南京凌鸥创芯电子有限公司

LKS32MC07x User Manual

目录

表格目录

图片目录

如图 23-1 所示，可配置逻辑 CL0 主要包括 4 个独立的可配置逻辑单元 CLU。每个 CLU 都可以独立

1 文档约定

1.1 寄存器读写权限

RW	读/写，软件可以读写这些位。
RO	只读，软件只能读取这些位。
WO	只写，软件只能写入该位。读取该位时将返回默认值。
RW1C(Read and Write 1 to Clear)	可读，写 1 清零。

1.2 缩略词汇表

字:32 位数据/指令。

半字:16 位数据/指令。

字节:8 位数据。

双字:64 位数据。

ADC:Analog-Digital Converter，模数转换器

DAC:Digital-Analog Converter，数模转换器

BGP:Bandgap，带隙基准

WDT:Watch dog，看门狗

LSI:Low Speed Internal Clock，即低速 LRC 时钟

HSI:High Speed Internal Clock，即高速 HRC 时钟

PLL:Phase Lock Loop Clock，锁相环时钟，通常用作系统高速时钟

POR:Power-On Reset，即上电复位，芯片系统上电时产生的复位信号

NVR:Non-Volatile Register，flash 中区别于 main 区域之外的一块存储区域

IAP（在应用中编程）:IAP 是指可以在用户程序运行期间对微控制器的 Flash 进行重新编程。

ICP（在线编程）:ICP 是指可以在器件安装于用户应用电路板上时使用 JTAG 协议、 SWD 协议或自举程序对微控制器的 Flash 进行编程。

CW:Clock wise，顺时针

CCW:Counter clock wise，逆时针

Option bytes: 选项字节，保存在 Flash 中的 MCU 配置字节

2 系统概况

2.1 简述

LKS32MC07x 系列 MCU 是凌鸥增强版主线型 MCU，配备 96MHz ARM Cortex-M0 内核，以及必要的模拟和外设资源。

2.2 特性

- 96MHz 32 位 Cortex-M0 内核，32bit 硬件除法协处理器
- 4 通道 DMA
- 10uA 低功耗休眠模式
- -40~105℃工业级工作温度范围
- MCU 采用 2.5V~5.5V 单电源供电，内部集成数字供电 LDO
- 超强抗静电和群脉冲能力

2.2.1 存储

- 64/128kB Flash，数据防盗功能
- 12kB RAM

2.2.2 时钟

- 内置 8MHz 高精度 RC 时钟，全温度范围精度±1%
- 内置 32kHz 低速时钟，供低功耗模式使用
- 内部 PLL 可提供最高 96MHz 时钟

2.2.3 外设

- 2 路 UART
- 1 路 SPI
- 1 路 IIC
- 通用 16/32 位 Timer，支持捕捉和边沿对齐 PWM，Timer0 支持中心对齐 PWM
- 电机控制专用 PWM 模块，支持 2 组各 3 对 PWM 输出，死区可配置

- Hall 信号专用接口，支持测速、去抖
- 硬件看门狗
- 最多支持 64 路 GPIO

2.2.4 模拟模块

- 集成 2 路 12bit SAR ADC，3Msps 采样及转换速率，共 14 个外部通道
- 集成 4 路 OPA，可设置为差分 PGA 模式
- 集成 3 路比较器
- 集成 2 路 12bit DAC 数模转换器，作为内部比较器输入
- 内置 1.2V 0.5%精度电压基准源
- 内置 1 路低功耗 LDO 和电源监测电路
- 集成高精度、低温飘高频 RC 时钟

2.3 系统框图

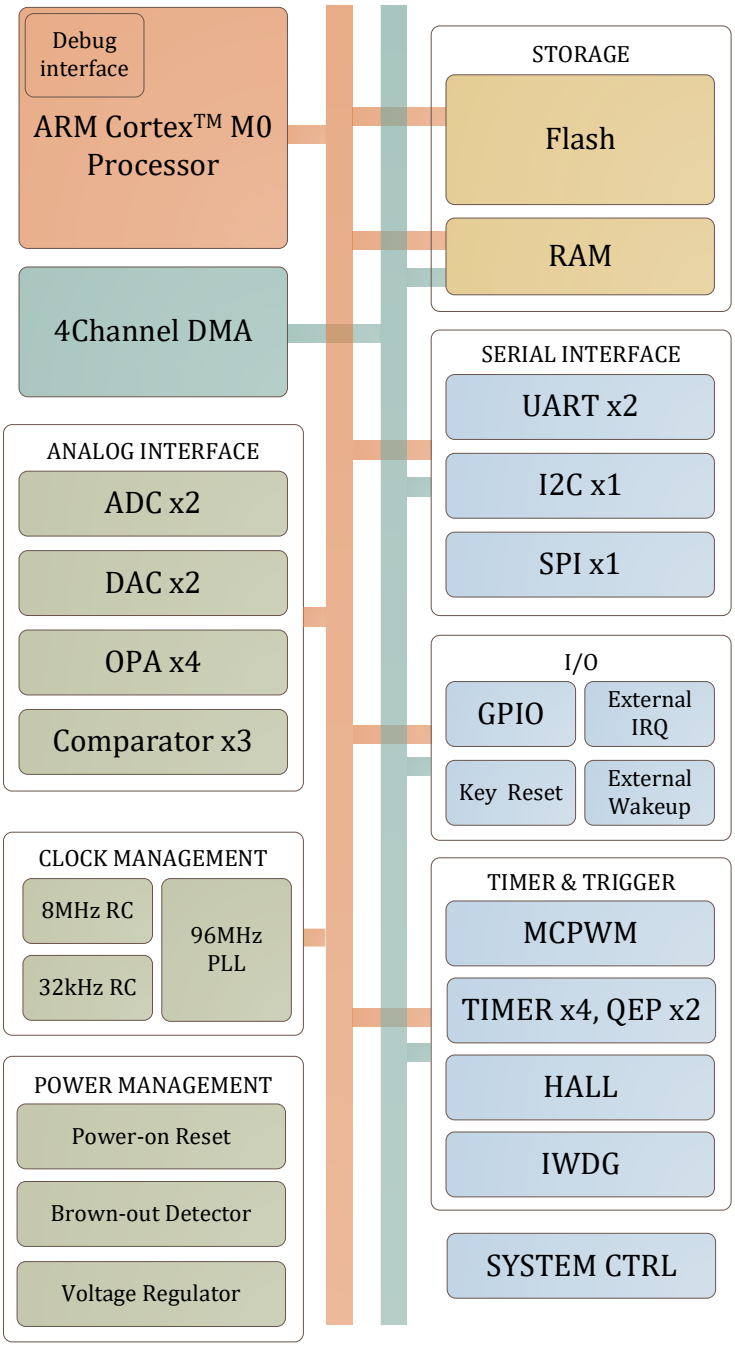


图 2-1 LKS32MC07x 系统框图

3 地址空间

数据字节以小端格式存放在存储器中。一个字里的最低地址字节被认为是该字的最低有效字节，而最高地址字节是最高有效字节。其他所有没有分配给片上存储器和外设的存储器空间都是保留的地址空间。

表 3-1 系统地址空间分配

类型	设备	开始地址	结束地址	空间大小	时钟/软复位
CODE	FLASH	0x0000_0000	0x0001_FFFF	128kB	同总线
	FLSCR	0x0002_0000	0x0002_00FF	256B	同总线
RAM	RAM	0x2000_0000	0x2000_2FFF	12kB	同总线
	SYS	0x4000_0000	0x4000_00FF	256B	同总线
	SPI0	0x4001_0000	0x4001_00FF	256B	外设门控时钟[0] 软复位[0]
	I2C0	0x4001_0100	0x4001_01FF	256B	外设门控时钟[1] 软复位[1]
	CMP	0x4001_0200	0x4001_02FF	256B	外设门控时钟[2] 软复位[2]
	HALL0	0x4001_0300	0x4001_03FF	256B	外设门控时钟[3] 软复位[3]
	ADC0	0x4001_0400	0x4001_04FF	256B	外设门控时钟[12] 软复位[12]
	ADC1	0x4001_0500	0x4001_05FF	256B	外设门控时钟[13] 软复位[13]
	TIMER0	0x4001_0600	0x4001_06FF	256B	外设门控时钟[4] 软复位[4]
	TIMER1	0x4001_0700	0x4001_07FF	256B	外设门控时钟[5] 软复位[5]
	TIMER2	0x4001_0800	0x4001_08FF	256B	外设门控时钟[6] 软复位[6]
	TIMER3	0x4001_0900	0x4001_09FF	256B	外设门控时钟[7] 软复位[7]
	QEP0	0x4001_0A00	0x4001_0AFF	256B	外设门控时钟[8] 软复位[8]
	QEP1	0x4001_0B00	0x4001_0BFF	256B	外设门控时钟[9] 软复位[9]
	MCPWM0	0x4001_0C00	0x4001_0CFF	256B	外设门控时钟[10] 软复位[10]
	GPIO	0x4001_0D00	0x4001_0EFF	512B	外设门控时钟[11] 软复位[11]
	CRC0	0x4001_0F00	0x4001_0FFF	256B	外设门控时钟[16] 软复位[16]
	UART0	0x4001_1000	0x4001_10FF	256B	外设门控时钟[14] 软复位[14]

	UART1	0x4001_1100	0x4001_11FF	256B	外设门控时钟[15] 软复位[15]
	DMA0	0x4001_1200	0x4001_12FF	256B	外设门控时钟[18] 软复位[18]
	CAN0	0x4001_1300	0x4001_14FF	512B	外设门控时钟[19] 软复位[19]
	SIF0	0x4001_1500	0x4001_15FF	256B	外设门控时钟[20] 软复位[20]
	Resv.	0x4001_1600	0x4001_16FF	256B	同总线
	AON	0x4001_1700	0x4001_17FF	256B	同总线
	CL0	0x4001_1800	0x4001_18FF	256B	外设门控时钟[21] 软复位[21]
	DSP0	0x4001_2000	0x4001_3FFF	8kB	外设门控时钟[17] 软复位[17]

以上外设门控时钟控制寄存器，请参考 6.3.7 SYS_CLK_FEN，软复位控制寄存器请参考 6.3.8 SYS_SFT_RST。

4 中断

嵌套向量中断控制器位于 CPU 核内部。当中断事件发生时，通知 CPU 暂停主程序执行，按照优先级设定跳转进入中断服务函数。

LKS32MC07x 系列芯片共有 25 个外部中断源。

中断控制器最多支持 4 个中断优先级可供编程选择。

表 4-1 中断号分布

中断号	说明	中断号	说明
-14	NMI		
-13	HardFault		
-12	保留		
-11			
-10			
-9			
-8			
-7			
-6			
-5	SVCall		
-4	保留		
-3			
-2	PendSV		
-1	SysTick		
0	TIMER0_IRQn	16	MCPWM0_IRQn
1	TIMER1_IRQn	17	MCPWM1_IRQn
2	TIMER2_IRQn	18	DMA0_IRQn
3	TIMER3_IRQn	19	CAN0_IRQn
4	QEP0_IRQn	20	SIF0_IRQn
5	QEP1_IRQn	21	唤醒中断 WAKE_IRQn
6	I2C0_IRQn	22	软件中断 SW_IRQn
7	SPI0_IRQn	23	AVDD 电压过低 PWRDN_IRQn
8	GPIO_IRQn	24	CL0_IRQn
9	HALL0_IRQn	25	Reserved
10	UART0_IRQn	26	Reserved
11	UART1_IRQn	27	Reserved
12	DSP0_IRQn	28	Reserved
13	CMP_IRQn	29	Reserved
14	ADC0_IRQn	30	Reserved
15	ADC1_IRQn	31	Reserved

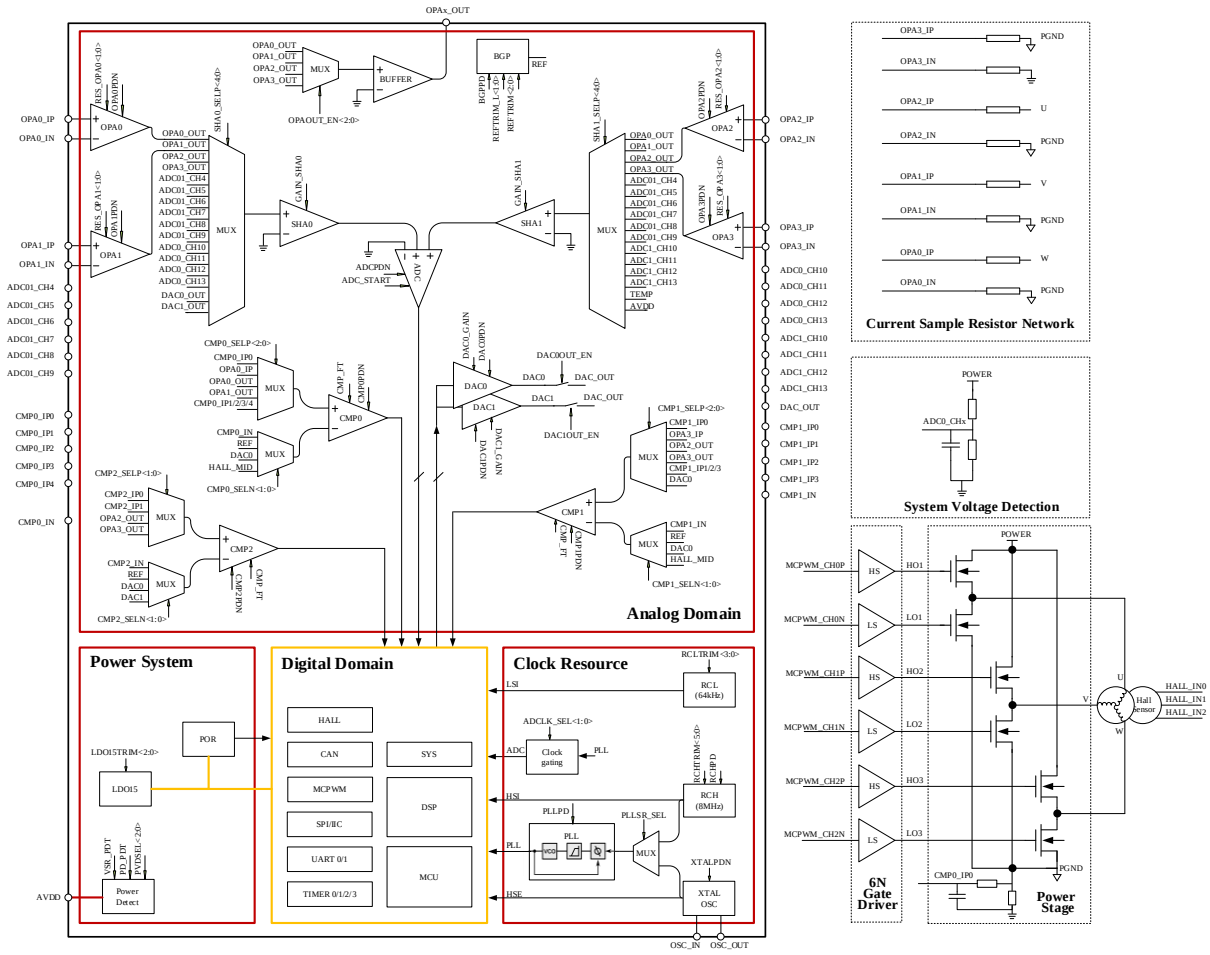
5 模拟电路

5.1 简述

模拟电路包含以下模块:

- 集成 2 路 12BIT SAR ADC，采样率 3MHz，每路最多支持 16 通道，包括 4 个运放输出及 10 个外部 ADC 通道共计 14 个可选 ADC 通道信号。
- 集成 4 路运算放大器，可设置为 PGA 模式
- 集成 3 路比较器，可设置迟滞模式
- 集成 2 路 12BIT 数模转换器
- 内置±2℃温度传感器
- 内置高精度基准源

各个模块之间的相互关系、以及各模块的控制寄存器（寄存器的说明见下文“模拟寄存器表”）如下图所示。



*ADC01_CH4~ADC01_CH9 为 ADC0 和 ADC1 公用通道

图 5-1 模拟电路功能框图

5.1.1 电源管理系统

电源管理系统由上电/掉电复位模块（POR）、电源检测模块（PVD）、LDO15 模块组成。

该芯片由 2.5~5V 单电源供电，以节省芯片外的电源成本。芯片内部集成一路 LDO15 给内部所有数字电路、PLL 模块供电。

LDO 上电后自动开启，无需软件配置。

POR 模块监测 LDO15 的电压，在 LDO15 电压低于 1.26V 时（例如上电之初，或者掉电时），为数字电路提供复位信号以避免数字电路工作产生异常。

PVD 模块对 5V 输入电源进行检测，如低于某一设定阈值，则产生报警（中断）信号以提醒 CPU。 中断提醒阈值可通过寄存器 PVDSEL[9:8]设置为不同的电压。PVD 模块可通过设置 PD_PDT=1 关闭。发生欠压事件时，仅产生欠压标志，不会产生复位。

当发生电压过低事件时，会触发产生电源电压过低中断，对应中断号 23，详见第 4 章。

PVDSEL[9:8]/ PD_PDT 的说明见模拟寄存器 SYS_AFE_REG6

电源电压过低标志见模拟前端信息寄存器 SYS_AFE_DEG

5.1.2 时钟系统

时钟系统包括内部 32kHz RC 时钟、内部 8MHz RC 时钟、PLL 电路组成。

32kHz RC 时钟 LSI 主要用于系统内的看门狗模块以及复位信号滤波等，也可用作 MCU 主时钟。8MHz RC 时钟可作为 MCU 主时钟使用。PLL 最高可提供 96MHz 的时钟，通常 MCU 使用 PLL 时钟作为系统主时钟。

32kHz 和 8MHz RC 时钟均带有出厂校正，32kHz RC 时钟在-40~105℃范围内的精度为±50%，8MHz RC 时钟在该温度范围的精度为±1%。

8MHz RC 时钟通过设置 RCHPD='0'打开（默认打开，写 1 关闭），RC 时钟需要 BGP 电压基准源模块提供基准电压和电流，因此开启 RC 时钟时实际上连带开启了 BGP 模块（BGPPD='0'）。芯片上电的默认状态下，8MHz RC 时钟和 BGP 模块都是开启的。32kHz RC 时钟始终开启，不能关闭。

PLL 对 8MHz RC 时钟进行倍频，给 MCU、ADC 等模块提供更高速的工作时钟。MCU 和 PWM 模块的最高时钟为 96MHz，ADC 模块最高时钟 48MHz。

PLL 通过设置 PLLPDN='1'打开（默认关闭，设 1 打开），开启 PLL 模块之前，同样也需要开启 BGP 模块。开启 PLL 之后，PLL 需要 8us 的稳定时间来输出稳定时钟。芯片上电的默认状态下，RCH 时钟和 BGP 模块都是开启的，但 PLL 默认是关闭的，需要软件来开启。

晶体起振电路内置了放大器，需在 OSC_IN/OSC_OUT 管脚之间接入一个晶体，且在 OSC_IN/OSC_OUT 引脚各接一个 15pF 电容到地，设置 XTALPDN=1 即可起振。OSC_IN 也可以直接输入一个 8MHz 的方波代替晶体。

BGPPD /PLLPDN 的说明见模拟寄存器 SYS_AFE_REG5

5.1.3 基准电压源

基准源电路(BGP REF: Bandgap reference)为 ADC、DAC、RC 时钟、PLL、温度传感器、运算放大器、比较器和 FLASH 提供基准电压和电流，使用上述任何一个模块之前，都需要开启 BGP 基准电压源。

芯片上电的默认状态下，BGP 模块是开启的。通过设置 BGPPD ='0'将基准源打开，从关闭到开启，BGP 需要约 6us 达到稳定。BGP 输出电压约 1.2V，精度为±0.8%

BGPPD 的说明见模拟寄存器 SYS_AFE_REG5

5.1.4 ADC 模块

请参考第 12 章。

5.1.5 运算放大器

芯片集成 4 路输入输出轨到轨（rail-to-rail）运算放大器，内置反馈电阻，外部引脚上需串联一个电阻 R_0 到信号源。反馈电阻 R_2:R_1 的阻值可通过寄存器 RES_OPAx[1:0]设置，以实现不同的放大倍数。

RES_OPAx<1:0>的说明见模拟寄存器 SYS_AFE_REG0

放大器的结构示意图如下所示:

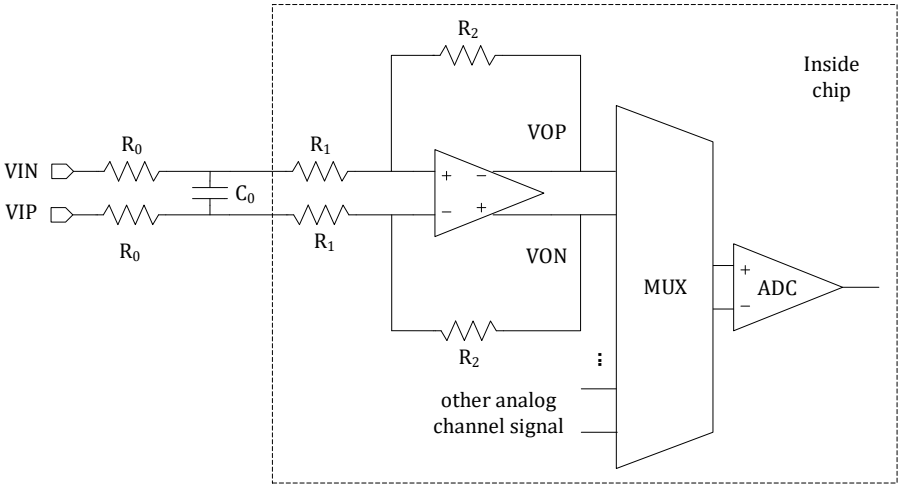


图 5-2 放大器框图

图中两个 R_0 是片外需放置的电阻，阻值必须相等，最终的放大倍数为 $R_2/(R_1+R_0)$。

对于 MOS 管电阻直接采样的应用，由于 MOS 下管关断、上管导通时信号会升高到数十 V 的电源电压，为减小此时往芯片引脚里流入的电流，建议接>20kΩ 的外部电阻。

对于分流电阻采样的应用，建议接 100~2KΩ 的外部电阻。C_0 为信号滤波电容，和 R_0 形成一阶 RC 滤波电路。R_0 的具体阻值可根据 R_0*C_0 的滤波常数而定。如果信号上噪声较小不需要滤波、或者信号需要很大的带宽（较快的响应速度），则 C_0 可以不加。

放大器可通过设置 OPAOUT_EN 选择将放大器的输出信号通过 BUFFER 送至 P2.7 管脚口进行测量和应用（对应关系见 datasheet 芯片管脚说明）。因为有 BUFFER 存在，在运放正常工作模式下也可以选择送一路运放输出信号出来。送至 IO 口的信号为运放的正端输出信号（对应放大器详细框图与说明可见 Wiki：运放差分和单端工作模式的区别）。

OPAOUT_EN 的说明见模拟寄存器 SYS_AFE_REG2

芯片上电的默认状态下，放大器模块是关闭的。放大器可通过设置 OPAxPDN=1 打开。开启放大器之前，需要先开启 BGP 模块。

OPAxPDN 的说明见模拟寄存器 SYS_AFE_REG5

运放输入正负端内置钳位二极管，电机相线通过一个匹配电阻后直接接入输入端，从而简化了 MOSFET 电流采样的外置电路。

5.1.6 比较器

内置 3 路输入轨到轨（rail-to-rail）比较器，比较器比较速度可编程、迟滞电压可编程、信号源可编程。

比较器的比较延时可通过寄存器 CMP_FT 设置为＜30nS/200nS。迟滞电压通过 CMP_HYS 设置为 20mV/0mV。

比较器正端输入信号来源可以通过寄存器 CMP2_SELP[1:0]/ CMP1_SELP[2:0]/ CMP0_SELP[2:0]

进行设置；负端输入信号来源可以通过寄存器 CMPx_SELN[1:0]进行设置（x=0/1/2，代表比较器 CMP0/CMP1/CMP2）。

需说明的是，两个比较器负输入端的 BEMFx_MID 信号，是对比较器正输入端信号 CMPx_IP1/ CMPx_IP2/ CMPx_IP3 信号的平均，具体连接方式见图 5-3。其中电阻 R=8.2k 欧，图中的开关只有在比较器负输入端信号选择为 BEMFx_MID 之后才会导通，否则开关都处于断开状态。

BEMFx_MID 主要用于 BLDC 方波模式控制时，虚拟电机相线中心点电压，用于反电势过零点检测。三个相线分压后，分别接 CMPx_IP1、CMPx_IP2、CMPx_IP3，MCU 控制比较器负端选择 BEMFx_MID，比较器正端的多路选择器以分时复用的方式分别选择 CMPx_IP1、CMPx_IP2、CMPx_IP3，就可以比较出反电势过零点。

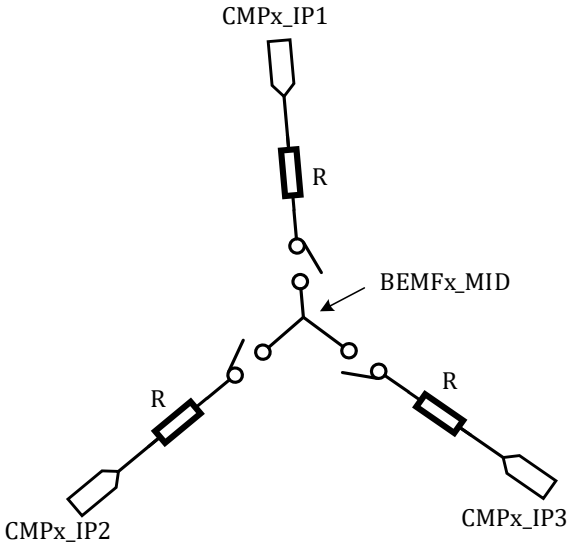


图 5-3 BEMFx_MID 信号

比较器输出结果，可以通过 CMP_DATA 输出数据寄存器读出。

CMP_FT 的说明见模拟寄存器 SYS_AFE_REG1

CMPx_SELN<1:0>/ CMPx_SELP<2:0>/ CMP_HYS 的说明见模拟寄存器 SYS_AFE_REG2/模拟寄存器 SYS_AFE_REG3

芯片上电的默认状态下，比较器模块是关闭的。比较器通过设置 CMPxPDN=1 (x=0,1)打开，开启比较器之前，需要先开启 BGP 模块。

CMPxPDN 的说明见模拟寄存器 SYS_AFE_REG5

5.1.7 温度传感器

芯片内置温度传感器，在-40~85℃范围内精度为 2℃。85~105℃范围内精度为 3℃。

测量时需选择内部基准，选择外部基准会引起结果的较大偏差。

芯片出厂前会经温度校正，校正值保存在 flash info 区。出厂校正时所用增益为 ADC_GAIN=0，建议应用时也选择此增益，可以使校正值更准确。

芯片上电的默认状态下，温度传感器模块是关闭的。开启传感器之前，需要先开启 BGP 模块。

温度传感器通过设置 TMPPDN=1 打开，开启到稳定需要约 2us，因此需在 ADC 测量传感器之前 2us 打开。

温度传感器信号连至 ADC1 的通道 14。

ADC 部分的设置参考模数转换器(ADC)章节

TMPPDN 的说明见模拟寄存器 SYS_AFE_REG5

温度传感器的典型曲线如下图所示:

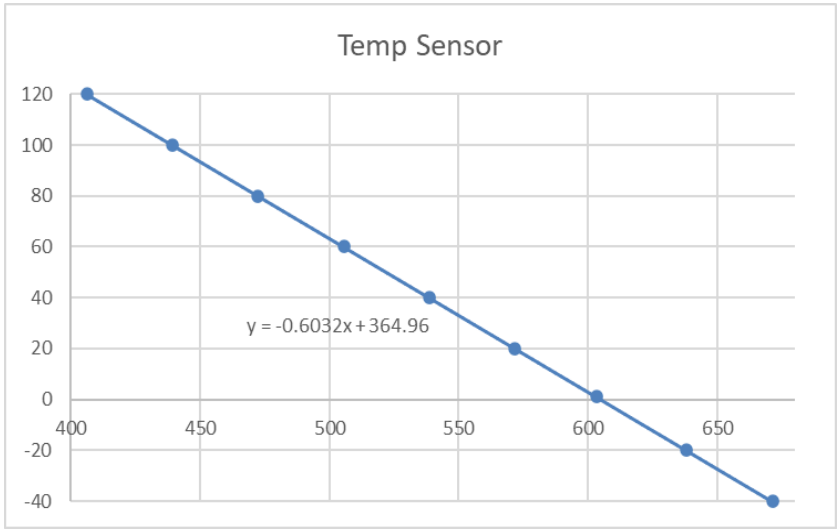


图 5-4 温度传感器曲线

图中 X 轴为温度传感器的温度信号所对应的 ADC 值，Y 轴为传感器所处的温度。测温时，按照如上要求配置传感器相关寄存器，并得到 ADC 值后，将 ADC 值作为 X 代入公式:

y=-0.6032x+364.96

求得的 Y 值即为此时的温度。

公式中有两个系数，a=-0.6032, b=364.96。对于不同的芯片，b 系数的值是不一样的。芯片出厂前会经过温度标定，将每颗芯片所对应的系数 b 写入 flash 的 info 区，地址为 0x000014D4。存储时，会将 b 系数小数点右移一位（乘 10）存入 info 区，小数点后第二位不进行保存。

同时为方便客户操作，系数 a 也会存入 flash info 区，地址为 0x000014D0。存储时，将 a 系数小数点右移四位（乘 10000）存入 info 区。

实际使用中，应从 flash info 区对应地址读出 a/b 系数，同时将读取到的 ADC 测到的当下温度传感器值代入公式，即可计算得到当下温度值，单位为摄氏度。计算时，需注意系数 a/b 在保存时小数点的位移数，即 a 系数应除以 10000，b 系数除以 10。

注意，上述计算公式，基于 ADC 右对齐实现。若换成左对齐，ADC 采样值需右移 4 位后，才能代入上述公式。

5.1.8 DAC 模块

芯片内置两路 12bit DAC，输出信号的最大量程可通过寄存器 DAC0_GAIN、DAC1_GAIN 设置为 1.2V/4.85V。

DAC0 可通过配置寄存器 DAC0OUT_EN=1，将 DAC0 输出送至 P0.0 管脚，；DAC1 可通过配置寄存器 DAC1OUT_EN=1，将 DAC1 输出送至 P0.0 管脚，可驱动>5kΩ 的负载电阻和 50pF 的负载电容。通常不会同时输出 DAC0 和 DAC1，以免造成信号竞争。

DAC 最大输出码率为 1MHz。

芯片上电的默认状态下，DAC 模块是关闭的。DAC0 可通过设置 DAC0PDN =1 打开，DAC1 可通过设置 DAC1PDN =1 打开，开启 DAC 模块之前，需要先开启 BGP 模块。

DAC 的输入数字信号寄存器为 SYS_AFE_DACx，低 12BIT 有效。信号范围是 0x000~0xFFF。0x000 对应零模拟量输出 0V，0xFFF 对应满量程模拟量输出为DAC_{fs}，如上文所述，DAC_{fs}的值可由 DACx_GAIN 寄存器进行设置。每一档信号(LSB)所对应的模拟信号幅度为$\frac{DAC_{fs}}{4096}$。若 SYS_AFE_DACx 的数字值为 Din，则该数字信号所对应的 DAC 输出模拟信号为$\frac{DAC_{fs}}{4096} * Din$

不同芯片，DAC 存在制造偏差，为抵消偏差，DAC 自带校准硬件模块。DAC 输出遵循公式 y=ax+b。x 为 SYS_AFE_DACx 填入值（理想值对应数字量）。a 来自 SYS_AFE_DACx_AMC 寄存器，b 来自 SYS_AFE_DACx_DC。硬件根据 SYS_AFE_DACx、SYS_AFE_DACx_AMC 和 SYS_AFE_DACx_DC 进行乘加从而求得校正后的数字量，送至 DACn 输入端，使得 DACn 最终输出的模拟值就是填入的理想数字量对应的值。系统上电后，默认加载 4.85V 的校准值，若换成其它量程，软件需读取 flash info 区域，重新加载到相应寄存器。其中 x=0,1，对应 DAC0 和 DAC1。

DAC0：

地址 0x00001450，为 4.85V 量程的 a 参数，地址 0x00001454，为 4.85V 量程的 b 参数。

地址 0x00001458，为 1.2V 量程的 a 参数，地址 0x0000145C，为 1.2V 量程的 b 参数。

DAC1：

地址 0x00001460，为 4.85V 量程的 a 参数，地址 0x00001464，为 4.85V 量程的 b 参数。

地址 0x00001468，为 1.2V 量程的 a 参数，地址 0x0000146C，为 1.2V 量程的 b 参数。

DAC 输出的模拟信号，除了可以送至 IO 口供外部模块使用外，还可通过配置寄存器连至芯片内部的 2 路比较器负端，作为比较器的基准信号使用。详见比较器章节。

DAC0OUT_EN/DAC1OUT_EN 的说明见模拟寄存器 SYS_AFE_REG3

DAC0_GAIN/DAC1_GAIN 的说明见模拟寄存器 SYS_AFE_REG1

DAC0PDN/DAC1PDN 的说明见模拟寄存器 SYS_AFE_REG5

SYS_AFE_DAC0/SYS_AFE_DAC1 的说明见寄存器 SYS_AFE_DAC0, SYS_AFE_DAC1

5.2 寄存器

5.2.1 地址分配

模拟寄存器 SYS_AFE_REG0~SYS_AFE_REG6，对应地址为 0x4000_0010~0x4000_0028。其中保留寄存器(Res)必须全部配置为 0（芯片上电后会被复位为 0）。其他寄存器根据应用场合需要进行配置。

模拟寄存器基地址为 0x4000_0000。

表 5-1 系统控制寄存器

名称	偏移	说明
SYS_AFE_INFO	0x04	芯片版本信息寄存器
SYS_AFE_DBG	0x08	调试寄存器
SYS_AFE_REG0	0x10	模拟配置寄存器 0
SYS_AFE_REG1	0x14	模拟配置寄存器 1
SYS_AFE_REG2	0x18	模拟配置寄存器 2
SYS_AFE_REG3	0x1C	模拟配置寄存器 3
SYS_AFE_REG4	0x20	模拟配置寄存器 4
SYS_AFE_REG5	0x24	模拟配置寄存器 5
SYS_AFE_REG6	0x28	模拟配置寄存器 6
SYS_AFE_REG7	0x2C	模拟配置寄存器 7
SYS_AFE_DAC_CTRL	0x5C	DAC 控制寄存器
SYS_AFE_DAC0	0x60	DAC0 数字量寄存器
SYS_AFE_DAC1	0x64	DAC1 数字量寄存器
SYS_AFE_DAC0_AMC	0x68	DAC0 数字量增益校正寄存器
SYS_AFE_DAC0_DC	0x6C	DAC0 数字量直流偏置寄存器
SYS_AFE_DAC1_AMC	0x70	DAC1 数字量增益校正寄存器
SYS_AFE_DAC1_DC	0x74	DAC1 数字量直流偏置寄存器

5.2.2 SYS_AFE_INFO 芯片版本信息寄存器

地址:0x4000_0004

复位值:根据 wafer 版本不同

表 5-2 SYS_AFE_INFO 芯片版本信息寄存器

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

位置	位名称	说明

[31:8]		未使用
[7:4]	Series	芯片型号信息
[3:0]	Version	芯片版本信息

此寄存器与用户无关。

5.2.3 SYS_AFE_DBG 调试寄存器

地址:0x4000_0008

复位值:0x0

表 5-3 SYS_AFE_DBG AFE 调试寄存器

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

位置	位名称	说明
[31:8]		未使用
[15]	PWR_WEAK	供电低于掉电监测阈值
[14:0]		未使用

PWR_WEAK 标志在当供电电压低于掉电监测电路设定的阈值时置位，同时将产生 CPU 掉电中断（中断号 23）。在供电恢复，高于阈值后清零。此标志为只读位，无法软件清除。

掉电监测的使用请参考 5.1.1 电源管理系统。

5.2.4 SYS_AFE_REG0 模拟配置寄存器 0

地址：0x4000_0010

复位值：0x0

表 5-4 SYS_AFE_REG0 模拟配置寄存器 0

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

位置	位名称	说明
[31:16]		未使用
[15:10]		保留位，必须为 0
[9:8]	IT_OPA	OPA 偏置电流调节，默认配置 0 00: ×1，建议 RES_OPAx 选择非最大放大倍数时，选用此配置 01: ×1.2

		10: ×1.5，建议 RES_OPAx 选择 00 最大放大倍数时，选用此配置 11: ×2
[7:6]	RES_OPA3	运放 3 反馈电阻 00: 320k:10k 01: 160k:10k 10: 80k:10k 11: 40k:10k
[5:4]	RES_OPA2	运放 2 反馈电阻 00: 320k:10k 01: 160k:10k 10: 80k:10k 11: 40k:10k
[3:2]	RES_OPA1	运放 1 反馈电阻 00: 320k:10k 01: 160k:10k 10: 80k:10k 11: 40k:10k
[1:0]	RES_OPA0	运放 0 反馈电阻 00: 320k:10k 01: 160k:10k 10: 80k:10k 11: 40k:10k

5.2.5 SYS_AFE_REG1 模拟配置寄存器 1

地址：0x4000_0014

复位值：0x0

表 5-5 SYS_AFE_REG1 模拟配置寄存器 1

15	14	13	12	11 10 9 8	7	6	5	4	3 2 1	0
Res.	Res.	Res.	Res.	Res.	DAC1_GAIN	DAC0_GAIN	Res.	REF2VDD	Res.	CMP_FT
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:16]		未使用
[15]	Reserved	保留位，必须为 0
[14]	Reserved	保留位，必须为 0
[13]	Reserved	保留位，必须为 0
[12]	Reserved	保留位，必须为 0
[11:8]	Reserved	保留位，必须为 0
[7]	DAC1_GAIN	DAC1 输出档位配置 0:满量程为 4.85V; 1:满量程为 1.2V
[6]	DAC0_GAIN	DAC0 输出档位配置

		0:满量程为 4.85V; 1:满量程为 1.2V
[5]	Reserved	保留位，必须为 0
[4]	REF2VDD	ADC 基准选择 1: 使用外部输入电源作为 ADC 基准; 0: 使用默认内部 REF 2.4V 作为 ADC 基准源
[3:1]	Reserved	保留位，必须为 0
[0]	CMP_FT	比较器快速比较 1:使能，比较速度小于 50ns 0: 不使能，比较速度小于 200ns

当 REF2VDD 设置为 1，使用外部电源作为 ADC 基准时，如果选择 ADCx_GAIN=0，则量程变为 7.5V；如果选择 ADC_GAIN=1，则量程变为 15V。

在 DAC 输出挡位配置中，4.85V 量程，在接近 4.45V 时提前饱和；1.2V 量程，在接近 1.13V 时提前饱和。

5.2.6 SYS_AFE_REG2 模拟配置寄存器 2

地址：0x4000_0018

复位值：0x0

表 5-6 SYS_AFE_REG2 模拟配置寄存器 2

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP2_SELP		Res.					XTRSEL	Res.			REF_AD_EN	LDOOUT_EN	OPAOUT_EN		
RW		RW					RW	RW			RW	RW	RW		
0		0					0	0			0	0	0		

位置	位名称	说明
[31:16]		未使用
[15:14]	CMP2_SELP	比较器 2 信号正端选择 00: 连 CMP2_IP0 01: 连 CMP2_IP1 10: 连 OPA2_OUT 11: 连 OPA3_OUT
[13:9]	Reserved	保留位，必须为 0
[8]	XTRSEL	晶体起振电路电阻调节 XTRSEL=1: P 端电阻增加一倍，从而降低晶振起振速度
[7:5]	Reserved	保留位，必须为 0
[4]	REF_AD_EN	输出 1.2V Reference 到 P2.7
[3]	LDOOUT_EN	输出 LDO15 到 P2.7
[2:0]	OPAOUT_EN	使能 OPAx 输出信号送至 IO 口 p2_7 000:不输出; 001:输出 OPA0 信号到 IO 口;

		010:输出 OPA1 信号到 IO 口; 011:输出 OPA2 信号到 IO 口; 100:输出 OPA3 信号到 IO 口; 101~111: 不允许此配置

5.2.7 SYS_AFE_REG3 模拟配置寄存器 3

地址：0x4000_001C

复位值：0x0

表 5-7 SYS_AFE_REG3 模拟配置寄存器 3

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAC1OUT_EN	CMP1_SELP			DAC0OUT_EN	CMP0_SELP			CMP_HYS	Res.	CMP1_SELN		CMP0_SELN		CMP2_SELN	
RW	RW			RW	RW			RW	RW	RW		RW		RW	
0	0			0	0			0	0	0		0		0	

位置	位名称	说明
[31:16]		未使用
[15]	DAC1OUT_EN	DAC1 输出到 IO 使能 0: 不使能 1:使能输出到 IO P0[0]
[14:12]	CMP1_SELP	比较器 1 信号正端选择 000: CMP1_IP0 001: OPA3_IP 010: OPA2_OUT 011: OPA3_OUT 100: CMP1_IP1 101: CMP1_IP2 110: CMP1_IP3 111: DAC0 输出 说明:上述除 AVSS/OPA2_OUT/OPA3_OUT 外都为管脚名称，请参看 datasheet 里管脚定义章节
[11]	DAC0OUT_EN	DAC0 输出到 IO 使能 0: 不使能 1:使能输出到 IO P0[0]
[10:8]	CMP0_SELP	比较器 0 信号正端选择 000: CMP0_IP0 001: OPA0_IP 010: OPA0_OUT 011: OPA1_OUT 100: CMP0_IP1 101: CMP0_IP2 110: CMP0_IP3 111: CMP0_IP4

		说明:上述除 OPA0_OUT/OPA1_OUT 外都为管脚名称，请参看 datasheet 里管脚定义章节
[7]	CMP_HYS	比较器回差选择，采用默认配置 0: 20mv; 1: 0mv
[6]	Reserved	保留位，必须为 0
[5:4]	CMP1_SELN	比较器 1 信号负端选择 00: CMP1_IN 01: REF 10: DAC0 输出 11: BEMF1_MID 说明:上述 CMP1_IN 为管脚名称，请参看 datasheet 里管脚定义章节；REF 为芯片内部 1.2V BANDGAP 基准源；DAC 输出即为芯片内部 DAC 模块输出模拟信号；BEMF1_MID 为 CMP1_IP1, CMP1_IP2, CMP1_IP3 信号经电阻星形连接后得到的平均值
[3:2]	CMP0_SELN	比较器 0 信号负端选择 00: CMP0_IN 01: REF 10: DAC0 输出 11: BEMF0_MID 说明: 上述 CMP0_IN 为管脚名称，请参看 datasheet 里管脚定义章节；REF 为芯片内部 1.2V BANDGAP 基准源；DAC 输出即为芯片内部 DAC 模块输出模拟信号；BEMF0_MID 为 CMP0_IP1, CMP0_IP2, CMP0_IP3 信号经电阻星形连接后得到的平均值
[1:0]	CMP2_SELN	比较器 2 信号负端选择 00: CMP2_IN 01: REF 10: DAC0 输出 11: DAC1 输出

5.2.8 SYS_AFE_REG4 模拟配置寄存器 4

地址：0x4000_0020

复位值：0x0

表 5-8 SYS_AFE_REG4 模拟配置寄存器 4

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

位置	位名称	说明
[31:16]		
[15]		未使用

[14]	Reserved	保留位，必须为 0
[13]		未使用
[12]	Reserved	保留位，必须为 0
[11:8]		未使用
[7]		未使用
[6]	Reserved	保留位，必须为 0
[5]		未使用
[4]	Reserved	保留位，必须为 0
[3:0]		未使用

5.2.9 SYS_AFE_REG5 模拟配置寄存器 5

地址：0x4000_0024

复位值：0x0

表 5-9 SYS_AFE_REG5 模拟配置寄存器 5

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLLPDN	XTALPDN	TMPPDN	DAC0PDN	CMP2PDN	RCHPD	DAC1PDN	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

位置	位名称	说明
[31:16]		未使用
[15]	PLLPDN	PLL 关闭控制 0:关闭 PLL（默认） 1:开启 PLL
[14]	XTALPDN	晶体起振电路开关 0:关闭（默认） 1:开启
[13]	TMPPDN	温度传感器开关 0:关闭（默认） 1:开启
[12]	DAC0PDN	DAC0 开关 0:关闭（默认） 1:开启
[11]	CMP2PDN	CMP2 开启使能 0:关闭（默认） 1:开启
[10]	RCHPD	RCH 时钟关闭使能 0:开启（默认） 1:关闭

[9]	DAC1PDN	DAC1 开关 0:关闭（默认） 1:开启
[8]	BGPPD	BGP 关闭使能 0:开启（默认） 1:关闭
[7]	CMP1PDN	CMP1 开启使能 0:关闭（默认） 1:开启
[6]	CMP0PDN	CMP0 开启使能 0:关闭（默认） 1:开启
[5]	OPA3PDN	OPA3 开启使能 0:关闭（默认） 1:开启
[4]	OPA2PDN	OPA2 开启使能 0:关闭（默认） 1:开启
[3]	OPA1PDN	OPA1 开启使能 0:关闭（默认） 1:开启
[2]	OPA0PDN	OPA0 开启使能 0:关闭（默认） 1:开启
[1]	Reserved	保留位，需全部为’0’
[0]	ADCPDN	ADC 开启使能 0:关闭（默认） 1:开启

如果 SYS_CLK_CFG 选择 PLL 时钟，则 PLLPDN 是被硬件控制的，软件配置 PLLPDN 关闭 PLL 无效。关闭 PLL 需要 PLLPDN=0，且 SYS_CLK_CFG 不选择 PLL 作为芯片主时钟，这两个条件都须满足。同理如果 SYS_CLK_CFG 选择 HRC 时钟，则 RCHPD 是被硬件控制的，软件直接配置 RCHPD 关闭 RCH 无效。关闭 PLL 需要 RCHPD=1，且芯片进入休眠。如果芯片主时钟为 PLL 时钟，且 HRC 为 PLL 参考时钟，则 RCH 也是被硬件控制的。由于 RCH 和 PLL 依赖 BGP，所以 BGPPD 也是硬件控制的，在芯片使用了 RCH 或 PLL 时，软件直接配置 BGPPD 关闭 BGP 无效。关闭 BGP 需要先顺序关闭 PLL 和 RCH，且芯片进入休眠。

5.2.10 SYS_AFE_REG6 模拟配置寄存器 6

地址：0x4000_0028

复位值：0x0

表 5-10 SYS_AFE_REG6 模拟配置寄存器 6

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

PLLSR_SEL	Res.	PVDSEL	Res.	Res.	VSR_PDT	PD_PDT
RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0

位置	位名称	说明
[31:16]		未使用
[15]	PLLSR_SEL	PLL 时钟源选择 0: 使用 RCH 作为输入时钟源; 1: 使用 XTAL OSC 作为输入时钟源
[14:10]	Reserved	保留位，必须为 0
[9:8]	PVDSEL	电源掉电监测阈值 V_{th} 选择 00: 4.00V 01: 3.75V 10: 3.50V 11: 3.25V (SYS_AFE_INFO.VERSION<=3) 2.50V (SYS_AFE_INFO.VERSION=4) 以上阈值为 VSR_PDT=0，即选择低功耗基准源 V_{BGP} 时的设定值；如果 VSR_PDT=1，选择 DAC 输出作为基准，则阈值相应调整为 V_{th}*DAC0_OUT/V_{BGP}
[7:2]	Reserved	保留位，必须为 0
[1]	VSR_PDT	掉电检测基准源选择，其中低功耗基准源 V_{BGP} 为 1.3V 左右，DAC 输出则可以通过软件配置 0: 选择低功耗基准源 V_{BGP}; 1: 选择 DAC0 输出
[0]	PD_PDT	关闭掉电检测电路 0: 开启 1: 关闭

5.2.11 SYS_AFE_REG7 模拟配置寄存器 7

地址：0x4000_002C

复位值：0x0

表 5-11 SYS_AFE_REG7 模拟配置寄存器 7

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

位置	位名称	说明
[31:16]		未使用
[15:5]	Reserved	保留位，必须为 0
[4]	ADCLKSEL	ADC 时钟频率选择 0: 48MHz 1: 24MHz
[3:0]	Reserved	保留位，需全部为'0'

5.2.12 SYS_AFE_DAC_CTRL DAC 控制寄存器

地址:0x4000_005C

复位值:0x0

表 5-12 SYS_AFE_DAC_CTRL DAC 控制寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DAC1_STEP			TIMER3_TRIG_DAC1	TIMER2_TRIG_DAC1	TIMER1_TRIG_DAC1	TIMER0_TRIG_DAC1		DAC0_STEP			TIMER3_TRIG_DAC0	TIMER2_TRIG_DAC0	TIMER1_TRIG_DAC0	TIMER0_TRIG_DAC0
	RW			RW	RW	RW	RW		RW			RW	RW	RW	RW
	0			0	0	0	0		0			0	0	0	0

位置	位名称	说明
[31:15]		未使用
[14:12]	DAC1_STEP	SYS_AFE_DAC1 步进值，3bit 有符号数，范围-4~3
[11]	TIMER3_TRIG_DAC1	TIMER3 过零事件触发 DAC1 步进使能，高有效
[10]	TIMER2_TRIG_DAC1	TIMER2 过零事件触发 DAC1 步进使能，高有效
[9]	TIMER1_TRIG_DAC1	TIMER1 过零事件触发 DAC1 步进使能，高有效
[8]	TIMER0_TRIG_DAC1	TIMER0 过零事件触发 DAC1 步进使能，高有效
[7]		未使用
[6:4]	DAC0_STEP	SYS_AFE_DAC0 步进值，3bit 有符号数，范围-4~3
[3]	TIMER3_TRIG_DAC0	TIMER3 过零事件触发 DAC0 步进使能，高有效
[2]	TIMER2_TRIG_DAC0	TIMER2 过零事件触发 DAC0 步进使能，高有效
[1]	TIMER1_TRIG_DAC0	TIMER1 过零事件触发 DAC0 步进使能，高有效
[0]	TIMER0_TRIG_DAC0	TIMER0 过零事件触发 DAC0 步进使能，高有效

DACx 允许使用 Timer 的过零事件作为触发来更新 SYS_AFE_DACx 的值，每触发一次，DAC 的设置值在原值的基础上步进一次（递增或递减），步进值由 SYS_AFE_CTRL.DACx_STEP 设定，x 为 0/1。

5.2.13 SYS_AFE_DAC0 DAC0 数字量寄存器

地址:0x4000_0060

复位值:0x0

表 5-13 SYS_AFE_DAC 0DAC0 数字量寄存器

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

位置	位名称	说明
[31:12]		未使用
[11:0]	DAC_IN	DAC 待转换的数字量输入

5.2.14 SYS_AFE_DAC1 DAC1 数字量寄存器

地址:0x4000_0064

复位值:0x0

表 5-14 SYS_AFE_DAC1 DAC1 数字量寄存器

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

位置	位名称	说明
[31:12]		未使用
[11:0]	DAC_IN	DAC 待转换的数字量输入

5.2.15 SYS_AFE_DAC0_AMC DAC0 增益校正寄存器

地址：0x4000_0068

复位值：0x0

表 5-15 SYS_AFE_DAC0_AMC DAC0 增益校正寄存器

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

位置	位名称	说明
[31:10]		未使用
[9:0]	AMC	DAC0 增益校正值，10bit 无符号定点数，B[9]为整数，B[8:0]为小数

5.2.16 SYS_AFE_DAC0_DC DAC0 直流偏置寄存器

地址：0x4000_006C

复位值：0x0

表 5-16 SYS_AFE_DAC0_DC DAC0 直流偏置寄存器

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

位置	位名称	说明
[31:10]		未使用
[9:0]	DC	DAC0 直流偏置，10bit 有符号数，B[9]为符号位

5.2.17 SYS_AFE_DAC1_AMC DAC1 增益校正寄存器

地址：0x4000_0070

复位值：0x0

表 5-17 SYS_AFE_DAC1_AMC DAC1 增益校正寄存器

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

位置	位名称	说明
[31:10]		未使用
[9:0]	AMC	DAC1 增益校正值，10bit 无符号定点数，B[9]为整数，B[8:0]为小数

5.2.18 SYS_AFE_DAC1_DC DAC1 直流偏置寄存器

地址：0x4000_0074

复位值：0x0

表 5-18 SYS_AFE_DAC1_DC DAC1 直流偏置寄存器

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

位置	位名称	说明
[31:10]		未使用
[9:0]	DC	DAC1 直流偏置，10bit 有符号数，B[9]为符号位

DAC 增益校正，记模拟输出的 12bitDAC 数值为 DAC_raw，经过校正后的 12bit DAC 数值为

DAC_cali

DAC_cali = DAC_raw * AMC –DC;

其中 AMC 为 DAC 增益校正系数，为 10bit 定点无符号数，B[9]为整数，B[8:0]为小数，大小为 1 左右，例如 AMC = 10’b10_0001_0000 = 1+1/32, 或

AMC = 10’b01_1110_1100 = 1-5/128

DC 位 DAC 直流偏置，为 10bit 有符号整数。

增益校正计算结果进行截断保留，最终 DAC_cali 仍为 12bit 整数。

且增益校正加直流偏置校正后的数值会进行饱和处理，最大为 0xfff，最小为 0x000。

需要注意的是，DAC 有 2 个输出档位，系统上电后，加载默认档位的 DAC 校准值，若切换到其它档位，请使用原厂提供的库函数。

6 系统时钟复位

6.1 时钟

6.1.1 时钟源

如下表所示，系统包括 4 个时钟源。其中内部低速 RC 振荡时钟 LSI/内部高速 RC 振荡时钟 HSI 不会停振。

表 6-1 系统时钟源

时钟源	频率	来源	误差	说明
LSI/LRC	32kHz	内部 RC 振荡器	全温度范围误差<50%	内部系统管理时钟，用于 WDT，复位信号的滤波和展宽，亦可作为 MCU 运行主时钟
HSI/HRC	8MHz	内部 RC 振荡器	全温度范围误差<1%	作为 PLL 源时钟
HSE	8MHz	外部晶体	与晶体有关	外部晶振
PLL	96MHz	PLL 时钟	取决于 PLL 参考时钟。使用内部 HRC 作为 PLL 参考时钟时，全温度范围误差<1%	PLL 输出时钟，以 HSI 作为参考时钟输入，倍频 12 倍后输出 PLL 时钟作为系统主时钟。
JTAG/SWD	<10MHz	调试器	-	JTAG/SWD 时钟，取决于上位机设置

SWD 时钟速率与上位机或下载器设置有关。

系统可以使用内部高速 RC 时钟 HSI 作为 PLL 的参考时钟。PLL 将 8MHz 的参考时钟 HSI 倍频 12 倍至 96MHz。

通常，PLL 经过 n/8 分频后，可以得到 96MHz×n/8 的高速时钟，作为系统主时钟 MCLK。系统主时钟 MCLK 可以通过 SYS_CLK_CFG.CLK_DIV 进行 n/8 分频，可以产生 12,24,48MHz 等频率值。

系统复位时，PLL 默认关闭，HSI 默认开启，即系统上电选择 HSI 时钟，8MHz 作为系统主时钟进行工作，从而保证系统上电之初功耗处于较低水平。

LSI 和晶振时钟 HSE 也可以作为系统主时钟，通过 SYS_CLK_CFG.CLK_SEL[1:0]进行选择。通过设置 SYS_CLK_CFG.CLK_SEL[1:0]=2 可以切换系统时钟为 LSI 时钟，此时系统工作于 32kHz 时钟，PLL/HRC/BGP 等模拟模块均可关闭以降低功耗。

SYS_CLK_CFG.CLK_DIV 作为 PLL 的分频系数。当 SYS_CLK_CFG.CLK_SEL 不为 1 时，即系统时钟 HIS/HSE 或 LSI 作为系统工作时钟，此时 SYS_CLK_CFG.CLK_DIV 不再起分频作用。

表 6-2 PLL 作为 MCLK 时钟时的分频配置

SYS_CLK_CFG	分频系数	频率/MHz	是否均匀
0x0101	1/8	12	是

0x0111	2/8	24	是
0x0155	4/8	48	是
0x01FF	8/8	96	是

系统高速时钟 MCLK 经过 SYS_CLK_FEN 寄存器控制的开关之后供给外设。I2C 时钟由 SYS_CLK_DIV0 寄存器控制可以进一步分频，UART 时钟由 SYS_CLK_DIV2 寄存器控制可以进一步分频。

PLL 输出的时钟经过 2 分频后送至 ADC（典型工作频率 48MHz），即 ACLK。

内部 32kHz RC 产生一路 LSI 时钟 LCLK，主要用于看门狗 WDT 工作时钟，以及部分系统控制，复位的滤波展宽等。当系统无过多工作任务且系统降低功耗时，可以设置 SYS_CLK_CFG.CLK_SEL=2，将系统时钟切换为 LSI。

晶振时钟可以作为系统运行时钟，也可以作为 PLL 的参考时钟源。通过设置 SYS_CLK_CFG.CLK_SEL=3，可以将系统时钟切换为晶振时钟 HSE。如果要使用晶振时钟，需要通过设置 SYS_AFE_REG5.XTALPDN=1 使能晶振起振电路，晶振起振电路默认是关闭的。如果要使用晶振时钟作为 PLL 参考时钟，则需要设置 SYS_AFE_REG6.PLLSR_SEL=1，PLL 默认参考时钟为内部 8MHz HRC 时钟。

需要注意的是，外部晶体在某些情况下可能会停振失效。芯片内部有晶振停振检测电路，当外部晶体停振时，会自动将使用晶振时钟的模块切换至内部 HRC 时钟。

通过 SYS_CLK_CFG.HSE_FAIL 和 SYS_CLK_CFG.HSE_FAILED 可以读取晶体的工作状态。

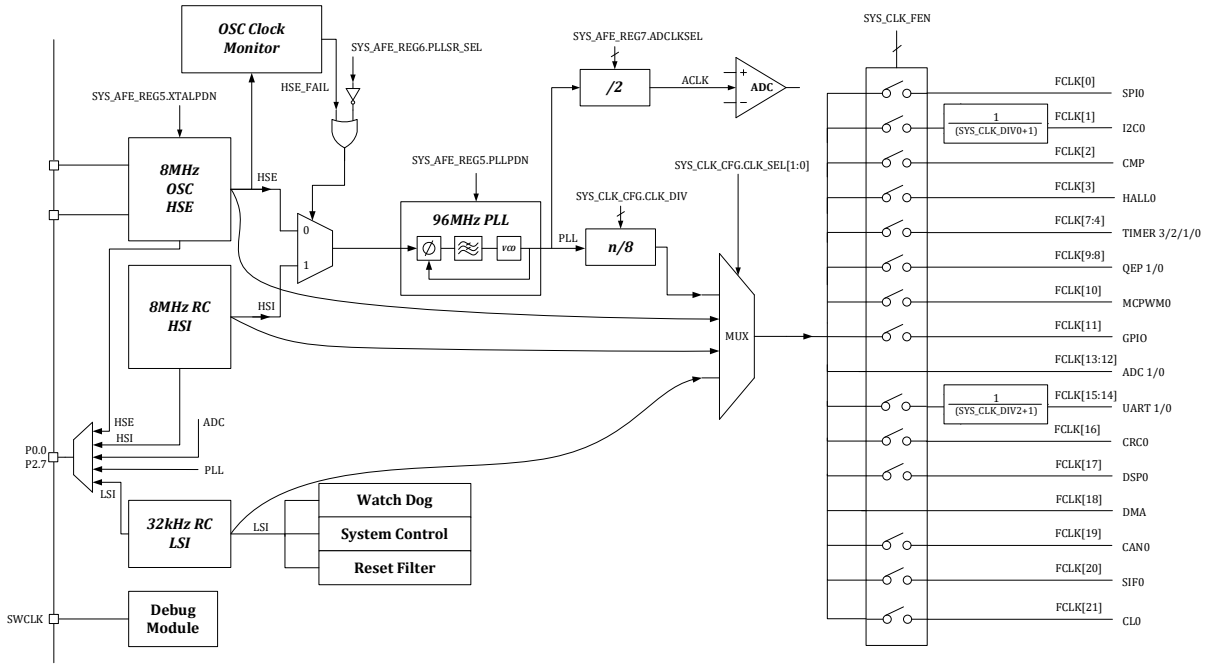


图 6-1 时钟架构

为了保证系统可靠工作，时钟系统有防止时钟被误操作关闭的机制。如当 PLL 被用作系统工作主时钟时，PLL 电路本身无法被关闭，作为 PLL 参考时钟的 HSI 无法被软件关闭；32kHz LSI 时钟上电即工作，且无法关闭。SWCLK 由调试器提供，频率可在上位机调试界面进行选择。

为便于调试和出厂校正，高速 RC 时钟 HSI 和低速时钟 LSI 可以通过配置 GPIO 的第二功能通过芯片管脚输出。

6.2 复位

6.2.1 复位源

芯片的复位来源包括硬件复位与软件复位。

6.2.1.1 硬件复位

如表 6-3 硬件复位源所示，系统包括 3 个硬件复位源，产生的复位均为芯片全局复位，复位产生后处理器程序计数器(PC)回到 0 地址，所有寄存器恢复到默认值。

表 6-3 系统复位源

名称	来源	说明
PORn	内部电源管理	同时监控 1.5V 数字电源和 5V 电源，1.5V 电源低于 1.26V 时产生复位，5V 电源低于 2.2V 时产生复位。其欠压阈值说明如表 6-4 所示。
RSTn	外部按键	外部按键复位电路，低电平有效
WDTn	硬件看门狗	如果不进行软件喂狗则定时产生复位，复位间隔可配置

表 6-4 欠压阈值说明

	Vth when Version<=3			Vth when Version>=4	
VSR_PDT / PVDSEL	0	1	VSR_PDT / PVDSEL	0	1
0	4.00	4.00*DAC_OUT/VBGP（欠压）	0	4.00	4.00*DAC_OUT/VBGP（欠压）
1	3.75	3.75*DAC_OUT/VBGP（欠压）	1	3.75	3.75*DAC_OUT/VBGP（欠压）
2	3.50	3.50*DAC_OUT/VBGP（欠压）	2	3.50	3.50*DAC_OUT/VBGP（欠压）
3	3.25	3.25*DAC_OUT/VBGP（欠压）	3	2.50	2.50*DAC_OUT/VBGP（欠压）

6.2.1.1.1 硬件复位架构

如下图所示，PORn 来自内部模拟电路， RSTn 来自外部按键。

WDTn 为 1 个 LSI 时钟周期宽度信号，是内部数字信号。WDTn 信号在 Debug 模式下可以被屏蔽，由 SYS_DBG_CFG 进行配置。

经过滤波展宽预处理的复位信号进行与运算得到一个全局复位信号。

3 个复位信号复位等级和作用域一致，均为全局复位。

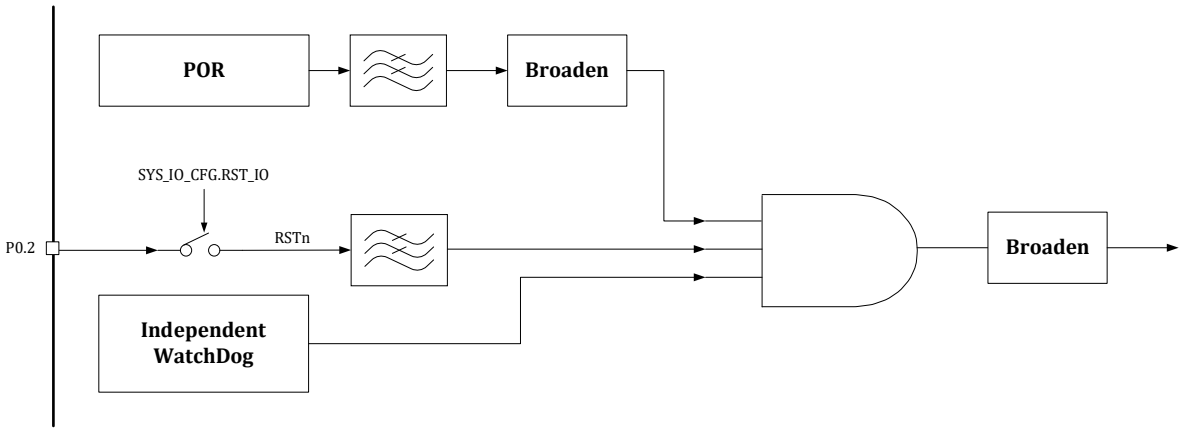


图 6-2 复位架构

6.2.1.1.2 硬件复位记录

AON_EVT_RCD 寄存器用于保存硬件复位事件，当某个硬件复位发生后，AON_EVT_RCD 对应位置记录值变为 1。AON_EVT_RCD 寄存器本身无法被复位信号复位，只能通过向 AON_EVT_RCD 寄存器写入 0xCA40 清空记录，复位记录可以方便地了解是否发生复位以及发生过哪种复位。

6.2.1.2 软件复位

CPU 的软复位操作可以使程序计数器(PC)回到 0 地址，软复位是否复位外设寄存器，取决于 SYS_DBG_CFG.SFT_RST_PERI 设置。

在集成开发环境(IDE: Integrated Development Environment)中的调试模式下，点击 Reset 与 CPU 软复位操作作用相同，仅仅使得 PC 回到 0 地址，对外设中的寄存器没有影响。但如果在 bootloader 中进行了外设模块的软复位，则会使得外设寄存器被复位为默认值。具体 bootloader 实现请咨询芯片供应商。如果设置了 SYS_DBG_CFG. SFT_RST_PERI=1，则可以在 IDE 软复位的同时复位外设寄存器。

部分外设模块有模块级软复位，可以使用 SYS_SFT_RST 寄存器进行复位，写入对应的位，可以将模块状态机恢复到初始状态，同时将模块的寄存器恢复到默认值，详见 6.3.8。

6.2.2 复位作用域

表 6-5 复位作用域

复位源	作用域
PORn	内部电源管理，全局复位
RSTn	外部按键，全局复位，除极少数寄存器
WDTn	硬件看门狗，全局复位，除极少数寄存器
SYS_SFT_RST.SPI0_ SFT_RST	SPI0
SYS_SFT_RST.I2C0_SFT_RST	I2C0
SYS_SFT_RST.CMP_SFT_RST	CMP 数字接口模块（不包括模拟比较器本身）
SYS_SFT_RST.HALL0_SFT_RST	HALL0
SYS_SFT_RST.TIMER0_ SFT_RST	TIMER0

SYS_SFT_RST.TIMER1_ SFT_RST	TIMER1
SYS_SFT_RST.TIMER2_ SFT_RST	TIMER2
SYS_SFT_RST.TIMER3_ SFT_RST	TIMER3
SYS_SFT_RST.QEP0_ SFT_RST	QEP0
SYS_SFT_RST.QEP1_ SFT_RST	QEP1
SYS_SFT_RST.MCPWM0_ SFT_RST	MCPWM
SYS_SFT_RST.GPIO_ SFT_RST	GPIO
SYS_SFT_RST.ADC0_ SFT_RST	ADC0 数字接口模块
SYS_SFT_RST.ADC1_ SFT_RST	ADC1 数字接口模块
SYS_SFT_RST.UART0_SFT_RST	UART0
SYS_SFT_RST.UART1_SFT_RST	UART1
SYS_SFT_RST.CRC0_ SFT_RST	CRC0
SYS_SFT_RST.DSP0_ SFT_RST	DSP0
SYS_SFT_RST.DMA0_ SFT_RST	DMA0
SYS_SFT_RST.CAN0_ SFT_RST	CAN0
SYS_SFT_RST.SIF0_ SFT_RST	SIF0
SYS_SFT_RST.CL0_ SFT_RST	CL0
NVIC_SystemReset();	CPU 软复位，仅复位 CPU 内核，将 PC 重置为 0，若 SYS_DBG_CFG.SFT_RST_PERI=0 所有外设寄存器值仍然维持，SYS_DBG_CFG.SFT_RST_PERI=1，所有外设寄存器同时复位。

其中用于控制 P0[2]作为 GPIO 使用还是作为外部复位脚使用的 SYS_IO_CFG.RST_IO，复位记录寄存器 AON_EVT_RCD 只受 POR 复位。

全局复位会复位全芯片寄存器，包括 CPU 内核寄存器以及所有外设寄存器，除上述极少数寄存器。

若 CPU 软复位设置为仅仅复位 CPU 内核，而不复位外设寄存器。建议重新烧录程序后使用掉电重新上电或外部复位的方式重置外设寄存器。

Flash 存储内容，SRAM 存储内容不受复位影响。

6.3 寄存器

6.3.1 地址分配

系统模块寄存器基地址为 0x4000_0000。

表 6-6 系统控制寄存器

名称	偏移	说明
SYS_CLK_CFG	0x80	时钟控制寄存器
SYS_IO_CFG	0x84	IO 控制寄存器
SYS_DBG_CFG	0x88	Debug 控制寄存器
SYS_CLK_DIV0	0x90	外设时钟分频寄存器 0
SYS_CLK_DIV2	0x98	外设时钟分频寄存器 2
SYS_CLK_FEN	0x9C	外设时钟门控寄存器
SYS_SFT_RST	0xA4	软复位寄存器

SYS_PROTECT	0xA8	写保护寄存器
SYS_FLSE	0xD0	Flash 擦除保护寄存器
SYS_FLSP	0xD4	Flash 编程保护寄存器

6.3.2 SYS_CLK_CFG 时钟控制寄存器

地址:0x4000_0080

复位值:0x0

表 6-7 SYS_CLK_CFG 时钟控制寄存器

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

位置	位名称	说明
[31:14]		未使用
[13]	HSE_FAILED	晶振时钟在被开启后是否停振过 如果晶振有停振情况，则该位置位，即使晶体后续恢复振荡，仍保持记录为 1。写 1 清零
[12]	HSE_FAIL	晶振时钟当前是否已经停振 0：外部晶体当前正常工作 1：外部晶体当前已经停振
[11:10]		未使用
[9:8]	CLK_SEL	系统时钟 MCLK 的来源选择信号。默认选择 HRC 0: HRC 1: PLL 2: LRC 3: XTAL 注意，PLL/XTAL 在上电后默认关闭，需要软件来开启。
[7:0]	CLK_DIV	PLL 输出分频控制，默认为 0x00: 1/8 分频 0x01: 1/8 分频 0x11: 1/4 分频 0x55: 1/2 分频 0xFF: 1/1 分频 不推荐其它配置值。

当 CLK_SEL = 0 时，MCLK 选择 HSI 时钟（8MHz），此时 SYS_CLK_CFG[7:0]的分频系数无效。最终输出的系统时钟频率即为 8MHz。

当 CLK_SEL = 2 时，MCLK 选择 LRC 时钟（32kHz），此时 SYS_CLK_CFG[7:0]的分频系数无效。

最终输出的系统时钟频率即为 32kHz。

当 CLK_SEL = 3 时，MCLK 选择 HSE 时钟（8MHz），此时 SYS_CLK_CFG[7:0]的分频系数无效。最终输出的系统时钟频率即为 8MHz。

6.3.3 SYS_IO_CFG IO 控制寄存器

地址:0x4000_0084

复位值:0x40

表 6-8 SYS_IO_CFG IO 控制寄存器

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

位置	位名称	说明
[31:7]		未使用
[6]	SWDMUX	SWD 复用控制信号，默认配置为 SWD 0:P2.14 复用为 SWCLK，P2.15 复用为 SWDIO 1:P2.14, P2.15 作为正常 GPIO 使用
[5]	RST_IO	RSTn/P0.2 复用控制信号，默认配置为 RSTn 0:RSTn 1:P0.2 注意，上电后默认是 RSTn，后续软件可使能此位，RSTn 功能失效
[4:0]		未使用

为了安全起见，上电后 16ms 内，SWD 的 IO 不能切换为 GPIO 功能，只能作为 SWD 使用。即使软件改写了 SYS_IO_CFG [6]将不起作用，需要在 16ms 之后再次对 SYS_IO_CFG [6]改写。现象为:在 16ms 内，软件可以向 SYS_IO_CFG[6] 1'b1，但读回该 bit 仍为 1'b 0，如果经过 16ms 后再次写入 1'b 1，则可读回 1'b 1。即在上电后的 16ms 内，软件 SYS_IO_CFG[6]写入 1'b 1 无效，上电 16ms 后，软件对 SYS_IO_CFG[6]写入 1'b 1 才有效。这是为了防止应用上上电即切换 SWD 功能为 GPIO，导致芯片后续无法通过 SWD 进行擦写烧录 flash。

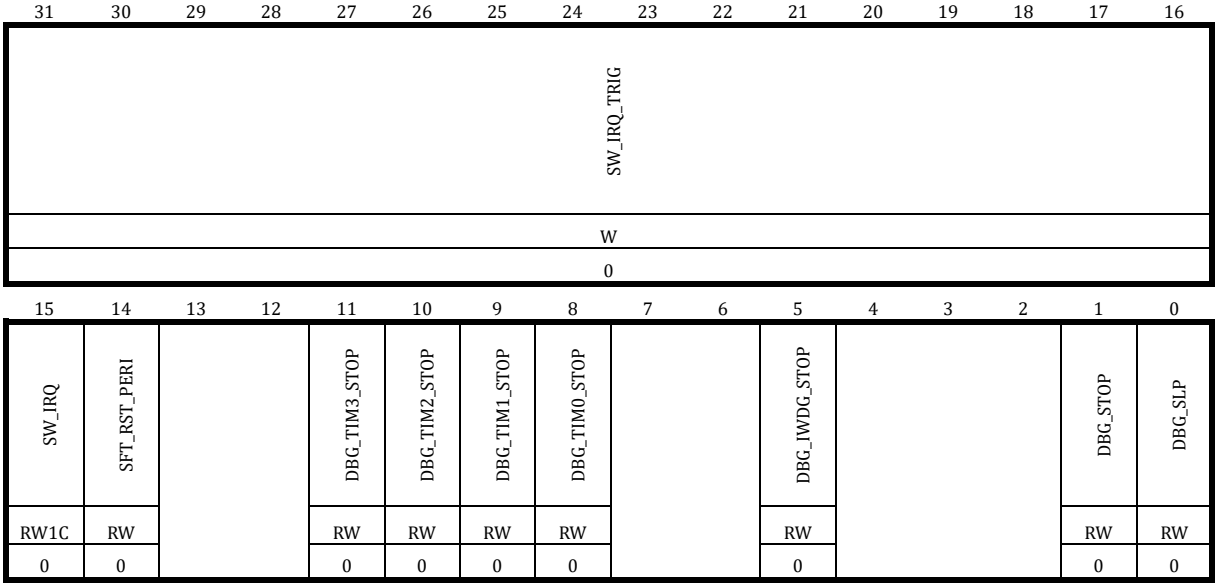
注意，SWD IO 切换为 GPIO 功能后，用户无法再通过 SWD 对芯片进行调试，因此建议上电后留有一定的时间窗口，经过一定延时后再将 SWD 切换为 GPIO 功能。而且软件上建议留有其他设计考虑，使得 SWD IO 可以切换回 SWD 功能。

6.3.4 SYS_DBG_CFG Debug 控制寄存器

地址:0x4000_0088

复位值:0x40

表 6-9 SYS_DBG_CFG Debug 控制寄存器



位置	位名称	说明
[31:16]	SW_IRQ_TRIG	向此位段写入 0x5AA5 触发软件中断，SW_IRQ 置 1
[15]	SW_IRQ	软件中断标志，对应中断号 30，写 1 清零。
[14]	SFT_RST_PERI	Debug 软复位是否复位除 Flash/SYS_AFE 以外的外设寄存器
[13:12]		未使用
[11]	DBG_TIM3_STOP	调试模式下 CPU halt 状态 Timer3 停止 1:Timer3 在 CPU halt 状态时停止计数 0:Timer3 在 CPU halt 状态时继续计数
[10]	DBG_TIM2_STOP	调试模式下 CPU halt 状态 Timer2 停止 1:Timer2 在 CPU halt 状态时停止计数 0:Timer2 在 CPU halt 状态时继续计数
[9]	DBG_TIM1_STOP	调试模式下 CPU halt 状态 Timer1 停止 1:Timer1 在 CPU halt 状态时停止计数 0:Timer1 在 CPU halt 状态时继续计数
[8]	DBG_TIM0_STOP	调试模式下 CPU halt 状态 Timer0 停止 1:Timer0 在 CPU halt 状态时停止计数 0:Timer0 在 CPU halt 状态时继续计数
[7:6]		未使用
[5]	DBG_IWDG_STOP	调试模式下 CPU halt 状态独立看门狗停止 1:独立看门狗在 CPU halt 状态时停止计数 0:独立看门狗在 CPU halt 状态时继续计数
[4:2]		未使用
[1]	DBG_STOP	调试 STOP 模式

		0: (FCLK=Off, HCLK=Off) 在 STOP 模式下，时钟管理模块关闭 HCLK 和 FCLK，即处理器时钟和所有外设时钟。 当退出 STOP 模式时，时钟管理模块不经历复位，并保持原有配置，可以恢复到 STOP 模式前的时钟设置。在 STOP 模式中，所有外设寄存器均保持，因此退出后无需重新配置。 1: (FCLK=On, HCLK=On) 如果设置 DBG_STOP 为 1，进入 STOP 模式后 HCLK 和 FCLK 均不被关闭。 通过设置 SCB->SCR = (1UL<<2)，然后使用__WFI()/__WFE()指令进入 STOP 模式。
[0]	DBG_SLP	调试睡眠(SLEEP)模式 0: (FCLK=On, HCLK=Off) 在睡眠模式中，FCLK 作为所有外设时钟，不关闭； HCLK 作为 CPU 时钟，被关闭。 在睡眠模式中，只有 CPU 时钟被暂时挂起，而所有外设包括系统时钟管理模块均保持原有配置状态。因此退出睡眠模式时，软件无需重新配置外设寄存器。 1: (FCLK=On, HCLK=On) 如果配置 DBG_SLP 为 1，则当进入睡眠模式时，HCLK 不会关闭。 通过__WFI()/__WFE()指令可以令处理器进入睡眠模式

6.3.5 SYS_CLK_DIV0 外设时钟分频寄存器 0

地址:0x4000_0090

复位值:0x0

表 6-10 SYS_CLK_DIV0 外设时钟分频寄存器 0

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

位置	位名称	说明
[31:16]		未使用
[15:0]	DIV0	I2C 工作时钟=MCLK/(CLK_DIV0+1)。其中 MCLK 由 SYS_CLK_CFG 分频系数决定

6.3.6 SYS_CLK_DIV2 外设时钟分频寄存器 2

地址:0x4000_0098

复位值:0x0

表 6-4 SYS_CLK_DIV2 外设时钟分频寄存器 2

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

位置	位名称	说明
[31:16]		未使用
[15:0]	DIV2	UART 工作时钟=MCLK/(CLK_DIV2+1)。UART0/UART1 共享此分频配置，波特率根据 UART 波特率寄存器进一步分频，其中 MCLK 由 SYS_CLK_CFG 分频系数决定。

6.3.7 SYS_CLK_FEN 外设时钟门控寄存器

地址:0x4000_009C

复位值:0x0

表 6-5 SYS_CLK_FEN 外设时钟门控寄存器

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
										CL0_CLK_EN	SIF0_CLK_EN	CAN0_CLK_EN	Res.	DSP0_CLK_EN	CRC0_CLK_EN
										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
UART1_CLK_EN	UART0_CLK_EN	Res.	Res.	GPIO_CLK_EN	MCPWM0_CLK_EN	QEP1_CLK_EN	QEP0_CLK_EN	TIMER3_CLK_EN	TIMER2_CLK_EN	TIMER1_CLK_EN	TIMER0_CLK_EN	HALL0_CLK_EN	CMP_CLK_EN	I2C0_CLK_EN	SPI0_CLK_EN
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:21]		未使用
[21]	CL0_CLK_EN	CL0 模块时钟控制信号，默认关闭 CL0 模块时钟 1:使能 CL0 模块时钟 0:关闭 CL0 模块时钟
[20]	SIF0_CLK_EN	SIF0 模块时钟控制信号，默认关闭 SIF0 模块时钟 1:使能 SIF0 模块时钟 0:关闭 SIF0 模块时钟
[19]	CAN0_CLK_EN	CAN0 模块时钟控制信号，默认关闭 CAN0 模块时钟 1:使能 CAN0 模块时钟 0:关闭 CAN0 模块时钟
[18]		保留
[17]	DSP0_CLK_EN	DSP0 模块时钟控制信号，默认关闭 DSP0 模块时钟 1:使能 DSP0 模块时钟

		0:关闭 DSP0 模块时钟
[16]	CRC0_CLK_EN	CRC0 模块时钟控制信号，默认关闭 CRC0 模块时钟 1:使能 CRC0 模块时钟 0:关闭 CRC0 模块时钟
[15]	UART1_CLK_EN	UART1 模块时钟控制信号，默认关闭 UART1 模块时钟 1:使能 UART1 模块时钟 0:关闭 UART1 模块时钟
[14]	UART0_CLK_EN	UART0 模块时钟控制信号，默认关闭 UART0 模块时钟 1:使能 UART0 模块时钟 0:关闭 UART0 模块时钟
[13]		保留
[12]		保留
[11]	GPIO_CLK_EN	GPIO 模块时钟控制信号，默认关闭 GPIO 模块时钟 1:使能 GPIO 模块时钟 0:关闭 GPIO 模块时钟
[10]	MCPWM0_CLK_EN	MCPWM0 模块时钟控制信号，默认关闭 MCPWM0 模块时钟 1:使能 MCPWM0 模块时钟 0:关闭 MCPWM0 模块时钟
[9]	QEP1_CLK_EN	QEP1 模块时钟控制信号，默认关闭 QEP1 模块时钟 1:使能 QEP1 模块时钟 0:关闭 QEP1 模块时钟
[8]	QEP0_CLK_EN	QEP0 模块时钟控制信号，默认关闭 QEP0 模块时钟 1:使能 QEP0 模块时钟 0:关闭 QEP0 模块时钟
[7]	TIMER3_CLK_EN	TIMER3 模块时钟控制信号，默认关闭 TIMER3 模块时钟 1:使能 TIMER3 模块时钟 0:关闭 TIMER3 模块时钟
[6]	TIMER2_CLK_EN	TIMER2 模块时钟控制信号，默认关闭 TIMER2 模块时钟 1:使能 TIMER2 模块时钟 0:关闭 TIMER2 模块时钟
[5]	TIMER1_CLK_EN	TIMER1 模块时钟控制信号，默认关闭 TIMER1 模块时钟 1:使能 TIMER1 模块时钟 0:关闭 TIMER1 模块时钟
[4]	TIMER0_CLK_EN	TIMER0 模块时钟控制信号，默认关闭 TIMER0 模块时钟 1:使能 TIMER0 模块时钟 0:关闭 TIMER0 模块时钟
[3]	HALL0_CLK_EN	HALL0 模块时钟控制信号，默认关闭 HALL0 模块时钟 1:使能 HALL0 模块时钟 0:关闭 HALL0 模块时钟
[2]	CMP_CLK_EN	CMP 模块时钟控制信号，默认关闭 CMP 模块时钟 1:使能 CMP 模块时钟 0:关闭 CMP 模块时钟
[1]	I2C0_CLK_EN	I2C0 模块时钟控制信号，默认关闭 I2C0 模块时钟 1:使能 I2C0 模块时钟 0:关闭 I2C0 模块时钟

[0]	SPI0_CLK_EN	SPI0 模块时钟控制信号，默认关闭 SPI0 模块时钟 1:使能 SPI0 模块时钟 0:关闭 SPI0 模块时钟

注意，上述每个模块的时钟为各自模块内部电路的工作时钟，即使不开启各自模块的时钟，也不影响软件访问各自模块的寄存器。

6.3.8 SYS_SFT_RST 软复位寄存器

地址:0x4000_00A4

复位值:0x0

表 6-6 SYS_SFT_RST 软复位寄存器

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
										CL0_SFT_RST	SIF0_SFT_RST	CAN0_SFT_RST	DMA0_SFT_RST	DSP0_SFT_RST	CRC0_SFT_RST
										RW	RW	RW	RW	RW	RW
										0	0	0	0	0	0

15	4	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UART1_SFT_RST	UART0_SFT_RST	ADC1_SFT_RST	ADC0_SFT_RST	GPIO_SFT_RST	MCPWM0_SFT_RST	QEP1_SFT_RST	QEP0_SFT_RST	TIMER3_SFT_RST	TIMER2_SFT_RST	TIMER1_SFT_RST	TIMER0_SFT_RST	HALL0_SFT_RST	CMP0_SFT_RST	I2C0_SFT_RST	SPI0_SFT_RST
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

位置	位名称	说明
[31:21]		未使用
[21]	CL0_SFT_RST	CL0 模块软复位信号，默认不复位 CL0 模块 1:复位 CL0 模块 0:释放 CL0 模块
[20]	SIF0_SFT_RST	SIF0 模块软复位信号，默认不复位 SIF0 模块 1:复位 SIF0 模块 0:释放 SIF0 模块
[19]	CAN0_SFT_RST	CAN0 模块软复位信号，默认不复位 CAN0 模块 1:复位 CAN0 模块 0:释放 CAN0 模块
[18]	DMA0_SFT_RST	DMA0 模块软复位信号，默认不复位 DMA0 模块 1:复位 DMA0 模块 0:释放 DMA0 模块
[17]	DSP0_SFT_RST	DSP0 模块软复位信号，默认不复位 DSP0 模块 1:复位 DSP0 模块 0:释放 DSP0 模块

[16]	CRC0_ SFT_RST	CRC0 数字接口模块软复位信号，默认不复位 CRC0 模块 1:复位 CRC0 数字接口模块 0:释放 CRC0 数字接口模块
[15]	UART1_SFT_RST	UART1 模块软复位信号，默认不复位 UART1 模块 1:复位 UART1 模块 0:释放 UART1 模块
[14]	UART0_SFT_RST	UART0 模块软复位信号，默认不复位 UART0 模块 1:复位 UART0 模块 0:释放 UART0 模块
[13]	ADC1_ SFT_RST	ADC1 数字接口模块软复位信号，默认不复位 ADC1 模块 1:复位 ADC1 数字接口模块 0:释放 ADC1 数字接口模块
[12]	ADC0_ SFT_RST	ADC0 数字接口模块软复位信号，默认不复位 ADC0 模块 1:复位 ADC0 数字接口模块 0:释放 ADC0 数字接口模块
[11]	GPIO_ SFT_RST	GPIO 模块软复位信号，默认不复位 GPIO 模块 1:复位 GPIO 模块 0:释放 GPIO 模块
[10]	MCPWM0_ SFT_RST	MCPWM0 模块软复位信号，默认不复位 MCPWM0 模块 1:复位 MCPWM0 模块 0:释放 MCPWM0 模块
[9]	QEP1_ SFT_RST	QEP1 模块软复位信号，默认不复位 QEP1 模块 1:复位 QEP1 模块 0:释放 QEP1 模块
[8]	QEP0_ SFT_RST	QEP0 模块软复位信号，默认不复位 QEP0 模块 1:复位 QEP0 模块 0:释放 QEP0 模块
[7]	TIMER3_ SFT_RST	TIMER3 模块软复位信号，默认不复位 TIMER3 模块 1:复位 TIMER3 模块 0:释放 TIMER3 模块
[6]	TIMER2_ SFT_RST	TIMER2 模块软复位信号，默认不复位 TIMER2 模块 1:复位 TIMER2 模块 0:释放 TIMER2 模块
[5]	TIMER1_ SFT_RST	TIMER1 模块软复位信号，默认不复位 TIMER1 模块 1:复位 TIMER1 模块 0:释放 TIMER1 模块
[4]	TIMER0_ SFT_RST	TIMER0 模块软复位信号，默认不复位 TIMER0 模块 1:复位 TIMER0 模块 0:释放 TIMER0 模块
[3]	HALL0_SFT_RST	HALL0 模块软复位信号，默认不复位 HALL0 模块 1:复位 HALL0 模块 0:释放 HALL0 模块
[2]	CMP_ SFT_RST	CMP 模块软复位信号，默认不复位 CMP 模块 1:复位 CMP 模块 0:释放 CMP 模块

[1]	I2C_0SFT_RST	I2C0 模块软复位信号，默认不复位 I2C0 模块 1:复位 I2C0 模块 0:释放 I2C0 模块
[0]	SPI0_ SFT_RST	SPI0 模块软复位信号，默认不复位 SPI0 模块 1:复位 SPI0 模块 0:释放 SPI0 模块

注意，模块软复位在 SYS_SFT_RST 对应位写入 1 后会保持在复位状态，需要再次写入 0 才能解除复位状态。

6.3.9 SYS_PROTECT 写保护寄存器

地址:0x4000_00A8

复位值:0x0

表 6-7 SYS_PROTECT 写保护寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	PSW	除 SYS_AFE_DAC0、SYS_AFE_DAC0_AMC、SYS_AFE_DAC0_DC、SYS_AFE_DAC1、SYS_AFE_DAC1_AMC、SYS_AFE_DAC1_DC 外，其他系统寄存器(SYS_开头的寄存器，包括时钟复位管理以及模拟寄存器)受写保护，写入前需先写入密码解除写保护 写入 0x7A83，解除写保护，使能寄存器写操作 写入其它值，开启写保护，禁止寄存器写操作 读回恒为 0

6.3.10 SYS_FLSE 擦除保护寄存器

地址:0x4000_00D0

复位值:0x0

表 6-8 擦除保护寄存器 SYS_FLSE

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

位置	位名称	说明
[31:16]		未使用
[15:0]	FLSE	**FLASH 擦除保护寄存器。本寄存器写入 0x8FCA，FLASH 的擦除功能才能最**

		终生效。写入其他值，FLASH 的擦除功能均无法生效。为防止误擦除，建议不使用擦除功能时，不生效此寄存器。

6.3.11 SYS_FLSP 编程保护寄存器

地址:0x4000_00D4

复位值:0x0

表 6-9 编程保护寄存器 SYS_FLSE

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

位置	位名称	说明
[31:16]		未使用
[15:0]	FLSP	**FLASH 编程保护寄存器。本寄存器写入 0x8F35，FLASH 的编程功能才能最终生效。写入其他值，FLASH 的编程功能均无法生效。为防止误编程，建议不使用编程功能时，不生效此寄存器。**

7 非易失性存储体

7.1 概述

非易失性存储体包含两个部分:FLASH 和 ROM。

FLASH 存储体包含两个部分：NVR 和 MAIN。NVR 大小为 1.5KB；MAIN 有三个大小尺寸：32KB、64KB 和 128KB。

- FLASH 存储体主闪存存储区（MAIN），包括应用程序和用户数据区
- FLASH 信息存储区（NVR），预留给用户使用数据区

ROM 存储体，出厂前固化特定程序，有两个大小尺寸 32KB 和 64KB。

LKS32MC07x 系列芯片，非易失性存储体包含五个型号：

- 型号 1：1.5KB NVR，32KB FLASH
- 型号 2：1.5KB NVR，32KB FLASH，32KB ROM
- 型号 3：1.5KB NVR，64KB FLASH
- 型号 4：1.5KB NVR，64KB FLASH，64KB ROM
- 型号 5：1.5KB NVR，128KB FLASH

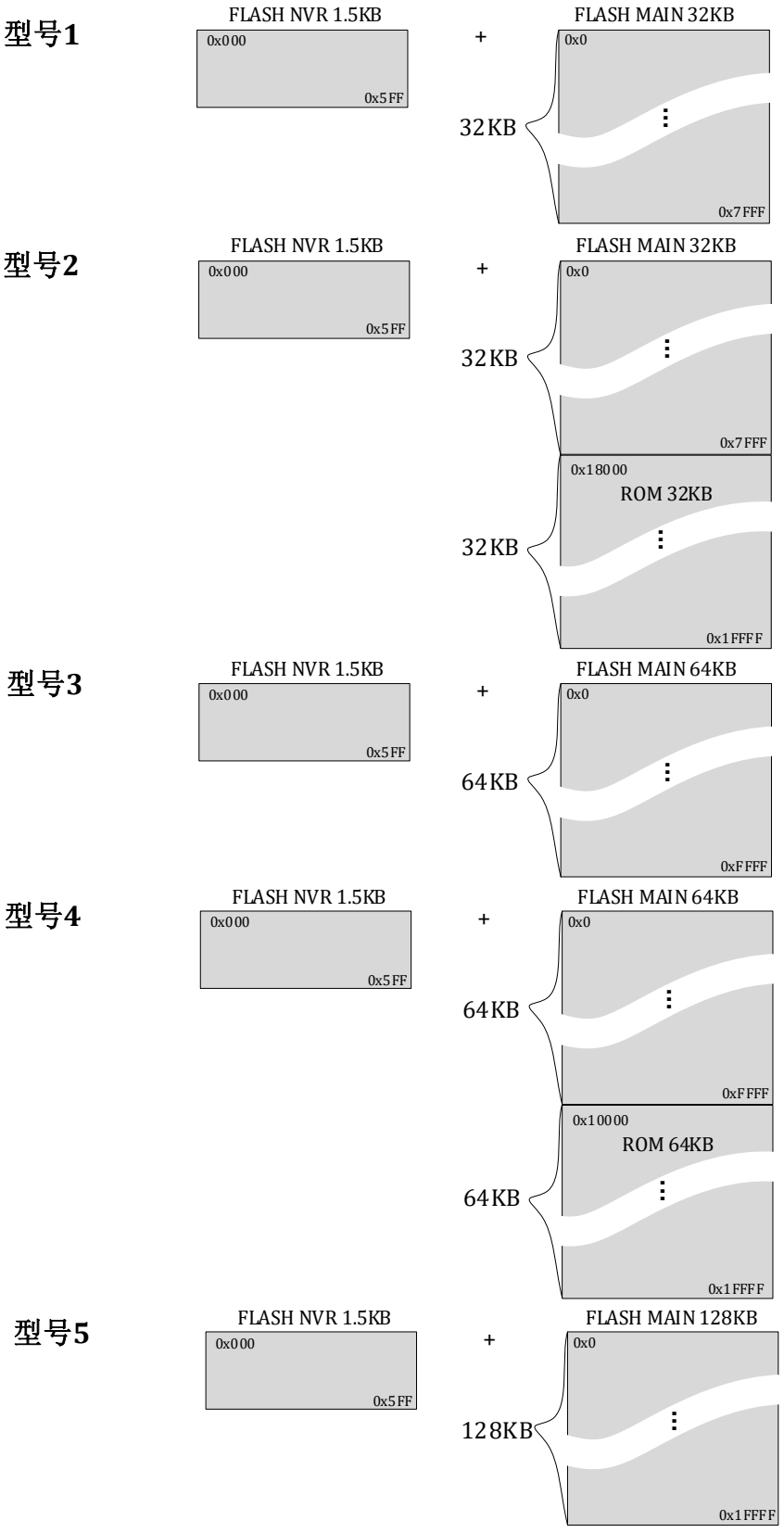


图 7-1 非易失性存储体空间划分框图及型号

7.2 功能特点

非易失性存储体控制器，主要实现对 FLASH 存储体的相关操作；ROM 存储体仅包含执行操作。具体包括：

- FLASH 读取数据的操作，包括对 NVR 部分的读取和对 MAIN 部分的读取。
- FLASH 写入数据的操作，包括对 NVR 部分的写入和对 MAIN 部分的写入。
- FLASH 擦除操作，包括 Full 擦除和 Sector 擦除。NVR 部分仅支持 Sector 擦除，MAIN 部分支持 Full 擦除和 Sector 擦除。
- FLASH 深度休眠的操作，以降低芯片的休眠功耗。
- FLASH 存储体内容的加密操作。
- FLASH 的读取加速操作，以提升芯片整体运行效率。
- FLASH 控制寄存器的访问。
- ROM 区域，仅可执行，不可拷贝。
- FLASH 保护状态下，NVR 的 Sector 2（从 0 开始计算）支持读取，编程和擦除。

7.2.1 功能描述

非易失性存储体控制器，实现了对 FLASH 存储体的复位/读出/写入/擦除/休眠等操作。如下为控制状态转换图:

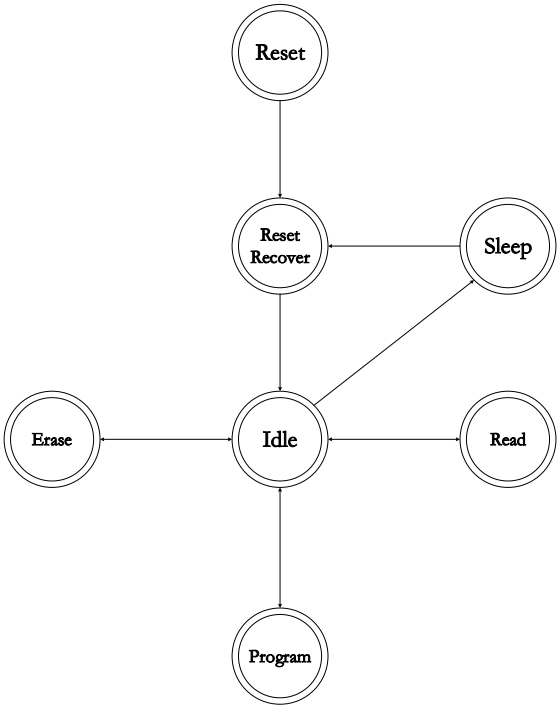


图 7-2 FLASH 控制状态转换图

7.2.1.1 复位操作

系统完成复位后，FLASH 需要一段时间恢复。其目的是保证 FLASH 存储体内部的电路稳定。待稳定后，MCU 才可对 FLASH 执行对应操作。此操作由硬件自动实现，无需软件干预。

7.2.1.2 休眠操作

FLASH 的休眠操作分成两个部分：Standby 和 Deep Sleep。当系统不执行对 FLASH 的操作时，FLASH 可自动进入 StandBy 状态（若开启预取，此功能失效）。当系统执行 Deep Sleep 操作时，将触发 FLASH 也进入 Deep Sleep，实现降低功耗的目的。FLASH 进入 Deep Sleep 的操作，硬件自动完成，无需软件介入。

当外界唤醒系统，同时将唤醒 FLASH。经过一段时间恢复后，FLASH 可执行正常操作。此唤醒恢复操作，硬件自动完成，无需软件介入。

7.2.1.3 读取操作

读操作为 FLASH 的基本操作。系统可通过两条路径访问 FLASH 内部的数据。

- MCU 通过 AHB 总线，直接对 FLASH 执行取指、取数操作，宽度为 32bit，且只能访问 MAIN 空间的数据。为了加快 MCU 的取指、取数据的速度，硬件提供了加速的功能。
- MCU 通过 AHB 总线，访问控制器的寄存器，间接实现读取 FLASH 内部数据的操作。可以访问 MAIN 和 NVR 空间的数据；若执行连续读取操作，硬件可自动完成地址累加，无需每次都更新地址寄存器的值。

ROM 不可被读取，仅可执行基本操作。系统只有一条路径访问 ROM 内部的数据

- MCU 通过 AHB 总线，直接对 ROM 执行取指、取数操作，宽度为 32bit。

FLASH_CFG.REGION 位指示当前访问的是哪个空间。具体表格如下:

表 7-1 FLASH 访问空间分配表

FLASH_CFG.REGION	访问区域
0	MAIN 区域
1	NVR 区域

访问本控制器的寄存器，实现间接读取 FLASH 内部数据的操作的执行流程如下:

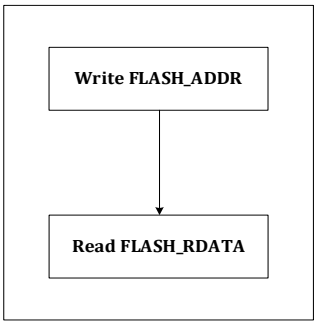


图 7-3 FLASH 间接读取操作流程图

7.2.1.4 FLASH 编程操作

执行对 FLASH 存储体的编程操作。一般而言，我们先执行擦除操作，然后执行数据编程操作。同时，只能通过访问 FLASH 控制器的寄存器，实现编程操作。具体流程为:

- SYS_FLSP 寄存器，写入 0x8F35，开启编程使能开关 1
- FLASH_CFG.PRG_EN 写 1，开启编程使能开关 2
- FLASH_ADDR，写入编程地址
- FLASH_WDATA，写入编程数据

访问本控制器的寄存器，实现 FLASH 编程操作的执行流程如下:

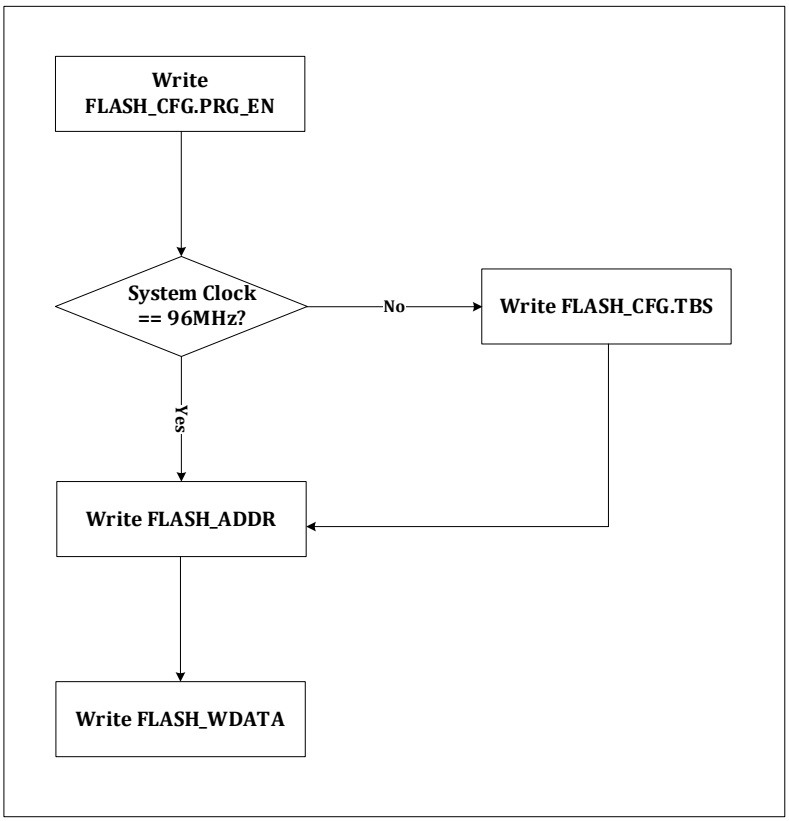


图 7-4 FLASH 模块编程操作流程图

为了防止 FLASH 区域被误编程，额外增加一组编程的使能开关--SYS_FLSP。SYS_FLSP 保护 FLASH，防止误编程。写入 0x8F35，开启编程使能，再配置 FLASH_CFG.PRG_EN，最终开启 FLASH 的编程功能。

系统工作频率的判断，需要参考 SYS_CLK_CFG 的配置。FLASH 编程/擦除操作的绝对时间是固定的，FLASH 控制器需要保存这些绝对时间对应的计数值。FLASH_CFG.TBS 默认值是 96MHz 时钟频率下的计数值；当芯片工作在其他频率下时，需要配置 FLASH_CFG.TBS 的值，以实现 48MHz 和 24MHz 的计数值（其它频率暂不支持）。最终保证计数值的值×时钟频率等于恒定的时间。不同频率下对应的 FLASH_CFG.TBS 值，已经在 FLASH_CFG 寄存器列出。切记，FLASH_CFG.TBS 仅能配置寄存器说明中提供的几组值，不能被写入其它值，否则可能导致 FLASH 编程/擦除失败。建议对 FLASH_CFG 的操作，执行先读回，然后按照或/与的方法操作。另外，在执行 FLASH 的编程/擦除

操作时，CPU 将暂停工作直至 FLASH 的编程/擦除操作完毕。

图 7-4 仅展示了一次编程的流程。若执行连续编程时，可以在写入 FLASH_ADDR 寄存器前，配置 FLASH_CFG.ADR_INC，开启地址自动递增模式，后续只需要反复写 FLASH_WDATA 寄存器即可，FLASH_ADDR 每次写入一次数据会自动增加 0x4。连续读操作类似。连续编程的流程如下:

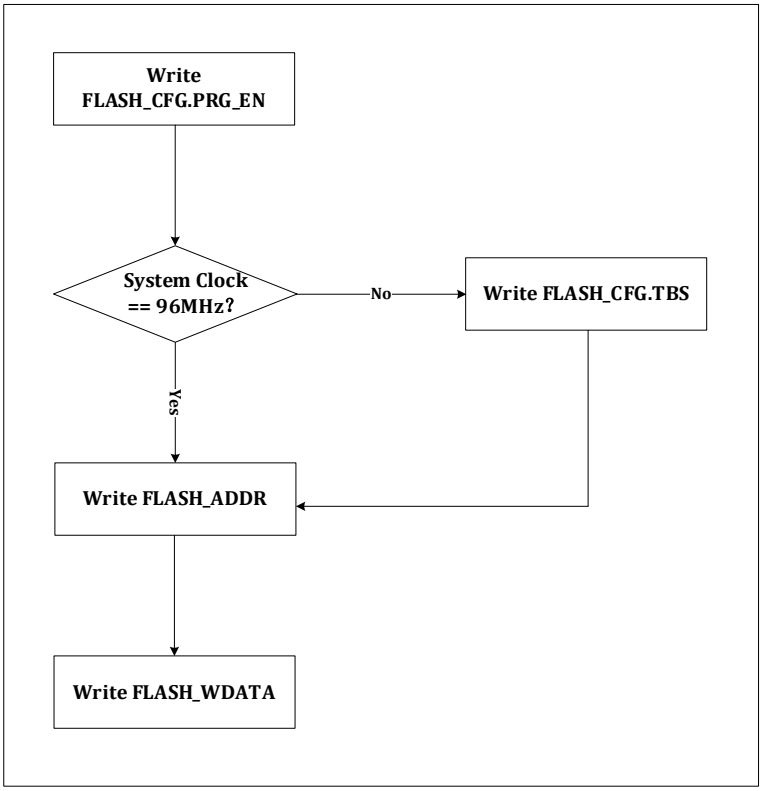


图 7-5 FLASH 模块编程操作流程图

7.2.1.5 FLASH 擦除操作

擦除操作为 FLASH 的基本操作。系统只能通过访问 FLASH 控制器的寄存器实现。具体流程为:

- SYS_FLSE 寄存器，写入 0x8FCA，擦除操作使能开关 1
- FLASH_CFG.ERS_EN 写 1，擦除操作使能开关 2
- FLASH_ADDR，写入擦除地址
- FLASH_ERASE，触发擦除操作

为了防止 FLASH 区域被误擦，额外增加一组擦除的使能开关--SYS_FLSE。SYS_FLSE 保护 FLASH，防止误擦，写入 0x8FCA，开启擦除使能，再配置 FLASH_CFG.ERS_EN，最终开启 FLASH 的擦除功能。

执行对 FLASH 存储体的擦除操作。擦除分成 Sector 和 Full。分别对应，512Byte 的擦除和 32KB/64KB/128KB 的擦除。通过配置 FLASH 控制寄存器决定执行哪一种类型的擦除操作。

下表为 Secotor 地址分配空间。

表 7-2 FLASH Sector 地址分配表

Name	Addresses	Size(Bytes)
Sector 0	0x0000 0000 - 0x0000 01FF	512
Sector1	0x0000 0200 - 0x0000 03FF	512
Sector2	0x0000 0400 - 0x0000 05FF	512
…	…	…
Sector255	0x0001 FE00 - 0x0001 FFFF	512

NVR 区域只能实现 Sector 擦除，NVR 区域只有三个 Sector(0/1/2)；MAIN 区域可以实现 Sector 擦除和 Full 擦除。具体表格如下:

NVR（FLASH_CFG .REGION）	Sector Erase	Full Erase
0	Main 区域	Main 区域
1	NVR 区域	Main 区域

FLASH 擦除操作流程如下所示。

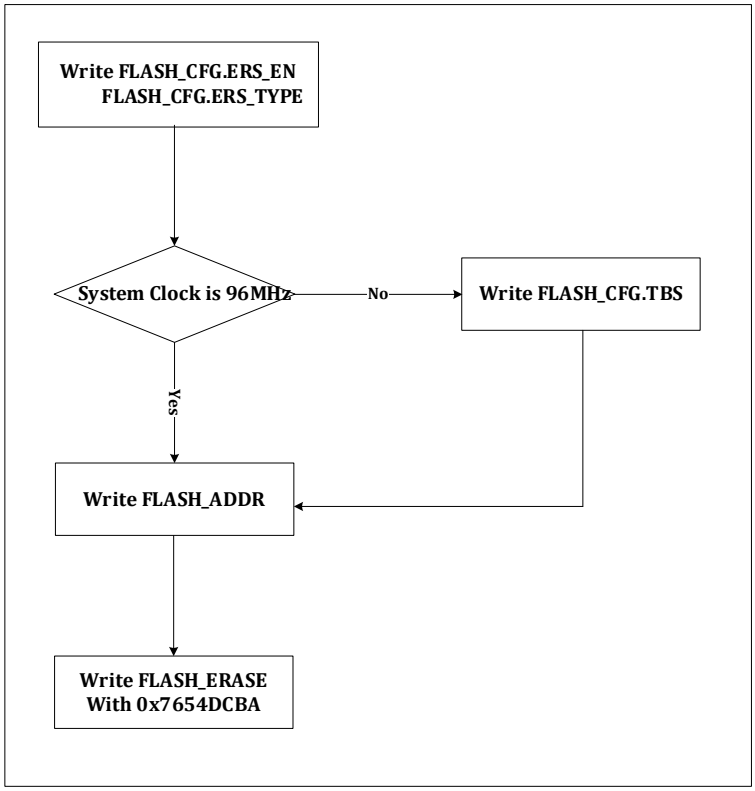


图 7-6 FLASH 模块擦除操作流程图

若选择 Secotor 擦除，需要通过 FLASH_ADDR 确定哪个 Secotor 被擦除，若是 Full 模式的话，FLASH_ADDR 的值将失效。FLASH_ERASE 写入 0x7654DCBA 触发擦除操作。

7.2.1.6 FLASH 预取操作

因 FLASH 存储体的速度限制，无法达到 96MHz 的速度。对 FLASH 进行读取操作时，需要大于一个 96MHz 的时钟周期，才能完成数据的读出。为了加快数据的读出，FLASH 控制器增加了预

取功能。当 FLASH 控制器完成当前的读取操作后，在不影响正常程序执行的前提下，顺序预取下一个 WORD 的数据。预取操作的开启和关闭，只需要设置 FLASH_CFG.PREF 即可。

当系统频率降低到 48MHz 及以下，对 FLASH 进行读取操作，无需两个时钟周期的等待。此时，可以配置 FLASH_CFG.WAIT 为 0，去掉一个额外的等待周期。切记，从高频调整到低频的时候，先调慢时钟，然后将 FLASH_CFG.WAIT 改成 0；从低频调整到高频的时候，先将 FLASH_CFG.WAIT 改成 1，然后再调快时钟。

7.2.1.7 非易失性存储体保护

非易失性存储体加密保护，保护 FLASH 和 ROM。因 ROM 只能执行，无法拷贝，已经处于保护状态且无法更改；本小节主要解释 FLASH 部分的保护。

FLASH 保护的目的：防止外界非法获取 FLASH 的内容。

若 FLASH 存储体内的数据处于保护状态，用户可执行去保护操作；相反，若 FLASH 存储体内的数据处于去保护状态，用户可执行保护操作。默认情况下，FLASH 存储体内的数据处于保护的状态。芯片上电复位完成后，硬件自动执行一次状态更新操作，若是保护状态则保持保护的状态，否则，变成去保护状态。

FLASH 存储体有 32KB、64KB 和 128KB，各自的最后一个 WORD 设计为保护字（32KB 对应 0x7FFC，64KB 对应 0xFFFC，128KB 对应 0x1FFFC）。当这个 WORD 内容为全 1 时，说明，FLASH 的内容无需保护，读取 FLASH_PROTECT 寄存器，完成状态更新即可；当这个 WORD 的内容被写为非全 1 时，说明，FLASH 的内容需要保护。若需要保护，仅需要将最后一个 WORD 的内容，写入非全 1 的值，读取 FLASH_PROTECT 寄存器，完成状态的更新（读取 FLASH_PROTECT 返回值无意义），启动保护了。

去保护分成两种情况。若最后一个 WORD 没有执行过写入非全 1 的操作，读取 FLASH_PROTECT 寄存器，即完成状态更新（读取 FLASH_PROTECT 返回值无意义），解除保护。若最后一个 WORD 已经非全 1，需执行擦除操作才能解除保护。先对 FLASH 执行擦除操作，将最后一个 WORD 恢复为全 1 值，然后读取 FLASH_PROTECT 寄存器，完成状态的更新，取消保护（读取 FLASH_PROTECT 返回值无意义）。

在 FLASH 处于保护状态下，为了方便客户实现二次烧录等特殊应用需求。LKS32MC07x 放开且仅放开了 NVR 区域的 Sector 2，供客户访问操作，包括读取、编程和擦除。注意，只有 FLASH 处于保护状态下，此功能才有效。同时，执行此操作的时候，我们只能通过访问如下寄存器 FLASH_NCFG/NADDR/NWDATA/NRDATA/NERASE/NKEY，实现此功能。

具体流程为：

- SYS_FLSE 寄存器，写入 0x8FCA，擦除操作使能开关 1
- SYS_FLSP 寄存器，写入 0x8F35，开启编程使能开关 2
- 读取 FLASH_PROTECT 寄存器，确定 FLASH 处于保护状态
- FLASH_NKEY，写入特殊值（0x0000_000F），开启保护状态下对 NVR Sector 2 的操作
- FLASH_NCFG，开启编程和擦除开关（若需要）
- FLASH_NADDR，写入编程或擦除地址

- FLASH_NERASE，实现对 NVR Sector 2 的擦除（若需要）
- FLASH_NWDATA，实现对 NVR Sector 2 的编程（若需要）
- FLASH_NRDATA，实现对 NVR Sector 2 的读取

切记，此通路只能访问到 NVR Sector 2，且一般访问完毕后，执行硬复位。

7.2.1.8 FLASH 在线升级(IAP)

IAP 模式，实现中断向量表的重映射。在 LKS32MC07x 系列芯片中，包含了系统寄存器 VTOR，其地址为 0xE000_ED08。用于重新映射中断向量表入口地址。

表 7-3 IAP VTOR 寄存器描述

名称	复位值	偏移	位置	权限	说明
VTOR	0x0		[31:7]	RW	执行写入操作，写入中断向量表入口地址

默认值为 0x0，此时中断向量表入口地址为 0x0。当写入非 0 值时，中断向量表入口地址将映射到 VTOR 寄存器所对应的地址上，立即生效。

在 LKS32MC07x 系列芯片中，因为有 VTOR 寄存器。用户可根据自己需求，更新整个 FLASH 的内容。在线升级过程中可以使用中断，也可以关闭中断。

7.2.1.8.1 开启中断的在线升级

推荐软件配置流程:

关闭 MCU 的中断控制器，暂时不接收新的中断响应；

在新的中断入口地址处，放置中断处理函数代码；

将新的中断入口地址写入 VTOR 寄存器；

开启 MCU 的中断控制器，使能中断；

用户跳转至在线升级函数，开始在线升级功能；

完成升级，关闭 MCU 的中断控制器，配置 VTOR 为默认值 0；

执行 MCU 软复位，系统 PC 重新从 0 地址开始执行升级后的程序；

7.2.1.8.2 关闭中断的在线升级

关闭 MCU 的中断控制器，暂时不接收新的中断响应；

用户跳转至在线升级函数，开始在线升级功能；如果在线升级使用了类似 UART 的外设通讯，需要 MCU 轮询处理 UART 的中断标志位。

执行 MCU 软复位，系统 PC 重新从 0 地址开始执行升级后的程序；

7.2.1.8.3 在线升级函数的位置

如果需要将 FLASH 全部擦除，则需要将在线升级函数放置在 RAM 中，如果需要使用中断则新的中断向量入口地址也需要位于 RAM 地址空间。

如果只需要擦除应用程序占用的部分 FLASH 区域，则可以将在线升级函数放置在 FLASH 高段地址的空闲区域，使用块擦除 FLASH 旧的应用程序，写入新的应用程序。

7.3 寄存器

7.3.1 地址分配

FLASH 控制器模块寄存器的基地址是 0x0002_0000 寄存器列表

表 7-4 FLASH 控制器模块寄存器列表

名称	偏移	说明
FLASH_CFG	0x00	FLASH 配置寄存器
FLASH_ADDR	0x04	地址寄存器
FLASH_WDATA	0x08	写数据寄存器
FLASH_RDATA	0x0C	读数据寄存器
FLASH_ERASE	0x10	擦除控制寄存器
FLASH_PROTECT	0x14	FLASH 保护状态寄存器
FLASH_READY	0x18	FLASH 工作状态寄存器
FLASH_SIZE	0x1C	FLASH 容量状态寄存器
FLASH_NCFG	0x20	FLASH 配置寄存器（保护状态下，针对 NVR 操作）
FLASH_NADDR	0x24	地址寄存器（保护状态下，针对 NVR 操作）
FLASH_NWDATA	0x28	写数据寄存器（保护状态下，针对 NVR 操作）
FLASH_NRDATA	0x2C	读数据寄存器（保护状态下，针对 NVR 操作）
FLASH_NERASE	0x30	擦除控制寄存器（保护状态下，针对 NVR 操作）
FLASH_NKEY	0x34	FLASH 密钥寄存器（保护状态下，针对 NVR 操作）

7.3.2 FLASH_CFG 配置寄存器（推荐先读回，按或/与方式修改）

地址:0x0002_0000

复位值:0xE0

表 7-5 FLASH_CFG 配置寄存器

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

RW		RW		RW		RW	
0		0		0		0	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERS_TYPE				REGION				WAIT		TBS					
RW				RW				RW		RW					
0				0				1		0x60					

位置	位名称	说明
[31]	ERS_EN	FLASH 擦除使能。默认为 0。 0:关闭擦除 1:开启擦除
[27]	PRG_EN	FLASH 编程使能。默认为 0。 0:关闭编程 1:开启编程
[23]	ADR_INC	FLASH 地址递增使能。默认为 0。 0:关闭递增使能 1:开启递增使能 当执行 FLASH 连续读写访问时，可以开启此功能减少对地址的操作。
[19]	PREF	FLASH 预取加速使能。默认为 0。 0:关闭加速 1:开启加速
[15]	ERS_TYPE	FLASH 擦除类型选择。默认为 0。 0:Sector 1:FULL
[11]	REGION	访问 FLASH 区域选择。默认为 0。 0:MAIN 1:NVR
[7]	WAIT	读取 FLASH 数据，等待开关。默认为 1。 0:读取，等待一个周期 1:读取，等待两个周期
[6:0]	TBS	编程/擦除时间基数寄存器,默认值为 0x60。只能配成如下几个值 0x60:96Mhz 系统频率下，FLASH 编程/擦除时间基数配置值。 0x30:48Mhz 系统频率下，FLASH 编程/擦除时间基数配置值。 0x17:24Mhz 系统频率下，FLASH 编程/擦除时间基数配置值。

7.3.3 FLASH_ADDR 地址寄存器

地址:0x0002_0004

复位值:0x0

表 7-6 FLASH_ADDR 地址寄存器

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

RW
0

位置	位名称	说明
[31:15]		未使用
[14:0]	ADDR	地址寄存器。读/写/擦除操作对应的地址寄存器。因按照 WORD 操作，最低两位会被 FLASH 控制器忽略。 执行擦除操作时，需要根据擦除类型，地址需要对齐。一个 Sector 是 512-Byte。若执行 Sector 擦除，地址需要是 512 的整数倍（若带偏移，偏移量会被忽略）。全芯片擦除，不会参考这个寄存器的值

7.3.4 FLASH_WDATA 写数据寄存器

地址:0x0002_0008

复位值:0x0

表 7-7 FLASH_WDATA 写数据寄存器

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

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

位置	位名称	说明
[31:0]	WDATA	执行写入操作，写入 FLASH 的值

7.3.5 FLASH_RDATA 读数据寄存器

地址:0x0002_000C

复位值:0x0

表 7-8 FLASH_RDATA 读数据寄存器

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															

位置	位名称	说明
[31:0]	RDATA	执行读取操作，读出 FLASH 的值

7.3.6 FLASH_ERASE 擦除控制寄存器

地址:0x0002_0010

复位值:0x0

表 7-9 FLASH_ERASE 擦除控制寄存器

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															

位置	位名称	说明
[31:0]	ERASE	写入 0x7654DCBA，触发擦除操作

7.3.7 FLASH_PROTECT 保护状态寄存器

地址:0x0002_0014

复位值:0x0

表 7-10 FLASH_PROTECT 保护状态寄存器

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															

位置	位名称	说明
[31:0]	PROTECT	读取该寄存器，更新加密/解密状态读取返回值无参考意义。

7.3.8 FLASH_READY

地址:0x0002_0018

复位值:0x0

表 7-11 FLASH_READY 工作状态寄存器

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

位置	位名称	说明
[31:1]		未使用
[0]	READY	1:FLASH 处于 Idle 状态；0:FLASH 处于 Busy 状态

7.3.9 FLASH_SIZE 容量寄存器

地址:0x0002_001C

复位值:0x00

表 7-12 FLASH_SIZE 容量寄存器

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

位置	位名称	说明
[31:8]	--	未使用
[7:6]	FLASH_SIZE	FLASH 容量寄存器，仅可读 0：32KB 1：64KB 2：128KB 3：128KB
[5:2]	--	未使用
[3:2]	ROM_SIZE	ROM 容量寄存器，仅可读 0：0KB 1：32KB 2：64KB 3：0KB

7.3.10 FLASH_NCFG 配置寄存器（推荐先读回，按或/与方式修改）

地址:0x0002_0020

复位值:0xB0

表 7-13 FLASH_NCFG 配置寄存器

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		RO		RW	
0		0		0		0	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERS_TYPE				REGION				WAIT		TBS					
RO				RW				RO		RO					
0				0				1		60					

位置	位名称	说明
[31]	ERS_EN	FLASH 擦除使能。默认为 0。 0:关闭擦除 1:开启擦除
[27]	PRG_EN	FLASH 编程使能。默认为 0。 0:关闭编程 1:开启编程
[23]	ADR_INC	FLASH 地址递增使能。默认为 0。此寄存器，只能读取不能修改。 0:关闭递增使能 1:开启递增使能 当执行 FLASH 连续读写访问时，可以开启此功能减少对地址的操作。
[19]	PREF	FLASH 预取加速使能。默认为 0。 0:关闭加速 1:开启加速
[15]	ERS_TYPE	FLASH 擦除类型选择。默认为 0。此寄存器，只能读取不能修改。 0:Sector 1:FULL
[11]	REGION	访问 FLASH 区域选择。默认为 0。修改此寄存器，无论写入值是 0 还是 1，最终，只能被强制写成 1。 0:MAIN 1:NVR
[7]	WAIT	读取 FLASH 数据，等待开关。默认为 1。此寄存器，只能读取不能修改。 0:读取，等待一个周期 1:读取，等待两个周期
[5:0]	TBS	编程/擦除时间基数寄存器,默认值为 0x60。**只能配成如下几个值。**此寄存器，只能读取不能修改。 0x60:96Mhz 系统频率下，FLASH 编程/擦除时间基数配置值。 0x30:48Mhz 系统频率下，FLASH 编程/擦除时间基数配置值。 0x17:24Mhz 系统频率下，FLASH 编程/擦除时间基数配置值。

7.3.11 FLASH_NADDR 地址寄存器

地址:0x0002_0024

复位值:0x0

表 7-14 FLASH_NADDR 地址寄存器

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

位置	位名称	说明
[31:15]		未使用
[14:0]	ADDR	地址寄存器。读/写/擦除操作对应的地址寄存器。因按照 WORD 操作，最低两位会被 FLASH 控制器忽略。 执行擦除操作时，需要根据擦除类型，地址需要对齐。一个 Sector 是 512-Byte。若执行 Sector 擦除，地址需要是 512 的整数倍（若带偏移，偏移量会被忽略）。全芯片擦除，不会参考这个寄存器的值

7.3.12 FLASH_NWDATA 写数据寄存器

地址:0x0002_0028

复位值:0x0

表 7-15 FLASH_NWDATA 写数据寄存器

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

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

位置	位名称	说明
[31:0]	WDATA	执行写入操作，写入 FLASH 的值

7.3.13 FLASH_NRDATA 读数据寄存器

地址:0x0002_002C

复位值:0x0

表 7-16 FLASH_NRDATA 读数据寄存器

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															

位置	位名称	说明
[31:0]	RDATA	执行读取操作，读出 FLASH 的值

7.3.14 FLASH_NERASE 擦除控制寄存器

地址:0x0002_0030

复位值:0x0

表 7-17 FLASH_NERASE 擦除控制寄存器

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															

位置	位名称	说明
[31:0]	ERASE	写入 0x7654DCBA，触发擦除操作

7.3.15 FLASH_NKEY 密钥寄存器

地址:0x0002_0034

复位值:0x0

表 7-18 FLASH_NKEY 密钥寄存器

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

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

位置	位名称	说明

[31:4]	--	未使用
[3:0]	KEY	当 FLASH 处于保护状态时，无法通过 SWD 读取，编程和擦除 FLASH。此时，用户可以对此寄存器写入特殊值（0x0000_000F），使能开关，配置 FLASH_NCFG/NADDR/NWDATA/NRDATA/NERASE 寄存器，对 NVR Sector 2 进行读取，编程和擦除。

7.3.16 NVR 校准参数地址信息

LKS32MC7x 校准参数使用

LKS32MC07x 产品存储芯片校准参数：

校准参数，每颗产品均独立校准，每颗产品不支持校准参数混用；

校准参数，在出厂前进行写入，出厂后不支持编程和擦除，仅支持读取；

校准参数，通过 LKS 公司提供的库函数进行读取访问；

校准参数，推荐关闭系统中断后，执行访问操作；

lks32mc07x_nvr.o 文件包含了校准参数的读取函数：

读取函数：uint32_t Read_Trim(uint32_t adr);

表 7-19 LKS32MC07x 校准值存放地址

地址	内容
0x00001420	ADC0_DC0 校准值，3.6V 量程 OFFSET
0x00001424	ADC0_AMC0 校准值，3.6V 量程 GAIN
0x00001428	ADC0_DC1 校准值，7.6V 量程 OFFSET
0x0000142C	ADC0_AMC1 校准值，7.6V 量程 GAIN
0x00001430	ADC1_DC0 校准值，3.6V 量程 OFFSET
0x00001434	ADC1_AMC0 校准值，3.6V 量程 GAIN
0x00001438	ADC1_DC1 校准值，7.6V 量程 OFFSET
0x0000143C	ADC1_AMC1 校准值，7.6V 量程 GAIN
0x00001450	DAC0 选择 4.85V 档位，SYS_AFE_DAC_AMC 校准值（扩大 512 倍结果）
0x00001454	DAC0 选择 4.85V 档位，SYS_AFE_DAC_DC 校准值
0x00001458	DAC0 选择 1.20V 档位，SYS_AFE_DAC_AMC 校准值（扩大 512 倍结果）
0x0000145C	DAC0 选择 1.20V 档位，SYS_AFE_DAC_DC 校准值
0x00001460	DAC1 选择 4.85V 档位，SYS_AFE_DAC_AMC 校准值（扩大 512 倍结果）
0x00001464	DAC1 选择 4.85V 档位，SYS_AFE_DAC_DC 校准值
0x00001468	DAC1 选择 1.20V 档位，SYS_AFE_DAC_AMC 校准值（扩大 512 倍结果）
0x0000146C	DAC1 选择 1.20V 档位，SYS_AFE_DAC_DC 校准值
0x00001480	OPA0，320K 欧姆 VS 10K 欧姆，GAIN 校准值（扩大 1000 倍结果）,R0 为 0 欧姆。
0x00001484	OPA0，160K 欧姆 VS 10K 欧姆，GAIN 校准值（扩大 1000 倍结果）,R0 为 0 欧姆。
0x00001488	OPA0，080K 欧姆 VS 10K 欧姆，GAIN 校准值（扩大 1000 倍结果）,R0 为 0 欧姆。

0x0000148C	OPA0，040K 欧姆 VS 10K 欧姆，GAIN 校准值（扩大 1000 倍结果）,R0 为 0 欧姆。
0x00001490	OPA1，320K 欧姆 VS 10K 欧姆，GAIN 校准值（扩大 1000 倍结果）,R0 为 0 欧姆。
0x00001494	OPA1，160K 欧姆 VS 10K 欧姆，GAIN 校准值（扩大 1000 倍结果）,R0 为 0 欧姆。
0x00001498	OPA1，080K 欧姆 VS 10K 欧姆，GAIN 校准值（扩大 1000 倍结果）,R0 为 0 欧姆。
0x0000149C	OPA1，040K 欧姆 VS 10K 欧姆，GAIN 校准值（扩大 1000 倍结果）,R0 为 0 欧姆。
0x000014A0	OPA2，320K 欧姆 VS 10K 欧姆，GAIN 校准值（扩大 1000 倍结果）,R0 为 0 欧姆。
0x000014A4	OPA2，160K 欧姆 VS 10K 欧姆，GAIN 校准值（扩大 1000 倍结果）,R0 为 0 欧姆。
0x000014A8	OPA2，080K 欧姆 VS 10K 欧姆，GAIN 校准值（扩大 1000 倍结果）,R0 为 0 欧姆。
0x000014AC	OPA2，040K 欧姆 VS 10K 欧姆，GAIN 校准值（扩大 1000 倍结果）,R0 为 0 欧姆。
0x000014B0	OPA3，320K 欧姆 VS 10K 欧姆，GAIN 校准值（扩大 1000 倍结果）,R0 为 0 欧姆。
0x000014B4	OPA3，160K 欧姆 VS 10K 欧姆，GAIN 校准值（扩大 1000 倍结果）,R0 为 0 欧姆。
0x000014B8	OPA3，080K 欧姆 VS 10K 欧姆，GAIN 校准值（扩大 1000 倍结果）,R0 为 0 欧姆。
0x000014BC	OPA3，040K 欧姆 VS 10K 欧姆，GAIN 校准值（扩大 1000 倍结果）,R0 为 0 欧姆。
0x000014C0	OPA0，共模电压（扩大 10000 倍结果）
0x000014C4	OPA1，共模电压（扩大 10000 倍结果）
0x000014C8	OPA2，共模电压（扩大 10000 倍结果）
0x000014CC	OPA3，共模电压（扩大 10000 倍结果）
0x000014D0	温度传感器，斜率校准值
0x000014D4	温度传感器，偏置校准值
0x00001500	OPA0，320K 欧姆:10K 欧姆挡位。高 16 位存放 R2 实际阻值，低 16 位存放 R1 实际阻值（扩大 100 倍结果）
0x00001504	OPA0，160K 欧姆:10K 欧姆挡位。高 16 位存放 R2 实际阻值，低 16 位存放 R1 实际阻值（扩大 100 倍结果）
0x00001508	OPA0，080K 欧姆:10K 欧姆挡位。高 16 位存放 R2 实际阻值，低 16 位存放 R1 实际阻值（扩大 100 倍结果）
0x0000150C	OPA0，040K 欧姆:10K 欧姆挡位。高 16 位存放 R2 实际阻值，低 16 位存放 R1 实际阻值（扩大 100 倍结果）
0x00001510	OPA1，320K 欧姆:10K 欧姆挡位。高 16 位存放 R2 实际阻值，低 16 位存放 R1 实际阻值（扩大 100 倍结果）
0x00001514	OPA1，160K 欧姆:10K 欧姆挡位。高 16 位存放 R2 实际阻值，低 16 位存放 R1 实际阻值（扩大 100 倍结果）
0x00001518	OPA1，080K 欧姆:10K 欧姆挡位。高 16 位存放 R2 实际阻值，低 16 位存放 R1 实际阻值（扩大 100 倍结果）
0x0000151C	OPA1，040K 欧姆:10K 欧姆挡位。高 16 位存放 R2 实际阻值，低 16 位存放 R1 实际阻值（扩大 100 倍结果）
0x00001520	OPA2，320K 欧姆:10K 欧姆挡位。高 16 位存放 R2 实际阻值，低 16 位存放 R1 实际阻值（扩大 100 倍结果）
0x00001524	OPA2，160K 欧姆:10K 欧姆挡位。高 16 位存放 R2 实际阻值，低 16 位存放 R1 实际阻值（扩大 100 倍结果）
0x00001528	OPA2，080K 欧姆:10K 欧姆挡位。高 16 位存放 R2 实际阻值，低 16 位存放 R1 实际阻值（扩大 100 倍结果）
0x0000152C	OPA2，040K 欧姆:10K 欧姆挡位。高 16 位存放 R2 实际阻值，低 16 位存放 R1 实际阻值（扩大 100 倍结果）
0x00001530	OPA3，320K 欧姆:10K 欧姆挡位。高 16 位存放 R2 实际阻值，低 16 位存放 R1 实际阻

	值（扩大 100 倍结果）
0x00001534	OPA3，160K 欧姆:10K 欧姆挡位。高 16 位存放 R2 实际阻值，低 16 位存放 R1 实际阻值（扩大 100 倍结果）
0x00001538	OPA3，080K 欧姆:10K 欧姆挡位。高 16 位存放 R2 实际阻值，低 16 位存放 R1 实际阻值（扩大 100 倍结果）
0x0000153C	OPA3，040K 欧姆:10K 欧姆挡位。高 16 位存放 R2 实际阻值，低 16 位存放 R1 实际阻值（扩大 100 倍结果）
0x00001560	OPA0 挡位下的 Offset 值（有符号类型，单位：V，放大 10000 倍） Bit[31:16] OPA0 4 倍放大倍数 Bit[15:0] OPA0 8 倍放大倍数
0x00001564	OPA0 挡位下的 Offset 值（有符号类型，单位：V，放大 10000 倍） Bit[31:16] OPA0 16 倍放大倍数 Bit[15:0] OPA0 32 倍放大倍数
0x00001568	OPA1 挡位下的 Offset 值（有符号类型，单位：V，放大 10000 倍） Bit[31:16] OPA1 4 倍放大倍数 Bit[15:0] OPA1 8 倍放大倍数
0x0000156C	OPA1 挡位下的 Offset 值（有符号类型，单位：V，放大 10000 倍） Bit[31:16] OPA1 16 倍放大倍数 Bit[15:0] OPA1 32 倍放大倍数
0x00001570	OPA2 挡位下的 Offset 值（有符号类型，单位：V，放大 10000 倍） Bit[31:16] OPA2 4 倍放大倍数 Bit[15:0] OPA2 8 倍放大倍数
0x00001574	OPA2 挡位下的 Offset 值（有符号类型，单位：V，放大 10000 倍） Bit[31:16] OPA2 16 倍放大倍数 Bit[15:0] OPA2 32 倍放大倍数
0x00001578	OPA3 挡位下的 Offset 值（有符号类型，单位：V，放大 10000 倍） Bit[31:16] OPA3 4 倍放大倍数 Bit[15:0] OPA3 8 倍放大倍数
0x0000157C	OPA3 挡位下的 Offset 值（有符号类型，单位：V，放大 10000 倍） Bit[31:16] OPA3 16 倍放大倍数 Bit[15:0] OPA3 32 倍放大倍数

8 SPI

8.1 概述

SPI 接口主要使用在，外部设计采用 SPI 协议的应用场景下。SPI 的工作模式软件可选，默认为 SPI Motorola 模式。SPI 接口支持全双工传输和半双工传输。当接口配置为 Master 模式时，可发送时钟信号供外部 Slave 设备使用。

8.2 主要特性

- 支持 Master 和 Slave 操作
- 全双工传输，可根据应用情况，使用 3 根或者 4 根信号线
- 支持半双工传输，可根据应用情况，使用 2 根信号线
- 可编程的时钟极性和相位
- 可编程的数据顺序：MSB 或 LSB
- 最快传输速度为系统最高时钟频率的 1/8
- 片选信号均可选。Master 模式下，片选信号可以软件控制也可以硬件产生；Slave 模式下，片选信号可以恒定有效，也可以来自外界设备
- 无本地 FIFO，支持 DMA 操作。包含溢出检测和片选信号异常检测
- 数据长度 8-Bit～16-Bit 可调

8.3 功能描述

8.3.1 功能框图

本接口采用同步串行设计，实现 MCU 同外部设备之间的 SPI 传输。支持轮询和中断方式获得传输状态信息。本接口的主要功能模块如下图所示。

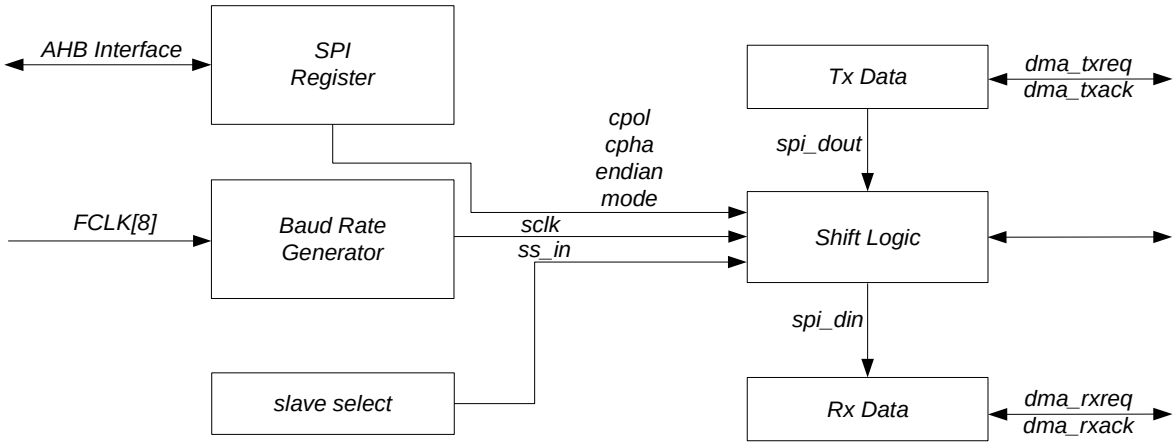


图 8-1 SPI 模块结构框图

接口信号包括，spi_din，spi_dout，sclk_in，sclk_out，ss_in 和 ss_out。

spi_din:接口接收的数据信号。同 SPI 协议比较，当接口配置为 Master 模式时，其等效为 MISO；当接口配置为 Slave 模式时，其等效为 MOSI。

spi_dout:接口发送的数据信号。同 SPI 协议比较，当接口配置为 Master 模式时，其等效为 MOSI；当接口配置为 Slave 模式时，其等效为 MISO。

sclk_in:接口接收的时钟信号。此时，接口的工作模式为 Slave。非 Slave 模式下，此信号输入无效。

sclk_out:接口发送的时钟信号。此时，接口的工作模式为 Master，非 Master 模式下，此信号输出恒定为 0。

ss_in:接口接收的片选信号。此时，接口的工作模式为 Slave。非 Slave 模式下，此信号输入无效。

ss_out:接口发送的片选信号。此时，接口的工作模式为 Master。非 Master 模式下，此信号输出恒定为 1。

8.3.2 功能说明

8.3.2.1 全双工模式

默认情况下，SPI 接口配置为全双工模式。此时，数据传输需要两根数据线。数据信号的变化，发生在时钟信号的边沿，即同步于时钟信号。

接口为 Master 模式时:

- spi_din 为数据输入，接外部 Slave 设备的 MISO
- spi_dout 为数据输出，接外部 Slave 设备的 MOSI

- spi_ss_out 为片选信号，根据应用情况选择是使用该信号还是软件控制其它 GPIO 实现

接口为 Slave 模式时:

- spi_din 为数据输入，接外部 Master 设备的 MOSI
- spi_dout 为数据输出，接外部 Master 设备的 MISO
- spi_ss_in 为片选信号，根据应用情况是使用该信号还是片选恒有效

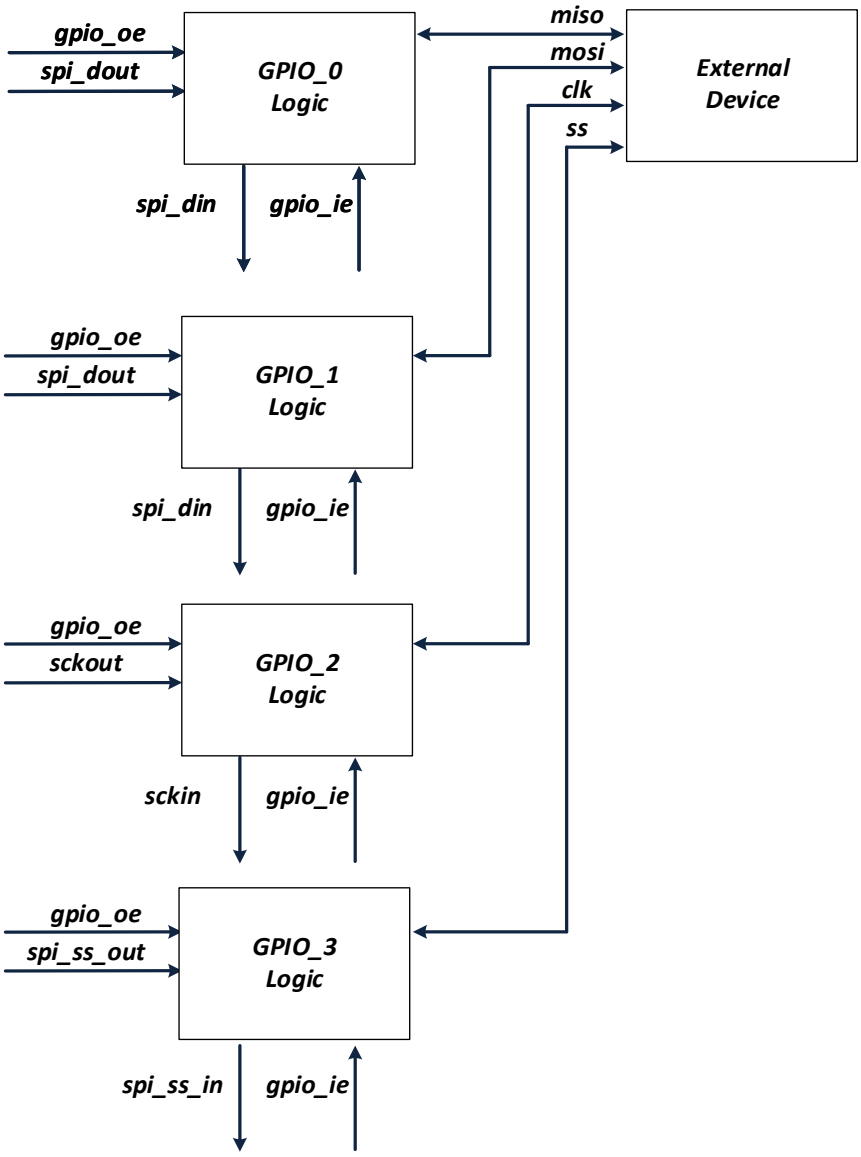


图 8-2 SPI 接口全双工模式互连框图

从上图可知，GPIO 若配置为输出，则 SPI 接口可发送数据；GPIO 若配置为输入，则 SPI 接口可接收数据。

8.3.2.2 半双工模式

SPI 接口可配置为半双工模式。此时，数据传输只需要一根数据线。数据信号的变化，发生在时钟信号的边沿，即同步于时钟信号。一次传输只能是一个方向的，要不就是发送，要不就是接

收。

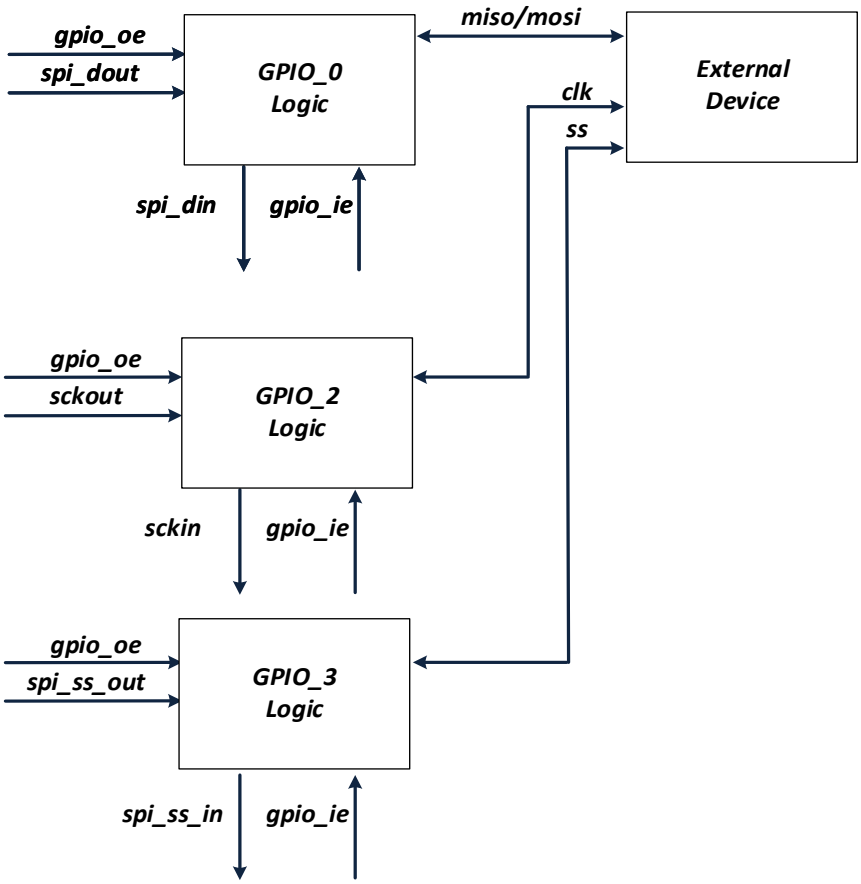


图 8-3 SPI 接口半双工模式互连框图

注意，上图中若本接口做 Master，则 clk 为本接口的输出信号；若本接口做 Slave，则 clk 为本接口的输入信号。

仅发送

SPI_CFG. DUPLEX 配置为 2，半双工发送模式有效。此时，本接口只能发送数据。GPIO_0 的 oe 使能，发送 spi_dout 数据到外界；GPIO_0 的 ie 关闭，spi_din 恒定输入为 0。此模式下，支持 Master/Slave 模式下的发送。

仅接收

SPI_CFG. DUPLEX 配置为 3，半双工接收模式有效。此时，本接口只能接收数据。GPIO_0 的 oe 关闭，spi_dout 无法发送数据到外界；GPIO_0 的 ie 开启，spi_din 接收来自外部的数据。此模式下，支持 Master/Slave 模式下的接收。

注意，全双工下是两个 GPIO 用于数据传输，半双工下可从中任意选一个 GPIO 用于数据传输。

8.3.2.3 片选信号

本接口做 Slave 模式时，片选信号可选，SPI_CFG.CS 决定片选来源。ss 为 Master 设备发出的选通使能信号，低电平有效。

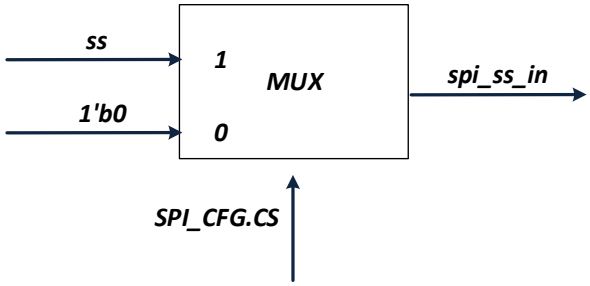


图 8-4 SPI 模块 Slave 模式片选信号选择

本接口做 Master 模式时，片选信号亦可选。模块硬件产生了标准的片选信号，实际应用可屏蔽此信号通过软件操作额外的 GPIO 实现。

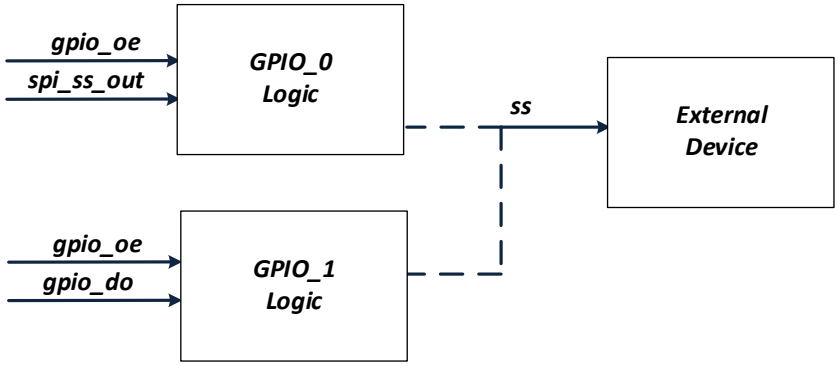


图 8-5 SPI 模块 Master 模式片选信号选择

注意，图 16-5 虚线仅表示不确定。若使用 spi_ss_out 为 ss 的源头，那么将 GPIO_0 同外界设备互连；若使用软件操作 GPIO 的方式，那么可将 GPIO_1 同外界设备互连。

8.3.2.4 通讯格式

在 SPI 通讯过程中，发送或者接收操作均是基于 SPI 时钟的。通讯格式受到 SPI_CFG. SAMPLE 和 SPI_ CLK_POL 控制。SPI_CFG. SAMPLE 为 Phase 控制位，SPI_ CLK_POL 为 Polarity 控制位。

Polarity 控制了 SPI 时钟信号在默认情况下的电平状态。Polarity 为 0 时，默认时钟电平为低电平；Polarity 为 1 时，默认电平为高电平。

Phase 控制了 SPI 数据的发送/接收时刻。Phase 为 0 时，时钟从默认电平到第一个跳变边沿为采样数据时刻，Phase 为 1 时，时钟从默认电平到第一个跳变边沿为发送数据时刻。

8.3.2.5 数据格式及长度

SPI 数据传输格式分成两种:MSB 和 LSB。数据传输格式受到 SPI_CFG.ENDIAN 控制。注意，在数据传输过程中硬件自动实现传输格式的转换，无需软件介入。

SPI 数据长度可配，范围是从 8-Bit 到 16-Bit。SPI_SIZE.BITSIZE 控制长度。

8.3.2.6 DMA 传输

在大容量数据传输应用下，SPI 接口支持 DMA 传输，减轻 MCU 的负担。一次传输，最大传输量为由 DMA 模块决定，最小传输量为 1 字节。在全双工模式下，接收和发送均可实现 DMA 传输；

在半双工模式下，仅接收或发送实现 DMA 传输。

在接收到新的数据后，硬件自动产生 DMA 请求，通过 DMA 模块将数据搬移到 RAM 中。在发送新数据前，硬件自动产生 DMA 请求，通过 DMA 模块将数据从 RAM 中搬移到 SPI 接口。因 SPI 无 FIFO，SPI 发送一次 DMA 请求，DMA 只能搬移一个字节数据。**若要实现多字节搬移，DMA 需配置为多轮搬移，每一轮搬移一个字节的方式**。

DMA 传输需要配置 DMA 模块相应寄存器。

因 SPI 接口支持 DMA 传输，也支持 MCU 传输。两者区别在于--DMA 传输，发送的数据来自 DMA 的搬移；MCU 传输，发送的数据来自 MCU 的搬移。

DMA 传输，推荐软件配置流程如下：

- 初始化 DMA 模块，将本次发送的数据来源，接收的数据去向配置好，传输长度配置完毕。
- 初始化 GPIO 模块，将 SPI 复用的 GPIO 配置完毕。
- 初始化 SPI 接口，SPI_IE/SPI_CFG/SPI_BAUD/SPI_SIZE 等寄存器配置完毕。
- 触发 SPI 接口，进入发送/接收状态。触发条件是 MCU 对 SPI_TXDATA 寄存器执行写操作，因最终发送的数据来自 DMA，本次 MCU 写入的数据不会混入 SPI 发送流程。

8.3.2.7 MCU 传输

一次只能发送/接收一个 SPI_SIZE.BITSIZE 长度的数据，每次完成后需要通过中断或者轮询的方式判断传输是否完成。无论是主模式还是从模式，写 SPI_TXDATA 寄存器，才能触发传输。主模式为主动发送，从模式为加载数据到发送队列等待主模式发出时钟信号，开始传输。推荐软件配置流程如下:

- 初始化 GPIO 模块，将 SPI 复用的 GPIO 配置完毕。
- 初始化 SPI 接口，SPI_IE/SPI_CFG/SPI_BAUD/SPI_SIZE 等寄存器配置完毕。
- MCU 对 SPI_TXDATA 寄存器执行写操作，触发 SPI 接口进入发送流程。从模式下，数据加载到内部状态机等待主设备发起读取操作；主模式下，触发发送。

注意：若需要连续发送，则需要重新配置 SPI_TXDATA 寄存器。通过中断或者轮询方式获得传输完毕信息。

8.3.2.8 外部事件传输

在大容量数据传输应用下，SPI 接口支持 DMA 传输。传输过程中，可被打断也可不被打断。所谓打断是指，当前字节传输完成，需等待外部事件才开始下一个字节的传输；不打断是指，当前字节传输完成，直接下一个字节的传输。打断模式需要同其它模块配合使用。例如定时器模块，定时器可触发下一字节的传输。打断模式仅在 Master 模式有效。SPI_IE.TRANS_TRIG 控制了是否使用打断模式。

8.3.2.9 中断处理

SPI 接口包含三种类型的中断事件，分别是：数据传输完成事件，异常事件和溢出事件。

- 数据完成事件，当前数据传输完成。高电平有效，对 SPI_IE. CMPLT_IF 写 1 清除。
- 异常事件，SPI 接口为 Slave 模式，若在传输过程中片选信号受到干扰，被拉高，将产生片选异常事件。高电平有效，对 SPI_IE. AB_IF 写 1 清除。
- 溢出事件，SPI_RX_DATA 寄存器数据没有及时被读走，，将产生溢出事件。高电平有效，对 SPI_IE. OV_IF 写 1 清除。

上述事件，默认是不触发 SPI 中断，可以通过配置 SPI_IE[7:4]使能事件产生中断。

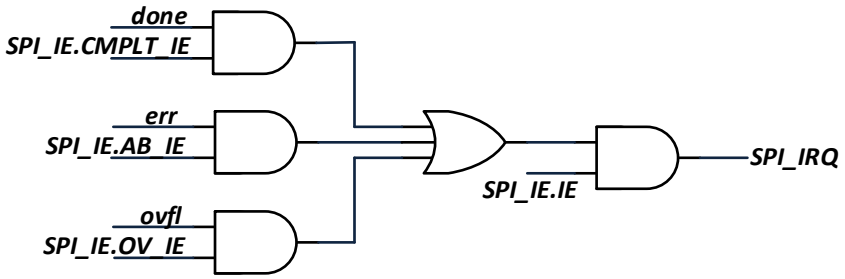


图 8-6 SPI 模块中断选信号产生图

数据传输完毕后，可通过 DMA 中断或者 SPI 自身中断判断结束。

- 发送模式下，DMA 先完成搬移操作，SPI 后发送完毕。SPI 发送完毕，可作为本次传输完成标识。
- 接收模式下，SPI 接收完毕，触发 DMA 搬移。DMA 搬移完成，可作为本次传输完成标识。

溢出事件。在全双工模式下，发送和接收的 DMA 均有效，一般情况下不会发送溢出事件；在半双工模式下，只有发送（或接收），此时硬件屏蔽了接收（或发送）的溢出判断

8.3.2.10 波特率设置

SPI 接口时钟通过对系统时钟分频获得，分频系数来自 SPI_BAUD. BAUD。SPI 传输波特率配置计算公式为:

SPI 传输波特率 = 系统时钟 / (2*(BAUD + 1))

SPI 协议为半拍协议，上升沿发送数据，下降沿采集数据；或者下降沿发送数据，上升沿采样数据。

SPI 接口采用同步设计，需要对外部设备的信号进行同步采样，同步时钟为系统时钟。数据和时钟信号（此时为 Slave 模式）的同步，需要两拍系统时钟。考虑到时钟相位的偏移，此时需要一拍系统时钟的冗余，由此推导出最快的 SPI 传输波特率为系统时钟的 1/8，高电平周期为四拍系统时钟，低电平周期为四拍系统时钟。因此，SPI_BAUD.BAUD 的配置值不能小于 3。

8.4 寄存器

8.4.1 地址分配

SPI 模块寄存器的基地址是 0x4001_0000，模块寄存器列表如下。

表 8-1 SPI 模块控制寄存器列表

名称	偏移	说明
SPI0_CFG	0x00	SPI 配置寄存器
SPI0_IE	0x04	SPI 中断寄存器
SPI0_BAUD	0x08	SPI 波特率寄存器
SPI0_TXDATA	0x0C	SPI 发送数据寄存器
SPI0_RXDATA	0x10	SPI 接收数据寄存器
SPI0_SIZE	0x14	SPI 传输数据长度寄存器

8.4.2 SPI0_CFG SPI 控制寄存器

地址:0x4001_0000

复位值:0x0

表 8-2 系统控制寄存器 SPI0_CFG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								DUPLEX	CS	MS	CPHA	CPOL	ENDIAN	EN	
								RW	RW	RW	RW	RW	RW	RW	
								0	1	0	0	1	0	0	

位置	位名称	说明
[31:8]		未使用
[7:6]	DUPLEX	半双工模式设置 0X:关闭半双工模式 10:开启半双工模式，仅发送 11:开启半双工模式，仅接收
[5]	CS	SPI 从设备下，片选信号来源。默认值为 1。 0:Slave 模式下，片选信号恒为有效值--0 1:Slave 模式下，片选信号来自 Master 设备
[4]	MS	SPI 主从模式选择。默认值为 0。 0:Slave 模式 1:Master 模式
[3]	CPHA	SPI 相位选择。默认值为 0。 0:Phase 为 0 1:Phase 为 1
[2]	CPOL	SPI 极性选择。默认值为 0。 0:Polarity 为 0 1:Polarity 为 1
[1]	ENDIAN	SPI 模块传输顺序。默认值为 0。 0:MSB，高位先传输 1:LSB，低位先传输

[0]	EN	SPI 模块使能信号。默认值为 0。 0:关闭 SPI 模块 1:开启 SPI 模块

8.4.3 SPI0_IE SPI 中断寄存器

地址:0x4001_0004

复位值:0x0

表 8-3 SPI0_IE 中断寄存器

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

位置	位名称	说明
[31:8]		未使用
[7]	IE	SPI 中断使能开关。默认值为 0。 0:关闭 SPI 中断 1:使能 SPI 中断
[6]	CMPLT_IE	SPI 传输，完成事件中断使能信号。 0:屏蔽此中断源 1:使能此中断源
[5]	AB_IE	SPI 传输，异常事件中断使能信号。 0:屏蔽此中断源 1:使能此中断源
[4]	OV_IE	SPI 传输，溢出事件中断使能信号。默认值为 0。 0:屏蔽此中断源 1:使能此中断源
[3]	TRANS_TRIG	传输触发选择。 1：外部触发 0：内部自动执行。仅主模式有效
[2]	CMPLT_IF	SPI 传输，完成事件。高电平有效，写 1 清除。
[1]	AB_IF	SPI 传输，异常事件。Slave 模式下，传输未完成，发生片选信号无效事件。高电平有效，写 1 清除。
[0]	OV_IF	SPI 传输，溢出事件。前次接收的旧数据没有被取得走，本次接收的新数据已经到达。 高电平有效，写 1 清除。

8.4.4 SPI0_BAUD SPI 波特率寄存器

地址:0x4001_0008

复位值:0x0

表 8-4 SPI0_BAUD 控制寄存器

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

位置	位名称	说明
[31:9]		未使用
[15]	TRANS_MODE	SPI 数据搬移方式。默认为 0，DMA 方式。 0：SPI 接口支持 DMA 搬移数据到 SPI 接口，完成发送和接收。 1：SPI 接口支持 MCU 搬移数据到 SPI 接口，完成发送和接收。
[14:12]		未使用
[11:0]	BAUD	SPI 传输波特率配置，SPI 实际传输速度计算公式为: SPI 传输速度 = 系统时钟 / (2*(BAUD + 1)) 切记，BAUD 的配置值不能小于 3。

8.4.5 SPI0_TXDATA SPI 数据发送寄存器

地址:0x4001_000C

复位值:0x0

表 8-5 SPI0_TXDATA 数据发送寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	TX_DATA	SPI 数据发送寄存器

8.4.6 SPI0_RXDATA SPI 数据接收寄存器

地址:0x4001_0010

复位值:0x0

表 8-6 SPI0_RXDATA 数据接收寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	RX_DATA	SPI 数据接收寄存器

8.4.7 SPI0_SIZE SPI 数据传输长度寄存器

地址:0x4001_0014

复位值:0x0

表 8-7 SPI0_SIZE 数据传输长度寄存器

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

位置	位名称	说明
[31:5]		未使用
[4:0]	BITSIZE	字节长度寄存器。 0x00:非法值 0x07:非法值 0x08:8-Bit 0x09:9-Bit … 0x0E:14-Bit 0x0F:15-Bit 0x10:16-Bit

9 I2C

9.1 概述

I2C 总线接口连接微控制器和串行 I2C 总线。它提供多主机功能，控制所有 I2C 总线特定的时序、协议、仲裁和定时。支持标准和快速两种模式。

9.2 主要特性

- 多主机功能：该模块既可做主设备也可做从设备。

  I2C 主设备功能：产生时钟、START 和 STOP 事件。

  I2C 从设备功能：可编程的 I2C 硬件地址比较（仅支持 7 位硬件地址）、停止位检测。
- 根据系统分频，实现不同的通讯速度。
- 状态标志：发送器/接收器模式标志、字节发送结束标志、I2C 总线忙标志。
- 错误标志：主模式时的仲裁丢失、地址/数据传输后的应答(ACK)错误、检测到错位的起始或停止条件。
- 一个中断向量，包含五个中断源：总线错误中断源、完成中断源、NACK 中断源、硬件地址匹配中断源和传输完成中断源。
- 具单字节缓冲器的 DMA。

9.3 功能描述

9.3.1 功能框图

本接口采用同步串行设计，实现 MCU 同外部设备之间的 I2C 传输。支持轮询和中断方式获得传输状态信息。本接口的主要功能模块如下图所示。

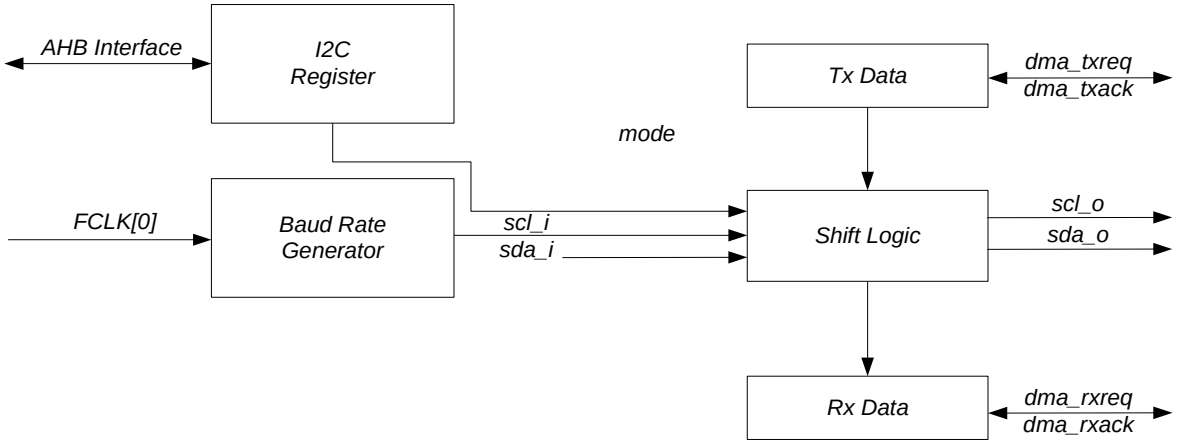


图 9-1I2C 模块顶层功能框图

I2C 接口同外界通讯只有 SCL 和 SDA 两根信号线。SDA 为双向复用信号线，受到 sda_oe 控制。模块级，I2C 接口信号包括，scl_i，sda_i，scl_o，sda_o 和 sda_oe。

scl_i:时钟信号。当 I2C 接口配置为从模式时，此为 I2C 总线的时钟输入信号。

sda_i:数据信号。当 I2C 接口接收数据时（无论主模式还是从模式），此为 I2C 总线的数据输入信号。

scl_o:时钟信号。当 I2C 接口配置为主模式时，此为 I2C 总线的时钟输出信号。

sda_o:数据信号。当 I2C 接口发送数据时（无论主模式还是从模式），此为 I2C 总线的数据输出信号。

sda_oe:数据使能信号。当 sda_o 输出时，sda_oe 有效；当 sda_i 输入时，sda_oe 无效。

9.3.2 功能说明

I2C 模块接收和发送数据，并将数据从串行转换成并行，或并行转换成串行。可以开启或禁止中断。接口通过数据引脚(SDA)和时钟引脚(SCL)连接到 I2C 总线。

9.3.2.1 模式选择

接口可以下述 4 种模式中的一种运行:

- 从发送模式
- 从接收模式
- 主发送模式
- 主接收模式

I2C 接口默认主从均不使能。接口根据配置情况，进入主模式或者从模式。当仲裁丢失或产生停止信号时，主模式自动释放总线并产生相应异常中断。允许多主机功能。

主模式时，I2C 接口启动数据传输并产生时钟信号。串行数据传输总是以起始条件开始并以停止条件结束。起始条件和停止条件都是在主模式下由软件控制产生。

从模式时，I2C 接口能识别它自己的地址(7 位)。软件能够控制开启或禁止硬件地址比较功能，硬件地址比较功能可降低 MCU 的负担。只有地址匹配才通知 MCU 进行相关处理。

数据和地址按 8 位/字节进行传输，高位在前。跟在起始条件后的 1 个字节是地址。地址只在主模式发送。

在一个字节传输的 8 个时钟后的第 9 个时钟期间，接收器必须回送一个应答位(ACK)给发送器。

软件可以开启或禁止应答(ACK)，并可以设置 I2C 接口的地址。

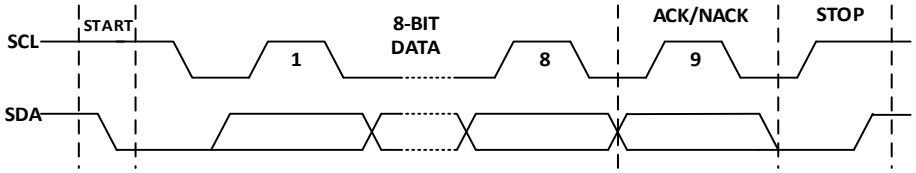


图 9-2 基本 I2C 传输时序图

I2C 接口没有 FIFO，若一次性发送大量数据，为了降低 MCU 的负担，需 DMA 配合。I2C 接口支持 DMA 传输（多字节传输）和非 DMA 传输（单字节传输）。上述四种传输模式进一步扩展为：

- 从模式单字节发送，从模式 DMA 发送
- 从模式单字节接收，从模式 DMA 接收
- 主模式单字节发送，主模式 DMA 发送
- 主模式单字节接收，主模式 DMA 接收

一般情况下，非 DMA 方式时，一次传输一个字节（可反复单次传输，需软件介入提供数据）。DMA 方式，一次连续传输可以多字节（最大不超过 32 字节，极端情况一次传输一个字节，因无 FIFO，每次 DMA 请求，仅传输一个字节，多轮完成本次数据传输）。

上述所有模式，遵循如下基本原则：

- 单字节发送，中断将在 8-bit 数据发送完毕且收到响应后（ACK/NACK 均可）产生。
- 单字节接收，中断将在 8-bit 数据接收完毕后产生。
- DMA 发送，正常情况下，中断将在数据发送完毕且收到响应后（ACK/NACK 均可）产生。
- DMA 接收，正常情况下，中断将在数据接收完毕后产生。
- 当 I2C 接口配置为主模式时，检测到错误后，I2C 接口会主动释放总线，恢复到起始状态并产生中断信号。

9.3.2.2 从模式

默认情况下，I2C 接口主模式和从模式均关闭。若工作在从模式，需使能从模式。为了产生正确的时序，必须通过系统寄存器 SYS_CLK_DIV0 设定 I2C 接口的工作时钟频率，I2C 接口时钟基于系统高速主时钟进行分频，SYS_CLK_DIV0 是 I2C 接口工作时钟的分频系数。

- 从模式下，I2C 接口时刻在监控总线上的信号。一旦检测到起始条件，其将保存地址位数据和读写位数据。
- 从模式下，若硬件地址匹配功能开启，只有地址匹配的情况下，才会产生中断，通知 MCU 进行后续处理。若没有开启，每次收到地址及读写位数据，都将产生中断。
- 从模式下，单字节接收模式。每次收到一个字节的数据后，产生中断，此时 I2C 接口可拉低 SCL，直至中断完成，继续后续操作。
- 从模式下，单字节发送模式。每次发送一个字节完毕后且收到响应（ACK/NACK），产生中断，此时 I2C 接口可拉低 SCL，直至中断完成，继续后续操作。
- 从模式下，DMA 接收模式。每次收到 SIZE 约定后的数据，产生中断，此时 I2C 接口可拉低 SCL，直至中断完成。
- 从模式下，DMA 发送模式。每次发送 SIZE 约定后的数据并收到响应（ACK/NACK），产生中

断，此时 I2C 接口可拉低 SCL，直至中断完成。

9.3.2.2.1 从模式传输

单字节传输，并不意味着仅仅传输一个字节数据，其含义为每次传输完一个字节的数据后，将产生中断判断是否还要继续传输。单字节传输的极端情况是仅传输一个字节的数据。下图为单字节传输的总线示意图。从图可知，一般单字节传输的流程如下：

- 地址匹配，产生地址匹配中断，准备开始传输。
- 若是接收模式，一个字节接收完毕后，产生中断，软件判断是否继续接收，返回 ACK/NACK 响应。
- 若是发送模式，一个字节发送完毕后，等待响应（ACK/NACK），产生中断，根据响应判断后续操作。
- 获得总线 STOP 事件，本次传输完成。

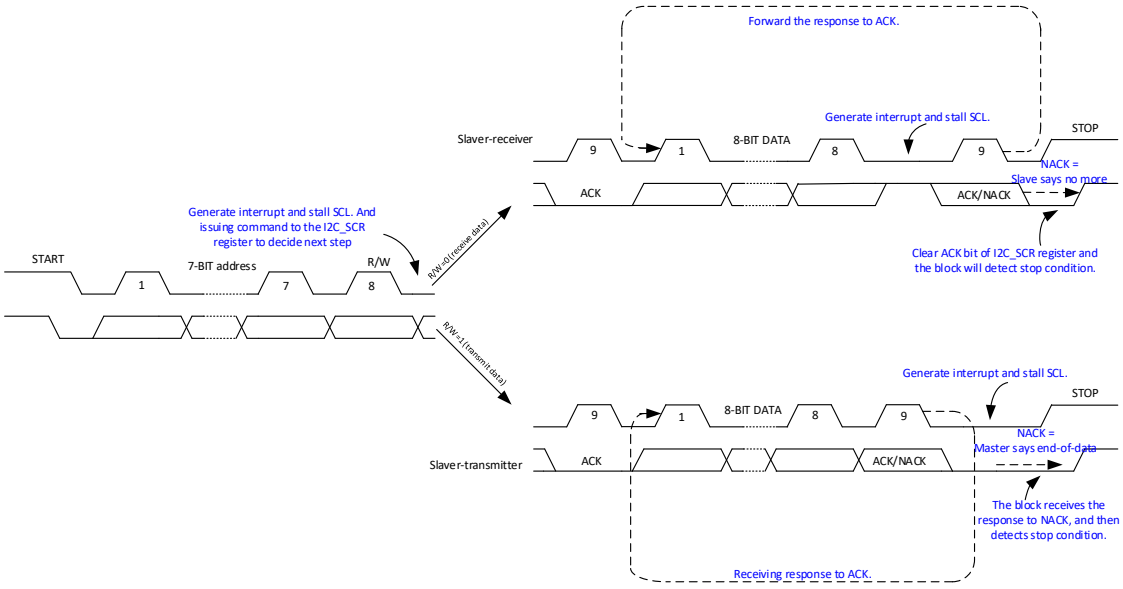


图 9-3 从模式传输示意图

9.3.2.2.2 从模式发送

地址匹配后，从发送器将字节从 I2C_DATA 寄存器经由内部移位寄存器发送到 SDA 线上。在 I2C_DATA 数据没有准备好之前，从设备可拉底 SCL，直到待发送数据已写入 I2C_DATA 寄存器。I2C 接口在发送完毕每个字节后都执行下列操作：

- 如果接收到 ACK 位，装载下一个字节数据，继续传输。装载过程中，可拉底 SCL。
- 如果接收到 NACK 位，停止下一个字节的装载。

- 等待 STOP 事件，停止本次传输。

9.3.2.2.3 从模式单字节接收

地址匹配后，从接收器将通过内部移位寄存器从 SDA 线接收到的数据存入 I2C_DATA 寄存器。I2C 接口在接收到每个字节后都执行下列操作:

- 如果设置了 ACK 位，在一个字节接收完毕后，产生一个 ACK 应答脉冲。
- 如果清除了 ACK 位，在一个字节接收完毕后，产生一个 NACK 应答脉冲。
- 等待 STOP 事件，结束本次传输。

9.3.2.3 从模式 DMA 传输

从模式下，一般仅配置 I2C 时钟、从地址以及硬件地址匹配使能，等待总线有访问请求后，根据芯片实际情况决定是否响应本次传输请求。DMA 传输，其含义为每次传输多个字节的数据后，将产生中断判断是否还要继续传输。DMA 传输的极端情况是仅传输一个字节的数据。DMA 传输，一般建议开启硬件地址比较功能，NACK 中断，传输完成中断。一般 DMA 传输的流程如下:

- 配置 I2C 从地址，使能 I2C 中断（可使能硬件地址比较中断）。地址匹配，产生 I2C 地址匹配中断，在中断处理函数中，配置 DMA，准备好发送数据或者准备好接收地址。然后写 I2C_SCR，准备开始传输或者停止本次传输。
- 若是接收模式，I2C_BSIZE.SIZE 约定的字节接收完毕后，产生中断，软件判断是否继续接收，返回 ACK/NACK 响应。
- 若是发送模式，I2C_BSIZE.SIZE 约定的字节发送完毕后，等待响应（ACK/NACK），产生中断，根据响应判断后续操作。
- 获得总线完成标志，本次传输完成。

9.3.2.3.1 从模式 DMA 发送

地址匹配后，配置完毕 DMA。通过发送 DMA 请求，将字节从 RAM 搬移到 I2C_DATA 寄存器，然后经由内部移位寄存器发送到 SDA 线上。在 I2C_DATA 数据没有准备好之前，从设备可拉底 SCL，直到待发送数据已写入 I2C_DATA 寄存器。从模式下发送数据，需要软件协助触发第一次 DMA 搬移，配置 I2C_BCR.BYTE_CMPLT 为 1 即可。I2C 接口在发送完毕 I2C_BSIZE.SIZE 约定的字节数据后都执行下列操作:

- 如果接收到 ACK 位，配置 DMA，准备下一批数据，继续传输。准备过程中，可拉低 SCL。
- 如果接收到 NACK 位，停止准备下一批数据，停止本次传输。

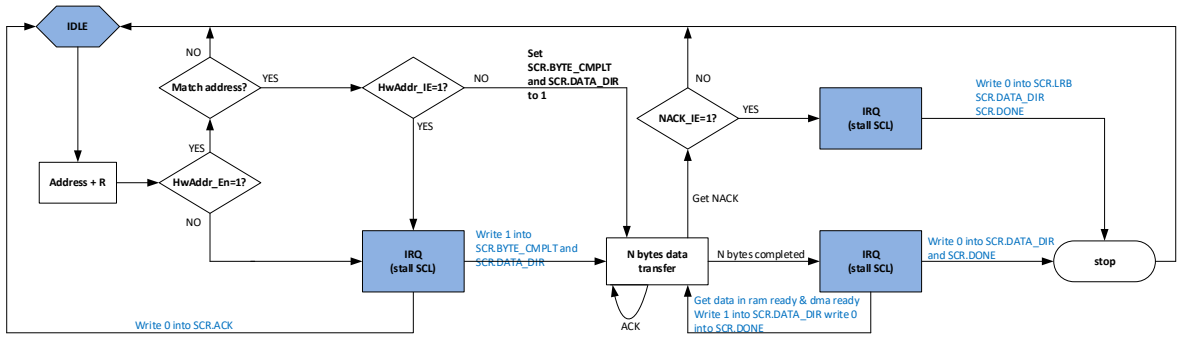


图 9-4 从模式下多字节发送示意图

9.3.2.3.2 从模式 DMA 接收

地址匹配后，配置好 DMA，从 SDA 线接收到的数据先存入 I2C_DATA 寄存器，然后通过 DMA 搬移到 RAM。I2C 接口在接收到一个字节后都将执行 DMA 请求。完成 I2C_BCR.SIZE 约定的数据传输后，执行下列操作:

- 如果设置了 ACK 位，在 I2C_BCR.SIZE 约定数据接收完毕后，产生一个 ACK 应答脉冲。
- 如果清除了 ACK 位，在 I2C_BCR.SIZE 约定数据接收完毕后，产生一个 NACK 应答脉冲。

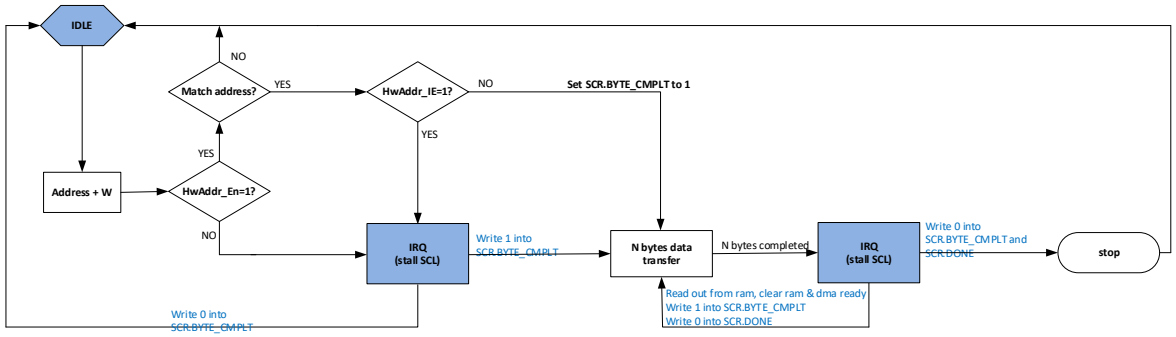


图 9-5 从模式下多字节接收示意图

9.3.2.4 主模式

默认情况下，I2C 接口主模式和从模式均关闭。若工作在主模式，需使能主模式。为了产生正确的时序，必须在系统寄存器 CLK_DIV0 中设定 I2C 接口的工作时钟。

I2C 接口执行主模式传输之前，需要判断总线是否空闲。可读取 I2C_MSCR 寄存器的 BIT3，查询当前总线状态。若总线处于忙的状态，可以开启 I2C 中断，通过收到 STOP 中断事件判断总线是否空闲下来。只有空闲状态下，才能正常发送 START 状态，以及后续的数据。

9.3.2.4.1 主模式单字节传输

单字节传输，并不意味着仅仅传输一个字节数据，其含义为每次传输完一个字节的数据后，

将产生中断判断是否还要继续传输。单字节传输的极端情况是仅传输一个字节的数据。图 6 为单字节传输的总线示意图。从图可知，一般单字节传输的流程如下:

- 判断总线是否空闲，若空闲，准备开始传输。
- 若是接收模式，一个字节接收完毕后，产生中断，软件判断是否继续接收，返回 ACK/NACK 响应。
- 若是发送模式，一个字节发送完毕后，等待响应（ACK/NACK），产生中断，根据响应判断后续操作。
- 发送总线 STOP 事件，本次传输完成。

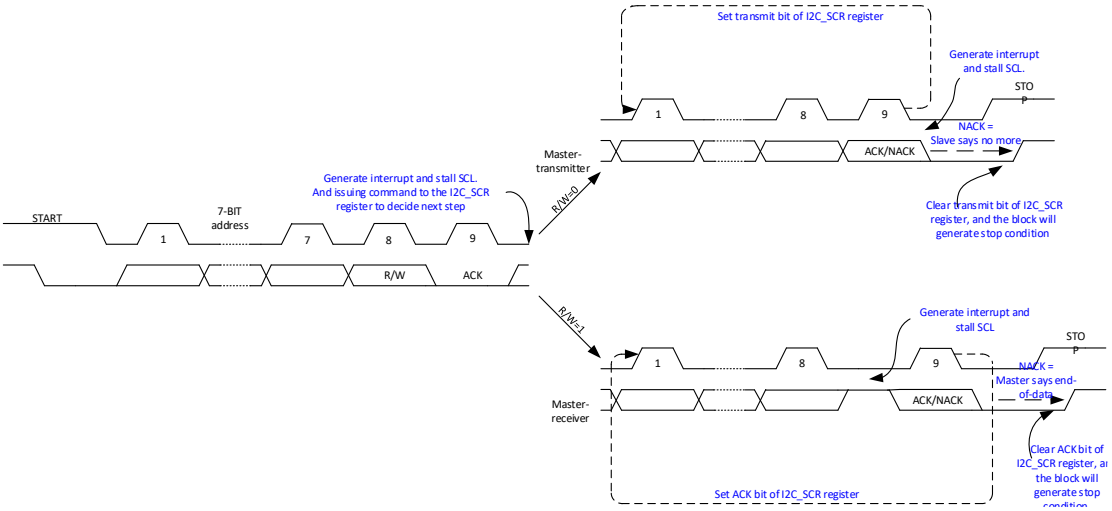


图 9-6 主模式下单字节传输示意图

9.3.2.4.2 主模式单字节发送

开始传输后，I2C 接口将字节从 I2C_DATA 寄存器经由内部移位寄存器发送到 SDA 线上。在 I2C_DATA 数据没有准备好之前，主设备可不产生 SCL 时钟信号，直到待发送数据已写入 I2C_DATA 寄存器。I2C 接口在发送完毕每个字节后都执行下列操作:

- 如果接收到 ACK 位，装载下一个字节数据，继续传输。装载过程中，可拉底 SCL。
- 如果接收到 NACK 位，停止下一个字节的装载。
- 产生 STOP 事件，结束本次传输。

9.3.2.4.3 主模式单字节接收

开始传输后，I2C 接口将通过内部移位寄存器从 SDA 线接收到的数据存入 I2C_DATA 寄存器。I2C 接口在接收到每个字节后都执行下列操作:

- 如果设置了 ACK 位，在一个字节接收完毕后，产生一个 ACK 应答脉冲。
- 如果清除了 ACK 位，在一个字节接收完毕后，产生一个 NACK 应答脉冲。

- 产生 STOP 事件，结束本次传输。

9.3.2.4.4 主模式 DMA 传输

DMA 传输，其含义为每次传输多个字节的数据后，将产生中断判断是否还要继续传输。DMA 传输的极端情况是仅传输一个字节的数据。DMA 传输，一般建议开启 NACK 中断，传输完成中断。一般 DMA 传输的流程如下:

- 总线空闲，准备开始传输。
- 若是接收模式，I2C_BSIZE.SIZE 约定的字节接收完毕后，产生中断，软件判断是否继续接收，返回 ACK/NACK 响应。
- 若是发送模式，I2C_BSIZE.SIZE 约定的字节发送完毕后，等待响应（ACK/NACK），产生中断，根据响应判断后续操作。
- 发送 STOP 事件，本次传输完成。

9.3.2.4.5 主模式 DMA 发送

总线空闲，配置完毕 DMA。通过发送 DMA 请求，将字节从 RAM 搬移到 I2C_DATA 寄存器，然后经由内部移位寄存器发送到 SDA 线上。在 I2C_DATA 数据没有准备好之前，主设备可不产生 SCL 时钟，直到待发送数据已写入 I2C_DATA 寄存器。I2C 接口在发送完毕 SIZE 约定的字节数据后都执行下列操作:

- 如果接收到 ACK 位，配置 DMA，准备下一批数据，继续传输。准备过程中，可拉低 SCL。
- 如果接收到 NACK 位，停止加载下一批数据。
- 如果本次数据发送完毕，停止后续发送。
- 产生 STOP 事件，停止本次传输。

异常情况是:

- 若从设备地址不匹配或者从设备没有准备好，此时从设备将返回 NACK
- 主设备产生 STOP 事件，停止本次传输。

等待一段时间后，重新配置 I2C 寄存器，关闭 DMA 对应的通道使能信号，重新配置 DMA 寄存器，再次发送传输请求。关闭 DMA 对应通道是因为 I2C 有预取，DMA 已经不是初始状态。

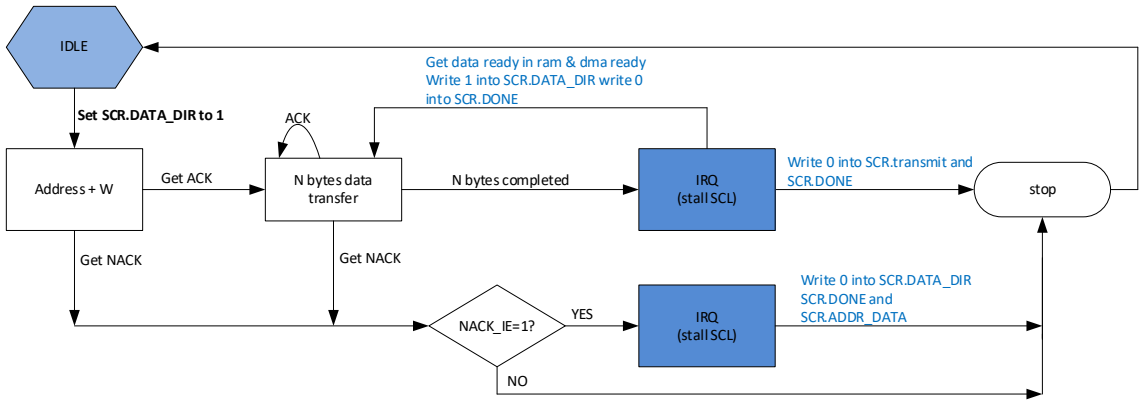


图 9-7 主模式下多字节发送示意图

9.3.2.4.6 主模式 DMA 接收

总线空闲，配置好 DMA 从 SDA 线接收到的数据先存入 I2C_DATA 寄存器，然后通过 DMA 搬移到 RAM。I2C 接口在接收到一个字节后都将执行 DMA 请求。完成 I2C_BCR.BURST_SIZE 约定的数据传输后，执行下列操作:

- 如果设置了 ACK 位，在 I2C_BCR.SIZE 约定数据接收完毕后，产生一个 ACK 应答脉冲。
- 如果清除了 ACK 位，在 I2C_BCR.SIZE 约定数据接收完毕后，产生一个 NACK 应答脉冲。
- 产生 STOP 事件，停止本次传输。

异常情况是:

- 若从设备地址不匹配或者从设备没有准备好，此时从设备将返回 NACK。
- 主设备产生 STOP 事件，停止本次传输。
- 等待一段时间后，重新配置 I2C 寄存器，再次发送传输请求。因为上一次没有收到有效数据，DMA 并没有任何动作，所以不用重置 DMA。

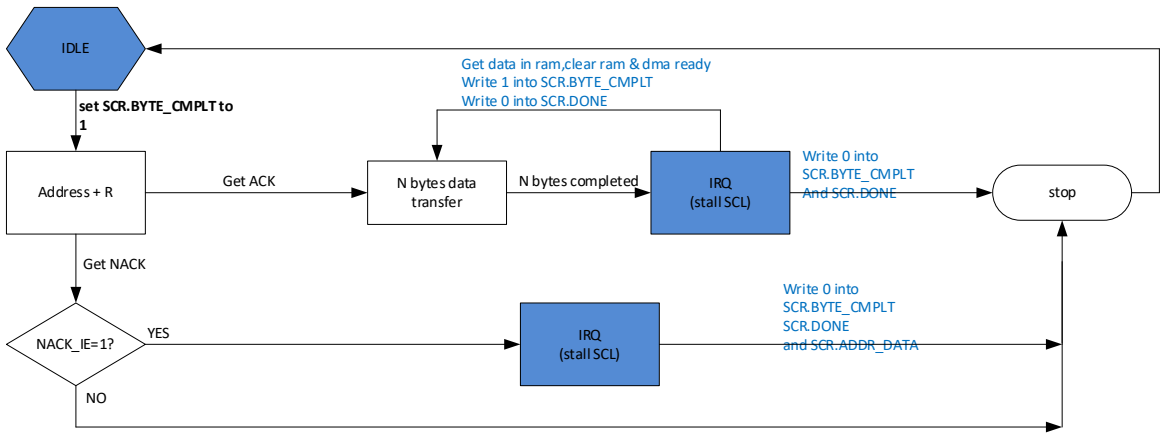


图 9-8 主模式下多字节接收示意图

9.3.2.5 DMA 传输

在大容量数据传输应用下，I2C 接口支持 DMA 传输，减轻 MCU 的负担。一次传输，最大传输量为 32 字节，最小传输量为 1 字节。因 I2C 无 FIFO，I2C 发送一次 DMA 请求，DMA 只能搬移一个字节数据。若要实现多字节搬移，DMA 需配置为多轮搬移，每一轮搬移一个字节的方式。

在接收到新的数据后，硬件自动产生 DMA 请求，通过 DMA 模块将数据搬移到 RAM 中。在发送新数据前，硬件自动产生 DMA 请求，通过 DMA 模块将数据从 RAM 中搬移到 I2C 接口。

DMA 传输需要配置 DMA 模块相应寄存器。

因 I2C 接口支持 DMA 传输，也支持 MCU 传输。两者区别在于 DMA 传输，发送的数据来自 DMA 的搬移；MCU 传输，发送的数据来自 MCU 的搬移。

DMA 传输，推荐软件配置流程如下：

- 初始化 DMA 模块，将本次发送的数据来源，接收的数据去向配置好，传输长度配置完毕。
- 初始化 GPIO 模块，将 I2C 复用的 GPIO 配置完毕。
- 初始化 I2C 接口，I2C_CFG/I2C_BCR/I2C_BSIZE 等寄存器配置完毕。
- 主模式下，触发 I2C 接口，进入发送状态；从模式下，等待主发送传输请求。

若 I2C 接口为从发送模式，需要考虑预取，I2C_BCR.SLV_DMA_PREF 为预取开关。一般从模式传输的时候，可以开启硬件地址比较(I2C_ADDR. ADDR_CMP)，可选择是否开启硬件地址比较中断使能(I2C_BCR.BURST_ADDR_CMP)。

- 开启硬件地址比较中断，从设备收到的地址同自身匹配成功后，会产生中断。通知软件此时主设备请求获得从设备数据，软件判断是否接收，若接收就回 ACK，软件需要 I2C_BCR. SLV_DMA_PREF 置 1，协助硬件预取第一次传输的数据；否则就回 NACK。

关闭硬件地址比较中断，一旦总线上出现 START 事件，从设备硬件预取第一次传输数据，无论从设备是否地址匹配成功。匹配成功，也不会产生地址匹配中断，直接开始数据传输。匹配不成功，从设备不会传输数据。此时，若匹配不成功，因为 I2C 有预取的操作，为后续匹配成功后传输正常，需要对 DMA 进行清除操作

9.3.2.6 I2C 总线异常处理

在一个地址或数据字节传输期间，当 I2C 接口检测到一个外部的停止或起始条件则产生总线错误。一般而言，产生总线错误是由于总线上有干扰、某些 I2C 设备没有同步于本 I2C 网络自行发送了 START 事件/STOP 事件。根据 I2C 协议规定，发生总线错误的时候，在收到 START 事件/STOP 事件后要重置本 I2C 设备的接口逻辑。对于从设备而言，这个操作是没有问题的；对于主设备而言，总线错误强行要求其释放总线并重置其 I2C 接口逻辑。因为主设备是不响应外部 START 和 STOP 事件的，发生总线错误后，需要中断处理函数处理本次异常，并指导主设备继续监视总线情况，以便后续执行 I2C 总线传输。

本 I2C 接口。主模式下，总线错误可被检测到同时总线错误中断也会产生；从模式下，总线错误将触发地址数据被接收，同时让 I2C 接口恢复空闲状态并产生中断。

9.3.2.7 中断处理

I2C 接口包含三种类型的中断事件，分别是：数据传输完成事件，总线错误事件、STOP 事件、NACK 事件和硬件地址匹配事件。

- 数据完成事件。当前数据传输完成，高电平有效，对 I2C_SCR.Done 写 0 清除。
- 总线错误事件。传输过程中，总线产生错误的 START 事件/STOP 事件，高电平有效，对 I2C_SCR.STT_ERR 写 0 清除。
- STOP 事件。当前数据传输完成，主设备发送 STOP 事件，从设备收到 STOP 事件并产生相应中断。高电平有效，对 I2C_SCR.STOP_EVT 写 0 清除。
- NACK 事件。发送端接收到 NACK 响应，表明接收端无法继续后续传输。高电平有效，对 I2C_SCR.RX_ACK 写 0 清除。
- 硬件地址匹配事件。从模式下接收到的地址同本设备地址匹配，产生相应中断。高电平有效，对 I2C_SCR.ADDR_DATA 写 0 清除。

9.3.2.8 通讯速度设置

I2C 接口的工作时钟来自系统时钟的分频，分频寄存器为 SYS 模块的 CLK_DIV0。

I2C 接口采用同步设计，需要对外部设备的信号进行同步采样，同步时钟为 I2C 接口工作时钟。

- I2C 模块工作时钟频率 = 系统频率 / (CLK_DIV0 + 1)
- I2C 模块数据信号(SDA)和时钟信号(SCL)的时钟频率= I2C 模块工作时钟频率 / 17。
- I2C 波特率 = I2C 模块工作时钟频率 / 17

9.4 寄存器

9.4.1 地址分配

I2C 模块寄存器的基地址是 0x4001_0100，寄存器列表如下:

表 9-1 I2C 寄存器地址分配表

名称	偏移	说明
I2C0_ADDR	0x00	I2C 地址寄存器
I2C0_CFG	0x04	I2C 配置寄存器
I2C0_SCR	0x08	I2C 状态寄存器
I2C0_DATA	0x0C	I2C 数据寄存器
I2C0_MSCR	0x10	I2C 主模式寄存器
I2C0_BCR	0x14	I2C 传输控制寄存器
I2C0_BSIZE	0x18	I2C 传输长度寄存器

9.4.2 I2C0_ADDR 地址寄存器

地址:0x4001_0100

复位值:0x0

表 9-2 地址寄存器 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						

位置	位名称	说明
[31:8]		未使用
[7]	ADDR_CMP	I2C 硬件地址比较使能开关，默认值为 0。 0:关闭 1:开启
[6:0]	ADDR	仅用于从模式下，I2C 设备硬件地址。主模式下，从设备地址写入 I2C_DATA 寄存器。

9.4.3 I2C0_CFG 系统控制寄存器

地址:0x4001_0104

复位值:0x0

表 9-3 系统控制寄存器 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
								RW	RW	RW	RW			RW	RW
								0	0	0	0			0	0

位置	位名称	说明
[31:8]		未使用
[7]	IE	I2C 中断使能信号。默认值为 0。 1:使能 I2C 中断 0:关闭 I2C 中断
[6]	TC_IE	I2C 数据传输完成中断使能信号。默认值为 0。 1:使能此中断源

		0:屏蔽此中断源
[5]	BUS_ERR_IE	I2C 总线错误事件中断使能信号。默认值为 0。 1:使能此中断源 0:屏蔽此中断源
[4]	STOP_IE	I2CSTOP 事件中断使能信号。默认值为 0。 1:使能此中断源 0:屏蔽此中断源
[3:2]		NA
[1]	MST_MODE	I2C 主模式使能信号。默认值为 0。 1:使能主模式 0:关闭主模式
[0]	SLV_MODE	I2C 从模式使能信号。默认值为 0。 1:使能从模式 0:关闭从模式

9.4.4 I2C0_SCR 状态控制寄存器

地址:0x4001_0108

复位值:0x0

表 9-4 状态控制寄存器 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

位置	位名称	说明
[31:8]		未使用
[7]	STT_ERR	总线错误状态标志位，用于主模式发送/主模式接收，写 0 清除。 0:无 START/STOP 总线错误 1:有 START/STOP 总线错误
[6]	LOST_ARB	总线仲裁丢失状态标志位，用于主模式发送/主模式接收，发生总线仲裁丢失事件将此位置 1，无中断事件产生，在字节完成中断中需查此位。 总线上任何 START 事件将导致硬件清除此位。 0:无总线仲裁丢失错误发生 1:有总线仲裁丢失错误发生
[5]	STOP_EVT	STOP 事件状态标志位，用于主模式发送/从模式发送/主模式接收/从模式接收。写 0 清除。

		0:无 STOP 事件 1:有 STOP 事件
[4]	BYTE_CMPLT	ACK 控制位，用于主模式接收/从模式接收。发送方发送完毕当前字节，接收方对此的响应。若是发送方，此位保留 0 值。接收方，根据实际情况配置。 0:字节发送完成，返回 NACK 回应，表示接收方不能接收更多数据 1:字节发送完成，返回 ACK 回应，表示接收方可以继续接收数据
[3]	ADDR_DATA	地址数据标志位，用于主模式发送/从模式发送/主模式接收/从模式接收。START 后，第一个字节为地址数据，此位是一个提示位。写 0 清除。 0:当前传输的数据非地址数据。 1:当前传输的数据是地址数据。
[2]	DATA_DIR	发送或接收控制位，主模式发送/从模式发送，此位置 1，触发发送，硬件自动清零；主模式接收/从模式接收，此位置 0，等待接收。 0:接收 1:触发发送
[1]	RX_ACK	接收响应标志位，用于主模式发送/从模式发送，告知发送方，接收方的反馈。发送方收到反馈后，对该位执行清零操作。 0:本 I2C 接口发送数据，接收到 ACK 响应。 1:本 I2C 接口发送数据，接收到 NACK 响应。
[0]	Done	传输完成状态标志位，用于主模式发送/从模式发送/主模式接收/从模式接收。写 0 清除。 0:传输未完成 1:传输已完成

一般，进入中断后，需读取 I2C_SCR 寄存器，获得当前 I2C 总线状态及当前传输处于什么阶段；然后，对 I2C_SCR 进行写操作，写入不同的值，软件通知硬件下一步如何处理。

