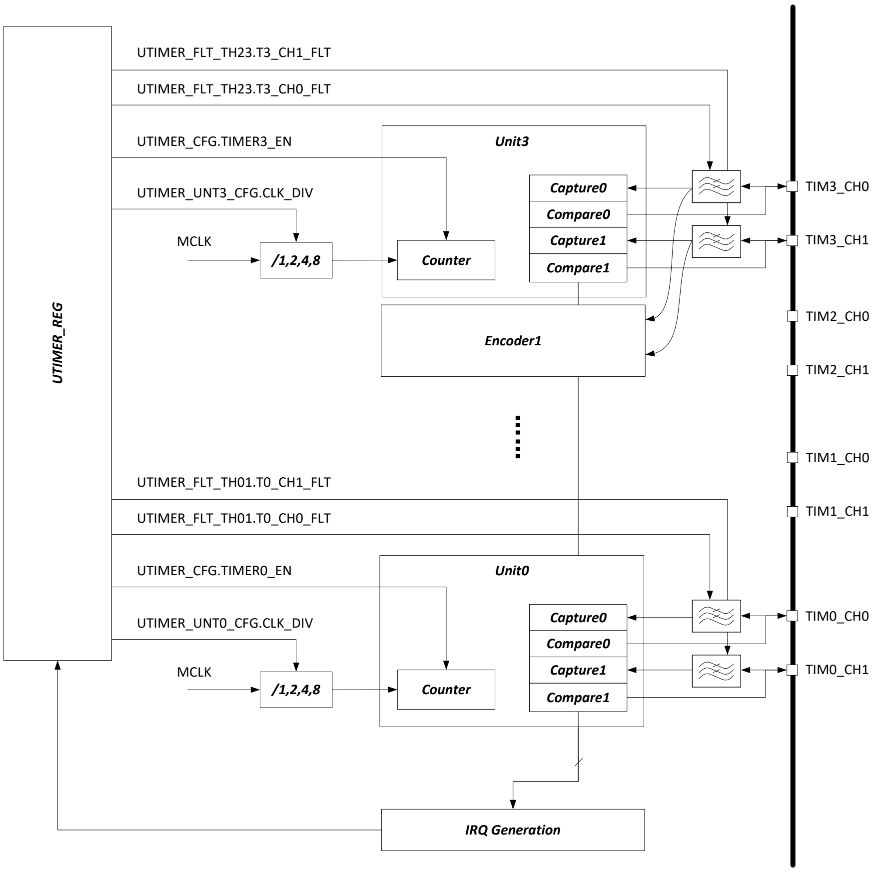


Fig. 11-1 Block Top Functional Block Diagram

11.1.1.1 Bus Interface Module

The bus interface module includes:

Translate the access signals from the AHB bus into register read and write signals, control the clock of the register module, and initiate read and write to the register module.

CG clock gating module. When the AHB bus has no access, turn off the register module clock to reduce power consumption.

11.1.1.2 Register Module

Register module (utimer_reg) for realizing:

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.

11.1.1.3 IO Filter Module

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

11.1.1.4 General Timer Module

The utimer_unt module implements general timer functions, including comparison and capture modes, which can process two external input signals or generate two pulse signals to be transmitted outside the chip. The timer module includes a total of four independent general timers. The bit width of Timer0/Unit0 and Timer1/Unit1 is 16-bit, and the bit width of Timer2/Unit2 and Timer3/Unit3 is 32-bit. Each timer has two channels.

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.

11.1.1.5 Encoder Module

The encoder module is used to count the encoder code signals sent from outside the chip. There are two encoder modules integrated on the timer module, where the encoder inputs are from Timer 2 channel 0/1 and Timer 3 channel 0/1, respectively, and it will not affect the Timer function when using the encoder.

11.1.1.6 Clock Divider Module

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

11.1.2 Features

The timer module has the following characteristics:

- Available for working in at different frequencies independently
- Timer0 and Timer1 are 16-bit general timers
- Timer2 and Timer3 are 32bit general timers
- Each general timer processes two external input signals (capture mode) or generates two output signals (comparison mode)
- Two independent counters

- Available for filtering each input signal of up to 120 main clocks, that is, when the chip works at a clock frequency of 96MHz, it can filter out glitches below 1.25uS width.
- Support DMA transmission

11.2 Implementation Description

11.2.1 Clock Divider

Each timer works at the main system frequency, and the frequency division counter is adopted to reduce the counting frequency, thus to realize the independent frequency division of each timer for better write interrupt/count value.

11.2.2 Filtering

The timer module has a total of 8/4 pairs of channel inputs, and the timer can filter each input to varying degrees.

The filter width can be adjusted by setting the filter register, 0 ~ 120 system clock widths are available.

Usually, high-speed clock is used for filtering input signals, which generally a 96MHz PLL clock. UTIMER_UNTx_CFG.CLK_DIV has no impact on the division of the timer count clock and the filtered clock.

As shown in Fig. 11-2, The original input signal was reversed at several moments from t1 to t6, and the filter width was set as T. It can be seen that only the inversions that occur at times t3 and t6 are maintained for a time greater than T, and we can see from the filter output that the signal has only inverted twice.

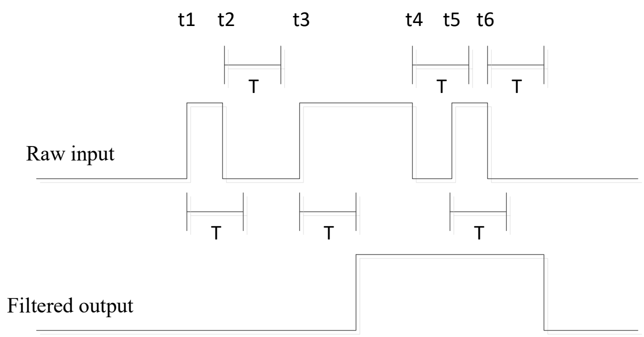


Fig. 11-2 Filter Diagram

11.2.3 Mode

11.2.3.1 Counter

The counter in Timer counts in increasing direction.

The counter counts from 0 to the TH value, and then returns to 0 to restart counting. When the counter returns to 0, a zero return interrupt is generated. The timer period is clk_freq*(TH+1).

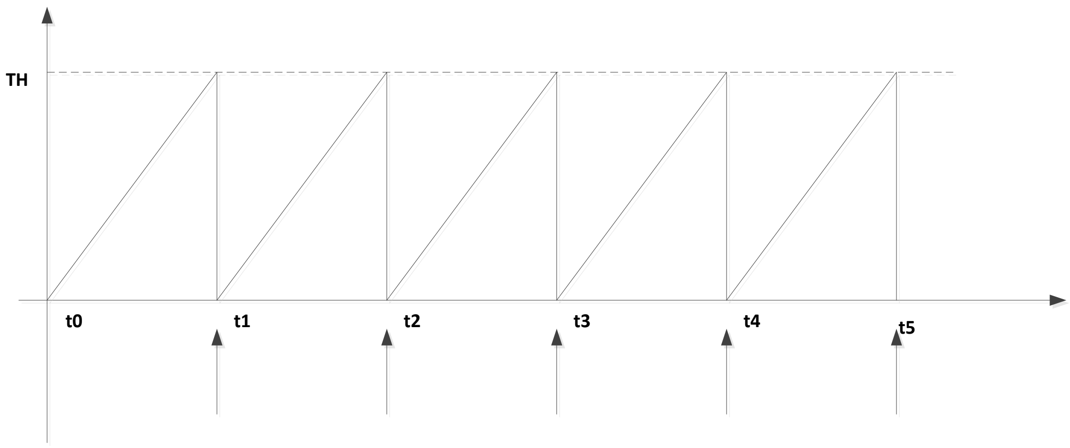


Fig. 11-3 General Counter

11.2.3.2 Comparison Mode

In compare mode, a compare pulse can be driven, and a compare interrupt is generated when the counter counts to the UTIMER_UNTx_CMP value. 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.

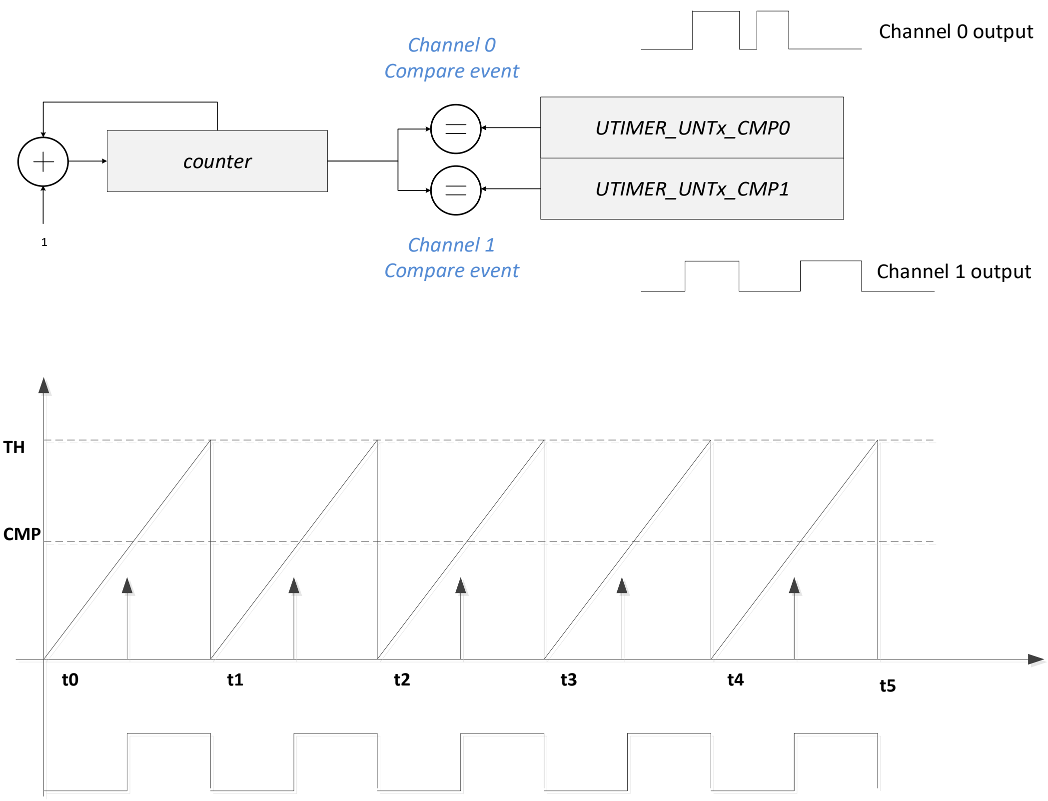


Fig. 11-4 Comparison Mode

11.2.3.3 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.

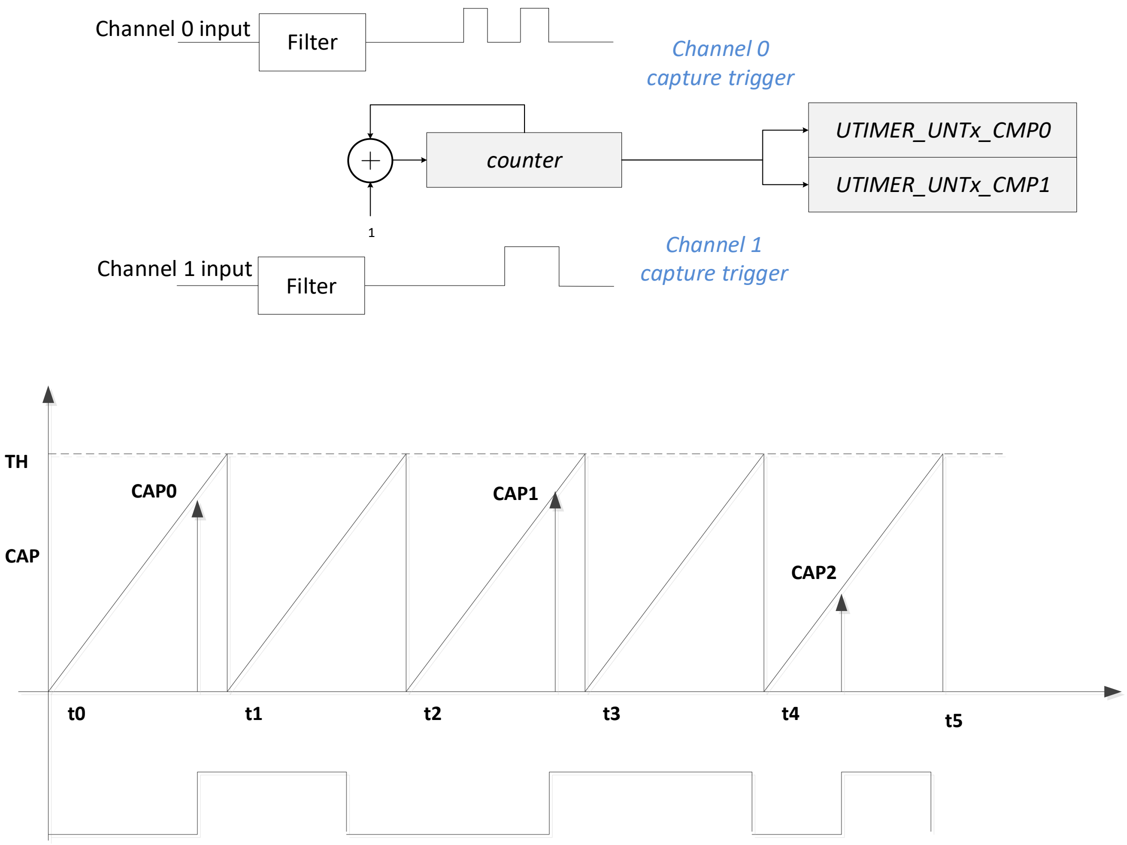


Fig. 11-5 Capture Mode

As shown in Figure 10-5, the timer is set to capture on the rising edge. When changes in the rising edge of the input signal is captured at timing CAP0, CAP1, and CAP2, the timer count value at the corresponding time will be stored in the UTIMER_UNTx_CMP register.

11.2.4 Encoder

The encoder interface supports three modes: orthogonal coded signal, pulse signal with symbolic data, and CW/CCW double pulse signal.

T1 and T2 input signals of Encoder0 come from GPIO input corresponding to Timer2 Channel0 and Channel1 respectively; T1 and T2 input signals of Encoder1 come from GPIO input corresponding to Timer3 Channel0 and Channel1 respectively. Enabling the encoder function does not affect the normal use of the Timer function.

11.2.4.1 Orthogonal Coded Signal

The quadrature coded signal is mostly used to count the number of encoder turns. The input signals are T1 and T2, which support the two modes in the following table.

Generally speaking, the transition edge of T1 and T2 will cause the counter to increment or decrement. The counter counting direction (increasing or decreasing) is determined by the level of another steady-state signal other than the transition signal.

If T1 has a rising edge transition, then check whether T2 is high or low level. If it is a high level, the counter decrements, if it is a low level, the counter increments; The change of T1 falling edge counter is just the opposite.

If T2 has a rising edge transition, then check whether T1 is high or low level. If it is a high level, the counter decrements, if it is a low level, the counter increments; The change of T2 falling edge counter is just the opposite.

The formula is as follows:

Counter Up = (T1 !=T2) @(T1 triggering edges) | (T1 == T2) @ (T2 triggering edges)

Counter Down = (T1 == T2) @ (T1 triggering edges) | (T1 != T2) @ (T2 triggering edges)

Table 11-1 Working Mode of Encoder Orthogonal Coding

Counting Mode	T1/T2 level state (steady state signal)	T1 change edge state		T2 change edge state	
		Rising edge	Falling edge	Rising edge	Falling edge
T1 count only	T2 high level	Decrement	Increment	Not count	Not count
	T2 low level	Increment	Decrement	Not count	Not count
T1/T2 count	T2 high level	Decrement	Increment	Not count	Not count
	T2 low level	Increment	Decrement	Not count	Not count
	T1 high level	Not count	Not count	Increment	Decrement
	T1 low level	Not count	Not count	Decrement	Increment

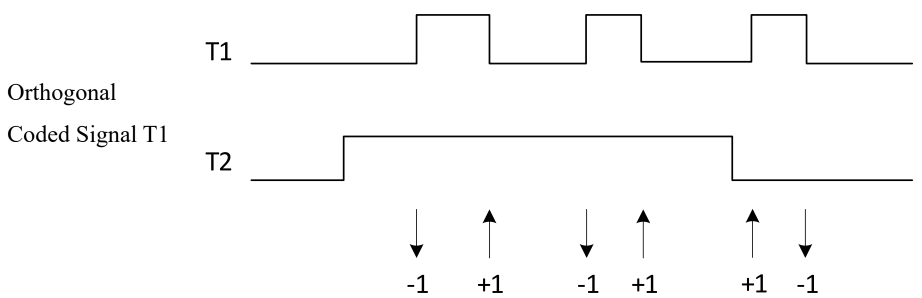


Fig. 11-6 The Counts of Orthogonal Coded Signal when the Encoder Counts Only at Time T1

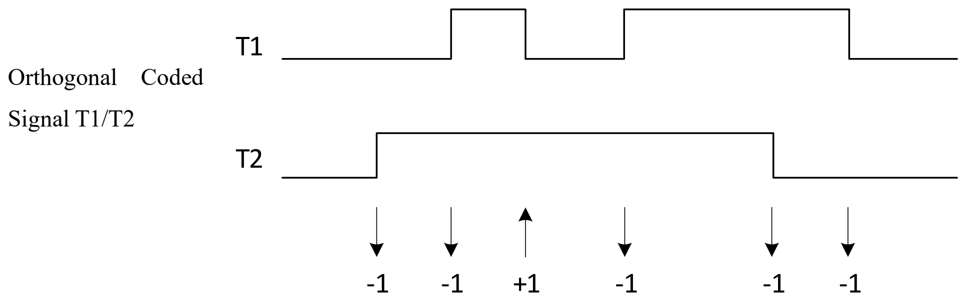


Fig. 11-7 The Counts of Orthogonal Coded Signal when the Encoder Counts at Time T1 or T2

11.2.4.2 Pulse Signal with Symbolic Data Type

In this mode of operation, T1 is a pulse signal, and T2 is a signed signal. The edge of T1 triggers counting and the level of T2 controls the counting direction; if it is a high level, it increases, and if it is a low level, it decreases. It can also be set to count only T1 rising edges or both T1 rising and falling edges.

Counter Up = (T2==1) @ (T1 triggering edges)

Counter Down = (T2==0) @ (T1 triggering edges)

Table 11-2 Working Mode of Pulse Signal with Symbolic Data Type

Counting Mode	T2 level state (steady state signal)	T1 change edge state	
		Rising edge	Falling edge
Only T1 rising edge	High	Increment	Not count
	Low	Decrement	Not count
T1 rising and falling edges	High	Increment	Decrement
	Low	Decrement	Increment

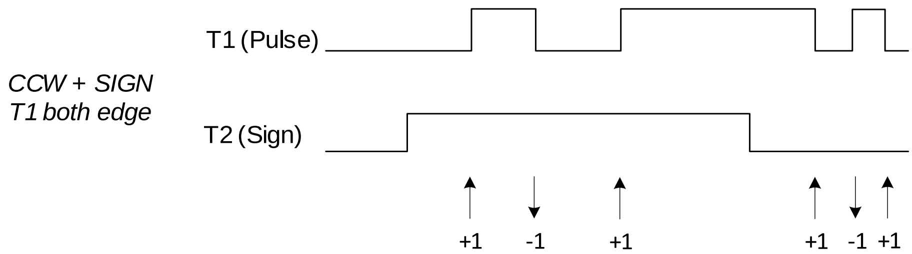


Fig. 11-8 The Counts of Pulse Signal with Symbolic Data Type when the Encoder Counts both on T1 Rising and Falling Edges

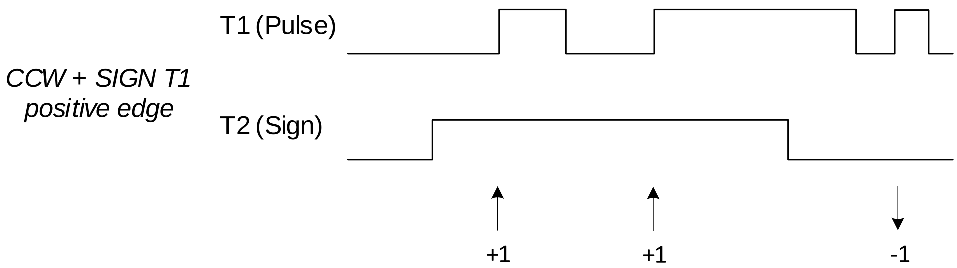


Fig. 11-9 The Counts of Pulse Signal with Symbolic Data Type when the Encoder Counts only on T1 Rising Edges

11.2.4.3 CCW/CW Double Pulse Signal

The counter increments when T1 transitions, and decrements when T2 transitions. It can also be set to count only the rising edges or both the rising and falling edges. The formula is as follows:

Counter Up = 1 @ (T1 triggering edges)

Counter Down = 1 @ (T2 triggering edges)

Table 11-3 Encoder CCW/CW Double Pulse Working Mode

Counting Mode	Changing edge state			
	T1 rising edge	T1 falling edge	T2 rising edge	T2 falling edge
T1/T2 rising edge	Increment	Not count	Decrement	Not count
T1/T2 rising and falling edges	Increment	Increment	Decrement	Decrement

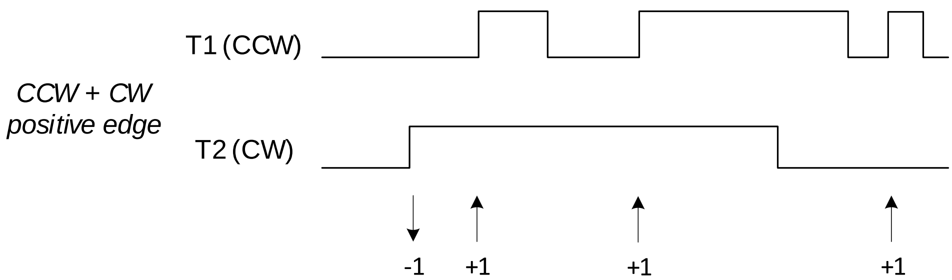


Fig. 11-10 CCW/CW Double Pulse Signal Counting when Encoder Counts only on the T1 and T2 Rising Edge

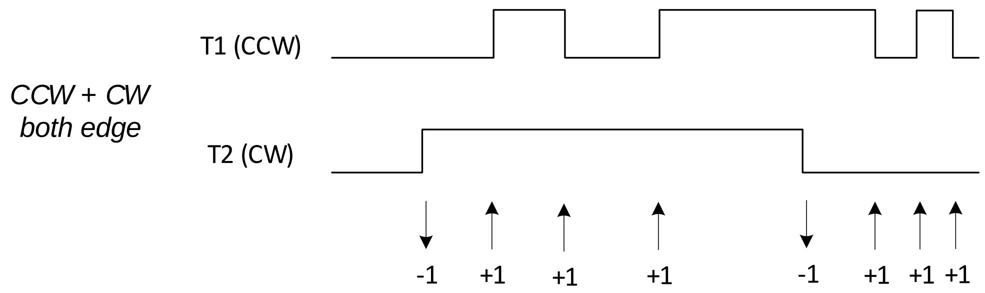


Fig. 11-11 CCW/CW Double Pulse Signal Counting when Encoder Counts both on the Rising and Falling Edge T1 and T2

11.3 Register

11.3.1 Address Allocation

The base address of the universal timer module in the chip is 0x4001_1800, and the register list is as follows:

Table 11-4 Address Allocation of General Timer Register

Name	Offset	Description
UTIMER_UNT0_CFG	0x00	Timer0 configuration register
UTIMER_UNT0_TH	0x04	Timer0 count threshold register
UTIMER_UNT0_CNT	0x08	Timer0 count value register
UTIMER_UNT0_CMP0	0x0C	Timer0 compare/capture register 0
UTIMER_UNT0_CMP1	0x10	Timer0 compare/capture register 1
UTIMER_UNT0_EVT	0x14	Timer0 external event selection register
UTIMER_UNT1_CFG	0x20	Timer1 configuration register
UTIMER_UNT1_TH	0x24	Timer1 count threshold register
UTIMER_UNT1_CNT	0x28	Timer1 count value register
UTIMER_UNT1_CMP0	0x2C	Timer1 compare/capture register 0
UTIMER_UNT1_CMP1	0x30	Timer1 compare/capture register 1
UTIMER_UNT1_EVT	0x34	Timer1 external event selection register
UTIMER_UNT2_CFG	0x40	Timer2 configuration register
UTIMER_UNT2_TH	0x44	Timer2 count threshold register
UTIMER_UNT2_CNT	0x48	Timer2 count value register
UTIMER_UNT2_CMP0	0x4C	Timer2 compare/capture register 0
UTIMER_UNT2_CMP1	0x50	Timer2 compare/capture register 1
UTIMER_UNT2_EVT	0x54	Timer2 external event selection register
UTIMER_UNT3_CFG	0x60	Timer3 configuration register
UTIMER_UNT3_TH	0x64	Timer3 count threshold register
UTIMER_UNT3_CNT	0x68	Timer3 count value register
UTIMER_UNT3_CMP0	0x6C	Timer3 compare/capture register 0
UTIMER_UNT3_CMP1	0x70	Timer3 compare/capture register 1

UTIMER_UNT3_EVT	0x74	Timer3 external event selection register
UTIMER_ECD0_CFG	0x80	Encoder0 configuration register
UTIMER_ECD0_TH	0x84	Encoder0 count threshold register
UTIMER_ECD0_CNT	0x88	Encoder0 count value register
UTIMER_ECD1_CFG	0x90	Encoder1 configuration register
UTIMER_ECD1_TH	0x94	Encoder1 count threshold register
UTIMER_ECD1_CNT	0x98	Encoder1 count value register
UTIMER_FLT_TH01	0xA0	Filter Threshold Register 01
UTIMER_FLT_TH23	0xA4	Filter Threshold Register 23
UTIMER_CFG	0xF0	General timer register
UTIMER_IE	0xF4	Interrupt enable register
UTIMER_IF	0xF8	Interrupt flag register
UTIMER_RE	0xFC	DMA management register

11.3.2 System Control Register

11.3.2.1 UTIMER_CFG

Address: 0x4001_18F0

Reset value: 0x0

Table 11-5 UTIMER Configuration Register (UTIMER_CFG)

15 14 13 12 11 10	9	8	7	6	5	4	3 2 1 0
	ENC1_EN	ENC0_EN	TIMER3_EN	TIMER2_EN	TIMER1_EN	TIMER0_EN	
	RW	RW	RW	RW	RW	RW	
	0	0	0	0	0	0	

Location	Bit name	Description
[31:10]		Unused
[9]	ENC1_EN	1: Turn on the encoder 1, 0: Turn off the encoder 1
[8]	ENC0_EN	1: Turn on the encoder 0, 0: Turn off the encoder 0
[7]	TIMER3_EN	Timer3 enable. When TIMER3_EN is 0, timer3 stops counting, and all interrupt flags output 0 at the same time.
[6]	TIMER2_EN	Timer2 enable. When TIMER2_EN is 0, timer2 stops counting, and all interrupt flags output 0 at the same time.
[5]	TIMER1_EN	Timer1 enable. When TIMER1_EN is 0, timer1 stops counting, and all interrupt flags output 0 at the same time.
[4]	TIMER0_EN	Timer0 enable. When TIMER0_EN is 0, timer0 stops counting, and all interrupt flags output 0 at the same time.
[3:0]		Reserved bit, better write 0

11.3.3 Filter Control Register

11.3.3.1 UTIMER_FLT_TH01

Address: 0x4001_18A0

Reset value: 0x0

Table 11-6 Filter Control Register (UTIMER_FLT_TH01)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T1_CH1_FLT				T1_CH0_FLT				T0_CH1_FLT				T0_CH0_FLT			
RW				RW				RW				RW			
0				0				0				0			

Location	Bit name	Description
[31:16]		Unused
[3:12 PM]	T1_CH1_FLT	TIM1_CH1 signal filter width selection, value range 0 ~ 15. [15:12] When it's 0, TIM1_CH1 is not filtered. [15:12] When it is not 0, the TIM1_CH1 signal is filtered: the filter width is T1_CH1_FLT×8. When the TIM1_CH1 level is stable beyond the width of T1_CH1_FLT×8 system clock cycles, the filter output is updated to the TIM1_CH1 signal value; otherwise, the filter keeps the current output unchanged.
[11:8]	T1_CH0_FLT	TIM1_CH0 signal filter width selection. The meaning is the same as T1_CH1_FLT.
[7:4]	T0_CH1_FLT	TIM0_CH1 signal filter width selection. The meaning is the same as T1_CH1_FLT.
[3:0]	T0_CH0_FLT	TIM0_CH0 signal filter width selection. The meaning is the same as T1_CH1_FLT.

11.3.3.2 UTIMER_FLT_TH23

Address: 0x4001_18A4

Reset value: 0x0

Table 11-7 Filter Control Register (UTIMER_FLT_TH23)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T3_CH1_FLT				T3_CH0_FLT				T2_CH1_FLT				T2_CH0_FLT			
RW				RW				RW				RW			
0				0				0				0			

Location	Bit name	Description

[31:16]		Unused
[3:12 PM]	T3_CH1_FLT	TIM3_CH1 signal filter width selection, value range 0 ~ 15. [15:12] When it is 0, TIM3_CH1 is not filtered. [15:12] When it is not 0, the TIM3_CH1 signal is filtered: the filter width is T3_CH1_FLT×8. When the TIM3_CH1 level is stable beyond the width of T3_CH1_FLT×8 system clock cycles, the filter output is updated to the TIM3_CH1 signal value; otherwise, the filter keeps the current output unchanged.
[11:8]	T3_CH0_FLT	TIM3_CH0 signal filter width selection. The meaning is the same as T3_CH1_FLT.
[7:4]	T2_CH1_FLT	TIM2_CH1 signal filter width selection. The meaning is the same as T3_CH1_FLT.
[3:0]	T2_CH0_FLT	TIM2_CH0 signal filter width selection. The meaning is the same as T3_CH1_FLT.

11.3.4 Timer Register

Timer0 and Time1 are the same. Here the example given is Timer0 register.

Timer2 and Timer3 are the same. The difference with Timer0 and Timer1 is that the related registers of Timer2 and Timer3 counter are 32 bits wide, while the related registers of Timer0 and Timer1 counter are 16 bits wide.

Encoder0 multiplexes the input port of Timer2, and Encoder1 multiplexes the input port of Timer3; when the Encoder function is enabled, it won't affect the use of the corresponding Timer.

11.3.4.1 Timer0 Configuration Register (UTIMER_UNT0_CFG)

Address: 0x4001_1800

Reset value: 0x0

Table 11-8 Timer 0 Configuration Register (UTIMER_UNT0_CFG)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				ETON		CLK_DIV		CH1_POL	CH1_MODE	CH1_FE_CAP_EN	CH1_RE_CAP_EN	CH0_POL	CH0_MODE	CH0_FE_CAP_EN	CH0_RE_CAP_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
[31:12]		Unused
[11]	ETON	Timer0 counter count enable source. 0: Auto run.

		1: Wait for external event to trigger counting and stops after one period; The default value is 0. External event could be selected by setting UTIMER_UNT0_EVT.
[10]		Reserved bit. Always read as 0.
[9:8]	CLK_DIV	Timer0 counter frequency configuration. The counting frequency of the counter is divided by 1/2/4/8 of the main clock frequency: 00: 1 frequency division, 01: 2 frequency division, 10: 4 frequency division, 11: 8 frequency division
[7]	CH1_POL	Channel 1 output polarity control in compare mode: the output value when the counter count value returns to zero.
[6]	CH1_MODE	Channel 1 working mode: 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.
[5]	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.
[4]	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.
[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	Channel 0 working mode: 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.

11.3.4.2 Timer1 Configuration Register (UTIMER_UNT1_CFG)

Address: 0x4001_1820

Reset value: 0x0

Table 11-9 Timer 0 Configuration Register (UTIMER_UNT0_CFG)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				ETON		CLK_DIV		CH1_POL	CH1_MODE	CH1_FE_CAP_EN	CH1_RE_CAP_EN	CH0_POL	CH0_MODE	CH0_FE_CAP_EN	CH0_RE_CAP_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
[31:12]		Unused
[11]	ETON	Timer1 counter count enable source. 0: Auto run. 1: Wait for external event to trigger counting and stops after one period; The default value is 0. External event could be selected by setting UTIMER_UNT1_EVT.
[10]		Reserved bit. Always read as 0.
[9:8]	CLK_DIV	Timer1 counter frequency configuration. The counting frequency of the counter is divided by 1/2/4/8 of the main clock frequency: 00: 1 frequency division, 01: 2 frequency division, 10: 4 frequency division, 11: 8 frequency division
[7]	CH1_POL	Channel 1 output polarity control in compare mode: the output value when the counter count value returns to zero.
[6]	CH1_MODE	Channel 1 working mode: 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.
[5]	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.
[4]	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.
[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	Channel 0 working mode: 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.

