南京凌鸥创芯电子有限公司

LKS32MC03x User Manual

图片目录

目录

表格目录

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 64kHz RC 时钟

HSI:High Speed Internal Clock，即 4MHz RC 时钟

PLL:Phase Lock Loop Clock，即锁相环时钟，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

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

LKS32MC03x 系列 MCU 是极致性价比的紧凑型 MCU，配备 48MHz Cortex-M0 内核，以及必要的模拟和外设资源。模块集成更为精炼，广泛适用于轻量级应用。

2.2 特性

- 48MHz 32 位 RISC 内核，32bit 硬件除法协处理器
- 4 通道 DMA
- 低功耗休眠模式
- -40~105℃工业级工作温度范围
- 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

- 16/32kB Flash，数据防盗功能
- 4kB RAM

2.2.2 时钟

- 内置 4MHz 高精度 RC 时钟，全温度范围精度±1%
- 内置 64kHz 低速时钟，供低功耗模式使用
- 内部 PLL 可提供最高 48MHz 时钟

2.2.3 外设

- 一路 UART
- 一路 SPI
- 一路 IIC
- 通用 16/32 位 Timer，支持捕捉和边沿对齐 PWM
- 电机控制专用 PWM 模块，支持 6 路 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
- 26 路 GPIO

2.2.4 模拟模块

- 集成 1 路 12bit SAR ADC，1Msps 采样及转换速率，共 11 通道
- 集成 1 路 OPA，可设置为差分 PGA 模式
- 集成两路比较器
- 集成 8bit 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

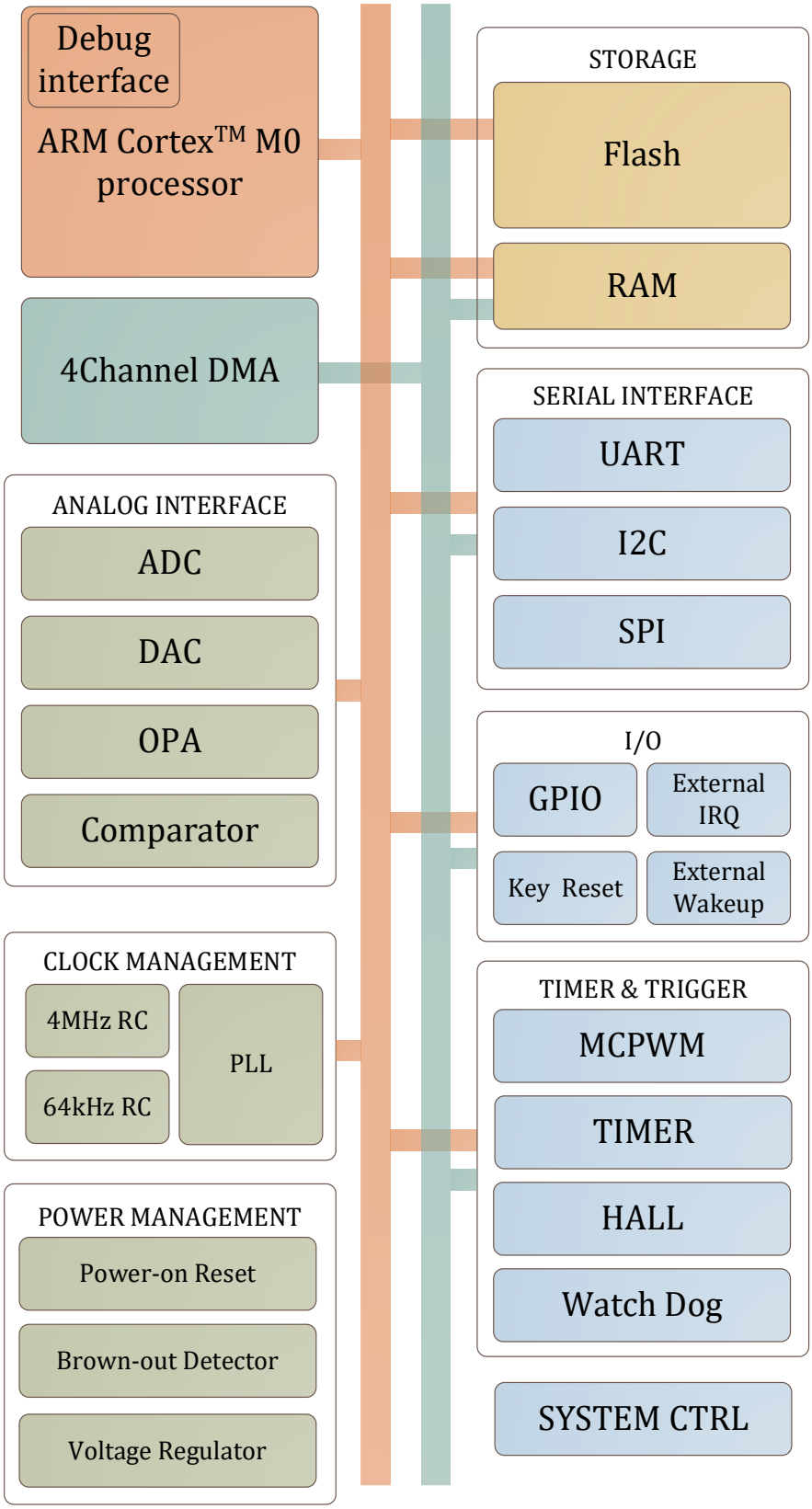


图 2-1 LKS32MC03x 系统框图

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	0x0000_7FFF	32kB	同总线
	FLSCR	0x0001_0000	0x0001_00FF	256B	同总线
RAM	RAM	0x2000_0000	0x2000_0FFF	4kB	同总线
	SYS	0x4000_0000	0x4000_00FF	256B	同总线
	SPI	0x4001_0000	0x4001_00FF	256B	外设门控时钟[10] 软复位[10]
	I2C	0x4001_0100	0x4001_01FF	256B	外设门控时钟[0] 软复位[0]
	CMP	0x4001_0200	0x4001_02FF	256B	外设门控时钟[3] 软复位[3]
	HALL	0x4001_0300	0x4001_03FF	256B	外设门控时钟[1] 软复位[1]
	ADC	0x4001_0400	0x4001_04FF	256B	外设门控时钟[9] 软复位[9]
	TIMER0	0x4001_0500	0x4001_05FF	256B	外设门控时钟[5] 软复位[5]
	TIMER1	0x4001_0600	0x4001_06FF	256B	外设门控时钟[6] 软复位[6]
	MCPWM	0x4001_0700	0x4001_07FF	256B	外设门控时钟[4] 软复位[4]
	GPIO	0x4001_0800	0x4001_08FF	256B	外设门控时钟[7] 软复位[7]
	UART	0x4001_0900	0x4001_09FF	256B	外设门控时钟[2] 软复位[2]
	DMA	0x4001_0A00	0x4001_0AFF	256B	外设门控时钟[11] 软复位[11]
	DIV	0x4001_0B00	0x4001_0BFF	256B	外设门控时钟[8] 软复位[8]
	AON	0x4001_0C00	0x4001_0CFF	256B	同总线

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。

LKS32MC03x 系列芯片共有 14 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	16	Reserved
1	TIMER1	17	Reserved
2	MCPWM0	18	Reserved
3	MCPWM1	19	Reserved
4	I2C	20	Reserved
5	SPI	21	Reserved
6	GPIO	22	Reserved
7	HALL	23	Reserved
8	UART	24	Reserved
9	CMP	25	Reserved
10	ADC	26	Reserved
11	DMA	27	Reserved
12	WAKEUP，唤醒中断	28	Reserved
13	软件中断	29	Reserved
14	Reserved	30	Reserved
15	Reserved	31	Reserved

5 模拟电路

5.1 简述

模拟电路包含以下模块:

- 集成 1 路 12BIT SAR ADC，采样率 1MHz，每路采样电路最多 11 通道信号可选。
- 集成 1 路运算放大器，可设置为 PGA 模式
- 集成 2 路比较器，可设置迟滞模式
- 集成 8BIT 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。

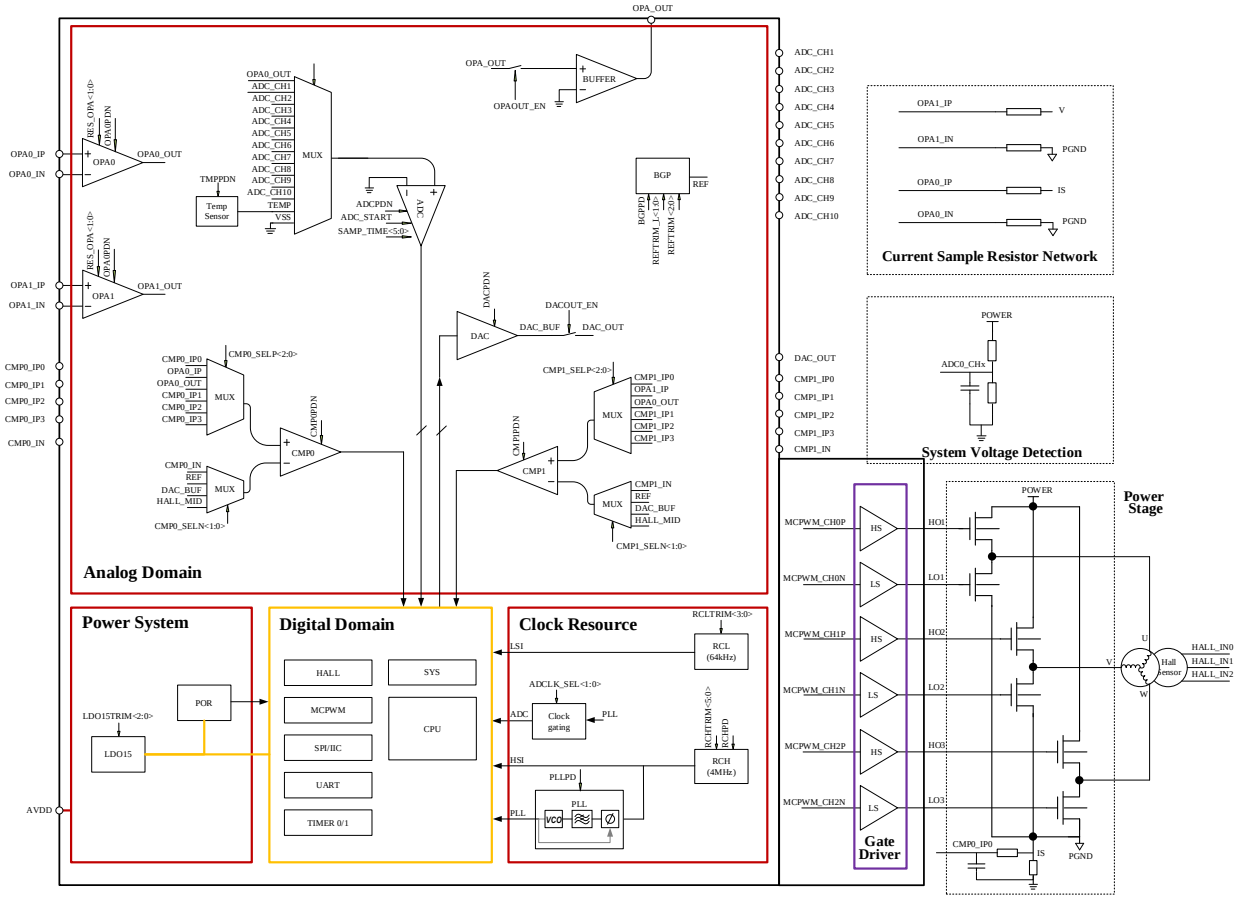


图 5-1 模拟电路功能框图

5.1.1 电源管理系统

电源管理系统由 LDO15 模块、上电/掉电复位模块（POR）组成。

该芯片由 3.3~5V 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 1.25V 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。

5.1.2 时钟系统

时钟系统包括内部 64kHz RC 时钟、内部 4MHz RC 时钟、PLL 电路组成。

64kHz 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。4MHz RC 时钟可作为 MCU 主时钟使用。PLL 最高可提供 48MHz 的时钟，通常 MCU 使用 PLL 时钟作为系统主时钟。

64kHz 和 4MHz RC 时钟均带有出厂校正，64kHz RC 时钟在-40~105℃范围内的精度为±50%，4MHz RC 时钟在该温度范围的精度为±1%。

4MHz RC 时钟在 5V 供电和 3.3V 供电下的校准参数略有差别，对于大多数的 5V 应用来说，芯片的 4M RC 时钟已处于校准状态。如应用于 3.3V 供电，则建议由软件加载校正参数到相应的寄存器，具体参考相应应用笔记。如不加载，4M RC 时钟的偏差约增加 1.5%。

4MHz RC 时钟通过设置 BGPPD='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，4MHz RC 时钟和 BGP 模块都是开启的。64kHz RC 时钟始终开启，不能关闭。

PLL 对 4MHz 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 48MHz，ADC 模块最高时钟 24MHz。

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 6us 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。

BGPPD /PLLPDN 的说明见模拟寄存器 SYS_AFE_REG0

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_REG0

5.1.4 ADC 模块

请参加第 12 章。

5.1.5 运算放大器

芯片集成 1 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_OPA[1:0]设置，以实现不同的放大倍数。

RES_OPA<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:

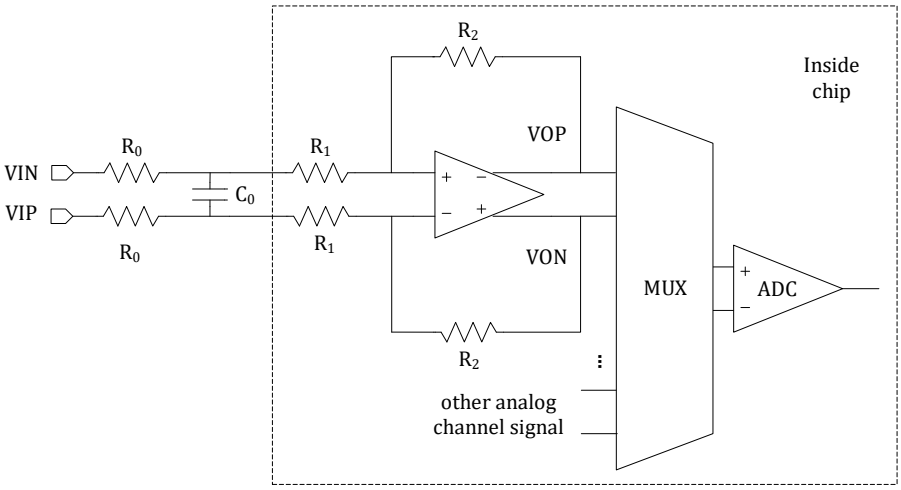


图 5-2 放大器框图

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 最小为 15pF。

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 P0.7 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。

OPAOUT_EN 的说明见模拟寄存器 SYS_AFE_REG0

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 OPAPDN=1 打开。开启放大器之前，需要先开启 BGP 模块。

OPAPDN 的说明见模拟寄存器 SYS_AFE_REG0

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。

OPA 在芯片端口上有多组差分输入。其中 OPA0_IP/OPA0_IN 和 OPA1_IP/OPA1_IN 时分复用内部

的运算放大器。当 ADC 的采样通道配置为 ADC_CH8 时即 ADC0_CHN0[3:0]=8，实际是采样 OPA1 的差分输入信号经过 OPA 放大后的输出，当 ADC 的采样通道配置为 0 时即 ADC0_CHN0[3:0]=0，实际是采样 OPA0 的差分输入信号经过 OPA 放大后的输出。

此外，OPA0 的差分输入有两组，OPA0_IP/OPA0_IN 和 OPA0_IP_B/OPA0_IN_B，通过 SYS_AFE_REG0.OPA0_B_EN 进行选择。这一设置通常根据具体芯片型号引脚功能分布进行配置，在应用期间维持不变。

OPA 的多路差分输入及选择控制逻辑如图所示。

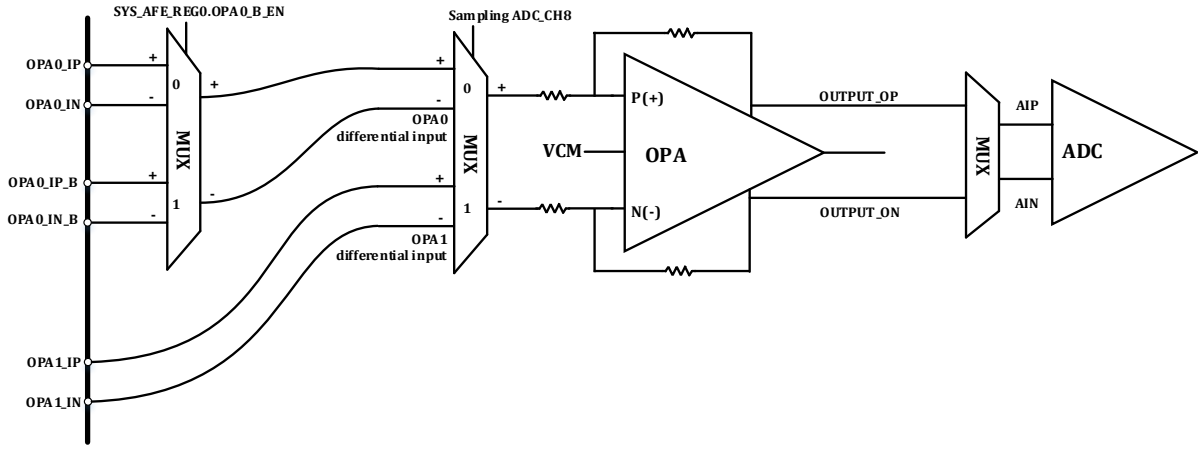


图 5-3 OPA 多路差分输入选择控制逻辑

ADC 在采样 ADC_CH8 时，实际上是采样 OPA 放大后的 OPA1 信号。 OPA 输入开关信号将始终与要采样的下一个 ADC 通道保持一致。 建议将 OPA1(即 ADC_CH8)设置为 ADC 采样序列中的第一个通道，这将为 OPA1_IP/OPA1_IN 作为 OPA 输入提供足够的稳定时间。

当 OPA 输入时在 OPA0_IP/OPA0_IN 和 OPA1_IP/OPA1_IN 之间切换时，需要至少 1us 的稳定时间。不建议立即对 OPA0/1 连续进行采样。

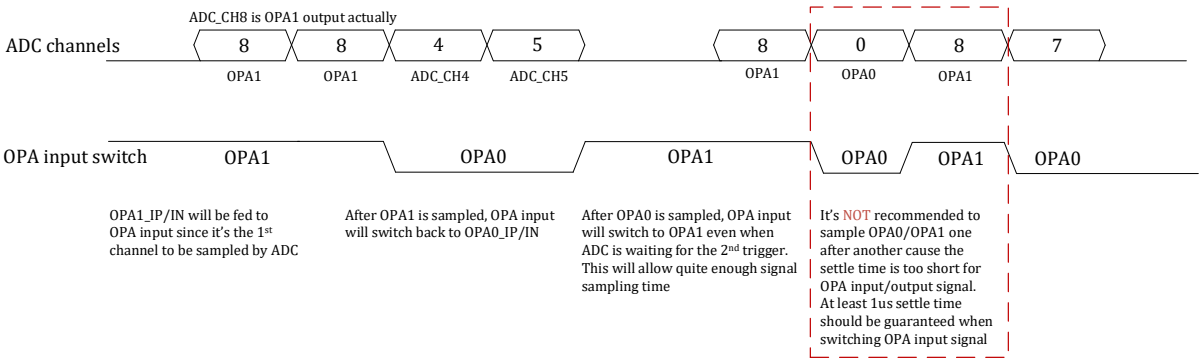


图 5-4 OPA 多路差分输入选择与 ADC 采样联动

5.1.6 比较器

内置 2 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。

比较器正负两个输入端的信号来源都可通过寄存器 CMPx_SELP[2:0]和 CMPx_SELN[1:0]进行设置（x=0/1，代表 CMP0/CMP1 两个比较器）。

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 5-5。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。

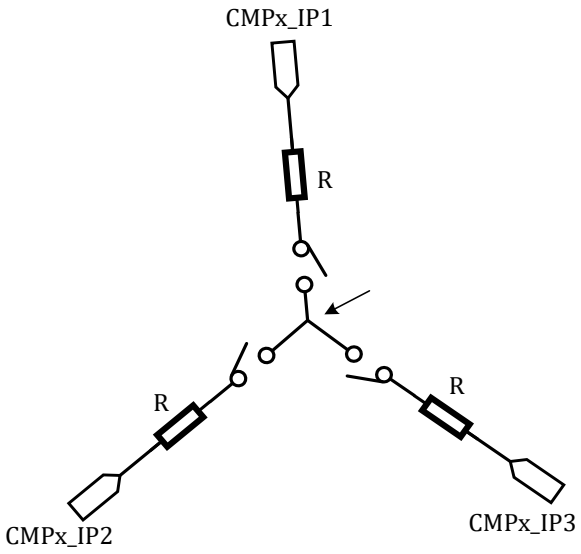


图 5-5 BEMFx_MID 信号

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_REG1

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_REG0

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℃。

芯片出厂前会经温度校正，校正值保存在 flash info 区。

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 ADC 的通道 11。

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_REG0

温度传感器的典型曲线如下图所示:

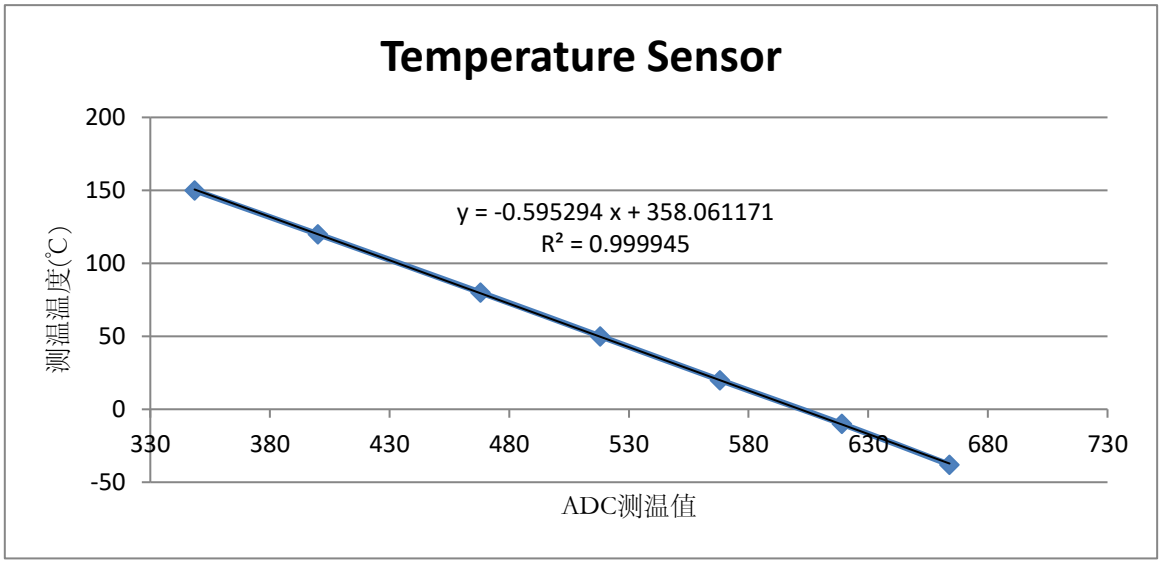


图 5-4 温度传感器曲线

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=-0.595x+358.1

求得的 Y 值即为此时的温度。

公式中有两个系数，a=-0.595, b=358.1。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 写入芯片特定 FLASH 区域（通过库函数读取）。存储时，会将 b 系数小数点右移一位（乘 10）存入，小数点后第二位不进行保存。

同时为方便客户操作，系数 a 也会存入芯片特定 FLASH 区域（通过库函数读取）。存储时，将 a 系数小数点右移四位（乘 10000）存入。

实际使用中，应通过库函数读出 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

芯片内置一路 8bit DAC，输出信号的最大量程为 3V；C 版本最大量程为 4.8V，若需要使用 1.2V 量程，需要设置 SYS_AFE_REG2.BIT15=1。

DAC 可通过配置寄存器 DACOUT_EN=1，将 DAC 输出送至 P0.0 管脚，可驱动>5kΩ 的负载电阻和 50pF 的负载电容。

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。DAC 可通过设置 DACPDN =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_DAC，低 8BIT 有效。信号范围是 0x00~0xFF。0x00 对应零模拟量输出 0V，0xFF 对应满量程模拟量输出为DAC_{fs}，。每一档信号(LSB)所对应的模拟信号幅度为$\frac{DAC_{fs}}{256}$。若 SYS_AFE_DAC 的数字值为 Din，则该数字信号所对应的 DAC 输出模拟信号为$\frac{DAC_{fs}}{256} * Din$

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。

NVR 中存储了 DAC_AMC/DC 校正值，可用于校准 DAC 输出，具体用法请参考官方库函数。

DACOUT_EN 的说明见 SYS_AFE_REG1

DACPDN 的说明见 SYS_AFE_REG0

SYS_AFE_DAC 的说明见 SYS_AFE_DAC DAC

5.2 寄存器

5.2.1 地址分配

模拟寄存器 SYS_AFE_REG0~ SYS_AFE_REG2，对应地址为 0x4000_0010~0x4000_0018。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_ADC	0x00	ADC 数据寄存器
SYS_AFE_INFO	0x04	芯片版本信息寄存器
	0x08	保留
SYS_OPA_SEL	0x0C	运放切换控制寄存器
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_DAC	0x2C	DAC 数字量寄存器

5.2.2 SYS_AFE_ADC ADC 原始数据寄存器

地址:0x4000_0000

复位值:0x0

表 5-2 ADC 原始数据寄存器 SYS_AFE_ADC

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

位置	位名称	说明
[31:12]		未使用
[11:0]	ADC_RAW_D	ADC 输出的原始数据

5.2.3 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 5-3 芯片版本信息寄存器 SYS_AFE_INFO

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

位置	位名称	说明
[31:3]		未使用
[2:0]	Version	芯片版本信息 1: DAC 输出范围 0~3V 2: DAC 输出范围 0~3V/4.8V 3: DAC 输出范围 0~1.2V/3V/ 4.8V

此寄存器与用户无关。

5.2.4 SYS_OPA_SEL 运放切换控制寄存器

地址:0x4000_000C

复位值:0x1

表 5-4 运放切换控制寄存器 SYS_OPA_SEL

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

位置	位名称	说明
[31:1]		未使用
[0]	OPA_SEL_EN	OPA0/1 输入分时复用芯片内唯一的一个 OPA，复用使能开关，高电平有效。 B 版本之后此 BIT 寄存器可读写。

通过 SYS_OPA_SEL=0xCA 可以将该寄存器置 1，使能复用功能。通过 SYS_OPA_SEL &= ~BIT0 可以将该寄存器置 0，关闭复用功能。

根据 OPA_SEL_EN 的使能与否，模拟内部的二选一多选器会控制。

芯片只有 1 个 OPA 模块，但有 2 路 OPA 输入信号，因此 OPA0/OPA1 信号复用 OPA 模块。

OPASELA 由 ADC 接口模块硬件控制，跟模拟寄存器 ADC_SELP<3:0>一样。

OPASELA	运放 A 输入通道选择	0: 选择 OPA0_IP/OPA0_IN 引脚作为运放信号通道 1: 选择 OPA1_IP/OPA1_IN 引脚作为运放信号通道

ADC 物理通道的选择寄存器对应关系如下:

ADC_SELP<3:0>	ADC 通道正端选择	0000:OPA0 输出; 0001: ADC_CH1; 0010: ADC_CH2; 0011: ADC_CH3; 0100: ADC_CH4; 0101: ADC_CH5; 0110: ADC_CH6; 0111: ADC_CH7; 1000:ADC_CH8/OPA1 输出; 1001:ADC_CH9; 1010:ADC_CH10; 1011:连温度传感器; 1100:连内部地; 1101:连 2.4V 基准;

其中 CH8 是复用的。

表 5-5 ADC 通道复用 OPA 列表

OPA_SEL_EN	当前 ADC 采样通道	ADC 实际采样通道
0	8	ADC_CH8

1	8	OPA1_OUT

当 OPA_SEL_EN=1，OPA 输入通道切换由硬件电路自动控制；OPA 输入的切换控制信号 OPASELA 仅分别在 ADC 采样第 8 通道时为 1，其余时间均为 0。

通常建议运放 OPA0 和 OPA1 的采样，间隔一定时间，比如 ADC 单段采样模式下，一轮采样 6 个通道。第一次采样 OPA1，第 6 次采样 OPA0，给 OPA 输入信号以充足的建立时间。如果某轮采样第一次即采样 OPA1，则 OPA 输入控制信号在采样开始很长时间之前就切换至 OPA1，使得 OPA1 的采样有足够的建立时间。OPA1 采样完毕后，OPA 输入信号就会切换回 OPA0，如果间隔通道数足够，OPA0 输入信号也给予运放以充足的建立时间。

5.2.5 SYS_AFE_REG0 模拟配置寄存器 0

地址:0x4000_0010

复位值:0x0

表 5-6 模拟配置寄存器 0 SYS_AFE_REG0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLLPDN	CMP1PDN	CMP0PDN	TMPPDN	DACPDN	BGPPD	OPAPDN	ADCPDN	REF2VDD	GA_AD	OPA0_B_EN	OPASEL	Res.	OPAOUT_EN	RES_OPA	
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

位置	位名称	说明
[31:16]		未使用
[15]	PLLPDN	0:关闭 PLL 1:开启 PLL
[14]	CMP1PDN	0:关闭比较器 1 1:开启比较器 1
[13]	CMP0PDN	0:关闭比较器 0 1:开启比较器 0
[12]	TMPPDN	0:关闭温度传感器 1:开启温度传感器
[11]	DACPDN	0:关闭 DAC 1:开启 DAC
[10]	BGPPD	0:开启 Bandgap 1:关闭 Bandgap，注意极性与其他模块相反
[9]	OPAPDN	0:关闭 OPA

		1:开启 OPA
[8]	ADCPDN	0:关闭 ADC 1:开启 ADC
[7]	REF2VDD	ADC 基准源选择 0:选择内部 2.4V 作为 ADC 基准源 1:选择 AVDD 电源电压作为 ADC 基准源，此时建议设置 ADC 量程 GA_AD=1，ADC 满量程对应 AVDD。
[6]	GA_AD	ADC 量程选择 0: REF*3/2，当使用内部 2.4V 基准时，ADC 满量程为 2.4*3/2=3.6V，当使用 AVDD 电源电压作为基准时，ADC 满量程为 AVDD*3/2，如使用 5V 供电，即满量程为 7.5V; 1: REF。当使用内部 2.4V 基准时，ADC 满量程为 2.4V，当使用 AVDD 电源电压作为基准时，ADC 满量程为 AVDD，如使用 5V 供电，即满量程为 5V。
[5]	OPA0_B_EN	0: OPA0 使用 OPA0_IN/OPA0_IP 作为差分输入 1: OPA0 使用 OPA0_IN_B/OPA0_IP_B 作为差分输入
[4]	保留位	必须为 0
[3]	保留位	必须为 0
[2]	OPAOUT_EN	0:不输出; 1:输出 OPA 放大输出信号到 IO 口 P0.7
[1:0]	RES_OPA	运放反馈电阻 00: 200k:10k 01: 190k:20k 10: 180k:30k 11: 170k:40k

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 选择了 HRC 时钟，则 RCHPD 是被硬件控制的，软件配置 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（bandgap），所以 BGPPD 也是硬件控制的，在芯片使用了 RCH 或 PLL 时，软件配置 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。

5.2.6 SYS_AFE_REG1 模拟配置寄存器 1

地址:0x4000_0014

复位值:0x0

表 5-7 模拟配置寄存器 1 SYS_AFE_REG1

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

Res.	CMP1_SELP	CMP_FT	CMP0_SELP	CMP_HYS	REFOUT_EN	CMP1_SELN	CMP0_SELN	DACOUT_EN	LDOOUT_EN
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:16]		未使用
[15]	保留	必须为 0
[14:12]	CMP1_SELP	比较器 1 信号正端选择 000: CMP1_IP0 001: OPA0_IP 010: OPA0_OUT 011: CMP1_IP1 100: CMP1_IP2 101: CMP1_IP3 110: NC 111: NC
[11]	CMP_FT	比较器快速比较使能 1: 比较器比较速度小于 30ns 0: 不使能，200ns
[10:8]	CMP0_SELP	比较器 0 信号正端选择 000: CMP0_IP0 001: OPA0_IP 010: OPA0_OUT 011: CMP0_IP1 100: CMP0_IP2 101: CMP0_IP3 110: NC 111: NC
[7]	CMP_HYS	比较器回差选择 0: 20mV 1: 0mv
[6]	保留	必须为 0
[5:4]	CMP1_SELN	比较器 1 信号负端选择 00: CMP1_IN 01: REF 10: DAC 输出 11: BEMFx_MID
[3:2]	CMP0_SELN	比较器 0 信号负端选择 00: CMP0_IN 01: REF

		10: DAC 输出 11: BEMFx_MID
[1]	DACOUT_EN	DAC 输出到 IO 使能 0: 不输出 1: 输出 DAC 到 P0.0
[0]	LDOOUT_EN	LDO 输出到 IO 使能 0: 不输出 1: 输出 LDO15 到 P0.0

5.2.7 SYS_AFE_REG2 模拟配置寄存器 2

地址:0x4000_0018

复位值:0x0

表 5-8 模拟配置寄存器 2 SYS_AFE_REG2

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

位置	位名称	说明
[31:16]		未使用
[15]	C 版本 DAC_GAIN	仅对 C 版本有效 0: 3.0V/4.8V 量程 1: 1.2V 量程
[14:13]	保留	必须为 0
[12:8]	SAMP_TIME	ADC 采样时间选择 00000->11111:对应采样时间从 5->37 个 ADC 时钟周期，逐次增加
[7:0]	保留	必须为 0

