南京凌鸥创芯电子有限公司

LKS32MC45x User Manual

目录

表格目录

图片目录

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。

RS	读/写 1，写 0 无效

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。

CRAM：Code RAM

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 32kHZ RC 时钟

HSI：High Speed Internal Clock，即 12MHz RC 时钟

HSE：High Speed External Clock，即 10~24MHz 外部晶振时钟

PLL：Phase Lock Loop Clock，即 192MHz 锁相环时钟，通常用作系统高速时钟

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

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

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

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

CW：Clock wise，顺时针

CCW：Counter clock wise，逆时针

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

2 系统概况

2.1 简述

LKS32MC45x 系列 MCU 是双电机驱动的高端 MCU，配备 192MHz ARM Cortex-M4F 内核，以及丰富的模拟和外设资源，广泛适用于各种对算力有较高需求的电机控制应用。

2.2 特性

- 192MHz 32 位 Cortex-M4F 内核
- 8 通道 DMA
- 6uA 低功耗休眠模式
- -40~105℃工业级工作温度范围
- 2.2V~3.6V 单电源供电，内部集成数字供电 LDO，部分型号支持 5V 供电
- 部分 IO 兼容 5V 电平
- 超强抗静电和群脉冲能力

2.2.1 存储

- 128/256kB Flash，数据防盗功能
- 40kB RAM

2.2.2 时钟

- 内置 12MHz 高精度 RC 时钟，全温度范围精度±1%
- 内置 32kHz 低速时钟，供低功耗模式使用
- 内部 PLL 可提供最高 192MHz 时钟

2.2.3 外设

- 3 路 UART
- 2 路 SPI
- 2 路 IIC

- 3 路通用 16 位 Timer ，2 路通用 32 位 Timer，支持捕捉和边沿对齐 PWM
- 2 个电机控制专用 PWM 模块，各支持 4 对/8 路 PWM 输出，独立死区控制
- 2 路 Hall 信号专用接口，支持测速、去抖
- 硬件独立看门狗+硬件窗口看门狗

2.2.4 模拟模块

- 集成 3 路 14bit SAR ADC，2Msps 采样及转换速率，每路 ADC 有 16 个信号通道，且正端信号通道和负端信号通道可自由配置。
- 集成 6 路 OPA，可设置为差分 PGA 模式
- 集成 6 路比较器
- 集成 2 路 12bit DAC 数模转换器，作为内部比较器输入
- 内置 1.2V 0.5%精度电压基准源
- 内置 1 路低功耗 LDO 和电源监测电路
- 集成高精度、低温飘高频 RC 时钟

2.3 系统框图

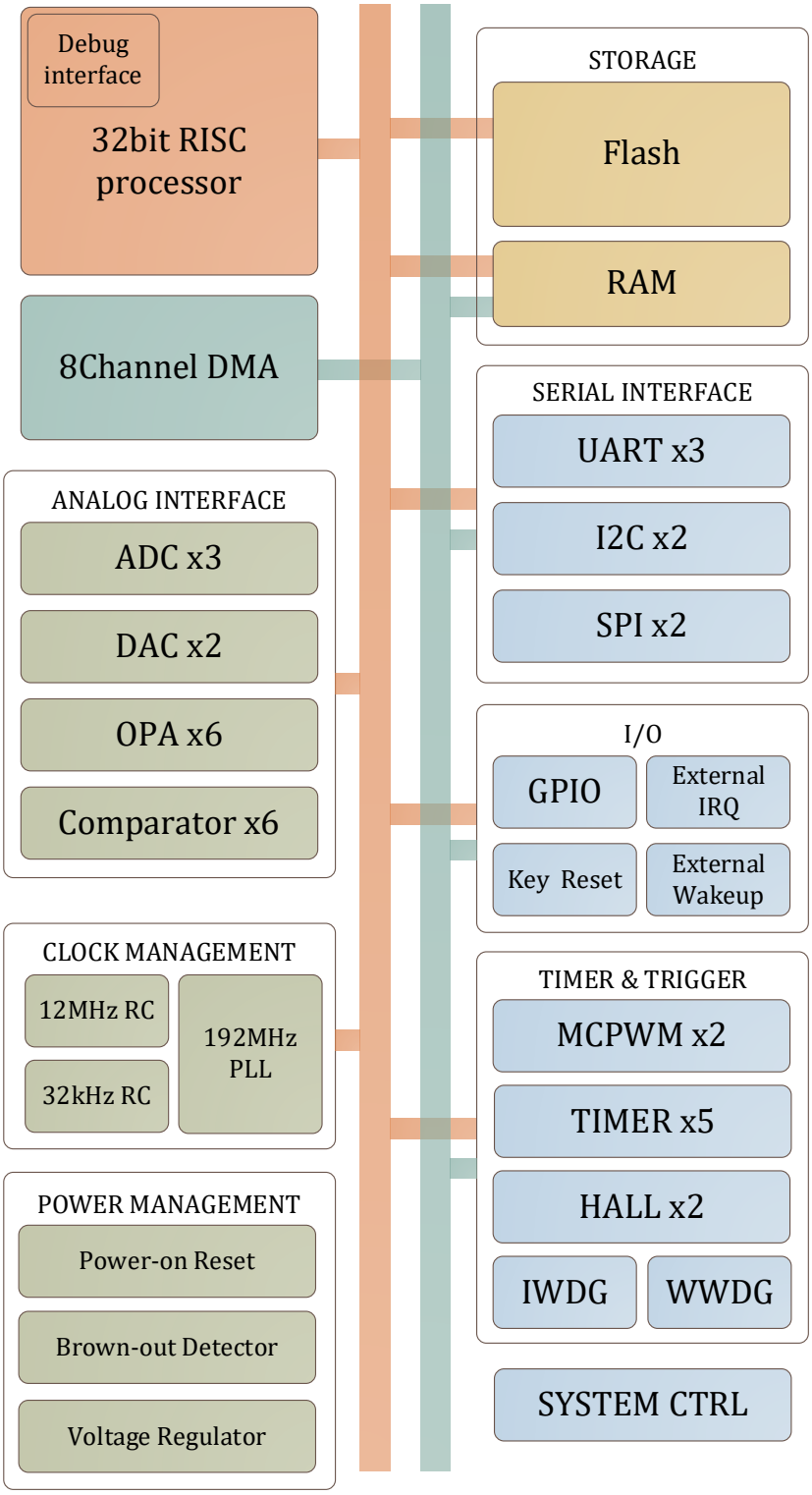


图 2-1 LKS32MC45x 系统框图

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。

由于 RAM 的读取速度远高于 FLASH，将程序从 FLASH 空间拷贝到 CRAM 空间可以加速程序的取指和执行。对于某些对于 Data RAM 需求较小的应用，可以将 SRAM 用作 CRAM，用于存储需要加速运行的程序片段。

通过 SYS_REMAP 寄存器共可以配置 4 种地址空间，主要用于将 SRAM 映射到 0x0000_0000~0x1FFF_FFFF 以扩大 Code RAM 空间，以便程序加载到 RAM 中执行提高关键代码运行效率。

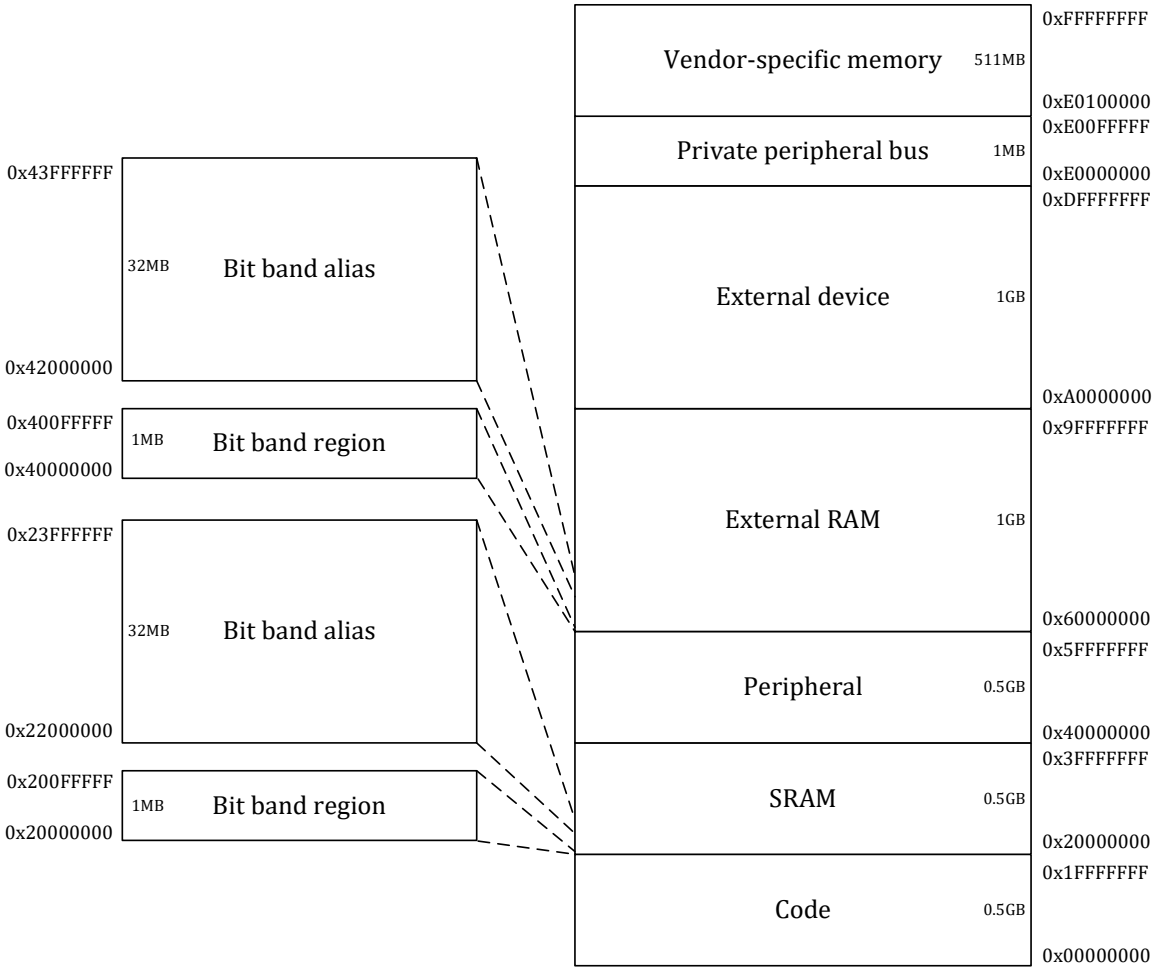


图 3-1 RISC CPU Memory Model

3.1 CODE 段

地址空间范围：0x0000_0000~0x1FFF_FFFF，通过 RISC CPU 的 I/D Bus 出口访问。

CODE 段主要包含片内 Flash(On-chip Flash)、片外 Flash(Ext-Flash)以及 CRAM(Code RAM)。

表 3-1 存储区地址空间

起止地址		空间大小
0x0000_0000	On-chip Flash	256kB
0x0003_FFFF		
0x0100_0000	Ext-Flash	最大支持 16MB
0x01FF_FFFF		
0x0200_0000	保留	
0x0FFF_FFFF		
0x1000_0000	CRAM	最大支持 24kB
0x1000_7FFF		

Ext-Flash 最大支持 16MB，实际颗粒容量可以小于 16MB。

但是 On-chip Flash 和 Ext-Flash 的烧写算法是不同的，因此烧写算法中要根据地址选择不同的烧写方法。

CRAM 最大支持 24kB。芯片默认没有 CRAM，需要修改寄存器 SYS_REMAP 来切换地址空间映射，将 SRAM 作为 CRAM 使用。在加载文件中，需要放入 CRAM 中执行的函数需要特殊指明，利用分散加载机制，将这部分 Code 从加载域拷贝到执行域。

需要注意的是，由于芯片默认没有 CRAM，因此在 SystemInit()函数中需要修改 SYS_REMAP 来切换地址空间映射,之后__main()会在进入 main()函数之前完成 RW/ZI 段的数据初始化以及从 flash 中拷贝程序到 CRAM 中。

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


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


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

图 3-2 Reset_Handler 实现

3.2 RAM 段

地址空间范围：0x2000_0000~0x3FFF_FFFF，通过 RISC CPU 的 S Bus 出口访问。

SRAM 共分为 4 块。

表 3-2 SRAM 的大小和用途

	大小	用途
SRAM0	16kB	Data RAM
SRAM1	8kB	Data RAM/Code RAM
SRAM2	8kB	Data RAM/Code RAM
SRAM3	8kB	Data RAM/Code RAM

表 3-3 SRAM 在不同地址空间配置下的用途

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

3.3 PERIPHERAL 段

地址空间范围：0x4000_0000~0x5FFF_FFFF，通过 RISC CPU 的 S Bus 出口访问，是所有外设所在地址空间。

3.4 地址空间 0 (SYS_REMAP=0)

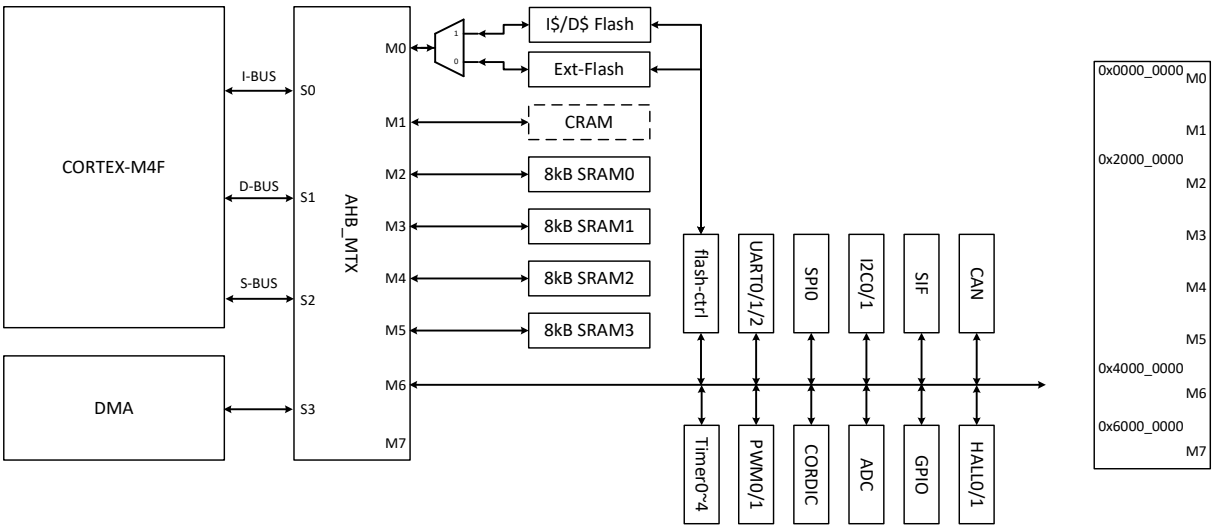


图 3-3 总线架构(REMAP=0)

默认的总线架构，复位后为此种配置，所有 SRAM 都是 Data RAM，无 Code RAM 用于加速。

表 3-4 REMAP=0 时系统地址空间分配

类型	外设	开始地址	结束地址	工作时钟/软复位	空间大小
CODE	FLASH	0x0000_0000	0x0003_FFFF	同总线/无	256kB
	EXT-FLASH	0x0100_0000	0x01FF_FFFF	同总线/无	16MB
RAM	SRAM0	0x2000_0000	0x2000_3FFF	同总线/无	16kB
	SRAM1	0x2000_4000	0x2000_5FFF	同总线/无	8kB
	SRAM2	0x2000_6000	0x2000_7FFF	同总线/无	8kB
	SRAM3	0x2000_8000	0x2000_9FFF	同总线/无	8kB
	保留	0x2000_A000	0x21FF_FFFF		
	Bit band alias	0x2200_0000	0x23FF_FFFF		32MB
	保留	0x2400_0000	0x3FFF_FFFF		
PERIP	保留	0x4000_0000	0x4000_03FF		1kB
	保留	0x4000_0400	0x4000_07FF		1kB
	保留	0x4000_0800	0x4000_0BFF		1kB
	保留	0x4000_0C00	0x4000_0FFF		1kB
	保留	0x4001_0000	0x4001_03FF		1kB
	保留	0x4001_0400	0x4001_07FF		1kB
	保留	0x4001_0800	0x4001_0BFF		1kB
	保留	0x4001_0C00	0x4001_0FFF		1kB
	I2C0	0x4001_1000	0x4001_13FF	外设门控时钟[0]/软复位[0]	1kB
	I2C1	0x4001_1400	0x4001_17FF	外设门控时钟[1]/软复位[1]	1kB
	HALL0	0x4001_1800	0x4001_1BFF	外设门控时钟[2]/软复位[2]	1kB
	HALL1	0x4001_1C00	0x4001_1FFF	外设门控时钟[3]/软复位[3]	1kB

	UART0	0x4001_2000	0x4001_23FF	外设门控时钟[4]/软复位[4]	1kB
	UART1	0x4001_2400	0x4001_27FF	外设门控时钟[5]/软复位[5]	1kB
	UART2	0x4001_2800	0x4001_2BFF	外设门控时钟[6]/软复位[6]	1kB
	保留	0x4001_2C00	0x4001_2FFF		1kB
	CMP	0x4001_3000	0x4001_33FF	外设门控时钟[8]/软复位[8]	1kB
	MCPWM0	0x4001_3400	0x4001_37FF	外设门控时钟[9]/软复位[9]	1kB
	MCPWM1	0x4001_3800	0x4001_3BFF	外设门控时钟[10]/软复位[10]	1kB
	保留	0x4001_3C00	0x4001_3FFF		1kB
	TIMER0	0x4001_4000	0x4001_40FF	外设门控时钟[12]/软复位[12]	256B
	TIMER1	0x4001_4100	0x4001_41FF	外设门控时钟[13]/软复位[13]	256B
	TIMER2	0x4001_4200	0x4001_42FF	外设门控时钟[14]/软复位[14]	256B
	TIMER3	0x4001_4300	0x4001_43FF	外设门控时钟[15]/软复位[15]	256B
	TIMER4	0x4001_4400	0x4001_44FF	外设门控时钟[16]/软复位[16]	256B
	QEP0	0x4001_4500	0x4001_45FF	外设门控时钟[17]/软复位[17]	256B
	QEP1	0x4001_4600	0x4001_46FF	外设门控时钟[18]/软复位[18]	256B
	QEP2	0x4001_4700	0x4001_47FF	外设门控时钟[19]/软复位[19]	256B
	QEP3	0x4001_4800	0x4001_48FF	外设门控时钟[20]/软复位[20]	256B
	保留	0x4001_4900	0x4001_63FF		
	GPIO	0x4001_6400	0x4001_67FF	外设门控时钟[22]/软复位[22]	1kB
	CRC	0x4001_6800	0x4001_6BFF	外设门控时钟[23]/软复位[23]	1kB
	CORDIC	0x4001_6C00	0x4001_6FFF	外设门控时钟[24]/软复位[24]	1kB
	FMAC	0x4001_7000	0x4001_77FF	外设门控时钟[25]/软复位[25]	2kB
	ADC0	0x4001_7800	0x4001_7BFF	ACLK/软复位[26]	1kB
	ADC1	0x4001_7C00	0x4001_7FFF	ACLK/软复位[27]	1kB
	ADC2	0x4001_8000	0x4001_83FF	ACLK/软复位[28]	1kB
	保留	0x4001_8400	0x4001_87FF		1kB
	CAN	0x4001_8800	0x4001_8BFF	同总线/软复位[29]	1kB
	保留	0x4001_8C00	0x4001_8FFF		1kB
	AFE	0x4001_9000	0x4001_93FF		1kB
	保留	0x4001_9400	0x4001_97FF		1kB
	SYS	0x4001_9800	0x4001_9BFF	同总线/无	1kB
	DMA	0x4001_9C00	0x4001_9FFF	同总线/无	1kB
	FLSCR	0x4001_A000	0x4001_A3FF	同总线/无	1kB
	保留	0x4001_A400	0x4001_A7FF		1kB
	SPI0	0x4001_A800	0x4001_ABFF	外设门控时钟[30]/软复位[30]	1kB
	SPI1	0x4001_AC00	0x4001_AFFF	外设门控时钟[31]/软复位[31]	1kB
	EXT_FLSCR	0x4001_B000	0x4001_B3FF		1kB
	WWDG	0x4001_B400	0x4001_B7FF		1kB
always on	BRAM	0x4001_B800	0x4001_B87F		128B
	AONCR	0x4001_B880	0x4001_B8BF		64B
	IWDG	0x4001_B8C0	0x4001_B8FF		64B
	保留				

	Bit band alias	0x4200_0000	0x43FF_FFFF		32MB

3.5 地址空间 1 (SYS_REMAP=1)

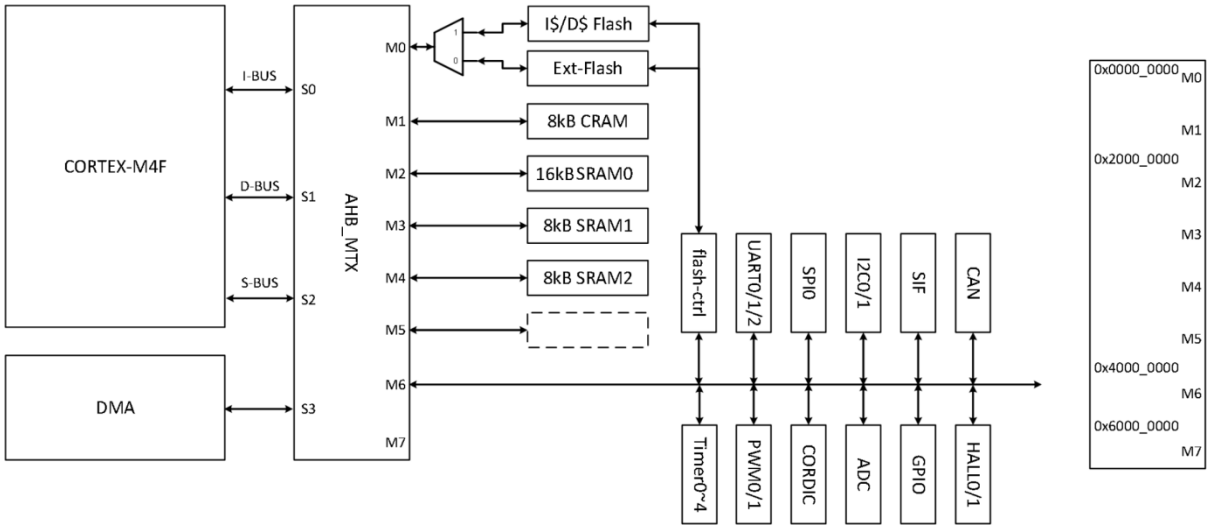


图 3-4 总线架构(REMAP=1)

将 8kB SRAM3 映射为 CRAM 使用，其余 SRAM0/1/2 仍用作 Data RAM，CRAM 共 8kB，Data RAM 共 32kB。

表 3-5 REMAP=1 时系统地址空间分配

类型	外设	开始地址	结束地址	工作时钟/软复位	空间大小
CODE	FLASH	0x0000_0000	0x0003_FFFF	同总线/无	256kB
	EXT-FLASH	0x0100_0000	0x01FF_FFFF	同总线/无	16MB
RAM	SRAM0	0x2000_0000	0x2000_3FFF	同总线/无	16kB
	SRAM1	0x2000_4000	0x2000_5FFF	同总线/无	8kB
	SRAM2	0x2000_6000	0x2000_7FFF	同总线/无	8kB
	SRAM3	0x1000_8000	0x1000_9FFF	同总线/无	8kB
	保留	0x2000_A000	0x21FF_FFFF		
	Bit band alias	0x2200_0000	0x23FF_FFFF		32MB
	保留	0x2400_0000	0x3FFF_FFFF		
PERIP	与 REMAP=0 时相同				

3.6 地址空间 2 (SYS_REMAP=2)

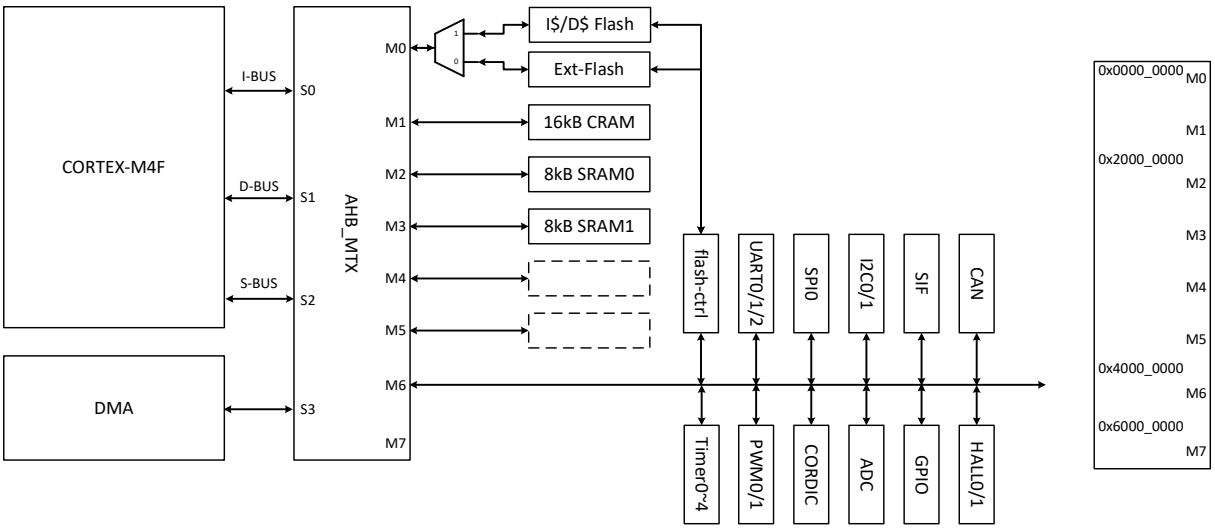


图 3-5 总线架构(REMAP=2)

将 8kB 的 SRAM3/SRAM2 分别映射为 CRAM0/1 使用，其余 SRAM0/1 仍用作 Data RAM。CRAM 共 16kB，Data RAM 共 16kB。

表 3-6 REMAP=2 时系统地址空间分配

类型	外设	开始地址	结束地址	工作时钟/软复位	空间大小
CODE	FLASH	0x0000_0000	0x0003_FFFF	同总线/无	256kB
	EXT-FLASH	0x0100_0000	0x01FF_FFFF	同总线/无	16MB
RAM	SRAM0	0x2000_0000	0x2000_3FFF	同总线/无	16kB
	SRAM1	0x2000_4000	0x2000_5FFF	同总线/无	8kB
	SRAM2	0x1000_6000	0x1000_7FFF	同总线/无	8kB
	SRAM3	0x1000_8000	0x1000_9FFF	同总线/无	8kB
	保留	0x2000_A000	0x21FF_FFFF		
	Bit band alias	0x2200_0000	0x23FF_FFFF		32MB
	保留	0x2400_0000	0x3FFF_FFFF		
PERIP	与 REMAP=0 时相同				

3.7 地址空间 3 (SYS_REMAP=3)

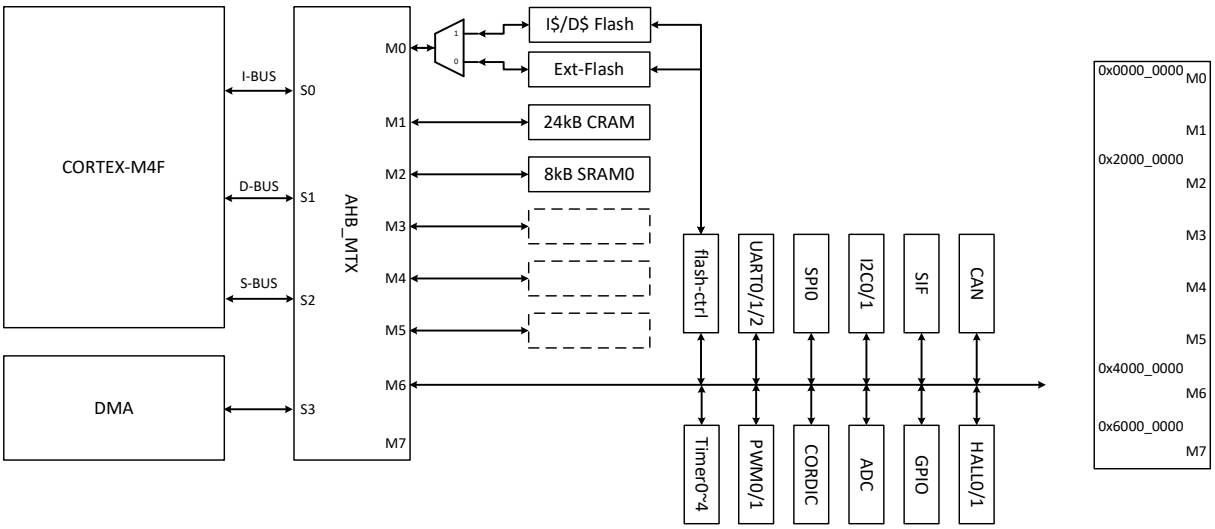


图 3-6 总线架构(REMAP=3)

将 8kB 的 SRAM3/2/1 分别映射为 CRAM0/1/2 使用，其余 SRAM0 仍用作 Data RAM。CRAM 共 24kB，Data RAM 共 8kB。

表 3-7 REMAP=1 时系统地址空间分配

类型	外设	开始地址	结束地址	工作时钟/软复位	空间大小
CODE	FLASH	0x0000_0000	0x0003_FFFF	同总线/无	256kB
	EXT-FLASH	0x0100_0000	0x01FF_FFFF	同总线/无	16MB
RAM	SRAM0	0x2000_0000	0x2000_3FFF	同总线/无	16kB
	SRAM1	0x1000_4000	0x1000_5FFF	同总线/无	8kB
	SRAM2	0x1000_6000	0x1000_7FFF	同总线/无	8kB
	SRAM3	0x1000_8000	0x1000_9FFF	同总线/无	8kB
	保留	0x2000_A000	0x21FF_FFFF		
	Bit band alias	0x2200_0000	0x23FF_FFFF		32MB
	保留	0x2400_0000	0x3FFF_FFFF		
PERIP	与 REMAP=0 时相同				

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。

除系统中断以外，LKS32MC45x 系列芯片共有 38 个独立的中断源及中断向量。最多支持 8bit 共 256 个中断优先级(0~255)可供编程选择。中断优先级数值越大表示优先级越低，对于非系统中断，0 代表最高优先级。NMI 中断为不可屏蔽中断，对应 SRAM 校验错事件，当 SYS_MEM_CFG.PCK_EN 为 1 且 SYS_MEM_CFG.PCK_ERR 为 1 时发生不可屏蔽中断。

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	I2C0	20	TIMER3
1	I2C1	21	TIMER4
2	HALL0	22	QEP0
3	HALL1	23	QEP1
4	UART0	24	QEP2
5	UART1	25	QEP3
6	UART2	26	GPIO(EXTI)
7	CMP0	27	ADC0
8	CMP1	28	ADC1
9	CMP2	29	ADC2
10	CMP3	30	CAN
11	CMP4	31	SPI0
12	CMP5	32	SPI1
13	MCPWM0_IRQ0	33	WWDG
14	MCPWM0_IRQ1	34	DMA
15	MCPWM1_IRQ0	35	WAKEUP
16	MCPWM1_IRQ1	36	PWRDN(电源电压过低)

17	TIMER0	37	FMAC
18	TIMER1	38	SW
19	TIMER2		

5 模拟电路

5.1 简述

模拟电路包含以下模块：

- 集成 3 路 14BIT SAR ADC，采样率 2MHz，每个 ADC 最多 16 通道信号可选，且正端信号通道和负端信号通道可自由配置。
- 集成 6 路运算放大器，可设置为 PGA 模式
- 集成 6 路比较器，可设置迟滞模式
- 集成 2 路 12BIT 数模转换器
- 内置±2℃温度传感器
- 内置高精度基准源

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

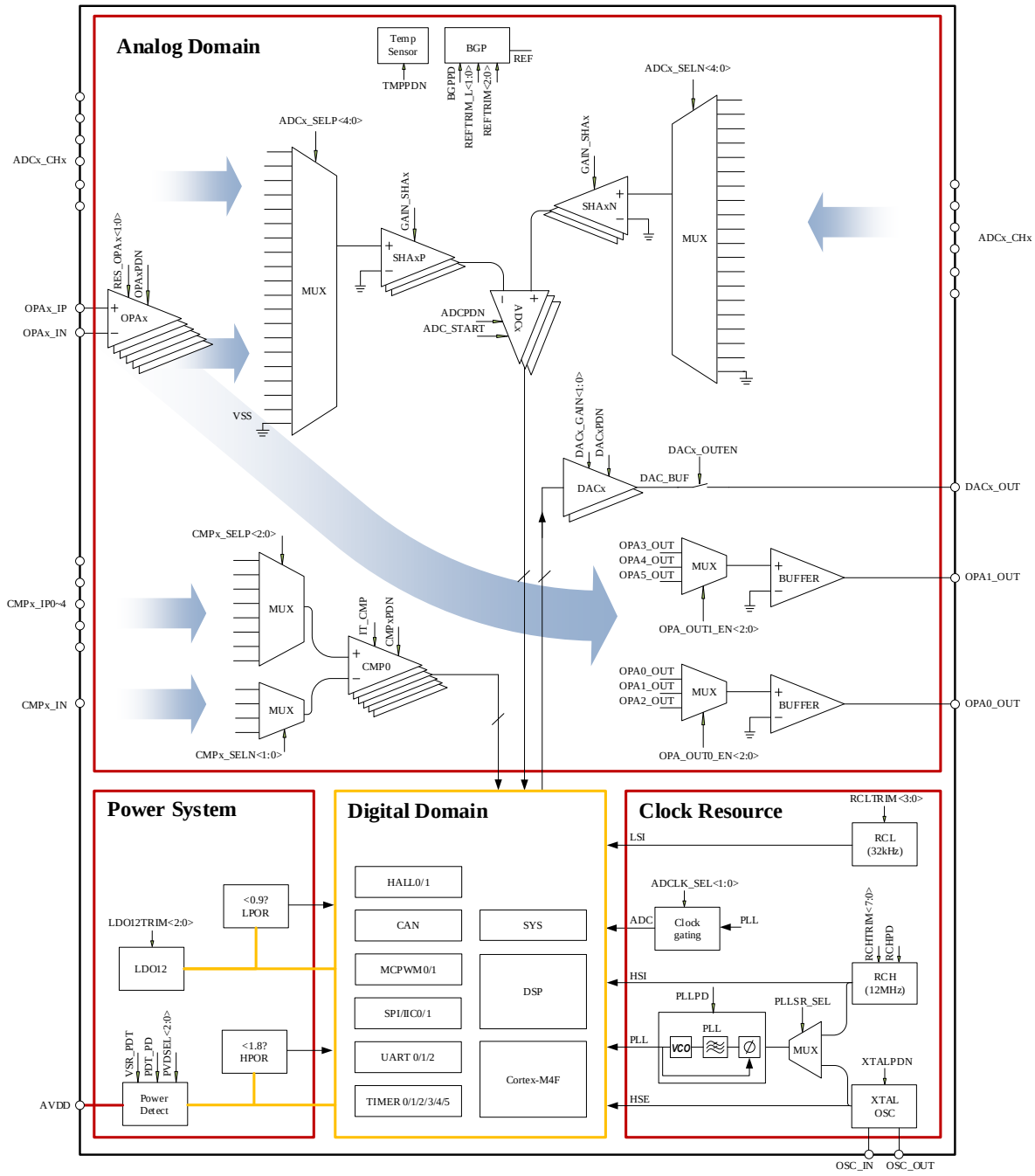


图 5-1 模拟电路功能框图

5.1.1 电源管理系统

电源管理系统由 LDO12 模块、电源检测模块（PVD）、上电/掉电复位模块（POR）组成。

该芯片由 3.3V 单电源供电，以节省芯片外的电源成本。芯片内部集成一路 LDO12 给内部所有数字电路、PLL 模块提供 1.2V 供电。

LDO 上电后自动开启，无需软件配置，但 LDO 输出电压可通过软件实现微调。

LPOR 模块监测 LDO12 的电压，在 LDO12 电压低于 0.9V 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。