9.4.5 I2C0_DATA 数据寄存器

地址:0x4001_010C

复位值:0x0

表 9-5 数据寄存器 I2C0_DATA

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

位置	位名称	说明
[31:8]		未使用
[7:0]	DATA	数据寄存器，用于主模式发送/从模式发送/主模式接收/从模式接收。发

		送方，写入发送数据；接收方，读取接收数据。注意，地址数据也是数据，主模式只能将要发送地址数据写入此寄存器。

9.4.6 I2C0_MSCR 主模式寄存器

地址:0x4001_0110

复位值:0x0

表 9-6 主模式寄存器 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

位置	位名称	说明
[31:4]		未使用
[3]	BUSY	I2C 总线，闲忙状态。 0:检测到 STOP 事件，空闲。 1:检测到 START 事件，忙碌。
[2]	MST_CHECK	主模式争抢总线标志位。争抢到总线，置 1；STOP 事件或者发生总线冲突本模块释放总线，置 0。
[1]	RESTART	再次触发 START 事件，写 1 有效。发送 START 完毕，硬件清 0。I2C_CFG[1]置 1，才能实现写 1 操作。
[0]	START	触发 START 事件并发送地址数据至总线，写 1 有效。I2C_CFG[1]置 1，才能实现写 1 操作。

9.4.7 I2C0_BCR I2C 传输控制寄存器

地址:0x4001_0114

复位值:0x0

表 9-7 DMA 传输控制寄存器 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	SLV_DMA_PREF				
								RW	RW	RW	RW				
								0	0	0	0				

位置	位名称	说明
[31:8]		未使用
[7]	BURST_NACK	I2C 传输，NACK 事件中断使能信号。 1：使能此中断源 0：屏蔽此中断源
[6]	BURST_ADDR_CMP	I2C 传输，硬件地址匹配中断使能信号。 1：使能此中断源 0：屏蔽此中断源
[5]	BUSRT_EN	I2C 多数据传输使能，需要采用 DMA 方式。 1：使能 0：关闭
[4]	SLV_DMA_PREF	I2C 多数据传输。从模式执行 DMA 方式发送，触发硬件预取第一个字节。硬件自动清零。 1：使能 0：关闭
[3:0]		未使用

10 CMP

10.1 概述

比较器信号处理模块(以下简称CMP模块，为便于区分后续图示中模拟比较器使用Comparator表示，数字比较器信号处理模块使用 CMP 表示)，用于处理 3 个模拟轨到轨比较器产生的输出信号，由一系列使能、极性控制、滤波等数字电路组成。信号处理时钟由系统主时钟分频得到，此模块也用于向 CPU 产生比较器中断。

CMP0/1 可以使用 MCPWM 的 4 路 P 通道进行开窗控制。

模拟比较器未经滤波的原始输出值，可以通过读取 CMP_DATA 值获得。同时模拟比较器未经滤波的原始输出值也可以通过配置 GPIO 的 AF1 功能送出，具体 GPIO 功能配置及引进位置，请参考器件 datasheet。

关于模拟比较器的更多说明，包括其输入端信号的选择，迟滞的配置，请参考章节 5.1.6。

10.2 寄存器

10.2.1 地址分配

CMP 模块寄存器的基地址是 0x4001_0200，寄存器列表如下:

表 10-1 比较器寄存器列表

名称	偏移地址	说明
CMP_IE	0x00	比较器中断使能寄存器
CMP_IF	0x04	比较器中断标志寄存器
CMP_TCLK	0x08	比较器分频时钟控制寄存器
CMP_CFG	0x0C	比较器控制寄存器
CMP_BLCWIN	0x10	比较器开窗控制寄存器 0
CMP_DATA	0x14	比较器输出数值寄存器

10.2.2 CMP_IE 中断使能寄存器

地址:0x4001_0200

复位值:0x0

表 10-2 CMP_IE 比较器中断使能寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					CMP2_RE	CMP1_RE	CMP0_RE						CMP2_IE	CMP1_IE	CMP0_IE
					RW	RW	RW						RW	RW	RW

	0	0	0		0	0	0

位置	位名称	说明
[31:11]		未使用
[10]	CMP2_RE	比较器 2 DMA 请求使能，高有效
[9]	CMP1_RE	比较器 1 DMA 请求使能，高有效
[8]	CMP0_RE	比较器 0 DMA 请求使能，高有效
[7:3]		未使用
[2]	CMP2_IE	比较器 2 中断使能，高有效
[1]	CMP1_IE	比较器 1 中断使能，高有效
[0]	CMP0_IE	比较器 0 中断使能，高有效

10.2.3 CMP_IF 中断标志寄存器

地址:0x4001_0204

复位值:0x0

表 10-3 CMP_IF 比较器中断标志寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													CMP2_IF	CMP1_IF	CMP0_IF
													RW	RW	RW
													0	0	0

位置	位名称	说明
[31:2]		未使用
[2]	CMP2_IF	比较器 2 中断标志，高有效，写 1 清零
[1]	CMP1_IF	比较器 1 中断标志，高有效，写 1 清零
[0]	CMP0_IF	比较器 0 中断标志，高有效，写 1 清零

当使能 CMP DMA 请求时，即 CMP_IE.CMPx_RE=1，在 DMA 收到相应请求并开始进行数据搬移时，DMA 自动将对应的 CMPx_IF 位清除，不再需要软件清除。

10.2.4 CMP_TCLK 分频时钟控制寄存器

地址:0x4001_0208

复位值:0x0

表 10-4 CMP_TCLK 比较器分频时钟控制寄存器

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
					COMM_DIV			FIL2_CLK_DIV16				CLK2_EN	FIL2_CLK_DIV2		

	RW	RW	RW	RW
	3'h4	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIL1_CLK_DIV16				CLK1_EN	FIL1_CLK_DIV2			FIL0_CLK_DIV16				CLK0_EN	FIL0_CLK_DIV2		
RW				RW	RW			RW				RW	RW		
0				0	0			0				0	0		

位置	位名称	说明
[31:27]		未使用
[26:24]	COMM_DIV	比较器 0/1/2 共用滤波时钟分频系数，影响进入比较器中断的时间，最大支持到 4
[23:20]	FIL2_CLK_DIV16	比较器 2 滤波时钟分频，基于 MCLK 进行 1~16 分频，影响进入比较器中断的时间
[19]	CLK2_EN	比较器 2 滤波时钟使能，高有效
[18:16]	FIL2_CLK_DIV2	比较器 2 滤波时钟分频 0:1 分频 1:2 分频 2:4 分频 3:8 分频 4:16 分频 5:32 分频, 6:64 分频 7:128 分频
[15:12]	FIL1_CLK_DIV16	比较器 1 滤波时钟分频，基于 MCLK 进行 1~16 分频，影响进入比较器中断的时间
[11]	CLK1_EN	比较器 1 滤波时钟使能，高有效
[10:8]	FIL1_CLK_DIV2	比较器 1 滤波时钟分频 0:1 分频 1:2 分频 2:4 分频 3:8 分频 4:16 分频 5:32 分频, 6:64 分频 7:128 分频
[7:4]	FIL0_CLK_DIV16	比较器 0 滤波时钟分频，基于 MCLK 进行 1~16 分频，影响进入比较器中断的时间
[3]	CLK0_EN	比较器 0 滤波时钟使能，高有效
[2:0]	FIL0_CLK_DIV2	比较器 0 滤波时钟分频 0:1 分频 1:2 分频

		2:4 分频 3:8 分频 4:16 分频 5:32 分频, 6:64 分频 7:128 分频

以比较器 0 为例，CMP0 滤波时间宽度使用如下公式进行计算

滤波时间宽度= Period(FCLK)*(2$^{(COMM_DIV+CMP_TCLK.FIL0_CLK_DIV2)}$*(CMP_TCLK.FIL0_CLK_DIV16+1)-1)，FCLK 即芯片主时钟，作为 CMP 模块的时钟，受 SYS_CLK_FEN 的门控控制。需要注意的是，产生 CMP 滤波时钟需要使能 CMP_TCLK.CLK_EN 位。

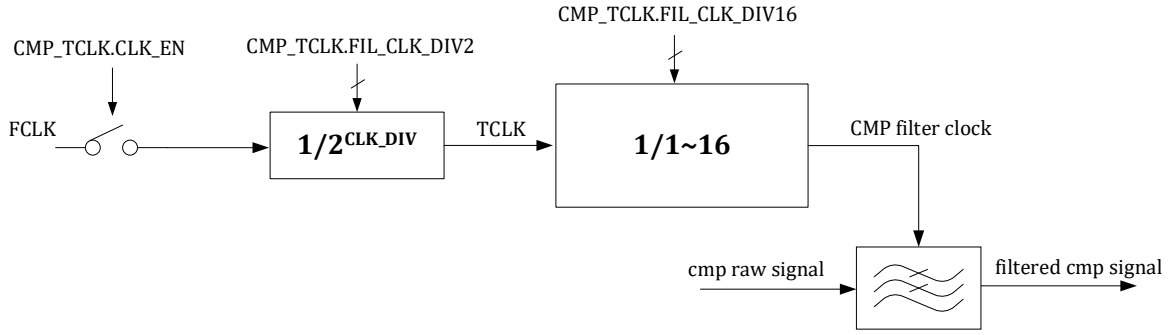


图 10-1 比较器滤波时钟产生

CMP 模块使用此滤波时钟对模拟比较器的输出信号进行滤波，即只有信号稳定时间超过设定的滤波时间宽度才能通过滤波器，CMP 模块输出的滤波后的信号才会发生变化，并产生相应中断。如果输入信号稳定时间不足滤波时间宽度即发生变化，则 CMP 模块输出的滤波后的信号维持原值不变。

因为滤波时钟可以进行分频，因此 CMP 信号的滤波宽度范围是 0~128*16*16-1 个总线周期，即 1~32767 个系统主时钟周期。

10.2.5 CMP_CFG 控制寄存器

地址:0x4001_020C

复位值:0x0

表 10-5 CMP_CFG 比较器控制寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					CMP2_IRQ_TRIG	CMP2_IN_EN	CMP2_POL	CMP1_W_PWM_POL	CMP1_IRQ_TRIG	CMP1_IN_EN	CMP1_POL	CMP0_W_PWM_POL	CMP0_IRQ_TRIG	CMP0_IN_EN	CMP0_POL
					RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
					0	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明

[31:11]		未使用
[10]	CMP2_IRQ_TRIG	比较器 2 中断触发类型，0:电平触发，1:边沿触发
[9]	CMP2_IN_EN	比较器 2 信号输入使能
[8]	CMP2_POL	比较器 2 极性选择，0:高电平有效；1:低电平有效
[7]	CMP1_W_PWM_POL	比较器 1 开窗 PWM 信号极性选择，在 CMP_BLCWIN1 使能情况下使用
[6]	CMP1_IRQ_TRIG	比较器 1 中断触发类型，0:电平触发，1:边沿触发
[5]	CMP1_IN_EN	比较器 1 信号输入使能
[4]	CMP1_POL	比较器 1 极性选择，0:高电平有效；1:低电平有效
[3]	CMP0_W_PWM_POL	比较器 0 开窗 PWM 信号极性选择，在 CMP_BLCWIN0 使能情况下使用
[2]	CMP0_IRQ_TRIG	比较器 0 中断触发类型，0:电平触发，1:边沿触发
[1]	CMP0_IN_EN	比较器 0 信号输入使能
[0]	CMP0_POL	比较器 0 极性选择，0:高电平有效；1:低电平有效

比较器的极性及使能控制如图 10-2 所示。

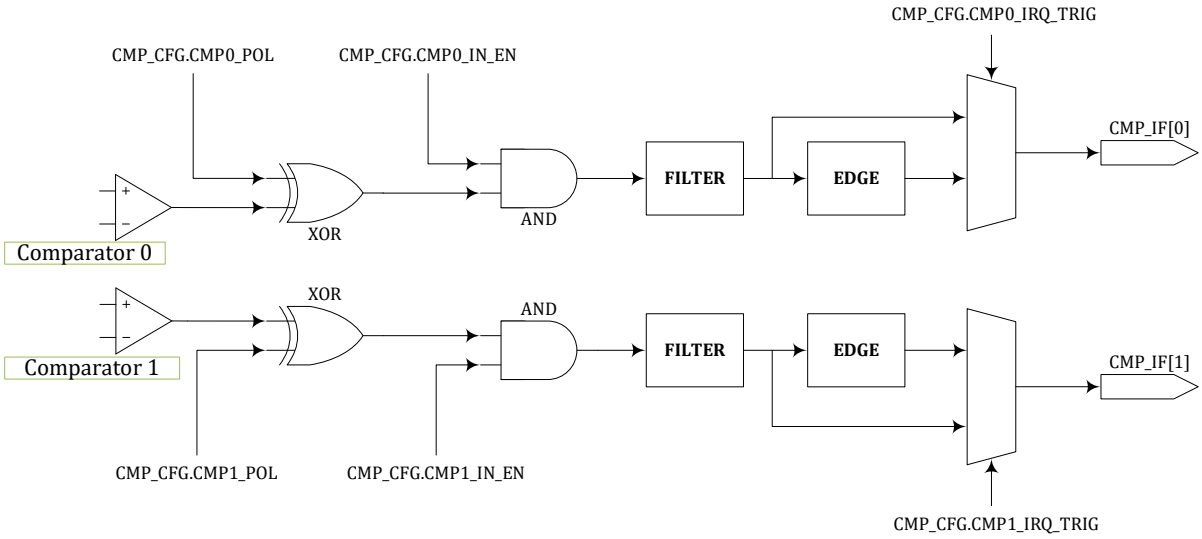


图 10-2 比较器控制及中断产生逻辑

比较器 0 和比较器 1 可以与 MCPWM 模块联合动作，其中 MCPWM 模块的 P 管控制信号可以作为比较器开窗的控制信号。但比较器自身的中断信号产生与开窗控制无关，仅仅受 CMP_CFG 寄存器影响。比较器 2 不支持开窗控制。

MCPWM 的 fail 信号可以来自 GPIO，也可以来自比较器模块，使用 MCPWM0_FAIL 寄存器进行控制。如果 MCPWM 的 fail 信号来自比较器，则是经过比较器模块内部的开窗控制的。fail 信号进入 MCPWM 后也会进行极性使能以及滤波等处理，与比较器模块的实现类似，但完全独立，由 MCPWM 内部的寄存器进行控制。MCPWM 内部与 fail 相关的错误中断信号产生收到 MCPWM 内部有关极性使能滤波控制寄存器的影响。具体请参考 MCPWM 章节。

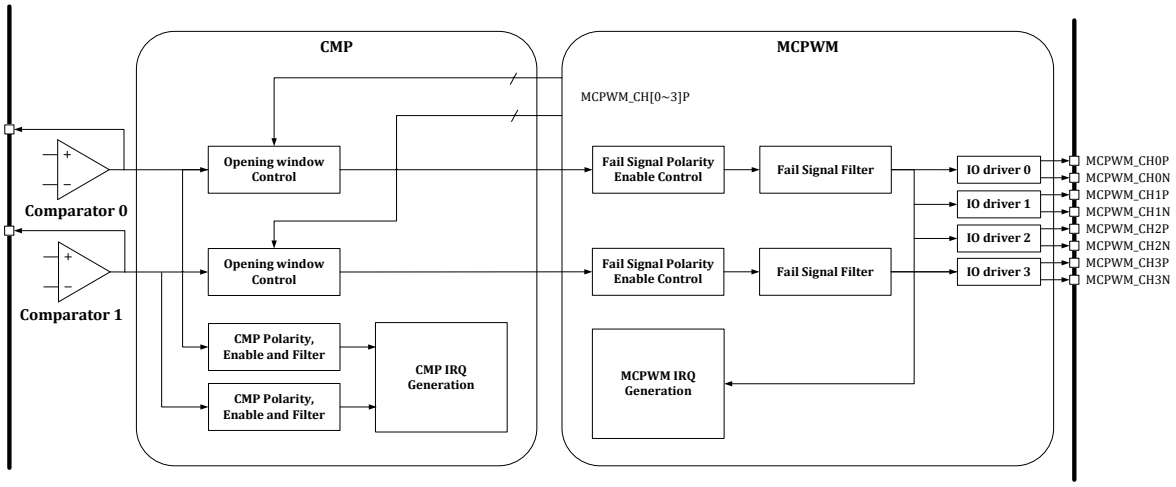


图 10-3 CMP 与 MCPWM 的联动

对于比较器的开窗功能，若 CMP_CFG.CMP0_PWM_POL=1，则在对应 MCPWM CHNx_P 信号为 1 时，比较器 0 可以产生比较信号输出，其他时刻比较信号为 0；反之，若 CMP_CFG.CMP0_PWM_POL=0，则在对应 MCPWM CHNx_P 信号为 0 时，比较器 0 可以产生比较信号输出，其他时刻比较信号为 0。比较器 1 的开窗控制信号极性由 CMP_CFG.CMP1_PWM_POL 位进行控制，逻辑相同。

注意:CMP_CFG.CMP0_PWM_POL 和 CMP_CFG.CMP1_PWM_POL 同时会影响送入 MCPWM 模块作为 FAIL 信号的比较器信号，如图 10-4 所示。来自比较器的 MCPWM FAIL 信号为模拟比较器输出的原始信号，未经过比较器数字接口模块的滤波处理，但是可以被 MCPWM 的通道信号进行开窗控制，开窗控制设置见比较器数字接口模块。FAIL 信号进入 MCPWM 模块后，可以通过设置 MCPWM_TCLK 进行滤波。

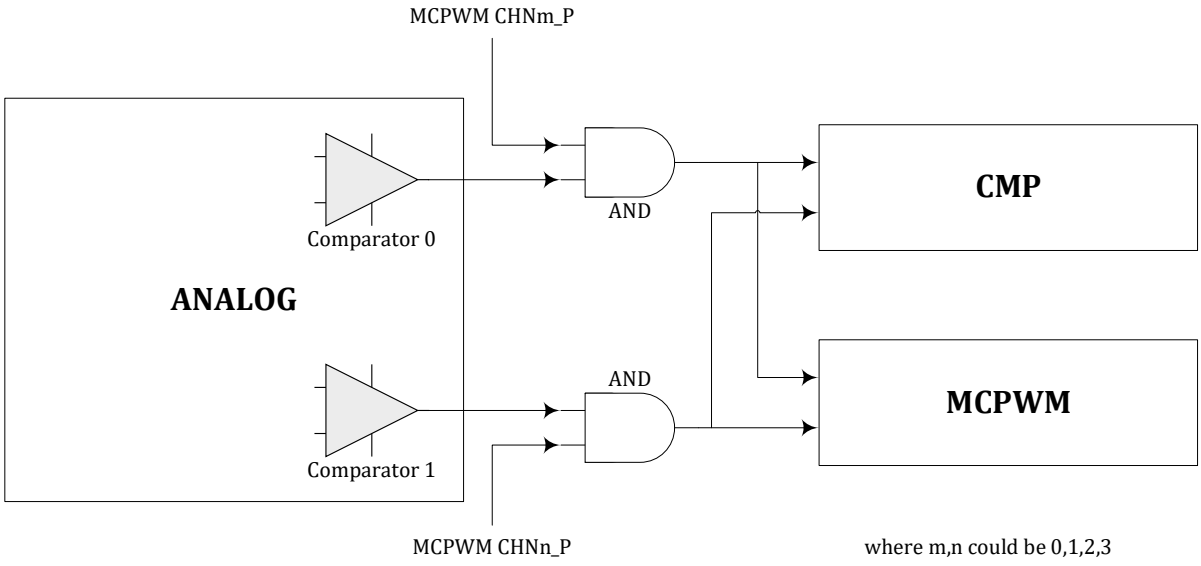


图 10-4 比较器开窗功能图示

10.2.6 CMP_BLCWIN 开窗控制寄存器

地址:0x4001_0210

复位值:0x0

表 10-6 CMP_ BLCWIN 比较器开窗控制寄存器

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

位置	位名称	说明
[31:8]		保留
[7]	CMP1_CHN3P_WIN_EN	使用 MCPWM 模块 CHN3_P 通道输出的 P 管开关控制信号作为比较器 1 开窗使能
[6]	CMP1_CHN2P_WIN_EN	使用 MCPWM 模块 CHN2_P 通道输出的 P 管开关控制信号作为比较器 1 开窗使能
[5]	CMP1_CHN1P_WIN_EN	使用 MCPWM 模块 CHN1_P 通道输出的 P 管开关控制信号作为比较器 1 开窗使能
[4]	CMP1_CHN0P_WIN_EN	使用 MCPWM 模块 CHN0_P 通道输出的 P 管开关控制信号作为比较器 1 开窗使能
[3]	CMP0_CHN3P_WIN_EN	使用 MCPWM 模块 CHN3_P 通道输出的 P 管开关控制信号作为比较器 0 开窗使能
[2]	CMP0_CHN2P_WIN_EN	使用 MCPWM 模块 CHN2_P 通道输出的 P 管开关控制信号作为比较器 0 开窗使能
[1]	CMP0_CHN1P_WIN_EN	使用 MCPWM 模块 CHN1_P 通道输出的 P 管开关控制信号作为比较器 0 开窗使能
[0]	CMP0_CHN0P_WIN_EN	使用 MCPWM 模块 CHN0_P 通道输出的 P 管开关控制信号作为比较器 0 开窗使能

10.2.7 CMP_DATA 输出数据寄存器

地址:0x4001_0214

复位值:0x0

表 10-7 CMP_ DATA 比较器输出数据寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					CMP2_FLT_DATA	CMP1_FLT_DATA	CMP0_FLT_DATA						CMP2_RAW_DATA	CMP1_RAW_DATA	CMP0_RAW_DATA
					R	R	R						R	R	R

	0	0	0		0	0	0

位置	位名称	说明
[31:11]		未使用
[10]	CMP2_FLT_DATA	比较器 2 经过滤波后的信号
[9]	CMP1_FLT_DATA	比较器 1 经过滤波后的信号
[8]	CMP0_FLT_DATA	比较器 0 经过滤波后的信号
[7:3]		未使用
[2]	CMP2_RAW_DATA	比较器 2 原始输出信号，直接来自模拟比较器 2
[1]	CMP1_RAW_DATA	比较器 1 原始输出信号，直接来自模拟比较器 1
[0]	CMP0_RAW_DATA	比较器 0 原始输出信号，直接来自模拟比较器 0

11 HALL

11.1 综述

芯片共支持 3 路 HALL 信号输入。

对于输入的 HALL 传感器信号，所进行的处理包括:

滤波，消除 HALL 信号毛刺的影响

捕获，HALL 输入有变化时，记录当前的定时器值，并输出中断

溢出，HALL 信号长时间不发生变化导致计数器溢出，并输出中断

11.2 实现说明

11.2.1 信号来源

HALL 信号来源于 GPIO，对于每一路 HALL 信号，芯片有两个 IO 可以作为该信号的来源。通过配置 GPIO 寄存器，用户可以选择将其中一个 GPIO 的输入信号做为 HALL 信号使用。

详细管脚位置说明见芯片器件 datasheet。

11.2.2 工作时钟

HALL 模块工作频率可调。通过配置 HALL_CFG.CLK_DIV 寄存器，可以选择系统主时钟的 1/2/4/8 分频作为 HALL 模块工作频率，滤波和计数均采用该频率工作。

11.2.3 信号滤波

滤波模块主要用于去除 HALL 信号上的毛刺。

滤波包括两级滤波器，两级滤波电路可单独开启，也可全部开启:

第一级采用 7 判 5 进行滤波，即连续 7 个采样点中，如果达到或超过 5 个 1 则输出 1，如果达到或超过 5 个 0 则输出 0，否则输出保持上一次的滤波结果。通过配置 HALL_CFG.FIL_75 可以选择是否使能第一级滤波器。具体如下图所示:

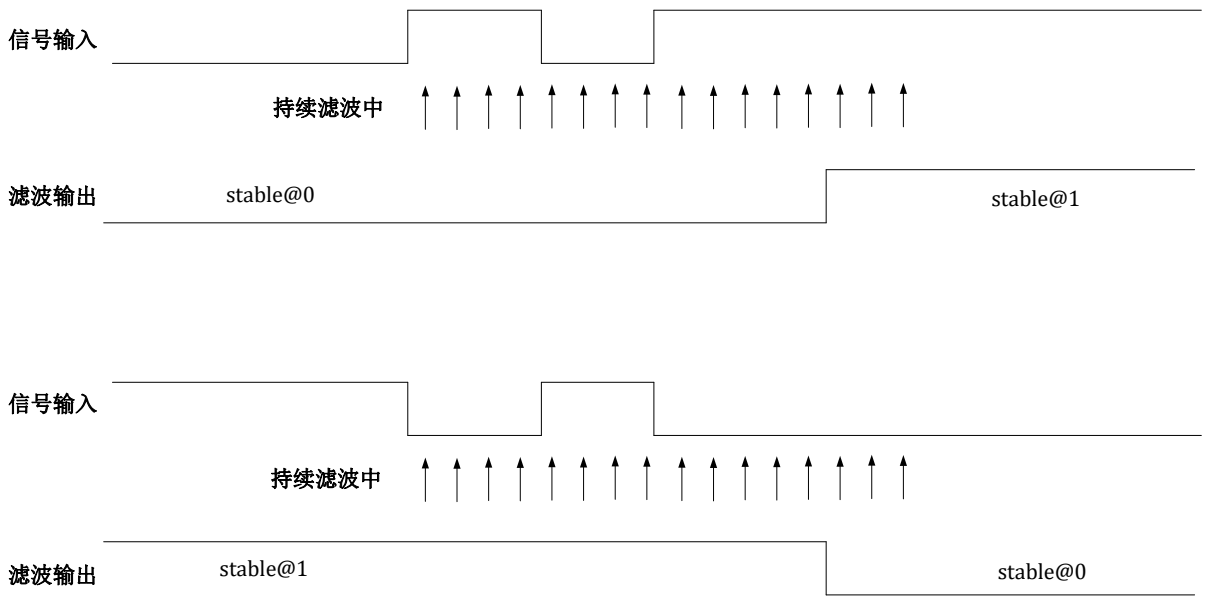


图 11-1 7/5 滤波模块框图

第二级采用连续滤波，在连续 N 个采样点中，如全为 0 则输出 0，如全为 1 则输出 1，否则输出保持上一次的滤波结果。

通过配置 HALL_CFG.FIL_LEN 可以配置第二级滤波器滤波深度，即连续采样个数。连续采样个数最大为 2^{15}，滤波时间常数计算公式如下:

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

举例，在 96MHz 工作频率下，周期 T_{clk} 是 10.42ns，寄存器配置最大为 32767，最长滤波宽度为约 10.42ns×32768≈340us。

通过访问 HALL_INFO.FIL_DATA[2:0]可以捕捉滤波后的 HALL 信号；HALL_INFO.RAW_DATA[2:0]则是滤波前原始 HALL 输入信号，详见 11.3.3。

11.2.4 捕获

捕获模块用于测量两次 HALL 信号变化之间的时间，其核心为一个 24 位计数器，在 96MHz 工作频率下，如果通过 HALL_CFG.CLK_DIV=3 设置 Hall 时钟 8 分频，则最大可以记录约 2^24*8/96e6=1. 4s 的时间宽度，达到 10.42ns 的时间分辨率。

HALL_CNT 从 0 开始计数，当发生 HALL 信号变化时，将此时刻的 HALL_CNT 值保存到 HALL_WIDTH 寄存器，将此时刻的 HALL 信号保存到 HALL_INFO.FIL_DATA，输出 HALL 信号变化中断，HALL_CNT 重新从 0 开始计数。

当计数器计数值达到 HALL_TH 时，输出 HALL 计数器溢出中断，计数器重新从 0 开始计数。

11.2.5 中断

捕获、溢出事件触发中断，中断使能控制位位于 HALL_CFG.CHG_IE 和 HALL_CFG.OV_IE，中断标志位位于 HALL_INFO.CHG_IF 和 HALL_INFO.OV_IF。终端标志可以通过对 HALL_INFO.CHG_IF 和 HALL_INFO.OV_IF 写 1 清空。

HALL 中断事件可以作为 DMA 请求，但需要在 HALL_CFG 中使能 CHG_RE，OV_RE 或 SW_RE。当 DMA 收到传输请求开始数据搬运后，会将中断标志清除，不再需要软件清除。

11.2.6 数据流程

HALL 模块的数据流程如下图所示，FCLK 为受 SYS_CLK_FEN.HALL_CLK_EN 门控信号控制的 HALL 时钟，开通后为 96MHz 的 PLL 时钟。

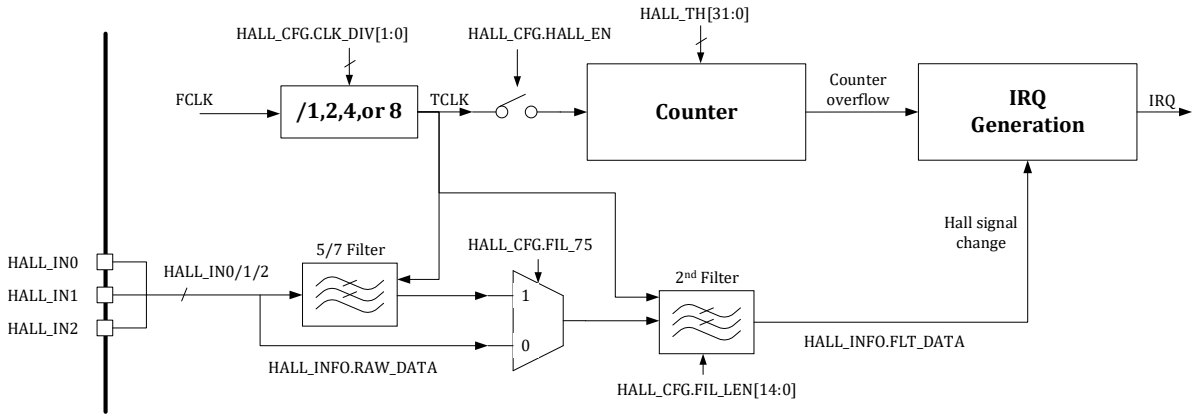


图 11-2 数据流程框图

11.3 寄存器

11.3.1 地址分配

HALL 模块寄存器的基地址是 0x4001_0300，寄存器列表如下:

表 11-1 HALL 模块寄存器地址分配

名称	偏移	描述
HALL0_CFG	0x00	HALL 模块配置寄存器
HALL0_INFO	0x04	HALL 模块信息寄存器
HALL0_WIDTH	0x08	HALL 宽度计数值寄存器
HALL0_TH	0x0C	HALL 模块计数器门限值寄存器
HALL0_CNT	0x10	HALL 计数寄存器

11.3.2 HALL0_CFG HALL 模块配置寄存器

地址:0x4001_0300

复位值:0x0

表 11-2 HALL0_CFG HALL 模块配置寄存器

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

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

位置	位名称	说明
[31]		未使用
[30]	SW_IE	软件触发 HALL 信号变化中断使能，高电平有效。当 SW_IE=1 时，软件向 HALL_INFO.SW_IF 写 1，可以手动产生 HALL 信号变化中断，并令 HALL_INFO.CHG_IF 置位。
[29]	OV_IE	HALL 计数器溢出中断使能开关。默认关闭。 0: 关闭 1: 使能
[28]	CHG_IE	HALL 信号变化中断使能开关。默认关闭。 0: 关闭 1: 使能
[27]		保留
[26]	OV_RE	HALL 计数器溢出 DMA 请求使能开关。默认关闭。 0: 关闭 1: 使能
[25]	CHG_RE	HALL 信号变化 DMA 请求使能开关。默认关闭。 0: 关闭 1: 使能
[24]	HALL_EN	HALL 模块使能开关。默认关闭。 0: 关闭 1: 使能
[23:21]		未使用
[20]	FIL_75	7/5 滤波开关（连续采样 7 次，5 次值一致）。默认关闭。 0: 关闭 1: 使能
[19:18]		未使用
[17:16]	CLK_DIV	HALL 时钟分频系数。默认不分频。 00:不分频 01:2 分频 10:4 分频 11:8 分频
[15]		未使用
[14:0]	FIL_LEN	滤波宽度，低于对应脉冲宽度的信号将被硬件自动过滤掉。滤波宽度的计算公式为[14:0] + 1。

11.3.3 HALL0_INFO HALL 模块信息寄存器

地址:0x4001_0304

复位值:0x0

表 11-3 HALL0_INFO HALL 模块信息寄存器

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

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

位置	位名称	说明
[31:19]		未使用
[18]	SW_IF	软件触发 HALL 信号变化中断。写 1 触发，自动清零。
[17]	OV_IF	HALL 计数器溢出事件标志，写 1 清空
[16]	CHG_IF	HALL 信号变化事件标志，写 1 清空
[15:11]		系统保留，必须写入 0，读出 0
[10:8]	RAW_DATA	HALL 值，未滤波结果
[7:3]		系统保留，必须写入 0，读出 0
[2:0]	FLT_DATA	HALL 值，滤波结果

11.3.4 HALL0_WIDTH HALL 宽度计数值寄存器

地址:0x4001_0310

复位值:0x0

表 11-4 HALL0_WIDTH HALL 宽度计数值寄存器

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															

位置	位名称	说明
[31:24]		未使用
[23:0]	CAP_CNT	HALL 宽度计数器值

11.3.5 HALL0_TH HALL 模块计数器门限值寄存器

地址:0x4001_030C

复位值:0x0

表 11-5 HALL0_TH HALL 模块计数器门限值寄存器

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															

位置	位名称	说明
[31:24]		未使用
[23:0]	TH	HALL 计数器门限值

11.3.6 HALL0_CNT HALL 计数寄存器

地址:0x4001_0308

复位值:0x0

表 11-6 HALL0_CNT HALL 计数寄存器

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															

位置	位名称	说明
[31:24]		未使用
[23:0]	CNT	HALL 计数值，写入任意值可清零

12 ADC

12.1 概述

LKS32MC07x 系列芯片集成了 2 个 12bit SAR-ADC 转换核心，14 个模拟 IO 输入信号/4 个 OPA 输出/2 个 DAC 输出/温度传感器输出/内部电源 AVDD 均可作为 ADC 的被采样信号。主要有以下特性：

- 2 个 ADC 数字接口分别对应 2 个 ADC
- 3Msps 采样率，48MHz 工作频率
- 每个 ADC 接口支持 16 路通道选择
- 每个 ADC 接口配备 1 个模拟看门狗，可同时检测上下阈值
- 支持过采样
- 支持软件、硬件触发功能
- 可以与 MCPWM/Timer/CL 单元联动，触发指示信号可通过 GPIO 送出调试
- 支持自定义采样序列，序列次数和通道号可灵活配置
- 支持左对齐、右对齐模式
- 支持差分输入模拟信号
- 支持单段采样和两段采样
- 支持硬件过采样，过采样后有效位数可达 16bit 以上

ADC 采样的量词含义约定：

1 次采样：完成对应的一次模拟信号量到数字信号量的采样转换并存储至 ADC_DATx 寄存器；

1 段采样：可能包含 1 次或若干次采样，若干次采样可以是相同的模拟信号，也可以是不同的模拟信号。采样开始通常由 MCPWM、TIMER 或软件进行触发，一个触发信号完成一段采样，采样完成后产生相应的段采样完成中断。

12.1.1 功能框图

每个 ADC 接口包括 16 个数据寄存器（ADC 16 次采样各个通道模拟量对应的数字量），以及若干控制寄存器，14 个常规触发采样寄存器，2 个空闲采样触发寄存器。

可以同一时刻进行 2 个通道的采样。

以 ADC0 为例，数据寄存器 ADC0_DATx 用于存储 ADC0 第 x 次采样得到的数字量。被转换的模拟信号来源由寄存器 ADC0_CHNx 中的某 4bit 进行选择（详见 12.2.3）。以 ADC0_CHN0 为例，位[3:0]选择第 0 次采样的模拟信号来源，ADC01_CH4~ADC01_CH9，(ADC0 和 ADC1 共用的通道 4-9)ADC0_CH10~ADC0_CH13 任选，若 ADC0_CHN0[3:0]=0，ADC0_CHN0[7:4]=4，则第 0 个采样的模拟量正端信号来自 OPA0_OUT，第 1 个采样的模拟量正端信号对应 IO ADC01_CH4，以此类推。

采样通道数寄存器 ADC_CHNT 控制每段采样的次数，1~15 对应 1~15 次，不可设置为 0 次。两段采样次数之和不应超过 16 次。

控制逻辑根据触发配置寄存器 ADC_TRIG 选择来自 MCPWM 或 TIMER 的触发信号启动一段采样或者软件触发启动。MCPWM/TIMER 会送出定时触发信号。

一段转换（一段内的所有通道采样转换完毕）完成，触发 ADC 转换完成中断。

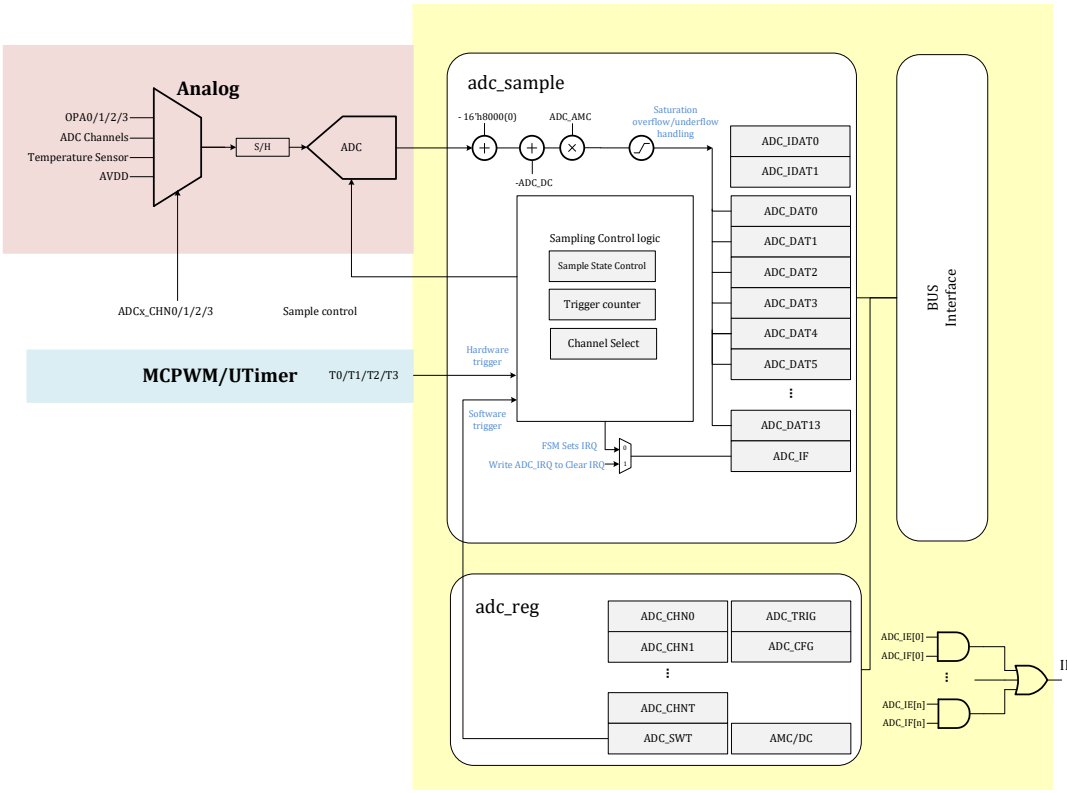


图 12-1 ADC 采集模块功能框图

来源可选使得用户可以灵活配置采样顺序、以及采样信号来源，甚至实现对单个信号多次采样的目的。控制寄存器使得用户可以配置采样个数，提高采样频率/降低采样功耗。

正负端信号来源均可配置，使得用户可以灵活设置每次采样的差分信号来源，灵活使用 IO 资源。

过采样允许用户在采样率要求不高但精度要求的场景下，通过多次采样对信号进行平均从而得到更高的信噪比。

12.1.2 ADC 触发方式

- 支持单段触发和两段触发
- 可以设置采样触发事件发生次数从而实现间隔触发
- 可以软件触发
- 每段采样触发完成可产生对应中断标志，如果使能中断请求，则会向 NVIC 产生中断服务请求
- 触发指示信号可以通过 GPIO 送出用于调试

- 支持空闲触发采样

ADC 触发分为常规触发(NT: Normal Trigger)和空闲触发(IT: Idle Trigger)两种触发。常规触发比空闲触发有更高优先级。如果 ADC 正在进行常规采样转换，此时发生了空闲触发，则 ADC 会继续完成常规触发的通道采样转换，完成后再进行空闲触发采样转换；如果 ADC 正在进行空闲采样转换，此时发生了常规触发，则常规触发会打断空闲采样转换，等常规触发采样完成后再进行空闲触发。

SYS_AFE_INFO.VERSION<=3 版本的芯片，空闲触发与常规触发不可同时使用；SYS_AFE_INFO.VERSION=4 版本的芯片，空闲触发与常规触发可以同时使用。

12.1.3 ADC 通道选择

每个 ADC 模块有 5 个通道信号来源寄存器，控制采样序列 0~15 的信号选择。ADC_CHN0 控制第 0~3 次采样， ADC_CHN1 控制第 4~7 次采样，ADC_CHN2 控制第 8~11 次采样，ADC_CHN3 控制第 12~13 次采样，ADC_ICHN 控制第 0~1 次空闲采样。每次采样通道值选择范围都是 0~15，对应通道 0~15， 也就是可以对某一个通道进行多次采样。每一个采样对应一个结果寄存器，转换结束后，可以直接到对应结果寄存器中获取到 ADC 采样结果。

表 12-1 ADC 通道选择与寄存器配置对应关系

ADC 采样序列	对应的采样数据寄存器	对应信号来源寄存器
第 0 次常规采样	ADC_DAT0	ADC_CHN0[3:0]
第 1 次常规采样	ADC_DAT1	ADC_CHN0[7:4]
第 2 次常规采样	ADC_DAT2	ADC_CHN0[11:8]
第 3 次常规采样	ADC_DAT3	ADC_CHN0[15:12]
第 4 次常规采样	ADC_DAT4	ADC_CHN1[3:0]
……		
第 13 次常规采样	ADC_DAT13	ADC_CHN3[7:4]
第 0 次空闲采样	ADC_IDAT0	ADC_ICHN[3:0]
第 1 次空闲采样	ADC_IDAT1	ADC_ICHN[7:4]

12.1.4 ADC 中断

ADC 中断信号在每段采样完成后置高。

软件或硬件触发事件如果在 ADC 工作时发生，则产生异常触发中断

ADC_DAT0 具备阈值中断，阈值可以设置为上阈值/下阈值，通过 ADC_CFG[1]设置，当 ADC_DAT0 超过 ADC_DAT0_TH 时发生中断。

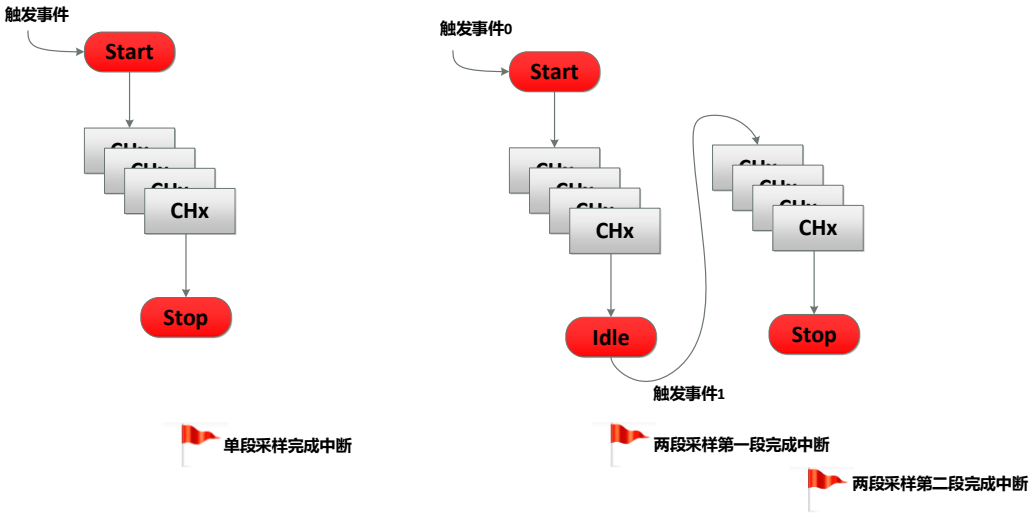


图 12-2 ADC 中断产生

12.1.5 ADC 输出数制

ADC 输出数据为 12bit 补码，输入信号 0 对应 12’b0000_0000_0000。

ADC 有两种量程设置 3.6V 和 7.2V。以 3.6V 量程为例，输入信号-3.6V 对应 12’b1000_0000_0000，输入信号+3.6V 对应 12’b0111_1111_1111。ADC 转换后的 12bit 补码需进行符号扩展后存入 16bit 位宽的采样数据寄存器，左对齐/右对齐可根据配置寄存器进行设置。以 12’b1000_0000_1101 为例，如果配置为左对齐，右侧补 4 个 0，存入 ADC_DAT 的值为 16’b1000_0000_1101_0000；如果配置为右对齐，左侧进行符号扩展，存入 ADC_DAT 的值为 16’b1111_1000_0000_1101。推荐统一使用左对齐方式。

由于芯片使用 5V 供电，因此 7.2V 量程实际支持的差分信号范围是±5V。因此 7.2V 量程无法用满整个 ADC 量程。

表 12-2 ADC 输出数字量数制转换

ADC 3.6V 量程时不同输入信号/V	ADC 7.2V 量程时不同输入信号/V	转为有符号数后的数值
3.6	7.2	12’b0111_1111_1111
0	0	12’b0000_0000_0000
-3.6	-7.2	12’b1000_0000_0000

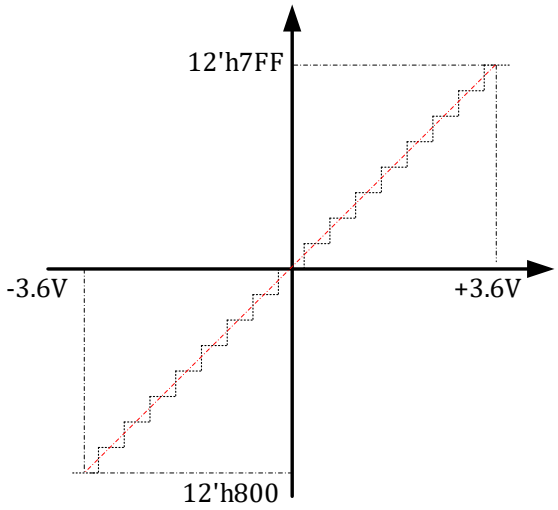


图 12-3 2/3 增益设置下 ADC 模数转换数制量程

12.1.6 ADC 量程

ADC 在 2.4V 的默认基准电压配置下，有两种量程设置：3.6V 和 7.2V。7.2V 量程下，对应最大±5V 的输入信号幅度，3.6V 量程下，对应最大±3.6V 的输入信号幅度。

在 ADC 采样通道配置为运放的输出信号时（即 OPA0~OPA3），应选择合适的运放增益，使得具体应用上的最大信号可被放大到接近+/-3.3V 的水平，同时将 ADC 配置为 3.6V 量程。举例来说，相线电流最大 100A（正弦波有效值），MOS 内阻（假设为 MOS 内阻采样）为 5mR，则运放的最大输入信号幅值为+/-707mV。此时应该选择运放的放大倍数为 4.5 倍，则放大后的信号约为+/-3.18V。

如果因为客观原因，运放的输出信号经放大后，最大信号仍然小于+/-2.4V，则应将 ADC 配置为 3.6V 量程。

在 ADC 采样通道配置为 GPIO 复用口输入的信号时，同样根据信号的最大幅度来选择 ADC 增益。由于 IO 口的限制，GPIO 复用口输入的信号范围只能在-0.3V~AVDD+0.3V 之间。

12.1.7 ADC 校正

ADC 硬件接口模块可以进行直流偏置校正与增益校正。

ADC_AMC 存储的是增益校正系数 $AMP_{correction}$，为 10bit 无符号定点数，ADC_AMC[9]为整数部分，ADC_AMC[8:0]为小数部分。可以表示数值在 1 附近的定点数。

记 ADC 输出的数字量为 D_{ADC}，D_{ADC} 对应的真实值为 D，D_0 为编码数制的 0，则

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

最终硬件会将进行校正后的 D 存入相应的采样数据寄存器。ADC 接口硬件电路会根据每个通道的增益配置(ADCx_GAIN)来自动选择 $AMP_{correction}$ 与 DC_{offset}。

12.1.8 ADC 阈值监测(模拟看门狗)

每个 ADC 接口模块配备有 1 组阈值监测电路，同时进行上阈值和下阈值监测，ADC 的数据寄存器可以单独配置是否启用某组阈值监测，1 个数据寄存器可以同时启用多组阈值监测，但通常不这样配置。如果使能阈值监测，当 ADC 完成某次转换，且将经过校正的数据写入 ADCx_DAT 寄存器时，会进行阈值比较，如果写入的数据大于上阈值或小于下阈值则对应的阈值超限中断标志会置 1。

阈值为 12bit 有符号数，因此阈值比较与数据的左右对齐无关。左对齐时 ADCx_DATx[15:4]与阈值进行比较，右对齐时，ADCx_DATx[11:0]与阈值进行比较。

注意：

ADC 阈值监测逻辑会把 ADC_DATA 的值以及 ADC_LTH 和 ADC_HTH 的值都当作无符号数进行阈值监测。因此阈值检测功能的使用场景有所限制，具体的使用场景如下：

1、 当 ADC_LTH 和 ADC_HTH 的值换算成有符号的数都为正值，即 0<ADC_LTH<ADC_HTH 时，阈值监测功能可以正常使用。

2、 当 ADC_LTH 和 ADC_HTH 的值换算成有符号的数都为负值，即 ADC_LTH<ADC_HTH<0 时，阈值监测功能可以正常使用。

除了上述场景之外，阈值监测功能无法正常使用。如 ADC_HTH 和 ADC_LTH 的值换算成有符号数时符号位不一致，并且在有符号数的情况下 ADC_HTH>ADC_LTH，那么在换算成无符号数时会导致 ADC_HTH<ADC_LTH，阈值采样中断会不断产生。

12.1.9 过采样

ADC 支持对信号过采样求平均后再写入数据寄存器，过采样率范围是 1~128 倍，但只能是 2 的幂次，通过 ADCx_CFG.OVSR 配置。默认 1 次采样即写入数据寄存器，即不进行过采样。如果配置了过采样倍数，则由 ADCx_CHNx 指定的所有信号，均进行多次采样求平均后才写入数据寄存器。写入数据寄存器的数值与 ADC 多次转换数值的关系如下公式所示。

$$\text{ADCx\_DAT} = \frac{1}{OVSR}\sum_{i=0}^{OVSR-1}\text{ADCx\_DAT\_RAW}_i$$

过采样可以配合阈值监测使用，此时只有当平均后的数值超过阈值范围时才产生阈值超限事件。

当配置了过采样模式后，每段采样转换时间按倍数增加，并在采样转换完成后产生中断。

通过 ADCx_CFG.TROVS，可以配置一次触发即完成多次(ADCx_CFG.OVSR 次)采样并进行数据平均，如图 12-4 所示，ADCx_CFG.TROVS=0，ADCx_CFG.OVSR=1 即过采样率为 2，触发后，ADC 对每个信号都采样两次平均后才将结果写入对应的 ADCx_DAT 寄存器；如果 ADCx_CFG.TROVS=1，则需要多次(ADCx_CFG.OVSR 次)触发才积累足够数据进行平均，如图 12-5 所示，ADCx_CFG.OVSR=1 即过采样率仍为 2，则每 2 次触发才累计 2 次数据进行平均并写入数据寄存器。当需要多次触发进行累加时，每次触发只能采样一个通道，即要求 ADCx_CHNT=1，如果采样多个不同信号，当通道切换时，会导致不同通道数据累加在一起而得到错误的转换后数据。

过采样可以配合连续采样进行使用，这种场景中，ADC 连续反复对信号进行采样，累计到过采样次数后将数据存入数据寄存器。

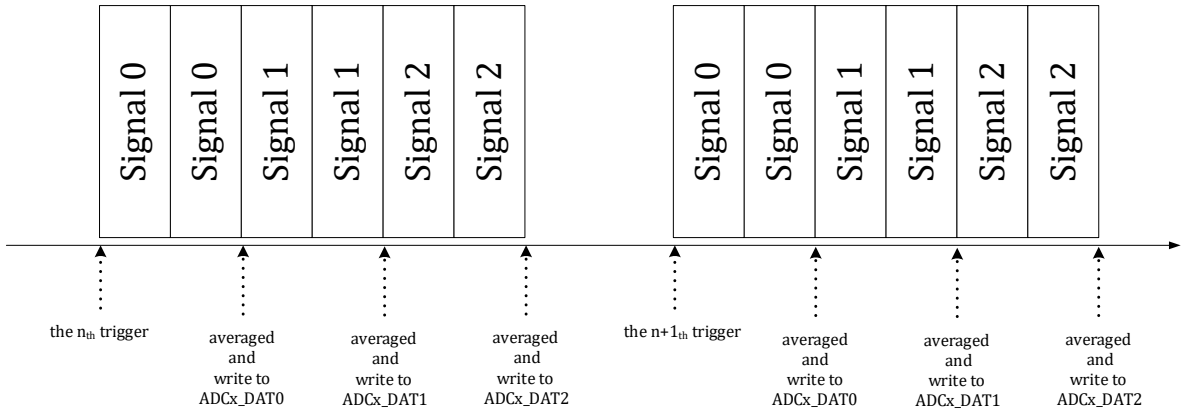


图 12-4 过采样转换时序示例 1

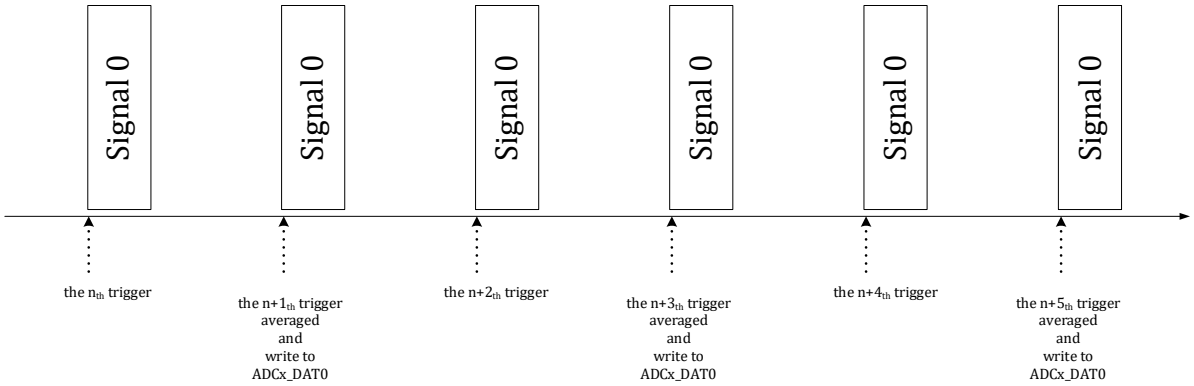


图 12-5 过采样转换时序示例 2

只有常规采样支持过采样，常规采样过程中出现空闲采样不会影响过采样的数据累积。

12.2 寄存器

12.2.1 地址分配

ADC0 在芯片中的基地址是 0x4001_0400；

ADC1 在芯片中的基地址是 0x4001_0500。

表 12-3 ADCx 寄存器列表

名称	偏移地址	说明
ADCx_DAT0	0x00	ADCx 第 0 次常规采样数据
ADCx_DAT1	0x04	ADCx 第 1 次常规采样数据
ADCx_DAT2	0x08	ADCx 第 2 次常规采样数据
ADCx_DAT3	0x0C	ADCx 第 3 次常规采样数据
ADCx_DAT4	0x10	ADCx 第 4 次常规采样数据
ADCx_DAT5	0x14	ADCx 第 5 次常规采样数据
ADCx_DAT6	0x18	ADCx 第 6 次常规采样数据
ADCx_DAT7	0x1C	ADCx 第 7 次常规采样数据

ADCx_DAT8	0x20	ADCx 第 8 次常规采样数据
ADCx_DAT9	0x24	ADCx 第 9 次常规采样数据
ADCx_DAT10	0x28	ADCx 第 10 次常规采样数据
ADCx_DAT11	0x2C	ADCx 第 11 次常规采样数据
ADCx_DAT12	0x30	ADCx 第 12 次常规采样数据
ADCx_DAT13	0x34	ADCx 第 13 次常规采样数据
ADCx_IDAT0	0x38	ADCx 第 0 次空闲采样数据
ADCx_IDAT1	0x3C	ADCx 第 1 次空闲采样数据
ADCx_ICHN	0x40	ADCx 空闲采样通道配置
ADCx_CHN0	0x50	ADCx 第 0~ 3 次常规采样输入信号选择
ADCx_CHN1	0x54	ADCx 第 4~ 7 次常规采样输入信号选择
ADCx_CHN2	0x58	ADCx 第 8~11 次常规采样输入信号选择
ADCx_CHN3	0x5C	ADCx 第 12~15 次常规采样输入信号选择
ADCx_CHNT	0x60	ADCx 各种触发模式下采样次数
ADCx_GAIN	0x64	ADCx 增益选择
ADCx_CFG	0x74	ADCx 模式配置
ADCx_TRIG	0x78	ADCx 采样触发配置
ADCx_SWT	0x7C	ADCx 软件触发
ADCx_DC0	0x80	ADCx 增益为 0 时 DC offset
ADCx_AMC0	0x84	ADCx 增益为 0 时增益校正系数
ADCx_DC1	0x88	ADCx 增益为 1 时 DC offset
ADCx_AMC1	0x8C	ADCx 增益为 1 时增益校正系数
ADCx_IE	0x90	ADCx 中断使能
ADCx_IF	0x94	ADCx 中断标志
ADCx_LTH	0xC4	ADCx 数据低阈值
ADCx_HTH	0xC8	ADCx 数据高阈值
ADCx_GEN	0xCC	ADCx 阈值监测使能

12.2.2 采样数据寄存器

12.2.2.1 ADCx_DAT0(x = 0,1)

地址分别为：0x4001_0400，0x4001_0500

复位值：0x0

表 12-4 采样数据寄存器 ADCx_DAT0

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

位置	位名称	说明
[31:16]		未使用
[15:0]	DAT0	ADCx 第 0 次采样数据

12.2.2.2 ADCx_DAT1(x = 0,1)

地址分别为：0x4001_0404，0x4001_0504

复位值：0x0

表 12-5 采样数据寄存器 ADCx_DAT1

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

位置	位名称	说明
[31:16]		未使用
[15:0]	DAT1	ADCx 第 1 次采样数据

12.2.2.3 ADCx_DAT2(x = 0,1)

地址分别为：0x4001_0408，0x4001_0508

复位值：0x0

表 12-6 采样数据寄存器 ADCx_DAT2

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

位置	位名称	说明
[31:16]		未使用
[15:0]	DAT2	ADCx 第 2 次采样数据

12.2.2.4 ADCx_DAT3(x = 0,1)

地址分别为：0x4001_040C，0x4001_050C

复位值：0x0

表 12-7 采样数据寄存器 ADCx_DAT3

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

位置	位名称	说明
[31:16]		未使用
[15:0]	DAT3	ADCx 第 3 次采样数据

12.2.2.5 ADCx_DAT4(x = 0,1)

地址分别为：0x4001_0410，0x4001_0510

复位值：0x0

表 12-8 采样数据寄存器 ADCx_DAT4

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

位置	位名称	说明
[31:16]		未使用
[15:0]	DAT4	ADCx 第 4 次采样数据

12.2.2.6 ADCx_DAT5(x = 0,1)

地址分别为：0x4001_0414，0x4001_0514

复位值：0x0

表 12-9 采样数据寄存器 ADCx_DAT5

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

位置	位名称	说明
[31:16]		未使用
[15:0]	DAT5	ADCx 第 5 次采样数据

12.2.2.7 ADCx_DAT6(x = 0,1)

地址分别为：0x4001_0418，0x4001_0518

复位值：0x0

表 12-10 采样数据寄存器 ADCx_DAT6

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

位置	位名称	说明
[31:16]		未使用
[15:0]	DAT6	ADCx 第 6 次采样数据

12.2.2.8 ADCx_DAT7(x = 0,1)

地址分别为：0x4001_041C，0x4001_051C

复位值：0x0

表 12-11 采样数据寄存器 ADCx_DAT7

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

位置	位名称	说明
[31:16]		未使用
[15:0]	DAT7	ADCx 第 7 次采样数据

12.2.2.9 ADCx_DAT8(x = 0,1)

地址分别为：0x4001_0420，0x4001_0520

复位值：0x0

表 12-12 采样数据寄存器 ADCx_DAT8

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT8															
RW															
0															

位置	位名称	说明
[31:16]		未使用
[15:0]	DAT8	ADCx 第 8 次采样数据

12.2.2.10 ADCx_DAT9(x = 0,1)

地址分别为：0x4001_0424，0x4001_0524

复位值：0x0

表 12-13 采样数据寄存器 ADCx_DAT9

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

位置	位名称	说明
[31:16]		未使用
[15:0]	DAT9	ADCx 第 9 次采样数据

12.2.2.11 ADCx_DAT10(x = 0,1)

地址分别为：0x4001_0428，0x4001_0528

复位值：0x0

表 12-14 采样数据寄存器 ADCx_DAT10

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

位置	位名称	说明
[31:16]		未使用
[15:0]	DAT10	ADCx 第 10 次采样数据

12.2.2.12 ADCx_DAT11(x = 0,1)

地址分别为：0x4001_042C，0x4001_052C

复位值：0x0

表 12-15 采样数据寄存器 ADCx_DAT11

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

位置	位名称	说明
[31:16]		未使用
[15:0]	DAT11	ADCx 第 11 次采样数据

12.2.2.13 ADCx_DAT12(x = 0,1)

地址分别为：0x4001_0430，0x4001_0530

复位值：0x0

表 12-16 采样数据寄存器 ADCx_DAT12

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

位置	位名称	说明
[31:16]		未使用
[15:0]	DAT12	ADCx 第 12 次采样数据

12.2.2.14 ADCx_DAT13(x = 0,1)

地址分别为：0x4001_0434，0x4001_0534

复位值：0x0

表 12-17 采样数据寄存器 ADCx_DAT13

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

位置	位名称	说明
[31:16]		未使用
[15:0]	DAT13	ADCx 第 13 次采样数据

12.2.2.15 ADCx_IDAT0(x = 0,1)

地址分别为：0x4001_0438，0x4001_0538

复位值：0x0

表 12-18 空闲采样数据寄存器 ADCx_IDAT0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDAT0															
RW															
0															

位置	位名称	说明
[31:16]		未使用
[15:0]	IDAT0	ADCx 第 0 次空闲采样数据

12.2.2.16 ADCx_IDAT1(x = 0,1)

地址分别为：0x4001_043C，0x4001_053C

复位值：0x0

表 12-19 空闲采样数据寄存器 ADCx_IDAT1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDAT1															
RW															
0															

位置	位名称	说明
[31:16]		未使用
[15:0]	IDAT15	ADCx 第 1 次空闲采样数据

12.2.3 信号来源寄存器

12.2.3.1 ADCx_CHN0(x = 0,1)

地址分别为：0x4001_0450，0x4001_0550

复位值：0x0

表 12-20 信号来源寄存器 ADCx_CHN0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDS3				PDS2				PDS1				PDS0			
RW				RW				RW				RW			
0				0				0				0			

位置	位名称	说明
[31:16]		未使用

[15:12]	PDS3	ADCx 第 3 次采样正端输入信号选择
[11:8]	PDS2	ADCx 第 2 次采样正端输入信号选择
[7:4]	PDS1	ADCx 第 1 次采样正端输入信号选择
[3:0]	PDS0	ADCx 第 0 次采样正端输入信号选择

12.2.3.2 ADCx_CHN1(x = 0,1)

地址分别为：0x4001_0454，0x4001_0554

复位值：0x0

表 12-21 信号来源寄存器 ADCx_CHN1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDS7				PDS6				PDS5				PDS4			
RW				RW				RW				RW			
0				0				0				0			

位置	位名称	说明
[31:16]		未使用
[15:12]	PDS7	ADCx 第 7 次采样正端输入信号选择
[11:8]	PDS6	ADCx 第 6 次采样正端输入信号选择
[7:4]	PDS5	ADCx 第 5 次采样正端输入信号选择
[3:0]	PDS4	ADCx 第 4 次采样正端输入信号选择

12.2.3.3 ADCx_CHN2(x = 0,1)

地址分别为：0x4001_0458，0x4001_0558

复位值：0x0

表 12-22 信号来源寄存器 ADCx_CHN2

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDS11				PDS10				PDS9				PDS8			
RW				RW				RW				RW			
0				0				0				0			

位置	位名称	说明
[31:16]		未使用
[15:12]	PDS11	ADCx 第 11 次采样正端输入信号选择
[11:8]	PDS10	ADCx 第 10 次采样正端输入信号选择
[7:4]	PDS9	ADCx 第 9 次采样正端输入信号选择
[3:0]	PDS8	ADCx 第 8 次采样正端输入信号选择

12.2.3.4 ADCx_CHN3(x = 0,1)

地址分别为：0x4001_045C，0x4001_055C

复位值：0x0

表 12-23 信号来源寄存器 ADCx_CHN3

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								PDS13				PDS12			
								RW				RW			
								0				0			

位置	位名称	说明
[31:8]		未使用
[7:4]	PDS13	ADCx 第 13 次采样正端输入信号选择
[3:0]	PDS12	ADCx 第 12 次采样正端输入信号选择

12.2.3.5 ADCx_ICHN(x = 0,1)

地址分别为：0x4001_0440，0x4001_0540

复位值：0x0

表 12-24 信号来源寄存器 ADCx_ICHN

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								IDS1				IDS0			
								RW				RW			
								0				0			

位置	位名称	说明
[31:8]		未使用
[7:4]	IDS1	ADCx 第 1 次空闲采样输入信号选择
[3:0]	IDS0	ADCx 第 0 次空闲采样输入信号选择

12.2.3.6 ADC 通道信号选择

2 个 ADC 接口可以选择的被采集模拟信号不完全相同，具体如表 12-25 所示。

表 12-25 ADC 模拟信号来源分布

寄存器	不同配置值对应通道信号
ADC0_CHNx	0x0: OPA0 输出 0x1: OPA1 输出 0x2: OPA2 输出 0x3: OPA3 输出 0x4: ADC01_CH4 0x5: ADC01_CH5

	0x6: ADC01_CH6 0x7: ADC01_CH7 0x8: ADC01_CH8 0x9: ADC01_CH9 0xA: ADC0_CH10 0xB: ADC0_CH11 0xC: ADC0_CH12 0xD: ADC0_CH13 0xE: DAC0_OUT 0xF: DAC1_OUT
ADC1_CHNx	0x0: OPA0 输出 0x1: OPA1 输出 0x2: OPA2 输出 0x3: OPA3 输出 0x4: ADC01_CH4 0x5: ADC01_CH5 0x6: ADC01_CH6 0x7: ADC01_CH7 0x8: ADC01_CH8 0x9: ADC01_CH9 0xA: ADC1_CH10 0xB: ADC1_CH11 0xC: ADC1_CH12 0xD: ADC1_CH13 0xE:温度传感器 0xF:内部电源 AVDD

*ADC01_CH4~ADC01_CH9 为 ADC0 和 ADC1 公用通道

正常情况下 ADC 对运放通道的采样，采的都是差分信号。

12.2.4 采样通道数寄存器

12.2.4.1 ADCx_CHNT(x = 0,1)

地址分别为：0x4001_0460，0x4001_0560

复位值：0x0

表 12-26 采样通道数寄存器 ADCx_CHNT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						IS1		S2				S1			
						RW		RW				RW			
						0		0				0			

位置	位名称	说明
[31:10]		未使用

[9:8]	IS1	空闲采样次数
[7:4]	S2	第二段常规采样次数
[3:0]	S1	第一段常规采样次数

注意：采样次数不允许配置为 0。1 表示 1 个通道，2 表示 2 个通道，…… ，16 表示 16 个通道。如果常规采样使用两段采样，则两段采样通道数之和不应超过 16。

空闲采样仅支持单段采样，采样次数可以配置为 1 或 2。

12.2.5 配置寄存器

12.2.5.1 ADCx_GAIN(x = 0,1)

地址分别为：0x4001_0464，0x4001_0564

复位值：0x0

表 12-27 增益选择寄存器 ADCx_GAIN

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IG1	IG0	G13	G12	G11	G10	G9	G8	G7	G6	G5	G4	G3	G2	G1	G0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:16]		未使用
[15]	IG1	IDAT1 增益选择，0:±3.6V 量程，1:±7.2V 量程
[14]	IG0	IDAT0 增益选择，0:±3.6V 量程，1:±7.2V 量程
[13]	G13	DAT13 增益选择，0:±3.6V 量程，1:±7.2V 量程
[12]	G12	DAT12 增益选择，0:±3.6V 量程，1:±7.2V 量程
[11]	G11	DAT11 增益选择，0:±3.6V 量程，1:±7.2V 量程
[10]	G10	DAT10 增益选择，0:±3.6V 量程，1:±7.2V 量程
[9]	G9	DAT9 增益选择，0:±3.6V 量程，1:±7.2V 量程
[8]	G8	DAT8 增益选择，0:±3.6V 量程，1:±7.2V 量程
[7]	G7	DAT7 增益选择，0:±3.6V 量程，1:±7.2V 量程
[6]	G6	DAT6 增益选择，0:±3.6V 量程，1:±7.2V 量程
[5]	G5	DAT5 增益选择，0:±3.6V 量程，1:±7.2V 量程
[4]	G4	DAT4 增益选择，0:±3.6V 量程，1:±7.2V 量程
[3]	G3	DAT3 增益选择，0:±3.6V 量程，1:±7.2V 量程
[2]	G2	DAT2 增益选择，0:±3.6V 量程，1:±7.2V 量程
[1]	G1	DAT1 增益选择，0:±3.6V 量程，1:±7.2V 量程
[0]	G0	DAT0 增益选择，0:±3.6V 量程，1:±7.2V 量程

受限于 5V 电源，7.2V 量程只能用到±5V 范围。

12.2.5.2 ADCx_CFG (x = 0,1)

地址分别为：0x4001_0474，0x4001_0574

复位值：0x0

表 12-28 模式配置寄存器 ADCx_CFG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			NSMP	FSM_RS	DATA_ALIGN		CSMP	TCNT				TROVS	OVSR		
			RW	RW	RW		RW	RW				RW	RW		
			0	0	0		0	0				0	0		

位置	位名称	说明
[31:13]		未使用
[12]	NSMP	0:单段采样，1:两段采样
[11]	FSM_RS	状态机复位控制信号。软件写入 1 产生复位，将 ADC 内部状态机回到初始状态，完成后自动回到 0。该复位控制，不影响 ADC 其它寄存器的配置值。 读取该位返回 ADC 当前状态，1 表示 ADC 目前正在采样转换工作中，0 表示空闲。
[10]	DATA_ALIGN	ADC_DAT 对齐方式 0：左对齐，右端补 4'h0， 1：右对齐，左端补 4bit 符号位
[9]		未使用
[8]	CSMP	连续采样模式 0：不开启 1：开启连续采样模式，触发到来后，ADC 即开始采样转换，完成所有信号的转换后，无需等待触发立即从第 0 个信号开始新一轮的采样转换
[7:4]	TCNT	触发一次采样所需的事件数。 0：表示需要发生 1 次事件才能触发一次采样 1：表示需要发生 2 次事件才能触发一次采样 …… 15：表示需要发生 16 次事件才能触发一次采样
[3]	TROVS	过采样触发模式 0：一次触发即连续过采样多次至 OVSR，并将数据平均后存入，可以配置采样多个通道，每个通道都采样 OVSR 次后才进行下一个通道的采样。 1：一次触发只进行一次采样转换，需要 OVSR 次触发才完成足够数据的采集并进行平均，而后存入数据寄存器。受到累加器个数限制，在此种配置下，ADCx_CHNT 只能配置采

		样一个通道。
[2:0]	OVSR	过采样率 0：1，默认 1 次采样即存入数据 1：2 次采样后存入数据 2：4 次采样后存入数据 3：8 次采样后存入数据 4：16 次采样后存入数据 5：32 次采样后存入数据 6：64 次采样后存入数据 7：128 次采样后存入数据 过采样率设置为非 0 值时需要保证 ADC0 和 ADC1 不会同时工作。即其中一个 ADC 采样转换过程中，另外一个 ADC 的采样触发事件不能发生。

注意：ADC_CFG 寄存器中 ADCx_CFG.OVSR 字段过采样率的设置可以设置为 0~7。但是将过采样率设置为非 0 值时需要保证 ADC0 和 ADC1 不会同时工作。即其中一个 ADC 采样转换过程中，另外一个 ADC 的采样触发事件不能发生。

12.2.5.3 ADC0_TRIG

地址分别为：0x4001_0478

复位值：0x0

表 12-29 采样触发配置寄存器 ADC0_TRIG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CL_IT3_EN	CL_IT2_EN	CL_IT1_EN	CL_IT0_EN	CL_NT3_EN	CL_NT2_EN	CL_NT1_EN	CL_NT0_EN	UTIME2_CMP1_EN	UTIME2_CMP0_EN	UTIME0_CMP1_EN	UTIME0_CMP0_EN	MCPWM0_T3_EN	MCPWM0_T2_EN	MCPWM0_T1_EN	MCPWM0_T0_EN
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:16]		未使用
[15]	CL_IT3_EN	CL_OUTPUT[3]触发 ADC 空闲采样使能，高有效
[14]	CL_IT2_EN	CL_OUTPUT[2]触发 ADC 空闲采样使能，高有效
[13]	CL_IT1_EN	CL_OUTPUT[1]触发 ADC 空闲采样使能，高有效
[12]	CL_IT0_EN	CL_OUTPUT[0]触发 ADC 空闲采样使能，高有效
[11]	CL_NT3_EN	CL_OUTPUT[3]触发 ADC 常规采样使能，高有效
[10]	CL_NT2_EN	CL_OUTPUT[2]触发 ADC 常规采样使能，高有效
[9]	CL_NT1_EN	CL_OUTPUT[1]触发 ADC 常规采样使能，高有效
[8]	CL_NT0_EN	CL_OUTPUT[0]触发 ADC 常规采样使能，高有效
[7]	TIMER2_CMP1_EN	TIMER2 比较事件 1 触发 ADC 常规采样使能，高有效

[6]	TIMER2_CMP0_EN	TIMER2 比较事件 0 触发 ADC 常规采样使能，高有效
[5]	TIMER0_CMP1_EN	TIMER0 比较事件 1 触发 ADC 常规采样使能，高有效
[4]	TIMER0_CMP0_EN	TIMER0 比较事件 0 触发 ADC 常规采样使能，高有效
[3]	MCPWM0_T3_EN	MCPWM0 T3 事件触发 ADC 常规采样使能，高有效
[2]	MCPWM0_T2_EN	MCPWM0 T2 事件触发 ADC 常规采样使能，高有效
[1]	MCPWM0_T1_EN	MCPWM0 T1 事件触发 ADC 常规采样使能，高有效
[0]	MCPWM0_T0_EN	MCPWM0 T0 事件触发 ADC 常规采样使能，高有效

12.2.5.4 ADC1_TRIG

地址分别为：，0x4001_0578

复位值：0x0

表 12-30 采样触发配置寄存器 ADC1_TRIG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CL_IT3_EN	CL_IT2_EN	CL_IT1_EN	CL_IT0_EN	CL_NT3_EN	CL_NT2_EN	CL_NT1_EN	CL_NT0_EN	UTIME3_CMP1_EN	UTIME3_CMP0_EN	UTIME1_CMP1_EN	UTIME1_CMP0_EN	MCPWM0_T3_EN	MCPWM0_T2_EN	MCPWM0_T1_EN	MCPWM0_T0_EN
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:16]		未使用
[15]	CL_IT3_EN	CL_OUTPUT[3]触发 ADC 空闲采样使能，高有效
[14]	CL_IT2_EN	CL_OUTPUT[2]触发 ADC 空闲采样使能，高有效
[13]	CL_IT1_EN	CL_OUTPUT[1]触发 ADC 空闲采样使能，高有效
[12]	CL_IT0_EN	CL_OUTPUT[0]触发 ADC 空闲采样使能，高有效
[11]	CL_NT3_EN	CL_OUTPUT[3]触发 ADC 常规采样使能，高有效
[10]	CL_NT2_EN	CL_OUTPUT[2]触发 ADC 常规采样使能，高有效
[9]	CL_NT1_EN	CL_OUTPUT[1]触发 ADC 常规采样使能，高有效
[8]	CL_NT0_EN	CL_OUTPUT[0]触发 ADC 常规采样使能，高有效
[7]	TIMER3_CMP1_EN	TIMER3 比较事件 1 触发 ADC 常规采样使能，高有效
[6]	TIMER3_CMP0_EN	TIMER3 比较事件 0 触发 ADC 常规采样使能，高有效
[5]	TIMER1_CMP1_EN	TIMER1 比较事件 1 触发 ADC 常规采样使能，高有效
[4]	TIMER1_CMP0_EN	TIMER1 比较事件 0 触发 ADC 常规采样使能，高有效
[3]	MCPWM0_T3_EN	MCPWM0 T3 事件触发 ADC 常规采样使能，高有效
[2]	MCPWM0_T2_EN	MCPWM0 T2 事件触发 ADC 常规采样使能，高有效
[1]	MCPWM0_T1_EN	MCPWM0 T1 事件触发 ADC 常规采样使能，高有效
[0]	MCPWM0_T0_EN	MCPWM0 T0 事件触发 ADC 常规采样使能，高有效

ADC 的触发来源为 MCPWM 或 TIMER，通常不会同时使用 MCPWM 和 TIMER 进行触发，只

在二者之中选一。

TIMER 对 ADC 的触发信号可以通过配置 GPIO 为 Timer 功能(第 7/8 功能)，通过输出 Timer 通道信号进行观测。

MCPWM 对 ADC 的触发信号可以通过配置 GPIO 为第 9 功能，即 ADC_TRIGGER 功能送出用于捕捉调试。每发生一次 ADC 触发，ADC_TRIGGER 信号翻转一次。

当使用两段触发时，只有 BIT0/2/4/6.../16/18/...等位置的触发信号可以作为第一段触发信号，只有 BIT1/3/5/.../17/19/...等位置的触发信号可以作为第二段的触发信号。

举例来说当 MCPWM0_T1_EN 和 MCPWM0_T0_EN 都使能的时候，MCPWM0_T0_EN 会触发第一段采样，但不会触发第二段采样；MCPWM0_T1_EN 会触发第二段采样，但不会触发第一段采样。

12.2.6 软件触发寄存器

12.2.6.1 ADCx_SWT(x = 0,1)

地址分别为：0x4001_047C，0x4001_057C

复位值：0x0

表 12-31 软件触发寄存器 ADCx_SWT

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

位置	位名称	说明
[31:16]		未使用
[15:0]	SWT	写入数据为 0x5AA5 时，产生一次常规触发 写入数据位 0xF00F 时，产生一次空闲触发

注意，软件触发采集寄存器为只写寄存器，且只有写入数据为 0x5AA5/0xF00F 时产生软件触发事件，一次总线的写入产生一次软件触发，数据写入产生一个软件触发后寄存器自动清零，下一次软件触发则再次写入 0x5AA5/0xF00F。

12.2.7 校正寄存器

ADC 的可选模拟通道中一个信号源为 GND，通过对这个通道的测量可以得到系统的 DC 偏置。

通常系统初始化后会进行一次 GND 信号的测量，并将测量值存入 DC offset 寄存器，后续每次其他 channel 的信号的采样值会自动减去 DC offset，再存入转换后的数字量寄存器。

考虑到信号误差，去除 DC offset 的信号可能会发生溢出，对于溢出的数据会做饱和处理，以使信号范围在-1.2V-+1.2V 或-2.4V-+2.4V 之间。

ADC 有两种量程：3.6V 和 7.2V。3.6V 量程设置下，对应最大±3.6V 的输入信号幅度，7.2V 量程设置下，对应最大±5V 的输入信号幅度。两种量程使用不同的 DC offset 和增益校正系数。因此存在两组校正寄存器，分别为 ADCx_DC0，ADCx_AMC0 和 ADCx_DC1，ADCx_AMC1。芯片出厂时已经过工厂标定，标定数据存放在 Flash info 中，芯片上电自动完成校准参数的加载。ADC 模块在初始化的时候，需要根据数据左右对齐模式配置 DC offset, 可以参看芯片供应商提供的库函数。使用 SYS_SFT_RST 软复位 ADC 模块后，ADC 内部的寄存器会被复位。需要重新使用 ADC 初始化函数从 NVR 中读取 DC/AMC 等校准参数。

12.2.7.1 ADCx_DC0(x = 0,1)

地址分别为：0x4001_0480，0x4001_0580

复位值：0x0

表 12-32 0 档增益直流偏置寄存器 ADCx_DC0

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

位置	位名称	说明
[31:16]		未使用
[15:0]	DC_OFFSET	ADC 采样电路 DC offset

考虑到 ADC 的 DC offset 是接近 0 的数值，因此此处寄存器仅有低 10bit 是有实际实现的，高 6bit 仅仅是 ADCx_DC[9]的符号扩展，同时 ADCx_DC 寄存器参与 ADC 数据校正时也会进行符号扩展。

即如果向 ADCx_DC 写入 0x12B0；实际写入的是 0x2B0，而读出时会读出 0xFFB0。

此处存入的 ADC_DC 值应该对应的是右对齐的 offset 数值，当 ADCx_CFG 寄存器配置为左对齐时，硬件会根据对齐的设置，自动调整 ADCx_DC 参与 ADC 校正。具体来说，右对齐时，ADCx_DC[15:0]直接参与校正运算，左对齐时，ADCx_DC[15:0]会先左移 4 位然后参与 ADC 校正运算。

12.2.7.2 ADCx_AMC0(x = 0,1)

地址分别为：0x4001_0484，0x4001_0584

复位值：0x200

表 12-33 0 档增益增益校正寄存器 ADCx_AMC0

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

	RW
	0x200

位置	位名称	说明
[31:10]		未使用
[9:0]	AM_CALI	ADC 采样电路增益校正系数

ADC 的增益校正系数是一个接近 1 的数值，0x200 标定为 1，举例来说 0x1F0 小于 1，0x205 则大于 1。

12.2.7.3 ADCx_DC1(x = 0,1)

地址分别为：0x4001_0488，0x4001_0588

复位值：0x0

表 12-34 1 档增益直流偏置寄存器 ADCx_DC1

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

位置	位名称	说明
[31:16]		未使用
[15:0]	DC_OFFSET	ADC 采样电路 DC offset

12.2.7.4 ADCx_AMC1(x = 0,1)

地址分别为：0x4001_048C，0x4001_058C

复位值：0x200

表 12-35 1 档增益增益校正寄存器 ADCx_AMC1

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

位置	位名称	说明
[31:10]		未使用
[9:0]	AM_CALI	ADC 采样电路增益校正系数

12.2.8 中断寄存器

12.2.8.1 ADCx_IE(x = 0,1)

地址分别为：0x4001_0490，0x4001_0590

复位值：0x0

表 12-36 中断使能寄存器 ADCx_IE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISF_RE	HERR_RE	SERR_RE			AWD0_RE	SF2_RE	SF1_RE	ISF_IE	HERR_IE	SERR_IE			AWD0_IE	SF2_IE	SF1_IE
RW	RW	RW			RW	RW	RW	RW	RW	RW			RW	RW	RW
0	0	0			0	0	0	0	0	0			0	0	0

位置	位名称	说明
[31:16]		未使用
[15]	ISF_RE	空闲采样完成 DMA 请求使能
[14]	HERR_RE	硬件触发发生在非空闲状态 DMA 请求使能
[13]	SERR_RE	软件触发发生在非空闲状态 DMA 请求使能
[12:11]		未使用
[10]	AWD0_RE	阈值监测 0 超限 DMA 请求使能
[9]	SF2_RE	第二段采样完成 DMA 请求使能
[8]	SF1_RE	第一段采样完成 DMA 请求使能
[7]	ISF_IE	空闲采样完成中断使能
[6]	HERR_IE	硬件触发发生在非空闲状态中断使能
[5]	SERR_IE	软件触发发生在非空闲状态中断使能
[4:3]		未使用
[2]	AWD0_IE	阈值监测 0 超限中断使能
[1]	SF2_IE	第二段常规采样完成中断使能
[0]	SF1_IE	第一段常规采样完成中断使能

12.2.8.2 ADCx_IF(x = 0,1)

地址分别为：0x4001_0494，0x4001_0594

复位值：0x0

表 12-37 中断标志寄存器 ADCx_IF

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

	ISF_IF	HERR_IF	SERR_IF		AWD0_IF	SF2_IF	SF1_IF
	RW1C	RW1C	RW1C		RW1C	RW1C	RW1C
	0	0	0		0	0	0

位置	位名称	说明
[31:8]		未使用
[7]	ISF_IF	空闲采样完成中断标志
[6]	HERR_IF	硬件触发发生在非空闲状态中断标志
[5]	SERR_IF	软件触发发生在非空闲状态中断标志
[4:3]		未使用
[2]	AWD0_IF	阈值监测 0 超限中断标志
[1]	SF2_IF	第二段常规采样完成中断标志
[0]	SF1_IF	第一段常规采样完成中断标志

只有当软件/硬件触发 ADC 进行常规采样，且 ADC 此时正在进行常规采样；或软件/硬件触发 ADC 进行空闲采样，且 ADC 此时正在进行空闲采样时会产生触发错误中断标志 ADC_IF[6:5]。

以上 ADCx_IF 标志位，0：表示未发生中断，1: 表示发生过中断，写 1 清零。

12.2.9 模拟看门狗

12.2.9.1 ADC0_LTH

地址为:0x4001_04C4

复位值:0x0000_F800

表 12-38 下阈值寄存器 ADC0_ LTH

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

位置	位名称	说明
[31:16]		未使用
[15:12]		符号扩展
[11:0]	LTH	ADC 模拟看门狗 0 下阈值

ADC0_LTH 只有[11:0]可以写入，读出时[15:12]为符号扩展位，ADC0_LTH 以 BIT[11]作为符号位。

12.2.9.2 ADC0_HTH

地址为:0x4001_04C8

复位值:0x0000_07FF

表 12-39 上阈值寄存器 ADC0_ HTH

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

位置	位名称	说明
[31:16]		未使用
[15:12]		符号扩展
[11:0]	HTH	ADC 模拟看门狗 0 上阈值

ADC0_HTH 只有[11:0]可以写入，读出时[15:12]为符号扩展位，ADC0_HTH 以 BIT[11]作为符号位。

12.2.9.3 ADC0_GEN

地址为:0x4001_04CC

复位值:0x0

表 12-40 监测使能寄存器 ADC0_ GEN

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

位置	位名称	说明
[31:16]		未使用
[15:0]	GEN	ADC 模拟看门狗 0 对应使能位 BIT0: DAT0 看门狗监测使能 BIT1: DAT1 看门狗监测使能 …… BIT13: DAT13 看门狗监测使能 BIT14: IDAT0 看门狗监测使能 BIT15: IDAT1 看门狗监测使能

由于只有一个模拟看门狗，即只有一组高低阈值。同一时刻一般只使能一个 ADC_DAT 寄存器的阈值监测。如果同时使能多组，则多组 ADC_DAT 寄存器被相同的高低阈值监测。

举例来说，如果设置 ADC_GEN[1]=1，则看门狗 0 会监测 ADCx_DAT1 的值，如果大于 ADCx_TH0.HTH 或小于 ADCx_TH0.LTH 则产生对应中断。

12.3 应用指南

12.3.1 ADC 采样触发模式

ADC 支持一段、两段采样模式，每段采样需要特定的外部事件来触发开始，每段采样支持不同采样次数和采样信号通道配置。

第一次触发

来自 MCPWM/TIMER/CL 模块的定时事件 TADC[0]/TADC[1]/TADC[2]/TADC[3]可以触发 ADC 采样。可以选择四个触发源的任何一个或者几个触发采样。也可以通过向 ADCx_SWT 写入命令字的方式 16'h5AA5 软件触发 ADC 采样。

第一段采样

判断是否为一段采样。

是：采样次数达到预设值 ADCx_CHNT[3:0]，ADC 回到空闲状态 0；采样次数未达到预设值，继续采样。

否：采样次数达到预设值 ADCx_CHNT[3:0]，ADC 进入空闲状态 1（两段第一段完成，等待触发第二段）；采样次数未达到预设值，继续第一段采样。

第二段触发

第二段采样

第二段采样次数到达预设值 ADC_CHNT[7:4]，结束本次采样，回到空闲状态 0。

各种硬件触发模式的触发条件汇总如表 12-41 所示。其中单段采样模式较为特殊，可以通过 ADC_CFG 寄存器设置，一次 TADC 事件即触发采样，还是多次 TADC 事件才触发采样；而两段采样模式仅支持一次相应的 TADC 事件即触发一段采样。

此外 ADC 模块也支持通过软件写入特殊数值的方式触发采样，软件触发也仅支持写入一次即触发。

表 12-41 ADC 采样触发模式

	单段触发	两段触发
TADC 触发	None(TADC 触发使能未打开)	第一段 TADC[0] 第二段 TADC[1]
	C 次 TADC[0]	
	C 次 TADC[1]	
	C 次 TADC[2]	
	C 次 TADC[3]	
	C 次* TADC[0]/TADC[1]/ TADC[2]/TADC[3]	
软件触发	向 ADC_SWT 写入 16'h5aa5	第一段向 ADC_SWT 写入 16'h5aa5 第二段向 ADC_SWT 写入 16'h5aa5

*C 次通过 ADC_CFG.SINGLE_TCNT 设置。ADC_CFG.SINGLE_TCNT 只在单段触发下使用，如果同时使能了 TADC[3:0]，则 4 个触发源都会被计数，到 SINGLE_TCNT 次触发一次 ADC 采样转换。

12.3.1.1 单段触发模式

单段触发模式是指 ADC 收到一次触发完成一段采样动作，一段采样可能包含多次对模拟信号的采样，次数由分段采样次数寄存器配 ADCx_CHNT 进行配置，寄存器数值为 1~15 时，对应的采样次数为 1~15。

假设单段采样配置通道数目为4，则采样转换后的数据会依次填充到ADC_DAT0、ADC_DAT1、ADC_DAT2、ADC_DAT3。

触发事件可以是来自外部的MCPWM/TIMER定时信号TADC[0]、TADC[1]、TADC[2]、TADC[3]、发生到预设次数、或者为软件触发。

每个采样的信号源通过信号来源寄存器 ADC_CHN0/1/2/3 进行配置选定，信号源的选定需在触发前完成，且在一次采样过程完成前不应该改变。

完成一段采样动作后，进入空闲状态，并产生采样完成中断。

以 MCPWM 触发单段采样为例，设置 ADC_CFG.TCNT=4，ADCx_TRIG.MCPWM0_T2_EN =1，即 MCPWM0 的 TADC[2]发生 4 次才进行触发，状态转移如图 12-6 所示。

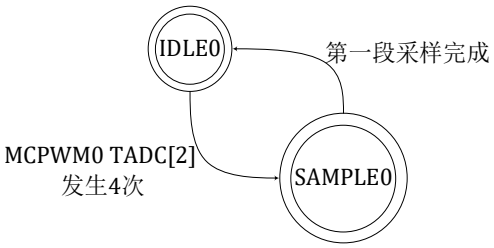


图 12-6 ADC 单段采样状态转移图

12.3.1.2 两段触发模式

两段触发需要两次触发才能完成完整的一轮采样。第一个触发到达时进行第一段采样，第二个触发到达时进行第二段采样。

假设两段采样配置通道数目分别为 2 和 3，则第一段采样转换后的数据会依次填充到 ADC_DAT0、ADC_DAT1，第二段采样转换后的数据会依次填充到 ADC_DAT2、ADC_DAT3、ADC_DAT4。

触发事件可以是来自外部的 MCPWM 定时信号 TADC[0]和 TADC[1]或两次软件触发。

TADC[0]或软件触发发生后，先进行 ADC_CHNT[3:0]次采样，完成后进入空闲状态并等待下一个触发信号的到来；TADC[1]或软件触发作为第二个触发信号发生后，再进行 ADC_CHNT[7:4]次采样。采样次数均通过分段采样次数寄存器 ADC_CHNT 进行配置。

每个采样的信号源通过寄存器配置选定，信号源的选定需在触发前完成，且在一次采样过程完成前不应该改变。

软件触发较硬件触发的优先级低，在硬件触发采样的过程中发生软件触发，状态机不予处理，而产生一个错误中断。即只有状态机处于空闲状态时才会处理软件触发的采样请求。如果需要使用软件触发采样，需要确保硬件触发已经关闭。然后通过向 ADC_SWT 寄存器写入 0x5AA5 以产生一次软件触发。

以两次软件触发两段采样为例，状态转移如图 12-7 所示。

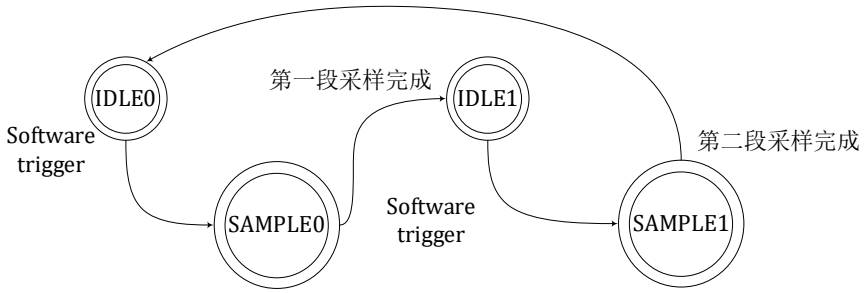


图 12-8 ADC 两段采样状态转移图

12.3.2 中断

12.3.2.1 单段触发采样完成中断

采样完成产生一个中断。

12.3.2.2 两段触发采样完成中断

第一段采样完成产生一个中断，第二段采样完成产生一个中断。

12.3.2.3 ADC_IF 触发所需时间

CPU 软件运行于 PLL 时钟域，ADC 模块运行于 ADC 时钟域。典型地，ADC 时钟周期为 PLL 时钟周期（总线时钟周期）2 倍。 ADC_IF 的清除需要操作 ADC 时钟域内的寄存器，最多需要 20 个总线周期。 建议 ADC_IF 标志在中断函数入口处进行读取清除。 如果清除语句在中断函数最后，可能导致中断函数退出后标志仍未清除，导致再次进入中断。

同理，如果软件在清除 ADC_IF 后立刻查询 ADC_IF，可能 ADC_IF 仍为原始值。

12.3.3 配置修改

建议在 ADC 中断中进行 ADC_CHNx 的配置和修改，因为进入 ADC 中断后说明 ADC 此时已完成一段采样且处于空闲状态。而在主程序中，无法确认 ADC 运行状态，因此在主程序中如需修改 ADC_CHNx 和 ADC_CHNT 等寄存器，需要先关闭 ADC 触发，并向 ADC_CFG[11]写入 1,以复位 ADC 接口电路状态机，确保 ADC 不在工作状态。如果 ADC 在运行中配置发生变化会发生不可预判的行为。

示例程序如下

```
ADCx_CFG_temp = ADCx_CFG;          //Save ADCx_CFG
ADCx_CFG = 0x0000;                 //Disable ADC trigger
ADCx_CFG = 0x0800;                 //Reset ADC state machine
/*
    Add your code below, like:
ADCx_CHNT = 0x0005
ADCx_CHN0 = 0x3210;
ADCx_CHN1 = 0x7654;
```

```
*/
ADCx_CFG = ADCx_CFG_temp;          //restore ADCx_CFG
```

12.3.4 选择对应的模拟通道

ADC 采样信号来源可以参考表 12-25，也可以参考请查阅 DATASHEET 中表 2.2 引脚功能选择。关闭对应 IO 的 IE 和 OE，即可使用其模拟功能。

13 TIMER 通用定时器

13.1 概述

13.1.1 功能框图

如图 13-1 所示，通用定时器 TIMER 主要包括 4 个独立的 Timer。分别可以独立配置运行计数时钟和滤波常数。每个 Timer 可以用于输出特定周期占空比的波形，也可以捕获外部波形进行周期占空比的检测。

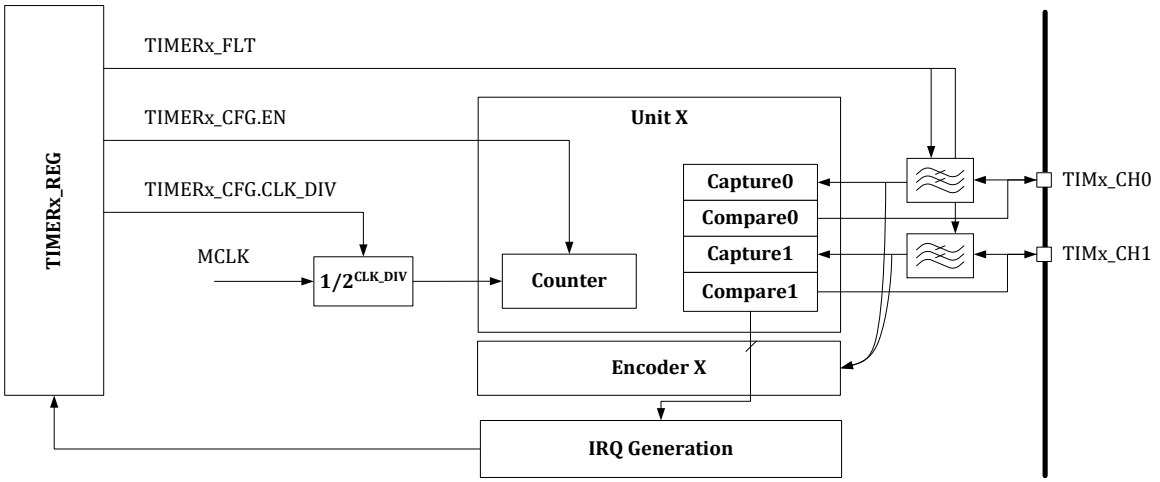


图 13-1 TIMER 模块顶层功能框图

13.1.1.1 寄存器模块

对各个子模块控制寄存器的读写。

对各个子模块状态、结果寄存器的访问。

对各个子模块中断信号的处理和中断产生。

13.1.1.2 IO 滤波模块

IO 滤波模块对来自芯片外部的输入信号进行滤波，降低毛刺对定时器功能的影响。

13.1.1.3 通用定时器模块

通用定时器模块实现了通用的定时器功能，包括比较和捕获工作模式。每个定时器包含两个通道，可以处理两个外部输入信号或者产生两个脉冲信号送到芯片外部。

通用定时器模块，支持外部事件触发开始计数。外部事件源头可配。当外部事件触发后，定时器开始自增。支持使用外部信号作为 timer 时钟进行计数。

13.1.1.4 时钟分频模块

时钟分频模块用于产生定时器工作所需的各种分频时钟。

13.1.2 功能特点

定时器模块有以下特点:

- 独立工作，可工作在不同频率下
- Timer0/1 为 16bit 通用定时器
- Timer2/3 为 32bit 通用定时器
- 每个通用定时器处理 2 个外部输入信号（捕获模式），或者产生 2 个输出信号（比较模式）
- 对每个输入信号可以进行最大 2040 个系统主时钟的滤波，即，当芯片工作在 96MHz 时钟频率下时，可以滤除 21uS 宽度以下毛刺

13.2 特性

13.2.1 时钟分频

Timer 使用 TIMERx_CFG.CLK_DIV 进行分频，可以降低计数器的计数频率，分频系数可以设置为 1/2/4/8/16/32/64/128.