11.3.4.3 Timer2 Configuration Register (UTIMER_UNT2_CFG)

Address: 0x4001_1840

Reset value: 0x0

Table 11-10 Timer 2 Configuration Register (UTIMER_UNT2_CFG)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				ETON		CLK_DIV		CH1_POL	CH1_MODE	CH1_FE_CAP_EN	CH1_RE_CAP_EN	CH0_POL	CH0_MODE	CH0_FE_CAP_EN	CH0_RE_CAP_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
[31:12]		Unused
[11]	ETON	Timer2 counter count enable source. 0: Auto run. 1: Wait for external event to trigger counting and stops after one period; The default value is 0. External event could be selected by setting UTIMER_UNT2_EVT.
[10]		Reserved bit. Always read as 0.

[9:8]	CLK_DIV	Timer2 counter frequency configuration. The counting frequency of the counter is divided by 1/2/4/8 of the main clock frequency: 00: 1 frequency division, 01: 2 frequency division, 10: 4 frequency division, 11: 8 frequency division
[7]	CH1_POL	Channel 1 output polarity control in compare mode: the output value when the counter count value returns to zero.
[6]	CH1_MODE	Channel 1 working mode: 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.
[5]	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.
[4]	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.
[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	Channel 0 working mode: 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.

11.3.4.4 Timer3 Configuration Register (UTIMER_UNT3_CFG)

Address: 0x4001_1860

Reset value: 0x0

Table 11-11 Timer 3 Configuration Register (UTIMER_UNT3_CFG)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				ETON		CLK_DIV		CH1_POL	CH1_MODE	CH1_FE_CAP_EN	CH1_RE_CAP_EN	CH0_POL	CH0_MODE	CH0_FE_CAP_EN	CH0_RE_CAP_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
[31:12]		Unused
[11]	ETON	Timer3 counter count enable source. 0: Auto run. 1: Wait for external event to trigger counting and stops after one period; The default value is 0. External event could be selected by setting UTIMER_UNT3_EVT.
[10]		Reserved bit. Always read as 0.
[9:8]	CLK_DIV	Timer3 counter frequency configuration. The counting frequency of the counter is divided by 1/2/4/8 of the main clock frequency: 00: 1 frequency division, 01: 2 frequency division, 10: 4 frequency division, 11: 8 frequency division
[7]	CH1_POL	Channel 1 output polarity control in compare mode: the output value when the counter count value returns to zero.
[6]	CH1_MODE	Channel 1 working mode: 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.
[5]	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.
[4]	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.
[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	Channel 0 working mode: 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.

Among the LKS32MC08X series, Timer0/1 is 16bit; Timer2/3 is 32bit, so the following related registers of Timer2/3 are 32bit.

11.3.4.5 Timer 0 Threshold Register (UTIMER_UNT0_TH)

Address: 0x4001_1804

Reset value: 0x0

Table 11-12 Timer 0 Threshold Register (UTIMER_UNT0_TH)

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">UNT0_TH</td></tr>
<tr><td colspan="16">RW</td></tr>
<tr><td colspan="16">0</td></tr>
</table>

Location	Bit name	Description
[31:16]		Unused
[15:0]	UNT0_TH	Timer 0 counter count threshold. The counter counts from 0 to UTIMER_UNT0_TH, and then returns to 0 to start counting.

11.3.4.6 Timer 1 Threshold Register (UTIMER_UNT1_TH)

Address: 0x4001_1824

Reset value: 0x0

Table 11-13 Timer 1 Threshold Register (UTIMER_UNT1_TH)

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">UNT1_TH</td></tr>
<tr><td colspan="16">RW</td></tr>
</table>

0

Location	Bit name	Description
[31:16]		Unused
[15:0]	UNT1_TH	Timer 1 counter count threshold. The counter counts from 0 to UTIMER_UNT1_TH, and then returns to 0 to start counting.

11.3.4.7 Timer 2 Threshold Register (UTIMER_UNT2_TH)

Address: 0x4001_1844

Reset value: 0x0

Table 11-14 Timer 2 Threshold Register (UTIMER_UNT2_TH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UNT2_TH															
RW															
0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNT2_TH															
RW															
0															

Location	Bit name	Description
[31:0]	UNT2_TH	Timer 2 counter count threshold. The counter counts from 0 to UTIMER_UNT2_TH, and then returns to 0 to start counting.

11.3.4.8 Timer 3 Threshold Register (UTIMER_UNT3_TH)

Address: 0x4001_1864

Reset value: 0x0

Table 11-15 Timer 3 Threshold Register (UTIMER_UNT3_TH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UNT3_TH															
RW															
0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNT3_TH															
RW															
0															

Location	Bit name	Description
[31:0]	UNT3_TH	Timer 3 counter count threshold. The counter counts from 0 to UTIMER_UNT3_TH, and then returns to 0 to start counting.

11.3.4.9 Timer 0 Count Register (UTIMER_UNT0_CNT)

Address: 0x4001_1808

Reset value: 0x0

Table 11-16 Timer 0 Count Register (UTIMER_UNT0_CNT)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNT0_CNT															
RW															
0															

Location	Bit name	Description
[31:16]		Unused
[15:0]	UNT0_CNT	The current count value of the Timer 0 counter. New count value can be written.

11.3.4.10 Timer 1Count Register (UTIMER_UNT1_CNT)

Address: 0x4001_1828

Reset value: 0x0

Table 11-17 Timer 1 Count Register (UTIMER_UNT1_CNT)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNT1_CNT															
RW															
0															

Location	Bit name	Description
[31:16]		Unused
[15:0]	UNT1_CNT	The current count value of the Timer 1 counter. New count value can be written.

11.3.4.11 Timer 2 Counter Register (UTIMER_UNT2_CNT)

Address: 0x4001_1848

Reset value: 0x0

Table 11-18 Timer 2 Count Register (UTIMER_UNT2_CNT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UNT2_CNT															
RW															
0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNT2_CNT															
RW															
0															

Location	Bit name	Description
[31:0]	UNT2_CNT	The current count value of the Timer 2 counter. New count value can be written.

11.3.4.12 Timer 3 Count Register (UTIMER_UNT3_CNT)

Address: 0x4001_1868

Reset value: 0x0

Table 11-19 Timer 3 Count Register (UTIMER_UNT3_CNT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UNT3_CNT															
RW															
0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNT3_CNT															
RW															
0															

Location	Bit name	Description
[31:0]	UNT3_CNT	The current count value of the Timer3 counter. New count value can be written.

11.3.4.13 Timer 0 Channel 0 Compare Capture Register (UTIMER_UNT0_CMP0)

Address: 0x4001_180C

Reset value: 0x0

Table 11-20 Timer 0 Channel 0 Compare Capture Register (UTIMER_UNT0_CMP0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNT0_ CMP0															
RW															
0															

Location	Bit name	Description
[31:16]		Unused
[15:0]	UNT0_ CMP0	When Timer 0 channel 0 works in compare mode and the counter count value is equal to UTIMER_UNT0_CMP0, a comparison event occurs. When Timer 0 channel 0 works in the capture mode, the counter count value when the capture event occurs is stored in the UTIMER_UNT0_CMP0 register.

11.3.4.14 Timer 0 Channel 1 Compare Capture Register (UTIMER_UNT0_CMP1)

Address: 0x4001_1810

Reset value: 0x0

Table 11-21 Timer 0 Channel 1 Compare Capture Register (UTIMER_UNT0_CMP1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNT0_CMP1															
RW															
0															

Location	Bit name	Description
[31:16]		Unused
[15:0]	UNT0_CMP1	When Timer 0 channel 1 works in compare mode and the counter count value is equal to UTIMER_UNT0_CMP1, a comparison event occurs. When Timer 0 channel 1 works in the capture mode, the counter count value when the capture event occurs is stored in the UTIMER_UNT0_CMP1 register.

11.3.4.15 Timer 1 Channel 0 Compare Capture Register (UTIMER_UNT1_CMP0)

Address: 0x4001_182C

Reset value: 0x0

Table 11-22 Timer 1 Channel 0 Compare Capture Register (UTIMER_UNT1_CMP0)

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

UNT1_ CMP0
RW
0

Location	Bit name	Description
[31:16]		Unused
[15:0]	UNT1_ CMP0	When Timer 1 channel 0 works in compare mode and the counter count value is equal to UTIMER_UNT1_CMP0, a comparison event occurs. When Timer 1 channel 0 works in the capture mode, the counter count value when the capture event occurs is stored in the UTIMER_UNT1_ CMP0 register.

11.3.4.16 Timer 1 Channel 0 Compare Capture Register (UTIMER_UNT1_CMP1)

Address: 0x4001_1830

Reset value: 0x0

Table 11-23 Timer 1 Channel 1 Compare Capture Register (UTIMER_UNT1_CMP1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNT1_CMP1															
RW															
0															

Location	Bit name	Description
[31:16]		Unused
[15:0]	UNT1_CMP1	When Timer 1 channel 1 works in compare mode and the counter count value is equal to UTIMER_UNT1_CMP1, a comparison event occurs. When Timer 1 channel 1 works in the capture mode, the counter count value when the capture event occurs is stored in the UTIMER_UNT1_CMP1 register.

11.3.4.17 Timer 2 Channel 0 Compare Capture Register (UTIMER_UNT2_CMP0)

Address: 0x4001_184C

Reset value: 0x0

Table 11-24 Timer 2 Channel 0 Compare Capture Register (UTIMER_UNT2_CMP0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UNT2_ CMP0															
RW															

0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNT2_ CMP0															
RW															
0															

Location	Bit name	Description
[31:0]	UNT2_ CMP0	When Timer 2 channel 0 works in compare mode and the counter count value is equal to UTIMER_UNT2_CMP0, a comparison event occurs. When Timer 2 channel 0 works in the capture mode, the counter count value when the capture event occurs is stored in the 入 UTIMER_UNT2_ CMP0 register.

11.3.4.18 Timer 2 Channel 1 Compare Capture Register (UTIMER_UNT2_CMP1)

Address: 0x4001_1850

Reset value: 0x0

Table 11-25 Timer 2 Channel 1 Compare Capture Register (UTIMER_UNT2_CMP1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UNT2_CMP1															
RW															
0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNT2_CMP1															
RW															
0															

Location	Bit name	Description
[31:0]	UNT2_CMP1	When Timer 2 channel 1 works in compare mode and the counter count value is equal to UTIMER_UNT2_CMP1, a comparison event occurs. When Timer 2 channel 1 works in the capture mode, the counter count value when the capture event occurs is stored in the UTIMER_UNT2_CMP1 register.

11.3.4.19 Timer 3 Channel 0 Compare Capture Register (UTIMER_UNT3_CMP0)

Address: 0x4001_186C

Reset value: 0x0

Table 11-26 Timer 3 Channel 0 Compare Capture Register (UTIMER_UNT3_CMP0)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UNT3_ CMP0															
RW															
0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNT3_ CMP0															
RW															
0															

Location	Bit name	Description
[31:0]	UNT3_ CMP0	When Timer 3 channel 0 works in compare mode and the counter count value is equal to UTIMER_UNT3_CMP0, a comparison event occurs. When Timer 3 channel 0 works in the capture mode, the counter count value when the capture event occurs is stored in the UTIMER_UNT3_ CMP0 register.

11.3.4.20 Timer 3 Channel 1 Compare Capture Register (UTIMER_UNT3_CMP1)

Address: 0x4001_1870

Reset value: 0x0

Table 11-27 Timer 3 Channel 1 Compare Capture Register (UTIMER_UNT3_CMP1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UNT3_CMP1															
RW															
0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNT3_CMP1															
RW															
0															

Location	Bit name	Description
[31:0]	UNT3_CMP1	When Timer 3 channel 1 works in compare mode and the counter count value is equal to UTIMER_UNT3_CMP1, a comparison event occurs. When Timer 3 channel 1 works in the capture mode, the counter count value when the capture event occurs is stored in the UTIMER_UNT3_CMP1 register.

11.3.4.21 Timer0 External Event Select Register (UTIMER_UNT0_EVT)

Address: 0x4001_1814

Reset value: 0x0

Table 11-28 Timer 0 External Event Select Register (UTIMER_UNT0_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	Timer0 external event selection register. This register should be used with UTIMER_UNT0_CFG [11]. When U0CFG [11] is high, select the event that triggers Timer0 count by this register. 0: MCPWM TADC [2] comparison event 1: MCPWM TADC [3] comparison event 2: TIMER1 channel 0 comparison event 3: TIMER1 channel 1 comparison event 4: TIMER2 channel 0 comparison event 5: TIMER2 channel 1 comparison event 6: TIMER3 channel 0 comparison event 7: TIMER3 channel 1 comparison event

11.3.4.22 Timer1 External Event Select Register (UTIMER_UNT1_EVT)

Address: 0x4001_1834

Reset value: 0x0

Table 11-29 Timer 1 External Event Select Register (UTIMER_UNT1_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	Timer1 external event selection register. This register should be used with UTIMER_UNT1_CFG[11]. UTIMER_UNT1_CFG[11] is high, select the event that triggers Timer1 count by this register.

		0: TIMER0 channel 0 comparison event 1: TIMER0 channel 1 comparison event 2: MCPWM TADC [2] comparison event 3: MCPWM TADC [3] comparison event 4: TIMER2 channel 0 comparison event 5: TIMER2 channel 1 comparison event 6: TIMER3 channel 0 comparison event 7: TIMER3 channel 1 comparison event

11.3.4.23 Timer2 External Event Select Register (UTIMER_UNT2_EVT)

Address: 0x4001_1854

Reset value: 0x0

Table 11-30 Timer 2 External Event Select Register (UTIMER_UNT2_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	Timer2 external event selection register. This register should be used with UTIMER_UNT2_CFG[11]. When UTIMER_UNT2_CFG[11] is high, select the event that triggers Timer2 count by this register. 0: TIMER0 channel 0 comparison event 1: TIMER0 channel 1 comparison event 2: TIMER1 channel 0 comparison event 3: TIMER1 channel 1 comparison event 4: MCPWM TADC [2] comparison event 5: MCPWM TADC [3] comparison event 6: TIMER3 channel 0 comparison event 7: TIMER3 channel 1 comparison event

11.3.4.24 Timer3 External Event Select Register (UTIMER_UNT3_EVT)

Address: 0x4001_1874

Reset value: 0x0

Table 11-31 Timer 3 External Event Select Register (UTIMER_UNT3_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	Timer3 external event selection register. This register should be used with UTIMER_UNT3_CFG[11]. When UTIMER_UNT3_CFG[11] is high, select the event that triggers Timer3 count by this register. 0: TIMER0 channel 0 comparison event 1: TIMER0 channel 1 comparison event 2: TIMER1 channel 0 comparison event 3: TIMER1 channel 1 comparison event 4: TIMER2 channel 0 comparison event 5: TIMER2 channel 1 comparison event 6: MCPWM TADC [2] comparison event 7: MCPWM TADC [3] comparison event

11.3.5 Encoder Register

Encoder0 and Encoder1 have the same function, only the register address allocation is different.

11.3.5.1 EncoderX Configuration Register (UTIMER_ECDx_CFG)

UTIMER_ECD0_CFG address: 0x4001_1880

UTIMER_ECD1_CFG address: 0x4001_1890

Reset value: 0x0

Table 11-32 EncoderX Configuration Register (UTIMER_ECDx_CFG)

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

Location	Bit name	Description
[31:11]		Unused
[10]	FE_CNT_EN	Whether to count on the falling edge (rising edge is always counted) in CCW+SIGN and CCW+CW mode
[9:8]	MODE	EncoderX encoder mode selection 00: counting on T1 01: counting on T1 & T2 The above are the counting modes of the orthogonal code signal

		10: CCW+SIGN, pulse signal counting with symbolic data 11: CCW+CW, CCW+CW double pulse signal counting
[7:0]		Unused

11.3.5.2 EncoderX Count Threshold Register (UTIMER_ECDx_TH)

Address of UTIMER_ECD0_TH: 0x4001_1884

Address of UTIMER_ECD1_TH: 0x4001_1894

Reset value: 0x0

Table 11-33 EncoderX Count Threshold Register (UTIMER_ECDx_TH)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECD0_TH															
RW															
0															

Location	Bit name	Description
[31:16]		Unused
[15:0]	ECDx_TH	EncoderX count threshold. After the encoder counts up (increases) to the threshold value, counting up again will cause the counter to return to zero. After the encoder counts down (decreases) to zero, counting down again will cause the counter to return to the threshold value.

11.3.5.3 EncoderX Count Value Register (UTIMER_ECDx_CNT)

UTIMER_ECD0_CNT address: 0x4001_1888

UTIMER_ECD1_CNT address: 0x4001_1898

Reset value: 0x0

Table 11-34 EncoderX Count Value Register (UTIMER_ECDx_CNT)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECD0_CNT															
RO															
0															

Location	Bit name	Description
[31:16]		Unused
[15:0]	ECDx_CNT	EncoderX count value.

11.3.6 Interrupt Management Register

The interrupt management register includes the interrupt flag register UTIMER_IF and the interrupt enable register UTIMER_IE. Each bit of the two registers corresponds to the same interrupt.

11.3.6.1 Interrupt Enable Register (UTIMER_IE)

Address: 0x4001_18F4

Reset value: 0x0

Table 11-35 Interrupt Enable Register (UTIMER_IE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENC1_OF_IE	ENC1_UF_IE	ENC0_OF_IE	ENC0_UF_IE	T3_CH1_IE	T3_CH0_IE	T3_ZC_IE	T2_CH1_IE	T2_CH0_IE	T2_ZC_IE	T1_CH1_IE	T1_CH0_IE	T1_ZC_IE	T0_CH1_IE	T0_CH0_IE	T0_ZC_IE
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]	ENC1_OF_IE	Encoder1 overflow interrupt enable. Active high. When the Encoder1 counter reaches the count threshold, the up-count event triggers an overflow interrupt.
[14]	ENC1_UF_IE	Encoder1 underflow interrupt enable. Active high. When the Encoder1 counter reaches zero, the down-count event triggers an overflow interrupt.
[13]	ENC0_OF_IE	Encoder0 overflow interrupt enable. Active high.
[12]	ENC0_UF_IE	Encoder0 underflow interrupt enable. Active high.
[11]	T3_CH1_IE	Timer3 channel 1 compare/capture interrupt enable. Active high.
[10]	T3_CH0_IE	Timer3 channel 0 compare/capture interrupt enable. Active high.
[9]	T3_ZC_IE	Timer3 counter over-zero interrupt enable. Active high.
[8]	T2_CH1_IE	Timer2 channel 1 compare/capture interrupt enable. Active high.
[7]	T2_CH0_IE	Timer2 channel 0 compare/capture interrupt enable. Active high.
[6]	T2_ZC_IE	Timer2 counter over-zero interrupt enable. Active high.
[5]	T1_CH1_IE	Timer1 channel 1 compare/capture interrupt enable. Active high.
[4]	T1_CH0_IE	Timer1 channel 0 compare/capture interrupt enable. Active high.
[3]	T1_ZC_IE	Timer1 counter over-zero interrupt enable. Active high.
[2]	T0_CH1_IE	Timer0 channel 1 compare/capture interrupt enable. Active high.
[1]	T0_CH0_IE	Timer0 channel 0 compare/capture interrupt enable. Active high.
[0]	T0_ZC_IE	Timer0 counter over-zero interrupt enable. Active high.

11.3.6.2 Interrupt Flag Register (UTIMER_IF)

Address: 0x4001_18F8

Reset value: 0x0

Table 11-36 Interrupt Flag Register (UTIMER_IF)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENC1_OF_IF	ENC1_UF_IF	ENC0_OF_IF	ENC0_UF_IF	T3_CH1_IF	T3_CH0_IF	T3_ZC_IF	T2_CH1_IF	T2_CH0_IF	T2_ZC_IF	T1_CH1_IF	T1_CH0_IF	T1_ZC_IF	T0_CH1_IF	T0_CH0_IF	T0_ZC_IF
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]	ENC1_OF_IF	Encoder1 overflow interrupt flag. Active high. Write 1 to clear this bit. When the Encoder1 counter reaches the count threshold, the up-count event triggers an overflow interrupt.
[14]	ENC1_UF_IF	Encoder1 underflow interrupt flag. Active high. Write 1 to clear this bit. When the Encoder1 counter reaches zero, the down-count event triggers an overflow interrupt.
[13]	ENC0_OF_IF	Encoder0 overflow interrupt flag. Active high. Write 1 to clear this bit. When the Encoder0 counter reaches the count threshold, the up-count event triggers an overflow interrupt.
[12]	ENC0_UF_IF	Encoder0 underflow interrupt flag. Active high. Write 1 to clear this bit. When the Encoder0 counter reaches zero, the down-count event triggers an overflow interrupt.
[11]	T3_CH1_IF	Timer3 channel 1 compare/capture interrupt flag. Active high. Write 1 to clear this bit.
[10]	T3_CH0_IF	Timer3 channel 0 compare/capture interrupt flag. Active high. Write 1 to clear this bit.
[9]	T3_ZC_IF	Timer3 counter over-zero interrupt flag. Active high. Write 1 to clear this bit.
[8]	T2_CH1_IF	Timer2 channel 1 compare/capture interrupt flag. Active high. Write 1 to clear this bit.
[7]	T2_CH0_IF	Timer2 channel 0 compare/capture interrupt flag. Active high. Write 1 to clear this bit.
[6]	T2_ZC_IF	Timer2 counter over-zero interrupt flag. Active high. Write 1 to clear this bit.

[5]	T1_CH1_IF	Timer1 channel 1 compare/capture interrupt flag. Active high. Write 1 to clear this bit.
[4]	T1_CH0_IF	Timer1 channel 0 compare/capture interrupt flag. Active high. Write 1 to clear this bit.
[3]	T1_ZC_IF	Timer1 counter over-zero interrupt flag. Active high. Write 1 to clear this bit.
[2]	T0_CH1_IF	Timer0 channel 1 compare/capture interrupt flag. Active high. Write 1 to clear this bit.
[1]	T0_CH0_IF	Timer0 channel 0 compare/capture interrupt flag. Active high. Write 1 to clear this bit.
[0]	T0_ZC_IF	Timer0 counter over-zero interrupt flag. Active high. Write 1 to clear this bit.

The comparison event of Timer2 channel 0/1 and Timer3 channel 0/1 can be used as ADC sampling trigger event UTimer_T0, UTimer_T1, UTimer_T2, and UTimer_T3;

MCPWM_T0, MCPWM_T1, MCPWM_T2, and MCPWM_T3 generated by MCPWM are enabled controlled within the ADC, along with these four events, and obtained four ADC sampling trigger events.

11.3.7 DMA Management Register

The interrupt management register includes the interrupt flag register IF and the interrupt enable register IE. Each bit of the two registers corresponds to the same interrupt.

11.3.7.1 DMA Request Enable Register (UTIMER_RE)

Address: 0x4001_18FC

Reset value: 0x0

Table 11-37 DMA Request Enable Register (UTIMER_RE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				T3_CH1_DMA_RE	T3_CH0_DMA_RE	T3_DMA_RE	T2_CH1_DMA_RE	T2_CH0_DMA_RE	T2_DMA_RE	T1_CH1_DMA_RE	T1_CH0_DMA_RE	T1_DMA_RE	T0_CH1_DMA_RE	T0_CH0_DMA_RE	T0_DMA_RE
				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]	T3_CH1_DMA_RE	Timer3 channel 1 compare/capture DMA request enable. Active high.

[10]	T3_CH0_DMA_RE	Timer3 channel 0 compare/capture DMA request enable. Active high.
[9]	T3_DMA_RE	Timer3 counter over-zero DMA request enable. Active high.
[8]	T2_CH1_DMA_RE	Timer2 channel 1 compare/capture DMA request enable. Active high.
[7]	T2_CH0_DMA_RE	Timer2 channel 0 compare/capture DMA request enable. Active high.
[6]	T2_DMA_RE	Timer2 counter over-zero DMA request enable. Active high.
[5]	T1_CH1_DMA_RE	Timer1 channel 1 compare/capture DMA request enable. Active high.
[4]	T1_CH0_DMA_RE	Timer1 channel 0 compare/capture DMA request enable. Active high.
[3]	T1_DMA_RE	Timer1 counter over-zero DMA request enable. Active high.
[2]	T0_CH1_DMA_RE	Timer0 channel 1 compare/capture DMA request enable. Active high.
[1]	T0_CH0_DMA_RE	Timer0 channel 0 compare/capture DMA request enable. Active high.
[0]	T0_DMA_RE	Timer0 counter over-zero DMA request enable. Active high.

12 HALL Signal Processing Module

12.1 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.

12.2 Implementation Description

12.2.1 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.

12.2.2 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.

12.2.3 Signal Filtering

The filter module is mainly used to remove glitches on the HALL signal.

The filtering includes two stages of filters:

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. The details are shown below:

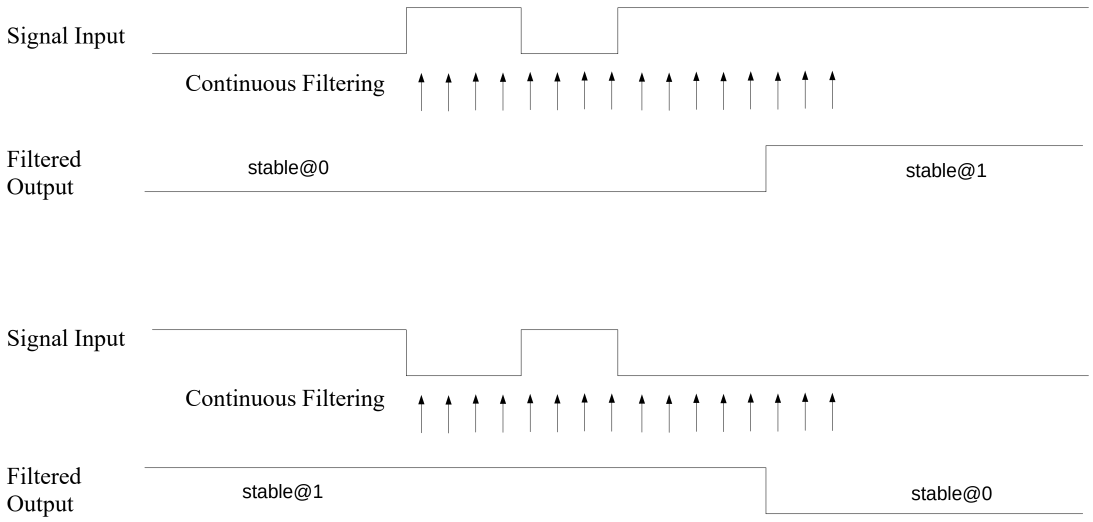


Fig. 12-1 7/5 Filter Module Block Diagram

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 whether to enable the first-stage filter by setting HALL_CFG.FIL_75.

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^{15}, and the calculation formula of the filter time constant is as follows:

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

For example, at a 96MHz operating frequency, the period T_{clk} is 10.4ns, the maximum register configuration is 32767, and the longest filter width is about 10.4ns × 32768≈340us.

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 12.3.3 for details.

12.2.4 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 can record a maximum time width of about 1.39 seconds at a 96MHz operating frequency, and achieve a time resolution of 10ns.

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.

12.2.5 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.

12.2.6 Data Flow

The data flow of the HALL module is shown in the figure below. FCLK [1] is the main clock controlled by the SYS_CLK_FEN gate control, which is usually a 96MHz PLL clock.

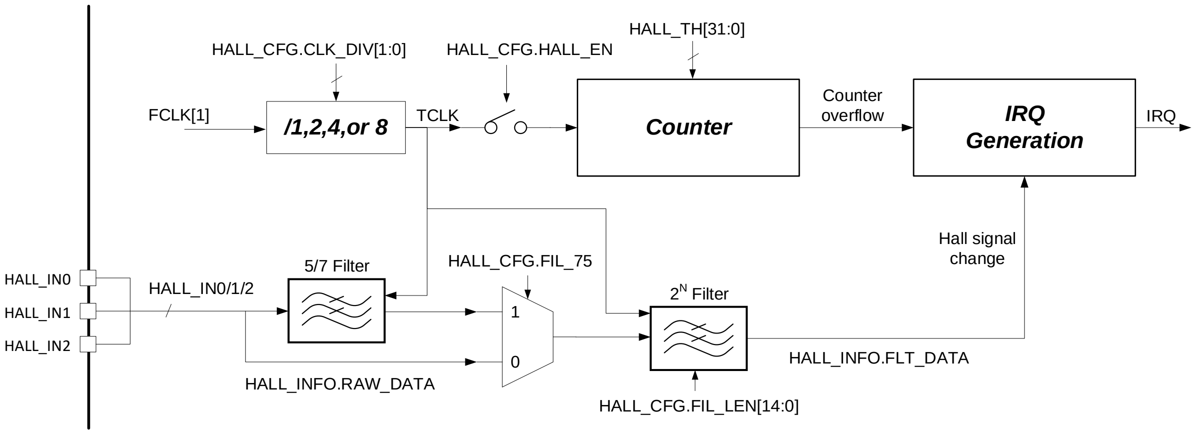


Fig. 12-2 Data Flow Diagram

12.3 Register

12.3.1 Address Allocation

The base address of the HALL module register is 0x4001_1000, and the register list is as follows:

Table 12-1 HALL 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

12.3.2 HALL Module Configuration Register (HALL_CFG)

Address: 0x4001_1000

Reset value: 0x0

Table 12-2 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				HALL_EN				FIL_75			CLK_DIV	
	RW	RW	RW				RW				RW			RW	
	0	0	0				0				0			0	

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

Location	Bit name	Description
[31]		Unused
[30]	SW_IE	Software-triggered HALL signal change interrupt enable. Active high. After this bit is valid, writing 1 to INFO [18], a HALL signal change interrupt will be generated manually.
[29]	OV_IE	HALL counter overflow interrupt enable switch. Off by default. 1: Enable 0: Disable
[28]	CHG_IE	HALL signal change interrupt enable switch. Off by default. 1: Enable 0: Disable
[27:25]		Unused
[24]	HALL_EN	HALL module enable switch. Off by default. 1: Enable 0: Disable
[11:21 PM]		Unused
[20]	FIL_75	7/5 filter switch (sequential sampling for seven times, and five results should be the same). Off by default. 1: Enable 0: Disable
[7:18 PM]		Unused
[5:16 PM]	CLK_DIV	HALL clock division factor 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.

12.3.3 HALL Module Information Register (HALL_INFO)

Address: 0x4001_1004

Reset value: 0x0

Table 12-3 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
													RW	RW	RW
													0	0	0

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

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
[3:11 PM]		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. Flter result

12.3.4 HALL Width Count Value Register (HALL_WIDTH)

Address: 0x4001_1008

Reset value: 0x0

Table 12-4 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							
								0							

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

0

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

12.3.5 HALL Module Counter Threshold Register (HALL_TH)

Address: 0x4001_100C

Reset value: 0x0

Table 12-5 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

12.3.6 HALL Count Register (HALL_CNT)

Address: 0x4001_1010

Reset value: 0x0

Table 12-6 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

13 MCPWM

13.1 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 a basic period. The counter has four clock frequency divisions, 1-, 2-, 4-, and 8-divided frequency division, and the divided clock frequencies generated are 96MHz, 48MHz, 24MHz and 12MHz, respectively.

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).

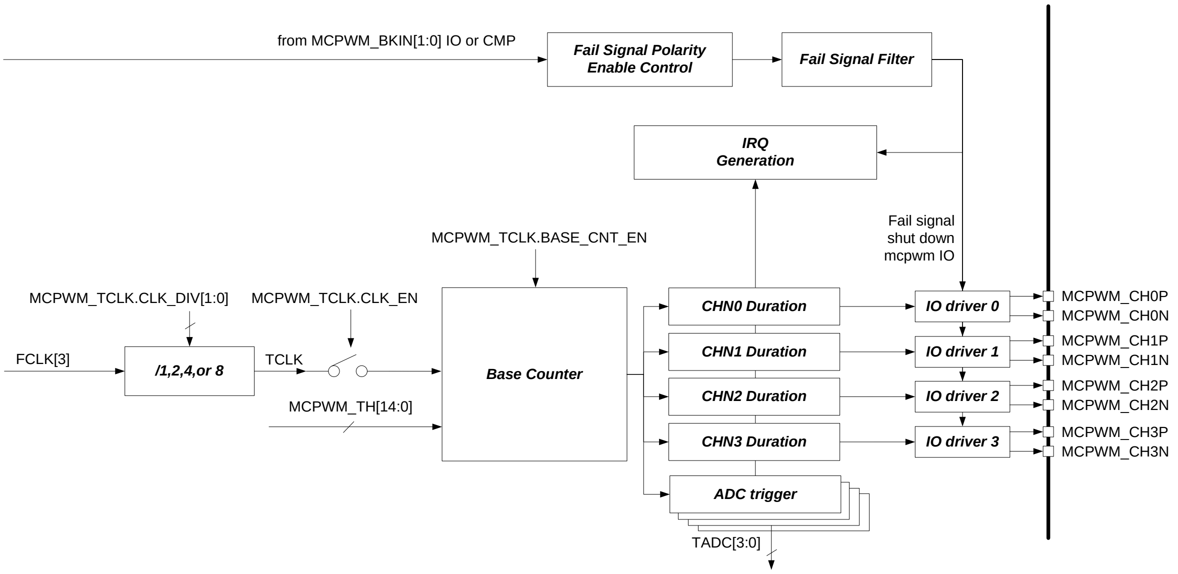


Fig. 13-1 MCPWM Module Block Diagram

Usually, a 96MHz clock is used as the operating frequency of the MCPWM module to ensure the timing accuracy.