HPOR 模块监测 AVDD 的电压，在 AVDD 电压低于 1.8V 时（例如上电之初，或者掉电时），为数字电路提供复位信号以避免数字电路工作产生异常。

PVD 模块对 3.3V 输入电源进行检测，如低于某一设定阈值，则产生报警（中断）信号以提醒 MCU。中断提醒阈值可通过寄存器 PVDSEL[1:0]设置为不同的电压。PVD 模块可通过设置 PDT_PD=1 关闭。

PVDSEL[1:0]/ PD_PDT 的说明见 SYS_AFE_REGB 模拟配置寄存器 B

5.1.2 时钟系统

时钟系统包括内部 32kHz RC 时钟、内部 12MHz RC 时钟、PLL 电路组成。

32kHz RC 时钟 LSI 主要用于系统内的看门狗模块以及复位信号滤波等。12MHz RC 时钟可作为 MCU 主时钟使用，MCU 也可以使用 PLL 时钟，PLL 最高可提供 192MHz 的时钟。

32kHz 和 12MHz RC 时钟均带有出厂校正，32kHz RC 时钟在-40~105℃范围内的精度为±50%，12MHz RC 时钟在该温度范围的精度为±1%。

芯片出厂前时钟已经过校正。

12MHz RC 时钟通过设置 RCHPD='0'打开（默认打开，写 1 关闭），RC 时钟需要 BGP 电压基准源模块提供基准电压和电流，因此开启 RC 时钟需要先开启 BGP 模块（BGPPD='0'）。芯片上电的默认状态下，12MHz RC 时钟和 BGP 模块都是开启的。32kHz RC 时钟始终开启，不能关闭。

PLL 对 12MHz 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 192MHz，ADC 模块最高时钟 32MHz，通过寄存器 ADCCLKSEL[1:0]可设置不同的 ADC 工作频率。

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

ADCCLKSEL<1:0>的说明见 SYS_AFE_REG3 模拟配置寄存器 3

BGPPD/RCHPD/PLLPDN 的说明见 SYS_AFE_REGA 模拟配置寄存器 A

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%

芯片出厂前基准源已经过校正。

5.1.4 ADC 模块

请参考第 13 章。

5.1.5 运算放大器

芯片集成 6 路输入输出轨到轨（rail-to-rail）运算放大器，内置反馈电阻，外部引脚上需串联一个电阻 R_0 到信号源。反馈电阻 R_2:R_1 的阻值可通过寄存器 RES_OPAx[1:0]设置，以实现不同的放大倍数。

RES_OPAx<1:0>的说明见 SYS_AFE_REG0 模拟配置寄存器 0 及 SYS_AFE_REG1 模拟配置寄存器 1

放大器的结构示意图如下所示：

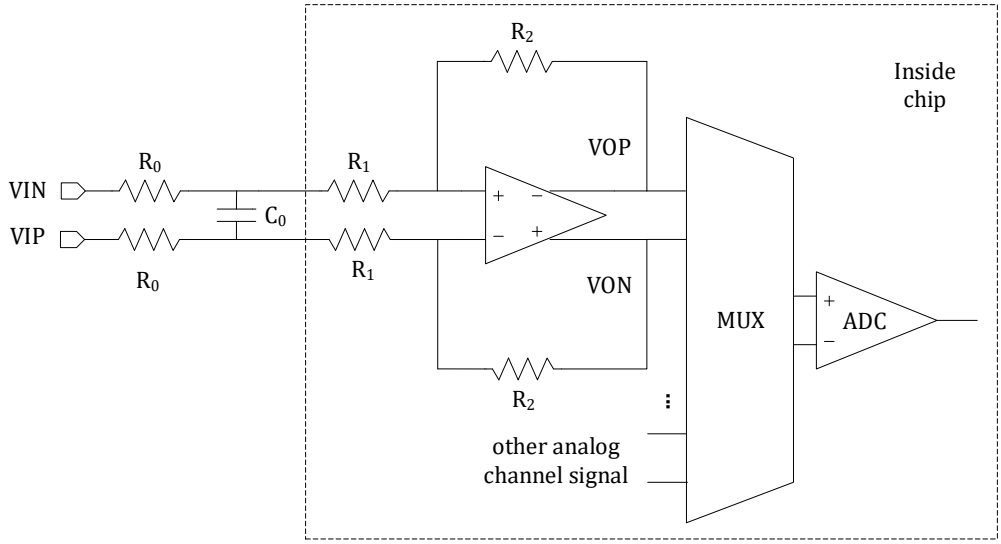


图 5-2 放大器框图

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

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

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

OPA0/1/2 可通过设置 OPAOUT0_EN<1:0>选择将 3 路放大器中的一路输出信号通过 BUFFER 送至 P3.11 管脚口进行测量和应用，OPA3/4/5 可通过设置 OPAOUT1_EN<1:0>选择将 3 路放大器中的一路输出信号通过 BUFFER 送至 P4.6 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。

OPAOUT0_EN<1:0>的说明见 SYS_AFE_REG2 模拟配置寄存器 2

OPAOUT1_EN<1:0>的说明见 SYS_AFE_REG2 模拟配置寄存器 2

芯片上电的默认状态下，放大器模块是关闭的。放大器可通过设置 OPAxPDN=1 (x=0,1,2,3,4,5)打开。开启放大器之前，需要先开启 BGP 模块。

OPAxPDN 的说明见 SYS_AFE_REG9 模拟配置寄存器 9

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

5.1.6 比较器

内置 6 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 IT_CMP 设置为 150nS/600nS。迟滞电压通过 CMP_HYS 设置为 20mV/40mV。

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=0/1/2/3/4/5，代表比较器 0/1/2/3/4/5）。

需说明的是，两个比较器负输入端的 HALLx_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，具体连接方式见下图 5-2。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 HALLx_MID 之后才会导通，否则开关都处于断开状态。当 CMPx_IP1/ CMPx_IP2/ CMPx_IP3 管脚连的是 HALL 信号时，通过将 HALL 信号与 HALLx_MID 信号进行比较，可快速得到 HALL 信号的状态。

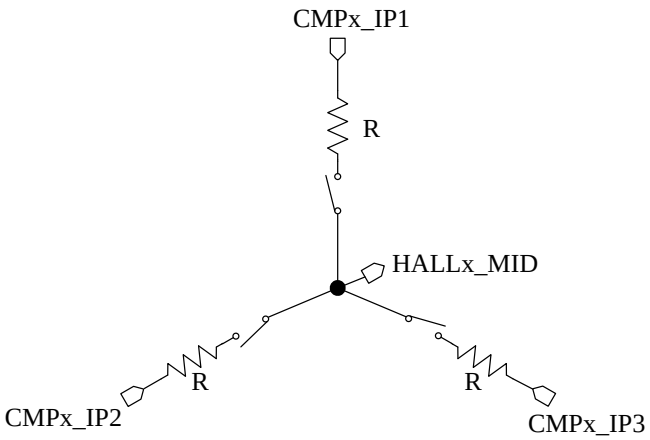


图 5-2 HALLx_MID 信号

比较器输出结果，可以通过 CMP_DATA 寄存器读出。

IT_CMP<1:0>的说明见 SYS_AFE_REG2 模拟配置寄存器 2

CMP_HYS 的说明见 SYS_AFE_REG4 模拟配置寄存器 4

CMPx_SELN<1:0>的说明见 SYS_AFE_REG4 模拟配置寄存器 4 和 SYS_AFE_REG5 模拟配置寄存器 5

CMPx_SELP<2:0>的说明见 SYS_AFE_REG5 模拟配置寄存器 5 和 SYS_AFE_REG6 模拟配置寄存器 6

比较器的原始输出状态值见 CMP_DATA 输出数据寄存器

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=0,1,2,3,4,5)打开，开启比较器之前，需要先开启 BGP 模块。

CMPxPDN 的说明见 SYS_AFE_REG9 模拟配置寄存器 9

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 ADC2 的通道 14。

ADC 部分的设置参考 ADC 章节

TMPPDN 的说明见 SYS_AFE_REGA 模拟配置寄存器 A

温度传感器的典型曲线如下图所示：

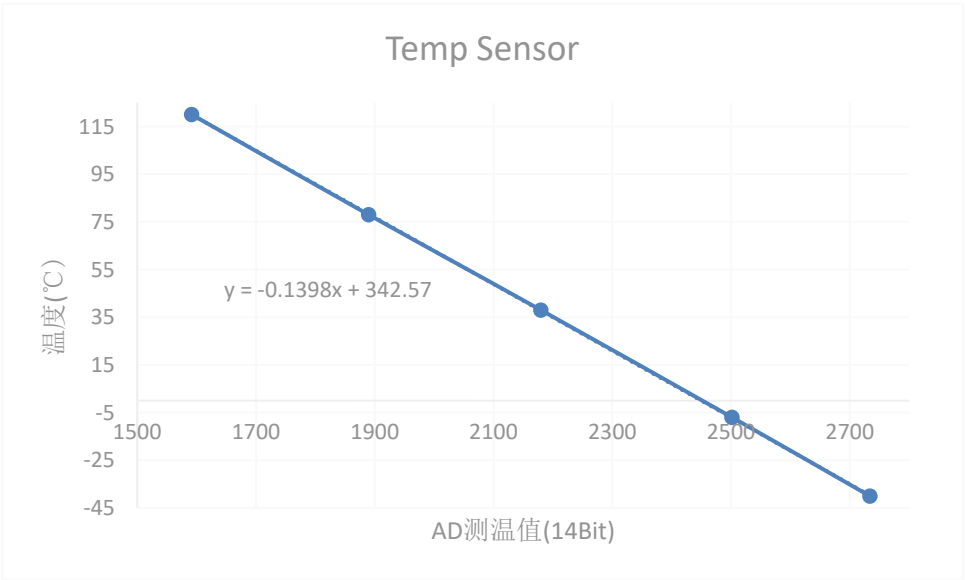


图 5-4 温度传感器曲线

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=-0.1398x+342.57

求得的 Y 值即为此时的温度。

公式中有两个系数，a=-0.1398, b=342.57。对于不同的芯片，b 系数的值是不一样的。芯片出厂前会经过温度标定，将每颗芯片所对应的系数 b 写入 flash 的 info 区，地址为 0x00000944。存储时，会将 b 系数小数点右移一位（乘 10）存入 info 区，小数点后第二位不进行保存。

同时为方便客户操作，系数 a 也会存入 flash info 区，地址为 0x00000940。存储时，将 a 系数小数点右移四位（乘 10000）存入 info 区。

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

注意，上述计算公式，基于 ADC 右对齐实现。若换成左对齐，ADC 采样值需右移 2 位后，才能代入上述公式。

5.1.8 DAC 模块

芯片内置 2 路 12bit DAC，输出信号的最大量程可通过寄存器 DACx_GAIN<1:0>设置为 1.2V/3V，x=0,1 代表 DAC0/1。

DAC0 可通过配置寄存器 DAC0_OUTEN=1，将 DAC 输出送至 P3.4 管脚，DAC1 可通过配置寄存器 DAC1_OUTEN=1，将 DAC 输出送至 P4.7 管脚，可驱动>5kΩ 的对地负载电阻和 50pF 的负载电容。但只能输出电流，不能输入灌电流。

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 DACxPDN =1 打开，开启 DAC 模块之前，需要先开启 BGP 模块。

DAC 的输入数字信号寄存器为 SYS_AFE_DACx，低 12BIT 有效。信号范围是 0x000~0xFFF。0x000 对应零模拟量输出 0V，0xFFF 对应满量程模拟量输出为DAC_{fs}，如上文所述，DAC_{fs}的值可由 DAC12B_FS 寄存器进行设置。每一档信号(LSB)所对应的模拟信号幅度为$\frac{DAC_{fs}}{4096}$。若 SYS_AFE_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$\frac{DAC_{fs}}{4096} * Din$。

不同芯片，DAC 存在制造偏差，为抵消偏差，DAC 自带校准硬件模块。DAC 输出遵循公式 y=ax+b。x 为 SYS_AFE_DACx 填入值（理想值对应数字量）。a 来自 SYS_AFE_DACx_AMC 寄存器，b 来自 SYS_AFE_DACx_DC。硬件根据 SYS_AFE_DACx、SYS_AFE_DACx_AMC 和 SYS_AFE_DACx_DC 进行乘加从而求得校正后的数字量，送至 DACx 输入端，使得 DACx 最终输出的模拟值就是填入的理想数字量对应的值。系统上电后，默认加载 3V 的校准值，若换成其它量程，软件需读取 flash info 区域，重新加载到相应寄存器。

DAC0：

地址 0x00000860，为 3V 量程的 a 参数，地址 0x00000868，为 3V 量程的 b 参数。

地址 0x00000864，为 1.2V 量程的 a 参数，地址 0x0000086C，为 1.2V 量程的 b 参数。

DAC1：

地址 0x00000870，为 3V 量程的 a 参数，地址 0x00000878，为 3V 量程的 b 参数。

地址 0x00000874，为 1.2V 量程的 a 参数，地址 0x0000087C，为 1.2V 量程的 b 参数。

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

DACx_OUTEN 和 DACx_GAIN<1:0>的说明见 SYS_AFE_REG4 模拟配置寄存器 4

DACxPDN 的说明见 SYS_AFE_REGA 模拟配置寄存器 A

5.2 寄存器

5.2.1 地址分配

模拟寄存器 SYS_AFE_REG0 ~ SYS_AFE_REGB，对应地址为 0x40019810 ~ 0x4001983C。地址 0x40019820~0x4001983C 的寄存器中保留寄存器(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 0x40019800。

表 5-1 系统控制寄存器

名称	偏移	说明
SYS_AFE_INFO	0x04	芯片版本信息寄存器
SYS_AFE_DBG0	0x08	调试寄存器
SYS_AFE_REG0	0x10	模拟配置寄存器 0
SYS_AFE_REG1	0x14	模拟配置寄存器 1
SYS_AFE_REG2	0x18	模拟配置寄存器 2
SYS_AFE_REG3	0x1C	模拟配置寄存器 3
SYS_AFE_REG4	0x20	模拟配置寄存器 4
SYS_AFE_REG5	0x24	模拟配置寄存器 5
SYS_AFE_REG6	0x28	模拟配置寄存器 6
SYS_AFE_REG9	0x34	模拟配置寄存器 9
SYS_AFE_REGA	0x38	模拟配置寄存器 A
SYS_AFE_REGB	0x3C	模拟配置寄存器 B
SYS_TMP_A	0x58	温度传感器系数 A
SYS_TMP_B	0x5C	温度传感器系数 B
SYS_AFE_DAC0	0x60	DAC0 数字量寄存器
SYS_AFE_DAC0_AMC	0x64	DAC0 增益校正寄存器
SYS_AFE_DAC0_DC	0x68	DAC0 直流偏置寄存器
SYS_AFE_DAC1	0x70	DAC1 数字量寄存器
SYS_AFE_DAC1_AMC	0x74	DAC1 增益校正寄存器
SYS_AFE_DAC1_DC	0x78	DAC1 直流偏置寄存器

5.2.2 SYS_AFE_INFO 芯片版本信息寄存器

地址：0x4001_9804

复位值:根据 wafer 版本不同

表 5-2 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
								Res.				Version			
												RO			
												Depends			

位置	位名称	说明
[31:8]		未使用
[7:4]		保留
[3:0]	Version	芯片版本信息

5.2.3 SYS_AFE_DBG0 调试寄存器

地址:0x4001_9808

复位值:0x0

表 5-3 SYS_AFE_DBG0 调试寄存器

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

位置	位名称	说明
[31:8]		未使用
[13]	PWR_WEAK	供电低于掉电监测阈值时为高
[12:0]		未使用

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

掉电监测的使用请参考 5.1.1 电源管理系统。

5.2.4 SYS_AFE_REG0 模拟配置寄存器 0

地址：0x4001_9810

复位值：0x0

表 5-4 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
Res.	RES_OPA3			Res.	RES_OPA2			Res.	RES_OPA1			Res.	RES_OPA0		
RW	RW			RW	RW			RW	RW			RW	RW		
0	0			0	0			0	0			0	0		

位置	位名称	说明
[31:16]		未使用
[15]	Reserved	保留，必须为 0
[14:12]	RES_OPA3	运放 3 反馈电阻 000: 20k:10k 001: 40k:10k 010: 80k:10k 011:160k:10k 100: 320k:10k 101: 320k:5k 110/111: 320k:10k;
[11]	Reserved	保留，必须为 0
[10:8]	RES_OPA2	运放 2 反馈电阻 000: 20k:10k 001: 40k:10k 010: 80k:10k 011:160k:10k 100: 320k:10k 101: 320k:5k 110/111: 320k:10k;
[7]	Reserved	保留，必须为 0
[6:4]	RES_OPA1	运放 1 反馈电阻 000: 20k:10k 001: 40k:10k 010: 80k:10k 011:160k:10k 100: 320k:10k 101: 320k:5k 110/111: 320k:10k;
[3]	Reserved	保留，必须为 0

[2:0]	RES_OPA0	运放 0 反馈电阻 000: 20k:10k 001: 40k:10k 010: 80k:10k 011:160k:10k 100: 320k:10k 101: 320k:5k 110/111: 320k:10k;

5.2.5 SYS_AFE_REG1 模拟配置寄存器 1

地址：0x4001_9814

复位值：0x0

表 5-5 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.					XDIV	XCSEL		Res.	RES_OPA5			Res.	RES_OPA4		
RW					RW	RW		RW	RW			RW	RW		
0					0	0		0	0			0	0		

位置	位名称	说明
[31:16]		未使用
[15:11]	Reserved	保留，必须为 0
[10]	XDIV	晶体频率除以 2 0:使用 12MHz 外部晶体 1:使用 24MHz 外部晶体，同时需要将 SYS_AFE_REG4.XTRSEL[0]置为 1
[9:8]	XCSEL	晶体电容调整
[7]	Reserved	保留，必须为 0
[6:4]	RES_OPA5	运放 5 反馈电阻 000: 20k:10k 001: 40k:10k 010: 80k:10k 011:160k:10k 100: 320k:10k 101: 320k:5k 110/111: 320k:10k;
[3]	Reserved	保留，必须为 0
[2:0]	RES_OPA4	运放 4 反馈电阻 000: 20k:10k 001: 40k:10k 010: 80k:10k 011:160k:10k

		100: 320k:10k 101: 320k:5k 110/111: 320k:10k;

5.2.6 SYS_AFE_REG2 模拟配置寄存器 2

地址：0x4001_9818

复位值：0x0

表 5-6 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
LDOOUT_EN	Res.	Res.	REF2VDD	Res.	Res.	IT_CMP	CMP_FT	Res.	Res.	OPA_OUT1_EN	Res.	Res.	OPA_OUT0_EN
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:16]		未使用
[15]	LDOOUT_EN	LDO 输出到 IO 使能 1:使能 0:关闭
[14]	Reserved	保留，必须为 0
[13]	Reserved	保留，必须为 0
[12]	REF2VDD	使用外部输入电源作为 ADC REF 1: 使用外部输入电源作为 ADC REF; 0: 使用默认内部 REF 作为 ADC 基准
[11]	Reserved	保留，必须为 0
[10]	Reserved	保留，必须为 0
[9]	IT_CMP	比较器 0~5 比较速度选择 0:比较速度 150ns 1:比较速度 600ns
[8]	CMP_FT	使能比较器快速比较 1:在 IT_CMP 为默认 0 的时候，使能比较器比较速度小于 30ns 0:不使能
[7]	Reserved	保留，必须为 0
[6]	Reserved	保留，必须为 0
[5:4]	OPAOUT1_EN	使能 OPAx(x=3,4,5)输出信号送至 IO 口 00:不输出 01:输出 OPA3 信号到 IO 口 P4.6 10:输出 OPA4 信号到 IO 口 P4.6 11:输出 OPA5 信号到 IO 口 P4.6

[3]	Reserved	保留，必须为 0
[2]	Reserved	保留，必须为 0
[1:0]	OPAOUT0_EN	使能 OPAx(x=0,1,2)输出信号送至 IO 口 00:不输出 01:输出 OPA0 信号到 IO 口 P3.11 10:输出 OPA1 信号到 IO 口 P3.11 11:输出 OPA2 信号到 IO 口 P3.11

5.2.7 SYS_AFE_REG3 模拟配置寄存器 3

地址：0x4001_981C

复位值：0x0

表 5-7 模拟配置寄存器 3 SYS_AFE_REG3

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

位置	位名称	说明
[31:16]		未使用
[15]	Reserved	保留，必须为 0
[14]	Reserved	保留，必须为 0
[13]	Reserved	保留，必须为 0
[12]	Reserved	保留，必须为 0
[11:10]	ADCCLKSEL	ADC 时钟频率选择 01:禁止 00: 32MHz 11: 16MHz 10: 8MHz
[9]	Reserved	保留，必须为 0
[8]	Reserved	保留，必须为 0
[7:6]	Reserved	保留，必须为 0
[5:4]	Reserved	保留，必须为 0
[3:2]	Reserved	保留，必须为 0
[1:0]	Reserved	保留，必须为 0

5.2.8 SYS_AFE_REG4 模拟配置寄存器 4

地址：0x4001_9820

复位值：0x0

表 5-8 模拟配置寄存器 4 SYS_AFE_REG4

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

位置	位名称	说明
[31:16]		未使用
[15:14]	CMP0_SELN	比较器 0 信号负端选择 00: CMP0_IN 01: REF 10: DAC0 输出 11: CMP0_IP1/2/3 的平均值
[13:12]	CMP1_SELN	比较器 1 信号负端选择 00: CMP1_IN 01: REF 10: DAC1 输出 11: CMP1_IP1/2/3 的平均值
[11:10]	Reserved	保留，必须为 0
[9]	Reserved	保留，必须为 0
[8]	Reserved	保留，必须为 0
[7:6]	XTRSEL	晶体起振电路电阻调节 XTRSEL<1>=1: N 端阻值增加 XTRSEL<0>=1: P 端电阻减小一倍
[5]	DAC1_OUTEN	DAC1 输出到 IO 使能 1: 使能 DAC1 输出到 IO P4.7 0: 不输出
[4]	DAC0_OUTEN	DAC0 输出到 IO 使能 1: 使能 DAC0 输出到 IO P3.4 0: 不输出
[3]	CMP_HYS	比较器回差选择 0: 20mv 1: 40mv
[2]	DAC1_GAIN	DAC1 量程选择 0: 满量程为 3V

		1: 满量程为 1.2V
[1]	Reserved	保留，必须为 0
[0]	DAC0_GAIN	DAC0 量程选择 0: 满量程为 3V 1: 满量程为 1.2V

5.2.9 SYS_AFE_REG5 模拟配置寄存器 5

地址：0x4001_9824

复位值：0x0

表 5-9 模拟配置寄存器 5 SYS_AFE_REG5

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

位置	位名称	说明
[31:16]		未使用
[15]	Reserved	保留位，必须为 0
[14:12]	CMP0_SELP	比较器 0 信号正端选择 000: CMP0_IP0 001: OPA0_IP 010: OPA0_OUT_HF，即 OPA0 差分输出的一半，详见运放应用笔记 011: OPA1_OUT_HF 100: CMP0_IP1 101: CMP0_IP2 110: CMP0_IP3 111: AVSS
[11]	Reserved	保留位，必须为 0
[10:8]	CMP1_SELP	比较器 1 信号正端选择 000: CMP1_IP0 001: OPA1_IP 010: OPA1_OUT_HF 011: OPA2_OUT_HF 100: CMP1_IP1 101: CMP1_IP2 110: CMP1_IP3

		111: AVSS
[7:6]	CMP4_SELN	比较器 4 信号负端选择 00: CMP4_IN 01: REF 10: DAC0 输出 11: DAC1 输出
[5:4]	CMP5_SELN	比较器 5 信号负端选择 00: CMP5_IN 01: REF 10: DAC0 输出 11: DAC1 输出
[3:2]	CMP2_SELN	比较器 2 信号负端选择 00: CMP2_IN 01: REF 10: DAC0 输出 11: DAC1 输出
[1:0]	CMP3_SELN	比较器 3 信号负端选择 00: CMP3_IN 01: REF 10: DAC0 输出 11: DAC1 输出

5.2.10 SYS_AFE_REG6 模拟配置寄存器 6

地址：0x4001_9828

复位值：0x0

表 5-10 模拟配置寄存器 6 SYS_AFE_REG6

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

位置	位名称	说明
[31:16]		未使用
[15]	Reserved	保留位，必须为 0
[14:12]	CMP4_SELP	比较器 4 信号正端选择 000: CMP4_IP0 001: OPA4_IP 010: OPA4_OUT_HF 011: OPA5_OUT_HF

		100: CMP4_IP1 101: CMP4_IP2 110: CMP4_IP3 111: AVSS
[11]	Reserved	保留位，必须为 0
[10:8]	CMP5_SELP	比较器 5 信号正端选择 000: CMP5_IP0 001: OPA5_IP 010: OPA5_OUT_HF 011: OPA0_OUT_HF 100: CMP5_IP1 101: CMP5_IP2 110: CMP5_IP3 111: AVSS
[7]	Reserved	保留位，必须为 0
[6:4]	CMP2_SELP	比较器 2 信号正端选择 000: CMP2_IP0 001: OPA2_IP 010: OPA2_OUT_HF 011: OPA3_OUT_HF 100: CMP2_IP1 101: CMP2_IP2 110: CMP2_IP3 111: AVSS
[3]	Reserved	保留位，必须为 0
[2:0]	CMP3_SELP	比较器 3 信号正端选择 000: CMP3_IP0 001: OPA3_IP 010: OPA3_OUT_HF 011: OPA4_OUT_HF 100: CMP3_IP1 101: CMP3_IP2 110: CMP3_IP3 111: AVSS

5.2.11 SYS_AFE_REG9 模拟配置寄存器 9

地址：0x4001_9834

复位值：0x0

表 5-11 模拟配置寄存器 9 SYS_AFE_REG9

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

ADCPDN	Res.	CMP5PDN	CMP4PDN	CMP3PDN	CMP2PDN	CMP1PDN	CMP0PDN	Res.	OPA5PDN	OPA4PDN	OPA3PDN	OPA2PDN	OPA1PDN	OPA0PDN
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:16]		未使用
[15]	ADCPDN	ADC 使能 0: 关闭 1: 使能
[14]	Reserved	保留，必须为 0
[13]	CMP5PDN	CMP5 使能 0: 关闭 1: 使能
[12]	CMP4PDN	CMP4 使能 0: 关闭 1: 使能
[11]	CMP3PDN	CMP3 使能 0: 关闭 1: 使能
[10]	CMP2PDN	CMP2 使能 0: 关闭 1: 使能
[9]	CMP1PDN	CMP1 使能 0: 关闭 1: 使能
[8]	CMP0PDN	CMP0 使能 0: 关闭 1: 使能
[7:6]	Reserved	保留，必须为 0
[5]	OPA5PDN	OPA5 使能 0: 关闭 1: 使能
[4]	OPA4PDN	OPA4 使能 0: 关闭 1: 使能
[3]	OPA3PDN	OPA3 使能 0: 关闭 1: 使能
[2]	OPA2PDN	OPA2 使能 0: 关闭

		1: 使能
[1]	OPA1PDN	OPA1 使能 0: 关闭 1: 使能
[0]	OPA0PDN	OPA0 使能 0: 关闭 1: 使能

5.2.12 SYS_AFE_REGA 模拟配置寄存器 A

地址：0x4001_9838

复位值：0x0

表 5-12 模拟配置寄存器 A SYS_AFE_REGA

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

位置	位名称	说明
[31:16]		未使用
[15:8]	Reserved	保留位，必须为 0
[7]	PLLPDN	PLL 使能 0: 关闭 1: 使能
[6]	XTALPDN	晶体起振电路使能 0: 关闭 1: 使能
[5]	TMPPDN	温度传感器使能 0: 关闭 1: 使能
[4]	LDO2_CTL	可掉电 LDO 电源关闭控制信号 0: 开启 1: 关闭
[3]	DAC1PDN	DAC1 使能 0: 关闭 1: 使能
[2]	DAC0PDN	DAC0 使能 0: 关闭

		1: 使能
[1]	RCHPD	RCH 模块关闭信号 0: 开启 1: 关闭
[0]	BGPPD	BGP 模块关闭信号 0: 开启 1: 关闭

如果 SYS_CLK_CFG 选择 PLL 时钟，则 PLLPDN 是被硬件控制的，软件配置 PLLPDN 关闭 PLL 无效。关闭 PLL 需要 PLLPDN=0，且 SYS_CLK_CFG 不选择 PLL 作为芯片主时钟，这两个条件都须满足。同理如果 SYS_CLK_CFG 选择 HRC 时钟，则 RCHPD 是被硬件控制的，软件直接配置 RCHPD 关闭 RCH 无效。关闭 PLL 需要 RCHPD=1，且芯片进入休眠。 如果芯片主时钟为 PLL 时钟，且 HRC 为 PLL 参考时钟，则 RCH 也是被硬件控制的。由于 RCH 和 PLL 依赖 BGP，所以 BGPPD 也是硬件控制的，在芯片使用了 RCH 或 PLL 时，软件直接配置 BGPPD 关闭 BGP 无效。关闭 BGP 需要先顺序关闭 PLL 和 RCH，且芯片进入休眠。LDO2_CTL 也是由休眠状态机控制的(详见 24.4 章节)，软件直接写入关闭 LDO 无效。

5.2.13 SYS_AFE_REGB 模拟配置寄存器 B

地址：0x4001_983C

复位值：0x0

表 5-13 模拟配置寄存器 B SYS_AFE_REGB

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								PLLSR_SEL	PORFIL_EN	Res.	Res.	PVDSEL		VSR_PDT	PDT_PD
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
[15:8]	Reserved	保留位，需全部为’0’
[7]	PLLSR_SEL	PLL 参考时钟选择 0: PLL 使用 RCH 时钟 1: PLL 使用晶体时钟 该信号在不使用晶体时钟的产品或应用下，时钟输出 0;在使用晶体时钟且晶体未停振时输出 1，如晶体停振则输出 0
[6]	PORFIL_EN	POR 滤波使能 0: 关闭 1: 使能
[5]	Reserved	保留位，需全部为’0’
[4]	Reserved	保留位，需全部为’0’

[3:2]	PVDSEL	电源掉电监测阈值选择 00: 3.0V 01: 2.7 10: 2.4V 11: 2.1V
[1]	VSR_PDT	掉电检测基准源选择 0:选择低功耗基准源 1:选择 DAC0 的输出
[0]	PDT_PD	掉电检测电路关闭控制信号 0: 开启 1: 关闭

5.2.14 SYS_TMP_A 温度传感器系数 A 寄存器

地址：0x4001_9858

复位值：0x0

表 5-14 温度传感器系数 A 寄存器 SYS_TMP_A

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

位置	位名称	说明
[31:16]		未使用
[15:0]	TMP_GAIN_A	温度传感器增益校正系数 A

5.2.15 SYS_TMP_B 温度传感器系数 B 寄存器

地址：0x4001_985C

复位值：0x0

表 5-15 温度传感器系数 B 寄存器 SYS_TMP_B

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

位置	位名称	说明
[31:16]		未使用

[15:12]	TMP_SIGN	温度传感器 offset 校正系数 B 符号扩展位
[11:0]	TMP_OFFSET_B	温度传感器 offset 校正系数 B

5.2.16 SYS_AFE_DAC0 DAC0 数字量寄存器

地址：0x4001_9860

复位值：0x0

表 5-16 DAC0 数字量寄存器 SYS_AFE_DAC0

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

位置	位名称	说明
[31:12]		未使用
[11:0]	DAC_IN	DAC0 待转换的数字量输入

5.2.17 SYS_AFE_DAC0_AMC DAC0 增益校正寄存器

地址：0x4001_9864

复位值：0x0

表 5-17 DAC0 增益校正寄存器 SYS_AFE_DAC0_AMC

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

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

5.2.18 SYS_AFE_DAC0_DC DAC0 直流偏置寄存器

地址：0x4001_9868

复位值：0x0

表 5-18 DAC0 直流偏置寄存器 SYS_AFE_DAC0_DC

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

SIGN_EXT	DAC_DC
RO	RW
0	0

位置	位名称	说明
[31:16]		未使用
[15:10]	SIGN_EXT	符号位扩展，B[15:10]={6{B[9]}}，即 B[15:10]都等于 B[9]
[9:0]	DAC_DC	DAC 直流偏置，10bit 有符号数，B[9]为符号位

5.2.19 SYS_AFE_DAC1 DAC1 数字量寄存器

地址：0x4001_9870

复位值：0x0

表 5-19 DAC1 数字量寄存器 SYS_AFE_DAC1

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

位置	位名称	说明
[31:12]		未使用
[11:0]	DAC_IN	DAC1 待转换的数字量输入

5.2.20 SYS_AFE_DAC1_AMC DAC1 增益校正寄存器

地址：0x4001_9874

复位值：0x0

表 5-20 DAC1 增益校正寄存器 SYS_AFE_DAC1_AMC

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

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

5.2.21 SYS_AFE_DAC1_DC DAC1 直流偏置寄存器

地址：0x4001_9878

复位值：0x0

表 5-21 DAC1 直流偏置寄存器 SYS_AFE_DAC1_DC

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

位置	位名称	说明
[31:16]		未使用
[15:10]	SIGN_EXT	符号位扩展，B[15:10]={6{B[9]}}，即 B[15:10]都等于 B[9]
[9:0]	DAC_DC	DAC 直流偏置，10bit 有符号数，B[9]为符号位

DAC 增益校正，记数字输出的 12bitDAC 数值为 DAC_raw，经过校正后的 12bit DAC 数值为 DAC_cali

DAC_cali = DAC_raw * DAC_AMC – DAC_DC;

其中 DAC_AMC 为 DAC 增益校正系数，为 10bit 定点无符号数，B[9]为整数，B[8:0]为小数，大小为 1 左右，例如 DAC_AMC = 10'b10_0001_0000 = 1+1/32, 或 DAC_AMC = 10'b01_1110_1100 = 1-5/128。

DAC_DC 为 DAC 直流偏置，为 10bit 有符号整数，经过符号扩展为 16bit 有符号整数，便于软件直接当做 signed short 进行读取。

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

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

需要注意的是，DAC 有两个输出档位，系统上电后，加载默认档位的 DAC 校准值，若切换到其它档位，请使用原厂提供的库函数。

6 系统时钟复位

6.1 时钟

6.1.1 时钟源

如下表所示，系统包括 5 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	32KHz	内部 RC 振荡器	全温度范围误差 <50%	内部系统管理时钟，用于 WDT，复位信号的滤波和展宽
HSI	12MHz	内部 RC 振荡器	全温度范围误差<1%	可作为 PLL 源时钟
PLL	192MHz	PLL 时钟	0	PLL 输出时钟，以 HSI 作为参考时钟输入，倍频 16 倍后输出 PLL 时钟作为系统主时钟。
JTAG/SWD	取决于设置	调试器		JTAG/SWD 时钟
XTAL	10~24MHz	晶体		外挂晶体

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 12MHz 的参考时钟 HSI 倍频 16 倍至 192MHz。