13.2.2 中断标志清零

采用了通过对每个中断标志位写 1 来清除标志位的设计。

13.2.3 滤波

每个定时器模块有两个通道，在工作于输入捕获模式时，两个通道共享一个滤波系数。

通过配置滤波寄存器 TIMERx_FLT，可以调整滤波宽度为 0~2040 个 Timer 计数时钟宽度。

输入信号滤波使用 Timer 计数的分频时钟，**TIMERx_CFG.CLK_DIV 对 Timer 计数时钟的分频对滤波时钟有影响，即 Timer 分频后滤波宽度随之变大**。

如图 13-2 所示，原始输入信号在 t1~t6 几个时刻发生了翻转，滤波器宽度配置成 T。可以看到只有 t3 和 t6 时刻发生的翻转维持了大于 T 的时间，因此从滤波器的输出看，信号仅发生了两次翻转。

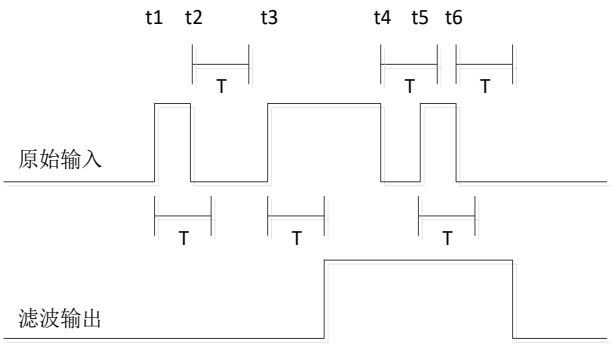


图 13-2 TIMER 滤波示意图

13.2.4 模式

13.2.4.1 计数器

Timer 中的计数器可以选择递增或递减方向计数，默认为递增计数。

若为递增计数，则计数器从 0 计数到 TH 值，再回到 0 重新开始计数，计数器回到 0 时，产生回零中断。若为递减计数，则计数器从 TH 计数到 0 值，再回到 TH 重新开始计数，计数器回到 0 时，产生回零中断。实际计数周期为 clk_freq*(TH+1)。

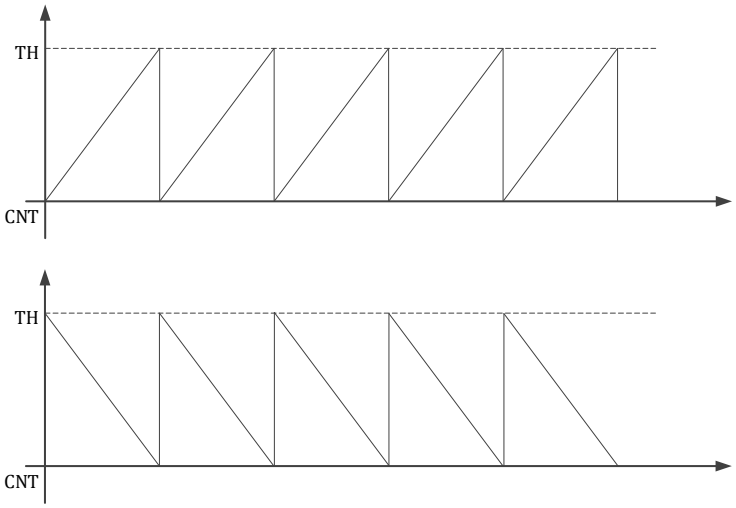


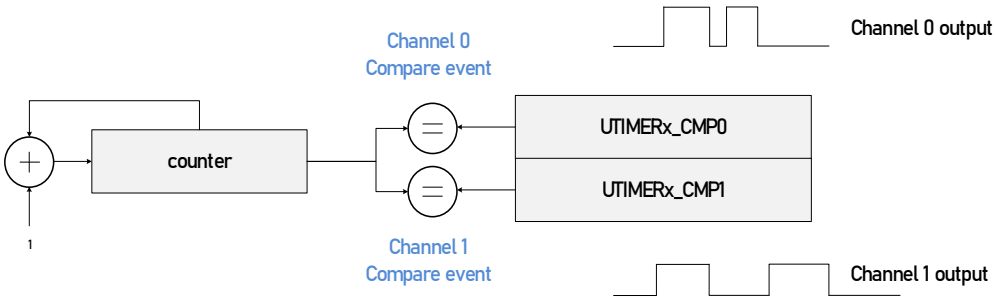
图 13-3 TIMER 通用计数器

计数器的计数时钟可以通过 TIMERx_CFG.XCLK_EN 进行配置，可以使用芯片内部的系统时钟（通常为芯片内部 96MHz PLL 时钟）或使用 Timer 通道 0/1 信号作为外部时钟进行计数。

计数器的计数时钟可以通过 TIMERx_CFG.CLK_DIV 进行分频，以降低计数器的计数频率。

13.2.4.2 捕获模式

捕获模式下，可以使用 Timer 来检测输入信号的上升/下降或者双沿，发生捕获事件（即输入信号电平变化）时，定时器计数值存入 TIMER_UNTx_CMP 寄存器，并产生捕获中断。计数器回零时，仍然会产生回零中断。



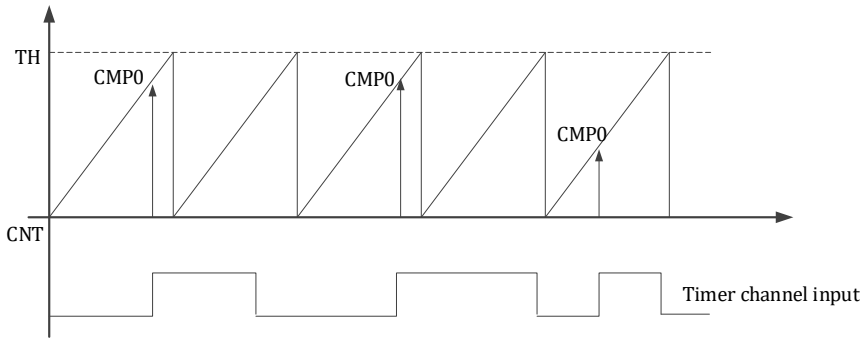


图 13-4 TIMER 捕获模式

如图 13-4 所示，定时器设置为上升沿捕获。在 CAP0/CAP1/CAP2 三个时刻点，捕获到输入信号发生上升沿变化，对应时刻点的定时器计数值将存入 TIMER_UNTx_CMP 寄存器中。

捕获模式下，通道 0/1 捕获的信号来源可以通过 TIMERx_CFG.SRC0 和 TIMERx_SRC1 进行设置，信号源可以设置为来自 IO 的通道信号，或来自模拟比较器，以及来自 IO 的两个通道信号的异或。

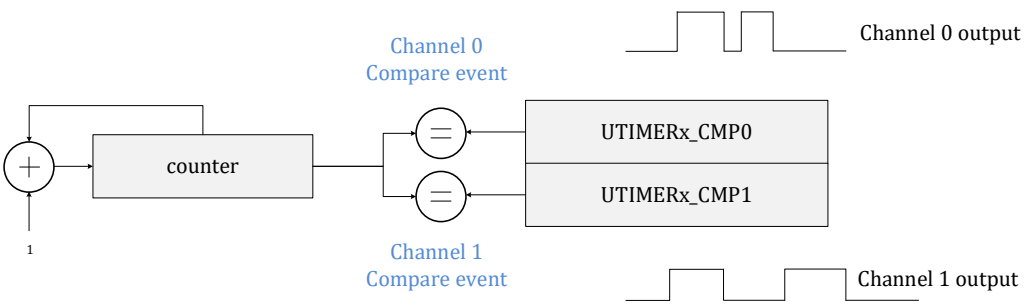
可以通过设置 TIMERx_CFG.CAP0_CLR_EN 或 TIMERx_CFG.CAP1_CLR_EN 使能 Timer 自动清零，即当通道 0/1 发生捕获事件时，TIMER_CNT 自动回零重新开始计数。这一功能便于 Timer 计算捕获信号的周期和占空比。

例如，如果设置了 Timer0 的通道 0/1 同时捕获同一个信号，通道 0 捕获上升沿，通道 1 捕获下降沿。且使能 CAP0_CLR_EN，即发生通道 0 捕获事件时 TIMER_CNT 自动回零。则当发生通道 0 捕获事件（CAP0）时，TIMERx_CMP0 记录的值为信号上一个周波的周期，TIMERx_CMP1 记录的值为信号上一个周波的高电平占空比。

同理，如果设置了 Timer0 的通道 0/1 同时捕获同一个信号，通道 0 捕获上升沿，通道 1 捕获下降沿。且使能 CAP1_CLR_EN，即发生通道 1 捕获事件时 TIMER_CNT 自动回零。则当发生通道 1 捕获事件（CAP1）时，TIMERx_CMP1 记录的值为信号上一个周波的周期，TIMERx_CMP0 记录的值为信号上一个周波的低电平占空比。

13.2.4.3 比较模式(边沿对齐 PWM)

比较模式下，计数器计数到 TIMERx_CMP 值时，产生比较中断。比较模式可以驱动一个比较脉冲发生，在回零时，向 IO 口输出一个电平（极性可配置），在比较事件发生时，电平翻转，向 IO 口输出另一个电平。计数器回零时，仍然会产生回零中断。设置 TIMERx_CMP0=0，可使得 Timer X 通道 0 为恒 1，设置 TIMERx_CMP0=TIMERx_TH+1，可使得 Timer 通道 0 为恒 0。



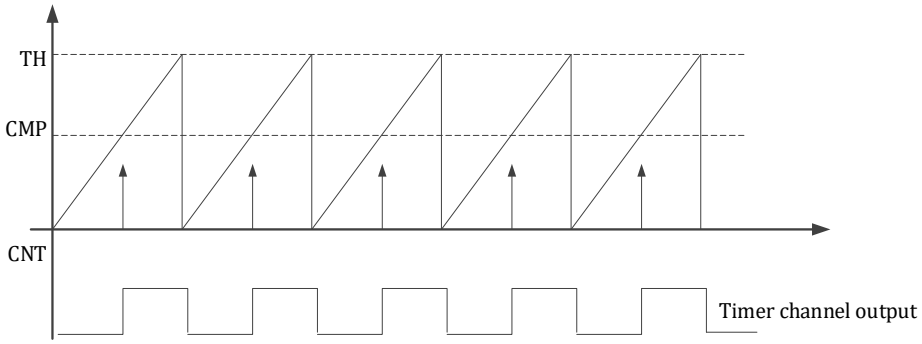


图 13-5 TIMER 比较模式

13.2.4.4 单次触发

在比较模式下，当 TIMERx_CFG.EN=0，即 Timer 没有开始计数时，通过向 TIMERx_CFG.ONE_TRIG 写入 1，可以触发 Timer 发出一个周期特定占空比的波形。结束后 Timer 回到空闲状态，通道不再动作。如果 TIMER 单次触发仍在计数周期内再次发生单次触发，TIMER 不会响应，仍继续完成本次周期计数。即在单次触发计数过程中发生的过于频繁的软件单次触发不被 TIMER 接受。如下图，第 3 次向 ONE_TRIG 写 1 无效，不会触发 TIMER 重新开始计数。

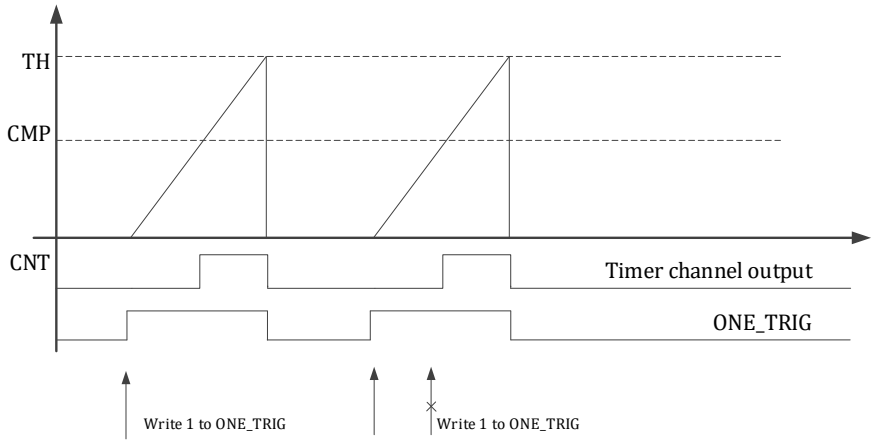


图 13-6 TIMER 单次触发

13.2.4.5 中心模式(互补 PWM)

除递增计数和递减外，计数器还可以设置为中心计数模式，即 TIMERx_CFG.CENTER=1。此时，计数器从 0 递增计数至 TH 后再递减计数回 0，开始下一个周期的中心计数。实际计数周期为 clk_freq*(2*TH+1)。中心计数模式可以用于输出互补 PWM 波形，通过合理设置 TIMERx_CMP0/1 可以产生上下管导通死区，死区时间=TIMERx_CMP1-TIMERx_CMP0。此外，需要合理设置通道的初始值以产生互补波形。Timer 在递增计数和递减计数期间都会有 TIMERx_CNT=TIMER_CMP0/1 的时刻，但只有递增计数阶段会产生中断事件。

此外，Timer0 支持 FAIL 事件及时关断 PWM 输出。发生 FAIL 事件时，Timer 通道 0/1 输出值由 TIMER0_IO.CH0_DEFAULT 和 TIMER0_IO.CH1_DEFAULT 指定，同时 TIMER0_IO.MOE 被 FAIL 事件清零，软件将 MOE 置 1 后，Timer 通过恢复输出 PWM 波形。

Timer0 同时配备了影子寄存器 TIMER0_STH/ TIMER0_SCMP0/ TIMER0_SCMP1。当 TIMER0_CFG.SHADOW=1 时，表示使用影子寄存器作为计数器的动作指示，TIMER0_TH/ TIMER0_CMP0/ TIMER0_CMP1 仅仅作为预装载寄存器，向 TIMER0_TH/ TIMER0_CMP0/ TIMER0_CMP1 写入仅仅修改预装载值，不改变影子寄存器值。TIMER0 的影子寄存器可以通过向 TIMER0_CFG.UPDATE 写 1 进行软件更新；或者可以等待 TIMER 过零事件自动更新。

当 TIMER0_CFG.SHADOW=0 时，与普通 Timer 相同，仍使用 TIMER0_TH/ TIMER0_CMP0/ TIMER0_CMP1 作为计数器的动作指示。

读取 TIMER0_TH/ TIMER0_CMP0/ TIMER0_CMP1 时，总是返回影子寄存器值 TIMER0_STH/ TIMER0_SCMP0/ TIMER0_SCMP1。

表 13-1 TIMER0 影子寄存器对应关系

TIMER0_CFG.SHADOW 设定值	0	1
TIMER0_CNT 命中 TH	TIMER0_TH	TIMER0_STH
TIMER0_CNT 命中 CMP0	TIMER0_CMP0	TIMER0_SCMP0
TIMER0_CNT 命中 CMP1	TIMER0_CMP1	TIMER0_SCMP1

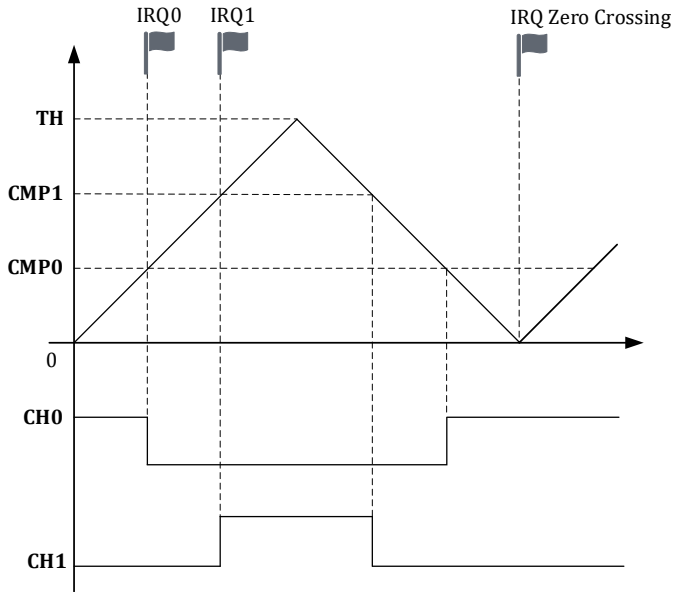


图 13-7 TIMER 中心计数模式

TIMER0/1/2/3 均支持中心计数模式进行互补 PWM 输出；但只有 TIMER0 配备了影子寄存器和 FAIL 停机机制。即只有 TIMER0 存在 TIMER0_IO 寄存器。

13.2.5 外部事件

Timer 支持与其他 Timer 通道、MCPWM 的 ADC 触发事件以及 CLU 输出进行联动。外部信号可以作为 Timer 的外部启动信号、外部时钟信号、外部重置信号、外部门控信号。

外部事件在 Timer 内部只有一种解释，因此只能使用一种外部事件功能，以上 4 种功能不可以同时使用，亦即 TIMERx_CFG.ETON，TIMERx_CFG.XCLK_EN，TIMERx_CFG.RL_EN，TIMERx_CFG.GATE_EN 4 个设置位只能有一个为 1。

以下说明均以 TIMER 正向递增计数为例，递减计数同理。

外部事件通过 TIMERx_EVT 进行选择。

13.2.5.1.1 外部启动信号

当 TIMERx_CFG.EN=0 且 TIMERx_CFG.ETON=1 时，TIMERx 在发生外部事件前 CNT 停留在 0，在发生外部事件时启动计数。外部启动信号被处理为一个脉冲信号后进行 TIMER 的启动触发。

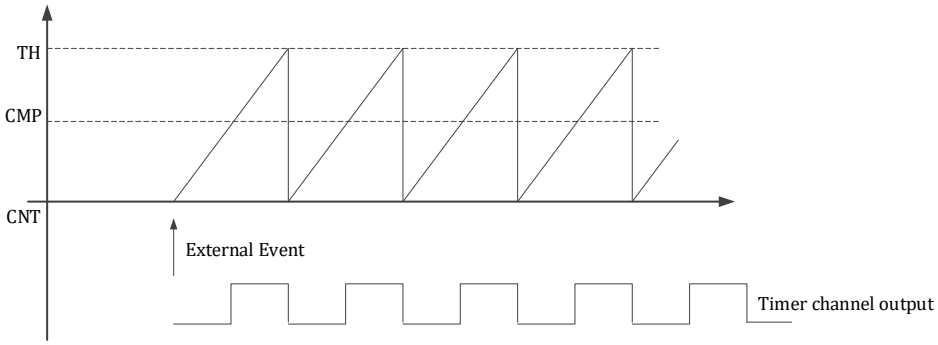


图 13-8 TIMER 外部事件触发启动

外部启动也可以与单次触发进行联合使用，此时软件需要先配置 ETON=1，然后配置 ONE_TRIG=1，等待外部事件触发后，TIMER 进行一个周期的计数，周期结束后 ONE_TRIG 被硬件清零。

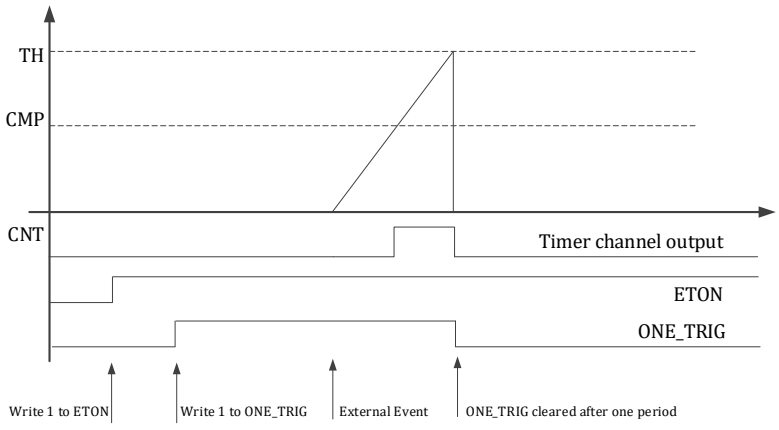


图 13-9 TIMER 外部事件触发单次（单周期）计数

13.2.5.1.2 外部时钟信号

通过设置 TIMERx_CFG.XCLK_EN=1，TIMER 可以使用外部信号作为计数时钟。外部时钟的选择通过 TIMERx_EVT 进行设置。

当外部信号出现上升沿时，TIMERx_CNT 递增（或递减）。特别地，如果使用 TIMERm 的通道输入作为 TIMERx 的外部时钟，则 TIMERm 的通道输入为受到 TIMERm 滤波后的信号。

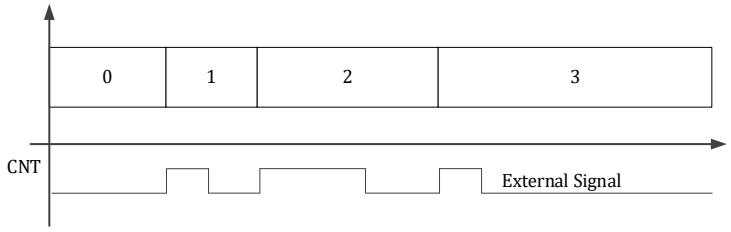


图 13-10 TIMER 使用外部时钟计数

13.2.5.1.3 外部重置信号

通过设置 TIMERx_CFG.RL_EN=1，TIMER 可以使用外部信号作为重置信号。外部重置信号的选择通过 TIMERx_EVT 进行设置。如果 TIMER 设置为递增计数或中心计数，发生外部信号重置时，TIMERx_CNT 被重置为 0；如果 TIMER 设置为递减计数，发生外部信号重置时，TIMERx_CNT 被重置为 TIMERX_TH。外部重置信号被处理为一个脉冲信号后对 TIMER 进行重置。

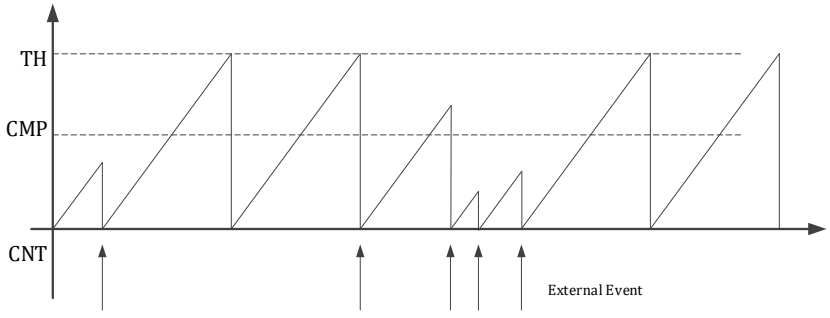


图 13-11 TIMER 外部信号重置功能

13.2.5.1.4 外部门控信号

通过设置 TIMERx_CFG.GATE_EN=1，TIMER 可以使用外部信号作为门控信号。外部门控信号的选择通过 TIMERx_EVT 进行设置。门控信号电平为高时，TIMER 进行计数，电平为低时，TIMER 停止计数，TIMERx_CNT 保持不变。由于门控信号需要为电平信号，因此 MCPWM 的 ADC 触发信号无法作为外部门控信号使用。特别地，如果使用 TIMERm 的通道输入作为 TIMERx 计数的门控信号，通道输入信号受到 TIMERm 的滤波影响。

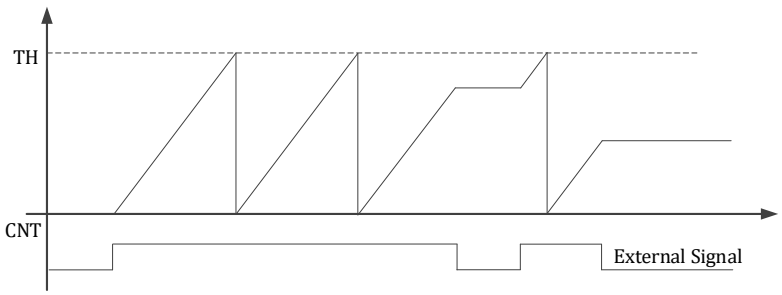


图 13-12 TIMER 外部信号门控功能

以 TIMER2 通道 1 作为 TIMER3 的外部事件信号为例，需要通过 TIMER2_CFG.SRC1 设置 TIMER2 的通道 1 信号来自 IO TIM2_CH1，通过 TIMER2_CFG.CH1_MODE 设置 TIMER2 通道 1 为捕获模式。此外，需要注意 TIMER2_FLT 设置，对 TIMER2 通道 1 输入信号仍有滤波作用。GPIO 需要做相应设置以允许信号通过 GPIO 进入 TIMER2 通道 1。最后通过 TIMER3_EVT 设置 TIMER2 通道 1 作为 TIMER3 外部信号的作用，通常只设置 TIMER3_CFG.ETON，TIMER3_CFG.XCLK_EN，TIMER3_CFG.RL_EN，TIMER3_CFG.GATE_EN 4 个设置位只能有一个为 1。

13.2.6 ADC 触发

Timer0/1 的比较事件（CMP0/1）可作为 ADC 采样触发事件。

13.2.7 编码器

编码器接口支持正交编码信号、符号加脉冲信号、CW/CCW 双脉冲信号三种模式。

其中 QEP0 的两个输入信号 T1/T2 分别来自 Timer2 Channel0/1 对应的 GPIO 输入，并经过 Timer 内部的滤波；QEP1 的 T1/T2 信号分别来自 Timer3 Channel0/1 对应的 GPIO 输入，并经过 Timer 内部的滤波。开启编码器功能时并不影响 Timer 功能的正常使用。

编码器的输入 T1/T2/Z 信号均受对应的 Timer 滤波系数控制，举例来说，QEP0 的 Z 信号受到 TIMER2_FLT 滤波时间控制，QEP1 的 Z 信号受到 TIMER3_FLT 滤波时间控制。

由于编码器复用的是 Timer 通道输入，且受 Timer 通道滤波影响，因此使用编码器，需要开启对应的 SYS_CLK_FEN 中对应的 Timer 时钟使能。如使用 QEP0 需要开启 Time2 外设时钟使能使用 QEP1 需要开启 Time3 外设时钟使能。

13.2.7.1 正交编码信号

正交编码信号多用于计数编码器圈数，输入为 T1/T2 两个信号，支持下表中两个模式。

概括来讲，T1/T2 的跳变沿会导致计数器递增或递减。而计数器计数方向（递增或递减）由跳变信号之外的另一个稳态信号的电平高低决定。

如果 T1 发生了上升沿跳变，则看 T2 是高电平还是低电平，如果是高电平则计数器递减，如果是低电平计数器递增，T1 下降沿计数器变化相反。

如果 T2 发生了上升沿跳变，则看 T1 是高电平还是低电平，如果是高电平则计数器递增，如

果是低电平计数器递减，T2 下降沿计数器变化相反。

逻辑关系如下公式所示：

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

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

表 13-2 编码器正交编码工作模式

计数模式	T1/T2 电平状态(稳态信号)	T1 变化边沿状态		T2 变化边沿状态	
		上升沿	下降沿	上升沿	下降沿
仅 T1 计数	T2 高	递减	递增	不计数	不计数
	T2 低	递增	递减	不计数	不计数
T1/T2 都计数	T2 高	递减	递增	不计数	不计数
	T2 低	递增	递减	不计数	不计数
	T1 高	不计数	不计数	递增	递减
	T1 低	不计数	不计数	递减	递增

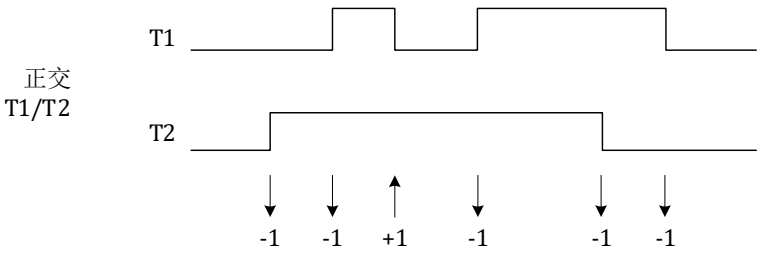


图 13-13 编码器只在 T1 时刻计数的正交编码信号计数情况

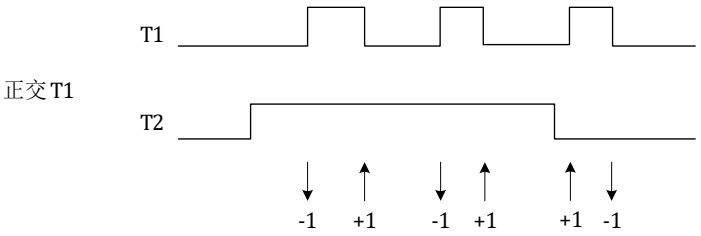


图 13-14 编码器在 T1 或 T2 时刻计数的正交编码信号计数情况

13.2.7.2 符号加脉冲信号

这种工作模式下，T1 为脉冲信号，T2 为符号信号。T1 的边沿触发计数，T2 电平控制计数方向，高则递增，低则递减。可以配置仅 T1 上升沿计数还是 T1 上升下降沿都计数。

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

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

表 13-3 编码器符号加脉冲工作模式

计数模式	T2 电平状态(稳态信号)	T1 变化边沿状态	
		上升沿	下降沿
仅 T1 上升沿	高	递增	不计数
	低	递减	不计数
T1 上升下降沿	高	递增	递减
	低	递减	递增

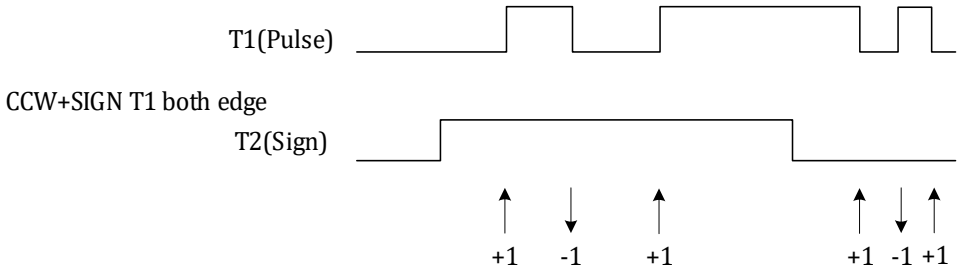


图 13-15 编码器在 T1 上升下降沿都计数的符号加脉冲信号计数情况

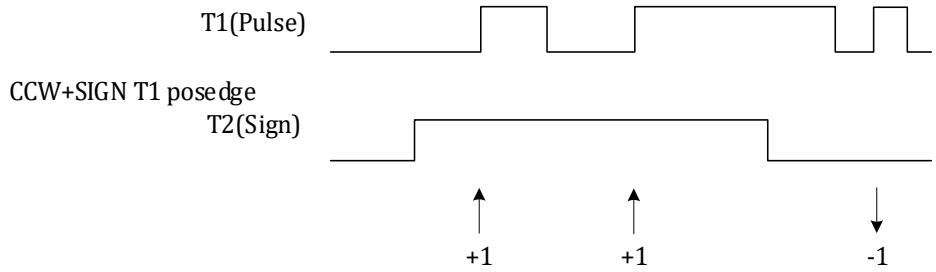


图 13-16 编码器在仅 T1 上升沿计数的符号加脉冲信号计数情况

13.2.7.3 CCW/CW 双脉冲信号

在 T1 跳变时计数器递增，在 T2 跳变时计数器递减。可以配置计数器仅在上升沿变化或者在上升下降沿都变化。以下式表示

Counter Up = 1 @ (T1 triggering edges)

Counter Down = 1 @ (T2 triggering edges)

表 13-4 编码器 CCW/CW 双脉冲工作模式

计数模式	变化边沿状态			
	T1 上升沿	T1 下降沿	T2 上升沿	T2 下降沿
T1/T2 上升沿	递增	不计数	递减	不计数
T1/T2 上升下降沿	递增	递增	递减	递减

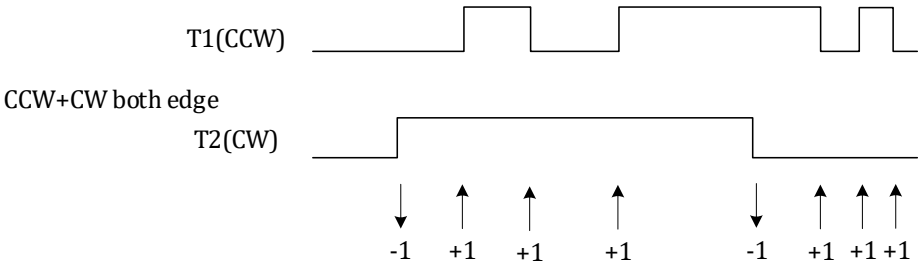


图 13-17 编码器仅在 T1/T2 上升沿计数的 CCW/CW 双脉冲信号计数情况

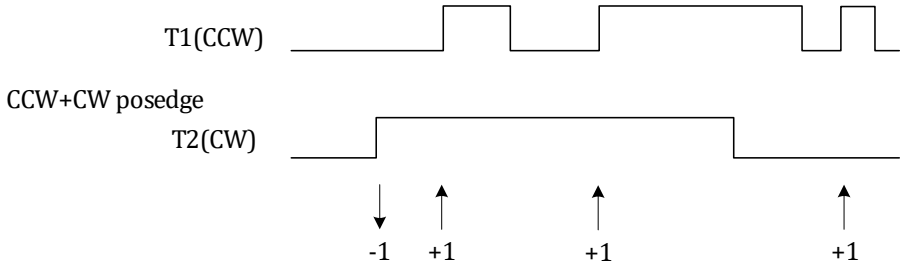


图 13-18 编码器在 T1/T2 上升下降沿计数的 CCW/CW 双脉冲信号计数情况

13.3 寄存器

13.3.1 地址分配

Timer0 在芯片中的基地址是 0x4001_0600

表 13-5 Timer0 寄存器地址分配

名称	偏移	描述
TIMER0_CFG	0x00	Timer0 配置寄存器
TIMER0_TH	0x04	Timer0 计数门限寄存器
TIMER0_CNT	0x08	Timer0 计数值寄存器
TIMER0_CMP0	0x0C	Timer0 比较/捕获寄存器 0
TIMER0_CMP1	0x10	Timer0 比较/捕获寄存器 1
TIMER0_EVT	0x14	Timer0 外部事件选择寄存器
TIMER0_FLT	0x18	Timer0 滤波控制寄存器
TIMER0_IE	0x1C	Timer0 中断使能寄存器
TIMER0_IF	0x20	Timer0 中断标志寄存器
TIMER0_IO	0x24	Timer0 IO 控制寄存器

Timer1 在芯片中的基地址是 0x4001_0700

表 13-6 Timer1 寄存器地址分配

TIMER1_CFG	0x00	Timer1 配置寄存器
TIMER1_TH	0x04	Timer1 计数门限寄存器
TIMER1_CNT	0x08	Timer1 计数值寄存器
TIMER1_CMP0	0x0C	Timer1 比较/捕获寄存器 0
TIMER1_CMP1	0x10	Timer1 比较/捕获寄存器 1
TIMER1_EVT	0x14	Timer1 外部事件选择寄存器
TIMER1_FLT	0x18	Timer1 滤波控制寄存器
TIMER1_IE	0x1C	Timer1 中断使能寄存器
TIMER1_IF	0x20	Timer1 中断标志寄存器

Timer2 在芯片中的基地址是 0x4001_0800

表 13-7 Timer2 寄存器地址分配

TIMER2_CFG	0x00	Timer2 配置寄存器
TIMER2_TH	0x04	Timer2 计数门限寄存器
TIMER2_CNT	0x08	Timer2 计数值寄存器
TIMER2_CMP0	0x0C	Timer2 比较/捕获寄存器 0
TIMER2_CMP1	0x10	Timer2 比较/捕获寄存器 1
TIMER2_EVT	0x14	Timer2 外部事件选择寄存器
TIMER2_FLT	0x18	Timer2 滤波控制寄存器
TIMER2_IE	0x1C	Timer2 中断使能寄存器
TIMER2_IF	0x20	Timer2 中断标志寄存器

Timer3 在芯片中的基地址是 0x4001_0900

表 13-8 Timer3 寄存器地址分配

TIMER3_CFG	0x00	Timer3 配置寄存器
TIMER3_TH	0x04	Timer3 计数门限寄存器
TIMER3_CNT	0x08	Timer3 计数值寄存器
TIMER3_CMP0	0x0C	Timer3 比较/捕获寄存器 0
TIMER3_CMP1	0x10	Timer3 比较/捕获寄存器 1
TIMER3_EVT	0x14	Timer3 外部事件选择寄存器
TIMER3_FLT	0x18	Timer3 滤波控制寄存器
TIMER3_IE	0x1C	Timer3 中断使能寄存器
TIMER3_IF	0x20	Timer3 中断标志寄存器

Timer0/1/2/3 不同之处在于 Timer0/1 计数器相关寄存器为 16 位宽，而 Timer2/3 计数器相关寄存器为 32 位宽，即 TIMERx_TH，TIMERx_CMP0，TIMERx_CMP1 的位宽不同。

且 Timer0 的 TH/CMP0/CMP1 存在影子寄存器，可以选择是否启用影子寄存器，如果不启用，则 Timer0 和 Timer1 相同。如果启用影子寄存器，则向以上 3 个寄存器写入的仅仅是预装载值，当 Timer0 发生过零事件时，预装载值被加载进入影子寄存器；读取 TIMER0_TH/CMP0/CMP1 读

取的影子寄存器的值。

QEP0 在芯片中的基地址是 0x4001_0A00

表 13-9 QEP0 寄存器地址分配

QEP0_CFG	0x00	QEP0 配置寄存器
QEP0_TH	0x04	QEP0 计数门限寄存器
QEP0_CNT	0x08	QEP0 计数值寄存器
QEP0_IE	0x0C	QEP0 中断使能寄存器
QEP0_IF	0x10	QEP0 中断标志寄存器

QEP1 在芯片中的基地址是 0x4001_0B00

表 13-10 QEP1 寄存器地址分配

QEP1_CFG	0x00	QEP1 配置寄存器
QEP1_TH	0x04	QEP1 计数门限寄存器
QEP1_CNT	0x08	QEP1 计数值寄存器
QEP1_IE	0x0C	QEP1 中断使能寄存器
QEP1_IF	0x10	QEP1 中断标志寄存器

13.3.2 TIMER0 寄存器

13.3.2.1 TIMER0_CFG Timer0 配置寄存器

地址:0x4001_0600

复位值:0x0

表 13-11 Timer0 配置寄存器 TIMER0_CFG

31	30	29	28	27	26	25	24	23	22 21 20	19	18	17	16
EN		CAP1_CLR_EN	CAP0_CLR_EN	UPDATE	SHADOW	ONE_TRIG	CENTER	DIR	CLK_DIV	ETON	GATE_EN	RL_EN	XCLK_EN
RW		RW	RW	WO	RW	RW	RW	RW	RW	RW	RW	RW	RW
0		0	0	0	0	0	0	0	0	0	0	0	0

15 14 13 12	11	10	9	8	7 6 5 4	3	2	1	0
SRC1	CH1_POL	CH1_MODE	CH1_FE_CAP_EN	CH1_RE_CAP_EN	SRC0	CH0_POL	CH0_MODE	CH0_FE_CAP_EN	CH0_RE_CAP_EN
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0001	0	0	0	0	0	0	0	0	0

位置	位名称	说明

[31]	EN	Timer 模块整体使能，高有效
[30]		未使用
[29]	CAP1_CLR_EN	当发生 CAP1 捕获事件时，清零 Timer 计数器，高有效
[28]	CAP0_CLR_EN	当发生 CAP0 捕获事件时，清零 Timer 计数器，高有效
[27]	UPDATE	软件更新 启用影子寄存器时，向 UPDATE 写 1 将 TH/CMP0/CMP1 的预装载值加载到影子寄存器 STH/SCMP0/SCMP1 之中。自动清零，读回恒为 0。写 0 无作用。 **推荐使用 TIMER0_CFG|=BIT27;的方式进行手动更新影子寄存器的操作，防止手动更新时误将其他配置清零。**
[26]	SHADOW	0: 不启用影子寄存器，软件写入 TH/CMP0/CMP1 时直接更新对应的影子寄存器 1: 启用影子寄存器，软件写入 TH/CMP0/CMP1 时进仅仅更新预装载值，当 Timer 发生过零事件时才将预装载值写入影子寄存器
[25]	ONE_TRIG	单次发送模式，此位需要在 Timer 比较模式下使用，且 EN 需设置为 0 ETON 为 0 时，软件向 ONE_TRIG 写 1 触发 Timer 发送一个周期的特定占空比的脉冲，Timer 计数一个周期后停止。此位在 Timer 计数的当前周期内读为 1，Timer 停止后，自动清零。 ETON 为 1 时，即使软件配置 ONE_TRIG=1，也需要等待外部事件触发，Timer 计数一个周期后停止。ONE_TRIG 位在外部触发 Timer 计数一个周期停止后被硬件清零。
[24]	CENTER	中心计数模式使能 0: Timer 向上从 0 计数至 TH，然后回 0，或 Timer 向下从 TH 计数至 0，然后回到 TH 1: Timer 向上从 0 计数至 TH，然后向下计数至 0
[23]	DIR	0: 0->TH 递增计数，1：TH->递减计数 此位在 CENTER=1 时只读，用于指示当前计数方向
[22:20]	CLK_DIV	Timer 计数器频率配置，计数器计数频率是系统主时钟频率的 2^{CLK_DIV} 分频。默认值为 0，不分频。 0: 1 分频 1: 2 分频 2: 4 分频 3: 8 分频 4: 16 分频 5: 32 分频 6: 64 分频 7: 128 分频
[19]	ETON	Timer 计数器外部启动使能 0: 自动运行 1:等待外部事件触发计数，外部触发信号根据 TIMER0_EVT.EVT_SRC 进行选择 需要注意的是，使用外部触发也需要设置 TIMERx_CFG.EN=1，除非是外部信号进行的单次触发，即 TIMERx_CFG.ONE_TRIG=1。
[18]	GATE_EN	Timer 暂停使能

		0：不暂停 1：当外部信号为低时，Timer 暂停计数，外部信号根据 TIMER0_EVT.EVT_SRC 进行选择
[17]	RL_EN	Timer 重装使能 0：禁用外部事件重装，Timer 向上计数至 TH 回 0，或向下计数值 0 回到 TH 1：使能外部事件重装，重装信号根据 TIMER0_EVT.EVT_SRC 进行选择，当向上计数时，发生外部事件，Timer 重装为 0,；当向下计数时，发生外部事件，Timer 重装为 TH。
[16]	XCLK_EN	Timer 时钟源 0: 芯片内部时钟 1: 外部时钟，时钟源根据 TIMER0_EVT.EVT_SRC 进行选择，当使用外部时钟时，CLK_DIV 不再起作用，即时钟不再进行分频
[15:12]	SRC1	Timer 捕获模式通道 1 信号来源。默认为 3'h1。 0: Timer 通道 0，来自芯片 GPIO（参见 Datasheet 及应用配置） 1: Timer 通道 1，来自芯片 GPIO（参见 Datasheet 及应用配置） 2: CLU0 输出 3: CLU1 输出 4: CLU2 输出 5: CLU3 输出 6: 比较器 0 的输出 7: 比较器 1 的输出 8: 比较器 2 的输出 9: Timer 通道 0 和 1 的异或
[11]	CH1_POL	Timer 通道 1 在比较模式下的输出极性控制，当计数器 CNT<CMP1 时的输出值。
[10]	CH1_MODE	Timer 通道 1 的工作模式选择，默认值为 0。 0: 比较模式。输出方波，在 Timer 通道 1 计数器计数值等于 0 或等于 Timer 比较捕获寄存器值时，IO 发生翻转。 1: 捕获模式。当 Timer 通道 1 输入信号发生捕获事件时，将计数器计数值存入 Timer 通道 1 比较捕获寄存器。
[9]	CH1_FE_CAP_EN	Timer 通道 1 下降沿捕获事件使能。1:使能；0:关闭。 Timer 通道 1 输入信号发生 1→0 跳变被视为捕获事件。下降沿事件使能可以与上升沿事件使能并存。
[8]	CH1_RE_CAP_EN	Timer 通道 1 上升沿捕获事件使能。1:使能；0:关闭。 Timer 通道 1 输入信号发生 0→1 跳变被视为捕获事件。上升沿事件使能可以与下降沿事件使能并存。
[7:4]	SRC0	Timer 捕获模式通道 0 信号来源。默认为 3'h0。 0: Timer 通道 0，来自芯片 GPIO（参见 Datasheet 及应用配置） 1: Timer 通道 1，来自芯片 GPIO（参见 Datasheet 及应用配置） 2: CLU0 输出 3: CLU1 输出 4: CLU2 输出 5: CLU3 输出 6: 比较器 0 的输出

		7: 比较器 1 的输出 8: 比较器 2 的输出 9: Timer 通道 0 和 1 的异或
[3]	CH0_POL	Timer 通道 0 在比较模式下的输出极性控制:当计数器 CNT<CMP0 时的输出值。时的输出值。
[2]	CH0_MODE	Timer 通道 0 的工作模式选择，默认值为 0。 0: 比较模式。输出方波，在 Timer 通道 0 计数器计数值等于 0 或等于 Timer 比较捕获寄存器值时，IO 发生翻转。 1: 捕获模式。当 Timer 通道 0 输入信号发生捕获事件时，将计数器计数值存入 Timer 通道 0 比较捕获寄存器。
[1]	CH0_FE_CAP_EN	Timer 通道 0 下降沿捕获事件使能。1:使能；0:关闭。 Timer 通道 0 输入信号发生 1→0 跳变被视为捕获事件。下降沿事件使能可以与上升沿事件使能并存。
[0]	CH0_RE_CAP_EN	Timer 通道 0 上升沿捕获事件使能。1:使能；0:关闭。 Timer 通道 0 输入信号发生 0→1 跳变被视为捕获事件。上升沿事件使能可以与下降沿事件使能并存。

当使用外部时钟计数时，也需要设置 TIMERx_CFG.TON=1，且需要开启 SYS_CLK_FEN 中对于 Timer 时钟的使能。使用外部时钟计数时，也需要设置 TIMERx_TH。

通常外部事件仅仅作为外部启动、外部门控、外部时钟和外部清零的一个用途使用，即通常 TIMERx_CFG[19:16]中仅设置 1bit 为 1。

如果设置 SRC0 为 4'h8 捕获两通道异或值，（通常为捕获编码器正交的两个通道），需要同时设置 CMP0_CLR_EN=1，CH0_FE_CAP_EN=1，CH0_RE_CAP_EN=1，CH0_MODE=1，即捕获上升沿、下降沿，且每次有信号边沿都清零计数器。CMP0 即为捕获的两次信号边沿之间的计数值。

13.3.2.2 TIMER0_TH Timer0 门限寄存器

地址:0x4001_0604

复位值:0x0

表 13-12 Timer0 门限寄存器 TIMER0_TH

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

位置	位名称	说明
[31:16]		未使用
[15:0]	TH	Timer 计数器计数门限。向上计数从 0 计数到 TH 值后回 0，或向下计数从 TH 计数到 0 后回到 TH

13.3.2.3 TIMER0_CNT Timer0 计数寄存器

地址:0x4001_0608

复位值:0x0

表 13-13 Timer0 计数寄存器 TIMER0_CNT

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

位置	位名称	说明
[31:16]		未使用
[15:0]	CNT	Timer 0 计数器当前计数值。写操作可以写入新的计数值。

注意:写入 TIMER0_CNT 前需要先通过 SYS_CLK_FEN.TIMER0_CLK_EN 开启 Timer0 时钟。

13.3.2.4 TIMER0_CMP0 Timer0 通道 0 比较捕获寄存器

地址:0x4001_060C

复位值:0x0

表 13-14 Timer0 通道 0 比较捕获寄存器 TIMER0_CMP0

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

位置	位名称	说明
[31:16]		未使用
[15:0]	CMP0	Timer 通道 0 工作在比较模式时，当计数器计数值等于 CMP0 时，发生比较事件。 Timer 通道 0 工作在捕获模式时，发生捕获事件时的计数器计数值存入 CMP0 寄存器。

设置 CMP0=0，可使得通道 0 恒为!CH0_POL，设置 CMP0=TH+1，可使得通道 0 恒为 CH0_POL。

13.3.2.5 TIMER0_CMP1 Timer0 通道 1 比较捕获寄存器

地址:0x4001_0610

复位值:0x0

表 13-15 Timer0 通道 1 比较捕获寄存器 TIMER0_CMP1

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

位置	位名称	说明
[31:16]		未使用
[15:0]	CMP1	Timer 通道 1 工作在比较模式时，当计数器计数值等于 CMP1 时，发生比较事件。 Timer 通道 1 工作在捕获模式时，发生捕获事件时的计数器计数值存入 CMP1 寄存器。

设置 CMP1=0，可使得通道恒为!CH1_POL，设置 CMP1=TH+1，可使得通道恒为 CH1_POL。

13.3.2.6 TIMER0_EVT Timer0 外部事件选择寄存器

地址:0x4001_0614

复位值:0x0

表 13-16 Timer0 外部事件选择寄存器 TIMER0_EVT

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

位置	位名称	说明
[31:4]		未使用
[3:0]	EVT_SRC	Timer 外部事件选择寄存器，本寄存器需要配合 TIMER0_CFG.ETON，TIMER0_CFG.GATE_EN，TIMER0_CFG.RL_EN，TIMER0_CFG.XCLK_EN 使用。 通常 4 个设置位不同时使用。 当设置 ETON=1 时，根据本寄存器选择触发 Timer 计数的事件。 需要注意的是，Timer 自身的比较事件无法触发 Timer 进行计数。当 ETON=1 时，触发信号的上升沿触发 Timer 开始计数，用作触发源的 Timer 需要工作在比较模式，无须从外部输入信号至 Timer 通道 0:不可用 1:不可用 2:TIMER1 通道 0 比较/捕获事件 3:TIMER1 通道 1 比较/捕获事件 4:TIMER2 通道 0 比较/捕获事件 5:TIMER2 通道 1 比较/捕获事件 6:TIMER3 通道 0 比较/捕获事件 7:TIMER3 通道 1 比较/捕获事件 8:CLU0 输出上升沿 9:CLU1 输出上升沿 10:CLU2 输出上升沿 11:CLU3 输出上升沿 12:MCPWM TADC[0]比较事件 13:MCPWM TADC[1]比较事件 14:MCPWM TADC[2]比较事件

		15:MCPWM TADC[3]比较事件 当 GATE_EN=1，且外部信号为高时 Timer 计数，外部信号为低时，Timer 暂停，当 GATE_EN=1 时，Timer 使用对应外部信号的电平信号作为计数门控。GATE_EN 仅支持配合 0~11 的外部事件使用，当 EVT_SRC 选择为 0~7 时，外部信号为 Timer 通道输入信号经过对应 Timer 滤波后的信号，作为门控信号的 Timer 通道需要配置为输入使能。 0:TIMER0 通道 0 滤波后输入信号 1:TIMER0 通道 1 滤波后输入信号 2:TIMER1 通道 0 滤波后输入信号 3:TIMER1 通道 1 滤波后输入信号 4:TIMER2 通道 0 滤波后输入信号 5:TIMER2 通道 1 滤波后输入信号 6:TIMER3 通道 0 滤波后输入信号 7:TIMER3 通道 1 滤波后输入信号 8:CLU0 输出信号 9:CLU1 输出信号 10:CLU2 输出信号 11:CLU3 输出信号 当 RL_EN 为高时，Timer 使用外部信号作为重装载信号 作为重装载信号的 Timer 通道需要配置为输入使能 0:TIMER0 通道 0 比较/捕获事件 1:TIMER0 通道 1 比较/捕获事件 2:TIMER1 通道 0 比较/捕获事件 3:TIMER1 通道 1 比较/捕获事件 4:TIMER2 通道 0 比较/捕获事件 5:TIMER2 通道 1 比较/捕获事件 6:TIMER3 通道 0 比较/捕获事件 7:TIMER3 通道 1 比较/捕获事件 8:CLU0 输出 9:CLU1 输出 10:CLU2 输出 11:CLU3 输出 12:MCPWM TADC[0]比较事件 13:MCPWM TADC[1]比较事件 14:MCPWM TADC[2]比较事件 15:MCPWM TADC[3]比较事件 当 XCLK_EN 为高时，Timer 使用外部信号作为 Timer 外部时钟，Timer 在时钟上升沿进行计数动作。如果使用 Timer 通道作为外部时钟，通道信号受到对应 Timer 滤波设置影响。 0:TIMER0 通道 0 输入信号 1:TIMER0 通道 1 输入信号 2:TIMER1 通道 0 输入信号

		3:TIMER1 通道 1 输入信号 4:TIMER2 通道 0 输入信号 5:TIMER2 通道 1 输入信号 6:TIMER3 通道 0 输入信号 7:TIMER3 通道 1 输入信号 8:CLU0 输出 9:CLU1 输出 10:CLU2 输出 11:CLU3 输出 12:MCPWM TADC[0]比较事件 13:MCPWM TADC[1]比较事件 14:MCPWM TADC[2]比较事件 15:MCPWM TADC[3]比较事件

13.3.2.7 TIMER0_FLT Timer0 滤波控制寄存器

地址:0x4001_0618

复位值:0x0

表 13-17 Timer0 滤波控制寄存器 TIMER0_FLT

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

位置	位名称	说明
[31:8]		未使用
[7:0]	FLT	通道 0/1 信号滤波宽度选择。取值范围 0~255。 FLT 为 0 时，不对通道进行滤波。 FLT 不为 0 时，对通道信号进行滤波:滤波宽度为 FLT×8。当通道信号电平稳定超过 FLT×8 个系统时钟周期宽度时，滤波器输出更新；否则，滤波器保持当前的输出不变。

注意，以上滤波器的工作时钟与对应的 Timer 的分频后的工作时钟为相同时钟，受 TIMERx_CFG.CLK_DIV 的分频系数控制。

假设 TIMERx_FLT = 0x6；TIMERx_CFG.CLK_DIV = 0x2; 则 Timer 的运行时钟相对系统时钟进行了 4 分频。通道输入信号需要经过 8×6 倍 Timer 的运行时钟的滤波，亦即 8×6×4 倍系统时钟的滤波。

13.3.2.8 TIMER0_IE Timer0 中断使能寄存器

地址:0x4001_061C

复位值:0x0

表 13-18 Timer0 中断使能寄存器 TIMER0_IE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				FAIL_RE	ZC_RE	CH1_RE	CH0_RE					FAIL_IE	ZC_IE	CH1_IE	CH0_IE
				RW	RW	RW	RW					RW	RW	RW	RW
				0	0	0	0					0	0	0	0

位置	位名称	说明
[31:12]		未使用
[11]	FAIL_RE	Timer FAIL 事件 DMA 请求使能，高电平有效。
[10]	ZC_RE	Timer 计数器过 0 DMA 请求使能，高电平有效。
[9]	CH1_RE	Timer 通道 1 比较/捕获 DMA 请求使能，高电平有效。
[8]	CH0_RE	Timer 通道 0 比较/捕获 DMA 请求使能，高电平有效。
[7:4]		未使用
[3]	FAIL_IE	Timer FAIL 事件中断使能，高电平有效。
[2]	ZC_IE	Timer 计数器过 0 中断使能，高电平有效。
[1]	CH1_IE	Timer 通道 1 比较/捕获中断使能，高电平有效。
[0]	CH0_IE	Timer 通道 0 比较/捕获中断使能，高电平有效。

DMA 请求事件，与中断为相同事件标志，DMA 请求被 DMA 受理后，中断标志会被 DMA 硬件自动清除，无需 CPU 软件干预。需要注意的是，通常开启某个事件的 DMA 请求则关闭对应的中断使能。

13.3.2.9 TIMER0_IF Timer0 中断标志寄存器

地址:0x4001_0620

复位值:0x0

表 13-19 Timer0 中断标志寄存器 TIMER0_IF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												FAIL_IF	ZC_IF	CH1_IF	CH0_IF
												RW1C	RW1C	RW1C	RW1C
												0	0	0	0

位置	位名称	说明
[31:4]		未使用
[3]	FAIL_IF	Timer Fail 事件中断标志。高电平有效，写 1 清 0
[2]	ZC_IF	Timer 计数器过 0 中断标志。高电平有效，写 1 清 0

[1]	CH1_IF	Timer 通道 1 比较/捕获中断标志。高电平有效，写 1 清 0
[0]	CH0_IF	Timer 通道 0 比较/捕获中断标志。高电平有效，写 1 清 0

在比较模式下，当 Timer 设置为中心计数（0->TH->0），即 TIMERx_CFG.CENTER=1 时。CH0_IF 和 CH1_IF 只有在 Timer 从 0 递增计数至 TH 的过程中，当 TIMERx_CNT=TIMERx_CMP0/1 时置位；在 Timer 从 TH 递减计数到 0 的过程中不会置位。

当 TIMERx_CFG.DIR=0

中断标志写 1 清零，一般不建议用如下|=方式清零，因为|=是先读取中断标志，将对应位改为 1 再写入清零，如果同时有其他中断标志置位，会被一起清零，而这通常不是软件所期望的。例如，如下写法本意是清零 ZC_IF，但如果同时 CH0_IF 在写入执行前置 1 了，则软件先读取回 TIMERx_IF 值为 0x24，然后执行或操作 0x4|0x1=0x5，然后写入，同时对 CH0_IF 和 ZC_IF 进行了清零，可能导致 Timer 少进入一次因捕获而产生的中断。

TIMERx_IF|=0x4;

如果希望清零 ZC_IF 标志位，应以如下方式，直接对 BIT2 写 1.

TIMERx_IF=0x4;

13.3.2.10TIMER0_IO Timer0 IO 控制寄存器

地址:0x4001_0624

复位值:0x0

表 13-20 Timer0 IO 控制寄存器 TIMER0_IO

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						CH1_DEFAULT	CH0_DEFAULT	HALT_PRT	MOE				FAIL_SEL	FAIL_POL	FAIL_EN
						RW	RW	RW	RW				RW	RW	RW
						0	0	0	0				0	0	0

位置	位名称	说明
[31:10]		未使用
[9]	CH1_DEFAULT	发生 Fail 事件 MOE 清零后，CH1 通道输出值
[8]	CH0_DEFAULT	发生 Fail 事件 MOE 清零后，CH0 通道输出值
[7]	HALT_PRT	MCU 进入 HALT 状态，Timer 输出值选择。 1: 输出 CH1_DEFAULT 和 CH0_DEFAULT；0:正常输出
[6]	MOE	当 TIMER0_CFG.CENTER=1 时，Timer 通道 0/1 输出使能 1:输出正常信号 0:输出 CH0_DEFAULT 和 CH1_DEFAULT 默认值 TIMER0_IF.FAIL_IF 变 1 将触发 MOE 变成 0，通道输出默认值。 若要恢复输出正常信号，需要先将 FAIL_IF 清零，然后软件设置 MOE=1
[5:3]		未使用

[2]	FAIL_SEL	FAIL 信号选择 0：TIMER0_FAIL，来自 GPIO 1：CLU0 输出
[1]	FAIL_POL	FAIL 信号极性 0：高电平 FAIL 1：低电平 FAIL
[0]	FAIL_EN	FAIL 信号使能 0：禁用 FAIL 1：使能 FAIL

13.3.3 TIMER1 寄存器

13.3.3.1 TIMER1_CFG Timer1 配置寄存器

地址:0x4001_0700

复位值:0x0

表 13-21 Timer1 配置寄存器 TIMER1_CFG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN		CAP1_CLR_EN	CAP0_CLR_EN			ONE_TRIG	CENTER	DIR	CLK_DIV			ETON	GATE_EN	RL_EN	XCLK_EN
RW		RW	RW			RW	RW	RW	RW			RW	RW	RW	RW
0		0	0			0	0	0	0			0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRC1				CH1_POL	CH1_MODE	CH1_FE_CAP_EN	CH1_RE_CAP_EN	SRC0				CH0_POL	CH0_MODE	CH0_FE_CAP_EN	CH0_RE_CAP_EN
RW				RW	RW	RW	RW	RW				RW	RW	RW	RW
0001				0	0	0	0	0				0	0	0	0

位置	位名称	说明
[31]	EN	Timer 模块整体使能，高有效
[30]		未使用
[29]	CAP1_CLR_EN	当发生 CAP1 捕获事件时，清零 Timer 计数器，高有效
[28]	CAP0_CLR_EN	当发生 CAP0 捕获事件时，清零 Timer 计数器，高有效
[27:26]		未使用
[25]	ONE_TRIG	单次发送模式，此位需要在 Timer 比较模式下使用，且 EN 需设置为 0 ETON 为 0 时，软件向 ONE_TRIG 写 1 触发 Timer 发送一个周期的特定占空比的脉冲，Timer 计数一个周期后停止。此位在 Timer 计数的当前周期内读为 1，Timer 停止后，自动清零。

		ETON 为 1 时，即使软件配置 ONE_TRIG=1，也需要等待外部事件触发，Timer 计数一个周期后停止。ONE_TRIG 位在外部触发 Timer 计数一个周期停止后被硬件清零。
[24]	CENTER	中心计数模式使能 0: Timer 向上从 0 计数至 TH，然后回 0，或 Timer 向下从 TH 计数至 0，然后回到 TH 1: Timer 向上从 0 计数至 TH，然后向下计数至 0
[23]	DIR	0: 0->TH 递增计数，1：TH->递减计数 此位在 CENTER=1 时只读，用于指示当前计数方向
[22:20]	CLK_DIV	Timer 计数器频率配置，计数器计数频率是系统主时钟频率的 2^{CLK_DIV} 分频。默认值为 0，不分频。 0: 1 分频 1: 2 分频 2: 4 分频 3: 8 分频 4: 16 分频 5: 32 分频 6: 64 分频 7: 128 分频
[19]	ETON	Timer 计数器计数外部启动使能 0: 自动运行 1: 等待外部事件触发计数，外部触发信号根据 TIMER1_EVT.EVT_SRC 进行选择 需要注意的是，使用外部触发也需要设置 TIMER1_CFG.EN=1，除非是外部信号进行的单次触发，即 TIMER1_CFG.ONE_TRIG=1。
[18]	GATE_EN	Timer 暂停使能 0：不暂停 1：当外部信号为低时，Timer 暂停计数，外部信号根据 TIMER1_EVT.EVT_SRC 进行选择
[17]	RL_EN	Timer 重装使能 0：禁用外部事件重装，Timer 向上计数至 TH 回 0，或向下计数值 0 回到 TH 1：使能外部事件重装，重装信号根据 TIMER1_EVT.EVT_SRC 进行选择，当向上计数时，发生外部事件，Timer 重装为 0,；当向下计数时，发生外部事件，Timer 重装为 TH。
[16]	XCLK_EN	Timer 时钟源 0: 芯片内部时钟 1: 外部时钟，时钟源根据 TIMER1_EVT.EVT_SRC 进行选择，当使用外部时钟时，CLK_DIV 不再起作用，即时钟不再进行分频
[15:12]	SRC1	Timer 捕获模式通道 1 信号来源。默认为 3’h1。 0: Timer 通道 0，来自芯片 GPIO（参见 Datasheet 及应用配置） 1: Timer 通道 1，来自芯片 GPIO（参见 Datasheet 及应用配置） 2: CLU0 输出 3: CLU1 输出 4: CLU2 输出

		5: CLU3 输出 6: 比较器 0 的输出 7: 比较器 1 的输出 8: 比较器 2 的输出 9: Timer 通道 0 和 1 的异或
[11]	CH1_POL	Timer 通道 1 在比较模式下的输出极性控制，当计数器 CNT<CMP1 时的输出值。
[10]	CH1_MODE	Timer 通道 1 的工作模式选择，默认值为 0。 0: 比较模式。输出方波，在 Timer 通道 1 计数器计数值等于 0 或等于 Timer 比较捕获寄存器值时，IO 发生翻转。 1: 捕获模式。当 Timer 通道 1 输入信号发生捕获事件时，将计数器计数值存入 Timer 通道 1 比较捕获寄存器。
[9]	CH1_FE_CAP_EN	Timer 通道 1 下降沿捕获事件使能。1:使能；0:关闭。 Timer 通道 1 输入信号发生 1→0 跳变被视为捕获事件。下降沿事件使能可以与上升沿事件使能并存。
[8]	CH1_RE_CAP_EN	Timer 通道 1 上升沿捕获事件使能。1:使能；0:关闭。 Timer 通道 1 输入信号发生 0→1 跳变被视为捕获事件。上升沿事件使能可以与下降沿事件使能并存。
[7:4]	SRC0	Timer 捕获模式通道 0 信号来源。默认为 3'h0。 0: Timer 通道 0，来自芯片 GPIO（参见 Datasheet 及应用配置） 1: Timer 通道 1，来自芯片 GPIO（参见 Datasheet 及应用配置） 2: CLU0 输出 3: CLU1 输出 4: CLU2 输出 5: CLU3 输出 6: 比较器 0 的输出 7: 比较器 1 的输出 8: 比较器 2 的输出 9: Timer 通道 0 和 1 的异或
[3]	CH0_POL	Timer 通道 0 在比较模式下的输出极性控制:当计数器 CNT<CMP0 时的输出值。
[2]	CH0_MODE	Timer 通道 0 的工作模式选择，默认值为 0。 0: 比较模式。输出方波，在 Timer 通道 0 计数器计数值等于 0 或等于 Timer 比较捕获寄存器值时，IO 发生翻转。 1: 捕获模式。当 Timer 通道 0 输入信号发生捕获事件时，将计数器计数值存入 Timer 通道 0 比较捕获寄存器。
[1]	CH0_FE_CAP_EN	Timer 通道 0 下降沿捕获事件使能。1:使能；0:关闭。 Timer 通道 0 输入信号发生 1→0 跳变被视为捕获事件。下降沿事件使能可以与上升沿事件使能并存。
[0]	CH0_RE_CAP_EN	Timer 通道 0 上升沿捕获事件使能。1:使能；0:关闭。 Timer 通道 0 输入信号发生 0→1 跳变被视为捕获事件。上升沿事件使能可以与下降沿事件使能并存。

当使用外部时钟计数时，也需要设置 TIMERx_CFG.TON=1，且需要开启 SYS_CLK_FEN 中对于 Timer 时钟的使能。使用外部时钟计数时，也需要设置 TIMERx_TH。

如果设置 SRC0 为 4'h8 捕获两通道异或值，（通常为捕获编码器正交的两个通道），需要同时设置 CMP0_CLR_EN=1，CH0_FE_CAP_EN=1，CH0_RE_CAP_EN=1，CH0_MODE=1，即捕获上升沿、下降沿，且每次有信号边沿都清零计数器。CMP0 即为捕获的两次信号边沿之间的计数值。

13.3.3.2 TIMER1_TH Timer1 门限寄存器

地址:0x4001_0704

复位值:0x0

表 13-22 Timer1 门限寄存器 TIMER1_TH

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

位置	位名称	说明
[31:16]		未使用
[15:0]	TH	Timer 计数器计数门限。向上计数从 0 计数到 TH 值后回 0，或向下计数从 TH 计数到 0 后回到 TH

13.3.3.3 TIMER1_CNT Timer1 计数寄存器

地址:0x4001_0708

复位值:0x0

表 13-23 Timer1 计数寄存器 TIMER1_CNT

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

位置	位名称	说明
[31:16]		未使用
[15:0]	CNT	Timer 0 计数器当前计数值。写操作可以写入新的计数值。

注意:写入 TIMER1_CNT 前需要先通过 SYS_CLK_FEN.TIMER1_CLK_EN 开启 Timer1 时钟。

13.3.3.4 TIMER1_CMP0 Timer1 通道 0 比较捕获寄存器

地址:0x4001_070C

复位值:0x0

表 13-24 Timer1 通道 0 比较捕获寄存器 TIMER1_CMP0

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

CMP0
RW
0

位置	位名称	说明
[31:16]		未使用
[15:0]	CMP0	Timer 通道 0 工作在比较模式时，当计数器计数值等于 CMP0 时，发生比较事件。 Timer 通道 0 工作在捕获模式时，发生捕获事件时的计数器计数值存入 CMP0 寄存器。

设置 CMP0=0，可使得通道 0 恒为!CH0_POL，设置 CMP0=TH+1，可使得通道 0 恒为 CH0_POL。

13.3.3.5 TIMER1_CMP1 Timer1 通道 1 比较捕获寄存器

地址:0x4001_0710

复位值:0x0

表 13-25 Timer1 通道 1 比较捕获寄存器 TIMER1_CMP1

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

位置	位名称	说明
[31:16]		未使用
[15:0]	CMP1	Timer 通道 1 工作在比较模式时，当计数器计数值等于 CMP1 时，发生比较事件。 Timer 通道 1 工作在捕获模式时，发生捕获事件时的计数器计数值存入 CMP1 寄存器。

设置 CMP1=0，可使得通道恒为!CH1_POL，设置 CMP1=TH+1，可使得通道恒为 CH1_POL。

13.3.3.6 TIMER1_EVT Timer1 外部事件选择寄存器

地址:0x4001_0714

复位值:0x0

表 13-26 Timer1 外部事件选择寄存器 TIMER1_EVT

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

位置	位名称	说明
[31:4]		未使用

[3:0]	EVT_SRC	Timer 外部事件选择寄存器，本寄存器需要配合 TIMER1_CFG.ETON，TIMER1_CFG.GATE_EN，TIMER1_CFG.RL_EN，TIMER1_CFG.XCLK_EN 使用。 通常 4 个设置位不同时使用。 当设置 ETON=1 时，根据本寄存器选择触发 Timer 计数的事件。 需要注意的是，Timer 自身的比较事件无法触发 Timer 进行计数。当 ETON=1 时，触发信号的上升沿触发 Timer 开始计数，用作触发源的 Timer 需要工作在比较模式，无须从外部输入信号至 Timer 通道 0:TIMER0 通道 0 比较/捕获事件 1:TIMER0 通道 1 比较/捕获事件 2:不可用 3:不可用 4:TIMER2 通道 0 比较/捕获事件 5:TIMER2 通道 1 比较/捕获事件 6:TIMER3 通道 0 比较/捕获事件 7:TIMER3 通道 1 比较/捕获事件 8:CLU0 输出上升沿 9:CLU1 输出上升沿 10:CLU2 输出上升沿 11:CLU3 输出上升沿 12:MCPWM TADC[0]比较事件 13:MCPWM TADC[1]比较事件 14:MCPWM TADC[2]比较事件 15:MCPWM TADC[3]比较事件 当 GATE_EN=1，且外部信号为高时 Timer 计数，外部信号为低时，Timer 暂停，当 GATE_EN=1 时，Timer 使用对应外部信号的电平信号作为计数门控。GATE_EN 仅支持配合 0~11 的外部事件使用，当 EVT_SRC 选择为 0~7 时，外部信号为 Timer 通道输入信号经过对应 Timer 滤波后的信号，作为门控信号的 Timer 通道需要配置为输入使能 0:TIMER0 通道 0 滤波后输入信号 1:TIMER0 通道 1 滤波后输入信号 2:TIMER1 通道 0 滤波后输入信号 3:TIMER1 通道 1 滤波后输入信号 4:TIMER2 通道 0 滤波后输入信号 5:TIMER2 通道 1 滤波后输入信号 6:TIMER3 通道 0 滤波后输入信号 7:TIMER3 通道 1 滤波后输入信号 8:CLU0 输出信号 9:CLU1 输出信号 10:CLU2 输出信号 11:CLU3 输出信号 当 RL_EN 为高时，Timer 使用外部信号作为重装载信号

		作为重装载信号的 Timer 通道需要配置为输入使能 0:TIMER0 通道 0 比较/捕获事件 1:TIMER0 通道 1 比较/捕获事件 2:TIMER1 通道 0 比较/捕获事件 3:TIMER1 通道 1 比较/捕获事件 4:TIMER2 通道 0 比较/捕获事件 5:TIMER2 通道 1 比较/捕获事件 6:TIMER3 通道 0 比较/捕获事件 7:TIMER3 通道 1 比较/捕获事件 8:CLU0 输出 9:CLU1 输出 10:CLU2 输出 11:CLU3 输出 12:MCPWM TADC[0]比较事件 13:MCPWM TADC[1]比较事件 14:MCPWM TADC[2]比较事件 15:MCPWM TADC[3]比较事件 当 XCLK_EN 为高时，Timer 使用外部信号作为 Timer 外部时钟，Timer 在时钟上升沿进行计数动作。如果使用 Timer 通道作为外部时钟，通道信号受到对应 Timer 滤波设置影响。 0:TIMER0 通道 0 输入信号 1:TIMER0 通道 1 输入信号 2:TIMER1 通道 0 输入信号 3:TIMER1 通道 1 输入信号 4:TIMER2 通道 0 输入信号 5:TIMER2 通道 1 输入信号 6:TIMER3 通道 0 输入信号 7:TIMER3 通道 1 输入信号 8:CLU0 输出 9:CLU1 输出 10:CLU2 输出 11:CLU3 输出 12:MCPWM TADC[0]比较事件 13:MCPWM TADC[1]比较事件 14:MCPWM TADC[2]比较事件 15:MCPWM TADC[3]比较事件

13.3.3.7 TIMER1_FLT Timer1 滤波控制寄存器

地址:0x4001_0718

复位值:0x0

表 13-27 Timer1 滤波控制寄存器 TIMER1_FLT

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

	FLT
	RW
	0

位置	位名称	说明
[31:8]		未使用
[7:0]	FLT	通道 0/1 信号滤波宽度选择。取值范围 0~255。 FLT 为 0 时，不对通道进行滤波。 FLT 不为 0 时，对通道信号进行滤波:滤波宽度为 FLT×8。当通道信号电平稳定超过 FLT×8 个系统时钟周期宽度时，滤波器输出更新；否则，滤波器保持当前的输出不变。

注意，以上滤波器的工作时钟与对应的 Timer 的分频后的工作时钟为相同时钟，受 TIMERx_CFG.CLK_DIV 的分频系数控制。

假设 TIMERx_FLT = 0x6；TIMERx_CFG.CLK_DIV = 0x2; 则 Timer 的运行时钟相对系统时钟进行了 4 分频。通道输入信号需要经过 8×6 倍 Timer 的运行时钟的滤波，亦即 8×6×4 倍系统时钟的滤波。

13.3.3.8 TIMER1_IE Timer1 中断使能寄存器

地址:0x4001_071C

复位值:0x0

表 13-28 Timer1 中断使能寄存器 TIMER1_IE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					ZC_RE	CH1_RE	CH0_RE						ZC_IE	CH1_IE	CH0_IE
					RW	RW	RW						RW	RW	RW
					0	0	0						0	0	0

位置	位名称	说明
[31:11]		未使用
[10]	ZC_RE	Timer 计数器过 0 DMA 请求使能，高电平有效。
[9]	CH1_RE	Timer 通道 1 比较/捕获 DMA 请求使能，高电平有效。
[8]	CH0_RE	Timer 通道 0 比较/捕获 DMA 请求使能，高电平有效。
[7:3]		未使用
[2]	ZC_IE	Timer 计数器过 0 中断使能，高电平有效。
[1]	CH1_IE	Timer 通道 1 比较/捕获中断使能，高电平有效。
[0]	CH0_IE	Timer 通道 0 比较/捕获中断使能，高电平有效。

DMA 请求事件，与中断为相同事件标志，DMA 请求被 DMA 受理后，中断标志会被 DMA 硬件自动清除，无需 CPU 软件干预。需要注意的是，通常开启某个事件的 DMA 请求则关闭对应的中断

使能。

13.3.3.9 TIMER1_IF Timer1 中断标志寄存器

地址:0x4001_0720

复位值:0x0

表 13-29 Timer1 中断标志寄存器 TIMER1_IF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													ZC_IF	CH1_IF	CH0_IF
													RW1C	RW1C	RW1C
													0	0	0

位置	位名称	说明
[31:3]		未使用
[2]	ZC_IF	Timer 计数器过 0 中断标志。高电平有效，写 1 清 0
[1]	CH1_IF	Timer 通道 1 比较/捕获中断标志。高电平有效，写 1 清 0
[0]	CH0_IF	Timer 通道 0 比较/捕获中断标志。高电平有效，写 1 清 0

在比较模式下，当 Timer 设置为中心计数（0->TH->0），即 TIMERx_CFG.CENTER=1 时。CH0_IF 和 CH1_IF 只有在 Timer 从 0 递增计数至 TH 的过程中，当 TIMERx_CNT=TIMERx_CMP0/1 时置位；在 Timer 从 TH 递减计数到 0 的过程中不会置位。

中断标志写 1 清零，一般不建议用如下|=方式清零，因为|=是先读取中断标志，将对应位改为 1 再写入清零，如果同时有其他中断标志置位，会被一起清零，而这通常不是软件所期望的。例如，如下写法本意是清零 ZC_IF，但如果同时 CH0_IF 在写入执行前置 1 了，则软件先读取回 TIMERx_IF 值为 0x24，然后执行或操作 0x4|0x1=0x5，然后写入，同时对 CH0_IF 和 ZC_IF 进行了清零，可能导致 Timer 少进入一次因捕获而产生的中断。

TIMERx_IF|=0x4;

如果希望清零 T0_ZC_IF 标志位，应以如下方式，直接对 BIT2 写 1.

TIMERx_IF=0x4;

13.3.4 TIMER2 寄存器

13.3.4.1 TIMER2_CFG Timer2 配置寄存器

地址:0x4001_0800

复位值:0x0

表 13-30 Timer2 配置寄存器 TIMER2_CFG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN		CAP1_CLR_EN	CAP0_CLR_EN			ONE_TRIG	CENTER	DIR	CLK_DIV			ETON	GATE_EN	RL_EN	XCLK_EN
RW		RW	RW			RW	RW	RW	RW			RW	RW	RW	RW
0		0	0			0	0	0	0			0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRC1				CH1_POL	CH1_MODE	CH1_FE_CAP_EN	CH1_RE_CAP_EN	SRC0				CH0_POL	CH0_MODE	CH0_FE_CAP_EN	CH0_RE_CAP_EN
RW				RW	RW	RW	RW	RW				RW	RW	RW	RW
0001				0	0	0	0	0				0	0	0	0

位置	位名称	说明
[31]	EN	Timer 模块整体使能，高有效
[30]		未使用
[29]	CAP1_CLR_EN	当发生 CAP1 捕获事件时，清零 Timer 计数器，高有效
[28]	CAP0_CLR_EN	当发生 CAP0 捕获事件时，清零 Timer 计数器，高有效
[27:26]		未使用
[25]	ONE_TRIG	单次发送模式，此位需要在 Timer 比较模式下使用，且 EN 需设置为 0 ETON 为 0 时，软件向 ONE_TRIG 写 1 触发 Timer 发送一个周期的特定占空比的脉冲，Timer 计数一个周期后停止。此位在 Timer 计数的当前周期内读为 1，Timer 停止后，自动清零。 ETON 为 1 时，即使软件配置 ONE_TRIG=1，也需要等待外部事件触发，Timer 计数一个周期后停止。ONE_TRIG 位在外部触发 Timer 计数一个周期停止后被硬件清零。
[24]	CENTER	中心计数模式使能 0: Timer 向上从 0 计数至 TH，然后回 0，或 Timer 向下从 TH 计数至 0，然后回到 TH 1: Timer 向上从 0 计数至 TH，然后向下计数至 0
[23]	DIR	0: 0->TH 递增计数，1：TH->递减计数 此位在 CENTER=1 时只读，用于指示当前计数方向
[22:20]	CLK_DIV	Timer 计数器频率配置，计数器计数频率是系统主时钟频率的 2^{CLK_DIV} 分频。默认值为 0，不分频。

		0: 1 分频 1: 2 分频 2: 4 分频 3: 8 分频 4: 16 分频 5: 32 分频 6: 64 分频 7: 128 分频
[19]	ETON	Timer 计数器外部启动使能 0: 自动运行 1: 等待外部事件触发计数，外部触发信号根据 TIMER2_EVT.EVT_SRC 进行选择 需要注意的是，使用外部触发也需要设置 TIMER2_CFG.EN=1，除非是外部信号进行的单次触发，即 TIMER2_CFG.ONE_TRIG=1。
[18]	GATE_EN	Timer 暂停使能 0：不暂停 1：当外部信号为低时，Timer 暂停计数，外部信号根据 TIMER2_EVT.EVT_SRC 进行选择
[17]	RL_EN	Timer 重装使能 0：禁用外部事件重装，Timer 向上计数至 TH 回 0，或向下计数值 0 回到 TH 1：使能外部事件重装，重装信号根据 TIMER2_EVT.EVT_SRC 进行选择，当向上计数时，发生外部事件，Timer 重装为 0,；当向下计数时，发生外部事件，Timer 重装为 TH。
[16]	XCLK_EN	Timer 时钟源 0: 芯片内部时钟 1: 外部时钟，时钟源根据 TIMER2_EVT.EVT_SRC 进行选择，当使用外部时钟时，CLK_DIV 不再起作用，即时钟不再进行分频
[15:12]	SRC1	Timer 捕获模式通道 1 信号来源。默认为 3’h1。 0: Timer 通道 0，来自芯片 GPIO（参见 Datasheet 及应用配置） 1: Timer 通道 1，来自芯片 GPIO（参见 Datasheet 及应用配置） 2: CLU0 输出 3: CLU1 输出 4: CLU2 输出 5: CLU3 输出 6: 比较器 0 的输出 7: 比较器 1 的输出 8: 比较器 2 的输出 9: Timer 通道 0 和 1 的异或
[11]	CH1_POL	Timer 通道 1 在比较模式下的输出极性控制，当计数器 CNT<CMP1 时的输出值。
[10]	CH1_MODE	Timer 通道 1 的工作模式选择，默认值为 0。 0: 比较模式。输出方波，在 Timer 通道 1 计数器计数值等于 0 或等于 Timer 比较捕获寄存器值时，IO 发生翻转。 1: 捕获模式。当 Timer 通道 1 输入信号发生捕获事件时，将计数器

		计数值存入 Timer 通道 1 比较捕获寄存器
[9]	CH1_FE_CAP_EN	Timer 通道 1 下降沿捕获事件使能。1:使能；0:关闭。 Timer 通道 1 输入信号发生 1→0 跳变被视为捕获事件。下降沿事件使能可以与上升沿事件使能并存。
[8]	CH1_RE_CAP_EN	Timer 通道 1 上升沿捕获事件使能。1:使能；0:关闭。 Timer 通道 1 输入信号发生 0→1 跳变被视为捕获事件。上升沿事件使能可以与下降沿事件使能并存。
[7:4]	SRC0	Timer 捕获模式通道 0 信号来源。默认为 3'h0。 0: Timer 通道 0，来自芯片 GPIO（参见 Datasheet 及应用配置） 1: Timer 通道 1，来自芯片 GPIO（参见 Datasheet 及应用配置） 2: CLU0 输出 3: CLU1 输出 4: CLU2 输出 5: CLU3 输出 6: 比较器 0 的输出 7: 比较器 1 的输出 8: 比较器 2 的输出 9: Timer 通道 0 和 1 的异或
[3]	CH0_POL	Timer 通道 0 在比较模式下的输出极性控制:当计数器 CNT<CMP0 时的输出值。
[2]	CH0_MODE	Timer 通道 0 的工作模式选择，默认值为 0。 0: 比较模式。输出方波，在 Timer 通道 0 计数器计数值等于 0 或等于 Timer 比较捕获寄存器值时，IO 发生翻转。 1: 捕获模式。当 Timer 通道 0 输入信号发生捕获事件时，将计数器计数值存入 Timer 通道 0 比较捕获寄存器。
[1]	CH0_FE_CAP_EN	Timer 通道 0 下降沿捕获事件使能。1:使能；0:关闭。 Timer 通道 0 输入信号发生 1→0 跳变被视为捕获事件。下降沿事件使能可以与上升沿事件使能并存。
[0]	CH0_RE_CAP_EN	Timer 通道 0 上升沿捕获事件使能。1:使能；0:关闭。 Timer 通道 0 输入信号发生 0→1 跳变被视为捕获事件。上升沿事件使能可以与下降沿事件使能并存。

13.3.4.2 TIMER2_TH Timer2 门限寄存器

地址:0x4001_0804

复位值:0x0

表 13-31 Timer2 门限寄存器 TIMER2_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															

位置	位名称	说明
[31:0]	TH	Timer 计数器计数门限。向上计数从 0 计数到 TH 值后回 0，或向下计数从 TH 计数到 0 后回到 TH

13.3.4.3 TIMER2_CNT Timer2 计数寄存器

地址:0x4001_0808

复位值:0x0

表 13-32 Timer2 计数寄存器 TIMER2_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															

位置	位名称	说明
[31:0]	CNT	Timer2 计数器当前计数值。写操作可以写入新的计数值。

注意:写入 TIMER2_CNT 前需要先通过 SYS_CLK_FEN.TIMER2_CLK_EN 开启 Timer2 时钟。

13.3.4.4 TIMER2_CMP0 Timer2 通道 0 比较捕获寄存器

地址:0x4001_080C

复位值:0x0

表 13-33 Timer2 通道 0 比较捕获寄存器 TIMER2_CMP0

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

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

位置	位名称	说明
[31:0]	CMP0	Time 通道 0 工作在比较模式时，当计数器计数值等于 CMP0 时，发生比较事件。

		Timer 通道 0 工作在捕获模式时，发生捕获事件时的计数器计数值存入 CMP0 寄存器。

设置 CMP0=0，可使得通道 0 恒为!CH0_POL，设置 CMP0=TH+1，可使得通道 0 恒为 CH0_POL。

13.3.4.5 TIMER2_CMP1 Timer2 通道 1 比较捕获寄存器

地址:0x4001_0810

复位值:0x0

表 13-34 Timer2 通道 1 比较捕获寄存器 TIMER2_CMP1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CMP1															
RW															
0															

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

位置	位名称	说明
[31:0]	CMP1	Timer 通道 1 工作在比较模式时，当计数器计数值等于 CMP1 时，发生比较事件。 Timer 通道 1 工作在捕获模式时，发生捕获事件时的计数器计数值存入 CMP1 寄存器。

设置 CMP1=0，可使得通道恒为!CH1_POL，设置 CMP1=TH+1，可使得通道恒为 CH1_POL。

13.3.4.6 TIMER2_EVT Timer2 外部事件选择寄存器

地址:0x4001_0814

复位值:0x0

表 13-35 Timer2 外部事件选择寄存器 TIMER2_EVT

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

位置	位名称	说明
[31:4]		未使用
[3:0]	EVT_SRC	Timer 外部事件选择寄存器，本寄存器需要配合 TIMER2_CFG.ETON，TIMER2_CFG.GATE_EN，TIMER2_CFG.RL_EN，TIMER2_CFG.XCLK_EN 使用。 通常 4 个设置位不同时使用。 当设置 ETON=1 时，根据本寄存器选择触发 Timer 计数的事件。

		需要注意的是，Timer 自身的比较事件无法触发 Timer 进行计数。当 ETON=1 时，触发信号的上升沿触发 Timer 开始计数，用作触发源的 Timer 需要工作在比较模式，无须从外部输入信号至 Timer 通道 0:TIMER0 通道 0 比较/捕获事件 1:TIMER0 通道 1 比较/捕获事件 2:TIMER1 通道 0 比较/捕获事件 3:TIMER1 通道 1 比较/捕获事件 4:不可用 5:不可用 6:TIMER3 通道 0 比较/捕获事件 7:TIMER3 通道 1 比较/捕获事件 8:CLU0 输出上升沿 9:CLU1 输出上升沿 10:CLU2 输出上升沿 11:CLU3 输出上升沿 12:MCPWM TADC[0]比较事件 13:MCPWM TADC[1]比较事件 14:MCPWM TADC[2]比较事件 15:MCPWM TADC[3]比较事件 当 GATE_EN=1，且外部信号为高时 Timer 计数，外部信号为低时，Timer 暂停，当 GATE_EN=1 时，Timer 使用对应外部信号的电平信号作为计数门控。GATE_EN 仅支持配合 0~11 的外部事件使用，当 EVT_SRC 选择为 0~7 时，外部信号为 Timer 通道输入信号经过对应 Timer 滤波后的信号，作为门控信号的 Timer 通道需要配置为输入使能 0:TIMER0 通道 0 滤波后输入信号 1:TIMER0 通道 1 滤波后输入信号 2:TIMER1 通道 0 滤波后输入信号 3:TIMER1 通道 1 滤波后输入信号 4:TIMER2 通道 0 滤波后输入信号 5:TIMER2 通道 1 滤波后输入信号 6:TIMER3 通道 0 滤波后输入信号 7:TIMER3 通道 1 滤波后输入信号 8:CLU0 输出信号 9:CLU1 输出信号 10:CLU2 输出信号 11:CLU3 输出信号 当 RL_EN 为高时，Timer 使用外部信号作为重装载信号 作为重装载信号的 Timer 通道需要配置为输入使能 0:TIMER0 通道 0 比较/捕获事件 1:TIMER0 通道 1 比较/捕获事件 2:TIMER1 通道 0 比较/捕获事件 3:TIMER1 通道 1 比较/捕获事件 4:TIMER2 通道 0 比较/捕获事件

		5:TIMER2 通道 1 比较/捕获事件 6:TIMER3 通道 0 比较/捕获事件 7:TIMER3 通道 1 比较/捕获事件 8:CLU0 输出 9:CLU1 输出 10:CLU2 输出 11:CLU3 输出 12:MCPWM TADC[0]比较事件 13:MCPWM TADC[1]比较事件 14:MCPWM TADC[2]比较事件 15:MCPWM TADC[3]比较事件 当 XCLK_EN 为高时，Timer 使用外部信号作为 Timer 外部时钟，Timer 在时钟上升沿进行计数动作。如果使用 Timer 通道作为外部时钟，通道信号受到对应 Timer 滤波设置影响。 0:TIMER0 通道 0 输入信号 1:TIMER0 通道 1 输入信号 2:TIMER1 通道 0 输入信号 3:TIMER1 通道 1 输入信号 4:TIMER2 通道 0 输入信号 5:TIMER2 通道 1 输入信号 6:TIMER3 通道 0 输入信号 7:TIMER3 通道 1 输入信号 8:CLU0 输出 9:CLU1 输出 10:CLU2 输出 11:CLU3 输出 12:MCPWM TADC[0]比较事件 13:MCPWM TADC[1]比较事件 14:MCPWM TADC[2]比较事件 15:MCPWM TADC[3]比较事件

13.3.4.7 TIMER2_FLT Timer2 滤波控制寄存器

地址:0x4001_0818

复位值:0x0

表 13-36 Timer2 滤波控制寄存器 TIMER2_FLT

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

位置	位名称	说明

[31:8]		未使用
[7:0]	FLT	通道 0/1 信号滤波宽度选择。取值范围 0~255。 FLT 为 0 时，不对通道进行滤波。 FLT 不为 0 时，对通道信号进行滤波:滤波宽度为 FLT×8。当通道信号电平稳定超过 FLT×8 个系统时钟周期宽度时，滤波器输出更新；否则，滤波器保持当前的输出不变。

注意，以上滤波器的工作时钟与对应的 Timer 的分频后的工作时钟为相同时钟，受 TIMERx_CFG.CLK_DIV 的分频系数控制。

假设 TIMERx_FLT.FLT = 0x6；TIMERx_CFG.CLK_DIV = 0x2; 则 Timer 的运行时钟相对系统时钟进行了 4 分频。通道输入信号需要经过 8×6 倍 Timer 的运行时钟的滤波，亦即 8×6×4 倍系统时钟的滤波。

13.3.4.8 TIMER2_IE Timer2 中断使能寄存器

地址:0x4001_081C

复位值:0x0

表 13-37 Timer2 中断使能寄存器 TIMER2_IE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					ZC_RE	CH1_RE	CH0_RE						ZC_IE	CH1_IE	CH0_IE
					RW	RW	RW						RW	RW	RW
					0	0	0						0	0	0

位置	位名称	说明
[31:11]		未使用
[10]	ZC_RE	Timer 计数器过 0 DMA 请求使能，高电平有效。
[9]	CH1_RE	Timer 通道 1 比较/捕获 DMA 请求使能，高电平有效。
[8]	CH0_RE	Timer 通道 0 比较/捕获 DMA 请求使能，高电平有效。
[7:3]		未使用
[2]	ZC_IE	Timer 计数器过 0 中断使能，高电平有效。
[1]	CH1_IE	Timer 通道 1 比较/捕获中断使能，高电平有效。
[0]	CH0_IE	Timer 通道 0 比较/捕获中断使能，高电平有效。

DMA 请求事件，与中断为相同事件标志，DMA 请求被 DMA 受理后，中断标志会被 DMA 硬件自动清除，无需 CPU 软件干预。需要注意的是，通常开启某个事件的 DMA 请求则关闭对应的中断使能。

13.3.4.9 TIMER2_IF Timer2 中断标志寄存器

地址:0x4001_0820

复位值:0x0

表 13-38 Timer2 中断标志寄存器 TIMER2_IF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													ZC_IF	CH1_IF	CH0_IF
													RW1C	RW1C	RW1C
													0	0	0

位置	位名称	说明
[31:3]		未使用
[2]	ZC_IF	Timer 计数器过 0 中断标志。高电平有效，写 1 清 0
[1]	CH1_IF	Timer 通道 1 比较/捕获中断标志。高电平有效，写 1 清 0
[0]	CH0_IF	Timer 通道 0 比较/捕获中断标志。高电平有效，写 1 清 0

在比较模式下，当 Timer 设置为中心计数（0->TH->0），即 TIMERx_CFG.CENTER=1 时。CH0_IF 和 CH1_IF 只有在 Timer 从 0 递增计数至 TH 的过程中，当 TIMERx_CNT=TIMERx_CMP0/1 时置位；在 Timer 从 TH 递减计数到 0 的过程中不会置位。

中断标志写 1 清零，一般不建议用如下|=方式清零，因为|=是先读取中断标志，将对应位改为 1 再写入清零，如果同时有其他中断标志置位，会被一起清零，而这通常不是软件所期望的。例如，如下写法本意是清零 ZC_IF，但如果同时 CH0_IF 在写入执行前置 1 了，则软件先读取回 TIMERx_IF 值为 0x24，然后执行或操作 0x4|0x1=0x5，然后写入，同时对 CH0_IF 和 ZC_IF 进行了清零，可能导致 Timer 少进入一次因捕获而产生的中断。

TIMERx_IF|=0x4;

如果希望清零 T0_ZC_IF 标志位，应以如下方式，直接对 BIT2 写 1.