5.2.8 SYS_AFE_DAC DAC 数字量寄存器

地址:0x4000_002C

复位值:0x0

表 5-9 DAC 数字量寄存器 SYS_AFE_DAC

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
[31:8]		未使用
[7:0]	DAC_IN	DAC 待转换的数字量输入

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	64kHz	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	4MHz	内部 RC 振荡器	全温度范围误差<1%	作为 PLL 源时钟
PLL	48MHz	PLL 时钟	0	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 4MHz 的参考时钟 HSI 倍频 12 倍至 48MHz。

PLL 经过 n/8 分频后，可以得到 48MHz×n/8 的高速时钟，SYS_CLK_CFG.CLK_SEL[0]在此分频后的高速时钟与 4MHz 的 HSI 之间进行二选一，作为系统主时钟 MCLK。系统复位时，PLL 默认关闭，HSI 默认开启，系统选择 HSI 时钟，即 4MHz 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。

系统内部时钟 MCLK 是系统主时钟。可以通过 SYS_CLK_CFG 寄存器 CLK_DIV 位域控制进行 n/8 分频，可以产生 12，24,48MHz 等频率值。SYS_CLK_CFG.CLK_SEL[0]表示选择 PLL 或 HSI 作为系统主时钟。此外，通过设置 SYS_CLK_CFG.CLK_SEL[1]=1 可以切换系统时钟为 LSI 时钟，此时系统工作于 64kHz 时钟，PLL/HRC/BGP 等模拟模块均可关闭以降低功耗。

当 SYS_CLK_CFG.CLK_SEL 为 1 时，SYS_CLK_CFG.CLK_DIV 作为 PLL 的分频系数。当 SYS_CLK_CFG.CLK_SEL 不为 1 时，系统时钟 HSI 或 LSI 作为工作时钟，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	6	是
0x0111	2/8	12	是
0x0155	4/8	24	是
0x01FF	8/8	48	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 24MHz），即 ACLK。

内部 64kHz RC 产生一路 LSI 时钟 LCLK，主要用于 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。

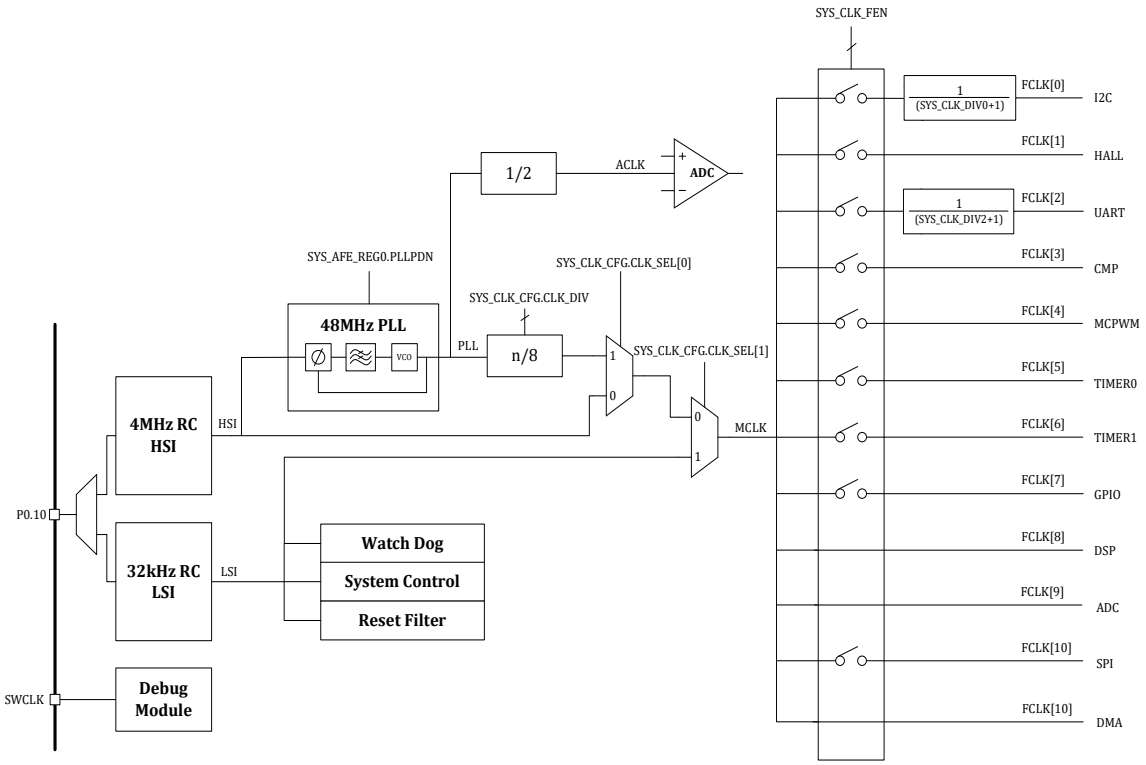


图 6-1 时钟架构

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 本身无法被关闭，作为 PLL 参考时钟的 HSI 无法被软件关闭；64kHz 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 1.25V 时产生复位，5V 电源低于 2.5V 时产生复位
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

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。

P0.2 引脚低于 16us 宽度的外部复位会被复位滤波滤除，要求可靠外部复位宽度大于 200us。

3 个复位信号复位等级和作用域一致，均为全局复位。

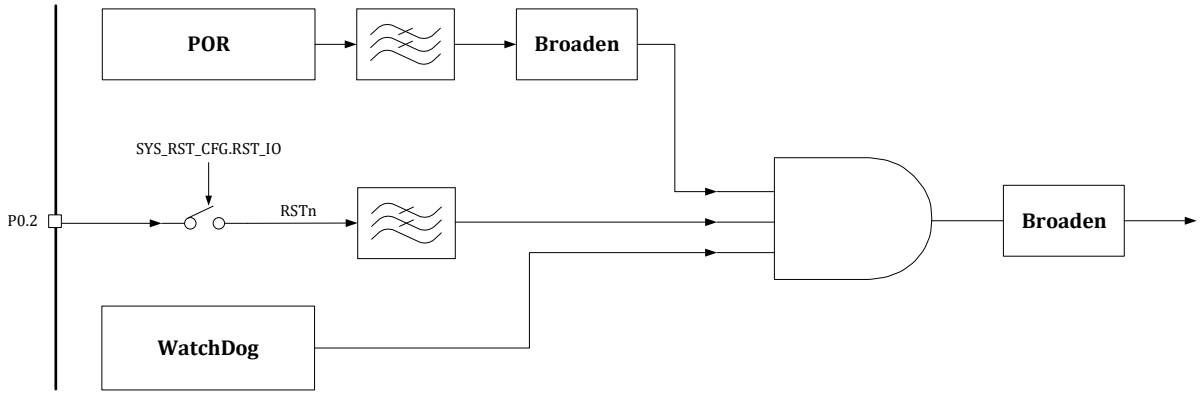


图 6-2 复位架构

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，但对所有外设中的寄存器没有影响。

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。

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 6-4 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.SPI_ SFT_RST	SPI
SYS_SFT_RST.ADC_ SFT_RST	ADC 数字接口模块
SYS_SFT_RST.DIV_ SFT_RST	DIV
SYS_SFT_RST.GPIO_ SFT_RST	GPIO
SYS_SFT_RST.TIMER1_ SFT_RST	TIMER1
SYS_SFT_RST.TIMER0_ SFT_RST	TIMER0
SYS_SFT_RST.MCPWM_ SFT_RST	MCPWM
SYS_SFT_RST.CMP_ SFT_RST	比较器数字接口模块
SYS_SFT_RST.UART_SFT_RST	UART
SYS_SFT_RST.HALL_SFT_RST	HALL
SYS_SFT_RST.I2C_SFT_RST	I2C
NVIC_SystemReset();	CPU 软复位，仅复位 CPU 内核，将 PC 重置为 0，所有外设寄存器值仍然维持。

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_RST_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。

由于 CPU 软复位仅仅复位 CPU 内核，而不复位外设寄存器，因此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 6-5 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 6-6 时钟控制寄存器 SYS_CLK_CFG

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

位置	位名称	说明
[31:10]		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: LRC 注意，PLL 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 HRC 时钟（4MHz），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 4MHz。

当 CLK_SEL = 2 时，MCLK 选择 LRC 时钟（64kHz），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 64kHz。

6.3.3 SYS_IO_CFG IO 控制寄存器

地址:0x4000_0084

复位值:0x40

表 6-7 IO 控制寄存器 SYS_IO_CFG

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

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:P1.8, P1.9 作为正常 GPIO 使用 1:P1.8 复用为 SWCLK，P1.9 复用为 SWDIO
[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 30ms 内，P1.8 和 P1.9 不能切换为 GPIO 功能，只能作为 SWD 使用。即使软件改写了 SYS_MISC_CFG[6]也需要在 30ms 之后才会生效。现象为:在 30ms 内，软件可以向 SYS_IO_CFG[6]写入 1'b0，但读回该 bit 仍为 1'b1，如果经过 30ms 后再读，则可读回 1'b0。即软件写入是有效的，但不会立即生效。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，P1.8 和 P1.9 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 P1.8 和 P1.9 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 6-8 Debug 控制寄存器 SYS_DBG_CFG

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

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

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。
[14:11]		未使用
[10]	SFT_RST_PERI	Debug 软复位是否复位除 Flash/SYS_AFE 以外的外设寄存器 需要向 BIT[14:10]写入 0x1F，将此位置位
[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 6-9 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 6-10 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/UART2 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 6-11 SYS_CLK_FEN 外设时钟门控寄存器

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

位置	位名称	说明
[31:12]		未使用
[11]		保留
[10]	SPI_CLK_EN	SPI 模块整体时钟控制信号，默认关闭 SPI 模块时钟 1:使能 SPI 模块时钟 0:关闭 SPI 模块时钟
[9]		保留
[8]	DSP_CLK_EN	DSP 模块整体时钟控制信号，默认关闭 DSP 模块时钟 1:使能 DSP 模块时钟 0:关闭 DSP 模块时钟
[7]		保留
[6]	TIMER1_CLK_EN	TIMER1 模块整体时钟控制信号，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
[5]	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
[4]	MCPWM_CLK_EN	MCPWM 模块整体时钟控制信号，默认关闭 MCPWM 模块时钟 1:使能 MCPWM 模块时钟 0:关闭 MCPWM 模块时钟
[3]	CMP_CLK_EN	CMP 模块整体时钟控制信号，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
[2]	UART_CLK_EN	UART 模块整体时钟控制信号，默认关闭 UART 模块时钟

		1:使能 UART 模块时钟 0:关闭 UART 模块时钟
[1]	HALL_CLK_EN	HALL 模块整体时钟控制信号，默认关闭 HALL 模块时钟 1:使能 HALL 模块时钟 0:关闭 HALL 模块时钟
[0]	I2C_CLK_EN	I2C 模块整体时钟控制信号，默认关闭 I2C 模块时钟 1:使能 I2C 模块时钟 0:关闭 I2C 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 6-12 软复位寄存器 SYS_SFT_RST

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

位置	位名称	说明
[31:12]		未使用
[11]	DMA_ SFT_RST	DMA 模块软复位信号，默认不复位 DMA 模块 1:复位 DMA 模块 0:释放 DMA 模块
[10]	SPI_ SFT_RST	SPI 模块软复位信号，默认不复位 SPI 模块 1:复位 SPI 模块 0:释放 SPI 模块
[9]	ADC_ SFT_RST	ADC 数字接口模块软复位信号，默认不复位 ADC 模块 1:复位 ADC 数字接口模块 0:释放 ADC 数字接口模块
[8]	DSP_ SFT_RST	DSP 模块软复位信号，默认不复位 DSP 模块 1:复位 DSP 模块 0:释放 DSP 模块
[7]	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
[6]	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
[5]	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
[4]	MCPWM_ SFT_RST	MCPWM 模块软复位信号，默认不复位 MCPWM 模块 1:复位 MCPWM 模块 0:释放 MCPWM 模块
[3]	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
[2]	UART_SFT_RST	UART 模块软复位信号，默认不复位 UART 模块 1:复位 UART 模块 0:释放 UART 模块
[1]	HALL_SFT_RST	HALL 模块软复位信号，默认不复位 HALL 模块 1:复位 HALL 模块 0:释放 HALL 模块
[0]	I2C_SFT_RST	I2C 模块软复位信号，默认不复位 I2C 模块 1:复位 I2C 模块 0:释放 I2C 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。

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 初始化函数读取 DC/AMC 等校准参数

6.3.9 SYS_PROTECT/SYS_WR_PROTECT 写保护寄存器

地址:0x4000_00A8

复位值:0x0

表 6-13 写保护寄存器 SYS_PROTECT/SYS_WR_PROTECT

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_DAC、SYS_AFE_DAC_AMC、SYS_AFE_DAC_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，使能寄存器写操作 写入其它值，禁止寄存器写操作

6.3.10 SYS_FLSE 擦除保护寄存器

地址:0x4000_00D0

复位值:0x0

表 6-14 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。

6.3.11 SYS_FLSP 编程保护寄存器

地址:0x4000_00D4

复位值:0x0

表 6-15 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。

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,主要为 MIAN 区域,包括应用程序和用户数据区,有两个大小尺寸:16KB 和 32KB。

ROM 存储体，出厂前固化特定程序，大小为 16KB。

LKS32MC03x 系列芯片，非易失性存储体包含三个型号:

- 型号 1：32KB FLASH
- 型号 2：16KB FLASH，16KB ROM
- 型号 3：16KB FLASH

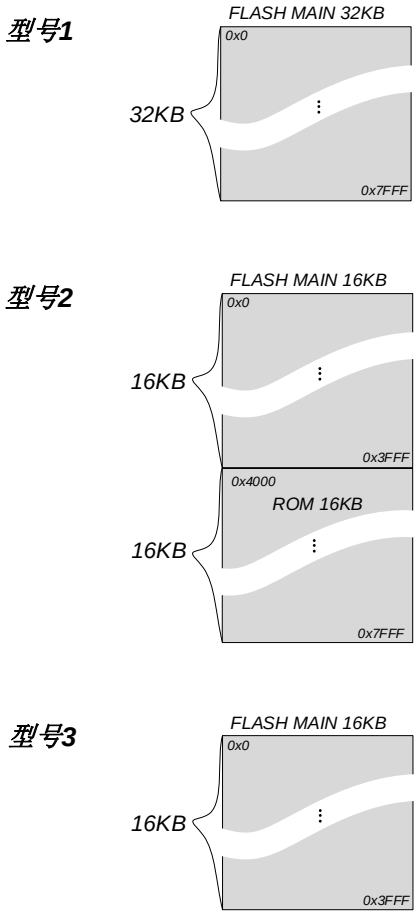


图 7-1 非易失性存储体空间划分框图及型号

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。
- FLASH 写入数据的操作。
- FLASH 擦除操作，包括 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。

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:

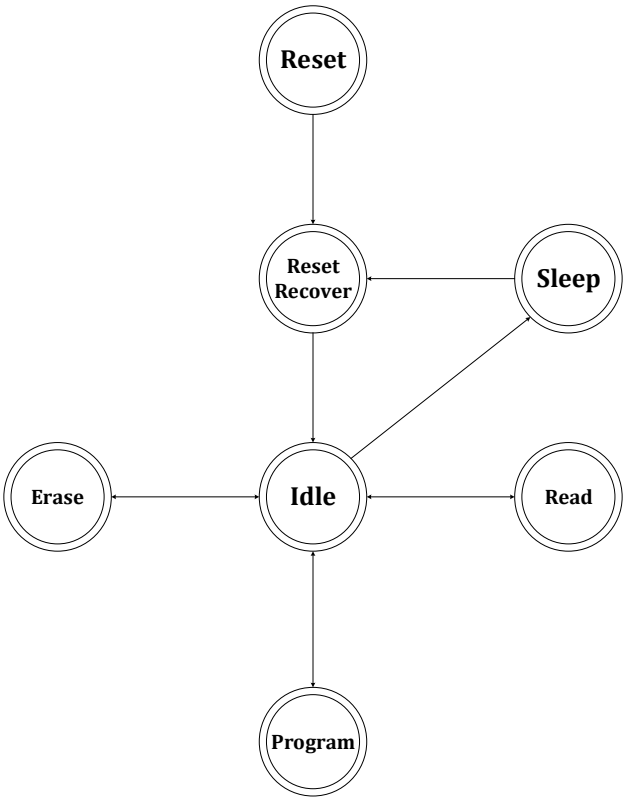


图 7-2 FLASH 控制状态转换图

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

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:

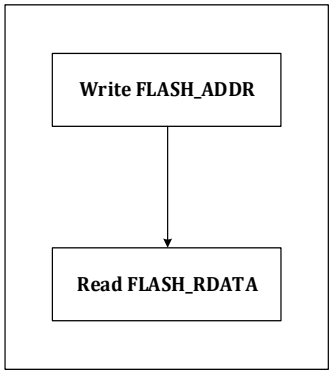


图 7-3 FLASH 间接读取操作流程图

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:

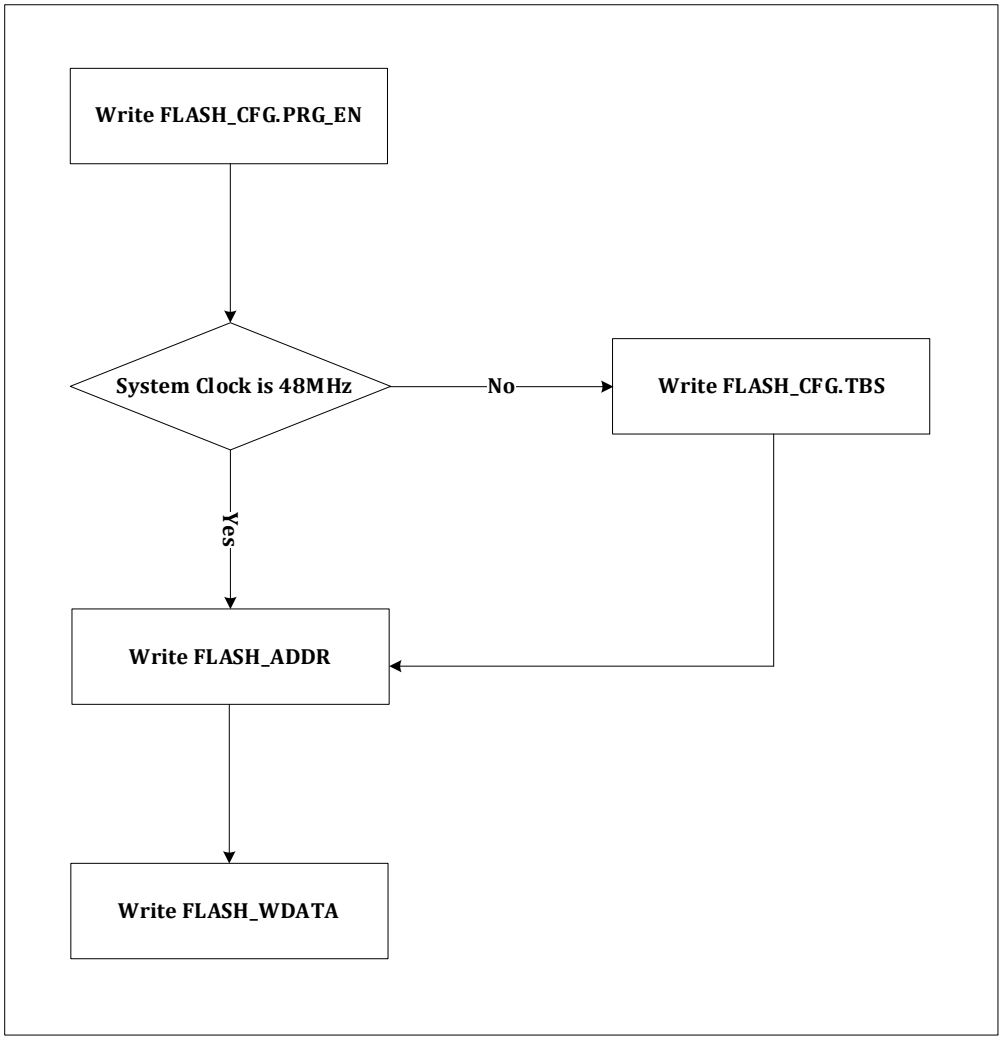


图 7-4 FLASH 模块编程操作流程图

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 48MHz 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 24MHz 和 12MHz 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:

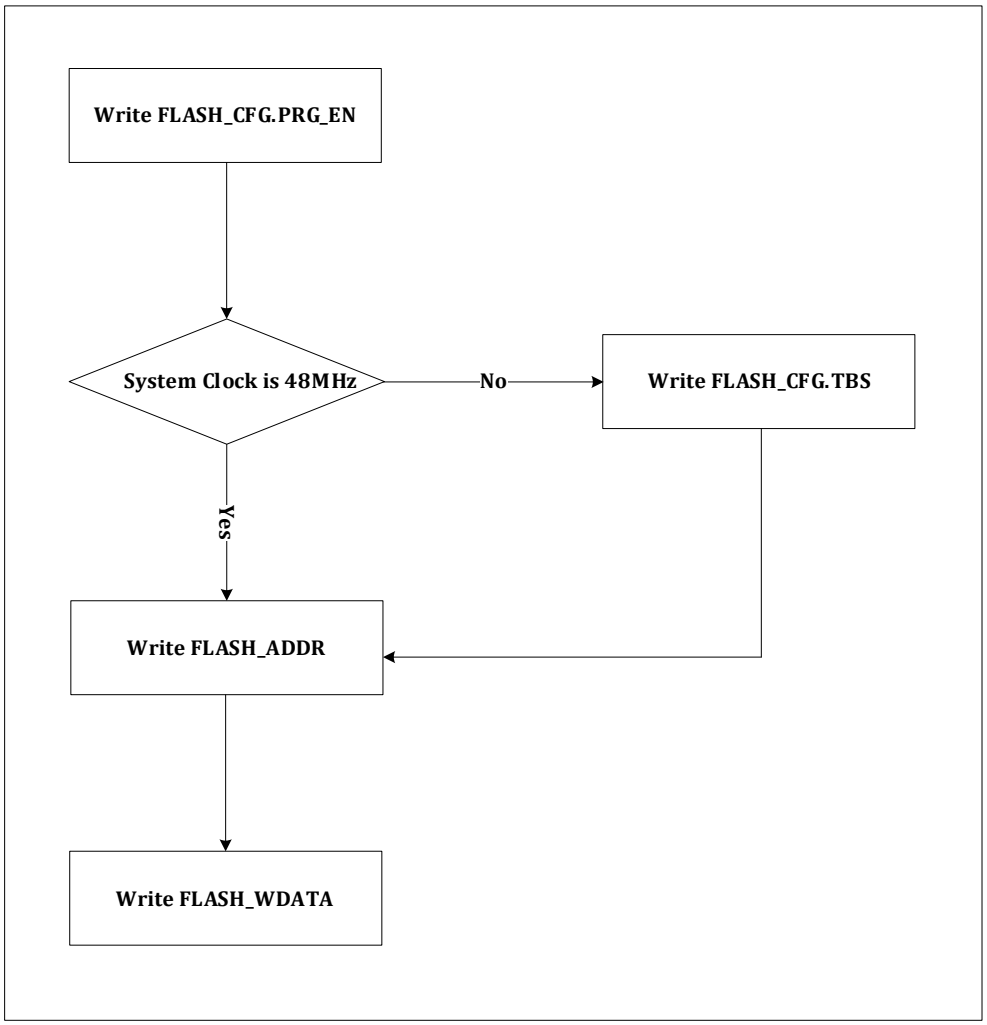


图 7-5 FLASH 模块编程操作流程图

注意，ROM 区域，只支持一次编程。MAIN 区域不受此限制。MAIN 区域推荐使用库函数执行编程操作。ROM 区域编程，推荐使用凌鸥公司指定离线烧录器完成。

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 16KB/32KB 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(16KB 版本，无 Sector32—Sector63)。

表 7-1 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
…	…	…
Sector63	0x0000 7E00 - 0x0000 7FFF	512

FLASH 擦除操作流程如下所示。

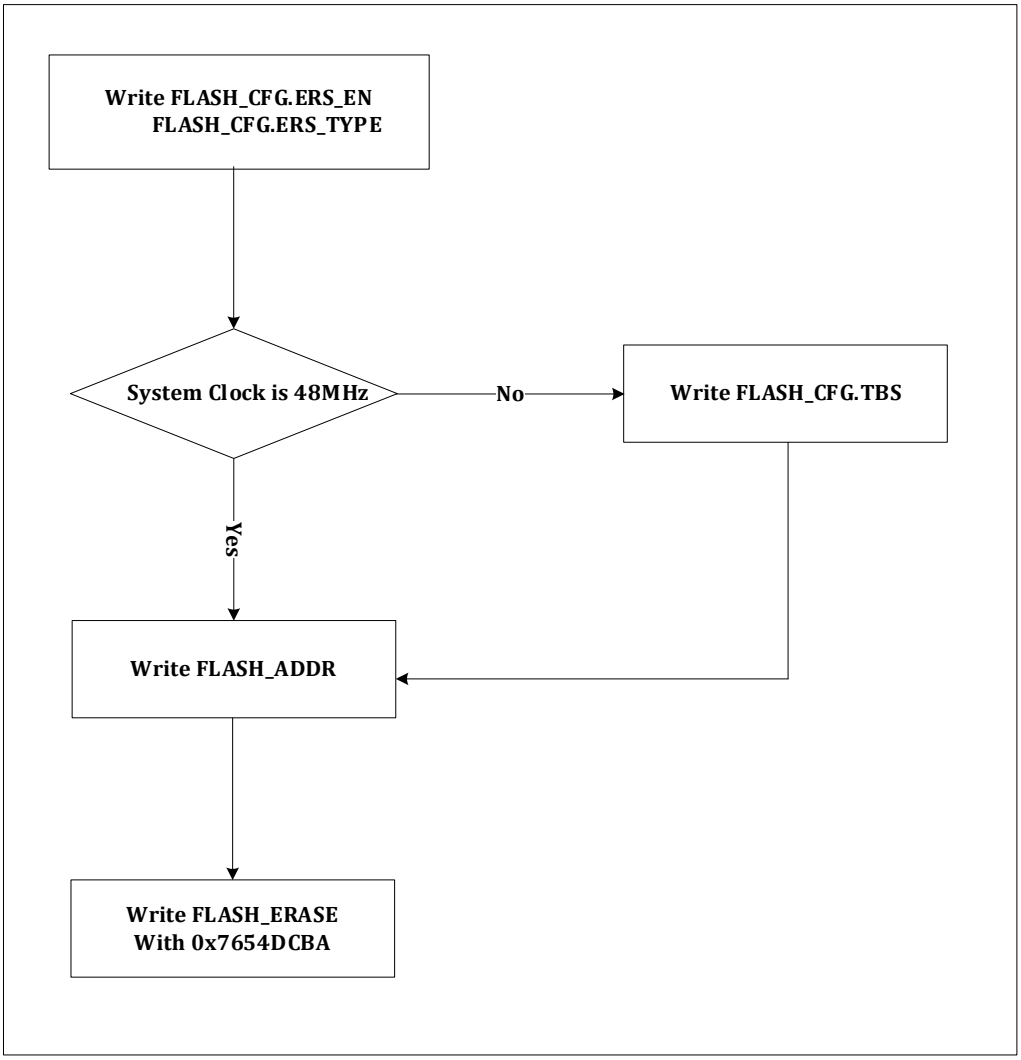


图 7-6 FLASH 模块擦除操作流程图

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。

注意，ROM 区域，不支持擦除操作。MAIN 区域不受此限制。MAIN 区域推荐使用库函数执行擦除操作。

7.2.1.6 FLASH 预取操作

因 FLASH 存储体的速度限制，无法达到 48MHz 的速度。对 FLASH 进行读取操作时，需要大于一个 48MHz 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 24MHz 及以下，对 FLASH 进行读取操作，无需大于一个时钟周期。此时，可以配置 FLASH_CFG.WAIT 为 0，去掉额外的等待。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 16KB 和 32KB，各自的最后一个 WORD 设计为保护字（16KB 对应 0x3FFC，32KB 对应 0x7FFC）。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）。

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 LKS32MC03x 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 7-2 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 LKS32MC03x 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 0x0001_0000 寄存器列表

表 7-3FLASH 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

7.3.2 配置寄存器 FLASH_CFG（推荐先读回，按或/与方式修改）

地址:0x0001_0000

复位值:0xB0

表 7-4 配置寄存器 FLASH_CFG

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

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

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
[7]	WAIT	读取 FLASH 数据，等待开关。默认为 1。 0:读取，无需等待一个周期

		1:读取，等待一个周期
[5:0]	TBS	编程/擦除时间基数寄存器,默认值为 0x30。只能配成如下几个值 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。 0x0B:12Mhz 系统频率下，FLASH 编程/擦除时间基数配置值。