PLL 经过 n/8 分频后，可以得到 192MHz×n/8 的高速时钟，SYS_CLK_CFG.CLK_SEL 在此分频后的高速时钟与 12MHz 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 12MHz 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。

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 24, 48, 96MHz 等频率值。SYS_CLK_CFG.CLK_SEL 表示选择 PLL 或 12MHz RC 时钟作为系统主时钟。SYS_CLK_CFG.CLK_DIV 作为 PLL 的分频系数。当 SYS_CLK_CFG.CLK_SEL 不为 1 时，即系统时钟 HIS/HSE 或 LSI 作为系统工作时钟，此时 SYS_CLK_CFG.CLK_DIV 不再起分频作用。

表 6-2 PLL 作为 MCLK 时钟时的分频配置

SYS_CLK_CFG	分频系数	频率/MHz	是否均匀
0x0101	1/8	24	是
0x0111	2/8	48	是
0x0155	4/8	96	是
0x01FF	8/8	192	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。I2C0/1 时钟由 SYS_CLK_DIV0 寄存器控制可以进一步分频，UART0/1/2 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 SYS_AFE_REG3.ADCCLKSEL 控制的 2/4/8 分频后送至 ADC（典型工作频率 96MHz），即 ACLK。

内部 32kHz 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。

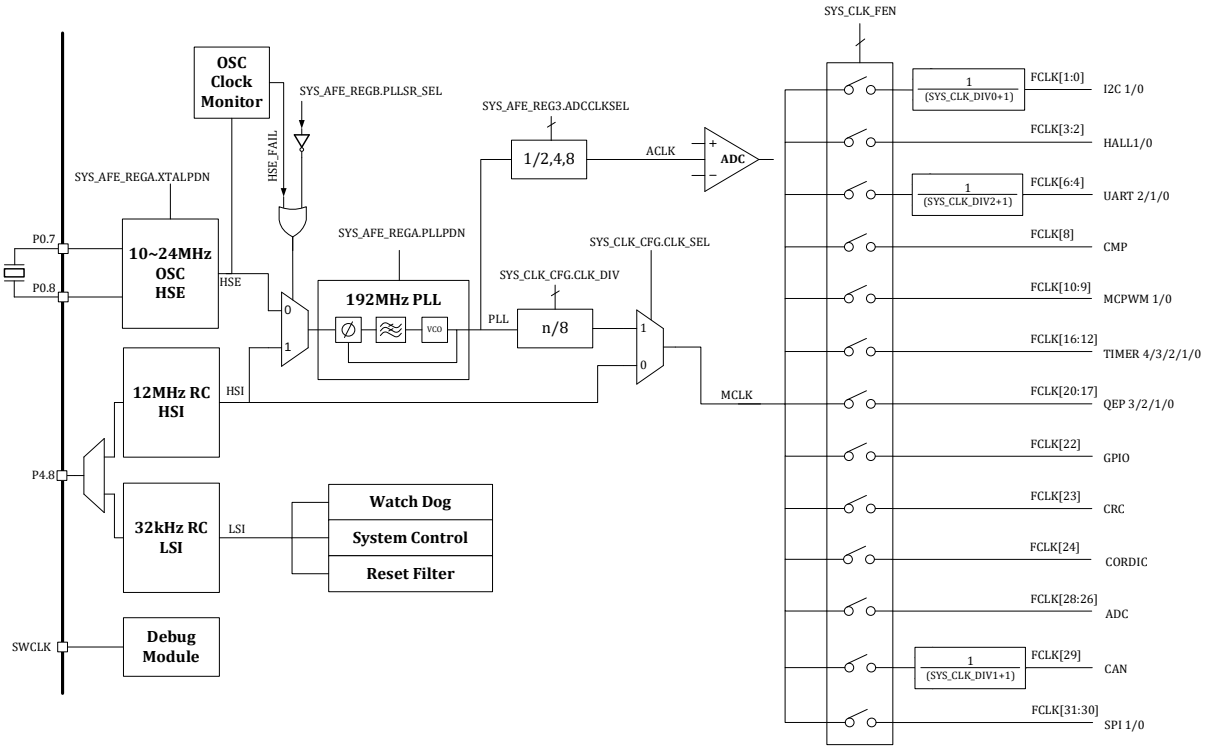


图 6-1 时钟架构

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；32kHz LSI 时钟上电即工作，且无法关闭。SWCLK 由调试器提供，时钟频率取决于上位机调试界面的设定。

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观测。

6.2 复位

6.2.1 复位源

芯片的复位包括硬件复位与软件复位。

6.2.1.1 硬件复位

如表 6-3 所示，系统包括 4 个硬件复位源，产生的复位均为芯片全局复位，复位产生后芯片程序计数器回到 0 地址，所有寄存器恢复到默认值。4 个硬件复位均为低电平有效。

表 6-3 系统复位源

名称	来源	说明
LPORn	内部 1.5V 电源管理	监控 1.5V 数字电源，低于 1.25V 时产生复位
HPORn	内部 3.3V 电源管理	监控 3.3V 电源，低于 2.1V 时产生复位
RSTn	外部按键	外部 RC 组成按键复位电路
IWDTn	独立看门狗	使用 LSI 时钟进行计数，如果不进行喂狗则定时产生复位，复位间隔可配置
WWDTn	窗口看门狗	使用系统高速时钟进行计数，如果不进行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

LPORn/HPORn 来自内部模拟电路，RSTn 来自外部按键。IWDTn 为独立看门狗复位，使用 32kHz LSI 时钟，芯片上电后默认开启，即使在晶振停振或 PLL 未工作时 IWDTn 也可以产生复位信号。WWDTn 为窗口看门狗复位信号，使用系统高速时钟进行计数，通常为 PLL 主时钟，主要用于对复位时间间隔有精确要求的应用场景。IWDTn 和 WWDTn 信号在调试模式下可以被禁用，通过 SYS_DBG_CFG 进行设置。

所有复位信号经过滤波展宽预处理后，形成一个全局复位信号，对全芯片进行复位。flash 和 RAM 存储器内容不受复位影响。

P0.6 引脚宽度小于 32us 的复位脉冲会被滤除，要求可靠复位宽度大于 200us。

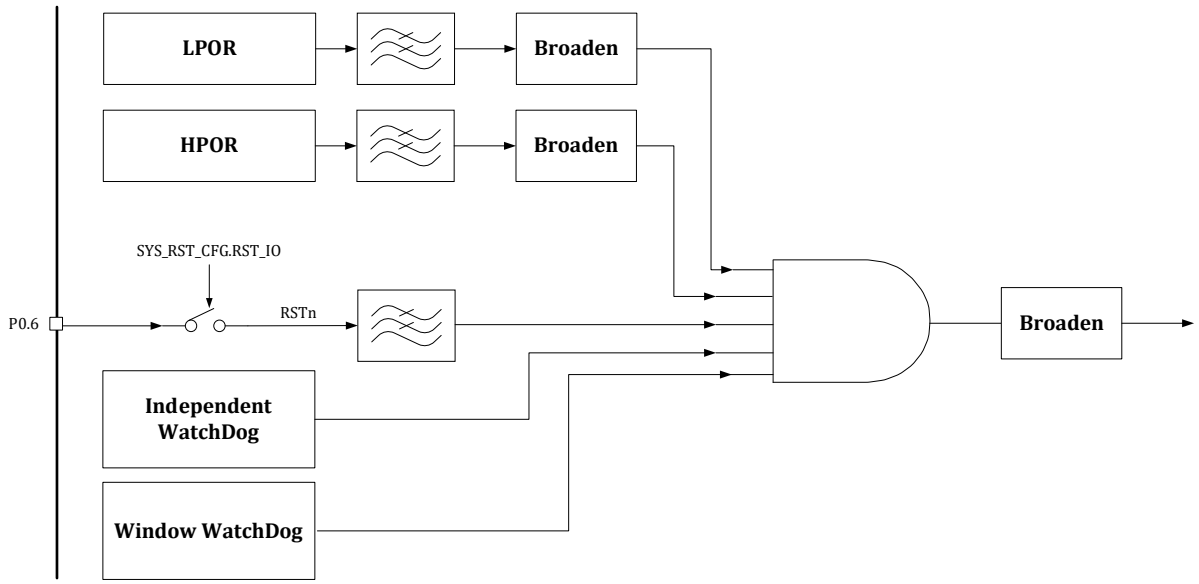


图 6-2 复位架构

6.2.1.1.2 硬件复位记录

在 PMU 功耗管理模块中，AON_EVT_RCD 寄存器用于保存包括硬件复位在内的各种事件记录，当某个硬件复位发生后，AON_EVT_RCD 对应位置位。AON_EVT_RCD 寄存器本身只能被 LPORn 复位信号复位，通过向 AON_EVT_RCD 寄存器写入 0xCA40 可以清空 AON_EVT_RCD 记录，复位记录可以方便地了解是否发生以及发生过何种复位。

6.2.1.2 软件复位

CPU 的软复位操作可以使程序计数器(PC: Program Counter)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 6.3.9。

6.2.2 复位作用域

表 6-4 复位作用域

复位源	作用域
LPORn	内部 1.5V 电源管理，全局复位
HPOR	内部 3.3V 电源管理，全局复位，除极少数寄存器
RSTn	外部按键，全局复位，除极少数寄存器
IWDTn	独立看门狗，全局复位，除独立看门狗模块和极少数寄存器
WWDTn	窗口看门狗，全局复位，除独立看门狗模块和极少数寄存器
SYS_SFT_RST.SPI1_ SFT_RST	SPI1
SYS_SFT_RST.SPI0_ SFT_RST	SPI0
SYS_SFT_RST.CAN_ SFT_RST	CAN
SYS_SFT_RST.ADC2_ SFT_RST	ADC2 数字接口模块
SYS_SFT_RST.ADC1_ SFT_RST	ADC1 数字接口模块
SYS_SFT_RST.ADC0_ SFT_RST	ADC0 数字接口模块
SYS_SFT_RST.CORDIC_ SFT_RST	CORDIC
SYS_SFT_RST.CRC_ SFT_RST	CRC
SYS_SFT_RST.GPIO_ SFT_RST	GPIO
SYS_SFT_RST.QEP3_SFT_RST	QEP3
SYS_SFT_RST.QEP2_SFT_RST	QEP2
SYS_SFT_RST.QEP1_SFT_RST	QEP1
SYS_SFT_RST.QEP0_SFT_RST	QEP0
SYS_SFT_RST.TIMER4_ SFT_RST	TIMER4
SYS_SFT_RST.TIMER3_ SFT_RST	TIMER3
SYS_SFT_RST.TIMER2_ SFT_RST	TIMER2

SYS_SFT_RST.TIMER1_ SFT_RST	TIMER1
SYS_SFT_RST.TIMER0_ SFT_RST	TIMER0
SYS_SFT_RST.MCPWM1_ SFT_RST	MCPWM1
SYS_SFT_RST.MCPWM0_ SFT_RST	MCPWM0
SYS_SFT_RST.CMP_ SFT_RST	比较器数字接口模块
SYS_SFT_RST.UART2_ SFT_RST	UART2
SYS_SFT_RST.UART1_SFT_RST	UART1
SYS_SFT_RST.UART0_SFT_RST	UART0
SYS_SFT_RST.HALL1_SFT_RST	HALL1
SYS_SFT_RST.HALL0_SFT_RST	HALL0
SYS_SFT_RST.I2C1_SFT_RST	I2C1
SYS_SFT_RST.I2C0_SFT_RST	I2C0
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[6]作为 GPIO 使用还是作为外部复位脚使用的 SYS_IO_CFG.RST_IO，以及复位记录寄存器 AON_EVT_RCD 只受 LPOR 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 0x40019800。

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_DIV1	0x94	外设时钟分频寄存器 1
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_CACHE_CFG	0xAC	Cache 控制寄存器
SYS_MEM_CFG	0xC4	Memory 控制寄存器
SYS_FLSE	0xD0	On-Chip FLASH 擦除控制寄存器
SYS_ FLSP	0xD4	On-Chip FLASH 编程控制寄存器

6.3.2 SYS_CLK_CFG 时钟控制寄存器

地址：0x4001_9880

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
		XTAL_FAILED		XTAL_FAIL		CLK_SEL		CLK_DIV							
		RW1C		RO		RW		RW							
		1		0		0		0							

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

当 CLK_SEL = 0 时，MCLK 选择 CLK_HS 时钟，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 CLK_HS 时钟频率。根据 SYS_AFE_REGB.PLLSR_SEL，当 SYS_AFE_REGB.PLLSR_SEL=0 时，CLK_HS 为芯片内部 12MHz RCH 时钟 HSI，当 SYS_AFE_REGB.PLLSR_SEL=1 时，CLK_HS 为晶体时钟。

6.3.3 SYS_IO_CFG IO 控制寄存器

地址：0x4001_9884

复位值：0x02

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
								TRACEMUX	SWDMUX	RST_IO				IO_DS	
								RW	RW	RW				RW	
								0	0	0				2	

位置	位名称	说明
[31:8]		未使用
[7]	TRACEMUX	TRACE 复用控制信号，默认配置为 TRACE 功能 0: P1[3:0]，P0[15:14]作为正常 GPIO 使用 1: P1[3]复用 SWV，{P0[15],P1[0:2]}复用为 TRACEDATA[3:0]，P0[14]复用为 TRACECLK
[6]	SWDMUX	JTAG/SWD 复用控制信号，默认配置为 JTAG/SWD 0: IO 功能为 JTAG/SWD P0.9--->nTRST P0.10--->SWDIOTMS P0.11 --->TDI P0.12--->SWCLKTCLK P0.13--->TDO 1:P0[13:9]作为正常 GPIO 使用
[5]	RST_IO	RSTn/P0.6 复用控制信号，默认配置为 RSTn 0:RSTn 1:P0.6 注意，上电后默认是 RSTn，后续软件可使能此位，RSTn 功能失效
[4]	POR_FIL	未对用户开放 此寄存器使用 SYS_AFE_REG6[11]作为复位，所以需要先配置 SYS_AFE_REG6[11]=1，然后才可以使能 POR 滤波，POR 滤波对 LPOR 和 HPOR 均有效
[3:2]		未使用
[1:0]	IO_DS	IO 驱动能力选择

		0: 4.5mA 1: 9mA 2: 13.5mA 3: 18mA 上电默认驱动能力为 13.5mA

为了安全起见，上电后 30ms 内，SWD/Jtag IO 不能切换为 GPIO 功能，只能作为 SWD/Jtag，这是为了保证芯片在上电初期一定可以通过调试器/下载器进行控制。即使软件改写了 SYS_IO_CFG.SWDMUX 也需要在 30ms 之后才会生效。在上电之后的 30ms 内，软件可以向 SYS_IO_CFG.SWDMUX 写入 1'b0，但读回为 1'b1，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。

注意，SWD/Jtag IO 切换为 GPIO 功能后，除非应用软件特殊设计，否则只能使用原厂下载器擦除下载程序。

6.3.4 SYS_DBG_CFG Debug 控制寄存器

地址：0x4001_9888

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_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP	DBG_TIM1_STOP	DBG_TIM0_STOP			DBG_IWDG_STOP	DBG_WWDG_STOP		DBG_STDBY	DBG_STOP	DBG_SLP
RW1C	RW		RW	RW	RW	RW	RW			RW	RW		RW	RW	RW
0	0		0	0	0	0	0			0	0		0	0	0

位置	位名称	说明
[31:16]	SW_IRQ_TRIG	向此位段写入 0x5AA5 触发软件中断，SW_IRQ 置 1
[15]	SW_IRQ	软件中断标志，对应中断号 30，写 1 清零。
[14]	SFT_RST_PERI	Debug 软复位是否复位除 Flash/SYS_AFE 以外的外设寄存器

[13]		未使用
[12]	DBG_TIM4_STOP	调试模式下 CPU halt 状态 Timer4 停止 1：Timer4 在 CPU halt 状态时停止计数 0：Timer4 在 CPU halt 状态时继续计数
[11]	DBG_TIM3_STOP	调试模式下 CPU halt 状态 Timer3 停止 1：Timer3 在 CPU halt 状态时停止计数 0：Timer3 在 CPU halt 状态时继续计数
[10]	DBG_TIM2_STOP	调试模式下 CPU halt 状态 Timer2 停止 1：Timer2 在 CPU halt 状态时停止计数 0：Timer2 在 CPU halt 状态时继续计数
[9]	DBG_TIM1_STOP	调试模式下 CPU halt 状态 Timer1 停止 1：Timer1 在 CPU halt 状态时停止计数 0：Timer1 在 CPU halt 状态时继续计数
[8]	DBG_TIM0_STOP	调试模式下 CPU halt 状态 Timer0 停止 1：Timer0 在 CPU halt 状态时停止计数 0：Timer0 在 CPU halt 状态时继续计数
[7:6]		未使用
[5]	DBG_IWDG_STOP	调试模式下 CPU halt 状态独立看门狗停止 1：独立看门狗在 CPU halt 状态时停止计数 0：独立看门狗在 CPU halt 状态时继续计数
[4]	DBG_WWDG_STOP	调试模式下 CPU halt 状态窗口看门狗停止 1：窗口看门狗在 CPU halt 状态时停止计数 0：窗口看门狗在 CPU halt 状态时继续计数
[3]		未使用
[2]	DBG_STDBY	调试 StandBy 模式 0: (FCLK=Off, HCLK=Off) 整个数字电路进入关电状态(除 PMU)。从软件角度看，退出 Standby 休眠与上电复位相同，只是 AON_EVT_RCD 会指示休眠唤醒等状态位 1: (FCLK=On, HCLK=On) 如果设置了 DBG_STDBY=1，数字部分不会掉电，而且内部高速均不关闭。但 PMU 仍会在退出休眠时产生复位信号，重新开始取指执行。使得执行过程与不设置 DBG_STDBY 时相同
[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。
[0]	DBG_SLP	调试睡眠模式 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：0x4001_9890

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	I2C0/I2C1 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_DIV1 外设时钟分频寄存器 1

地址：0x4001_9894

复位值：0x0

表 6-10 SYS_CLK_DIV1 外设时钟分频寄存器 1

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]	DIV1	CAN 工作时钟=MCLK/(CLK_DIV1+1)。其中 MCLK 由 SYS_CLK_CFG 分频系数决定

6.3.7 SYS_CLK_DIV2 外设时钟分频寄存器 2

地址：0x4000_0098

复位值：0x0

表 6-11 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。

6.3.8 SYS_CLK_FEN 外设时钟门控寄存器

地址：0x4001_989C

复位值：0x0

表 6-12 SYS_CLK_FEN 外设时钟门控寄存器

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

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

位置	位名称	说明
[31]	SPI1_CLK_EN	SPI1 模块时钟控制信号，默认关闭 SPI0 模块时钟 1:使能 SPI0 模块时钟 0:关闭 SPI0 模块时钟
[30]	SPI0_CLK_EN	SPI0 模块时钟控制信号，默认关闭 SPI0 模块时钟 1:使能 SPI0 模块时钟

		0:关闭 SPI0 模块时钟
[29]	CAN_CLK_EN	CAN 模块时钟控制信号，默认关闭 CAN 模块时钟 1:使能 CAN 模块时钟 0:关闭 CAN 模块时钟
[28]	ADC2_CLK_EN	ADC2 模块时钟控制信号，默认关闭 ADC2 模块时钟 1:使能 ADC2 模块时钟 0:关闭 ADC2 模块时钟
[27]	ADC1_CLK_EN	ADC1 模块时钟控制信号，默认关闭 ADC1 模块时钟 1:使能 ADC1 模块时钟 0:关闭 ADC1 模块时钟
[26]	ADC0_CLK_EN	ADC0 模块时钟控制信号，默认关闭 ADC0 模块时钟 1:使能 ADC0 模块时钟 0:关闭 ADC0 模块时钟
[25]		未使用
[24]	CORDIC_CLK_EN	CORDIC 模块时钟控制信号，默认关闭 CORDIC 模块时钟 1:使能 CORDIC 模块时钟 0:关闭 CORDIC 模块时钟
[23]	CRC_CLK_EN	CRC 模块时钟控制信号，默认关闭 CRC 模块时钟 1:使能 CRC 模块时钟 0:关闭 CRC 模块时钟
[22]	GPIO_CLK_EN	GPIO 模块时钟控制信号，默认关闭 GPIO 模块时钟 1:使能 GPIO 模块时钟 0:关闭 GPIO 模块时钟
[21]		未使用
[20]	QEP3_CLK_EN	QEP3 模块时钟控制信号，默认关闭 QEP3 模块时钟 1:使能 QEP3 模块时钟 0:关闭 QEP3 模块时钟
[19]	QEP2_CLK_EN	QEP2 模块时钟控制信号，默认关闭 QEP2 模块时钟 1:使能 QEP2 模块时钟 0:关闭 QEP2 模块时钟
[18]	QEP1_CLK_EN	QEP1 模块时钟控制信号，默认关闭 QEP1 模块时钟 1:使能 QEP1 模块时钟 0:关闭 QEP1 模块时钟
[17]	QEP0_CLK_EN	QEP0 模块时钟控制信号，默认关闭 QEP0 模块时钟 1:使能 QEP0 模块时钟 0:关闭 QEP0 模块时钟
[16]	TIMER4_CKL_EN	TIMER4 模块时钟控制信号，默认关闭 TIMER4 模块时钟 1:使能 TIMER4 模块时钟 0:关闭 TIMER4 模块时钟
[15]	TIMER3_CKL_EN	TIMER3 模块时钟控制信号，默认关闭 TIMER3 模块时钟 1:使能 TIMER3 模块时钟 0:关闭 TIMER3 模块时钟
[14]	TIMER2_CKL_EN	TIMER2 模块时钟控制信号，默认关闭 TIMER2 模块时钟

		1:使能 TIMER2 模块时钟 0:关闭 TIMER2 模块时钟
[13]	TIMER1_CKL_EN	TIMER1 模块时钟控制信号，默认关闭 TIMER1 模块时钟 1:使能 TIMER1 模块时钟 0:关闭 TIMER1 模块时钟
[12]	TIMER0_CKL_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
[11]		未使用
[10]	MCPWM1_CKL_EN	MCPWM1 模块时钟控制信号，默认关闭 MCPWM1 模块时钟 1:使能 MCPWM1 模块时钟 0:关闭 MCPWM1 模块时钟
[9]	MCPWM0_CKL_EN	MCPWM0 模块时钟控制信号，默认关闭 MCPWM0 模块时钟 1:使能 MCPWM0 模块时钟 0:关闭 MCPWM0 模块时钟
[8]	CMP_CKL_EN	CMP 模块时钟控制信号，默认关闭 CMP 模块时钟 1:使能 CMP 模块时钟 0:关闭 CMP 模块时钟
[7]		未使用
[6]	UART2_CLK_EN	UART2 模块时钟控制信号，默认关闭 UART2 模块时钟 1:使能 UART2 模块时钟 0:关闭 UART2 模块时钟
[5]	UART1_CLK_EN	UART1 模块时钟控制信号，默认关闭 UART1 模块时钟 1:使能 UART1 模块时钟 0:关闭 UART1 模块时钟
[4]	UART0_CLK_EN	UART0 模块时钟控制信号，默认关闭 UART0 模块时钟 1:使能 UART0 模块时钟 0:关闭 UART0 模块时钟
[3]	HALL1_CLK_EN	HALL1 模块时钟控制信号，默认关闭 HALL1 模块时钟 1:使能 HALL1 模块时钟 0:关闭 HALL1 模块时钟
[2]	HALL0_CLK_EN	HALL0 模块时钟控制信号，默认关闭 HALL0 模块时钟 1:使能 HALL0 模块时钟 0:关闭 HALL0 模块时钟
[0]	I2C1_CLK_EN	I2C1 模块时钟控制信号，默认关闭 I2C1 模块时钟 1:使能 I2C1 模块时钟 0:关闭 I2C1 模块时钟
[0]	I2C0_CLK_EN	I2C0 模块时钟控制信号，默认关闭 I2C0 模块时钟 1:使能 I2C0 模块时钟 0:关闭 I2C0 模块时钟

注意，上述每个模块的时钟门控，只控制各模块的工作时钟。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配置寄存器。

6.3.9 SYS_SFT_RST 软复位寄存器

地址：0x4001_98A4

复位值：0x0

表 6-13 软复位寄存器 SYS_SFT_RST

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

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

位置	位名称	说明
[31]	SPI1_ SFT_RST	SPI1 模块软复位信号，默认不复位 SPI0 模块 1:复位 SPI0 模块 0:释放 SPI0 模块
[30]	SPI0_ SFT_RST	SPI0 模块软复位信号，默认不复位 SPI0 模块 1:复位 SPI0 模块 0:释放 SPI0 模块
[29]	CAN_ SFT_RST	CAN 模块软复位信号，默认不复位 CAN 模块 1:复位 CAN 模块 0:释放 CAN 模块
[28]	ADC2_ SFT_RST	ADC2 模块软复位信号，默认不复位 ADC2 模块 1:复位 ADC2 模块 0:释放 ADC2 模块
[27]	ADC1_ SFT_RST	ADC1 模块软复位信号，默认不复位 ADC1 模块 1:复位 ADC1 模块 0:释放 ADC1 模块
[26]	ADC0_ SFT_RST	ADC0 模块软复位信号，默认不复位 ADC0 模块 1:复位 ADC0 模块 0:释放 ADC0 模块
[25]		未使用

[24]	CORDIC_ SFT_RST	CORDIC 模块软复位信号，默认不复位 CORDIC 模块 1:复位 CORDIC 模块 0:释放 CORDIC 模块
[23]	CRC_ SFT_RST	CRC 模块软复位信号，默认不复位 CRC 模块 1:复位 CRC 模块 0:释放 CRC 模块
[22]	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
[21]	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
[20]	QEP3_SFT_RST	QEP3 模块软复位信号，默认不复位 QEP3 模块 1:复位 QEP3 模块 0:释放 QEP3 模块
[19]	QEP2_SFT_RST	QEP2 模块软复位信号，默认不复位 QEP2 模块 1:复位 QEP2 模块 0:释放 QEP2 模块
[18]	QEP1_SFT_RST	QEP1 模块软复位信号，默认不复位 QEP1 模块 1:复位 QEP1 模块 0:释放 QEP1 模块
[17]	QEP0_SFT_RST	QEP0 模块软复位信号，默认不复位 QEP0 模块 1:复位 QEP0 模块 0:释放 QEP0 模块
[16]	TIMER4_ SFT_RST	TIMER4 模块软复位信号，默认不复位 TIMER4 模块 1:复位 TIMER4 模块 0:释放 TIMER4 模块
[15]	TIMER3_ SFT_RST	TIMER3 模块软复位信号，默认不复位 TIMER3 模块 1:复位 TIMER3 模块 0:释放 TIMER3 模块
[14]	TIMER2_ SFT_RST	TIMER2 模块软复位信号，默认不复位 TIMER2 模块 1:复位 TIMER2 模块 0:释放 TIMER2 模块
[13]	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
[12]	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
[11]		未使用
[10]	MCPWM1_ SFT_RST	MCPWM1 模块软复位信号，默认不复位 MCPWM1 模块 1:复位 MCPWM1 模块 0:释放 MCPWM1 模块

[9]	MCPWM0_ SFT_RST	MCPWM0 模块软复位信号，默认不复位 MCPWM0 模块 1:复位 MCPWM0 模块 0:释放 MCPWM0 模块
[8]	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
[7]		未使用
[6]	UART2_ SFT_RST	UART2 模块软复位信号，默认不复位 UART2 模块 1:复位 UART2 模块 0:释放 UART2 模块
[5]	UART1_SFT_RST	UART1 模块软复位信号，默认不复位 UART1 模块 1:复位 UART1 模块 0:释放 UART1 模块
[4]	UART0_SFT_RST	UART0 模块软复位信号，默认不复位 UART0 模块 1:复位 UART0 模块 0:释放 UART0 模块
[3]	HALL1_SFT_RST	HALL1 模块软复位信号，默认不复位 HALL1 模块 1:复位 HALL1 模块 0:释放 HALL1 模块
[2]	HALL0_SFT_RST	HALL0 模块软复位信号，默认不复位 HALL0 模块 1:复位 HALL0 模块 0:释放 HALL0 模块
[1]	I2C1_SFT_RST	I2C1 模块软复位信号，默认不复位 I2C1 模块 1:复位 I2C1 模块 0:释放 I2C1 模块
[0]	I2C0_SFT_RST	I2C0 模块软复位信号，默认不复位 I2C0 模块 1:复位 I2C0 模块 0:释放 I2C0 模块

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。由于复位信号内部会进行展宽。所以软复位释放后，至少等待 1 个总线周期后再对外设进行读写。可以通过插入__asm("NOP");实现等待。

6.3.10 SYS_PROTECT 写保护寄存器

地址：0x4001_98A8

复位值：0x0

表 6-14 写保护寄存器 SYS_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_DACx、SYS_AFE_DACx_AMC、SYS_AFE_DACx_DC(x=0,1)外，其他系统寄存器均受写保护，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

6.3.11 SYS_CAHCE_CFG 缓存控制寄存器

地址：0x4001_98AC

复位值：0x0

表 6-15 SYS_CACHE_CFG 缓存寄存器

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	Cache 使能，1：启用 Cache，0：禁用 Cache

6.3.12 SYS_MEM_CFG 内存控制寄存器

地址：0x4001_98C4

复位值：0x0

表 6-16 SYS_MEM_CFG 内存控制寄存器

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

位置	位名称	说明
[31:10]		未使用
[9:8]	REMAP	地址空间重映射，用于将 SRAM1/2/3 映射至 0x1FFFFFFF 之前的地址空间做为 Code RAM 使用，Code RAM 可用于存放关键代码，由于 RAM 访问速度更快，可以加速关键代码的的执行。

		0：SRAM0/1/2/3 均用为 Data RAM，无 Code RAM 1：SRAM0/1/2 用为 Data RAM，SRAM3 用作 Code RAM 2：SRAM0/1 用为 Data RAM，SRAM2/3 用作 Code RAM 3：SRAM0 用为 Data RAM，SRAM1/2/3 用作 Code RAM 具体地址空间请参考第 3 章地址空间章节
[7:4]	PCK_EN	SRAM3/2/1/0 内存奇偶校验使能
[3:0]	PCK_ERR	SRAM3/2/1/0 内存奇偶校验错误，0：无错误发生，1：有错误发生，写 1 清零

只有当使能校验时，对应的奇偶位才可能在校验出错时置位。当奇偶校验出错时，产生不可屏蔽中断 NMI。

6.3.13 SYS_FLSE On-Chip FLASH 擦除控制寄存器

地址：0x4001_98D0

复位值：0x0

表 6-17 擦除控制寄存器 SYS_FLSE

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

位置	位名称	说明
[31:16]		未使用
[15:0]	FLSE	**On-Chip FLASH 擦除控制寄存器。本寄存器写入 0x8FCA，On-Chip FLASH 的擦除功能才能最终生效。写入其他值，On-Chip FLASH 的擦除功能均无法生效。为防止误擦除，建议不使用擦除功能时，不生效此寄存器。**

6.3.14 SYS_FLSP On-Chip FLASH 编程控制寄存器

地址：0x4001_98D4

复位值：0x0

表 6-18 编程控制寄存器 SYS_FLSP

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

位置	位名称	说明

[31:16]		未使用
[15:0]	FLSP	**On-Chip FLASH 编程控制寄存器。本寄存器写入 0x8F35，On-Chip FLASH 的编程功能才能最终生效。写入其他值，On-Chip FLASH 的编程功能均无法生效。为防止误编程，建议不使用编程功能时，不生效此寄存器。**

7 非易失存储体

7.1 概述

LKS32MC45x 系列芯片，非易失性存储体包含两个部分:On-Chip FLASH 存储体及其控制器和 Ext-Chip FLASH 控制器。

On-Chip FLASH 存储体包含三个区域：NVR、MAIN 和 ROM。NVR 大小为 2KB；MAIN 大小根据具体型号有不同划分，ROM 大小根据具体型号有不同划分。

- On-Chip FLASH 存储体主存储区域（MAIN），包括应用程序和用户数据区
- On-Chip FLASH 存储体附存储区域（ROM），包括应用程序和用户数据区
- On-Chip FLASH 存储体信息存储区域（NVR），预留给用户使用数据区

On-Chip FLASH 存储体 ROM 区域，预留给客户一次编程机会，客户可将自有的程序写入 ROM 区域并锁定。MCU 可执行 ROM 区域程序，但无法搬移 ROM 区域的内容。

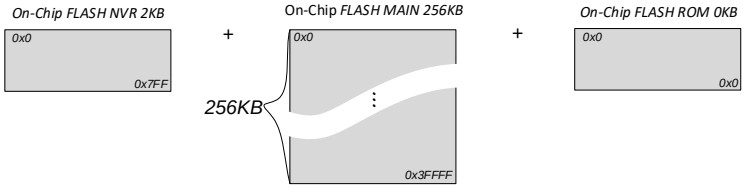
LKS32MC45x 系列芯片，On-Chip FLASH 存储体共有七种组合。

Ext-Chip FLASH 控制器最大支持容量为 16MB 的外挂 SPI 接口 FLASH。外挂 SPI 接口 FLASH，需要客户自行采购，LKS32MC45x 的 Ext-FLASH 控制器支持市面上主流的外挂 SPI 接口 FLASH。

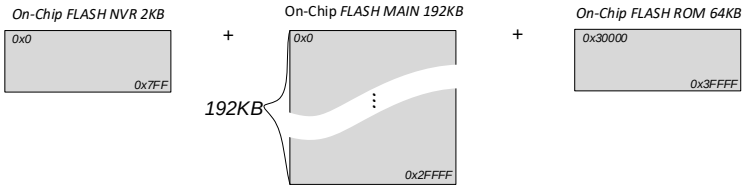
- Ext-Chip FLASH 控制器，仅支持 SPI 接口 FLASH，支持单线/双线/四线三种数据传输模式
- Ext-Chip FLASH 控制器，支持 MCU 直接运行外挂 SPI 接口 FLASH 中的应用程序，同时，也支持将外挂 SPI 接口 FLASH 中的应用程序搬移到片内执行

为提升 LKS32MC45x 系列芯片的程序执行效率，减少非易失性存储体运行速度低于系统频率所带来的性能损失，额外设计了一个 Cache 加速模块。On-Chip Flash 的内容能预取存入 Cache，加速其运行效率；但 Ext-Chip FLASH 的内容不进 Cache，无法获得加速。