13.1.1 Base Counter Module

The module is mainly composed of an count-up counter, and its count threshold is MCPWM_TH. The counter starts at time t0, and counts up from -TH. It passes the time zero at time t1, counts at time t2 to TH to complete a counting cycle, and then returns to -TH to restart counting. The count period is (TH×2+1) times the count clock period.

Timed event interrupt can be generated at t0/t1 (current time t0 is the previous t2), MCPWM_IF.T0_IF and MCPWM_IF.T1_IF will be set.

The start and stop of the Base Counter can be controlled by register configuration MCPWM_TCLK.BASE_CNT_EN.

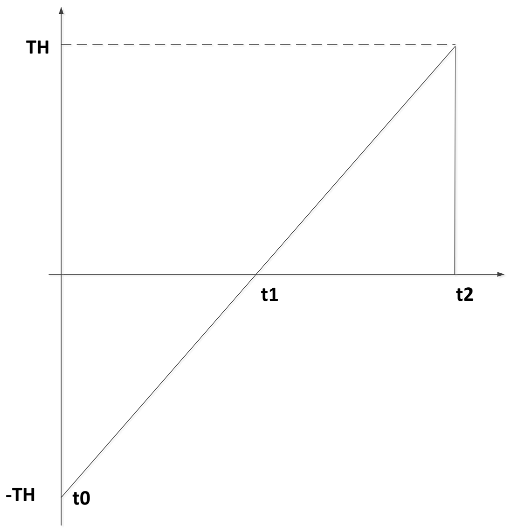


Fig. 13-2 Base Counter t0/t1 Timing

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_TH) 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.

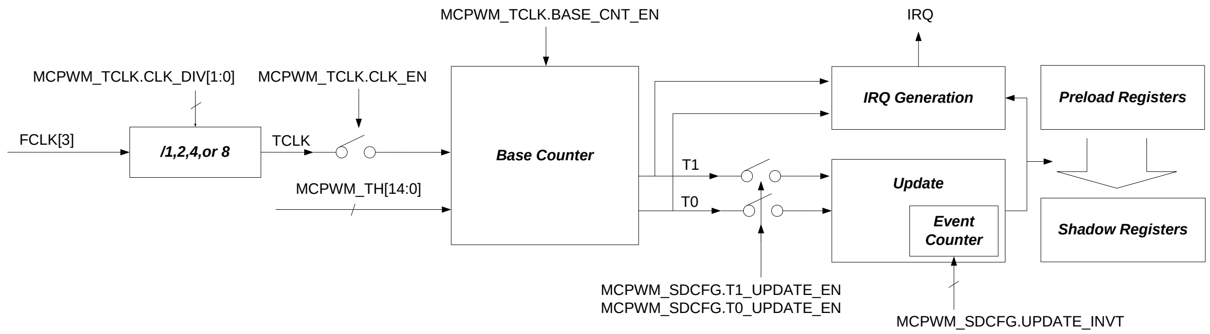


Fig. 13-3 MCPWM Update Mechanism

13.1.2 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.

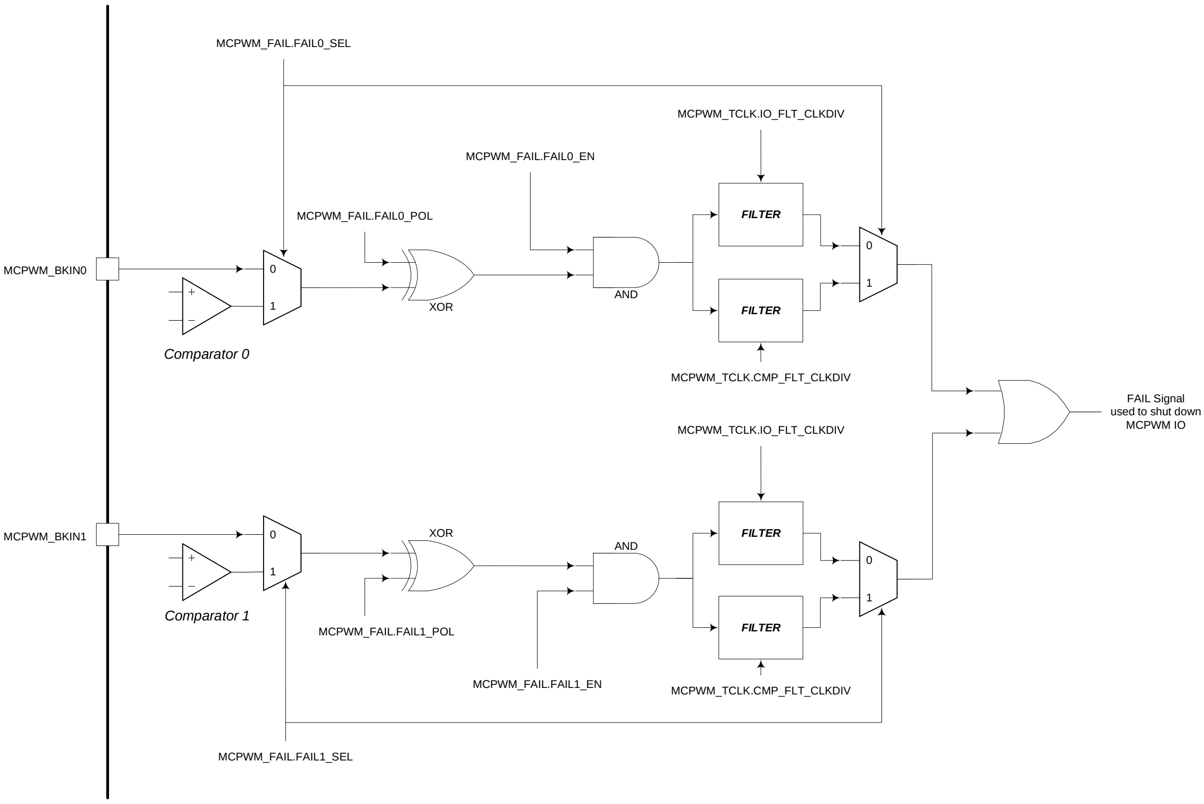


Fig. 13-4 MCPWM FAIL Logic Diagram

The clock of the Filter module comes from the gated clock FCLK [3] of the system main clock MCLK, and is divided by two stages. The first-stage frequency division is controlled by MCPWM_TCLK.CLK_DIV, which divides by 1/2/4/8 times. The second-stage frequency division can achieve 1 ~ 16 times the frequency division. If the Fail signal comes from MCPWM_BKIN [1: 0], then use MCPWM_TCLK.IO_FLT_CLKDIV [3: 0] as the second-stage frequency division coefficient; If the Fail signal comes from the internal comparator output, then use MCPWM_TCLK.CMP_FLT_CLKDIV [3: 0] as the frequency division factor of the second stage, as shown in Fig. 13-5.

The MCPWM module uses the frequency-divided clock to filter the Fail signal. The filter width is fixed at 16 cycles, that is, the input signal must be stable for at least 16 clock cycles (clock frequency divided by two) before the hardware determines it as a valid input signal. The formula for the filter time constant is as follows, where T_{MCLK} is the clock period of MCLK/FCLK [3], 96MHz corresponds to 10.4ns. MCPWM_TCLK.FLT_CLKDIV may be MCPWM_TCLK.IO_FLT_CLKDIV or MCPWM_TCLK.CMP_FLT_CLKDIV depending on the configuration.

$$T = T_{MCLK} \times (\text{MCPWM\_TCLK.CLK\_DIV}) \times (\text{MCPWM\_TCLK.FLT\_CLKDIV}+1) \times 16$$

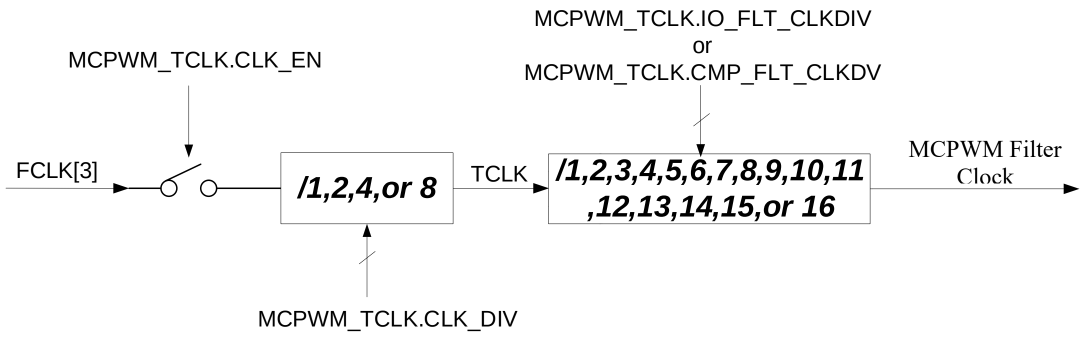


Fig. 13-5 MCPWM Fail Signal Filtering Clock Generation Logic

Once a Fail event occurs, that is

Fig. 13-4 Finally, the Fail signal is output, and the hardware forces the IO output to the default value of the fault specified in the MCPWM_FAIL.CHxN_DEFAULT and MCPWM_FAIL.CHxP_DEFAULT registers. After then, the values of MCPWM_FAIL.CHxN_DEFAULT and MCPWM_FAIL.CHxP_DEFAULT are directly output to the IO port, and are no longer affected by the polarity control such as MCPWM_FAIL.FAIL_POL.

The 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.

13.1.3 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

13.1.4 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:

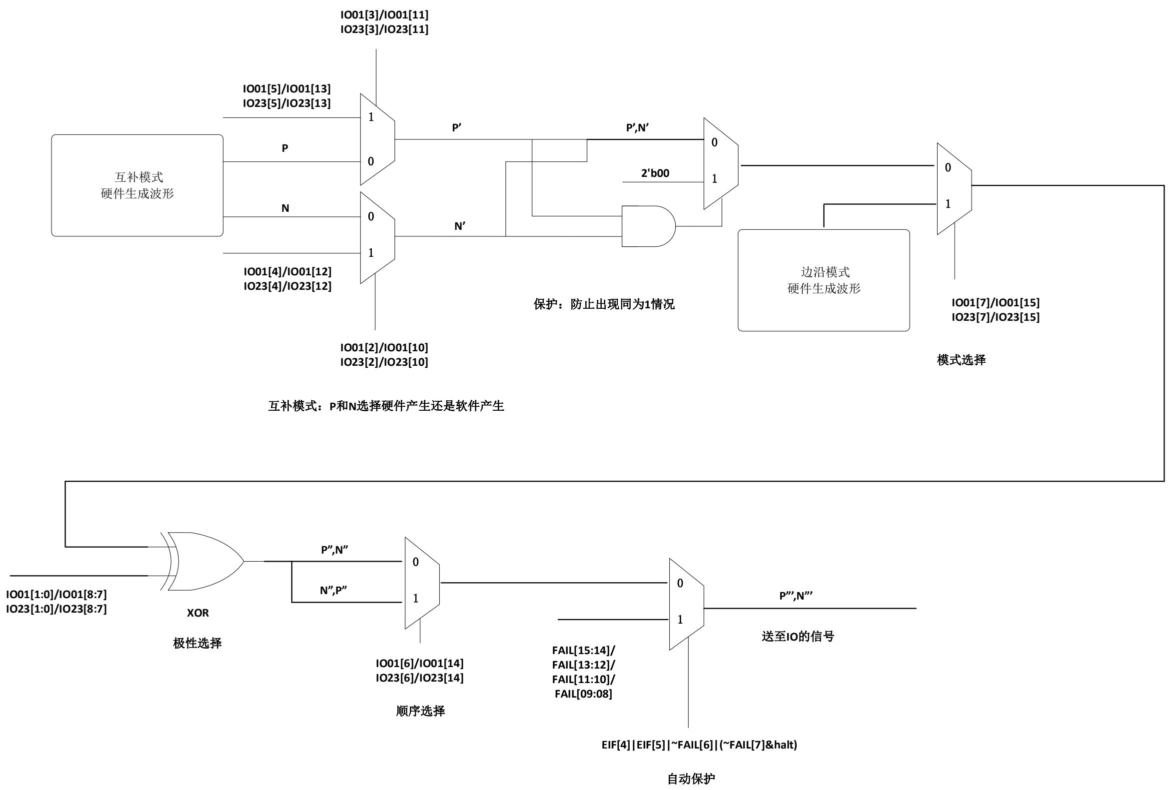


Fig. 13-6 IO Driver Module Data Flow Chart

13.1.4.1 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.

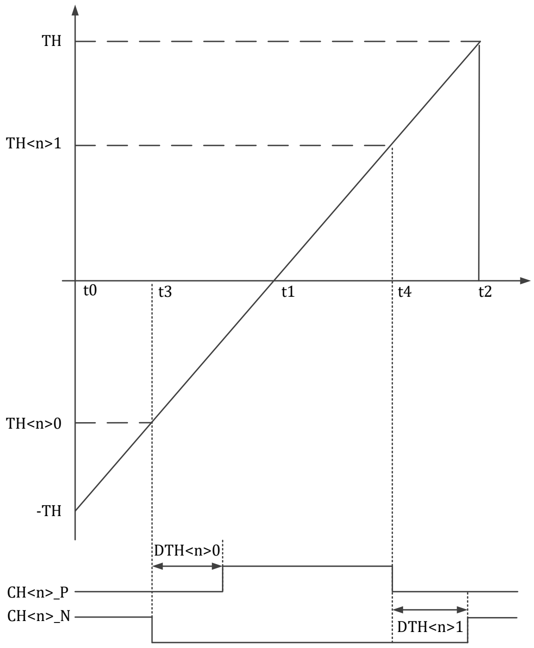


Fig. 13-7 MCPWM Timing TH <n> 0 and TH <n> 1——Complementary Mode

13.1.4.2 MCPWM Wave-form Control: Center-aligned Push-pull Mode

Complementary push-pull mode. In the first cycle, CH <n> P is set to 1 at time t3, and CH <n> P goes low at time t4. In the second cycle, CH <n> N is set to 1 at time t3, and CH <n> N goes low at time t4.

Both t3 and t4 will generate corresponding interrupts.

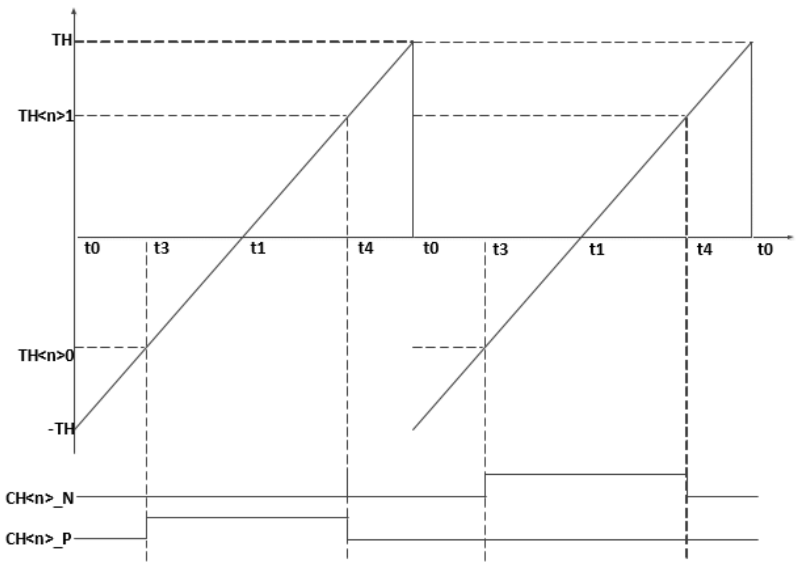


Fig. 13-8 MCPWM Timing TH <n> 0 and TH <n> 1——Complementary Mode

13.1.4.3 MCPWM Wave-form Output: Edge-aligned Mode

In edge-aligned mode, CH <n> P and CH <n> N are reset to 0 at time t0 at the same time, then

CH<n>N goes high at time t3, and CH<n>P goes high 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.

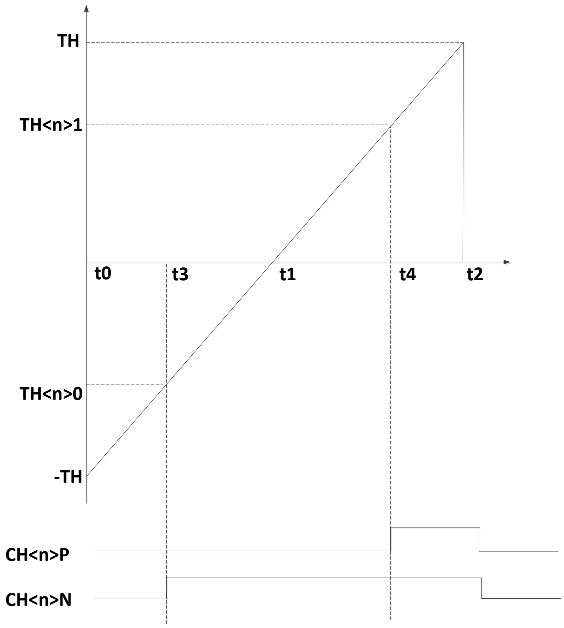


Fig. 13-9 MCPWM Timing Edge-aligned Mode

13.1.4.4 MCPWM Wave-form Control: Edge-aligned Push-pull Mode

Edge-aligned push-pull mode. In the first cycle, CH <n> P is set to 1 at time t0, and CH <n> P goes low at time t3. In the second cycle, CH <n> N is set to 1 at time t0, and CH <n> N goes low at time t3.

Both t0 and t3 will generate corresponding interrupts.

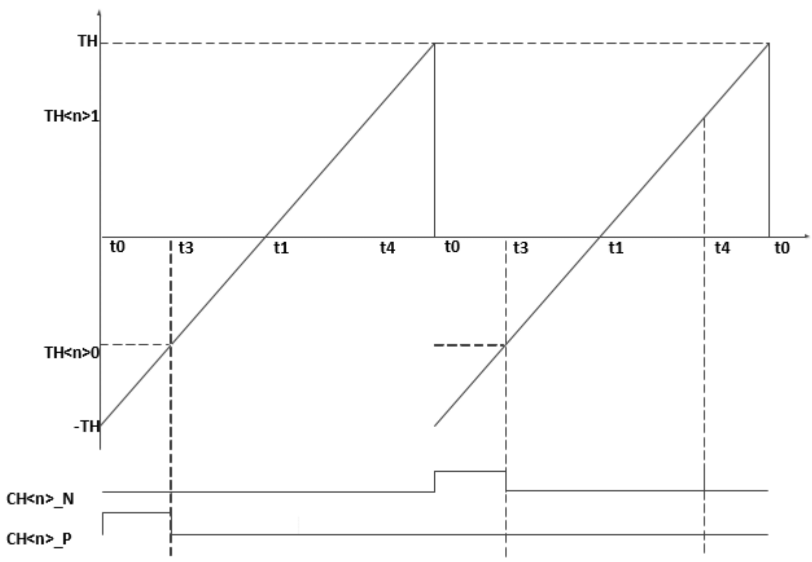


Fig. 13-10 MCPWM Timing TH <n> 0 and TH <n> 1 Edge-aligned Push-pull Mode

13.1.4.5 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 be adjusted independently.

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 (interrupt output remains until MCU writes 0).

MCPWM IO can also be output by software configuration, that is, the dead-zone control is controlled by software. If the PWM is complementary mode, it is still guaranteed by the hardware that it is not high or low at the same time.

The position of CH <n> P and CH <n> N output to IO can be interchanged.

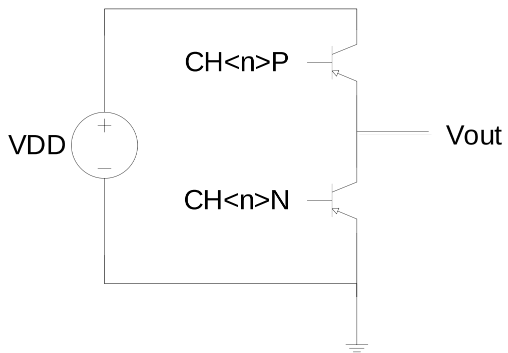


Fig. 13-11 MCPWM IO Control Diagram

13.1.4.6 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.

13.1.4.7 MCPWM IO: Auto-protection

When an emergency stop event (Fail event) occurs, CH <n> P and CH <n> N should be switched to the off state automatically. Remember to turn off the level configuration (MCPWM_FAIL.CHxN_DEFAULT and MCPWM_FAIL.CHxP_DEFAULT control the default level).

- After the chip works normally, the default output level of IO is the specified value of register MCPWM_FAIL.CHxN_DEFAULT and MCPWM_FAIL.CHxP_DEFAULT. When the user configuration is completed and MCPWM works normally, set MCPWM_FAIL.MCPWM_OE (ie MOE) to 1, and the IO output level is controlled by MCPWM IO 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, PWM stops output, and output the FAIL [15: 8] value.

13.1.5 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.

13.1.6 MCPWM Main Events

The counter thresholds of the MCPWM module and corresponding events are shown in the following table.

Table 13-1 MCPWM Counter Threshold and Events

T0	-MCPWM_TH
T1	0
TIO0[0]	MCPWM_TH00
TIO0[1]	MCPWM_TH01
TIO1[0]	MCPWM_TH10
TIO1[1]	MCPWM_TH11
TIO2[0]	MCPWM_TH20
TIO2[1]	MCPWM_TH21
TIO3[0]	MCPWM_TH30
TIO3[1]	MCPWM_TH31
TADC[0]	MCPWM_TMR0
TADC [1]	MCPWM_TMR1
TADC [2]	MCPWM_TMR2
TADC [3]	MCPWM_TMR3

13.2 Register

13.2.1 Address Allocation

The base address of the MCPWM module register is 0x4001_1C00, and the register list is as follows:

Table 13-2 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_TH	0x30	MCPWM threshold register
MCPWM_UPDATE	0x34	MCPWM load control register
MCPWM_IE	0x38	MCPWM interrupt control register
MCPWM_IF	0x3C	MCPWM interrupt flag register
MCPWM_EIE	0x40	MCPWM abnormal interrupt control register
MCPWM_EIF	0x44	MCPWM abnormal interrupt flag register
MCPWM_RE	0x48	MCPWM DMA request enable register
MCPWM_PP	0x4C	MCPWM push-pull mode enable register
MCPWM_IO01	0x50	MCPWM IO01 control register
MCPWM_IO23	0x54	MCPWM IO23 control register
MCPWM_SDCFG	0x58	MCPWM load configuration register
MCPWM_TCLK	0x60	MCPWM clock divider control register
MCPWM_FAIL	0x64	MCPWM short circuit control register
MCPWM_PRT	0x74	MCPWM protection register
MCPWM_CNT	0x78	MCPWM counter register
MCPWM_DTH00	0x80	MCPWM CH0 N channel dead-zone width control register
MCPWM_DTH01	0x84	MCPWM CH0 P channel dead-zone width control register
MCPWM_DTH10	0x88	MCPWM CH1 N channel dead-zone width control register

MCPWM_DTH11	0x8C	MCPWM CH1 P channel dead-zone width control register
MCPWM_DTH20	0x90	MCPWM CH2 N channel dead-zone width control register
MCPWM_DTH21	0x94	MCPWM CH2 P channel dead-zone width control register
MCPWM_DTH30	0x98	MCPWM CH3 N channel dead-zone width control register
MCPWM_DTH31	0x9C	MCPWM CH3 P channel dead-zone width control register

Table 13-3Registers Protected by MCPWM_PRT

Name	Offset Address	Description
MCPWM_TH	0x30	MCPWM threshold register
MCPWM_IE	0x38	MCPWM interrupt control register
MCPWM_EIE	0x40	MCPWM abnormal interrupt control register
MCPWM_RE	0x48	MCPWM DMA request enable register
MCPWM_PP	0x4C	MCPWM push-pull mode enable register
MCPWM_IO01	0x50	MCPWM IO01 control register
MCPWM_IO23	0x54	MCPWM IO23 control register
MCPWM_SDCFG	0x58	MCPWM load configuration register
MCPWM_TCLK	0x60	MCPWM clock divider control register
MCPWM_FAIL	0x64	MCPWM short circuit control register
MCPWM_DTH00	0x80	MCPWM CH0 N channel dead-zone width control register
MCPWM_DTH01	0x84	MCPWM CH0 P channel dead-zone width control register
MCPWM_DTH10	0x88	MCPWM CH1 N channel dead-zone width control register
MCPWM_DTH11	0x8C	MCPWM CH1 P channel dead-zone width control register
MCPWM_DTH20	0x90	MCPWM CH2 N channel dead-zone width control register
MCPWM_DTH21	0x94	MCPWM CH2 P channel dead-zone width control register
MCPWM_DTH30	0x98	MCPWM CH3 N channel dead-zone width control register
MCPWM_DTH31	0x9C	MCPWM CH3 P channel dead-zone width control register

Table 13-4 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_TH	0x30	MCPWM threshold register

13.2.2 MCPWM_TH00

Unprotected register

Address: 0x4001_1C00

Reset value: 0x0

Table 13-5 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.

13.2.3 MCPWM_TH01

Unprotected register

Address: 0x4001_1C04

Reset value: 0x0

Table 13-6 MCPWM_TH01 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.

13.2.4 MCPWM_TH10

Unprotected register

Address: 0x4001_1C08

Reset value: 0x0

Table 13-7 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.

13.2.5 MCPWM_TH11

Unprotected register

Address: 0x4001_1C0C

Reset value: 0x0

Table 13-8 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.

13.2.6 MCPWM_TH20

Unprotected register

Address: 0x4001_1C10

Reset value: 0x0

Table 13-9 MCPWM_TH20 Configuration Register

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.

13.2.7 MCPWM_TH21

Unprotected register

Address: 0x4001_1C14

Reset value: 0x0

Table 13-10 MCPWM_TH21 Configuration Register

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.

13.2.8 MCPWM_TH30

Unprotected register

Address: 0x4001_1C18

Reset value: 0x0

Table 13-11 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.

13.2.9 MCPWM_TH31

Unprotected register

Address: 0x4001_1C1C

Reset value: 0x0

Table 13-12 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.

13.2.10 MCPWM_TMR0

Unprotected register

Address: 0x4001_1C20

Reset value: 0x0

Table 13-13 MCPWM_TMR0 Configuration Register

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

Location	Bit name	Description
[31:16]		Unused
[15:0]	TMR0	Compare threshold 0 register for ADC sampling timer, 16-bit signed number; after an update event occurs, this register is loaded into the MCPWM operating system.

13.2.11 MCPWM_TMR1

Unprotected register

Address: 0x4001_1C24

Reset value: 0x0

Table 13-14 MCPWM_TMR1 Configuration Register

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

Location	Bit name	Description
[31:16]		Unused
[15:0]	TMR1	Compare threshold 1 register for ADC sampling timer, 16-bit signed number; after an update event occurs, this register is loaded into the MCPWM operating system.

13.2.12 MCPWM_TMR2

Unprotected register

Address: 0x4001_1C28

Reset value: 0x0

Table 13-15 MCPWM_TMR2 Configuration Register

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">TMR2</td></tr>
<tr><td colspan="16">RW</td></tr>
<tr><td colspan="16">0</td></tr>
</table>

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.

13.2.13 MCPWM_TMR3

Unprotected register

Address: 0x4001_1C2C

Reset value: 0x0

Table 13-16 MCPWM_TMR3 Configuration Register

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">TMR3</td></tr>
<tr><td colspan="16">RW</td></tr>
<tr><td colspan="16">0</td></tr>
</table>

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.

13.2.14 MCPWM_TH

Write-protected register

Address: 0x4001_1C30

Reset value: 0x0

Table 13-17 MCPWM_TH Configuration Register

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 counter threshold value, 15-bit unsigned number, the counter in the operating system of MCPWM counts from -TH to TH; after an update event occurs, this register is loaded into the MCPWM operating system.

13.2.15 MCPWM_UPDATE

Unprotected register

Address: 0x4001_1C34

Reset value: 0x0

Table 13-18 MCPWM_UPDATE Configuration Register

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

Location	Bit name	Description
[31:13]		Unused
[12]	TH_UPDATE	Manually load the contents of the MCPWM_TH register into the MCPWM operating system. 1: Load; 0: Do not load.
[11]	TMR3_UPDATE	Manually load the contents of the MCPWM_TMR3 register into the MCPWM operating system. 1: Load; 0: Do not load.
[10]	TMR2_UPDATE	Manually load the contents of the MCPWM_TMR2 register into the MCPWM operating system. 1: Load; 0: Do not load.
[9]	TMR1_UPDATE	Manually load the contents of the MCPWM_TMR1 register into the MCPWM operating system.

		1: Load; 0: Do not load.
[8]	TMR0_UPDATE	Manually load the contents of the MCPWM_TMR0 register into the MCPWM operating system. 1: Load; 0: Do not load.
[7]	TH31_UPDATE	Manually load the contents of the MCPWM_TH31 register into the MCPWM operating system. 1: Load; 0: Do not load.
[6]	TH30_UPDATE	Manually load the contents of the MCPWM_TH30 register into the MCPWM operating system. 1: Load; 0: Do not load.
[5]	TH21_UPDATE	Manually load the contents of the MCPWM_TH21 register into the MCPWM operating system. 1: Load; 0: Do not load.
[4]	TH20_UPDATE	Manually load the contents of the MCPWM_TH20 register into the MCPWM operating system. 1: Load; 0: Do not load.
[3]	TH11_UPDATE	Manually load the contents of the MCPWM_TH11 register into the MCPWM operating system. 1: Load; 0: Do not load.
[2]	TH10_UPDATE	Manually load the contents of the MCPWM_TH10 register into the MCPWM operating system. 1: Load; 0: Do not load.
[1]	TH01_UPDATE	Manually load the contents of the MCPWM_TH01 register into the MCPWM operating system. 1: Load; 0: Do not load.
[0]	TH00_UPDATE	Manually load the contents of the MCPWM_TH00 register into the MCPWM operating system. 1: Load; 0: Do not load.

Writing 1 to corresponding bits of MCPWM_UPDATE will trigger shadow registers to update their values to preload values. MCPWM_UPDATE is write-only and will be cleared automatically. Each time we write to MCPWM_UPDATE's specific bits will trigger shadow register update for once.

13.2.16 MCPWM_IE

Write-protected register

Address: 0x4001_1C38

Reset value: 0x0

Table 13-19 MCPWM_IE Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SHADE_IE	TMR3_IE	TMR2_IE	TMR1_IE	TMR0_IE	TH31_IE	TH30_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	RW	RW

	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Location	Bit name	Description
[31:15]		Unused
[14]	SHADE_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.

13.2.17 MCPWM_IF

Unprotected register

Address: 0x4001_1C3C

Reset value: 0x0

Table 13-20 MCPWM_IF Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SHADE_IF	TMR3_IF	TMR2_IF	TMR1_IF	TMR0_IF	TH31_IF	TH30_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	RW1C	RW1C
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Location	Bit name	Description
[31:15]		Unused
[14]	SHADE_IF	MCPWM_TH/MCPWM_TH00 ~ MCPWM_TH31/MCPWM_TMR0 ~ MCPWM_TMR3 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 the 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 the 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 the 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 the 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 the 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 the 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 the 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 the 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

		the 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 the 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 the 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 the 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.

13.2.18 MCPWM_EIE

Write-protected register

Address: 0x4001_1C40

Reset value: 0x0

Table 13-21 MCPWM_EIE Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										FAIL1_IE	FAIL0_IE	CH3_SHORT_IE	CH2_SHORT_IE	CH1_SHORT_IE	CH0_SHORT_IE
										RW	RW	RW	RW	RW	RW
										0	0	0	0	0	0

Location	Bit name	Description
[31:6]		Unused
[5]	FAIL1_IE	FAIL1 interrupt source enable. 1: enable; 0: disable.
[4]	FAIL0_IE	FAIL0 interrupt source enable. 1: enable; 0: disable.
[3]	CH3_SHORT_IE	MCPWM CH3_P and CH3_N are valid at the same time, the interrupt source is enabled. 1: enable; 0: disable.
[2]	CH2_SHORT_IE	MCPWM CH2_P and CH2_N are valid at the same time, the interrupt source is enabled. 1: enable; 0: disable.
[1]	CH1_SHORT_IE	MCPWM CH1_P and CH1_N are valid at the same time, the interrupt

		source is enabled. 1: enable; 0: disable.
[0]	CH0_SHORT_IE	MCPWM CH0_P and CH0_N are valid at the same time, and the interrupt source is enabled. 1: enable; 0: disable.