7.3.3 地址寄存器 FLASH_ADDR

地址:0x0001_0004

复位值:0x0

表 7-5 地址寄存器 FLASH_ADDR

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 写数据寄存器 FLASH_WDATA

地址:0x0001_0008

复位值:0x0

表 7-6 写数据寄存器 FLASH_WDATA

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

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

位置	位名称	说明

[31:0]	WDATA	执行写入操作，写入 FLASH 的值

7.3.5 读数据寄存器 FLASH_RDATA

地址:0x0001_000C

复位值:0x0

表 7-7 读数据寄存器 FLASH_RDATA

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

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

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

7.3.6 擦除控制寄存器 FLASH_ERASE

地址:0x0001_0010

复位值:0x0

表 7-8 擦除控制寄存器 FLASH_ERASE

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

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

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

7.3.7 保护状态寄存器 FLASH_PROTECT

地址:0x0001_0014

复位值:0x0

表 7-9 保护状态寄存器 FLASH_PROTECT

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

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

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 工作状态寄存器 FLASH_READY

地址:0x0001_0018

复位值:0x0

表 7-10 工作状态寄存器 FLASH_READY

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

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。

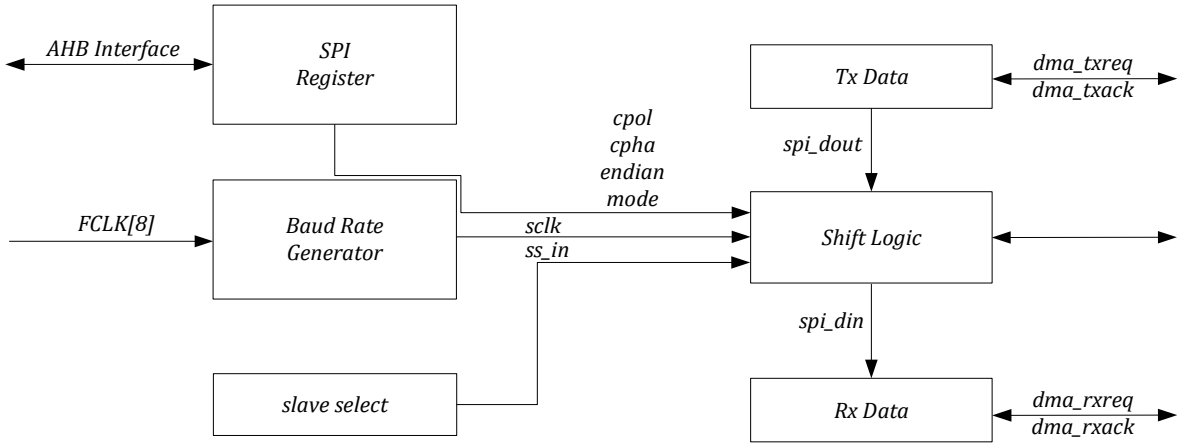


图 8-1 SPI 模块结构框图

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

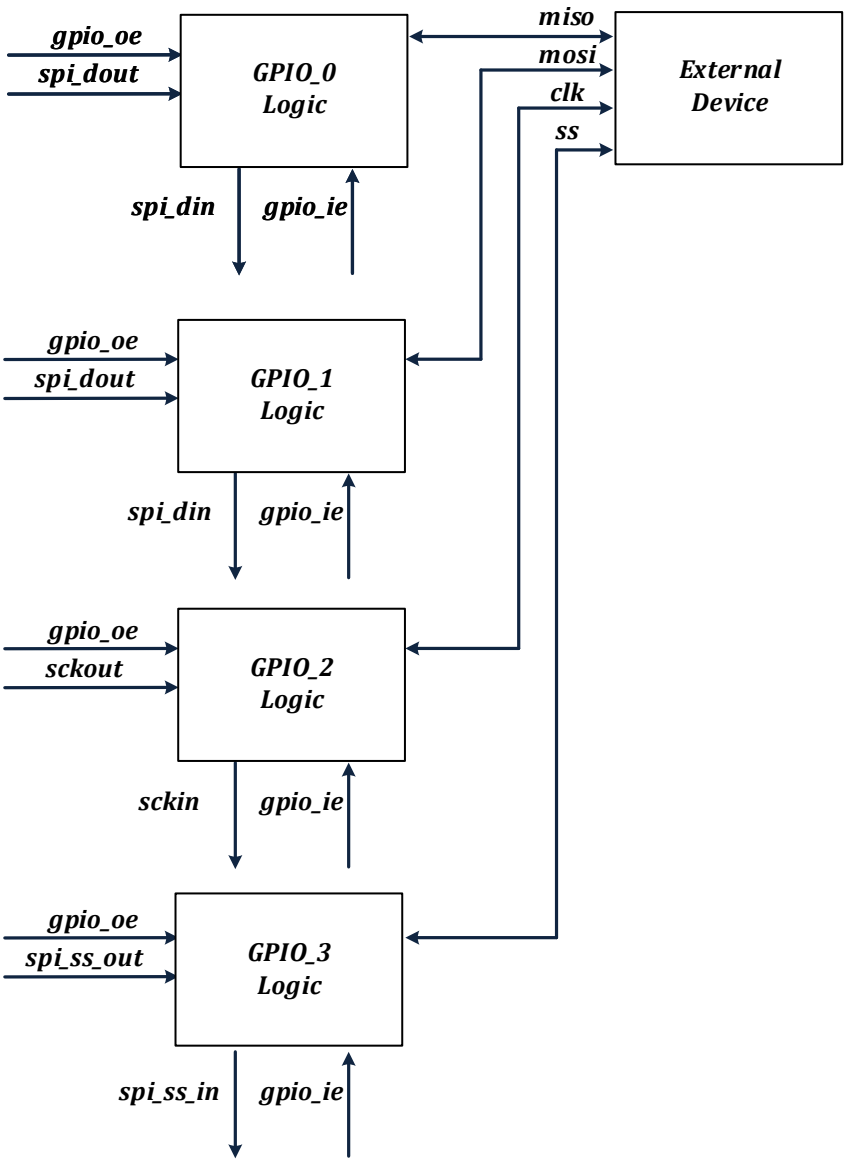


图 8-2 SPI 接口全双工模式互连框图

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。

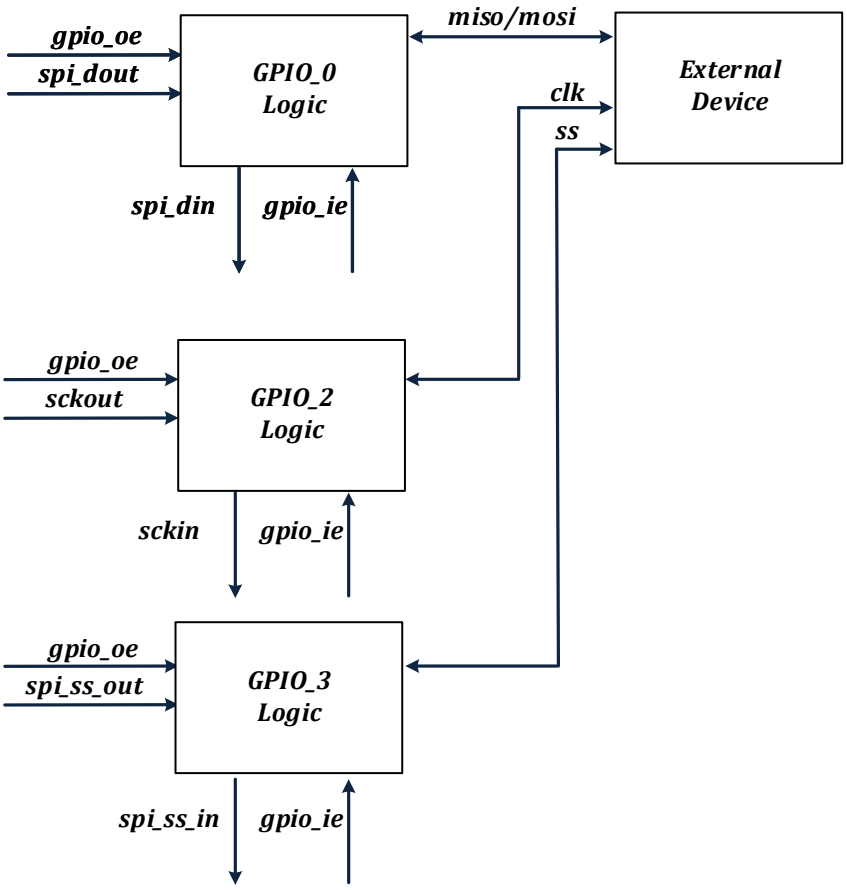


图 8-3 SPI 接口半双工模式互连框图

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。

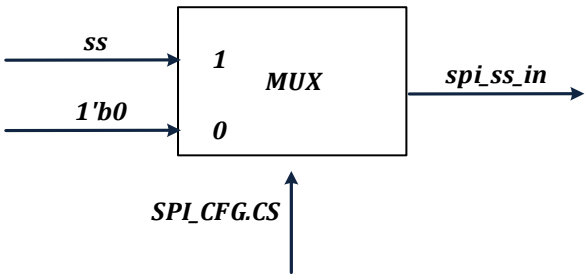


图 8-4 SPI 模块 Slave 模式片选信号选择

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。

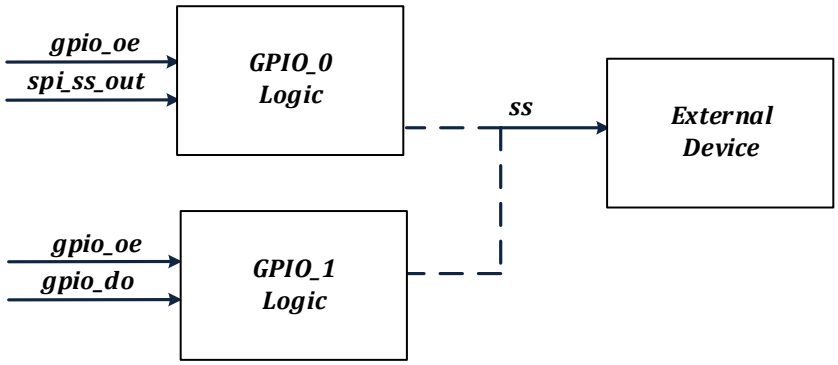


图 8-5 SPI 模块 Master 模式片选信号选择

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。

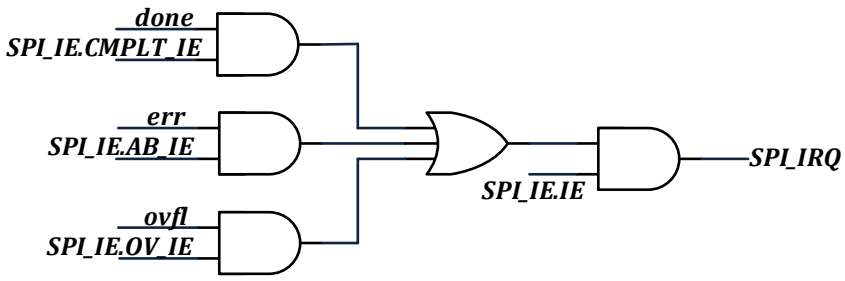


图 8-6 SPI 模块中断选信号产生图

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

SPI_CFG	0x00	SPI 配置寄存器
SPI_IE	0x04	SPI 中断寄存器
SPI_BAUD	0x08	SPI 波特率寄存器
SPI_TXDATA	0x0C	SPI 发送数据寄存器
SPI_RXDATA	0x10	SPI 接收数据寄存器
SPI_SIZE	0x14	SPI 传输数据长度寄存器

8.4.2 SPI_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 SPI_CFG

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

位置	位名称	说明
[31:8]		未使用
[7:6]	DUPLEX	半双工模式设置 00,01: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 SPI_IE SPI 中断寄存器

地址:0x4001_0004

复位值:0x0

表 8-3 SPI_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 SPI_BAUD SPI 波特率寄存器

地址:0x4001_0008

复位值:0x0

表 8-4 SPI_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
[7]	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。
[6]		未使用
[5:0]	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 SPI_TXDATA SPI 数据发送寄存器

地址:0x4001_000C

复位值:0x0

表 8-5 SPI_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 SPI_RXDATA SPI 数据接收寄存器

地址:0x4001_0010

复位值:0x0

表 8-6 SPI_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 SPI_SIZE SPI 数据传输长度寄存器

地址:0x4001_0014

复位值:0x0

表 8-7 SPI_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。

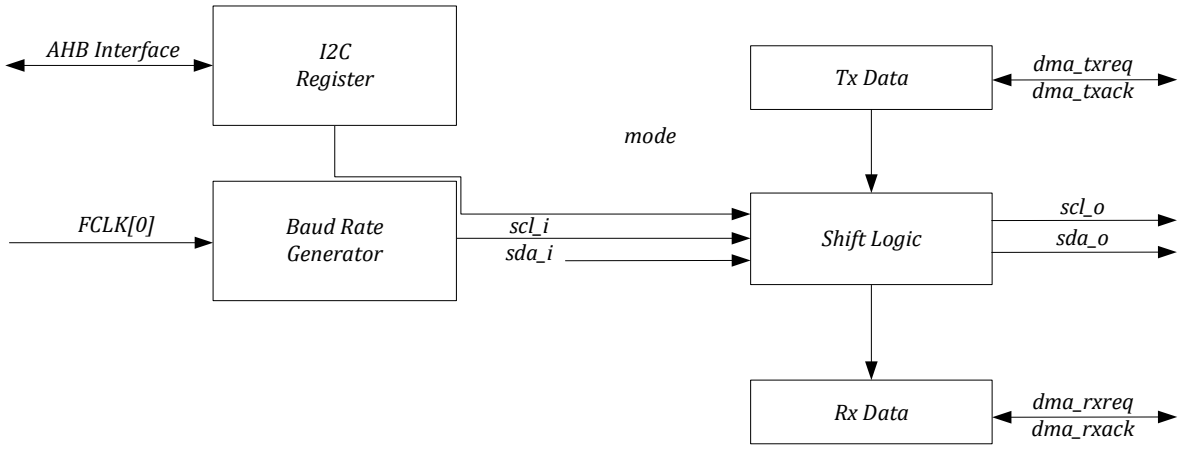


图 9-1I2C 模块顶层功能框图

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。

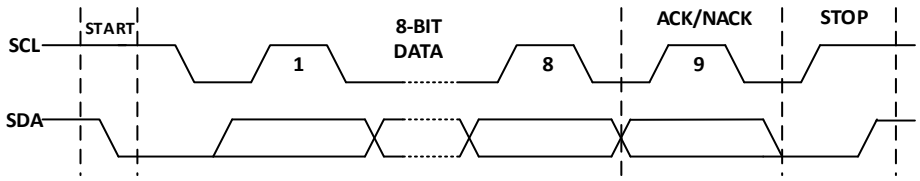


图 9-2 基本 I2C 传输时序图

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

从模式下可以配置进行单字节传输。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。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。

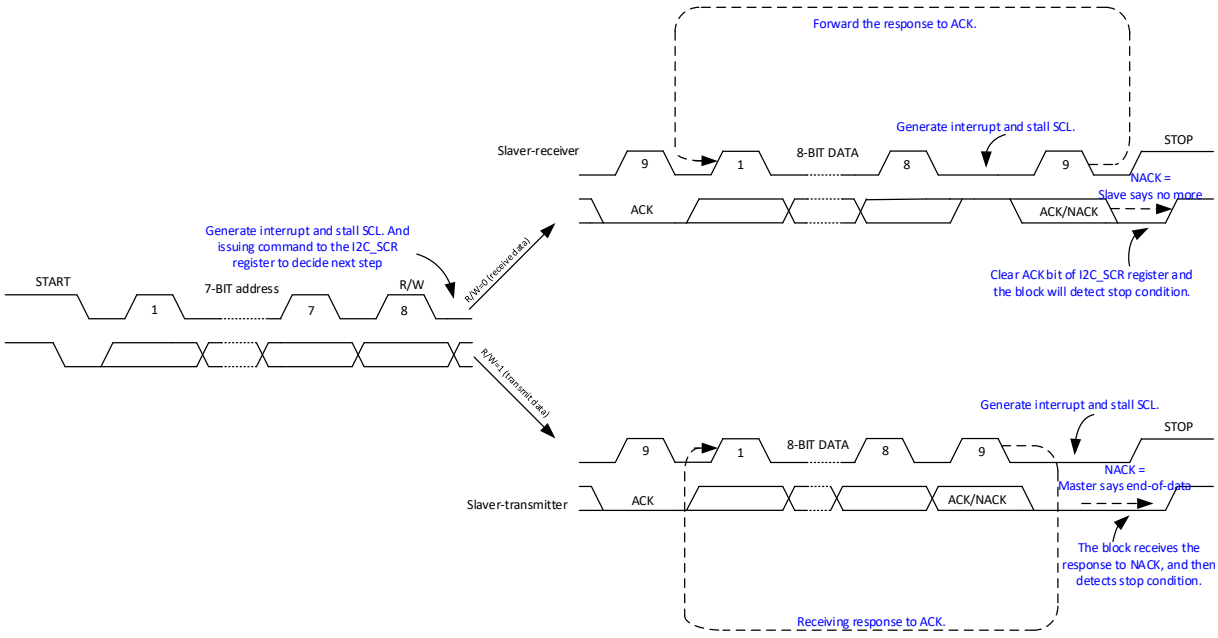


图 9-3 从模式传输示意图

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.BURST_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.BURST_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.BURST_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。

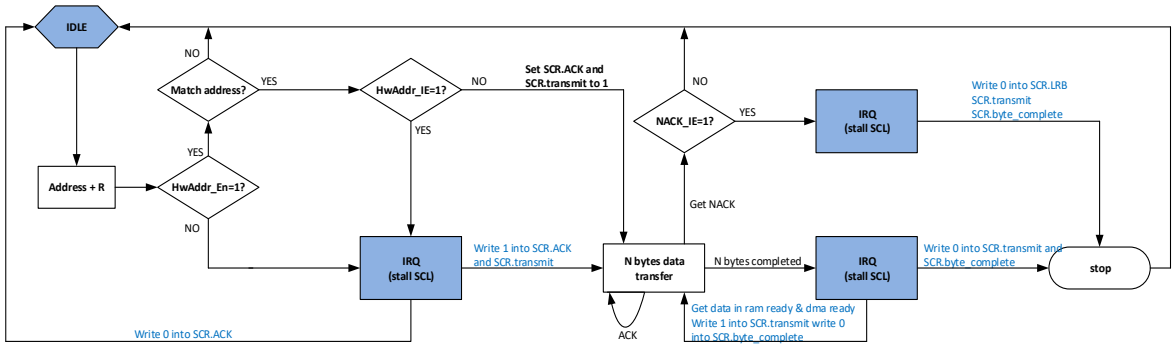


图 9-4 从模式下多字节发送示意图

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.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.BURST_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.BURST_SIZE 约定数据接收完毕后，产生一个 NACK 应答脉冲。

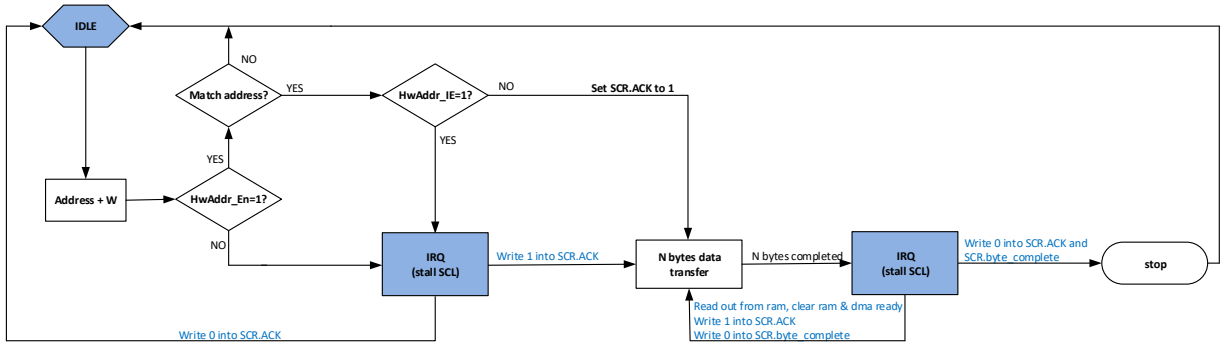


图 9-5 从模式下多字节接收示意图

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。

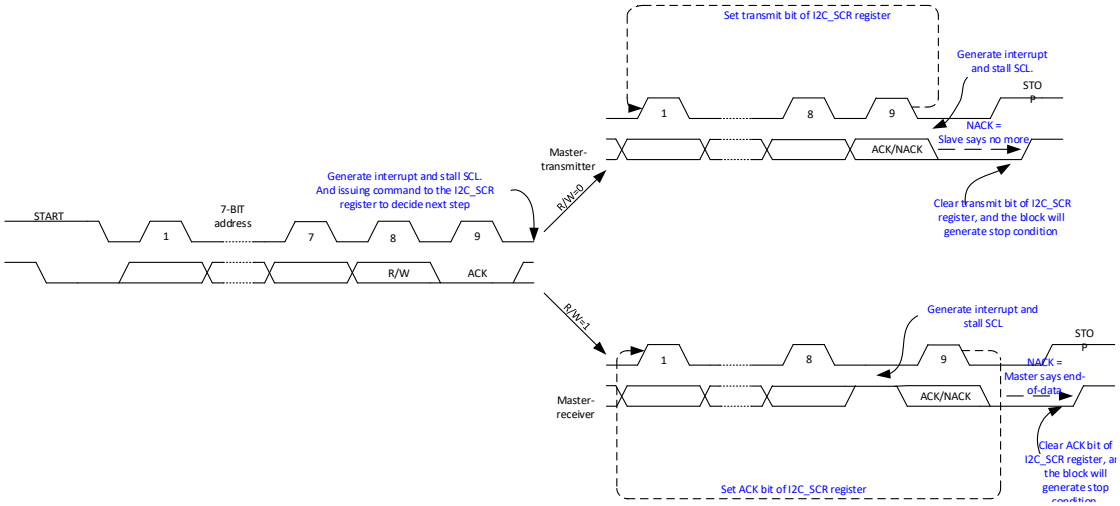


图 9-6 主模式下单字节传输示意图

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_BCR.BURST_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_BCR.BURST_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。

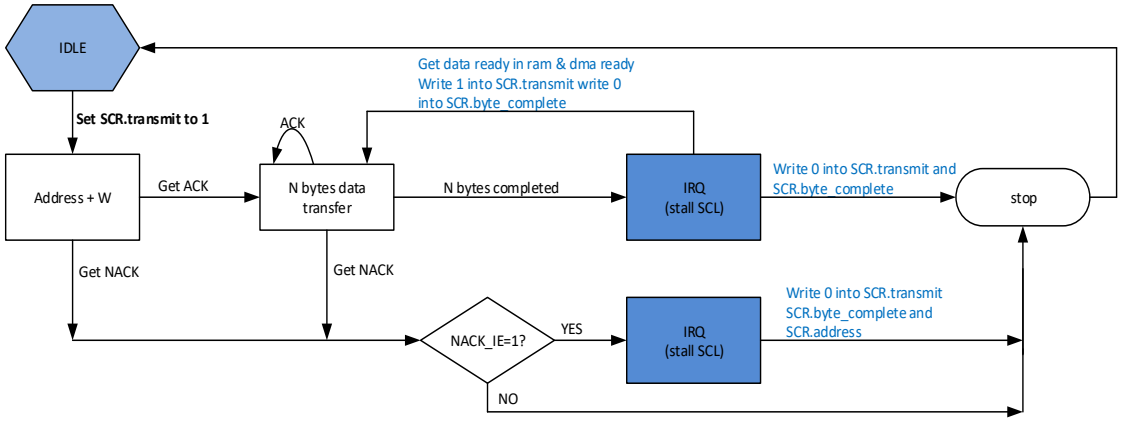


图 9-7 主模式下多字节发送示意图

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.BURST_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.BURST_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。

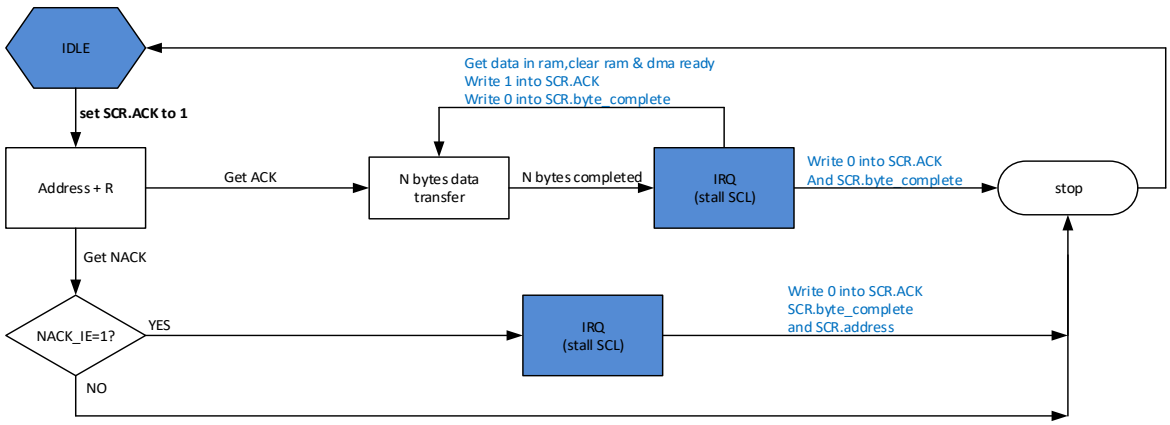


图 9-8 主模式下多字节接收示意图

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。数据和时钟信号的时钟频率为接口工作时钟/17。

- I2C模块工作时钟频率 = 系统频率 / (CLK_DIV0 + 1)
- 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
I2C_ADDR	0x00	I2C 地址寄存器
I2C_CFG	0x04	I2C 配置寄存器
I2C_SCR	0x08	I2C 状态寄存器
I2C_DATA	0x0C	I2C 数据寄存器
I2C_MSCR	0x10	I2C 主模式寄存器
I2C_BCR	0x14	I2C 传输控制寄存器
I2C_BSIZE	0x18	I2C 传输长度寄存器

9.4.2 I2C_ADDR 地址寄存器

地址:0x4001_0100

复位值:0x0

表 9-2 地址寄存器 I2C_ADDR

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

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 I2C_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 I2C_CFG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								IE	TC_IE	BUS_ERR_IE	STOP_IE			MST_MODE	SLV_MODE

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

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 I2C_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 I2C_SCR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								STT_ERR	LOST_ARB	STOP_EVT	BYTE_CMPLT	ADDR_DATA	DATA_DIR	RX_ACK	Done
								RW	RW	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

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 I2C_DATA 数据寄存器

地址:0x4001_010C

复位值:0x0

表 9-5 数据寄存器 I2C_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 I2C_MSCR 主模式寄存器

地址:0x4001_0110

复位值:0x0

表 9-6 主模式寄存器 I2C_MSCR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												BUSY	MST_CHECK	RESTART	START
												RW	RW	RW	RW
												0	0	0	0

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 I2C_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 I2C_BCR

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

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

9.4.8 I2C_BSIZE I2C 传输长度寄存器

地址:0x4001_0118

复位值:0x0

表 9-8 DMA 传输长度寄存器 I2C_BSIZE

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

位置	位名称	说明
[31:8]		未使用

[4:0]	SIZE	I I2C 数据传输长度寄存器，用于多字节传输。 实际传输字节数 = B[3:0] + 1

10 CMP 比较器

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 2 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。

比较器可用于以下功能：

1. 反电势过零点检测
2. 硬件过流检测
3. 作为 MCPWM 的 fail 信号来源

比较器的主要特性如下：

1. 每个比较器都具备多种的信号来源可供选择：
   - 多路 GPIO 口输入信号
   - 运放输出信号
   - 运放正端输出信号
   - 1.2V BANDGAP 基准源
   - DAC 输出信号
2. 比较速度可编程，迟滞电压可编程
3. 输出信号可滤波，滤波深度可选
4. 可产生 CMP 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 的第二功能送出，具体 GPIO 第二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。

推荐配置流程：

1. 配置 CMP 的模拟开关

   芯片上电的默认状态下，比较器模块是关闭的，通过配置寄存器 SYS_AFE_REG0.CMPxPDN 可以打开比较器（x=0/1，代表 CMP0/CMP1 两个比较器）。开启比较器之前，需要先开启 BGP 模块。

2. 配置 CMP 的数字时钟开关、时钟滤波和信号输入开关

   通过配置寄存器 CMP_TCLK.CLK10_EN 可以打开 CMP 的数字时钟开关，1 表示打开，0 表示关闭；通过配置寄存器 CMP_TCLK.FIL_CLK10_DIV16[7:4]可以配置滤波时钟，数值设置范围是 0~15，0 表示 1 分频滤波，15 表示 16 分频滤波；通过配置寄存器 CMP_CFG.CMPx_IN_EN 可以打开信号输入开关，1 表示打开，0 表示关闭。

3. 配置 CMP 的比较速度和迟滞电压

   比较器的比较延时可通过配置寄存器 SYS_AFE_REG1.CMP_FT 设置为＜30nS/200nS。迟滞电压可通过配置寄存器 SYS_AFE_REG1.CMP_HYS 设置为 20mV/0mV。

4. 选择 CMP 的正负端信号来源

   CMP 的正端信号有 8 种信号来源可选择，可以通过配置寄存器 SYS_AFE_REG1.CMPx_SELP 进行设置，负端有 4 种信号来源可选择，可以通过配置寄存器 SYS_AFE_REG1.CMPx_SELN 进行设置。

5. 配置 CMP 的中断

   通过配置寄存器 CMP_IE.CMPx_IE 可以打开 CMP 中断，1 表示打开，0 表示关闭。

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 比较器中断使能寄存器 CMP_IE

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

	CMP1_RE	CMP0_RE		CMP1_IE	CMP0_IE	
	RW	RW		RW	RW	
	0	0		0	0	

位置	位名称	说明
[31:10]		未使用
[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
[7:2]		未使用
[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 比较器中断标志寄存器 CMP_IF

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

位置	位名称	说明
[31:2]		未使用
[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_3008

复位值:0x0

表 10-4 比较器分频时钟控制寄存器 CMP_TCLK

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

	FIL_CLK10_DIV16	CLK10_EN	FIL_CLK10_DIV2
	RW	RW	RW
	0	0	0

位置	位名称	说明
[31:8]		未使用
[7:4]	FIL_CLK10_DIV16	比较器 1/0 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]	CLK10_EN	比较器 1/0 滤波时钟使能，高有效
[2:0]	FIL_CLK10_DIV2	比较器 1/0 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

CMP 滤波时钟频率使用如下公式进行计算

Freq(CMP_Filter) = Freq(MCLK)/$2^{CMP\_TCLK.FIL\_CLK\_DIV2}$/(CMP_TCLK.FIL_CLK_DIV16+1)，其中 FCLK[3]为 CMP 模块的主时钟，受 SYS_CLK_FEN[3]控制。需要注意的是，产生 CMP 滤波时钟需要使能 CMP_TCLK.CLK_EN 位。

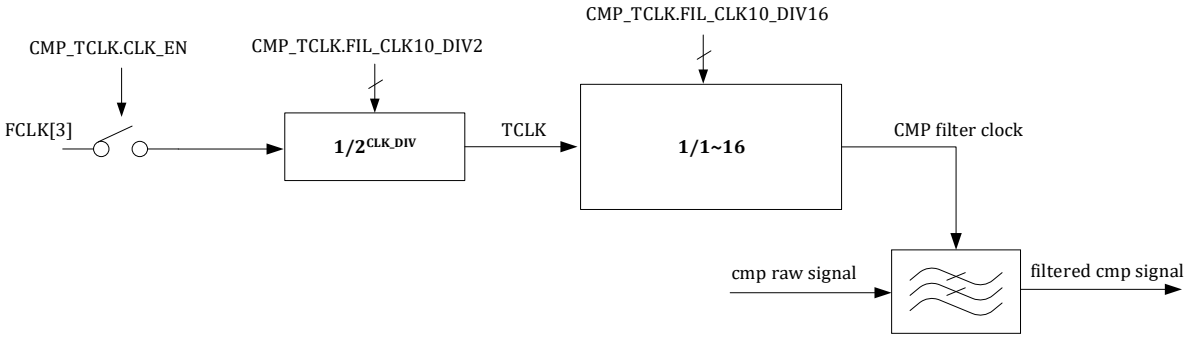


图 10-1 比较器滤波时钟产生

CMP 模块使用此滤波时钟对模拟比较器的输出信号进行 16 时钟周期长的滤波，即只有信号稳定时间超过 16 个滤波时钟周期才能通过滤波器，CMP 模块输出的滤波后的信号才会发生变化，如果输入信号稳定时间不足 16 个滤波时钟周期即发生变化，则 CMP 模块输出的滤波后的信号维持原值不变。**即滤波宽度=滤波时钟周期*16。**

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 1*1*16~128*16*16 个总线周期，

即 16~32768 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 比较器控制寄存器 CMP_CFG

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

位置	位名称	说明
[31:8]		未使用
[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。

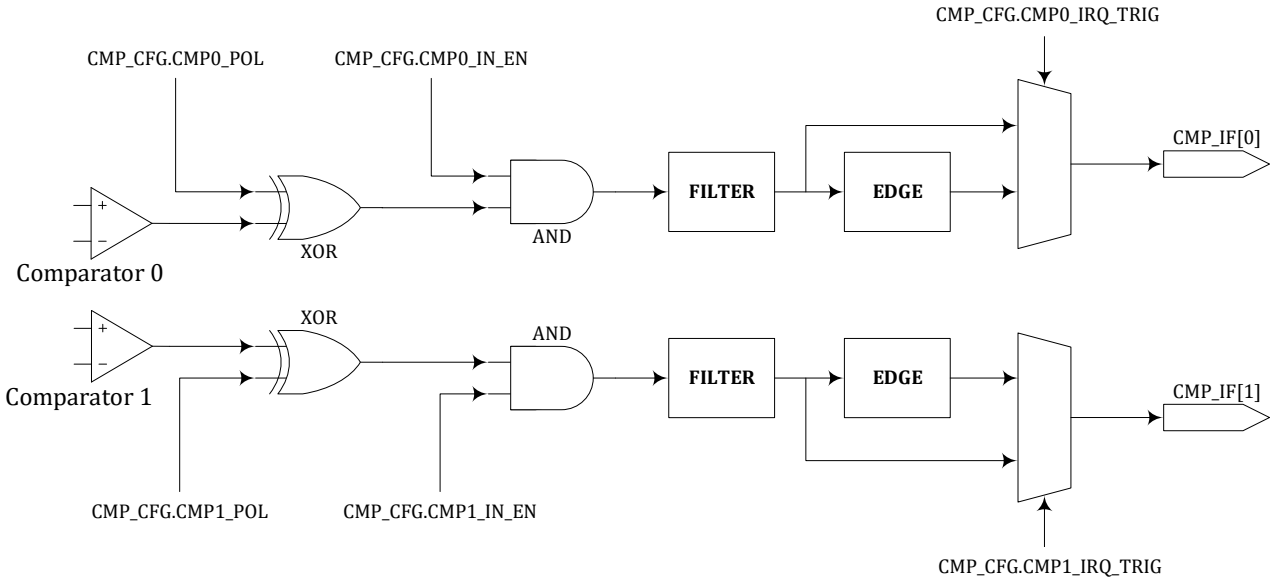


图 10-2 比较器控制及中断产生逻辑

比较器模块与 MCPWM 模块可以联合动作，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。

MCPWM 的 fail 信号可以来自 GPIO，也可以来自比较器模块，使用 MCPWM_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。

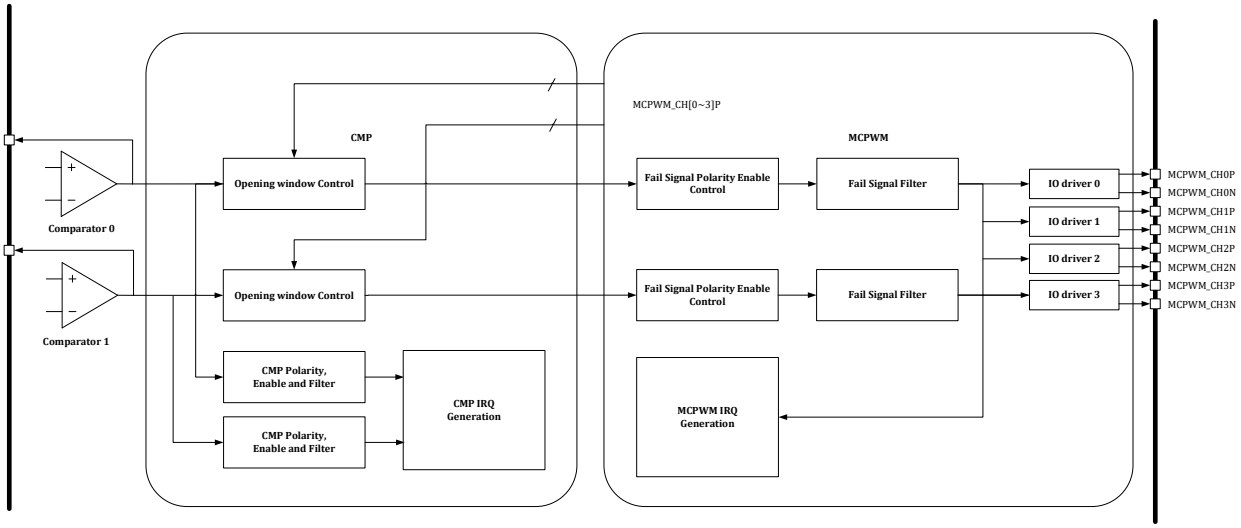


图 10-3 CMP 与 MCPWM 的联动

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。

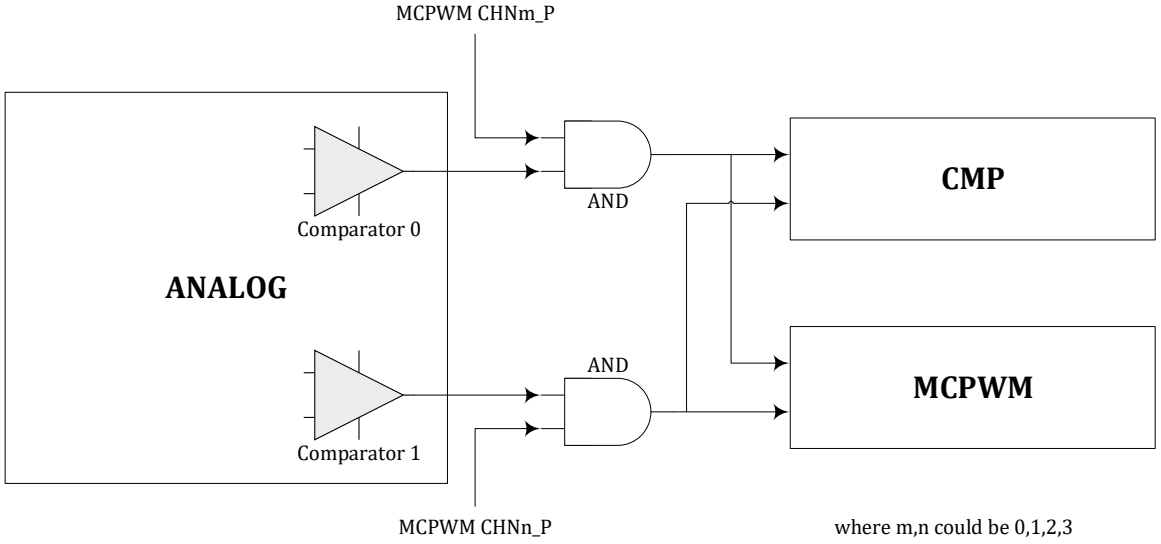


图 10-4 比较器开窗功能图示

10.2.6 CMP_BLCWIN 开窗控制寄存器

地址:0x4001_3010

复位值:0x0

表 10-6 比较器开窗控制寄存器 CMP_ BLCWIN

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								CMP1_CHN3P_WIN_EN	CMP1_CHN2P_WIN_EN	CMP1_CHN1P_WIN_EN	CMP1_CHN0P_WIN_EN	CMP0_CHN3P_WIN_EN	CMP0_CHN2P_WIN_EN	CMP0_CHN1P_WIN_EN	CMP0_CHN0P_WIN_EN
								RW	RW	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