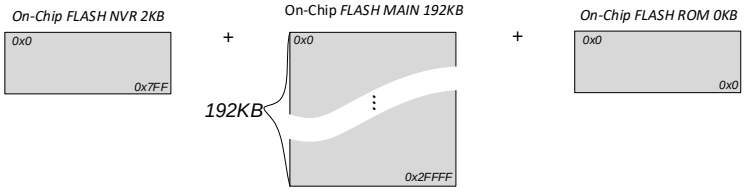
NVR 2KB/MAIN 256KB/ROM 0KB



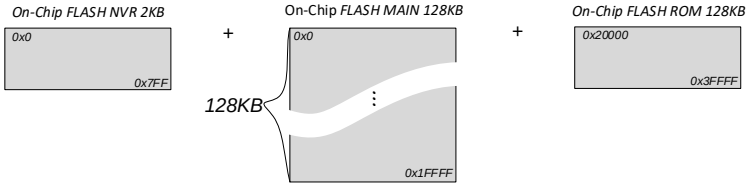
NVR 2KB/MAIN 192KB/ROM 64KB



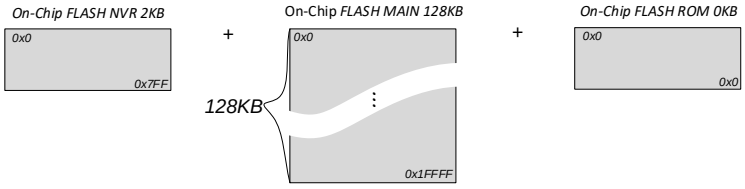
NVR 2KB/MAIN 192KB/ROM 0KB



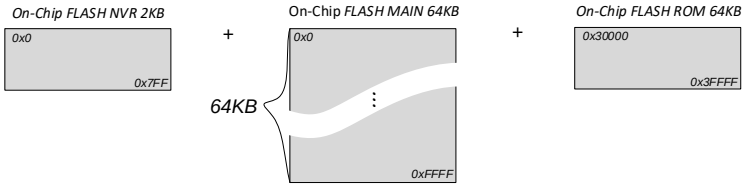
NVR 2KB/MAIN 128KB/ROM 128KB



NVR 2KB/MAIN 128KB/ROM 0KB



NVR 2KB/MAIN 64KB/ROM 64KB



NVR 2KB/MAIN 64KB/ROM 0KB

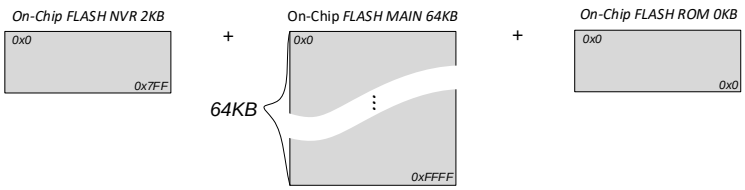


图 7-1 On-Chip FLASH 存储体空间划分框图

7.2 功能特点

On-Chip FLASH 控制器模块，主要实现对 On-Chip FLASH 存储体的相关操作。包括：

- On-Chip FLASH 读取数据的操作。包括对 NVR 区域的读取和对 MAIN 区域的读取。锁定后的 ROM 区域，不支持读取。
- On-Chip FLASH 编程数据的操作。包括对 NVR 区域的编程和对 MAIN 区域的编程。ROM 区域不支持二次编程。
- On-Chip FLASH 擦除操作。支持 Sector 擦除。NVR 区域和 MAIN 区域支持 Sector 擦除。ROM 部分不支持擦除。
- On-Chip FLASH 可执行操作。即应用程序可直接运行在 MAIN 区域和 ROM 区域，NVR 区域不可执行，只能存储数据。
- On-Chip FLASH 深度休眠的操作，以降低芯片的休眠功耗。
- On-Chip FLASH 存储体内容的保护操作。
- On-Chip FLASH 的 Cache 加速模块，以提升芯片整体运行效率。
- On-Chip FLASH 控制寄存器的访问。

Ext-Chip FLASH 控制器模块，主要实现对 Ext-Chip FLASH 存储体的相关操作。包括：

- Ext -Chip FLASH 读取数据的操作。
- Ext -Chip FLASH 编程数据的操作。
- Ext -Chip FLASH 擦除操作。
- Ext -Chip FLASH 控制寄存器的访问。

7.2.1 On-Chip FLASH 功能描述

控制模块，实现了对 On-Chip FLASH 存储体的复位/读出/编程/擦除/休眠等操作。如下为控制状态转换图：

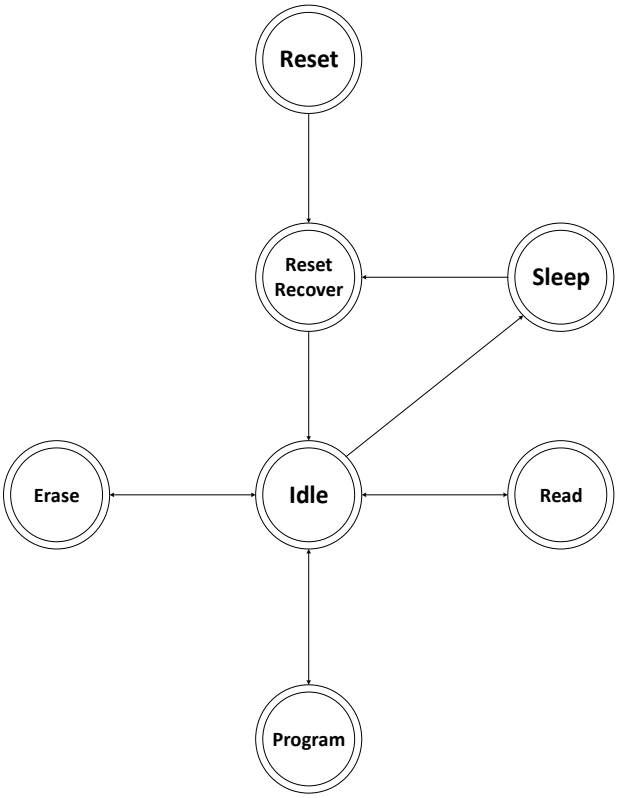


图 7-2 On-Chip FLASH 控制状态转换图

7.2.1.1 复位操作

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

7.2.1.2 休眠操作

On-Chip FLASH 的休眠操作分成两个部分：Standby 和 Deep Sleep。当系统不执行对 On-Chip FLASH 的操作时，On-Chip FLASH 可自动进入 StandBy 状态(若开启 Cache 加速模块，此功能失效)。当系统执行 Deep Sleep 操作时，将触发 On-Chip FLASH 也进入 Deep Sleep，实现降低功耗的目的。On-Chip 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 On-Chip FLASH。经过一段时间恢复后，On-Chip 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 On-Chip FLASH 读取操作

读操作为 On-Chip 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 总线的 I 总线和 D 总线，直接（Direct）对 On-Chip FLASH 执行取指、取数操作。总线宽度为 32Bit，且只能访问 MAIN 区域和 ROM 区域（只运行不能被读取）的数据。为了提升 MCU 的读取效率，硬件提供了 Cache 加速的功能。
- MCU 通过 AHB 总线中的 S 总线，访问控制器的寄存器，间接（Indirect）实现读取 On-Chip FLASH

内部数据的操作。可以访问 MAIN 区域和 NVR 区域的数据，ROM 区域为有条件访问；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。此时，存在 I/D 总线和 S 总线，同时访问 FLASH 的情况，On-Chip FLASH 内部有仲裁机制，优先 I/D 总线访问，S 总线需等待 I/D 总线空闲才能完成对应操作；一旦 S 总线获得访问 On-Chip FLASH 的权力，I/D 总线无法抢夺直至 S 总线访问完毕。

- 寄存器 FSMC_REDY.READY 位，指示当前 I/D 总线是否完成了当前读取访问操作，1 表示完成，0 表示未完成。
- 寄存器 FSMC_REDY.IREADY 位，指示当前 S 总线是否完成了当前读取访问操作，1 表示完成，0 表示未完成。

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

表 7-1 On-Chip FLASH 访问空间分配表

FSMC_ICFG.REGION	访问区域
0	MAIN 区域和 ROM 区域*
1	NVR 区域

*注意，ROM 区域默认支持一次编程，编程完毕后，此区域被限定 XN（eXecute regioN），锁定后不能被读取，也不允许二次编程和擦除。

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

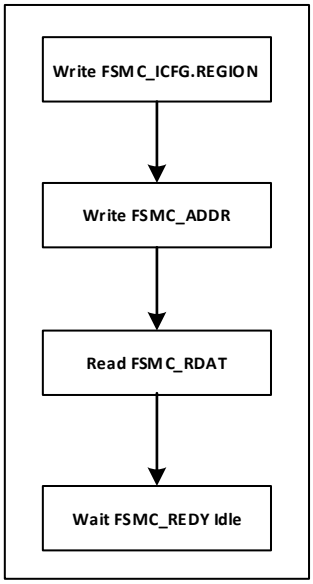


图 7-3 On-Chip FLASH 间接读取操作流程图

7.2.1.4 On-Chip FLASH 编程操作

执行对 On-Chip FLASH 存储体的编程操作。一般而言，我们先执行擦除操作（保证对应编程背景为默认值，杜绝同一地址连续两次编程），再执行数据编程操作。同时，只能通过访问 On-Chip FLASH 控制器的寄存器，实现编程操作。LKS32MC45x 芯片的编程宽度为 128-Bit，即一次写入 16 个 Byte。

类似读取操作。此时，存在 I/D 总线和 S 总线，同时访问 FLASH 的情况。I/D 总线读取 On-Chip FLASH 数据，S 总线执行编程操作。On-Chip FLASH 内部有仲裁机制，优先 I/D 总线访问，S 总线需等待 I/D 总线空闲才能完成对应操作；一旦 S 总线获得访问 On-Chip FLASH 的权力，I/D 总线无法抢夺直至 S 总线访问完毕。

不同型号产品，编程地址空间不同，下表为不同组合下地址分配情况：

表 7-2 On-Chip FLASH 编程空间地址分配表

组合	MAIN 区域	ROM 区域
组合 1	0x0000 0000 - 0x0003 FFFF	0x0000 0000 - 0x0000 0000
组合 2	0x0000 0000 - 0x0002 FFFF	0x0003 0000 - 0x0003 FFFF
组合 3	0x0000 0000 - 0x0002 FFFF	0x0003 0000 - 0x0003 FFFF
组合 4	0x0000 0000 - 0x0001 FFFF	0x0002 0000 - 0x0003 FFFF
组合 5	0x0000 0000 - 0x0001 FFFF	0x0000 0000 - 0x0000 0000
组合 6	0x0000 0000 - 0x0000 FFFF	0x0003 0000 - 0x0003 FFFF
组合 7	0x0000 0000 - 0x0000 FFFF	0x0000 0000 - 0x0000 0000

编程操作，具体流程为：

- 系统寄存器 SYS_FLSP，写入密钥，解除编程保护
- On-Chip FLASH 控制寄存器 FSMC_ICFG，开启编程使能
- On-Chip FLASH 地址寄存器 FSMC_ADDR，写入编程地址
- On-Chip FLASH 编程数据寄存器 FSMC_WDAT0 至 FSMC_WDAT3，写入编程数据
- On-Chip FLASH 编程触发寄存器 FSMC_WDAT，写入触发值，触发编程
- 寄存器 FSMC_REDY.IREADY 位，指示当前 S 总线是否完成了当前编程访问操作，1 表示完成，0 表示未完成。

访问本控制器的寄存器，实现 On-Chip FLASH 编程操作的执行流程如下：

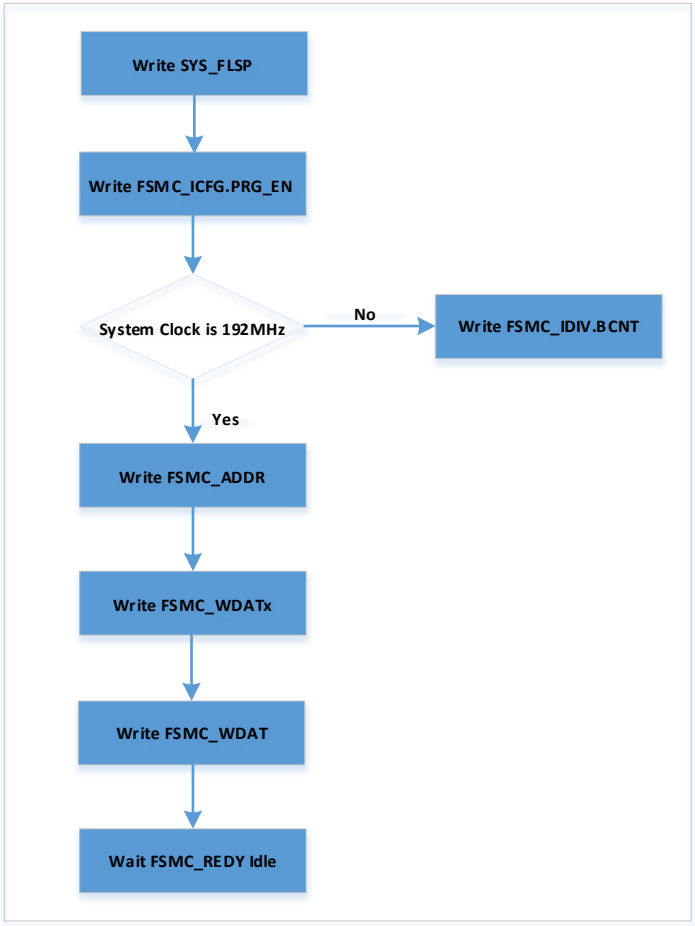


图 7-4 On-Chip FLASH 模块编程操作流程图

系统工作频率的判断，需要参考 SYS_CLK_CFG 的配置。On-Chip FLASH 编程/擦除操作的绝对时间是固定的，On-Chip FLASH 控制器需要基于绝对时间，保存对应频率下的计数值。FSMC_IDIV.BCNT0 和 FSMC_IDIV.BCNT1 默认值是 192MHz 时钟频率下的计数值；当芯片工作在其他频率下时，需要重新配置 FSMC_IDIV.BCNT0 和 FSMC_IDIV.BCNT1 的值，以实现 96MHz/48MHz 和 24MHz 的计数值（其它频率暂不支持）。最终保证计数值×时钟频率等于恒定的时间。不同频率下对应的 FSMC_IDIV.BCNT0 和 FSMC_IDIV.BCNT1 的值，请参考本章节寄存器描述。切记，FSMC_IDIV.BCNT0 和 FSMC_IDIV.BCNT1 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 On-Chip FLASH 编程/擦除失败。建议对 FSMC_IDIV 的操作，执行先读回，然后按照或/与的方法操作。

在执行 On-Chip FLASH 的编程/擦除操作时，FLASH 处于忙状态，若 Cache 不命中，无后续指令/数据取回 MCU，MCU 将暂停工作，等待 On-Chip FLASH 的编程/擦除操作完毕，才能继续执行程序。此时，为保证数据一致性，一旦 FLASH 执行了编程和擦除操作，Cache 所存数据均无效，需重新取指令和数据。

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 FSMC_ADDR 寄存器前，配置 FSMC_ICFG.ADR_INC，开启地址自动递增模式。当前的编程操作完毕后， FSMC_ADDR 的值，会硬件自动增加 0x10。连续读操作类似，但递增的值为 0x04。连续编程的流程如下：

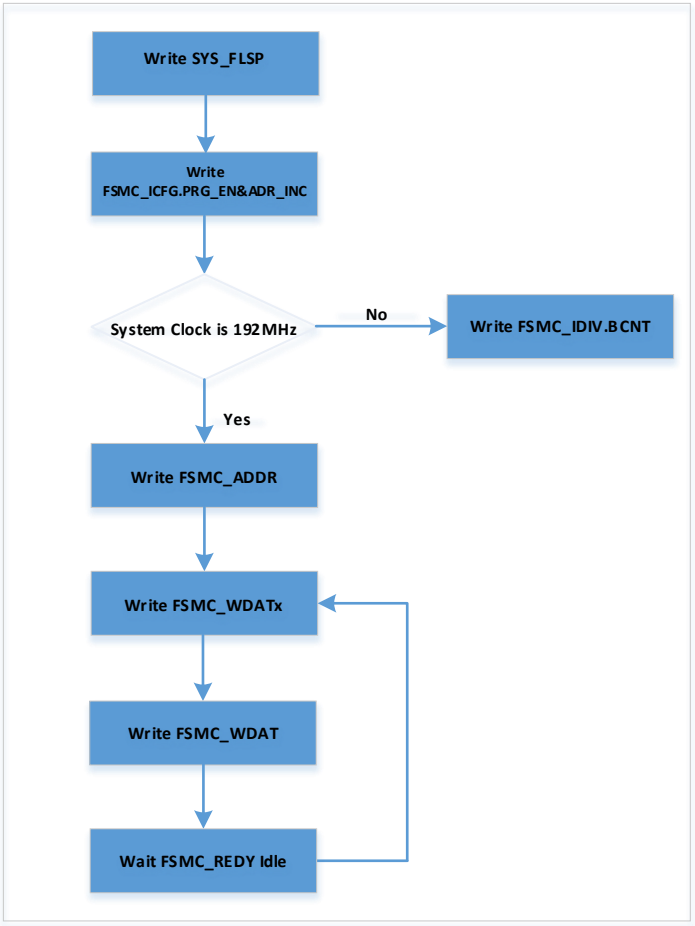


图 7-5 On-Chip FLASH 模块连续编程操作流程图

7.2.1.5 On-Chip FLASH 擦除操作

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

- 系统寄存器 SYS_FLSE，写入密钥，解除擦除保护
- On-Chip FLASH 擦除操作使能
- On-Chip FLASH 地址寄存器 FSMC_ADDR，写入擦除地址
- On-Chip FLASH 擦除寄存器 FSMC_ERAS，触发擦除操作
- 寄存器 FSMC_REDY.IREADY 位，指示当前 S 总线是否完成了当前擦除访问操作，1 表示完成，0 表示未完成。

类似读取操作。此时，存在 I/D 总线和 S 总线，同时访问 FLASH 的情况。I/D 总线读取 On-Chip FLASH 数据，S 总线执行擦除操作。On-Chip FLASH 内部有仲裁机制，优先 I/D 总线访问，S 总线需等待 I/D 总线空闲才能完成对应操作；一旦 S 总线获得访问 On-Chip FLASH 的权力，I/D 总线无法抢夺直至 S 总线访问完毕。

执行对 On-Chip FLASH 存储体的擦除操作。一个 Sector 的大小为 1024 字节。

下表为 MAIN 区域（最大情况）Sector 地址分配空间。

表 7-3 On-Chip FLASH MAIN 区域 Sector 地址分配表

块名	地址	尺寸(Byte)
Sector 0	0x0000 0000 - 0x0000 03FF	1024
Sector 1	0x0000 0400 - 0x0000 07FF	1024
Sector 2	0x0000 0800 - 0x0000 0BFF	1024
…	…	…
Sector256	0x0003 FC00 - 0x0003 FFFF	1024

ROM 区域不支持擦除；

NVR 区域实现 Sector 擦除；

表 7-4 On-Chip FLASH NVR 区域 Sector 地址分配表

块名	地址	尺寸(Byte)
Sector 0	0x0000 0000 - 0x0000 03FF	1024
Sector 1	0x0000 0400 - 0x0000 07FF	1024

MAIN 区域可以实现 Sector 擦除。

FLASH 擦除操作流程如下所示。

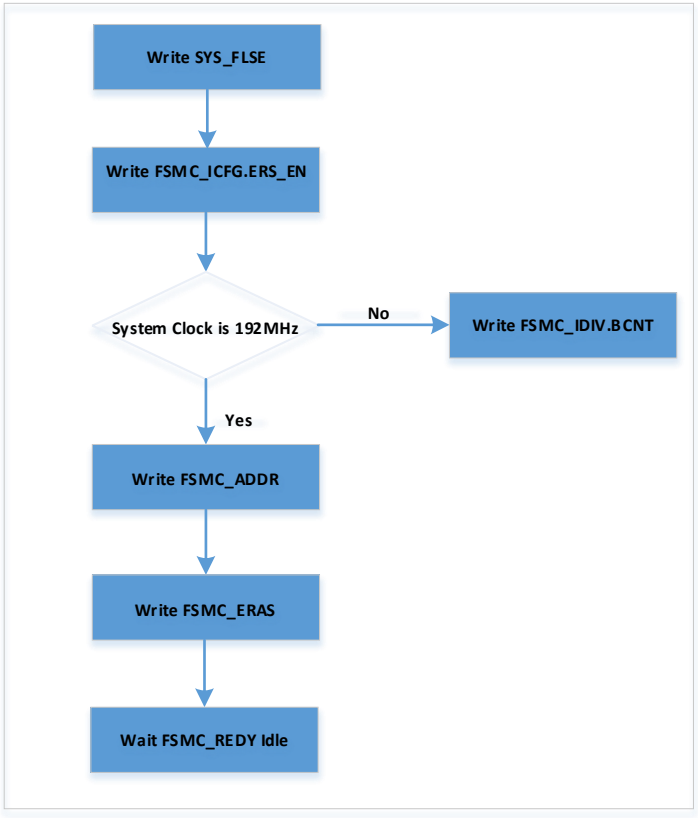


图 7-6 On-Chip FLASH 模块擦除操作流程图

Secotor 擦除，需要通过 FSMC_ADDR 确定哪个 Secotor 被擦除。FSMC_ERAS 写入 0x7654DCBA 触发擦除操作。

On-Chip FLASH 存储体，默认擦写次数为 100K 次。本模块支持快速擦除模块，理论上可以成倍增加擦除次数（最少时间实现正确擦除）。快速擦除模式，支持 Sector 擦除。相比正常擦除操作流程，快速擦除流程如下：

- 系统寄存器 SYS_FLSE，写入密钥，解除擦除保护
- On-Chip FLASH 擦除操作，配置为块擦除且使能快速擦除，快速擦除计数器配置为 0
- On-Chip FLASH 擦除操作使能
- On-Chip FLASH 地址寄存器 FSMC_ADDR，写入擦除地址
- On-Chip FLASH 擦除寄存器 FSMC_ERAS，触发擦除操作
- 寄存器 FSMC_REDY.IREADY 位，指示当前 S 总线是否完成了当前擦除访问操作，1 表示完成，0 表示未完成。
- On-Chip FLASH，用户执行读取当前擦除块地址对应的所有数据，判断是否都为 1。为全 1，说明快速擦除成功；不为全 1，说明快速擦除失败，快速擦除计数器软件加 1，再次触发擦除操作。如此反复直至擦除成功。

注，快速擦除计数器最大值为 3，执行完配置为 3 的擦除后，当前 Sector 地址对应的数据仍然非全 1，说明 On-Chip FLASH 已经到了使用极限。

7.2.1.6 On-Chip FLASH Cache 加速操作

因 On-Chip FLASH 存储体的速度限制，无法达到系统最快速度。当对 On-Chip FLASH 进行读取操作时，需要大于 1 个时钟周期才能完成。为了加快数据的读出，On-Chip FLASH 控制器增加了 Cache 功能。Cache 功能的开启和关闭，只需要设置 SYS_CACHE_CFG.EN 即可。

注，若执行了 On-Chip FLASH 存储体擦除/编程操作，Cache 模块并不会同步刷新同步，但 Cache 会将存入的数据，硬件会自动全配置为无效，保证不发生因两者存在潜在的数据失配导致的运行错误。

7.2.1.7 On-Chip FLASH 保护

On-Chip FLASH 存储体的保护策略分成两个部分：On-Chip FLASH 存储体（包括 MAIN、NVR 和 ROM）整体保护策略和 ROM 区域独有保护策略。On-Chip FLASH 存储体整体保护策略的开启和关闭，通过其保护字实现。保护字的状态为开启，On-Chip FLASH 存储体整体保护开启；保护字的状态为关闭，On-Chip FLASH 存储体整体保护关闭。ROM 区域的独有保护策略，主要针对 ROM 区域的读保护，On-Chip FLASH 存储体整体保护的开启和关闭，不影响 ROM 区域的独有保护策略的开启和关闭。保护策略的具体逻辑，如下：

- ROM 区域未执行编程操作，保护字未开启保护，MCU 和外界均可访问并读出 ROM 区域

- ROM 区域已执行编程操作，保护字未开启保护，MCU 和外界均可访问并读出 ROM 区域
- ROM 区域已执行编程操作，保护字开启保护，外界不可以访问并读出，MCU 不可以读出但可以执行 ROM 区域
- On-Chip FLASH 存储体整体保护，保护字未开启保护，MCU 和外界均可访问并读出到 On-Chip FLASH 存储体非 ROM 区域的内容；ROM 区域内容是否可被访问并被读出，取决于 ROM 区域的保护字的状态，但 MCU 可运行 ROM 区域应用程序
- On-Chip FLASH 存储体整体保护，保护字开启保护，外界无法访问并读出到 On-Chip FLASH 存储体的内容（包括 MAIN/ROM/NVR）；MCU 可访问并读出到 On-Chip FLASH 存储体非 ROM 区域的内容；ROM 区域内容是否可被访问并被读出，取决于 ROM 区域的保护字的状态，但 MCU 可运行 ROM 区域应用程序

On-Chip FLASH 存储体根据 MAIN 区域和 ROM 区域的尺寸，On-Chip FLASH 存储体（包括 MAIN、NVR 和 ROM）整体保护策略和 ROM 区域独有保护策略，各自对应的保护字所分配的地址略有不同。下表为保护字地址分配表。

表 7-5 On-Chip FLASH 保护字地址分配表

组合	整体保护策略_MAIN 区域			ROM 独有保护策略_ROM 区域		
	起始	结束	保护字地址	起始	结束	保护字地址
组合 1	0x0000 0000	0x0003 FFFF	0x0003 FFF0	0x0000 0000	0x0000 0000	无
组合 2	0x0000 0000	0x0002 FFFF	0x0002 FFF0	0x0003 0000	0x0003 FFFF	0x0003 FFF0
组合 3	0x0000 0000	0x0002 FFFF	0x0002 FFF0	0x0000 0000	0x0000 0000	无
组合 4	0x0000 0000	0x0001 FFFF	0x0001 FFF0	0x0002 0000	0x0003 FFFF	0x0003 FFF0
组合 5	0x0000 0000	0x0001 FFFF	0x0001 FFF0	0x0000 0000	0x0000 0000	无
组合 6	0x0000 0000	0x0000 FFFF	0x0000 FFF0	0x0003 0000	0x0003 FFFF	0x0003 FFF0
组合 7	0x0000 0000	0x0000 FFFF	0x0000 FFF0	0x0000 0000	0x0000 0000	无

若 On-Chip FLASH 存储体处于整体保护状态，用户可执行去保护操作，解除对 On-Chip FLASH 存储体内数据的保护（ROM 区域需考虑 ROM 区域的保护状态）。相反，若 On-Chip FLASH 存储体处于去保护状态，用户可执行保护操作，对 On-Chip FLASH 存储体的数据进行整体保护。默认情况下，On-Chip FLASH 存储体的数据处于整体保护状态。芯片上电复位后，硬件自动执行一次保护状态更新操作，是保护的话继续处于保护状态，否则变成去保护状态。

从上表可以看到，我们一般将当前组合下最后 4 个 WORD，分配为保护字。保护字的内容为全 1 时，表明对应策略处于去保护状态；保护字的内容被写为非全 1 时，表明对应策略处于保护状态。因此，若需要开启保护，执行对应保护字的编程，写入非全 1 的值，读取 FSMC_PROT 寄存器，即触发一次保护状态更新的操作，完成保护（读取 FSMC_PROT 返回值可知晓当前保护状态）。因两套保护策略，有两个保护字，我们需明确当前是哪个保护状态的更新。

表 7-6 On-Chip FLASH 保护字地址分配表

FSMC_ICFG.SEL_PROT	区域
0	整体保护策略区域
1	ROM 独有保护策略区域

当 FSMC_ICFG.PROT 配置为 0，读取 FSMC_PROT 寄存器，将完成 On-Chip FLASH 存储体处于整体保护状态的更新操作；当 FSMC_ICFG.PROT 配置为 1，读取 FSMC_PROT 寄存器，将完成 ROM 区域独有保护状态的更新操作。

7.2.2 Ext-Chip FLASH 控制器功能描述

LKS32MC45x 硬件实现 Ext-Chip FLASH 控制器，并不包含外挂 FLASH 存储体。外挂 FLASH 的地址映射空间起始于 0x0100_0000，结束于 0x01FF_FFFF，大小为 16MB。

Ext-Chip FLASH 控制器特性：

- 地址线宽度为 24 位，支持外挂 FLASH 容量范围：小于等于 16MB，大于 64KB。
- 仅 SPI 模式 3 的传输方式，即默认 SCLK 为高电平，下降沿发送数据，上升沿采样数据。
- 支持单线 SPI 传输，双线 SPI 传输和四线传输。
- 可通过 MCU 直接访问，也可以通过寄存器间接访问。

7.2.2.1 Ext-Chip FLASH 访问方式

Ext-Chip FLASH 有两种访问方式：直接访问和间接访问。所谓直接访问，就是 MCU 直接通过地址寻址到对应 Ext-Chip FLASH。间接访问，就是通过 Ext-Chip FLASH 控制器的寄存器访问。一般，读取/运行 Ext-Chip FLASH 的内容，可以使用直接访问方式；读取/编程 Ext-Chip FLASH 使用间接访问方式。

7.2.2.2 Ext-Chip FLASH 传输并行数

Ext-Chip FLASH 控制器支持三种访问并行数：单线、双线和四线。不同并行数量的切换，需要通过 Ext-Chip FLASH 控制器的寄存器发送配置命令，才能顺利切换。Ext-Chip FLASH 常见 PIN 复用方案如下：

表 7-7 Ext-Chip FLASH PIN 描述及同 LKS32MC45x 互联关系表

Symbol	Extension	Remarks	LKS45x
CS#		Chip select	EFLS_CSN
SO	SIO1	Serial data output for 1 x I/O Serial data input and output for 4 x I/O read mode	EFLS_DAT[1]
WP#	SIO2	Write protection active low Serial data input and output for 4 x I/O read mode	EFLS_DAT[2]
SI	SIO0	Serial data input for 1x I/O Serial data input and output for 4 x I/O read mode	EFLS_DAT[0]
SCLK		Serial interface clock input	EFLS_CLK
HOLD#/RESET#	SIO3	Hardware Reset Pin Active low or to pause the device without deselecting the device Serial data input and output for 4 x I/O read mode	EFLS_DAT[3]

默认情况下，使用单线模式，一根信号线发送，一根信号线接收。Ext-Chip FLASH 的 WP#带电阻下拉，防止误编程，HOLD#/RESET#带电阻上拉。LKS32MC45x 对应的信号，默认为输入，不影响复用功能。切换到其它两种模式，需要 Ext-Chip FLASH 控制器先发送命令，Ext-Chip FLASH 接收命令后，复用开启。Ext-Chip FLASH 实现模式切换，同时 Ext-Chip FLASH 控制器同步开启多线传输。

7.2.2.3 Ext-Chip FLASH 命令格式

一般情况下，Ext-Chip FLASH 命令由如下几个部分组成，但根据具体命令不同，某些部分无需发送，具体需参见 Ext-Chip FLASH 文档。Ext-Chip FLASH 控制器在发送某命令前，每个部分均需要配置，若某部分无需发送，对应部分的数据长度可配置为 0。具体命令组成如下：

- 指令部分（Command）
- 地址部分（Address）
- 模式部分（Mode）
- 冗余部分（Dummy）
- 数据部分（Data）

以 P25Q32H 为例子，提供如下几组配置，供参考。

7.2.2.3.1 Ext-Chip FLASH 间接方式四线读取

间接访问，主要配置 FSMC_UCFG 等寄存器。访问波形如下：

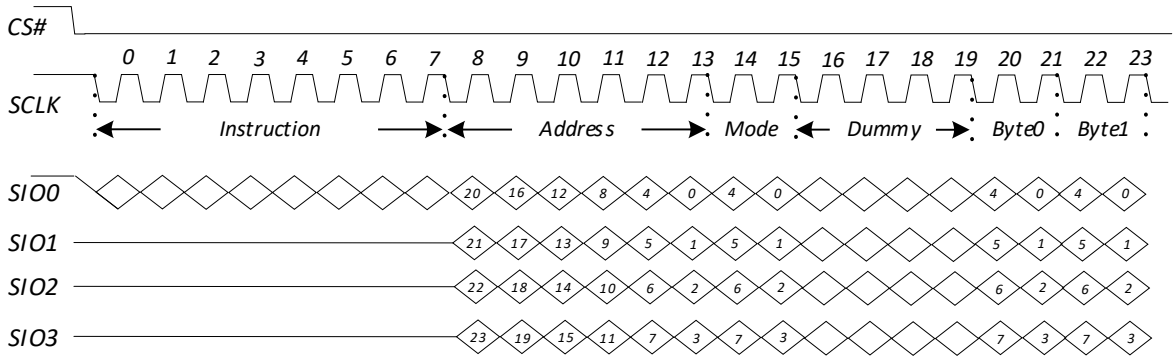


图 7-7 Ext-Chip FLASH 间接方式四线读取数据波形

所有部分均包含，除开命令外，其它部分均为四线模式。则 FSMC_UCFG 的配置为：

```
FSMC_UCFG = 0x3FC0_4123;
```

此处，FSMC_UCFG.DAT_LEN 配置为 4，表示读取四个字节的数据，可以配置为其它数值。

7.2.2.3.2 Ext-Chip FLASH 直接方式四线读取

直接访问，主要配置 FSMC_FCFG 等寄存器。访问波形如下：

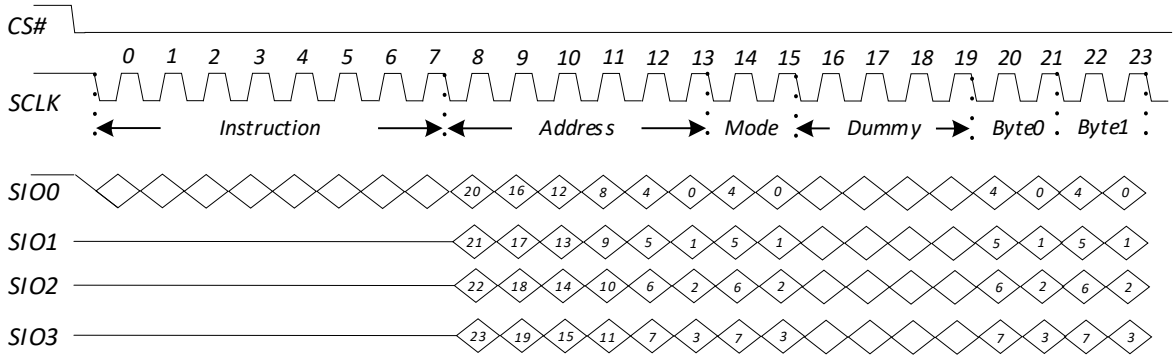


图 7-8 Ext-Chip FLASH 直接方式四线读取数据波形

所有部分均包含，除开命令外，其它部分均为四线模式。则 FSMC_FCFG 的配置为：

FSMC_FCFG = 0x03FC_0421;

此处，FSMC_FCFG.DAT_LEN 配置为 4，表示读取四个字节的数据，可以配置为其它数值。不过，因为 MCU 一次只需要 32-bit 数据，所以，直接访问的数据长度设置为 4，比较合理。

7.2.2.4 Ext-Chip FLASH 存储顺序

Ext-Chip FLASH 存储的最小单位是字节。字节内按照高低顺序存放。当发生读取操作时，按照地址递增的方式，逐一发出字节；接收方，若是间接方式访问，根据 FSMC_UCFG.ORDER 的配置情况，决定读回的顺序。

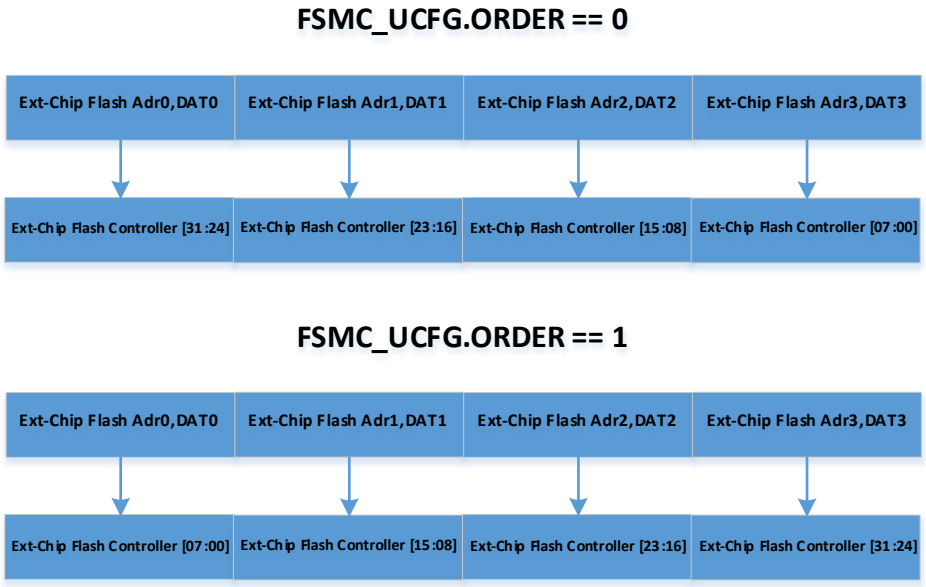


图 7-9 Ext-Chip FLASH 间接方式访问读回的数据顺序

若采用直接方式访问，情况类似，此时参考 FSMC_FCFG.ORDER 的值。

图 7-10 Ext-Chip FLASH 直接方式访问读回的数据顺序

7.2.2.5 DMA 传输

Ext-Chip FLASH 控制器支持 DMA 传输。间接方式和直接方式，DMA 传输方式不相同。

在间接方式下，配置 FSMC_UTRG.DMA_EN 为 1，即可开启 DMA 功能。传输数据的长度，为 4 的整数倍。即，每次完成 4 个字节的传输后，产生一次 DMA 传输请求。举例：