TIMERx_IF=0x4;

13.3.5 TIMER3 寄存器

13.3.5.1 TIMER3_CFG Timer3 配置寄存器

地址:0x4001_0900

复位值:0x0

表 13-39 Timer3 配置寄存器 TIMER3_CFG

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

EN		CAP1_CLR_EN	CAP0_CLR_EN		ONE_TRIG	CENTER	DIR	CLK_DIV	ETON	GATE_EN	RL_EN	XCLK_EN
RW		RW	RW		RW	RW	RW	RW	RW	RW	RW	RW
0		0	0		0	0	0	0	0	0	0	0

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

SRC1				CH1_POL	CH1_MODE	CH1_FE_CAP_EN	CH1_RE_CAP_EN	SRC0				CH0_POL	CH0_MODE	CH0_FE_CAP_EN	CH0_RE_CAP_EN
RW				RW	RW	RW	RW	RW				RW	RW	RW	RW
0001				0	0	0	0	0				0	0	0	0

位置	位名称	说明
[31]	EN	Timer 模块整体使能，高有效
[30]		未使用
[29]	CAP1_CLR_EN	当发生 CAP1 捕获事件时，清零 Timer 计数器，高有效
[28]	CAP0_CLR_EN	当发生 CAP0 捕获事件时，清零 Timer 计数器，高有效
[27:26]		未使用
[25]	ONE_TRIG	单次发送模式，此位需要在 Timer 比较模式下使用，且 EN 需设置为 0 ETON 为 0 时，软件向 ONE_TRIG 写 1 触发 Timer 发送一个周期的特定占空比的脉冲，Timer 计数一个周期后停止。此位在 Timer 计数的当前周期内读为 1，Timer 停止后，自动清零。 ETON 为 1 时，即使软件配置 ONE_TRIG=1，也需要等待外部事件触发，Timer 计数一个周期后停止。ONE_TRIG 位在外部触发 Timer 计数一个周期停止后被硬件清零。
[24]	CENTER	中心计数模式使能 0: Timer 向上从 0 计数至 TH，然后回 0，或 Timer 向下从 TH 计数至 0，然后回到 TH 1: Timer 向上从 0 计数至 TH，然后向下计数至 0
[23]	DIR	0: 0->TH 递增计数，1：TH->递减计数 此位在 CENTER=1 时只读，用于指示当前计数方向
[22:20]	CLK_DIV	Timer 计数器频率配置，计数器计数频率是系统主时钟频率的 2^{CLK_DIV} 分频。默认值为 0，不分频。 0: 1 分频 1: 2 分频 2: 4 分频 3: 8 分频 4: 16 分频 5: 32 分频 6: 64 分频 7: 128 分频

[19]	ETON	Timer 计数器外部启动使能 0: 自动运行 1: 等待外部事件触发计数，外部触发信号根据 TIMER3_EVT.EVT_SRC 进行选择 需要注意的是，使用外部触发也需要设置 TIMER3_CFG.EN=1，除非是外部信号进行的单次触发，即 TIMER3_CFG.ONE_TRIG=1。
[18]	GATE_EN	Timer 暂停使能 0：不暂停 1：当外部信号为低时，Timer 暂停计数，外部信号根据 TIMER3_EVT.EVT_SRC 进行选择
[17]	RL_EN	Timer 重装使能 0：禁用外部事件重装，Timer 向上计数至 TH 回 0，或向下计数值 0 回到 TH 1：使能外部事件重装，重装信号根据 TIMER3_EVT.EVT_SRC 进行选择，当向上计数时，发生外部事件，Timer 重装为 0,；当向下计数时，发生外部事件，Timer 重装为 TH。
[16]	XCLK_EN	Timer 时钟源 0: 芯片内部时钟 1: 外部时钟，时钟源根据 TIMER3_EVT.EVT_SRC 进行选择，当使用外部时钟时，CLK_DIV 不再起作用，即时钟不再进行分频
[15:12]	SRC1	Timer 捕获模式通道 1 信号来源。默认为 3'h1。 0: Timer 通道 0，来自芯片 GPIO（参见 Datasheet 及应用配置） 1: Timer 通道 1，来自芯片 GPIO（参见 Datasheet 及应用配置） 2: CLU0 输出 3: CLU1 输出 4: CLU2 输出 5: CLU3 输出 6: 比较器 0 的输出 7: 比较器 1 的输出 8: 比较器 2 的输出 9: Timer 通道 0 和 1 的异或
[11]	CH1_POL	Timer 通道 1 在比较模式下的输出极性控制，当计数器 CNT<CMP1 时的输出值。
[10]	CH1_MODE	Timer 通道 1 的工作模式选择，默认值为 0。 0: 比较模式。输出方波，在 Timer 通道 1 计数器计数值等于 0 或等于 Timer 比较捕获寄存器值时，IO 发生翻转。 1: 捕获模式。当 Timer 通道 1 输入信号发生捕获事件时，将计数器计数值存入 Timer 通道 1 比较捕获寄存器
[9]	CH1_FE_CAP_EN	Timer 通道 1 下降沿捕获事件使能。1:使能；0:关闭。 Timer 通道 1 输入信号发生 1→0 跳变被视为捕获事件。下降沿事件使能可以与上升沿事件使能并存。
[8]	CH1_RE_CAP_EN	Timer 通道 1 上升沿捕获事件使能。1:使能；0:关闭。 Timer 通道 1 输入信号发生 0→1 跳变被视为捕获事件。上升沿事件使能可以与下降沿事件使能并存。
[7:4]	SRC0	Timer 捕获模式通道 0 信号来源。默认为 3'h0。

		0: Timer 通道 0，来自芯片 GPIO（参见 Datasheet 及应用配置） 1: Timer 通道 1，来自芯片 GPIO（参见 Datasheet 及应用配置） 2: CLU0 输出 3: CLU1 输出 4: CLU2 输出 5: CLU3 输出 6: 比较器 0 的输出 7: 比较器 1 的输出 8: 比较器 2 的输出 9: Timer 通道 0 和 1 的异或
[3]	CH0_POL	Timer 通道 0 在比较模式下的输出极性控制:当计数器 CNT<CMP0 时的输出值。
[2]	CH0_MODE	Timer 通道 0 的工作模式选择，默认值为 0。 0: 比较模式。输出方波，在 Timer 通道 0 计数器计数值等于 0 或等于 Timer 比较捕获寄存器值时，IO 发生翻转。 1: 捕获模式。当 Timer 通道 0 输入信号发生捕获事件时，将计数器计数值存入 Timer 通道 0 比较捕获寄存器。
[1]	CH0_FE_CAP_EN	Timer 通道 0 下降沿捕获事件使能。1:使能；0:关闭。 Timer 通道 0 输入信号发生 1→0 跳变被视为捕获事件。下降沿事件使能可以与上升沿事件使能并存。
[0]	CH0_RE_CAP_EN	Timer 通道 0 上升沿捕获事件使能。1:使能；0:关闭。 Timer 通道 0 输入信号发生 0→1 跳变被视为捕获事件。上升沿事件使能可以与下降沿事件使能并存。

13.3.5.2 TIMER3_TH Timer3 门限寄存器

地址:0x4001_0904

复位值:0x0

表 13-40 Timer3 门限寄存器 TIMER3_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															

位置	位名称	说明
[31:0]	TH	Timer 计数器计数门限。向上计数从 0 计数到 TH 值后回 0，或向下计数从 TH 计数到 0 后回到 TH

13.3.5.3 TIMER3_CNT Timer3 计数寄存器

地址:0x4001_0908

复位值:0x0

表 13-41 Timer3 计数寄存器 TIMER3_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															

位置	位名称	说明
[31:0]	CNT	Timer3 计数器当前计数值。写操作可以写入新的计数值。

注意:写入 TIMER3_CNT 前需要先通过 SYS_CLK_FEN.TIMER3_CLK_EN 开启 Timer3 时钟。

13.3.5.4 TIMER3_CMP0 Timer3 通道 0 比较捕获寄存器

地址:0x4001_090C

复位值:0x0

表 13-42 Timer3 通道 0 比较捕获寄存器 TIMER3_CMP0

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

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

位置	位名称	说明
[31:0]	CMP0	Time 通道 0 工作在比较模式时，当计数器计数值等于 CMP0 时，发生比较事件。 Timer 通道 0 工作在捕获模式时，发生捕获事件时的计数器计数值存入 CMP0 寄存器。

设置 CMP0=0，可使得通道 0 恒为!CH0_POL，设置 CMP0=TH+1，可使得通道 0 恒为 CH0_POL。

13.3.5.5 TIMER3_CMP1 Timer3 通道 1 比较捕获寄存器

地址:0x4001_0910

复位值:0x0

表 13-43 Timer3 通道 1 比较捕获寄存器 TIMER3_CMP1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CMP1															
RW															
0															

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

位置	位名称	说明
[31:0]	CMP1	Timer 通道 1 工作在比较模式时，当计数器计数值等于 CMP1 时，发生比较事件。 Timer 通道 1 工作在捕获模式时，发生捕获事件时的计数器计数值存入 CMP1 寄存器。

设置 CMP1=0，可使得通道恒为!CH1_POL，设置 CMP1=TH+1，可使得通道恒为 CH1_POL。

13.3.5.6 TIMER3_EVT Timer3 外部事件选择寄存器

地址:0x4001_0914

复位值:0x0

表 13-44 Timer3 外部事件选择寄存器 TIMER3_EVT

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

位置	位名称	说明
[31:4]		未使用
[3:0]	EVT_SRC	Timer 外部事件选择寄存器，本寄存器需要配合 TIMER3_CFG.ETON，TIMER3_CFG.GATE_EN，TIMER3_CFG.RL_EN，TIMER3_CFG.XCLK_EN 使用。 通常 4 个设置位不同时使用。 当设置 ETON=1 时，根据本寄存器选择触发 Timer 计数的事件。 需要注意的是，Timer 自身的比较事件无法触发 Timer 进行计数。当 ETON=1 时，触发信号的上升沿触发 Timer 开始计数，用作触发源的 Timer 需要工作在比较模式，无须从外部输入信号至 Timer 通道 0:TIMER0 通道 0 比较/捕获事件 1:TIMER0 通道 1 比较/捕获事件 2:TIMER1 通道 0 比较/捕获事件 3:TIMER1 通道 1 比较/捕获事件

		4:TIMER2 通道 0 比较/捕获事件 5:TIMER2 通道 1 比较/捕获事件 6:不可用 7:不可用 8:CLU0 输出上升沿 9:CLU1 输出上升沿 10:CLU2 输出上升沿 11:CLU3 输出上升沿 12:MCPWM TADC[0]比较事件 13:MCPWM TADC[1]比较事件 14:MCPWM TADC[2]比较事件 15:MCPWM TADC[3]比较事件 当 GATE_EN=1，且外部信号为高时 Timer 计数，外部信号为低时，Timer 暂停，当 GATE_EN=1 时，Timer 使用对应外部信号的电平信号作为计数门控。GATE_EN 仅支持配合 0~11 的外部事件使用，当 EVT_SRC 选择为 0~7 时，外部信号为 Timer 通道输入信号经过对应 Timer 滤波后的信号，作为门控信号的 Timer 通道需要配置为输入使能。 0:TIMER0 通道 0 滤波后输入信号 1:TIMER0 通道 1 滤波后输入信号 2:TIMER1 通道 0 滤波后输入信号 3:TIMER1 通道 1 滤波后输入信号 4:TIMER2 通道 0 滤波后输入信号 5:TIMER2 通道 1 滤波后输入信号 6:TIMER3 通道 0 滤波后输入信号 7:TIMER3 通道 1 滤波后输入信号 8:CLU0 输出信号 9:CLU1 输出信号 10:CLU2 输出信号 11:CLU3 输出信号 当 RL_EN 为高时，Timer 使用外部信号作为重装载信号 作为重装载信号的 Timer 通道需要配置为输入使能 0:TIMER0 通道 0 比较/捕获事件 1:TIMER0 通道 1 比较/捕获事件 2:TIMER1 通道 0 比较/捕获事件 3:TIMER1 通道 1 比较/捕获事件 4:TIMER2 通道 0 比较/捕获事件 5:TIMER2 通道 1 比较/捕获事件 6:TIMER3 通道 0 比较/捕获事件 7:TIMER3 通道 1 比较/捕获事件 8:CLU0 输出 9:CLU1 输出 10:CLU2 输出 11:CLU3 输出

		12:MCPWM TADC[0]比较事件 13:MCPWM TADC[1]比较事件 14:MCPWM TADC[2]比较事件 15:MCPWM TADC[3]比较事件 当 XCLK_EN 为高时，Timer 使用外部信号作为 Timer 外部时钟，Timer 在时钟上升沿进行计数动作。如果使用 Timer 通道作为外部时钟，通道信号受到对应 Timer 滤波设置影响。 0:TIMER0 通道 0 输入信号 1:TIMER0 通道 1 输入信号 2:TIMER1 通道 0 输入信号 3:TIMER1 通道 1 输入信号 4:TIMER2 通道 0 输入信号 5:TIMER2 通道 1 输入信号 6:TIMER3 通道 0 输入信号 7:TIMER3 通道 1 输入信号 8:CLU0 输出 9:CLU1 输出 10:CLU2 输出 11:CLU3 输出 12:MCPWM TADC[0]比较事件 13:MCPWM TADC[1]比较事件 14:MCPWM TADC[2]比较事件 15:MCPWM TADC[3]比较事件

13.3.5.7 TIMER3_FLT Timer3 滤波控制寄存器

地址:0x4001_0918

复位值:0x0

表 13-45 Timer3 滤波控制寄存器 TIMER3_FLT

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

位置	位名称	说明
[31:8]		未使用
[7:0]	FLT	通道 0/1 信号滤波宽度选择。取值范围 0~255。 FLT 为 0 时，不对通道进行滤波。 FLT 不为 0 时，对通道信号进行滤波:滤波宽度为 FLT×8。当通道信号电平稳定超过 FLT×8 个系统时钟周期宽度时，滤波器输出更新；否则，滤波器保持当前的输出不变。

注意，以上滤波器的工作时钟与对应的 Timer 的分频后的工作时钟为相同时钟，受 TIMERx_CFG.CLK_DIV 的分频系数控制。

假设 TIMERx_FLT.FLT = 0x6；TIMERx_CFG.CLK_DIV = 0x2; 则 Timer 的运行时钟相对系统时钟进行了 4 分频。通道输入信号需要经过 8×6 倍 Timer 的运行时钟的滤波，亦即 8×6×4 倍系统时钟的滤波。

13.3.5.8 TIMER3_IE Timer3 中断使能寄存器

地址:0x4001_091C

复位值:0x0

表 13-46 Timer3 中断使能寄存器 TIMER3_IE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					ZC_RE	CH1_RE	CH0_RE						ZC_IE	CH1_IE	CH0_IE
					RW	RW	RW						RW	RW	RW
					0	0	0						0	0	0

位置	位名称	说明
[31:11]		未使用
[10]	ZC_RE	Timer 计数器过 0 DMA 请求使能，高电平有效。
[9]	CH1_RE	Timer 通道 1 比较/捕获 DMA 请求使能，高电平有效。
[8]	CH0_RE	Timer 通道 0 比较/捕获 DMA 请求使能，高电平有效。
[7:3]		未使用
[2]	ZC_IE	Timer 计数器过 0 中断使能，高电平有效。
[1]	CH1_IE	Timer 通道 1 比较/捕获中断使能，高电平有效。
[0]	CH0_IE	Timer 通道 0 比较/捕获中断使能，高电平有效。

DMA 请求事件，与中断为相同事件标志，DMA 请求被 DMA 受理后，中断标志会被 DMA 硬件自动清除，无需 CPU 软件干预。需要注意的是，通常开启某个事件的 DMA 请求则关闭对应的中断使能。

13.3.5.9 TIMER3_IF Timer3 中断标志寄存器

地址:0x4001_0920

复位值:0x0

表 13-47 Timer3 中断标志寄存器 TIMER3_IF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													ZC_IF	CH1_IF	CH0_IF

	RW1C	RW1C	RW1C
	0	0	0

位置	位名称	说明
[31:3]		未使用
[2]	ZC_IF	Timer 计数器过 0 中断标志。高电平有效，写 1 清 0
[1]	CH1_IF	Timer 通道 1 比较/捕获中断标志。高电平有效，写 1 清 0
[0]	CH0_IF	Timer 通道 0 比较/捕获中断标志。高电平有效，写 1 清 0

在比较模式下，当 Timer 设置为中心计数（0->TH->0），即 TIMERx_CFG.CENTER=1 时。CH0_IF 和 CH1_IF 只有在 Timer 从 0 递增计数至 TH 的过程中，当 TIMERx_CNT=TIMERx_CMP0/1 时置位；在 Timer 从 TH 递减计数到 0 的过程中不会置位。

中断标志写 1 清零，一般不建议用如下|=方式清零，因为|=是先读取中断标志，将对应位改为 1 再写入清零，如果同时有其他中断标志置位，会被一起清零，而这通常不是软件所期望的。例如，如下写法本意是清零 ZC_IF，但如果同时 CH0_IF 在写入执行前置 1 了，则软件先读取回 TIMERx_IF 值为 0x24，然后执行或操作 0x4|0x1=0x5，然后写入，同时对 CH0_IF 和 ZC_IF 进行了清零，可能导致 Timer 少进入一次因捕获而产生的中断。

TIMERx_IF|=0x4;

如果希望清零 T0_ZC_IF 标志位，应以如下方式，直接对 BIT2 写 1.

TIMERx_IF=0x4;

13.3.6 QEP0 寄存器

13.3.6.1 QEP0_CFG QEP0 配置寄存器

QEP0_CFG 地址：0x4001_0A00

复位值：0x0

表 13-48 QEP0 配置寄存器 QEP0_CFG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN					FE_CNT_EN	MODE						ZNC	ZPC	ZLC	ZEC
RW					RW	RW						RW	RW	RW	RW
0					0	0						0	0	0	0

位置	位名称	说明
[31:16]		未使用
[15]		QEP 模块整体使能
[14:11]		未使用

[10]	FE_CNT_EN	CCW+SIGN/CCW+CW 两种模式下，是否在下降沿进行计数（上升沿总是计数）
[9:8]	MODE	编码器模式选择 00：counting on T1， 01：counting on T1 & T2 以上两种模式都为正交编码信号计数模式 10：CCW+SIGN，符号加脉冲信号计数模式 11：CCW+CW，CCW+CW 双脉冲信号计数模式
[7:4]		未使用
[3]	ZNC	Z 信号清零极性选择：低电平/下降沿清零使能
[2]	ZPC	Z 信号清零极性选择：高电平/上升沿清零使能
[1]	ZLC	Z 信号电平清零 QEP 计数器使能
[0]	ZEC	Z 信号边沿清零 QEP 计数器使能

当 ZEC=1 时，使用 Z 信号的跳变沿进行 QEP 计数的清零，当 ZNC 同时为 1 时，Z 信号的下降沿清零 QEP 计数器，当 ZPC 同时为 1 时，Z 信号的上升沿清零 QEP 计数器；ZNC 和 ZPC 可以同时为 1，则 Z 信号发生变化则清零 QEP 计数器，清零后立即释放复位，QEP 计数器立即准备好进行后续计数。

当 ZLC=1 时，使用 Z 信号的有效电平进行 QEP 计数的清零，当 ZNC 同时为 1 时，Z 信号低电平时清零 QEP 计数器，当 ZPC 同时为 1 时，Z 信号高电平清零 QEP 计数器；通常使用 Z 信号有效电平清零 QEP 计数器时，ZNC 和 ZPC 不会同时为 1，否则计数器一直处于清零状态。使用有效电平清零时，只有 Z 信号翻转后，QEP 计数器才会恢复计数。

13.3.6.2 QEP0_TH QEP0 计数门限寄存器

QEP0_TH 地址：0x4001_0A04

复位值：0x0

表 13-49 QEP0 计数门限寄存器 QEP0_TH

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

位置	位名称	说明
[31:16]		未使用
[15:0]	TH	计数门限 TH。编码器向上计数（增）到 TH 值后，再次向上计数会导致计数器回到 0。编码器向下计数（减）到 0 值后，再次向下计数会导致计数器回到 TH。

13.3.6.3 QEP0_CNT QEP0 计数值寄存器

QEP0_CNT 地址：0x4001_0A08

复位值：0x0

表 13-50 QEP0 计数值寄存器 QEP0_CNT

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

位置	位名称	说明
[31:16]		未使用
[15:0]	CNT	QEP0 计数值。

13.3.6.4 QEP0_IE QEP0 中断使能寄存器

地址：0x4001_0A0C

复位值：0x0

表 13-51 QEP0 中断使能寄存器 QEP0_IE

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

位置	位名称	说明
[31:2]		未使用
[1]	OF_IE	上溢出中断使能，高电平有效。
[0]	UF_IE	下溢出中断使能，高电平有效。

13.3.6.5 QEP0_IF QEP0 中断标志寄存器

地址：0x4001_0A10

复位值：0x0

表 13-52 QEP0 中断标志寄存器 QEP0_IF

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

	RW1C	RW1C
	0	0

位置	位名称	说明
[31:2]		未使用
[1]	OF_IF	上溢出中断标志。高电平有效，写 1 清 0。当计数器计数达到计数门限时，上计数事件触发上溢出中断。
[0]	UF_IF	下溢出中断标志。高电平有效，写 1 清 0。当计数器计数达到 0 时，下计数事件触发下溢出中断。

13.3.7 QEP1 寄存器

13.3.7.1 QEP1_CFG QEP1 配置寄存器

QEP1_CFG 地址：0x4001_0B00

复位值：0x0

表 13-53 QEP1 配置寄存器 QEP1_CFG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN					FE_CNT_EN	MODE						ZNC	ZPC	ZLC	ZEC
RW					RW	RW						RW	RW	RW	RW
0					0	0						0	0	0	0

位置	位名称	说明
[31:11]		未使用
[10]	FE_CNT_EN	CCW+SIGN/CCW+CW 两种模式下，是否在下降沿进行计数（上升沿总是计数）
[9:8]	MODE	编码器模式选择 00：counting on T1， 01：counting on T1 & T2 以上两种模式都为正交编码信号计数模式 10：CCW+SIGN，符号加脉冲信号计数模式 11：CCW+CW，CCW+CW 双脉冲信号计数模式
[7:4]		未使用
[3]	ZNC	Z 信号清零极性选择：低电平/下降沿清零使能
[2]	ZPC	Z 信号清零极性选择：高电平/上升沿清零使能
[1]	ZLC	Z 信号电平清零 QEP 计数器使能
[0]	ZEC	Z 信号边沿清零 QEP 计数器使能

当 ZEC=1 时，使用 Z 信号的跳变沿进行 QEP 计数的清零，当 ZNC 同时为 1 时，Z 信号的下降沿清零 QEP 计数器，当 ZPC 同时为 1 时，Z 信号的上升沿清零 QEP 计数器；ZNC 和 ZPC 可以同时为

1，则 Z 信号发生变化则清零 QEP 计数器，清零后立即释放复位，QEP 计数器立即准备好进行后续计数。

当 ZLC=1 时，使用 Z 信号的有效电平进行 QEP 计数的清零，当 ZNC 同时为 1 时，Z 信号低电平时清零 QEP 计数器，当 ZPC 同时为 1 时，Z 信号高电平清零 QEP 计数器；通常使用 Z 信号有效电平清零 QEP 计数器时，ZNC 和 ZPC 不会同时为 1，否则计数器一直处于清零状态。使用有效电平清零时，只有 Z 信号翻转后，QEP 计数器才会恢复计数。

13.3.7.2 QEP1_TH QEP1 计数门限寄存器

QEP1_TH 地址：0x4001_0B04

复位值：0x0

表 13-54 QEP1 计数门限寄存器 QEP1_TH

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

位置	位名称	说明
[31:16]		未使用
[15:0]	TH	QEP 计数门限 TH。编码器向上计数（增）到 TH 值后，再次向上计数会导致计数器回到 0。编码器向下计数（减）到 0 值后，再次向下计数会导致计数器回到 TH。

13.3.7.3 QEP1_CNT QEP1 计数值寄存器

QEP1_CNT 地址：0x4001_0B08

复位值：0x0

表 13-55 QEP1 计数值寄存器 QEP1_CNT

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

位置	位名称	说明
[31:16]		未使用
[15:0]	CNT	QEP 计数值。

13.3.7.4 QEP1_IE QEP1 中断使能寄存器

地址：0x4001_0B0C

复位值：0x0

表 13-56 QEP1 中断使能寄存器 QEP1_IE

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

位置	位名称	说明
[31:2]		未使用
[1]	OF_IE	上溢出中断使能，高电平有效。
[0]	UF_IE	下溢出中断使能，高电平有效。

13.3.7.5 QEP1_IF QEP1 中断标志寄存器

地址：0x4001_0B10

复位值：0x0

表 13-57 QEP1 中断标志寄存器 QEP1_IF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														OF_IF	UF_IF
														RW1C	RW1C
														0	0

位置	位名称	说明
[31:2]		未使用
[1]	OF_IF	上溢出中断标志。高电平有效，写 1 清 0。当计数器计数达到计数门限时，上计数事件触发上溢出中断。
[0]	UF_IF	下溢出中断标志。高电平有效，写 1 清 0。当计数器计数达到 0 时，下计数事件触发下溢出中断。

14 MCPWM

14.1 概述

MCPWM 模块，是一个精确控制电机驱动波形输出的模块。

包含两个 16 位递增计数器，用于提供两个计数周期（以下称为两个时基）。计数器的时钟分频有 1/2/4/8 四种选项，产生的的分频时钟频率分别为 96MHz/48MHz/24MHz/12MHz。通道 0/1/2 固定使用时基 0，通道 3/4/5 固定使用时基 1。

MCPWM 模块共包含 6 个 PWM 生成子模块。

- 可以产生 6 对（互补信号）或 12 路独立（边沿模式）不交叠的 PWM 信号;

- 支持边沿对齐 PWM

- 中心对齐 PWM

- 移相 PWM

同时可以产生 4 路与 MCPWM 同时基的定时信息，用于触发 ADC 模块同步采样，进行与 MCPWM 的联动。

包含一组急停保护模块，用于不依赖 CPU 软件的处理快速关断 MCPWM 模块输出。MCPWM 模块可输入 4 路急停信号，其中 2 路来自芯片 IO，2 路来自片内比较器的输出。当急停事件发生时（支持有效电平极性选择），根据 MCPWM0_FAIL012/MCPWM0_FAIL345/MCPWM0_CH_MSK 的设置关闭相应的 PWM 通道，以避免短路发生。

对急停信号有独立滤波模块。

MCPWM 的每个输出 IO 支持两种控制模式:PWM 硬件控制或者软件直接控制（用于 EABS 软刹车，或 BLDC 方波换相控制）。

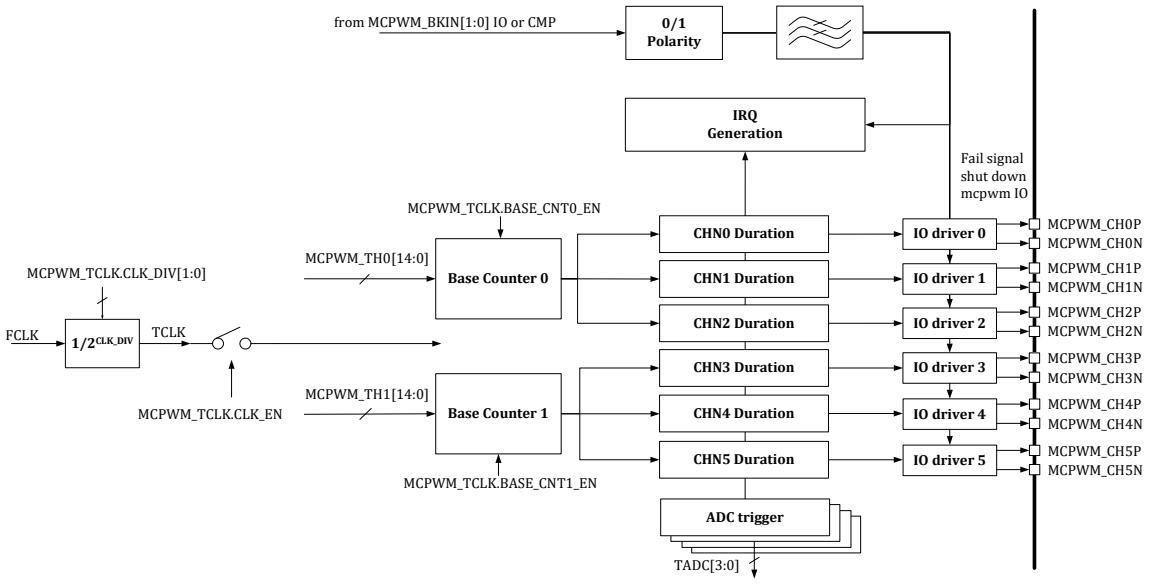


图 14-1 MCPWM 模块框图

为了保证定时精度，通常采用 96MHz 的时钟作为 MCPWM 模块工作频率。

14.1.1 Base Counter 模块

该模块主要是由两个递增计数器组成，其计数门限值为 MCPWM0_TH0/MCPWM0_TH1，计数器从 t0 时刻开始从-TH 递增计数，在 t1 时刻过 0 点，在 t2 时刻计数到 TH 完成一次计数循环，回到-TH，重新开始计数。计数周期为(TH×2+1)×计数时钟周期。

在 t0/t1(本次 t0 即上一次 t2)可产生定时事件中断，MCPWM0_IFx.T0_IF 和 MCPWM0_IFx.T1_IF 将被置位。

可通过寄存器配置 MCPWM0_TCLK.BASE_CNT0_EN/MCPWM0_TCLK.BASE_CNT1_EN 控制 Base Counter 0/1 的启动和停止。

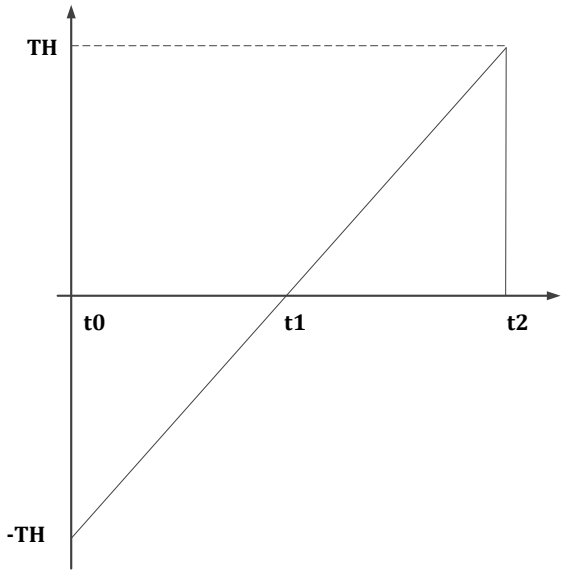


图 14-2 Base Counter t0/t1 时序

在运行 MCPWM 模块前，用户一般需要将门限值 MCPWM0_TH00~MCPWM0_TH51，死区寄存器 MCPWM0_DTH00/ MCPWM0_DTH01/ MCPWM0_DTH10/ MCPWM0_DTH11 配置好。在实际运行过程中，也可动态改变比较门限值和死区寄存器。这些配置好的寄存器，可通过写 MCPWM0_UPDATE 寄存器实现手动更新，也可以通过配置 MCPWM0_SDCFG.T1_UPDATE_EN 及 MCPWM0_SDCFG.T0_UPDATE_EN 进行硬件自动更新。硬件更新，仅在 t0 及 t1 时刻（可配置 t0 或 t1 更新和 t0 t1 时刻都更新）才能产生更新事件，硬件把加载寄存器的值载入到实际运行的寄存器中。而更新事件的发生频率可以配置，即每间隔 N 个 t0 及 t1 时刻才发生更新。无论是否发生更新，t0/t1 时刻均可产生相应的中断。若硬件把加载寄存器的值到载入实际运行的寄存器后，产生装载中断。

通过配置 MCPWM0_SDCFG 寄存器选择更新发生在 t0 或者 t1 或者 t0/t1 都更新，配置更新间隔数，间隔数为 1~16。最频繁的更新配置为更新发生在 t0 和 t1，连续发生。最低速的更新配置为更新发生在 t1，每 16 个 t1 更新一次。

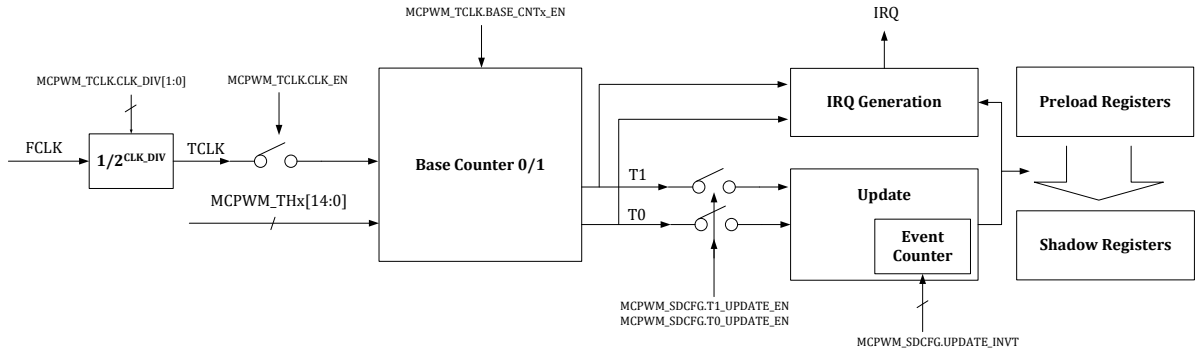


图 14-3 MCPWM 更新机制

时基 0 和时基 1 对应的 TH 寄存器 MCPWM0_THx 和计数寄存器 MCPWM0_CNTx（x=0,1）均存在影子寄存器，且均支持手动更新（通过软件向 MCPWM0_UPDATE 写入对应位进行更新）和自动更新（发生特定硬件事件触发更新）。影子寄存器可以在不开启 MCPWM TCLK 时钟（即 MCPWM0_TCLK.TCLK_EN=0）的情况下进行更新。MCPWM0_TCLK.BASE_CNT0_EN 和 MCPWM0_TCLK.BASE_CNT1_EN 分别控制时基 0 和时基 1 的计数使能。当 MCPWM0_TCLK.BASE_CNT0_EN=0 时，时基 0 的 CNT 在完成更新后保持值不变，时基 1 的 CNT 同理，实现完全相同。

此外，为了精确控制第一个 MCPWM 周期的计数，建议将 MCPWM0_THx 和 MCPWM0_CNTx 同时更新。当然也支持 MCPWM0_THx 和 MCPWM0_CNTx 分别更新。然后再通过软件写入或外部事件触发计数开始，从而将 MCWPM_TCLk.BASE_CNT0_EN/MCWPM_TCLk.BASE_CNT1_EN 置位，分别使能时基 0 和时基 1 的 CNT 计数。

14.1.2 FAIL 信号处理

FAIL 信号即急停信号，主要用于在出现异常时迅速关断功率管，以免造成不可逆的硬件损坏。该信号处理模块主要是根据实际情况设置急停事件，实现快速关断 PWM 的输出。有 2 路 fail 信号输入 MCPWM，即 FAIL0~FAIL1，分别可以来自芯片 I0 MCPWM_BKIN[1:0]或芯片内部比较器的输出 CMP[1:0]。

其中 MCPWM 的 P/N 通道 0/1/2 可以选择使用 CMP[0]或 MCPWM_BKIN[0],MCPWM 的 P/N 通道 3 可以选择使用 CMP[1]或 MCPWM_BKIN[1]。

07x MCU FAIL 信号分布表

	MCPWM CH0/1/2		MCPWM CH3/4/5	
	Fail0	Fail1	Fail2	Fail3
MCPWM_BKIN0	√		√	
MCPWM_BKIN1		√		√
CMP0	√		√	
CMP1		√		√
CLUOUT0	√		√	

CLUOUT1	√		√	
CLUOUT2		√		√
CLUOUT3		√		√

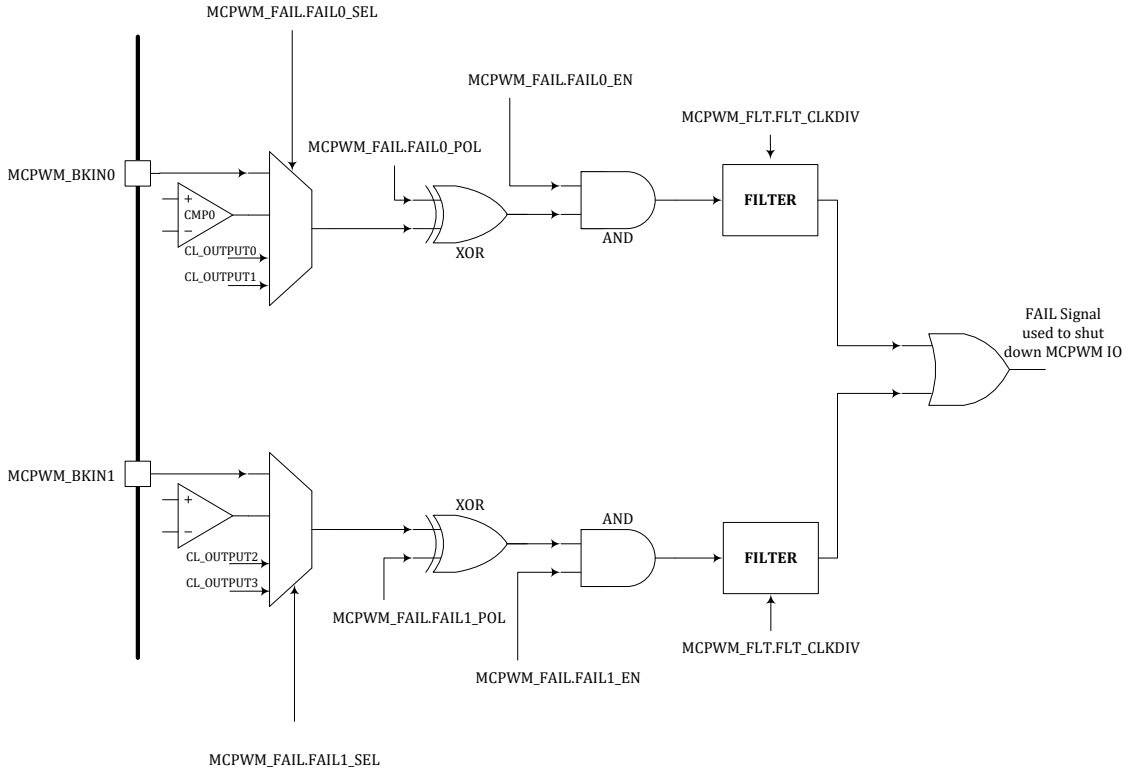


图 14-4 MCPWM FAIL 逻辑示意图

Filter 滤波模块的时钟，来自系统主时钟 MCLK 的门控时钟 FCLK，见 SYS_CLK_FEN，并经过可选的两级分频，第一级分频由 MCPWM0_TCLK.CLK_DIV 控制，进行 1/2/4/8 倍分频。第二级分频可实现 1~256 倍的分频，使用 MCPWM0_FLT.FLT_CLKDIV[7:0]作为第二级的分频系数；如图 14-5 所示。

MCPWM 模块使用分频后的时钟进行 Fail 信号滤波，滤波宽度固定为 16 个周期，即输入信号必须保持至少 16 个分频时钟周期（两级分频后的时钟）稳定后，硬件才判定其为有效输入信号。滤波时间常数的公式为，其中 T_{MCLK} 为 MCLK/FCLK[4]的时钟周期，96MHz 对应 10.42ns。

$$T = T_{MCLK} \times (MCPWM0\_TCLK.CLK\_DIV) \times (FLT\_CLKDIV+1) \times 16$$

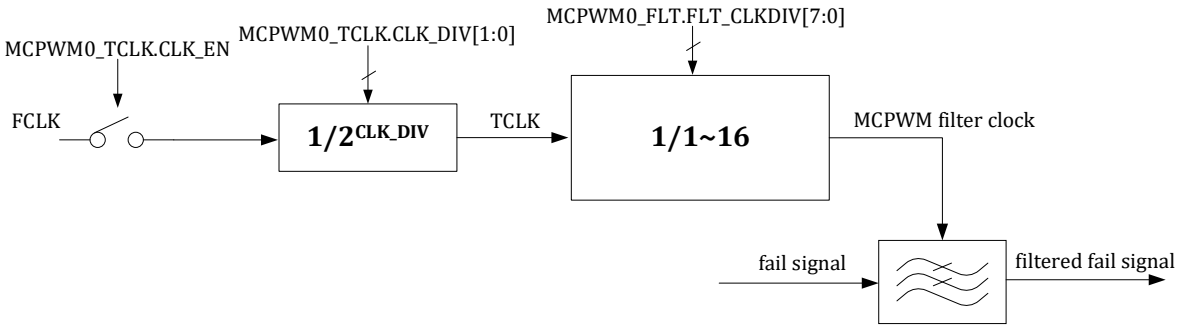


图 14-5 MCPWM Fail 信号滤波时钟生成逻辑

一旦发生 FAIL 事件，硬件将 IO 输出强制变为 MCPWM0_FAILx.CHxN_DEFAULT 和 MCPWM0_FAILx.CHxP_DEFAULT 寄存器所指定的故障缺省值，此时 MCPWM0_FAILx.CHxN_DEFAULT 和 MCPWM0_FAILx.CHxP_DEFAULT 的值直接输出到 IO 口，不再受到 MCPWM0_FAILx.FAIL_POL 等极性控制的影响。其中 MCPWM0_FAIL012 用于控制通道 0/1/2 的故障缺省设置，MCPWM0_FAIL345 用于控制通道 3/4/5 的故障缺省设置。

来自比较器的 MCPWM FAIL 信号为模拟比较器输出的原始信号，未经过比较器数字接口模块的滤波处理，但是可以被 MCPWM 的通道信号进行开窗控制，开窗控制设置见比较器数字接口模块。FAIL 信号进入 MCPWM 模块后，可以通过设置 MCPWM_TCLK 进行滤波。

14.1.3 MCPWM 特殊输出状态

电机控制中经常会用到全 0 和全 1 输出状态，以下互补模式设置可以得到期望的输出。

1. 如果 THn0≥THn1，芯片处于恒 0 状态（CH<n>P 关闭，CH<n>N 开启），无死区
2. 如果 THn0=-TH，THn1=TH，芯片处于恒 1 状态（CH<n>P 开启，CH<n>N 关闭），无死区

14.1.4 IO DRIVER 模块

该模块根据实际 MCPWM 的寄存器配置情况，将 IO 设置到相应电平。IO Driver 模块的整体数据流程图如下:

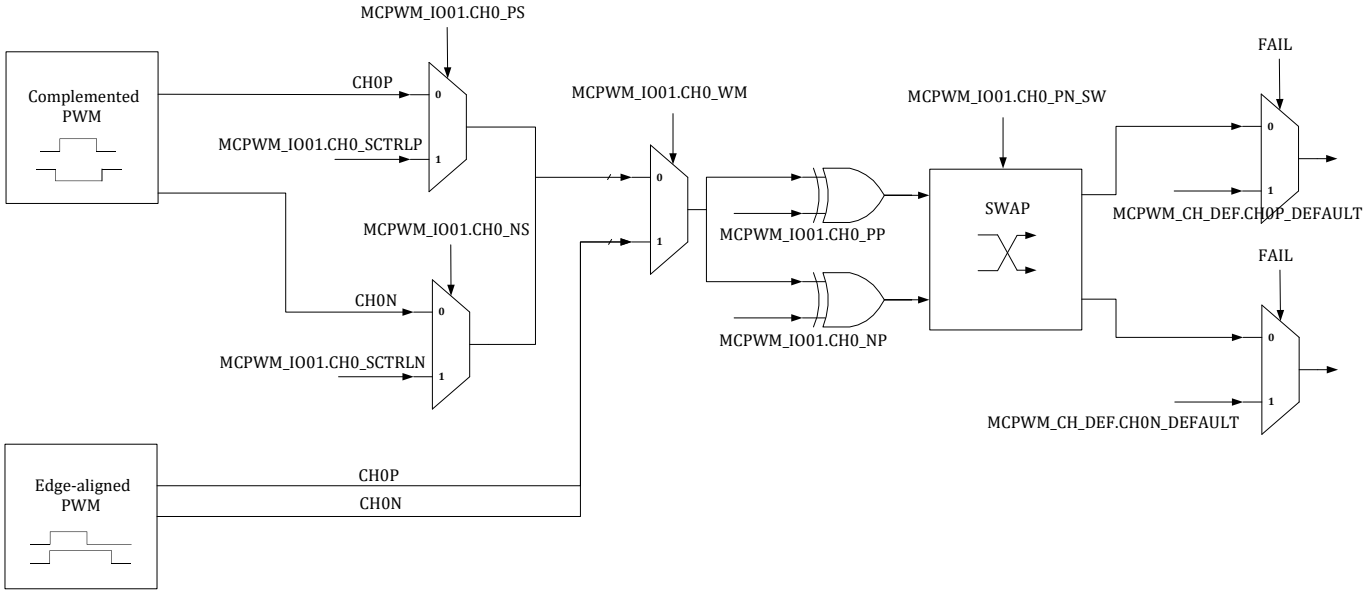


图 14-6 IO Driver 模块数据流程图

14.1.4.1 MCPWM 波形输出-中心对齐模式

4 个 MCPWM IO Driver 采用独立的控制门限，独立死区宽度（每一对互补 IO 的死区需要独立配置，即 4 个死区配置寄存器），共享数据更新事件。

采用 TH<n>0 和 TH<n>1 控制第<n>个 MCPWM IO 的启动、关闭动作，n 为 0/1/2/3/4/5。

当计数器 CNT 值向上计数达到 TH<n>0(即 t3 时刻)，关闭 CH<n>N，经过死区延时 DTH0，打开 CH<n>P。

当计数器 CNT 值向上计数达到 TH<n>1(即 t4 时刻)，关闭 CH<n>P，经过死区延 DTH1，打开 CH<n>N。

采用独立的启动和关闭时间控制，可以提供相位控制的能力。

死区延时保证 CH<n>P/CH<n>N 不会同时打开，避免短路发生。

t3/t4 时刻均会产生相应中断。

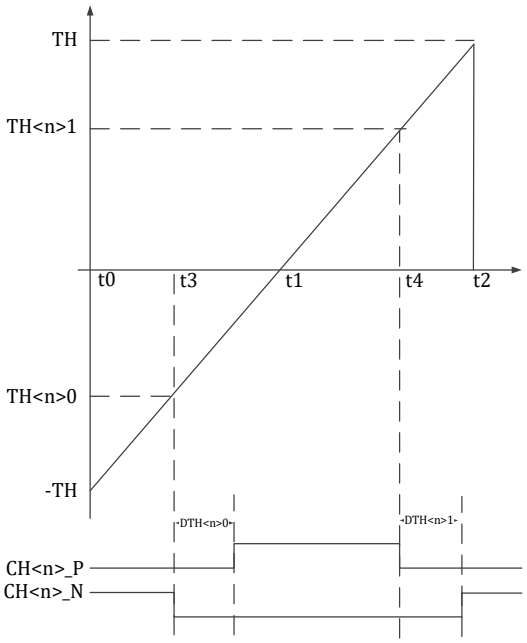


图 14-7MCPWM 时序 TH<n>0 和 TH<n>1-中心对齐模式

14.1.4.2 MCPWM 波形控制-中心对齐推挽模式

中心对齐推挽模式。第一个周期内，在 t3 时刻 CH<n>P 打开，在 t4 时刻，CH<n>P 关闭。第二个周期内，在 t3 时刻 CH<n>N 打开，在 t4 时刻，CH<n>N 关闭。

t3/t4 时刻均会产生相应中断。

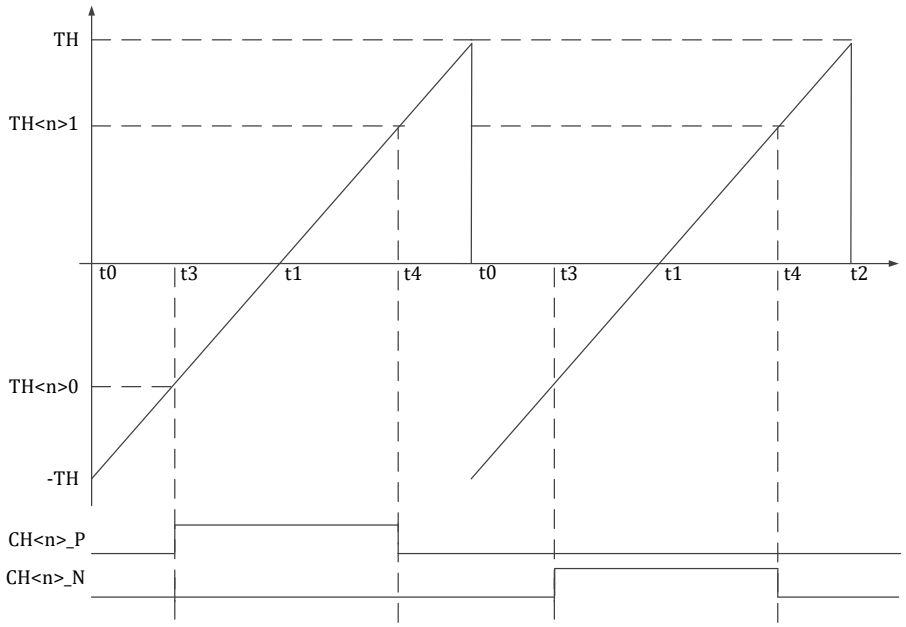


图 14-8 MCPWM 时序 TH<n>0 和 TH<n>1-中心对齐推挽模式

14.1.4.3 MCPWM 波形输出-边沿对齐模式

边沿对齐模式。在 t0 时刻 CH<n>P/CH<n>N 同时打开，在 t3 时刻，CH<n>P 关闭；在 t4 时刻，CH<n>N 关闭。

t3/t4 时刻均会产生相应中断。

边沿对齐模式下，CH<n>P/CH<n>N 无需死区保护。

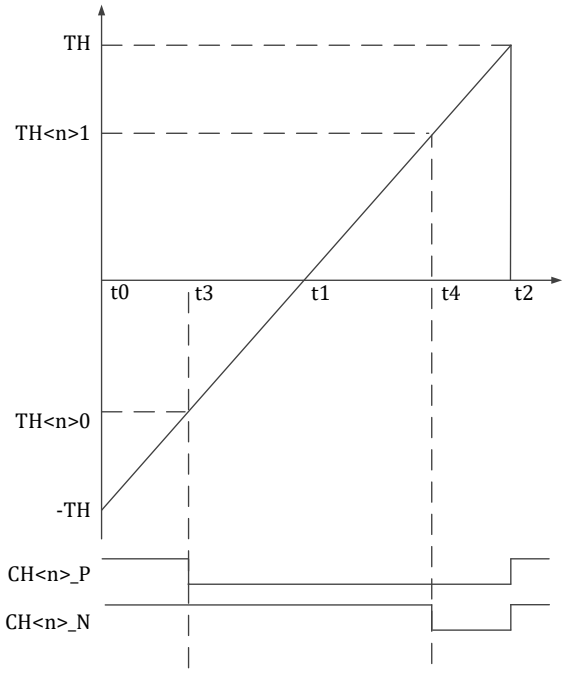


图 14-9MCPWM 时序边沿对齐模式

14.1.4.4 MCPWM 波形控制-边沿对齐推挽模式

边沿对齐推挽模式。第一个周期内，在 t0 时刻 CH<n>P 打开，在 t3 时刻，CH<n>P 关闭。第二个周期内，在 t0 时刻 CH<n>N 打开，在 t3 时刻，CH<n>N 关闭。

t0/t3 时刻均会产生相应中断。

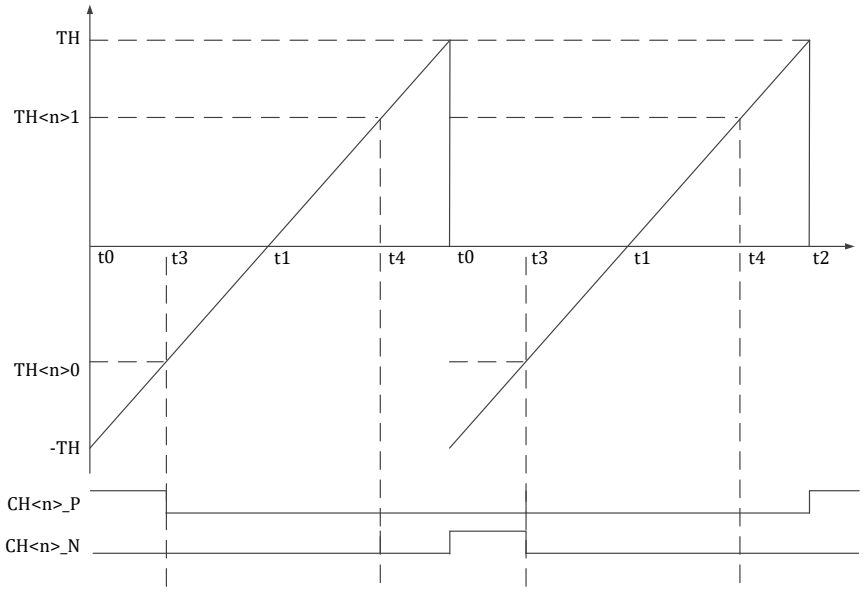


图 14-10MCPWM 时序 TH<n>0 和 TH<n>1 边沿对齐推挽模式

14.1.4.5 MCPWM IO 死区控制

MCPWM IO 是一对互斥控制信号 CH<n>P/CH<n>N，控制如下图所示的电路，

当 CH<n>P 为高/CH<n>N 为低时，Vout 输出高（VDD）；

当 CH<n>P 为低/CH<n>N 为高时，Vout 输出低（VSS）；

当 CH<n>P 为高/CH<n>N 为高时，Vout 输出不确定，但是会产生 VDD 到 VSS 的短路；

当 CH<n>P 为低/CH<n>N 为低时，Vout 输出不确定。

必须避免 CH<n>P/CH<n>N 同时打开的情况。死区的引入，可以有效避免 VDD 到 VSS 的短路。

四组 MCPWM IO 的死区宽度共享同一个死区宽度设置 MCPWM0_DTH0/1。

对于互补模式 MCPWM IO 自动插入死区。

对于边沿对齐模式，MCPWM IO 无死区。

在 IO Driver 模块中增加 CH<n>P/CH<n>N 冲突检测，发生冲突时，自动将 IO 拉低，同时给出错误中断（错误中断标志位可软件清除或者硬件自动清除）。

MCPWM IO 也可通过软件配置的方式输出，此时，死区控制通过软件实现，如果 MCPWM 模块配置为中心对齐模式，硬件短路保护机制仍有效，保证 P 和 N 不同时打开。

CH<n>P/CH<n>N，在 IO 上可以互换。

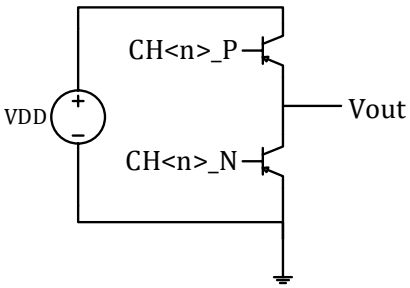


图 14-11MCPWM IO 控制示意图

14.1.4.6 MCPWM IO 极性设置

CH<n>P/CH<n>N 的有效电平可以配置为高有效/低有效，每个 IO 的有效电平单独可配。CH<n>P/CH<n>N 输出到 IO 的位置通过软件配置可以互换。

14.1.4.7 MCPWM IO 自动保护

当发生急停事件（Fail 事件），应立刻将 CH<n>P/CH<n>N 自动切换到默认安全电平状态。需要注意默认电平配置（MCPWM0_CH_DEF.CHxN_DEFAULT 和 MCPWM0_CH_DEF.CHxP_DEFAULT 控制默认电平）。

- 芯片正常工作后，IO 默认输出的电平是寄存器 MCPWM0_CH_DEF.CHxN_DEFAULT 和 MCPWM0_CH_DEF.CHxP_DEFAULT 指定值，当用户配置完毕，MCPWM 正常工作后，配置 MCPWM0_FAILx.MCPWM0_OE（即 MOE）为 1，IO 输出电平受到 MCPWM IO 模块控制。
- 当发生 FAIL 短路状况时，硬件立即切换到 IO 默认输出电平。
- 当芯片调试中，CPU 被上位机软件暂停运行时，可暂停 MCPWM 正常输出，转而输出默认电平值。

14.1.5 ADC Trigger Timer 模块

MCPWM 可以提供 ADC 采样控制。当计数器计数到 MCPWM0_TMR0/ MCPWM0_TMR1/ MCPWM0_TMR2/ MCPWM0_TMR3 时，可产生定时事件，触发 ADC 采样。该触发信号可同时输出到 GPIO，便于调试之用。输出的具体 GPIO，参见对应器件的 datasheet。其中 MCPWM0_TMR0/ MCPWM0_TMR1 固定使用时基 0，MCPWM0_TMR2/ MCPWM0_TMR3 可以选择使用时基 0/1，见 14.2.33 MCPWM0_TCLK。

表 14-1 MCPWM 计数器阈值与事件对应表

MCPWMx 时基 0 t0 事件	MCPWMx_CNT0==-MCPWMx_TH0
MCPWMx 时基 0 t1 事件	MCPWMx_CNT0==0
MCPWMx 时基 1 t0 事件	MCPWMx_CNT1==-MCPWMx_TH1
MCPWMx 时基 1 t1 事件	MCPWMx_CNT1==0
MCPWMx TIO0[0] 事件	MCPWMx_CNT0==MCPWMx_TH00
MCPWMx TIO0[1] 事件	MCPWMx_CNT0==MCPWMx_TH01
MCPWMx TIO1[0] 事件	MCPWMx_CNT0==MCPWMx_TH10
MCPWMx TIO1[1] 事件	MCPWMx_CNT0==MCPWMx_TH11

MCPWMx TIO2[0] 事件	MCPWMx_CNT0==MCPWMx_TH20
MCPWMx TIO2[1] 事件	MCPWMx_CNT0==MCPWMx_TH21
MCPWMx TIO3[0] 事件	MCPWMx_CNT1==MCPWMx_TH30
MCPWMx TIO3[1] 事件	MCPWMx_CNT1==MCPWMx_TH31
MCPWMx TIO4[0] 事件	MCPWMx_CNT1==MCPWMx_TH40
MCPWMx TIO4[1] 事件	MCPWMx_CNT1==MCPWMx_TH41
MCPWMx TIO5[0] 事件	MCPWMx_CNT1==MCPWMx_TH50
MCPWMx TIO5[1] 事件	MCPWMx_CNT1==MCPWMx_TH51
MCPWMx TADC[0] 事件	TMR0
MCPWMx TADC [1] 事件	TMR1
MCPWMx TADC [2] 事件	TMR2
MCPWMx TADC [3] 事件	TMR3

14.1.6 数字电源应用

在数字电源应用中，MCPWM 可以和比较器、Timer、DAC 联动，在此种配置下，MCPWM 内部的状态机完全由比较器输出的比较值进行硬件控制，MCPWM0_CNT0/1 不再进行自动递增计数，即 MCPWM 通道输出电平受到比较器输出值控制。

Timer 的过零事件定时触发 DAC 进行数值更新，从而使得 DAC 输出一个递增或递减的斜坡信号，并连接至比较器负端，比较器正端接外部采样信号。待 DAC 信号上升或下降到一定数值时，触发比较器翻转动作，从而进一步触发 MCPWM 内部状态机动作。通过 SYS_AFE_DAC_CTRL.TIMERm_TRIG_DACx 可以设置使用哪个 Timer 的过零事件作为 DAC 数值更新事件，m=0,1,2,3 对应 TIMER0/1/2/3，x=0,1 对应 DAC0 或 DAC1。DAC 数值的更新步长由 SYS_AFE_DAC_CTRL.DACx_STEP 设置。

数字电源应用通常只需要使用 MCPWM 三相通道中的一相，举例来说，可以设置 MCPWM0_TCLK.CMP_CTRL_CNT0=1 使用 CH0N/P 通道，也可以设置 MCPWM0_TCLK.CMP_CTRL_CNT1=1 使用 CH3N/P 通道。通道 1/2/4/5 不支持与比较器联动。

MCPWM 在此类应用中分为 HS_ON、LS_ON、ALL_OFF 三种状态。HS_ON、LS_ON 之间状态切换有最小切换时间，或者可以认为是状态最小停留时间，即当 MCPWM 从 HS_ON 切换到 LS_ON 状态，至少停留一段时间才能再切换回 HS_ON 状态；反之亦然。状态最小停留时间通过 MCPWM0_STT_HYST 设置。

状态切换时，MCPWM 会同步产生 t0 事件（MCPWM0_CNTx==-MCPWM0_THx）或 t1 事件（MCPWM0_CNTx==0）。如果使用 CH0N/P 做数字电源应用，一般需要同步设置 MCPWM0_RE.TR0_T0_RE=1，MCPWM0_RE.TR0_T1_RE=1，使能 MCPWM 时基 0 的 t0/t1 事件的 DMA 触发使能；同理，如果使用 CH3N/P 做数字电源应用，一般需要同步设置 MCPWM0_RE.TR1_T0_RE=1，MCPWM0_RE.TR1_T1_RE=1，使能 MCPWM 时基 1 的 t0/t1 事件的 DMA 触发使能。使得 MCPWM 在刚进入新状态时不久，DMA 将新的 DAC 参考值存入 SYS_AFE_DAC 寄存器，后续 Timer 定时触发时，DAC 输出会以新的 DAC 参考值为起点生成斜坡信号。

受限于比较器资源，通常不会同时设置 MCPWM0_TCLK.CMP_CTRL_CNT0=1 和 MCPWM0_TCLK.CMP_CTRL_CNT0=1。一般同时只有 CH0N/P 或 CH3N/P 中的一相通道用作数字电源应用。

HS_ON 状态：PWM 上管输出为高，下管输出低

Timer 清零开始计数，定时触发 DAC0 输入数字量 SYS_AFE_DAC0 更新，DAC0 产生斜率信号

比较器 CMP1 输出为高时，且满足状态最小停留时间，切换至 LS_ON 状态。先关闭上管，经过死区时间后打开下管。同时，MCPWM 的过零事件触发 DMA 装载 SYS_AFE_DAC0 为 LS_ON 参考值。

过流发生时，MCPWM 高侧低侧通道全关闭。进入 ALL_OFF 状态。

LS_ON 状态：PWM 上管输出为低，下管输出高

Timer 清零开始计数，定时触发 DAC0 输入数字量更新，DAC0 产生斜率信号

比较器 CMP1 输出为低时，且满足状态最小停留时间，切换至 HS_ON 状态。先关闭下管，经过死区时间后打开上管。同时，MCPWM 的过零事件触发 DMA 装载 SYS_AFE_DAC0 为 HS_ON 参考值。

过流发生时，高侧低侧开关全关闭。进入 ALL_OFF 状态。

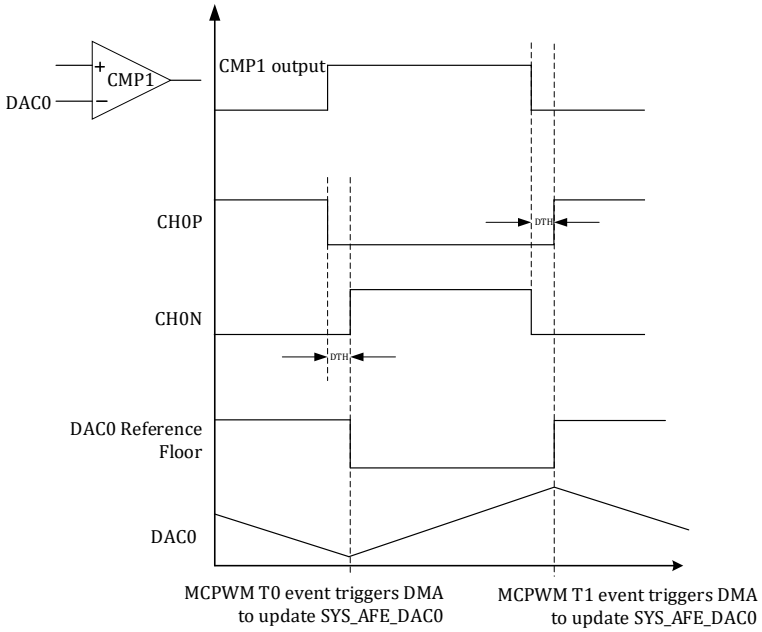


图 14-12 MCPWM 通道 0 受比较器 1 控制进行状态动作时序图

特别地，在 HS_ON 状态时，如果当电流极性信号 CMP2 输出为低时，为 ZCS 事件，ZCS 发生时，需要同时关闭上管和下管，然后延时 Tzcs 后，Tzcs 时间可通过 MCPWM0_ZCS_DELAY 设置，进入 LS_ON 状态。在 LS_ON 状态时，如果当电流极性信号 CMP2 输出为高时，亦为 ZCS 事件，需要同时关闭上管和下管，然后延时 Tzcs 后，进入 HS_ON 状态。

Tzcs 时间可通过 MCPWM0_ZCS_DELAY 设置。

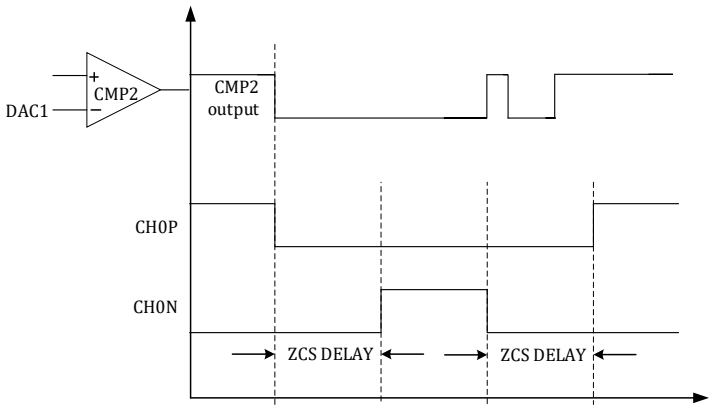


图 14-13 比较器 2 触发 ZCS 事件发生时 MCPWM 通道 0 动作时序

14.1.7 中断

MCPMW_IF0 和 MCPWM0_EIF[5:4]、MCPWM0_EIF[8]标志置位且使能的情况下，产生 MCPWM0_IRQ0 中断;

MCPMW_IF1 和 MCPWM0_EIF[7:6] 、MCPWM0_EIF[9]标志置位且使能的情况下，产生 MCPWM0_IRQ1 中断。

14.2 寄存器

14.2.1 地址分配

MCPWM 模块寄存器的基地址是 0x4001_0C00，

寄存器列表如下:

表 14-2 MCPWM 模块寄存器列表

名称	偏移地址	说明
MCPWM0_TH00	0x00	MCPWM CH0_P 比较门限值寄存器
MCPWM0_TH01	0x04	MCPWM CH0_N 比较门限值寄存器
MCPWM0_TH10	0x08	MCPWM CH1_P 比较门限值寄存器
MCPWM0_TH11	0x0C	MCPWM CH1_N 比较门限值寄存器
MCPWM0_TH20	0x10	MCPWM CH2_P 比较门限值寄存器
MCPWM0_TH21	0x14	MCPWM CH2_N 比较门限值寄存器
MCPWM0_TH30	0x18	MCPWM CH3_P 比较门限值寄存器
MCPWM0_TH31	0x1C	MCPWM CH3_N 比较门限值寄存器
MCPWM0_TH40	0x20	MCPWM CH4_P 比较门限值寄存器
MCPWM0_TH41	0x24	MCPWM CH4_N 比较门限值寄存器
MCPWM0_TH50	0x28	MCPWM CH5_P 比较门限值寄存器
MCPWM0_TH51	0x2C	MCPWM CH5_N 比较门限值寄存器
MCPWM0_TMR0	0x30	ADC 采样定时器比较门限 0 寄存器

MCPWM0_TMR1	0x34	ADC 采样定时器比较门限 1 寄存器
MCPWM0_TMR2	0x38	ADC 采样定时器比较门限 2 寄存器
MCPWM0_TMR3	0x3C	ADC 采样定时器比较门限 3 寄存器
MCPWM0_TH0	0x40	MCPWM 时基 0 门限值寄存器
MCPWM0_TH1	0x44	MCPWM 时基 1 门限值寄存器
MCPWM0_CNT0	0x48	MCPWM 时基 0 计数器寄存器
MCPWM0_CNT1	0x4C	MCPWM 时基 1 计数器寄存器
MCPWM0_UPDATE	0x50	MCPWM 加载控制寄存器
MCPWM0_FCNT	0x54	MCPWM FAIL 时刻 CNT 值
MCPWM0_EVT0	0x58	MCPWM 时基 0 外部触发
MCPWM0_EVT1	0x5C	MCPWM 时基 1 外部触发
MCPWM0_DTH00	0x60	MCPWM CH0/1/2 N 通道死区宽度控制寄存器
MCPWM0_DTH01	0x64	MCPWM CH0/1/2 P 通道死区宽度控制寄存器
MCPWM0_DTH10	0x68	MCPWM CH3/4/5 N 通道死区宽度控制寄存器
MCPWM0_DTH11	0x6C	MCPWM CH3/4/5 P 通道死区宽度控制寄存器
MCPWM0_FLT	0x70	MCPWM 滤波时钟分频寄存器
MCPWM0_SDCFG	0x74	MCPWM 加载配置寄存器
MCPWM0_AUEN	0x78	MCPWM 自动更新使能寄存器
MCPWM0_TCLK	0x7C	MCPWM 时钟分频控制寄存器
MCPWM0_IE0	0x80	MCPWM 时基 0 中断控制寄存器
MCPWM0_IF0	0x84	MCPWM 时基 0 中断标志位寄存器
MCPWM0_IE1	0x88	MCPWM 中断控制寄存器
MCPWM0_IF1	0x8C	MCPWM 中断标志位寄存器
MCPWM0_EIE	0x90	MCPWM 异常中断控制寄存器
MCPWM0_EIF	0x94	MCPWM 异常中断标志位寄存器
MCPWM0_RE	0x98	MCPWM DMA 请求控制寄存器
MCPWM0_PP	0x9C	MCPWM 推挽模式使能寄存器
MCPWM0_IO01	0xA0	MCPWM CH0 CH1 IO 控制寄存器
MCPWM0_IO23	0xA4	MCPWM CH2 CH3 IO 控制寄存器
MCPWM0_IO45	0xA8	MCPWM CH4 CH5 IO 控制寄存器
MCPWM0_FAIL012	0xB0	MCPWM CH0 CH1 CH2 短路控制寄存器
MCPWM0_FAIL345	0xB4	MCPWM CH3 CH4 CH5 短路控制寄存器
MCPWM0_CH_DEF	0xB8	MCPWM 短路保护通道输出默认值
MCPWM0_CH_MSK	0xBC	MCPWM 通道屏蔽寄存器
MCPWM0_PRT	0xC0	MCPWM 保护寄存器
MCPWM0_STT_HYST	0xC8	MCPWM 状态停留延时寄存器
MCPWM0_ZCS_DELAY	0xCC	MCPWM ZCS 状态延时寄存器

表 14-3 受 MCPWM0_PRT 保护的寄存器

名称	偏移地址	说明
MCPWM0_TH0	0x30	MCPWM 门限值寄存器
MCPWM0_TH1	0x34	MCPWM 门限值寄存器
MCPWM0_DTH00	0x60	MCPWM CH0/1/2 N 通道死区宽度控制寄存器

MCPWM0_DTH01	0x64	MCPWM CH0/1/2 P 通道死区宽度控制寄存器
MCPWM0_DTH10	0x68	MCPWM CH3/4/5 N 通道死区宽度控制寄存器
MCPWM0_DTH11	0x6C	MCPWM CH3/4/5 P 通道死区宽度控制寄存器
MCPWM0_FLT	0x70	MCPWM 滤波时钟分频寄存器
MCPWM0_SDCFG	0x74	MCPWM 加载配置寄存器
MCPWM0_AUEN	0x78	MCPWM 自动加载使能寄存器
MCPWM0_TCLK	0x7C	MCPWM 时钟分频控制寄存器
MCPWM0_IE0	0x80	MCPWM 时基 0 中断控制寄存器
MCPWM0_IE1	0x88	MCPWM 时基 1 中断控制寄存器
MCPWM0_EIE	0x90	MCPWM 异常中断控制寄存器
MCPWM0_RE	0x98	MCPWMDMA 请求控制寄存器
MCPWM0_PP	0x9C	MCPWM 推挽模式使能寄存器
MCPWM0_IO01	0xA0	MCPWM CH0 CH1 IO 控制寄存器
MCPWM0_IO23	0xA4	MCPWM CH2 CH3 IO 控制寄存器
MCPWM0_IO45	0xA8	MCPWM CH4 CH5 IO 控制寄存器
MCPWM0_FAIL012	0xB0	MCPWM CH0 CH1 CH2 短路控制寄存器
MCPWM0_FAIL345	0xB4	MCPWM CH3 CH4 CH5 短路控制寄存器
MCPWM0_CH_DEF	0xB8	MCPWM 短路保护通道输出值
MCPWM0_CH_MSK	0xBC	MCPWM 通道屏蔽寄存器
MCPWM0_STT_HYST	0xC8	MCPWM 状态停留延时寄存器
MCPWM0_ZCS_DELAY	0xCC	MCPWM ZCS 状态延时寄存器

表 14-4 存在影子寄存器的寄存器

名称	偏移地址	说明
MCPWM0_TH00	0x00	MCPWM CH0_P 比较门限值寄存器
MCPWM0_TH01	0x04	MCPWM CH0_N 比较门限值寄存器
MCPWM0_TH10	0x08	MCPWM CH1_P 比较门限值寄存器
MCPWM0_TH11	0x0C	MCPWM CH1_N 比较门限值寄存器
MCPWM0_TH20	0x10	MCPWM CH2_P 比较门限值寄存器
MCPWM0_TH21	0x14	MCPWM CH2_N 比较门限值寄存器
MCPWM0_TH30	0x18	MCPWM CH3_P 比较门限值寄存器
MCPWM0_TH31	0x1C	MCPWM CH3_N 比较门限值寄存器
MCPWM0_TH40	0x20	MCPWM CH4_P 比较门限值寄存器
MCPWM0_TH41	0x24	MCPWM CH4_N 比较门限值寄存器
MCPWM0_TH50	0x28	MCPWM CH5_P 比较门限值寄存器
MCPWM0_TH51	0x2C	MCPWM CH5_N 比较门限值寄存器
MCPWM0_TMR0	0x30	ADC 采样定时器比较门限 0 寄存器
MCPWM0_TMR1	0x34	ADC 采样定时器比较门限 1 寄存器
MCPWM0_TMR2	0x38	ADC 采样定时器比较门限 2 寄存器
MCPWM0_TMR3	0x3C	ADC 采样定时器比较门限 3 寄存器
MCPWM0_TH0	0x40	MCPWM 时基 0 门限值寄存器
MCPWM0_TH1	0x44	MCPWM 时基 1 门限值寄存器
MCPWM0_CNT0	0x48	MCPWM 时基 0 计数器寄存器

MCPWM0_CNT1	0x4C	MCPWM 时基 1 计数器寄存器

对于所有存在影子寄存器的 MCPWM 配置寄存器，写入时，写入的是预装载寄存器，更新事件发生时才会把预装载寄存器值写入影子寄存器，读出时读出的是影子寄存器的值。

14.2.2 MCPWM0_TH00

无写保护的寄存器

地址:0x4001_0C00

复位值:0x0

表 14-5 MCPWM0_TH00 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	TH00	MCPWM CH0_P 比较门限值，16 位有符号数；发生更新事件时，本寄存器加载到 MCPWM 实际运行系统中。写入的是预装载寄存器，更新事件发生时才会把预装载寄存器值写入影子寄存器，读出时读出的是影子寄存器的值。

14.2.3 MCPWM0_TH01

无写保护的寄存器

地址:0x4001_0C04

复位值:0x0

表 14-6 MCPWM0_TH00 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	TH01	MCPWM CH0_N 比较门限值，16 位有符号数；发生更新事件时，本寄存器加载到 MCPWM 实际运行系统中。

14.2.4 MCPWM0_TH10

无写保护的寄存器

地址:0x4001_0C08

复位值:0x0

表 14-7 MCPWM0_TH10 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
15:0]	TH10	MCPWM CH1_P 比较门限值，16 位有符号数；发生更新事件时，本寄存器加载到 MCPWM 实际运行系统中。

14.2.5 MCPWM0_TH11

无写保护的寄存器

地址:0x4001_0C0C

复位值:0x0

表 14-8 MCPWM0_TH11 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	TH11	MCPWM CH1_N 比较门限值，16 位有符号数；发生更新事件时，本寄存器加载到 MCPWM 实际运行系统中。

14.2.6 MCPWM0_TH20

无写保护的寄存器

地址:0x4001_0C10

复位值:0x0

表 14-9 MCPWM0_TH20 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	TH20	MCPWM CH2_P 比较门限值，16 位有符号数；发生更新事件时，本寄存器加载到 MCPWM 实际运行系统中。

14.2.7 MCPWM0_TH21

无写保护的寄存器

地址:0x4001_0C14

复位值:0x0

表 14-10 MCPWM0_TH21 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	TH21	MCPWM CH2_N 比较门限值，16 位有符号数；发生更新事件时，本寄存器加载到 MCPWM 实际运行系统中。

14.2.8 MCPWM0_TH30

无写保护的寄存器

地址:0x4001_0C18

复位值:0x0

表 14-11 MCPWM0_TH30 配置寄存器

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

位置	位名称	说明

[31:16]		未使用
[15:0]	TH30	MCPWM CH3_P 比较门限值，16 位有符号数；发生更新事件时，本寄存器加载到 MCPWM 实际运行系统中。

14.2.9 MCPWM0_TH31

无写保护的寄存器

地址:0x4001_0C1C

复位值:0x0

表 14-12 MCPWM0_TH31 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	TH31	MCPWM CH3_N 比较门限值，16 位有符号数；发生更新事件时，本寄存器加载到 MCPWM 实际运行系统中。

14.2.10 MCPWM0_TH40

无写保护的寄存器

地址:0x4001_0C20

复位值:0x0

表 14-13 MCPWM0_TH40 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	TH40	MCPWM CH4_P 比较门限值，16 位有符号数；发生更新事件时，本寄存器加载到 MCPWM 实际运行系统中。

14.2.11 MCPWM0_TH41

无写保护的寄存器

地址:0x4001_0C24

复位值:0x0

表 14-14 MCPWM0_TH41 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	TH41	MCPWM CH4_N 比较门限值，16 位有符号数；发生更新事件时，本寄存器加载到 MCPWM 实际运行系统中。

14.2.12 MCPWM0_TH50

无写保护的寄存器

地址:0x4001_0C28

复位值:0x0

表 14-15 MCPWM0_TH50 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	TH50	MCPWM CH5_P 比较门限值，16 位有符号数；发生更新事件时，本寄存器加载到 MCPWM 实际运行系统中。

14.2.13 MCPWM0_TH51

无写保护的寄存器

地址:0x4001_0C2C

复位值:0x0

表 14-16 MCPWM0_TH51 配置寄存器

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

位置	位名称	说明

[31:16]		未使用
[15:0]	TH51	MCPWM CH5_N 比较门限值，16 位有符号数；发生更新事件时，本寄存器加载到 MCPWM 实际运行系统中。

14.2.14 MCPWM0_TMR0

无写保护的寄存器

地址:0x4001_0C30

复位值:0x0

表 14-17 MCPWM0_TMR0 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	TMR0	ADC 采样定时器比较门限 0 寄存器，16 位有符号数；当 MCPWM0_CNT0=TMR0 时产生 TADC[0]事件触发 ADC 进行采样。MCPWM 发生更新事件后，本寄存器加载到 MCPWM 实际运行系统中。

14.2.15 MCPWM0_TMR1

无写保护的寄存器

地址:0x4001_0C34

复位值:0x0

表 14-18 MCPWM0_TMR1 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	TMR1	ADC 采样定时器比较门限 1 寄存器，16 位有符号数；当 MCPWM0_CNT=TMR1 时产生 TADC[1]事件触发 ADC 进行采样。MCPWM 发生更新事件后，本寄存器加载到 MCPWM 实际运行系统中。

14.2.16 MCPWM0_TMR2

无写保护的寄存器

地址:0x4001_0C38

复位值:0x0

表 14-19 MCPWM0_TMR2 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	TMR2	ADC 采样定时器比较门限 2 寄存器，16 位有符号数；发生更新事件后，本寄存器加载到 MCPWM 实际运行系统中。

14.2.17 MCPWM0_TMR3

无写保护的寄存器

地址:0x4001_0C3C

复位值:0x0

表 14-20 MCPWM0_TMR3 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	TMR3	ADC 采样定时器比较门限 3 寄存器，16 位有符号数；发生更新事件后，本寄存器加载到 MCPWM 实际运行系统中。。

14.2.18 MCPWM0_TH0

写保护的寄存器

地址:0x4001_0C40

复位值:0x0

表 14-21 MCPWM0_TH0 时基 0 寄存器

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

位置	位名称	说明
[31:15]		未使用
[14:0]	TH	MCPWM 时基 0 计数器门限值，15 位无符号数，MCPWM 实际运行系统中的时基 0 计数器从-TH 计数到 TH；发生更新事件后，本寄存器加载到 MCPWM 实际运行系统中。

14.2.19 MCPWM0_TH1

写保护的寄存器

地址:0x4001_0C44

复位值:0x0

表 14-22 MCPWM0_TH1 时基 1 寄存器

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

位置	位名称	说明
[31:15]		未使用
[14:0]	TH	MCPWM 时基 1 计数器门限值，15 位无符号数，MCPWM 实际运行系统中的时基 1 计数器从-TH 计数到 TH；发生更新事件后，本寄存器加载到 MCPWM 实际运行系统中。。

14.2.20 MCPWM0_CNT0

无写保护的寄存器

地址:0x4001_0C48

复位值:0x8000

表 14-23 MCPWM0_CNT0 寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	CNT	向此寄存器写入，更改时基 0 计数器的设定值，发生更新事件后，本寄存器加载到 MCPWM 实际运行系统的时基 0 CNT 中。 读出的数据为 MCPWM 实际运行系统中时基 0 计数器的值。实际读出的计数范围为-TH ~ +TH

14.2.21 MCPWM0_CNT1

无写保护的寄存器

地址:0x4001_0C4C

复位值:0x8000

表 14-24 MCPWM0_CNT1 寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	CNT	向此寄存器写入，更改时基 1 计数器的设定值，发生更新事件后，本寄存器加载到 MCPWM 实际运行系统的时基 1 CNT 中。 读出的数据为 MCPWM 实际运行系统中时基 1 计数器的值。实际读出的计数范围为-TH ~ +TH

14.2.22 MCPWM0_UPDATE

无写保护的寄存器

地址:0x4001_0C50

复位值:0x0

表 14-25 MCPWM0_UPDATE MCPWM 手动更新寄存器

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
				CNT1_UPDATE	TH1_UPDATE	TMR3_UPDATE	TMR2_UPDATE			TH51_UPDATE	TH50_UPDATE	TH41_UPDATE	TH40_UPDATE	TH31_UPDATE	TH30_UPDATE
				WO	WO	WO	WO			WO	WO	WO	WO	WO	WO
				0	0	0	0			0	0	0	0	0	0

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

	CNT0_UPDATE	TH0_UPDATE	TMR1_UPDATE	TMR0_UPDATE		TH21_UPDATE	TH20_UPDATE	TH11_UPDATE	TH10_UPDATE	TH01_UPDATE	TH00_UPDATE
	WO	WO	WO	WO		WO	WO	WO	WO	WO	WO
	0	0	0	0		0	0	0	0	0	0

位置	位名称	说明
[31:28]		未使用
[27]	CNT1_UPDATE	写 1 产生软件（手动）触发，将 MCPWM0_CNT1 由预装载寄存器加载到 MCPWM 运行所使用的影子寄存器
[26]	TH1_UPDATE	写 1 产生软件（手动）触发，将 MCPWM0_TH1 由预装载寄存器加载到 MCPWM 运行所使用的影子寄存器
[25]	TMR3_UPDATE	写 1 产生软件（手动）触发，将 MCPWM0_TMR3 由预装载寄存器加载到 MCPWM 运行所使用的影子寄存器
[24]	TMR2_UPDATE	写 1 产生软件（手动）触发，将 MCPWM0_TMR2 由预装载寄存器加载到 MCPWM 运行所使用的影子寄存器
[23:22]		未使用
[21]	TH51_UPDATE	写 1 产生软件（手动）触发，将 MCPWM0_TH51 由预装载寄存器加载到 MCPWM 运行所使用的影子寄存器
[20]	TH50_UPDATE	写 1 产生软件（手动）触发，将 MCPWM0_TH50 由预装载寄存器加载到 MCPWM 运行所使用的影子寄存器
[19]	TH41_UPDATE	写 1 产生软件（手动）触发，将 MCPWM0_TH41 由预装载寄存器加载到 MCPWM 运行所使用的影子寄存器
[18]	TH40_UPDATE	写 1 产生软件（手动）触发，将 MCPWM0_TH40 由预装载寄存器加载到 MCPWM 运行所使用的影子寄存器
[17]	TH31_UPDATE	写 1 产生软件（手动）触发，将 MCPWM0_TH31 由预装载寄存器加载到 MCPWM 运行所使用的影子寄存器
[16]	TH30_UPDATE	写 1 产生软件（手动）触发，将 MCPWM0_TH30 由预装载寄存器加载到 MCPWM 运行所使用的影子寄存器
[15:12]		未使用
[11]	CNT0_UPDATE	写 1 产生软件（手动）触发，将 MCPWM0_CNT0 由预装载寄存器加载到 MCPWM 运行所使用的影子寄存器
[10]	TH0_UPDATE	写 1 产生软件（手动）触发，将 MCPWM0_TH0 由预装载寄存器加载到 MCPWM 运行所使用的影子寄存器
[9]	TMR1_UPDATE	写 1 产生软件（手动）触发，将 MCPWM0_TMR1 由预装载寄存器加载到 MCPWM 运行所使用的影子寄存器
[8]	TMR0_UPDATE	写 1 产生软件（手动）触发，将 MCPWM0_TMR0 由预装载寄存器加载到 MCPWM 运行所使用的影子寄存器
[7:6]		未使用
[5]	TH21_UPDATE	写 1 产生软件（手动）触发，将 MCPWM0_TH21 由预装载寄存器加载到 MCPWM 运行所使用的影子寄存器
[4]	TH20_UPDATE	写 1 产生软件（手动）触发，将 MCPWM0_TH20 由预装载寄存器加载到 MCPWM 运行所使用的影子寄存器

[3]	TH11_UPDATE	写 1 产生软件（手动）触发，将 MCPWM0_TH11 由预装载寄存器加载到 MCPWM 运行所使用的影子寄存器
[2]	TH10_UPDATE	写 1 产生软件（手动）触发，将 MCPWM0_TH10 由预装载寄存器加载到 MCPWM 运行所使用的影子寄存器
[1]	TH01_UPDATE	写 1 产生软件（手动）触发，将 MCPWM0_TH01 由预装载寄存器加载到 MCPWM 运行所使用的影子寄存器
[0]	TH00_UPDATE	写 1 产生软件（手动）触发，将 MCPWM0_TH00 由预装载寄存器加载到 MCPWM 运行所使用的影子寄存器

向 MCPWM0_UPDATE 对应位写 1 可以触发寄存器值从预装载寄存器写入影子寄存器，MCPWM0_UPDATE 写入后自动清零，无需软件清零。每写 1 一次，进行一次软件/手动触发。对 MCPWM0_UPDATE[15:14]置位不会对 MCPWM0_CNT0/1 造成影响。

向 MCPWM0_UPDATE 写 0 不作用，不推荐写入 0。

14.2.23 MCPWM0_FCNT

无写保护的寄存器

地址:0x4001_0C54

复位值:0x0

表 14-26 MCPWM0_FCNT 寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	FCNT	若 MCPWM0_FAIL012[15]=1，当发生 fail0/1 事件时，记录 MCPWM0_CNT0 值，存入 MCPWM0_FCNT；若 MCPWM0_FAIL345[15]=1，当发生 fail2/3 事件时，记录 MCPWM0_CNT1 值，存入 MCPWM0_FCNT

14.2.24 MCPWM0_EVT0

无写保护的寄存器

地址:0x4001_0C58

复位值:0x0

表 14-27 MCPWM0_EVT0 MCPWM 时基 0 外部触发寄存器

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

TIM3_CMP1	TIM3_CMP0	TIM2_CMP1	TIM2_CMP0	TIM1_CMP1	TIM1_CMP0	TIM0_CMP1	TIM0_CMP0		PWM0_TMR3	PWM0_TMR2	PWM0_TMR1	PWM0_TMR0
RW	RW	RW	RW	RW	RW	RW	RW		RW	RW	RW	RW
0	0	0	0	0	0	0	0		0	0	0	0

位置	位名称	说明
[31:16]		未使用
[15]	TIM3_CMP1	TIMER3 CMP1 事件触发时基 0 开始计数
[14]	TIM3_CMP0	TIMER3 CMP0 事件触发时基 0 开始计数
[13]	TIM2_CMP1	TIMER2 CMP1 事件触发时基 0 开始计数
[12]	TIM2_CMP0	TIMER2 CMP0 事件触发时基 0 开始计数
[11]	TIM1_CMP1	TIMER1 CMP1 事件触发时基 0 开始计数
[10]	TIM1_CMP0	TIMER1 CMP0 事件触发时基 0 开始计数
[9]	TIM0_CMP1	TIMER0 CMP1 事件触发时基 0 开始计数
[8]	TIM0_CMP0	TIMER0 CMP0 事件触发时基 0 开始计数
[7:4]		未使用
[3]	PWM0_TMR3	MCPWM0 TMR3 事件触发时基 0 开始计数
[2]	PWM0_TMR2	MCPWM0 TMR2 事件触发时基 0 开始计数
[1]	PWM0_TMR1	MCPWM0 TMR1 事件触发时基 0 开始计数
[0]	PWM0_TMR0	MCPWM0 TMR0 事件触发时基 0 开始计数

14.2.25 MCPWM0_EVT1

无写保护的寄存器

地址:0x4001_0C5C

复位值:0x0

表 14-28 MCPWM0_EVT1 MCPWM 时基 1 外部触发寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM3_CMP1	TIM3_CMP0	TIM2_CMP1	TIM2_CMP0	TIM1_CMP1	TIM1_CMP0	TIM0_CMP1	TIM0_CMP0					PWM0_TMR3	PWM0_TMR2	PWM0_TMR1	PWM0_TMR0
RW	RW	RW	RW	RW	RW	RW	RW					RW	RW	RW	RW
0	0	0	0	0	0	0	0					0	0	0	0

位置	位名称	说明
[31:12]		未使用
[15]	TIM3_CMP1	TIMER3CMP1 事件触发时基 1 开始计数
[14]	TIM3_CMP0	TIMER3 CMP0 事件触发时基 1 开始计数
[13]	TIM2_CMP1	TIMER2 CMP1 事件触发时基 1 开始计数

[12]	TIM2_CMP0	TIMER2 CMP0 事件触发时基 1 开始计数
[11]	TIM1_CMP1	TIMER1 CMP1 事件触发时基 1 开始计数
[10]	TIM1_CMP0	TIMER1 CMP0 事件触发时基 1 开始计数
[9]	TIM0_CMP1	TIMER0 CMP1 事件触发时基 1 开始计数
[8]	TIM0_CMP0	TIMER0 CMP0 事件触发时基 1 开始计数
[7:4]		未使用
[3]	PWM0_TMR3	MCPWM0 TMR3 事件触发时基 1 开始计数
[2]	PWM0_TMR2	MCPWM0 TMR2 事件触发时基 1 开始计数
[1]	PWM0_TMR1	MCPWM0 TMR1 事件触发时基 1 开始计数
[0]	PWM0_TMR0	MCPWM0 TMR0 事件触发时基 1 开始计数

14.2.26 MCPWM0_DTH00

写保护的寄存器

地址:0x4001_0C60

复位值:0x0

表 14-29 MCPWM0_DTH00 配置寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						DTH00									
						RW									
						0									

位置	位名称	说明
[31:10]		未使用
[9:0]	DTH00	MCPWM CH0/1/2 N 通道死区宽度控制寄存器，10bit 无符号数

14.2.27 MCPWM0_DTH01

写保护的寄存器

地址:0x4001_0C64

复位值:0x0

表 14-30 MCPWM0_DTH01 配置寄存器

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

位置	位名称	说明
[31:10]		未使用
[9:0]	DTH01	MCPWM CH0/1/2 P 通道死区宽度控制寄存器，10bit 无符号数

14.2.28 MCPWM0_DTH10

写保护的寄存器

地址:0x4001_0C68

复位值:0x0

表 14-31 MCPWM0_DTH10 配置寄存器

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

位置	位名称	说明
[31:10]		未使用
[9:0]	DTH10	MCPWM CH3/4/5 N 通道死区宽度控制寄存器，10bit 无符号数

14.2.29 MCPWM0_DTH11

写保护的寄存器

地址:0x4001_0C6C

复位值:0x0

表 14-32 MCPWM0_DTH11 配置寄存器

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

位置	位名称	说明
[31:10]		未使用
[9:0]	DTH11	MCPWM CH3/4/5 P 通道死区宽度控制寄存器，10bit 无符号数

14.2.30 MCPWM0_FLT

写保护的寄存器

地址:0x4001_0C70

复位值:0x0

表 14-33 MCPWM0_FLT MCPWM 滤波时钟分频寄存器

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

	0

位置	位名称	说明
[31:8]		未使用
[7:0]	FLT_CLKDIV	FAIL 信号输入的滤波时钟分频寄存器，基于系统时钟分频，影响 MCPWM0_FAIL[1:0]。计算公式如下: 系统时钟 / (FLT_CLKDIV + 1)。分频范围是 1-256。

MCPWM 使用分频后的时钟对 FAIL 信号固定进行 16 周期的滤波。当使用 256 分频时，滤波宽度为 4096 个 TCLK 时钟宽度。

14.2.31 MCPWM0_SDCFG

写保护的寄存器

地址:0x4001_0C74

复位值:0x0

表 14-34 MCPWM0_SDCFG 配置寄存器

15	14	13	12	11 10 9 8	7	6	5	4	3 2 1 0
	TR1_AEC	TR1_T1_UEN	TR1_T0_UEN	TR1_UP_INTV		TR0_AEC	TR0_T1_UEN	TR0_T0_UEN	TR0_UP_INTV
	RW	RW	RW	RW		RW	RW	RW	RW
	0	0	0	0		0	0	0	0

位置	位名称	说明
[31:15]		未使用
[14]	TR1_AEC	更新事件是否自动清除 MCPWM0_EIF[7:6]并置位 MCPWM0_FAIL345.MOE，恢复 MCPWM 通道 3/4/5 信号输出。 1:使能自动故障清除功能；0:关闭自动故障清除功能。
[13]	TR1_T1_UEN	时基 1 t1（过零）事件更新使能。1:使能；0，关闭。
[12]	TR1_T0_UEN	时基 1 t0（起点）事件更新使能。1:使能；0，关闭。
[11:8]	TR1_UP_INTV	时基 1 更新间隔。一旦 t0 和 t1 事件发生次数同 TR1_UP_INTV 相等，MCPWM 系统自动触发 MCPWM0_TH1，TH30~TH51 和 MCPWM0_TMR 寄存器加载到 MCPWM 运行系统的操作。若 TR1_T1_UEN 和 TR1_T0_UEN 均关闭，将不会触发此类型加载，只能手动触发加载。
[7]		未使用
[6]	TR0_AEC	更新事件是否自动清除 MCPWM0_EIF[5:4]并置位 MCPWM0_FAIL012.MOE，恢复 MCPWM 通道 0/1/2 信号输出。 1:使能自动故障清除功能；0:关闭自动故障清除功能。
[5]	TR0_T1_UEN	时基 0 t1（过零）事件更新使能。1:使能；0，关闭。
[4]	TR0_T0_UEN	时基 0 t0（起点）事件更新使能。1:使能；0，关闭。

[3:0]	TR0_UP_INTV	时基 0 更新间隔。一旦 t0 和 t1 事件发生次数同 TR0_UP_INTV+1 相等，MCPWM 系统自动触发 MCPWM0_TH0，MCPWM0_TH00~TH21 和 MCPWM0_TMR 寄存器加载到 MCPWM 运行系统的操作。若 TR0_T1_UEN 和 TR0_T0_UEN 均关闭，将不会触发此类型加载，只能手动触发加载。

14.2.32 MCPWM0_AUEN

写保护的寄存器

地址:0x4001_0C78

复位值:0x0

表 14-35 MCPWM0_AUEN MCPWM 自动更新使能寄存器

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
				CNT1_AUPDATE	TH1_AUPDATE	TMR3_AUPDATE	TMR2_AUPDATE			TH51_AUPDATE	TH50_AUPDATE	TH41_AUPDATE	TH40_AUPDATE	TH31_AUPDATE	TH30_AUPDATE
				RW	RW	RW	RW			RW	RW	RW	RW	RW	RW
				0	1	1	1			1	1	1	1	1	1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				CNT0_AUPDATE	TH0_AUPDATE	TMR1_AUPDATE	TMR0_AUPDATE			TH21_AUPDATE	TH20_AUPDATE	TH11_AUPDATE	TH10_AUPDATE	TH01_AUPDATE	TH00_AUPDATE
				RW	RW	RW	RW			RW	RW	RW	RW	RW	RW
				0	1	1	1			1	1	1	1	1	1

位置	位名称	说明
[31:28]		未使用
[27]	CNT1_AUPDATE	MCPWM0_CNT1 自动加载使能。1:加载；0:不加载。
[26]	TH1_AUPDATE	MCPWM0_TH1 自动加载使能。1:加载；0:不加载。
[25]	TMR3_AUPDATE	MCPWM0_TMR3 自动加载使能。1:加载；0:不加载。
[24]	TMR2_AUPDATE	MCPWM0_TMR2 自动加载使能。1:加载；0:不加载。
[23:22]		未使用
[21]	TH51_AUPDATE	MCPWM0_TH51 自动加载使能。1:加载；0:不加载。
[20]	TH50_AUPDATE	MCPWM0_TH50 自动加载使能。1:加载；0:不加载。
[19]	TH41_AUPDATE	MCPWM0_TH41 自动加载使能。1:加载；0:不加载。
[18]	TH40_AUPDATE	MCPWM0_TH40 自动加载使能。1:加载；0:不加载。
[17]	TH31_AUPDATE	MCPWM0_TH31 自动加载使能。1:加载；0:不加载。
[16]	TH30_AUPDATE	MCPWM0_TH30 自动加载使能。1:加载；0:不加载。
[15:12]		未使用
[11]	CNT0_AUPDATE	MCPWM0_CNT0 自动加载使能。1:加载；0:不加载。

[10]	TH0_AUPDATE	MCPWM0_TH0 自动加载使能。1:加载；0:不加载。
[9]	TMR1_AUPDATE	MCPWM0_TMR1 自动加载使能。1:加载；0:不加载。
[8]	TMR0_AUPDATE	MCPWM0_TMR0 自动加载使能。1:加载；0:不加载。
[7:6]		未使用
[5]	TH21_AUPDATE	MCPWM0_TH21 自动加载使能。1:加载；0:不加载。
[4]	TH20_AUPDATE	MCPWM0_TH20 自动加载使能。1:加载；0:不加载。
[3]	TH11_AUPDATE	MCPWM0_TH11 自动加载使能。1:加载；0:不加载。
[2]	TH10_AUPDATE	MCPWM0_TH10 自动加载使能。1:加载；0:不加载。
[1]	TH01_AUPDATE	MCPWM0_TH01 自动加载使能。1:加载；0:不加载。
[0]	TH00_AUPDATE	MCPWM0_TH00 自动加载使能。1:加载；0:不加载。

14.2.33 MCPWM0_TCLK

写保护的寄存器

地址:0x4001_077C

复位值:0x0

表 14-36 MCPWM0_TCLK 配置寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				CMP_CTRL_CNT1	CMP_CTRL_CNT0	EVT_CNT1_EN	EVT_CNT0_EN	BASE_CNT1_EN	BASE_CNT0_EN	TMR3_TB	TMR2_TB	ZCS_EN	CLK_EN	CLK_DIV	
				RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	
				0	0	0	0	0	0	0	0	0	0	0	

位置	位名称	说明
[31:12]		未使用
[11]	CMP_CTRL_CNT1	比较器控制 MCPWM0_CNT1 计数
[10]	CMP_CTRL_CNT0	比较器控制 MCPWM0_CNT0 计数
[9]	EVT_CNT1_EN	时基 1 外部触发使能
[8]	EVT_CNT0_EN	时基 0 外部触发使能
[7]	BASE_CNT1_EN	MCPWM 时基 1 计数器使能开关。1:使能；0:关闭。
[6]	BASE_CNT0_EN	MCPWM 时基 0 计数器使能开关。1:使能；0:关闭。
[5]	TMR3_TB	TMR3 时基选择，0:时基 0，1:时基 1
[4]	TMR2_TB	TMR2 时基选择，0:时基 0，1:时基 1
[3]	ZCS_EN	使能 ZCS 事件检测，高有效。当 ZCS_EN=1，发生 ZCS 事件时，将 P/N 通道同时关断，并延时 MCPWM0_ZCS_DELAY 后切换状态；当 ZCS_EN=0 时，发生 ZCS 事件不延时也不进行状态切换。
[2]	CLK_EN	MCPWM 工作时钟使能。1:使能；0:关闭。
[1:0]	CLK_DIV	MCPWM 工作时钟分频寄存器。 0:系统时钟

		1:系统时钟/2 2:系统时钟/4 3:系统时钟/8

只有使能 MCPWM0_TCLK.CLK_EN，才能完成影子寄存器更新。配置完成影子寄存器之后，同时开启 MCPWM0_TCLK.BASE_CNT0/1_EN 可以使得时基 0 和时基 1 同步开始计数。如果 TH0 和 TH1 设置为相同值，则时基 0 和时基 1 完全同频。

当使用外部触发 MCPWM 开始计数时，需要配置 BASE_CNTx_EN 为 0，EVT_CNTx_EN 为 1，同时设置 MCPWM0_EVTx 选择合适的外部触发源。待触发事件发生后，BASE_CNTx_EN 会由硬件置 1，MCPWM 对应计数器开始计数。

14.2.34 MCPWM0_IE0

写保护的寄存器

地址:0x4001_0C80

复位值:0x0

表 14-37 MCPWM0_IE0 MCPWM 时基 0 中断控制寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	UP_IE	TMR3_IE	TMR2IE	TMR1_IE	TMR0_IE			TH21_IE	TH20_IE	TH11_IE	TH10_IE	TH01_IE	TH00_IE	T1_IE	T0_IE
	RW	RW	RW	RW	RW			RW	RW	RW	RW	RW	RW	RW	RW
	0	0	0	0	0			0	0	0	0	0	0	0	0

位置	位名称	说明
[31:15]		未使用
[14]	UP_IE	时基 0 更新事件中断源使能。1，使能；0，关闭。
[13]	TMR3_IE	MCPWM0_CNT0 等于 MCPWM0_TMR3 中断源使能。1，使能；0，关闭。
[12]	TMR2_IE	MCPWM0_CNT0 等于 MCPWM0_TMR2 中断源使能。1，使能；0，关闭。
[11]	TMR1_IE	MCPWM0_CNT0 等于 MCPWM0_TMR1 中断源使能。1，使能；0，关闭。
[10]	TMR0_IE	MCPWM0_CNT0 等于 MCPWM0_TMR0 中断源使能。1，使能；0，关闭。
[9:8]		未使用
[7]	TH21_IE	MCPWM0_CNT0 等于 MCPWM0_TH21 中断源使能。1，使能；0，关闭。
[6]	TH20_IE	MCPWM0_CNT0 等于 MCPWM0_TH20 中断源使能。1，使能；0，关闭。
[5]	TH11_IE	MCPWM0_CNT0 等于 MCPWM0_TH11 中断源使能。1，使能；0，关闭。
[4]	TH10_IE	MCPWM0_CNT0 等于 MCPWM0_TH10 中断源使能。1，使能；0，关闭。
[3]	TH01_IE	MCPWM0_CNT0 等于 MCPWM0_TH01 中断源使能。1，使能；0，关闭。
[2]	TH00_IE	MCPWM0_CNT0 等于 MCPWM0_TH00 中断源使能。1，使能；0，关闭。
[1]	T1_IE	时基 0 T1 事件，MCPWM0_CNT0 的计数值回 0 中断源使能。 1，使能；0，关闭。
[0]	T0_IE	时基 0 T0 事件，MCPWM0_CNT0 的计数值等于 MCPWM0_TH0 中断源使能。 1，使能；0，关闭。