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_3014

复位值:0x0

表 10-7 比较器输出数据寄存器 CMP_ DATA

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

位置	位名称	说明
[31:10]		未使用
[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
[2:7]		未使用
[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 信号处理模块

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:

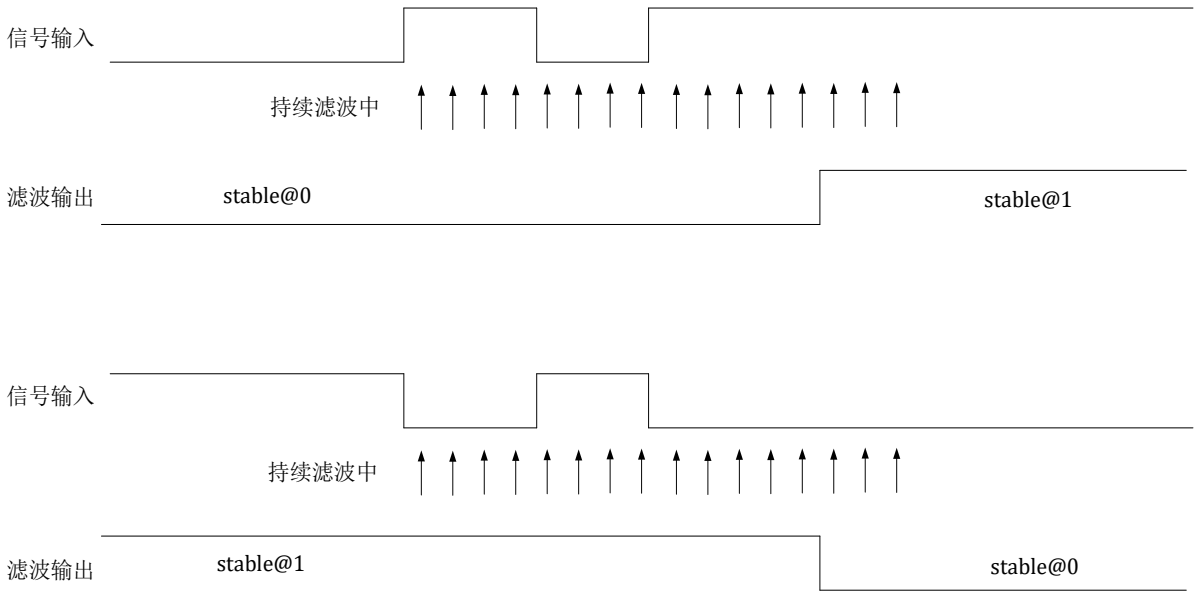


图 11-1 7/5 滤波模块框图

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 48MHz 工作频率下，周期 T_{clk} 是 20.8ns，寄存器配置最大为 32767，最长滤波宽度为约 20.8ns×32768≈680us。

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 48MHz 工作频率下，最大可以记录约 2. 8 秒的时间宽度，达到 20.8ns 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。中断标志可以通过对 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[1]为受 SYS_CLK_FEN.HALL_CLK_EN 门控信号控制的 HALL 时钟，开通后为 48MHz 的 PLL 时钟。

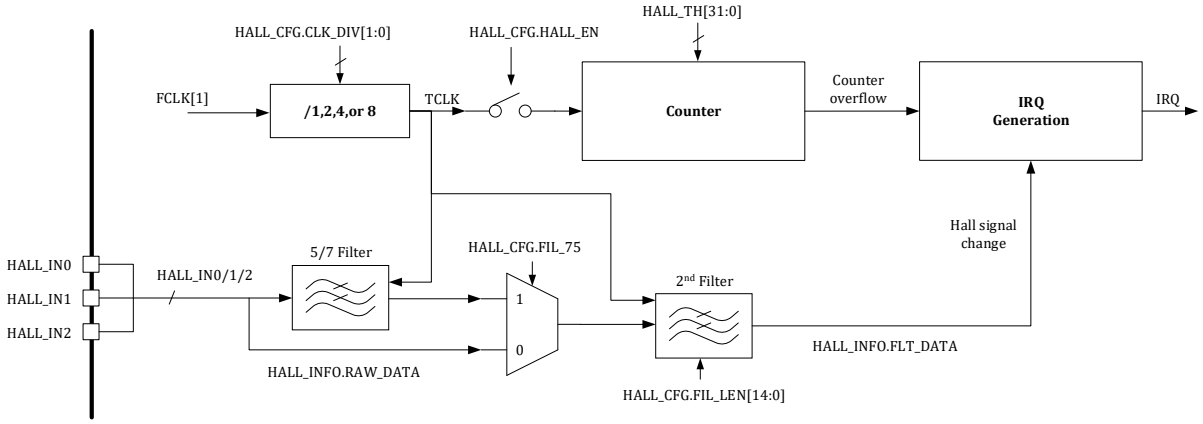


图 11-2 数据流程框图

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 11-1HALL 模块寄存器地址分配

名称	偏移	描述
HALL_CFG	0x00	HALL 模块配置寄存器
HALL_INFO	0x04	HALL 模块信息寄存器
HALL_WIDTH	0x08	HALL 宽度计数值寄存器
HALL_TH	0x0C	HALL 模块计数器门限值寄存器
HALL_CNT	0x10	HALL 计数寄存器

11.3.2 HALL 模块配置寄存器 HALL_CFG

地址:0x4001_0300

复位值:0x0

表 11-2 HALL 模块配置寄存器 HALL_CFG

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

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

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

	FIL_LEN
	RW
	0

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 HALL 模块信息寄存器 HALL_INFO

地址:0x4001_0304

复位值:0x0

表 11-3 HALL 模块信息寄存器 HALL_INFO

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

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

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 HALL 宽度计数值寄存器 HALL_WIDTH

地址:0x4001_0308

复位值:0x0

表 11-4 HALL 宽度计数值寄存器 HALL_WIDTH

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

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

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 HALL 模块计数器门限值寄存器 HALL_TH

地址:0x4001_030C

复位值:0x0

表 11-5 HALL 模块计数器门限值寄存器 HALL_TH

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

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

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 HALL 计数寄存器 HALL_CNT

地址:0x4001_0310

复位值:0x0

表 11-6 HALL 计数寄存器 HALL_CNT

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

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

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

LKS32MC03x 系列芯片集成了 1 个 12BIT SARADC。

- 1MSPS 采样率，24MHz 工作频率。
- 支持 14 通道选择。
- 支持软件、硬件触发功能。
- 可以与 MCPWM、UTimer 单元联动进行定时触发
- 支持触发指示信号可通过 GPIO 送出调试
- 支持单段、双段、四段自定义采样序列采样，序列次数和通道号可灵活配置。
- 支持左对齐、右对齐模式。

ADC 采样的量词含义约定:

1 次采样:完成对应的一个通道的模拟信号量到数字信号量的采样转换并存储数字量至 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量通道，也可以是不同的模拟量通道。采样开始通常由 MCPWM、UTimer 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；以 MCPWM 触发的四段采样为例，每段采样 3 次（即完成 3 个模拟量的采样），TADC[0]触发 ADC 开始第一段采样，第一段采样完成后 ADC 进入等待状态，等待 TADC[1]触发事件发生；TADC[1]发生后，触发 ADC 开始第二段采样；同理，TADC[2]/TADC[3]分别触发第三段和第四段采样。

1 轮采样:可能包含 1 段、2 段或 4 段采样，每段分别由特定触发信号触发；ADC 完成一轮采样后回归空闲状态等待下次触发

12.1.1 功能框图

ADC 接口包括 10 个数据寄存器（ADC 在最多 10 次采样中存储的数字量），以及若干控制寄存器。

数据寄存器 ADC_DATx 用于存储 ADC 第 x 次采样得到的数据量。被转换的模拟信号来源由寄存器 ADC_CHNx 中的 4bit 进行选择（详见 12.2.4）。以 ADC_CHN0 为例，BIT[3:0]选择第 0 次采样的模拟通道号，通道号 CH0~CH15 任选，若 ADC_CHN0[3:0]=0，ADC_CHN0[7:4]=3，则第 0 个采样的模拟量对应通道 CH0，第 1 个采样的模拟量对应通道 CH3，以此类推。

分段采样次数寄存器 ADC_CHNT 控制每段采样的次数，1~15 对应 1~15 次。

控制逻辑根据配置寄存器 ADC_CFG 选择来自 MCPWM 或 UTimer 的触发信号启动一段采样或者软件触发启动。MCPWM/UTimer 会送出定时触发信号 TADC[0]/TADC[1]/TADC[2]/TADC[3]，可选择 TADC[0]/TADC[1]/TADC[2]/TADC[3]作为触发信号。触发信号的选择保存在控制寄存器中。

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。多段触发模式下，每一段转换完成可触发产生一个转换完成中断。

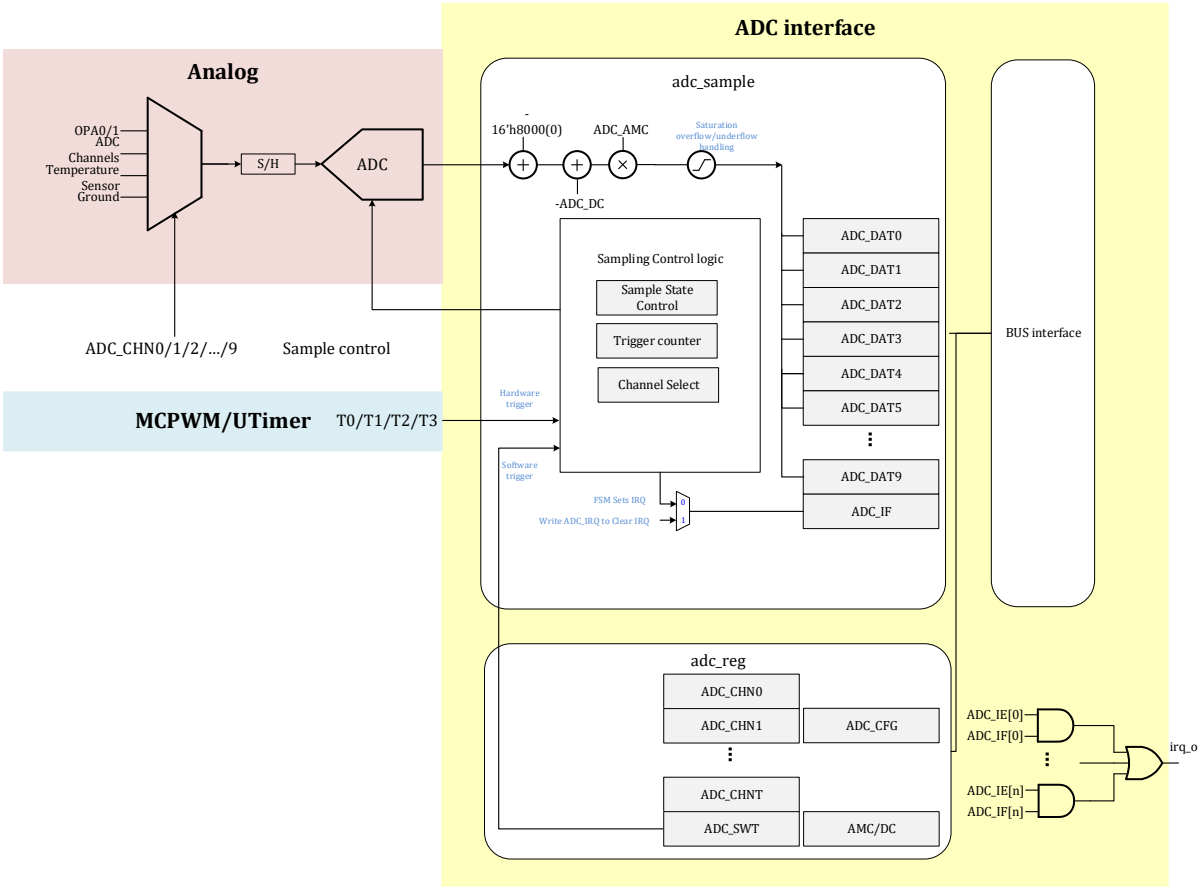


图 12-1 ADC 采集模块功能框图

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。

ADC 最高工作时钟为 24MHz，采样时间 5-37 个 ADC 时钟周期（Cycle）可配置，数据转换时间固定为 12Cycle。以采样时间 5Cycle 为例，采样+转换最少 17Cycle，即 ADC 最高转换速率是 24/17=1.4Msps。但考虑到采样建立时间，建议设置采样时间 8Cycle 或以上，对应转换速度率 24/(8+12)=1.2Msps。采样时间可以通过寄存器 SYS_AFE_REG2 设置。

12.1.2 ADC 触发方式

- 支持单段触发、两段触发、四段触发完成采样
- 单段触发可以设置触发事件发生次数
- 两段触发的触发源只能为 MCPWM/UTimer 的定时信号 TADC[0]+TADC[1]，或两次软件触发
- 四段触发的触发源只能为 MCPWM/UTimer 的定时信号 TADC[0]+TADC[1]+TADC[2]+TADC[3]，四次硬件触发事件必须是按顺序发生；或四次软件触发
- 每段触发完成均可产生中断

- 触发指示信号可以通过 GPIO 送出用于调试

12.1.3 ADC 输出数制

ADC 输出数据为 12bit 补码，输入信号 0 对应 12h'0000_0000_0000，以 2.4V 量程为例，输入信号-2.4V 对应 12h'1000_0000_0000，输入信号+2.4V 对应 12h'0111_1111_1111。ADC 转换后的 12bit 补码需扩展为 16BIT 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'h1000_0000_1101 为例，如果配置为左对齐，右侧补 4 个 0，存入 ADCx_DAT 的值为 16'h1000_0000_1101_0000；如果配置为右对齐，左侧进行符号扩展，存入 ADCx_DAT 的值为 16'h1111_1000_0000_1101。推荐统一使用左对齐方式。

表 12-1 ADC 输出数字量数制转换

ADC 2.4V 量程输入模拟量数值/V	ADC 3.6V 量程输入模拟量数值/V	转为有符号数后的数值
2.4	3.6	12'h0111_1111_1111
0	0	12'h0000_0000_0000
-2.4	-3.6	12'h1000_0000_0000

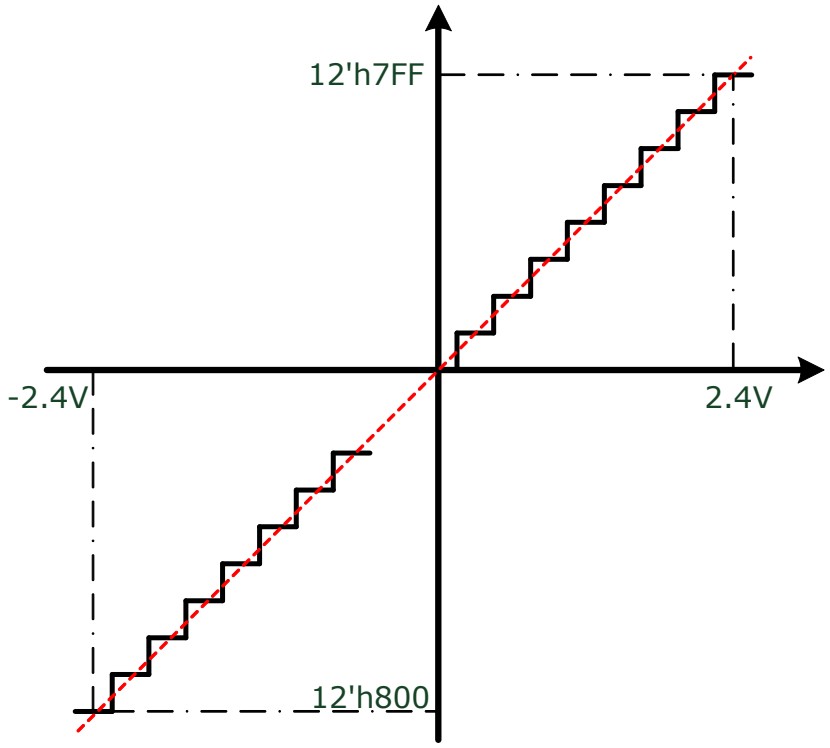


图 12-2 ADC 2.4V 量程设置下转换数制量程

12.1.4 ADC 量程

ADC 有两种量程：2.4V 和 3.6V。2.4V 量程模式下，对应最大±2.4V 的输入信号幅度；3.6V 量程模式模式下，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~OPA1），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（放大倍数选择方式详见 5.2.5），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 的量程配置为 2.4V。

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 量程。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。

两种量程选择由 SYS_AFE_REG0.GA_AD 寄存器进行控制。不同于 LKS32MC08x/05x，量程仅能通过软件直接控制寄存器进行选择，不支持硬件根据通道配置自动切换。

12.1.5 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。

ADC_DC 存储的是 ADC 的直流偏置，通常在校正阶段通过测量通道 15（从 0 开始计数）的 AVSS（内部地）得到 ADC 直流偏置数值并存入 flash 中,并在系统加载阶段由软件将直流偏置写入 ADC_DC 寄存器中。

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

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

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。

12.1.6 ADC 阈值监测(模拟看门狗)

ADC 接口模块配备有 1 组阈值监测电路，阈值监测可以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。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，如果写入数据大于上阈值或小于下阈值则置位对应的阈值超限中断标志。

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

12.1.7 ADC 配置流程

推荐配置流程:

1. 打开 ADC 模拟开关，选择 ADC 工作频率

   通过配置寄存器 SYS_AFE_REG0.ADCPDN 可以开启 ADC，ADC 开启前，需要先开启 BGP、4M RC 时钟和 PLL 模块。

2. 配置 ADC 数据输出格式

   ADC 的输出格式可配置为左对齐或者右对齐,配置的是 ADC_CFG.DATA_ALIGN 寄存器，0 为左对齐，1 为右对齐。

3. 配置 ADC 采样模式

   ADC 的采样模式可配置为单段、双段、四段采样模式，配置的是 ADC_CFG.TRG_MODE[9:8]寄存器，00 为单段采样模式，01 为两段采样模式，11 为四段采样模式。

4. 配置 ADC 触发事件

   ADC 采样的触发事件选择配置的是 ADC_CFG.SEL 寄存器。单段采样模式下可以设置触发一次采样所需的事件数，配置的是 ADC_CFG.SINGLE_TCNT[7:4]寄存器，设置范围是 0~15，0 表示一次事件即触发，15 表示 16 次事件才触发。

5. 配置 ADC 量程

   ADC 的两种增益模式可以通过配置 SYS_AFE_REG0.GA_AD 增益寄存器进行选择，0 为低增益（2/3 倍），1 为高增益（1 倍）。

6. 配置 ADC 通道数，选择采样信号源

   配置各段采样模式下，采样的通道个数，配置的是 ADC_CHNT 寄存器，设置范围是 1~16，1 代表一个通道。ADC 的采样信号源配置，通过配置 ADC_CHN0、ADC_CHN1、ADC_CHN2 寄存器选择。

7. 配置 ADC 中断

   通过配置 ADC_IE 寄存器可以使能 ADC 中断。即使未开启中断，中断事件仍能置位 ADC_IF，但不会提出中断请求。

12.2 寄存器

12.2.1 地址分配

ADC 在芯片中的基地址是 0x4001_0400；

表 12-2 ADC0 寄存器列表

名称	偏移地址	说明
ADC_DAT0	0x00	ADC 通道第 0 次采样数据
ADC_DAT1	0x04	ADC 通道第 1 次采样数据
ADC_DAT2	0x08	ADC 通道第 2 次采样数据
ADC_DAT3	0x0C	ADC 通道第 3 次采样数据
ADC_DAT4	0x10	ADC 通道第 4 次采样数据
ADC_DAT5	0x14	ADC 通道第 5 次采样数据
ADC_DAT6	0x18	ADC 通道第 6 次采样数据
ADC_DAT7	0x1C	ADC 通道第 7 次采样数据
ADC_DAT8	0x20	ADC 通道第 8 次采样数据
ADC_DAT9	0x24	ADC 通道第 9 次采样数据
ADC_LTH	0x30	ADC 模拟看门狗低阈值

ADC_HTH	0x34	ADC 模拟看门狗高阈值
ADC_GEN	0x38	ADC 模拟看门狗监测使能
ADC_CHN0	0x40	ADC 第 0~3 次采样信号选择
ADC_CHN1	0x44	ADC 第 4~7 次采样信号选择
ADC_CHN2	0x48	ADC 第 8~11 次采样信号选择
	0x4C	
ADC_CHNT	0x50	ADC 各种触发模式下采样通道数
ADC_CFG	0x54	ADC 对齐模式配置
ADC_SWT	0x58	ADC 软件触发
ADC_DC	0x60	ADC DC offset
ADC_AMC	0x64	ADC 增益校正
ADC_IE	0x68	ADC 中断使能
ADC_IF	0x6C	ADC 中断标志

12.2.2 采样数据寄存器

12.2.2.1 ADC_DAT0

地址:0x4001_0400

复位值:0x0

表 12-3 采样数据寄存器 ADC_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	ADC 第 0 次采样数据

12.2.2.2 ADC_DAT1

地址:0x4001_0404

复位值:0x0

表 12-4 采样数据寄存器 ADC_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	ADC 第 1 次采样数据

12.2.2.3 ADC_DAT2

地址:0x4001_0408

复位值:0x0

表 12-5 采样数据寄存器 ADC_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	ADC 第 2 次采样数据

12.2.2.4 ADC_DAT3

地址:0x4001_040C

复位值:0x0

表 12-6 采样数据寄存器 ADC_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	ADC 第 3 次采样数据

12.2.2.5 ADC_DAT4

地址:0x4001_0410

复位值:0x0

表 12-7 采样数据寄存器 ADC_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	ADC 第 4 次采样数据

12.2.2.6 ADC_DAT5

地址:0x4001_0414

复位值:0x0

表 12-8 采样数据寄存器 ADC_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	ADC 第 5 次采样数据

12.2.2.7 ADC_DAT6

地址:0x4001_0418

复位值:0x0

表 12-9 采样数据寄存器 ADC_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	ADC 第 6 次采样数据

12.2.2.8 ADC_DAT7

地址:0x4001_041C

复位值:0x0

表 12-10 采样数据寄存器 ADC_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	ADC 第 7 次采样数据

12.2.2.9 ADC_DAT8

地址:0x4001_0420

复位值:0x0

表 12-11 采样数据寄存器 ADC_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	ADC 第 8 次采样数据

12.2.2.10 ADC_DAT9

地址:0x4001_0424

复位值:0x0

表 12-12 采样数据寄存器 ADC_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	ADC 第 9 次采样数据

12.2.3 模拟看门狗

12.2.3.1 ADC_LTH

地址为:0x4001_0430

复位值:0x0000_F800

表 12-13 下阈值寄存器 ADC_ 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

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

12.2.3.2 ADC_HTH

地址为:0x4001_0434

复位值:0x0000_07FF

表 12-14 上阈值寄存器 ADC_ 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

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

12.2.3.3 ADC_GEN

地址为:0x4001_0438

复位值:0x0

表 12-15 监测使能寄存器 ADC_ 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
[31:10]		未使用
[9:0]	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 …… BIT9: DAT9 看门狗监测使能

12.2.4 信号来源寄存器

12.2.4.1 ADC_CHN0

地址:0x4001_0440

复位值:0x0

表 12-16 信号来源寄存器 ADC_CHN0

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

0	0	0	0

位置	位名称	说明
[31:16]		未使用
[15:12]	DS3	ADC 第 3 次采样信号选择
[11:8]	DS2	ADC 第 2 次采样信号选择
[7:4]	DS1	ADC 第 1 次采样信号选择
[3:0]	DS0	ADC 第 0 次采样信号选择

12.2.4.2 ADC_CHN1

地址:0x4001_0444

复位值:0x0

表 12-17 信号来源寄存器 ADC_CHN1

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

位置	位名称	说明
[31:16]		未使用
[15:12]	DS7	ADC 第 7 次采样信号选择
[11:8]	DS6	ADC 第 6 次采样信号选择
[7:4]	DS5	ADC 第 5 次采样信号选择
[3:0]	DS4	ADC 第 4 次采样信号选择

12.2.4.3 ADC_CHN2

地址:0x4001_0448

复位值:0x0

表 12-18 信号来源寄存器 ADC_CHN2

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

位置	位名称	说明
[31:8]		未使用
[7:4]	DS9	ADC 第 9 次采样信号选择
[3:0]	DS8	ADC 第 8 次采样信号选择

12.2.4.4 采样信号选择

表 12-19 ADC 采样通道信号选择

ADC_CHNx.DSn 设置值	采样的模拟通道
0x0	OPA0_OUT
0x1	ADC_CH1
0x2	ADC_CH2
0x3	ADC_CH3
0x4	ADC_CH4
0x5	ADC_CH5
0x6	ADC_CH6
0x7	ADC_CH7
0x8	ADC_CH8/OPA1_OUT
0x9	ADC_CH9
0xA	ADC_CH10
0xB	温度传感器
0xC	内部地
0xD	2.4V 基准源

12.2.5 分段通道数寄存器

12.2.5.1 ADC_CHNT

地址:0x4001_0450

复位值:0x0

表 12-20 分段通道数寄存器 ADC_CHNT

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

位置	位名称	说明
[31:16]		未使用
[15:12]	S4	四段采样模式下第四段采样次数
[11:8]	S3	四段采样模式下第三段采样次数
[7:4]	S2	两段或四段采样模式下第二段采样次数
[3:0]	S1	单段、两段或四段采样模式下第一段采样次数

注意:每段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，…… ，12 表示 12 个通道，…… ，16 表示 16 个通道。

12.2.6 配置寄存器

12.2.6.1 ADC_CFG

地址:0x4001_0454

复位值:0x0

表 12-21 模式配置寄存器 ADC_CFG

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

位置	位名称	说明
[31:13]		未使用
[12]	SEL	TADC 触发来源选择。0:MCPWM，1:UTimer
[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 读出为 1，说明 ADC 正在进行转换，出于 busy 状态
[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
[9:8]	TRG_MODE	触发模式选择 0:单段触发；1:两段触发；2:保留；3:四段触发
[7:4]	SINGLE_TCNT	单段触发模式下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
[3:0]	TRG_EN	TADC 触发 ADC 采样使能信号。高电平有效，默认为低。 TRG_EN[0]:TADC[0]使能开关 TRG_EN[1]:TADC[1]使能开关 TRG_EN[2]:TADC[2]使能开关 TRG_EN[3]:TADC[3]使能开关 对应位置高表示使能，拉低表示关闭。

ADC 的触发来源为 MCPWM 或 UTimer，不可以同时使用 MCPWM 和 UTimer 进行触发，只能在二者之中选一。MCPWM/UTimer 可以提供的触发事件共计四个--TADC[3:0]。

若 ADC 的触发来源为 UTimer，ADC 采样触发事件为 UTimer_T0/ UTimer_T1/ UTimer_T2/ UTimer_T3；其依次对应 Timer0 通道 0/1、Timer1 通道 0/1 的比较事件。

若 ADC 的触发来源为 MCPWM。ADC 采样触发事件为 MCPWM 产生的 MCPWM_T0/ MCPWM_T1/ MCPWM_T2/ MCPWM_T3，分别是 MCPWM 内部计数器计数到 MCPWM_TMR0/1/2/3 时产生的事件。

UTimer 对 ADC 的触发信号可以通过配置 GPIO 为第 7/8 功能，即通过 Timer0/1 功能送出用于捕捉调试。

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。

12.2.7 软件触发寄存器

12.2.7.1 ADC_SWT

地址:0x4001_0458

复位值:0x0

表 12-22 软件触发寄存器 ADC_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，产生一次软件触发

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 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，等待后续的软件触

发到来。

12.2.8 直流偏置寄存器

ADC 的一个信号源为 GND，通过对这个通道的测量可以得到 ADC 模块的 DC 偏置。

通常芯片在出厂测试时会将 DC 偏置写入 flash 信息区,芯片上电后会自动从 flash 信息区加载 DC 偏置值，存入 ADC_DC 寄存器，后续每次其他通道的信号的采样值会自动减去 ADC_DC，再存入对应的数字量寄存器(ADCx_DAT)。

考虑到信号误差，去除 DC offset 的信号可能会发生溢出，对于溢出的情况 ADC 接口模块会进行饱和处理，以使信号范围在-2.4V~+2.4V 或-3.6V~+3.6V 之间。

12.2.8.1 ADC_DC

地址:0x4001_0460

复位值:0x0

表 12-23 直流偏置寄存器 ADC_DC

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
[31:8]		未使用
[7:0]	DC_OFFSET	采样电路增益 ADC DC offset

考虑到 ADC 的 DC offset 是接近 0 的数值，高 8bit 仅仅是 ADC_DC[7]的符号扩展，同时 ADC_DC 寄存器参与 ADC 数据校正时也会进行符号扩展。如果向 ADC_DC 写入 0x10B0; 实际写入的是 0xB0，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 ADC_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 ADC_DC 参与 ADC 校正。具体来说，右对齐时，ADC_DC[15:0]直接参与校正运算，左对齐时，ADC_DC[15:0]会先左移 4 位然后参与 ADC 校正运算。

12.2.9 增益校正寄存器

12.2.9.1 ADC_AMC

地址:0x4001_0464

复位值:0x200

表 12-24 增益校正寄存器 ADC_AMC

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	采样电路 ADC 增益校正寄存器

12.2.10 中断寄存器

12.2.10.1 ADC_IE

地址:0x4001_0468

复位值:0x0

表 12-25 中断使能寄存器 ADC_IE

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

位置	位名称	说明
[31:15]		未使用
[14]	AWD_RE	模拟看门狗超限 DMA 请求使能
[13]	HERR_RE	硬件触发发生在非空闲状态 DMA 请求使能
[12]	SERR_RE	软件触发发生在非空闲状态 DMA 请求使能
[11]	S4_RE	第四段采样完成 DMA 请求使能
[10]	S3_RE	第三段采样完成 DMA 请求使能
[9]	S2_RE	第二段采样完成 DMA 请求使能
[8]	S1_RE	第一段采样完成 DMA 请求使能
[7]		未使用
[6]	AWD_IE	模拟看门狗超限中断使能
[5]	HERR_IE	硬件触发发生在非空闲状态中断使能
[4]	SERR_IE	软件触发发生在非空闲状态中断使能
[3]	S4_IE	第四段采样完成中断使能

[2]	S3_IE	第三段采样完成中断使能
[1]	S2_IE	第二段采样完成中断使能
[0]	S1_IE	第一段采样完成中断使能

12.2.10.2 ADC_IF

地址:0x4001_046C

复位值:0x0

表 12-26 中断标志寄存器 ADC_IF

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

位置	位名称	说明
[31:7]		未使用
[6]	AWD_IF	模拟看门狗超限中断标志
[5]	HERR_IF	硬件触发发生在非空闲状态中断标志
[4]	SERR_IF	软件触发发生在非空闲状态中断标志
[3]	S4_IF	第四段采样完成中断标志
[2]	S3_IF	第三段采样完成中断标志
[1]	S2_IF	第二段采样完成中断标志
[0]	S1_IF	第一段采样完成中断标志

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

12.3 应用指南

12.3.1 ADC 采样触发模式

ADC 支持一段、两段、四段采样模式，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。ADC 内部的状态转移描述如下，共有 8 个状态分别为采样状态 0~3，空闲状态 0~3。

第一次触发

来自 MCPWM/UTimer 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 ADC_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 ADC_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 ADC_CHNT[3:0]，ADC 进入空闲状态 1（两段或四段采样第一段完成，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]，判断是否为两段采样。

是:结束本次采样，回到空闲状态 0。

否:进入空闲状态 2，等待第三次触发及第四次触发完成采样。

第三段触发

第三段采样

第三段采样次数到达预设值 ADC_CHNT[11:8]，进入空闲状态 3。

第四段触发

第四段采样

第四段采样通道数到达预设值 ADC_CHNT[15:12]，回到空闲状态 0。

各种硬件触发模式的触发条件汇总如表 12-27 ADC 采样触发模式所示。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、四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 12-27 ADC 采样触发模式

	单段触发	两段触发	四段触发
TADC 触发	None(TADC 触发使能未打开)	第一段 TADC[0] 第二段 TADC[1]	第一段 TADC[0] 第二段 TADC[1] 第三段 TADC[2] 第四段 TADC[3]
	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	第一段向 ADC_SWT 写入 16‘h5aa5 第二段向 ADC_SWT 写入 16‘h5aa5 第三段向 ADC_SWT 写入 16‘h5aa5 第四段向 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 ADC_CHNT 进行配置，寄存器数值为 1~10 时，对应的采样次数为 1~10。

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/UTimer 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 TADC[2]发生 4 次才进行触发，状态转移如图 12-3 所示。