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

DMA_DADR4 = 0x20001000;// ram space

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

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

DMA_CCR4      = 0x00000A11;// word-alignment

DMA_CTRL       = 0x00000001;// enable DMA

                                                    //wait DMA interrupt
```

在直接方式下，无需使能 FSMC_UTRG.DMA_EN 控制位，使用 DMA 软件触发的方式，直接发起对 Ext-Chip FLASH 的读取访问请求。举例：

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

DMA_DADR4 = 0x20001000;// ram space

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

DMA_REN4     = 0x80000000;// soft trig
```

```
DMA_CCR4      = 0x00000AF1;// word-alignment
DMA_CTRL      = 0x00000001;// enable DMA
                                        //wait DMA interrupt
```

7.3 寄存器

7.3.1 地址分配

非易失存储体模块寄存器的基地址是 0x4001_A000 寄存器列表

表 7-8 非易失存储体模块寄存器列表

名称	偏移	说明
FSMC_UCMD	0x00	Ext-Chip FLASH 间接方式访问命令寄存器
FSMC_UMOD	0x04	Ext-Chip FLASH 间接方式访问模式寄存器
FSMC_UADR	0x08	Ext-Chip FLASH 间接方式访问地址寄存器
FSMC_UWDA	0x0C	Ext-Chip FLASH 间接方式访问编程数据寄存器
FSMC_URDA	0x10	Ext-Chip FLASH 间接方式访问读取数据寄存器
FSMC_UCFG	0x20	Ext-Chip FLASH 间接方式访问控制寄存器
FSMC_UTRG	0x24	Ext-Chip FLASH 间接方式访问操作控制寄存器
FSMC_FCMD	0x30	Ext-Chip FLASH 直接方式访问命令寄存器
FSMC_FMOD	0x34	Ext-Chip FLASH 直接方式访问模式寄存器
FSMC_FCFG	0x50	Ext-Chip FLASH 直接方式访问控制寄存器
FSMC_FTRG	0x54	Ext-Chip FLASH 直接方式访问操作寄存器
FSMC_EDIV	0x60	Ext -Chip FLASH 波特率寄存器
FSMC_WDAT0	0x70	On-Chip FLASH 编程数据寄存器，BIT[031:000]
FSMC_WDAT1	0x74	On-Chip FLASH 编程数据寄存器，BIT[063:032]
FSMC_WDAT2	0x78	On-Chip FLASH 编程数据寄存器，BIT[095:064]
FSMC_WDAT3	0x7C	On-Chip FLASH 编程数据寄存器，BIT[127:096]
FSMC_ICFG	0x80	On-Chip FLASH 配置寄存器
FSMC_ADDR	0x84	On-Chip FLASH 地址寄存器
FSMC_WDAT	0x88	On-Chip FLASH 编程操作触发寄存器
FSMC_RDAT	0x8C	On-Chip FLASH 读数据寄存器
FSMC_ERAS	0x90	On-Chip FLASH 擦除操作触发寄存器
FSMC_PROT	0x94	On-Chip FLASH 保护状态寄存器
FSMC_REDY	0x98	On-Chip FLASH 工作状态寄存器
FSMC_IDIV	0x9C	On-Chip FLASH 时基寄存器

7.3.2 FSMC_UCMD Ext-Chip FLASH 间接方式访问命令寄存器

地址：0x4001_A000

复位值：0x0

表 7-9 Ext-Chip FLASH 间接方式访问命令寄存器 FSMC_UCMD

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

位置	位名称	说明
[7:0]	UCMD	Ext-Chip FLASH 间接方式访问命令寄存器

7.3.3 FSMC_UMOD Ext-Chip FLASH 间接方式访问模式寄存器

地址：0x4001_A004

复位值：0x0

表 7-10 Ext-Chip FLASH 间接方式访问模式寄存器 FSMC_UMOD

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

位置	位名称	说明
[7:0]	UMOD	Ext-Chip FLASH 间接方式访问模式寄存器

7.3.4 FSMC_UADR Ext-Chip FLASH 间接方式访问地址寄存器

地址：0x4001_A008

复位值：0x0

表 7-11 Ext-Chip FLASH 间接方式访问地址寄存器 FSMC_UADR

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

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

0

位置	位名称	说明
[23:0]	FSMC_UADR	Ext-Chip FLASH 间接方式访问地址寄存器

7.3.5 FSMC_UWDA Ext-Chip FLASH 间接方式访问编程数据寄存器

地址：0x4001_A00C

复位值：0x0

表 7-12 Ext-Chip FLASH 间接方式访问编程数据寄存器 FSMC_UWDA

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

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

位置	位名称	说明
[31:0]	FSMC_UWDA	Ext-Chip FLASH 间接方式访问编程数据寄存器

7.3.6 FSMC_URDA Ext-Chip FLASH 间接方式访问读取数据寄存器

地址：0x4001_A010

复位值：0x0

表 7-13 Ext-Chip FLASH 间接方式访问读取数据寄存器 FSMC_URDA

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

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

位置	位名称	说明
[31:0]	FSMC_URDA	Ext-Chip FLASH 间接方式访问读取数据寄存器

7.3.7 FSMC_UCFG Ext-Chip FLASH 间接方式访问控制寄存器

地址：0x4001_A020

复位值：0x0

表 7-14 Ext-Chip FLASH 间接方式访问控制寄存器 FSMC_UCFG

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

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

位置	位名称	说明
31	ORDER	直接方式访问，读取数据大小头设置 0：Ext-Chip FLASH 存储的数据，按照大头存放 1：Ext-Chip FLASH 存储的数据，按照小头存放
30	--	保留位，固定配置为 0
[29:28]	DAT_SPD	数据部分并行发送配置 00：一根线传输 01：两根线传输 11：四根线传输 10：非法配置
[27:26]	DMY_SPD	冗余部分并行发送配置 00：一根线传输 01：两根线传输 11：四根线传输 10：非法配置
[25:24]	MOD_SPD	模式部分并行发送配置 00：一根线传输 01：两根线传输 11：四根线传输 10：非法配置
[23:22]	ADR_SPD	地址部分并行发送配置 00：一根线传输

		01：两根线传输 11：四根线传输 10：非法配置
[21:20]	CMD_SPD	指令部分并行发送配置 00：一根线传输 01：两根线传输 11：四根线传输 10：非法配置
[19:12]	DAT_LEN	数据部分数据长度，字节为最小单位，0 代表无此部分，1 代表 1 个字节，一次类推，最大 15 个字节
[11:8]	DMY_LEN	冗余部分数据长度，字节为最小单位，0 代表无此部分，1 代表 1 个字节，一次类推，最大 15 个字节
[7:4]	MOD_LEN	模式部分数据长度，字节为最小单位，0 代表无此部分，1 代表 1 个字节，一次类推，最大 15 个字节
[3:0]	--	保留位，固定配置为 0

7.3.8 FSMC_UTRG Ext-Chip FLASH 间接方式访问操作寄存器

地址：0x4001_A024

复位值：0x0

表 7-15 Ext-Chip FLASH 间接方式访问操作寄存器 FSMC_UTRG

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

位置	位名称	说明
[31:3]	--	保留位，固定配置为 0
[2]	DMA_EN	Ext-Chip FLASH 控制器使能 DMA 搬移操作
[1]	OP	通过 Ext-Chip FLASH 控制器间接方式访问 Ext-Chip FLASH，操作类型 1：编程/擦除 0：读取
[0]	SEL	通过 Ext-Chip FLASH 控制器间接方式访问 Ext-Chip FLASH。同时，此位也是状态位。 1：触发操作，处理中 0：不触发操作，空闲

7.3.9 FSMC_FCMD Ext-Chip FLASH 直接方式访问命令寄存器

地址：0x4001_A030

复位值：0x0

表 7-16 Ext-Chip FLASH 直接方式访问命令寄存器 FSMC_FCMD

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

位置	位名称	说明
[7:0]	FCMD	Ext-Chip FLASH 直接方式访问命令寄存器

7.3.10 FSMC_FMOD Ext-Chip FLASH 直接方式访问模式寄存器

地址：0x4001_A034

复位值：0x0

表 7-17 Ext-Chip FLASH 直接方式访问模式寄存器 FSMC_FMOD

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

位置	位名称	说明
[7:0]	FMOD	Ext-Chip FLASH 直接方式访问模式寄存器

7.3.11 FSMC_FCFG Ext-Chip FLASH 直接方式访问控制寄存器

地址：0x4001_A050

复位值：0x400

表 7-18 Ext-Chip FLASH 直接方式访问控制寄存器 FSMC_FCFG

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

	0		0	0	0	0	0

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

位置	位名称	说明
27	ORDER	直接方式访问，读取数据大小头设置 0：Ext-Chip FLASH 存储的数据，按照大头存放 1：Ext-Chip FLASH 存储的数据，按照小头存放
26	--	保留位，固定配置为 0
[25:24]	DAT_SPD	数据部分并行发送配置 00：一根线传输 01：两根线传输 11：四根线传输 10：非法配置
[23:22]	DMY_SPD	冗余部分并行发送配置 00：一根线传输 01：两根线传输 11：四根线传输 10：非法配置
[21:20]	MOD_SPD	模式部分并行发送配置 00：一根线传输 01：两根线传输 11：四根线传输 10：非法配置
[19:18]	ADR_SPD	地址部分并行发送配置 00：一根线传输 01：两根线传输 11：四根线传输 10：非法配置
[17:16]	CMD_SPD	指令部分并行发送配置 00：一根线传输 01：两根线传输 11：四根线传输 10：非法配置
[15:8]	DAT_LEN	数据部分数据长度，字节为最小单位，0 代表无此部分，1 代表 1 个字节，一次类推，最大 15 个字节
[7:4]	DMY_LEN	冗余部分数据长度，字节为最小单位，0 代表无此部分，1 代表 1 个字节，一次类推，最大 15 个字节
[3:0]	MOD_LEN	模式部分数据长度，字节为最小单位，0 代表无此部分，1 代表 1 个字节，一次类推，最大 15 个字节

7.3.12 FSMC_FTRG Ext-Chip FLASH 直接方式访问操作寄存器

地址：0x4001_A054

复位值：0x0

表 7-19 Ext-Chip FLASH 直接方式访问操作寄存器 FSMC_FTRG

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

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

位置	位名称	说明
[31:1]	--	保留位，固定配置为 0
[0]	UPDATE	Ext-Chip FLASH 控制器直接方式访问寄存器更新。因存在可能 MCU 一直使用直接方式获取 Ext-Chip FLASH 的数据，为了安全考虑，需要在空闲期间更新直接访问相关寄存器，此位也相当于一个状态，表明是否更新完毕。 1：触发更新操作/更新过程中 0：不触发更新操作/更新完毕

7.3.13 FSMC_EDIV Ext-Chip FLASH 波特率寄存器

地址：0x4001_A060

复位值：0x44444

表 7-20 Ext-Chip FLASH 波特率寄存器 FSMC_EDIV

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

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

位置	位名称	说明
[19:18]	DAT_BAUDL	数据部分波特率配置，低电平持续周期数 0：一个系统时钟周期 1：两个系统时钟周期 2：三个系统时钟周期 3：四个系统时钟周期
[17:16]	DAT_BAUDH	数据部分波特率配置，高电平持续周期数 0：一个系统时钟周期 1：两个系统时钟周期 2：三个系统时钟周期 3：四个系统时钟周期
[15:14]	DMY_BAUDL	冗余部分波特率配置，低电平持续周期数 0：一个系统时钟周期 1：两个系统时钟周期 2：三个系统时钟周期 3：四个系统时钟周期
[13:12]	DMY_BAUDH	冗余部分波特率配置，高电平持续周期数 0：一个系统时钟周期 1：两个系统时钟周期 2：三个系统时钟周期 3：四个系统时钟周期
[11:10]	MOD_BAUDL	模式部分波特率配置，低电平持续周期数 0：一个系统时钟周期 1：两个系统时钟周期 2：三个系统时钟周期 3：四个系统时钟周期
[09:08]	MOD_BAUDH	模式部分波特率配置，高电平持续周期数 0：一个系统时钟周期 1：两个系统时钟周期 2：三个系统时钟周期 3：四个系统时钟周期
[07:06]	ADR_BAUDL	地址部分波特率配置，低电平持续周期数 0：一个系统时钟周期 1：两个系统时钟周期 2：三个系统时钟周期 3：四个系统时钟周期
[05:04]	ADR_BAUDH	地址部分波特率配置，高电平持续周期数 0：一个系统时钟周期 1：两个系统时钟周期 2：三个系统时钟周期 3：四个系统时钟周期
[03:02]	CMD_BAUDL	指令部分波特率配置，低电平持续周期数 0：一个系统时钟周期

		1：两个系统时钟周期 2：三个系统时钟周期 3：四个系统时钟周期
[01:00]	CMD_BAUDH	指令部分波特率配置，高电平持续周期数 0：一个系统时钟周期 1：两个系统时钟周期 2：三个系统时钟周期 3：四个系统时钟周期

注，波特率为 L 系统时钟周期和 H 系统时钟周期之和。

7.3.14 FSMC_WDAT0 On-Chip FLASH 编程数据寄存器

地址：0x4001_A070

复位值：0x0

表 7-21On-Chip FLASH 编程数据寄存器 FSMC_WDAT0

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

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

位置	位名称	说明
[31:0]	WDATA	On-Chip FLASH 编程数据寄存器，BIT[031:000]

7.3.15 FSMC_WDAT1 On-Chip FLASH 编程数据寄存器

地址：0x4001_A074

复位值：0x0

表 7-22On-Chip FLASH 编程数据寄存器 FSMC_WDAT1

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

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

位置	位名称	说明
[31:0]	WDATA	On-Chip FLASH 编程数据寄存器，BIT[063:032]

7.3.16 FSMC_WDAT2 On-Chip FLASH 编程数据寄存器

地址：0x4001_A078

复位值：0x0

表 7-23On-Chip FLASH 编程数据寄存器 FSMC_WDAT2

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

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

位置	位名称	说明
[31:0]	WDATA	On-Chip FLASH 编程数据寄存器，BIT[095:064]

7.3.17 FSMC_WDAT3 On-Chip FLASH 编程数据寄存器

地址：0x4001_A07C

复位值：0x0

表 7-24On-Chip FLASH 编程数据寄存器 FSMC_WDAT3

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

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

0

位置	位名称	说明
[31:0]	WDATA	On-Chip FLASH 编程数据寄存器，BIT[127:096]

7.3.18 FSMC_ICFG On-Chip FLASH 配置寄存器

地址：0x4001_A080

复位值：0x60

表 7-25On-Chip FLASH 配置寄存器 FSMC_ICFG

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

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

位置	位名称	说明
[31]	ERS_EN	On-Chip FLASH 擦除使能。默认为 0。 0：关闭擦除 1：开启擦除
[27]	PRG_EN	On-Chip FLASH 编程使能。默认为 0。 0：关闭编程 1：开启编程
[23]	ADR_INC	On-Chip 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 On-Chip FLASH 连续读写访问时，开启此功能，可以减少对地址的操作。
[19]	SEL_PROT	On-Chip FLAS 保护字选择 0：选择 On-Chip FLASH 存储体整体保护策略的保护字 1：选择 ROM 区域独有保护策略的保护字
[11]	REGION	访问 On-Chip FLASH 区域选择。默认为 0。

		0：MAIN/ROM（根据地址不同选择不同区域） 1：NVR
[7]	RETRY_EN	快速擦除使能 0：关闭 1：使能
[3:2]	RETRY	快速擦除计数器

7.3.19 FSMC_ADDR On-Chip FLASH 地址寄存器

地址：0x4001_A084

复位值：0x0

表 7-26On-Chip FLASH 地址寄存器 FSMC_ADDR

17	16	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
[31:16]		未使用
[17:0]	ADDR	地址寄存器。读/写/擦除操作对应的地址寄存器。 读取操作，一次读取 32-BIT 数据，ADDR 的最低两位会被 On-Chip FLASH 控制器忽略。 编程操作，一次编程 128-BIT 数据，ADDR 的最低四位会被 On-Chip FLASH 控制器忽略。 擦除操作，Sector 大小为 1024-Byte，执行 Sector 擦除时，ADDR 的最低十位会被 On-Chip FLASH 控制器忽略。执行 Full 擦除时，ADDR 的值被 On-Chip FLASH 控制器忽略。

7.3.20 FSMC_WDAT On-Chip FLASH 编程操作触发寄存器

地址：0x4001_A088

复位值：0x0

表 7-27On-Chip FLASH 编程操作触发寄存器 FSMC_WDAT

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

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

WO
0

位置	位名称	说明
[31:0]	WDATA	写入 0x7654DCBA，触发 On-Chip FLASH 编程操作

7.3.21 FSMC_RDAT On-Chip FLASH 读数据寄存器

地址：0x4001_ A08C

复位值：0x0

表 7-28On-Chip FLASH 读数据寄存器 FSMC_RDAT

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

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

位置	位名称	说明
[31:0]	RDATA	执行读取操作，读出 On-Chip FLASH 对应地址的值

7.3.22 FSMC_ERAS On-Chip FLASH 擦除操作触发寄存器

地址：0x4001_ A090

复位值：0x0

表 7-29On-Chip FLASH 擦除控制寄存器 FSMC_ERAS

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

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

位置	位名称	说明
[31:0]	ERASE	写入 0x7654DCBA，触发 On-Chip FLASH 擦除操作

7.3.23 FSMC_PROT On-Chip FLASH 保护状态寄存器

地址：0x4001_ A094

复位值：0x0808

表 7-30On-Chip FLASH 保护状态寄存器 FSMC_PROT

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

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

位置	位名称	说明
[16]	RPROT	ROM 区域独有保护策略的保护状态 0：未保护 1：保护中
[0]	PROT	On-Chip FLASH 存储体整体保护策略的保护状态 0：未保护 1：保护中

7.3.24 FSMC_REDY On-Chip FLASH 工作状态寄存器

地址：0x4001_ A098

复位值：0x00000101

表 7-31On-Chip FLASH 工作状态寄存器 FSMC_REDY

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

位置	位名称	说明
[31:9]		未使用

[8]	IREADY	S 总线访问 On-Chip FLASH 状态 0：S 总线在访问 On-Chip FLASH，不接受 S 总线的发起新请求 1：S 总线未访问 On-Chip FLASH，可接受 S 总线的发起新请求
[7:1]		未使用
[0]	READY	I/D 总线访问 On-Chip FLASH 状态 0：I/D 总线在访问 On-Chip FLASH，不接受 I/D 总线的发起新请求 1：I/D 总线未访问 On-Chip FLASH，可接受 I/D 总线的发起新请求

注，S 总线是外设访问总线，通过寄存器方式间接访问 On-Chip FLASH，支持读取/编程/擦除；I/D 总线是 MCU 直接访问 On-Chip FLASH 的总线，只支持读取；

7.3.25 FSMC_IDIV On-Chip FLASH 时基寄存器

地址：0x4001_ A09C

复位值：0x0

表 7-32On-Chip FLASH 工作状态寄存器 FSMC_IDIV

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

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

位置	位名称	说明
[30:16]	BCNT1	On-Chip FLASH 编程擦除时基寄存器 1。不同系统频率，值可以微调。默认为 0x5800，对应 192M 系统时钟。 192M：0x5800 096M：0x2C00 048M：0x1600
[14:08]	BCNT0	On-Chip FLASH 编程擦除时基寄存器 0。不同系统频率，值可以微调。默认为 0x40，对应 192M 系统时钟。一般不同频率调整
[07:04]	CKL	On-Chip FLASH，时钟信号低电平持续时间 0x1：仅当系统频率低于 96M，可使用的值 0x2：默认值 其它值，均为非法值
[03:00]	CKH	On-Chip FLASH，时钟信号高电平持续时 0x1：仅当系统频率低于 96M，可使用的值

		0x2：默认值 其它值，均为非法值

8 DMA

8.1 概述

DMA 和 CPU 同为芯片总线的主设备。

如图 8-1 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。

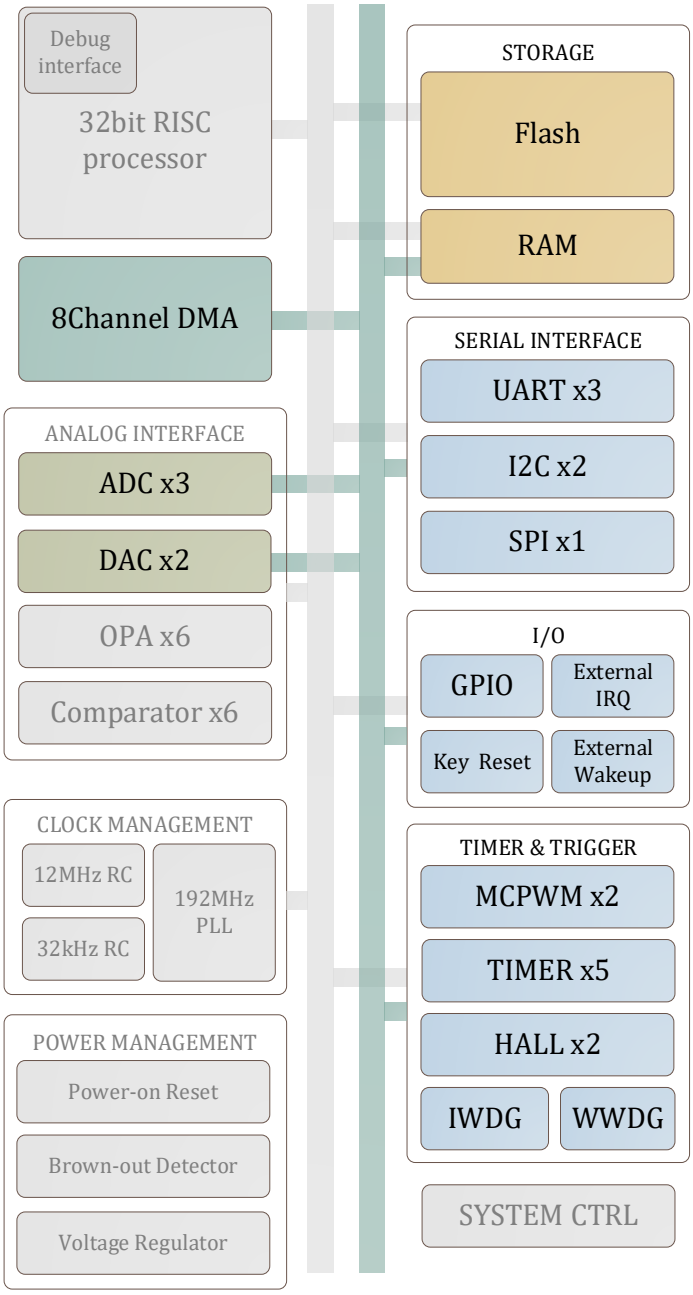


图 8-1 DMA AHB 总线架构

DMA 有 8 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一轮搬运，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 的传输分为多个轮次，根据 DMA_CCRx.RMODE 设置，可以每轮传输一次传输多轮，也可以只传输一轮但是连续进行多次数据搬运。

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。

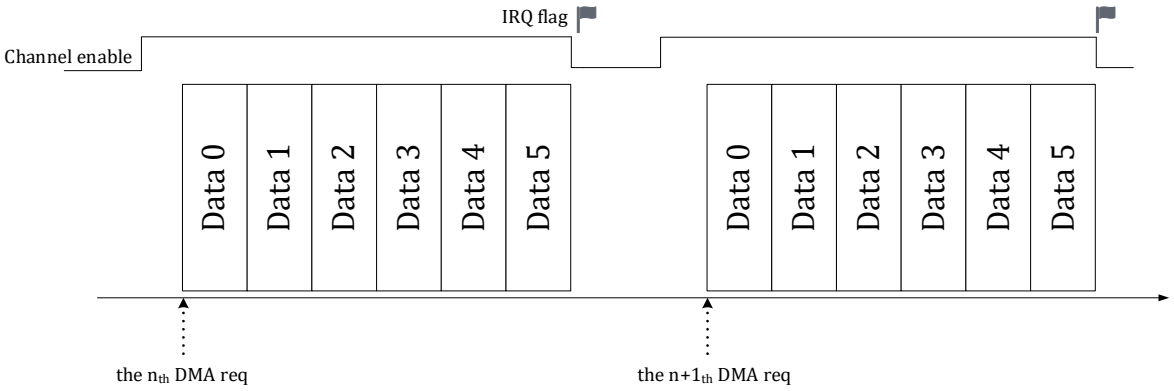


图 8-2 RMODE=0 DMA 传输情况

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。

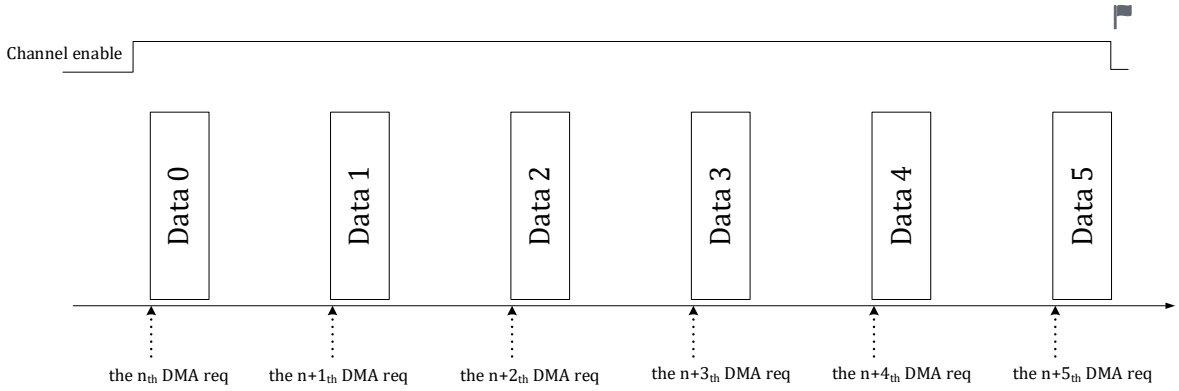


图 8-3 RMODE=1 DMA 传输情况

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，所以通常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。下图以源地址为例，说明了 SINC 为不同配置时，DMA 访问数据源地址的变化情况，目的地址同理。配置为一轮传输 4 次，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。

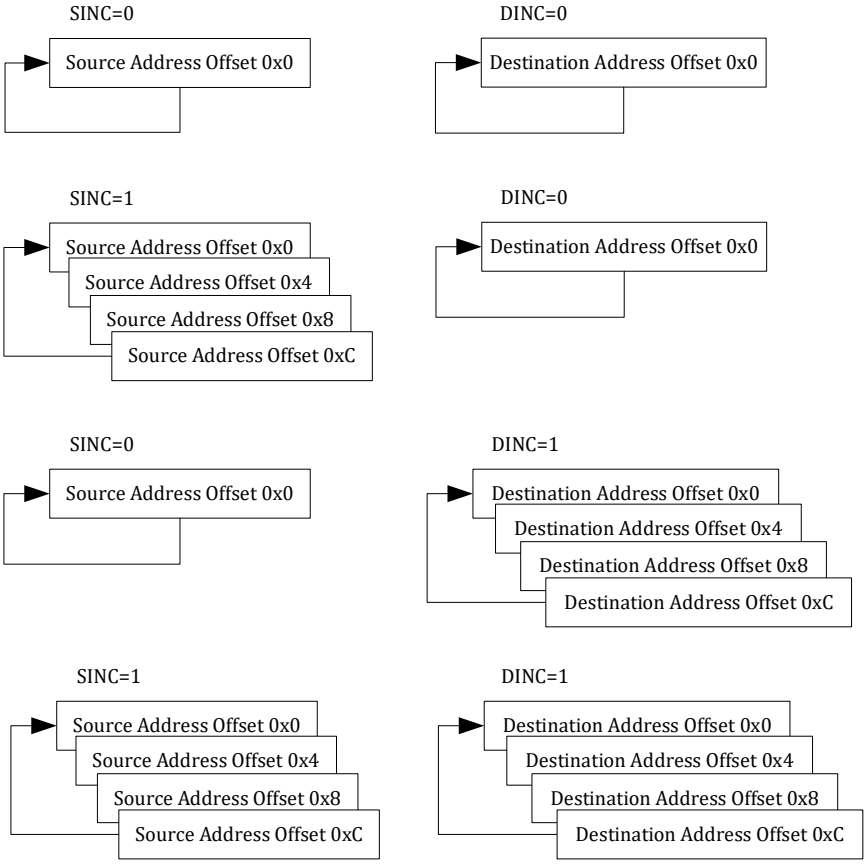


图 8-4 DMA 地址递增控制

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。

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 8 个通道对应的使能 DMA_CCRx.EN），此时 DMA 时钟被门控关闭。DMA 包含配置寄存器（可以被 CPU 配置）和数据搬运模块（向总线发起各种设备访问请求）。

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 支持循环模式，由 DMA_CCRx.CIRC 控制，通常循环模式和 DMA_CCRx.RMODE 配合使用。

如果配置 DMA_CCRx.CIRC=1，且 DMA_CCRx.RMODE=1，即循环多轮模式。一次 DMA 请求信号进行触发，只进行一次数据搬运（如 UART 数据搬运至 RAM，完成一轮搬运等待下一次请求。在此模式下，DMA_CTMSx 可以配置为一个较大的值，每进行一次搬运 DMA_CTMSx 减 1。由于循环模式 DMA 进行多轮搬运不再产生中断标志，CPU 可以通过读取 DMA_CTMSx 来判断当前已经传输的数据量，如果已经足够则开始处理。当传输了 DMA_CTMSx 轮以后，地址会回到初始地址。DMA_CTMSx 会重新装载为预设值，开始新一轮的递减。循环多轮模式下，DMA 的通道使能 DMA_CCRx.EN 位由

软件置位，一直为高，即使 DMA_CTMSx 递减至 0。如果要退出传输，可以软件清零 DMA_CCRx.EN。

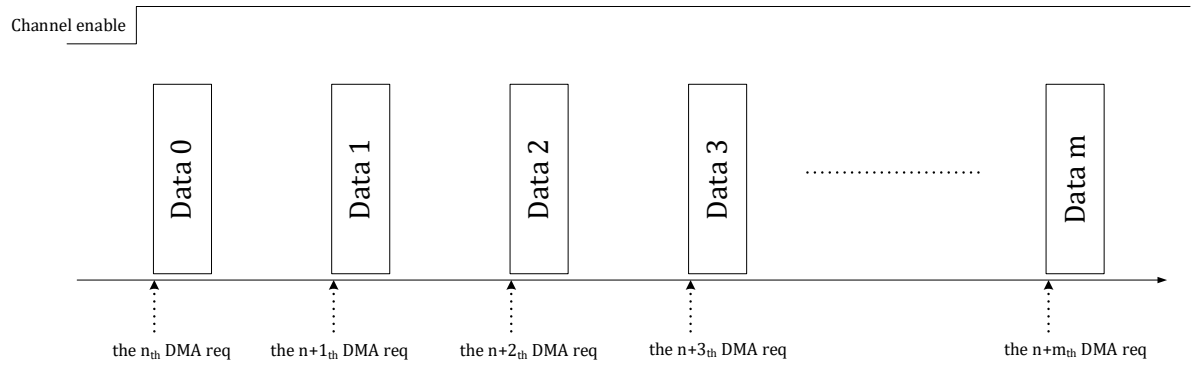


图 8-5 CIRC=1 且 RMODE=1 DMA 传输情况

如果配置 DMA_CCRx.CIRC=1，且 DMA_CCRx.RMODE=0，即循环单轮模式则 DMA 收到一次请求后会连续地进行 DMA_CTMSx 次数据搬运，并产生中断标志。在此模式下，DMA_CCRx.EN 无须软件置位，当 DMA 通道收到请求时会自动置位，传输完成一轮后自动清零。如果要退出此种模式，可以清零 DMA_CCRx.CIRC 位，并清零 DMA_CCRx.EN 位。

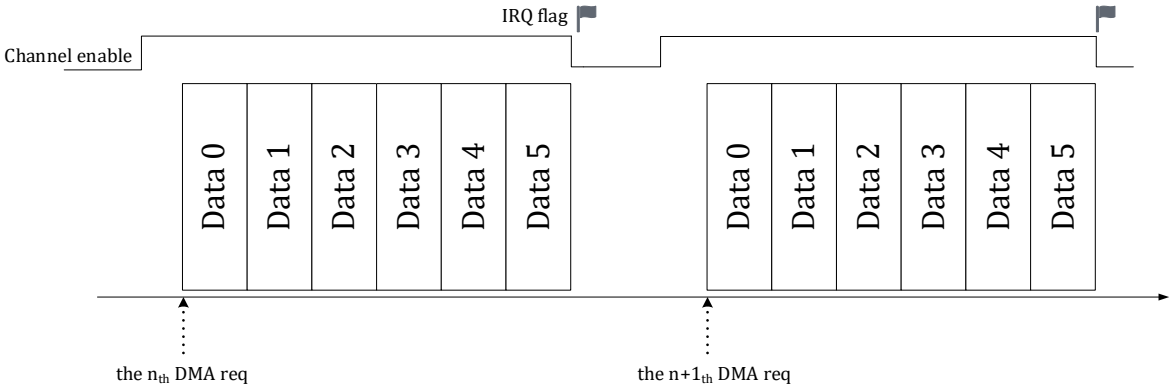


图 8-6 CIRC=1 且 RMODE=0 DMA 传输情况

循环模式下，DMA_CTMSx 寄存器随着搬运递减为 0 后重新装载为初始配置值。如果是搬运数据到内存，则覆盖之前搬运至内存的数据；如果是搬运至外设，则重复一轮数据发射。以将 ADC 数据搬运至内存为例，设定数据块大小为 16bit×12channel×8 次，则在循环模式下 DMA 完成一轮 96 个 half word 搬运后重新开始下一轮搬运，并覆盖之前使用的内存地址，不设置 DMA 完成中断标志位。单次模式下，DMA 完成一定大小的数据块搬运后即终止本次 DMA 操作，设置当前通道 DMA 完成中断标志位，同时硬件自动关闭对应的 DMA 通道，即硬件电路自动在该通道传输完成后将 DMA_CCRx.EN 设为 0。DMA 需要搬运的轮次由 DMA_CTMS 寄存器进行控制。

8.2 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 通道受理握手后会被 DMA 硬件清零，无法软件清除事件标志。