14.2.35 MCPWM0_IF0

无写保护的寄存器

地址:0x4001_0C84

复位值:0x0

表 14-38 MCPWM0_IF0 MCPWM 时基 0 中断标志寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	UP_IF	TMR3_IF	TMR2_IF	TMR1_IF	TMR0_IF			TH21_IF	TH20_IF	TH11_IF	TH10_IF	TH01_IF	TH00_IF	T1_IF	T0_IF
	RW1C	RW1C	RW1C	RW1C	RW1C			RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C
	0	0	0	0	0			0	0	0	0	0	0	0	0

位置	位名称	说明
[31:15]		未使用
[14]	UP_IF	时基 0 更新事件 MCPWM0_TH0/MCPWM0_TH00~MCPWM0_TH21/MCPWM0_TMR 等寄存器更新到 MCPWM 实际运行系统的中断事件。 1，发生；0，没发生。写 1 清零。
[13]	TMR3_IF	MCPWM0_CNT0 等于 MCPWM0_TMR3 中断事件。 1，发生；0，没发生。写 1 清零。
[12]	TMR2_IF	MCPWM0_CNT0 等于 MCPWM0_TMR2 中断事件。 1，发生；0，没发生。写 1 清零。
[11]	TMR1_IF	MCPWM0_CNT0 等于 MCPWM0_TMR1 中断事件。 1，发生；0，没发生。写 1 清零。
[10]	TMR0_IF	MCPWM0_CNT0 等于 MCPWM0_TMR0 中断事件。 1，发生；0，没发生。写 1 清零。
[9:8]		未使用
[7]	TH21_IF	MCPWM0_CNT0 等于 MCPWM0_TH21 中断事件。 1，发生；0，没发生。写 1 清零。
[6]	TH20_IF	MCPWM0_CNT0 等于 MCPWM0_TH20 中断事件。 1，发生；0，没发生。写 1 清零。
[5]	TH11_IF	MCPWM0_CNT0 等于 MCPWM0_TH11 中断事件。 1，发生；0，没发生。写 1 清零。
[4]	TH10_IF	MCPWM0_CNT0 等于 MCPWM0_TH10 中断事件。 1，发生；0，没发生。写 1 清零。
[3]	TH01_IF	MCPWM0_CNT0 等于 MCPWM0_TH01 中断事件。 1，发生；0，没发生。写 1 清零。
[2]	TH00_IF	MCPWM0_CNT0 等于 MCPWM0_TH00 中断事件。 1，发生；0，没发生。写 1 清零。
[1]	T1_IF	t1 事件，MCPWM0_CNT0 等于 0 中断事件。

		1，发生；0，没发生。写 1 清零。
[0]	T0_IF	t0 事件，MCPWM0_CNT0 等于 MCPWM0_TH 中断事件。 1，发生；0，没发生。写 1 清零。

14.2.36 MCPWM0_IE1

写保护的寄存器

地址:0x4001_0C88

复位值:0x0

表 14-39 MCPWM0_IE1 MCPWM 时基 1 中断控制寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	UP_IE	TMR3_IE	TMR2_IE					TH51_IE	TH50_IE	TH41_IE	TH40_IE	TH31_IE	TH30_IE	T1_IE	T0_IE
	RW	RW	RW					RW	RW	RW	RW	RW	RW	RW	RW
	0	0	0					0	0	0	0	0	0	0	0

位置	位名称	说明
[31:15]		未使用
[14]	UP_IE	时基 1 更新事件 MCPWM0_TH1/MCPWM0_TH30~MCPWM0_TH51/MCPWM0_TMR2~MCPWM0_TMR3 等寄存器更新到 MCPWM 实际运行系统的中断使能。 1，使能；0，关闭。
[13]	TMR3_IE	MCPWM0_CNT1 等于 MCPWM0_TMR3 中断使能。1，使能；0，关闭。
[12]	TMR2_IE	MCPWM0_CNT1 等于 MCPWM0_TMR2 中断使能。1，使能；0，关闭。
[11:8]		未使用
[7]	TH51_IE	MCPWM0_CNT1 等于 MCPWM0_TH51 中断使能。1，使能；0，关闭。
[6]	TH50_IE	MCPWM0_CNT1 等于 MCPWM0_TH50 中断使能。1，使能；0，关闭。
[5]	TH41_IE	MCPWM0_CNT1 等于 MCPWM0_TH41 中断使能。1，使能；0，关闭。
[4]	TH40_IE	MCPWM0_CNT1 等于 MCPWM0_TH40 中断使能。1，使能；0，关闭。
[3]	TH31_IE	MCPWM0_CNT1 等于 MCPWM0_TH31 中断使能。1，使能；0，关闭。
[2]	TH30_IE	MCPWM0_CNT1 等于 MCPWM0_TH30 中断使能。1，使能；0，关闭。
[1]	T1_IE	T1 事件，MCPWM0_CNT1 等于 0 中断使能。 1，使能；0，关闭。
[0]	T0_IE	T0 事件，MCPWM0_CNT1 等于 MCPWM0_TH1 中断使能。 1，使能；0，关闭。

14.2.37 MCPWM0_IF1

无写保护的寄存器

地址:0x4001_0C8C

复位值:0x0

表 14-40 MCPWM0_IF1 MCPWM 时基 1 中断标志寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	UP_IF	TMR3_IF	TMR2_IF					TH51_IF	TH50_IF	TH41_IF	TH40_IF	TH31_IF	TH30_IF	T1_IF	T0_IF
	RW1C	RW1C	RW1C					RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C
	0	0	0					0	0	0	0	0	0	0	0

位置	位名称	说明
[31:15]		未使用
[14]	UP_IF	时基 1 更新事件 MCPWM0_TH1/MCPWM0_TH30~MCPWM0_TH51/MCPWM0_TMR 等寄存器更新到 MCPWM 实际运行系统的中断事件。 1，发生；0，没发生。写 1 清零。
[13]	TMR3_IF	MCPWM0_CNT1 等于 MCPWM0_TMR3 中断事件。 1，发生；0，没发生。写 1 清零。
[12]	TMR2_IF	MCPWM0_CNT1 等于 MCPWM0_TMR2 中断事件。 1，发生；0，没发生。写 1 清零。
[11:8]		未使用
[7]	TH51_IF	MCPWM0_CNT1 等于 MCPWM0_TH51 中断事件。 1，发生；0，没发生。写 1 清零。
[6]	TH50_IF	MCPWM0_CNT1 等于 MCPWM0_TH50 中断事件。 1，发生；0，没发生。写 1 清零。
[5]	TH41_IF	MCPWM0_CNT1 等于 MCPWM0_TH41 中断事件。 1，发生；0，没发生。写 1 清零。
[4]	TH40_IF	MCPWM0_CNT1 等于 MCPWM0_TH40 中断事件。 1，发生；0，没发生。写 1 清零。
[3]	TH31_IF	MCPWM0_CNT1 等于 MCPWM0_TH31 中断事件。 1，发生；0，没发生。写 1 清零。
[2]	TH30_IF	MCPWM0_CNT1 等于 MCPWM0_TH30 中断事件。 1，发生；0，没发生。写 1 清零。
[1]	T1_IF	T1 事件，MCPWM0_CNT1 等于 0 中断事件。 1，发生；0，没发生。写 1 清零。
[0]	T0_IF	T0 事件，MCPWM0_CNT1 等于 MCPWM0_TH 中断事件。 1，发生；0，没发生。写 1 清零。

14.2.38 MCPWM0_EIE

写保护的寄存器

地址:0x4001_0C90

复位值:0x0

表 14-41 MCPWM0_EIE 配置寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						ZCS1_IE	ZCS0_IE	FAIL3_IE	FAIL2_IE	FAIL1_IE	FAIL0_IE				
						RW	RW	RW	RW	RW	RW				
						0	0	0	0	0	0				

位置	位名称	说明
[31:10]		未使用
[9]	ZCS1_IE	时基 1 ZCS 事件中断源使能。1，使能；0，关闭。
[8]	ZCS0_IE	时基 0 ZCS 事件中断源使能。1，使能；0，关闭。
[7]	FAIL3_IE	FAIL3 中断源使能。1，使能；0，关闭。
[6]	FAIL2_IE	FAIL2 中断源使能。1，使能；0，关闭。
[5]	FAIL1_IE	FAIL1 中断源使能。1，使能；0，关闭。
[4]	FAIL0_IE	FAIL0 中断源使能。1，使能；0，关闭。
[3:0]		未使用

14.2.39 MCPWM0_EIF

无写保护的寄存器

地址:0x4001_0C94

复位值:0x0

表 14-42 MCPWM0_EIF 配置寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						ZCS1_IF	ZCS1_IF	FAIL3_IF	FAIL2_IF	FAIL1_IF	FAIL0_IF				
						RW1C	RW1C	RW1C	RW1C	RW1C	RW1C				
						0	0	0	0	0	0				

位置	位名称	说明
[31:10]		未使用
[9]	ZCS1_IF	时基 1 ZCS 中断源事件。1，发生；0，没发生。写 1 清零。
[8]	ZCS0_IF	时基 0 ZCS 中断源事件。1，发生；0，没发生。写 1 清零。
[7]	FAIL3_IF	FAIL3 中断源事件。1，发生；0，没发生。写 1 清零。
[6]	FAIL2_IF	FAIL2 中断源事件。1，发生；0，没发生。写 1 清零。
[5]	FAIL1_IF	FAIL1 中断源事件。1，发生；0，没发生。写 1 清零。
[4]	FAIL0_IF	FAIL0 中断源事件。1，发生；0，没发生。写 1 清零。

[3:0]		未使用

14.2.40 MCPWM0_RE

写保护的寄存器

地址:0x4001_0C98

复位值:0x0

表 14-43 MCPWM0_RE 配置寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR1_T1_RE1	TR1_T0_RE1	TR0_T1_RE1	TR0_T0_RE1	TMR3_RE1	TMR2_RE1	TMR1_RE1	TMR0_RE1	TR1_T1_RE0	TR1_T0_RE0	TR0_T1_RE0	TR0_T0_RE0	TMR3_RE0	TMR2_RE0	TMR1_RE0	TMR0_RE0
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

位置	位名称	说明
[31:16]		未使用
[15]	TR1_T1_RE1	时基 1 T1 事件 DMA 请求 1 使能。1:使能；0:关闭。
[14]	TR1_T0_RE1	时基 1 T0 事件 DMA 请求 1 使能。1:使能；0:关闭。
[13]	TR0_T1_RE1	时基 0 T1 事件 DMA 请求 1 使能。1:使能；0:关闭。
[12]	TR0_T0_RE1	时基 0 T0 事件 DMA 请求 1 使能。1:使能；0:关闭。
[11]	TMR3_RE1	MCPWM 计数器命中 TMR3，DMA 请求 1 使能。1:使能；0:关闭。
[10]	TMR2_RE1	MCPWM 计数器命中 TMR2，DMA 请求 1 使能。1:使能；0:关闭。
[9]	TMR1_RE1	MCPWM 计数器命中 TMR1，DMA 请求 1 使能。1:使能；0:关闭。
[8]	TMR0_RE1	MCPWM 计数器命中 TMR0，DMA 请求 1 使能。1:使能；0:关闭。
[7]	TR1_T1_RE0	时基 1 T1 事件 DMA 请求 0 使能。1:使能；0:关闭。
[6]	TR1_T0_RE0	时基 1 T0 事件 DMA 请求 0 使能。1:使能；0:关闭。
[5]	TR0_T1_RE0	时基 0 T1 事件 DMA 请求 0 使能。1:使能；0:关闭。
[4]	TR0_T0_RE0	时基 0 T0 事件 DMA 请求 0 使能。1:使能；0:关闭。
[3]	TMR3_RE0	MCPWM 计数器命中 TMR3，DMA 请求 0 使能。1:使能；0:关闭。
[2]	TMR2_RE0	MCPWM 计数器命中 TMR2，DMA 请求 0 使能。1:使能；0:关闭。
[1]	TMR1_RE0	MCPWM 计数器命中 TMR1，DMA 请求 0 使能。1:使能；0:关闭。
[0]	TMR0_RE0	MCPWM 计数器命中 TMR0，DMA 请求 0 使能。1:使能；0:关闭。

MCPWM 可以产生两个 DMA 请求信号。如果使能 MCPWM0_RE 寄存器中的 RE0，且对应的事件标志置位，会产生 DMA 请求信号 0；如果使能 MCPWM0_RE 寄存器中的 RE1，且对应的事件标志置位，会产生 DMA 请求信号 1。两个请求信号可以分别触发两个 DMA 通道进行搬运。

14.2.41 MCPWM0_PP

写保护的寄存器

地址:0x4001_0C9C

复位值:0x0

表 14-44 MCPWM0_PP 配置寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										IO5_PPE	IO4_PPE	IO3_PPE	IO2_PPE	IO1_PPE	IO0_PPE
										RW	RW	RW	RW	RW	RW
										0	0	0	0	0	0

位置	位名称	说明
[31:6]		未使用
[5]	IO5_PPE	IO5 推挽模式使能信号。写 1，使能；写 0，关闭。
[4]	IO4_PPE	IO4 推挽模式使能信号。写 1，使能；写 0，关闭。
[3]	IO3_PPE	IO3 推挽模式使能信号。写 1，使能；写 0，关闭。
[2]	IO2_PPE	IO2 推挽模式使能信号。写 1，使能；写 0，关闭。
[1]	IO1_PPE	IO1 推挽模式使能信号。写 1，使能；写 0，关闭。
[0]	IO0_PPE	IO0 推挽模式使能信号。写 1，使能；写 0，关闭。

推挽模式使能信号。根据工作模式不同而不同。边沿模式，开启边沿模式的推挽模式；中心对齐，开启中心对齐的推挽模式。

14.2.42 MCPWM0_IO01

写保护的寄存器

地址:0x4001_0CA0

复位值:0x0

表 14-45 MCPWM0_IO01 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15]	CH1_WM	CH1 工作模式选择。1:边沿模式；0:互补模式。
[14]	CH1_PN_SW	CH1 的 P 和 N 通道输出互换选择。即 P 通道信号最后从 N 通道输出，N 通道的信号最后从 P 通道输出。1:互换；0:不互换。
[13]	CH1_SCTRLP	当 CH1_PS=1 时，输出到 CH1 P 通道的值。
[12]	CH1_SCTRLN	当 CH1_NS=1 时，输出到 CH1 N 通道的值。

[11]	CH1_PS	CH1 P 来源。1:来自 CH1_SCTRLP；0:MCPWM 内部计数器产生。
[10]	CH1_NS	CH1 N 来源。1:来自 CH1_SCTRLN；0:MCPWM 内部计数器产生。
[9]	CH1_PP	CH1 P 极性选择。1:CH1 P 信号取反输出；0:CH1 P 信号正常输出。
[8]	CH1_NP	CH1 N 极性选择。1:CH1 N 信号取反输出；0:CH1 N 信号正常输出。
[7]	CH0_WM	CH0 工作模式选择。1:边沿模式；0:互补模式。
[6]	CH0_PN_SW	CH0 的 P 和 N 通道输出互换选择。即 P 通道信号最后从 N 通道输出，N 通道的信号最后从 P 通道输出。1:互换；0:不互换。
[5]	CH0_SCTRLP	当 CH0_PS =1 时，输出到 CH0 P 通道的值。
[4]	CH0_SCTRLN	当 CH0_NS =1 时，输出到 CH0 N 通道的值。
[3]	CH0_PS	CH0 P 来源。1:来自 CH0_SCTRLP；0:MCPWM 实际运行系统中计数器产生。
[2]	CH0_NS	CH0 N 来源。1:来自 CH0_SCTRLN；0:MCPWM 实际运行系统中计数器产生。
[1]	CH0_PP	CH0 P 极性选择。1:CH0 P 信号取反输出；0:CH0 P 信号正常输出。
[0]	CH0_NP	CH0 N 极性选择。1:CH0 N 信号取反输出；0:CH0 N 信号正常输出。**极性选择跟随通道交换，例如 CH0 N 选择取反输出，同时选择了通道交换，则交换后的 CH0 N 仍是取反输出。**

14.2.43 MCPWM0_IO23

写保护的寄存器

地址:0x4001_0CA4

复位值:0x0

表 14-46 MCPWM0_IO23 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15]	CH3_WM	CH3 工作模式选择。1:Edge 模式；0:互补模式。
[14]	CH3_PN_SW	CH3 的 P 和 N 通道输出互换选择。即 P 通道信号最后从 N 通道输出，N 通道的信号最后从 P 通道输出。1:互换；0:不互换。
[13]	CH3_SCTRLP	当 CH3_PS =1 时，输出到 CH3 P 通道的值。
[12]	CH3_SCTRLN	当 CH3_NS =1 时，输出到 CH3 N 通道的值。
[11]	CH3_PS	CH3 P 来源。1:来自 CH3_SCTRLP；0:MCPWM 实际运行系统中计数器产生。
[10]	CH3_NS	CH3 N 来源。1:来自 CH3_SCTRLN；0:MCPWM 实际运行系统中计数

		器产生。
[9]	CH3_PP	CH3 P 极性选择。1:CH3 P 信号取反输出；0:CH3 P 信号正常输出。
[8]	CH3_NP	CH3 N 极性选择。1:CH3 N 信号取反输出；0:CH3 N 信号正常输出。
[7]	CH2_WM	CH2 工作模式选择。1:Edge 模式；0:互补模式。
[6]	CH2_PN_SW	CH2 的 P 和 N 通道输出互换选择。即 P 通道信号最后从 N 通道输出，N 通道的信号最后从 P 通道输出。1:互换；0:不互换。
[5]	CH2_SCTRLP	当 CH2_PS =1 时，输出到 CH2 P 通道的值。
[4]	CH2_SCTRLN	当 CH2_NS =1 时，输出到 CH2 N 通道的值。
[3]	CH2_PS	CH2 P 来源。1:来自 CH2_SCTRLP；0:MCPWM 实际运行系统中计数器产生。
[2]	CH2_NS	CH2 N 来源。1:来自 CH2_SCTRLN；0:MCPWM 实际运行系统中计数器产生。
[1]	CH20_PP	CH2 P 极性选择。1:CH2 P 信号取反输出；0:CH2 P 信号正常输出。
[0]	CH2_NP	CH2 N 极性选择。1:CH2 N 信号取反输出；0:CH2 N 信号正常输出。**极性选择跟随通道交换，例如 CH2 N 选择取反输出，同时选择了通道交换，则交换后的 CH2 N 仍是取反输出。**

14.2.44 MCPWM0_IO45

写保护的寄存器

地址:0x4001_0CA8

复位值:0x0

表 14-47 MCPWM0_IO45 配置寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH5_WM	CH5_PN_SW	CH5_SCTRLP	CH5_SCTRLN	CH5_PS	CH5_NS	CH5_PP	CH5_NP	CH4_WM	CH4_PN_SW	CH4_SCTRLP	CH4_SCTRLN	CH4_PS	CH4_NS	CH4_PP	CH4_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

位置	位名称	说明
[31:16]		未使用
[15]	CH5_WM	CH5 工作模式选择。1:Edge 模式；0:互补模式。
[14]	CH5_PN_SW	CH5 的 P 和 N 通道输出互换选择。即 P 通道信号最后从 N 通道输出，N 通道的信号最后从 P 通道输出。1:互换；0:不互换。
[13]	CH5_SCTRLP	当 CH5_PS =1 时，输出到 CH5 P 通道的值。
[12]	CH5_SCTRLN	当 CH5_NS =1 时，输出到 CH5 N 通道的值。
[11]	CH5_PS	CH5 P 来源。1:来自 CH5_SCTRLP；0:MCPWM 实际运行系统中计数器产生。
[10]	CH5_NS	CH5 N 来源。1:来自 CH5_SCTRLN；0:MCPWM 实际运行系统中计数器产生。
[9]	CH5_PP	CH5 P 极性选择。1:CH5 P 信号取反输出；0:CH5 P 信号正常输出。

[8]	CH5_NP	CH5 N 极性选择。1:CH5 N 信号取反输出；0:CH5 N 信号正常输出。
[7]	CH4_WM	CH4 工作模式选择。1:Edge 模式；0:互补模式。
[6]	CH4_PN_SW	CH4 的 P 和 N 通道输出互换选择。即 P 通道信号最后从 N 通道输出，N 通道的信号最后从 P 通道输出。1:互换；0:不互换。
[5]	CH4_SCTRLP	当 CH4_PS =1 时，输出到 CH4 P 通道的值。
[4]	CH4_SCTRLN	当 CH4_NS =1 时，输出到 CH4 N 通道的值。
[3]	CH4_PS	CH4 P 来源。1:来自 CH4_SCTRLP；0:MCPWM 实际运行系统中计数器产生。
[2]	CH4_NS	CH4 N 来源。1:来自 CH4_SCTRLN；0:MCPWM 实际运行系统中计数器产生。
[1]	CH40_PP	CH4 P 极性选择。1:CH4 P 信号取反输出；0:CH4 P 信号正常输出。
[0]	CH4_NP	CH4 N 极性选择。1:CH4 N 信号取反输出；0:CH4 N 信号正常输出。**极性选择跟随通道交换，例如 CH4 N 选择取反输出，同时选择了通道交换，则交换后的 CH4 N 仍是取反输出。**

14.2.45 MCPWM0_FAIL012

写保护的寄存器

地址:0x4001_0CB0

复位值:0x0

表 14-48 MCPWM0_FAIL012 配置寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FAIL_0CAP				FAIL1_SEL		FAIL0_SEL		HALT_PRT	MOE	FAIL1_EN	FAIL0_EN	FAIL1_POL	FAIL0_POL		
RW				RW		RW		RW	RW	RW	RW	RW	RW		
0				0		0		0	0	0	0	0	0		

位置	位名称	说明
[31:16]		未使用
[15]	FAIL_0CAP	当发生 fail0/1 事件时，将 MCPWM0_CNT0 值存入 MCPWM0_FCNT
[14:12]		未使用
[11:10]	FAIL1_SEL	FAIL1 来源选择 0: IO MCPWM0_BKIN1 1: 比较器 1 原始输出 2: CLUOUT2 3: CLUOUT3
[9:8]	FAIL0_SEL	FAIL0 来源选择 0: IO MCPWM0_BKIN0 1: 比较器 0 原始输出 2: CLUOUT0

		3: CLUOUT1
[7]	HALT_PRT	MCU 进入 HALT 状态，MCPWM 输出值选择。 1:正常输出；0:强制 MCPWM 输出保护值。
[6]	MOE	MOE 控制 MCPWM CH0/CH1/CH2 通道 P 和 N 输出值。 1:输出 MCPWM 产生的正常信号 0:输出 MCPWM0_CH_DEF 设置的 CHxN_DEFAULT 和 CHxP_DEFAULT 默认值，此默认值不受极性/通道选择等控制。 MCPWM0_EIF.FAIL1_IF 和 MCPWM0_EIF.FAIL0_IF 任意一位变 1 将触发 MOE 变成 0，输出默认值。
[5]	FAIL1_EN	FAIL1 输入使能。1:使能；0:关闭。
[4]	FAIL0_EN	FAIL0 输入使能。1:使能；0:关闭。
[3]	FAIL1_POL	FAIL1 极性选择。1:信号极性取反输入，信号输入低为有效电平；0:信号极性正常输入，信号输入高为有效电平。
[2]	FAIL0_POL	FAIL0 极性选择。1:信号极性取反输入，信号输入低为有效电平；0:信号极性正常输入，信号输入高为有效电平。
[1:0]		未使用

FAIL0/1 信号用于工作于时基 0 的 MCPWM 通道 0/1/2。当 FAIL0/1 有效时，将暂停 MCPWM 时基 0 对应的计数器 MCPWM0_CNT0，并产生对应的错误中断标志 MCPWM0_EIF。

14.2.46 MCPWM0_FAIL345

写保护的寄存器

地址:0x4001_0CB4

复位值:0x0

表 14-49 MCPWM0_FAIL345 配置寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FAIL_1CAP				FAIL3_SEL		FAIL2_SEL		HALT_PRT	MOE	FAIL1_EN	FAIL0_EN	FAIL3_POL	FAIL2_POL		
RW				RW		RW		RW	RW	RW	RW	RW	RW		
0				0		0		0	0	0	0	0	0		

位置	位名称	说明
[31:16]		未使用
[15]	FAIL_1CAP	当发生 fail2/3 事件时，将 MCPWM0_CNT1 值存入 MCPWM0_FCNT
[14:12]		未使用
[11:10]	FAIL3_SEL	FAIL3 来源选择 0: IO MCPWM0_BKIN1 1:比较器 1 的结果 2: CLUOUT2

		3: CLUOUT3
[9:8]	FAIL2_SEL	FAIL2 来源选择 0: IO MCPWM0_BKIN0 1:比较器 0 的结果 2: CLUOUT0 3: CLUOUT1
[7]	HALT_PRT	MCU 进入 HALT 状态，MCPWM 输出值选择。 1:正常输出；0:强制 MCPWM 输出保护值。
[6]	MOE	MOE 控制 MCPWM CH3/CH4/CH5 通道 P 和 N 输出值。 1:输出 MCPWM 产生的正常信号 0:输出 CH3N_DEFAULT 和 CH3P_DEFAULT 默认值，此默认值不受极性/通道选择等控制。 MCPWM0_EIF.FAIL2_IF 和 MCPWM0_EIF.FAIL3_IF 任意一位变 1 将触发 MOE 变成 0，输出默认值。
[5]	FAIL3_EN	FAIL3 输入使能。1:使能；0:关闭。
[4]	FAIL2_EN	FAIL2 输入使能。1:使能；0:关闭。
[3]	FAIL3_POL	FAIL3 极性选择。1:信号极性取反输入，信号输入低为有效电平；0:信号极性正常输入，信号输入高为有效电平。
[2]	FAIL2_POL	FAIL2 极性选择。1:信号极性取反输入，信号输入低为有效电平；0:信号极性正常输入，信号输入高为有效电平。
[1:0]		未使用

FAIL2/3 信号用于工作于时基 1 的 MCPWM 通道 3/4/5。当 FAIL2/3 有效时，将暂停 MCPWM 时基 1 对应的计数器 MCPWM0_CNT1，并产生对应的错误中断标志 MCPWM0_EIF。

MCPWM0_FAIL 可以用来设置紧急停车事件，封锁 MCPWM 的信号输出。通道 0/1/2 的急停事件有两个 FAIL0 和 FAIL1，通道 3/4/5 的急停事件有两个 FAIL2 和 FAIL3。 每个 FAIL 信号共有 3 个信号来源可选，比较器输出，MCPWM0_BKIN 和 CLUOUT。

FAIL 的输入信号可以使用数字滤波，滤波时钟的第一级分频由 MCPWM0_TCLK.CLK_DIV 寄存器设置，第二级分频由 MCPWM0_FLT.FLT_CLKDIV[7:0]寄存器设置。

最后滤波电路会对 FAIL 信号进行 16 个滤波时钟的滤波，即只有信号稳定时间超过 16 个滤波周期才能通过滤波器。**即滤波宽度=滤波时钟周期*16。**

更多信息可以参考 FAIL 信号处理。

14.2.47 MCPWM0_CH_DEF

写保护的寄存器

地址:0x4001_0CB8

复位值:0x0

表 14-50 MCPWM0_CH_DEF 短路保护通道输出值寄存器

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

	CH5P_DEFAULT	CH5N_DEFAULT	CH4P_DEFAULT	CH4N_DEFAULT	CH3P_DEFAULT	CH3N_DEFAULT	CH2P_DEFAULT	CH2N_DEFAULT	CH1P_DEFAULT	CH1N_DEFAULT	CH0P_DEFAULT	CH0N_DEFAULT
	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
	0	0	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:12]		未使用
[11]	CH5P_DEFAULT	CH5_P 通道默认值
[10]	CH5N_DEFAULT	CH5_N 通道默认值
[9]	CH4P_DEFAULT	CH4_P 通道默认值
[8]	CH4N_DEFAULT	CH4_N 通道默认值
[7]	CH3P_DEFAULT	CH3 P 通道默认值
[6]	CH3N_DEFAULT	CH3 N 通道默认值。当发生 FAIL 事件或 MOE 为 0 时，相应通道输出默认电平。**默认电平输出不受 MCPWM0_IO23 的 BIT0、BIT1、BIT8、BIT9、BIT6、BIT14 通道交换和极性控制的影响，直接控制通道输出。**
[5]	CH2P_DEFAULT	CH2_P 通道默认值
[4]	CH2N_DEFAULT	CH2_N 通道默认值
[3]	CH1P_DEFAULT	CH1_P 通道默认值
[2]	CH1N_DEFAULT	CH1_N 通道默认值
[1]	CH0P_DEFAULT	CH0_P 通道默认值
[0]	CH0N_DEFAULT	CH0_N 通道默认值。当发生 FAIL 事件或 MOE 为 0 时，相应通道输出默认电平。**默认电平输出不受 MCPWM0_IO01 和 MCPWM0_IO23 的 BIT0、BIT1、BIT8、BIT9、BIT6、BIT14 通道交换和极性控制的影响，直接控制通道输出。**

14.2.48 MCPWM0_CH_MSK

写保护的寄存器

地址:0x4001_0CBC

复位值:0x0

表 14-51 MCPWM0_CH_MSK 通道屏蔽位寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				CH5P_FAIL_EN	CH5N_FAIL_EN	CH4P_FAIL_EN	CH4N_FAIL_EN	CH3P_FAIL_EN	CH3N_FAIL_EN	CH2P_FAIL_EN	CH2N_FAIL_EN	CH1P_FAIL_EN	CH1N_FAIL_EN	CH0P_FAIL_EN	CH0N_FAIL_EN
				RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
				1	1	1	1	1	1	1	1	1	1	1	1

位置	位名称	说明
[31:12]		未使用
[11]	CH5P_FAIL_EN	CH5_P 通道 FAIL 事件使能，高有效，默认开启
[10]	CH5N_FAIL_EN	CH5_N 通道 FAIL 事件使能，高有效，默认开启
[9]	CH4P_FAIL_EN	CH4_P 通道 FAIL 事件使能，高有效，默认开启
[8]	CH4N_FAIL_EN	CH4_N 通道 FAIL 事件使能，高有效，默认开启
[7]	CH3P_FAIL_EN	CH3_P 通道 FAIL 事件使能，高有效，默认开启
[6]	CH3N_FAIL_EN	CH3_N 通道 FAIL 事件使能，高有效，默认开启
[5]	CH2P_FAIL_EN	CH2_P 通道 FAIL 事件使能，高有效，默认开启
[4]	CH2N_FAIL_EN	CH2_N 通道 FAIL 事件使能，高有效，默认开启
[3]	CH1P_FAIL_EN	CH1_P 通道 FAIL 事件使能，高有效，默认开启
[2]	CH1N_FAIL_EN	CH1_N 通道 FAIL 事件使能，高有效，默认开启
[1]	CH0P_FAIL_EN	CH0_P 通道 FAIL 事件使能，高有效，默认开启
[0]	CH0N_FAIL_EN	CH0_N 通道 FAIL 事件使能，1：发生 FAIL 事件或 MCPWM0_FAIL012.MOE=0 时，CH0_N 通道电平输出为默认值，0：发生 FAIL 事件或 MCPWM0_FAIL012.MOE=0 时，CH0_N 通道电平不受影响，仍由 MCPWM 内部硬件控制。默认开启 FAIL 使能

14.2.49 MCPWM0_PRT

无写保护的寄存器

地址:0x4001_0CC0

复位值:0x0

表 14-52 MCPWM0_PRT 寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	PRT	写入 0xDEAD，解除 MCPWM 寄存器写保护；写入其它值，MCPWM 寄存器进入写保护。此寄存器读出恒为 0。

14.2.50 MCPWM0_SWAP

移动至 GPIO 模块中的 PWM_SWAP 寄存器。

14.2.51 MCPWM0_STT_HYST

写保护的寄存器

地址:0x4001_0CC8

复位值:0x0

表 14-53 MCPWM0_STT_HYST 状态停留延迟寄存器

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

位置	位名称	说明
[31:12]		未使用
[11:0]	STT_HYST	状态切换延时，仅在 MCPWM0_TCLK.CMP_CTRL_CNT0=1 或 MCPWM0_TCLK.CMP_CTRL_CNT1=1 时起作用。12bit 无符号数

14.2.52 MCPWM0_ZCS_DELAY

写保护的寄存器

地址:0x4001_0CCC

复位值:0x0

表 14-54 MCPWM0_ZCS_DELAY 状态停留时间寄存器

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

位置	位名称	说明
[31:15]		未使用
[14:0]	ZCS_DELAY	ZCS 状态停留时间，仅在 MCPWM0_TCLK.CMP_CTRL_CNT0=1 或 MCPWM0_TCLK.CMP_CTRL_CNT1=1 时起作用。15bit 无符号数。ZCS_DELAY 必须大于 STT_HYST

15 GPIO

15.1 概述

LSK32MC07x 系列芯片共集成了 4 组 GPIO，每组包含 16 个 GPIO。部分 GPIO 可以作为系统的休眠模式唤醒源。部分 GPIO 可以用作外部中断源输入。

15.1.1 功能框图

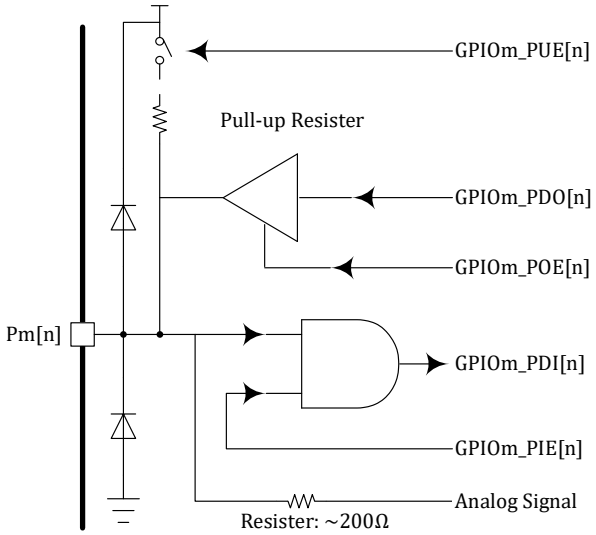


图 15-1 GPIO 功能框图

如上图所示，Pm[n]为芯片 PAD，m 可以是 0~3，表示 4 组 GPIO 中的任意一组，n 可以是 0~15，表示一组 16 个 GPIO 中的一个 IO。模拟信号通过一个约 200Ω 的电阻串联直接连接到 PAD。数字信号经过一个三态门输出，当输出使能 GPIOm_POE[n]=0 时，buffer 输出高阻态，否则 buffer 输出与 GPIOm_PDO[n]电平相同。数字信号输入通过一个与门进入芯片内部，当 GPIOm_PIE[n]=0 时，GPIOm_PDI[n]恒为 0，当 GPIOm_PIE[n]=1，即输入使能打开时，GPIOm_PDI[n]的电平与 Pm[n]电平相同。芯片 PAD 可以配置上拉，P0[2]引脚因为复用为外部复位脚 RSTN 上拉电阻为 300kΩ，注意，并非所有 PAD 都配备上拉电阻，具体哪些 PAD 具有上拉电阻资源，请参考 15.2.6 章节。没有上拉电阻的 PAD 也可以配置 GPIOm_PUE[n]寄存器，但无实际作用。

15.1.2 产品特点

- 64 路 GPIO
- 推挽开漏模式可配置
- 部分 GPIO 支持外部中断
- 部分 GPIO 可作为休眠唤醒源
- 部分 GPIO 支持输入信号滤波

具体哪些 GPIO 支持外部中断，请参考 15.2.15.4

具体哪些 GPIO 可以作为休眠唤醒源，请参考 21.6.5

具体哪些 GPIO 可以进行输入信号滤波，请参考 15.3.2

GPIO 可配置成最多 13 种外部功能。GPIO 功能分布，请参考对应器件手册(datasheet)。

表 15-1 GPIO 第二功能

GPIO 第二功能值	功能描述
0	模拟功能，当 GPIO 作为模拟功能使用时，需要关闭对应 IO 的输出使能，上拉使能等，方式 IO 输出信号或片内上拉电阻对模拟信号造成干扰
1	CMP_OUT/CLKO，模拟比较器直接输出/时钟输出
2	HALL，HALL 信号输入
3	MCPWM，MCPWM 通道输出或停机信号输入
4	UART，串口 RXD 或 TXD
5	SPI，SPI 时钟、片选，数据输入、数据输出
6	I2C，I2C 时钟、I2C 数据
7	TIMER0/1，TIMER0/1 的通道 0/1，作为输入用于捕获模式或外部时钟源，作为输出用于比较模式，TIMER0 的停机信号输入
8	TIMER2/3，TIMER2/3 的通道 0/1，作为输入用于捕获模式或外部时钟源，作为输出用于比较模式，QEP0/1 的 Z 轴清零信号输入
9	ADC_TRIGGER0/1，ADC0/1 采样触发信号输出，每发生一次 ADC 采样触发，ADC_TRIGGER 信号翻转一次
10	CAN，CAN RX 或 TX
11	SIF，一线通串行通讯接口
12	CL，可配置逻辑（Configurable Logic）阵列 UNIT 输出

15.2 寄存器

15.2.1 地址分配

GPIO 0 模块在芯片中的基地址是 0x4001_0D00。

GPIO 1 模块在芯片中的基地址是 0x4001_0D40。

GPIO 2 模块在芯片中的基地址是 0x4001_0D80。

GPIO 3 模块在芯片中的基地址是 0x4001_0DC0。

GPIO 0/1/2/3 的寄存器定义完全相同，仅基地址不同。

表 15-2 GPIOx 寄存器列表

名称	偏移地址	说明
GPIOx_PIE	0x00	GPIO x 输入使能
GPIOx_POE	0x04	GPIO x 输出使能
GPIOx_PDI	0x08	GPIO x 输入数据
GPIOx_PDO	0x0C	GPIO x 输出数据

GPIOx_PUE	0x10	GPIO x 上拉使能
GPIOx_PODE	0x18	GPIO x 开漏使能
GPIOx_PFLT	0x1C	GPIO x 滤波使能
GPIOx_F3210	0x20	GPIO x [3:0]功能选择
GPIOx_F7654	0x24	GPIO x [7:4]功能选择
GPIOx_FBA98	0x28	GPIO x [11:8]功能选择
GPIOx_FFEDC	0x2C	GPIO x [15:12]功能选择
GPIOx_BSRR	0x30	GPIO x 位操作寄存器
GPIOx_BRR	0x34	GPIO x 位清零寄存器

GPIO 中断模块的基地址是 0x4001_0E00。

表 15-3 GPIO 中断模块寄存器列表

名称	偏移地址	说明
EXTI_CR0	0x00	外部中断配置寄存器 0
EXTI_CR1	0x04	外部中断配置寄存器 1
EXTI_IE	0x08	GPIO 中断/DMA 使能
EXTI_IF	0x0C	GPIO 中断标志
CLKO_SEL	0x10	输出时钟选择信号
PWM_SWAP	0x14	MCPWM 通道重排

15.2.2 GPIOx_PIE

地址分别是:0x4001_0D00, 0x4001_0D40, 0x4001_0D80, 0x4001_0DC0

复位值:0x0

表 15-4 GPIOx 输入使能寄存器 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

位置	位名称	说明
[31:16]		未使用
[15]	PIE15	GPIO x[15] / Px[15] 输入使能
[14]	PIE14	GPIO x[14] / Px[14] 输入使能
[13]	PIE13	GPIO x[13] / Px[13] 输入使能
[12]	PIE12	GPIO x[12] / Px[12] 输入使能
[11]	PIE11	GPIO x[11] / Px[11] 输入使能
[10]	PIE10	GPIO x[10] / Px[10] 输入使能
[9]	PIE9	GPIO x[9] / Px[9] 输入使能
[8]	PIE8	GPIO x[8] / Px[8] 输入使能
[7]	PIE7	GPIO x[7] / Px[7] 输入使能

[6]	PIE6	GPIO x[6] / Px[6] 输入使能
[5]	PIE5	GPIO x[5] / Px[5] 输入使能
[4]	PIE4	GPIO x[4] / Px[4] 输入使能
[3]	PIE3	GPIO x[3] / Px[3] 输入使能
[2]	PIE2	GPIO x[2] / Px[2] 输入使能
[1]	PIE1	GPIO x[1] / Px[1] 输入使能
[0]	PIE0	GPIO x[0] / Px[0] 输入使能

15.2.3 GPIOx_POE

地址分别是:0x4001_0D04, 0x4001_0D44, 0x4001_0D84, 0x4001_0DC4

复位值:0x0

表 15-5 GPIOx 输出使能寄存器 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

位置	位名称	说明
[31:16]		未使用
[15]	POE15	GPIO x[15] / Px[15] 输出使能
[14]	POE14	GPIO x[14] / Px[14] 输出使能
[13]	POE13	GPIO x[13] / Px[13] 输出使能
[12]	POE12	GPIO x[12] / Px[12] 输出使能
[11]	POE11	GPIO x[11] / Px[11] 输出使能
[10]	POE10	GPIO x[10] / Px[10] 输出使能
[9]	POE9	GPIO x[9] / Px[9] 输出使能
[8]	POE8	GPIO x[8] / Px[8] 输出使能
[7]	POE7	GPIO x[7] / Px[7] 输出使能
[6]	POE6	GPIO x[6] / Px[6] 输出使能
[5]	POE5	GPIO x[5] / Px[5] 输出使能
[4]	POE4	GPIO x[4] / Px[4] 输出使能
[3]	POE3	GPIO x[3] / Px[3] 输出使能
[2]	POE2	GPIO x[2] / Px[2] 输出使能
[1]	POE1	GPIO x[1] / Px[1] 输出使能
[0]	POE0	GPIO x[0] / Px[0] 输出使能

15.2.4 GPIOx_PDI

地址分别是:0x4001_0D08, 0x4001_0D48, 0x4001_0D88, 0x4001_0DC8

复位值:0x0

表 15-6 GPIOx 输入数据寄存器 GPIOx_PDI

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

位置	位名称	说明
[31:16]		未使用
[15:0]	PDI	GPIO x 输入数据

15.2.5 GPIOx_PDO

地址分别是:0x4001_0D0C, 0x4001_0D4C, 0x4001_0D8C, 0x4001_0DCC

复位值:0x0

表 15-7 GPIOx 输出数据寄存器 GPIOx_PDO

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

位置	位名称	说明
[31:16]		未使用
[15:0]	PDO	GPIO x 输出数据

15.2.6 GPIOx_PUE

地址分别是:0x4001_0D10, 0x4001_0D50, 0x4001_0D90, 0x4001_0DD0

复位值:0x0

表 15-8 GPIOx 上拉使能寄存器 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

位置	位名称	说明
[31:16]		未使用

[15]	PUE15	GPIO x[15] / Px[15] 上拉使能
[14]	PUE14	GPIO x[14] / Px[14] 上拉使能
[13]	PUE13	GPIO x[13] / Px[13] 上拉使能
[12]	PUE12	GPIO x[12] / Px[12] 上拉使能
[11]	PUE11	GPIO x[11] / Px[11] 上拉使能
[10]	PUE10	GPIO x[10] / Px[10] 上拉使能
[9]	PUE9	GPIO x[9] / Px[9] 上拉使能
[8]	PUE8	GPIO x[8] / Px[8] 上拉使能
[7]	PUE7	GPIO x[7] / Px[7] 上拉使能
[6]	PUE6	GPIO x[6] / Px[6] 上拉使能
[5]	PUE5	GPIO x[5] / Px[5] 上拉使能
[4]	PUE4	GPIO x[4] / Px[4] 上拉使能
[3]	PUE3	GPIO x[3] / Px[3] 上拉使能
[2]	PUE2	GPIO x[2] / Px[2] 上拉使能
[1]	PUE1	GPIO x[1] / Px[1] 上拉使能
[0]	PUE0	GPIO x[0] / Px[0] 上拉使能

注意:并非所有 IO 都具有上拉功能，具体哪些 IO 具有上拉功能请参考 15.3.1。没有上拉功能的 IO 对应的 PUE 寄存器也没有实现，因此向这些位置写入 1 无效，读回恒为 0。

15.2.7 GPIOx_PODE

地址分别是:0x4001_0D18, 0x4001_0D58, 0x4001_0D98, 0x4001_0DD8

复位值:0x0

表 15-9 GPIOx 开漏使能寄存器 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

位置	位名称	说明
[31:16]		未使用
[15]	PODE15	GPIO x[15] / Px[15] 开漏使能
[14]	PODE14	GPIO x[14] / Px[14] 开漏使能
[13]	PODE13	GPIO x[13] / Px[13] 开漏使能
[12]	PODE12	GPIO x[12] / Px[12] 开漏使能
[11]	PODE11	GPIO x[11] / Px[11] 开漏使能
[10]	PODE10	GPIO x[10] / Px[10] 开漏使能
[9]	PODE9	GPIO x[9] / Px[9] 开漏使能
[8]	PODE8	GPIO x[8] / Px[8] 开漏使能
[7]	PODE7	GPIO x[7] / Px[7] 开漏使能

[6]	PODE6	GPIO x[6] / Px[6] 开漏使能
[5]	PODE5	GPIO x[5] / Px[5] 开漏使能
[4]	PODE4	GPIO x[4] / Px[4] 开漏使能
[3]	PODE3	GPIO x[3] / Px[3] 开漏使能
[2]	PODE2	GPIO x[2] / Px[2] 开漏使能
[1]	PODE1	GPIO x[1] / Px[1] 开漏使能
[0]	PODE0	GPIO x[0] / Px[0] 开漏使能

15.2.8 GPIOx_PFLT

地址分别是:0x4001_0D1C, 0x4001_0D5C, 0x4001_0D9C, 0x4001_0DDC

复位值:0x0

表 15-10 GPIOx 滤波寄存器 GPIOx_PFLT

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

位置	位名称	说明
[31:16]		未使用
[15]	PFLT15	GPIO x[15] / Px[15] 滤波使能
[14]	PFLT14	GPIO x[14] / Px[14] 滤波使能
[13]	PFLT13	GPIO x[13] / Px[13] 滤波使能
[12]	PFLT12	GPIO x[12] / Px[12] 滤波使能
[11]	PFLT11	GPIO x[11] / Px[11] 滤波使能
[10]	PFLT10	GPIO x[10] / Px[10] 滤波使能
[9]	PFLT9	GPIO x[9] / Px[9] 滤波使能
[8]	PFLT8	GPIO x[8] / Px[8] 滤波使能
[7]	PFLT7	GPIO x[7] / Px[7] 滤波使能
[6]	PFLT6	GPIO x[6] / Px[6] 滤波使能
[5]	PFLT 5	GPIO x[5] / Px[5] 滤波使能
[4]	PFLT 4	GPIO x[4] / Px[4] 滤波使能
[3]	PFLT 3	GPIO x[3] / Px[3] 滤波使能
[2]	PFLT 2	GPIO x[2] / Px[2] 滤波使能
[1]	PFLT1	GPIO x[1] / Px[1] 滤波使能
[0]	PFLT0	GPIO x[0] / Px[0] 滤波使能

只有部分 GPIO 支持输入信号滤波，具体哪些 GPIO 支持滤波，请参考 15.3.2。

15.2.9 GPIOx_F3210

地址分别是:0x4001_0D20, 0x4001_0D60, 0x4001_0DA0, 0x4001_0DE0

复位值:0x0

表 15-11 GPIOx 功能选择寄存器 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			

位置	位名称	说明
[31:16]		未使用
[15:12]	F3	GPIO x[3] / Px[3] 功能选择
[11:8]	F2	GPIO x[2] / Px[2] 功能选择
[7:4]	F1	GPIO x[1] / Px[1] 功能选择
[3:0]	F0	GPIO x[0] / Px[0] 功能选择

其中 F3/F2/F1/F0 可以设置的 GPIO 第二功能如表 15-1 所示，GPIO 功能为稀疏映射，即每个 GPIO 仅可配置部分第二功能使用，需要 GPIO 功能分布，请参考对应器件手册(datasheet)。以下 GPIOx_F7654，GPIOx_FBA98，GPIOx_FFEDC 与 GPIOx_F3210 相同。

15.2.10 GPIOx_F7654

地址分别是:0x4001_0D24, 0x4001_0D64, 0x4001_0DA4, 0x4001_0DE4

复位值:0x0

表 15-12 GPIOx 功能选择寄存器 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			

位置	位名称	说明
[31:16]		未使用
[15:12]	F7	GPIO x[7] / Px[7] 功能选择
[11:8]	F6	GPIO x[6] / Px[6] 功能选择
[7:4]	F5	GPIO x[5] / Px[5] 功能选择
[3:0]	F4	GPIO x[4] / Px[4] 功能选择

15.2.11 GPIOx_FBA98

地址分别是:0x4001_0D28, 0x4001_0D68, 0x4001_0DA8, 0x4001_0DE8

复位值:0x0

表 15-13 GPIOx 功能选择寄存器 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

位置	位名称	说明
[31:16]		未使用
[15:12]	F11	GPIO x[11] / Px[11] 功能选择
[11:8]	F10	GPIO x[10] / Px[10] 功能选择
[7:4]	F9	GPIO x[9] / Px[9] 功能选择
[3:0]	F8	GPIO x[8] / Px[8] 功能选择

15.2.12 GPIOx_FFEDC

地址分别是:0x4001_0D2C, 0x4001_0D6C, 0x4001_0DAC, 0x4001_0DEC

复位值:0x0

表 15-14 GPIOx 功能选择寄存器 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

位置	位名称	说明
[31:16]		未使用
[15:12]	F15	GPIO x[15] / Px[15] 功能选择
[11:8]	F14	GPIO x[14] / Px[14] 功能选择
[7:4]	F13	GPIO x[13] / Px[13] 功能选择
[3:0]	F12	GPIO x[12] / Px[12] 功能选择

GPIO 的功能复用详细列表请见对应产品 DATASHEET 管脚分布章节。

15.2.13 GPIOx_BSRR

地址分别是:0x4001_0D30, 0x4001_0D70, 0x4001_0DB0, 0x4001_0DF0

复位值:0x0

表 15-15 GPIOx 位操作寄存器 GPIOx_BSRR

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

CLR15	CLR14	CLR13	CLR12	CLR11	CLR10	CLR9	CLR8	CLR7	CLR6	CLR5	CLR4	CLR3	CLR2	CLR1	CLR0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

位置	位名称	说明
[31]	CLR15	写 1 将 GPIO x[15]清零，写 0 无作用
[30]	CLR14	写 1 将 GPIO x[14]清零，写 0 无作用
[29]	CLR13	写 1 将 GPIO x[13]清零，写 0 无作用
[28]	CLR12	写 1 将 GPIO x[12]清零，写 0 无作用
[27]	CLR11	写 1 将 GPIO x[11]清零，写 0 无作用
[26]	CLR10	写 1 将 GPIO x[10]清零，写 0 无作用
[25]	CLR9	写 1 将 GPIO x[9]清零，写 0 无作用
[24]	CLR8	写 1 将 GPIO x[8]清零，写 0 无作用
[23]	CLR7	写 1 将 GPIO x[7]清零，写 0 无作用
[22]	CLR6	写 1 将 GPIO x[6]清零，写 0 无作用
[21]	CLR5	写 1 将 GPIO x[5]清零，写 0 无作用
[20]	CLR4	写 1 将 GPIO x[4]清零，写 0 无作用
[19]	CLR3	写 1 将 GPIO x[3]清零，写 0 无作用
[18]	CLR2	写 1 将 GPIO x[2]清零，写 0 无作用
[17]	CLR1	写 1 将 GPIO x[1]清零，写 0 无作用
[16]	CLR0	写 1 将 GPIO x[0]清零，写 0 无作用
[15]	SET15	写 1 将 GPIO x[15]置 1，写 0 无作用
[14]	SET14	写 1 将 GPIO x[14]置 1，写 0 无作用
[13]	SET13	写 1 将 GPIO x[13]置 1，写 0 无作用
[12]	SET12	写 1 将 GPIO x[12]置 1，写 0 无作用
[11]	SET11	写 1 将 GPIO x[11]置 1，写 0 无作用
[10]	SET10	写 1 将 GPIO x[10]置 1，写 0 无作用
[9]	SET9	写 1 将 GPIO x[9]置 1，写 0 无作用
[8]	SET8	写 1 将 GPIO x[8]置 1，写 0 无作用
[7]	SET7	写 1 将 GPIO x[7]置 1，写 0 无作用
[6]	SET6	写 1 将 GPIO x[6]置 1，写 0 无作用
[5]	SET5	写 1 将 GPIO x[5]置 1，写 0 无作用

[4]	SET4	写 1 将 GPIO x[4]置 1，写 0 无作用
[3]	SET3	写 1 将 GPIO x[3]置 1，写 0 无作用
[2]	SET2	写 1 将 GPIO x[2]置 1，写 0 无作用
[1]	SET1	写 1 将 GPIO x[1]置 1，写 0 无作用
[0]	SET0	写 1 将 GPIO x[0]置 1，写 0 无作用

若用 BSRR 的高 16 位与低 16 位同时对 GPIO 同一位置既置 1 又清零，则该位被清零。

15.2.14 GPIOx_BRR

地址分别是:0x4001_0D34, 0x4001_0D74, 0x4001_0DB4, 0x4001_0DF4

复位值:0x0

表 15-16 GPIOx 位清零寄存器 GPIOx_BRR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLR15	CLR14	CLR13	CLR12	CLR11	CLR10	CLR9	CLR8	CLR7	CLR6	CLR5	CLR4	CLR3	CLR2	CLR1	CLR0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:16]		未使用
[15]	CLR15	写 1 将 GPIO x[15]清零，写 0 无作用
[14]	CLR14	写 1 将 GPIO x[14]清零，写 0 无作用
[13]	CLR13	写 1 将 GPIO x[13]清零，写 0 无作用
[12]	CLR12	写 1 将 GPIO x[12]清零，写 0 无作用
[11]	CLR11	写 1 将 GPIO x[11]清零，写 0 无作用
[10]	CLR10	写 1 将 GPIO x[10]清零，写 0 无作用
[9]	CLR9	写 1 将 GPIO x[9]清零，写 0 无作用
[8]	CLR8	写 1 将 GPIO x[8]清零，写 0 无作用
[7]	CLR7	写 1 将 GPIO x[7]清零，写 0 无作用
[6]	CLR6	写 1 将 GPIO x[6]清零，写 0 无作用
[5]	CLR5	写 1 将 GPIO x[5]清零，写 0 无作用
[4]	CLR4	写 1 将 GPIO x[4]清零，写 0 无作用
[3]	CLR3	写 1 将 GPIO x[3]清零，写 0 无作用
[2]	CLR2	写 1 将 GPIO x[2]清零，写 0 无作用
[1]	CLR1	写 1 将 GPIO x[1]清零，写 0 无作用
[0]	CLR0	写 1 将 GPIO x[0]清零，写 0 无作用

15.2.15 外部事件

EXTI_CR0 和 EXTI_CR1 用于选择 GPIO 外部信号产生中断或 DMA 请求的上升沿/下降沿触发类型。

15.2.15.1 EXTI_CR0

地址:0x4001_0E00

复位值:0x0

表 15-17 EXTI_CR0 外部触发配置寄存器 0

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

位置	位名称	说明
[31:16]		未使用
[15:14]	T7	GPIO 0[15]/ P0[15]外部触发类型选择 00:不触发 01:下降沿触发 10:上升沿触发 11:上升沿、下降沿均触发
[13:12]	T6	GPIO 0[14]/ P0[14]外部触发类型选择 00:不触发 01:下降沿触发 10:上升沿触发 11:上升沿、下降沿均触发
[11:10]	T5	GPIO 0[11]/ P0[11]外部触发类型选择 00:不触发 01:下降沿触发 10:上升沿触发 11:上升沿、下降沿均触发
[9:8]	T4	GPIO 0[6]/ P0[6]外部触发类型选择 00:不触发 01:下降沿触发 10:上升沿触发 11:上升沿、下降沿均触发
[7:6]	T3	GPIO 0[3]/ P0[3]外部触发类型选择 00:不触发 01:下降沿触发 10:上升沿触发 11:上升沿、下降沿均触发
[5:4]	T2	GPIO 0[2]/ P0[2]外部触发类型选择 00:不触发 01:下降沿触发 10:上升沿触发 11:上升沿、下降沿均触发
[3:2]	T1	GPIO 0[1]/ P0[1]外部触发类型选择 00:不触发

		01:下降沿触发 10:上升沿触发 11:上升沿、下降沿均触发
[1:0]	T0	GPIO 0[0]/ P0[0]外部触发类型选择 00:不触发 01:下降沿触发 10:上升沿触发 11:上升沿、下降沿均触发

15.2.15.2 EXTI_CR1

地址:0x4001_0E04

复位值:0x0

表 15-18 EXTI_CR1 外部触发配置寄存器 1

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	

位置	位名称	说明
[31:16]		未使用
[15:14]	T15	GPIO 2[15]/P2[15]外部触发类型选择 00:不触发 01:下降沿触发 10:上升沿触发 11:上升沿、下降沿均触发
[13:12]	T14	GPIO 2[12]/P2[12]外部触发类型选择 00:不触发 01:下降沿触发 10:上升沿触发 11:上升沿、下降沿均触发
[11:10]	T13	GPIO 2[7]/P2[7]外部触发类型选择 00:不触发 01:下降沿触发 10:上升沿触发 11:上升沿、下降沿均触发
[9:8]	T12	GPIO 2[4]/P2[4]外部触发类型选择 00:不触发 01:下降沿触发 10:上升沿触发 11:上升沿、下降沿均触发
[7:6]	T11	GPIO 2[3]/P2[3]外部触发类型选择 00:不触发

		01:下降沿触发 10:上升沿触发 11:上升沿、下降沿均触发
[5:4]	T10	GPIO 1[10]/P1[10]外部触发类型选择 00:不触发 01:下降沿触发 10:上升沿触发 11:上升沿、下降沿均触发
[3:2]	T9	GPIO 1[1]/P1[1]外部触发类型选择 00:不触发 01:下降沿触发 10:上升沿触发 11:上升沿、下降沿均触发
[1:0]	T8	GPIO 1[0]/P1[0]外部触发类型选择 00:不触发 01:下降沿触发 10:上升沿触发 11:上升沿、下降沿均触发

表 15-19 GPIO 中断/DMA 请求事件资源分布表

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

15.2.15.3 EXTI_IE

地址:0x4001_0E08

复位值:0x0

表 15-20 EXTI_IE GPIO 中断使能寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				EXIT3_RE	EXIT2_RE	EXIT1_RE	EXTI0_RE					EXIT3_IE	EXIT2_IE	EXIT1_IE	EXTI0_IE
				RW	RW	RW	RW					RW	RW	RW	RW
				0	0	0	0					0	0	0	0

位置	位名称	说明
[31:12]		未使用
[11]	EXTI3_RE	GPIO3 DMA 请求使能，需要配合 EXTI_CR0/1 使用
[10]	EXTI2_RE	GPIO2 DMA 请求使能，需要配合 EXTI_CR0/1 使用

[9]	EXTI1_RE	GPIO1 DMA 请求使能，需要配合 EXTI_CR0/1 使用
[8]	EXTI0_RE	GPIO0 DMA 请求使能，需要配合 EXTI_CR0/1 使用
[7:4]		未使用
[3]	EXTI3_IE	GPIO3 中断使能，需要配合 EXTI_CR0/1 使用
[2]	EXTI2_IE	GPIO2 中断使能，需要配合 EXTI_CR0/1 使用
[1]	EXTI1_IE	GPIO1 中断使能，需要配合 EXTI_CR0/1 使用
[0]	EXTI0_IE	GPIO0 中断使能，需要配合 EXTI_CR0/1 使用

每一组 GPIO 外部请求信号可以统一用作中断请求，或者 DMA 请求。但通常每组信号不会同时用作中断和 DMA 请求。DMA 请求信号的产生同样来自中断标志置位，当中断发生后，如果使能了 EXTIx_RE，则 DMA 会在响应请求后清除该组 GPIO 的所有中断标志。举例来说 GPIO0 中，通常只会使用 16 个 GPIO P0.0~P0.15 中的一个作为外部事件来源，且要么当做中断请求使用，要么当做 DMA 请求使用。如果同时使能 P0.0 和 P0.1 的外部事件请求，则两个 IO 都会产生外部事件。如果作为 DMA 请求，则 P0.0 和 P0.1 中任意一个外部事件发生时，DMA 都会清除 P0.0~P0.15 所有的外部事件标志。

但是允许使用 GPIO0 中的一个 IO 作为中断事件请求，同时使用 GPIO2 中的一个 IO 作为 DMA 请求使能。则 GPIO2 的 IO 发生 DMA 请求时，DMA 只会硬件清零 GPIO2 的外部事件标志，不会影响 GPIO0 的外部事件标志，GPIO0 的外部事件会产生中断请求，由中断服务程序进行处理。

15.2.15.4 EXTI_IF

地址:0x4001_0E0C

复位值:0x0

表 15-21 EXTI_IF 外部中断标志寄存器

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

位置	位名称	说明
[31:16]		未使用
[15]	IF15	GPIO 2[15] / P2[15] 外部中断标志。中断标志高有效，写 1 清零
[14]	IF14	GPIO 2[12] / P2[12] 外部中断标志。中断标志高有效，写 1 清零
[13]	IF13	GPIO 2[7] / P2[7] 外部中断标志。中断标志高有效，写 1 清零
[12]	IF12	GPIO 2[4] / P2[4] 外部中断标志。中断标志高有效，写 1 清零
[11]	IF11	GPIO 2[3] / P2[3] 外部中断标志。中断标志高有效，写 1 清零
[10]	IF10	GPIO 1[10] / P1[10] 外部中断标志。中断标志高有效，写 1 清零
[9]	IF9	GPIO 1[1] / P1[1] 外部中断标志。中断标志高有效，写 1 清零
[8]	IF8	GPIO 1[0] / P1[0] 外部中断标志。中断标志高有效，写 1 清零

[7]	IF7	GPIO 0[15] / P0[15] 外部中断标志。中断标志高有效，写 1 清零
[6]	IF6	GPIO 0[14] / P0[14] 外部中断标志。中断标志高有效，写 1 清零
[5]	IF5	GPIO 0[11] / P0[11] 外部中断标志。中断标志高有效，写 1 清零
[4]	IF4	GPIO 0[6] / P0[6] 外部中断标志。中断标志高有效，写 1 清零
[3]	IF3	GPIO 0[3] / P0[3] 外部中断标志。中断标志高有效，写 1 清零
[2]	IF2	GPIO 0[2] / P0[2] 外部中断标志。中断标志高有效，写 1 清零
[1]	IF1	GPIO 0[1] / P0[1] 外部中断标志。中断标志高有效，写 1 清零
[0]	IF0	GPIO 0[0] / P0[0] 外部中断标志。中断标志高有效，写 1 清零

即使不使能 EXTI_IE，EXTI_IF 仍会置位，但不会产生中断请求或 DMA 请求。

15.2.15.5 CLKO_SEL

地址:0x4001_0E10

复位值:0x0

表 15-22 CLKO_SEL GPIO 输出时钟信号选择寄存器

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

位置	位名称	说明
[31:5]		未使用
[4]	HSE_OE	晶振时钟输出使能。1:使能；0:禁用。
[3]	ADC_OE	ADC 时钟输出使能。1:使能；0:禁用。
[2]	PLL_OE	PLL 输出使能。1:使能；0:禁用。
[1]	HSI_OE	HRC 输出使能。1:使能；0:禁用。
[0]	LSI_OE	LRC 输出使能。1:使能；0:禁用。

芯片各时钟信号可以从 P0.0/P2.7 进行输出观测。注意由于 ADC/PLL 为高频时钟信号，可能无法驱动片外大负载。

如果需要输出时钟，需要配置 P0.0 的第二功能为 1，即 GPIO0_F3210 = 0x0001，并使能 P0.0 输出使能 GPIO0_POE=0x0001，并配置 CLKO_SEL，选择某一路时钟通过 P0.0 进行输出。

15.2.15.6 PWM_SWAP

地址:0x4001_0E14

复位值:0x0

表 15-23 PWM_SWAP 寄存器

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

	SWAP
	RW
	0

位置	位名称	说明
[31:2]		未使用
[1:0]	SWAP	向此寄存器写入 0x67 可将 BIT[1:0]写为 1，写入 0x69 可将 BIT[1:0]写为 2，写其他值则将 BIT[1:0]写为 0

MCPWM0_SWAP 的值为 0 时，MCPWM 通道输出与 GPIO 关系如下:

表 15-24 MCPWM 默认输出表

MCPWM 输出顺序	GPIO 对应顺序	
MCPWM0_CH0P	P1.4	P0.15
MCPWM0_CH0N	P1.5	P1.0
MCPWM0_CH1P	P1.6	P2.11
MCPWM0_CH1N	P1.7	P2.12
MCPWM0_CH2P	P1.8	P2.4
MCPWM0_CH2N	P1.9	P2.5
MCPWM0_CH3P	P1.10	P3.2
MCPWM0_CH3N	P1.11	P3.4
MCPWM0_CH4P	P3.10	P0.3
MCPWM0_CH4N	P3.11	P0.4
MCPWM0_CH5P	P2.9	P0.5
MCPWM0_CH5N	P2.10	P0.6

为了便于 MCU 芯片与 Gate Driver 芯片进行多 die 封装 SIP 设计，GPIO 模块内 PWM 通道输出可以进行重新映射。下表中最左侧一列 MCPWM 通道进行了重排，右侧两列不变，仍与芯片的 IO 排布顺序一致。

PWM_SWAP 的值为 1 时，对应关系如下:

表 15-25 PWM_SWAP=1 时 MCPWM 输出映射表表

MCPWM 输出顺序	GPIO 对应顺序	
MCPWM0_CH0N	P1.4	
MCPWM0_CH1N	P1.5	
MCPWM0_CH2N	P1.6	
MCPWM0_CH0P	P1.7	
MCPWM0_CH1P	P1.8	
MCPWM0_CH2P	P1.9	
MCPWM0_CH3N	P1.10	P3.2
MCPWM0_CH4N	P1.11	P3.4
MCPWM0_CH5N	P3.10	P0.3
MCPWM0_CH3P	P3.11	P0.4
MCPWM0_CH4P	P2.9	P0.5

MCPWM0_CH5P	P2.10	P0.6

PWM_SWAP 的值为 2 时，对应关系如下:

表 15-26 PWM_SWAP=2 时 MCPWM 输出映射表

MCPWM 输出顺序	GPIO 对应顺序	
MCPWM0_CH0N	P1.13	
MCPWM0_CH1N	P1.14	
MCPWM0_CH2N	P1.15	
MCPWM0_CH0P	P2.0	
MCPWM0_CH1P	P1.4	
MCPWM0_CH2P	P1.5	
MCPWM0_CH3N	P1.6	
MCPWM0_CH4N	P1.7	
MCPWM0_CH5N	P1.8	
MCPWM0_CH3P	P1.9	
MCPWM0_CH4P	P1.10	
MCPWM0_CH5P	P1.11	

15.3 实现说明

15.3.1 上拉实现

LKS32MC07x 系列芯片，部分 GPIO 配备有上拉电阻。配备上拉功能的 GPIO 如下:

表 15-27 GPIO 上拉资源分布表

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

15.3.2 滤波实现

LKS32MC07x 系列芯片，部分 GPIO 支持对输入信号进行滤波操作，滤波时间宽度为 4 个 LSI 时钟周期，约 120us，即不足 120us 的变化会被滤波滤除。注意由于滤波使用 LSI 时钟，而 RC 本身精度有限，所以具体滤波时间常数会存在个体偏差。

表 15-28 GPIO 滤波资源分布表

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

15.3.3 开漏实现

LKS32MC07x 系列芯片，当 IO 功能配置为 0，即作为普通 IO 使用时，所有 IO 均具有开漏功能；但当 GPIO 功能配置不为 0，即复用为某些特定接口而非普通 IO 功能时，只有部分 IO 具备开

漏功能，如下表所示。

表 15-29 GPIO 开漏资源分布表

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

15.4 应用指南

15.4.1 外部中断

示例如下:

```
GPIO0_PIE = 0x0001;                 // 使能 P0[0]输入
NVIC_EnableIRQ(GPIO_IRQn);          //使能 GPIO 中断
__enable_irq();                     //使能中断
i = 1000;
while(i--);
// P0[7] IO 上外接方波信号
EXTI_CR0 = 0x0002;                  // 使能 p0[7]上升沿触发，产生外部中断
    while(irq_flag != 2);               // 外部信号翻转两次产生两次中断，irq_flag 在 GPIO 中断处理程序中递增两次
EXTI_CR0 = 0x0001;                  // 使能 p0[7] 下降沿触发，产生外部中断
    while(irq_flag != 4);
EXTI_CR0 = 0x00003;                 // 同时使能 P0[7]上升沿、下降沿触发，产生外部中断
    while(irq_flag != 8);
EXTI_CR0 = 0x0000;                  // 同时禁用 P[7]上下沿触发，将无法产生外部中断
```

15.4.2 使用 GPIO 的模拟功能

将 GPIO 的 IE 和 OE 关闭，即可使用模拟功能。此时，PAD 通过内部电阻直接与模拟模块相连。

16 CRC

16.1 概述

CRC 即循环冗余校验码（Cyclic Redundancy Check）：是数据通信领域中最常用的一种查错校验码，其特征是信息字段和校验字段的长度可以任意选定。循环冗余检查（CRC）是一种数据传输检错功能，对数据进行多项式计算，并将得到的结果附在帧的后面，接收设备也执行类似的算法，以保证数据传输的正确性和完整性。

利用 CRC 进行检错的过程可简单描述为：在发送端根据要传送的 K 位二进制码序列，以一定的规则产生一个校验用的 R 位监督码(CRC 码)，附在原始信息后边，构成一个新的二进制码序列数共 K+R 位，然后发送出去。在接收端，根据信息码和 CRC 码之间所遵循的规则进行检验，以确定传送中是否出错。这个规则，在差错控制理论中称为“生成多项式”。

16.2 基本原理

循环冗余校验码（CRC）的基本原理是：在 K 位信息码后再拼接 R 位的校验码，整个编码长度为 N 位，因此，这种编码也叫（N，K）码。对于一个给定的（N，K）码，可以证明存在一个最高次幂为 N-K=R 的多项式 G(x)。根据 G(x)可以生成 K 位信息的校验码，而 G(x)叫做这个 CRC 码的生成多项式。校验码的具体生成过程为：假设要发送的信息用多项式 C(X)表示，将 C(x)左移 R 位（可表示成 $C(x)*2^R$），这样 C(x)的右边就会空出 R 位，这就是校验码的位置。用 $C(x)*2^R$ 除以生成多项式 G(x)得到的余数就是校验码。

任意一个由二进制位串组成的代码都可以和一个系数仅为‘0’和‘1’取值的多项式一一对应。例如：代码 1010111 对应的多项式为 $x^6+x^4+x^2+x+1$，而多项式为 $x^5+x^3+x^2+x+1$ 对应的代码 101111。

16.3 基本概念

16.3.1 对应关系

多项式和二进制数有直接对应关系：X 的最高幂次对应二进制数的最高位，以下各位对应多项式的各幂次，有此幂次项对应 1，无此幂次项对应 0。可以看出：X 的最高幂次为 R，转换成对应的二进制数有 R+1 位。

多项式包括生成多项式 G(X)和信息多项式 C(X)。

如生成多项式为 $G(X)=X^4+X^3+X+1$，可转换为二进制数码 11011。

而发送信息为 101111，可转换为数据多项式为 $C(X)=X^5+X^3+X^2+X+1$。

16.3.2 生成多项式

生成多项式是接受方和发送方的一个约定，也就是一个二进制数，在整个传输过程中，这个数始终保持不变。

在发送方，利用生成多项式对信息多项式做模 2 除生成校验码。在接收方利用生成多项式对收到的编码多项式做模 2 除检测和确定错误位置。

应满足以下条件：

A、生成多项式的最高位和最低位必须为 1。

B、当被传送信息（CRC 码）任何一位发生错误时，被生成多项式做除后应该使余数不为 0。

C、不同位发生错误时，应该使余数不同。

D、对余数继续做除，应使余数循环。

16.3.3 校验码位数

CRC 校验码位数 = 生成多项式位数 - 1。注意有些生成多项式的简记式中将生成多项式的最高位 1 省略了。

16.3.4 生成步骤

1、将 X 的最高次幂为 R 的生成多项式 G(X)转换成对应的 R+1 位二进制数。

2、将信息码左移 R 位，相当于对应的信息多项式 $C(X)*2^R$。

3、用生成多项式（二进制数）对信息码做除，得到 R 位的余数(注意：这里的二进制做除法得到的余数其实是模 2 除法得到的余数，并不等于其对应十进制数做除法得到的余数。)。

4、将余数拼到信息码左移后空出的位置，得到完整的 CRC 码。

【例】假设使用的生成多项式是 $G(X)=X^3+X+1$。4 位的原始报文为 1010，求编码后的报文。

解：

1、将生成多项式 $G(X)=X^3+X+1$ 转换成对应的二进制除数 1011。

2、此题生成多项式有 4 位（R+1）(注意：4 位的生成多项式计算所得的校验码为 3 位，R 为校验码位数)，要把原始报文 C(X)左移 3（R）位变成 1010 000

3、用生成多项式对应的二进制数对左移 3 位后的原始报文进行模 2 除（高位对齐），相当于按位异或：

1010000

1011

0001000

0001011

0000011

得到的余位 011，所以最终编码为：1010 011

POL=0x13，data=0x77

011101110000000

10010011

01111101000000

10010011

0110100100000

10010011

010000010000

10010011

00010001000

10010011

00011011

16.4 寄存器

16.4.1 地址分配

CRC 的基地址是 0x4001_0F00，寄存器列表如下：

表 16-1 CRC 寄存器列表

名称	偏移地址	说明
CRC0_DR	0x00	CRC 数据（输入信息码/输出编码）寄存器
CRC0_CR	0x04	CRC 控制寄存器
CRC0_INIT	0x08	CRC 初始码寄存器
CRC0_POL	0x0C	CRC 生成多项式对应的二进制码寄存器

16.4.2 寄存器描述

16.4.2.1CRC0_DR CRC 信息码寄存器

地址：0x4001_0F00

复位值：0x0

表 16-2 CRC0_DR CRC 数据寄存器

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

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

位置	位名称	说明
[31:0]	DR	存放待编码的信息码和经 CRC 校验后的编码

CRC0_DR 寄存器既用于放入待校验数据，也用于返回校验结果。写入 CRC0_DR 寄存器即触发一次 CRC 计算。待编码数据应在 CR 等寄存器配置完成后最后写入，以触发 CRC 计算开始。

16.4.2.2CRC0_CR CRC 控制寄存器

地址：0x4001_0F04

复位值：0x0

表 16-3 CRC0_CR CRC 控制寄存器

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

位置	位名称	说明
[31:13]		未使用
[12]	REV_OUT_TYPE	是否将CRC校验后的编码反转后输出，即b[31]=b[0], b[30]=b[1],…[b0]=b[31]
[11:10]		未使用

[9:8]	REV_IN_TYPE	待编码数据反转类型 00:不反转 01:按字节反转，即 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:按半字（16bit）反转，即 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:按字反转，即 b[31]=b[0], b[30]=b[1],… [b0]=b[31]
[7:6]		未使用
[5:4]	POLY_SIZE	输出编码（多项式）位宽 00: 32bits 01: 16bits 10: 8bits 11: 7bits
[3:1]		未使用
[0]	RESET	与输入信息码进行 CRC 计算的数据来源 0:来自于上一次的计算结果 1:来自于 CRC0_INIT 写入 1 实现 CRC 数据重置并自动清零，读回恒为 0.

同时需要注意的是，向 CRC0_CR.RESET 写入 1 会将 CRC0_INIT 寄存器复位为 0xFFFFFFFF。

如果需要清除 CRC 的计算结果，应向 CRC0_CR.RESET 写入 1，否则后续 CRC 计算会以之前的计算结果为初值进行。

16.4.2.3CRC0_INIT CRC 初始码寄存器

地址：0x4001_0F08

复位值：0x0

表 16-4 CRC0_INIT CRC 初始码寄存器

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															

位置	位名称	说明
[31:0]	INIT	存放初始码

初 CRC0_DR 与 CRC0_INIT 相异或后开始进行 CRC 校验计算。

16.4.2.4CRC0_POL CRC 生成码寄存器

地址：0x4001_0F0C

复位值：0x0

表 16-5 CRC0_POL CRC 生成码寄存器

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															

位置	位名称	说明
[31:0]	POL	存放生成多项式对应的生成码

17 UART

17.1 概述

通用异步收发传输器（Universal Asynchrounous Receiver/Transmitter），通常称作 UART，是一种异步收发传输器。

UART 主要特征如下:

- 支持全双工工作
- 支持单线半双工工作
- 支持 8/9 位数据位
- 支持 1/2 停止位
- 支持奇/偶/无校验模式
- 带 1 字节发送缓存
- 带 1 字节接收缓存
- 支持 LIN 模式 break character 收发
- 支持空闲帧检测
- 支持一主多从的 Multi-drop Slave/Master 模式

17.2 功能说明

17.2.1 发送

UART 包括一个字节发送缓冲区，当发送缓冲区有数据时，UART 将发送缓冲区的数据移入串行移位寄存器，并通过 UART_TXD 端口发送出去。只要 UART 发送缓冲区有数据，UART 就会自动进行发送。

完成加载后，产生发送缓冲区空中断，此时，用户可以往发送缓冲区填入下一个需要发送的字节，这样，发送完成后，UART 将加载这个字节进行发送。

完成发送后，会产生发送完成中断。

发送流程:

设置 SYS_CLK_FEN. UART_CLK_EN = 1 开启 UART 时钟

设置 UART_CTRL.BYTE_LEN，选择 8/9bit 数据字节长度

设置 UART_CTRL.CK_EN，选择是否启用数据字节校验

设置 UART_CTRL.CK_TYPE，选择使用奇校验还是偶校验

设置 UART_CTRL.BIT_ORDER，选择发送时 LSB first 还是 MSB first

设置 UART_CTRL.STOP_LEN，选择停止比特长度为 1bit/2bit

如果使用 DMA 搬运数据，则设置 UART_RE. TX_BUF_EMPTY_RE=1，即 TX buffer 空就触发 DMA 搬运新数据到 UART；如果使用软件轮询或中断的方式，则设置 UART_IE. TX_BUF_EMPTY_IE=1

如果使用 DMA 搬运数据到 UART 发送，需要对 DMA 进行相应设置。

如果使用软件搬运数据到 UART 发送，需要软件向 UART_BUFF 写入数据，可以触发 UART 开始通过 UART_TXD 端口发送数据。

17.2.2 接收

UART 包括一个字节的接收缓冲区，当完成一个字节的接收后，会产生接收中断，并将接收到字节存储到接收缓冲区，用户应当在 UART 接收完成下一个字节前完成此字节的读取，否则缓冲区会被写入新接收的字节。接收部分内部使用高速时钟对 UART_RX 信号进行过采样来判定稳态的数据信号，防止噪声干扰造成错误接收。

17.2.3 UART 帧格式

UART 信号上收发的数据格式通常如下：

信号线空闲；

起始比特 (1 bit Start bit： 1 bit Zero)

数据字节 (data word, 8bits or 9bits, LSB first or MSB first)

停止比特 (1/2 bit Stop bit: 1/2bit Ones)

数据字节的长度可以通过 UART_CTRL.BYTE_LEN 进行选择，8bit (UART_CTRL.BYTE_LEN=0)或 9bit (UART_CTRL.BYTE_LEN=1)。起始比特为 1bit 零，TX 信号线为低，停止比特期间 TX 信号线为高。停止比特的长度由 UART_CTRL.STOP_LEN 进行选择。

需要注意的是，当启用校验位时(UART_CTR.CK_EN=1)，校验位会替换掉数据字节的最高 bit，即 MSB，此时 8bit 数据字节实际是 7bit 数据+1bit 校验字，9bit 数据字节实际是 8bit 数据+1bit 校验字。

图 17-1 UART 帧格式

17.2.4 波特率配置

UART 输入时钟为系统主时钟，波特率通过两级分频实现。

波特率=UART 模块时钟/（256*DIVH+DIVL+1)

可以通过 SYS_CLK_DIV2 对 UART 模块时钟进行分频.

UART 模块时钟=系统主时钟/(1+SYS_CLK_DIV2)

表 17-1 UART 波特率配置示例

UART 波特率	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

注意，同一波特率，其配置系数可能不唯一。

17.2.5 收发端口互换(TX/RX 互换)

UART 模块支持 Tx 与 Rx 端口互换。通过将 Tx 对应的 GPIO 配置为输入使能，Rx 对应的 GPIO 配置为输出使能，即可实现 Tx 与 Rx 端口的互换。此时，GPIO 第二功能仍选择为 UART，UART 本身配置无需修改。

此外，如果要使用一个 GPIO 同时作为 Tx 和 Rx，需要将 IO 分时复用为输入或输出，对应 Rx 或 Tx，即可实现单口半双工逻辑。

17.2.6 多机通讯

多机通讯场景通常是一个设备作为主设备 master，另有若干个从设备 slave，主设备的 UARTm_TXD 端口连接到所有从设备的 UARTs_RXD 端口，从设备的 UARTs_TXD 相与连接到主设备的 UARTm_RXD 端口。如图 17-2 UART 多机通讯互联拓扑所示，为一个主设备和 3 个从设备（地址分别为 0x51,0x52,0x53）的互联情况。

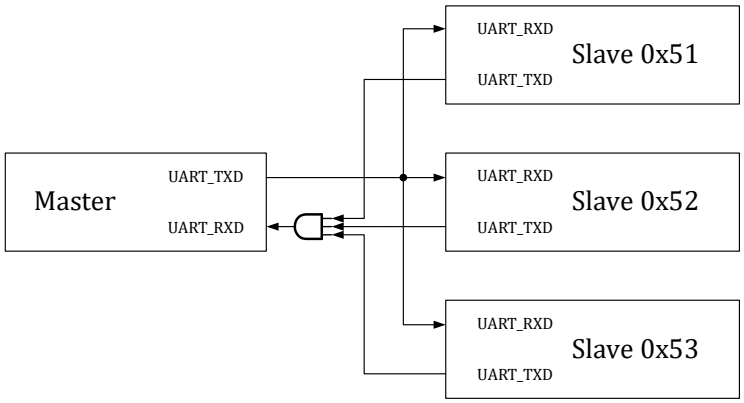


图 17-2 UART 多机通讯互联拓扑

为了降低从机报文处理的负荷，从设备应该只处理发送给他的 UART 数据，而忽略发送给其他从设备的数据。在多机通讯场景中，每一个从机有一个 8bit 的设备地址，即 UART_ADR。从设备设置 UART_CTRL.MD_EN=1 后，开启多机通讯过滤模式，主机无须设置 MD_EN。主机、从机均需要使用 9bit 模式发送数据，即设置 UART_CTRL.BYTE_LEN=1。当发送数据字节 MSB(BIT8)为 1 时，低 8 位(BIT7:BIT0)是主机发出的从机地址，即**地址字节**；当发送数据字节 MSB(BIT8)为 0 时，低 8 位(BIT7:BIT0)是主机发给特定从机的数据，即**数据字节**。

所有从机接收到 MSB(BIT8)=1 的数据字节后，判断低 8 位(BIT7:BIT0)地址是否与自己的 UART_ADR 配置值相等。如果相等，则说明后续数据都是发送给此从机，否则从机进入静默状态，忽略所有后续主机发出的数据。当从机再次接收到 MSB(BIT8)=1 的数据字节且低 8 位(BIT7:BIT0)地址与自己的 UART_ADR 配置值相等时，表示此从机被选中，退出静默状态。从机设置 UART_CTRL.MD_EN=1 开启多机通讯过滤模式后，立即进入静默状态。如果主机没有发送 MSB=1 的地址字节，直接发送数据字节，则所有从机都处于静默状态，不会接收任何数据。

在多机通讯模式中，主设备无需设置 UART_CTRL.MD_EN=1，从设备需要设置 UART_CTRL.MD_EN=1。如果无需地址过滤，从设备也可以不设置 UART_CTRL.MD_EN=1，另使用软件对接收数据进行判读，此时从设备会接收并处理所有报文数据。

在多机通讯模式中，如果从设备设置了 UART_CTRL.MD_EN=1，则地址字节无论是否命中 UART_ADR，UART_IF.RX_DONE_IF 标志均不置位；如果地址不匹配，即从设备处于静默状态，即使主设备发送数据，UART_IF.RX_DONE_IF 仍不置位。如果地址匹配，当接收到主设备发送的数据字节时，从设备的 UART_IF.RX_DONE_IF 标志置位为 1，表示从设备接收到一个数据字节。

由于检验位需要占用 1bit 数据位，而多机通讯需要使用 9bit 字节长度。因此多机通讯场景下不支持校验位，在设置 UART_CTRL.MD_EN=1 时，请勿设置 UART_CTRL.CK_EN=1。主机、从机均需设置 UART_CTRL.BYTE_LEN=1 使用 9bit 模式，并设置 UART_CTRL.MD_EN。多机通讯支持 MSB first，即 UART_CTRL.BIT_ORDER=1。

如图 17-3 UART 多机通讯示例所示，主设备先发送了数据 0x03，但此时没有从设备地址命中，因此 3 个从设备均不接收 0x03。主设备发送地址字节 0x151，从设备 0x51 被选中。但之后主设备没有发送数据，而是直接发送地址字节 0x153，从设备 0x53 被选中，主设备连续发送 3 个数据字节 0x03,0x53,0x04，均被从设备 0x53 接收。之后主设备发送地址字节 0x152，之后发送数据字节 0x07，被从设备 0x52 接收。

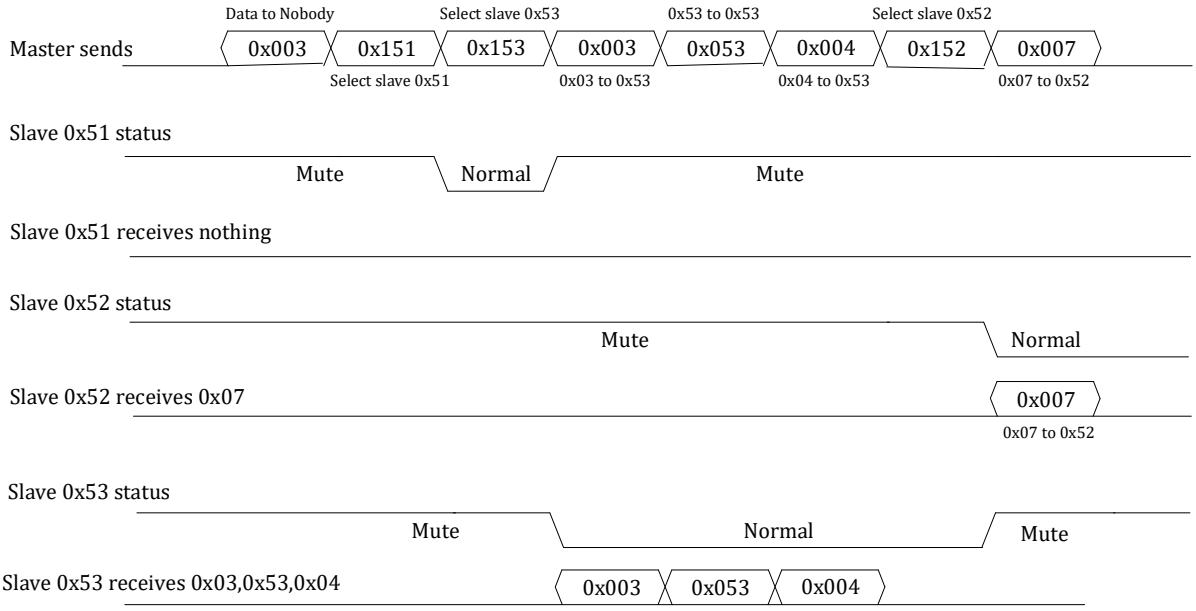


图 17-3 UART 多机通讯示例

17.2.7 校验位

可以通过设置 UART_CTRL.CK_EN=1 来使能校验位。同时根据字节长度 UART_CTRL.BYTE_LEN 不同，UART 的帧格式有 4 种情况。

表 17-2 UART 帧格式

BYTE_LEN	CK_EN	UART frame
0	0	| StartBit | 8bit data | StopBit |

0	1	| StartBit | 7bit data | Parity | StopBit |
1	0	| StartBit | 9bit data | StopBit |
1	1	| StartBit | 8bit data | Parity | StopBit |

17.3 寄存器

17.3.1 地址分配

UART0 和 UART1 实现完全相同。

UART0 基地址 0x4001_1000。

UART1 基地址 0x4001_1100。

表 17-3 UART 地址分配列表

名称	偏移地址	说明
UARTx_CTRL	0x00	UART 控制寄存器
UARTx_DIVH	0x04	UART 波特率设置高字节寄存器
UARTx_DIVL	0x08	UART 波特率设置低字节寄存器
UARTx_BUFF	0x0C	UART 收发缓冲寄存器
UARTx_ADR	0x10	485 通信地址匹配寄存器
UARTx_STT	0x14	UART 状态寄存器
UARTx_RE	0x18	UART DMA 请求使能寄存器
UARTx_IE	0x1C	UART 中断使能寄存器
UARTx_IF	0x20	UART 中断标志寄存器
UARTx_IOC	0x24	UART IO 控制

17.3.2 UARTx_CTRL UARTx 控制寄存器 (x = 0,1)

地址分别是：0x4001_1000，0x4001_1100

复位值:0x0

表 17-4 UART 控制寄存器 UARTx_CTRL

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

位置	位名称	说明
[31:8]		未使用

[7]	DUPLEX	双工，默认值为 0。 0:全双工；1:半双工
[6]	LIN_EN	LIN 模式使能，默认值为 0。 0:关闭；1:开启
[5]	MD_EN	使能 Multi-drop，默认值为 0。 0:关闭；1:开启
[4]	CK_EN	数据校验开关，默认值为 0。 0:关闭；1:开启
[3]	CK_TYPE	奇偶校验配置，默认值为 0。 0:偶校验（EVEN）；1: 奇校验（ODD）
[2]	BIT_ORDER	数据发送顺序配置，默认值为 0。 0:LSB；1:MSB
[1]	STOP_LEN	停止位长度配置，默认值为 0。 0:1-Bit；1:2-Bit
[0]	BYTE_LEN	数据长度配置，默认值为 0。 0:8-Bit；1:9-Bit

17.3.3 UARTx_DIVH UARTx 波特率设置高字节寄存器 (x = 0,1)

地址分别是：0x4001_1004，0x4001_1104

复位值:0x0

表 17-5 UART 波特率设置高字节寄存器 UARTx_DIVH

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

位置	位名称	说明
[31:8]		未使用
[7:0]	DIVH	波特率设置高字节 BAUDRATE =主时钟/(1+256* UART_DIVH+UART_DIVL)

17.3.4 UARTx_DIVL UARTx 波特率设置低字节寄存器 (x = 0,1)

地址分别是：0x4001_1008，0x4001_1108

复位值:0x0

表 17-6 UART 波特率设置低字节寄存器 UART_DIVL

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

位置	位名称	说明
[31:8]		未使用
[7:0]	DIVL	波特率设置低字节 BAUDRATE =主时钟/(1+256* UART_DIVH+UART_DIVL)

17.3.5 UARTx_BUFF UARTx 收发缓冲寄存器 (x = 0,1)

地址分别是：0x4001_100C，0x4001_110C

表 17-7 UART 收发缓冲寄存器 UART_BUFF

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

位置	位名称	说明
[31:9]		未使用
[8:0]	BUFF	写:发送数据缓存；读:接收数据寄存器

UART 的 Tx_buffer 和 Rx_buffer 共享 UART_BUFF。其中，Tx_buffer 是只写的，Rx_buffer 是只读的。因此，读 UART_BUFF 寄存器是访问 UART_RX_BUFF，写 UART_BUFF 寄存器是访问 UART_TX_BUFF。

当 UARTx_CTRL.BYTE_LEN=0 时，UART 收发只使用 buffer 低 8 位，即 UARTx_BUFF[7:0]；

当 UARTx_CTRL.BYTE_LEN=1 时，UART 收发使用 buffer 全部 9 位，即 UARTx_BUFF[8:0]。

如果使能 UARTx_CTRL.CK_EN，即使用校验位，则 UARTx_BUFF 的最高位（BIT8 when UARTx_CTRL.BYTE_LEN=1 or BIT7 UARTx_CTRL.BYTE_LEN=0）数据无效，发送时会被替换为校验位，接收时会作为校验位，而不会写入 UARTx_BUFF。

17.3.6 UARTx_ADR UARTx 地址匹配寄存器 (x = 0,1)

地址分别是：0x4001_1010，0x4001_1110

复位值:0x0

表 17-8 UART 地址匹配寄存器 UART_ADR

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

位置	位名称	说明
[31:8]		未使用
[7:0]	ADR	多机通讯时的从机地址

17.3.7 UARTx_STT UARTx 状态寄存器 (x = 0,1)

地址分别是：0x4001_1014，0x4001_1114

复位值:0x0

表 17-9 UART 状态寄存器 UART_STT

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

位置	位名称	说明
[31:4]		未使用
[3]	RX_BUSY	1:UART 已检测到起始符并正处于接收状态 0:UART 接收端空闲 此状态位只读，置位与清零均由硬件完成
[2]	ADR_MATCH	Multi-drop 模式下，地址匹配标志位。 1:匹配；0:未匹配。
[1]	TX_DONE	发送完成标志位。 1:完成；0:未完成。
[0]	TX_BUF_EMPTY	发送缓存状态位。 1:空；0:非空。

17.3.8 UARTx_RE UARTx DMA 请求使能寄存器 (x = 0,1)

地址分别是：0x4001_1018，0x4001_1118

复位值:0x0

表 17-10 UART DMA 请求使能寄存器 UART_RE

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

位置	位名称	说明
[31:3]		未使用
[2]	TX_BUF_EMPTY	发送缓冲区空 DMA 请求开关，默认值为 0。 0:关闭；1:开启。
[1]	RX_DONE	接收完成 DMA 请求开关，默认值为 0。 0:关闭；1:开启。
[0]	TX_DONE	发送完成 DMA 请求开关，默认值为 0。 0:关闭；1:开启。

17.3.9 UARTx_IE UARTx 中断使能寄存器 (x = 0,1)

地址分别是：0x4001_101C，0x4001_111C

复位值:0x0

表 17-11 UART 中断使能寄存器 UART_IE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							IDLE	LBD	RX_OV	TX_OV	CK_ERR	STOP_ERR	TX_BUF_EMPTY	RX_DONE	TX_DONE
							RW	RW	RW	RW	RW	RW	RW	RW	RW
							0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:9]		未使用
[8]	IDLE	空闲帧中断使能 0:关闭；1:开启
[7]	LBD	LIN break character 检测中断使能 0:关闭；1:开启
[6]	RX_OV	接收缓冲区溢出中断使能 0:关闭；1:开启
[5]	TX_OV	发送缓冲区溢出中断使能 0:关闭；1:开启
[4]	CK_ERR	校验错误中断开关，默认值为 0。 0:关闭；1:开启
[3]	STOP_ERR	停止位错误中断开关，默认值为 0。 0:关闭；1:开启
[2]	TX_BUF_EMPTY	发送缓冲区空中断开关，默认值为 0。 0:关闭；1:开启
[1]	RX_DONE	接收完成中断开关，默认值为 0。 0:关闭；1:开启
[0]	TX_DONE	发送完成中断开关，默认值为 0。

		0:关闭；1:开启

17.3.10 UARTx_IF UARTx 中断标志寄存器 (x = 0,1)

地址分别是：0x4001_1020，0x4001_1120

复位值:0x0

表 17-12 UART 中断使能寄存器 UART_IF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							IDLE	LBD	RX_OV	TX_OV	CK_ERR	STOP_ERR	TX_BUF_EMPTY	RX_DONE	TX_DONE
							RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C
							0	0	0	0	0	0	1	0	1

位置	位名称	说明
[31:9]		未使用
[8]	IDLE	空闲帧中断标志，高有效，写 1 清零 当 UART 检测到空闲帧后，硬件置位这一标志，如果 UARTx_IE.IDLE_IE=1，则产生中断请求；写 1 清零后，只有待 RX_DONE_IF 标志再次置位后，IDLE_IF 标志才会置位。在多从机（Multi-drop）场景中，只有从机地址命中时，即 UARTx_STT.ADR_MATCH=1 时，表示主机此时欲与本从机通讯，IDLE_IF 才会置位，否则即使 UARTx_RX 信号为常高，IDLE_IF 也不会置位。
[7]	LBD	LIN break character 检测中断标志，高有效，写 1 清零
[6]	RX_OV	接收缓冲区溢出中断标志，高有效，写 1 清零
[5]	TX_OV	发送缓冲区溢出中断标志，高有效，写 1 清零
[4]	CK_ERR	校验错误中断标志，高有效，写 1 清零
[3]	STOP_ERR	停止位错误中断标志，高有效，写 1 清零
[2]	TX_BUF_EMPTY	发送缓冲区空中断标志，高有效，写 1 清零
[1]	RX_DONE	接收完成中断标志，高有效，写 1 清零
[0]	TX_DONE	发送完成中断标志，高有效，写 1 清零

中断标志写 1 清零，一般不建议用如下|=方式清零，因为|=是先读取中断标志，将对应位改为 1 再写入清零，如果同时有其他中断标志置位，会被一起清零，而这通常不是软件所期望的。例如，如下写法本意是清零 TX_DONE_IF，但如果同时 RX_DONE_IF 在写入执行前置 1 了，则软件先读取回 UART_IF 值为 0x2，然后执行或操作 0x2|0x1=0x3，然后写入，同时对 RX_DONE_IF 和 TX_DONE_IF 进行了清零，可能导致 UART 少进入一次因接收数据产生的中断，从而少接收到一字节数据。

UART_IF|=0x1;

如果希望清零 TX_DONE_IF 标志位，应以如下方式，直接对 BIT0 写 1.

```
UART_IF=0x1;
```

17.3.11 UARTx_IOC UARTx IO 控制寄存器 (x = 0,1)

地址分别是：0x4001_1024，0x4001_1124

复位值:0x0

表 17-13 UARTIO 控制寄存器 UART_IOC

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

位置	位名称	说明
[31:8]		未使用
[7]	SBK	LIN 模式下，写 1 发送一次 break character，即连续 13 个 0，完成后自动清零，未完成时读回为 1，向此位写 1 之前需要先配置 UARTx_CTRL.LIN_EN=1，否则无法进行 break character 的发送
[6]	LBDL	LIN break character 检测长度，10/11 个 0 0:10bits，1:11bits
[5]		未使用
[4]	AUTO	波特率自适应 IO 端口使能开关。 0:关闭；1:开启
[3:2]		未使用
[1]	TXD_INV	TXD 输出极性使能开关，默认值为 0。 0:正常输出；1:取反输出。 正常输出极性，即软件发送 1，硬件发送 1；取反输出极性，即应用发送 1，硬件发送 0。
[0]	RXD_INV	RXD 输入极性使能开关，默认值为 0。 0:正常输入；1:取反输入。 正常输入极性，即硬件接收到 1，软件接收的是 1；取反输入极性，即硬件接收 1，软件接收的是 0。