13.2.19 MCPWM_EIF

Unprotected register

Address: 0x4001_1C44

Reset value: 0x0

Table 13-22 MCPWM_EIF Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										FAIL1_IF	FAIL0_IF	CH3_SHORT_IF	CH2_SHORT_IF	CH1_SHORT_IF	CH0_SHORT_IF
										RW1C	RW1C	RW1C	RW1C	RW1C	RW1C
										0	0	0	0	0	0

Location	Bit name	Description
[31:6]		Unused
[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]	CH3_SHORT_IF	MCPWM CH3_P and CH3_N are active at the same time, interrupt source event occurred. 1: occurred; 0: did not occurred. Write 1 to clear.
[2]	CH2_SHORT_IF	MCPWM CH2_P and CH2_N are valid at the same time, interrupt source event occurred. 1: occurred; 0: did not occurred. Write 1 to clear.
[1]	CH1_SHORT_IF	MCPWM CH1_P and CH1_N are valid at the same time, interrupt source event occurred. 1: occurred; 0: did not occurred. Write 1 to clear.
[0]	CH0_SHORT_IF	MCPWM CH0_P and CH0_N are valid at the same time, interrupt source event occurred. 1: occurred; 0: did not occurred. Write 1 to clear.

13.2.20 MCPWM_RE

Write-protected register

Address: 0x4001_1C48

Reset value: 0x0

Table 13-23 MCPWM_RE Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										ZC_RE	STOV_RE	TMR3_RE	TMR2_RE	TMR1_RE	TMR0_RE
										RW	RW	RW	RW	RW	RW
										0	0	0	0	0	0

Location	Bit name	Description
[31:6]		Unused
[5]	ZC_RE	DMA request enable signal. Over-zero request. Write 1 to enable; write 0 to disable.
[4]	STOV_RE	DMA request enable signal. Starting event request. Write 1 to enable; write 0 to disable.
[3]	TMR3_RE	DMA request enable signal. ADC channel 3 request. Write 1 to enable; write 0 to disable.
[2]	TMR2_RE	DMA request enable signal. ADC channel 2 request. Write 1 to enable; write 0 to disable.
[1]	TMR1_RE	DMA request enable signal. ADC channel 1 request. Write 1 to enable; write 0 to disable.
[0]	TMR0_RE	DMA request enable signal. ADC channel 0 request. Write 1 to enable; write 0 to disable.

13.2.21 MCPWM_PP

Write-protected register

Address: 0x4001_1C4C

Reset value: 0x0

Table 13-24 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. Write 1 to enable; write 0 to disable.
[2]	IO2_PPE	IO2 push-pull mode enable signal. Write 1 to enable; write 0 to disable.

[1]	IO1_PPE	IO1 push-pull mode enable signal. Write 1 to enable; write 0 to disable.
[0]	IO0_PPE	IO0 push-pull mode enable signal. Write 1 to enable; write 0 to 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.

13.2.22 MCPWM_IO01

Write-protected register

Address: 0x4001_1C50

Reset value: 0x0

Table 13-25 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.**

13.2.23 MCPWM_IO23

Write-protected register

Address: 0x4001_1C54

Reset value: 0x0

Table 13-26 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 source. 1: From CH3_SCTRLP; 0: The counter is generated in the MCPWM operating system.

[10]	CH3_NS	CH3 N source. 1: From CH3_SCTRLN; 0: The counter is generated in the MCPWM operating system.
[9]	CH3_PP	CH3 P polarity selection. 1: CH3 P signal is inverted and output; 0: CH3 P signal is output normally.
[8]	CH3_NP	CH3 N polarity selection. 1: CH3 N signal is inverted and output; 0: CH3 N signal is output normally.
[7]	CH2_WM	CH2 working mode selection. 1: Edge mode; 0: complementary mode.
[6]	CH2_PN_SW	CH2 P and N channel output interchange selection. That is, the P channel signal is finally output from the N channel, and the N channel signal is finally output from the P channel. 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.**

13.2.24 MCPWM_SDCFG

Write-protected register

Address: 0x4001_1C58

Reset value: 0x0

Table 13-27 MCPWM_SDCFG Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									AUTO_ERR_CLR	T1_UPDATE_EN	T0_UPDATE_EN	UPDATE_INTV			
									RW	RW	RW	RW			
									0	0	0	0			

Location	Bit name	Description
[31:7]		Unused
[6]	AUTO_ERR_CLR	Whether the AUTO_ERR_CLR update event automatically clears MCPWM_EIF [5: 4] and sets MOE to restore MCPWM signal output.

		1: Enable automatic fault clearing function; 0: Disable automatic fault clearing function.
[5]	T1_UPDATE_EN	The t1 (over-zero) event update is enabled. 1: enable; 0, disable.
[4]	T0_UPDATE_EN	t0 (starting point) event update enable. 1: enable; 0, disable.
[3:0]	UPDATE_INTV	Update interval. Once the number of t0 and t1 events is equal to UPDATE_INTV + 1, the MCPWM system triggers the operation of the MCPWM_TH (including THxx) and MCPWM_TMR registers automatically, and loaded into the MCPWM operating system. If both B [5] and B [4] are closed, this type of loading will not be triggered, and the loading can only be triggered manually.

13.2.25 MCPWM_TCLK

Write-protected register

Address: 0x4001_1C60

Reset value: 0x0

Table 13-28 MCPWM_TCLK Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP_FLT_CLKDIV				IO_FLT_CLKDIV								BASE_CNT_EN	CLK_EN	CLK_DIV	
RW				RW								RW	RW	RW	
0				0								0	0	0	

Location	Bit name	Description
[31:16]		Unused
[3:12 PM]	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/(B[15:12]+1). The frequency division range is 1 to 16.
[11:8]	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/(B[11:8]+1). The frequency division range is 1 to 16.
[7:4]		Unused
[3]	BASE_CNT_EN	MCPWM operation counter enable switch. 1: enable; 0: disable.
[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

13.2.26 MCPWM_FAIL

Write-protected register

Address: 0x4001_1C64

Reset value: 0x0

Table 13-29 MCPWM_FAIL Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH3P_DEFAULT	CH3N_DEFAULT	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	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]	CH3N_DEFAULT	CH3 N channel default value
[14]	CH3P_DEFAULT	CH3 P channel default value
[13]	CH2N_DEFAULT	CH2 N channel default value
[12]	CH2P_DEFAULT	CH2 P channel default value
[11]	CH1N_DEFAULT	CH1 N channel default value
[10]	CH1P_DEFAULT	CH1 P channel default value
[9]	CH0N_DEFAULT	CH0 N channel default value
[8]	CH0P_DEFAULT	CH0 P channel default value. When a FAIL event occurs or MOE is 0, the corresponding channel outputs the default level. **The default level output controls the channel output, and 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]	MCPWM_OE	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 No. 1.
[0]	FAIL0_SEL	FAIL0 source selection. 1: Comparator 0 result; 0: From GPIO No. 0.

MCPWM_FAIL can be used to set emergency stop events and block MCPWM signal output. There are two main emergency stop events, FAIL0 and FAIL1. There are four signal sources, Comparator 0 output and Comparator 1 output, and MCPWM_BKIN0 and MCPWM_BKIN0. FAIL0 can come from comparator 0 output or chip IO MCPWM_BKIN0, FAIL1 can come from comparator 1 output or chip IO MCPWM_BKIN1.

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 signal source Comparator 0 output and the filtered clock output of Comparator 1 are set by MCPWM_TCLK.CMP_FLT_CLKDIV; The filter clock frequency division of the signal sources MCPWM_BKIN0 and MCPWM_BKIN0 is set by MCPWM_TCLK.IO_FLT_CLKDIV.

Finally, FAIL0 and FAIL1 filter the 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 13.1.2 Fail .

13.2.27 MCPWM_PRT

Unprotected register

Address: 0x4001_1C74

Reset value: 0x0

Table 13-30 MCPWM_PRT Configuration 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.

13.2.28 MCPWM_CNT

Unprotected register

Address: 0x4001_1C78

Reset value: 0x0

Table 13-31 MCPWM_CNT Configuration Register

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">CNT</td></tr>
<tr><td colspan="16">RW</td></tr>
<tr><td colspan="16">0</td></tr>
</table>

Location	Bit name	Description
[31:16]		Unused
[15:0]	CNT	Count value in MCPWM operating system.

13.2.29 MCPWM_DTH00

Write-protected register

Address: 0x4001_1C80

Reset value: 0x0

Table 13-32 MCPWM_DTH00 Configuration Register

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">DTH00</td></tr>
<tr><td colspan="16">RW</td></tr>
<tr><td colspan="16">0</td></tr>
</table>

Location	Bit name	Description
[31:16]		Unused
[15:0]	DTH00	MCPWM CH0 P-channel dead-zone width control register, 10-bit unsigned number

13.2.30 MCPWM_DTH01

Write-protected register

Address: 0x4001_1C84

Reset value: 0x0

Table 13-33 MCPWM_DTH01 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTH01															
RW															
0															

Location	Bit name	Description
[31:16]		Unused
[15:0]	DTH01	MCPWM CH0 N-channel dead-zone width control register, 10-bit unsigned number

13.2.31 MCPWM_DTH10

Write-protected register

Address: 0x4001_1C88

Reset value: 0x0

Table 13-34 MCPWM_DTH10 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTH10															
RW															
0															

Location	Bit name	Description
[31:16]		Unused
[15:0]	DTH10	MCPWM CH1 P-channel dead-zone width control register, 10-bit unsigned number

13.2.32 MCPWM_DTH11

Write-protected register

Address: 0x4001_1C8C

Reset value: 0x0

Table 13-35 MCPWM_DTH11 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTH11															
RW															
0															

Location	Bit name	Description
[31:16]		Unused
[15:0]	DTH11	MCPWM CH1 N-channel dead-zone width control register, 10-bit unsigned number

13.2.33 MCPWM_DTH20

Write-protected register

Address: 0x4001_1C90

Reset value: 0x0

Table 13-36 MCPWM_DTH20 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTH20															
RW															
0															

Location	Bit name	Description
[31:16]		Unused
[15:0]	DTH20	MCPWM CH2 P-channel dead-zone width control register, 10-bit unsigned number

13.2.34 MCPWM_DTH21

Write-protected register

Address: 0x4001_1C94

Reset value: 0x0

Table 13-37 MCPWM_DTH21 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTH21															
RW															
0															

Location	Bit name	Description
[31:16]		Unused
[15:0]	DTH21	MCPWM CH2 N-channel dead-zone width control register, 10-bit unsigned number

13.2.35 MCPWM_DTH30

Write-protected register

Address: 0x4001_1C98

Reset value: 0x0

Table 13-38 MCPWM_DTH30 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTH30															
RW															
0															

Location	Bit name	Description
[31:16]		Unused
[15:0]	DTH30	MCPWM CH3 P-channel dead-zone width control register, 10-bit unsigned number

13.2.36 MCPWM_DTH31

Write-protected register

Address: 0x4001_1C9C

Reset value: 0x0

Table 13-39 MCPWM_DTH31 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTH31															
RW															
0															

Location	Bit name	Description
[31:16]		Unused
[15:0]	DTH31	MCPWM CH3 N-channel dead-zone width control register, 10-bit unsigned number

14 UART

14.1 Introduction

UART features are as follows:

full-duplex operation

Support 7/8 data bit

Support 1/2 stop bi

Support odd/even/no parity mode

1 byte tx buffer

1 byte rx buffer

Support Multi-drop Slave/Master mode

14.2 Pin Function Description

14.2.1 Transport (TX)

The UART includes a byte tx buffer. When the tx buffer has data, the UART loads the data of the tx buffer and sends it out via TX.

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.

14.2.2 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.

14.2.3 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*UARTx_DIVH+UARTx_DIVL+1)

The UART module clock can be divided by SYS_CLK_DIV2,

UART module clock = main clock/(1+SYS_CLK_DIV2)

Table 14-1 Example of UART Baud Rate Configuration

UART Baud Rate	SYS_CLK_DIV2	UART_DIVH	UART_DIVL
300	0x0007	0x9C	0x3F
600	0x0003	0x9C	0x3F
1200	0x0001	0x9C	0x3F
2400	0x0000	0x9C	0x3F
4800	0x0000	0x4E	0x1F
9600	0x0000	0x27	0x0F
19200	0x0000	0x13	0x87
38400	0x0000	0x09	0xC3
43000	0x0000	0x08	0xB8
56000	0x0000	0x06	0xB1
57600	0x0000	0x06	0x82
115200	0x0000	0x03	0x40

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

14.2.4 DMA Configuration

UART module, support DMA operation, and realize DMA to move data, which greatly reduces the burden of MCU. UART transfers data by using DMA, and the precautions are as follow.

DMA couldn't receive and transmit data at the same time. DMA could only be used when receive only or transmit only.

For the transmission mode, there are two options for DMA configuration.

Option 1: If the UARTx_IE.TX_BUF_EMPTY_RE configuration is valid. The UARTx module will prefetch the first byte for transmission; Once the data enters the transmit queue, UARTx_IE.UARTx_BUFF is empty, and the hardware will automatically request DMA to move the next byte until the data is moved. After the DMA is moved, the DMA completion interrupt will be generated; However, UARTx may not have sent the last byte, and an exception may occur if to operate UARTx immediately. It is recommended to enable the UARTx_IE.TX_DONE_IE interrupt in the DMA interrupt handler. When UARTx finished sending the last byte, and generated a transmission completion interrupt, turned off UARTx_IE.TX_DONE_IE in the UARTx interrupt processing function.

Option 2: If the UARTx_IE.TX_DONE_RE is 1. The UART module does prefetch the first byte. Compared with option 1, if the data length of the current transmission is Len, the number of bytes transmitted by the DMA configuration is Len; Turn on the DMA interrupt, after the DMA transfer is completed, the UART is also sent, and the soft reset UARTx module reinitializes UARTx to start the next UARTx transmission.

14.3 Register

14.3.1 Address Allocation

The UART0 and UART1 implementations are identical.

The base address of UART0 is 0x4001_2800.

The base address of UART1 is 0x4001_2C00.

Table 14-2 UARTx Address Allocation List

Name	Offset Address	Description
UARTx_CTRL	0x00	UART control register
UARTx_DIVH	0x04	High byte register with UART baud rate setting
UARTx_DIVL	0x08	Low byte register with UART baud rate setting
UARTx_BUFF	0x0C	UART transceiver buffer register
UARTx_ADR	0x10	485 communication address matching register
UARTx_STT	0x14	UART status register
UARTx_IE	0x18	UART interrupt enable register
UARTx_IF	0x1C	UART interrupt flag register
UARTx_INV	0x20	UART IO flip enable

14.3.2 UARTx Control Register (UARTx_CTRL)

UART0_CTRL address: 0x4001_2800

UART1_CTRL address: 0x4001_2C00

Reset value: 0x0

Table 14-3 UARTx Control Register (UARTx_CTRL)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									MDMASTER_BIT9	MD_EN	CK_EN	CK_TYPE	BIT_ORDER	STOP_LEN	DAT_LEN
									RW	RW	RW	RW	RW	RW	RW
									0	0	0	0	0	0	0

Location	Bit name	Description
[31:8]		Unused
[6]	MDMASTER_BIT9	The 9th data bit value in Multi-drop Master mode
[5]	MD_EN	Enable Multi-drop. 0: disable, 1: enable
[4]	CK_EN	Enable verification. 0: disable, 1: enable

[3]	CK_TYPE	Parity check. 0: EVEN 1: ODD
[2]	BIT_ORDER	Bits transmitted first. 0: LSB, 1: MSB
[1]	STOP_LEN	Stop bit length. 0: 1bit, 1: 2bit
[0]	DAT_LEN	Data length. 0: 8bit, 1: 7bit

14.3.3 UARTx Baud Rate High-byte Register (UARTx_DIVH)

UART0_DIVH address: 0x4001_2804

UART1_DIVH address: 0x4001_2C04

Reset value: 0x0

Table 14-4 UARTx Baud Rate High-byte Register (UARTx_DIVH)

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

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

14.3.4 UARTx Baud Rate Low-byte Register (UARTx_DIVL)

UART0_DIVL address: 0x4001_2808

UART1_DIVL address: 0x4001_2C08

Reset value: 0x0

Table 14-5 UARTx Baud Rate Low-byte Register (UARTx_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* UARTx_DIVH+UARTx_DIVL)

14.3.5 UARTx Transceiver Buffer Register (UARTx_BUFF)

UART0_BUFF address: 0x4001_280C

UART1_BUFF address: 0x4001_2C0C

Reset value: 0x0

Table 14-6 UARTx Transceiver Buffer Register (UARTx_BUFF)

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8" rowspan="3"></td><td colspan="8">BUFF</td></tr>
<tr><td colspan="8">RW</td></tr>
<tr><td colspan="8">0</td></tr>
</table>

Location	Bit name	Description
[31:8]		Unused
[7:0]	BUFF	Write: data transmit buffer; read: data receive register

The Tx_buffer and Rx_buffer of the UART share the address 0x0C. Among them, Tx_buffer is write-only, Rx_buffer is read-only. Therefore, read access to UARTx_BUFF is to access UARTx_RX_BUFF, and write access to UARTx_BUFF is to access UARTx_TX_BUFF.

14.3.6 UARTx Address Match Register (UARTx_ADR)

UART0_ADR address: 0x4001_2810

UART0_ADR address: 0x4001_2810...

Reset value: 0x0

Table 14-7 UARTx Address Match Register (UARTx_ADR)

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8" rowspan="3"></td><td colspan="8">ADR</td></tr>
<tr><td colspan="8">RW</td></tr>
<tr><td colspan="8">0</td></tr>
</table>

Location	Bit name	Description
[31:8]		Unused
[7:0]	ADR	Used as address for 485 communication

14.3.7 UARTx Status Register (UARTx_STT)

UART0_STT address: 0x4001_2814

UART1_STT address: 0x4001_2C14

Reset value: 0x0

Table 14-8 UARTx Status Register (UARTx_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 (if the tx buffer is not empty, continue to transmit the data in the tx buffer) 1: done; 0: undone.
[0]	TX_BUF_EMPTY	Tx buffer status bit. 1: empty; 0: not empty.

14.3.8 UARTx Interrupt Enable Register (UARTx_IE)

UART0_IE address: 0x4001_2818

UART1_IE address: 0x4001_2C18

Reset value: 0x0

Table 14-9 UARTx Interrupt Enable Register (UARTx_IE)

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	CK_ERR_IE	STOP_ERR_IE	TX_BUF_EMPTY_IE	RX_DONE_IE	TX_DONE_IE
								RW	RW	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

Location	Bit name	Description
[31:8]		Unused
[7]	TX_BUF_EMPTY_RE	DMA request is enabled when Tx buffer is empty. The default value

		is 0. 0: Off; 1: On.
[6]	RX_DONE_RE	DMA request is enable upon reception. The default value is 0. 0: Off; 1: On.
[5]	TX_DONE_RE	DMA request is enable upon transmission. The default value is 0. 0: Off; 1: On.
[4]	CK_ERR_IE	Check error interrupt switch. The default value is 0. 0: Off; 1: On.
[3]	STOP_ERR_IE	Stop bit error interrupt switch. The default value is 0. 0: Off; 1: On.
[2]	TX_BUF_EMPTY_IE	Tx buffer empty interrupt switch. The default value is 0. 0: Off; 1: On.
[1]	RX_DONE_IE	Rx completion interrupt switch. The default value is 0. 0: Off; 1: On.
[0]	TX_DONE_IE	Tx completion interrupt switch. The default value is 0. 0: Off; 1: On.

14.3.9 UARTx Interrupt Flag Register (UARTx_IF)

UART0_IF address: 0x4001_281C

UART1_IF address: 0x4001_2C1C

Reset value: 0x0

Table 14-10 UARTx Interrupt Flag Register (UARTx_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	0	0	0

Location	Bit name	Description
[31:5]		Unused
[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.

14.3.10 UARTx IO Toggle Output Register (UARTx_INV)

UART0_INV address: 0x4001_2820

UART1_INV address: 0x4001_2C20

Reset value: 0x0

Table 14-11 UARTx Toggle Output Register (UARTx_INV)

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

Location	Bit name	Description
[31:2]		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 application sends "1", the hardware sends "1"; Invert output polarity means that the when the application 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 application receives "1"; Invert input polarity means that when the hardware receives "1", the application receives "0".

15 DSP

15.1 Introduction

The DSP module adopts a self-designed DSP instruction set, which can perform single-cycle arithmetic instructions such as addition, multiply-add (MAC), shift, and saturation, as well as multi-cycle arithmetic instructions such as division, square root, and trigonometric functions; It also has memory access instructions such as load and store, branch instructions such as unconditional jumps and conditional jumps, and miscellaneous instructions such as interrupt mention. Pseudo-instructions such as breakpoint instructions and register assignments can be used for debugging on the simulator.

DSP has two operating modes, autonomous operation and passive calling.

The so-called autonomous operation means that the DSP reads the instructions in the CODE MEM and the data in the DATA MEM to execute the DSP program, which is independent of the ARM Cortex M0. At this time, DSP_SC.PAUSED=0, that is, the DSP is in the running state; CODE MEM and DATA MEM allow the DSP Access but not allow CPU access to override.

Passive call means that the DSP is called by the ARM Cortex M0 as a peripheral module, and the CPU directly accesses the arithmetic operation resources inside the DSP, such as division, square root, and trigonometric functions. At this time, DSP_SC.PAUSED=1, that is, the DSP does not run the DSP program and is in a suspended state. The CODE MEM and DATA MEM allow the CPU to access and rewrite. For users who do not develop DSP programs, it is recommended to use this mode. The software running by CPU will call the arithmetic unit of the DSP directly.

DSP is equipped with independent program memory (CODE MEM) and data memory (DATA MEM). When DSP is paused, that is, DSP_SC.PAUSED = 1, users can access these two independent storage areas through the CPU; In the process of DSP initialization, the CPU should write the program and initial data of the DSP to the CODE MEM and DATA MEM of the DSP, respectively. The DSP has instructions for interrupt mention. After the interrupt is set, the DSP enters the suspended state, and the CPU is allowed to access the DATA MEM through the bus interface to interact with the DSP, including reading the DSP operation results and writing the data required for the subsequent operation of the DSP.

Besides, in order to make full use of DSP, when the DSP is suspended, it allows the CPU to directly access the arithmetic modules such as DSP divider, square root, and trigonometric functions through the DSP register interface, which allows the CPU to use DSP as a simple arithmetic co-processing module.

15.1.1 Functional Block Diagram

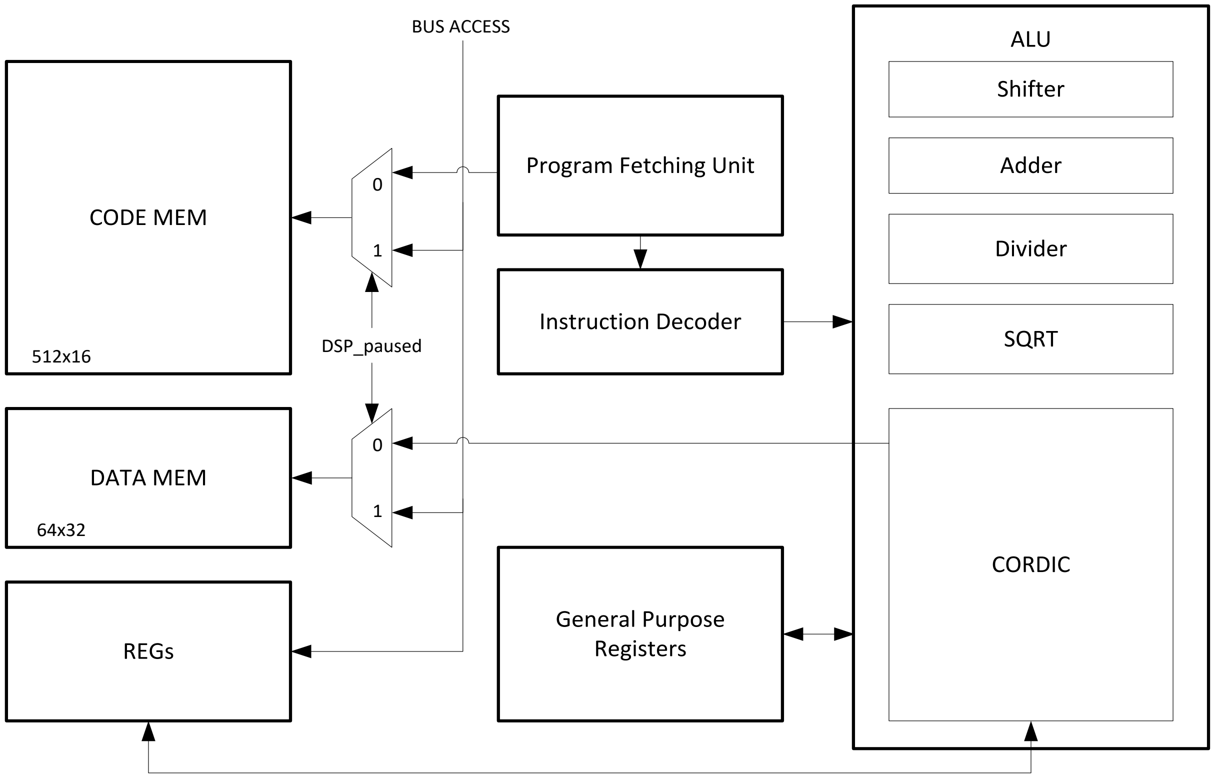


Fig. 15-1 DSP Module Functional Block Diagram

15.1.2 DSP Core Register

Table 15-1 DSP Core Register

Register	Bit Width	Usage
R0	32	Always read as 0
R1	32	
R2	32	
R3	32	
R4	32	
R5	32	
R6	32	ARCTAN module destination
R7	32	MAC result/DIV dividend
PC	16	Program Counter

The R0 register is a constant 0 register, data cannot be written, and the value is always 0 after readback. The R0 register can be used as a special operand to construct some pseudo-instructions. For example

ADD R0 R0 R0 is equivalent to NOP

MAC R1 R2 R0 is equivalent to MUL R1 R2, that is, the multiply-add (MAC) number is 0, which becomes multiply operation.

The R6 register is fixedly used to store the vector modulus value in the ARCTAN instruction.

The R7 register is used to store calculation results in the MAC instruction and to store the dividend operand in the DIV instruction.

The above convention is limited by the fixed-length instruction encoding, so it is necessary to fix an operand in the 4-operand instruction to use the convention register.

15.1.3 Bit Width

The dividend and quotient width of the division are both 32-bit signed numbers, and the divisor and remainder are 16-bit signed numbers.

The radicand is a 32-bit unsigned number, and the square root is a 16-bit unsigned number.

The two multiply-add (MAC) multipliers are 16-bit signed numbers, and the sum and result are 32-bit signed numbers.

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

Note: Regardless of whether it is a CPU call or a DSP arithmetic instruction call, please ensure that the operands do not exceed the representation range when using DSP arithmetic operation resources; otherwise, the calculation may be abnormal. For example, the two operands of multiplication are 16-bit signed numbers, and the divisor of division is 16-bit signed numbers. The range of 16-bit signed numbers is -32768 ~ 32767. If the input 32-bit data exceeds this range, for example, 50,000 or -40000 is used as the multiplication operand, a calculation error may occur since the lower 16 bits of the source operand have been intercepted.

The range of the dividend is -(2^31-1) ~ (2^31-1), the range of the divisor is -(2^15-1) ~ (2^15-1); the dividend shouldn't be assigned to -2^31, and the divisor shouldn't be assigned to -2^15.

In addition, since the CORDIC operation is based on successive approximations of multiple rotations, the error of the calculation result must not exceed 0.1%.

15.1.4 Instruction Cycle

The calculation of division instruction requires 10 cycles (96MHz).

The calculation of SQRT instruction requires 8 cycles (96MHz).

The calculation of trigonometric function instruction requires 8 cycles (96MHz).

The rest of the instructions are single-cycle instructions.

15.1.5 Address Space

Table 15-2 DSP Address Space

Module	Size	Address Space	Storage Size
code_mem	2kB	0x4001_4000 ~ 0x4001_47FF	512 x 16bit
data_mem	2kB	0x4001_4800 ~ 0x4001_4FFF	64 x 32bit
reg	2kB	0x4001_5000 ~ 0x4001_57FF	

Reserved	2kB	0x4001_5800 ~ 0x4001_5FFF	

The address space of DSP is divided into four sections, which are CODE MEM, DATA MEM, register MEM and reserved MEM; each MEM space occupies address space of 2kB, but the actual CODE MEM and DATA MEM storage space is less than 2kB. The CODE MEM is used to store the program code required by the DSP to run. It is a single-port SRAM that completes read and write operations in a single cycle; The DATA MEM is used to store the data required for DSP operation. It is a single-port SRAM that completes read and write operations in a single cycle; The register MEM is a DSP register that allows the CPU to access through the bus; the reserved memory is temporarily unused.

The CODE MEM bit width is 16, but it is still addressed by word, that is, the address is incremented by 4.

The DSP address space should be accessed at the appropriate time. The DSP status control register can be accessed at any time; DSP CODE MEM, DATA MEM, and CORDIC trigonometric function module, divider (DIV), Square Root (SQRT) in the register MEM can only access the register when the DSP is suspended, ie DSP_SC.PAUSED = 1. During the operation of the DSP, all arithmetic units may be used by the DSP. If the CPU access DSP through the register interface at the same time, it may cause an access conflict. Therefore, during the operation of the DSP, that is, when DSP_SC.PAUSED = 0, access to the arithmetic unit of the DSP through the register interface is prohibited.

When the DSP encounters an interrupt instruction, it will raise an interrupt and wait for the CPU to process. At the same time, the DSP enters the suspended state, and DSP_SC.PAUSED will be set to 1. Besides, the software can also set DSP_SC.PAUSED = 1 at any time to put the DSP into a suspended state. This mechanism is design to prevent the DSP from running permanently without losing power if there is no IRQ instruction in the program written by DSP.

15.2 Register

15.2.1 Address Allocation

The base address of the DSP module in the chip is 0x4001_4000. The base address of the DSP register in the chip is 0x4001_5000.

Table 15-3 DSP Register List

Name	Offset	Description
DSP_SC	0x00	DSP status control register
DSP_THETA	0x04	DSP sin/cos input angle register
DSP_X	0x08	DSP arctan/module calculate input coordinates X register
DSP_Y	0x0C	DSP arctan/module calculate input coordinates Y register
DSP_SIN	0x10	DSP sin/cos calculation result sin register
DSP_COS	0x14	DSP sin/cos calculation result cos register
DSP_MOD	0x18	DSP arctan calculation result sqrt(X^2+Y^2) Register
DSP_ARCTAN	0x1C	DSP arctan calculation result arctan(Y/X) angle register
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

15.2.2 DSP Status Control Register

15.2.2.1 DSP_SC

Address: 0x4001_5000

Reset value: 0x2

Table 15-4 DSP Status Control Register (DSP_SC)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												RESET_PC	CORDIC_MODE	PAUSED	IRQ
												WO	RW	RW	RWIC
												0	0	1	0

Location	Bit name	Description
[31:4]		Reserved
[3]	RESET_PC	When DSP is suspended, write "1" to reset DSP PC to address 0.
[2]	CORDIC_MODE	CORDIC mode, 0: arctan, 1: sin/cos
[1]	PAUSED	It indicates that the DSP is in a suspended state. When the DSP encounters an IRQ instruction, this bit gets set to "1", and could also be set to "1" by software writing. The software can start DSP operation after clearing this bit. 0： DSP is fetching instructions and data from CODE MEM and DATA MEM and running DSP programs 1： DSP is suspended to fetch instructions, allowing software to access DSP ALU via bus registers. Writing into registers will transfer ALU calculation oprands and starts calculation for one time, and reading from registers gets the calculation results For users who don't plan to write DSP program in CODE MEM, keep PAUSED=1
[0]	IRQ	DSP interrupt flag. Write 1 to clear

Note that the DSP is in a suspended state after reset, that is, when DSP_SC.PAUSED = 1; DSP_SC.CORDIC_MODE is used when the CPU accesses the CORDIC module through the register interface. The selection of sin/cos mode and arctan mode, and the calculation of sin/cos or arctan by the CORDIC module share the same hardware circuit. Therefore, the appropriate mode selection should be done by setting the DSP_SC calculator before performing a certain calculation.

The DSP_SC.CORDIC_MODE bit should be set only when the CPU calls the DSP CORDIC unit through the register interface. The DSP program can directly switch modes according to the SIN_COS or ARCTAN instructions, after then, DSP_SC.CORDIC_MODE no longer works.