表 8-1 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 采用完成中断，使中断不再被 CPU 响应。
Timer	Timer 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）
SPI	SPI 模块使用接收缓冲区满事件作为 DMA 请求信号，由于 SPI 是同时收发，所以接收缓冲区满既是接收完毕也是发送完毕的事件标志。读取 SPI FIFO 自动清除事件标志。
MCPWM	MCPWM 模块使用过零/计数周期结束/4 个 ADC 触发信号作为 DMA 请求，具体 DMA 操作根据配置寄存器设定
I2C	I2C 模块使用 I2C0_SCR.BYTE_CMPLT 即字节发送完成作为触发 DMA 请求，DMA 自动清除 I2C 的请求标志。其他 I2C 中断事件仍由 CPU 响应
UART	串口模块使用 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 发送完成事件产生 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。
CAN	CAN 中断可以作为 DMA 请求
HALL	HALL 中断可以作为 DMA 请求
CMP	CMP 中断可以作为 DMA 请求
FMAC	当设置 FMAC_CR.DMAWEN=1，则当 X1 buffer 有足够空闲空间时触发 DMA 向 FMAC 写入待滤波数据；当设置 FMAC_CR.DMAREN=1，则 Y buffer 有足够有效数据时触发 DMA 从 FMAC 读取滤波的数据。具体多少数据触发 DMA 进行搬运取决于 FMAC 的水线设置
GPIO	GPIO 中断可以作为 DMA 请求

8.3 优先级

DMA 的优先级采用固定优先级，优先级如图 8-7 所示。为避免出现来不及响应某些外设请求的情况，在设计应用软件时应考虑任务实时性，每个通道不配置搬运过于大量的数据导致其他通道得不到及时响应。

如图 8-7 所示，优先级由上至下降低。8 个 DMA 通道中，优先级关系为：通道 0>通道 1>通道 2>...>通道 7 (>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。同一个通道内多个硬件请求事件优先级相同。通常，一个 DMA 通道配置使能一个硬件事件请求，多个硬件请求不应在一个通道内同时发生。

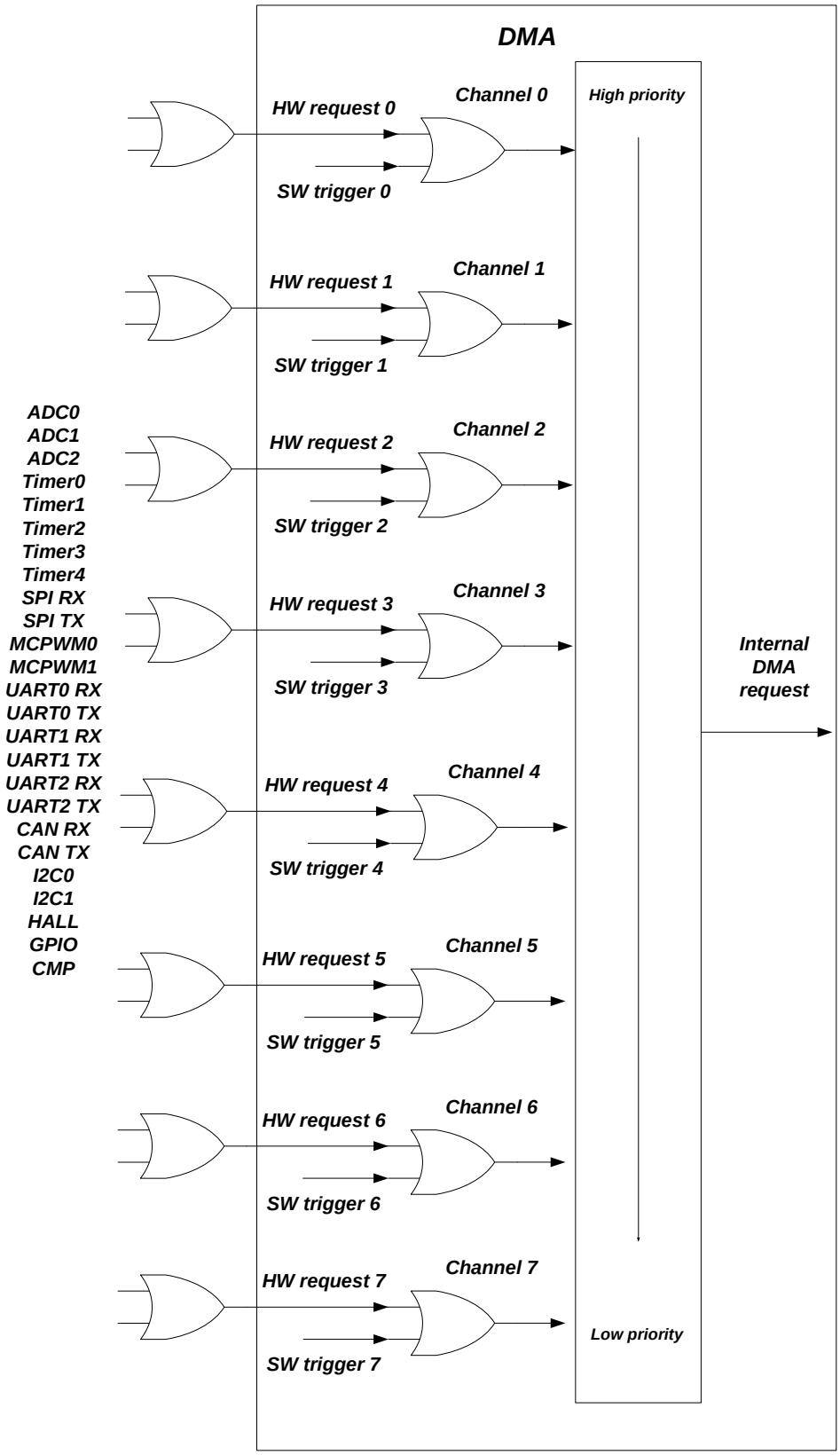


图 8-7 DMA 通道优先级

8.4 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，每搬运一个 byte 重新仲裁；对于 SPI/I2C 每搬运完一个 FIFO 的数据，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。

8.5 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。

8.6 寄存器

8.6.1 地址分配

DMA 控制器模块寄存器的基地址是 0x4001_9C00，寄存器列表如下：

表 8-2 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
	0x14	
DMA_CCR1	0x18	DMA 通道 1 通道配置寄存器
DMA_REN1	0x1C	DMA 通道 1 请求使能寄存器
DMA_CTMS1	0x20	DMA 通道 1 传输次数寄存器
DMA_SADR1	0x24	DMA 通道 1 源地址寄存器
DMA_DADR1	0x28	DMA 通道 1 目的地址寄存器
	0x2C	
DMA_CCR2	0x30	DMA 通道 2 通道配置寄存器
DMA_REN2	0x34	DMA 通道 2 请求使能寄存器
DMA_CTMS2	0x38	DMA 通道 2 传输次数寄存器
DMA_SADR2	0x3C	DMA 通道 2 源地址寄存器
DMA_DADR2	0x40	DMA 通道 2 目的地址寄存器

	0x44	
DMA_CCR3	0x48	DMA 通道 3 通道配置寄存器
DMA_REN3	0x4C	DMA 通道 3 请求使能寄存器
DMA_CTMS3	0x50	DMA 通道 3 传输次数寄存器
DMA_SADR3	0x54	DMA 通道 3 源地址寄存器
DMA_DADR3	0x58	DMA 通道 3 目的地址寄存器
	0x5C	
DMA_CCR4	0x60	DMA 通道 4 通道配置寄存器
DMA_REN4	0x64	DMA 通道 4 请求使能寄存器
DMA_CTMS4	0x68	DMA 通道 4 传输次数寄存器
DMA_SADR4	0x6C	DMA 通道 4 源地址寄存器
DMA_DADR4	0x70	DMA 通道 4 目的地址寄存器
	0x74	
DMA_CCR5	0x78	DMA 通道 5 通道配置寄存器
DMA_REN5	0x7C	DMA 通道 5 请求使能寄存器
DMA_CTMS5	0x80	DMA 通道 5 传输次数寄存器
DMA_SADR5	0x84	DMA 通道 5 源地址寄存器
DMA_DADR5	0x88	DMA 通道 5 目的地址寄存器
	0x8C	
DMA_CCR6	0x90	DMA 通道 6 通道配置寄存器
DMA_REN6	0x94	DMA 通道 6 请求使能寄存器
DMA_CTMS6	0x98	DMA 通道 6 传输次数寄存器
DMA_SADR6	0x9C	DMA 通道 6 源地址寄存器
DMA_DADR6	0xA0	DMA 通道 6 目的地址寄存器
	0xA4	
DMA_CCR7	0xA8	DMA 通道 7 通道配置寄存器
DMA_REN7	0xAC	DMA 通道 7 请求使能寄存器
DMA_CTMS7	0xB0	DMA 通道 7 传输次数寄存器
DMA_SADR7	0xB4	DMA 通道 7 源地址寄存器
DMA_DADR7	0xB8	DMA 通道 7 目的地址寄存器
	0xBC	
DMA_CTRL	0xC0	DMA 控制寄存器
DMA_IE	0xC4	DMA 中断使能寄存器
DMA_IF	0xC8	DMA 中断标志寄存器

8.6.2 DMA_CTRL DMA 控制寄存器

地址：0x4001_9CC0

复位值：0x0

表 8-3 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 使能

8.6.3 DMA_IE DMA 中断使能寄存器

地址：0x4001_9CC4

复位值：0x0

表 8-4 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
								CH7_FIE	CH6_FIE	CH5_FIE	CH4_FIE	CH3_FIE	CH2_FIE	CH1_FIE	CH0_FIE
								RW	RW	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

位置	位名称	说明
[31:8]		未使用
[7]	CH7_FIE	通道 7 完成中断使能
[6]	CH6_FIE	通道 6 完成中断使能
[5]	CH5_FIE	通道 5 完成中断使能
[4]	CH4_FIE	通道 4 完成中断使能
[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

8.6.4 DMA_IF DMA 中断标志寄存器

地址：0x4001_9CC8

复位值：0x0

表 8-5 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

	CH7_FIF	CH6_FIF	CH5_FIF	CH4_FIF	CH3_FIF	CH2_FIF	CH1_FIF	CH0_FIF
	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C
	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:8]		未使用
[7]	CH7_FIF	通道 7 完成中断标志，高有效，写 1 清零
[6]	CH6_FIF	通道 6 完成中断标志，高有效，写 1 清零
[5]	CH5_FIF	通道 5 完成中断标志，高有效，写 1 清零
[4]	CH4_FIF	通道 4 完成中断标志，高有效，写 1 清零
[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

8.6.5 DMA 通道配置寄存器

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

地址分别是：0x4001_9C00，0x4001_9C18，0x4001_9C30，0x4001_9C48，0x4001_9C60，0x4001_9C78，0x4001_9C90，0x4001_9CA8

复位值：0x0

表 8-6 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
[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:多轮传输，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

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

地址分别是：0x4001_9C04，0x4001_9C1C，0x4001_9C34，0x4001_9C4C，0x4001_9C64，0x4001_9C7C，0x4001_9C94，0x4001_9CAC

复位值：0x0

表 8-7 DMA 请求使能寄存器 DMA_ RENx

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SW_TRIG		GPIO	FMAC_Y	FMAC_X1	CMP	HALL1	HALL0	I2C1 TX	I2C1 RX	UART2 TX	UART2 RX	UART1 TX	UART1 RX	UART0 TX	UART0 RX
RW		RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0		0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I2C0 TX	I2C0 RX	MCPWM1	MCPWM0	SPI1 TX	SPI1 RX	SPI0 TX	SPI0 RX	TIMER4	TIMER3	TIMER2	TIMER1	TIMER0	ADC2	ADC1	ADC0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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。每一个 DMA 通道一般在同一时间通常只使能一个硬件 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。

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

地址分别是：0x4001_9C08，0x4001_9C20，0x4001_9C38，0x4001_9C50，0x4001_9C68，0x4001_9C80，0x4001_9C98，0x4001_9CB0

复位值：0x0

表 8-8 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
TIMES															
RW															
0															

位置	位名称	说明
[31:16]		未使用
[15:0]	TIMES	DMA 通道 x 数据搬运次数。此寄存器在该通道使能后变为只读。

DMA_CTMSx 寄存器只有在通道禁用，即 DMA_CCRx.EN=0 之后才可以写入数据。

当 DMA_CTRL=1 且 DMA_CCRx.EN=0 时，重新填写 CTMSx 值，可以将 DMA 内部已搬运的轮数次数计数器清零。

当 DMA_CCRx.RMODE=0 时，表示传输 1 轮，每轮传输 DMA_CTMS 次数据；

当 DMA_CCRx.RMODE=1 时，表示传输 DMA_CTMS 轮，每轮传输一次数据；

不支持多轮每轮传输多次数据。

以外设数据宽度为 16，内存数据宽度为 32，即 CTMS.TIMES=16，DMA_CCRx.RMODE=0 为例，DMA 每轮需要读取外设数据 16bit(2byte)×16=32byte，写入内存数据 32bit(4byte)×16=64byte。即读取外设 32byte，输入内存 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；当 DMA_CTMSx 递减为 0 时，重新装载为预设值，开启下一阶段的传输。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。在设置 DMA_CTMSx 寄存器时，需要保证 DMA_CTRL=1，即 DMA 被使能，否则 DMA_CTMSx 会写入无效。

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

地址分别是：0x4001_9C0C，0x4001_9C24，0x4001_9C3C，0x4001_9C54，0x4001_9C6C，0x4001_9C84，0x4001_9C9C，0x4001_9CB4

复位值：0x0

表 8-9 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 为单位从源地址搬运数据。DMA_SADRx. ADDR[0]值无效，源地址会以 2 为单位递增。

当 DMA_CCRx.SBTW=2'b10 时，即配置为以 32bit 为单位从源地址搬运数据。DMA_SADRx. ADDR [1:0]值无效，源地址会以 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！！！

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

地址分别是：0x4001_9C10，0x4001_9C28，0x4001_9C40，0x4001_9C58，0x4001_9C70，0x4001_9C88，0x4001_9CA0，0x4001_9CB8

复位值：0x0

表 8-10 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 为单位搬运数据到目的地址。DMA_DADRx.ADDR[0]值无效，目的地址会以 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。DMA_DADRx.ADDR[1:0]值无效，目的地址会以 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！！！

9 GPIO

9.1 概述

LSK32MC45x 系列芯片共集成了 84 个 GPIO。8 个 GPIO 可以作为系统的唤醒源。其中 34 个 GPIO 可以用作外部中断源输入。

9.1.1 功能框图

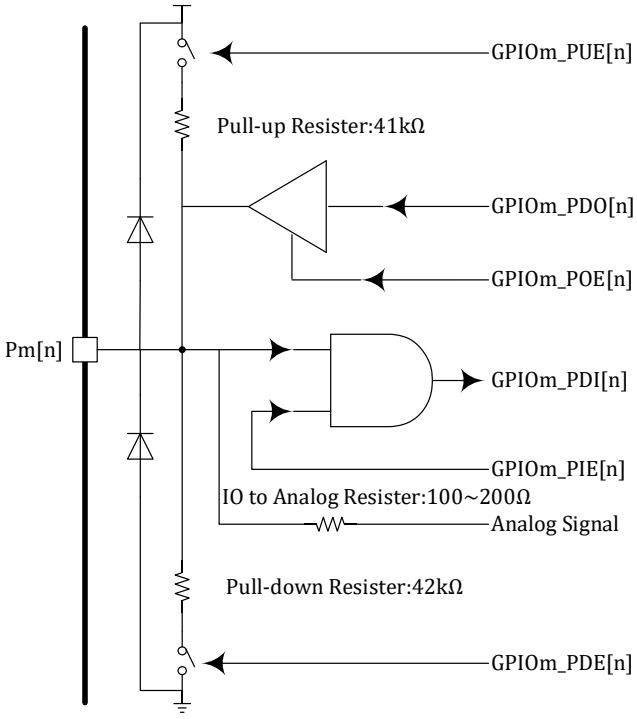


图 9-1 GPIO 功能框图

如上图所示，Pm[n]为芯片 PAD，m 可以是 0~5，表示 6 组 GPIO 中的任意一组，n 可以是 0~15，表示一组 16bit 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]的逻辑电平与 Pm[n]逻辑电平相同。芯片 PAD 可以配置上拉下拉，上拉电阻典型值为 41kΩ，下拉电阻典型值为 42kΩ。由于芯片制造工艺存在偏差，上下拉电阻值可能存在一定偏差。

9.1.2 产品特点

- 5 组 16bit GPIO，1 组 4bit GPIO，共 84 个 GPIO
- 推挽开漏模式可配置

- 支持配置锁定保护
- 支持 GPIO 外部中断
- 支持 GPIO 休眠唤醒

9.2 寄存器

9.2.1 地址分配

GPIO 0 模块基地址是 0x4001_6400。

GPIO 1 模块基地址是 0x4001_6440。

GPIO 2 模块基地址是 0x4001_6480。

GPIO 3 模块基地址是 0x4001_64C0。

GPIO 4 模块基地址是 0x4001_6500。

GPIO 5 模块基地址是 0x4001_6540。

EXTI 模块基地址是 0x4001_6580。

GPIO 0/1/2/3/4/5 的寄存器定义完全相同，仅基地址不同。但 P5 只有 P5[3:0]有实现，不存在 P5[15:4]这几个 IO。

表 9-1 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_PDE	0x14	GPIO x 下拉使能
GPIOx_PODE	0x18	GPIO x 开漏使能
GPIOx_PFLT	0x1C	GPIO x 滤波使能
GPIOx_F3210	0x20	GPIO x [3:0]功能选择
GPIOx_F7654	0x24	GPIO x [7:4]功能选择
GPIOx_FBA98	0x28	GPIO x [11:8]功能选择
GPIOx_FFEDC	0x2C	GPIO x [15:12]功能选择
GPIOx_BSRR	0x30	GPIO x 位操作寄存器
GPIOx_BRR	0x34	GPIO x 位清零寄存器

GPIO 中断/DMA 模块的基地址是 0x4001_6580。

表 9-2 GPIO 中断/DMA 模块寄存器列表

名称	偏移地址	说明
EXTI0_CR0	0x00	GPIO 0[7:0] 中断/DMA 触发类型
EXTI0_CR1	0x04	GPIO 0[15:8]中断/DMA 触发类型
EXTI1_CR0	0x08	GPIO 1[7:0] 中断/DMA 触发类型
EXTI1_CR1	0x0C	GPIO 1[15:8]中断/DMA 触发类型
EXTI2_CR0	0x10	GPIO 2[7:0] 中断/DMA 触发类型
EXTI2_CR1	0x14	GPIO 2[15:8]中断/DMA 触发类型
EXTI3_CR0	0x18	GPIO 3[7:0] 中断/DMA 触发类型
EXTI3_CR1	0x1C	GPIO 3[15:8]中断/DMA 触发类型
EXTI4_CR0	0x20	GPIO 4[7:0] 中断/DMA 触发类型
EXTI4_CR1	0x24	GPIO 4[15:8]中断/DMA 触发类型
EXTI5_CR0	0x28	GPIO 5[7:0] 中断/DMA 触发类型
EXTI5_CR1	0x2C	GPIO 5[15:8]中断/DMA 触发类型
EXTI0_IF	0x30	GPIO 0 中断标志
EXTI1_IF	0x34	GPIO 1 中断标志
EXTI2_IF	0x38	GPIO 2 中断标志
EXTI3_IF	0x3C	GPIO 3 中断标志
EXTI4_IF	0x40	GPIO 4 中断标志
EXTI5_IF	0x44	GPIO 5 中断标志
CLKO_SEL	0x50	时钟输出选择
LCKR_PRT	0x54	GPIO 保护
EXTI_REN	0x58	GPIO DMA 请求使能

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

GPIO0/1/2/3/4/5 地址分别是：0x4001_6400，0x4001_6440，0x4001_6480，0x4001_64C0，0x4001_6500，0x4001_6540

复位值：0x0

表 9-3 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

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

GPIO0/1/2/3/4/5 地址分别是：0x4001_6404，0x4001_6444，0x4001_6484，0x4001_64C4，0x4001_6504，0x4001_6544

复位值：0x0

表 9-4 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

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

地址分别是：0x4001_6408，0x4001_6448，0x4001_6488，0x4001_64C8，0x4001_6508，0x4001_6548

复位值：0x0

表 9-5 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。

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

GPIO0/1/2/3/4/5 地址分别是：0x4001_640C，0x4001_644C，0x4001_648C，0x4001_64CC，0x4001_650C，0x4001_654C

复位值：0x0

表 9-6 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

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

GPIO0/1/2/3/4/5 地址分别是：0x4001_6410，0x4001_6450，0x4001_6490，0x4001_64D0，0x4001_6510，0x4001_6550

复位值：0x0

表 9-7 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

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

GPIO0/1/2/3/4/5 地址分别是：0x4001_6414，0x4001_6454，0x4001_6494，0x4001_64D4，0x4001_6514，0x4001_6554

复位值：0x0

表 9-8 GPIOx 下拉使能寄存器 GPIOx_PDE

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

位置	位名称	说明
[31:16]		未使用
[15]	PDE15	GPIO x[15] / Px[15] 下拉使能
[14]	PDE14	GPIO x[14] / Px[14] 下拉使能
[13]	PDE13	GPIO x[13] / Px[13] 下拉使能
[12]	PDE12	GPIO x[12] / Px[12] 下拉使能
[11]	PDE11	GPIO x[11] / Px[11] 下拉使能
[10]	PDE10	GPIO x[10] / Px[10] 下拉使能
[9]	PDE9	GPIO x[9] / Px[9] 下拉使能
[8]	PDE8	GPIO x[8] / Px[8] 下拉使能
[7]	PDE7	GPIO x[7] / Px[7] 下拉使能
[6]	PDE6	GPIO x[6] / Px[6] 下拉使能
[5]	PDE5	GPIO x[5] / Px[5] 下拉使能
[4]	PDE4	GPIO x[4] / Px[4] 下拉使能
[3]	PDE3	GPIO x[3] / Px[3] 下拉使能
[2]	PDE2	GPIO x[2] / Px[2] 下拉使能
[1]	PDE1	GPIO x[1] / Px[1] 下拉使能
[0]	PDE0	GPIO x[0] / Px[0] 下拉使能

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

GPIO0/1/2/3/4/5 地址分别是：0x4001_6418，0x4001_6458，0x4001_6498，0x4001_64D8，0x4001_6518，0x4001_6558

复位值：0x0

表 9-9 GPIOx 开漏使能寄存器 GPIOx_PODE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PODE15	PODE14	PODE13	PODE12	PODE11	PODE10	PODE9	PODE8	PODE7	PODE6	PODE5	PODE4	PODE3	PODE2	PODE1	PODE0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:16]		未使用
[15]	PODE15	GPIO x[15] / Px[15] 开漏使能
[14]	PODE14	GPIO x[14] / Px[14] 开漏使能
[13]	PODE13	GPIO x[13] / Px[13] 开漏使能
[12]	PODE12	GPIO x[12] / Px[12] 开漏使能
[11]	PODE11	GPIO x[11] / Px[11] 开漏使能
[10]	PODE10	GPIO x[10] / Px[10] 开漏使能
[9]	PODE9	GPIO x[9] / Px[9] 开漏使能
[8]	PODE8	GPIO x[8] / Px[8] 开漏使能
[7]	PODE7	GPIO x[7] / Px[7] 开漏使能
[6]	PODE6	GPIO x[6] / Px[6] 开漏使能
[5]	PODE5	GPIO x[5] / Px[5] 开漏使能
[4]	PODE4	GPIO x[4] / Px[4] 开漏使能
[3]	PODE3	GPIO x[3] / Px[3] 开漏使能
[2]	PODE2	GPIO x[2] / Px[2] 开漏使能
[1]	PODE1	GPIO x[1] / Px[1] 开漏使能
[0]	PODE0	GPIO x[0] / Px[0] 开漏使能

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

GPIO0/1/2/3/4/5 地址分别是：0x4001_641C，0x4001_645C，0x4001_649C，0x4001_64DC，0x4001_651C，0x4001_655C

复位值：0x0

表 9-10 GPIOx 滤波寄存器 GPIOx_PFLT

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

位置	位名称	说明
[31:16]		未使用
[15]	PFLT15	GPIO x[15] / Px[15] 配置滤波
[14]	PFLT14	GPIO x[14] / Px[14] 配置滤波
[13]	PFLT13	GPIO x[13] / Px[13] 配置滤波
[12]	PFLT12	GPIO x[12] / Px[12] 配置滤波
[11]	PFLT11	GPIO x[11] / Px[11] 配置滤波

[10]	PFLT10	GPIO x[10] / Px[10] 配置滤波
[9]	PFLT9	GPIO x[9] / Px[9] 配置滤波
[8]	PFLT8	GPIO x[8] / Px[8] 配置滤波
[7]	PFLT7	GPIO x[7] / Px[7] 配置滤波
[6]	PFLT6	GPIO x[6] / Px[6] 配置滤波
[5]	PFLT 5	GPIO x[5] / Px[5] 配置滤波
[4]	PFLT 4	GPIO x[4] / Px[4] 配置滤波
[3]	PFLT 3	GPIO x[3] / Px[3] 配置滤波
[2]	PFLT 2	GPIO x[2] / Px[2] 配置滤波
[1]	PFLT1	GPIO x[1] / Px[1] 配置滤波
[0]	PFLT0	GPIO x[0] / Px[0] 配置滤波

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

GPIO0/1/2/3/4/5 地址分别是：0x4001_6420，0x4001_6460，0x4001_64A0，0x4001_64E0，0x4001_6520，0x4001_6560

复位值：0x0

表 9-11 GPIOx 功能选择寄存器 GPIOx_F3210

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

位置	位名称	说明
[31:16]		未使用
[15:12]	F3	GPIO x[3] / Px[3] 功能选择
[11:8]	F2	GPIO x[2] / Px[2] 功能选择
[7:4]	F1	GPIO x[1] / Px[1] 功能选择
[3:0]	F0	GPIO x[0] / Px[0] 功能选择

GPIO 的功能复用选择如表 9-12 所示

表 9-12 GPIO 功能复用

GPIOx_Fxxxx 配置值	第二功能代号	功能
0x0	AF0	模拟功能 在使用模拟功能时，需要关闭 GPIO 对应的 IE 和 OE
0x1	AF1	比较器输出或时钟输出功能
0x2	AF2	HALL
0x3	AF3	MCPWM
0x4	AF4	UART

0x5	AF5	SPI
0x6	AF6	IIC
0x7	AF7	Timer0(QEP0)/Time1(QEP1)/Time4
0x8	AF8	Timer2(QEP2)/Timer3(QEP3)
0x9	AF9	ADC trigger debug
0xA	AF10	CAN
0xB	AF11	外部 SPI flash

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

GPIO0/1/2/3/4/5 地址分别是：0x4001_6424，0x4001_6464，0x4001_64A4，0x4001_64E4，0x4001_6524，0x4001_6564

复位值：0x0

表 9-13 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

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

GPIO0/1/2/3/4/5 地址分别是：0x4001_6428，0x4001_6468，0x4001_64A8，0x4001_64E8，0x4001_6528，0x4001_6568

复位值：0x0

表 9-14 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

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

GPIO0/1/2/3/4/5 地址分别是：0x4001_642C，0x4001_646C，0x4001_64AC，0x4001_64EC，0x4001_652C，0x4001_656C

复位值：0x0

表 9-15 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。

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

GPIO0/1/2/3/4/5 地址分别是：0x4001_6430，0x4001_6470，0x4001_64B0，0x4001_64F0，0x4001_6530，0x4001_6570

复位值：0x0

表 9-16 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。

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

GPIO0/1/2/3/4/5 地址分别是：0x4001_6434，0x4001_6474，0x4001_64B4，0x4001_64F4，0x4001_6534，0x4001_6574

复位值：0x0

表 9-17 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
PFLT15	PFLT14	PFLT13	PFLT12	PFLT11	PFLT10	PFLT9	PFLT8	PFLT7	PFLT6	PFLT5	PFLT4	PFLT3	PFLT2	PFLT1	PFLT0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

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

GPIO0/1/2/3/4/5 地址分别是：0x4001_6580，0x4001_6588，0x4001_6590，0x4001_6598，0x4001_65A0，0x4001_65A8

复位值：0x0

表 9-18 GPIOx 外部中断/DMA 配置寄存器 EXTIx_CR0

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

位置	位名称	说明
[31:16]		未使用
[15:14]	T7	GPIO x[7]/ Px[7]外部中断/DMA 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 x[6]/ Px[6]外部中断/DMA 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 x[5]/ Px[5]外部中断/DMA 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 x[4]/ Px[4]外部中断/DMA 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 x[3]/ Px[3]外部中断触发类型选择 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 x[2]/ Px[2]外部中断/DMA 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 x[1]/ Px[1]外部中断/DMA 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 x[0]/ Px[0]外部中断/DMA 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

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

GPIO0/1/2/3/4/5 地址分别是：0x4001_6584，0x4001_658C，0x4001_6594，0x4001_659C，0x4001_65A4，0x4001_65AC

复位值：0x0

表 9-19 GPIOx 外部中断配置寄存器 EXTIx_CR1

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

位置	位名称	说明
[31:16]		未使用
[15:14]	T15	GPIO x[15]/Px[15]外部中断/DMA 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 x[14]/Px[14]外部中断/DMA 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 x[13]/Px[13]外部中断/DMA 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 x[12]/Px[12]外部中断/DMA 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 x[11]/Px[11]外部中断/DMA 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 x[10]/Px[10]外部中断/DMA 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 x[9]/Px[9]外部中断/DMA 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 x[8]/Px[8]外部中断/DMA 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

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

GPIO0/1/2/3/4/5 地址分别是：0x4001_65B0，0x4001_65B4，0x4001_65B8，0x4001_65BC，0x4001_65C0，0x4001_65C4

复位值：0x0

表 9-20 GPIOx 外部中断标志寄存器 EXTIx_IF

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

位置	位名称	说明
[31:16]		未使用
[15]	IF15	GPIO x[15] / Px[15] 外部中断标志。中断标志高有效，写 1 清零
[14]	IF14	GPIO x[14] / Px[14] 外部中断标志。中断标志高有效，写 1 清零
[13]	IF13	GPIO x[13] / Px[13] 外部中断标志。中断标志高有效，写 1 清零
[12]	IF12	GPIO x[12] / Px[12] 外部中断标志。中断标志高有效，写 1 清零
[11]	IF11	GPIO x[11] / Px[11] 外部中断标志。中断标志高有效，写 1 清零
[10]	IF10	GPIO x[10] / Px[10] 外部中断标志。中断标志高有效，写 1 清零
[9]	IF9	GPIO x[9] / Px[9] 外部中断标志。中断标志高有效，写 1 清零
[8]	IF8	GPIO x[8] / Px[8] 外部中断标志。中断标志高有效，写 1 清零
[7]	IF7	GPIO x[7] / Px[7] 外部中断标志。中断标志高有效，写 1 清零
[6]	IF6	GPIO x[6] / Px[6] 外部中断标志。中断标志高有效，写 1 清零
[5]	IF5	GPIO x[5] / Px[5] 外部中断标志。中断标志高有效，写 1 清零
[4]	IF4	GPIO x[4] / Px[4] 外部中断标志。中断标志高有效，写 1 清零
[3]	IF3	GPIO x[3] / Px[3] 外部中断标志。中断标志高有效，写 1 清零
[2]	IF2	GPIO x[2] / Px[2] 外部中断标志。中断标志高有效，写 1 清零
[1]	IF1	GPIO x[1] / Px[1] 外部中断标志。中断标志高有效，写 1 清零
[0]	IF0	GPIO x[0] / Px[0] 外部中断标志。中断标志高有效，写 1 清零

只有部分 GPIO 可以作为外部中断源。可以作为外部中断的 GPIO 如下表所示。

表 9-21 GPIO 中断资源分布

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

9.2.19 CLKO_SEL

地址：0x4001_65D0

复位值：0x0

表 9-22 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
											HSE_OE	ADC_OE	PLL_OE	HSI_OE	LSI_OE
											RW	RW	RW	RW	RW
											0	0	0	0	0

位置	位名称	说明
[31:5]		未使用
[4]	HSE_OE	晶振时钟输出使能。1：使能；0：禁用。
[3]	ADC_OE	ADC 时钟输出使能。1：使能；0：禁用。
[2]	PLL_OE	PLL 时钟输出使能。1：使能；0：禁用。
[1]	HSI_OE	HRC 时钟输出使能。1：使能；0：禁用。
[0]	LSI_OE	LRC 时钟输出使能。1：使能；0：禁用。

CLKO_SEL 主要用于输出芯片内部时钟，通常用于调试目的。

9.2.20 LCKR_PRT

地址：0x4001_65D4

复位值：0x0

表 9-23 GPIO 保护寄存器 LCKR_PRT

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

位置	位名称	说明
[31:16]		未使用
[15:1]	PSW[15:1]	GPIO 保护密码高 15 位
[0]	PSW[0]/LOCK	GPIO 保护密码最低位,向 LCKR_PRT[15:0]写入 0x5AC4 可解除 GPIO 配置保护，写入其他数值启用 GPIO 配置保护，读取 LCKR_PRT[0]可得到当前保护状态，LCKR_PRT[15:1]读回恒为 0。当启用配置保护后，GPIOx_PIE/GPIOx_POE/GPIOx_PDO/GPIOx_PUE/GPIOx_PDE/GPIOx_PODE/GPIOx_F3210/GPIOx_F7654/GPIOx_FBA98/GPIOx_FFEDC/GPIOx_BSRR/GPIOx_BRR 均无法写入。

9.2.21 EXTI_REN

地址：0x4001_65D8

复位值：0x0

表 9-24 GPIO DMA 请求使能寄存器 EXTI_REN

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

位置	位名称	说明
[31:6]		未使用
[5:0]	EXTI_REN	GPIO5~GPIO0 DMA 请求使能，需要配合 EXTIx_CR0/1 使用

每一组 GPIO 外部请求信号可以统一用作中断请求，或者 DMA 请求。但通常每组信号不会同时用作中断和 DMA 请求。DMA 请求信号的产生同样来自中断标志置位，当中断发生后，如果使能了 EXTI_REN 对应位，则 DMA 会在响应请求后清除该组 GPIO 的所有中断标志。

只有部分 GPIO 支持作为外部中断和 DMA 请求信号，具体请参考数据手册。

9.3 实现说明

9.3.1 上拉实现

LKS32MC45x 系列芯片,所有 GPIO 均配备有典型值 41kΩ 上拉电阻,通过 GPIOx_PUE 进行控制。

9.3.2 下拉实现

LKS32MC45x 系列芯片,所有 GPIO 均配备有典型值 42kΩ 下拉电阻,通过 GPIOx_PDE 进行控制。

9.3.3 滤波实现

LKS32MC45x 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 9-25 GPIO 滤波资源分布表

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

9.3.4 外部中断实现

LKS32MC45x 系列芯片，部分 GPIO 可以作为外部中断源，对应中断编号 26 GPIO(EXTI)。需要配置 EXTIx_CR0/EXTIx_CR1 使能后，才可以触发外部中断。

表 9-26 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				√	√				√		√					
P2	√			√				√	√					√	√	√

P3	√	√	√	√	√			√	√				√	√		
P4														√	√	√
P5															√	√

9.3.5 外部唤醒实现

LKS32MC45x 系列芯片，部分 GPIO 可以作为外部唤醒源。需要配置 AON_IO_WAKE_EN 使能并设置 AON_IO_WAKE_POL 为适当电平。