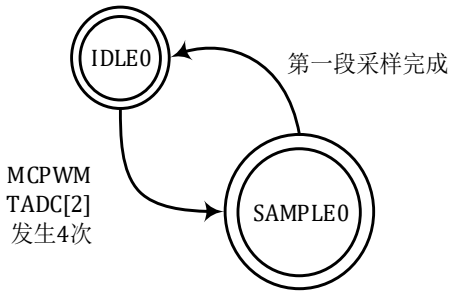


图 12-3 ADC 单段采样状态转移图

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 ADC0_DAT、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 12-4 所示。

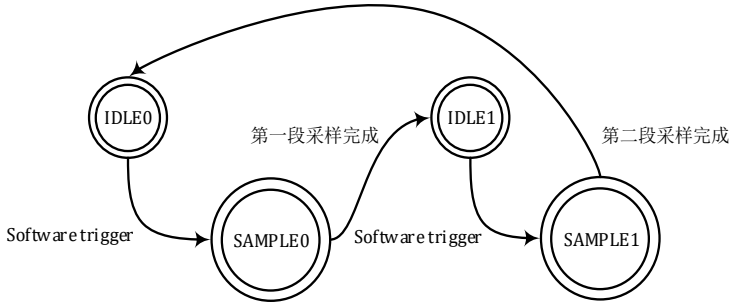


图 12-4 ADC 两段采样状态转移图

12.3.1.3 四段触发模式

与两段触发类似。四段的触发源分别为 TADC[0]、TADC[1]、TADC[2]、TADC[3]，且必须为 MCPWM TADC[0]/TADC[1]/TADC[2]/TADC[3]顺序触发 ADC 的四段采样；或者也可以是 4 次软件触发采样。四段采样的采样次数分别为 ADC_CHNT[7:4]、ADC_CHNT[3:0]、ADC_CHNT[7:4]、ADC_CHNT[3:0]。以 MCPWM TADC[0]/TADC[1]/TADC[2]/TADC[3]触发四段采样为例的状态转移如图 12-5 所示。

假设四段采样配置通道数目分别为 2、3、1、4，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，第三段采样转换后的数据会填充到 ADC_DAT5，第四段采样转换后的数据会依次填充到 ADC_DAT6、ADC_DAT7、ADC_DAT8、ADC_DAT9。

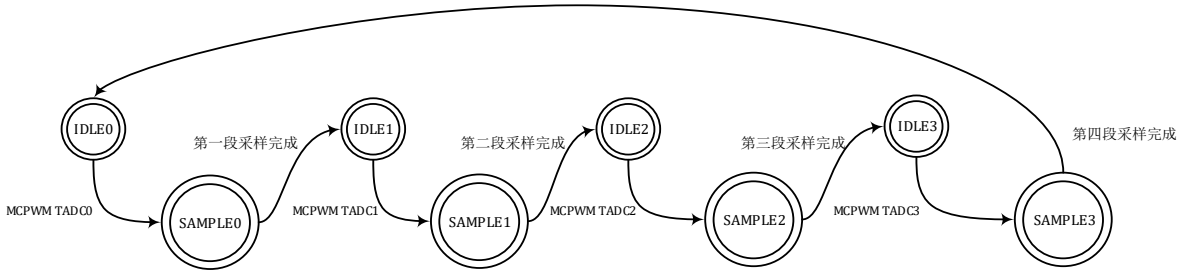


图 12-5 ADC 四段采样状态转移图

为使用 MCPWM 定时器产生 ADC 采样触发信号，需要配置 MCPWM_TMR0/ MCPWM_TMR1/ MCPWM_TMR2/ MCPWM_TMR3 等寄存器，对应 TADC0/1/2/3 发生时的 MCPWM 计数器值，此外需要配置 MCPWM_TH 设置计数器计数范围以及 MCPWM_TCLK 设置计数时钟频率并使能时钟。

12.3.2 中断

12.3.2.1 单段触发采样完成中断

采样完成产生中断 ADC_IF[0]。

12.3.2.2 两段触发采样完成中断

第一段采样完成产生中断 ADC_IF[0]，第二段采样完成产生中断 ADC_IF[1]。

12.3.2.3 四段触发采样完成中断

第一段采样完成产生中断 ADC_IF[0]，第二段采样完成产生中断 ADC_IF[1]，第三段采样完成产生中断 ADC_IF[2]，第四段采样完成产生中断 ADC_IF[3]。

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

12.3.4 选择对应的模拟通道

ADC 所采用信号对应的通道，请查阅 DATASHEET 中引脚功能选择。关闭对应 IO 的 IE 和 OE，即可使用其模拟功能。

13 UTimer 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 UTIMER 主要包括 2 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。

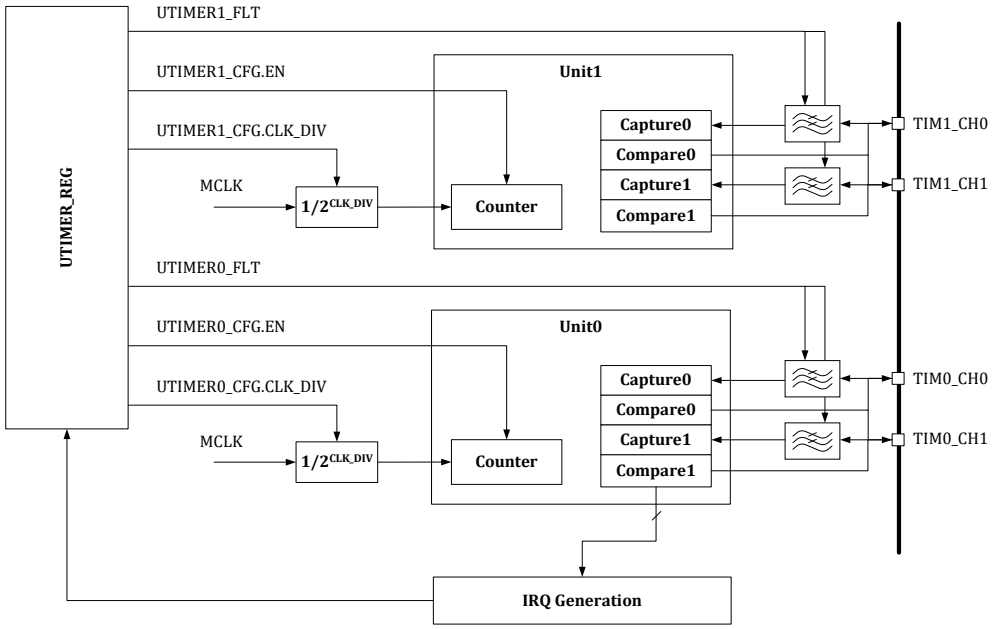


图 13-1 UTimer 模块顶层功能框图

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

utimer_unt 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。

utimer_unt 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，utimer_unt 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 为 16bit 通用定时器
- Timer1 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 48MHz 时钟频率下时，可以滤除 42uS 宽度以下毛刺

13.2 实现说明

13.2.1 时钟分频

Timer 使用 UTIMERx_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 UTIMERx_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，UTIMERx_CFG.CLK_DIV 对 Timer 计数时钟的分频对滤波时钟有影响。

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。

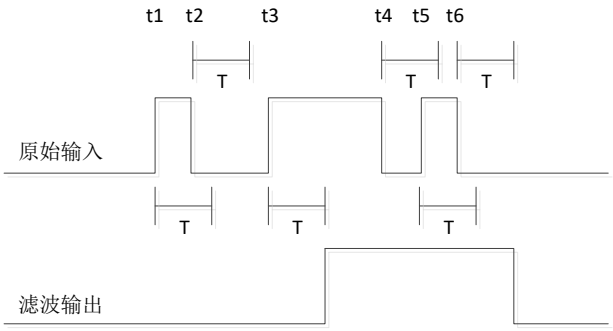


图 13-2 UTimer 滤波示意图

13.2.4 模式

13.2.4.1 计数器

Timer 中的计数器采用递增方向计数。

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。实际计数周期为 clk_freq*(TH+1)。

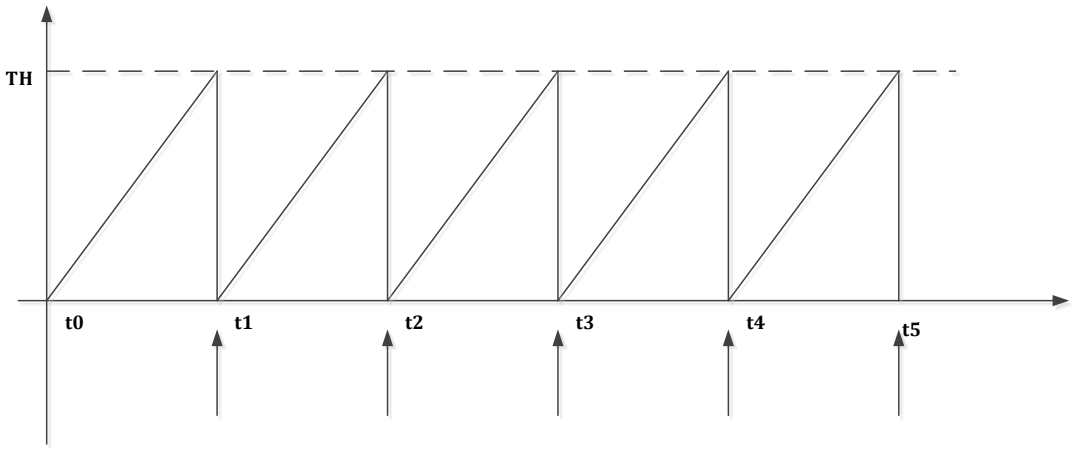


图 13-3 UTimer 通用计数器

计数器的计数时钟可以通过 UTIMERx_CFG.CLK_SRC 进行配置，可以使用芯片内部的系统时钟（通道为 48MHz PLL 时钟）或使用 Timer0/1 的通道 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 UTIMERx_CFG.CLK_DIV 进行分频，以降低计数器的计数频率。

13.2.4.2 比较模式

比较模式下，计数器计数到 UTIMERx_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 UTIMERx_CMP0=0，可使得 Timer X 通道 0 为恒 1，设置 UTIMERx_CMP0=UTIMERx_TH+1，可使得 Timer 通道 0 为恒 0。

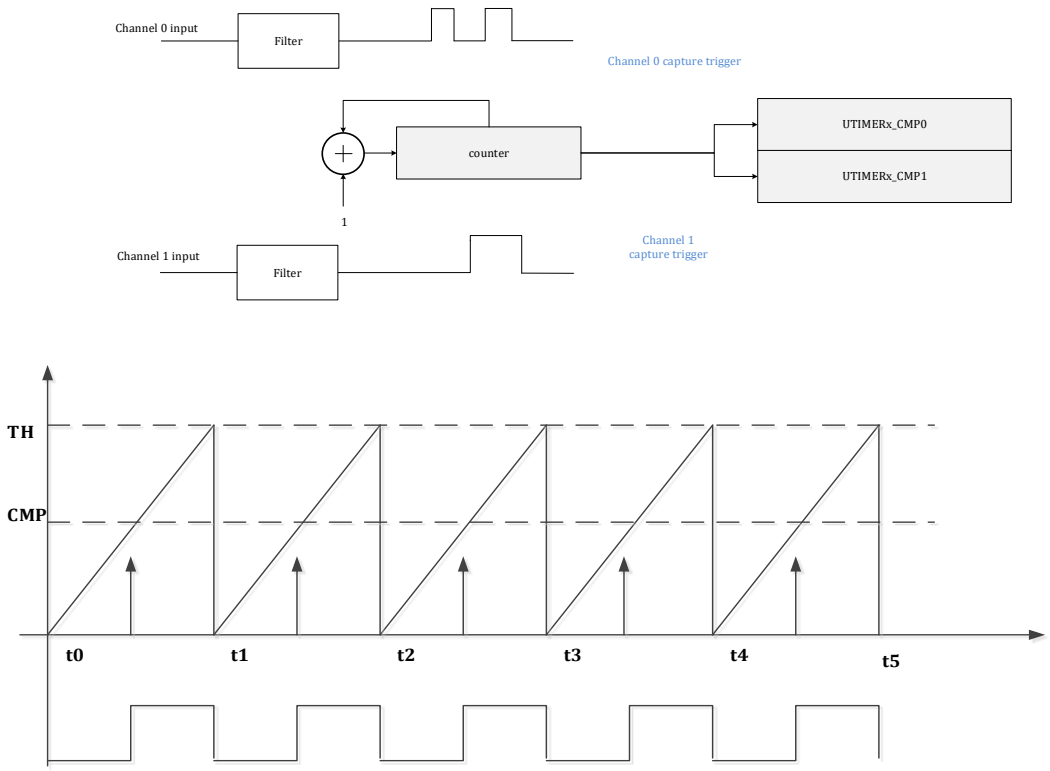
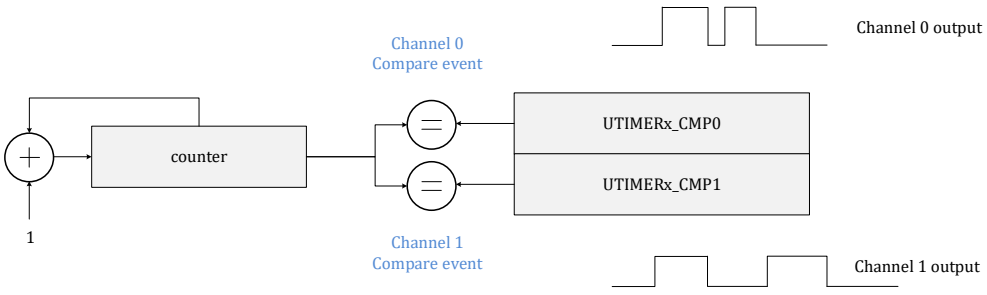


图 13-4 UTimer 比较模式

在比较模式下，当 UTIMERx_CFG.EN=0，即 Timer 没有开始计数时，通过向 UTIMERx_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。

13.2.4.3 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 UTIMER_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。



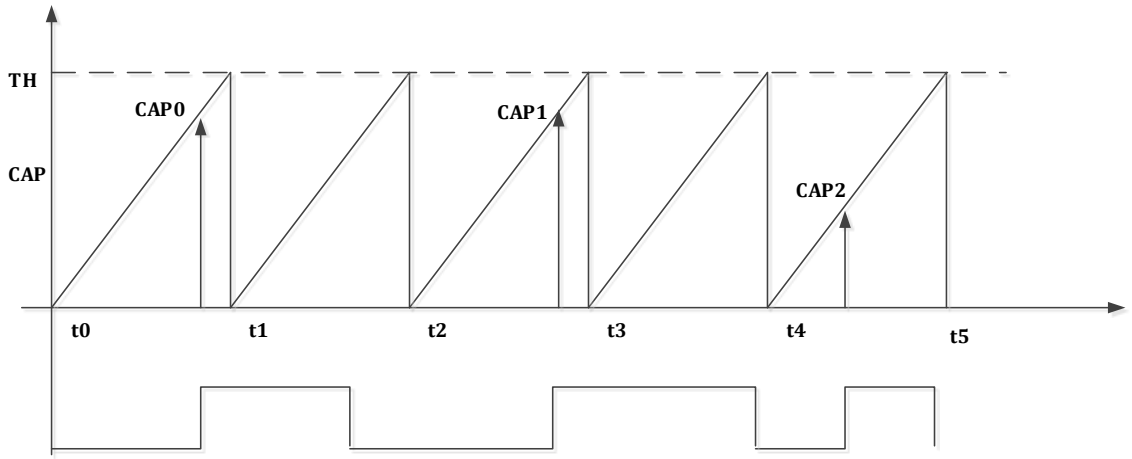


图 13-5 UTimer 捕获模式

如图 13-5 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 UTIMER_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 UTIMERx_CFG.SRC0 和 UTIMERx_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 UTIMERx_CFG.CAP0_CLR_EN 或 UTIMERx_CFG.CAP1_CLR_EN 使能 Timer 自动清零，即当通道 0/1 发生捕获事件时，UTIMER_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 UTIMER_CNT 自动回零。则当发生发生通道 0 捕获事件（CAP0）时，UTIMERx_CMP0 记录的值为信号上一个周波的周期，UTIMERx_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 UTIMER_CNT 自动回零。则当发生发生通道 1 捕获事件（CAP1）时，UTIMERx_CMP1 记录的值为信号上一个周波的周期，UTIMERx_CMP0 记录的值为信号上一个周波的低电平占空比。

13.2.4.4 单次触发

在比较模式下，当 UTIMERx_CFG.EN=0，即 Timer 没有开始计数时，通过向 UTIMERx_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。

当 UTIMERx_CFG.EN=1，通过向 UTIMERx_CFG.ONE_TRIG 写入 1，可以将 UTIMERx_CNT 重置，触发 TIMER 重新开始发出一个周期特定占空比的波形。一个周期后，TIMER 继续进行正常计数。

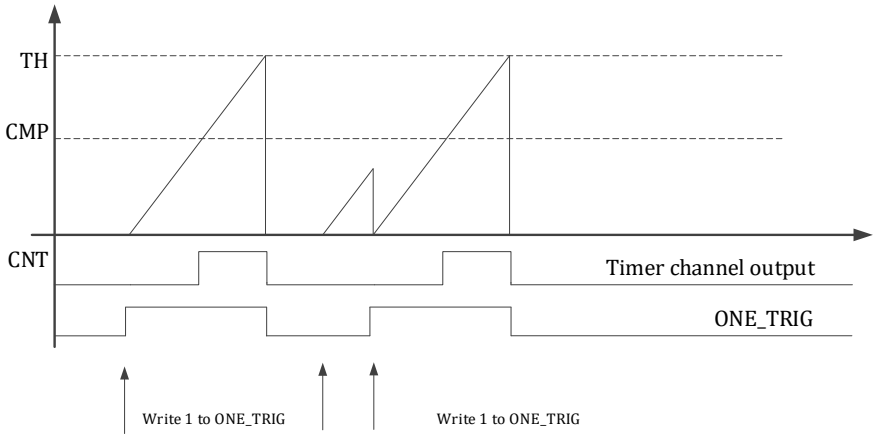


图 13-6 UTimer 单次触发模式

13.2.5 ADC 触发

Timer0/1 的比较事件（CMP0/1）可作为 ADC 采样触发事件。

13.3 寄存器

13.3.1 地址分配

Timer0 在芯片中的基地址是 0x4001_0500

表 13-1 Timer0 寄存器地址分配

名称	偏移	描述
UTIMER0_CFG	0x00	Timer0 配置寄存器
UTIMER0_TH	0x04	Timer0 计数门限寄存器
UTIMER0_CNT	0x08	Timer0 计数值寄存器
UTIMER0_CMP0	0x0C	Timer0 比较/捕获寄存器 0
UTIMER0_CMP1	0x10	Timer0 比较/捕获寄存器 1
UTIMER0_EVT	0x14	Timer0 外部事件选择寄存器
UTIMER0_FLT	0x18	Timer0 滤波控制寄存器
UTIMER0_IE	0x1C	Timer0 中断使能寄存器
UTIMER0_IF	0x20	Timer0 中断标志寄存器

Timer1 在芯片中的基地址是 0x4001_0600

表 13-2 Timer1 寄存器地址分配

UTIMER1_CFG	0x00	Timer1 配置寄存器
UTIMER1_TH	0x04	Timer1 计数门限寄存器
UTIMER1_CNT	0x08	Timer1 计数值寄存器
UTIMER1_CMP0	0x0C	Timer1 比较/捕获寄存器 0

UTIMER1_CMP1	0x10	Timer1 比较/捕获寄存器 1
UTIMER1_EVT	0x14	Timer1 外部事件选择寄存器
UTIMER1_FLT	0x18	Timer1 滤波控制寄存器
UTIMER1_IE	0x1C	Timer1 中断使能寄存器
UTIMER1_IF	0x20	Timer1 中断标志寄存器

Timer0/1 不同之处在于 Timer0 计数器相关寄存器为 16 位宽，而 Timer1 计数器相关寄存器为 32 位宽。

13.3.2 UTimer0 寄存器

13.3.2.1 UTIMER0_CFG Timer0 配置寄存器

地址:0x4001_0500

复位值:0x0

表 13-3 Timer0 配置寄存器 UTIMER0_CFG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN				CMP1_CLR_EN	CMP0_CLR_EN	ONE_TRIG	ETON		CLK_DIV			CLK_SRC			
RW				RW	RW	RW	RW		RW			RW			
0				0	0	0	0		0			0			

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

位置	位名称	说明
[31]	EN	Timer 模块整体使能，高有效
[30:28]		未使用
[27]	CAP1_CLR_EN	当发生 CAP1 捕获事件时，清零 Timer 计数器，高有效
[26]	CAP0_CLR_EN	当发生 CAP0 捕获事件时，清零 Timer 计数器，高有效
[25]	ONE_TRIG	在比较模式下，且 EN 为 0 时，写 1 触发 Timer 发送一个周期的特定占空比的脉冲，此位在脉冲发送期间内读为 1，一个 Timer 周期后，自动清零。
[24]	ETON	Timer 计数器计数使能配置。默认为 0。 0: 自动运行

		1: 等待外部事件触发计数，计数一个周期后停止。外部事件通过 UTIMER0_EVT 进行设置。
[23]		未使用
[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
[19:16]	CLK_SRC	Timer 时钟源 0: 芯片内部时钟 1: 保留 2: Timer0 通道 0 外部时钟信号 3: Timer0 通道 1 外部时钟信号 4: Timer1 通道 0 外部时钟信号 5: Timer1 通道 1 外部时钟信号
[15:12]	SRC1	Timer 捕获模式通道 1 信号来源。默认为 4'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: 比较器 0 的输出 3: 比较器 1 的输出 8: 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，当计数器计数值回零时的输出值。
[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: 比较器 0 的输出 3: 比较器 1 的输出

		8: 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:当计数器计数值回零时的输出值。
[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 UTIMERx_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 UTIMERx_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。

SRC0 和 SRC1 可以设置采样同一个 TIMER 通道 IO，分别捕获上升沿和下降沿，并在上升沿或下降沿时清零 TIMER_CNT,这种方式便于计算方波的周期和占空比。此种方式会占用 TIMER 两个通道。但只占用一个 TIMER 通道 IO，另一个 TIMER 通道 IO 可以设置为其他功能使用。

13.3.2.2 UTIMER0_TH Timer0 门限寄存器

地址:0x4001_0504

复位值:0x0

表 13-4 Timer0 门限寄存器 UTIMER0_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。计数器从 0 计数到 TH 值后再次回 0 开始计数

13.3.2.3 UTIMER0_CNT Timer0 计数寄存器

地址:0x4001_0508

复位值:0x0

表 13-5 Timer0 计数寄存器 UTIMER0_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 UTIMER0_CNT 前需要先通过 SYS_CLK_FEN.TIMER0_CLK_EN 开启 Timer0 时钟。

13.3.2.4 UTIMER0_CMP0 Timer0 通道 0 比较捕获寄存器

地址:0x4001_050C

复位值:0x0

表 13-6 Timer0 通道 0 比较捕获寄存器 UTIMER0_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 UTIMER0_CMP1 Timer0 通道 1 比较捕获寄存器

地址:0x4001_0510

复位值:0x0

表 13-7 Timer0 通道 1 比较捕获寄存器 UTIMER0_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 UTIMER0_EVT Timer0 外部事件选择寄存器

地址:0x4001_0514

复位值:0x0

表 13-8 Timer0 外部事件选择寄存器 UTIMER0_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
[31:5]		未使用
[4:0]	EVT_SRC	Timer 外部事件选择寄存器,本寄存器需要配合 UTIMER0_CFG.ETON 使用。ETON 为高时，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。 0:TIMER0 通道 0 比较事件 1:TIMER0 通道 1 比较事件 2:TIMER1 通道 0 比较事件 3:TIMER1 通道 1 比较事件 10:MCPWM TADC[0]比较事件 11:MCPWM TADC[1]比较事件 12:MCPWM TADC[2]比较事件 13:MCPWM TADC[3]比较事件

13.3.2.7 UTIMER0_FLT Timer0 滤波控制寄存器

地址:0x4001_0518

复位值:0x0

表 13-9 Timer0 滤波控制寄存器 UTIMER0_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 UTIMERx_CFG.CLK_DIV 的分频系数控制。

假设 UTIMERx_FLT = 0x6；UTIMERx_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 UTIMER0_IE Timer0 中断使能寄存器

地址:0x4001_051C

复位值:0x0

表 13-10 Timer0 中断使能寄存器 UTIMER0_IE

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

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。

13.3.2.9 UTIMER0_IF Timer0 中断标志寄存器

地址:0x4001_0520

复位值:0x0

表 13-11 Timer0 中断标志寄存器 UTIMER0_IF

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

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

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 UTIMER_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。

```
UTIMER_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.