When the software calls the CORDIC module to calculate sin/cos, the angle DSP_THETA is used as the input, and the sin/cos result is calculated and output to the DSP_SIN/DSP_COS register; When calculating arctan, the coordinates DSP_X/DSP_Y are used as input, and the angles theta = arctan (y/x) and module = sqrt (x^2+y^2) are calculated and output to the DSP_ARCTAN and DSP_MOD registers

15.2.3 DSP sin/cos Register

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

15.2.3.1 DSP_THETA

Address: 0x4001_5004

Reset value: 0x0

Table 15-5 DSP sin/cos Angle Input Register

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">THETA</td></tr>
<tr><td colspan="16">RW</td></tr>
<tr><td colspan="16">0</td></tr>
</table>

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

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

15.2.3.2 DSP_SIN

Address: 0x4001_5010

Reset value: 0x0

Table 15-6 DSP sin/cos Sine Result Register

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

SIN
RO
0

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

DSP_SIN is a 16-bit signed fixed-point number, including 1-bit sign bit, 1-bit integer bit, and 14-bit decimal bit; representing the range (−1 ~ 1).

15.2.3.3 DSP_COS

Address: 0x4001_5014

Reset value: 0x0

Table 15-7 DSP sin/cos Cosine Result Register

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

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

DSP_COS is a 16-bit signed fixed-point number, including 1-bit sign bit, 1-bit integer bit, and 14-bit decimal bit; representing the range (−1 ~ 1).

15.2.4 DSP arctan Register

15.2.4.1 DSP_X

Address: 0x4001_5008

Reset value: 0x0

Table 15-8 DSP Arctan/Module Coordinate X Input Register

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

RW
0

Location	Bit name	Description
[31:16]		Reserved bit. Sign extension when reading, ie {16{DSP_X[15]}}
[15:0]	X	DSP arctan/module calculate input coordinates X register

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

15.2.4.2 DSP_Y

Address: 0x4001_500C

Reset value: 0x0

Table 15-9 DSP Arctan/Module Calculate Coordinate Y Input Register

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

Location	Bit name	Description
[31:16]		Reserved bit. Sign extension when reading, ie {16{DSP_Y[15]}}
[15:0]	Y	DSP arctan/module calculate input coordinates Y register

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

15.2.4.3 DSP_MOD

Address: 0x4001_5018

Reset value: 0x0

Table 15-10 DSP Arctan Vector Modulus Result SQRT (X^2+Y^2) Register

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

Location	Bit name	Description
[31:16]		Reserved bit. Always read as 0.
[15:0]	MOD	DSP arctan calculation result sqrt(X^2+Y^2) Register

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

15.2.4.4 DSP_ARCTAN

Address: 0x4001_501C

Reset value: 0x0

Table 15-11 DSP Angle Result (Y/X) Register

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

Location	Bit name	Description
[31:16]		Reserved bit. Sign extension when reading, ie {16{DSP_ARCTAN[15]}}
[15:0]	ARCTAN	DSP Angle Result (Y/X) Register

DSP_ARCTAN is a 16-bit signed fixed-point number, representing the range (-32768 ~ 32767) corresponds to (−π ~ π).

15.2.5 DSP Divider Register

15.2.5.1 DSP_DID

Address: 0x4001_5020

Reset value: 0x0

Table 15-12 DSP Divider Register for Dividend

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

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

0

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

15.2.5.2 DSP_DIS

Address: 0x4001_5024

Reset value: 0x0

Table 15-13 DSP Divider Register for Divisor

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

Location	Bit name	Description
[31:16]		Reserved bit. Sign extension when reading, ie {16{DSP_DIS[15]}}
[15:0]	DIS	DSP Divider Register for Divisor

15.2.5.3 DSP_QUO

Address: 0x4001_5028

Reset value: 0x0

Table 15-14 DSP Division Quotient Register

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	DSP division quotient register

15.2.5.4 DSP_REM

Address: 0x4001_502C

Reset value: 0x0

Table 15-15 DSP Division Remainder Register

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

Location	Permission	Description
[31:16]		Reserved bit. Sign extension when reading, ie {16{DSP_REM[15]}}
[15:0]	REM	DSP division remainder register

When a DSP divider is required for the CPU, make sure that that the DSP is in a suspended state. Write the dividend to DSP, and then write the divisor. Writting a divisor can trigger a division calculation; The 32-bit and 16-bit division could finish in 8 cycles. While reading the division calculation result, DSP_SQRT or DSP_REM, the CPU will enter a wait state. The calculation result will be returned through the bus after the division calculation is done.

15.2.6 DSP SQRT Register

15.2.6.1 DSP_RAD

Address: 0x4001_5030

Reset value: 0x0

Table 15-16 DSP SQRT Register for Radicand

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

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

Location	Bit name	Description

[31:0]	RAD	DSP SQRT Register for Radicand

15.2.6.2 DSP_SQRT

Address: 0x4001_5034

Reset value: 0x0

Table 15-17 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

When a DSP SQRT controller is required for the CPU, make sure that that the DSP is in a suspended state. Write radicand to DSP to trigger an SQRT calculation; The 32-bit SQRT calculation could finish in 8 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.

15.3 DSP Instruction Set

15.3.1 Instruction Set Summary

Operation	Description	Assembler	Cycles
Add		ADD Rd1 Rs1 Rs2	1
	5bit Immediate	ADDI Rd1 Rs1 #<Imm>*[1]	1
Subtract		SUB Rd1 Rs1 Rs2	1
Shift	Arithmetic right shift	ASR Rd1 Rs1 Rs2	1
	5bit Immediate	ASRI Rd1 Rs1 #<Imm>	1
	Logical left shift	LSL Rd1 Rs1 Rs2	1
	5bit Immediate	LSLI Rd1 Rs1 #<Imm>	1
Multiply and accumulation		MAC Rs1 Rs2 Rs3	1
	5bit Immediate	MACI Rs1 Rs2 #<Imm>	1
Divide		DIV Rd1 Rs1 Rs2	10
Saturation		SAT Rd1 Rs1 Rs2	1
	4bit Immediate	SATI Rd1 #<Imm1> #<Imm2>	1

Cordic	SIN/COS	SIN_COS Rd1 Rd2 Rs1	8
	Arctan/Module	ARCTAN Rd1 Rs1 Rs2	8
Square root	Square root	SQRT Rd1 Rs1	8
Memory access	Load word	LDRWI Rd1 #<Imm>	1
	Load double half words	LDRDHI Rd1 Rd2 #<Imm>	1
	Store word	STRWI Rs1 #<Imm>	1
	Store double half words	STRDHI Rs1 Rs2 #<Imm>	1
Branch	Unconditional Jump	JUMP Rs1	2
	Immediate	JUMPI #<Imm>	2
	Jump if less than or equal to	JLE Rs1 Rs2 Rs3	2
	Immediate	JLEI Rs1 Rs2 #<Imm>	2
Miscellaneous	Generate IRQ and Pause DSP	IRQ	1

DSP uses 16bit fixed-length coding instructions. Since there are eight general registers, the register code should be 3-bit. Most of the instructions are 3-operand instructions, including two source operand registers and one destination operand register; some instructions contain immediate operand; some instructions involve four operands. Take the multiply-add (MAC) operation as an example. Rd = Rs1*Rs2+Rs3. Since the instruction length is not enough to indicate 4 registers, the Rd is fixed to R7 and is not displayed in the instruction code. The remaining 4 operand instructions also include ARCTAN/DIV. For specific operand assignment, please see the detailed explanation of the instructions below.

15.3.2 ADD

15.3.2.1 Encoding

15	14	13	12	11	10	9	8 7 6	5 4 3	2 1 0
0	0	0	0	0	0	0	Rs2	Rd1	Rs1

15.3.2.2 Assembly Syntax

ADD Rd1 Rs1 Rs2

15.3.2.3 Pseudocode

Rd1 = Rs1+Rs2

Result anti-overflow protection,

After overflow, Rd1 = 0x7FFF_FFFF,

After underflow, Rd1 = 0x8000_0000

15.3.3 ADDI (reserved)

The addition instruction with immediate operand is reserved in this version of DSP and is not implemented.

15.3.3.1 Instruction Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	Imm					Rd1			Rs1		

The immediate operand is a 5-bit signed number, representing the range of -16 ~ 15.

15.3.3.2 Assembly Syntax

ADDI Rd1 Rs1 Imm

15.3.3.3 Pseudocode

Rd1 = Rs1+Imm

Result anti-overflow protection,

After overflow, Rd1 = 0x7FFF_FFFF,

After underflow, Rd1 = 0x8000_0000

15.3.4 SUB

15.3.4.1 Instruction Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	0	0	Rs2			Rd1			Rs1		

15.3.4.2 Assembly Syntax

SUB Rd1 Rs1 Rs2

15.3.4.3 Pseudocode

Rd1 = Rs1 – Rs2

Result anti-overflow protection,

After overflow, Rd1 = 0x7FFF_FFFF,

After underflow, Rd1 = 0x8000_0000

15.3.5 ASR

15.3.5.1 Instruction Encoding

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td colspan="3">Rs2</td><td colspan="3">Rd1</td><td colspan="3">Rs1</td></tr>
</table>

15.3.5.2 Assembly Syntax

ASR Rd1 Rs1 Rs2

15.3.5.3 Pseudocode

Rd1 = Rs1 >> Rs2

The arithmetic right shift instruction only supports the right shift of 0 ~ 31 bits.

15.3.6 ASRI

15.3.6.1 Instruction Encoding

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td colspan="5">Imm</td><td colspan="3">Rd1</td><td colspan="3">Rs1</td></tr>
</table>

The immediate operand is a 5-bit unsigned number, representing the range of 0 ~ 31.

15.3.6.2 Assembly Syntax

ASRI Rd1 Rs1 Imm

15.3.6.3 Pseudocode

Rd1 = Rs1 >> Imm

The arithmetic right shift instruction with immediate operand only supports the right shift of 0 ~

31 bits.

15.3.7 LSL

15.3.7.1 Instruction Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	0	0	Rs2			Rd1			Rs1		

15.3.7.2 Assembly Syntax

LSL Rd1 Rs1 Rs2

15.3.7.3 Pseudocode

Rd1 = Rs1 << Rs2

Logic shift left only supports 0 ~ 31 bit shift left.

15.3.8 LSLI

15.3.8.1 Instruction Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	Imm					Rd1			Rs1		

The immediate operand is a 5-bit unsigned number, representing the range of 0 ~ 31.

15.3.8.2 Assembly Syntax

LSLI Rd1 Rs1 Imm

15.3.8.3 Pseudocode

Rd1 = Rs1 << Imm

The logical left shift with immediate operand only supports 0 ~ 31 bit left shift.

15.3.9 MAC

15.3.9.1 Instruction Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	Rs3			Rs2			Rs1		

15.3.9.2 Assembly Syntax

MAC Rs1 Rs2 Rs3

15.3.9.3 Pseudocode

Rd7 = Rs1 × Rs2+Rs3

Result anti-overflow protection,

After overflow, Rd7 = 0x7FFF_FFFF;

After underflow, Rd7 = 0x8000_0000.

Rs1 and Rs2 are 16-bit signed numbers, and Rs3 is a 32-bit signed number. Note that Rs1 and Rs2 as operands should not exceed the range of 16-bit signed numbers. When Rs3 is R0, MAC can be used as a multiplication instruction (MUL).

15.3.10 MACI (reserved)

Multiply-add (MAC) instructions with immediate operands are reserved in this version of DSP and are not implemented.

15.3.10.1 Instruction Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	Imm					Rs2			Rs1		

The immediate operand is a 5-bit signed number, representing the range of -16 ~ 15.

15.3.10.2 Assembly Syntax

MACI Rs1 Rs2 Imm

15.3.10.3 Pseudocode

Rd7 = Rs1 × Rs2+Imm

Result anti-overflow protection,

After overflow, Rd7 = 0x7FFF_FFFF;

After underflow, Rd7 = 0x8000_0000.

15.3.11 DIV

15.3.11.1 Instruction Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	0	0	Rd2			Rd1			Rs1		

15.3.11.2 Assembly Syntax

DIV Rd2 Rd1 Rs1

15.3.11.3 Pseudocode

Rd1 = Rd7/Rs1, Rd2 = Rd7 % Rs1

The division instruction takes 10 cycles to complete the submission. During the calculation of the division, the DSP should not issue other multi-cycle instructions, that is, only one multi-cycle instruction can be in the long-pipeline at a time. Other multi-cycle instructions include trigonometric function instructions and SQRT instructions. Multi-cycle instructions can run in the background, that is, while the multi-cycle instructions are operating, the DSP can execute other single-cycle instructions at the same time. However, only one multi-cycle instruction can run in the background at the same time. During the multi-cycle instruction calculation, the DSP can still use the destination operand of the multi-cycle instruction. However, it should be noted that the destination operand register will be rewritten when the multi-cycle calculation result is submitted.

Rd7 is a 32-bit signed number, Rs1 is a 16-bit signed number, Rd1 is a 32-bit signed number, and Rd2 is a 16-bit signed number.

15.3.12 SAT

15.3.12.1 Instruction Encoding

1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0

5	4	3	2	1	0										
0	1	1	0	0	0	0	Rs2			Rd1			Rs1		

15.3.12.2 Assembly Syntax

SAT Rd1 Rs1 Rs2

15.3.12.3 Pseudocode

If (Rd1<Rs1) Rd1=Rs1; else if (Rd1>Rs2) Rd1=Rs2

15.3.13 SATI (reserved)

Saturated instructions with immediate operands are reserved in this version of DSP and are not implemented.

15.3.13.1 Instruction Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	Imm2				Imm1				Rs1		

15.3.13.2 Assembly Syntax

SATI Rd1 Imm1 Imm2

15.3.13.3 Pseudocode

If (Rd1<Imm1) Rd1=Imm1; else if (Rd1>Imm2) Rd1=Imm2

15.3.14 SIN_COS

15.3.14.1 Instruction Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	0	0	Rd2			Rd1			Rs1		

15.3.14.2 Assembly Syntax

SIN_COS Rd1 Rd2 Rs1

15.3.14.3 Pseudocode

The sin/cos instruction ends and the calculation result is submitted within 8 cycles. During its execution, the DSP should not issue other multi-cycle instructions.

Rd1=cos (Rs1); Rd2=sin (Rs1)

15.3.15 ARCTAN

15.3.15.1 Instruction Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	0	0	Rs2			Rd1			Rs1		

15.3.15.2 Assembly Syntax

ARCTAN Rd1 Rs1 Rs2

15.3.15.3 Pseudocode

The ARCTAN instruction ends and the calculation result is submitted within 8 cycles. During its execution, the DSP should not issue other multi-cycle instructions.

Rd1= arctan(Rs2/Rs1); R6 = sqrt(Rs1^2+Rs2^2)

15.3.16 SQRT

15.3.16.1 Instruction Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	0	0	0	0	0	0	Rd1			Rs1		

15.3.16.2 Assembly Syntax

SQRT Rd1 Rs1

15.3.16.3 Pseudocode

The SQRT instruction ends and the calculation result is submitted within 8 cycles. During its execution, the DSP should not issue other multi-cycle instructions.

Rd1 = sqrt(Rs1)

15.3.17 LDRWI

15.3.17.1 Instruction Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	Imm						Rd1			0	0	0

15.3.17.2 Assembly Syntax

LDRWI Rd1 Imm

15.3.17.3 Pseudocode

Rd1=word(SRAM[imm])

Since the load instructions are all immediate instructions, the data address accessed can be generated during the decoding stage, so the load operation can be completed in one cycle. For similar load instructions, the CPU should access the register to calculate the address, which may take 2 cycles to complete.

Since DSP data MEM is composed of 64 32bit words, The range of immediate operand is 0 ~ 63.

15.3.18 LDRDHI

15.3.18.1 Instruction Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	Imm						Rd1			Rd2		

15.3.18.2 Assembly Syntax

LDRDHI Rd1 Rd2 Imm

15.3.18.3 Pseudocode

LDRDHI Rd1 Rd2

{Rd1,Rd2}= word(SRAM[Imm]),

The 32-bit data retrieved from DATA MEM will be sign-extended to 32 bits and then assigned to Rd1.

The lower 16 bits will be sign-extended to 32 bits and then assigned to Rd2.

Since DSP data MEM is composed of 64 32bit words, The range of immediate operand is 0 ~ 63.

15.3.19 STRWI

15.3.19.1 Instruction Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	Imm						0	0	0	Rs1		

15.3.19.2 Assembly Syntax

STRWI Rs1 Imm

15.3.19.3 Pseudocode

word{SRAM[imm]}=Rs1

The data of Store instructions comes from the register, so the Store operation cannot be completed immediately even if the address to be written is generated in the decoding stage. It's necessary to latch the address for one cycle, and then sent to the data memory interface together with the written data. Therefore, if the Store instruction is connected to the load instruction immediately after then, an access violation will occur. This sequence of instructions should be avoided when designing the assembler. If the STR instruction must be followed by the LDR instruction, the ADD R0 R0 R0 instruction can be inserted between the two as the instruction bubble.

Since DSP data MEM is composed of 64 32bit words, The range of immediate operand is 0 ~ 63.

15.3.20 STRDHI (reserved)

The store double halfword instruction with immediate operand is reserved in this version of DSP and is not implemented.

15.3.20.1 Instruction Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	Imm						Rs1			Rs2		

15.3.20.2 Assembly Syntax

STRDHI Rs1 Rs2 Imm

15.3.20.3 Pseudocode

word{SRAM[imm]}={Rs1, Rs2}

Since DSP data MEM is composed of 64 32bit words, The range of immediate operand is 0 ~ 63.

15.3.21 JUMP (reserved)

Register-based jump instructions are reserved in this version of DSP and are not implemented.

15.3.21.1 Instruction Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	0	0	0	0	0	0	0	0	Rs1		

15.3.21.2 Assembly Syntax

JUMP Rs1

15.3.21.3 Pseudocode

PC = PC+1+Rs1

15.3.22 JUMPI

15.3.22.1 Instruction Encoding

1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0

5	4	3	2	1	0	
1	1	0	0	1	0	Imm

15.3.22.2 Assembly Syntax

JUMPI Imm

15.3.22.3 Pseudocode

PC = PC+1+IMM

15.3.23 JLE

15.3.23.1 Instruction Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	0	0	0	Rs3			Rs2			Rs1		

15.3.23.2 Assembly Syntax

JLE Rs1 Rs2 Rs3

15.3.23.3 Pseudocode

PC = PC+1+Rs3, if (Rs1 <= Rs2)

15.3.24 JLEI

15.3.24.1 Instruction Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	1	Imm					Rs2			Rs1		

15.3.24.2 Assembly Syntax

JLEI Rs1 Rs2 Imm

15.3.24.3 Pseudocode

PC = PC+1+IMM, if (Rs1 <= Rs2)

15.3.25 IRQ

15.3.25.1 Instruction Encoding

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

15.3.25.2 Assembly Syntax

IRQ

15.3.25.3 Pseudocode

The IRQ instruction will generate an interrupt, wait for the CPU to process, and suspend the DSP at the same time.

15.3.26 R (Only for analog converter)

15.3.26.1 Instruction Encoding

N/A

15.3.26.2 Assembly Syntax

```
R5 0x5555        # Assign 0x5555 to R5
R1 20            # Assign 20 to R1
```

15.3.26.3 Pseudocode

Debug command. Only used in DSP Emulator. It can be used to set R1 ~ R7 to any specified value at any position in the DSP program.

15.3.27 BREAK(Only for analog converter)

15.3.27.1 Instruction Encoding

N/A

15.3.27.2 Assembly Syntax

BREAK

15.3.27.3 Pseudocode

Debug command. Only used in DSP Emulator. It can be used to insert a breakpoint at any position in the DSP program and print the R1 ~ R7 register value. After hitting the breakpoint, press Enter to continue the program.

15.3.28 END (Only for analog converter)

15.3.28.1 Instruction Encoding

N/A

15.3.28.2 Assembly Syntax

END

15.3.28.3 Pseudocode

Similar to breakpoint instruction. Only used in DSP Emulator. When the instruction appears in the instruction simulation, the operation of the simulator is interrupted, and the core register value is printed directly.

15.4 Application Guide

15.4.1 Memory Addressing

DSP immediate operand addressing. Due to the limited DATA MEM address space, the 6-bit immediate operand in the STR and LDR instructions can directly represent the entire address offset of the DATA MEM.

Take the following DATA MEM content as an example, the first row of data 0x01000100 corresponds to 0x0 address, the second row of data 0x30005000 corresponds to 0x1 address, and the third row of data 0x0003FFF8 corresponds to 0x2 address.

Use LDRDHI R1 R2 0x1 to assign values to R1 and R2, R1 = 0x3000, R2 = 0x5000

Use STRWI R3 0x3 to write 32bit data of R3 to address 0x3 and overwrite data 0xFFFFE000.

Although the DSP memory access addressing increases by 1 for each increase of 4Byte, for the CPU, each additional address increases by 4Byte. Therefore, when the CPU accesses the DSP DATA MEM, it should be calculated in the manner of DSP_DATA_MEM_BASE+offset*4. Take the following

DATA MEM content as an example, the CPU addressing address corresponding to the second row of data 0x30005000 is 0x4001_4804, and the CPU addressing address corresponding to the third row of data 0x0003FFF8 is 0x4001_4808.

DATA MEM:

0x00100010

0x30005000

0x0003FFF8

0xFFFFE000

0x30004000

0x7FFFFFFF

0x7FFFFFFF

0xF0000003

0x00007FFF # 8

0x00008000

0x80000000

0x00000000

15.4.2 Load after Store

The LDR cannot be used immediately after the DSP STR instruction, but can insert "ADD R0 R0 R0" between them as a "NOP" instruction.

15.4.3 Delayed Submission of Multi-cycle Instructions

DSP's multi-cycle arithmetic instructions include division, square root, and trigonometric functions.

The division instruction could be finished in 10 bus cycles (96MHz).

SQRT instructions could be finished in 8 bus cycles (96MHz).

Trigonometric function instructions could be finished in 8 bus cycles (96MHz).

In order to make full use of DSP performance, multi-cycle instruction background execution is allowed, that is, the DSP can still use other instructions during the calculation of the multi-cycle instruction without blocking the pipeline. As for this, the instruction sequence when programming should be taken into account, and insert other unrelated instructions between the issue of the multi-cycle instruction and the use of the result. The following instruction sequence, in which ADD R0 R0 R0 acts like NOP, can be replaced with other instructions in practical applications.

SIN_COS R1 R2 R3

ADD R0 R0 R0

ADD R0 R0 R0

ADD R0 R0 R0

ADD R0 R0 R0

ADD R0 R0 R0

ADD R0 R0 R0

ADD R0 R0 R0

LDRWI R4 0x10

MAC R1 R4 R0

15.4.4 Data Filling

DATA MEM supports writing by word, halfword and byte. In order to solve the problem of reading immediately after the AHB bus protocol is written, there is a word depth cache at the SRAM port. DSP CODE MEM and DATA MEM have consistency problems since two main devices, CPU and DSP, are allowed to access.. For example, as shown in Fig. 15-2. The CPU writes 4 words to DSP DATA MEM in sequence, and then reads the last word. Due to the existence of the cache, the last word is left in the cache without writing to the DSP DATA MEM. However, when the CPU reads this word data, it hits the cache, and the data is returned directly from the cache, so the data read back is correct. When the DSP reads the DATA MEM, it can only read the first 3 words, and the last word left in the cache is not accessible.

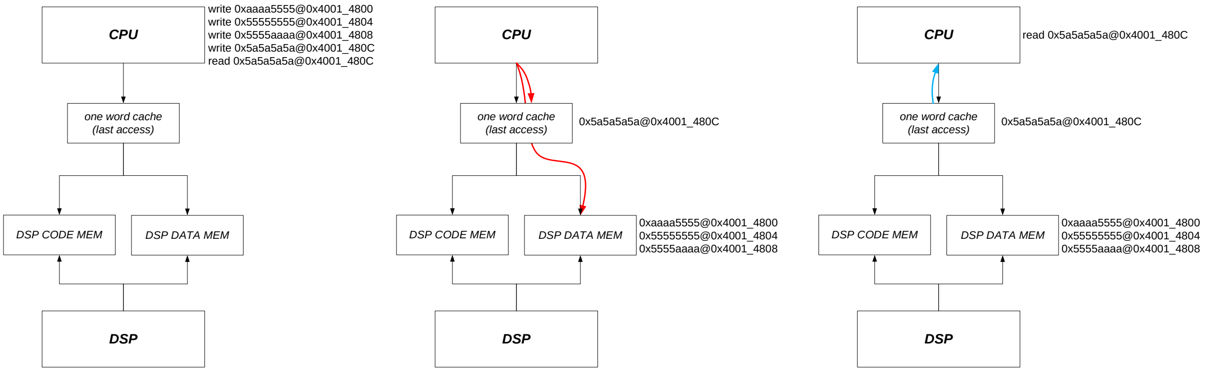


Fig. 15-2 Consistency of CPU Accessing DSP MEM

Therefore, it is required to write any other address after the last write, so that the data staying in the cache is truly written into the DSP MEM. For a system with a single master device, all read and write operations pass through the cache on the interface, so there will be no consistency problems; Since the MEM in the DSP is written by the CPU and accessed by the DSP through another channel, it is necessary to implement cache data push through a redundant write operation, as shown in Fig. 15-3.

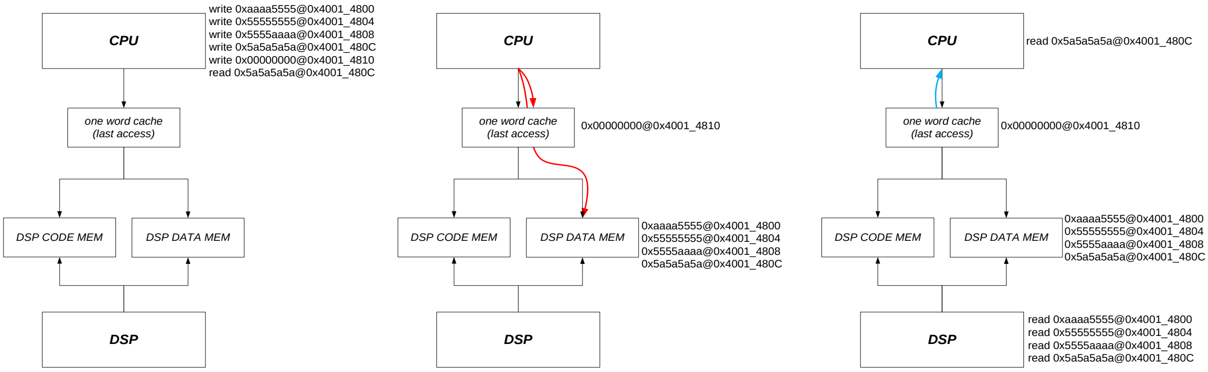


Fig. 15-3 Add Dummy Write Operation to Solve the Consistency Problem of DSP MEM

Since the address signal of the CPU's last read and write will remain in the Cache, if the DSP changes the DATA MEM data during operation, then the CPU reads the DSP DATA MEM data and hits the last R/W address, the CPU will fetch the data from the Cache, instead of DSP DATA MEM, that is, the CPU thinks the cache data is still valid. Therefore, the application should write dummy data to the CPU for the last time to avoid this situation.

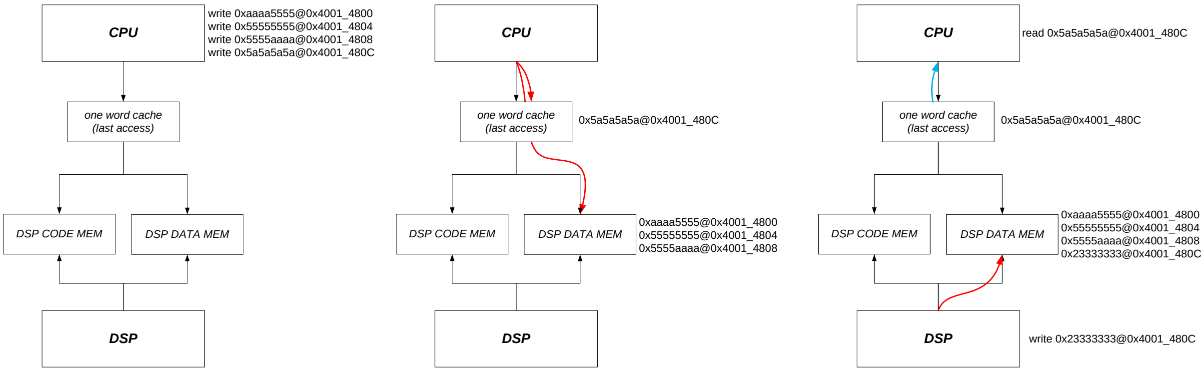


Fig. 15-4 CPU Failed to Read Real Data of DSP DATA MEM due to Hitting Cache

16 I2C

16.1 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, and can use DMA to reduce the burden of MCU according to the needs of specific equipment.

16.2 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.

16.3 Functional Description

16.3.1 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.

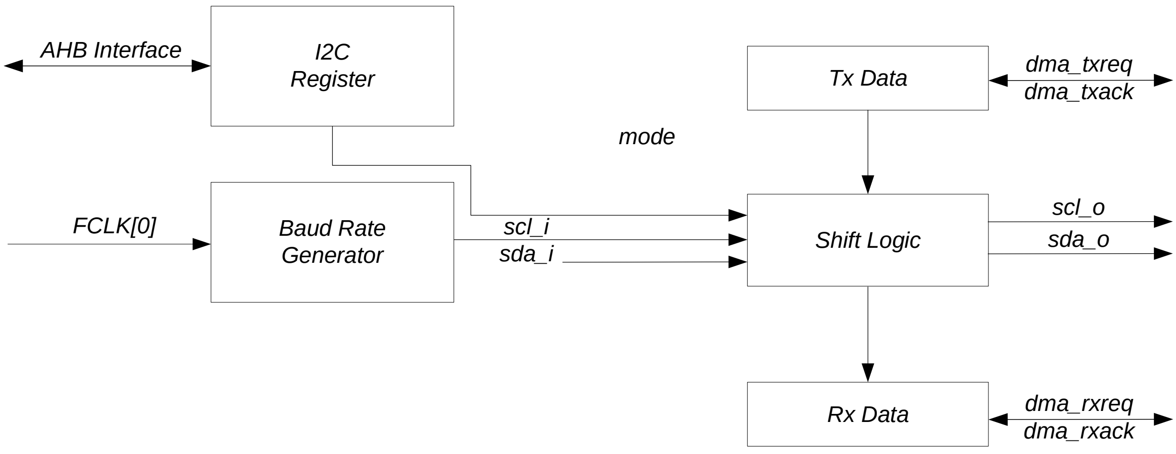


Fig. 16-1 I2C Module Top Functional Block Diagram

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.

16.3.2 Pin Function Description

The I2C module receives and sends data, and converts the data from serial to parallel, or parallel to serial , and can enable or disable interrupts. The interface is connected to the I2C bus via data pins (SDA) and clock pins (SCL).

16.3.2.1 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 is 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.

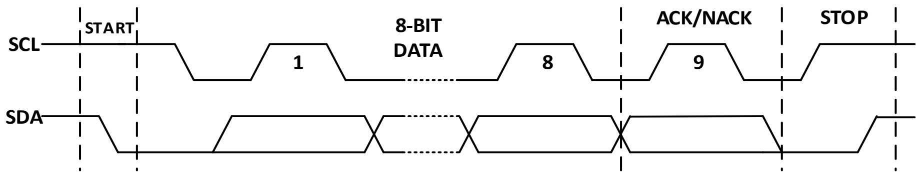


Fig. 16-2 Basic I2C Transmission Timing Diagram

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 16 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 mastermode, the I2C interface will release the bus after detecting an error, restore to the initial state and generate an interrupt signal.

16.3.2.2 I2C Interface 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.

16.3.2.2.1 Slave 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. 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 receving the reponse (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.