表 9-27 GPIO 唤醒资源分布表

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

9.4 应用指南

9.4.1 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
EXTI0_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
EXTI0_CR0 = 0x4000;                 // 使能 p0[7]下降沿触发，产生外部中断
while(irq_flag != 4);
EXTI0_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);
EXTI0_CR0 = 0x0000;                 // 同时禁用 P[7]上下沿触发，将无法产生外部中断
```

9.4.2 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。

10 CRC

10.1 概述

CRC 即循环冗余校验码（Cyclic Redundancy Check）：是数据通信领域中最常用的一种查错校验码，其特征是信息字段和校验字段的长度可以任意选定。循环冗余检查（CRC）是一种数据传输检错功能，对数据进行多项式计算，并将得到的结果附在帧的后面，接收设备也执行类似的算法，以保证数据传输的正确性和完整性。

利用 CRC 进行检错的过程可简单描述为：在发送端根据要传送的 K 位二进制码序列，以一定的规则产生一个校验用的 R 位监督码(CRC 码)，附在原始信息后边，构成一个新的二进制码序列数共 K+R 位，然后发送出去。在接收端，根据信息码和 CRC 码之间所遵循的规则进行检验，以确定传送中是否出错。这个规则，在差错控制理论中称为“生成多项式”。

10.2 基本原理

循环冗余校验码（CRC）的基本原理是：在 K 位信息码后再拼接 R 位的校验码，整个编码长度为 N 位，因此，这种编码也叫（N，K）码。对于一个给定的（N，K）码，可以证明存在一个最高次幂为 N-K=R 的多项式 G(x)。根据 G(x)可以生成 K 位信息的校验码，而 G(x)叫做这个 CRC 码的生成多项式。校验码的具体生成过程为：假设要发送的信息用多项式 C(X)表示，将 C(x)左移 R 位（可表示成 $C(x)*2^R$），这样 C(x)的右边就会空出 R 位，这就是校验码的位置。用 $C(x)*2^R$ 除以生成多项式 G(x)得到的余数就是校验码。

任意一个由二进制位串组成的代码都可以和一个系数仅为‘0’和‘1’取值的多项式一一对应。例如：代码 1010111 对应的多项式为 $x^6+x^4+x^2+x+1$，而多项式为 $x^5+x^3+x^2+x+1$ 对应的代码 101111。

10.3 基本概念

10.3.1 对应关系

多项式和二进制数有直接对应关系：X 的最高幂次对应二进制数的最高位，以下各位对应多项式的各幂次，有此幂次项对应 1，无此幂次项对应 0。可以看出：X 的最高幂次为 R，转换成对应的二进制数有 R+1 位。

多项式包括生成多项式 G(X)和信息多项式 C(X)。

如生成多项式为 $G(X)=X^4+X^3+X+1$，可转换为二进制数码 11011。

而发送信息为 101111，可转换为数据多项式为 $C(X)=X^5+X^3+X^2+X+1$。

10.3.2 生成多项式

生成多项式是接受方和发送方的一个约定，也就是一个二进制数，在整个传输过程中，这个数始终保持不变。

在发送方，利用生成多项式对信息多项式做模 2 除生成校验码。在接收方利用生成多项式对收到的编码多项式做模 2 除检测和确定错误位置。

应满足以下条件:

A、生成多项式的最高位和最低位必须为 1。

B、当被传送信息（CRC 码）任何一位发生错误时，被生成多项式做除后应该使余数不为 0。

C、不同位发生错误时，应该使余数不同。

D、对余数继续做除，应使余数循环。

10.3.3 校验码位数

CRC 校验码位数 = 生成多项式位数 - 1。注意有些生成多项式的简记式中将生成多项式的最高位 1 省略了。

10.3.4 生成步骤

1、将 X 的最高次幂为 R 的生成多项式 G(X)转换成对应的 R+1 位二进制数。

2、将信息码左移 R 位，相当于对应的信息多项式 C(X)*2^R。

3、用生成多项式（二进制数）对信息码做除，得到 R 位的余数(注意：这里的二进制做除法得到的余数其实是模 2 除法得到的余数，并不等于其对应十进制数做除法得到的余数。)。

4、将余数拼到信息码左移后空出的位置，得到完整的 CRC 码。

【例】假设使用的生成多项式是 G(X)=X3+X+1。4 位的原始报文为 1010，求编码后的报文。

解:

1、将生成多项式 G(X)=X3+X+1 转换成对应的二进制除数 1011。

2、此题生成多项式有 4 位（R+1）(注意：4 位的生成多项式计算所得的校验码为 3 位，R 为校验码位数)，要把原始报文 C(X)左移 3（R）位变成 1010 000

3、用生成多项式对应的二进制数对左移 3 位后的原始报文进行模 2 除（高位对齐），相当于按位异或:

1010000

1011

0001000

0001011

0000011

得到的余位 011，所以最终编码为：1010 011

POL=0x13，data=0x77

011101110000000

10010011

01111101000000

10010011

0110100100000

10010011

010000010000

10010011

00010001000

10010011

00011011

10.4 寄存器

10.4.1 地址分配

CRC 的基地址是 0x4001_6800，寄存器列表如下：

表 10-1 CRC 寄存器列表

名称	偏移地址	说明
CRC_DR	0x00	CRC 数据（输入信息码/输出编码）寄存器
CRC_CR	0x04	CRC 控制寄存器
CRC_INIT	0x08	CRC 初始码寄存器
CRC_POL	0x0C	CRC 生成多项式对应的二进制码寄存器

10.4.2 寄存器描述

10.4.2.1CRC_DR CRC 信息码寄存器

地址：0x4001_6800

复位值：0x0

表 10-2 CRC 数据寄存器 CRC_DR

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

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

位置	位名称	说明
[31:0]	DR	存放待编码的信息码和经 CRC 校验后的编码

CRC_DR 寄存器既用于放入待校验数据，也用于返回校验结果。写入 CRC_DR 寄存器即触发一次 CRC 计算。待编码数据应在 CR 等寄存器配置完成后最后写入，以触发 CRC 计算开始。

10.4.2.2CRC_CR CRC 控制寄存器

地址：0x4001_6804

复位值：0x0

表 10-3 CRC 控制寄存器 CRC_CR

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

位置	位名称	说明
[31:13]		未使用

[12]	REV_OUT_TYPE	是否将 CRC 校验后的编码反转后输出，即 b[31]=b[0], b[30]=b[1],... [b0]=b[31]
[11:10]		未使用
[9:8]	REV_IN_TYPE	待编码数据反转类型 00:不反转 01:按字节反转，即 b[31]=b[24], b[30]=b[25], ..., b[24]=b[31], ..., b[7]=b[0], b[6]=b[1], ..., b[0]=b[7] 10: 按半字（16bit）反转，即 b[31]=b[16], b[30]=b[17], ..., b[16]=b[31], ..., b[15]=b[0], b[14]=b[1], ..., b[0]=b[15] 11:按字反转，即 b[31]=b[0], b[30]=b[1],... [b0]=b[31]
[7:6]		未使用
[5:4]	POLY_SIZE	输出编码（多项式）位宽 00: 32bits 01: 16bits 10: 8bits 11: 7bits
[3:1]		未使用
[0]	RESET	与输入信息码进行 CRC 计算的数据来源 0:来自于上一次的计算结果 1:来自于 CRC_INIT 写入 1 实现 CRC 数据重置并自动清零，读回恒为 0.

同时需要注意的是，向 CRC_CR.RESET 写入 1 会将 CRC_INIT 寄存器复位为 0xFFFFFFFF。

如果需要清除 CRC 的计算结果，应向 CRC_CR.RESET 写入 1，否则后续 CRC 计算会以之前的计算结果为初值进行。

10.4.2.3CRC_INIT CRC 初始码寄存器

地址：0x4001_6808

复位值：0x0

表 10-4 CRC 初始码寄存器 CRC_INIT

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

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

位置	位名称	说明
[31:0]	INIT	存放初始码

CRC_DR 与 CRC_INIT 相异或后开始进行 CRC 校验计算。

10.4.2.4CRC_POL CRC 生成码寄存器

地址：0x4001_680C

复位值：0x0

表 10-5 CRC 生成码寄存器 CRC_POL

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

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

位置	位名称	说明
[31:0]	POL	存放生成多项式对应的生成码

11 CORDIC

11.1 概述

三角函数 Cordic 模块位宽为 16 位，Q15 定点数格式。

11.2 寄存器

11.2.1 地址分配

DSP 寄存器在芯片中的基地址是 0x4001_6C00。

表 11-1 DSP 寄存器列表

名称	偏移	说明
DSP_SC	0x00	DSP 状态控制寄存器
DSP_THETA	0x04	DSP sin/cos 输入角度寄存器
DSP_X	0x08	DSP arctan/module 计算输入坐标 X 寄存器
DSP_Y	0x0C	DSP arctan/module 计算输入坐标 Y 寄存器
DSP_SIN	0x10	DSP sin/cos 计算结果 sin 寄存器
DSP_COS	0x14	DSP sin/cos 计算结果 cos 寄存器
DSP_MOD	0x18	DSP arctan 计算结果 $sqrt(X^2+Y^2)$ 寄存器
DSP_ARCTAN	0x1C	DSP arctan 计算结果 arctan(Y/X) 角度寄存器

11.2.2 DSP_SC DSP 状态控制寄存器

地址：0x4001_6C00

复位值：0x2

表 11-2 DSP 状态控制寄存器 DSP_SC

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

位置	位名称	说明

[31:3]		保留
[2]	MODE	CORDIC mode, 0: arctan, 1: sin/cos
[1:0]		保留

MODE 位进行 sin/cos mode 和 arctan mode 的选择，CORDIC 模块计算 sin/cos 或 arctan 使用的是相同的硬件电路。因此在进行某一种计算前，应通过配置 DSP_SC.MODE 寄存器进行适当模式选择。DSP_SC.MODE=1 时，CORDIC 模块计算 sin/cos，以角度 theta 为输入，输出 sin/cos 结果；DSP_SC.MODE=0 时，CORDIC 模块计算 arctan，以坐标 x/y 为输入，计算并输出角度 theta= arctan (y/x) 和 module= sqrt (x^2+y^2)。

11.2.3 DSP sin/cos 相关寄存器

Cordic 模块计算 sin/cos 和 arctan 使用的是相同的数据通路，因此通过 CPU 使用 cordic 模块进行 sin/cos 计算，需要先将 DSP_SC[2]写为 1，使 cordic 进入 sin/cos 模式。

11.2.3.1 DSP_THETA

地址：0x4001_6C04

复位值：0x0

表 11-3 DSP sin/cos 角度输入寄存器

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

位置	位名称	说明
[31:16]		保留，读出时为符号扩展，即{16{DSP_THETA[15]}}
[15:0]	THETA	DSP sin/cos 输入角度寄存器

DSP_THETA 为 16 位有符号定点数，表示范围（-32768 ~ 32767）对应（–π ~ π）。DSP_THETA 为只写寄存器，写入触发一次三角函数计算，DSP_SIN/DSP_COS 使用 DSP_THETA 作为角度输入，同时计算完成。

11.2.3.2 DSP_SIN

地址：0x4001_6C10

复位值：0x0

表 11-4 DSP sin/cos 正弦结果寄存器

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

SIN
RO
0

位置	位名称	说明
[31:16]		保留，读出时为符号扩展，即{16{DSP_SIN[15]}}
[15:0]	SIN	DSP sin/cos 计算结果 sin 寄存器

DSP_SIN 为 16 位有符号定点数，其中 1bit 符号位，15bit 小数位；表示范围（–1 ~ 1）。

11.2.3.3 DSP_COS

地址：0x4001_6C14

复位值：0x0

表 11-5 DSP sin/cos 余弦结果寄存器

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

位置	位名称	说明
[31:16]		保留，读出时为符号扩展，即{16{DSP_COS[15]}}
[15:0]	COS	DSP sin/cos 计算结果 cos 寄存器

DSP_COS 为 16 位有符号定点数，其中 1bit 符号位，15bit 小数位；表示范围（–1 ~ 1）。

11.2.4 DSP arctan 相关寄存器

11.2.4.1 DSP_X

地址：0x4001_6C08

复位值：0x0

表 11-6 DSP arctan/module 坐标 X 输入寄存器

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

位置	位名称	说明
[15:0]	X	DSP arctan/module 计算输入坐标 Y 寄存器

DSP_X 为 16 位有符号定点数，其中 1bit 符号位，15bit 整数位，表示范围(–32768 ~ 32767)。

计算 arctan(DSP_Y/DSP_X)时，应先写入 DSP_X。

11.2.4.2 DSP_Y

地址：0x4001_6C0C

复位值：0x0

表 11-7 DSP arctan/module 坐标 Y 输入寄存器

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

位置	位名称	说明
[15:0]	Y	DSP arctan/module 计算输入坐标 Y 寄存器

DSP_Y 为 16 位有符号定点数，其中 1bit 符号位，15bit 整数位，表示范围(–32768 ~ 32767)。当计算 arctan(Y/X)时，一定要先写入 DSP_X，再写入 DSP_Y。写入 DSP_Y 触发一次 arctan 计算。

11.2.4.3 DSP_MOD

地址：0x4001_6C18

复位值：0x0

表 11-8 DSP arctan 角度结果 arctan(Y/X) 角度寄存器

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

位置	位名称	说明
[31:16]		保留，读恒为 0
[15:0]	MOD	DSP arctan 计算结果 sqrt(X^2+Y^2) 寄存器

DSP_MOD 为 16 位无符号整数，表示范围(0 ~ 65535)。

11.2.4.4 DSP_ARCTAN

地址：0x4001_6C1C

复位值：0x0

表 11-9 DSP arctan 角度结果 arctan(Y/X) 角度寄存器

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

位置	位名称	说明
[31:16]		保留，读出时为符号扩展，即{16{DSP_ARCTAN[15]}}
[15:0]	ARCTAN	DSP arctan 计算结果 arctan(Y/X) 角度寄存器

DSP_ARCTAN 为 16 位有符号定点数，表示范围（-32768 ~ 32767）对应（–π ~ π）。

12 FMAC 滤波加速器

12.1 概述

滤波加速器可进行向量算术操作，包含乘法器、累加器，以及本地存储器地址产生逻辑，用于在滤波加速器本地存储中索引向量数据。

滤波加速模块支持输入、输出循环滚动缓存，以便于实现有限/无限冲击响应数字滤波器。

滤波加速模块将 CPU 从冗长而频繁的滤波操作中解放出来，节省 CPU 算力用于其他任务。同时与 CPU 的软件滤波相比，硬件滤波加速模块可以较大程度提升滤波计算速度。

12.2 特性

- 16 位×16 位 乘法器
- 24+2 位累加器，支持饱和处理
- 16 位数据输入、输出
- 256×16 位本地数据存储
- 本地存储最多可以定义 3 个数据缓存区域（两个输入缓存，一个输出缓存）缓存基地址和缓存大小可以通过寄存器配置
- 输入、输出缓存可以作为循环 buffer 使用
- 滤波函数：FIR，直接 1 型 IIR
- 向量操作：点积，卷积，相关
- AHB 总线接口
- 支持 DMA 读写数据

12.3 功能描述

滤波加速模块如图 12-1 所示。

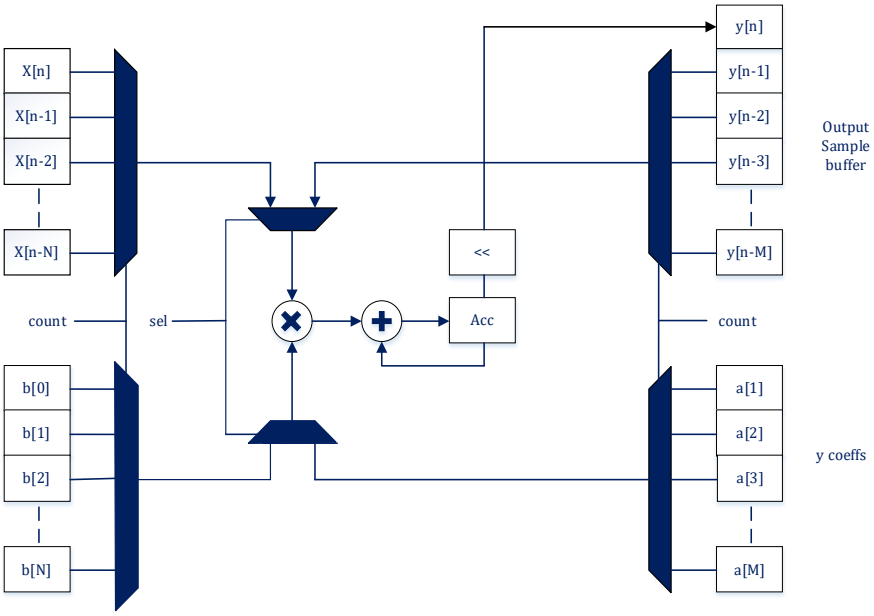


图 12-1 滤波加速模块框图

该模块的核心是定点乘累加器（MAC）。乘累加器从本地存储中读取两个 16 位有符号数作为输入，相乘以后与之前的和进行累加。本地存储中的数据地址由一系列的地址指针来决定。地址指针可以加载初始值，递增，递减，也可以被内部硬件复位回初值。所有的指针和 MAC 模块都是由内部的硬件逻辑控制来完成指定的操作序列。

如果要计算点积，两个输入向量被 CPU 或 DMA 存入 FMAC 本地存储中。当相应的操作被配置并开始执行时，每次硬件逻辑从本地存储读取来自两个向量的一对数据作为 MAC 输入，乘累加后暂存至内部中间结果寄存器，当所有向量元素均计算完成后，累加器的输出作为最终结果被写回本地存储，等待 CPU 或 DMA 读出。

如果要作为 FIR 滤波器（卷积）使用，FMAC 需要反复计算滤波器系数和输入向量的点积。每个采样周期，输入向量位移一个采样数据，最早的一个采样数据被丢弃，加入一个新的采样数据，然后使用相同的滤波器系数向量重新计算一次点积。如此往复。

如果要作为 IIR 滤波器使用,需要在 FIR 的基础上再进行计算输出结果与反馈系数向量的点积并与 FIR 的滤波结果相加。

更多的滤波器操作实现细节见后续章节介绍。

12.3.1 本地存储与缓存

FMAC 包含有一个 256×16 位的可读写存储器作为 FMAC 的本地存储

输入数据（输入向量元素）存储在两个输入 buffer X1 和 X2 中。

输出数据（操作结果）存储在 buffer Y 中。

缓存基地址和缓存大小由以下变量指定:

x1_base 为 X1 buffer 在本地存储中的基地址，

x2_base 为 X2 buffer 在本地存储中的基地址，

y_base 为 Y buffer 在本地存储中的基地址，

x1_buf_size 为 X1 buffer 的缓存大小，即其内包含多少个 16bit 有符号数

x2_buf_size 为 X2 buffer 的缓存大小，即其内包含多少个 16bit 有符号数

y_buf_size 为 Y buffer 的缓存大小，即其内包含多少个 16bit 有符号数

以上参数均可通过模块的寄存器进行配置。

CPU 或 DMA 可以通过 FMAC 模块的初始化功能，也可以通过直接写入，来初始化各个缓存的数据。写入的数据根据硬件写指针存入目标 buffer，每次数据写入，硬件写指针自动加 1，当写指针到达 buffer 尾时，再次回到 buffer 基地址。初始化功能用于滤波操作前的数据预装载，或者初始化滤波器系数。

缓存配置

缓存基地址和缓存大小通过 X1,X2,Y buffer 的配置寄存器进行配置。基地址可以是本地存储的任意位置，只要基地址+缓存大小不超过本地存储地址范围(0x00~0xFF)即可。即基地址+缓存大小≤256.

缓存的大小和位置没有限制，缓存甚至可以重叠，但是要解决好数据读写先后的问题。对于滤波操作来说，一般不建议缓存重叠，容易出现不易调试的错误。

当缓存作为循环 buffer 使用时，可以留有一定的动态余量（headroom，d）。同时可以设置 FIFO 的水线深度（watermark level）来调整 CPU 和 DMA 的访问频次。动态余量和水线深度应该根据应用需求来进行适当选择。为了最大化 FMAC 的吞吐率，输入 buffer 应该永不为空，因此 d 应该大于水线深度，以防止因为中断和 DMA 操作的延迟导致 FMAC 等待数据输入而暂时停滞。另一方面，如果输入数据的填装速度小于被 FMAC 处理的速度，输入缓存可能会被 FMAC 读空，并等待 CPU 或 DMA 写入下一个数据。d 可以等于水线深度，确保输入读取不会溢出。

12.3.2 输入缓存

X1 和 X2 buffer 为 FMAC 的输入缓存，用于存储输入数据。FMAC 的每次乘法操作从 X1 读取一个数据，从 X2 读取一个数据，然后相乘。FMAC 内部硬件指针，控制读取数据相对于缓存基地址的偏移。指针由硬件根据操作配置自动管理。

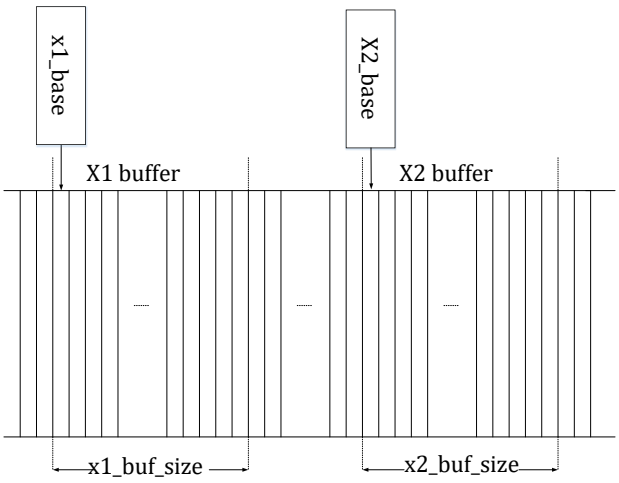


图 12-2 输入缓存区域

X1 buffer 可以作为循环 buffer 使用，只要 buffer 有空闲空间即可写入输入数据。buffer 在操作前可以进行初始化，也可以不进行初始化。如果操作开始时 buffer 内部没有数据，则 buffer 空的标志会触发 CPU 或 DMA 进行新数据装填直到有足够的数据进行滤波计算。

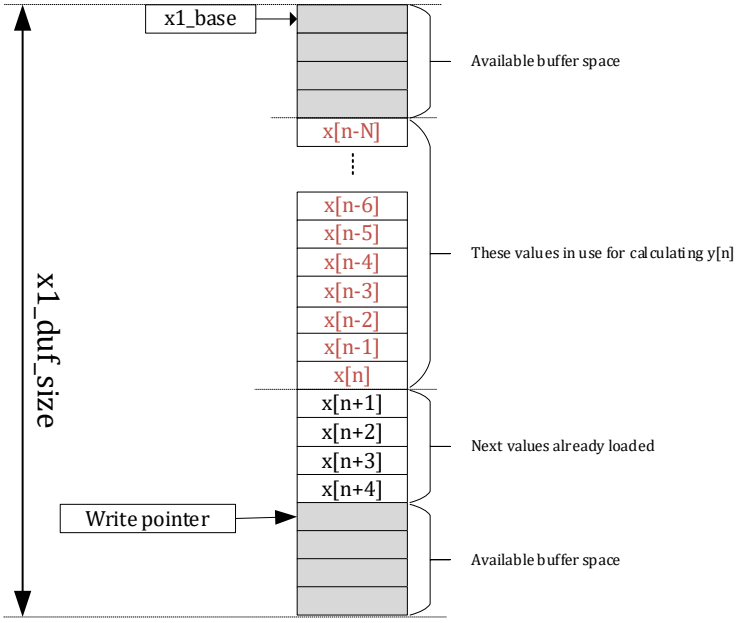


图 12-3 循环输入 buffer

X2 buffer 无法作为循环 buffer 使用，而且通常需要预装载。在滤波操作中，X2 buffer 用于存储滤波器系数。

当 X1 buffer 作为循环 buffer 使用时，x1_buf_size 应略大于当前操作所需数据个数（滤波器阶数），这样 buffer 可以一直存在未经 FMAC 使用的新数据。图 12-3 展示了 FMAC 在进行滤波操作时的缓存分布情况。当计算输出 y[n]时，FMAC 需要使用 N+1 个输入数据，x[n-N]至 x[n]。当计算完毕 y[n]，FMAC 开始使用 x[n+1-N]至 x[n+1]计算 y[n+1]，而最早输入的数据 x[n-N]不再被 FMAC 所需要，可以被新数据替换。

CPU 或 DMA 需要在 FMAC 需要数据时进行及时填装，否则 FMAC 会将 buffer 空标志置位，并暂停 FMAC 内部的滤波计算操作，直到新的数据填入 buffer。X1 buffer 硬件上保证了不会出现因为读取而造成的下溢出。

举例来说，如果采样滤波流程是受控于定时器或是 ADC 外设，则 buffer 会定期地进入空状态，因为 FMAC 处理数据的速度快于采样数据产生的速度。这是正常特性。

如果 buffer 内的空闲空间小于水线阈值(FMAC_X1BUFCFG.FULL_WM)，则 buffer 的满标志会置位。如果满标志未置位，则 FMAC 会产生中断请求，请求更多的数据填入 buffer。水线的存在使得一次中断填入多个数据成为可能，且不会发生 buffer 上溢出。

如果发生了上溢出，则 FMAC_SR.OVFL 会置位，且写入数据会被丢弃，写指针维持不变。X1 buffer 没有下溢出标志。

X1 buffer 在滤波操作器件的读写动作如图 12-4 所示。本例为一个 8 阶 FIR 滤波器，水线阈值设置为 4。

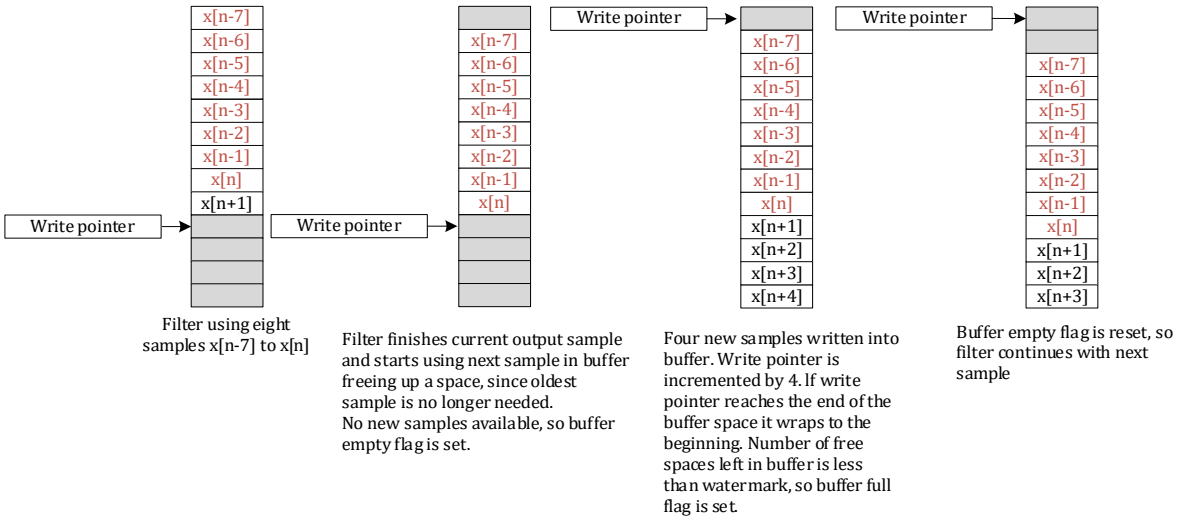


图 12-4 循环输入 buffer 的读写操作

12.3.3 输出缓存

Y buffer 用于存储累加器的输出，并等待 CPU 或 DMA 读取。每读取一次，Y buffer 的读指针加 1，当指针遇到 buffer 尾时，自动回到 buffer 基地址。

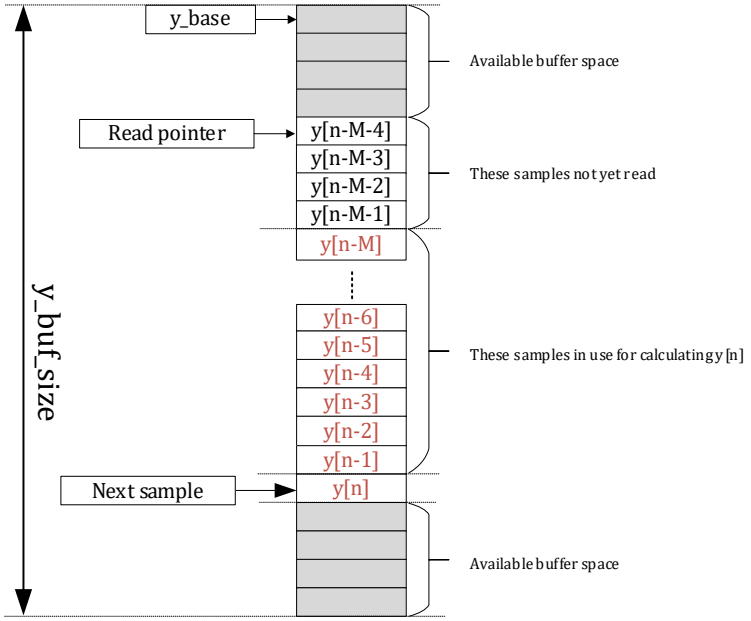


图 12-5 循环输出 buffer

Y buffer 也可以作为循环 buffer 使用。如果待写入的输出数据地址与读指针重合，即与未读取的

结果数据重叠，则 buffer 满标志置位，FMAC 会中断计算操作直到 Y buffer 中的结果数据被读走。

当进行 IIR 滤波操作时，Y buffer 需要预装载 M 个历史输出数据 y[n-M]至 y[n-1]，用于计算下一个输出数据 y[n]。每次 Y buffer 填入一个新的数据，最旧的数据 y[n-M]会被替换掉。