18 DMA

18.1 概述

DMA 和 CPU 同为芯片总线的主设备。

如图 18-1 所示。其中部分设备不需要被 DMA 访问，仅仅挂载于与 CPU 相连的总线上。ADC/DAC/SPI/I2C/MCPWM/UART/Timer/GPIO/Hall/SRAM 等设备可以被 CPU 和 DMA 共享访问。

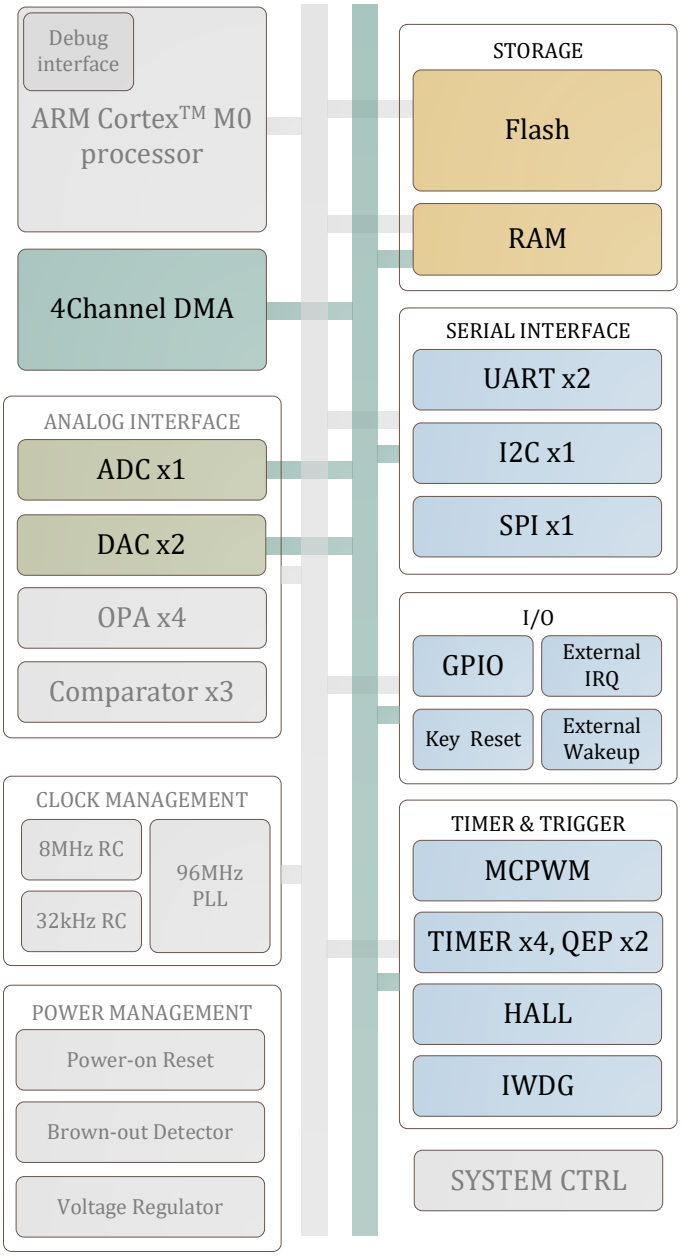


图 18-1 multi-layer AHB 总线架构

DMA 有 4 个通道，共享一个搬运引擎。当 DMA 空闲且多个通道同时发生请求时，按优先级进

行仲裁，但在搬运过程中不会发生抢占，即一定是某个通道完成搬运，DMA 空闲后，才会发起仲裁，判断下一个获得请求的通道是哪一个。

DMA 的传输有两种方式：DMA_CCRx.RMODE=1，多个轮次，每轮传输内传输一次；DMA_CCRx.RMODE=0，一轮，连续传输多次。

如果配置一轮传输多次数据，即 DMA_CCRx.RMODE=0，则一次 DMA 请求，DMA 连续发送 DMA_CTMSx 次数据，然后硬件将 DMA_CCRx.EN 位清零，置位中断标志。

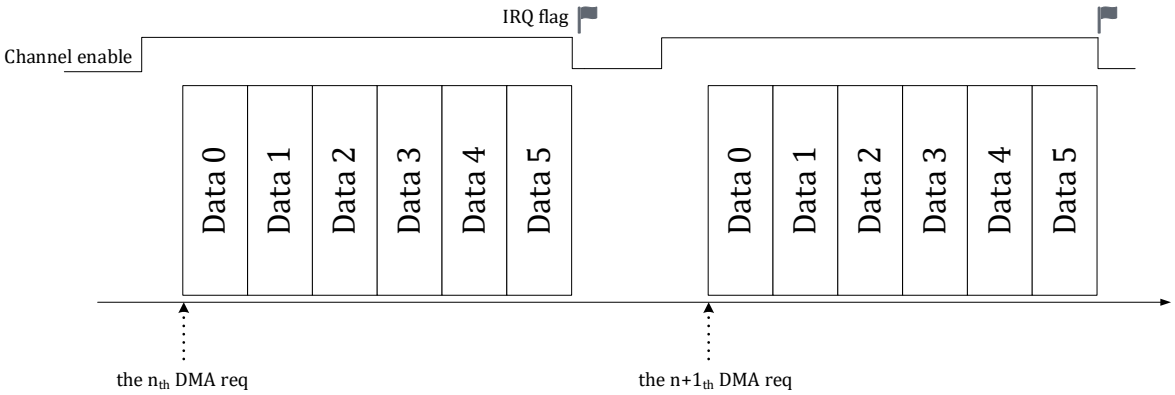


图 18-2 RMODE=0 DMA 传输情况

如果配置某个通道为多轮搬运，则在多轮搬运之间，DMA 可能会响应其他通道的搬运请求。

多轮传输每轮只搬运一次数据，搬运 DMA_CTMSx 轮后，硬件将 DMA_CCRx.EN 位清零，置位中断标志。

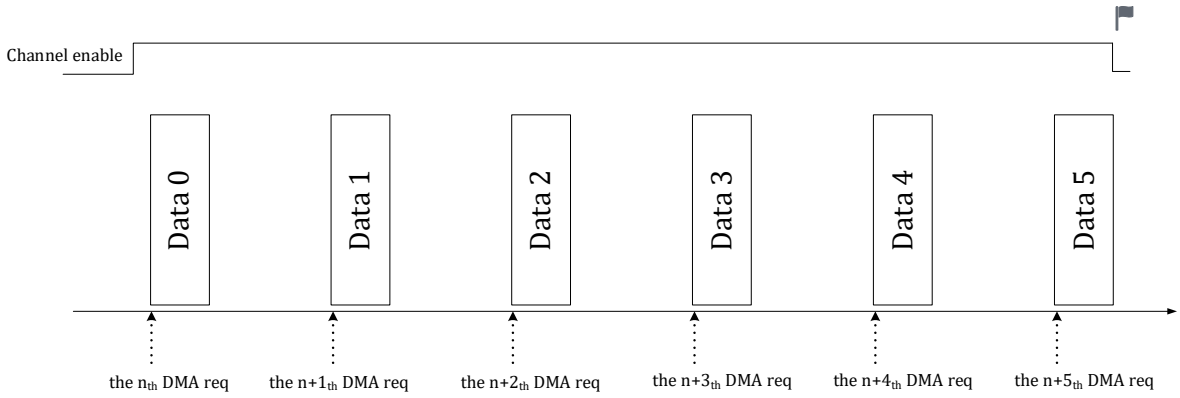


图 18-3 RMODE=1 DMA 传输情况

每发生一次 DMA 请求，DMA 开始一轮传输，完成一轮传输后，根据当前 DMA 通道的请求情况响应下一次请求，下一次请求可以仍为当前通道请求，也可以是其他通道请求，或者在无请求的情况下进入空闲状态，同时当前 DMA 通道待传输轮数减 1。当 DMA 待传输轮数为 0 时，当前通道的传输全部完成，DMA 产生中断，同时 DMA 通道使能 DMA_CCRx.EN 被硬件自动清零。在传输过程中，待传输轮数可以通过 DMA_CTMSx 寄存器读取。一轮传输内待传输次数不支持读取，因为 DMA 会连续进行多次传输，因此次数值会快速变化。

出于功耗控制考虑，DMA 模块可以通过设置 DMA_CTRL.EN 位为 0 来禁用（要求关闭 DMA 使能前先关闭通道对应的使能 DMA_CCRx.EN），此时 DMA 时钟被门控关闭。

DMA 支持 8, 16 或 32 bit (byte, half-word, or word) 三种位宽的传输操作，通过配置 DMA_CCRx.SBTW 和 DMA_CCRx.DBTW 来选择源地址和目的地址访问的位宽，源地址访问的位宽与目的地址访问的位宽可以不同。

DMA 每完成一次传输，地址根据 DMA_CCRx.SINC 和 DMA_CCRx.DINC 决定是否自动递增。所有的外设寄存器地址都是 word 对齐的，所以外设的地址递增应配置为 0/4。例如对于 UART/SPI/I2C，因为每次都是访问 UART_DATA 或 SPI/I2C FIFO 接口的固定地址，所以轮次间无需地址递增；如要访问 ADC 数据寄存器，则地址通常需要自动加 4，可以设置轮内递增。而对于内存，如果设置了轮内递增，每次地址递增的值，根据内存数据位宽(DMA_CCRx.SBTW/DBTW)设定，对于内存访问位宽为 byte 的情况，地址自动加 1，对于 half-word，地址自动加 2，对于 word，地址自动加 4。

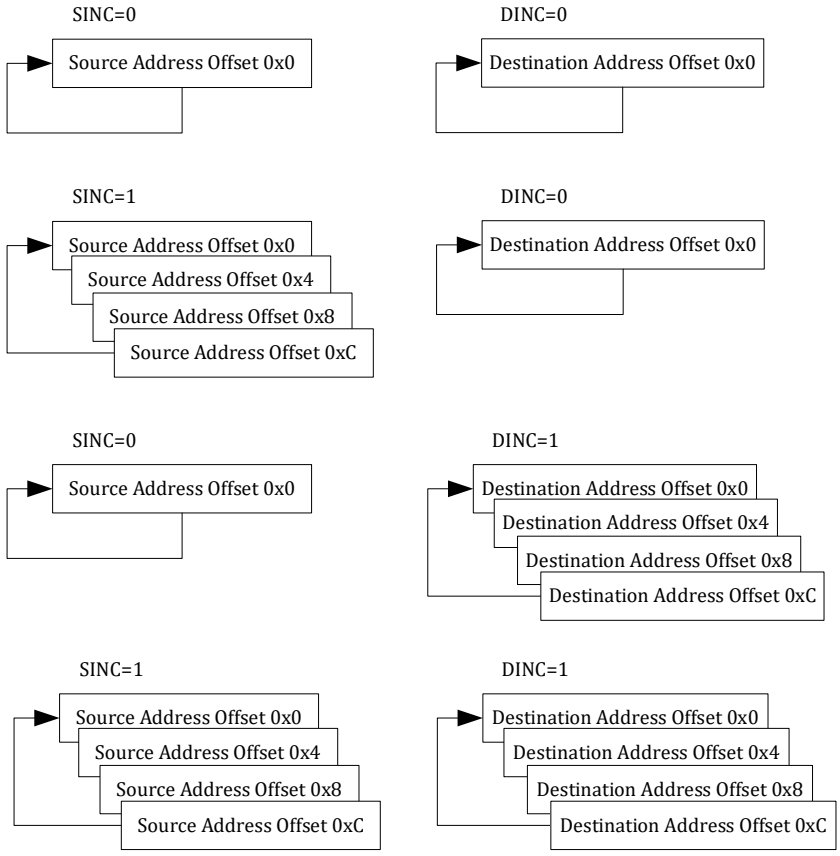


图 18-4 DMA 地址递增控制

图 18-4 仅为地址递增示例，配置为一轮传输 4 次，或传输 4 轮，源地址和目的地址数据尺寸均为 word。实际应用中，如果配置数据尺寸为 byte 或 halfword，则地址偏移按照 1/2 进行递增。

一般，SINC=0，DINC=0，源地址、目的地址均不递增，可能为外设到外设的搬运。

SINC=1，DINC=0，源地址递增，目的地址不递增，可能为内存数组到外设寄存器接口的搬运。

SINC=0，DINC=1，源地址不递增，目的地址递增，可能为外设寄存器接口到内存数组的搬运。

SINC=1，DINC=1，源地址、目的地址均递增，为外设多组数据(如 ADC_DATx)到内存数组的搬运。

如果配置 DMA_CCRx.CIRC=1，且 DMA_CCRx.RMODE=1，即循环多轮模式。一次 DMA 请求信

号进行触发，只进行一次数据搬运（如 UART 数据搬运至 RAM，完成一轮搬运等待下一次请求。在此模式下，DMA_CTMSx 可以配置为一个较大的值，每进行一次搬运 DMA_CTMSx 减 1。由于循环模式 DMA 进行多轮搬运不再产生中断标志，CPU 可以通过读取 DMA_CTMSx 来判断当前已经传输的数据量，如果已经足够则开始处理。当传输了 DMA_CTMSx 轮以后，地址会回到初始地址。DMA_CTMSx 会重新装载为预设值，开始新一轮的递减。循环多轮模式下，DMA 的通道使能 DMA_CCRx.EN 位由软件置位，一直为高，即使 DMA_CTMSx 递减至 0。如果要退出传输，可以软件清零 DMA_CCRx.EN。

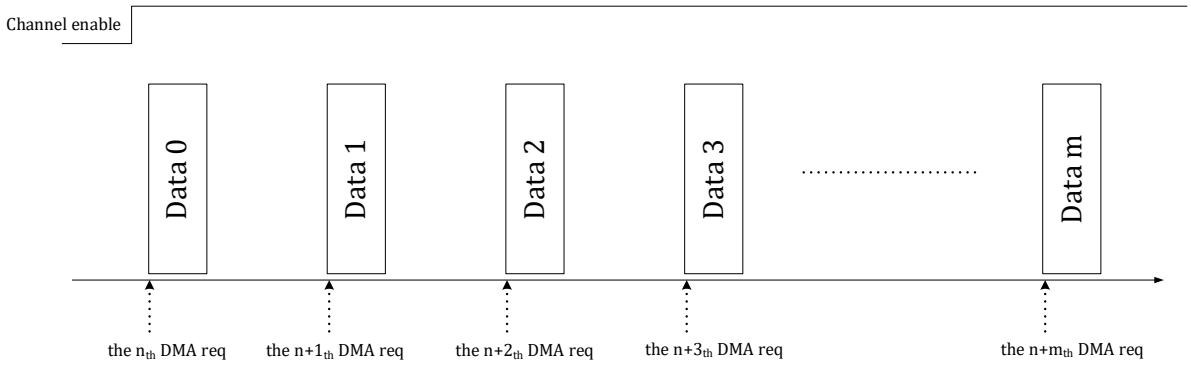


图 18-5 CIRC=1 且 RMODE=1 DMA 传输情况

如果配置 DMA_CCRx.CIRC=1，且 DMA_CCRx.RMODE=0，即循环单轮模式则 DMA 收到一次请求后会连续地进行 DMA_CTMSx 次数据搬运，并产生中断标志。在此模式下，DMA_CCRx.EN 无须软件置位，当 DMA 通道收到请求时会自动置位，传输完成一轮后自动清零。如果要退出此种模式，可以清零 DMA_CCRx.CIRC 位，并清零 DMA_CCRx.EN 位。

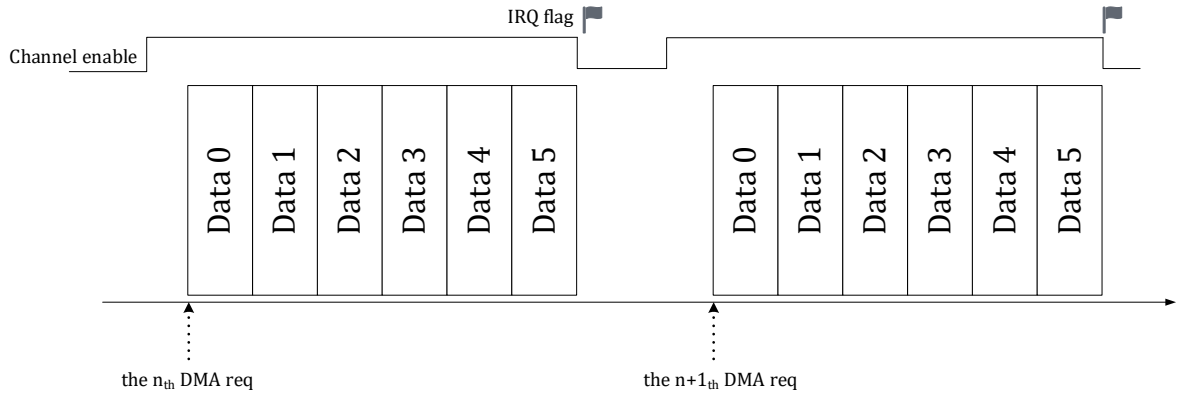


图 18-6 CIRC=1 且 RMODE=0 DMA 传输情况

循环模式下，DMA_CTMSx 寄存器随着搬运递减为 0 后重新装载为初始配置值。如果是搬运数据到内存，则覆盖之前搬运至内存的数据；如果是搬运至外设，则重复一轮数据发射。以将 ADC 数据搬运至内存为例，设定数据块大小为 16bit×12channel×8 次，则在循环模式下 DMA 完成一轮 96 个 half word 搬运后重新开始下一轮搬运，并覆盖之前使用的内存地址，不设置 DMA 完成中断标志位。单次模式下，DMA 完成一定大小的数据块搬运后即终止本次 DMA 操作，设置当前通道 DMA 完成中断标志位，同时硬件自动关闭对应的 DMA 通道，即硬件电路自动在该通道传输完成后将 DMA_CCRx.EN 设为 0。DMA 需要搬运的轮次由 DMA_CTMS 寄存器进行控制。

18.2 请求

DMA 的请求分为软件请求和硬件请求两类，软件请求通过设置对应 DMA 通道的 DMA_RENx.SW_TRIG=1 来产生，写 1 即产生，软件触发位设置后需要软件清零。硬件请求通常为外设的中断事件，当特定的外设中断事件作为 DMA 传输请求时，通常要禁用对应事件的中断响应，即不再需要 CPU 对中断进行响应，响应仅作为 DMA 传输请求。而且硬件 DMA 请求信号经过 DMA 通道受理后会被 DMA 硬件清零，无需软件清除提起请求的外设中断标志。

表 18-1 DMA 请求

触发来源	描述
软件	软件触发时进行的搬运操作由配置寄存器 DMA_CCRx 指定，当设置了 DMA_RENx.SW_TRIG，一旦通道被使能即开始进行 DMA 操作。
ADC	ADC 在单段触发模式下，一次完成若干个通道后采样后产生中断请求，由 DMA 把 ADC 转换后的值搬运到 SRAM。ADC 单段采样完成中断事件作为 DMA 请求信号，请求信号在得到 DMA 响应后由 DMA 将其清零，无需软件清零；注意此时需要软件同时禁用 ADC 采样完成中断，使中断不再被 CPU 响应。
DSP	DSP IRQ 指令产生的中断事件标志可以作为 DMA 请求
Timer	Timer 可以使用过零/比较/捕获事件作为 DMA 请求，具体 DMA 操作根据配置寄存器设定，通常作为定时事件（如每隔 10ms 触发一次 DMA 操作）
SPI	SPI 模块可以使用接收缓冲区满事件作为 DMA 请求信号，由于 SPI 是同时收发，所以接收缓冲区满既是接收完毕也是发送完毕的事件标志。读取 SPI FIFO 自动清除事件标志。
MCPWM	MCPWM 模块可以使用过零/计数周期结束/4 个 ADC 触发信号作为 DMA 请求，具体 DMA 操作根据配置寄存器设定
I2C	I2C 模块可以使用 I2C0_SCR.BYTE_CMPLT 即字节发送完成作为触发 DMA 请求，DMA 自动清除 I2C 的请求标志。其他 I2C 中断事件仍由 CPU 响应
UART	串口模块可以使用 UART_IF 触发 DMA 请求，如果 DMA 配置的传输方向是内存到串口，则使用 UART 发送缓冲区空标志作为 DMA 请求信号；如果传输方向是串口到内存，则应使用 UART 接收完成事件产生 DMA 请求信号。事件标志由 DMA 自动清零。UART 在由 DMA 操作时应同样禁用相应中断防止被 CPU 响应。
HALL	HALL 中断标志可以作为 DMA 请求
SIF	SIF 中断标志可以作为 DMA 请求*
比较器	CMP 中断标志可以作为 DMA 请求
GPIO	GPIO 中断标志可以作为 DMA 请求

*SYS_AFE_INFO.VERSION=4 版本的芯片，此功能不可用。

18.3 优先级

DMA 的优先级采用固定优先级，优先级如图 18-7 所示。为避免出现来不及响应某些外设请求的情况，在设计应用软件时应考虑任务实时性，每个通道不配置搬运过于大量的数据导致其他通道得不到及时响应。

如图 18-7 所示，优先级由上至下降低。4 个 DMA 通道中，优先级关系为:通道 0>通道 1>通道 2>通道 3 (>号表示优先级高于)。在 DMA 各通道内部，通常有多个硬件请求事件和一个软件请求事

件，硬件请求优先级高于软件请求。多个硬件请求事件优先级相同。通常，一个 DMA 通道配置一个硬件请求事件使能，多个硬件请求不应在一个通道内同时发生。

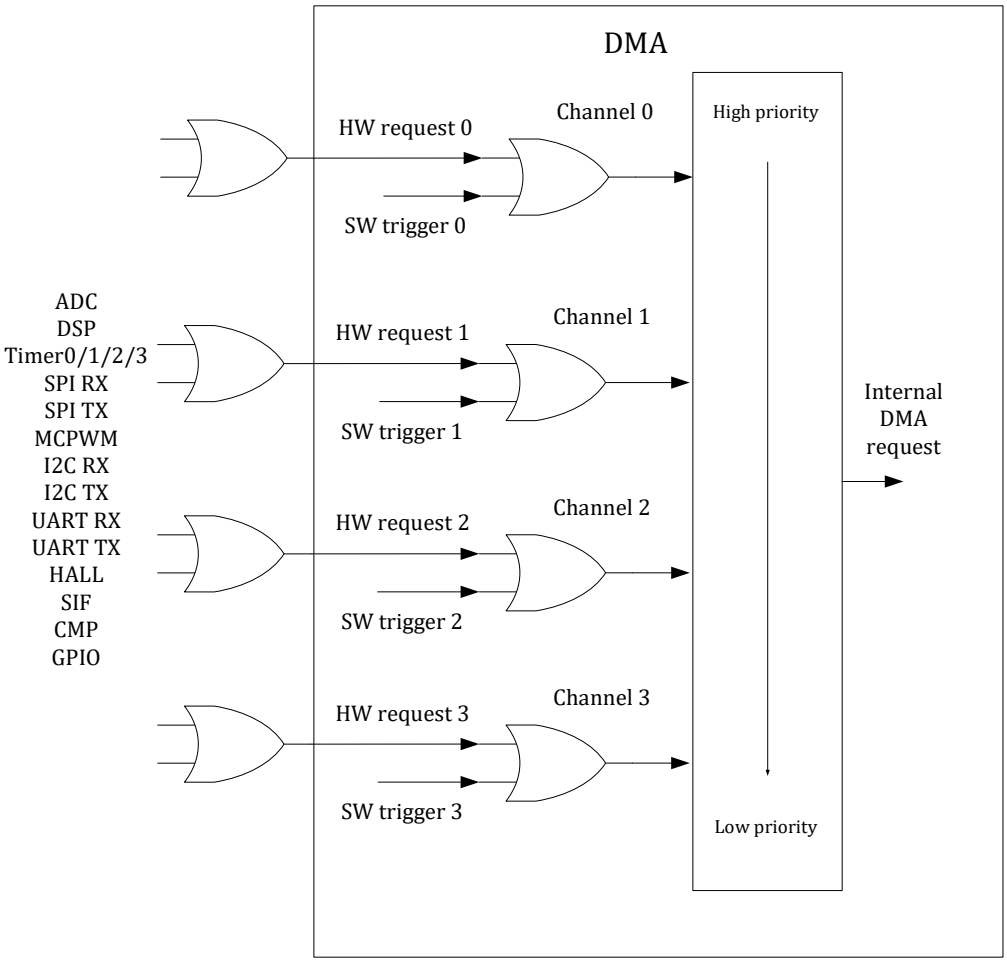


图 18-7 DMA 通道优先级

18.4 仲裁

当 DMA 处于空闲状态，或刚刚完成某一通道的 DMA 传输后，若此时恰好有一个或多个 DMA 请求发生，DMA 会根据优先级设定进行仲裁，优先级高的外设请求率先得到 DMA 服务。比如 ADC 连续模式下，每完成一轮 ADC 数据搬运，ADC 的采样完成事件标志被 DMA 清零，DMA 回到空闲状态或转而服务其他外设请求；对于 UART/SPI/I2C，每搬运一个 byte 重新仲裁。

为了避免 CPU 或 DMA 长期占用外设/SRAM，在外设/SRAM 的端口仲裁模块中加入了时间片机制，即一个主设备占用一段时间后释放访问权，仲裁模块观测另一个主设备是否在请求访问，如果是则转而允许另一个主设备访问，否则继续当前主设备未完成的访问。

18.5 中断

DMA 的一个通道完成 DMA 操作后产生 DMA 中断。当 DMA 某一个通道完成操作后，会自动关闭该通道的使能，通过 DMA_CCRx.EN。

18.6 寄存器

18.6.1 地址分配

DMA 控制器模块寄存器的基地址是 0x4001_1200，寄存器列表如下:

表 18-2 DMA 寄存器列表

名称	偏移地址	说明
DMA0_CCR0	0x00	DMA 通道 0 通道配置寄存器
DMA0_REN0	0x04	DMA 通道 0 请求使能寄存器
DMA0_CTMS0	0x08	DMA 通道 0 传输次数寄存器
DMA0_SADR0	0x0C	DMA 通道 0 源地址寄存器
DMA0_DADR0	0x10	DMA 通道 0 目的地址寄存器
DMA0_CCR1	0x20	DMA 通道 1 通道配置寄存器
DMA0_REN1	0x24	DMA 通道 1 请求使能寄存器
DMA0_CTMS1	0x28	DMA 通道 1 传输次数寄存器
DMA0_SADR1	0x2C	DMA 通道 1 源地址寄存器
DMA0_DADR1	0x30	DMA 通道 1 目的地址寄存器
DMA0_CCR2	0x40	DMA 通道 2 通道配置寄存器
DMA0_REN2	0x44	DMA 通道 2 请求使能寄存器
DMA0_CTMS2	0x48	DMA 通道 2 传输次数寄存器
DMA0_SADR2	0x4C	DMA 通道 2 源地址寄存器
DMA0_DADR2	0x50	DMA 通道 2 目的地址寄存器
DMA0_CCR3	0x60	DMA 通道 3 通道配置寄存器
DMA0_REN3	0x64	DMA 通道 3 请求使能寄存器
DMA0_CTMS3	0x68	DMA 通道 3 传输次数寄存器
DMA0_SADR3	0x6C	DMA 通道 3 源地址寄存器
DMA0_DADR3	0x70	DMA 通道 3 目的地址寄存器
DMA0_CTRL	0x80	DMA 控制寄存器
DMA0_IE	0x84	DMA 中断使能寄存器
DMA0_IF	0x88	DMA 中断标志寄存器

18.6.2 DMA0_CTRL DMA 控制寄存器

地址:0x4001_1280

复位值:0x0

表 18-3 DMA 控制寄存器 DMA0_CTRL

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

	RW
	0

位置	位名称	说明
[31:1]		未使用
[0]	EN	DMA 整体使能

18.6.3 DMA0_IE DMA 中断使能寄存器

地址:0x4001_1284

复位值:0x0

表 18-4 DMA 中断使能寄存器 DMA0_IE

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

位置	位名称	说明
[31:4]		未使用
[3]	CH3_FIE	通道 3 完成中断使能
[2]	CH2_FIE	通道 2 完成中断使能
[1]	CH1_FIE	通道 1 完成中断使能
[0]	CH0_FIE	通道 0 完成中断使能

18.6.4 DMA0_IF DMA 中断标志寄存器

地址:0x4001_1288

复位值:0x0

表 18-5 DMA 中断标志寄存器 DMA0_IF

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

位置	位名称	说明
[31:4]		未使用
[3]	CH3_FIF	通道 3 完成中断标志，高有效，写 1 清零
[2]	CH2_FIF	通道 2 完成中断标志，高有效，写 1 清零
[1]	CH1_FIF	通道 1 完成中断标志，高有效，写 1 清零
[0]	CH0_FIF	通道 0 完成中断标志，高有效，写 1 清零

18.6.5 DMA 通道配置寄存器

18.6.5.1 DMA0_CCRx (where x =0,1,2,3)

地址分别是:0x4001_1200，0x4001_1220，0x4001_1240，0x4001_1260

复位值:0x0

表 18-6 DMA 通道配置寄存器 DMA0_CCRx

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

位置	位名称	说明
[31:12]		未使用
[11:10]	SBTW	源地址访问位宽 0: Byte 1: Halfword 2: Word 3: 保留
[9:8]	DBTW	目的地址访问位宽 0: Byte 1: Halfword 2: Word 3: 保留
[7]		未使用
[6]	SINC	源地址递增方式 0:不递增 1:每传输一次，地址按照 SBTW 对应大小递增 1/2/4
[5]		未使用
[4]	DINC	目的地址递增方式 0:不递增 1:每传输一次，地址按照 DBTW 对应大小递增 1/2/4
[3]	CIRC	循环模式，高有效
[2]		未使用

[1]	RMODE	0:单轮传输，一轮连续传输多次，每收到 DMA 请求传输一轮，一个 DMA 请求即传输完毕 1:多轮，每轮进行一次数据传输，每收到 DMA 请求传输一轮，多个 DMA 请求才传输完毕 不支持多轮×多次传输
[0]	EN	通道使能，高有效，软件置 1 开启通道进行 DMA 搬运操作，搬运完成后 DMA 硬件将此位清零

18.6.5.2 DMA0_RENx (where x = 0,1,2,3)

地址分别是:0x4001_1204，0x4001_1224，0x4001_1244，0x4001_1264

复位值:0x0

表 18-7 DMA 请求使能寄存器 DMA0_ RENx

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SW_REN		GPIO_REN			CMP_REN	SIF_REN*	HALL_REN					UART1_TX_REN	UART1_RX_REN	UART0_TX_REN	UART0_RX_REN
RW		RW			RW	RW	RW					RW	RW	RW	RW
0		0			0	0	0					0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I2C0_TX_REN	I2C0_RX_REN	MCPWM1 REN	MCPWM0_REN	SPI TX_REN	SPI RX_REN			TIMER3_REN	TIMER2_REN	TIMER1_REN	TIMER0_REN	DSP_REN		ADC1_REN	ADC0_REN
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

* SYS_AFE_INFO.VERSION<=3 版本的芯片，SIF_REN 可用；SYS_AFE_INFO.VERSION=4 版本的芯片，SIF_REN 不可用。

一般，DMA 的通道中配置的内存、外设地址应与使能的外设中断请求相对应，这一点需要由应用软件保证。其中软件请求始终处于使能状态，即软件向 DMA_RENx.SW_TRIG 写 1 即开始一次 DMA 传输。来自外设的硬件请求进入 DMA 后通过 OR 逻辑形成一个请求信号，每一个 DMA 通道一般在同一时间只使能一个硬件 DMA 请求。软件触发标志 DMA_RENx.SW_TRIG 在请求被 DMA 受理后由硬件自动清除。

由于 CMP 信号可能是一个慢速的电平信号，在使用 CMP 触发 DMA 时，推荐将中断触发类型选择为边沿触发。

DMA_RENx 寄存器可以在 DMA 通道使能的情况下改写。

18.6.5.3 DMA0_CTMSx (where x = 0,1,2,3)

地址分别是:0x4001_1208，0x4001_1228，0x4001_1248，0x4001_1268

复位值:0x0

表 18-8 DMA 传输次数寄存器 DMA0_CTMSx

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

位置	位名称	说明
[7:0]	TMS	DMA 通道 x 数据搬运次数。此寄存器在该通道使能后变为只读。

DMA_CTMSx 寄存器只有在通道禁用，即 DMA_CCRx.EN=0 之后才可以写入数据。

当 DMA_CTRL=1 且 DMA_CCRx.EN=0 时，重新填写 CTMSx 值，可以将 DMA 内部已搬运的轮数次数计数器清零。

当 DMA_CCRx.RMODE=0 时，表示传输 1 轮，每轮传输 DMA_CTMS 次数据；

当 DMA_CCRx.RMODE=1 时，表示传输 DMA_CTMS 轮，每轮传输一次数据；

不支持多轮每轮传输多次数据。

以外设数据宽度为 16，内存数据宽度为 32，即 DMA_CTMSx=16，DMA_CCRx.RMODE=0 为例，DMA 每轮需要读取外设数据 16bit=2byte，写入内存数据 32bit=4byte。一共需要搬运 16 轮，即读取外设 32byte，存入内存 64byte；

即使仅仅搬运一轮，也需要设置 CTMS.ROUND =1，而不能让其为 0。

当设置 DMA_CCRx.CIRC=1（即循环模式）时，DMA_CTMSx 不再起作用，相当于无限轮；其他情况需要相应设置 DMA_CTMSx，如 DMA_CTMSx=1，且 DMA_CCRx.RMODE=1 时，用于搬运一轮数据。

当 DMA_CCRx.RMODE=1 时，DMA_CTMSx 读出为当前剩余未搬运轮数，当 DMA 传输完成后轮数会复位为配置值。次数读出时一直为配置值。举例来说，比如配置 DMA_CTMSx=4，读取时可能看到 DMA_CTMSx 由 4 开始递减，3,2,1...之后又变为 0。当 DMA 当前通道需要重新进行 4 轮传输时，需要重新向 DMA_CTMSx 写入轮数值。当 DMA_CCRx.RMODE=0 时，DMA_CTMSx 读出为当前这一轮剩余未搬运次数。但由于一轮内的数据是被 DMA 连续搬运的，所以软件读出的 DMA_CTMSx 会迅速变化。

注意，每次重新开启 DMA 通道前，都要重新配置 DMA_CTMSx 寄存器，已经在上一次传输中 DMA_CTMSx 寄存器已经递减为 0。

18.6.5.4 DMA0_SADRx (where x = 0,1,2,3)

地址分别是:0x4001_120C，0x4001_122C，0x4001_124C，0x4001_126C

复位值:0x0

表 18-9 DMA 源地址寄存器 DMA0_ SADRx

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

0

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

位置	位名称	说明
[31:0]	ADDR	DMA 通道 x 源地址

当 DMA_CCRx.SBTW=2'b01 时，即配置为以 16bit 为单位搬运源数据。DMA_SADRx.ADDR[0]值无效，外设地址会以 2 为单位递增。

当 DMA_CCRx.SBTW=2'b10 时，即配置为以 32bit 为单位搬运外设数据。DMA_SADRx.ADDR[1:0]值无效，外设地址会以 4 为单位递增。

注意:DMA_SADRx 寄存器只有在通道禁用，即 DMA_CCRx.EN=0 之后才可以写入数据！！！

18.6.5.5 DMA0_DADRx (where x = 0,1,2,3)

地址分别是:0x4001_1210，0x4001_1230，0x4001_1250，0x4001_1270

复位值:0x0

表 18-10 DMA 目的地址寄存器 DMA0_DADRx

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

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

位置	位名称	说明
[31:0]	ADDR	DMA 通道 x 目的地址

当 DMA_CCRx.DBTW=2'b01 时，即配置为以 16bit 为单位搬运内存数据。DADRx. ADDR[0]值无效，内存地址会以 2 为单位递增。

当 DMA_CCRx.DBTW=2'b10 时，即配置为以 32bit 为单位搬运内存数据。DADRx. ADDR[1:0]值无效，内存地址会以 4 为单位递增。

注意:DMA_DADRx 寄存器只有在通道禁用，即 DMA_CCRx.EN=0 之后才可以写入数据！！！

19 DSP

19.1 概述

DSP 模块使用自主设计的 DSP 指令集，可以进行加减、与或、乘累加、移位、饱和等单周期算术指令，以及除法、开方、三角函数等多周期算术运算指令；具备 load/store 等访存指令，无条件跳转以及条件跳转等分支指令，还有中断提起等杂项指令。有断点指令和寄存器赋值等伪指令可以在模拟器上用于调试。

DSP 有两种运行模式，自主运行、被动调用。

所谓自主运行即 DSP 读取 CODE MEM 中的指令和 DATA MEM 中的数据进行 DSP 程序执行，独立于 ARM Cortex M0，此时 DSP0_SC.PAUSED=0，即 DSP 处于运行状态；CODE MEM 和 DATA MEM 允许 DSP 访问但不允许 CPU 访问改写。

被动调用是指 DSP 作为一个外设模块被 ARM Cortex M0 所调用，CPU 直接访问 DSP 内部的算术运算资源如除法、开方、三角函数等。此时 DSP0_SC.PAUSED=1，即 DSP 不运行 DSP 程序，处于暂停状态，CODE MEM 和 DATA MEM 允许 CPU 进行访问改写，对于不进行 DSP 程序开发的用户，推荐使用此种模式，通过 CPU 运行的软件直接调用 DSP 的算术单元。

DSP 配备独立的程序存储器(CODE MEM)和数据存储器(DATA MEM)。在 DSP 暂停即 DSP0_SC.PAUSED=1 时可以通过 CPU 访问这两个独立的存储区域，在 DSP 初始化的过程中需要由 CPU 将 DSP 运行的程序和初始数据分别写入 DSP 的 CODE MEM 和 DATA MEM。DSP 具备提起中断的指令，中断置位后，DSP 同时进入暂停状态，此时允许 CPU 通过总线接口访问 DATA MEM 与 DSP 进行数据交互，包括读取 DSP 运算结果，以及写入 DSP 后续运行所需的数据等。

此外，为充分灵活利用 DSP，在 DSP 暂停时允许 CPU 通过 DSP 寄存器接口直接访问 DSP 除法器、开方器、三角函数等运算模块，即允许 CPU 将 DSP 当做简单的运算协处理模块使用。

19.1.1 功能框图

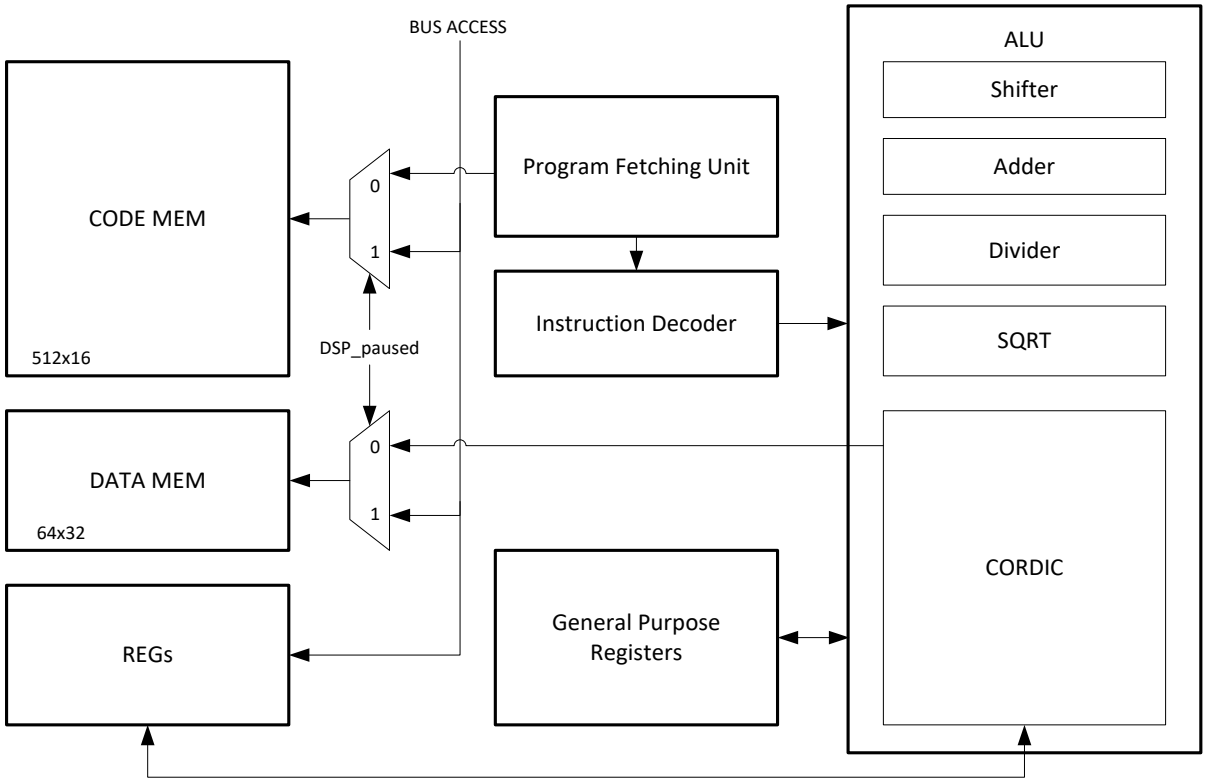


图 19-1 DSP 模块功能框图

19.1.2 DSP 核寄存器

表 19-1 DSP 核心寄存器

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

其中 R0 寄存器为常 0 寄存器，不能写入数据，读回恒为 0。R0 寄存器可作为特殊操作数构造一些伪指令。如

ADD R0 R0 R0 相当于 NOP

MAC R1 R2 R0 相当于 MUL R1 R2，即乘累加中累加的数为 0，变为乘法操作

R6 寄存器在 ARCTAN 指令中固定用于存放向量模值

R7 寄存器在 MAC 指令中固定用于存放计算结果，在 DIV 指令中固定用于存放被除数操作数。

以上约定是由于受定长指令编码限制，在 4 操作数指令中需要固定一个操作数使用约定寄存

器。

19.1.3 位宽

除法的被除数和商位宽均为 32 位有符号数，除数和余数为 32 位有符号数。

被开方数为 32 位无符号数，平方根为 16 位无符号数。

乘累加的两个乘数均为 32 位有符号数，加数和结果为 32 位有符号数。

三角函数 CORDIC 模块位宽为 16 位，Q15 定点数格式。

注意：在使用 DSP 算术运算资源时，无论是 CPU 调用还是通过 DSP 算术指令调用，请务必保证操作数不要超过表示范围，否则会出现计算异常。

除法除数为 0 时，无论被除数是任何值，商恒为 0，余数为被除数。

当被除数为 0x8000_0000，除数为-1 时，商超出了 32bit 有符号数的表示范围，饱和为 0x7fff_ffff。

表 19-2 除法特殊操作数情况表

被除数 DSP0_DID		除数 DSP0_DIS		商 DSP0_QUO		余数 DSP0_REM	
HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
0x7FFF_FFFF	2^{31}-1	0x8000_0000	-2^{31}	0x0	0	0x7FFF_FFFF	2^{31}-1
0x7FFF_FFFF	2^{31}-1	0x7FFF_FFFF	2^{31}-1	0x1	1	0x0	0
0x8000_0000	-2^{31}	0x7FFF_FFFF	2^{31}-1	0xFFFF_FFFF	-1	0xFFFF_FFFF	-1
0x8000_0000	-2^{31}	0x8000_0000	-2^{31}	0x1	1	0x0	0
0x8000_0000	-2^{31}	0xFFFF_FFFF	-1	0x7FFF_FFFF	2^{31}-1	0x0	0
0x7FFF_FFFF	2^{31}-1	0xFFFF_FFFF	-1	0x8000_0001	1-2^{31}	0x0	0
0x1234	4660	0x0	0	0x0	0	0x1234	4660
-0x1234	-4660	0x0	0	0x0	0	-0x1234	-4660

此外，由于 CORDIC 运算基于多次旋转逐次逼近，计算结果存在不超过 0.1%的计算误差。

19.1.4 指令周期

除法指令需要 12 个总线周期（96MHz）完成。

开方指令需要 8 个总线周期（96MHz）完成。

三角函数指令需要 20 个总线周期（96MHz）完成。

其余指令均为单周期指令

19.1.5 地址空间

表 19-3 DSP 地址空间

段	模块	CPU 软件视角地址空间	DSP 软件视角地址空间	实际存储体大小/说明
CODE	DSP Code Memory	0x4001_2000～0x4001_27FF	CODE 空间：0x0~0x1FF	512 x 16bit
DATA	保留	0x4001_2800 ~ 0x4001_2EFF		保留
	DSP Data Memory	0x4001_2F00 ~	DATA 空间：0x0~0x3B	60 x 32bit

		0x4001_2FEF		
	DSP0_PDI	0x4001_2FF0	DATA 空间：0x3C	GPIO_PDI 寄存器
	DSP0_BSRR	0x4001_2FF4	DATA 空间：0x3D	GPIO_PDO 置位寄存器
	DSP0_BRR/DSP0_OP	0x4001_2FF8	DATA 空间：0x3E	GPIO_PDO 清零寄存器/DSP 操作数寄存器
	DSP0_CL/DSP0_RES	0x4001_2FFC	DATA 空间：0x3F	CL_OUTPUT 寄存器/DSP 结果寄存器
REG	DSP 寄存器	0x4001_3000～0x4001_37FF		
		0x4001_3800～0x4001_3FFF		保留

DSP 的地址空间分为 4 段，分别为程序段、数据段、寄存器段、保留段，各段空间占用 2kB 的地址空间，但实际的 CODE MEM 和 DATA MEM 存储空间不足 2kB。程序段（CODE MEM）用于存放 DSP 运行所需程序代码，为单口 SRAM，单周期完成读写操作；数据段（DATA MEM）用于存放 DSP 运行所需数据，为单口 SRAM，单周期完成读写操作；寄存器段为允许 CPU 通过总线访问的 DSP 寄存器；保留段暂未使用。

CODE MEM 位宽为 16，但仍按 word 寻址，亦即地址每次增 4。

CODE MEM 和 DATA MEM 仅支持按 word 寻址，即每次写入 32bit，读出 32bit。

DSP 地址空间需要在适当的时机才能访问。其中 DSP 的状态控制寄存器可以在任何时间访问；DSP 的 CODE MEM、DATA MEM、以及寄存器段中 CODIC 三角函数模块、DIV 除法器、SQRT 开方器只有在 DSP 暂停即 DSP0_SC.PAUSED=1 时才能进行寄存器访问。因为在 DSP 运行期间，所有的运算单元可能在被 DSP 使用，CPU 同时通过寄存器接口进行访问会造成访问冲突。因此在 DSP 运行期间，即 DSP0_SC.PAUSED=0 时，禁止通过寄存器接口访问 DSP 的运算单元。

DSP 遇到中断指令会提起中断等待 CPU 处理，同时 DSP 进入暂停状态，DSP0_SC.PAUSED 会置位为 1,。此外软件也可以在任意时刻通过写 DSP0_SC.PAUSED=1 来使 DSP 进入暂停状态，这一机制主要是为了防止 DSP 写入的程序中没有 IRQ 指令，导致 DSP 在不掉电的情况下会永久运行。

下表列出了 CPU 和 DSP 读写访问 DSP data Memory 时访问的实际内容。

表 19-4 DSP data memory 空间读写访问内容

DSP 寻址地址	DSP		CPU 寻址地址	CPU	
	Read	Write		Read	Write
0	Data[0]	Data[0]	0x4001_2F00	Data[0]	Data[0]
1	Data[1]	Data[1]	0x4001_2F04	Data[1]	Data[1]
2	Data[2]	Data[2]	0x4001_2F08	Data[2]	Data[2]
⋮	⋮	⋮	⋮	⋮	⋮
0x3B	Data[59]	Data[59]	0x4001_2FEC	Data[59]	Data[59]
0x3C	DSP0_PDI		0x4001_2FF0		
0x3D		DSP0_BSRR	0x4001_2FF4		
0x3E	DSP0_OP	DSP0_BRR	0x4001_2FF8	DSP0_OP	DSP0_OP
0x3F	DSP0_CL	DSP0_RES	0x4001_2FFC	DSP0_RES	DSP0_RES

其中 Data[n]表示 DSP DATA Memory 中的第 n 个 word。

DSP0_RES 寄存器是 DSP data memory 位于 0x3F 的 word，即最后一个 word，是 Data[63]的

拷贝。对于 CPU 软件来说，读写操作 0x4001_2FFC，操作的均是 DSP0_RES 寄存器，CPU 软件可以直接读写 DSP0_RES，即使在 DSP 运行过程中。对于 DSP 来说，DSP 读取 0x3F 得到的是 DSP0_CL 的值，写 0x3F 地址，会改写 DSP0_RES 寄存器的值。

DSP0_OP 是 DSP data memory 位于 0x3E 的 word，即倒数第二个 word，对于 CPU 软件来说，读写操作 0x4001_2FF8，操作的均是 DSP0_OP 寄存器，CPU 软件可以直接读写 DSP0_OP，即使在 DSP 运行过程中。对于 DSP 来说，DSP 读取 0x3D 得到的是 DSP0_OP 的值，DSP 写 0x3D 地址，写的是 DSP0_BRR 寄存器，即 GPIO 位清零操作。

19.1.6 与 CPU, GPIO, CL 模块的交互

在 DSP DATA 段的最后 4 个 word，被用作特殊用途，不再作为 DATA Memory 使用。

DSP 通过汇编指令读取 0x3C 位置的 DATA Memory 时，实际是读取 DSP0_PDI；

DSP 通过汇编指令向 DATA MEM 地址 0x3D 或 0x3E 写入数据时，实际是在向 GPIOx_BSRR，GPIOx_BRR 寄存器写入数据进行 GPIO_PDO 的位操作。具体 GPIO 对应关系，请参考 DSP 寄存器章节。

DSP 通过汇编指令读取 DATA MEM 地址 0x3F 时，实际是读取 CL_OUTPUT。

此外，0x3E 和 0x3F 地址，允许 CPU 在 DSP 运行时与 DSP 进行数据交互。DSP0_OP 一般用于 CPU 写入数据，DSP 读出数据，例如使用 DSP 模拟串口发送时；DSP0_RES 一般用于 DSP 写入数据，CPU 读出数据，例如使用 DSP 模拟串口接收时。由于 DMA 在受理 DSP 的搬运请求时就会清零 DSP0_SC.IF，DSP0_SC.PAUSED 并重启 DSP 运行，而数据搬运是在 DSP 受理请求之后，因此需要有读写通道允许 DSP 允许期间进行数据交互。

19.2 寄存器

19.2.1 地址分配

DSP 模块在芯片中的基地址是 0x4001_2000。DSP 寄存器在芯片中的基地址是 0x4001_3000。

表 19-5 DSP 寄存器列表

名称	偏移	说明
DSP0_SC	0x00	DSP 状态控制寄存器
DSP0_THETA	0x04	DSP sin/cos 输入角度寄存器
DSP0_X	0x08	DSP arctan/module 计算输入坐标 X 寄存器
DSP0_Y	0x0C	DSP arctan/module 计算输入坐标 Y 寄存器
DSP0_SIN	0x10	DSP sin/cos 计算结果 sin 寄存器
DSP0_COS	0x14	DSP sin/cos 计算结果 cos 寄存器
DSP0_MOD	0x18	DSP arctan 计算结果 sqrt(X^2+Y^2) 寄存器
DSP0_ARCTAN	0x1C	DSP arctan 计算结果 arctan(Y/X) 角度寄存器
DSP0_DID	0x20	DSP 除法操作被除数
DSP0_DIS	0x24	DSP 除法操作除数
DSP0_QUO	0x28	DSP 除法操作商
DSP0_REM	0x2C	DSP 除法操作余数

DSP0_RAD	0x30	DSP 开方操作被开方数
DSP0_SQRT	0x34	DSP 开方操作平方根
DSP0_PC	0x38	DSP 程序指针（Program Count）

19.2.2 DSP0_SC

地址：0x4001_3000

复位值：0x2

表 19-6 DSP 状态控制寄存器 DSP0_SC

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					DMA_ACC_EN	RE	IE					RESET_PC	CORDIC_MODE	PAUSED	IF
					RW	RW	RW					WO	RW	RW	RWIC
					0	0	0					0	0	1	0

位置	位名称	说明
[31:11]		未使用
[10]	DMA_ACC_EN	RE=1 时，DMA 在清零 IF 的同时，是否清零 PAUSED，并 RESET_PC。 0：禁用，DMA 收到请求后只清零 IF 1：使能，DMA 收到请求后清零 IF，PAUSED，并 RESET_PC DMA_ACC_EN=1 通常用于 DSP 软件模拟串行通讯接口的收发
[9]	RE	DSP DMA 请求使能，高有效，当 RE=1，且 IF =1 时，产生 DMA 请求 DMA 接收到请求并受理开始数据搬运，并将 IF 标志清零，不再需要软件清零 IF。
[8]	IE	DSP 中断使能，高有效，当 IE=1，且 IF =1 时，产生中断请求，即使 IE=0，当 DSP 执行到 IRQ 指令时，IF 仍会置位。
[7:4]		未使用
[3]	RESET_PC	当 DSP 暂停时，写 1 重置 DSP PC 到 0 地址
[2]	CORDIC_MODE	CORDIC mode, 0: arctan, 1: sin/cos
[1]	PAUSED	指示 DSP 是否处于暂停状态，当 DSP 执行到 IRQ 指令时此 bit 置 1，软件写可以将此 bit 置 1。软件将此 bit 清零可以启动 DSP 运行 0：DSP 正读取 CODE MEM 和 DATA MEM 自主运行 DSP 程序 1：DSP 暂停取指令，允许 ARM 软件通过寄存器访问 DSP 内部算术单元，向寄存器写入操作数触发运算，读取寄存器取得运算结果 对于不编写 DSP CODE MEM 程序的用户，PASUED 需保持为 1
[0]	IF	DSP 中断标志，写 1 清零

注意，复位后 DSP 处于暂停状态，即 DSP0_SC.PAUSED=1 时；DSP0_SC.CORDIC_MODE 用于 CPU 通过寄存器接口访问 CORDIC 模块时，sin/cos mode 和 arctan mode 的选择，CORDIC 模块计算 sin/cos 或 arctan 使用的是相同的硬件电路。因此在进行某一种计算前，应通过配置 DSP0_SC 计

算器进行适当模式选择。

只有当 DSP 处于暂停状态时，CPU 才可以通过寄存器接口调用 DSP。当 DSP 自主运行时，CPU 无法通过寄存器接口访问 DSP 内部资源。

DSP0_SC.CORDIC_MODE 位仅在 CPU 通过寄存器接口调用 DSP CORDIC 单元时需要设置，DSP 程序可以根据 SIN_COS 或 ARCTAN 指令直接进行模式切换，DSP0_SC.CORDIC_MODE 不再起作用。

软件调用 CORDIC 模块计算 sin/cos 时以角度 DSP0_THETA 为输入，计算并输出 sin/cos 结果到 DSP0_SIN/DSP0_COS 寄存器；计算 arctan 时以坐标 DSP0_X/DSP0_Y 为输入，计算并输出角度 theta= arctan (y/x)和 module= sqrt (x^2+y^2)到 DSP0_ARCTAN 和 DSP0_MOD 寄存器。

19.2.3 DSP sin/cos 相关寄存器

CORDIC 模块计算 sin/cos 和 arctan 使用的是相同的数据通路，因此通过 CPU 使用 CORDIC 模块进行 sin/cos 计算，需要先将 DSP0_SC.CORDIC_MODE 写为 1，使 CORDIC 进入 sin/cos 模式。

19.2.3.1 DSP0_THETA

地址：0x4001_3004

复位值：0x0

表 19-7 DSP sin/cos 角度输入寄存器

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

位置	位名称	说明
[31:16]		保留，读出时为符号扩展，即{16{DSP0_THETA[15]}}
[15:0]	THETA	DSP sin/cos 输入角度寄存器

DSP0_THETA 为 16 位有符号定点数，表示范围 (-32768 ~ 32767) 对应 (–π ~ π) 。

19.2.3.2 DSP0_SIN

地址：0x4001_3010

复位值：0x0

表 19-8 DSP sin/cos 正弦结果寄存器

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

位置	位名称	说明

[31:16]		保留，读出时为符号扩展，即{16{DSP0_SIN[15]}}
[15:0]	SIN	DSP sin/cos 计算结果 sin 寄存器

DSP0_SIN 为 16 位有符号定点数，其中 1bit 符号位，15bit 小数位；表示范围 (–1 ~ 1) 。

19.2.3.3 DSP0_COS

地址：0x4001_3014

复位值：0x0

表 19-9 DSP sin/cos 余弦结果寄存器

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

位置	位名称	说明
[31:16]		保留，读出时为符号扩展，即{16{DSP0_COS[15]}}
[15:0]	COS	DSP sin/cos 计算结果 cos 寄存器

DSP0_COS 为 16 位有符号定点数，其中 1bit 符号位，15bit 小数位；表示范围 (–1 ~ 1) 。

19.2.4 DSP arctan 相关寄存器

19.2.4.1 DSP0_X

地址：0x4001_3008

复位值：0x0

表 19-10 DSP arctan/module 坐标 X 输入寄存器

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

位置	位名称	说明
[31:16]		保留，读出时为符号扩展，即{16{DSP0_X[15]}}
[15:0]	X	DSP arctan/module 计算输入坐标 X 寄存器

DSP0_X 为 16 位有符号定点数，Q15 格式，其中 1bit 符号位，15bit 整数位，表示范围(–32768 ~ 32767)。

19.2.4.2 DSP0_Y

地址：0x4001_300C

复位值：0x0

表 19-11 DSP arctan/module 计算坐标 Y 输入寄存器

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

位置	位名称	说明
[31:16]		保留，读出时为符号扩展，即{16{DSP0_Y[15]}}
[15:0]	Y	DSP arctan/module 计算输入坐标 Y 寄存器

DSP0_Y 为 16 位有符号定点数，其中 1bit 符号位，15bit 整数位，表示范围(–32768 ~ 32767) 。

19.2.4.3 DSP0_MOD

地址：0x4001_3018

复位值：0x0

表 19-12 DSP arctan 向量模结果 sqrt(X^2+Y^2) 寄存器

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

位置	位名称	说明
[31:16]		保留，读恒为 0
[15:0]	MOD	DSP arctan 计算结果 sqrt(X^2+Y^2) 寄存器

DSP0_MOD 为 16 位无符号定点数，16bit 均为整数位，表示范围(0~65535)。

19.2.4.4 DSP0_ARCTAN

地址：0x4001_301C

复位值：0x0

表 19-13 DSP arctan 角度结果 arctan(Y/X) 角度寄存器

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

0

位置	位名称	说明
[31:16]		保留，读出时为符号扩展，即{16{DSP0_ARCTAN[15]}}
[15:0]	ARCTAN	DSP arctan 计算结果 arctan(Y/X) 角度寄存器

DSP0_ARCTAN 为 16 位有符号定点数，表示范围 (-32768 ~ 32767) 对应 (–π ~ π) 。

19.2.5 **DSP 除法相关寄存器**

19.2.5.1 DSP0_DID

地址：0x4001_3020

复位值：0x0

表 19-14 DSP 除法被除数寄存器

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															

位置	位名称	说明
[31:0]	DID	DSP 除法被除数寄存器

19.2.5.2 DSP0_DIS

地址：0x4001_3024

复位值：0x0

表 19-15 DSP 除法除数寄存器

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

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

位置	位名称	说明
[31:0]	DIS	DSP 除法除数寄存器

19.2.5.3 DSP0_QUO

地址：0x4001_3028

复位值：0x0

表 19-16 DSP 除法商寄存器

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															

位置	位名称	说明
[31:0]	QUO	DSP 商寄存器

19.2.5.4 DSP0_REM

地址：0x4001_302C

复位值：0x0

表 19-17 DSP 除法余数寄存器

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

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

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

位置	权限	说明
[31:0]	REM	DSP 除法余数寄存器

当 CPU 需要使用 DSP 除法器时，应首先保证 DSP 处于暂停状态。先向 DSP 写入被除数，后写入除数；写入除数可以触发一次除法操作，32/32 位除法需要 12 周期完成，期间读取除法结果 DSP0_QUO 或 DSP0_REM 会使 CPU 进入等待状态，等待除法计算完成并通过总线返回计算结果。

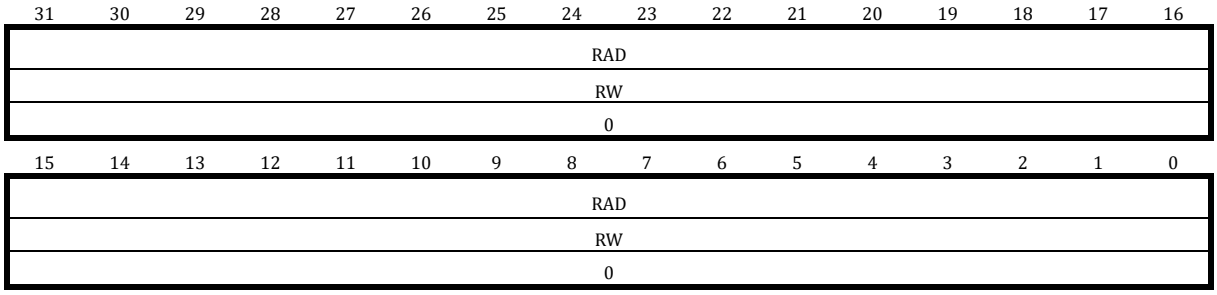
19.2.6 DSP 开方相关寄存器

19.2.6.1 DSP0_RAD

地址：0x4001_3030

复位值：0x0

表 19-18 DSP 被开方数寄存器



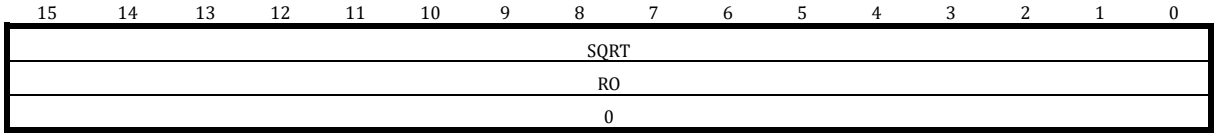
位置	位名称	说明
[31:0]	RAD	DSP 被开方数寄存器

19.2.6.2 DSP0_SQRT

地址：0x4001_3034

复位值：0x0

表 19-19 DSP 平方根寄存器



位置	位名称	说明
[31:16]		保留，读出时恒为 0
[15:0]	SQRT	DSP 平方根寄存器

当 CPU 需要使用 DSP 开方器时，应首先保证 DSP 处于暂停状态。向 DSP 写入被开方数；写入被开方数可以触发一次开方操作，32 位开方需要 8 周期完成，期间读取开方结果 DSP0_SQRT 会使 CPU 进入等待状态，等待开方计算完成并通过总线返回计算结果。

19.2.7 DSP0_PC

地址：0x4001_3038

复位值：0x0

表 19-20 DSP 程序指针寄存器 DSP0_PC

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

位置	位名称	说明
[31:9]		保留
[8:0]	PC	当 DSP 出于暂停状态时，允许软件直接写入 PC 值。 无论 DSP 是否出于暂停状态，PC 寄存器均可读

19.2.8 DSP0_PDI

通过 DSP 汇编程序访问，软件无法直接访问，对应 DSP DATA Memory 地址:0x3C

复位值:0x0

表 19-21 DSP0_PDI 寄存器

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

位置	位名称	说明
[31:16]		未使用
[15]	P2.14	GPIO2_PDI[14]
[14]	P2.7	GPIO2_PDI[7]
[13]	P2.6	GPIO2_PDI[6]
[12]	P2.5	GPIO2_PDI[5]
[11]	P2.4	GPIO2_PDI[4]
[10]	P2.3	GPIO2_PDI[3]
[9]	P1.0	GPIO1_PDI[0]
[8]	P0.15	GPIO0_PDI[15]
[7]	P0.14	GPIO0_PDI[14]
[6]	P0.13	GPIO0_PDI[13]
[5]	P0.12	GPIO0_PDI[12]
[4]	P0.11	GPIO0_PDI[11]

[3]	P0.4	GPIO0_PDI[4]
[2]	P0.3	GPIO0_PDI[3]
[1]	P0.2	GPIO0_PDI[2]
[0]	P0.0	GPIO0_PDI[0]

19.2.9 DSP0_BSRR

通过 DSP 汇编程序访问，软件无法直接访问，对应 DSP DATA Memory 地址:0x3D

复位值:0x0

表 19-22 DSP0_BSRR 寄存器

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

位置	位名称	说明
[31:16]		未使用
[15]	P2.14	GPIO2_BSRR[14]
[14]	P2.7	GPIO2_BSRR[7]
[13]	P2.6	GPIO2_BSRR[6]
[12]	P2.5	GPIO2_BSRR[5]
[11]	P2.4	GPIO2_BSRR[4]
[10]	P2.3	GPIO2_BSRR[3]
[9]	P1.0	GPIO1_BSRR[0]
[8]	P0.15	GPIO0_BSRR[15]
[7]	P0.14	GPIO0_BSRR[14]
[6]	P0.13	GPIO0_BSRR[13]
[5]	P0.12	GPIO0_BSRR[12]
[4]	P0.11	GPIO0_BSRR[11]
[3]	P0.4	GPIO0_BSRR[4]
[2]	P0.3	GPIO0_BSRR[3]
[1]	P0.2	GPIO0_BSRR[2]
[0]	P0.0	GPIO0_BSRR[0]

19.2.10 DSP0_BRR

通过 DSP 汇编程序访问，软件无法直接访问，对应 DSP DATA Memory 地址:0x3E

复位值:0x0

表 19-23 DSP0_BRR 寄存器

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

P2.14	P2.7	P2.6	P2.5	P2.4	P2.3	P1.0	P0.15	P0.14	P0.13	P0.12	P0.11	P0.4	P0.3	P0.2	P0.0
RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:16]		未使用
[15]	P2.14	GPIO2_BRR[14]
[14]	P2.7	GPIO2_BRR[7]
[13]	P2.6	GPIO2_BRR[6]
[12]	P2.5	GPIO2_BRR[5]
[11]	P2.4	GPIO2_BRR[4]
[10]	P2.3	GPIO2_BRR[3]
[9]	P1.0	GPIO1_BRR[0]
[8]	P0.15	GPIO0_BRR[15]
[7]	P0.14	GPIO0_BRR[14]
[6]	P0.13	GPIO0_BRR[13]
[5]	P0.12	GPIO0_BRR[12]
[4]	P0.11	GPIO0_BRR[11]
[3]	P0.4	GPIO0_BRR[4]
[2]	P0.3	GPIO0_BRR[3]
[1]	P0.2	GPIO0_BRR[2]
[0]	P0.0	GPIO0_BRR[0]

19.2.11 DSP0_CL

DSP0_CL 通过 DSP 汇编程序访问，软件无法直接访问，对应 DSP DATA Memory 地址:0x3F

复位值:0x0

表 19-24 DSP0_CL 寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												CL_OUTPUT3	CL_OUTPUT2	CL_OUTPUT1	CL_OUTPUT0
												RO	RO	RO	RO
												0	0	0	0

位置	位名称	说明
[31:4]		未使用
[3]	CL_OUTPUT3	GPIO0_BRR[4]
[2]	CL_OUTPUT2	GPIO0_BRR[3]
[1]	CL_OUTPUT1	GPIO0_BRR[2]
[0]	CL_OUTPUT0	GPIO0_BRR[0]

19.2.12 DSP0_OP

地址: 0x4001_2FF8

复位值:0x0

表 19-25 DSP0_OP 寄存器

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

位置	位名称	说明
[31:0]	OPERAND	DSP 操作数寄存器

DSP0_OP 是 DSP DATA memory 位于 0x3E 地址的内容，即倒数第二个 word 内容。对于 CPU 软件来说可读可写，且允许 DSP 运行期间进行读写。对于 DSP 来说，DSP0_OP 只读，DSP 写 0x3E 地址实际写的是 DSP0_BRR 寄存器。

19.2.13 DSP0_RES

地址: 0x4001_2FFC

复位值:0x0

表 19-26 DSP0_RES 寄存器

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

位置	位名称	说明
[31:0]	RESULT	DSP 运行结果寄存器

DSP0_RES 是 DSP DATA memory 位于 0x3F 地址的内容，即最后一个 word 内容的拷贝，对于 CPU 软件来说只读，且允许 DSP 运行期间进行读取。对于 DSP 来说，DSP0_RES 只可写，DSP 读取 0x3F 地址实际读取的是 DSP0_CL 的内容。

19.3 **DSP 指令集**

19.3.1 Instruction Set Summary

表 19-27 DSP 指令集

Operation	Description	Assembler	Cycles
Add		ADD Rd1 Rs1 Rs2	1
Subtract		SUB Rd1 Rs1 Rs2	1

And		AND	Rd1 Rs1 Rs2	1
Or		OR	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
	Logical right shift	LSR	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	12
Saturation		SAT	Rd1 Rs1 Rs2	1
	4bit Immediate	SATI	Rd1 #<Imm1> #<Imm2>	1
Cordic	SIN/COS	SIN_COS	Rd1 Rd2 Rs1	20
	Arctan/Module	ARCTAN	Rd1 Rs1 Rs2	20
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
	Jump if equal to	JEI	Rs1 Rs2 Rs3	2
	Immediate	JLEI	Rs1 Rs2 #<Imm>	2
Miscellaneous	Generate IRQ and Pause DSP	IRQ		1

DSP 使用 16bit 定长编码指令，因为通用寄存器共 8 个，所以寄存器编码需要 3bit。其中大部分指令为 3 操作数指令，包括两个源操作数寄存器和一个目的操作数寄存器；部分指令包含立即数；部分指令会涉及 4 个操作数，以乘加 (MAC) 操作为例，Rd = Rs1*Rs2+Rs3，由于指令长度不够表示 4 个寄存器，将 Rd 固定为 R7，在指令编码中不显示。其余 4 操作数指令还有 ARCTAN/DIV。具体操作数分配见下文指令的详细解释。

19.3.2 ADD

19.3.2.1编码

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

19.3.2.2汇编语法

```
ADD     Rd1 Rs1 Rs2
```

19.3.2.3伪代码

```
Rd1 = Rs1 + Rs2
```

结果防溢出保护，

上溢出后 Rd1 = 0x7FFF_FFFF，

下溢出后 Rd1 = 0x8000_0000

19.3.3 AND

19.3.3.1指令编码

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	Rs2			Rd1			Rs1		

19.3.3.2汇编语法

AND　　Rd1　Rs1 Rs2

19.3.3.3伪代码

Rd1 = Rs1&Rs2

19.3.4 OR

19.3.4.1指令编码

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	Rs2			Rd1			Rs1		

19.3.4.2汇编语法

OR　　Rd1　Rs1 Rs2

19.3.4.3伪代码

Rd1 = Rs1|Rs2

19.3.5 SUB

19.3.5.1指令编码

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

19.3.5.2汇编语法

SUB Rd1 Rs1 Rs2

19.3.5.3伪代码

Rd1 = Rs1 – Rs2

结果防溢出保护，

上溢出后 Rd1 = 0x7FFF_FFFF，

下溢出后 Rd1 = 0x8000_0000

19.3.6 ASR

19.3.6.1指令编码

15	14	13	12	11	10	9	8 7 6	5 4 3	2 1 0
0	0	1	0	0	0	0	Rs2	Rd1	Rs1

19.3.6.2汇编语法

ASR Rd1 Rs1 Rs2

19.3.6.3伪代码

Rd1 = Rs1 >> Rs2

算术右移指令只支持 0~31 bit 的右移。

19.3.7 ASRI

19.3.7.1指令编码

15	14	13	12	11	10 9 8 7 6	5 4 3	2 1 0
0	0	1	0	1	Imm	Rd1	Rs1

立即数为 5bit 无符号数，表示数值范围是 0~31.

19.3.7.2汇编语法

ASRI Rd1 Rs1 Imm

19.3.7.3伪代码

Rd1 = Rs1 >> Imm

带立即数的算术右移指令只支持 0~31 bit 的右移。

19.3.8 LSL

19.3.8.1指令编码

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		

19.3.8.2汇编语法

LSL Rd1 Rs1 Rs2

19.3.8.3伪代码

Rd1 = Rs1 << Rs2

逻辑左移只支持 0~31 bit 左移。

19.3.9 LSR

19.3.9.1指令编码

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	Rs2			Rd1			Rs1		

19.3.9.2汇编语法

LSL Rd1 Rs1 Rs2

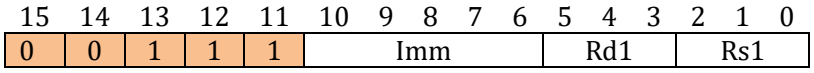
19.3.9.3伪代码

Rd1 = Rs1 >> Rs2

逻辑右移只支持 0~31 bit 左移。

19.3.10 LSLI

19.3.10.1指令编码



立即数为 5bit 无符号数，表示数值范围是 0~31.

19.3.10.2汇编语法

LSLI Rd1 Rs1 Imm

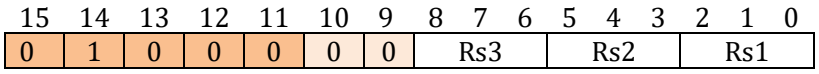
19.3.10.3伪代码

Rd1 = Rs1 << Imm

带立即数的逻辑左移只支持 0~31 bit 左移。

19.3.11 MAC

19.3.11.1指令编码



19.3.11.2汇编语法

MAC Rs1 Rs2 Rs3

19.3.11.3伪代码

Rd7 = Rs1 × Rs2 + Rs3

结果防溢出保护，

上溢出后 Rd7 = 0x7FFF_FFFF，

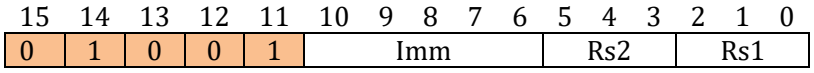
下溢出后 Rd7 = 0x8000_0000

其中 Rs1、Rs2 为 32 位有符号数，Rs3 为 32 位有符号数。当 Rs3 为 R0 时，MAC 可以当做乘法指令 MUL 使用。

19.3.12 MACI (reserved)

带立即数的乘加指令在此版本 DSP 中保留，未实现。

19.3.12.1指令编码



立即数为 5bit 有符号数，表示数值范围是-16~15.

19.3.12.2汇编语法

MACI Rs1 Rs2 Imm

19.3.12.3伪代码

Rd7 = Rs1 × Rs2 + Imm

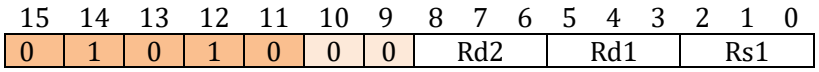
结果防溢出保护，

上溢出后 Rd7 = 0x7FFF_FFFF，

下溢出后 Rd7 = 0x8000_0000

19.3.13 DIV

19.3.13.1指令编码



19.3.13.2汇编语法

DIV Rd2 Rd1 Rs1

19.3.13.3伪代码

Rd1 = R7/Rs1, Rd2 = R7 % Rs1

除法指令需要 12 个周期才会结束提交，在除法进行计算的过程中，DSP 不应该再发射其他的多周期指令，即同一时刻只能由一个多周期指令在 long-pipeline 中。其他多周期指令包括三角函数指令以及开方指令。多周期指令可以后台运行，即多周期指令运算的同时，DSP 可以继续执行其他的单周期指令。但同一时间只能有一条多周期指令在后台运行。在多周期指令计算期间，DSP 仍

可以使用多周期指令的目的操作数，但需要注意是当多周期计算完成提交结果的时候，目的操作数寄存器会被改写。

其中 R7，Rs1，Rd1，Rd2 均为 32 位有符号数。除法指令的被除数固定为 R7。

19.3.14 SAT

19.3.14.1指令编码

15	14	13	12	11	10	9	8 7 6	5 4 3	2 1 0
0	1	1	0	0	0	0	Rs2	Rd1	Rs1

19.3.14.2汇编语法

SAT Rd1 Rs1 Rs2

19.3.14.3伪代码

If (Rd1<Rs1) Rd1=Rs1; else if (Rd1>Rs2) Rd1=Rs2

19.3.15 SATI (reserved)

带立即数的饱和指令在此版本 DSP 中保留，未实现。

19.3.15.1指令编码

15	14	13	12	11	10 9 8 7 6	5 4 3 2	1 0
0	1	1	0	1	Imm2	Imm1	Rs1

19.3.15.2汇编语法

SATI Rd1 Imm1 Imm2

19.3.15.3伪代码

If (Rd1<Imm1) Rd1=Imm1; else if (Rd1>Imm2) Rd1=Imm2

19.3.16 SIN_COS

19.3.16.1指令编码

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

19.3.16.2汇编语法

SIN_COS Rd1 Rd2 Rs1

19.3.16.3伪代码

Sin/cos 指令 20 周期结束并提交，在其执行期间，DSP 不应该发射其他的多周期指令。

Rd1=cos (Rs1); Rd2=sin (Rs1)

19.3.17 ARCTAN

19.3.17.1指令编码

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		

19.3.17.2汇编语法

ARCTAN Rd1 Rs1 Rs2

19.3.17.3伪代码

ARCTAN 指令 20 周期结束并提交，在其执行期间，DSP 不应该发射其他的多周期指令。

Rd1= arctan(Rs2/Rs1); R6 = sqrt(Rs1^2+Rs2^2)

19.3.18 SQRT

19.3.18.1指令编码

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		

19.3.18.2汇编语法

SQRT Rd1 Rs1

19.3.18.3伪代码

开方指令 8 周期结束并提交，在其执行期间，DSP 不应该发射其他的多周期指令。

Rd1 = sqrt(Rs1)

19.3.19 LDRWI

19.3.19.1指令编码

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

19.3.19.2汇编语法

LDRWI Rd1 Imm

19.3.19.3伪代码

Rd1=word(SRAM[imm])

由于 Load 指令都是立即数指令，访问的数据地址可以在译码阶段产生，因此 load 操作可以在一个周期内完成。而类似的 load 指令，CPU 需要访问寄存器来计算地址，因此需要 2 个周期。

立即数 imm 的表达范围是 0~63。因为 DSP data mem 是 64 个 32bit word 构成。

19.3.20 LDRDHI

19.3.20.1指令编码

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	Imm						Rd1			Rd2		

19.3.20.2汇编语法

LDRDHI Rd1 Rd2 Imm

19.3.20.3伪代码

LDRDHI Rd1 Rd2

{Rd1,Rd2}= word(SRAM[Imm]),

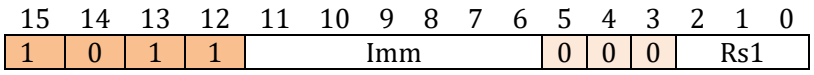
从 data mem 取回的 32bit 数据高 16 位会进行符号扩展为 32 位，然后赋值给 Rd1，

低 16 位会进行符号扩展为 32 位，然后赋值给 Rd2。

立即数 imm 的表达范围是 0~63。因为 DSP data mem 是 64 个 32bit word 构成。

19.3.21 STRWI

19.3.21.1指令编码



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		

19.3.21.2汇编语法

STRWI Rs1 Imm

19.3.21.3伪代码

word{SRAM[imm]}=Rs1

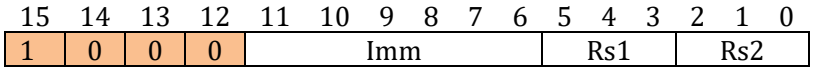
Store 指令的数据来自寄存器，因此即使待写入地址在译码阶段产生也无法立即完成 Store 操作。需要将地址锁存一个周期，然后与写入的数据一起送至 data memory 接口。因此 Store 指令后续如果立刻连接 load 指令会出现访问冲突。在设计汇编程序时应避免出现此种指令序列。如果 STR 指令后必须是 LDR 指令，可以在两者之间插入 ADD R0 R0 R0 指令作为指令 bubble。

立即数 imm 的表达范围是 0~63。因为 DSP data mem 是 64 个 32bit word 构成。

19.3.22 STRDHI (reserved)

带立即数的存储双半字指令在此版本 DSP 中保留，未实现。

19.3.22.1指令编码



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	Imm						Rs1			Rs2		

19.3.22.2汇编语法

STRDHI Rs1 Rs2 Imm

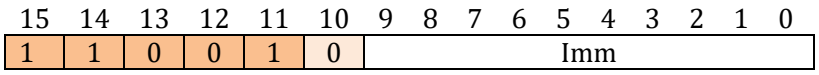
19.3.22.3伪代码

word{SRAM[imm]}={Rs1, Rs2}

立即数 imm 的表达范围是 0~63。因为 DSP data mem 是 64 个 32bit word 构成。

19.3.23 JUMPI

19.3.23.1指令编码



19.3.23.2汇编语法

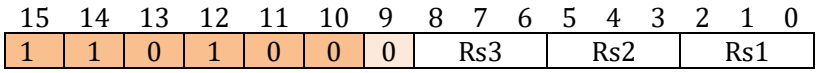
JUMPI Imm

19.3.23.3伪代码

PC = PC + 1 + IMM

19.3.24 JLE

19.3.24.1指令编码



19.3.24.2汇编语法

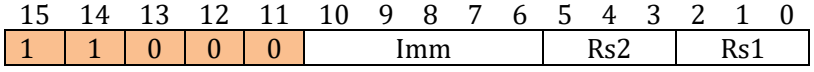
JLE Rs1 Rs2 Rs3

19.3.24.3伪代码

PC = PC + 1 + Rs3, if (Rs1 <= Rs2)

19.3.25 JEI

19.3.25.1指令编码



19.3.25.2汇编语法

JEI Rs1 Rs2 Imm

19.3.25.3伪代码

PC = PC + 1 + IMM, if (Rs1 == Rs2)

19.3.26 JLEI

19.3.26.1指令编码

15	14	13	12	11	10 9 8 7 6	5 4 3	2 1 0
1	1	0	1	1	Imm	Rs2	Rs1

19.3.26.2汇编语法

JLEI Rs1 Rs2 Imm

19.3.26.3伪代码

PC = PC + 1 + IMM, if (Rs1 <= Rs2)

19.3.27 IRQ

19.3.27.1指令编码

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

19.3.27.2汇编语法

IRQ

19.3.27.3伪代码

IRQ 指令会产生中断，等待 CPU 处理，并将 DSP 暂停运行。

19.3.28 R（仅用于模拟器）

19.3.28.1指令编码

无

19.3.28.2汇编语法

```
R5 0x5555          # Assign 0x5555 to R5
R1 20              # Assign 20 to R1
```

19.3.28.3伪代码

调试指令，仅在 DSP Emulator 中使用。可用于在 DSP 程序的任意位置将 R1~R7 置为任意指定数值。

19.3.29 BREAK（仅用于模拟器）

19.3.29.1指令编码

无

19.3.29.2汇编语法

```
BREAK
```

19.3.29.3伪代码

调试指令，仅在 DSP Emulator 中使用。可用于在 DSP 程序的任意位置插入断点并打印 R1~R7 寄存器值。断点命中后按回车继续程序运行。

19.3.30 END（仅用于模拟器）

19.3.30.1指令编码

无

19.3.30.2汇编语法

```
END
```

19.3.30.3伪代码

类似于断点指令，仅在 DSP Emulator 中使用。在指令模拟中遇到该指令即中断模拟器的运行，直接打印核心寄存器值。

19.4 应用指南

19.4.1 访存寻址

DSP 使用立即数寻址，由于 DATA MEM 地址空间有限，STR 和 LDR 指令中的 6bit 立即数可以直接表示 DATA MEM 全部地址偏移。

以如下 DATA MEM 内容为例，第一行数据 0x01000100 对应 0x0 地址，第二行数据 0x30005000 对应 0x1 地址，第三行数据 0x0003FFF8 对应 0x2 地址。

使用 LDRDHI R1 R2 0x1 可对 R1,R2 进行赋值 R1=0x3000，R2=0x5000

使用 STRWI R3 0x3 可以将 R3 的 32bit 数据写入 0x3 地址并覆盖掉数据 0xFFFFE000。

需要注意的是，虽然 DSP 访存寻址每增加 1 是增加 4Byte，对于 CPU 来说，每增加 4 地址才是增加 4Byte。因此 CPU 访存 DSP DATA MEM 时，应该按照 DSP0_DATA_MEM_BASE+offset*4 的方式进行计算。以如下 DATA MEM 内容为例，第二行数据 0x30005000 对应的 CPU 寻址地址为 0x4001_4804，第三行数据 0x0003FFF8 对应的 CPU 寻址地址为 0x4001_4808。

DATA MEM：

0x00100010

0x30005000

0x0003FFF8

0xFFFFE000

0x30004000

0x7FFFFFFF

0x7FFFFFFF

0xF0000003

0x00007FFF # 8

0x00008000

0x80000000

0x00000000

19.4.2 多周期指令的的延迟提交

DSP 的多周期算术指令包括除法、开方、三角函数。

除法指令需要 10 个总线周期（96MHz）完成。

开方指令需要 8 个总线周期（96MHz）完成。

三角函数指令需要 20 个总线周期（96MHz）完成。

为了充分利用 DSP 性能，允许多周期指令后台执行，即多周期指令计算期间 DSP 仍可其他指令，而不会阻塞流水线。这需要在编程时对指令序列有所考虑，在多周期指令的发射和结果使用之间插入其他不相关指令。如下指令序列，其中 ADD R0 R0 R0 作用类似 NOP，在实际应用中可以替换为其他指令。

```
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
```

19.4.3 软件模拟串行接口

通过 DSP0_PDI 和 DSP0_BSRR、DSP0_BRR 寄存器，DSP 程序可以与 GPIO 进行交互。不同于 CPU 程序会频繁进入中断处理程序导致软件运行实时性较差，DSP 程序运行于相对封闭的程序环境，程序执行具有较强的时间精确性。利用 DSP 程序的这种特性可以实现 DSP 模拟串行接口。以 UART 发送为例，通常 UART 的帧结构为 1bit 起始位+1 字节数据+1bit 停止位。DSP 可以按照一定的时间间隔通过操作 DSP0_BSRR、DSP0_BRR 寄存器进行 GPIO_PDO 的置位或清零，逐位发送上述串行数据，发送完成后通过中断指令通知 CPU/DMA 发送完毕。如果是进行 UART 接收，则 DSP 程序一直轮询 DSP0_PDI 寄存器，如果检测到对应为 0，则说明检测到 UART 起始位，则开始按照一定的时间间隔定时读取 DSP0_PDI 寄存器，读入完整一帧 UART 数据。

同理，DSP 也通过以上方式软件模拟 SPI 或 I2C 等其他串行接口。

20 IWDG 独立看门狗

20.1 概述

独立看门狗使用 32kHz 低速 RC(LRC/LSI)时钟进行工作，在系统主时钟停止的情况下仍可保证正常工作。

独立看门狗内部包含一个 21bit 递减计数器，启动后从配置值开始递减计数。独立看门狗可以配置产生系统休眠唤醒源，作为定时唤醒源。在超时情况下也可以产生全芯片复位信号。当 MCU 进入深度休眠状态时，包括 PLL/HRC 在内的系统时钟关闭，而 LRC 时钟是一直存在的，所以独立看门狗可以继续正常工作。

通常看门狗的定时唤醒门限值小于超时复位门限值，但大于 0x8，即独立看门狗重新装载后从 IWDG_RTH 开始递减计数，当计数到 IWDG_WTH 时产生唤醒信号。通常如果要使用 IWDG 进行休眠定时唤醒，在进入休眠前将 IWDG 刷新重置，IWDG 从 IWDG_RTH 开始递减计数，在经过一段时间后产生 IWDG 唤醒信号。如果不使能 IWDG 休眠定时唤醒，当 IWDG 产生全芯片复位时也可以将芯片从休眠中解除，但同时会复位 MCU 所有寄存器配置。

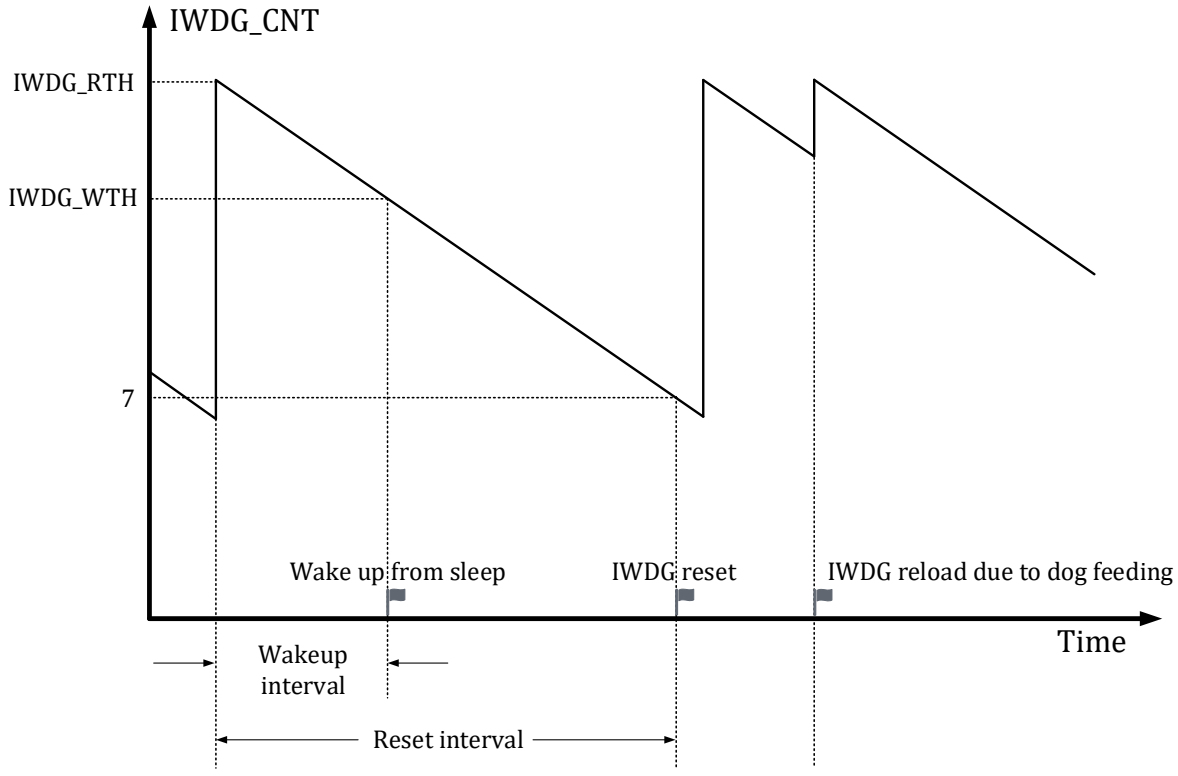


图 20-1 IWDG 递减计数事件发生示例

独立看门狗超时复位时间由如下公式计算

$Timeout_{IWDG}=t_{LRC}\times(IWDG\_RTH-7)$

独立看门狗唤醒时间由如下公式计算

$Wakeup_{IWDG} = t_{LRC} \times (IWDG\_RTH - IWDG\_WTH)$

其中 $Timeout_{IWDG}$ 为独立看门狗超时复位时间，$Wakeup_{IWDG}$ 为独立看门狗唤醒时间，t_{LRC} 为 LRC 时钟周期，1/32kHz=31.25uS，IWDG_RTH 为独立看门狗超时复位门限值，IWDG_WTH 为独立看门狗唤醒门限值。

IWDG_RTH=0x001000 对应独立看门狗最小复位时间间隔为 4096/32kHz≈128ms。

IWDG_RTH=0x1FF000 对应独立看门狗最大复位时间间隔为 511×4096/32kHz≈64s。

需要注意，由于低速 RC 时钟精度有限，不同芯片存在一定偏差，通常低速 RC 时钟偏差在全温度范围内不超过±50%。

20.2 寄存器

20.2.1 地址分配

IWDG 基地址为 0x40011700

表 20-1 独立看门狗寄存器

名称	偏移	说明
IWDG_PSW	0x00	独立看门狗密码寄存器
IWDG_CFG	0x04	独立看门狗配置寄存器
IWDG_CLR	0x08	独立看门狗清零寄存器
IWDG_WTH	0x0C	独立看门狗计数器定时唤醒门限值寄存器
IWDG_RTH	0x10	独立看门狗计数器超时复位门限值寄存器
IWDG_CNT	0x14	独立看门狗计数器当前计数值寄存器

20.2.2 IWDG_PSW 独立看门狗密码寄存器

地址:0x4001_1700

复位值:0x0

表 20-2 IWDG_PSW 独立看门狗密码寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	PSW	写入 0xA6B4，才能对 IWDG_CLR/ IWDG_RTH 等进行写操作，对 IWDG_CLR 或 IWDG_RTH 的写操作会将密码清空，因此每次对看门狗 IWDG_CLR/ IWDG_RTH 寄存器进行写操作前都需要重新写入密码

20.2.3 IWDG_CFG 独立看门狗配置寄存器

地址:0x4001_1704

复位值:0x0

表 20-3 IWDG_CFG 独立看门狗配置寄存器

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

位置	位名称	说明
[31:5]		未使用
[4]	DWK_EN	深度休眠定时唤醒使能，0:禁用，1:使能
[3:1]		未使用
[0]	WDG_EN	独立看门狗使能，0:禁用，1:使能。默认使能，写 1 置位，写 0 同时向[15:8]写入 0x3C 可清零。当看门狗被禁用时，不再产生复位信号，但仍可计数并产生定时唤醒信号

IWDG_CFG 的写入无需向 IWDG_PSW 写入密码。

20.2.4 IWDG_CLR 独立看门狗清零寄存器

地址:0x4001_1708

复位值:0x0

表 20-4 IWDG_CLR 看门狗清零寄存器

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

位置	位名称	说明
[31:16]	IWDG_CLR	未使用
[15:0]		写入字节 16'b0111_1001_1000_110B$_0$，高 15 位为密码，密码正确时，B[0]才能写入 B[0]写入 1，则重置 WDT 计数器为 TH，且该 bit 写入后自动清零，写入 0 无效

IWDG_CLR 的写入需要先向 IWDG_PSW 写入密码。

20.2.5 IWDG_WTH 独立看门狗定时唤醒门限寄存器

地址:0x4001_170C

复位值:0x001F_F000

表 20-5 IWDG_WTH 独立看门狗超时复位门限寄存器

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
											IWDG_WTH				
											RW				
											0x1F				

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

位置	位名称	说明
[31:21]	IWDG_WTH	未使用
[20:12]		看门狗定时唤醒门限值，看门狗使用 32kHz LRC 时钟从 IWDG_RTH 开始计数，递减计数至 IWDG_WTH 产生唤醒信号。
[11:0]		恒为 0

IWDG_ WTH 的写入无需向 IWDG_PSW 写入密码。

20.2.6 IWDG_RTH 独立看门狗超时复位门限寄存器

地址:0x4001_1710

复位值:0x001F_F000

表 20-6 IWDG_RTH 独立看门狗超时复位门限寄存器

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
											IWDG_RTH				
											RW				
											0x1F				

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

位置	位名称	说明
[31:21]	IWDG_RTH	未使用
[20:12]		看门狗超时复位门限值，也是重新装载值。看门狗使用 32kHz LRC 时钟从 IWDG_RTH 开始计数，计数至 0x7 产生复位。向该寄存器写入 0x0，会被硬件强制改写为 0x1000。先向 IWDG_PSW 写入正确密码才能改写 IWDG_RTH 寄存器。改写 IWDG_RTH 同样具有重置看门狗计数器的作用，看门狗会从新的 IWDG_RTH 开始计数。

[11:0]		恒为 0

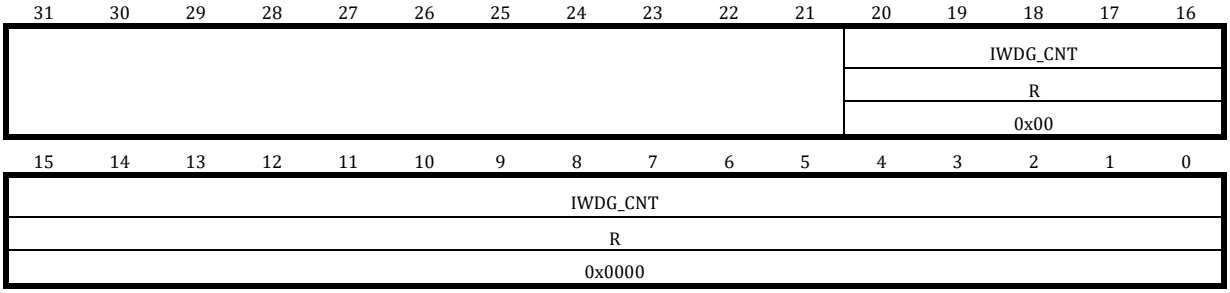
为防止 IWDG_RTH 被写入为 0，当软件写入值为全 0 时，硬件会强制改写为 0x1000，且寄存器低 12 位恒为 0。IWDG_RTG 的写入需要向 IWDG_PSW 写入密码。

20.2.7 IWDG_CNT 独立看门狗当前计数值寄存器

地址:0x4001_1714

复位值:0x0000_0000

表 20-7 IWDG_CNT 独立看门狗当前计数值寄存器



位置	位名称	说明
[31:21]	IWDG_CNT	未使用
[20:0]		看门狗当前计数值，此数值≤IWDG_TH

21 PMU 功耗管理模块

芯片可以通过降低运行时钟频率，关闭部分电路时钟来达到降低功耗的目的。

21.1 外设时钟门控

外设时钟由系统高速时钟 MCLK 经过门控或分频而来；当外设不需要使用时可以通过配置 SYS_CLK_FEN 寄存器关闭相应的外设时钟。对于每一个外设的工作时钟，均有一个时钟门控，详见 6.3.7。外设时钟上电后默认是关闭的，使用相应外设模块之前需要由软件来开启。

21.2 外设时钟分频

部分外设有独立的时钟分频模块使得该模块可以工作在合适的较低的的时钟频率上。

其中 I2C 使用 SYS_CLK_DIV0 作为分频系数，UART 使用 SYS_CLK_DIV2 作为分频系数，详见 6.3.3 和 6.3.6。UART 的波特率在 UART 模块内部还有一个额外的分频器。

21.3 低功耗模式

表 21-1 低功耗模式汇总

模式	模式进入	模式退出	PLL/HSI	LSI
休眠 Sleep	WFI/WFE	任意中断 Debug 操作 外部复位 IWDG 复位	PLL/HSI On，CPU 时钟 Off，NVIC/外设时钟 On	On
深度休眠 Deep Sleep	SLEEPDEEP + WFI/WFE	IWDG 定时唤醒 IO 唤醒 Debug 操作 外部复位 IWDG 复位	PLL/HSI Off，CPU/外设时钟 Off	