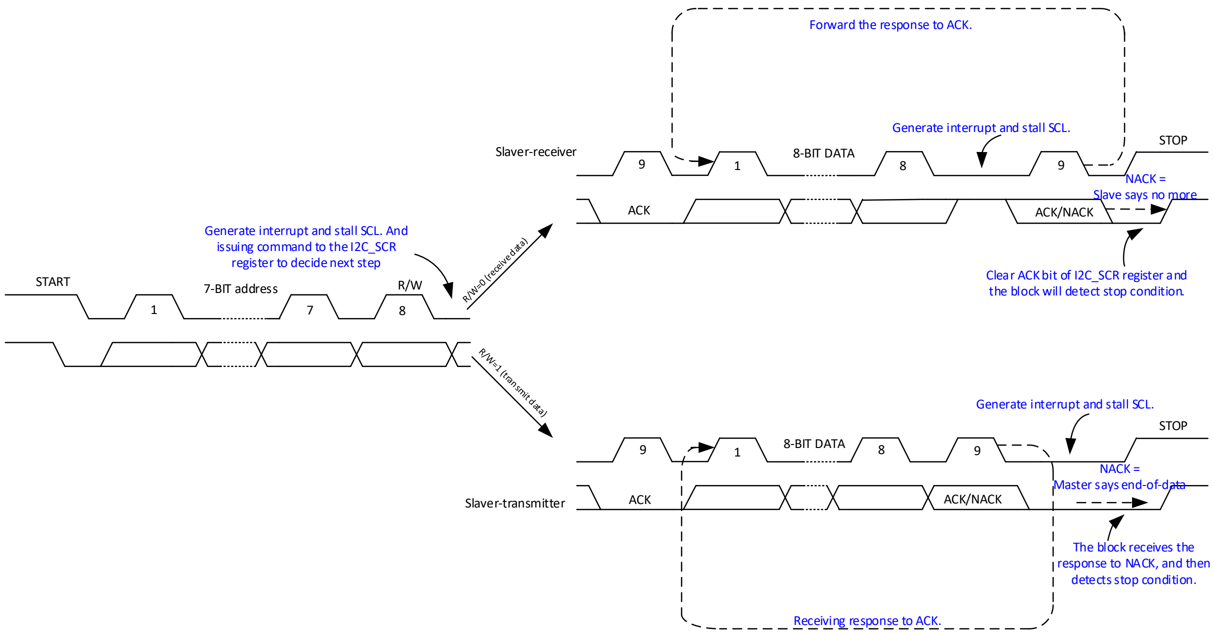


Fig. 16-3 Schematic Diagram of Single-byte Transmission in Slave Mode

16.3.2.2.2 Slave Mode Single-byte 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.

16.3.2.2.3 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.

16.3.2.3 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.

16.3.2.3.1 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.

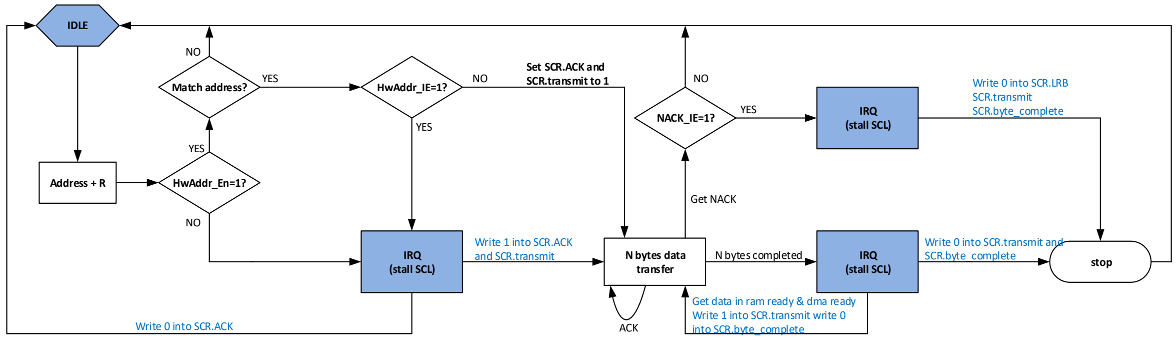


Fig. 16-4 Schematic Diagram of Multi-byte Transmission in Slave Mode

16.3.2.3.2 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.

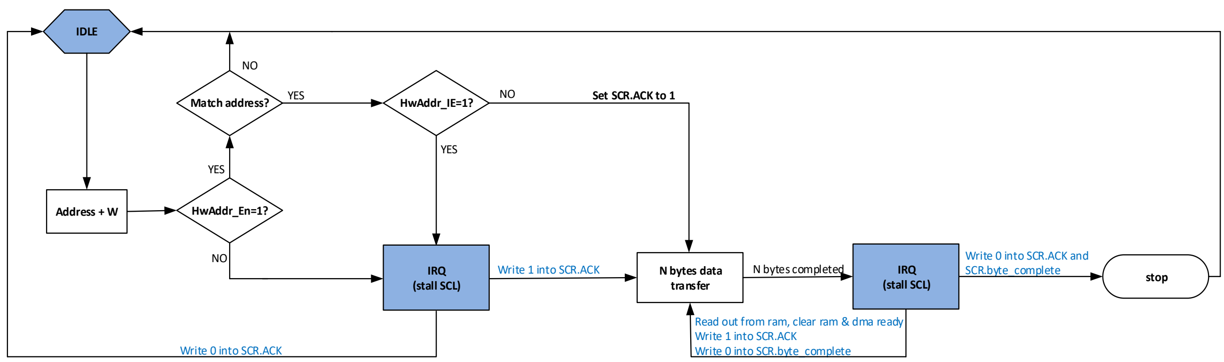


Fig. 16-5 Schematic Diagram of Multi-byte Reception in Slave Mode

16.3.2.4 I2C Interface 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.

16.3.2.4.1 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 receving the reponse (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.

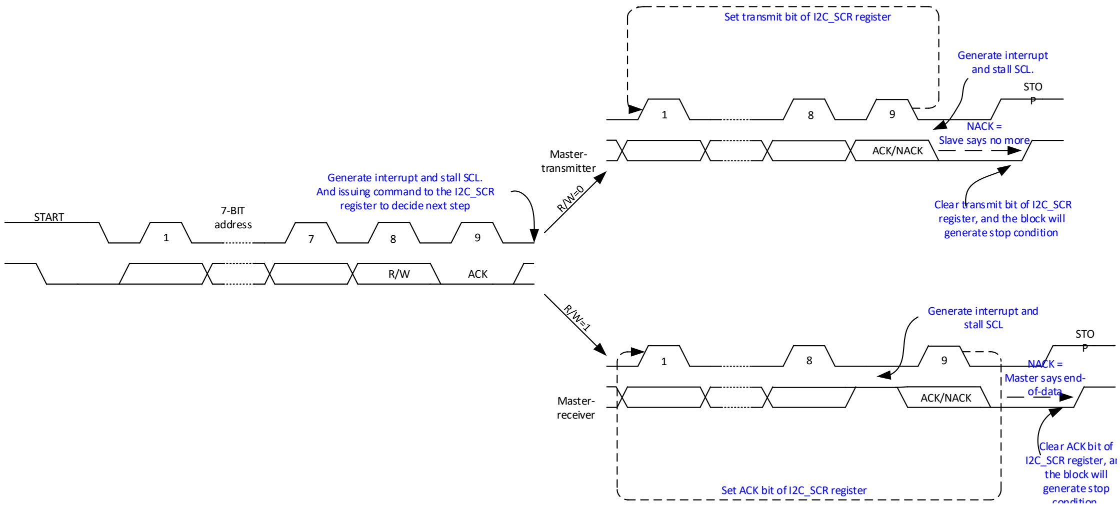


Fig. 16-6 Schematic Diagram of Single-byte Transmission in Master Mode

16.3.2.4.2 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.

16.3.2.4.3 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.

16.3.2.4.4 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.

16.3.2.4.5 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.

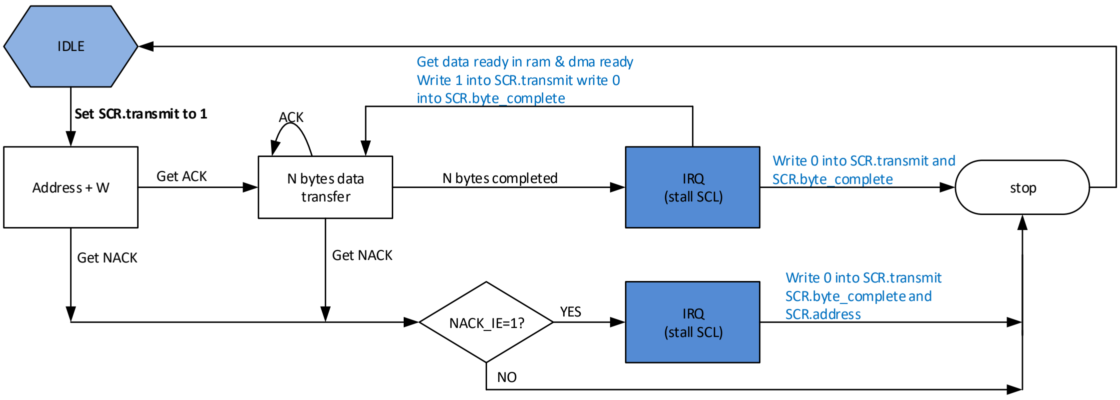


Fig. 16-7 Schematic Diagram of Multi-byte Transmission in Master Mode

16.3.2.4.6 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, then 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.

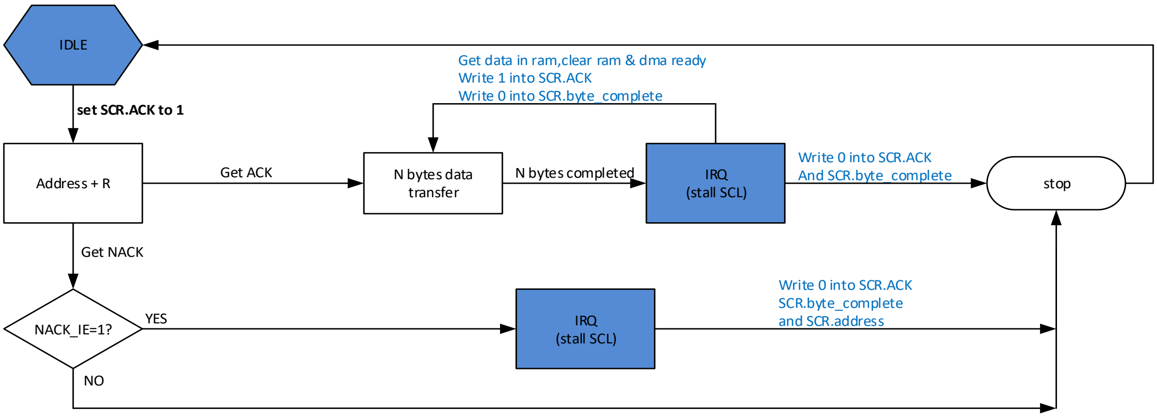


Fig. 16-8 Schematic Diagram of Multi-byte Reception in Master Mode

16.3.2.5 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.

16.3.2.6 DMA Transmission

Under the application of large-capacity data transmission, the I2C interface supports DMA transmission, thus 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 hardwaregenerates a DMA request automatically and moves the data from RAM to the I2C interface through the DMA module.

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

The I2C interface supports both DMA transmission and MCU transmission. The difference between the two is that the data sent by DMA transmission comes from the movement of DMA, and 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, set I2C_CFG, I2C_BCR 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 B [4] is the prefetching switch. Generally, the hardware address comparison (I2C_ADDR B [7]) can be turned on during transmission in the slave mode, and choose whether to enable the hardware address comparison interrupt (I2C_BCR B [6]).

- 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 B [4] 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.

16.3.2.7 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.

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

- Initialize the GPIO module and set the GPIO multiplexed with I2C.
- Initialize the I2C interface, set IE, CFG and other registers.
- The MCU triggers the I2C interface to enter the sending process. The data sent from the MCU is written to the I2C_DATA.

16.3.2.8 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.
- Bus error event. During the transmission process, the bus generates an erroneous START event/STOP event. Active high, write 0 to clear BIT7 of I2C_SCR.
- 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 BIT5 of I2C_SCR.
- NACK event. The sender receives a NACK response, indicating that the receiver cannot continue subsequent transmissions. Active high, write 0 to clear BIT1 of I2C_SCR.
- 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 BIT3 of I2C_SCR.
- Use DMA to assist data transfer. If the module is in receiving mode, I2C should move the data received to RAM by using the DMA. Then, it's necessary to check if the DMA data movement is completed. If the completion interrupt of I2C is used as the judgment basis, it is recommended to query the DMA status in the interrupt processing function. If this module is in transmission mode, there is no such problem , and the completion interrupt of I2C can be used directly as the basis for judgment.

16.3.2.9 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

16.4 Register

16.4.1 Address Allocation

The base address of the I2C module register is 0x4001_0400, and the register list is as follows:

Table 16-1 I2C Register Address Allocation List

Name	Offset	Description
I2C0_ADDR	0x00	I2C address register
I2C0_CFG	0x04	I2C configuration register
I2C0_SCR	0x08	I2C status register

I2C0_DATA	0x0C	I2C data register
I2C0_MSCR	0x10	I2C master mode register
I2C0_BCR	0x14	I2C DMA transfer control register

16.4.2 Register Description

16.4.2.1 Address Register (I2C0_ADDR)

Address: 0x4001_0400

Reset value: 0x0

Table 16-2 Address Register (I2C0_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. 1: enable 0: disable
[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; otherwise, the slave device address is written to the I2C_DATA register.

16.4.2.2 System Control Register (I2C0_CFG)

Address: 0x4001_0404

Reset value: 0x0

Table 16-3 System Control Register (I2C0_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

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

16.4.2.3 Status Control Register (I2C0_SCR)

Address: 0x4001_0408

Reset value: 0x0

Table 16-4 Status Control Register (I2C0_SCR)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								STT_ERR	LOST_ARB	STOP_EVT	BYTE_CMPLT	ADDR_DATA	DATA_DIR	RX_ACK	Done
								RW	RW	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

Location	Bit name	Description
[31:8]		Unused
[7]	STT_ERR	Bus error status flag, only used in master mode, write 0 to clear. 0: no START/STOP bus error 1: START/STOP bus error
[6]	LOST_ARB	Bus arbitration lost status flag bit. Only used in master mode, 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 modes can be used, write 0 to clear. 0: no STOP event 1: STOP event occurred
[4]	BYTE_CMPLT	ACK generation control bit. Both master and slave modes can be used. Configure the relocation before sending. In single-byte mode, the hardware is automatically cleared after the bytes are sent; in multi-byte mode, the hardware is automatically cleared after all the bytes are sent. 0: byte transmission is complete, return NACK response 1: byte transmission is completed, return ACK response
[3]	ADDR_DATA	Address data flag. Both master and slave modes can be used, 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. Both master and slave modes can be used, determine the direction of data transmission by this bit. This event can be cleared by the START event on the bus, and it can also be written to clear this bit. 0: Receive 1: triggered sent
[1]	RX_ACK	Receive response flag bit. Both master and slave modes can be used, the START event on the bus can clear this bit, and can also write 0 to clear this bit. 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 modes can be used, the START event on the bus can clear this bit, and can also write 0 to clear this bit. 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.

16.4.2.4 Data Register (I2C0_DATA)

Address: 0x4001_040C

Reset value: 0x0

Table 16-5 Data Register (2C0_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 mode can be used. When writing to the register, the data enters the I2C internal sending part, and the newly written data cannot be directly read. Read to get the data received by I2C. Master/slave receiving mode: store the received data. Single-byte transfers need to wait for the completion interrupt; for multi-byte transfers, the data will be moved to RAM by DMA, leaving only the last byte. Slave transmission mode: Store the transmitted data. Fill in the data first, and then trigger the BIT [2] of I2C_SCR. For multi-byte transfer, the data will be moved from RAM to I2C_DATA by DMA, and then sent by I2C interface. Master transmission mode: for single-byte transmission, I2C_DATA fills in the address; for multi-byte transmission, I2C_ADDR fills in the address, I2C_DATA data is transferred by DMA.

16.4.2.5 Main Mode Register (I2C0_MSCR)

Address: 0x4001_0410

Reset value: 0x0

Table 16-6 Main Mode Register (I2C0_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. 1: Detected START event, busy; 0: Detected STOP event, idle.
[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.

16.4.2.6 DMA Transmission Control Register (I2C0_BCR)

Address: 0x4001_0414

Reset value: 0x0

Table 16-7 DMA Transmission Control Register (I2C0_BCR)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								BURST_NACK	BURST_ADDR_CMP	BUSRT_EN		BURST_SIZE			
								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
[3:0]	BURST_SIZE	I2C data transmission length register. Used for multi-byte transmission. Actual bytes transferred = B [3: 0]+1

17 SPI

17.1 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.

17.2 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

17.3 Functional Description

17.3.1 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.

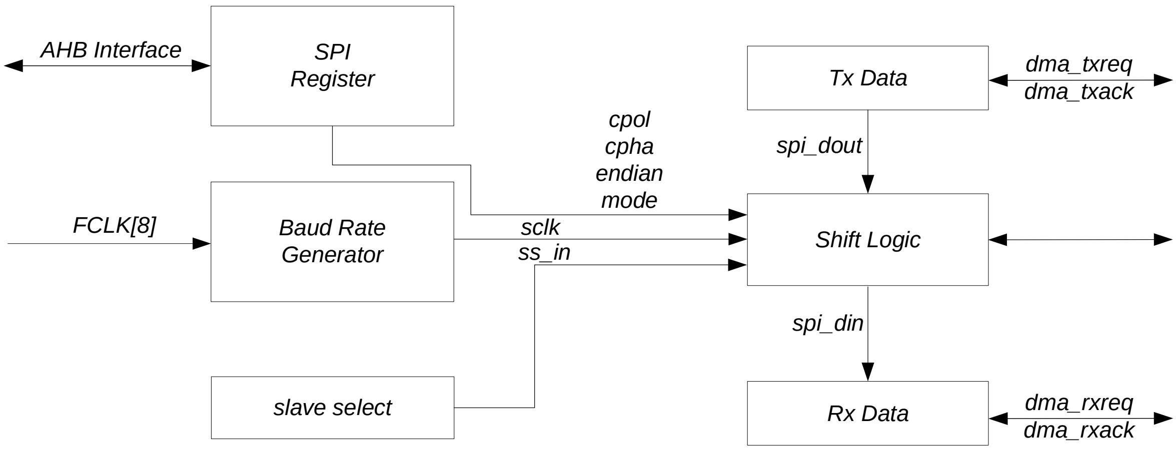


Fig. 17-1 SPI Module Structure Block Diagram

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.

sclk_out: clock signal sent by the interface. The working mode of the interface is Master, and in non-Master mode, the signal output is always 0.

ss_in: chip select signal received by the interface. The working mode of the interface is Slave, and this signal input is invalid in non-Slave mode.

ss_out: chip select signal sent by the interface. The working mode of the interface is Master, and this signal output is always 1 in non-Master mode.

17.3.2 Pin Function Description

17.3.2.1 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

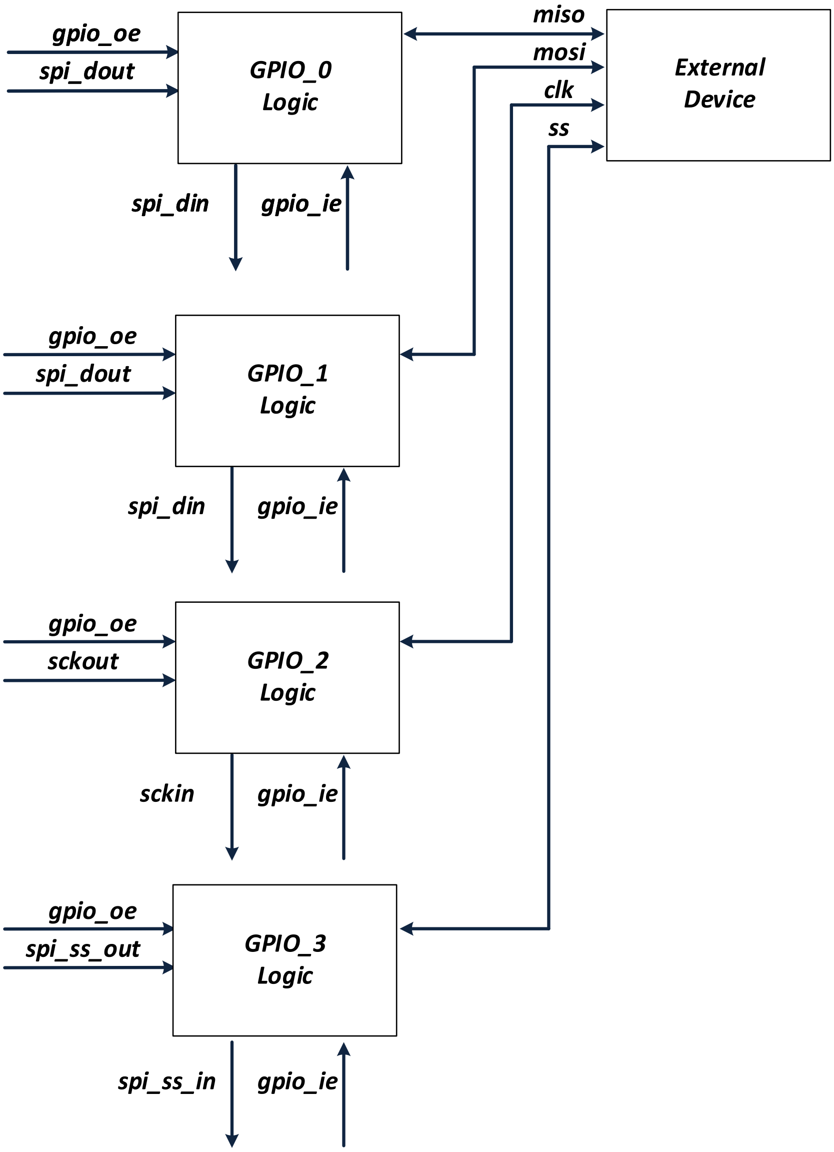


Fig. 17-2 SPI Interface Full Duplex Mode Interconnection Block Diagram

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.

17.3.2.2 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.

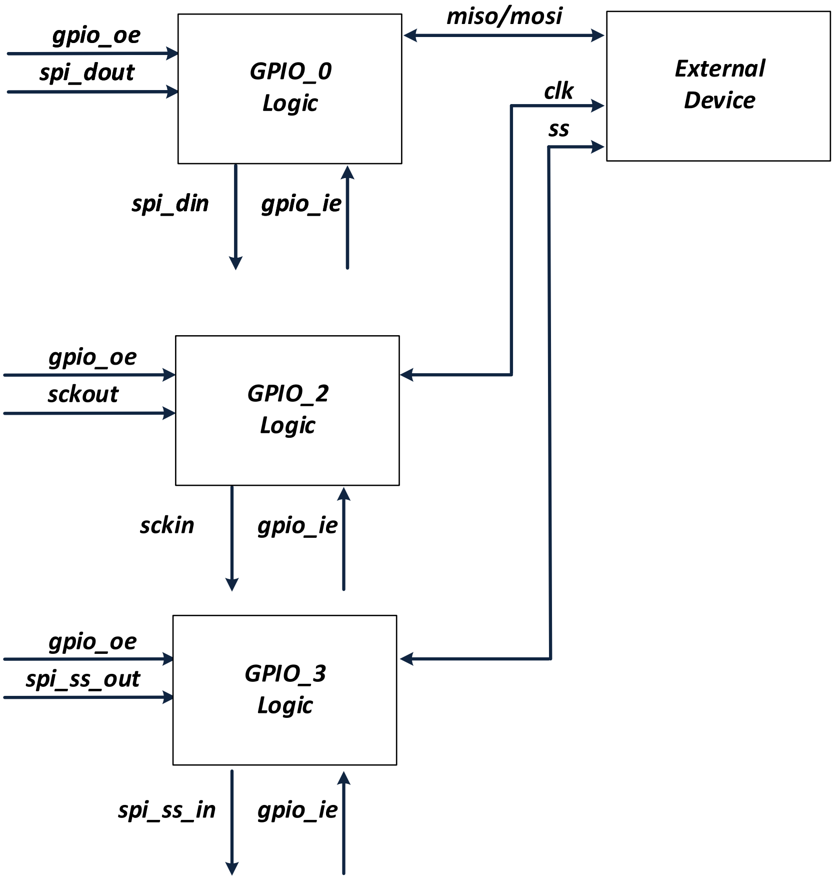


Fig. 17-3 SPI Interface Half-duplex Mode Interconnection Block Diagram

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

CFG [7: 6] 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 DMA transmission and supports sending in Master/Slave mode.

Receive only

CFG [7: 6] 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 DMA transmission and 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.

17.3.2.3 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.

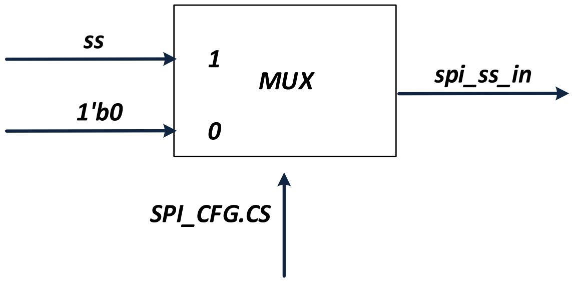


Fig. 17-4 SPI Module Chip Select Signal Selection in Slave Mode

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.

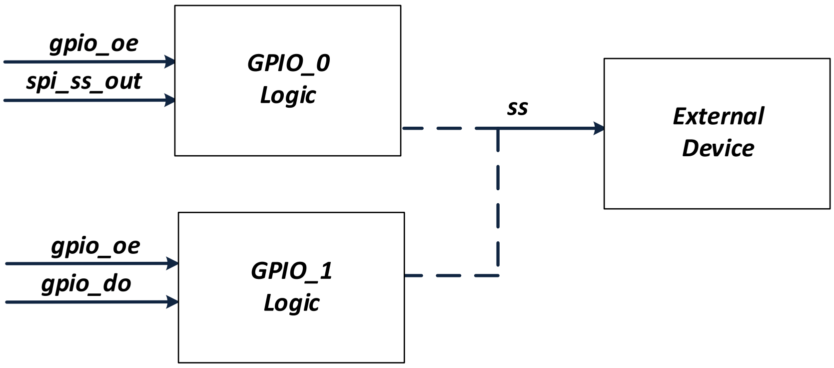


Fig. 17-5 SPI Module Chip Select Signal Selection in Master Mode

Note that the dotted line in Figure 14-5 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.

17.3.2.4 Communication Format

In the SPI communication process, the sending or receiving operation is based on the SPI clock. The communication format is controlled by CFG [3: 2]. CFG [3] is Phase control bit and CFG [2] 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.

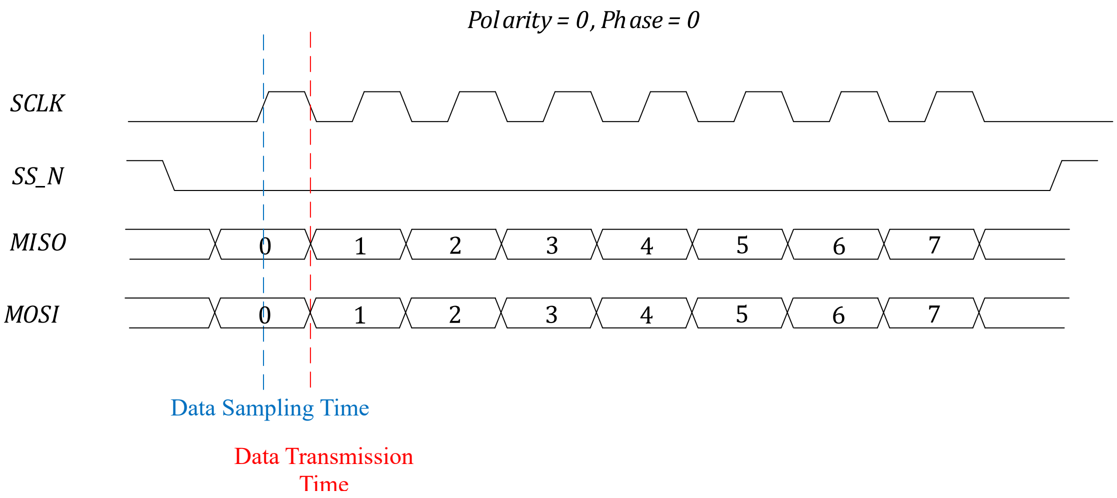


Fig. 17-6 SPI Communication Signal Polarity Phase (Polarity=0, Phase=0)

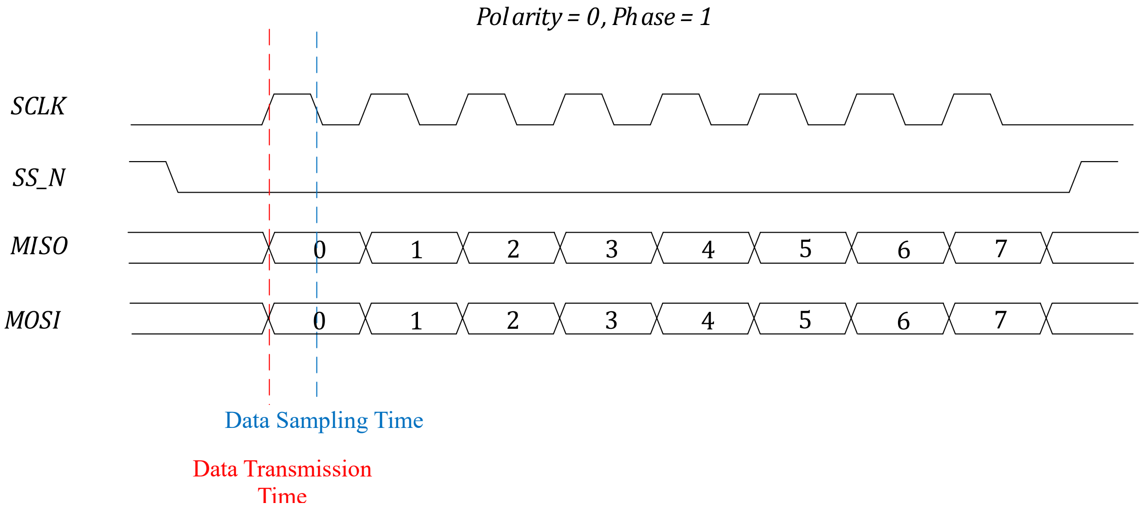


Fig. 17-7 SPI Communication Signal Polarity Phase (Polarity=0, Phase=1)

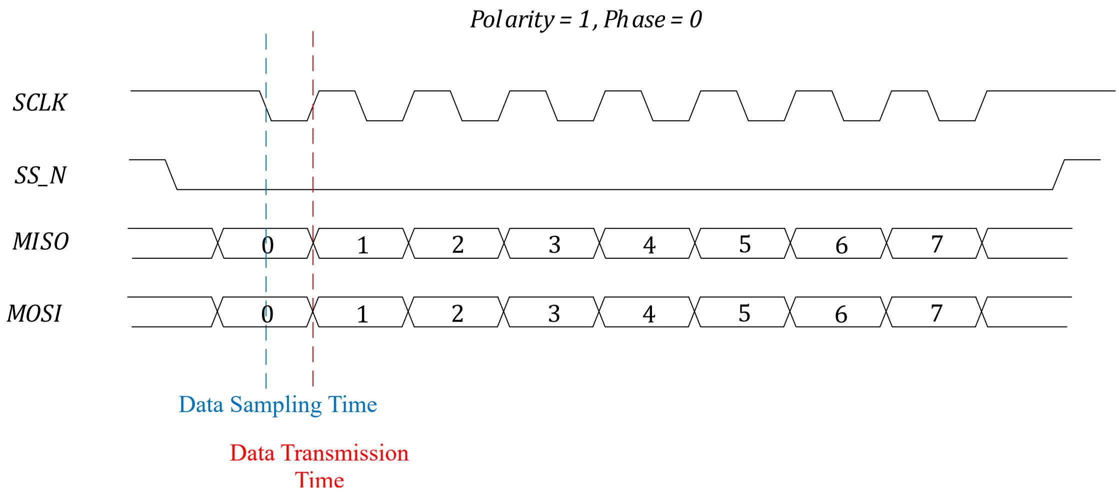


Fig. 17-8 SPI Communication Signal Polarity Phase (Polarity=1, Phase=0)

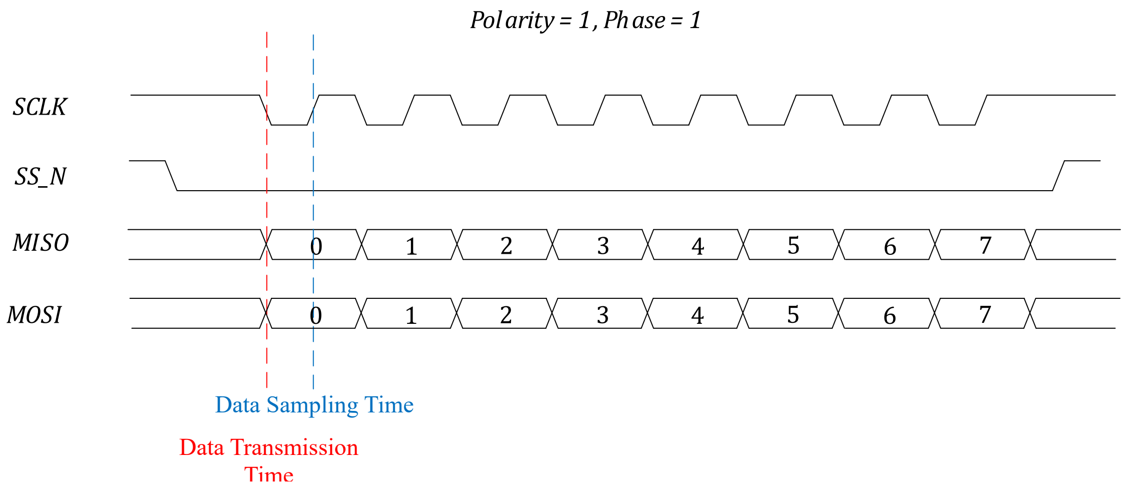


Fig. 17-9 SPI Communication Signal Polarity Phase (Polarity=1, Phase=1)

17.3.2.5 Data Format

SPI data transmission format is divided into two types: MSB and LSB. The data transmission format is controlled by CFG [1]. Note that the hardware automatically converts the transmission format during data transmission without software conversion.