```
UTIMER_IF=0x4;
```

13.3.3 UTimer1 寄存器

13.3.3.1 UTIMER1_CFG Timer1 配置寄存器

地址:0x4001_0600

复位值:0x0

表 13-12 Timer1 配置寄存器 UTIMER1_CFG

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

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

位置	位名称	说明
[31]	EN	Timer 模块整体使能，高有效
[30:28]		未使用
[27]	CAP1_CLR_EN	当发生 CAP1 捕获事件时，清零 Timer 计数器，高有效
[26]	CAP0_CLR_EN	当发生 CAP0 捕获事件时，清零 Timer 计数器，高有效
[25]	ONE_TRIG	在比较模式下，且 EN 为 0 时，写 1 触发 Timer 发送一个周期的特定占空比的脉冲，此位在脉冲发送期间内读为 1，一个 Timer 周期后，自动清零。
[24]	ETON	Timer 计数器计数使能配置。默认为 0。 0: 自动运行 1: 等待外部事件触发计数，计数一个周期后停止。外部事件通过 UTIMER0_EVT 进行设置。
[23]		未使用
[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
[19:16]	CLK_SRC	Timer 时钟源 0: 芯片内部时钟 1: 保留 2: Timer0 通道 0 外部时钟信号

		3: Timer0 通道 1 外部时钟信号 4: Timer1 通道 0 外部时钟信号 5: Timer1 通道 1 外部时钟信号
[15:12]	SRC1	Timer 捕获模式通道 1 信号来源。默认为 4'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: 比较器 0 的输出 3: 比较器 1 的输出 8: 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，当计数器计数值回零时的输出值。
[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: 比较器 0 的输出 3: 比较器 1 的输出 8: 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:当计数器计数值回零时的输出值。
[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。

13.3.3.2 UTIMER1_TH Timer1 门限寄存器

地址:0x4001_0604

复位值:0x0

表 13-13 Timer1 门限寄存器 UTIMER1_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。计数器从 0 计数到 TH 值后再次回 0 开始计数

13.3.3.3 UTIMER1_CNT Timer1 计数寄存器

地址:0x4001_0608

复位值:0x0

表 13-14 Timer1 计数寄存器 UTIMER1_CNT

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

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

注意:写入 UTIMER1_CNT 前需要先通过 SYS_CLK_FEN.TIMER1_CLK_EN 开启 Timer1 时钟。

位置	位名称	说明
[31:0]	CNT	Timer1 计数器当前计数值。写操作可以写入新的计数值。

13.3.3.4 UTIMER1_CMP0 Timer1 通道 0 比较捕获寄存器

地址:0x4001_060C

复位值:0x0

表 13-15 Timer1 通道 0 比较捕获寄存器 UTIMER1_CMP0

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

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

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。

13.3.3.5 UTIMER1_CMP1 Timer1 通道 1 比较捕获寄存器

地址:0x4001_0610

复位值:0x0

表 13-16 Timer1 通道 1 比较捕获寄存器 UTIMER1_CMP1

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

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

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。

13.3.3.6 UTIMER1_EVT Timer1 外部事件选择寄存器

地址:0x4001_0614

复位值:0x0

表 13-17 Timer1 外部事件选择寄存器 UTIMER1_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
[31:3]		未使用
[2:0]	EVT_SRC	Timer 外部事件选择寄存器,本寄存器需要配合 UTIMER1_CFG.ETON 使用。ETON 为高时，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。 0:TIMER0 通道 0 比较事件 1:TIMER0 通道 1 比较事件 2:TIMER1 通道 0 比较事件 3:TIMER1 通道 1 比较事件 10:MCPWM TADC[0]比较事件 11:MCPWM TADC[1]比较事件 12:MCPWM TADC[2]比较事件 13:MCPWM TADC[3]比较事件

13.3.3.7 UTIMER1_FLT Timer1 滤波控制寄存器

地址:0x4001_0618

复位值:0x0

表 13-18 Timer1 滤波控制寄存器 UTIMER1_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 UTIMERx_CFG.CLK_DIV 的分频系数控制。

假设 UTIMERx_FLT.FLT = 0x6；UTIMERx_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 UTIMER1_IE Timer1 中断使能寄存器

地址:0x4001_061C

复位值:0x0

表 13-19 Timer1 中断使能寄存器 UTIMER1_IE

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

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 UTIMER1_IF Timer1 中断标志寄存器

地址:0x4001_0620

复位值:0x0

表 13-20 Timer1 中断标志寄存器 UTIMER1_IF

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

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

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 48MHz/24MHz/12MHz/6MHz。通道 0/1/2 固定使用时基 0，通道 3 固定使用时基 1。

MCPWM 模块共包含四个 PWM 生成子模块。

- 可以产生 4 对（互补信号）或 8 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），把所有 MCPWM 输出信号复位到软件指定状态，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）。

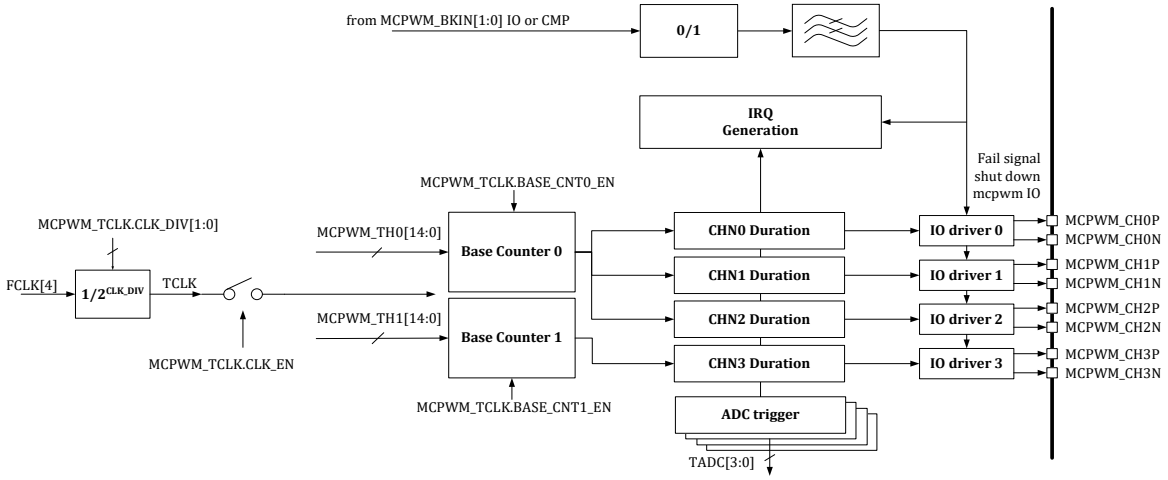


图 14-1 MCPWM 模块框图

为了保证定时精度，通常采用 48MHz 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 MCPWM_TH0/MCPWM_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，MCPWM_IFx.T0_IF 和 MCPWM_IFx.T1_IF 将被置位。

可通过寄存器配置 MCPWM_TCLK.BASE_CNT0_EN/MCPWM_TCLK.BASE_CNT1_EN 控制 Base Counter 0/1 的启动和停止。

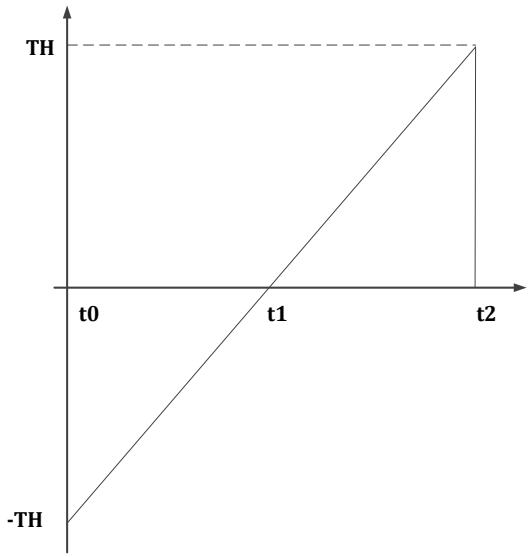


图 14-2 Base Counter t0/t1 时序

在运行 MCPWM 模块前，用户一般需要将门限值 MCPWM_TH00~MCWM_TH31，死区寄存器 MCPWM_DTH0/ MCPWM_DTH1 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 MCPWM_UPDATE 寄存器实现手动更新，也可以通过配置 MCPWM_SDCFG.T1_UPDATE_EN 及 MCPWM_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 MCPWM_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。

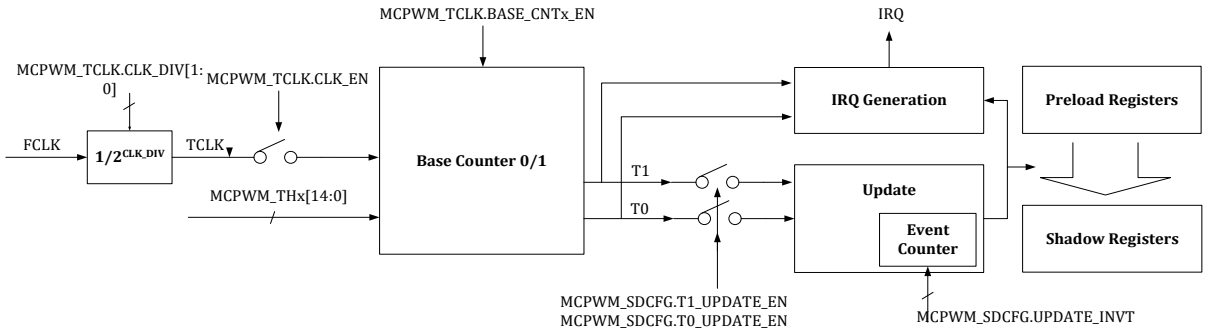


图 14-3 MCPWM 更新机制

时基 0 和时基 1 对应的 TH 寄存器 MCPWM_THx 和 CNT 寄存器 MCPWM_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 MCPWM_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 MCPWM_TCLK.TCLK_EN=0）的情况下进行更新。MCPWM_TCLK.BASE_CNT0_EN 和 MCPWM_TCLK.BASE_CNT1_EN 分别控制时基 0 和时基 1 的 CNT 计数使能。当 MCPWM_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 MCPWM_THx 和 MCPWM_CNTx 同时更新。当然也支持 MCPWM_THx 和 MCPWM_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 IO 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]。

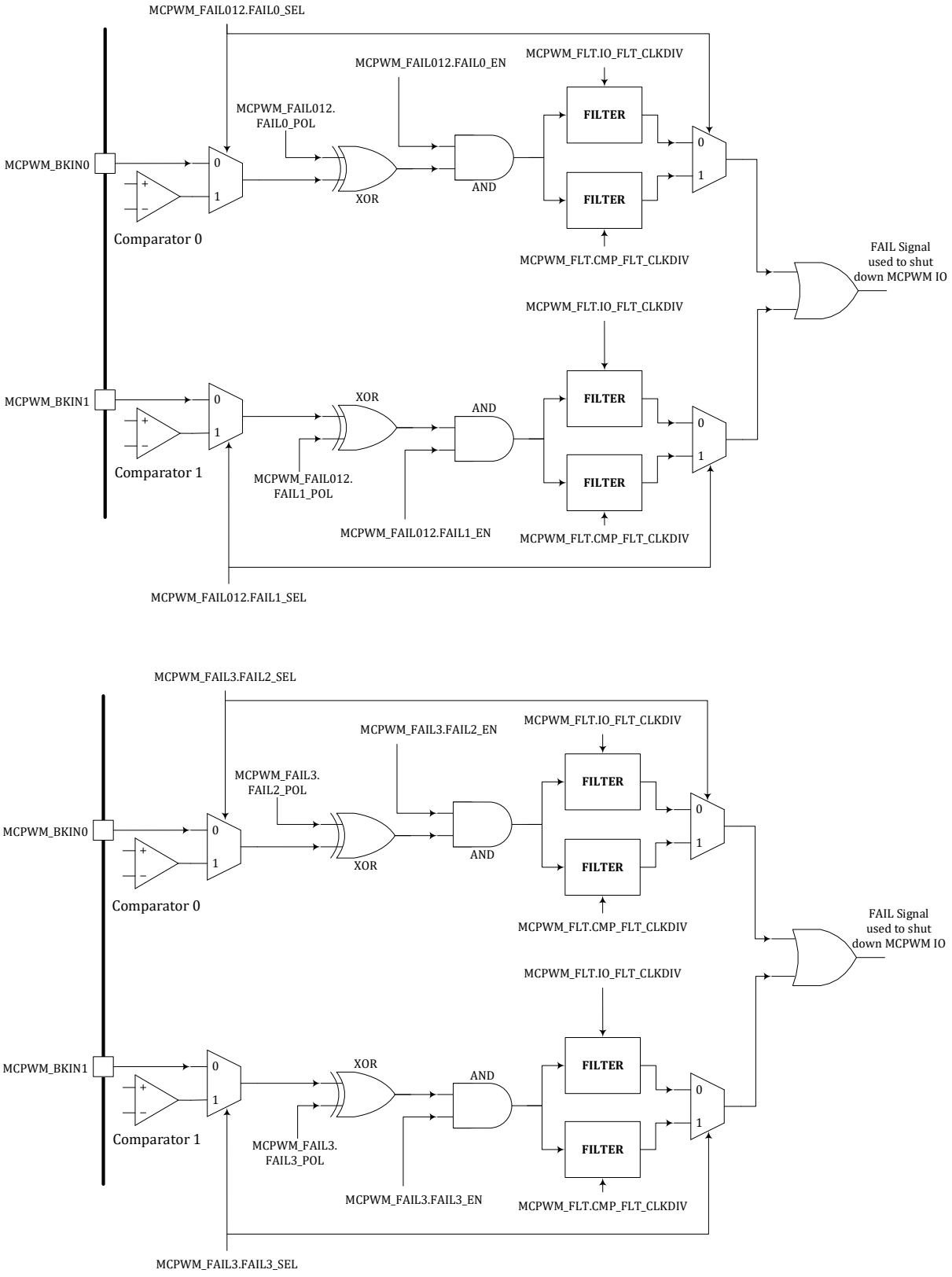


图 14-4 MCPWM FAIL 逻辑示意图

Filter 滤波模块的时钟，来自系统主时钟 MCLK 的门控时钟 FCLK[4]，见 SYS_CLK_FEN，并经过可选的两级分频，第一级分频由 MCPWM_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，如果 Fail 信号来自 MCPWM_BKIN[1:0]则使用 MCPWM_FLT.IO_FLT_CLKDIV[7:0]作为第二级的分频系数；如果 Fail 信号来自芯片内部比较器输出，则使用 MCPWM_FLT.CMP_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，48MHz 对应 20.8ns。FLT_CLKDIV 根据配置情况可能是 MCPWM_FLT.IO_FLT_CLKDIV 或 MCPWM_FLT.CMP_FLT_CLKDIV。

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

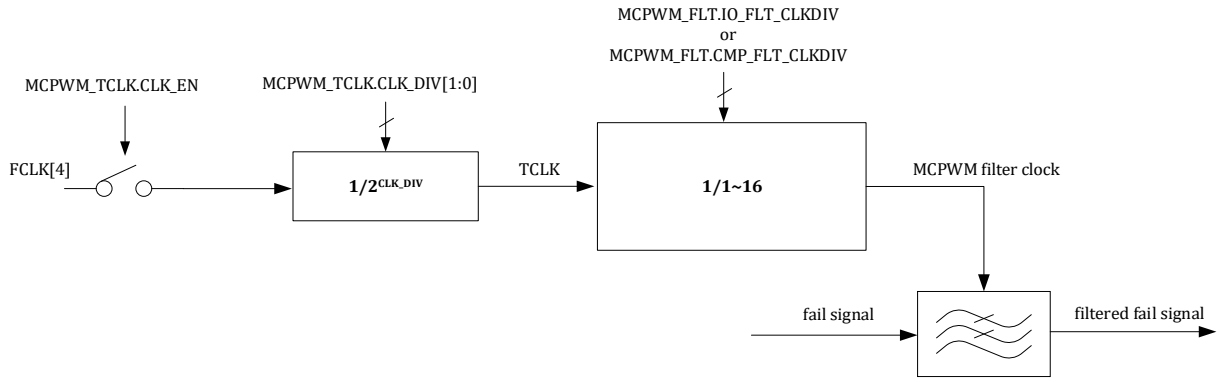


图 14-5 MCPWM Fail 信号滤波时钟生成逻辑

一旦发生 FAIL 事件，硬件将 IO 输出强制变为 MCPWM_FAILx.CHxN_DEFAULT 和 MCPWM_FAILx.CHxP_DEFAULT 寄存器所指定的故障缺省值，此时 MCPWM_FAILx.CHxN_DEFAULT 和 MCPWM_FAILx.CHxP_DEFAULT 的值直接输出到 IO 口，不再受到 MCPWM_FAILx.FAIL_POL 等极性控制的影响。其中 MCPWM_FAIL012 用于控制通道 0/1/2 的故障缺省设置，MCPWM_FAIL3 用于控制通道 3 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:

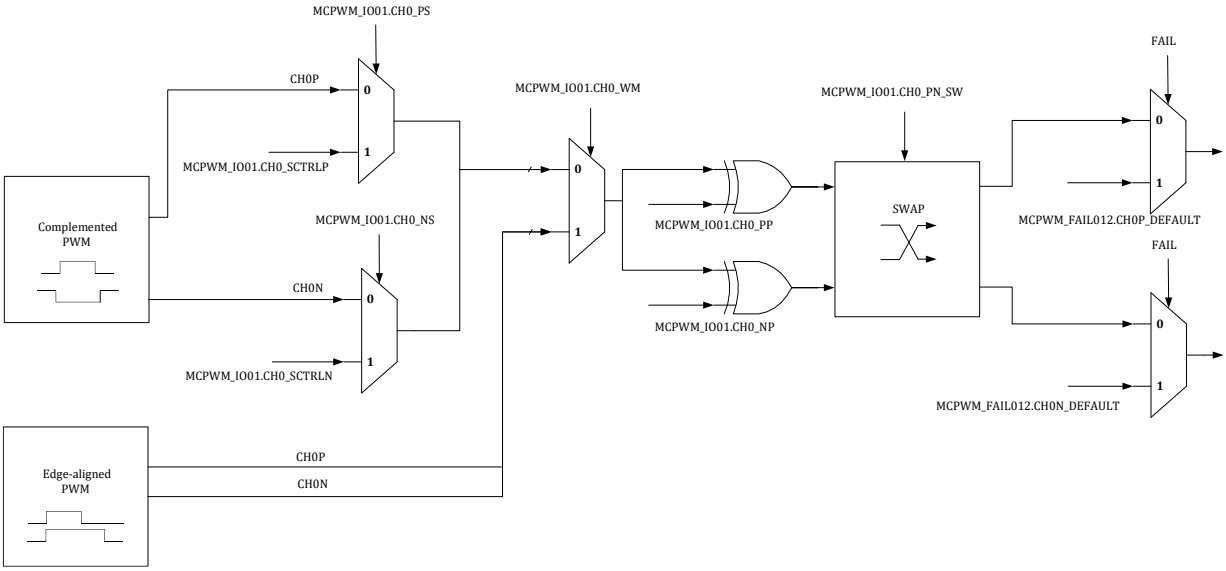


图 14-6 IO Driver 模块数据流程图

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。

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，经过死区延时 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。

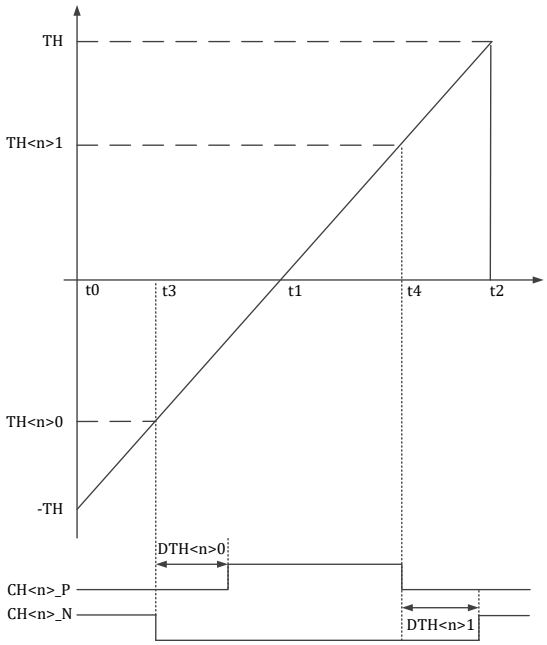


图 14-7MCPWM 时序 TH<n>0 和 TH<n>1-中心对齐模式

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。

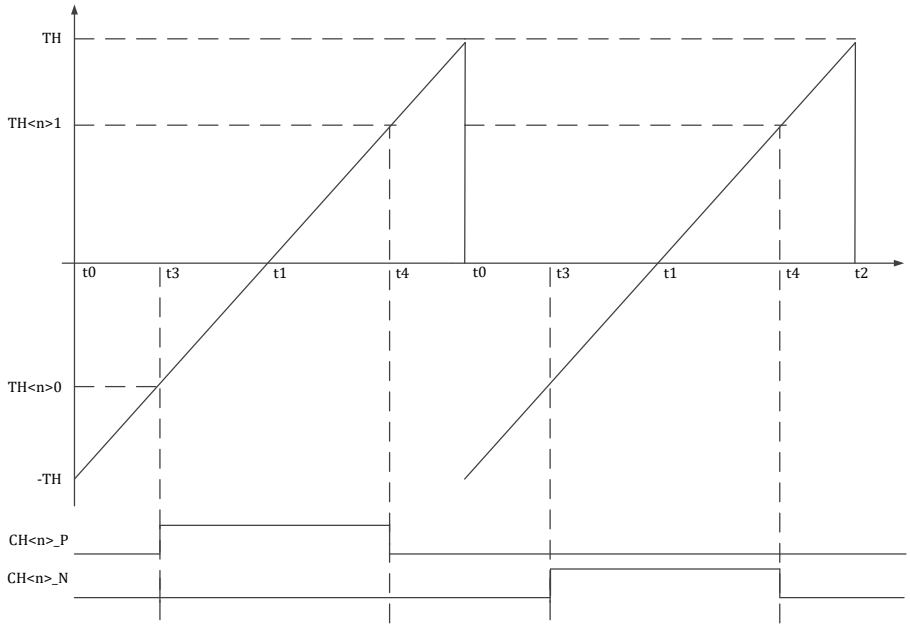


图 14-8 MCPWM 时序 TH<n>0 和 TH<n>1-中心对齐推挽模式

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。

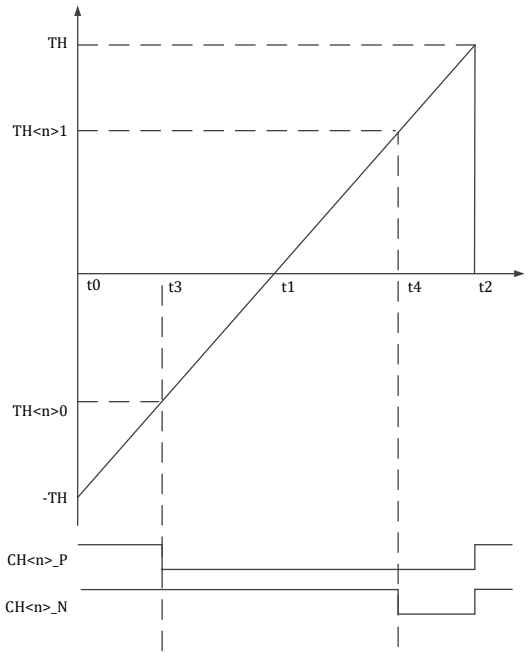


图 14-9MCPWM 时序边沿对齐模式

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。

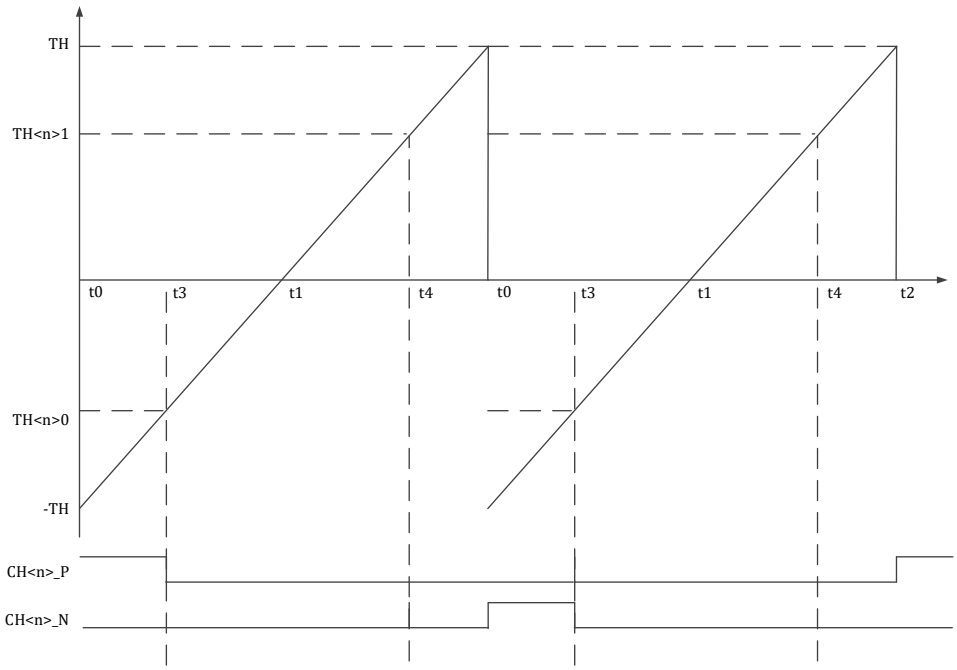


图 14-10MCPWM 时序 TH<n>0 和 TH<n>1 边沿对齐推挽模式

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 MCPWM_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。

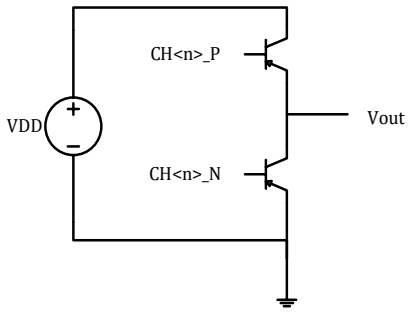


图 14-11MCPWM IO 控制示意图

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(MCPWM_FAILx.CHxN_DEFAULT 和 MCPWM_FAILx.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 MCPWM_FAILx.CHxN_DEFAULT 和 MCPWM_FAILx.CHxP_DEFAULT 指定值，当用户配置完毕，MCPWM 正常工作后，配置 MCPWM_FAILx.MCPWM_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 MCPWM_TMR0/ MCPWM_TMR1/ MCPWM_TMR2/ MCPWM_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 MCPWM_TMR0/ MCPWM_TMR1 固定使用时基 0,MCPWM_TMR2/ MCPWM_TMR3 可以选择使用时基 0/1,见 14.2.27 MCPWM_TCLK。

表 14-1 MCPWM 计数器阈值与事件对应表

t0	-TH
t1	0
TIO0[0]	TH00
TIO0[1]	TH01
TADC[0]	TMR0
TADC [1]	TMR1
TADC [2]	TMR2
TADC [3]	TMR3

14.1.6 中断

MCPMW_IF0 和 MCPWM_EIF[5:4]标志置位且使能的情况下，产生 MCPWM_IRQ0 中断；

MCPMW_IF1 和 MCPWM_EIF[7:6]标志置位且使能的情况下，产生 MCPWM_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_0700，

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
MCPWM_TH00	0x00	MCPWM CH0_P 比较门限值寄存器
MCPWM_TH01	0x04	MCPWM CH0_N 比较门限值寄存器
MCPWM_TH10	0x08	MCPWM CH1_P 比较门限值寄存器
MCPWM_TH11	0x0C	MCPWM CH1_N 比较门限值寄存器
MCPWM_TH20	0x10	MCPWM CH2_P 比较门限值寄存器
MCPWM_TH21	0x14	MCPWM CH2_N 比较门限值寄存器
MCPWM_TH30	0x18	MCPWM CH3_P 比较门限值寄存器
MCPWM_TH31	0x1C	MCPWM CH3_N 比较门限值寄存器
MCPWM_TMR0	0x20	ADC 采样定时器比较门限 0 寄存器
MCPWM_TMR1	0x24	ADC 采样定时器比较门限 1 寄存器
MCPWM_TMR2	0x28	ADC 采样定时器比较门限 2 寄存器
MCPWM_TMR3	0x2C	ADC 采样定时器比较门限 3 寄存器
MCPWM_TH0	0x30	MCPWM 时基 0 门限值寄存器
MCPWM_TH1	0x34	MCPWM 时基 1 门限值寄存器
MCPWM_CNT0	0x38	MCPWM 时基 0 计数器寄存器
MCPWM_CNT1	0x3C	MCPWM 时基 1 计数器寄存器
MCPWM_UPDATE	0x40	MCPWM 加载控制寄存器
MCPWM_FCNT	0x44	MCPWM FAIL 时刻 CNT 值
MCPWM_EVT0	0x48	MCPWM 时基 0 外部触发
MCPWM_EVT1	0x4C	MCPWM 时基 1 外部触发
MCPWM_DTH0	0x50	MCPWM CH0/1/2/3 N 通道死区宽度控制寄存器
MCPWM_DTH1	0x54	MCPWM CH0/1/2/3 P 通道死区宽度控制寄存器
MCPWM_FLT	0x70	MCPWM 滤波时钟分频寄存器
MCPWM_SDCFG	0x74	MCPWM 加载配置寄存器
MCPWM_AUEN	0x78	MCPWM 自动更新使能寄存器
MCPWM_TCLK	0x7C	MCPWM 时钟分频控制寄存器
MCPWM_IE0	0x80	MCPWM 时基 0 中断控制寄存器
MCPWM_IF0	0x84	MCPWM 时基 0 中断标志位寄存器

MCPWM_IE1	0x88	MCPWM 中断控制寄存器
MCPWM_IF1	0x8C	MCPWM 中断标志位寄存器
MCPWM_EIE	0x90	MCPWM 异常中断控制寄存器
MCPWM_EIF	0x94	MCPWM 异常中断标志位寄存器
MCPWM_RE	0x98	MCPWM DMA 请求控制寄存器
MCPWM_PP	0x9C	MCPWM 推挽模式使能寄存器
MCPWM_IO01	0xA0	MCPWM CH0 CH1 IO 控制寄存器
MCPWM_IO23	0xA4	MCPWM CH2 CH3 IO 控制寄存器
MCPWM_FAIL012	0xA8	MCPWM CH0 CH1 CH2 短路控制寄存器
MCPWM_FAIL3	0xAC	MCPWM CH3 短路控制寄存器
MCPWM_PRT	0xB0	MCPWM 保护寄存器
MCPWM_SWAP	0xB4	MCPWM 通道重映射寄存器
MCPWM_CHMSK	0xB8	MCPWM 通道屏蔽寄存器

表 14-3 受 MCPWM_PRT 保护的寄存器

名称	偏移地址	说明
MCPWM_TH0	0x30	MCPWM 门限值寄存器
MCPWM_TH1	0x34	MCPWM 门限值寄存器
MCPWM_DTH0	0x50	MCPWM CH0/1/2/3 N 通道死区宽度控制寄存器
MCPWM_DTH1	0x54	MCPWM CH0/1/2/3 P 通道死区宽度控制寄存器
MCPWM_FLT	0x70	MCPWM 滤波时钟分频寄存器
MCPWM_SDCFG	0x74	MCPWM 加载配置寄存器
MCPWM_AUEN	0x78	MCPWM 自动加载使能寄存器
MCPWM_TCLK	0x7C	MCPWM 时钟分频控制寄存器
MCPWM_IE0	0x80	MCPWM 时基 0 中断控制寄存器
MCPWM_IE1	0x88	MCPWM 时基 1 中断控制寄存器
MCPWM_EIE	0x90	MCPWM 异常中断控制寄存器
MCPWM_RE	0x98	MCPWMDMA 请求控制寄存器
MCPWM_PP	0x9C	MCPWM 推挽模式使能寄存器
MCPWM_IO01	0xA0	MCPWM CH0 CH1 IO 控制寄存器
MCPWM_IO23	0xA4	MCPWM CH2 CH3 IO 控制寄存器
MCPWM_FAIL012	0xA8	MCPWM CH0 CH1 CH2 短路控制寄存器
MCPWM_FAIL3	0xAC	MCPWM CH3 短路控制寄存器
MCPWM_CHMSK	0xB8	MCPWM 通道屏蔽寄存器

表 14-4 存在影子寄存器的寄存器

名称	偏移地址	说明
MCPWM_TH00	0x00	MCPWM CH0_P 比较门限值寄存器
MCPWM_TH01	0x04	MCPWM CH0_N 比较门限值寄存器
MCPWM_TH10	0x08	MCPWM CH1_P 比较门限值寄存器
MCPWM_TH11	0x0C	MCPWM CH1_N 比较门限值寄存器
MCPWM_TH20	0x10	MCPWM CH2_P 比较门限值寄存器

MCPWM_TH21	0x14	MCPWM CH2_N 比较门限值寄存器
MCPWM_TH30	0x18	MCPWM CH3_P 比较门限值寄存器
MCPWM_TH31	0x1C	MCPWM CH3_N 比较门限值寄存器
MCPWM_TMR0	0x20	ADC 采样定时器比较门限 0 寄存器
MCPWM_TMR1	0x24	ADC 采样定时器比较门限 1 寄存器
MCPWM_TMR2	0x28	ADC 采样定时器比较门限 2 寄存器
MCPWM_TMR3	0x2C	ADC 采样定时器比较门限 3 寄存器
MCPWM_TH0	0x30	MCPWM 时基 0 门限值寄存器
MCPWM_TH1	0x34	MCPWM 时基 1 门限值寄存器
MCPWM_CNT0	0x38	MCPWM 时基 0 计数器寄存器
MCPWM_CNT1	0x3C	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 MCPWM_TH00

无写保护的寄存器

地址:0x4001_0700

复位值:0x0

表 14-5 MCPWM_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。

14.2.3 MCPWM_TH01

无写保护的寄存器

地址:0x4001_0704

复位值:0x0

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

RW
0

表 14-6 MCPWM_TH00 配置寄存器

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 MCPWM_TH10

无写保护的寄存器

地址:0x4001_0708

复位值:0x0

表 14-7 MCPWM_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 MCPWM_TH11

无写保护的寄存器

地址:0x4001_070C

复位值:0x0

表 14-8 MCPWM_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 MCPWM_TH20

无写保护的寄存器

地址:0x4001_0710

复位值:0x0

表 14-9 MCPWM_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 MCPWM_TH21

无写保护的寄存器

地址:0x4001_0714

复位值:0x0

表 14-10 MCPWM_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 MCPWM_TH30

无写保护的寄存器

地址:0x4001_0718

复位值:0x0

表 14-11 MCPWM_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 MCPWM_TH31

无写保护的寄存器

地址:0x4001_071C

复位值:0x0

表 14-12 MCPWM_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 MCPWM_TMR0

无写保护的寄存器

地址:0x4001_0720

复位值:0x0

表 14-13 MCPWM_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 MCPWM_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。

14.2.11 MCPWM_TMR1

无写保护的寄存器

地址:0x4001_0724

复位值:0x0

表 14-14 MCPWM_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 MCPWM_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。

14.2.12 MCPWM_TMR2

无写保护的寄存器

地址:0x4001_0728

复位值:0x0

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

RW
0x7FFF

表 14-15 MCPWM_TMR2 配置寄存器

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。

14.2.13 MCPWM_TMR3

无写保护的寄存器

地址:0x4001_072C

复位值:0x0

表 14-16 MCPWM_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。。

14.2.14 MCPWM_TH0

写保护的寄存器

地址:0x4001_0730

复位值:0x0

表 14-17 MCPWM_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。

14.2.15 MCPWM_TH1

写保护的寄存器

地址:0x4001_0734

复位值:0x0

表 14-18 MCPWM_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。。

14.2.16 MCPWM_CNT0

无写保护的寄存器

地址:0x4001_0738

复位值:0x0

表 14-19 MCPWM_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

14.2.17 MCPWM_CNT1

无写保护的寄存器

地址:0x4001_073C

复位值:0x0

表 14-20 MCPWM_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

14.2.18 MCPWM_UPDATE

无写保护的寄存器

地址:0x4001_0740

复位值:0x0

表 14-21 MCPWM_UPDATE MCPWM 手动更新寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT1_UPDATE	CNT0_UPDATE	TH1_UPDATE	TH0_UPDATE	TMR3_UPDATE	TMR2_UPDATE	TMR1_UPDATE	TMR0_UPDATE	TH31_UPDATE	TH30_UPDATE	TH21_UPDATE	TH20_UPDATE	TH11_UPDATE	TH10_UPDATE	TH01_UPDATE	TH00_UPDATE
WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:16]		未使用
[15]	CNT1_UPDATE	写 1 产生软件（手动）触发，将 MCPWM_CNT1 由预装载寄存器加载到 MCPWM 运行所使用的影子寄存器

[14]	CNT0_UPDATE	写 1 产生软件（手动）触发，将 MCPWM_CNT0 由预装载寄存器加载到 MCPWM 运行所使用的影子寄存器
[13]	TH1_UPDATE	写 1 产生软件（手动）触发，将 MCPWM_TH1 由预装载寄存器加载到 MCPWM 运行所使用的影子寄存器
[12]	TH0_UPDATE	写 1 产生软件（手动）触发，将 MCPWM_TH0 由预装载寄存器加载到 MCPWM 运行所使用的影子寄存器
[11]	TMR3_UPDATE	写 1 产生软件（手动）触发，将 MCPWM_TMR3 由预装载寄存器加载到 MCPWM 运行所使用的影子寄存器
[10]	TMR2_UPDATE	写 1 产生软件（手动）触发，将 MCPWM_TMR2 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 MCPWM_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 MCPWM_TMR0 由预装载寄存器加载到 MCPWM 运行所使用的影子寄存器
[7]	TH31_UPDATE	写 1 产生软件（手动）触发，将 MCPWM_TH31 由预装载寄存器加载到 MCPWM 运行所使用的影子寄存器
[6]	TH30_UPDATE	写 1 产生软件（手动）触发，将 MCPWM_TH30 由预装载寄存器加载到 MCPWM 运行所使用的影子寄存器
[5]	TH21_UPDATE	写 1 产生软件（手动）触发，将 MCPWM_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 MCPWM_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 MCPWM_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 MCPWM_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 MCPWM_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 MCPWM_TH00 由预装载寄存器加载到 MCPWM 运行所使用的影子寄存器

向 MCPWM_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，MCPWM_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。向 MCPWM_UPDATE[15:14]会导致 MCPWM_CNT0/1 被更新为预装载值，请留意是否需要更新 CNT。

向 MCPWM_UPDATE 写 0 不作用，写 1 后立即写 0 可能导致影子寄存器装载失败，不推荐写 0。

14.2.19 MCPWM_FCNT

无写保护的寄存器

地址:0x4001_0744

复位值:0x0

表 14-22 MCPWM_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 MCPWM_FAIL012[15]=1，当发生 fail0/1 事件时，记录 MCPWM_CNT0 值，存入 MCPWM_FCNT；若 MCPWM_FAIL3[15]=1，当发生 fail2/3 事件时，记录 MCPWM_CNT0 值，存入 MCPWM_FCNT

14.2.20 MCPWM_EVT0

无写保护的寄存器

地址:0x4001_0748

复位值:0x0

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

位置	位名称	说明
[31:12]		未使用
[11]	TIMER1_CMP1	TIMER1 CMP1 事件触发时基 0 开始计数
[10]	TIMER1_CMP0	TIMER1 CMP0 事件触发时基 0 开始计数
[9]	TIMER0_CMP1	TIMER0 CMP1 事件触发时基 0 开始计数
[8]	TIMER0_CMP0	TIMER0 CMP0 事件触发时基 0 开始计数
[7:4]		未使用
[3]	MCPWM0_TMR3	MCPWM0 TMR3 事件触发时基 0 开始计数
[2]	MCPWM0_TMR2	MCPWM0 TMR2 事件触发时基 0 开始计数
[1]	MCPWM0_TMR1	MCPWM0 TMR1 事件触发时基 0 开始计数
[0]	MCPWM0_TMR0	MCPWM0 TMR0 事件触发时基 0 开始计数

14.2.21 MCPWM_EVT1

无写保护的寄存器

地址:0x4001_074C，0x4001_384C

复位值:0x0

表 14-24 MCPWM_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
				TIMER1_CMP1	TIMER1_CMP0	TIMER0_CMP1	TIMER0_CMP0					MCPWM0_TMR3	MCPWM0_TMR2	MCPWM0_TMR1	MCPWM0_TMR0
				RW	RW	RW	RW					RW	RW	RW	RW
				0	0	0	0					0	0	0	0

位置	位名称	说明
[31:12]		未使用
[11]	TIMER1_CMP1	TIMER1 CMP1 事件触发时基 1 开始计数
[10]	TIMER1_CMP0	TIMER1 CMP0 事件触发时基 1 开始计数
[9]	TIMER0_CMP1	TIMER0 CMP1 事件触发时基 1 开始计数
[8]	TIMER0_CMP0	TIMER0 CMP0 事件触发时基 1 开始计数
[7:4]		未使用
[3]	MCPWM0_TMR3	MCPWM0 TMR3 事件触发时基 1 开始计数
[2]	MCPWM0_TMR2	MCPWM0 TMR2 事件触发时基 1 开始计数
[1]	MCPWM0_TMR1	MCPWM0 TMR1 事件触发时基 1 开始计数
[0]	MCPWM0_TMR0	MCPWM0 TMR0 事件触发时基 1 开始计数

14.2.22　MCPWM_DTH0

写保护的寄存器

地址:0x4001_0750

复位值:0x0

表 14-25 MCPWM_DTH0 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	DTH0	MCPWM CH0/1/2/3 P 通道死区宽度控制寄存器，10bit 无符号数

14.2.23 MCPWM_DTH1

写保护的寄存器

地址:0x4001_0754

复位值:0x0

表 14-26 MCPWM_DTH1 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	DTH1	MCPWM CH0/1/2/3 N 通道死区宽度控制寄存器，10bit 无符号数

14.2.24 MCPWM_FLT

写保护的寄存器

地址:0x4001_0770

复位值:0x0

表 14-27 MCPWM_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
CMP_FLT_CLKDIV								IO_FLT_CLKDIV							
RW								RW							
0								0							

位置	位名称	说明
[31:16]		未使用
[15:8]	CMP_FLT_CLKDIV	比较器输出的滤波时钟分频寄存器，基于系统时钟分频，影响 MCPWM_FAIL[1:0]。计算公式如下: 系统时钟 / (CMP_FLT_CLKDIV + 1)。分频范围是 1-256。
[7:0]	IO_FLT_CLKDIV	GPIO 输入的滤波时钟分频寄存器，基于系统时钟分频，影响 MCPWM_FAIL[1:0]。计算公式如下: 系统时钟 / (IO_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。

14.2.25 MCPWM_SDCFG

写保护的寄存器

地址:0x4001_0774

复位值:0x0

表 14-28 MCPWM_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 MCPWM_EIF[7:6] 并置位 MCPWM_FAIL3.MOE，恢复 MCPWM 通道 3 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 MCPWM_TH1，TH30，TH31 和 MCPWM_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 MCPWM_EIF[5:4] 并置位 MCPWM_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 相等，MCPWM 系统自动触发 MCPWM_TH0，MCPWM_TH00~TH21 和 MCPWM_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。

14.2.26 MCPWM_AUEN

写保护的寄存器

地址:0x4001_0778

复位值:0x0

表 14-29 MCPWM_AUEN MCPWM 自动更新使能寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT1_AUPDATE	CNT0_AUPDATE	TH1_AUPDATE	TH0_AUPDATE	TMR3_AUPDATE	TMR2_AUPDATE	TMR1_AUPDATE	TMR0_AUPDATE	TH31_AUPDATE	TH30_AUPDATE	TH21_AUPDATE	TH20_AUPDATE	TH11_AUPDATE	TH10_AUPDATE	TH01_AUPDATE	TH00_AUPDATE
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:16]		未使用
[15]	CNT1_AUPDATE	MCPWM_CNT1 自动加载使能。1:加载；0:不加载。
[14]	CNT0_AUPDATE	MCPWM_CNT0 自动加载使能。1:加载；0:不加载。
[13]	TH1_AUPDATE	MCPWM_TH1 自动加载使能。1:加载；0:不加载。
[12]	TH0_AUPDATE	MCPWM_TH0 自动加载使能。1:加载；0:不加载。
[11]	TMR3_AUPDATE	MCPWM_TMR3 自动加载使能。1:加载；0:不加载。
[10]	TMR2_AUPDATE	MCPWM_TMR2 自动加载使能。1:加载；0:不加载。
[9]	TMR1_AUPDATE	MCPWM_TMR1 自动加载使能。1:加载；0:不加载。
[8]	TMR0_AUPDATE	MCPWM_TMR0 自动加载使能。1:加载；0:不加载。
[7]	TH31_AUPDATE	MCPWM_TH31 自动加载使能。1:加载；0:不加载。
[6]	TH30_AUPDATE	MCPWM_TH30 自动加载使能。1:加载；0:不加载。
[5]	TH21_AUPDATE	MCPWM_TH21 自动加载使能。1:加载；0:不加载。
[4]	TH20_AUPDATE	MCPWM_TH20 自动加载使能。1:加载；0:不加载。
[3]	TH11_AUPDATE	MCPWM_TH11 自动加载使能。1:加载；0:不加载。
[2]	TH10_AUPDATE	MCPWM_TH10 自动加载使能。1:加载；0:不加载。
[1]	TH01_AUPDATE	MCPWM_TH01 自动加载使能。1:加载；0:不加载。
[0]	TH00_AUPDATE	MCPWM_TH00 自动加载使能。1:加载；0:不加载。

14.2.27 MCPWM_TCLK

写保护的寄存器

地址:0x4001_077C

复位值:0x0

表 14-30 MCPWM_TCLK 配置寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						EVT_CNT1_EN	EVT_CNT0_EN	BASE_CNT1_EN	BASE_CNT0_EN	TMR3_TB	TMR2_TB		CLK_EN	CLK_DIV	
						RW	RW	RW	RW	RW	RW		RW	RW	
						0	0	0	0	0	0		0	0	

位置	位名称	说明
[31:10]		未使用
[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]		未使用
[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 MCPWM_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 MCPWM_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 MCPWM_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。

14.2.28 MCPWM_IE0

写保护的寄存器

地址:0x4001_0780

复位值:0x0

表 14-31 MCPWM_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	TMR2_IE	TMR1_IE	TMR0_IE		TH21_IE	TH20_IE	TH11_IE	TH10_IE	TH01_IE	TH00_IE	T1_IE	T0_IE
	RW	RW	RW	RW	RW		RW	RW	RW	RW	RW	RW	RW	RW
	0	0	0	0	0		0	0	0	0	0	0	0	0

位置	位名称	说明
[31:15]		未使用
[14]	UP_IE	MCPWM_TH/MCPWM_TH00~MCPWM_TH31/MCPWM_TMR0~MCPWM_TMR3 等寄存器更新到 MCPWM 实际运行系统的中断源使能。 1，使能；0，关闭。
[13]	TMR3_IE	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TMR3 中断源使能。 1，使能；0，关闭。
[12]	TMR2_IE	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TMR2 中断源使能。 1，使能；0，关闭。
[11]	TMR1_IE	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TMR1 中断源使能。 1，使能；0，关闭。
[10]	TMR0_IE	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TMR0 中断源使能。 1，使能；0，关闭。
[9]	TH31_IE	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH31 中断源使能。 1，使能；0，关闭。
[8]	TH30_IE	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH30 中断源使能。 1，使能；0，关闭。
[7]	TH21_IE	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH21 中断源使能。 1，使能；0，关闭。
[6]	TH20_IE	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH20 中断源使能。 1，使能；0，关闭。
[5]	TH11_IE	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH11 中断源使能。 1，使能；0，关闭。
[4]	TH10_IE	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH10 中断源使能。 1，使能；0，关闭。
[3]	TH01_IE	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH01 中断源使能。 1，使能；0，关闭。
[2]	TH00_IE	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH00 中断源使能。 1，使能；0，关闭。
[1]	T1_IE	t1 事件，计数器的计数值到达 0 中断源使能。 1，使能；0，关闭。
[0]	T0_IE	t0 事件，计数器的计数值回到 MCPWM_TH 中断源使能。 1，使能；0，关闭。

14.2.29 MCPWM_IF0

无写保护的寄存器

地址:0x4001_0784

复位值:0x0

表 14-32 MCPWM_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 MCPWM_TH0/MCPWM_TH00~MCPWM_TH21/MCPWM_TMR 等寄存器更新到 MCPWM 实际运行系统的中断源事件。 1，发生；0，没发生。写 1 清零。
[13]	TMR3_IF	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TMR3 中断源事件。 1，发生；0，没发生。写 1 清零。
[12]	TMR2_IF	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TMR2 中断源事件。 1，发生；0，没发生。写 1 清零。
[11]	TMR1_IF	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TMR1 中断源事件。 1，发生；0，没发生。写 1 清零。
[10]	TMR0_IF	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TMR0 中断源事件。 1，发生；0，没发生。写 1 清零。
[9]	TH31_IF	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH31 中断源事件。 1，发生；0，没发生。写 1 清零。
[8]	TH30_IF	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH30 中断源事件。 1，发生；0，没发生。写 1 清零。
[7]	TH21_IF	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH21 中断源事件。 1，发生；0，没发生。写 1 清零。
[6]	TH20_IF	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH20 中断源事件。 1，发生；0，没发生。写 1 清零。
[5]	TH11_IF	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH11 中断源事件。 1，发生；0，没发生。写 1 清零。
[4]	TH10_IF	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH10 中断源事件。 1，发生；0，没发生。写 1 清零。
[3]	TH01_IF	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH01 中断源事件。 1，发生；0，没发生。写 1 清零。

[2]	TH00_IF	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH00 中断源事件。 1，发生；0，没发生。写 1 清零。
[1]	T1_IF	t1 事件，计数器的计数值到达 0 中断源事件。 1，发生；0，没发生。写 1 清零。
[0]	T0_IF	t0 事件，计数器的计数值回到 MCPWM_TH 中断源事件。 1，发生；0，没发生。写 1 清零。

14.2.30 MCPWM_IE1

写保护的寄存器

地址:0x4001_0788

复位值:0x0

表 14-33 MCPWM_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			TH31_IE	TH30_IE							T1_IE	T0_IE
	RW	RW	RW			RW	RW							RW	RW
	0	0	0			0	0							0	0

位置	位名称	说明
[31:15]		未使用
[14]	UP_IE	MCPWM_TH/MCPWM_TH00~MCPWM_TH31/MCPWM_TMR0~MCPWM_TMR3 等寄存器更新到 MCPWM 实际运行系统的中断源使能。 1，使能；0，关闭。
[13]	TMR3_IE	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TMR3 中断源使能。 1，使能；0，关闭。
[12]	TMR2_IE	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TMR2 中断源使能。 1，使能；0，关闭。
[11]	TMR1_IE	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TMR1 中断源使能。 1，使能；0，关闭。
[10]	TMR0_IE	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TMR0 中断源使能。 1，使能；0，关闭。
[9]	TH31_IE	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH31 中断源使能。 1，使能；0，关闭。
[8]	TH30_IE	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH30 中断源使能。 1，使能；0，关闭。
[7]	TH21_IE	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH21 中断源使能。 1，使能；0，关闭。
[6]	TH20_IE	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH20 中断源使能。 1，使能；0，关闭。
[5]	TH11_IE	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH11 中断源使能。

		1，使能；0，关闭。
[4]	TH10_IE	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH10 中断源使能。 1，使能；0，关闭。
[3]	TH01_IE	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH01 中断源使能。 1，使能；0，关闭。
[2]	TH00_IE	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH00 中断源使能。 1，使能；0，关闭。
[1]	T1_IE	t1 事件，计数器的计数值到达 0 中断源使能。 1，使能；0，关闭。
[0]	T0_IE	t0 事件，计数器的计数值回到 MCPWM_TH 中断源使能。 1，使能；0，关闭。

14.2.31 MCPWM_IF1

无写保护的寄存器

地址:0x4001_078C

复位值:0x0

表 14-34 MCPWM_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			TH31_IF	TH30_IF							T1_IF	T0_IF
	RW1C	RW1C	RW1C			RW1C	RW1C							RW1C	RW1C
	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 MCPWM_TH1/MCPWM_TH30~MCPWM_TH31/MCPWM_TMR 等寄存器更新到 MCPWM 实际运行系统的中断源事件。 1，发生；0，没发生。写 1 清零。
[13]	TMR3_IF	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TMR3 中断源事件。 1，发生；0，没发生。写 1 清零。
[12]	TMR2_IF	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TMR2 中断源事件。 1，发生；0，没发生。写 1 清零。
[11:10]		未使用
[9]	TH31_IF	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH31 中断源事件。 1，发生；0，没发生。写 1 清零。
[8]	TH30_IF	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH30 中断源事件。 1，发生；0，没发生。写 1 清零。
[7:2]		未使用

[1]	T1_IF	t1 事件，计数器的计数值到达 0 中断源事件。 1，发生；0，没发生。写 1 清零。
[0]	T0_IF	t0 事件，计数器的计数值回到 MCPWM_TH 中断源事件。 1，发生；0，没发生。写 1 清零。

14.2.32 MCPWM_EIE

写保护的寄存器

地址:0x4001_0790

复位值:0x0

表 14-35 MCPWM_EIE 配置寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								FAIL3_IE	FAIL2_IE	FAIL1_IE	FAIL0_IE				
								RW	RW	RW	RW				
								0	0	0	0				

位置	位名称	说明
[31:8]		未使用
[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

14.2.33 MCPWM_EIF

无写保护的寄存器

地址:0x4001_0794

复位值:0x0

表 14-36 MCPWM_EIF 配置寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								FAIL3_IF	FAIL2_IF	FAIL1_IF	FAIL0_IF				
								RW1C	RW1C	RW1C	RW1C				

	0	0	0	0	

位置	位名称	说明
[31:8]		未使用
[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

14.2.34 MCPWM_RE

写保护的寄存器

地址:0x4001_0798

复位值:0x0

表 14-37 MCPWM_RE 配置寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								TR1_T1_RE	TR1_T0_RE	TR0_T1_RE	TR0_T0_RE	TMR3_RE	TMR2_RE	TMR1_RE	TMR0_RE
								RW	RW	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

位置	位名称	说明
[31:8]		未使用
[7]	TR1_T1_RE	时基 1 T1 事件 DMA 请求使能。1:使能；0:关闭。
[6]	TR1_T0_RE	时基 1 T0 事件 DMA 请求使能。1:使能；0:关闭。
[5]	TR0_T1_RE	时基 0 T1 事件 DMA 请求使能。1:使能；0:关闭。
[4]	TR0_T0_RE	时基 0 T0 事件 DMA 请求使能。1:使能；0:关闭。
[3]	TMR3_RE	MCPWM 计数器命中 TMR3，DMA 请求使能。1:使能；0:关闭。
[2]	TMR2_RE	MCPWM 计数器命中 TMR2，DMA 请求使能。1:使能；0:关闭。
[1]	TMR1_RE	MCPWM 计数器命中 TMR1，DMA 请求使能。1:使能；0:关闭。
[0]	TMR0_RE	MCPWM 计数器命中 TMR0，DMA 请求使能。1:使能；0:关闭。

14.2.35 MCPWM_PP

写保护的寄存器

地址:0x4001_079C

复位值:0x0

表 14-38 MCPWM_PP 配置寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												IO3_PPE	IO2_PPE	IO1_PPE	IO0_PPE
												RW	RW	RW	RW
												0	0	0	0

位置	位名称	说明
[31:4]		未使用
[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。

14.2.36 MCPWM_IO01

写保护的寄存器

地址:0x4001_07A0

复位值:0x0

表 14-39 MCPWM_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。**

14.2.37 MCPWM_IO23

写保护的寄存器

地址:0x4001_07A4

复位值:0x0

表 14-40 MCPWM_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 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。**

14.2.38 MCPWM_FAIL012

写保护的寄存器

地址:0x4001_07A8

复位值:0x0

表 14-41 MCPWM_FAIL012 配置寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FAIL_0CAP		CH2P_DEFAULT	CH2N_DEFAULT	CH1P_DEFAULT	CH1N_DEFAULT	CH0P_DEFAULT	CH0N_DEFAULT	HALT_PRT	MCPWM_OE	FAIL1_EN	FAIL0_EN	FAIL1_POL	FAIL0_POL	FAIL1_SEL	FAIL0_SEL
RW		RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0		0	0	0	0	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:16]		未使用
[15]	FAIL_0CAP	当发生 fail0/1 事件时，将 MCPWM_CNT0 值存入 MCPWM_FCNT
[14]		未使用
[13]	CH2P_DEFAULT	CH2_P 通道默认值
[12]	CH2N_DEFAULT	CH2_N 通道默认值
[11]	CH1P_DEFAULT	CH1_P 通道默认值
[10]	CH1N_DEFAULT	CH1_N 通道默认值

[9]	CH0P_DEFAULT	CH0_P 通道默认值
[8]	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 MCPWM_IO01 和 MCPWM_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。**
[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 P 和 N 输出值。 1:输出 MCPWM 产生的正常信号 0:输出 CHxN_DEFAULT 和 CHxP_DEFAULT 默认值，此默认值不受极性/通道选择等控制。 MCPWM_EIF.FAIL1_IF 和 MCPWM_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]	FAIL1_SEL	FAIL1 来源选择。1:比较器；0:来自 GPIO。
[0]	FAIL0_SEL	FAIL0 来源选择。1:比较器；0:来自 GPIO。

对于 MCPWM 的通道 0/1/2，FAIL0 信号来源为比较器 0 或 MCPWM0_BKIN0；FAIL1 信号来源为比较器 1 或 MCPWM0_BKIN1。

FAIL0/1 信号用于工作于时基 0 的 MCPWM 通道 0/1/2。

14.2.39 MCPWM_FAIL3

写保护的寄存器

地址:0x4001_07AC

复位值:0x0

表 14-42 MCPWM_FAIL3 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15]	FAIL_1CAP	当发生 fail2/3 事件时，将 MCPWM_CNT1 值存入 MCPWM_FCNT
[14:10]		未使用
[9]	CH3N_DEFAULT	CH3 N 通道默认值
[8]	CH3P_DEFAULT	CH3 P 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 MCPWM_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。**
[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 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。 MCPWM_EIF.FAIL2_IF 和 MCPWM_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]	FAIL3_SEL	FAIL3 来源选择。1:比较器 1 的结果；0:来自 GPIO 第 1 路。
[0]	FAIL2_SEL	FAIL2 来源选择。1:比较器 0 的结果；0:来自 GPIO 第 0 路。

对于 MCPWM 的通道 3，FAIL2 信号来源为比较器 0 或 MCPWM0_BKIN0；FAIL3 信号来源为比较器 1 或 MCPWM0_BKIN1。

FAIL2/3 信号用于工作于时基 1 的 MCPWM 通道 3。

MCPWM_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 的急停事件有两个 FAIL2 和 FAIL3。 每个 FAIL 信号共有 2 个信号来源可选，比较器输出或 MCPWM_BKIN。

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 MCPWM_TCLK.CLK_DIV 寄存器设置。信号来源比较器输出的滤波时钟分频由 MCPWM_TCLK.CMP_FLT_CLKDIV 设置；信号来源 MCPWM_BKIN 的滤波时钟分频由 MCPWM_TCLK.IO_FLT_CLKDIV 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。

14.2.40 MCPWM_PRT

无写保护的寄存器

地址:0x4001_07B0

复位值:0x0

表 14-43 MCPWM_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。

14.2.41 MCPWM_SWAP

写保护的寄存器，非开放寄存器

地址:0x4001_07B4

复位值:0x0

表 14-44 MCPWM_SWAP 寄存器

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

位置	位名称	说明
[31:1]		未使用
[0]	MCPWM_SWAP	向此寄存器写入 0x67 可将 BIT[0]写为 1，写其他值则将 BIT[0]写为 0

MCPWM_SWAP 的值为 0 时，MCPWM 通道输出与 GPIO 关系如下:

表 14-45 MCPWM 默认输出 IO

MCPWM 输出顺序	GPIO 对应顺序
MCPWM_CH0P	P0.10
MCPWM_CH0N	P0.11
MCPWM_CH1P	P0.12
MCPWM_CH1N	P0.13

MCPWM_CH2P	P0.14
MCPWM_CH2N	P0.15
MCPWM_CH3P	P1.8
MCPWM_CH3N	P1.9

MCPWM_SWAP 的值为 1 时，用于包含预驱芯片应用环境。在逻辑内部转换顺序，方便芯片封装，一般应用上只需要三组 MCPWM 通道，因此仅转换三组的顺序。关系如下:

表 14-46 MCPWM 通道映射后输出 IO

MCPWM 输出顺序	GPIO 对应顺序
MCPWM_CH0N	P0.10
MCPWM_CH1N	P0.11
MCPWM_CH2N	P0.12
MCPWM_CH0P	P0.13
MCPWM_CH1P	P0.14
MCPWM_CH2P	P0.15
MCPWM_CH3P	P1.8
MCPWM_CH3N	P1.9

14.2.42 MCPWM_CHMSK

写保护的寄存器

地址:0x4001_07B8

复位值:0x0

表 14-47 MCPWM_CHMSK 通道屏蔽位寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										CH2P_FAIL_EN	CH2N_FAIL_EN	CH1P_FAIL_EN	CH1N_FAIL_EN	CH0P_FAIL_EN	CH0N_FAIL_EN
										RW	RW	RW	RW	RW	RW
										1	1	1	1	1	1

位置	位名称	说明
[31:6]		未使用
[5]	CH2P_FAIL_EN	CH2_P FAIL 事件通道屏蔽使能，高有效，默认开启
[4]	CH2N_FAIL_EN	CH2_N FAIL 事件通道屏蔽使能，高有效，默认开启
[3]	CH1P_FAIL_EN	CH1_P FAIL 事件通道屏蔽使能，高有效，默认开启
[2]	CH1N_FAIL_EN	CH1_N FAIL 事件通道屏蔽使能，高有效，默认开启
[1]	CH0P_FAIL_EN	CH0_P FAIL 事件通道屏蔽使能，高有效，默认开启

[0]	CH0N_FAIL_EN	CH0_N FAIL 事件通道屏蔽使能，1：发生 FAIL 事件时，CH0_N 通道电平输出为默认值，0：发生 FAIL 事件时，CH0_N 通道电平不受影响，仍由 MCPWM 内部硬件控制。默认开启 FAIL 关闭机制

15 GPIO

15.1 概述

LSK32MC03x 系列芯片共集成了 2 组 GPIO，P0 包含 16 个 GPIO，P1 包含 10 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

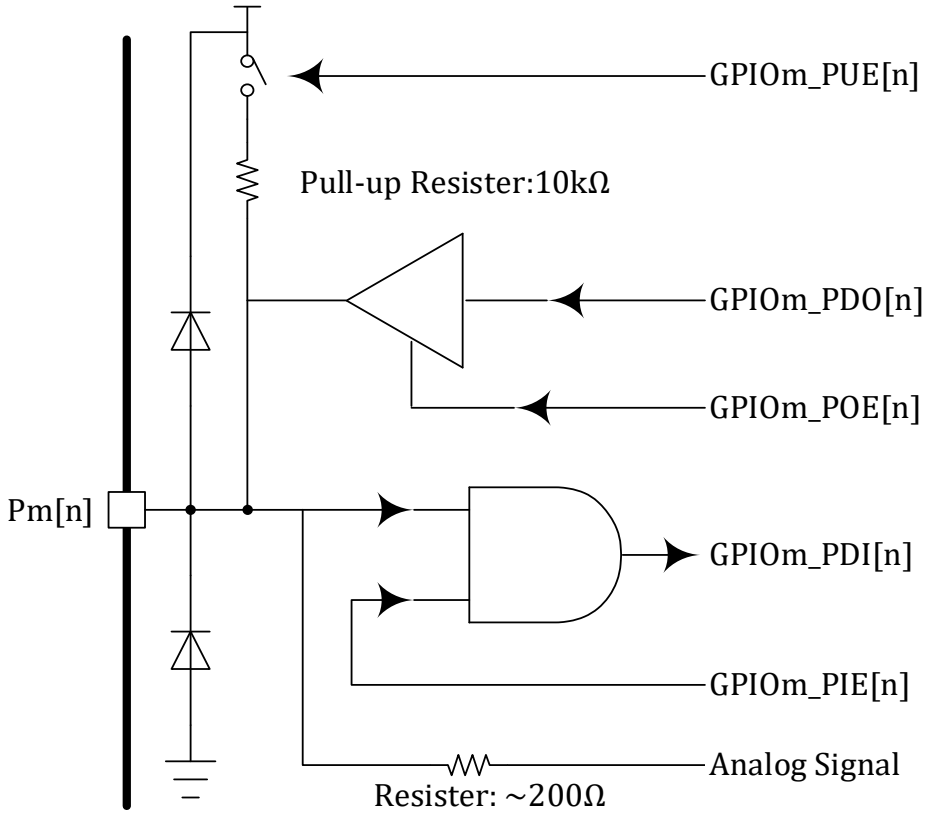


图 15-1 GPIO 功能框图

如上图所示，Pm[n]为芯片 PAD，m 可以是 0~1，表示 2 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。模拟信号通过一个电阻串联直接连接到 PAD，电阻阻值为 100~200Ω。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 100kΩ，其余上拉电阻为 10kΩ，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

- 26 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 20.6.5

具体哪些 GPIO 可以进行输入信号滤波，请参考 15.3.2

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_0800。

GPIO 1 模块在芯片中的基地址是 0x4001_0840。

GPIO 0/1 的寄存器定义完全相同，仅基地址不同。

表 15-1 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_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/唤醒/配置锁定模块的基地址是 0x4001_0880。

表 15-2 GPIO 中断/唤醒/配置锁定模块寄存器列表

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

15.2.2 GPIOx_PIE

地址分别是:0x4001_0800，0x4001_0840

复位值:0x0

表 15-3 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_0804，0x4001_0844

复位值:0x0

表 15-4 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_0808，0x4001_0848

复位值:0x0

表 15-5 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

当 GPIOx_PIE=0 时，GPIOx_PDI 读回为 0。

15.2.5 GPIOx_PDO

地址分别是:0x4001_080C，0x4001_084C

复位值:0x0

表 15-6 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_0810，0x4001_0850

复位值:0x0

表 15-7 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_0818，0x4001_0858

复位值:0x0

表 15-8 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_081C，0x4001_085C

复位值:0x0

表 15-9 GPIOx 配置锁定寄存器 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_0820，0x4001_0860

复位值:0x0

表 15-10 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 15-11 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。

表 15-11 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，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，TIMER0 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
8	TIMER1，TIMER1 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
9	ADC_TRIGGER，ADC 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

15.2.10 GPIOx_F7654

地址分别是:0x4001_0824，0x4001_0864

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_0828，0x4001_0868

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_082C，0x4001_086C

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_0830，0x4001_0870

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_0834，0x4001_0874

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 信号外部触发的电平类型。

15.2.15.1 EXTI_CR0

地址:0x4001_0880

复位值:0x0

表 15-17 外部触发配置寄存器 0 EXTI_CR0

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

位置	位名称	说明
[31:16]		未使用
[15:14]	T7	GPIO 0[9]/ P0[9]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[8]/ P0[8]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[7]/ P0[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]	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[5]/ P0[5]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[4]/ P0[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
[3:2]	T1	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
[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

15.2.15.2EXTI_CR1

地址:0x4001_0884

复位值:0x0

表 15-18 外部触发配置寄存器 1 EXTI_CR1

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

位置	位名称	说明
[31:16]		未使用
[15:14]	T15	GPIO 1[9]/P1[9]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 1[8]/P1[8]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 1[7]/P1[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 1[6]/P1[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]	T11	GPIO 1[5]/P1[5]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[4]/P1[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
[3:2]	T9	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
[1:0]	T8	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

表 15-19 GPIO 中断资源分布表

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

15.2.15.3EXTI_IE

地址:0x4001_0888

复位值:0x0

表 15-20 GPIO 中断使能寄存器 EXTI_IE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						EXIT1_RE	EXTI0_RE							EXIT1_IE	EXTI0_IE
						RW	RW							RW	RW
						0	0							0	0

位置	位名称	说明
[31:10]		未使用
[9]	EXTI1_RE	GPIO1 DMA 请求使能，需要配合 EXTI_CR1 使用
[8]	EXTI0_RE	GPIO0 DMA 请求使能，需要配合 EXTI_CR0 使用
[7:2]		未使用
[1]	EXTI1_IE	GPIO1 中断使能，需要配合 EXTI_CR1 使用
[0]	EXTI0_IE	GPIO0 中断使能，需要配合 EXTI_CR0 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。

15.2.15.4EXTI_IF

地址:0x4001_088C

复位值:0x0

表 15-21 外部中断标志寄存器 EXTI_IF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IF15	IF14	IF13	IF12	IF11	IF10	IF9	IF8	IF7	IF6	IF5	IF4	IF3	IF2	IF1	IF0
RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:16]		未使用
[15]	IF15	GPIO 1[9] / P1[9] 外部中断标志。中断标志高有效，写 1 清零
[14]	IF14	GPIO 1[8] / P1[8] 外部中断标志。中断标志高有效，写 1 清零
[13]	IF13	GPIO 1[7] / P1[7] 外部中断标志。中断标志高有效，写 1 清零
[12]	IF12	GPIO 1[6] / P1[6] 外部中断标志。中断标志高有效，写 1 清零

[11]	IF11	GPIO 1[5] / P1[5] 外部中断标志。中断标志高有效，写 1 清零
[10]	IF10	GPIO 1[4] / P1[4] 外部中断标志。中断标志高有效，写 1 清零
[9]	IF9	GPIO 0[15] / P0[15] 外部中断标志。中断标志高有效，写 1 清零
[8]	IF8	GPIO 0[14] / P0[14] 外部中断标志。中断标志高有效，写 1 清零
[7]	IF7	GPIO 0[9] / P0[9] 外部中断标志。中断标志高有效，写 1 清零
[6]	IF6	GPIO 0[8] / P0[8] 外部中断标志。中断标志高有效，写 1 清零
[5]	IF5	GPIO 0[7] / P0[7] 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[5] / P0[5] 外部中断标志。中断标志高有效，写 1 清零
[2]	IF2	GPIO 0[4] / P0[4] 外部中断标志。中断标志高有效，写 1 清零
[1]	IF1	GPIO 0[2] / P0[2] 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

15.2.15.5 CLKO_SEL

地址:0x4001_0890

复位值:0x0

表 15-22 GPIO 输出时钟信号选择寄存器 CLKO_SEL

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

位置	位名称	说明
[31:4]		未使用
[3]	ADC_OE	ADC 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	HSI 输出使能。1:使能；0:禁用。
[0]	LSI_OE	LSI 输出使能。1:使能；0:禁用。

芯片各时钟信号可以从 P0.10 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 P0.10 的第二功能为 1，即 GPIO0_FBA98 = 0x0100，并使能 P0.10 输出使能 GPIO0_POE=0x0400，并配置 CLKO_SEL，选择某一路时钟通过 P0.10 进行输出。

15.3 实现说明

15.3.1 上拉实现

LKS32MC03x 系列芯片，部分 GPIO 配备有 10kΩ 上拉电阻。配备上拉功能的 GPIO 如下:

表 15-23 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							√	√	√	√	√	√				

15.3.2 滤波实现

LKS32MC03x 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 15-24 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							√	√			√					

15.4 应用指南

15.4.1 外部中断

示例如下:

```
GPIO0_PIE = 0x0080;                 // 使能 P0[7]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 = 0x8000;                  // 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 = 0x4000;                  // 使能 p0[7]下降沿触发，产生外部中断
  while(irq_flag != 4);
EXTI_CR0 = 0xC000;                  // 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 UART

16.1 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
- 支持一主多从的 Multi-drop Slave/Master 模式

16.2 功能说明

16.2.1 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。

16.2.2 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。

16.2.3 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.

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 16-1 UART 帧格式

16.2.4 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 16-1 UART 波特率配置示例

UART 波特率	SYS_CLK_DIV2	UART_DIVH	UART_DIVL
150	0x0007	0x9C	0x3F
300	0x0003	0x9C	0x3F
600	0x0001	0x9C	0x3F
1200	0x0000	0x9C	0x3F
2400	0x0000	0x4E	0x1F
4800	0x0000	0x27	0x0F
9600	0x0000	0x13	0x87

19200	0x0000	0x09	0xC3
38400	0x0000	0x04	0xE1
57600	0x0000	0x03	0x40
115200	0x0000	0x01	0x9F

注意，同一波特率，其配置系数可能不唯一。

16.2.5 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。

16.2.6 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 16-2 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。

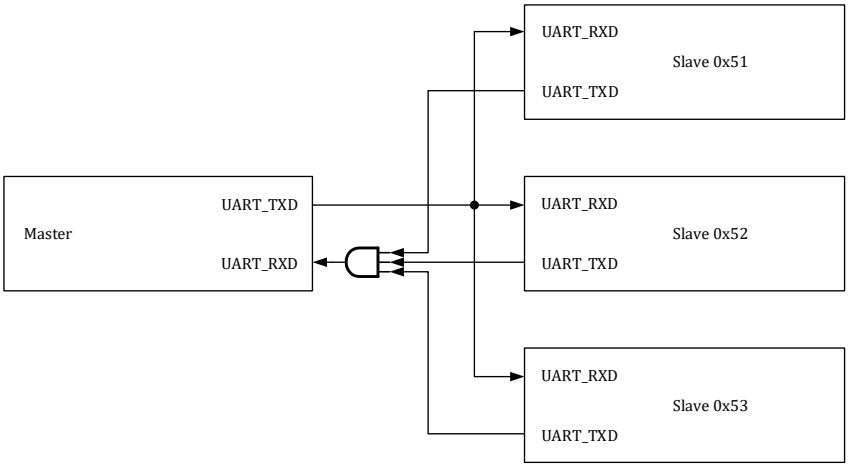


图 16-2 UART 多机通讯互联拓扑

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。每一个从机有一个 8bit 的设备地址，即 UART_ADR。主机、从机均需要设置 UART_CTRL.MD_EN=1 后，开启多机通讯过滤模式，同时设置 UART_CTRL.BYTE_LEN=1 使用 9bit 数据长度。主机发送数据字节 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。

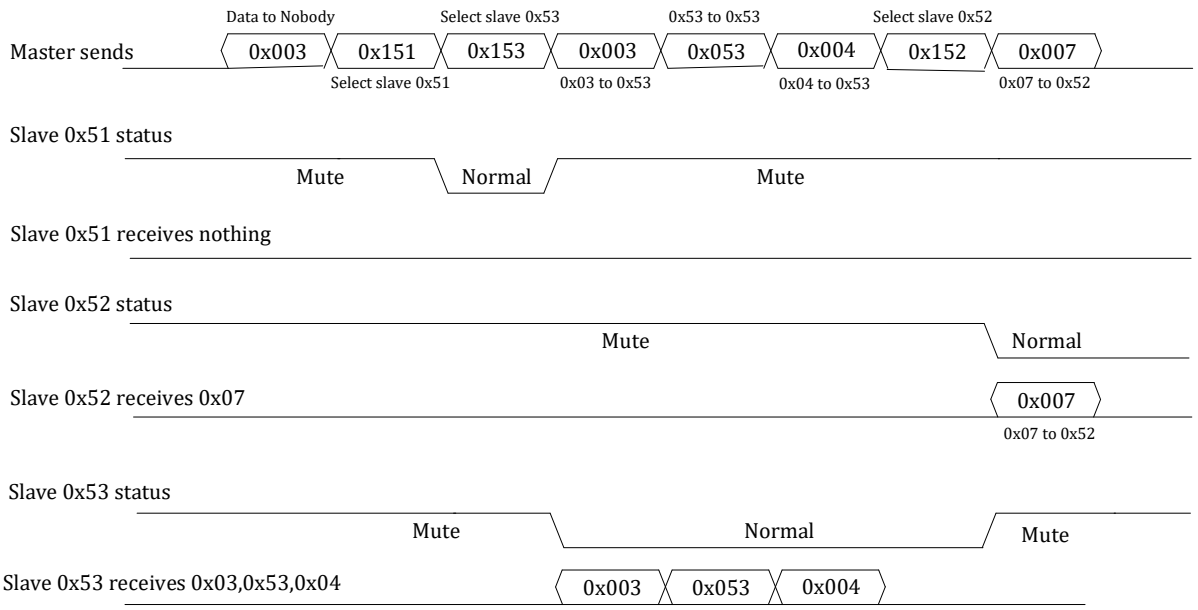
在多机通讯模式中，主设备亦需设置 UART_CTRL.MD_EN=1，从机发送数据给主机时，应先发送 1+主机 UART_ADR 的 9bit 数据，此时主机地址命中，后续无需再发送最高位为 1 的地址字节，可直接发送最高位为 0 的数据字节。其他从机亦可先发送 1+主机 UART_ADR 的 9bit 数据，保持主机地址命中，再发送最高位为 0 的数据字节，或者在主机地址命中的情况下，直接发送数据字节。