21.4 Sleep Mode 休眠模式

休眠模式下，CPU 时钟被关闭，但 NVIC 模块仍继续工作，所有的外设模块以及 IO 正常工作。

休眠模式对电路供电没有影响，所有寄存器状态和存储器数据在休眠过程中保持正常供电。

可参考休眠例程，进入休眠前关闭数字部分各个模块的时钟，关闭模拟 ADC/OPA/CMP/DAC 等模块。

21.4.1 模式进入

休眠模式可以通过执行 WFI/WFE 指令进入。根据 CPU 内核 SLEEPONEXIT 位的设置情况，休

眠模式的进入有两种不同方式。

Sleep-now:如果 SLEEPONEXIT 为 0，则 CPU 在执行 WFI/WFE 指令后立即进入休眠模式

Sleep-on-exit:如果设置 SLEEPONEXIT 为 1，CPU 在处理完所有中断后进入休眠模式

21.4.2 模式退出

如果使用 WFI 进入休眠，则任意中断可唤醒 CPU。

如果使用 WFE 进入休眠，则外设中断标志或 IO 唤醒事件可作为唤醒事件。当使用外设中断标志作为唤醒事件时，外设向 CPU 提起中断信号，但 NVIC 中并不使能对应中断，且需要设置 SEVONPEND 为 1。唤醒后外设中断标志以及 NVIC 中断 pending 位需要被软件清除。

使用此种方式进行休眠唤醒可以获得最短唤醒时间，因为不涉及中断的进入退出。

Debug 操作可以将芯片从休眠模式中唤醒。

21.5 Deep Sleep Mode 深度休眠模式

深度休眠模式通常会关闭所有系统高速时钟，包括 PLL/HSI。进入深度休眠前，需要设置 SYS_AFE_REG5=0x0500 来关闭 PLL/HRC/BGP。32kHz RC 时钟 LSI 仍正常工作。同时 LDO 会进入低功耗模式，BGP 模块被关闭。

相比休眠模式，深度休眠模式可以进一步降低功耗。

深度休眠模式对电路供电没有影响，所有寄存器状态和存储器数据在深度休眠模式中保持正常供电。

如果有外设需要使用高速时钟完成正在进行中的操作，例如 flash 的擦除写入，则深度休眠会推迟等待操作完成后再进入。

21.5.1 模式进入

设置 CPU 内核 System Control Register SLEEPDEEP 为 1，然后通过执行 WFI/WFE 指令进入深度休眠。

21.5.2 模式退出

IO 唤醒、CL 输出唤醒或 IWDG 定时唤醒信号可将芯片从深度休眠中唤醒。

外部复位或 IWDG 复位可复位全芯片，可以解除深度休眠状态。

Debug 操作可以将芯片从深度休眠模式中唤醒。

21.6 寄存器

21.6.1 地址分配

PMU 基地址为 0x40011720

表 21-2 功耗管理模块地址空间

名称	偏移	说明
AON_PWR_CFG	0x0	
AON_EVT_RCD	0x4	事件记录寄存器
AON_IO_WAKE_POL	0x8	IO 唤醒极性寄存器
AON_IO_WAKE_EN	0xC	IO 唤醒使能寄存器

21.6.2 AON_PWR_CFG 功耗管理配置寄存器

地址:0x4001_1720

复位值:0x0

表 21-3 AON_PWR_CFG 功耗管理配置寄存器

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

位置	位名称	说明
[31:2]		未使用
[1]	IOWK_FLT	IO 唤醒信号滤波使能，默认使能
[0]		未使用

软件可以选择对 IO 唤醒信号进行滤波或不滤波，如果启用片内滤波，则 IO 信号经过 120us 时间常数的滤波后，送入 PMU。使用片内 IO 信号滤波可以在一定的应用场景下代替板级滤波，但会引入 120us 左右的唤醒延时。如果不启用片内 IO 唤醒信号滤波，则应用场景需要保证 IO 唤醒输入信号稳定无干扰，可以考虑使用板级滤波，从而避免芯片被误唤醒造成不必要的功耗浪费。

21.6.3 AON_EVT_RCD 事件记录寄存器

地址:0x4001_1724

复位值:0x0

表 21-4 AON_EVT_RCD 事件记录寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		DEEPSLEEP	SLEEP			IWDG_WK	EVT_WK					IWDG_RST	KEY_RST		POR_RST
		RO	RO			RO	RO					RO	RO		RO
		0	0			0	0					0	0		1

位置	位名称	说明
[31:14]		未使用
[13]	DEEPSLEEP	深度休眠记录，高表示发生过
[12]	SLEEP	休眠记录，高表示发生过
[11:10]		未使用
[9]	IWDG_WK	IWDG 定时唤醒记录，高表示 Deep Sleep 休眠被 IWDG 定时唤醒
[8]	EVT_WK	IO 唤醒或 CLU 唤醒记录，高表示 Deep Sleep 休眠被 AON_IO_WAKE_EN 中对应的使能信号唤醒
[7:4]		未使用
[3]	IWDG_RST	独立看门狗复位发生记录，高表示发生过
[2]	KEY_RST_RCD	按键复位发生记录，高表示发生过
[1]		未使用
[0]	POR_RST_RCD	POR 复位发生记录，高表示发生过

向 AON_EVT_RCD 寄存器写入 0xCA40，清除全部事件记录。由于 EVT_RCD 工作于 LSI 时钟域，由软件写入密码到完成清除需要 32us。

21.6.4 AON_IO_WAKE_POL IO 唤醒极性寄存器

地址:0x4001_1728

复位值:0x0

表 21-5 AON_IO_WAKE_POL IO 唤醒源极性配置寄存器

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

位置	位名称	说明
[31:1]		未使用
[0]	WK_POL	IO 外部唤醒触发电平选择。1:高电平；0:低电平

所有 IO 唤醒信号使用同一个极性选择。

21.6.5 AON_IO_WAKE_EN IO 唤醒使能寄存器

地址:0x4001_172C

复位值:0x0

表 21-6 AON_IO_WAKE_EN IO 唤醒源使能寄存器

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

	CLUOUT3_EN	CLUOUT2_EN	CLUOUT1_EN	CLUOUT0_EN	P2_15_EN	P2_7_EN	P2_4_EN	P0_14_EN	P0_11_EN	P0_6_EN	P0_2_EN	P0_0_EN
	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
	0	0	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:12]		未使用
[11]	CLUOUT3_EN	CLUOUT3 作为唤醒使能
[10]	CLUOUT2_EN	CLUOUT2 作为唤醒使能
[9]	CLUOUT1_EN	CLUOUT1 作为唤醒使能
[8]	CLUOUT0_EN	CLUOUT0 作为唤醒使能
[7]	P2_15_EN	P2[15]外部唤醒使能
[6]	P2_7_EN	P2[7]外部唤醒使能
[5]	P2_4_EN	P2[4]外部唤醒使能
[4]	P0_14_EN	P0[14]外部唤醒使能
[3]	P0_11_EN	P0[11]外部唤醒使能
[2]	P0_6_EN	P0[6]外部唤醒使能
[1]	P0_2_EN	P0[2]外部唤醒使能
[0]	P0_0_EN	P0[0]外部唤醒使能

22 SIF 模块

22.1 简述

SIF 模块，适用于单线、单向通讯应用环境，例如车载仪表通讯。本章节描述了 SIF 的实现及使用方法。

22.2 主要特性

- 单线通讯
- 单向通讯，仅输出
- 支持同步信号、结束信号开启/关闭可配
- 支持同步信号、结束信号长度可配
- 支持同步信号、结束信号电平可配
- 支持 MSB/LSB 可配
- 支持数据占空比可配：2:1 和 3:1 两种模式
- 支持 DMA 传输

22.3 功能描述

22.3.1 功能框图

本接口采用同步串行设计，实现 MCU 同外部设备之间的 SIF 传输。支持轮询和中断方式获得传输状态信息。本接口的主要功能模块如下图所示：

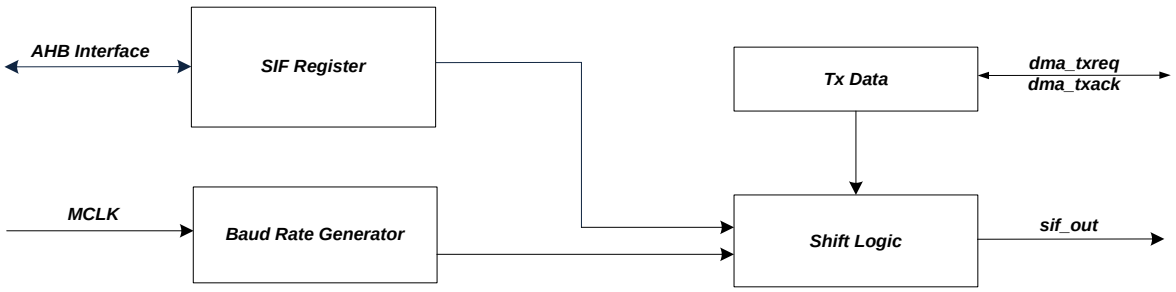


图 22-1 SIF 模块结构框图

SIF Register 模块，SIF 模块的相关寄存器。

Baud Rate Generator 模块，Tosc 时基产生电路。

Tx Data 模块，发送数据缓冲区。

Shift Logic 模块，数据发送模块。

22.3.2 功能说明

22.3.2.1 SIF 格式说明

SIF 通讯协议，分成几个部分：同步（SYNC），数据传输，结束（Done）和默认电平。

同步（SYNC）：一帧传输的开始波形，某些应用需要同步标记，某些不需要。

SYNC 分为有脉冲和无脉冲两个情况，有脉冲情况如下。无脉冲情况，SYNC 为持续电平。

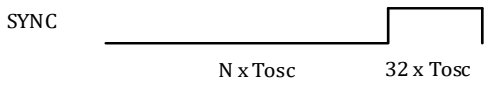


图 22-2 SIF 模块同步（SYNC）波形

输出传输：传输 0 和传输 1，两种状态。

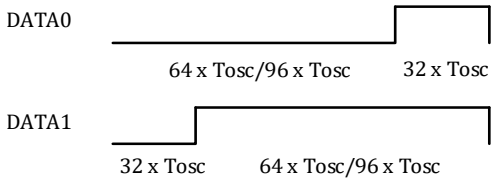


图 22-3 SIF 模块数据（DATA）波形

结束（Done）：一帧传输的结束波形，某些应用需要结束标记，某些不需要。结束信号没有翻转波形，按照应用需求，输出高/低电平。

22.3.2.2 Tosc 时基

Tosc 时基模块，产生基本时间计数单元。Tosc 时基的时钟来自系统时钟分频（分频后最高 3M），Tosc 最小时基为 333ns（去掉小数），最大时基约为 1.364ms（333ns*4096）。通过配置 SIF.TOSC 寄存器实现。

22.3.2.3 同步时间配置

同步（SYNC）信号时间的配置，同 SIF.STH1 寄存器相关。同步（SYNC）信号的默认电平在 SIF.CFG 寄存器中配置。

同步信号，默认为低电平的话，低电平时间最小为 1023*Tosc，高电平固定为 32*Tosc。

同步信号，默认为高电平的话，高电平时间最大为 1023*Tosc，低电平固定为 32*Tosc。

22.3.2.4 结束时间配置

结束（Done）信号时间的配置，同 SIF.DTH1 寄存器相关。结束（Done）信号的默认电平在 SIF.CFG 寄存器中配置。结束信号，默认电平可配，步长为 1ms，从 1ms—16ms 可选。结束时间是当前输出传输完毕后，到下一个数据发送的间隔时间。

22.3.2.5 DMA 传输

SIF 模块支持 DMA 传输。SIF 模块接收的数据宽度为一个字节（8-bit），因此，DMA 一轮搬移一个字节到 SIF 模块。传输轮数在 DMA 模块控制，一次 SIF 通讯包，由 N 个字节组成，则在 DMA 轮数寄存器总配置 N 即可。

22.4 寄存器

22.4.1 地址分配

SIF 模块寄存器的基地址是 0x4001_1500，模块寄存器列表如下。

表 22-1 SIF 模块控制寄存器列表

名称	偏移	说明
SIF0_CFG	0x00	SIF 配置寄存器
SIF0_TOSC	0x04	SIF TOSC 时基寄存器
SIF0_TSTH1	0x08	SIF 同步信号时间寄存器
SIF0_TDTH1	0x0C	SIF 结束信号时间寄存器
SIF0_IRQ	0x10	SIF 中断寄存器
SIF0_WDATA	0x14	SIF 传输数据寄存器

22.4.2 SIF0_CFG 配置寄存器

地址:0x4001_1500

复位值:0x0

表 22-2 地址寄存器 SIF0_CFG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								DONE	SYNC	IDLE	DONE_VLD	SYNC_VLD	RATIO	MSB	EN
								RW	RW	RW	RW	RW	RW	RW	RW

	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:8]		未使用
[7]	DONE	SIF 模块数据传输，完成信号默认电平 1：默认高电平 0：默认低电平
[6]	SYNC	SIF 模块数据传输，同步信号默认电平 1：默认高电平 0：默认低电平
[5]	SYNC_PULSE	SIF 模块数据传输，同步信号是否产生脉冲信号 1：同步信号有脉冲信号 0：同步信号无脉冲信号
[4]	DONE_VLD	SIF 模块数据传输，结束信号控制位 1：有结束信号 0：无结束信号
[3]	SYNC_VLD	SIF 模块数据传输，同步信号控制位 1：有同步信号 0：无同步信号
[2]	RATIO	SIF 模块数据占空比控制位 1：3:1 0：2:1
[1]	MSB	SIF 模块数据大小头控制位 1：MSB 0：LSB
[0]	EN	SIF 模块使能位 1：SIF 模块使能 0：SIF 模块关闭

22.4.3 SIF0_TOSC 时基寄存器

地址:0x4001_1504

复位值:0x0

表 22-3 SIF0_TOSC SIF 时基寄存器

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

位置	位名称	说明
[31:12]		未使用
[11:0]	OSC_TH	SIF 模块的时基 T_{osc} 设置 时基 T_{osc}= (OSC_TH+1)×(系统时钟周期×32)

22.4.4 SIF0_TSTH1 同步时间寄存器

地址:0x4001_1508

复位值:0x0

表 22-4 同步时间寄存器 SIF0_TSTH1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						TSTH1									
						RW									
						0									

位置	位名称	说明
[31:10]		未使用
[9:0]	T_{STH1}	SIF 模块同步信号周期 当 SYNC 无脉冲时，此周期即为整个 SYNC 时间宽度 当 SYNC 有脉冲时，此周期为 SYNC 脉冲前的时间宽度，SYNC 脉冲宽度固定为 32×Tosc Time = (T_{STH1}+1)×Tosc

22.4.5 SIF0_TDTH1 结束时间寄存器

地址:0x4001_150C

复位值:0x0

表 22-5 结束时间寄存器 SIF0_TDTH1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												TDTH1			
												RW			
												0			

位置	位名称	说明
[31:4]		未使用
[3:0]	TDTH1	SIF 模块结束信号周期 Time = (T_{DTH1}+1)×Tosc

22.4.6 SIF0_IRQ 中断寄存器

地址:0x4001_1510

复位值:0x0

表 22-6 中断寄存器 SIF0_IRQ

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

位置	位名称	说明
[31:8]		未使用
[7:5]		未使用
[4]	IRQ_IF	SIF 模块中断标志位，写 1 清零。 1：有 SIF 中断标志位 0：无 SIF 中断标志位
[3:2]		未使用
[1]	DMA_EN	SIF 模块 DMA 使能控制位 1：使能 0：关闭
[0]	IRQ_EN	SIF 模块中断使能控制位 1：使能中断 0：关闭中断

22.4.7 SIF0_WDATA 数据寄存器

地址:0x4001_1514

复位值:0x0

表 22-7 数据寄存器 SIF0_WDATA

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

	WDATA
	RW
	0

位置	位名称	说明
[31:8]		未使用
[7:0]	WDATA	SIF 模块数据寄存器

23 CL0 可配置逻辑

23.1 概述

可配置逻辑单元（Configurable Logic Unit 缩写为 CLU），提供了无需 CPU 干预操作的用户可配置的数字逻辑运算功能。可配置逻辑（CL）模块由四个独立的可配置逻辑单元（CLU）组成，支持用户可编程的异步和同步逻辑运算。CL 由于其灵活的逻辑功能，可以应用于多种数字功能实现，如替换系统胶合逻辑，生成特殊波形，或是同步系统事件的触发等。

23.2 主要特性

CL0 主要特性如下:

- 包含四个 CLU，具有 IO pin 与内部逻辑连接
- 每个 CLU 支持 256 种不同的组合逻辑功能（与、或、异或、多路复用等），还包含一个用于同步操作的时钟控制 D 触发器
- 每个 CLU 可以分别用于同步或者异步逻辑
- CLU 可以级联组合实现更复杂的逻辑功能
- 可以配合 SPI、I2C 或者 TIMER 与 MCPWM 进行操作
- 可以用于同步或者触发多个片上资源(ADC, UTIMTER 等)
- 异步逻辑输出可以用于唤醒操作

23.3 功能描述

23.3.1 功能框图

如图 23-1 所示，可配置逻辑 CL0 主要包括 4 个独立的可配置逻辑单元 CLU。每个 CLU 都可以独立配置为用户自定义的异步或同步数字逻辑功能，而无需 CPU 干预。许多内部和外部信号都可以作为各个 CLU 的输入，CLU 的输出也可以连接到 IO 输出或者直接连接到选中的外设输入。

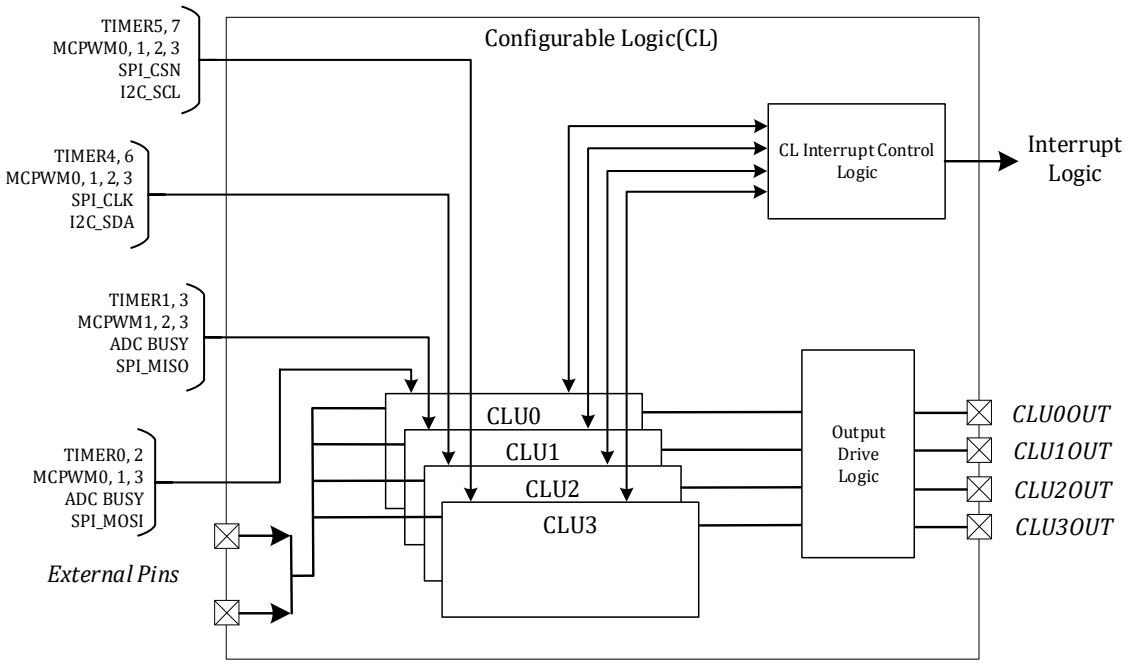


图 23-2 CL 模块顶层功能框图

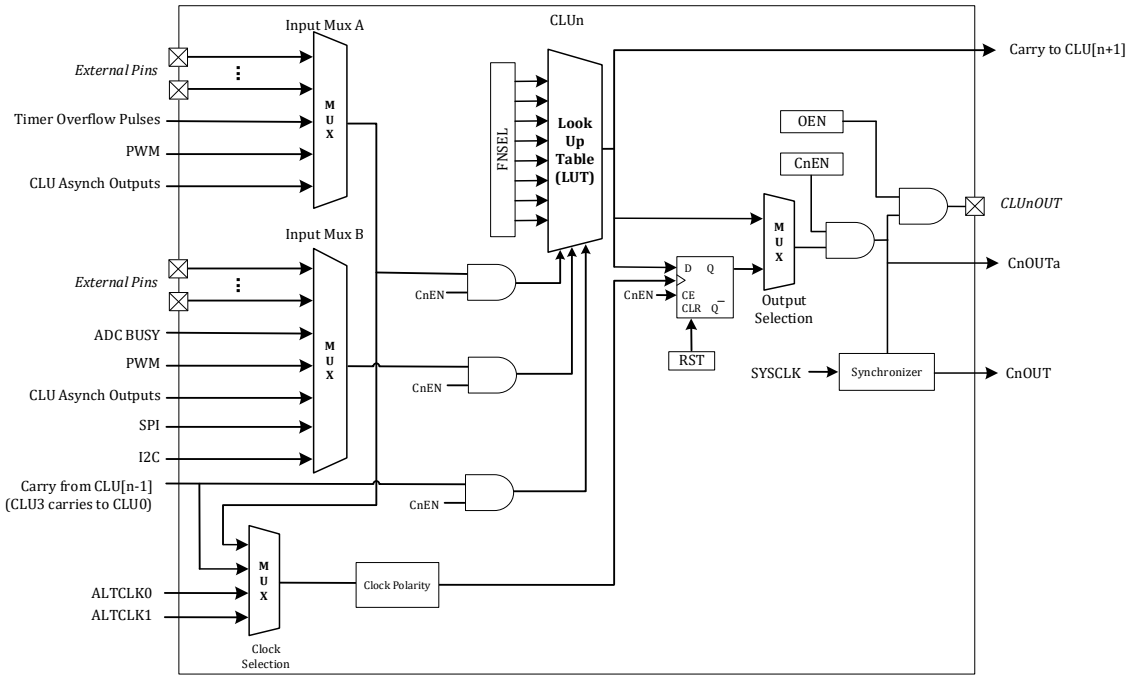


图 23-3 独立 CLU 模块功能框图

通过前述及 CL 模块的功能框图，可见 CL 的作用是，在无 CPU 干预下，若干内部和外部信号，可输入到各 CLU，通过“LUT（查找表）”的方法，得到自己想要的逻辑值结果，并输出到 IO 端口或直接用于外围设备的输入。

23.3.2 功能描述

23.3.2.1 配置流程

在使能独立 CLU 之前，应该先进行功能配置，输入复用选取和输出功能配置。CLU 初始化可以参考一下步骤：

1. 选择 C0_MX0 中到 LUT 的 A 与 B 输入
2. 配置 CL0_FN0 选择 LUT 功能
3. 配置 CLU 的 CL0_CF0 配置寄存器
4. 如果选择 CLU 的 D 触发器输出（OUTPSEL=1），则提前配置 RST=1 复位 D 触发的输出到 0
5. 配置 CL0_IE0 使能需要的中断信号。上升沿和下降沿中断分别通过 CnFIE 和 CnRIE 位使能
6. 配置 CL0_EN0 的 CnEN 使能对应的 CLUn，可以同时使能多个 CLU
7. 如果需要输出到 IO 端口，则使能 CL0_CF0 的对应 CnOEN 位

23.3.2.2 输入多路复选器

每个 CLU 都有两个主要的逻辑输入（A 与 B）和一个进位输入（C）。A 与 B 输入通过寄存器 CL0_MX0 的 CnMXA 和 CnMXB 进行选择，可以选择各内部和外部信号中的一个作为输入。当另一个 CLU 的输出被选为输入时，使用的是该 CLU 的异步输出，级联可以实现更复杂的逻辑功能。

Notes：当采用 Timer 溢出作为输入时，一次时钟溢出事件产生为一个系统时钟周期的高电平，其余时间为低电平。

进位输入 C 是前级 CLU 的异步输出。例如，CLU1 的进位输入 C 是 CLU0 的异步输出，CLU0 的进位输入 C 是 CLU3 的异步输出。

表 23-1 CLUnA 输入选择

CLUnMX.MXA	CLU0A	CLU1A	CLU2A	CLU3A
0000	CLU0OUT	CLU0OUT	CLU0OUT	CLU0OUT
0001	CLU1OUT	CLU1OUT	CLU1OUT	CLU1OUT
0010	CLU2OUT	CLU2OUT	CLU2OUT	CLU2OUT
0011	CLU3OUT	CLU3OUT	CLU3OUT	CLU3OUT
0100	Timer0 ch0 ovfl	Timer0 ch1 ovfl	Timer2 ch0 ovfl	Timer2 ch1 ovfl
0101	Timer1 ch0 ovfl	Timer1 ch1 ovfl	Timer3 ch0 ovfl	Timer3 ch1 ovfl
0110	PWM0	PWM2	PWM0	PWM1
0111	PWM1	PWM3	PWM2	PWM3
1000	P0.0	P0.0	P2.8	P2.8
1001	P0.3	P0.2	P1.13	P3.9
1010	P0.5	P0.5	P1.15	P1.15
1011	P0.7	P0.6	P1.10	P2.0
1100	P2.12	P2.12	P2.1	P2.1
1101	P0.12	P0.11	P2.4	P2.3
1110	P0.14	P0.14	P2.6	P2.6
1111	P1.0	P0.15	P2.14	P2.13

表 23-2 CLUnB 输入选择

CLUnMX.MXB	CLU0B	CLU1B	CLU2B	CLU3B
0000	CLU0OUT	CLU0OUT	CLU0OUT	CLU0OUT
0001	CLU1OUT	CLU1OUT	CLU1OUT	CLU1OUT
0010	CLU2OUT	CLU2OUT	CLU2OUT	CLU2OUT
0011	CLU3OUT	CLU3OUT	CLU3OUT	CLU3OUT

0100	ADBUSY0	ADBUSY0	PWM1	PWM0
0101	ADBUSY1	ADBUSY1	PWM3	PWM2
0110	PWM3	PWM1	I2CSDA	I2CSCL
0111	SPIMOSI	SPIMISO	SPICLK	SPICSN
1000	P0.2	P0.3	P3.9	P1.13
1001	P0.4	P0.4	P1.14	P1.14
1010	P0.6	P0.7	P2.0	P1.10
1011	P2.11	P2.11	P1.11	P1.11
1100	P0.11	P0.12	P2.3	P2.4
1101	P0.13	P0.13	P2.5	P2.5
1110	P0.15	P1.0	P2.13	P2.14
1111	P2.7	P2.7	P2.15	P2.15

注：

1、 CLU0OUT、CLU1OUT、CLU2OUT、CLU3OUT 分别代表 CLU 的 4 个输出端口；

2、 ADBUSY0 代表 ADC start 信号，即表示 ADC 正在采样；

3、 PWM0、PWM1、PWM2、PWM3 分别代表四个 PWM 模块的 TADC 事件触发信号（触发 ADC 采样）；

4、 SPIMOSI、SPIMISO、SPICLK、SPICSN 分别代表 SPI 模块的 MOSI、MISO、CLK、CS 端口；

5、 I2CSDA、I2CSCL 分别代表 I2C 模块的数据信号 SDA 和时钟信号 SCL；

6、TIMER0 ch0 ovfl、TIMER0 ch1 ovfl、TIMER1 ch0 ovfl、TIMER1 ch1 ovfl、TIMER2 ch0 ovfl、TIMER2 ch1 ovfl、TIMER3 ch0 ovfl、TIMER3 ch1 ovfl 代表 utimer 触发 ADC 采样事件（utimer 比较事件或者 utimer 的捕获事件）。

23.3.2.3 输出配置

每个 CLU 都有同步和异步输出，同步输出可以在任意时间通过寄存器 CLOUT0 输出，CLU 输出通过 CLUNCF 的 OUTSEL 控制是直接从 LUT 输出，还是从 DFF 输出。当 CLU 关闭时（CnEN 中的 CLEN0 等于 0），所有输出都维持在 0。

DFF 时钟通过 CLUnCF 的 CLKSEL 选择，共有四种来源。DFF 时钟可通过 bit CLKINV 进行翻转。每个 CLU 有下列四种 DFF 时钟选择：

1. CARRY_IN：前级的 CLU 的进位输出 C，第一个 CLU 使用最后一个 CLU 的进位；
2. MXA_INPUT：CLU 的 A 输入，通过寄存器 MXA 选择；
3. ALTCLK0：CLU0 的 Timer3 溢出，CLU1 的 Timer0 溢出，CLU2 的 Timer1 溢出，CLU3 的 Timer2 溢出。
4. ALTCLK1：CLU0 的 Timer1 溢出，CLU1 的 Timer2 溢出，CLU2 的 Timer3 溢出，CLU3 的 Timer0 溢出。

表 23-3 CLU 时钟/外部信号的时序

参数	符号	测试条件	Min	Typ	Max	Unit
传输延时	t_{DLY}	单个 CLU，连接到外部 pin	-	-	50	ns
		单个 CLU，内部连接	-	4	6	ns

时钟频率	f_{CLK}	1、2 或 3 个 CLU 级联	-	-	96	MHz
		4 个 CLU 级联	-	-	48	MHz

23.3.2.4 LUT 配置

每个 CLU 的逻辑功能由 LUT 配置决定，可以通过配置 CL0_FN0 的 CnFNSEL 来进行选取。CnFNSEL 的每位都被映射到 8 个多路选择器的输入的一种组合，也就是 LUT 的输出由 A，B 与 C 输入的组合选择。

表 23-4 LUT 真值表

A 输入	B 输入	C 输入	LUT 输出
0	0	0	FNSEL.0
0	0	1	FNSEL.1
0	1	0	FNSEL.2
0	1	1	FNSEL.3
1	0	0	FNSEL.4
1	0	1	FNSEL.5
1	1	0	FNSEL.6
1	1	1	FNSEL.7

参考上表所示的真值表，使用 LUT 可以实现三输入的所有逻辑功能。例如，实现与逻辑（A AND B），真值表的输出在 A 与 B 都为 1 时才为 1，也就是只有 FNSEL.7 与 FNSEL.6 为 1，所以 FNSEL 应该被写入 11000000b，也就是 0xC0.

23.4 寄存器

23.4.1 地址分配

CL0 在芯片中的基地址是 0x40011800，寄存器列表如下:

表 23-5 CL0 模块寄存器地址分配

名称	偏移	描述
CL0_EN0	0x00	CL0 使能寄存器
CL0_IE0	0x04	CL0 中断使能寄存器
CL0_IF0	0x08	CL0 中断标志寄存器
CL0_OUT0	0x0C	CL0 输出寄存器
CL0_MX0	0x10	CL0 输入多路复选寄存器
CL0_FN0	0x14	CL0 功能选择寄存器
CL0_CF0	0x18	CL0 配置寄存器

23.4.2 CL0_EN0 CL0 使能寄存器

地址: 0x40011800

复位值:0x0

表 23-6 CL0_EN0 CL0 使能寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												C3EN	C2EN	C1EN	C0EN
												RW	RW	RW	RW
												0	0	0	0

位置	位名称	说明
[31:4]		未使用
[3]	C3EN	CLU3 使能。默认关闭。 0: 关闭 1: 使能
[2]	C2EN	CLU2 使能。默认关闭。 0: 关闭 1: 使能
[1]	C1EN	CLU1 使能。默认关闭。 0: 关闭 1: 使能
[0]	C0EN	CLU0 使能。默认关闭。 0: 关闭 1: 使能

23.4.3 CL0_IE0 CL0 中断使能寄存器

地址: 0x40011804

复位值:0x0

表 23-7 CL0_IE0 CL0 中断使能寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								C3RIE	C3FIE	C2RIE	C2FIE	C1RIE	C1FIE	C0RIE	C0FIE
								RW	RW	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

位置	位名称	说明
[31:8]		未使用
[7]	C3RIE	CLU3 上升沿触发中断使能。默认关闭。 0: 关闭 1: 使能
[6]	C3FIE	CLU3 下降沿触发中断使能。默认关闭。 0: 关闭 1: 使能

[5]	C2RIE	CLU2 上升沿触发中断使能。默认关闭。 0: 关闭 1: 使能
[4]	C2FIE	CLU2 下降沿触发中断使能。默认关闭。 0: 关闭 1: 使能
[3]	C1RIE	CLU1 上升沿触发中断使能。默认关闭。 0: 关闭 1: 使能
[2]	C1FIE	CLU1 下降沿触发中断使能。默认关闭。 0: 关闭 1: 使能
[1]	C0RIE	CLU0 上升沿触发中断使能。默认关闭。 0: 关闭 1: 使能
[0]	C0FIE	CLU0 下降沿触发中断使能。默认关闭。 0: 关闭 1: 使能

23.4.4 CL0_IF0 CL0 中断标志寄存器

地址: 0x40011808

复位值:0x0

表 23-8 CL0_IF0 CL0 中断标志寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								C3RIF	C3FIF	C2RIF	C2FIF	C1RIF	C1FIF	C0RIF	C0FIF
								RW	RW	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

位置	位名称	说明
[31:8]		未使用
[7]	C3RIF	CLU3 上升沿中断标志位。中断标志高有效，写 1 清零。
[6]	C3FIF	CLU3 下降沿中断标志位。中断标志高有效，写 1 清零。
[5]	C2RIF	CLU2 上升沿中断标志位。中断标志高有效，写 1 清零。
[4]	C2FIF	CLU2 下降沿中断标志位。中断标志高有效，写 1 清零。
[3]	C1RIF	CLU1 上升沿中断标志位。中断标志高有效，写 1 清零。
[2]	C1FIF	CLU1 下降沿中断标志位。中断标志高有效，写 1 清零。
[1]	C0RIF	CLU0 上升沿中断标志位。中断标志高有效，写 1 清零。
[0]	C0FIF	CLU0 下降沿中断标志位。中断标志高有效，写 1 清零。

23.4.5 CL0_OUT0 CL0 输出寄存器

地址: 0x4001180C

复位值:0x0

表 23-9 CL0_OUT0 CL0 输出寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												C3OUT	C2OUT	C1OUT	C0OUT
												R	R	R	R
												0	0	0	0

位置	位名称	说明
[31:4]		未使用
[3]	C3OUT	CLU3 输出状态，与系统时钟同步。
[2]	C2OUT	CLU2 输出状态，与系统时钟同步。
[1]	C1OUT	CLU1 输出状态，与系统时钟同步。
[0]	C0OUT	CLU0 输出状态，与系统时钟同步。

23.4.6 CL0_MX0 CL0 输入多路复选寄存器

地址: 0x40011810

复位值:0x0

表 23-10 CL0_MX0 CL0 输入多路复选寄存器

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
C3MXA				C3MXB				C2MXA				C2MXB			
RW				RW				RW				RW			
0x0				0x0				0x0				0x0			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C1MXA				C1MXB				C0MXA				C0MXB			
RW				RW				RW				RW			
0x0				0x0				0x0				0x0			

位置	位名称	说明
[31:28]	C3MXA	CLU3 A 输入多路复用选择
[27:24]	C3MXB	CLU3 B 输入多路复用选择
[23:20]	C2MXA	CLU2 A 输入多路复用选择
[19:16]	C2MXB	CLU2 B 输入多路复用选择
[15:12]	C1MXA	CLU1 A 输入多路复用选择

[11:8]	C1MXB	CLU1 B 输入多路复用选择
[7:4]	C0MXA	CLU0 A 输入多路复用选择
[3:0]	C0MXB	CLU0 B 输入多路复用选择

23.4.7 CL0_FN0 CL0 功能选择寄存器

地址: 0x40011814

复位值:0x0

表 23-11 CL0_FN0 CL0 功能选择寄存器

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
C3FN								C2FN							
RW								RW							
0x00								0x00							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C1FN								C0FN							
RW								RW							
0x00								0x00							

位置	位名称	说明
[31:24]	C3FN	CLU3 查找表功能选择。
[23:16]	C2FN	CLU2 查找表功能选择。
[15:8]	C1FN	CLU1 查找表功能选择。
[7:0]	C0FN	CLU0 查找表功能选择。

23.4.8 CL0_CF0 CL0 配置寄存器

地址: 0x40011818

复位值:0x0

表 23-12 CL0_CF0 CL0 配置寄存器

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
C3OUTSEL	C3OEN			C3RST	C3CLKINV	C3CLKSEL		C2OUTSEL	C2OEN			C2RST	C2CLKINV	C2CLKSEL	
RW	RW			RW	RW	RW		RW	RW			RW	RW	RW	
0	0			0	0	0x0		0	0			0	0	0x0	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C1OUTSEL	C1OEN			C1RST	C1CLKINV	C1CLKSEL		C0OUTSEL	C0OEN			C0RST	C0CLKINV	C0CLKSEL	
RW	RW			RW	RW	RW		RW	RW			RW	RW	RW	

0	0		0	0	0x0	0	0		0	0	0x0

位置	位名称	说明
[31]	C3OUTSEL	CLU3 输出选取 0: D 触发器输出 1: LUT 输出
[30]	C3OEN	CLU3 输出使能。默认关闭。 0: 关闭 1: 使能
[29:28]		未使用
[27]	C3RST	CLU3 D 触发器复位。该位写 1 复位。读取恒为 0。
[26]	C3CLKINV	CLU3 D 触发器时钟电平反向。 0: 关闭 1: 使能
[25:24]	C3CLKSEL	CLU3 D 触发器时钟选择。 0x0: 进位输入 0x1: MXA 输入 0x2: ALTCLK0 0x3: ALTCLK1
[23]	C2OUTSEL	CLU2 输出选取 0: D 触发器输出 1: LUT 输出
[22]	C2OEN	CLU2 输出使能。默认关闭。 0: 关闭 1: 使能
[21:20]		未使用
[19]	C2RST	CLU2 D 触发器复位。该位写 1 复位。读取恒为 0。
[18]	C2CLKINV	CLU2 D 触发器时钟电平反向。 0: 关闭 1: 使能
[17:16]	C2CLKSEL	CLU2 D 触发器时钟选择 。 0x0: 进位输入 0x1: MXA 输入 0x2: ALTCLK0 0x3: ALTCLK1
[15]	C1OUTSEL	CLU1 输出选取 0: D 触发器输出 1: LUT 输出
[14]	C1OEN	CLU1 输出使能。默认关闭。 0: 关闭 1: 使能
[13:12]		未使用
[11]	C1RST	CLU1 D 触发器复位。该位写 1 复位。读取恒为 0。
[10]	C1CLKINV	CLU1 D 触发器时钟电平反向。 0: 关闭

		1: 使能
[9:8]	C1CLKSEL	CLU1 D 触发器时钟选择 。 0x0: 进位输入 0x1: MXA 输入 0x2: ALTCLK0 0x3: ALTCLK1
[7]	C0OUTSEL	CLU0 输出选取 0: D 触发器输出 1: LUT 输出
[6]	C0OEN	CLU0 输出使能。默认关闭。 0: 关闭 1: 使能
[5:4]		未使用
[3]	C0RST	CLU0 D 触发器复位。该位写 1 复位。读取恒为 0。
[2]	C0CLKINV	CLU0 D 触发器时钟电平反向。 0: 关闭 1: 使能
[1:0]	C0CLKSEL	CLU0 D 触发器时钟选择 。 0x0: 进位输入 0x1: MXA 输入 0x2: ALTCLK0 0x3: ALTCLK1

24 CAN

CAN（Controller Area Network）总线是一种可以在无主机情况下实现微处理器或者设备之间相互通信的总线标准。

24.1 主要特性

CAN 控制器遵循 CAN 总线支持 BOSCH 2.0A 和 2.0B 协议。2.0A 等效 CAN1.2，包含了 11 位 ID 格式；2.0B 包含了 11 位 ID 和 29 位 ID。

CAN 总线控制器可以处理总线上的数据收发，在本产品中，CAN 控制器具有 4 组筛选器。筛选器用于为应用程序选择要接收的消息。

CAN 控制器中，发送缓冲器（Tx Buffer）包含两组缓冲器：高优先级的主发送缓冲器（Primary Transmit Buffer，以下简称 PTB）和低优先级的辅发送缓冲器（Secondary Transmit Buffer，以下简称 STB）。PTB 包含一个 Slot，STB 包含两个 Slot，由硬件调度决定发送的顺序。接收缓冲器（Receive Buffer，以下简称 Rx Buffer）获取总线数据，Rx Buffer 包含十个 Slot。Slot 为存储 CAN 的单位（13-byte）

24.2 功能描述

24.2.1 功能框图

本接口采用同步串行设计，实现 MCU 同外部设备之间的 CAN 传输。支持轮询和中断方式获得传输状态信息。本接口的主要功能模块如下图所示。

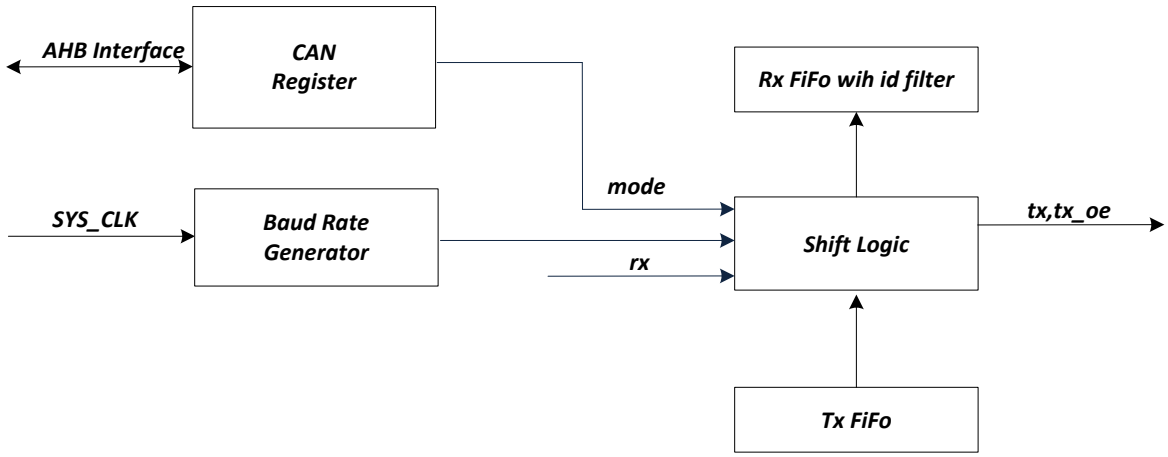


图 24-1 CAN 模块顶层功能框图

CAN 接口同外界通讯只有 tx 和 rx 两根信号线。tx 发送有效数据时候，tx_oe 有效。tx 不发送有效数据的时候，tx_oe 无效。

rx：数据信号。接收来自外界的 CAN 数据。

tx：数据信号。发送 CAN 数据到 CAN 总线。

tx_oe：数据使能信号。当 tx 输出时，tx_oe 有效；当 tx 无数据输出时，tx_oe 无效。

24.2.2 功能说明

CAN 模块接收和发送数据，并将数据从串行转换成并行，或并行转换成串行。可以开启或禁止中断。接口通过数据输出引脚(TX)和数据输入引脚(RX)连接到 CAN 总线的 PHY 芯片上。

24.2.2.1 工作模式

CAN 模块主要包含两个工作模式：正常工作模式和复位模式。

复位模式，时序参数、ID 配置、错误统计值等均在此模式下设定好。CAN_CFG_STAT. RESET 为 1，是复位模式。硬件复位结束后，CAN 模块处于复位模式下。

正常工作模式，CAN_CFG_STAT. RESET 为 0。此时，可以正常地响应 CAN 总线请求。

在上述两个模式下，扩展了监听模式（Listen Only）和自测试模式（Self Test）。前者，类似一个搜集器，只接收 CAN 总线上的数据，不发送任何数据。后者，是内部自测试，发送的数据同时被自己接收，检测内部功能是否正确。

24.2.2.2 传输

根据 SFF（Standard Frame Format），或 EFF（Extended Frame Format），搬移 11 个字节或者 13 个字节；完成后需要通过中断或者轮询的方式判断传输是否完成。

MCU 传输，推荐软件配置流程如下：

初始化 GPIO 模块，将 CAN 复用的 GPIO 配置完毕。

初始化 CAN 接口，控制寄存器配置完毕。

通过 CAN_TBUF 寄存器，写入 PTB/STB

触发 CAN 接口进入发送流程

24.2.2.3 中断处理

CAN 接口包含中断事件比较多，在 CAN_RTIE 和 CAN_ERRINT 寄存器描述中有相应说明。根据实际使用情况，开启对应的中断事件使能开关。

CAN 控制器提供了以下七种中断：

- 接收中断
- 发送中断
- 错误中断
- 数据溢出中断

- 被动错误中断
- 仲裁丢失中断
- 总线错误中断

24.2.2.3.1 接收中断

接收中断 CAN_RTIE.RIE，当 CAN Rx Buffer 接收到有效帧，CAN_RTIF.RIF 置位。接收中断进一步细化，还有三个中断源头 CAN_RTIE.ROIE、CAN_RTIE.RFIE 和 CAN_RTIE.RAFIE。分别是 CAN Rx Buffer 接收溢出中断，CAN Rx Buffer 接收满中断，和 CAN Rx Buffer 接收即将满中断。所以，产生一次接收中断，可能有几个标志位会置位。

只有先使能中断，才会置位标志位。

24.2.2.3.2 发送中断

CAN Tx Buffer 有两个源头，一个是 PTB，一个是 STB。对应的中断也是两个，CAN_RTIE.TPIE 和 CAN_RTIE.TSIE。另外，因本 CAN 设备支持发送终止传输，所以，有一个终止传输状态位（无中断使能标志）CAN_RTIF.AIF。

只有先使能中断，才会置位标志位。

24.2.2.3.3 错误中断

CAN 在传输过程中，发生总线错误事件，包括接收错误和发送错误。一般，产生错误，对应的错误计数器会递增，此时，根据计数器的值，将会产生不同类型的错误中断。

错误中断 CAN_RTIE.EIE。若 CAN_RECNT/CAN_TECNT 错误计数器的值从小于 CAN_ LIMIT.EWL 寄存器规定的值到超过 CAN_LIMIT.EWL 寄存器规定的值，或者反之；从非 BusOff 状态进入 BusOff 状态，或者反之，均会触发本中断，中断标志位是 CAN_RTIE.EIF。

被动错误中断 CAN_ERRINT.EPIE。若 CAN_RECNT/CAN_TECNT 错误计数器的值超过规定值（>127），本 CAN 设备从主动错误状态进入被动错误状态，或者从被动错误状态进入主动错误状态（此时错误计数器的值<127），均会触发本中断，中断标志位是 CAN_ERRINT.EPIF。

只有先使能中断，才会置位标志位。

24.2.2.3.4 数据溢出中断

当 CAN Rx Buffer 接收满，数据没有被 MCU 及时取走，若再接收到有效帧，此时，触发接收溢出中断 CAN_RTIE.ROIE，中断标志位是 CAN_RTIE.ROIF。

24.2.2.3.5 仲裁丢失中断

仲裁丢失中断 CAN_ERRINT.ALIE。CAN 总线支持仲裁机制，本 CAN 设备在总线竞争过程中，丢失了总线权利，将触发仲裁丢失中断。中断标志位是 CAN_ERRINT.ALIF。

只有先使能中断，才会置位标志位。

24.2.2.3.6 总线错误中断

总线错误中断 CAN_ERRINT.BEIE。CAN 在传输过程中，发生总线错误事件，将触发总线错误

中断。中断标志位是 CAN_ERRINT.BEIF。总线错误中断和错误中断的区别是，错误中断是总线错误积累到一定量后，才会触发。

只有先使能中断，才会置位标志位。

24.2.2.4 波特率设置

CAN 的波特率设定，主要依靠 CAN_SBAUD 寄存器完成。CAN_SBAUD 主要是配置 TQ 参数（TQ 的计算见 CAN_SBAUD 寄存器说明），采样点配置以及容差范围信息。

CAN_SBAUD.PRESC 设置传输基本时间单元参数 TQ：TQ = Tclk * (CAN_SBAUD.PRESC + 1)

LKS07x 时钟最快为 96M，对应的 Tclk 为 10.4ns，TQ 最大值为 2.6624us。

CAN_SBAUD 设置波特率；同时，可调节 BIT 信息中各个部分的宽度（TSEG1 和 TSEG2），找到理想采样点。

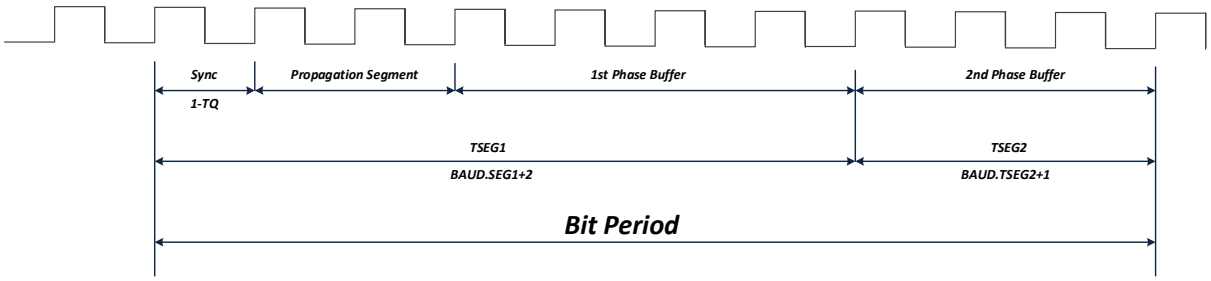


图 24-2 CAN 模块 Bit 周期介绍图

上图可知，TSEG1 包含三个部分：Sync，Propagation Segment 和 1st Phase Buffer。Sync 为此 BIT 的同步，固定 1 个 TQ。Propagation Segment 为传输延迟，1st Phase Buffer 为第一段相位缓冲。在具体 CAN 网络中，需要注意 Propagation Segment 这个参数，它决定了一个数据最少需要多长实际才能传递过来。同时，Propagation Segment 也直接影响了 SEG1 的配置。本设计将上述三者融合到一个参数：TSEG1。TSEG2 仅包含 2nd Phase Buffer 一个部分，其为第二相位缓冲，补偿边沿（Sync）阶段的误差，通过重新同步拉长或缩短。TSEG1 与 TSEG2 的长度决定了 CAN 数据的采样点，这种方式允许一定范围内的数据传输延迟和晶体的误差。其中，TSEG1 用来调整数据传输延迟时间造成的误差，而 TSEG2 则用来调整不同点节点晶体频率的误差。

SEG1 段时间计算公式：TSEG1 = TQ*(CAN_SBAUD.SEG1 + 2)

SEG2 段时间计算公式：TSEG2 = TQ*(CAN_SBAUD.SEG2 + 1)

波特率计算公式为：**CAN Baudrate = 1/(TSEG1 + TSEG2)**

CAN 通讯属于异步通讯，需要通过不断的重新同步才能保证收发节点的采样准确，所以，SJW（同步跳转宽度）决定了接收节点是否能有比较好的兼容性。CAN_SBAUD.SJW 是 SJW 位，其为重同步跳转寄存器。在一定波特率下通讯的各个设备，允许多大的通讯时间误差存在。

总线下界容差 < 总线波特率 < 总线上限容差

容差计算公式为：TQ * (SJW + 1)

配置 CAN_SBAUD 寄存器，一般可以参考如下几条：

- TSEG1 比 TSEG2 要略大点，此时采样点会略晚于中间点，保证信号的稳定。也可以根据实际情况调整，让采样点在此前后挪动。TSEG2 一般>=2（即 SEG2>=1）。BOSCH 规定 TSEG1+TSEG2>=8。一般芯片时钟快，但是波特率比较低的时候，可以通过组合 CAN_SBAUD.PRESC，CAN_SBAUD.SEG1 和 CAN_SBAUD.SEG2 实现。
- SJW 的值不能大于 SEG2，一般是后者的一半比较合理。
- 同一个 CAN 网络，波特率配置尽可能接近或一致，是比较好的选择。因各个芯片频率不同，CAN_SBAUD.PRESC 是不同的。波特率配置尽可能一致，是 CAN_SBAUD.PRESC 分频后产生的 TQ 尽可能一致。TQ 一致了，TSEG1 和 TSEG2 以及 SJW 就基本可以一致了。

基于 LKS07x 芯片，部分波特率索引值如下：

表 24-1 接收帧的结构

CAN 波特率	CAN_SBAUD.PRESC[7:0]	CAN_SBAUD.SJW[6:0]	CAN_SBAUD.SEG2[6:0]	CAN_SBAUD.SEG1[7:0]	Sample Point
1Mbps	0x05	0x02	0x05	0x08	63%
800Kbps	0x0B	0x01	0x03	0x04	60%
666Kbps	0x0B	0x01	0x03	0x06	67%
500Kbps	0x0B	0x02	0x05	0x08	63%
400Kbps	0x0B	0x02	0x06	0x0B	65%
250Kbps	0x17	0x02	0x05	0x08	63%
200Kbps	0x17	0x02	0x06	0x0B	65%
125Kbps	0x2F	0x02	0x05	0x08	63%
100Kbps	0x2F	0x02	0x06	0x0B	65%
80Kbps	0x35	0x02	0x06	0x0B	65%
50Kbps	0x5F	0x02	0x06	0x0B	65%
40Kbps	0x77	0x02	0x06	0x0B	65%
25Kbps	0xBF	0x02	0x06	0x0B	65%
20Kbps	0xEF	0x02	0x06	0x0B	65%
10Kbps	0xEF	0x06	0x0D	0x18	65%
05Kbps	0xEF	0x0D	0x1B	0x32	65%

24.2.2.5 ID 过滤器

CAN 总线，可以挂在很多设备。通过 ID 号，不同设备可以知道当前总线上发送的帧，是否需要被自己接收，或者发送出去的帧，是否有响应。本 CAN 设备，共计 4 组 ID 过滤器（ACF0/1/2/3）。只有当 CAN 设备处于复位模式时，MCU 才可以配置 ID 过滤器。CAN 帧的 ID 号有两种长度--11 位和 29 位。前者对应的 SFF（Standard Frame Format），后者对应 EFF（Extended Frame Format）。本 CAN 设备的 ID 滤波的长度均为 29 位。

如果当前接收到的帧，通过了其中一个 ID 过滤器，则它将被 CAN 设备接受。如果接受，此帧的数据将存储到 Rx Buffer 中，如果使能了 RIE，同时会置位 RIF。如果帧，不被接受，则不会置位 RIF 且 Rx Buffer 指针也不增加。未被接受的帧，将被丢弃，并被下一帧覆盖。任何存储的有效帧，都不会被任何未接受的帧覆盖。

通过 CAN_ACR 和 CAN_AMR 决定当前 CAN 设备可接收的 ID 范围。CAN_ACR 列出了一个特定的 ID，CAN_AMR 为对应的 MASK 寄存器，标识 CAN_ACR 中对应的位数据同接收到的 ID 对应的位的数据是否需要匹配。CAN_AMR 某一位是 0，表示这一位需要同 CAN_ACR 匹配；CAN_AMR 某一

位是 1，表示这一位不需要同 CAN_ACR 匹配。极端情况就是 CAN_ACR 上每一位都要匹配，或者每一位都不要匹配。

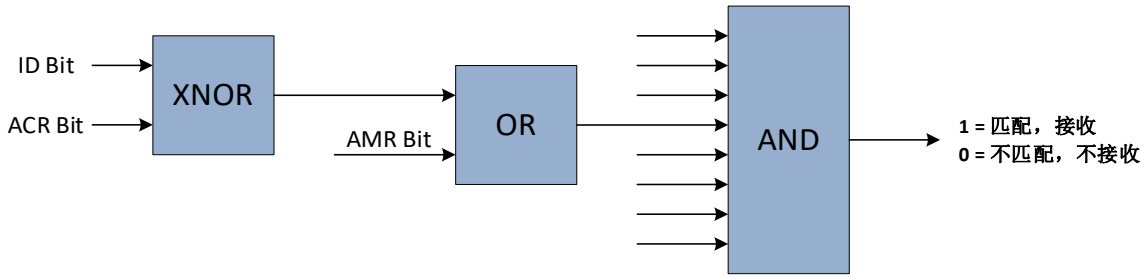


图 24-3 CAN 模块 ID 滤波逻辑关系

本 CAN 设备，通过 CAN_ACFCTRL、CAN_ACFEN 和 CAN_ACF 三组寄存器实现对 CAN_ACR0/1/2/3 和 CAN_AMR0/1/2/3 的配置。CAN_ACRx 和 CAN_AMRx 为一组，共计四组 ID 过滤器（ACF0/1/2/3）。CAN_ACFCTRL 为选择寄存器，选择上述哪组过滤器中将被 MCU 选中；CAN_ACFEN 为使能寄存器，选择上述哪组过滤器将被使能/关闭；CAN_ACRCTRL 选中 ACRx/AMRx，下一步通过访问 CAN_ACF，实现对 ACRx/AMRx 的读写。默认情况下，ACF0 使能，同时 CAN_ACR0 为全 0，CAN_AMR0 为全 1。

本 CAN 设备，CAN_ACRx 和 CAN_AMRx 均是按照最大 29-bit 设计，若是标准帧，ID 只有 11-bit，此时可以屏蔽 CAN_AMRx 对应高位（写 1）。

- 标准帧：CAN_ACRx[10:00]写入需要过滤的 ID 值；CAN_AMRx[28:11]写入全 1，CAN_AMRx[10:00]写入对应值。
- 扩展帧：CAN_ACRx[28:00]写入需要过滤的 ID 值；CAN_AMRx[28:00]写入对应值。

24.2.2.6 CAN 帧的接收

接收到的有效且被接受的帧，均存储到 Rx Buffer 中。一帧数据，存放到一个 Slot 中，本 CAN 设备，Rx Buffer 提供 10 个 Slot。Rx Buffer 按照 FIFO 模式存储和读出，即先存入 Rx Buffer 的帧，先被 MCU 读取。

- 如果使能了 CAN_RTIE.RIE，则每次接收到的有效且被接受的帧，CAN_RTIF.RIF 将置位为 1。
- 如果使能了 CAN_RTIE.RAFIE，则接收到的有效且被接受的帧的数量达到 CAN_LIMIT.AFWL 规定的值，CAN_RTIF.RAFIF 将置位为 1。
- 如果使能了 CAN_RTIE.RFIE，则接收到的有效且被接受的帧占满 Rx Buffer，CAN_RTIF.RFIF 将置位为 1。

通过 CAN_RBUF 寄存器来读取最早存入 Rx Buffer Slot 且没有被取走的帧。MCU 读取完毕后，将 CAN_RCTRL.RREL 设置为 1（硬件自动回零），释放已经读取的 Rx Buffer Slot。CAN_RBUF 寄存器自动指向下一个 Rx Buffer Slot。

哪怕 Rx Buffer 已满，本 CAN 设备仍然可以继续接收帧。一旦接收到的帧，是本 CAN 设备需要的（通过 ID 滤波器），根据 CAN_RCTRL.ROM 的配置，决定如何处理。CAN_RCTRL.ROM 为 1，此新接受的帧，被抛弃；CAN_RCTRL.ROM 为 0，最早进入 Rx Buffer 的帧，被抛弃。

CAN 帧的结构，分成 ID 部分和数据部分。第一个字节包含了帧分类等信息，确定是 SFF（标准）帧，还是 EFF（扩展）帧；确定是远程帧，还是数据帧；以及，确定数据长度。SFF 的 ID 长度为 2 个字节，EFF 的 ID 长度为 4 个字节。数据长度为最大 8 个字节。

表 24-2 接收帧的结构

SFF								
	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
CAN_RBUF0[31:24]	/							
CAN_RBUF0[23:16]	/							
CAN_RBUF0[15:08]	/					ID[10:08]		
CAN_RBUF0[07:00]	ID[07:00]							
CAN_RBUF1[31:24]	/							
CAN_RBUF1[23:16]	/							
CAN_RBUF1[15:08]	KOER[2:0]			TX	/			
CAN_RBUF1[07:00]	0	RTR	/		DLC[3:0]			
CAN_RBUF2[31:24]	DATA3							
CAN_RBUF2[23:16]	DATA2							
CAN_RBUF2[15:08]	DATA1							
CAN_RBUF2[07:00]	DATA0							
CAN_RBUF3[31:24]	DATA7							
CAN_RBUF3[23:16]	DATA6							
CAN_RBUF3[15:08]	DATA5							
CAN_RBUF3[07:00]	DATA4							
EFF								
	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
CAN_RBUF0[31:24]	/			ID[28:24]				
CAN_RBUF0[23:16]	ID[23:16]							
CAN_RBUF0[15:08]	ID[15:08]							
CAN_RBUF0[07:00]	ID[07:00]							
CAN_RBUF1[31:24]	/							
CAN_RBUF1[23:16]	/							
CAN_RBUF1[15:08]	KOER[2:0]			TX	/			
CAN_RBUF1[07:00]	1	RTR	/		DLC[3:0]			
CAN_RBUF2[31:24]	DATA3							
CAN_RBUF2[23:16]	DATA2							
CAN_RBUF2[15:08]	DATA1							
CAN_RBUF2[07:00]	DATA0							
CAN_RBUF3[31:24]	DATA7							
CAN_RBUF3[23:16]	DATA6							
CAN_RBUF3[15:08]	DATA5							
CAN_RBUF3[07:00]	DATA4							

IDE（IDentifier Extension）：0：标准帧格式；1：扩展帧格式

RTR（Remote Transmission Request）：0：数据帧；1：远程帧

DLC（Data Length Code）：数据长度码。设定范围为 0~8，对应数据长度为 0 Byte~8 Byte

KOER：同 CAN_EALCAP.KOER

TX：回环模式中接收自己发送的数据时此位置 1

CAN 帧的接收步骤如下：

- 配置 ID 滤波器
- 配置 CAN_RTIE.RFIE，CAN_RTIE.RAFIE 和 CAN_LIMIT.AFWL
- 等待 CAN_RTIF.RFIF 或者 CAN_RTIF.RAFIF（中断或者轮询皆可）
- 通过 CAN_RBUF 从 Rx Buffer 中读取最早接收到的 CAN 帧；将 CAN_RCTRL.RREL 设置为 1（硬件自动回零），释放已经读取的 Rx Buffer Slot，CAN_RBUF 寄存器自动指向下一个 Rx Buffer Slot。如此反复，直至通过 CAN_RCTRL.RSTAT 确认 Rx Buffer 为空。

24.2.2.7 CAN 帧的发送

CAN 模块要发送数据，必须在正常工作模式下。在开始任何传输之前，至少有一帧被写入传输缓冲区（Primary Transmit Buffer 或 Secondary Transmit Buffer）。CAN_TBUF 寄存器提供对 Primary Transmit Buffer（PTB）和 Secondary Transmit Buffer（STB）的访问。本 CAN 设备优先发送 PTB 的 CAN 帧。在发送空闲情况下，CAN_TCMD.TPE 配置为 1，即可触发 PTB 的数据发送，此时 STB 只能等待 PTB 发送完毕才能发送。

PTB 的存储深度是一，即只能存储一帧，STB 可以存储两帧。CAN_TCMD.TBSEL 为 1，选中 STB，为 0，选中 PTB。CAN_TCMD.TBSEL 为 1，MCU 可写入 STB。因可以存入两帧数据，CAN_TBUF 写入完成当前帧后，设置 CAN_TCTRL.TSNEXT，硬件自动跳转到下一个 STB Slot。所有 CAN_TBUF 的数据都可以按任意顺序写入。CAN_TCTRL.TSSTAT 可知道 STB 的状态。

CAN 帧的发送步骤如下：

- 配置 CAN_TCMD.TBSEL，选择发送哪个 Tx Buffer 的数据（0：PTB；1：STB）
- 将发送的数据，写入 CAN_TBUF 寄存器中
- 如果选择的是 STB，设置 TSNEXT=1 以完成 STB Slot 的装载
- CAN_TCMD.TPE 写 1，触发 PTB 的 CAN 帧的发送；CAN_TCMD.TSALL 或者 CAN_TCMD.TSONE 写 1，触发 STB 的 CAN 帧的发送
- 发送完成状态确认。CAN_RTIE.TPIE 使能，PTB 的 CAN 帧，发送完成后，CAN_RTIF.TPIF 置位；CAN_TCMD.TSONE 单帧模式，CAN_RTIE.TSIE 使能，STB 的 CAN 帧，发送完成后，CAN_RTIF.TSIF 置位；CAN_TCMD.TSALL 多帧模式，CAN_RTIE.TSIE 使能，STB 的所有 CAN 帧，发送完成后，CAN_RTIF.TSIF 置位
- 注意，若 STB 和 PTB 均触发的发送请求，同等条件下，PTB 先发送

本 CAN 设备发送 CAN 帧，支持 CAN 仲裁。一旦仲裁失败，可以等总线空闲重发此帧，也支持不再重发此帧。CAN_CFG_STAT. TPSS 配置 PTB 是否重发（0：重发；1：单次），CAN_CFG_STAT.

TSSS 配置 STB 是否重发（0：重发；1：单次）。配置为单次发送模式后，无论最后是否成功发送此帧，CAN_RTIF.TPIF/CAN_RTIF.TSIF 均会被置位，对应的 Slot 均会被清除（数据失效）。此时，需要借助另外两个状态位来判断：

- 有总线错误发生时，CAN_EALCAP.KOER 更新，CAN_ERRINT.BEIF 置位（CAN_ERRINT.BEIE 使能）
- 仲裁失败，CAN_ERRINT.ALIF 置位（CAN_ERRINT.ALIE 使能）

CAN 帧的最大有效负载长度为 8 字节。每帧的数据长度由 DLC 定义。对于远程帧（位 RTR），DLC 的值是无意义，因为远程帧的数据长度始终为 0 字节。

表 24-3 发送帧的结构

SFF								
	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
CAN_TBUF0[31:24]	/							
CAN_TBUF0[23:16]	/							
CAN_TBUF0[15:08]	/					ID[10:08]		
CAN_TBUF0[07:00]	ID[07:00]							
CAN_TBUF1[31:24]	/							
CAN_TBUF1[23:16]	/							
CAN_TBUF1[15:08]	/							
CAN_TBUF1[07:00]	0	RTR	/		DLC[3:0]			
CAN_TBUF2[31:24]	DATA3							
CAN_TBUF2[23:16]	DATA2							
CAN_TBUF2[15:08]	DATA1							
CAN_TBUF2[07:00]	DATA0							
CAN_TBUF3[31:24]	DATA7							
CAN_TBUF3[23:16]	DATA6							
CAN_TBUF3[15:08]	DATA5							
CAN_TBUF3[07:00]	DATA4							
EFF								
	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
CAN_TBUF0[31:24]	/			ID[28:24]				
CAN_TBUF0[23:16]	ID[23:16]							
CAN_TBUF0[15:08]	ID[15:08]							
CAN_TBUF0[07:00]	ID[07:00]							
CAN_TBUF1[31:24]	/							
CAN_TBUF1[23:16]	/							
CAN_TBUF1[15:08]	/							
CAN_TBUF1[07:00]	1	RTR	/		DLC[3:0]			
CAN_TBUF2[31:24]	DATA3							
CAN_TBUF2[23:16]	DATA2							
CAN_TBUF2[15:08]	DATA1							
CAN_TBUF2[07:00]	DATA0							
CAN_TBUF3[31:24]	DATA7							
CAN_TBUF3[23:16]	DATA6							
CAN_TBUF3[15:08]	DATA5							
CAN_TBUF3[07:00]	DATA4							

可以通过配置 CAN_TCMD.TPA 或者 CAN_TCMD.TSA 取消已请求但还没有被执行的数据发送。有如下几种情况，需注意：

- 仲裁中。节点仲裁失败，则取消数据发送；节点仲裁成功，则继续发送
- 数据发送中。成功发送数据且收到 ACK，对应的标志和状态正常置位，数据发送不取消；成功发送数据但没有收到 ACK，数据发送取消，错误计数器增加。CAN_TCMD.TSALL 发送，正在发送的 STB Slot 正常发送，没有开始发送的 STB Slot 被取消

24.2.2.8 错误管理

CAN 协议要求每个节点中都包含发送错误计数和接收错误计数。这两个错误计数的数值决定了控制器当前的错误状态（如主动错误、被动错误、离线）。控制器将数值分别存储在 CAN_RECNT 和 CAN_TECNT 中，MCU 可随时进行读取。除了错误状态之外，控制器还提供错误报警限制的功能（CAN_LIMIT.EWL），这个功能可在 CAN 控制器进入被动错误状态之前，提醒用户，当前发生的严重总线错误。

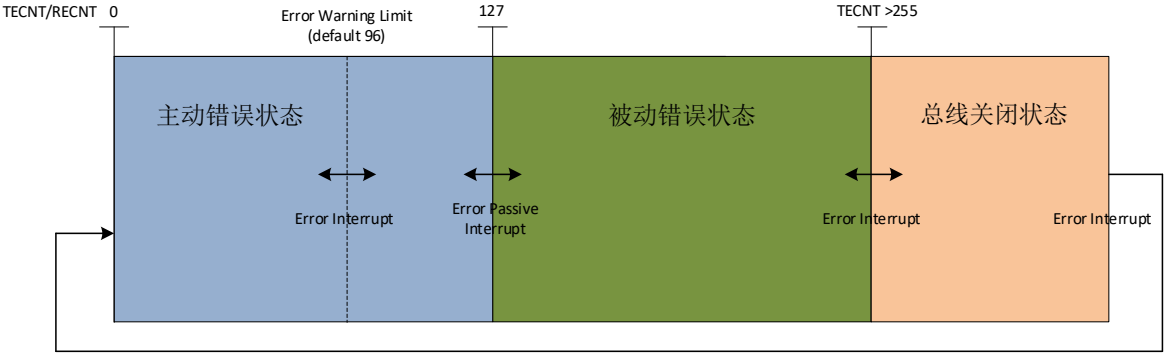


图 24-4 CAN 模块错误管理

24.2.2.9 错误计数

发送错误计数和接收错误计数根据下表规则递增/递减。注，一帧传输中可应用多个规则。

表 24-4 CAN 错误计数和接收错误计数规则

	接收和发送错误计数值的变动条件	发送错误计数值(TECNT)	接收错误计数值(RECNT)
1	接收单元检测出错误时 例外：接收单元在发送错误标志或过载标志中检测出“位错误”时，接收错误计数值不增加	-	+1
2	接收单元在发送完错误标志后检测到的第一个位为显性电平	-	+8
3	发送单元在输出错误标志时	+8	-
4	发送单元在发送主动错误标志或过载标志时，检测出位错误	+8	-
5	接收单元在发送主动错误标志或过载标志时，检测出位错误	-	+8
6	各单元从主动错误标志、过载标志的最开始检测出连续 14 个位的显性电平时，之后，每检测出连续的 8 个位的显性位时	发送时 +8	接收时 +8

7	检测出在被动错误标志后追加的连续 8 个显性位时	发送时 +8	接收时 +8
8	发送单元正常发送数据结束时（返回 ACK 并且到帧结束，也未检测出错误时）	TECNT=0 时， TECNT=0 TECNT-1	-
9	接收单元正常接收数据结束时（到 CRC 未检测出错误并且正常返回 ACK 时）	-	RECNT=0 时，RECNT=0 1<=RECNT<=127 时， RECNT-1 RECNT>127 时， RECNT=127
10	处于总线关闭状态的单元，检测到 128 次、连续 11 个隐性位	=0	=0

24.2.2.10 主动错误状态

当发送错误计数器（CAN_TECNT）和接收错误计数器（CAN_RECNT）的值都<=127 时，为主动错误状态；主动错误状态下的 CAN 设备，可以正常参与 CAN 总线通讯，并在检测到错误时能发出主动错误信号（连续 6 个显性信号）。

24.2.2.11 被动错误状态

当发送错误计数器（CAN_TECNT）或接收错误计数器（CAN_RECNT）的值>127 但<256 时，为被动错误状态；处于被动错误状态的 CAN 设备，检测到错误时，不能发出主动错误信号（连续 6 个显性信号）；只能发出被动错误信号（隐性信号），等待其它 CAN 设备发现错误。

24.2.2.12 关闭状态与关闭恢复

当发送错误计数器（CAN_TECNT）或接收错误计数器（CAN_RECNT）的值>255 时，本 CAN 设备自动进入关闭状态，从而不参与 CAN 总线的通信，直到返回到主动错误状态。可以通过 CAN_CFG_STAT.BUSOFF 位确认本 CAN 设备的状态。BUSOFF 被置位的同时，产生 CAN_RTIF.EIF 中断。

CAN 从关闭状态恢复到主动错误状态有以下两种方法：

- 上电复位
- 接收到连续 128 次、连续 11 位的隐性位序列（恢复序列）

节点关闭状态下，CAN_TECNT 值保持不变，CAN_RECNT 用于计数恢复序列。从节点关闭状态恢复后，CAN_TECNT 和 CAN_RECNT 被复位为 0。

24.2.2.13 仲裁

本 CAN 设备能够精确捕捉到仲裁失败的位置点，并反映到 CAN_EALCAP.ALC 寄存器中。CAN_EALCAP.ALC 寄存器中保存着，最近一次仲裁失败的位置点，如果本 CAN 设备赢得仲裁，则 CAN_EALCAP.ALC 位不更新。

CAN_ALC 值定义如下：

SOF 位后，第一个 ID 数据位 CAN_EALCAP.ALC 为 0，第二个 ID 数据位 CAN_EALCAP.ALC 为 1，依次类推。因为仲裁只发生在仲裁场内，所以 CAN_EALCAP.ALC 的最大值为 31。比如一个标准格式远程帧和一个扩展帧仲裁，扩展帧将在 IDE 位争抢总线失败，则 CAN_EALCAP.ALC=12。

24.2.3 寄存器

24.2.3.1 地址分配

CAN 模块寄存器的基地址是 0x4001_1300，寄存器列表如下。

表 24-5 CAN 寄存器地址分配

名称	偏移	说明
CAN_RBUF0	0x00	CAN Rx Buffer 寄存器 0
CAN_RBUF1	0x04	CAN Rx Buffer 寄存器 1
CAN_RBUF2	0x08	CAN Rx Buffer 寄存器 2
CAN_RBUF3	0x0C	CAN Rx Buffer 寄存器 3
CAN_TBUF0	0x50	CAN Tx Buffer 寄存器 0
CAN_TBUF1	0x54	CAN Tx Buffer 寄存器 1
CAN_TBUF2	0x58	CAN Tx Buffer 寄存器 2
CAN_TBUF3	0x5C	CAN Tx Buffer 寄存器 3
CAN_CFG_STAT	0xA0	CAN 配置和状态寄存器
CAN_TCMD	0xA1	CAN 发送命令寄存器
CAN_TCTRL	0xA2	CAN 发送控制寄存器
CAN_RCTRL	0xA3	CAN 接收控制寄存器
CAN_RTIE	0xA4	CAN 发送接收中断控制寄存器
CAN_RTIF	0xA5	CAN 发送接收中断标志寄存器
CAN_ERRINT	0xA6	CAN 错误中断使能和标志寄存器
CAN_LIMIT	0xA7	CAN 警告寄存器
CAN_SBAUD	0xA8	CAN 波特率配置寄存器
CAN_EALCAP	0xB0	CAN 错误信息和丢失仲裁信息记录寄存器
CAN_RECNT	0xB2	CAN 接收错误计数器寄存器
CAN_TECNT	0xB3	CAN 发送错误计数器寄存器
CAN_ACFCTRL	0xB4	CAN ID 滤波器控制寄存器
CAN_ACFEN	0xB6	CAN ID 滤波器使能寄存器
CAN_ACF	0xB8	CAN ID 滤波器选择寄存器

24.2.3.2 寄存器说明

24.2.3.2.1 CAN_RBUF0/1/2/3 寄存器

地址：0x4001_1300/4/8/C

复位值：X

表 24-6 读取寄存器 CAN_RBUF0/1/2/3

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

X

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

位置	位名称	说明
[31:0]	RDATA	接收寄存器，从 Rx Buffer Slot 中读出 CAN 帧的数据

不同帧的类型及数据长度不同，CAN_RBUF0/1/2/3 的数据内容也不同。根据文档对接收帧的结构解析即可。

24.2.3.2.2 CAN_TBUF0/1/2/3 寄存器器

地址：0x4001_1350/4/8/C

复位值：X

表 24-7 写入寄存器 CAN_TBUF0/1/2/3

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

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

位置	位名称	说明
[31:0]	WDATA	发送寄存器，写入数据到 Tx Buffer Slot 中（PTB 或者 STB）

不同帧的类型及数据长度不同，CAN_TBUF0/1/2/3 的数据内容也不同。根据文档对发送帧的结构解析即可。不用填写的位，补 0 即可。

24.2.3.2.3 CAN_CFG_STAT 配置和状态寄存器

地址：0x4001_13A0

复位值：0x0

表 24-8 配置和状态寄存器 CAN_CFG_STAT

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

	RESET	LBME	LBMI	TPSS	TSSS	RACTIVE	TACTIVE	BUSOFF
	RW	RW	RW	RW	RW	RO	RO	RW
	1	0	0	0	0	0	0	0

位置	位名称	说明
[7]	RESET	1：CAN 模块，复位模式 0：CAN 模块，正常工作模式 复位模式下，CAN 模块部分电路处于复位状态，部分寄存器只能在复位模式下配置；正常工作模式下，可以收发 CAN 帧。 注意，从复位模式切换到正常工作模式，需要等待 11 个 CAN Bit Time 的时间（例如波特率为 500K，一个 Bit Time 就是 2us，需要等待 22us 才能发送或者接收 CAN 帧）
[6]	LBME	外部回环模式使能位 1：使能 0：关闭 注意：正常通信中，禁止使能该位
[5]	LBMI	内部回环模式使能位 1：使能 0：关闭 注意：正常通信中，禁止使能该位
[4]	TPSS	PTB 单次传输模式使能位 1：使能 0：关闭
[3]	TSSS	STB 单次传输模式使能位 1：使能 0：关闭
[2]	RACTIVE	接收状态标志位 0：本 CAN 设备，接收空闲，总线空闲 1：本 CAN 设备，正在接收 CAN 帧
[1]	TACTIVE	发送状态标志位 0：本 CAN 设备，发送空闲 1：本 CAN 设备，正在发送 CAN 帧
[0]	BUSOFF	本 CAN 设备总线关闭状态位（Bus Off） 0：本 CAN 设备，不在总线关闭状态位 1：本 CAN 设备，处于总线关闭状态位 注意：BUSOFF 写 1 可以清零 TECNT 和 RECNT 寄存器，但仅限于项目调试

24.2.3.2.4 CAN_TCMD 发送命令寄存器

地址：0x4001_13A1

复位值：0x0

表 24-9 发送命令寄存器 CAN_TCMD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								TBSEL	LOM	-	TPE	TPA	TSONE	TSALL	TSA
								RW	RW	--	RO	RO	RO	RO	RO
								0	0	0	0	0	0	0	0

位置	位名称	说明
[7]	TBSEL	Tx Buffer 选择位，通过 CAN_TBUF 写入到对应 Buffer 中 1：STB 0：PTB 注意在写入 CAN_TBUF 过程中，TBSEL 的值不能改变；同时，在写入 STB 的时候，完成一帧的写入，需要更新 TSNEXT。
[6]	LOM	监听模式使能位 1：使能 0：关闭 如果设置了 TPE、TSONE 或 TSALL，则无法设置 LOM。如果 LOM 启用且 LBME 禁用，则无法启动传输。 LOM=1 和 LBME=0 禁止所有传输。 LOM=1 和 LBME=1 禁止应答相应接收到的帧和错误帧，但可以发送本 CAN 设备的帧 注意：正常通信中，禁止使能该位
[5]	--	保持默认值
[4]	TPE	PTB 发送使能位 1：触发 PTB 发送 0：不触发 PTB 发送 配置 TPE，在下一个可以发送的窗口，优先 STB 发送 PTB 的数据（除非当前已经在发送 STB，否则先发 PTB）。 该位写 1 后将保持为 1 直到 PTB 发送完成或者通过 TPA 取消发送。 软件不能通过写 0 清除该位。TPE 可以被清除的另外几种情况： 1. CAN_CFG_STAT.RESET=1 2. CAN_TCMD.LOM=1 同时 CAN_CFG_STAT.LBME=0
[3]	TPA	PTB 发送取消位 1：取消已经通过 TPE 置 1 请求但还未开始的 PTB 发送 0：不取消 软件置位，硬件自动清零。设置 TPA 会自动清零 TPE，一般 TPA 和 TPE 不同时置位。CAN_CFG_STAT.RESET=1 可清零 TPA。

[2]	TSONE	发送一帧 STB 数据控制位（Transmit Secondary ONE Frame） 1：发送一帧 STB 数据 0：不发送 TSONE 配置为 1，触发发送，成功发送或者 TSA 终止发送可实现硬件清零，否则一直保持 1。CAN_CFG_STAT.RESET=1，也可清零。
[1]	TSALL	发送多帧 STB 数据控制位（Transmit Secondary All Frames） 1：发送多帧 STB 数据 0：不发送 TSALL 配置为 1，触发发送，成功发送或者 TSA 终止发送可实现硬件清零，否则一直保持 1。CAN_CFG_STAT.RESET=1，也可清零。
[0]	TSA	STB 发送取消位 1：取消已经通过 TSONE 或者 TSALL 置 1 请求但还未开始的 STB 发送，软件写 1，硬件清零。写 1 可以清零 TSONE 或者 TSALL 位 0：不取消 一般 TSA 和 TSONE/TSALL，不同时置位。CAN_CFG_STAT.RESET=1 可清零 TSA

24.2.3.2.5 CAN_TCTRL 发送控制寄存器

地址：0x4001_13A2

复位值：0x0

表 24-10 发送控制寄存器 CAN_TCTRL

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

位置	位名称	说明
[7]	-	保持默认值
[6]	TSNEXT	更新 STB Slot 位置 1：更新 0：不更新 当前的 STB Slot 填满数据后，软件置位 TSNEXT，硬件自动指向下一个 STB Slot，此时，TSONE/TSALL 置位，即可发送被填满的 STB Slot。可以同时设置 TSNEXT 和 TSONE 或 TSALL。如果 TBSEL=0，则设置 TSNEXT 没有意义，TSNEXT 的配置将被忽略并自动清除。如果 STB 的所有 Slot 都已填满，则 TSNEXT 将保持置位状态，直到 Slot

		有空闲，才会被硬件清零。
[5:2]	-	保持默认值
[1:0]	TSSTAT	STB 状态位 3：STB 满 2：STB 大于半满 1：STB 小于等于半满 0：STB 空

24.2.3.2.6 CAN_RCTRL 接收控制寄存器

地址：0x4001_13A3

复位值：0x0

表 24-11 接收控制寄存器 CAN_RCTRL

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								SACK	ROM	ROV	RREL	RBALL		RSTAT	
								RW	RW	RW	RW	RW		RO	
								0	0	0	0	0		0	

位置	位名称	说明
[7]	SACK	自应答 1：LBME=1 时，使能自应答功能 0：无自应答
[6]	ROM	Rx Buffer 满，溢出控制位 1：新接收到的数据不被存储 0：最早接收到的数据被覆盖
[5]	ROV	Rx Buffer 溢出标志位 1：溢出，最少有一帧数据丢失。通过写 RREL 为 1 清零 0：无溢出
[4]	RREL	释放 Rx Buffer Slot 1：软件已经取走当前 Rx Buffer Slot 数据，可释放当前 Rx Buffer Slot，硬件指向下一个 Rx Buffer Slot 0：无释放操作
[3]	RBALL	接收 Rx Buffer 存储通过 ID 滤波的帧 1：存储所有的 CAN 帧，包括错误的帧 0：正常模式，仅存储正确的 CAN 帧
[2]	--	保持默认值
[1:0]	RSTAT	Rx Buffer 状态位

		3：满（溢出状态下，保持此值） 2：Rx Buffer 已经接收的帧的数量大于等于 AFWL 的值，但未满 1：Rx Buffer 已经接收的帧的数量小于 AFWL 的值 0：Rx Buffer 已经接收的帧的数量为 0，空

24.2.3.2.7 CAN_RTIE 发送接收中断控制寄存器

地址：0x4001_13A4

复位值：0x0

表 24-12 发送接收中断控制寄存器 CAN_RTIE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								RIE	ROIE	RFIE	RAFIE	TPIE	TSIE	EIE	TSFF
								RW	RW	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

位置	位名称	说明
[7]	RIE	接收中断使能（Receive Interrupt Enable） 1：使能 0：禁止
[6]	ROIE	接收溢出中断使能（Receive Overrun Interrupt Enable） 1：使能 0：禁止
[5]	RFIE	Rx Buffer 满中断使能（Rx Buffer Full Interrupt Enable） 1：使能 0：禁止
[4]	RAFIE	Rx Buffer 将满中断使能（Rx Buffer Almost Full Interrupt Enable） 1：使能 0：禁止
[3]	TPIE	PTB 发送中断使能（Transmission Primary Interrupt Enable） 1：使能 0：禁止
[2]	TSIE	STB 发送中断使能（Transmission Secondary Interrupt Enable） 1：使能 0：禁止
[1]	EIE	错误中断使能（Error Interrupt Enable） 1：使能 0：禁止
[0]	TSFF	Tx Buffer 标志位 1：STB Slot 被全部填满 0：STB Slot 没有被全部填满

24.2.3.2.8 CAN_RTIF 发送接收中断标志寄存器

地址：0x4001_13A5

复位值：0x0

表 24-13 发送接收中断标志寄存器 CAN_RTIF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								RIF	ROIF	RFIF	RAFIF	TPIF	TSIF	EIF	AIF
								RW	RW	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

位置	位名称	说明
[7]	RIF	接收中断标志位（Receive Interrupt Flag） 1：接收到有效帧（数据帧或者远程帧），写 1 清零 0：没有接收到有效帧
[6]	ROIF	接收溢出中断标志位（Receive Overrun Interrupt Flag） 1：Rx Buffer 至少有一帧未读走的数据被覆盖 0：Rx Buffer 无覆盖 溢出时 ROIF 和 RFIF 同时置 1，写 1 清零
[5]	RFIF	Rx Buffer 满中断标志位（Rx Buffer Full Interrupt Flag） 1：Rx Buffer 满，写 1 清零 0：Rx Buffer 未满
[4]	RAFIF	Rx Buffer 将满中断标志位（Rx Buffer Almost Full Interrupt Flag） 1：被填充的 Rx Buffer Slot 数目大于等于 AFWL 设定值 0：被填充的 Rx Buffer Slot 数目小于 AFWL 设定值
[3]	TPIF	PTB 发送中断标志（Transmission Primary Interrupt Flag） 1：触发 PTB 发送，发送成功完成。写 1 清零 0：无 PTB 发送请求，无完成标志
[2]	TSIF	STB 发送中断标志（Transmission Secondary Interrupt Flag） 1：触发 STB 发送，发送成功完成。写 1 清零 0：无 STB 发送请求，无完成标志
[1]	EIF	错误中断标志（Error Interrupt Flag） 1：错误计数器的值发生变化，大于或者小于错误警告寄存器的设定值。写 1 清零 0：无 错误计数器的值从小于错误警告寄存器的设定值变为大于设定值，或者从大于设定值变为小于设定值，均会触发中断标志。另外，进入 Bus Off 或从 Bus Off 退出，也会触发。
[0]	AIF	取消发送中断标志（Abort Interrupt Flag） 1：通过 TPA 和 TSA 请求的发送消息被成功取消。写 1 清零 0：未取消发送数据 AIF 没有对应的使能寄存器

注：对应使能寄存器使能后，中断标志位才会置位；进 AIF 是例外。

24.2.3.2.9 CAN_ERRINT 错误中断使能和标志寄存器

地址：0x4001_13A6

复位值：0x0

表 24-14 错误中断使能和标志寄存器 CAN_ERRINT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								EWARN	EPASS	EPIE	EPIF	ALIE	ALIF	BEIE	BEIF
								RW	R	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

位置	位名称	说明
[7]	EWARN	错误计数值超限中断标志位 1：RECNT 或者 TECNT 大于等于 EWL 设定值，写 1 清零 0：RECNT 或者 TECNT 小于 EWL 设定值
[6]	EPASS	CAN 设备处于被动错误状态 1：CAN 设备处于被动错误状态 0：CAN 设备处于主动错误状态
[5]	EPIE	被动错误中断使能（Error Passive Interrupt Enable） 1：使能 0：禁止
[4]	EPIF	被动错误中断标志（Error Passive Interrupt Flag） 1：发生主动错误到被动错误（或者相反）的改变，写 1 清零 0：未发生
[3]	ALIE	仲裁失败中断使能（Arbitration Lost Interrupt Enable） 1：使能 0：禁止
[2]	ALIF	仲裁失败中断标志位（Arbitration Lost Interrupt Flag） 1：仲裁失败，写 1 清零 0：仲裁成功
[1]	BEIE	总线错误中断使能（Bus Error Interrupt Enable） 1：使能 0：禁止
[0]	BEIF	总线错误中断标志（Bus Error Interrupt Flag） 1：总线错误，写 1 清零 0：无总线错误

24.2.3.2.10 CAN_LIMIT 错误&警告门限值寄存器

地址：0x4001_13A7

复位值：0x1B

表 24-15 错误&警告门限值寄存器 CAN_LIMIT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								AFWL				EWL			
								RW				RW			
								0x1				0xB			

位置	位名称	说明
[7:4]	AFWL	Rx Buffer 快满警告配置值 AFWL 同 Rx Buffer 实际接收到的数量进行比对，实际数量超过 AFWL，触发 RAFIF。 AFWL=0，无意义，硬件强制配置为 1；AFWL 超过 Rx Buffer 实际容量，无意义，硬件强制配置为 Rx Buffer 实际容量
[3:0]	EWL	可编程错误警告限值=（EWL+1）*8。可能的极限值：8，16，...128。 EWL 的值控制 EIF。

24.2.3.2.11 CAN_SBAUD 波特率配置寄存器

地址：0x4001_13A8

复位值：0x0

表 24-16 波特率配置寄存器 CAN_ SBAUD

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	S_SEG_2							S_SEG_1							
	RW							RW							
	0							0							

位置	位名称	说明
[31:24]	S_PRESC	TQ 预分频率设定值（S_Prescaler），主要配置 TQ 的大小。 TQ = system clock period *(S_PRESC+1)
[23]	--	保持默认值
[22:16]	S_SJW	再同步补偿宽度时间设定（Bit Timing Segment 2）

		再同步补偿宽度时间=(S_SJW+1)*TQ
[15]	--	保持默认值
[14:08]	S_SEG_2	段 2 时间单元设定（Bit Timing Segment 2） 段 2 时间=(S_SEG_2+1)*TQ
[07:00]	S_SEG_1	段 1 时间单元设定（Bit Timing Segment 1） 段 1 时间=(S_SEG_2+2)*TQ

24.2.3.2.12 CAN_EALCAP 错误信息和丢失仲裁信息记录寄存器

地址：0x4001_13B0

复位值：0x0

表 24-17 错误信息和丢失仲裁信息记录寄存器 CAN_EALCAP

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

位置	位名称	说明
[7:5]	KOER	错误类别（Kind Of Error） 0：无错误 1：位错误 2：形式错误 3：填充错误 4：应答错误 5：CRC 错误 6：其他错误 7：保留 有错误时 KOER 位更新，正常发送接收时 KOER 位保持不变。
[4:0]	ALC	仲裁失败位置捕捉（Arbitration Lost Capture） 仲裁失败时 ALC 记录一帧数据中仲裁失败时的位置

24.2.3.2.13 CAN_RECNT 接收错误计数器寄存器

地址：0x4001_13B2

复位值：0x0

表 24-18 接收错误计数器寄存器 CAN_ RECNT

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

	R
	0

位置	位名称	说明
[7:0]	RECNT	接收错误计数器（Receive Error Count） 接收错误计数器根据 CAN 协议规定的错误计数增加或者减少。该计数器不存在上溢，255 为最大值

24.2.3.2.14 CAN_TECNT 发送错误计数器寄存器

地址：0x4001_13B3

复位值：0x0

表 24-19 发送错误计数器寄存器 CAN_ TECNT

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

位置	位名称	说明
[7:0]	TECNT	发送错误计数器（Transmit Error Count） 发送错误计数器根据 CAN 协议规定的错误计数增加或者减少。该计数器不存在上溢，255 为最大值

24.2.3.2.15 CAN_ACFCTRL ID 滤波器控制寄存器

地址：0x4001_13B4

复位值：0x0

表 24-20 ID 滤波器控制寄存器 CAN_ ACFCTRL

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

位置	位名称	说明
[7:6]	--	保持默认值
[5]	SELMASK	选择 ID 滤波器的屏蔽寄存器（Select Acceptance MASK） 1：ACF 指向 ID 滤波器 AMR 寄存器（MASK） 0：ACF 指向 ID 滤波器 ACR 寄存器 通过 ACFADR 选择具体的筛选寄存器组
[4]	--	保持默认值
[3:0]	ACFADR	ID 滤波器地址（Acceptance Filter Address） ACFADR 指向具体的 ID 滤波器器，通过 SELMASK 去区分 ACR 和 AMR。 0：指向 ACF_0 1：指向 ACF_1 2：指向 ACF_2 3：指向 ACF_3 其它值，无效

24.2.3.2.16　CAN_ACFEN ID 滤波器使能寄存器

地址：0x4001_13B6

复位值：0x0

表 24-21 ID 滤波器使能寄存器 CAN_ACFEN

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												AE_3	AE_2	AE_1	AE_0
												RW	RW	RW	RW
												0	0	0	0

位置	位名称	说明
[31:8]		未使用
[3]	AE_3	ACF_3 使能（Acceptance Filter 4 Enable） 1：使能 0：禁止
[2]	AE_2	ACF_2 使能（Acceptance Filter 3 Enable） 1：使能 0：禁止
[1]	AE_1	ACF_1 使能（Acceptance Filter 2 Enable） 1：使能 0：禁止
[0]	AE_0	ACF_0 使能（Acceptance Filter 1 Enable） 1：使能 0：禁止

24.2.3.2.17 CAN_ACF ID 滤波器选择寄存器

地址：0x4001_13B8

复位值：0x0

表 24-22 ID 滤波器选择寄存器 CAN_ACF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	IDCMP	IDMASK	ID ACR [28:16] or AMR [28:16]												
	RW	RW	RW												
	0	0	0												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID ACR [15:00] or AMR [15:00]															
RW															
0															

位置	位名称	说明
[31]	--	保持默认值
[30]	IDCMP	SELMASK=1 时有效，ID AMR（MASK）选择作用范围 1：ID 滤波器接收 MASK 按照标准帧格式或者扩展帧格式处理 0：ID 滤波器接收 MASK 兼容标准帧格式和扩展帧格式两种格式
[29]	IDMASK	IDCMP 位为 1 时，MASK 选择哪种帧格式 1：ID 滤波器仅接收扩展帧格式 0：ID 滤波器仅接收标准帧格式
[28:00]	ID CODE/MASK	ACFADR 选择是那一组 ID 滤波器 SELMASK=0，表示 ID 滤波器选择的是 ID ACR SELMASK=1，表示 ID 滤波器选择的是 AMR 标准帧格式时 ID ACR/AMR 的范围是 Bit10-Bit0，扩展帧格式时的范围是 Bit28-Bit0

25 版本历史

表 25-1 文档版本历史

时间	版本号	说明
2025.08.26	1.35	增加欠压阈值说明
2025.08.15	1.34	DSP0_BSRR 寄存器改为 DSP0_BRR 寄存器
2025.07.31	1.33	07x 过采样设置注意事项
2025.07.31	1.32	修订 IO Driver 模块数据流程图内寄存器描述
2025.07.30	1.31	修订 FAIL 信号说明，增加 FAIL 信号分布表
2025.05.12	1.30	添加 PVD 欠压不发生复位事件说明
2025.04.16	1.29	修订 ROM_SIZE 位置到[3:2]
2025.03.21	1.28	删除 10k 电阻值
2025.01.25	1.27	更新 SIF SIF0_TOSC 寄存器描述
2025.01.02	1.26	更新 FLASH NVR 关于 OPA Offset 的地址
2024.12.17	1.25	增加 FLASH NVR 关于 OPA Offset 的地址
2024.12.02	1.24	增加 ADC_IF 触发所需时间说明
2024.09.29	1.23	CLU 输入多路复选部分补充
2024.08.17	1.21	修订 IT_OPA 描述
2024.08.08	1.20	ADC 阈值监测功能勘误
2024.06.17	1.19	添加 NVR 校准参数地址信息
2024.06.04	1.18	修改关于 PLL 时钟精度的说明，删除 BSIZE 寄存器描述
2024.05.29	1.17	修改 I2C 模块中关于主从模式下收发的流程图
2024.05.09	1.16	修订 DMA 的 SIF 请求，ADC 空闲采样，SYS_AFE_REN6.PVDSEL 以及 GPIO 开漏功能的说明
2024.01.19	1.15	修正 IIC 关于时钟分频的说明，补充关于休眠唤醒的说明，补充 ADC 模块通道数量说明
2023.11.20	1.14	添加 DAC Gain 的量程说明和 OPA 的单端输出模式说明
2023.09.28	1.13	修改 SPI、I2C、HALL、SIF 模块的寄存器命名，与器件库头文件中命名保持一致
2023.09.11	1.12	删除 OPAHFLF_EN 的说明，增加 ADC 用外部基准时量程挡位说明，GPIO 功能框图增加 ESD 二极管。
2023.08.10	1.11	修改了 DAC 校准值地址和 GPIO 部分的笔误
2023.07.24	1.1	修正 FLASH_CFG 配置寄存器的复位值
2023.07.05	1.09	修正部分笔误，修改比较器比较速度 CMP_FT 的说明
2023.07.04	1.08	修改 SYS_AFE_REG2 中 bit5 为保留位，mcu 供电范围
2023.05.07	1.07	修正 PWM 的 Filter 滤波模块的时钟分频的说明
2023.04.20	1.06	增加 PWR_WEAK 掉电检测标志位说明
2023.04.07	1.05	增加 OPAHFLF_EN 说明
2023.02.18	1.04	修改 MCPWM_SDCFG 的描述
2023.02.15	1.03	增加关于 TIMER 外部事件的描述
2023.02.10	1.02	补充关于 PLLPDN、BGPPD、BGPPD 的描述
2023.02.06	1.01	增加 SYS_FLSE/SYS_FLSP 寄存器的描述
2022.10.16	1.0	正式版发布

2022.01.02	0.9	更新 CAN 模块的描述，CMP_TCLK 滤波配置拆分，增加 MCPWM ZCS_IF
2021.12.29	0.8	增加 CAN 模块的描述
2021.12.22	0.7	增加 CL0 模块的描述
2021.12.17	0.6	更新 SYS_AFE_REG
2021.12.14	0.5	修订 ADC 通道，DSP 增加 GPIO、CL 访问的说明
2021.12.11	0.4	修订除 SIF、SPI、I2C 外的所有章节配图
2021.03.18	0.3	修改 HRC/LRC 时钟频率
2021.02.02	0.2	除 flash/i2c/spi 外，所有章节修改完毕
2020.12.01	0.1	初始版本，包含测试相关说明的内部版本

免责声明

LKS 和 LKO 为凌鸥创芯注册商标。

南京凌鸥创芯电子有限公司（以下简称：“Linko”）尽力确保本文档内容的准确和可靠，但是保留随时更改、更正、增强、修改产品和/或 文档的权利，恕不另行通知。用户可在下单前获取最新相关信息。

客户应针对应用需求选择合适的 Linko 产品，详细设计、验证和测试您的应用，以确保满足相应标准以及任何安全、安保或其它要求。客户应对此独自承担全部责任。

Linko 在此确认未以明示或暗示方式授予 Linko 或第三方的任何知识产权许可。

Linko 产品的转售，若其条款与此处规定不同，Linko 对此类产品的任何保修承诺无效。

如有更早期版本文档，一切信息以此文档为准。