17.3.2.6 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 is 255 bytes, 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 IE/CFG/BAUD/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.

17.3.2.7 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.

The SPI interface supports DMA transmission and MCU transmission. The SPI interface does not need to trigger the sending state during MCU transmission. As long as the MCU moves the data to the SPI interface, it can start sending data to the outside world.

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 IE/CFG/BAUD/SIZE and other registers. Note that SIZE can only be set as 1.
- The MCU performs a write operation on the TX_DATA register, triggering the SPI interface to enter the transmission process. The data sent from the MCU writes the value to TX_DATA.

Note: If continuous transmission is required, SIZE and TX_DATA registers should be set.[1]

17.3.2.8 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. IE [3] controls whether to use interrupt mode.

17.3.2.9 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 IE [0] 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 IE [1] to clear.
- Overflow event. If the data is not returned to RAM through DMA in time, or data is obtained from

1

RAM, an overflow event will occur. Active high, write 1 to IE [2] to clear.

The above events do not trigger the SPI interrupt by default, but can set IE [6: 4] to enable the event to generate an interrupt.

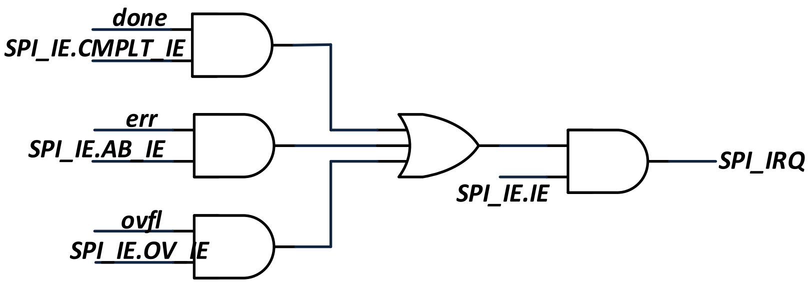


Fig. 17-10 Generation Diagram for SPI Module Interrupt Selection Signal

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.

17.3.2.10 Baud Rate Setting

The SPI interface clock is obtained by dividing the system clock by the frequency division factor from BAUD [5: 0]. The frequency division range is 1 to 128, and the corresponding BAUD [5: 0] values are 0 to 63.

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.

17.4 Register

17.4.1 Address Allocation

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

Table 17-1 List of SPI Module Control Register

Name	Offset	Description
SPI_CFG	0x00	SPI Configuration Register
SPI_IE	0x04	SPI Interrupt Register
SPI_DIV	0x08	SPI Baud Rate Setting Register
SPI_TX_DATA	0x0C	SPI Transmit Data Register
SPI_RX_DATA	0x10	SPI Receive Data Register
SPI_SIZE	0x14	SPI Transfer Data Length Register

17.4.2 System Control Register (SPI_CFG)

Address: 0x4001_0000

Reset value: 0x0

Table 17-2 System Control Register (SPI_CFG)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								DUPLEX		CS	MS	SAMPLE	CLK_POL	ENDIAN	EN
								RW		RW	RW	RW	RW	RW	RW
								0		0	0	0	0	0	0

Location	Bit name	Description
[31:8]		Unused
[7:6]	DUPLEX	Half-duplex mode setting 0X: turn off half-duplex mode 10: Turn on half-duplex mode, transmit only 11: Turn on half-duplex mode, receive only
[5]	CS	Source of chip select signal under SPI slave device. The default value is 1. 1: The chip select signal in Slave mode comes from the Master device 0: The chip select signal in Slave mode is always a valid value --0
[4]	MS	SPI master-slave mode selection. The default value is 0. 1: Master mode 0: Slave mode
[3]	SAMPLE	SPI phase selection. The default value is 0. 1: Phase is 1 0: Phase is 0
[2]	CLK_POL	SPI polarity selection. The default value is 0. 1: Polarity is 1 0: Polarity is 0
[1]	ENDIAN	SPI module transmission sequence. The default value is 0. 1: LSB, low bit is transmitted first 0: MSB, high bit is transmitted first

[0]	EN	SPI module enable signal. The default value is 0. 1: turn on the SPI module 0: turn off the SPI module

For SPI_CFG[3:2] corresponding communication waveform polarity and phase, please refer to 17.3.2.4.

17.4.3 Interrupt Register (SPI_IE SPI)

Address: 0x4001_0004

Reset value: 0x0

Table 17-3 Interrupt Register (SPI_IE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								IE	CMPLT_IE	AB_IE	OV_IE	TRANS_TRIG	CMPLT_IF	AB_IF	OV_IF
								RW	RW	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

Location	Bit name	Description
[31:8]		Unused
[7]	IE	SPI interrupt enable switch. The default value is 0. 1: enable SPI interrupt 0: disable SPI interrupt
[6]	CMPLT_IE	SPI transmission, complete event interrupt enable signal. 1: enable this interrupt source 0: disable this interrupt source
[5]	AB_IE	SPI transmission, abnormal event interrupt enable signal. 1: enable this interrupt source 0: disable this interrupt source
[4]	OV_IE	SPI transmission, interrupt enable signal for overflow event. The default value is 0. 1: enable this interrupt source 0: disable 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. There are two cases: 1. The old data received last time has not been taken away, the new data received this time has arrived. 2. Data has been sent, and the new data is not ready.

		Active high, write 1 to clear.

17.4.4 Baud Rate Setting Register (SPI_DIV)

Address: 0x4001_0008

Reset value: 0x0

Table 17-4 Baud Rate Setting Register (SPI_DIV)

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
[31:8]		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. SPI actual transmission speed calculation formula is: SPI transmission speed = system clock/(2*(BAUD+1)) Remember, the set value of BAUD cannot be less than 3.

17.4.5 SPI Transmit Data Register (SPI_TX_DATA)

Address: 0x4001_000C

Reset value: 0x0

Table 17-5 Transmit Data Register (SPI_TX_DATA)

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

Location	Bit name	Description
[31:8]		Unused
[5:0]	TX_DATA	SPI Transmit Data Register

17.4.6 SPI Receive Data Register (SPI_RX_DATA)

Address: 0x4001_0010

Reset value: 0x0

Table 17-6 Receive Data Register (SPI_RX_DATA)

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

Location	Bit name	Description
[31:8]		Unused
[5:0]	RX_DATA	SPI Receive Data Register

17.4.7 SPI Transfer Data Length Register (SPI_SIZE)

Address: 0x4001_0014

Reset value: 0x0

Table 17-7 Transfer Data Length Register (SPI_SIZE)

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
[5:0]	SIZE	SPI Transfer Data Length Register

18 CMP

18.1 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.

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

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. 18-1. 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.

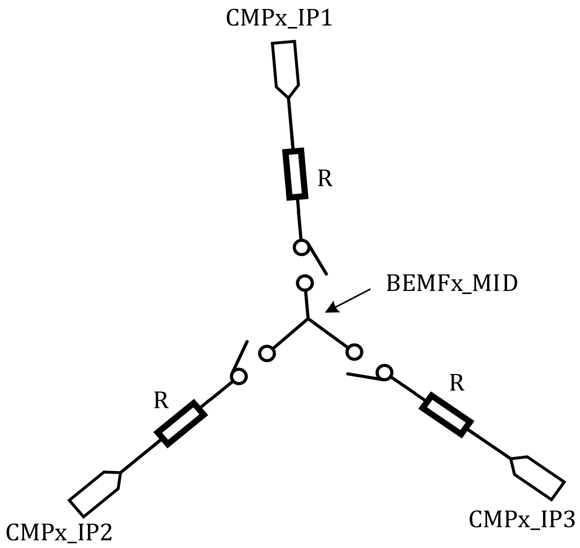


Fig. 18-2 BEMFx_MID Signal

The signal processing clock is obtained by dividing the main clock. The unfiltered original output value of the analog comparator can be obtained by reading the SYS_AFE_CMP value. The unfiltered original output value of the analog comparator can also be sent through P0.14 and P2.3 by configuring the second function of GPIO. For specific GPIO second function configuration and introduction location, please refer to the device datasheet.

Recommended configuration process:

1. Open the CMP analog switch

   Under the default state, the comparator module is turned off. By configuring the SYS_AFE_REG5.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.CLK_EN, 1 means opening, 0 means closed. Select the filter clock frequency division by configuring CMP_TCLK.FIL_CLK_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 can be set to 0.15uS/0.6uS by configuring SYS_AFE_REG1.IT_CM[1:0]. The comparator hysteresis can be set to 20mV/0mV by configuring SYS_AFE_REG3.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_REG3.CMPx_SELP[2:0]. There are 4 signal sources of the negative terminal to choose from, which can be set by configuring SYS_AFE_REG3.CMPx_SELN[1:0].

5. Configure the interrupt of CMP

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

18.2 Register

18.2.1 Address Allocation

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

Table 18-1 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

18.2.2 Register Description

18.2.2.1 CMP_IE

Address: 0x4001_0C00

Reset value: 0x0

Table 18-2 Comparator Interrupt Enable Register (CMP_IE)

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

Location	Bit name	Description
[31:2]		Unused
[1]	CMP1_IE	Comparator 1 interrupt enable, active high
[0]	CMP0_IE	Comparator 0 interrupt enable, active high

18.2.2.2 CMP_IF

Address: 0x4001_0C04

Reset value: 0x0

Table 18-3 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
														RW1C	RW1C
														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

18.2.2.3 CMP_TCLK

Address: 0x4001_0C08

Reset value: 0x0

Table 18-4 Comparator Divider Clock Control Register (CMP_TCLK)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								FIL_CLK_DIV16				CLK_EN		FIL_CLK_DIV1248	
								RW				RW		RW	
								0				0		0	

Location	Bit name	Description
[31:8]		Unused
[7:4]	FIL_CLK_DIV16	Comparator filter clock frequency division, based on MCLK to divide by 1 to 16, affecting the time to enter the comparator interrupt
[3]	CLK_EN	Clock enable, active high
[2]		Unused
[1:0]	FIL_CLK_DIV1248	Comparator filter clock divided by 2'b00: 1, 2'b01: 2, 2'b10: 4, 2'b11: 8

CMP Filter Operating Frequency

Freq(CMP_Filter) = Freq(MCLK)/$2^{CMP\_TCLK.FIL\_CLK\_DIV1248}$/(CMP_TCLK.FIL_CLK_DIV16+1), MCLK is the main clock, usually is a 96MHz full-speed clock. Note that the CMP_TCLK.CLK_EN bit should be enabled to generate the CMP filter clock.

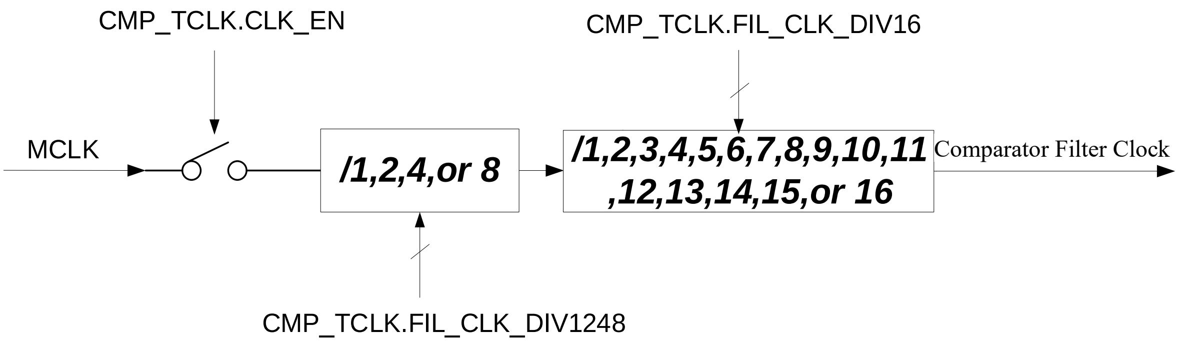


Fig. 18-1 Comparator Filter Clock Generation

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**.

18.2.2.4 CMP_CFG

Address: 0x4001_0C0C

Reset value: 0x0

Table 18-5 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_BLCWIN 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_BLCWIN 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. 18-2.

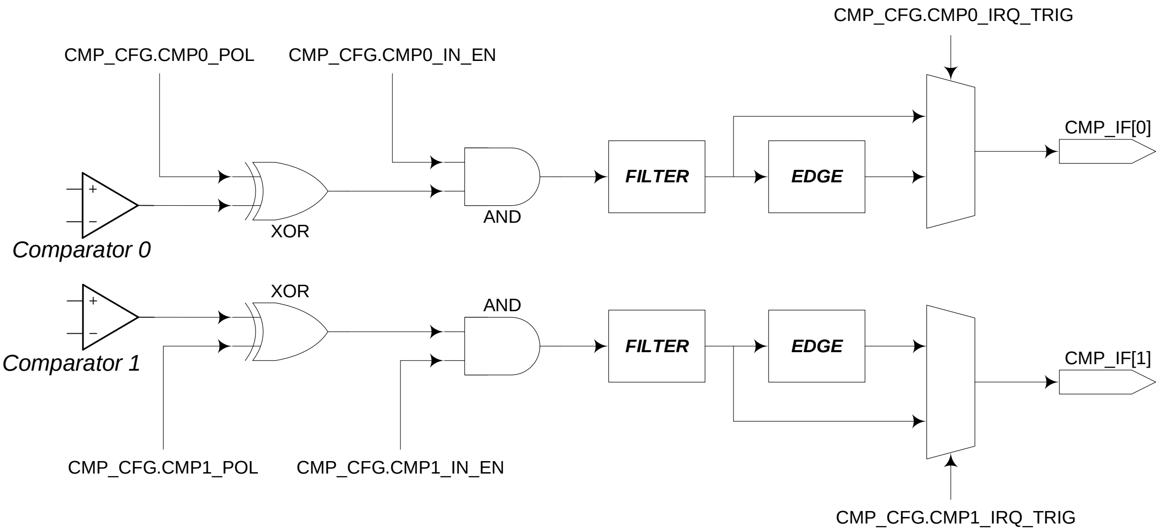


Fig. 18-2 Comparator Control and Interrupt Generation Logic

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.

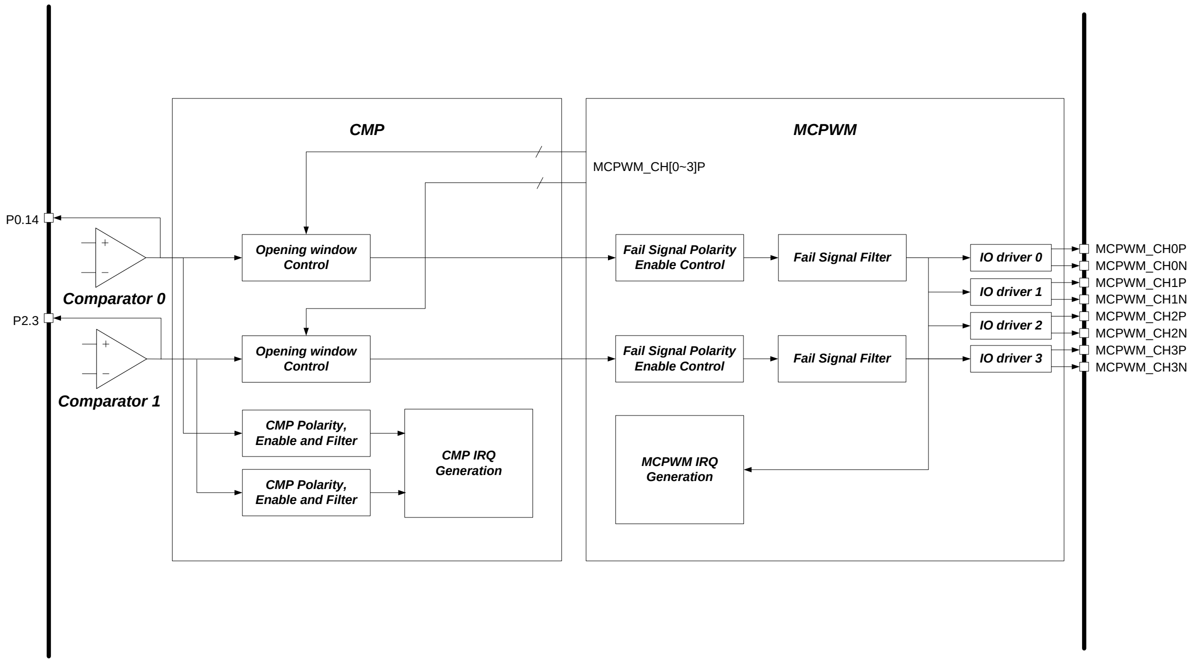


Fig. 18-3 CMP and MCPWM Linkage

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. 18-4.

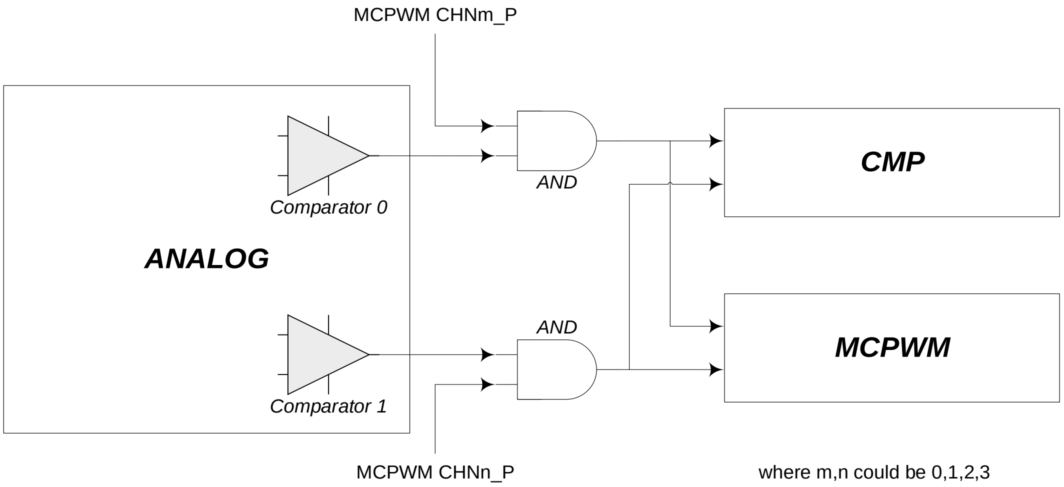


Fig. 18-4 Comparator Window Function Diagram

18.2.2.5 CMP_BLCWIN

Address: 0x4001_0C10

Reset value: 0x0

Table 18-6 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_WIN_EN	Use the P-tube switch control signal output from the CHN3_P channel 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 CHPWM 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 CHPWM 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 by the CHN0_P channel of the MCPWM module as the comparator 0 window enable

Usually 1-bit is 1 in CMP_ BLCWIN [3: 0] or CMP_ BLCWIN [7: 4], indicating that the corresponding CHNx_P is used to control the signal generation of the comparator 0/1. If CMP_ BLCWIN [3: 0] or CMP_ BLCWIN [7: 4] is 4’b0000, it means that the comparator 0/1 comparison signal is generated regardless of the PWM signal.

19 CAN

19.1 Introduction

The CAN bus interface connects the microcontroller and the serial CAN bus. According to the needs of specific equipment, this CAN module can use DMA to reduce the burden of MCU.

19.1 Main Features

Support BOSCH 2.0A and 2.0B protocols. 2.0A is equivalent to CAN1.2 and contains 11-bit ID format; 2.0B contains 11-bit ID and 29-bit ID.

- Support most functions of SJA1000 (some unsupported points are explained in the follow-up document).
- There are two modes: working mode and reset mode.
- Support monitoring and self-test. In monitor mode, only external signals are received but no signals are returned.
- DMA function.

19.2 Functional Description

19.2.1 Functional Block Diagram

This interface adopts synchronous serial design to realize CAN transmission between MCU and external equipment. and supports polling and interrupt mode to obtain transmission status information. The main functional modules of this interface are shown in the figure below.

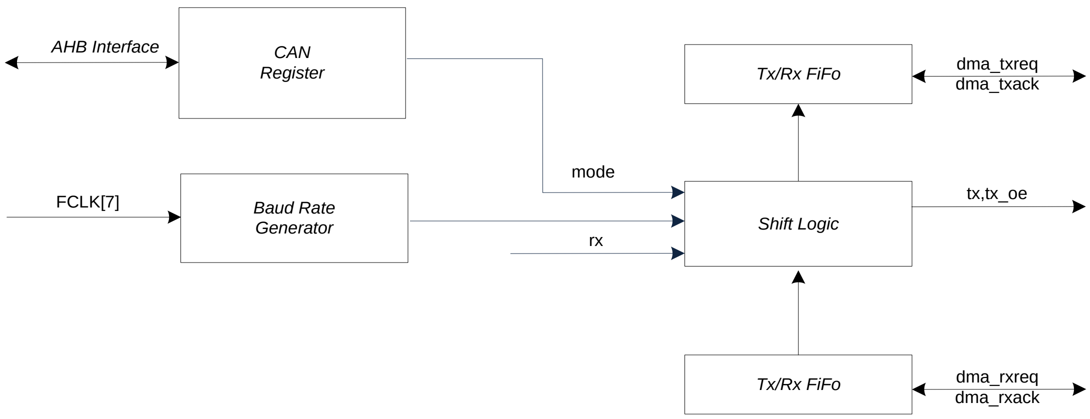


Fig. 19-1 CAN Module Top-level Functional Block Diagram

CAN interface communicates with the outside world only tx and rx two signal lines. When tx sends valid data, tx_oe is valid. When tx does not send valid data, tx_oe is invalid.

rx: data signal. Receive CAN data from outside.

tx: data signal. Send CAN data to the CAN bus.

tx_oe: data enable signal. When tx is output, tx_oe is valid; when tx has no data output, tx_oe is invalid.

19.2.2 Pin Function Description

The CAN 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 PHY chip of the CAN bus through a data output pin (TX) and a data input pin (RX).

19.2.2.1 Operating Mode

The CAN module mainly includes two working modes: normal working mode and reset mode.

Reset mode, timing parameters, ID configuration, error statistics, etc. are all set in this mode. CAN_MOD.0 is 1, it is the reset mode. After the hardware reset, the CAN module is in reset mode.

Normal working mode, CAN_MOD.0 is 0. it can respond to CAN bus request normally.

In the above two modes, the listening mode (Listen Only) and the self test mode (Self Test) have been expanded. The former, like a collector, only receives data on the CAN bus, and does not send any data. The latter is an internal self-test. The sent data is received by itself at the same time to check whether the internal function is correct.

19.2.2.2 DMA Transmission

Under the application of large-capacity data transmission, the CAN interface supports DMA transmission, reducing the burden on the MCU. One transmission, SFF is 11 bytes, EFF is 13 bytes.

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 CAN interface through the DMA module.

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

The CAN interface supports both DMA transmission and 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 by CAN.
- Initialize the CAN interface and set the control register.
- Trigger the CAN interface and enter the sending state.

The DMA designed by this CAN module is different from the DMA moving operation of other modules, which requires the MCU to intervene in some moving operations. Assuming that the currently configured CAN module sends N frames of data, then the first frame of data requires the MCU to be moved to the CAN module register, and the data of the subsequent frame (N-1) can be moved by DMA.

19.2.2.3 MCU Transmission

MCU transmission, according to SFF or EFF, move 11 bytes or 13 bytes; After the completion, determine whether the transmission is completed by interruption or polling.

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

- Initialize the GPIO module and set the GPIO multiplexed by CAN.
- Initialize the CAN interface and set the control register.
- The MCU triggers the CAN interface to enter the sending process. The sent data comes from the MCU writing the value to CAN_TXDATA.

19.2.2.4 Interrupt Handling

The CAN interface contains many interrupt events, which are described in the description of the CAN_IR and CAN_EIR registers. Turn on the corresponding interrupt event enable switch according to the actual usage.

19.2.2.5 Communication Sampling Settings

The setting of CAN baud rate mainly depends on the two registers CAN_BTR0 and CAN_BTR1. CAN_BTR0 is mainly to set TQ parameters (see the description of BTR0 register for TQ calculation), CAN_BTR1 mainly process sampling points, sampling times and width information of 1-bit data.

CAN_BTR0 sets the transmission basic time unit parameter TQ: TQ = 2*Tclk*(CAN_BTR0.BAUDRATE+1)

The fastest LKS08x clock is 96M, the corresponding Tclk is 10.4ns, and the maximum TQ is 1.3312us.

CAN_BTR1 sets the baud rate; The width of each part in the BIT information (TSEG1, TSEG2 and Sync.Seg) can be adjusted to find the ideal sampling point.

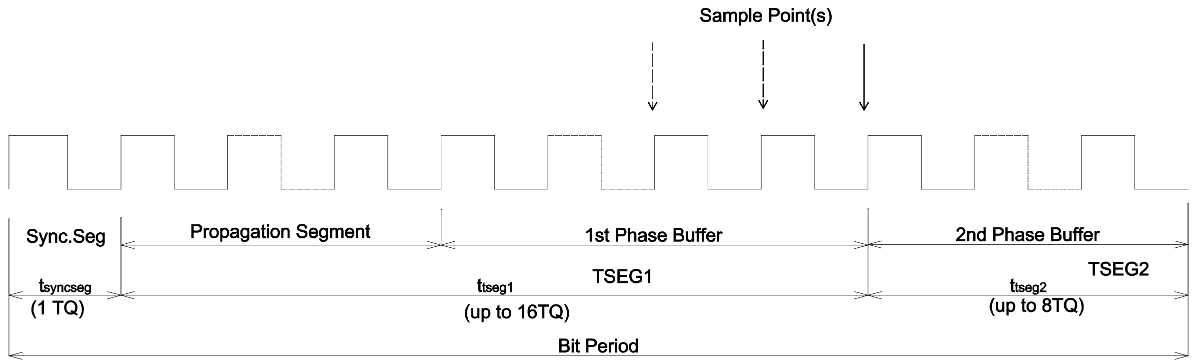


Fig. 19-2 CAN Module Bit Cycle Introduction Diagram

SEG1 period calculation formula: Tseg1 = TQ*(CAN_BTR1.SEG1+1)

SEG2 period calculation formula: Tseg2 = TQ*(CAN_BTR1.SEG2+1)

The formula for calculating the baud rate is: **Can Baudrate = 1/(1*TQ+Tseg1+Tseg2))**

CAN_BTR0.SJW is the SJW bit, which is the configuration register for the tolerance range. Each device communicating at a certain baud rate allows how much communication time error exists.

Bus lower bound tolerance <bus baud rate <bus upper limit tolerance

The tolerance calculation formula is:TQ*(SJW+1)

CAN_BTR1.SAM is the SAM bit, and it's the sampling times configuration register. 0: once; 1: three times. It can be set according to the actual use. This bit does not participate in the baud rate calculation.

The conventional baud rate index values based on the LKS08x chip are as follows:

CAN baud rate	BTR0	BTR1
1Mbps	0x05	0x14
800Kbps	0x05	0x16
666Kbps	0x85	0xB6
500Kbps	0x05	0x1C
400Kbps	0x05	0xFA
250Kbps	0x0B	0x1C
200Kbps	0x0B	0xFA
125Kbps	0x17	0x1C
100Kbps	0x1D	0x1C
83.33Kbps	0x17	0x6F
80Kbps	0x97	0xFF
66.66Kbps	0x1D	0x6F
50Kbps	0x3B	0x1C
40Kbps	0xAF	0xFF

In theory, we can achieve 1-bit data time width between 3TQ and 25TQ through the configuration register. In practical applications, we follow the BOSCH standard, and the time width of 1-bit data is directly from 8TQ to 25TQ. If we do not follow the BOSCH standard, our practical reach is between 4TQ and 25TQ.

19.2.2.6 ID Filtering

The CAN bus can be hung on many devices. Through the ID number, different devices can know whether the frame sent on the current bus needs to be received by themselves, or whether the frame sent out has a response.

The ID number of the CAN frame has two lengths, 11 bits and 29 bits. The former corresponds to SFF (standard frame format), the latter to EFF (extended frame format). Through CAN_ACR and CAN_AMR to judge the current ID range of CAN module. CAN_ACR lists a specific ID, CAN_AMR is a

MASK register, which identifies which bit data in CAN_ACR exactly matches the corresponding bit of the received ID, and which bits can be used. The extreme case is that every bit on CAN_ACR must match, or every bit should not match.

CAN_MOD.3 determines whether CAN_ACR contains two filter IDs or one filter ID. If it is 1, CAN_ACR contains a long filter ID; if it is 0, CAN_ACR contains two short filter IDs. In the case of two filtered IDs, as long as the ID of the received frame matches one of them, it will be received by CAN.

SFF, single filter ID

CAN_TXRX0	CAN_TXRX1	CAN_TXRX2	CAN_TXRX3
ID.28..ID21	ID.20..ID.18 RTR X X X X	Data Byte 1	Data Byte 2

Filter ID

ACR0[7:0]	ACR1[7:4] (ACR1[3:0] unused)	ACR2[7:0]	ACR3[7:0]
AMR0[7:0]	AMR1[7:4] (AMR1[3:0] unused)	AMR2[7:0]	AMR3[7:0]

SFF, double filter ID

CAN_TXRX0	CAN_TXRX1	CAN_TXRX2	CAN_TXRX3
ID.28..ID21	ID.20..ID.18 RTR X X X X	Data Byte 1	Data Byte 2

Filter ID1

ACR0[7:0]	ACR1[7:0]	ACR3[3:0]
AMR0[7:0]	AMR1[7:0]	AMR3[3:0]

Filter ID2

ACR2[7:0]	ACR3[7:4]
AMR2[7:0]	AMR3[7:4]

EFF, double filter ID

CAN_TXRX0	CAN_TXRX1	CAN_TXRX2	CAN_TXRX3
ID.28..ID21	ID.20..ID.13	ID.12..ID.5	ID.4..ID.0 RTR X X

Filter ID

ACR0[7:0]	ACR1[7:0]	ACR2[7:0]	ACR3[7:2]
AMR0[7:0]	AMR1[7:0]	AMR2[7:0]	AMR3[7:2]

EFF, double filter ID

CAN_TXRX0	CAN_TXRX1	CAN_TXRX2	CAN_TXRX3
ID.28..ID21	ID.20..ID.13	ID.12..ID.5(not matched)	ID.4..ID.0 RTR X X(not matched)

Filter ID1

ACR0[7:0]	ACR1[7:0]
AMR0[7:0]	AMR1[7:0]

Filter ID2

ACR2[7:0]	ACR3[7:0]
AMR2[7:0]	AMR3[7:0]

19.2.2.7 Send Frame Format

The transmission frame is divided into an ID part and a data part. The first byte contains information such as frame classification, determines whether it is an SFF (standard) frame or an EFF (extended) frame; determines whether it is a remote frame or a data frame; and, determines the data length. The ID length of SFF is 2 bytes, and the ID length of EFF is 4 bytes. The data length is a maximum of 8 bytes.

Table 19-1 Send Frame Structure

SFF		EFF	
Address	Domain	Address	Domain
0x40	TX frame info.	0x40	TX frame info.
0x44	TX ID0	0x44	TX ID0
0x48	TX ID1	0x48	TX ID1
0x4C	TX DATA0	0x4C	TX ID2
0x50	TX DATA1	0x50	TX ID3
0x54	TX DATA2	0x54	TX DATA0
0x58	TX DATA3	0x58	TX DATA1
0x5C	TX DATA4	0x5C	TX DATA2
0x60	TX DATA5	0x60	TX DATA3
0x64	TX DATA6	0x64	TX DATA4
0x68	TX DATA7	0x68	TX DATA5
0x6C	unused	0x6C	TX DATA6
0x70	unused	0x70	TX DATA7

SFF frame non-data information

Table 19-2 Send SFF Header Information

CAN Address	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
0x40	FF	RTR	X1	X1	DLC.3	DLC.2	DLC.1	DLC.0
0x44	ID.28	ID.27	ID.26	ID.25	ID.24	ID.23	ID.22	ID.21
0x48	ID.20	ID.19	ID.18	X2	X1	X1	X1	X1

EFF frame non-data info.