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。

注意:多机通讯暂不支持校验位,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。

如图 16-3 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。



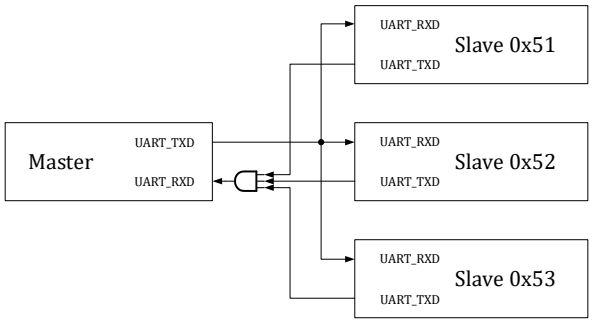


图 16-3 UART 多机通讯示例

16.2.7 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。UART_CTRL.BIT_ORDER 的设置，即 LSB first 或 MSB first 不影响帧格式。

表 16-2 UART 帧格式

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

16.3 寄存器

16.3.1 地址分配

UART 基地址 0x40010900。

表 16-3 UART 地址分配列表

名称	偏移地址	说明
UART_CTRL	0x00	UART 控制寄存器
UART_DIVH	0x04	UART 波特率设置高字节寄存器
UART_DIVL	0x08	UART 波特率设置低字节寄存器
UART_BUFF	0x0C	UART 收发缓冲寄存器
UART_ADR	0x10	485 通信地址匹配寄存器
UART_STT	0x14	UART 状态寄存器
UART_RE	0x18	UART DMA 请求使能寄存器
UART_IE	0x1C	UART 中断使能寄存器
UART_IF	0x20	UART 中断标志寄存器
UART_IOC	0x24	UART IO 控制

16.3.2 UART_CTRL UART 控制寄存器

地址:0x4001_0900

复位值:0x0

表 16-4 UART 控制寄存器 UART_CTRL

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

位置	位名称	说明
[31:7]		未使用
[6]		保留
[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

16.3.3 UART_DIVH UART 波特率设置高字节寄存器

地址:0x4001_0904

复位值:0x0

表 16-5 UART 波特率设置高字节寄存器 UART_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)

16.3.4 UART_DIVL UART 波特率设置低字节寄存器

地址:0x4001_0908

复位值:0x0

表 16-6 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)

16.3.5 UART_BUFF UART 收发缓冲寄存器

表 16-7 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。

16.3.6 UART_ADR UART 地址匹配寄存器

地址:0x4001_0910

复位值:0x0

表 16-8 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

16.3.7 UART_STT UART 状态寄存器

地址:0x4001_0914

复位值:0x0

表 16-9 UART 状态寄存器 UART_STT

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

位置	位名称	说明
[31:3]		未使用
[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。

16.3.8 UART_RE UART DMA 请求使能寄存器

地址分别是:0x4001_0918

复位值:0x0

表 16-10 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_RE	RX_DONE_RE	TX_DONE_RE
													RW	RW	RW
													0	0	0

位置	位名称	说明
[31:3]		未使用
[2]	TX_BUF_EMPTY_RE	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_RE	接收完成 DMA 请求开关，默认值为 0。 0:关闭；1:开启。
[0]	TX_DONE_RE	发送完成 DMA 请求开关，默认值为 0。

		0:关闭；1:开启。

16.3.9 UART_IE UART 中断使能寄存器

地址:0x4001_091C

复位值:0x0

表 16-11 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
									Resv.	Resv.	CK_ERR_IE	STOP_ERR_IE	TX_BUF_EMPTY_IE	RX_DONE_IE	TX_DONE_IE
									RW	RW	RW	RW	RW	RW	RW
									0	0	0	0	0	0	0

位置	位名称	说明
[31:7]		未使用
[6:5]		保留位，必须为 0
[4]	CK_ERR_IE	校验错误中断开关，默认值为 0。 0:关闭；1:开启
[3]	STOP_ERR_IE	停止位错误中断开关，默认值为 0。 0:关闭；1:开启
[2]	TX_BUF_EMPTY_IE	发送缓冲区空中断开关，默认值为 0。 0:关闭；1:开启
[1]	RX_DONE_IE	接收完成中断开关，默认值为 0。 0:关闭；1:开启
[0]	TX_DONE_IE	发送完成中断开关，默认值为 0。 0:关闭；1:开启

16.3.10 UART_IF UART 中断标志寄存器

地址:0x4001_0920

复位值:0x0

表 16-12 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
											CK_ERR_IF	STOP_ERR_IF	TX_BUF_EMPTY_IF	RX_DONE_IF	TX_DONE_IF

	RW1C	RW1C	RW1C	RW1C	RW1C
	0	0	1	0	1

位置	位名称	说明
[31:7]		未使用
[6:5]		保留
[4]	CK_ERR_IF	校验错误中断标志，高有效，写 1 清零
[3]	STOP_ERR_IF	停止位错误中断标志，高有效，写 1 清零
[2]	TX_BUF_EMPTY_IF	发送缓冲区空中断标志，高有效，写 1 清零
[1]	RX_DONE_IF	接收完成中断标志，高有效，写 1 清零
[0]	TX_DONE_IF	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;

16.3.11 UART_IOC UART IO 控制寄存器

地址:0x4001_0924

复位值:0x0

表 16-13 UART IO 控制寄存器 UART_IOC

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

位置	位名称	说明
[31:5]		未使用
[4]		保留
[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。

17 DMA

17.1 概述

DMA 和 CPU 同为芯片总线的主设备。

如图 17-1 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。

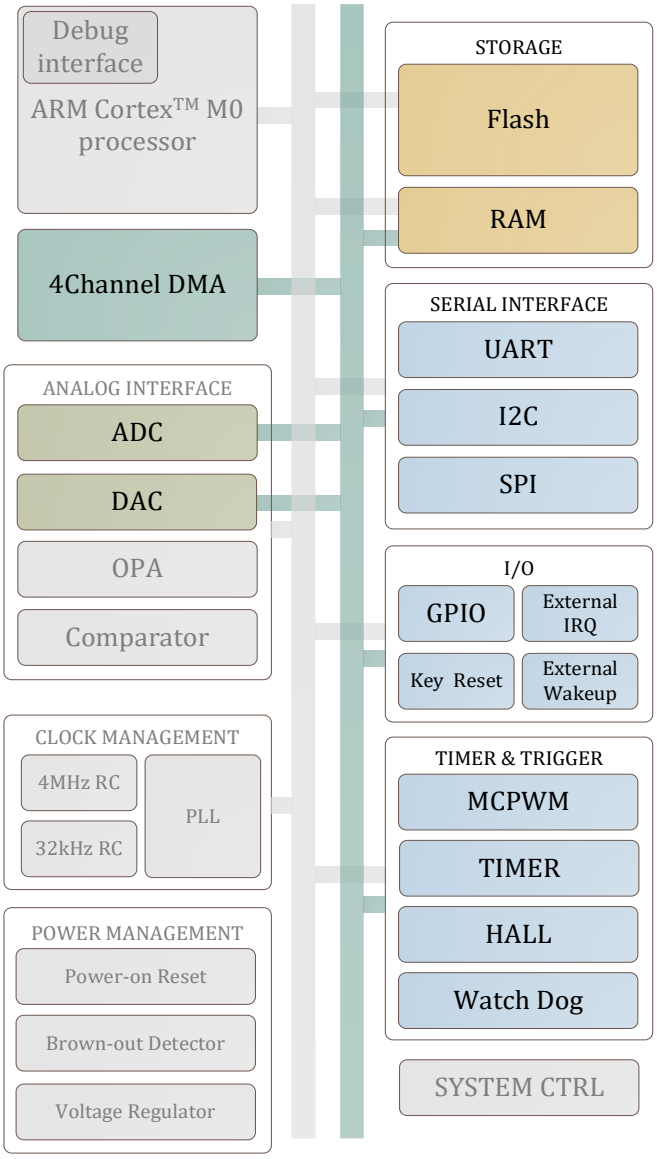


图 17-1 multi-layer AHB 总线架构

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。

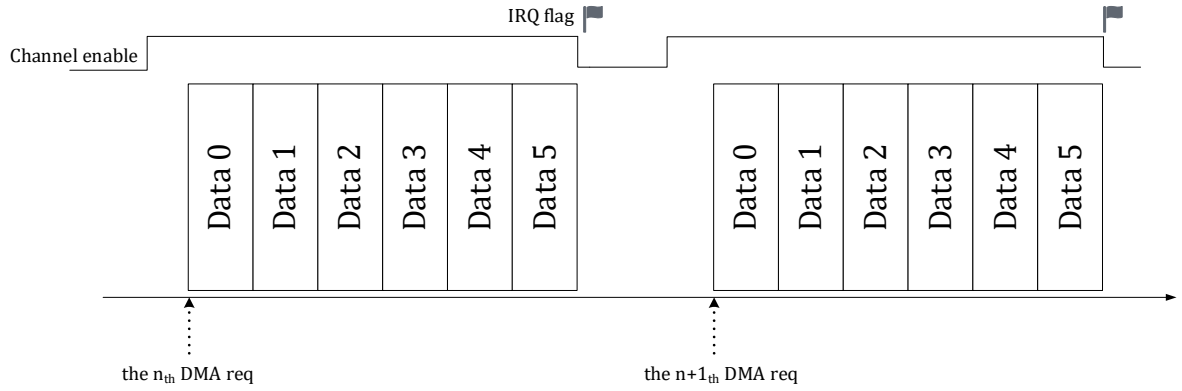


图 17-2 RMODE=0 DMA 传输情况

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。

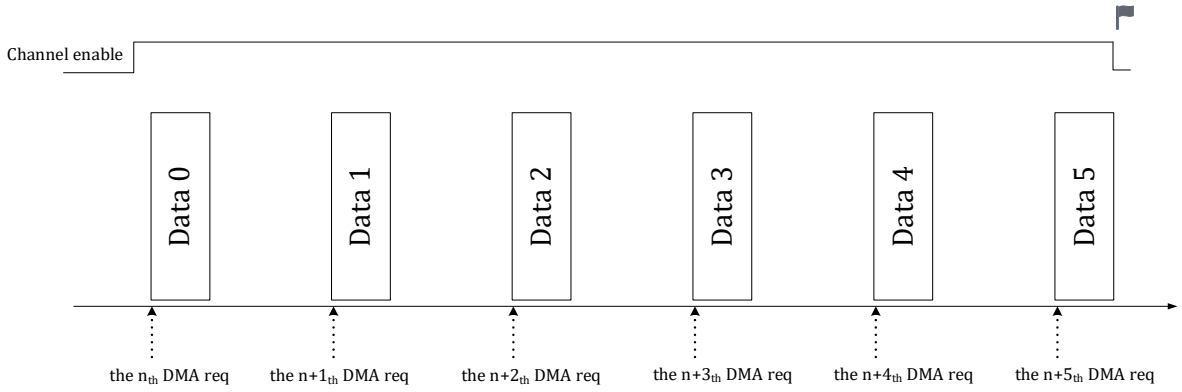


图 17-3 RMODE=1 DMA 传输情况

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。

每发生一次 DMA 请求，DMA 开始一轮传输，完成一轮(多次)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。

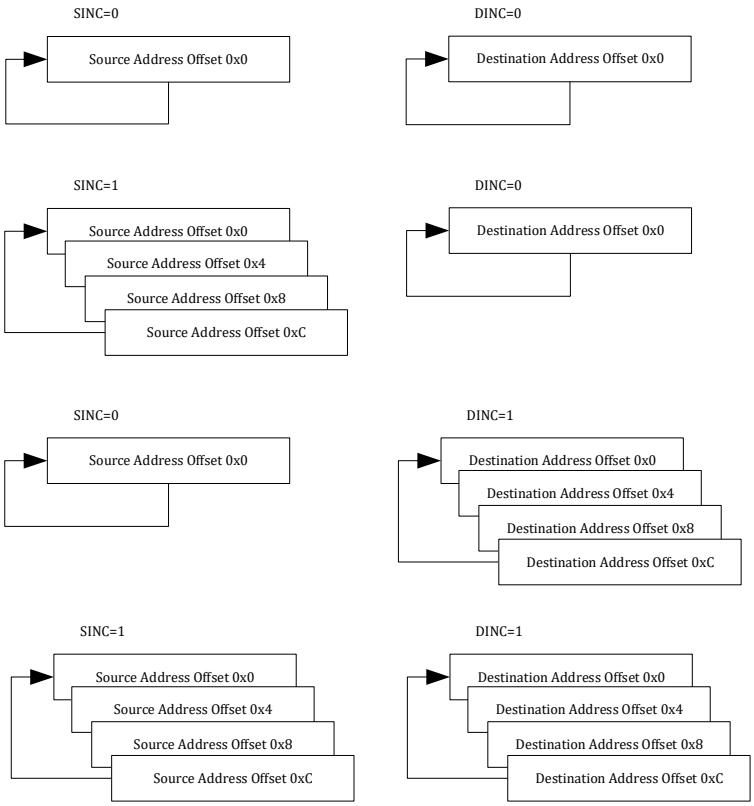


图 17-4 DMA 地址递增控制

图 17-4 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。

DMA 有循环模式和单次模式两种模式，由 DMA_CCRx.CIRC 控制。循环模式下 DMA 完成一轮数据块搬运后重新开始下一轮搬运，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 UART 数据搬运至内存为例，则在循环模式下 DMA 完成一轮（一字节）搬运后重新开始下一轮搬运，不设置 DMA 传输完成中断标志位。

注意：循环模式下，如果设置了地址递增，如 DMA_CCRx.SINC 或 DMA_CCRx.DINC=1，DMA 固定使用 256 大小的地址空间，达到 256 后计数回 0。通常，循环模式用于 UART/SPI/I2C 的数据搬运

至内存，通过查询 DMA_CTMSx 得知传输数据已经累积足够长度后，比如达到一个数据帧的长度，CPU 进行数据处理。因此，需要内存中开辟 256 大小的 char 型数组。

单次模式下，DMA 完成一定大小的数据块搬运后即完成本次 DMA 操作，设置完成 DMA 传输完成中断标志位，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。

17.2 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 17-1 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。
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。
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。
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
Timer	Timer 可以使用过零/比较事件作为 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）
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
GPIO	GPIO 中断可以作为 DMA 请求
HALL	HALL 中断可以作为 DMA 请求

17.3 优先级

DMA 的优先级采用固定优先级，优先级如图 17-5 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 17-5 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。

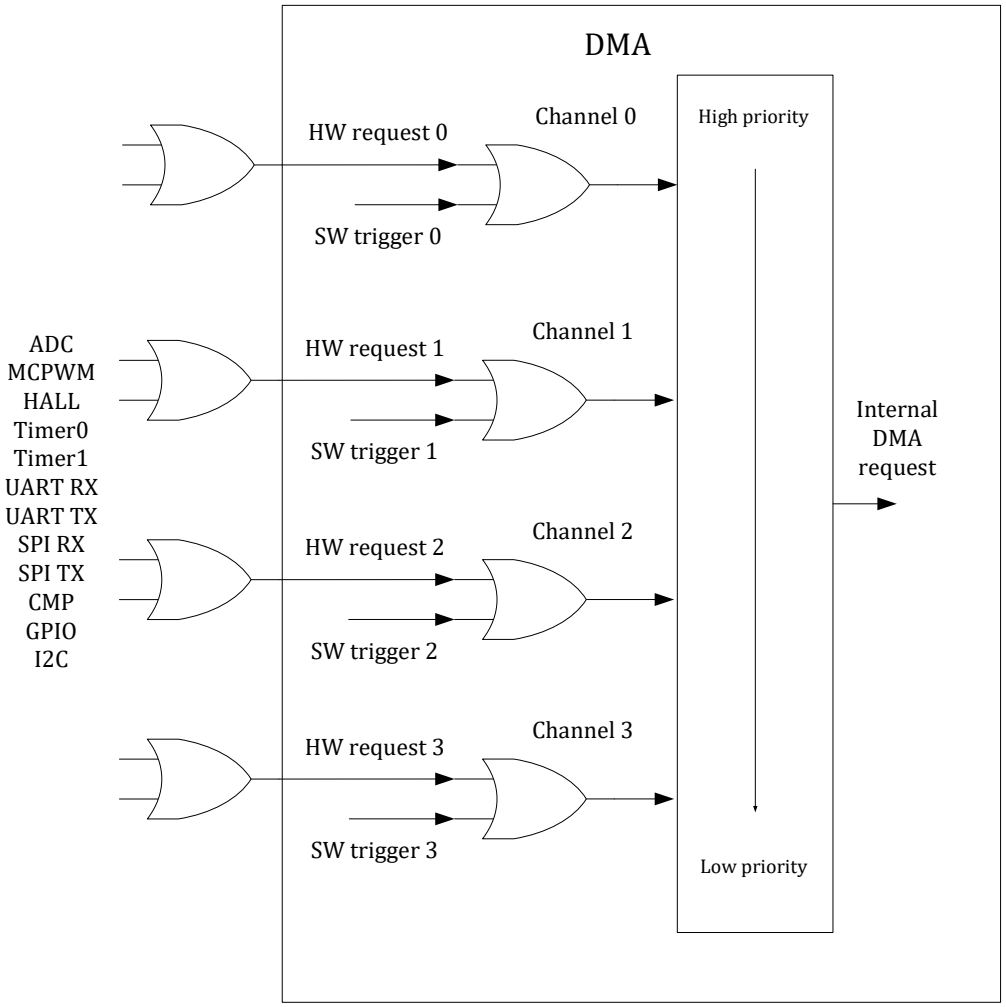


图 17-5 DMA 通道优先级

17.4 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。

17.5 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为 DMA_CCRx.EN。

17.6 寄存器

17.6.1 地址分配

DMA 控制器模块寄存器的基地址是 0x4001_0A00，寄存器列表如下:

表 17-2 DMA 寄存器列表

名称	偏移地址	说明
DMA_CCR0	0x00	DMA 通道 0 通道配置寄存器
DMA_REN0	0x04	DMA 通道 0 请求使能寄存器
DMA_CTMS0	0x08	DMA 通道 0 传输次数寄存器
DMA_SADR0	0x0C	DMA 通道 0 源地址寄存器
DMA_DADR0	0x10	DMA 通道 0 目的地址寄存器
DMA_CCR1	0x20	DMA 通道 1 通道配置寄存器
DMA_REN1	0x24	DMA 通道 1 请求使能寄存器
DMA_CTMS1	0x28	DMA 通道 1 传输次数寄存器
DMA_SADR1	0x2C	DMA 通道 1 源地址寄存器
DMA_DADR1	0x30	DMA 通道 1 目的地址寄存器
DMA_CCR2	0x40	DMA 通道 2 通道配置寄存器
DMA_REN2	0x44	DMA 通道 2 请求使能寄存器
DMA_CTMS2	0x48	DMA 通道 2 传输次数寄存器
DMA_SADR2	0x4C	DMA 通道 2 源地址寄存器
DMA_DADR2	0x50	DMA 通道 2 目的地址寄存器
DMA_CCR3	0x60	DMA 通道 3 通道配置寄存器
DMA_REN3	0x64	DMA 通道 3 请求使能寄存器
DMA_CTMS3	0x68	DMA 通道 3 传输次数寄存器
DMA_SADR3	0x6C	DMA 通道 3 源地址寄存器
DMA_DADR3	0x70	DMA 通道 3 目的地址寄存器
DMA_CTRL	0x80	DMA 控制寄存器
DMA_IE	0x84	DMA 中断使能寄存器

DMA_IF	0x88	DMA 中断标志寄存器

17.6.2 DMA_CTRL DMA 控制寄存器

地址:0x4001_0A80

复位值:0x0

表 17-3 DMA 控制寄存器 DMA_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

17.6.3 DMA_IE DMA 中断使能寄存器

地址:0x4001_0A84

复位值:0x0

表 17-4 DMA 中断使能寄存器 DMA_IE

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

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

17.6.4 DMA_IF DMA 中断标志寄存器

地址:0x4001_0A88

复位值:0x0

表 17-5 DMA 中断标志寄存器 DMA_IF

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

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

17.6.5 DMA 通道配置寄存器

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

地址分别是:0x4001_0A00，0x4001_0A18，0x4001_0A30，0x4001_0A48

复位值:0x0

表 17-6 DMA 通道配置寄存器 DMA_CCRx

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

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

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

地址分别是:0x4001_0A04，0x4001_0A1C，0x4001_0A34，0x4001_0A4C

复位值:0x0

表 17-7 DMA 请求使能寄存器 DMA_ RENx

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

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。

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

地址分别是:0x4001_0A08，0x4001_0A20，0x4001_0A38，0x4001_0A50

复位值:0x0

表 17-8 DMA 传输次数寄存器 DMA_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;

LKSMC03x 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。

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

地址分别是:0x4001_0A0C，0x4001_0A24，0x4001_0A3C，0x4001_0A54

复位值:0x0

表 17-9 DMA 源地址寄存器 DMA_ SADRx

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

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

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！！！

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

地址分别是:0x4001_0A10，0x4001_0A28，0x4001_0A40，0x4001_0A58

复位值:0x0

表 17-10 DMA 目的地址寄存器 DMA_DADRx

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

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

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！！！

18 DSP 协处理器

18.1 概述

除法器被除数和除数均为 32 位有符号数，32 个总线周期（最高 48MHz）完成计算。

除数为 0 时，除法计算结果不确定。

除法器的操作数：被除数、除数应限幅为-(2^31-1) ~ (2^31-1)，不支持赋值为-2^31。

使用除法器时，软件应先写入被除数 DSP_DID，然后写入除数 DSP_DIS。写入除数 DSP_DIS 的同时，触发一次除法操作。除法操作进行过程中，软件可以执行其他指令。但若在除法计算完成前读取结果(DSP_QUP 或 DSP_REM)，则 CPU 会等待除法器计算完成取回结果，再继续执行后续指令。在除法计算完成前向除法器写入新的操作数（DSP_DID 或 DSP_DIS）也会导致 CPU 进入等待状态，直到除法器完成当前计算才允许新操作数写入。以上两种情况除了导致 CPU 等待若干总线周期，无其他影响。

开方模块被开方数为 32 位无符号数，结果为 16 位无符号数，16 个总线周期（最高 48MHz）完成计算。

使用开方模块时，写入被开方数 DSP_RAD 即触发开方器开始运算，运算完毕可以从 DSP_SQRT 寄存器读取开方结果。开方运算计算过程中，软件可以执行其他指令。但若在计算完成前读取结果(DSP_SQRT)，则 CPU 会等待开方计算完成取回结果，再继续执行后续指令。在开方计算完成前向其写入新操作数(DSP_RAD)也会导致 CPU 进入等待状态，直到开方运算完成当前计算才允许新操作数的写入。以上两种情况除了导致 CPU 等待若干总线周期，无其他影响。

18.2 寄存器

18.2.1 地址分配

除法器寄存器在芯片中的基地址是 0x4001_0B00。

表 18-1 DSP 寄存器列表

名称	偏移	说明
DSP_DID	0x20	DSP 除法操作被除数
DSP_DIS	0x24	DSP 除法操作除数
DSP_QUO	0x28	DSP 除法操作商
DSP_REM	0x2C	DSP 除法操作余数
DSP_RAD	0x30	DSP 开方操作被开方数
DSP_SQRT	0x34	DSP 开方操作平方根

18.2.2 DSP 除法相关寄存器

18.2.2.1 DSP_DID

地址:0x4001_0B20

复位值:0x0

表 18-2 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															
WO															
0															

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

位置	位名称	说明
[31:0]	DID	除法被除数寄存器

18.2.2.2 DSP_DIS

地址:0x4001_0B24

复位值:0x0

表 18-3 除法除数寄存器

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

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

位置	位名称	说明
[31:0]	DIS	除法除数寄存器

18.2.2.3 DSP_QUO

地址:0x4001_0B28

复位值:0x0

表 18-4 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	商寄存器

18.2.2.4 DSP_REM

地址:0x4001_0B2C

复位值:0x0

表 18-5 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															

位置	权限	说明
[31:0]	REM	除法余数寄存器

18.2.3 DSP 开方相关寄存器

18.2.3.1 DSP_RAD

地址：0x4001_0B30

复位值：0x0

表 18-6 DSP 被开方数寄存器

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

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

位置	位名称	说明
[31:0]	RAD	DSP 被开方数寄存器

18.2.3.2 DSP_SQRT

地址：0x4001_0B34

复位值：0x0

表 18-7 DSP 平方根寄存器

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

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，向 DSP 写入被开方数可以触发一次开方操作，开方计算期间读取开方结果 DSP_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。软件可以在写入操作数后立即读取开方结果。

19 IWDG 独立看门狗

19.1 概述

独立看门狗使用 64kHz 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。

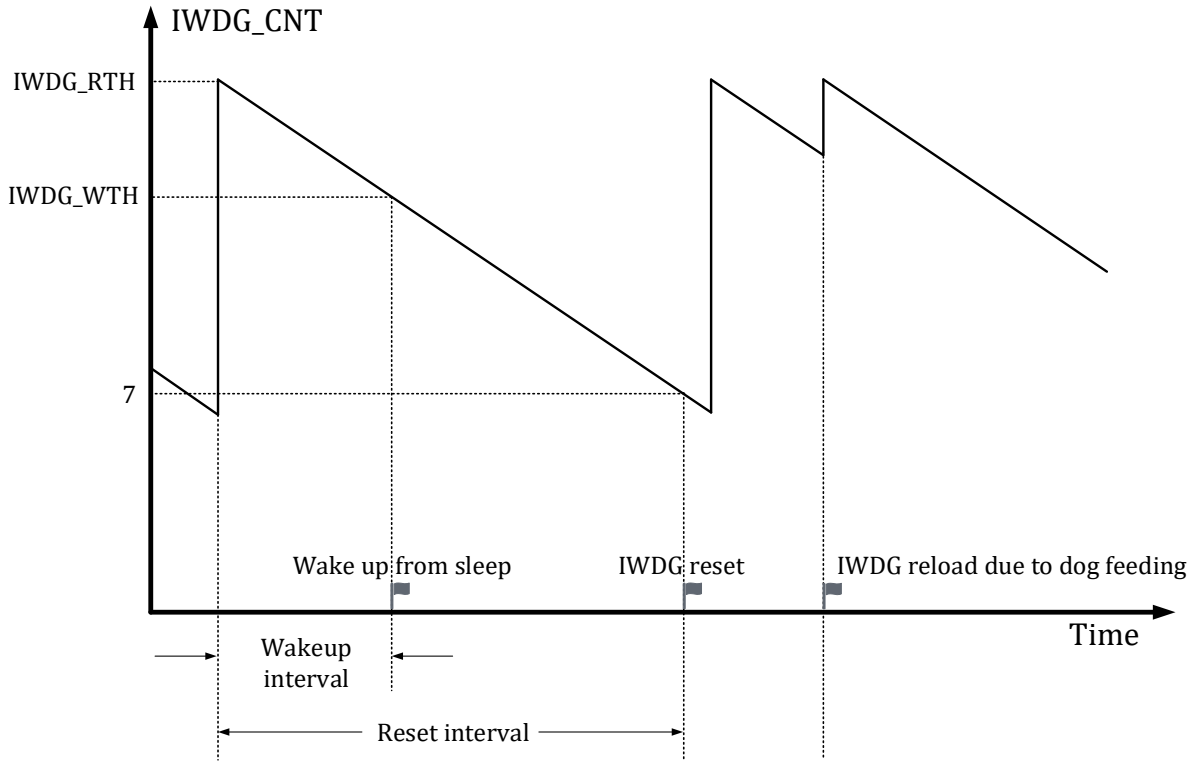


图 19-1 IWDG 计数时序

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，1/64kHz=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 4096/64kHz≈128ms。

IWDG_RTH=0x1FF000 对应独立看门狗最大复位时间间隔为 511×4096/64kHz≈64s。

关于 IWDG 的跨时钟喂狗，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%。

由于 IWDG 的计数器工作于 LRC 时钟域，而软件运行于系统主时钟域，通常为 PLL 时钟。当软件通过写入 IWDG_RTH 或写入 IWDG_CLR 进行喂狗时，从软件进行写入到看门狗计数器被重置，需要最多 3 个 LRC 时钟周期。

因此如果软件需要在喂狗后读取 IWDG_CNT，至少需要延时 3 个 LRC 时钟周期，读取 IWDG_CNT 才会发现计数器被重置。

软件写入修改 IWDG_RTH/IWDG_WTH/IWDG_CFG/IWDG_PSW 等寄存器，写入立即生效，无须等待。只有 IWDG_CNT 的重置会延时生效。

19.2 寄存器

19.2.1 地址分配

IWDG 基地址为 0x40010C00

表 19-1 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

19.2.2 IWDG_PSW 独立看门狗密码寄存器

地址:0x4001_0C00

复位值:0x0

表 19-2 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

19.2.3 IWDG_CFG 独立看门狗配置寄存器

地址:0x4001_0C04

复位值:0x1

表 19-3 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。

19.2.4 IWDG_CLR 独立看门狗清零寄存器

地址:0x4001_0C08

复位值:0x0

表 19-4 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。

19.2.5 IWDG_WTH 独立看门狗定时唤醒门限寄存器

地址:0x4001_0C0C

复位值:0x001F_F000

表 19-5 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 64kHz 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。

19.2.6 IWDG_RTH 独立看门狗超时复位门限寄存器

地址:0x4001_0C10

复位值:0x001F_F000

表 19-6 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 64kHz 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。

19.2.7 IWDG_CNT 独立看门狗当前计数值寄存器

地址:0x4001_0C14

复位值:0x0000_0000

表 19-7 IWDG_CNT 独立看门狗当前计数值寄存器

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

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

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

20 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。

20.1 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。

20.2 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。

20.3 低功耗模式

表 20-1 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	

20.4 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。

*建议参考官方休眠例程配置，休眠前需要先关闭所有数字模块的时钟，并关闭模拟 ADC/OPA/CMP/DAC 等模块。

20.4.1 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

20.4.2 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。

20.5 Deep Sleep Mode 深度休眠模式

深度休眠模式关闭所有系统高速时钟，包括 PLL/HSI。64kHz 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。

20.5.1 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。

20.5.2 模式退出

IO 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。

20.6 寄存器

20.6.1 地址分配

PMU 基地址为 0x40010C20

表 20-2 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

20.6.2 AON_PWR_CFG 功耗管理配置寄存器

地址:0x4001_0C20

复位值:0x0

表 20-3 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。

20.6.3 AON_EVT_RCD 事件记录寄存器

地址:0x4001_0C24

复位值:0x0

表 20-4 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	IO_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]	IO_WK	IO 唤醒记录，高表示 Deep Sleep 休眠被 IO 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。

20.6.4 AON_IO_WAKE_POL IO 唤醒极性寄存器

地址:0x4001_0C28

复位值:0x0

表 20-5 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。

20.6.5 AON_IO_WAKE_EN IO 唤醒使能寄存器

地址:0x4001_0C2C

复位值:0x0

表 20-6 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

	P1_9_EN	P1_8_EN	P1_5_EN	P0_10_EN	P0_9_EN	P0_8_EN	P0_2_EN	P0_0_EN
	RW	RW	RW	RW	RW	RW	RW	RW
	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:8]		未使用
[7]	P1_9_EN	P1[9]外部唤醒触发使能
[6]	P1_8_EN	P1[8]外部唤醒触发使能
[5]	P1_5_EN	P1[5]外部唤醒触发使能
[4]	P0_10_EN	P1[2]外部唤醒触发使能
[3]	P0_9_EN	P0[9]外部唤醒触发使能
[2]	P0_8_EN	P0[8]外部唤醒触发使能
[1]	P0_2_EN	P0[2]外部唤醒触发使能
[0]	P0_0_EN	P0[0]外部唤醒触发使能

21 版本历史

表 21-1 文档版本历史

时间	版本号	说明
2024.04.03	1.53	DAC 增加关于软件校正的说明
2024.03.25	1.52	增加 C 版本 DAC 说明
2024.03.13	1.51	优化 SYS_AFE_INFO 寄存器说明
2024.01.26	1.50	更正 MCPWM FAIL 信号处理的滤波系数部分说明
2024.01.15	1.49	增加关于休眠的说明
2024.01.10	1.48	UTIMER_CFG 寄存器 SRC1 位信息更正 ADC_DATx 信息更正
2023.09.25	1.47	增加关于 IWDG 喂狗跨时钟同步问题的说明 修正 MCPWM_FAIL3 寄存器说明
2023.08.30	1.46	增加版本号说明
2023.08.15	1.45	GPIO 图增加 ESD 二极管
2023.04.12	1.44	修改 OPA 复用的说明 增加 ADC 采样时间的说明
2023.02.11	1.43	增加关于关闭 PLL 等操作的说明
2022.12.18	1.42	去除 NVR 相关描述
2022.11.28	1.41	更新 LRC 时钟频率
2022.11.24	1.4	增加使用 SYS_SFT_RST 软复位 ADC 模块的说明
2022.11.12	1.39	更新 LRC 时钟频率和全温度范围偏差 修改 SYS_AFE_INFO.Version 描述
2022.11.07	1.38	增加 IO 与内部模拟电路间连接电阻描述
2022.10.28	1.37	增加 SYS_AFE_INFO.Version 描述 修改 MCPWM_DTH0/MCPWM_DTH1 寄存器描述
2022.10.25	1.36	修改 DSP_DIS/DSP_RAD 读写权限，修改供电电压范围
2022.10.24	1.35	增加 TIMER SRC0/1 采样同一个 IO 的描述
2022.10.14	1.34	修改 LRC 时钟频率和全温度范围偏差
2022.09.29	1.33	修改 SYS_OPA_SEL.OPA_SEL_EN 描述，增加写入功能
2022.09.15	1.32	增加 ADC/OPA/CMP 配置流程
2022.07.29	1.31	简化 SYS_AFE_REG 寄存器部分描述
2021.12.29	1.3	新增 SYS_FLSE 和 SYS_FLSP 寄存器描述
2021.06.23	1.0	正式版
2021.05.08	0.4	修改 ADC_CFG.FSM_RS 描述，去掉读取功能 修改 SYS_DBG_CFG
2021.03.18	0.3	修改 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。