Table 19-3 Send EFF Header Information

CAN Address	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
0x40	FF	RTR	X1	X1	DLC.3	DLC.2	DLC.1	DLC.0
0x44	ID.28	ID.27	ID.26	ID.25	ID.24	ID.23	ID.22	ID.21
0x48	ID.20	ID.19	ID.18	ID.17	ID.16	ID.15	ID.14	ID.13
0x4C	ID.12	ID.11	ID.10	ID.9	ID.8	ID.7	ID.6	ID.5
0x50	ID.4	ID.3	ID.2	ID.1	ID.0	X2	X1	X1

- FF: 1 means EFF (extended) frame, 0 means SFF (standard) frame.
- RTR: 1 indicates a remote frame, and 0 indicates a data frame.
- DLC: indicates the length of data to be sent in this frame. The maximum is 8 bytes and the minimum is 0 bytes.
- ID: Frame identification number. The ID length of the SFF frame is 11 bits (ID.28 to ID.18). The ID length of the EFF frame is 29 bits (ID.28 to ID.0). High order is sent first.
- DATA: data. The order between bytes, from big to small, that is, TX DATA7 is sent first. The order within the byte, from high to low.
- X2: It is best to be consistent with the RTR value
- X1: 1 or 0

19.2.2.8 Receive Frame Format

The received frame is divided into an ID part and a data part. The first byte contains information such as frame classification, determines whether it is an SFF (standard) frame or an EFF (extended) frame; determines whether it is a remote frame or a data frame; and, determines the data length. The ID length of SFF is 2 bytes, and the ID length of EFF is 4 bytes. The data length is a maximum of 8 bytes.

Table 19-4 Receive Frame Structure

SFF		EFF	
Address	Domain	Address	Domain
0x40	RX frame info	0x40	RX frame infor
0x44	RX ID0	0x44	RX ID0
0x48	RX ID1	0x48	RX ID1
0x4C	RX DATA0	0x4C	RX ID2

0x50	RX DATA1	0x50	RX ID3
0x54	RX DATA2	0x54	RX DATA0
0x58	RX DATA3	0x58	RX DATA1
0x5C	RX DATA4	0x5C	RX DATA2
0x60	RX DATA5	0x60	RX DATA3
0x64	RX DATA6	0x64	RX DATA4
0x68	RX DATA7	0x68	RX DATA5
0x6C	unused	0x6C	RX DATA6
0x70	unused	0x70	RX DATA7

SFF frame non-data information

Table 19-5 Receive SFF Header Information

CAN Address	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
0x40	FF	RTR	0	0	DLC.3	DLC.2	DLC.1	DLC.0
0x44	ID.28	ID.27	ID.26	ID.25	ID.24	ID.23	ID.22	ID.21
0x48	ID.20	ID.19	ID.18	RTR	0	0	0	0

EFF frame non-data info.

Table 19-6 Receive EFF Header Information

CAN Address	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
0x40	FF	RTR	0	0	DLC.3	DLC.2	DLC.1	DLC.0
0x44	ID.28	ID.27	ID.26	ID.25	ID.24	ID.23	ID.22	ID.21
0x48	ID.20	ID.19	ID.18	ID.17	ID.16	ID.15	ID.14	ID.13
0x4C	ID.12	ID.11	ID.10	ID.9	ID.8	ID.7	ID.6	ID.5
0x50	ID.4	ID.3	ID.2	ID.1	ID.0	RTR	0	0

19.2.2.9 Send

The CAN module must send data in the normal operating mode.

To perform the sending operation, it is generally recommended to turn on the sending interrupt source and the number of error warning interrupt sources. The source of the arbitration loss and bus error interruption is not forced to open, and the CAN module has an automatic retransmission mechanism.

The transmitted data is stored in the TX FIFO, which is 32-Byte, and can store several frames of data at a time, and once the data in the TX FIFO is successfully sent, it is released by the CAN module for new data to be written. The data length of one frame is 13 bytes, which considers that the maximum data amount is 8 bytes. Make sure that the module is in idle state before sending.

The format of the transmitted frame is divided into standard frame (SFF) and extended frame

(EFF). BIT0 of CAN_CMR is set to 1, trigger the CAN module to send data. Once the CAN bus is idle, data will be sent out. After the transmission is completed, check whether BIT2 of CAN_SR becomes 1 (1 means idle), or check whether the transmission is completed by interruption.

Before the data is sent to the bus, set BIT1 of CAN_CMR to stop sending and generate corresponding interrupts; check whether the current data is sent through BIT3 of CAN_SR.

19.2.2.10 Receive

The CAN module must receive data in the normal operating mode.

Reception starts when a Start frame is detected. The received data should pass the ID matching before stored in the RX FIFO. The RX FIFO is 32-Byte, which can store several frames of data at a time, and once the data in the RX FIFO is successfully taken by the MCU/DMA, it is released by the CAN module for new data to be written. The data length of one frame is 13 bytes, which considers that the maximum data amount is 8 bytes. There are several valid frames in the RX FIFO, which can be seen through the CAN_AMC register. The first received data frame can be obtained by reading the CAN_TXRX register.

19.2.3 Register

19.2.3.1 Address Allocation

The base address of the CAN module register is 0x4001_3400, and the register list is as follows.

Table 19-7 CAN Register Address Allocation

Name	Offset	Description
CAN_MOD	0x000	CAN mode register
CAN_CMR	0x004	CAN command register
CAN_SR	0x008	CAN status register
CAN_IR	0x00C	CAN interrupt status register
CAN_IER	0x010	CAN interrupt control register
CAN_BTR0	0x018	CAN bus timing control register 0
CAN_BTR1	0x01C	CAN bus timing control register 1
CAN_ALC	0x02C	CAN arbitration lost capture register
CAN_ECC	0x030	CAN error code capture register
CAN_EWLR	0x034	CAN error & warning threshold setting register
CAN_RXERR	0x038	CAN receive error counter
CAN_TXERR	0x03C	CAN send error counter
CAN_TXRX0	0x040	CAN send frame format register/CAN receive frame format register in normal working mode
CAN_TXRX1	0x044	CAN transmit data register 0/CAN receive data register 0 in normal working mode
CAN_TXRX2	0x048	CAN transmit data register 1/CAN receive data register 1 in normal working mode
CAN_TXRX3	0x04C	CAN transmit data register 2/CAN receive data register 2

		in normal working mode
CAN_TXRX4	0x050	CAN transmit data register 3/CAN receive data register 3 in normal working mode
CAN_TXRX5	0x054	CAN transmit data register 4/CAN receive data register 4 in normal working mode
CAN_TXRX6	0x058	CAN transmit data register 5/CAN receive data register 5 in normal working mode
CAN_TXRX7	0x05C	CAN transmit data register 6/CAN receive data register 6 in normal working mode
CAN_TXRX8	0x060	CAN transmit data register 7/CAN receive data register 7 in normal working mode
CAN_TXRX9	0x064	CAN transmit data register 8/CAN receive data register 8 in normal working mode
CAN_TXRXA	0x068	CAN transmit data register 9/CAN receive data register 9 in normal working mode
CAN_TXRXB	0x06C	CAN transmit data register 10/CAN receive data register 10 in normal working mode
CAN_TXRXC	0x070	CAN transmit data register 11/CAN receive data register 11 in normal working mode
CAN_ACR0	0x040	CAN ID code register 0 in reset mode
CAN_ACR1	0x044	CAN ID code register 1 in reset mode
CAN_ACR2	0x048	CAN ID code register 2 in reset mode
CAN_ACR3	0x04C	CAN ID code register 3 in reset mode
CAN_AMR0	0x050	CAN ID mask register 0 in reset mode
CAN_AMR1	0x054	CAN ID mask register 1 in reset mode
CAN_AMR2	0x058	CAN ID mask register 2 in reset mode
CAN_AMR3	0x05C	CAN ID mask register 3 in reset mode
CAN_RMC	0x074	CAN FIFO effective reception information counter
CAN_RBSA	0x078	Address register of the first CAN valid received message in FIFO
CAN_CDR	0x07C	CAN system clock divider register
CAN_RFIFO0~CAN_RFIFO31	0x080~0x144	CAN RX FIFO address register
CAN_TFIFO0 ~CAN_TFIFO31	0x180~0x1B0	CAN TX FIFO address register

19.2.3.2 Register Description

19.2.3.2.1 Mode Register (CAN_MOD)

Address: 0x4001_3400

Reset value: 0x0

Table 19-8 Mode Register (CAN_MOD)

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

Location	Bit name	Description
[31:5]		Unused
[4]	SLEEP	Write1 in normal working mode to trigger the sleep operation and enter sleep mode.
[3]	FLT_ID	CAN filter ID selection. The default value is 0. 1: single filter ID. A 32-bit filter ID 0: double filter ID. Two short filter IDs
[2]	TEST_MODE	CAN working mode selection. The default value is 0. 1: self-test mode 0: normal working mode
[1]	FUNC_MODE	CAN working mode selection. The default value is 0. 1: monitor mode 0: normal working mode
[0]	MODE	CAN working mode selection. The default value is 1. 1: reset mode 0: normal working mode

19.2.3.2.2 Command Register (CAN_CMR)

Address: 0x4001_3404

Reset value: 0x0

Table 19-9 Command Register (CAN_CMR)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										DMA_EN	RX_DUR_TX	CLR_OV	RELEASE_FIFO	INTR_TRANS	TRANS_REQ
										WO	WO	WO	WO	WO	WO
										0	0	0	0	0	0

Location	Bit name	Description
[31:6]		Unused
[5]	DMA_EN	Write 1, enable DMA function
[4]	RX_DUR_TX	Write 1, send and receive data
[3]	CLR_OV	Write 1 to clear the data overflow flag
[2]	RELEASE_FIFO	Write 1, release RFIFO
[1]	INTR_TRANS	Write 1, cancel the unexecuted transmission

[0]	TRANS_REQ	Write 1, generate CAN send transmission request

19.2.3.2.3 Status Register (CAN_SR)

Address: 0x4001_3408

Reset value: 0x0

Table 19-10 Status Register (CAN_SR)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								ON_BUS	ERR_OV	TXING	RXING	TRANS_DONE	TFIFO_EMPTY	RFIFO_EMPTY	DATA_AVAIL
								RO	RO	RO	RO	RO	RO	RO	RO
								0	0	0	0	0	0	0	0

Location	Bit name	Description
[31:8]		Unused
[7]	ON_BUS	1: CAN module, in BUS OFF state, no data sending and receiving action 0: CAN module, can send data to CAN bus, or receive CAN bus data
[6]	ERR_OV	1: The total number of errors generated by CAN transmission reaches or exceeds the CAN_EWL specified value 0: The total number of errors generated by CAN transmission is lower than the specified value of CAN_EWL
[5]	TXING	1: CAN module is sending a frame of data 0: CAN module does not send data
[4]	RXING	1: CAN module is receiving a frame of data 0: CAN module does not receive data
[3]	TRANS_DONE	1: The most recent transfer has been completed 0: The last transmission was not completed
[2]	TFIFO_EMPTY	1: TFIFO is empty, write and send data 0: TFIFO is not empty, internal data transmission was not completed
[1]	RFIFO_EMPTY	1: Too many frames stored in RFIFO, full, resulting in data loss 0: RFIFO is not full
[0]	DATA_AVAIL	1: There is one or more frames of data stored in RFIFO, which can be read through RFIFO register 0: No valid frame data in RFIFO

19.2.3.2.4 Interrupt Status Register (CAN_IR)

Address: 0x4001_340C

Reset value: 0x0

Table 19-11 Interrupt Status Register (CAN_IR)

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

Location	Bit name	Description
[31:8]		Unused
[7]	BUS_ERR_IF	1: Bus error interrupt (CAN_EIR.7 is valid)
[6]	LOST_ARB_IF	1: lost arbitration and changed to receive mode (CAN_EIR.6 is valid)
[5]	M_ERR_IF	1: Repeatedly enter the error state or the error always exceeds the specified value (CAN_EIR.5 is valid)
[4]	WK_IF	1: CAN module wakes up from sleep (CAN_EIR.4 is valid)
[3]	RFIFO_OV_IF	1: RFIFO data overflow (CAN_EIR.3 is valid)
[2]	CAN_SR67_IF	1: CAN_SR.7 or CAN_SR.6 has changed (CAN_EIR.2 is valid)
[1]	TX_DONE_IF	1: Current frame has been sent (CAN_EIR.1 is valid)
[0]	RFIFO_N_EMPTY_IF	1: New data has been received in RFIFO (CAN_EIR.0 is valid)

19.2.3.2.5 Interrupt Control Register (CAN_EIR)

Address: 0x4001_3410

Reset value: 0x0

Table 19-12 Interrupt Control Register (CAN_EIR)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								BUS_ERR_IE	LOST_ARB_IE	M_ERR_IE	WK_IE	RFIFO_OV_IE	CAN_SR67_IE	TX_DONE_IE	RFIFO_N_EMPTY_IE
								RW	RW	RW	RW	RW	RW	RW	RW

	0	0	0	0	0	0	0	0

Location	Bit name	Description
[31:8]		Unused
[7]	BUS_ERR_IE	1: Bus error interrupt, interrupt source enable 0: interrupt source is turned off
[6]	LOST_ARB_IE	1: lost arbitration, changed to receive mode, interrupt source enable 0: interrupt source is turned off
[5]	M_ERR_IE	1: Repeatedly enter the error state or the error always exceeds the specified value, the interrupt source is enabled 0: interrupt source is turned off
[4]	WK_IE	1: CAN module wakes up from sleep and interrupt source is enabled 0: interrupt source is turned off
[3]	RFIFO_OV_IE	1: RFIFO data overflow, interrupt source enable 0: interrupt source is turned off
[2]	CAN_SR67_IE	1: CAN_SR.7 or CAN_SR.6 has changed, the interrupt source is enabled 0: interrupt source is turned off
[1]	TX_DONE_IE	1: Current frame has been sent, the interrupt source is enabled 0: interrupt source is turned off
[0]	RFIFO_N_EMPTY_IE	1: New data is received in RFIFO and interrupt source is enabled 0: interrupt source is turned off

19.2.3.2.6 Baud Rate 0 Control Register (CAN_BTR0)

Address: 0x4001_3418

Reset value: 0x0

Table 19-13 Baud Rate 0 Control Register (CAN_BTR0)

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

Location	Bit name	Description
[31:8]		Unused
[7:6]	SJW	Synchronous jump width configuration. The formula is as follows: Tsjw = TQ*(SJW+1)
[5:0]	BAUDRATE	Baud rate setting. Calculate CAN transmission basic time unit

		parameters TQ, TCLK is the system clock frequency of CAN module TQ = 2*Tclk*(BAUDRATE+1)

19.2.3.2.7 Baud Rate 0 Control Register (CAN_BTR1)

Address: 0x4001_341C

Reset value: 0x0

Table 19-14 Baud Rate 0 Control Register (CAN_BTR1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								OSR	SEG2			SEG1			
								RW	RW			RW			
								0	0			0			

Location	Bit name	Description
[31:8]		Unused
[7]	OSR	1: 1-bit data is sampled three times 0: 1-bit data is sampled once
[6:4]	SEG2	SEG2 period Tseg2 = TQ*(SEG2+1)
[3:0]	SEG1	SEG1 period Tseg1 = TQ*(SEG1+1)

19.2.3.2.8 Arbitration Lost Capture Register (CAN_ALC)

Address: 0x4001_342C

Reset value: 0x0

Table 19-15 Arbitration Lost Capture Register (CAN_ALC)

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

Location	Bit name	Description
[31:5]		Unused
[4:0]	LOST_ARB	Record the specific location where bus arbitration is lost. 0: The first digit of the ID 1: The second digit of the ID

		2: The third digit of the ID 3: The fourth digit of the ID 4: The fifth digit of the ID 5: The sixth digit of ID 6: The seventh digit of ID 7: The eighth digit of the ID 8: The ninth digit of the ID 9: The tenth digit of the ID A: The eleventh digit of the ID B: SRTR bit C: IDE bit

19.2.3.2.9 Error Code Capture Register (CAN_ECC)

Address: 0x4001_3430

Reset value: 0x0

Table 19-16 Error Code Capture Register (CAN_ECC)

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

Location	Bit name	Description
[31:8]		Unused
[7:6]	BUS_ERR_TYPE	Bus error type 00: bit error 01: format error 10: Fill error 11: Other types of errors
[5]	ERR_TIMING	1: Bus error occurred during reception 0: Bus error occurred during transmission
[4:0]	ERR_POSITION	Error position 02: ID.28 to ID.21 03: Start frame 04: SRTR bit 05: IDE bit 06: ID.20 to ID.18 07: ID.17 to ID.13 08: CRC data segment 09: reserved bit, fixed to 0

		0A: Data Field 0B: Data Length Code 0C: RTR bit 0D: reserved bit, fixed to 1 0E: ID.4 to ID.0 0F: ID.12 to ID.5 11: active error flag 12: Intermission 13: Tolerate dominant bit 16: passive error flag 17: Error delimiter 18: CRC delimiter 19: Confirmation bit 1A: End of frame 1B: Confirm the delimiter 1C: Overload flag

19.2.3.2.10 Error & Warning Threshold Register (CAN_EWLR)

Address: 0x4001_3434

Reset value: 0x0

Table 19-17 Error & Warning Threshold Register (CAN_EWLR)

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

Location	Bit name	Description
[31:8]		Unused
[7:0]	EWLR	Read-only in normal operating mode; readable and writable in reset mode. After writing the new value, it takes effect in the normal operating mode.

19.2.3.2.11 Receive Error Counter Register (CAN_RXERR)

Address: 0x4001_3438

Reset value: 0x0

Table 19-18 Receive Error Counter Register (CAN_RXERR)

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

Location	Bit name	Description
[31:8]		Unused
[7:0]	RXERR	Read-only in normal operating mode; readable and writable in reset mode. Received detected errors. Add 1; After sending the error flag, a high level is received. Add 8; After sending the active error/overload error, a bit error is received. Add 8; The message is received normally. Add 1; The message is received normally, the counter is higher than 127, and the value is set between 119 and 127; After sending the active error/overload error, 14 consecutive highs are received. Add 8.

19.2.3.2.12 Transmit Error Counter Register (CAN_TXERR)

Address: 0x4001_343C

Reset value: 0x0

Table 19-19 Transmit Error Counter Register (CAN_TXERR)

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

Location	Bit name	Description
[31:8]		Unused
[7:0]	TXERR	Read-only in normal operating mode; readable and writable in reset mode. After sending active error/overload error, 14 consecutive highs are received. Add 8; Send error. Add 8; In the process of sending active error/overload error, if an error is found. Add 8; Send a message successfully. Add 1.

19.2.3.2.13 Transceiver Register (AN_TXRX0~ CAN_TXRXC)

Address space: 0x4001_3440 ~ 0x4001_3470

Reset value: 0x0

Table 19-20 Transceiver Register (CAN_TXRX)

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

Location	Bit name	Description
[31:8]		Unused
[7:0]	TXRX	Data tx or rx register. Under normal working mode, this register maps the first valid frame data in RFIFO; Under the normal working mode, write data to this register and store it directly in the TFIFO.

19.2.3.2.14 ID Register (CAN_ACR)

Address space: 0x4001_3440 ~ 0x4001_344C

Reset value: 0x0

Table 19-21 ID Register (CAN_ACR)

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

Location	Bit name	Description
[31:8]		Unused
[7:0]	ACR	Only available in reset mode; readable and writable in reset mode; Receivable ID register, when the ID of the input frame matches this, it is accepted by the CAN module. Use along with the AMR register, the same type ID can be matched.

19.2.3.2.15 ID Mask Register (CAN_AMR)

Address space: 0x4001_3450 ~ 0x4001_345C

Reset value: 0x0

Table 19-22 ID Mask Register (CAN_AMR)

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

Location	Bit name	Description
[31:8]		Unused
[7:0]	AMR	Only available in reset mode; readable and writable in reset mode; Use along with the ACR register, the same type ID can be matched. When a bit in the AMR register is 0, the corresponding bit of the identification ID should match the ACR; When a bit in the AMR register is 1, the corresponding bit of the identification ID does not need to match the ACR.

19.2.3.2.16 FIFO Effective Receive Data Register (CAN_RMC)

Address: 0x4001_3474

Reset value: 0x0

Table 19-23 FIFO Effective Receive Data Register (CAN_RMC)

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

Location	Bit name	Description
[31:5]		Unused
[4:0]	FRAME_CNT	This register is only readable in normal operating mode and reset mode. When CAN receives a frame of data, the counter automatically increments by 1; after the MCU reads a frame of data, the counter automatically decrements by 1.

19.2.3.2.17 Effective Receive Data Address Register (CAN_RBSA)

Address: 0x4001_3478

Reset value: 0x0

Table 19-24 Effective Receive Data Address Register (CAN_RBSA)

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

	RW
	0

Location	Bit name	Description
[31:5]		Unused
[4:0]	RBSA	Only readable in normal operating mode. Read and write in reset mode The size of the RFIFO is 32 bytes. This register indicates the location of the valid received data of the first frame in the RFIFO.

19.2.3.2.18 Clock Divider Register (CAN_CDR)

Address: 0x4001_347C

Reset value: 0x0

Table 19-25 Clock Divider Register (CAN_CDR)

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

Location	Bit name	Description
[31:3]		Unused
[2:0]	RBSA	Only readable in normal operating mode. Readable and writable in reset mode 0: MCLK/2 1: MCLK/4 2: MCLK/6 3: MCLK/8 4: MCLK/10 5: MCLK/12 6: MCLK/14 7: MCLK TCLK is the clock of MCLK after this frequency division

19.2.3.2.19 RX FIFO Register (CAN_ RFIFO0 ~ CAN_RFIFO31)

Address space: 0x4001_3480 ~ 0x4001_34FC

Reset value: 0x0

Table 19-26 RX FIFO Register (CAN_ RFIFO0 ~ CAN_RFIFO31)

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

Location	Bit name	Description
[31:8]		Unused
[7:0]	RX_DATA	RX FIFO address Only readable in operating mode; readable and writable in reset mode.

19.2.3.2.20 TX FIFO Register (CAN_TFIFO0 ~ CAN_TFIFO12)

Address: 0x4001_3580~0x4001_35B0

Reset value: 0x0

Table 19-27 TX FIFO Register (CAN_TFIFO0 ~ CAN_TFIFO12)

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

Location	Bit name	Description
[31:8]		Unused
[7:0]	TX_DATA	TX FIFO address Only be read in operating mode and reset mode.

20 SIF

20.1 Introduction

The SIF bus interface supports the SIF protocol, data transmission between the liquid crystal display and the electric vehicle controller, and realizes the detection of the running state and failure.

20.2 Main Features

- The international standard SIF communication protocol is adopted, and the interface is universal and convenient.
- The master-slave mode uses single-line one-way transmission, that is, only one transmission line is needed, the electric vehicle controller is the sender, and the instrument is the receiver.
- The transmission line shares the I/O port with the electric vehicle control failure running light, and does not occupy additional resources.
- The transmission baud rate has a wide adaptive range, and the host can use idle time to send data.
- The unit of time base is wide, 32us <Tosc <320us.
- The data level complies with the TTL specification.
- An interrupt vector.

20.3 Functional Description

20.3.1 Functional Block Diagram

This interface adopts synchronous serial design to realize SIF transmission between MCU and external equipment. and supports polling and interrupt mode to obtain transmission status information. The main functional modules of this interface are shown in the figure below.

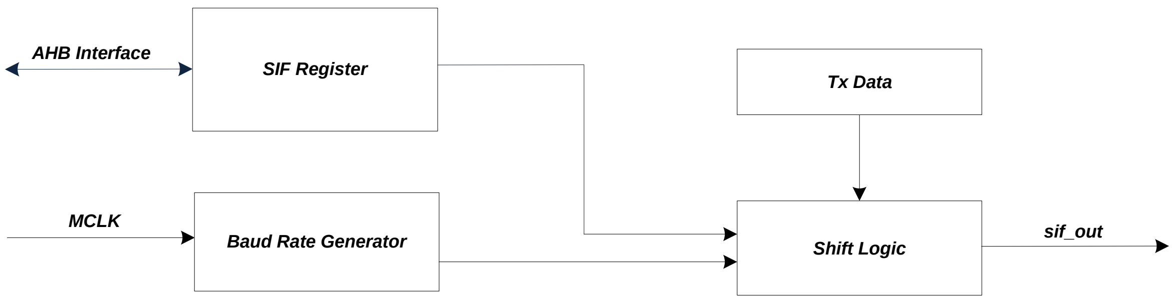


Fig. 20-1 SIF Module Top Functional Block Diagram

The SIF interface communicates with the outside world only with a signal line sif_out.

20.3.2 Pin Function Description

The SIF interface only supports master sending, and the module sends data to convert the data from parallel to serial. , and can enable or disable interrupts. The interface is connected to the SIF bus through the data pin sif_out.

20.3.3 Mode Selection

There is only one operating mode for the SIF interface: master transmit mode. The transmission speed of the SIF interface is very slow and does not support DMA transmission. An interrupt signal is generated after each byte is transferred.

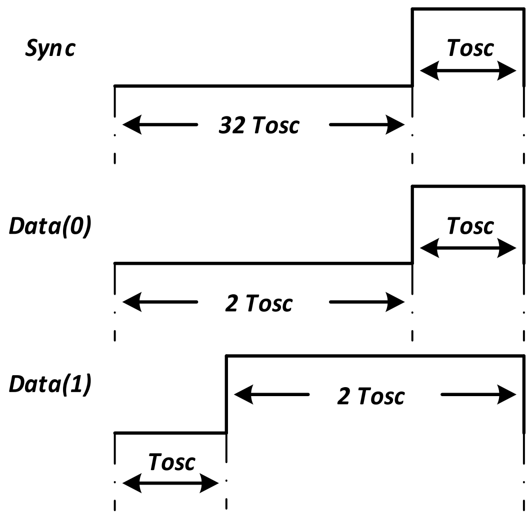


Fig. 20-2 SIF Basic Transmission Timing Diagram

Note: Figure 17-2 has a duty ratio of 2: 1, and an optional 3: 1. SIF_FREQ [0] controls the selection of duty cycle.

20.3.4 SIF Interface Transmission

The SIF interface is closed by default, and should be enabled before sending data. In order to generate the correct timing, the working clock of the SIF interface must be set in SFI_CFG.

20.3.5 Interrupt Handling

The SIF interface contains only one type of interrupt event, that is, the data transfer completion event.

20.3.6 Communication Speed Setting

The basic time unit of the SIF interface is 32us--Tosc. Tosc is generated from the system clock. By setting the SIF_FREQ register, we can get the right reference time at different system frequencies.

20.4 Register

20.4.1 Address Allocation

The base address of the SIF module register is 0x4001_3800, and the register list is as follows:

Table 20-1 List of SIF Module Control Register

Name	Offset	Description
SIF_CFG	0x00	SIF Configuration Register
SIF_FREQ	0x04	SIF Baud Rate Setting Register
SIF_IRQ	0x08	SIF Interrupt Register
SIF_WDATA	0x0C	SIF Transmit Data Register

20.4.2 Register Description

20.4.2.1 Configuration Register (SIF_CFG)

Address: 0x4001_3800

Reset value: 0x0

Table 20-2 Configuration Register (SIF_CFG)

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

Location	Bit name	Description
[31:5]		Unused
[4]	LAST_BYTE	After the current byte is transmitted, the transmission is ended. The SIF interface transmits the current byte. After the byte transmission is completed, an interrupt is generated. Write 1 to this bit in the interrupt handler to restore the SIF interface to its default state. If this bit is not written, it indicates the transmission is not ended. This bit is always “0” after read back.
[3:2]		Unused
[1]	ENDIAN	SIF big-end and little-end setting. 1:bMSB； 0:bLSB. The default value is 0.
[0]	EN	SIF module enable. The default value is 0. 1: Enable 0: Disable

20.4.2.2 Baud Rate Register (SIF_FREQ)

Address: 0x4001_3804

Reset value: 0x0

Table 20-3 Baud Rate Register (SIF_FREQ)

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

Location	Bit name	Description
[31:8]		Unused
[7:4]	TOSC	Tosc setting under the 96Mhz system clock. 01: 32us; 02: 64us; 03: 96us;04:128us 05:160us;06:192us;07:220us;08:256us 09:288us;10:320us; Others are 32us.
[3:1]		Unused
[0]	DUTY	SIF data transmission duty cycle 1: Duty ratio is 3: 1 0: Duty ratio is 2: 1

20.4.2.3 Status Control Register (SIF_IRQ)

Address: 0x4001_3808

Reset value: 0x0

Table 20-4 Status Control Register (SIF_IRQ)

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

Location	Bit name	Description
[31:5]		Unused
[4]	IF	SIF interrupt event. Active high. Write 1 to clear.
[3:1]		Unused
[0]	IE	SIF interrupt enable signal. Active high. The default value is 0.

20.4.2.4 Data Register (SIF_WDATA)

Address: 0x4001_380C

Reset value: 0x0

Table 20-5 Data Register (SIF_WDATA)

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

Location	Bit name	Description
[31:8]		Unused
[7:0]	WDATA	Data register. After the SIF interface is enabled, writing data to this register will trigger SIF data transmission.

21 Watchdog

21.1 Introduction

The watchdog works in the low-speed RC clock domain LSI, which uses 32kHz for counting, and 2s, 4s, 8s, 64s reset time is optional.

The reset control register SYS_RST_CFG.WDT_EN can be used to enable or disable the watchdog, and SYS_RST_CFG.WDT_EN = 1 enables the watchdog module. The reset source record register SYS_RST_SRC.WDT_RST_RCD records the watchdog reset event. When SYS_RST_SRC.WDT_RST_RCD is high, it indicates that a watchdog reset has occurred.

Watchdog reset is a hardware global reset, and its scope is equivalent to external pin reset and internal power-on reset.

21.2 Register

21.2.1 Address Allocation

The base address of Watchdog Module Register is 0x4000_0000.

Table 21-1 Watchdog Module Register

Name	Offset	Description
SYS_WDT_CLR	0x38	Watchdog Clear Register

21.2.2 Register Description

21.2.2.1 Watchdog Clear Register (SYS_WDT_CLR)

Table 21-2 Watchdog Clear Register (SYS_WDT_CLR)

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

Location	Bit name	Description
[31:16]		Unused
[15:0]	WDT_CLR	Write byte 16'b0111_1001_1000_$1B_2B_1B_0$, the upper 13-bit is the password, and B [2: 0] can be written only when the password is correct,. Wherein, B[2:1] is MODE

		2’b00: reset in 64 seconds 2’b01: reset in 8 seconds 2’b10: reset in 4 seconds 2’b11: reset in 2 seconds B [0] is CLR, write 1 to reset WDT counter

Before the watchdog is cleared, write 0xCAFE to SYS_WR_PROTECT, and enable the WatchDog SYS_WDT_CLR register to be written.

22 Version History

Table 22-1 Document's Version History

Date	Version No.	Description
Jun.10,2025	1.41	Modify the calculation formula of CAN baud rate
May.20,2025	1.40	Revise the accuracy of internal 4M RC clock, which is within ± 2.0% in the range of 40 ~ 125 °C
May.15,2025	1.39	Added description for temperature sensor chapter
May.13,2025	1.38	Added description of PVD
Jun.14,2024	1.37	NVR calibration parameter information added in FLAH chapter
Jan .15,2024	1.36	Added description on sleep mode Added description on I2C module communication speed
Dec. 6,2023	1.35	CAN regular baud rate index value modified, calculation formula modified
Apr .7, 2023	1.34	Modified the description of the ADC TIMER trigger Added a note that the ADC is needed to use the 1.2V reference when powered at 3.3V
Aug. 15,2023	1.33	Modified the GPIO functional block diagram
Mar. 18, 2023	1.32	Modified the description of clock accuracy
Feb. 28, 2023	1.31	Modified the description of PVDSEL
Feb. 18, 2023	1.3	Modified the description of MCPWM_SDCFG
Feb. 11, 2023	1.26	Added instructions on closing PLL and other operations Added the description of external crystal oscillator needed for CAN Modified the description of SYS_AFE_REG5.BGPPD
Jan. 3, 2023	1.25	Modified the description of Encoder Register
Dec. 29, 2022	1.24	Modified the description of AFE Register 6
Nov. 7, 2022	1.23	Added connection resistance between IO and internal analog circuit
Otc. 29, 2022	1.22	Modified the description of MCPWM_DTHx0/MCPWM_DTHx1
Otc. 17, 2022	1.21	Revise the DAC output formula
Sep. 23, 2022	1.20	Revise temperature sensor usage description when using 1.2V as ADC reference
Aug. 18, 2022	1.19	Add configuration procedure for ADC/CMP. Simplify some description of SYS_AFE_REG
May. 20, 2021	1.18	Modify the description of BEMFx_MID
Jun. 12, 2020	1.17	Modify the description of SYS_AFE_CMP
Feb. 18, 2020	1.16	Revised description
Dec. 3, 2019	1.15	Revised figures
Sep. 12, 2019	1.14	Revised description
Jun. 1, 2019	1.10	Revision the "Clock System" chapter
Mar. 18, 2019	1.0	Revisions before release

Nov. 29, 2018	0.1	Initial version

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.