如果 Y buffer 内的未读取数据个数少于水线阈值（FMAC_YBUFCFG.EMPTY_WM），则 Y buffer 空标志置位。如果 Y buffer 不为空，则会通过中断请求 CPU 或 DMA 读取其中的输出数据。水线阈值的存在使得 CPU 可以运行一次中断服务函数读取多个输出数据，而无下溢出风险。但如果发生了下溢出，FMAC_CR.UNFL 的标志会置位。这种情况下 Y buffer 读指针维持不变，读取的数据为读指针指向的数据。

Y buffer 作为循环 buffer 使用时的读写操作如图 12-6 所示。本例为一个 7 阶 IIR 滤波器，水线阈值设置为 4。

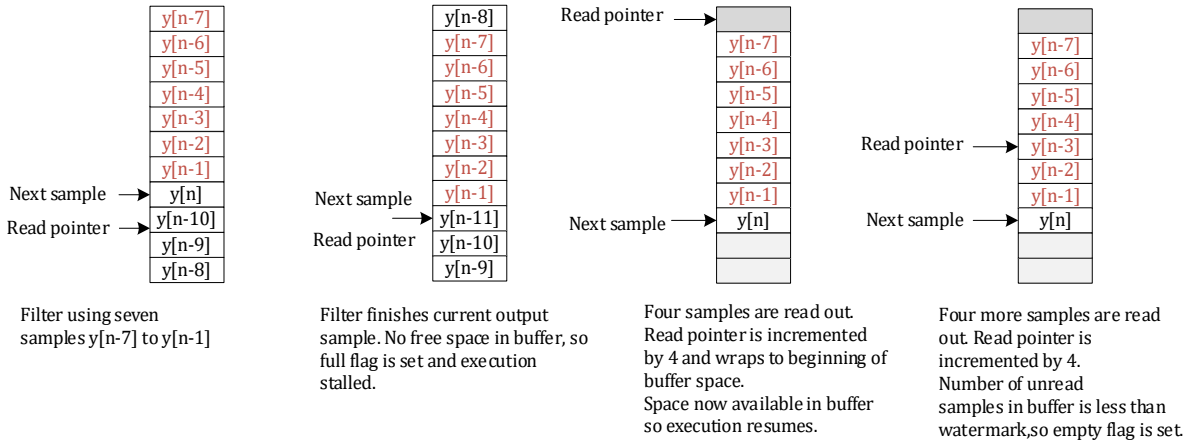


图 12-6 循环输出 buffer 读写操作

12.3.4 初始化函数

FMAC 除了常规的滤波操作外，还有初始化操作用于初始化输入 buffer 数据。初始化操作通过向 FMAC_PARAM.FUNC 写入适当的值，并将 FMAC_PARAM.START 写入为 1 来触发。同时 FMAC_PARAM.P/Q 字段也需要进行适当配置。当初始化操作完成后，START 由硬件自动复位为 0。

12.3.4.1 填装 X1 buffer

X1 buffer 初始化函数可以用于 X1 buffer 的预填装，从 X1_BASE 开始，连续写入 N+1 个数据到 FMAC_WDATA 寄存器，输入数据写入到本地存储，同时写指针跟随递增。初始化完成后，写指针指向 X1_BASE+N+1。

X1 buffer 初始化函数可以用于向 X1 buffer 填入向量元素，或是滤波器的输入数据。其相关参数如下：

P 表示待填入 X1 buffer 的数据个数 N+1；

Q 和 R 未使用

CPU 或 DMA 向 FMAC_WDATA 写入 N+1 个数据后，初始化操作完成。

12.3.4.2 填装 X2 buffer

X2 buffer 初始化函数可以用于 X2 buffer 的预填装，从 X2_BASE 开始，连续写入 N+1+M 个数据到 FMAC_WDATA 寄存器，输入数据写入到本地存储，同时写指针跟随递增。初始化完成后，写指针指向 X2_BASE+N+1+M。

X2 buffer 初始化函数可以用于向 X2 buffer 填入向量元素，或是滤波器的系数。其相关参数如下：

P 表示待填入 X2 buffer 的数据个数 N+1，从 X2_BASE 开始填装；

Q 表示待填入 X2 buffer 的数据个数 M，从 X2_BASE+N+1 开始填装；

R 未使用

CPU 或 DMA 向 FMAC_WDATA 写入 N+1+M 个数据后，初始化操作完成。

对于 FIR，x2 buffer 初始化系数数据从 h(0)开始，然后是 h(1),h(2),...,直到 h(N)。

对于 IIR，x2 buffer 的初始化系数数据从 b(0)开始，然后是 b(1),b(2),...,直到 b(N)，为 x 的系数；之后是 a(1),a(2),...,a(M)，为 y 的系数。y buffer 初始化样本数不超过 M 个。

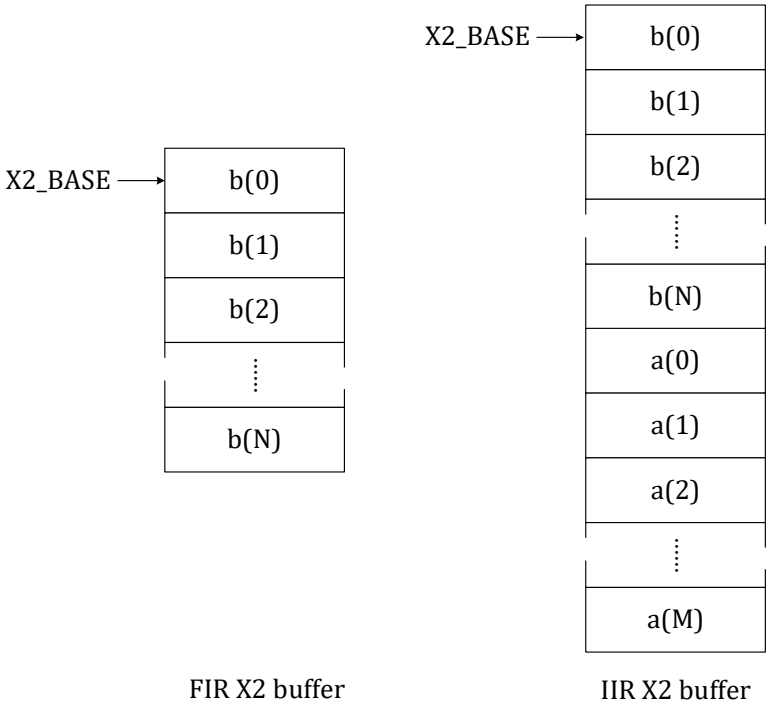


图 12-7 FIR 和 IIR 滤波器 x2 buffer 系数排列

12.3.4.3 填装 Y buffer

Y buffer 初始化函数可以用于 Y buffer 的预填装，从 Y_BASE 开始，连续写入 N+1 个数据到 FMAC_WDATA 寄存器，数据写入到本地存储，同时写指针跟随递增。初始化完成后，写指针指向 Y_BASE+N+1。

Y buffer 初始化函数可以用于向 Y buffer 填入 IIR 滤波中的反馈数据。其相关参数如下：

P 表示待填入 Y buffer 的数据个数 N+1，从 Y_BASE 开始填装；

Q 和 R 未使用

CPU 或 DMA 向 FMAC_WDATA 写入 N+1 个数据后，初始化操作完成。

12.3.5 滤波器函数

FMAC 可用于执行 FIR、IIR 等滤波操作。通过向 FMAC_PARAM.FUNC 写入适当的值，并将 FMAC_PARAM.START 写入为 1 可以触发滤波操作开始执行。同时，P/Q/R 等字段也需要设置适当的值。滤波操作会持续进行直到 START 位被软件复位。

12.3.5.1 FIR 滤波器（卷积）

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

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

此操作对长度为 N+1 的向量 B 和无限长的向量 X 进行卷积。$Y_n = B \cdot X_n$，其中$X_n = [x_{n-N}, \dots, x_n]$，包含有 X 的 N+1 个元素，下标为 n-N 到 n。这个操作与 FIR 滤波相对应，向量 B 即为滤波器系数，向量 X 为待滤波的采样数据。

滤波器结构如图 12-8 所示。

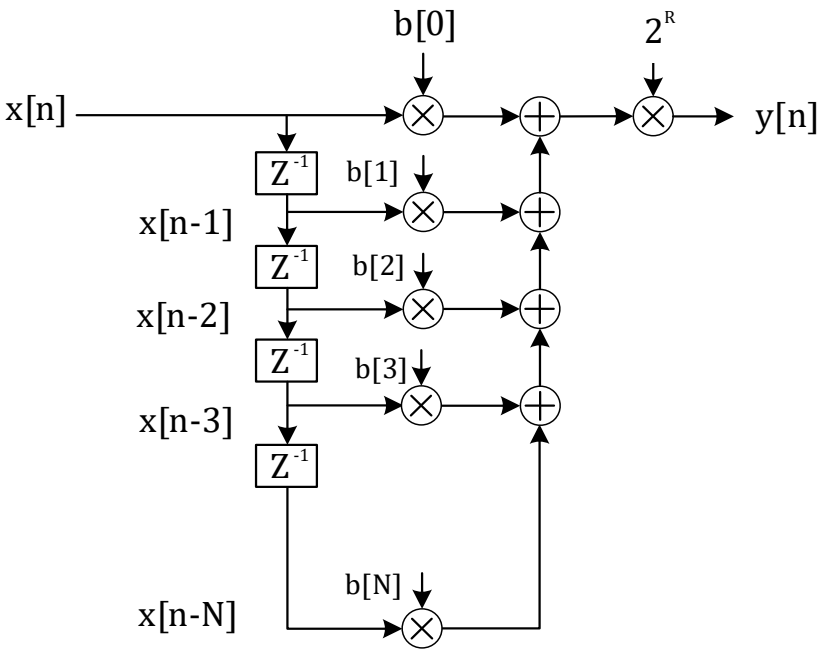


图 12-8 FIR 滤波器结构

FIR 滤波操作也可以用于计算向量的相关，这时需要将向量 B 首尾元素互换。

输入：

X1 buffer 存储有向量 X，是长度为 N+1+d 的循环 buffer，

X2 buffer 存储有向量 B，是长度为 N+1 的固定 buffer

输出：

Y buffer 存储有输出数据 y_n，是长度为 d 的循环 buffer。

参数：

P 为向量 B 的长度，N+1，取值范围是 2~127

R 是累加器输出增益，存入 Y buffer 的数据需要乘以 2^R，R 的范围是 0~7

Q 未使用。

FIR 滤波操作从 START 被置位为 1 开始持续进行，直到 START 被软件复位为 0 停止。

12.3.5.2 IIR 滤波器

$$\underline{Y}=\underline{B}*\underline{X}+\underline{A}*\underline{Y'}$$

$$y_n = 2^R \cdot \left(\sum_{k=0}^{N} b_k x_{n-k} + \sum_{k=1}^{M} a_k y_{n-k} \right)$$

滤波器输出 Y 是长度为 N+1 的向量 B 与无限长度向量 X 的卷积，加上 Y 的历史数据与长度为 M 的反馈向量 A 的卷积。$Y_n = B \cdot X_n + A \cdot Y_n$，其中$X_n = [x_{n-N}, \dots, x_n]$，包含有 X 的 N+1 个元素，下标为 n-N 到 n，$Y_{n-1} = [y_{n-M}, \dots, y_{n-1}]$，包含有 Y 的 M 个元素，下标为 n-M 到 n-1。

直接 1 型 IIR 滤波器结构如图 12-9 所示。

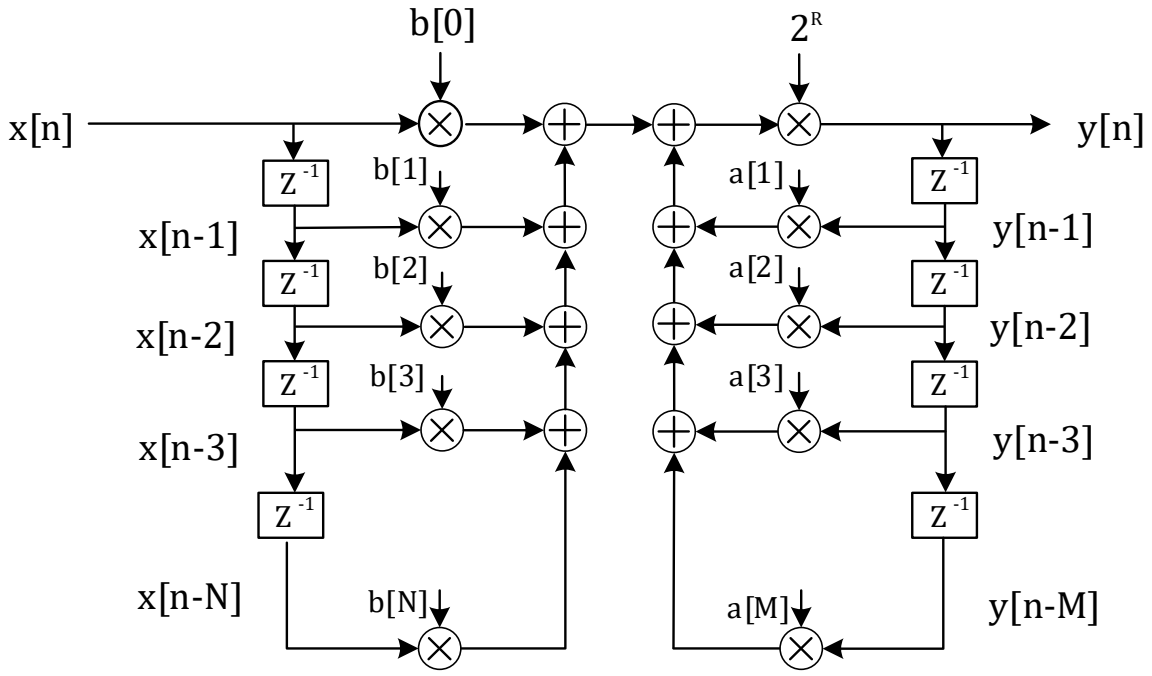


图 12-9 直接 1 型 IIR 滤波器结构

输入：

X1 buffer 存储有向量 X，是长度为 N+1+d 的循环 buffer，

X2 buffer 存储有向量 B 和向量 A，格式为 $(b_0, b_1, b_2..., b_N, a_1, a_2, ..., a_M)$，是长度为 M+N+1 的固定 buffer

输出：

Y buffer 存储有输出数据 y_n，是长度为 M+d 的循环 buffer。

参数：

P 为向量 B 的长度，N+1，取值范围是 2~127

Q 为向量 A 的长度，M，取值范围是 1~63

R 是累加器输出增益，存入 Y buffer 的数据需要乘以 2^R，R 的范围是 0~7

IIR 滤波操作从 START 被置位为 1 开始持续进行，直到 START 被软件复位为 0 停止。

12.3.6 定点数表示

FMAC 以定点数的数据格式进行计算操作。输入和输出数据都是 Q1.15 的定点数。

在 Q1.15 个时钟，数值表示为 1bit 符号位，和 15bit 的小数部分。数值范围为-1（0x8000）至 $1-2^{-15}$（0x7FFF）。

累加器位宽 26bit，其中 22bit 为小数部分，4bit 为整数和符号位，即 Q4.22 格式。因此累加器

和的数值范围为-8（0x2000000）至 7.99999976（0x1FFFFFF）。可编程增益步长为 6dB（即一档变化为原来的 2 倍），增益范围为 0db~42db，对应 R 的设置为 0~7，即增益范围为 2^0~2^7，增益施加于累加器输出，数据写入 Y buffer 之前。

注意，累加器的中间结果并没有饱和处理，因此如果数值超出定点数表示范围则会溢出。部分和可能会在中间计算过程中超出-8（0x2000000）至 7.99999976（0x1FFFFFF）的表示范围，但暂时的溢出并不会导致最终结果出错，因为滤波器输出的有效数据实际上位于数值的低位部分，只要最终结果再次回到定点数表示范围内则最终结果仍是正确的。但 FMAC_SR.SAT 会在溢出发生时置位。如果使能了 FMAC_CR.SATIEN，则会发生中断请求。这一特性有助于调试滤波器。

累加器的最终输出（经过增益处理之后）可以选择是否进行饱和处理。如果设置了 FMAC_CR.CLIPEN，则最终输出数据会被钳位在 Q1.15 的表示范围内，即-1（0x8000）至 $1-2^{-15}$（0x7FFF）。如果 CLIPEN 未使能，则会将累加器的高位截断丢弃，将低位存入 Y buffer。

12.3.7 使用 FMAC 实现 FIR 滤波器

FMAC 支持长度为 N+1 的 FIR 滤波操作，其中 N+1 为滤波器系数个数或抽头数。

长度为 N 的 FIR 滤波器，最小本地存储需求为 2N+3：

- N+1 个系数
- N+1 个输入数据
- 1 个输出数据

本地存储空间为 256，因此 FMAC 支持的最大 FIR 滤波器尺寸为 127 抽头。

为了最大化吞吐率，可以分别给输入输出缓存，留有一定额外空闲空间，大小分别为 d1 和 d2，以确保输入缓存永不为空，输出缓存永不为满。在这种设置下，本地存储需求为 2N+2+d1+d2。

buffer 配置如下：

X1_BUF_SIZE=N+1+d1；

X2_BUF_SIZE=N+1；

Y_BUF_SIZE=d2（或 1，如果不需要额外空闲空间）

buffer 基地址可以是本地存储的任意位置，但 X2 不应与其他 buffer 重叠，否则滤波器系数可能会被写覆盖，下面给出一个可能的基地址配置：

X2_BASE=0；

X1_BASE=N+1;

Y_BASE=2N+2+d1

FMAC_X1BUFCFG.FULL_WM 位域必须≤log2(d1)，否则 N 当输入数据写入还未完成 X1 buffer 就会被标记为满状态，导致输入数据无法继续写入。 同样，FMAC_YBUFCFG.EMPTY_WM 位域必须≤log2(d2)。

滤波器系数必须通过 X2 初始化操作进行预装载。X1 buffer 可以进行预装载，填充数据可以为

任意个，最多为 N+1 个，也可以不做预装载。Y buffer 在进行 FIR 操作时无需预装载，因为 FIR 不存在反馈路径。

完成 buffer 的配置和初始化之后，可以对 FMAC_CR 进行配置来指定 FMAC_WDATA 写入的方式和 FMAC_RDATA 读取的方式。有三种方式可选:

轮询:无 DMA 请求或中断请求产生。软件查询 X1_FULL 标志是否为低,为低则通过 FMAC_WDATA 向 X1 buffer 写入数据，查询 Y_EMPTY 标志是否为低，为低则通过 FMAC_RDATA 从 Y buffer 读取数据。

中断：当 X1_FULL 标志为低时，产生中断请求 CPU 通过 FMAC_WDATA 向 X1 buffer 写入数据，当 Y_EMPTY 为低时，产生中断请求 CPU 通过 FMAC_RDATA 从 Y buffer 读取数据。

DMA：当 X1_FULL 标志为低时，产生 DMA 传输请求，通过 FMAC_WDATA 向 X1 buffer 写入数据，当 Y_EMPTY 为低时，产生 DMA 传输请求，通过 FMAC_RDATA 从 Y buffer 读取数据。

读写操作方式可以不同。但一般不建议同一个操作同时使用中断和 DMA 请求，如果同时使能中断和 DMA，通常是 DMA 进行传输，中断不再进行传输。可选的读写操作方式组合如下

表 12-1 可选的读写操作方式组合

WIEN	RIEN	DMAWEN	DMAREN	X1 Write	Y Read
0	0	0	0	轮询	轮询
0	1	0	0	轮询	中断
1	0	0	0	中断	轮询
1	1	0	0	中断	中断
0	0	0	1	轮询	DMA
0	0	1	0	DMA	轮询
0	0	1	1	DMA	DMA
0	1	1	0	DMA	中断
1	0	0	1	中断	DMA

FIR 滤波操作启动前，需要对 FMAC_PARAM 进行如下配置:

FUNC=8（FIR 滤波功能）

P=N+1（滤波器系数个数）

Q=任意值（Don’t Care）

R=Gain

START=1（启动滤波操作）

如果填入 X1 buffer 的数据个数少于 $N+1+d-2^{FULL\_WM}$，则 X1_FULL 标志维持为低。如果 FMAC_CR.WIEN 为 1，则立即产生中断，请求 CPU 向 FMAC_WDATA 寄存器写入 2^{FULL_WM} 个新的采样数据。中断请求会一直存在,直到 FMAC_SR.X1_FULL 置位为 1。中断服务函数通常应该在写入 2^{FULL_WM} 个数据后查询 X1_FULL 标志，反复写入 2^{FULL_WM} 个新数据直到 X1_FULL 置位。

在 X1 buffer 写入了至少 N+1 个输入数据之后（包含预装载的数据个数），FMAC 开始计算第一个输出数据。

当 2^{EMPTY_WM} 个输出数据写入 Y buffer 之后，FMAC_SR.Y_EMPTY 标志置低。如果 FMAC_CR.RIEN 为 1，则立即产生中断，请求 CPU 从 FMAC_RDATA 读取 2^{EMPTY_WM} 个数据。中断请求会一直存在，直到 FMAC_SR.Y_EMPTY 置位为 1。中断服务函数通常应该在读取 2^{EMPTY_WM} 个数据后查询 Y_EMPTY 标志，反复读取 2^{EMPTY_WM} 个新数据直到 Y_EMPTY 置位。

滤波操作按照这种方式持续进行，直到软件向 START 写入 0。

12.3.8 使用 FMAC 实现 IIR 滤波器

FMAC 支持长度为 N+1 的 IIR 滤波操作，其中 N+1 为滤波器前馈系数个数或抽头数。滤波器反馈系数个数 M 可以是 1~N 的任意值。目前 FMAC 只能实现直接 1 型 IIR 滤波器，其他类型的 IIR 滤波器需要先转换为直接 1 型。

前馈系数为 N+1 个，反馈系数为 M 个的 IIR 滤波器，最小本地存储需求为 2N+2+2M：

- N+1+M 个系数
- N+1 个输入数据
- M 个输出数据

本地存储空间为 256，如果 M=N，FMAC 支持的最大 IIR 滤波器尺寸为 N+1=64。

为了最大化吞吐率，可以分别给输入输出缓存，留有一定额外空闲空间，大小分别为 d1 和 d2，以确保输入缓存永不为空，输出缓存永不为满。在这种设置下，本地存储需求为 2N+2M+2+d1+d2。

buffer 配置如下：

X1_BUF_SIZE=N+1+d1；

X2_BUF_SIZE=N+1+M；

Y_BUF_SIZE=M+d2

buffer 基地址可以是本地存储的任意位置，但不应与其他 buffer 重叠，下面给出一个可能的基地址配置：

X2_BASE=0；

X1_BASE=N+1+M;

Y_BASE=2N+2+M+d1

FMAC_X1BUFCFG.FULL_WM 位域必须≤log2(d1)，否则 N 当输入数据写入还未完成 X1 buffer 就会被标记为满状态，导致输入数据无法继续写入。 同样，FMAC_YBUFCFG.EMPTY_WM 位域必须≤log2(d2)。

滤波器系数（N+1 个前馈系数+M 个反馈系数）必须通过 X2 初始化操作进行预装载。X1 buffer 可以进行预装载，填充数据可以为任意个，最多为 N+1 个，也可以不做预装载。Y buffer 可以进行预装载，填充数据可以为任意个，最多为 M 个，也可以不做预装载

完成 buffer 的配置和初始化之后，可以对 FMAC_CR 进行配置来指定 FMAC_WDATA 写入的方式和 FMAC_RDATA 读取的方式。可选的读写操作方式组合可以参考 FIR 章节的表 12-1。

IIR 滤波操作启动前，需要对 FMAC_PARAM 进行如下配置：

FUNC=9（IIR 滤波功能）

P=N+1（滤波器前馈系数个数）

Q=M（滤波器反馈系数个数）

R=Gain

START=1（启动滤波操作）

如果填入 X1 buffer 的数据个数少于 N+1+d-2^{FULL_WM}，则 X1_FULL 标志维持为低。如果 FMAC_CR.WIEN 为 1，则立即产生中断，请求 CPU 向 FMAC_WDATA 寄存器写入 2^{FULL_WM} 个新的采样数据。中断请求会一直存在，直到 FMAC_SR.X1_FULL 置位为 1。中断服务函数通常应该在写入 2^{FULL_WM} 个数据后查询 X1_FULL 标志，反复写入 2^{FULL_WM} 个新数据直到 X1_FULL 置位。

在 X1 buffer 写入了至少 N+1 个输入数据之后（包含预装载的数据个数），FMAC 开始计算第一个输出数据。第一个输出数据的计算需要使用 X1 buffer 中的 N+1 个输入数据和 Y buffer 中的 M 个历史输出数据。第一个输出数据会写入到 Y buffer 的 Y_BASE+M 地址。

当 2^{EMPTY_WM} 个输出数据写入 Y buffer 之后，FMAC_SR.Y_EMPTY 标志置低。如果 FMAC_CR.RIEN 为 1，则立即产生中断，请求 CPU 从 FMAC_RDATA 读取 2^{EMPTY_WM} 个数据。中断请求会一直存在，直到 FMAC_SR.Y_EMPTY 置位为 1。中断服务函数通常应该在读取 2^{EMPTY_WM} 个数据后查询 Y_EMPTY 标志，反复读取 2^{EMPTY_WM} 个新数据直到 Y_EMPTY 置位。

滤波操作按照这种方式持续进行，直到软件向 START 写入 0。

12.3.9 滤波器初始化示例

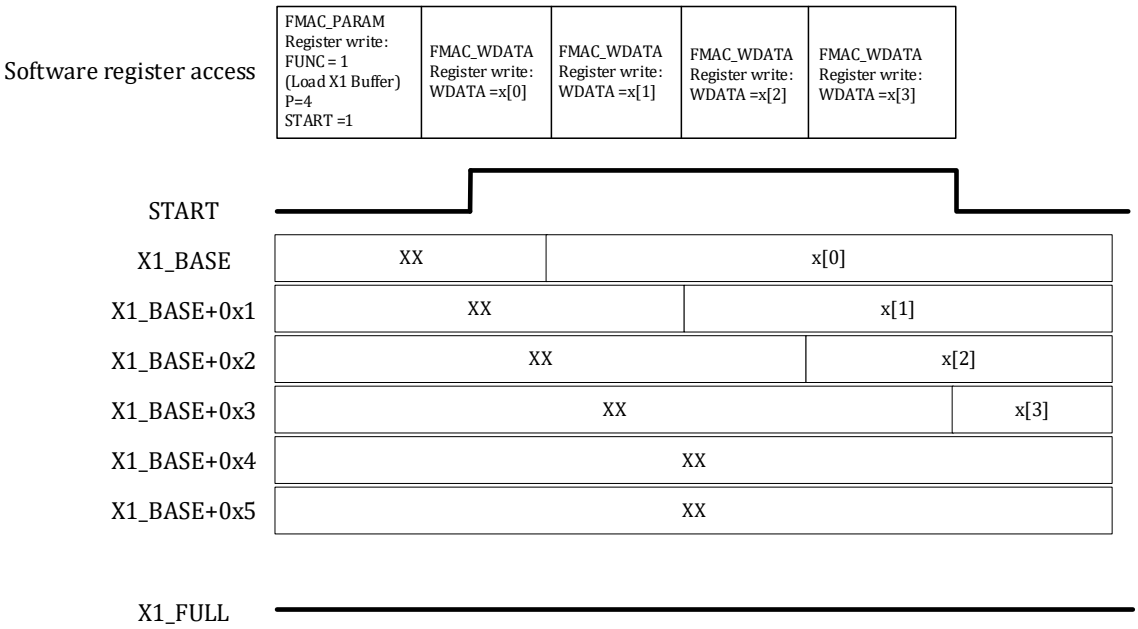


图 12-10 X1 buffer 初始化

图 12-10 展示了 X1 buffer 配置为 P=4 的预填装过程。X1 buffer 大小 X1_BUF_SIZE=6。向

FMAC_PARAM.START 写入 1，初始化操作启动。4 个数据写入 FMAC_WDATA 寄存器，并存入 FMAC 本地存储，对应地址从 X1_BASE 开始递增。4 个数据写入完成后，START 被硬件复位为 0。此时 X1 buffer 存储有 4 个有效的输入数据，同时写指针指向下一次写入数据的位置 X1_BASE+0x4。

12.3.10 滤波器操作示例

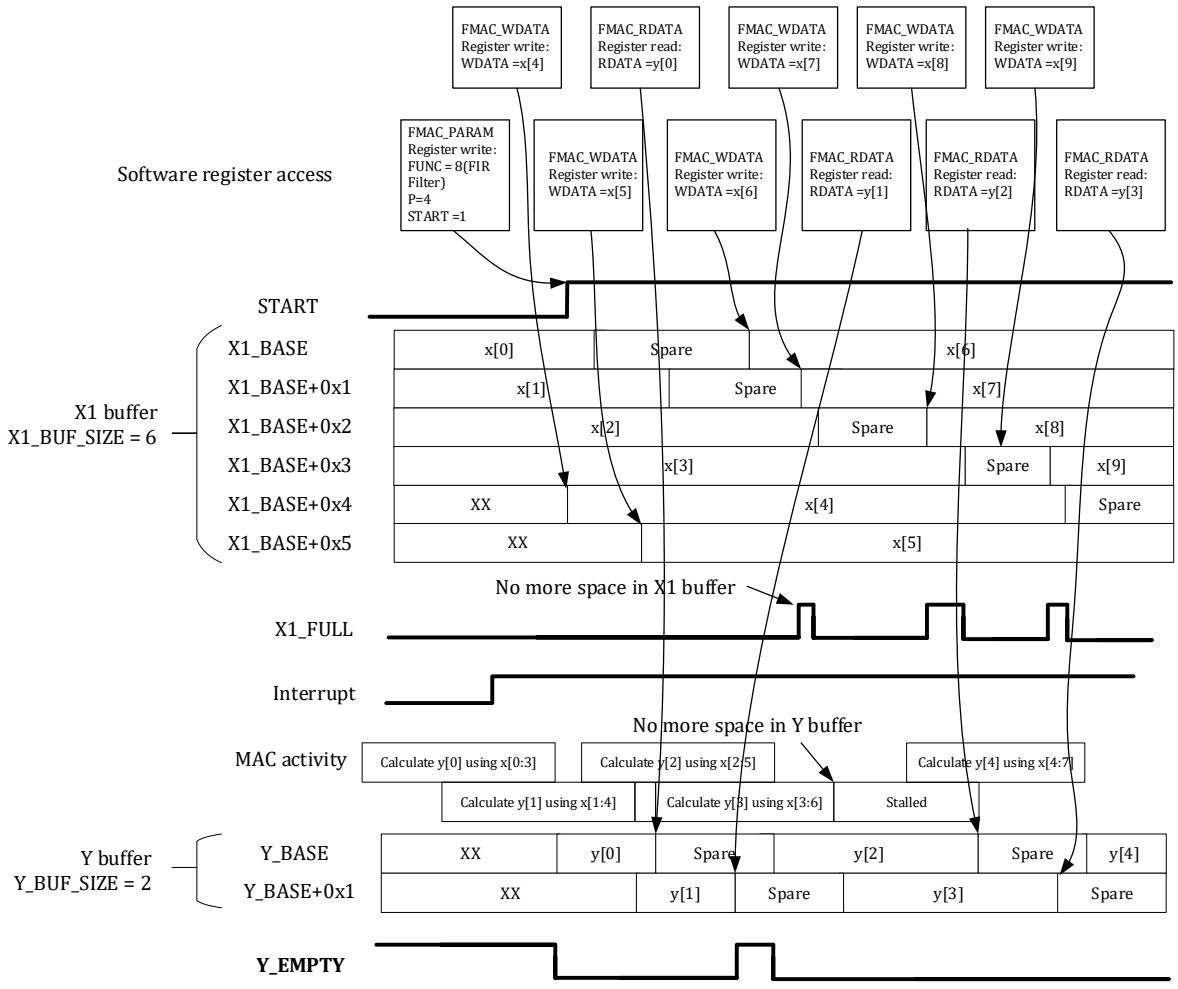


图 12-11 滤波器操作示例 1

图 12-11 展示了滤波器操作启动的情形。滤波器抽头数为 4（P=4）。X1 buffer 大小为 6，Y buffer 大小为 2。FULL_WM 和 EMPTY_WM 均为 0。滤波器启动前，X1 buffer 已经预填装了 4 个有效输入数据 x[0:3]。因此一旦 START 被写入为 1，滤波器即开始计算第一个输出数据 y[0]。因为 X1 buffer 还存在两个空闲位置，X1_FULL 标志维持为 0。FMAC 产生中断，请求新的输入数据。CPU 通过 FMAC_WDATA 再次写入两个输入数据 x[4]和 x[5]，两个数据被存入 FMAC 本地存储中的 X1 buffer 中。

与此同时，FMAC 计算完成第一个输出数据 y[0]，将其写入 Y buffer，使得 Y buffer 不再为空，Y_EMPTY 变低。计算完成 y[0]后，x[0]可以被丢弃，释放一个 X1 buffer 空间（位于 X1_BASE）。FMAC 立即启动计算第二个输出数据 y[1]，此时 y[1]所需的所有输入数据 x[1:5]均已就绪。

由于 Y_EMPTY 为低，FMAC 提起中断，请求 CPU 通过 FMAC_RDATA 读取 Y buffer 中的 y[0]，释放 Y buffer 的空间。此时 y[1]尚未计算完完成，Y buffer 中不存在有效输出数据，Y_EMPTY 变高。由于 X1 buffer 仍有空闲空间，FMAC 继续提起中断请求，请求 CPU 通过 FMAC_WDATA 向 X1 buffer

写入新的输入数据 x[6]等。

本例中,CPU 可以以更快的速度填入输入数据,填入的速度比 FMAC 处理速度更快,因此 X1_FULL 标志会偶尔变高为 1。但 CPU 读取 Y buffer 的速度不够快，导致 FMAC 偶尔需要等待 Y buffer 被读取以释放空间。这导致滤波操作没有全速进行,原因是滤波器长度比较小,而 CPU 处理速度比较慢。在这种情况下，增加 buffer 大小对于提高滤波器吞吐率**没有**改善效果。

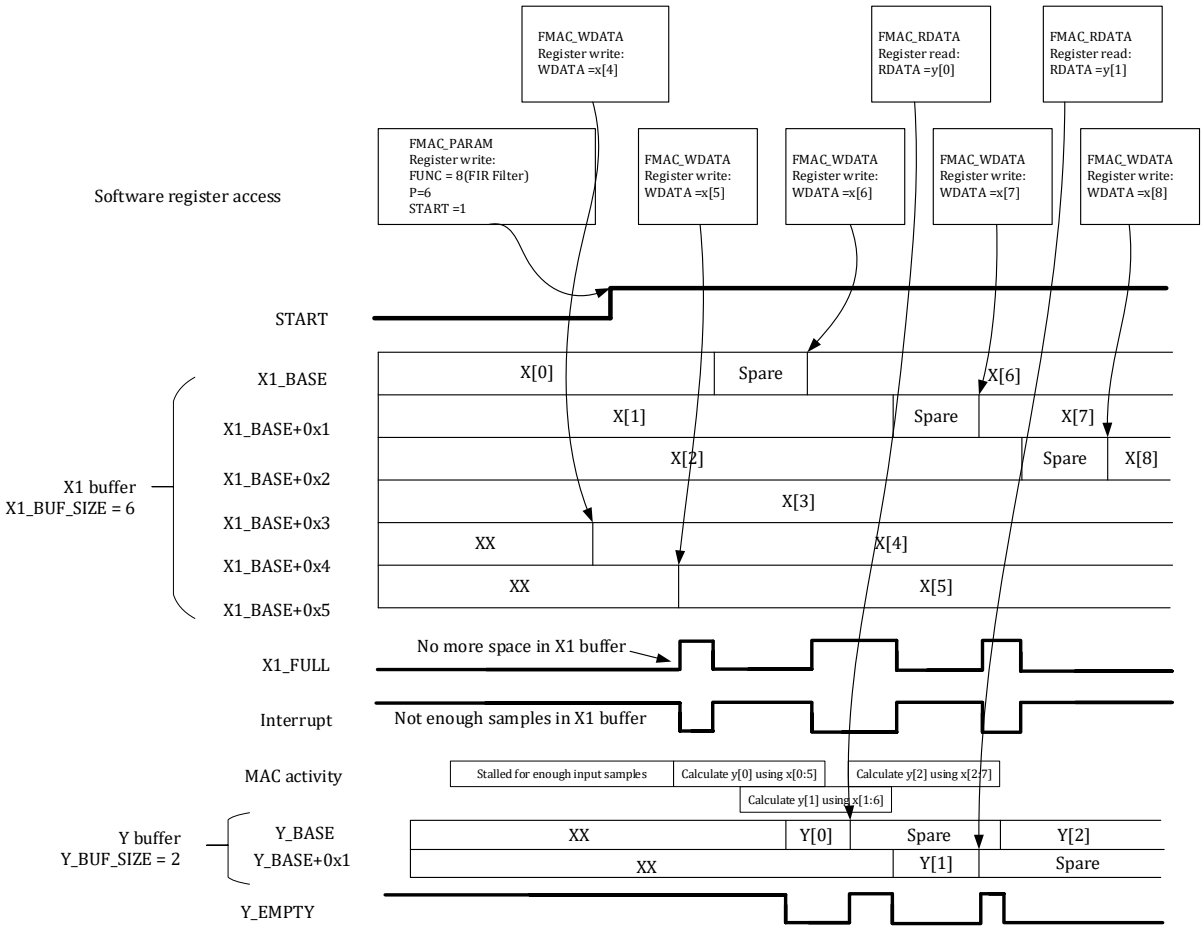


图 12-12 滤波器操作示例 2

图 12-12 的例子中是滤波器抽头数为 6 的 FIR。X1 buffer 大小为 6,Y buffer 大小为 2。FULL_WM 和 EMPTY_WM 均为 0。START 写入 1 之前，X1 buffer 中已经预填装了 4 个有效的输入数据 x[0:3]。但此时 X1 buffer 中的输入数据仍然不足，X1_FULL 标志没有置位。FMAC 继续提起中断，请求 CPU 填入更多的输入数据。且 FMAC 的滤波计算因等待足够的输入数据而处于暂停状态。

CPU 通过 FMAC_WDATA 写入两个新的输入数据 x[4]和 x[5]，存入 X1 buffer 中的空闲位置。此时 X1 buffer 中有 6 个有效输入数据，X1_FULL 变高为 1，FMAC 不再提起中断请求。FMAC 开始计算得到输出数据 y[0]，并将其存入 Y buffer。此时 x[0]被丢弃，释放掉 X1 buffer 的一个空间位置，X1_FULL 变低。与此同时 Y_EMPTY 变低。buffer 标志变化导致中断再次被提起，请求 CPU 写入新的输入数据，并读取输出数据。FMAC 处于停止状态，直到新的输入数据写入。

本例中 CPU 必须等待 FMAC 完成第一个输出数据的计算才能写入新的输入数据，因此第一个输出数据需要用到 x[0]，只有完成计算 x[0]位置才会成为 X1 buffer 中的空闲位置。因此 X1 buffer 会偶尔进入数据不足的状态，导致 FMAC 停止计算。通过增加 buffer 空间大小可以改善这个问题。

12.3.11 滤波器设计提示

FMAC 的设计架构对数字滤波器实现有某些限制

1. 直接 2 型和转置型需要转换为直接 1 型，才能在 FMAC 中实现

2. IIR 滤波器级联需要合并成单级滤波器，或者分开实现。如果选择分开实现，FMAC 的 X2 buffer 中可能存储有多组滤波器系数，在进行不同的滤波操作时需要更改 X2_BASE。实现多级滤波器的比较有效的方式是，先对 X1 buffer 进行预装载，然后使用第一级滤波器系数进行计算，将结果存入芯片的主存储器（从 FMAC 的本地存储中读出存入 SRAM）。修改 X2_BASE 指向第二级滤波器系数，然后使用第一级滤波器的输出数据再次初始化 X1 buffer。如此往复。最后一级滤波的输出数据被读入芯片主存之后，X1 buffer 可以填入新一轮的输入数据，FMAC 即可开启新一轮的计算。需要注意的是，每一级的 N+1 个输入数据都需要使用前一级的 N 的数据和一个新输入数据，保证各级间的处理连续。同样，IIR 滤波需要使用 M 个本级历史输出数据对输出 buffer 进行初始化。

3. 直接 1 型的 IIR 滤波器可能在累加器阶段产生绝对值很大的部分和，比如输入信号是一个很大的阶跃信号，或者滤波器系数绝对值大于 1。累加器位宽为 26bit，在不进行高位截断（wrapping）的情况下可处理的最大数值范围是 0x1FFFFFF~0x2000000，在 Q3.23 定点数格式下对应 3.99999988~-4。高位截断并不会导致最终结果错误，因为滤波输出的信息保存在数据的低位，高位部分大部分时间是维持不变的，可以视为 DC 分量，只要最终结果在表示范围内，则最终结果仍是正确的，可以看做高位截断被撤销了。当然，IIR 初值问题仍需要谨慎考虑，否则存在出错可能。

4. IIR 滤波器有前馈（分子）系数$[b_0, b_1, ..., b_N]$和反馈（分母）系数$[1, a_1, ..., a_M]$。某些 IIR 滤波器需要反馈系数绝对值大于 1 以得到一个陡峭的滚降频响曲线。但由于滤波器系数是 Q1.15 编码的，是无法直接表示大于 1 的系数的。可以利用增益 2^{-R}，使得 $2^{-R}[1, a_1, ..., a_M]$是小于 1 的，即可以用 Q1.15 定点数编码表示。在累加器的输出位置再将增益 2^R 乘回。但这个操作会影响信噪比。

12.4 寄存器

12.4.1 地址分配

FMAC 的基地址是 0x4001_7000，寄存器列表如下：

表 12-2 FMAC 寄存器列表

名称	偏移地址	说明
FMAC_X1BUFCFG	0x00	FMAC X1 buffer 配置寄存器
FMAC_X2BUFCFG	0x04	FMAC X2 buffer 配置寄存器
FMAC_YBUFCFG	0x08	FMAC Y buffer 配置寄存器
FMAC_PARAM	0x0C	FMAC 参数寄存器
FMAC_CR	0x10	FMAC 控制寄存器
FMAC_SR	0x14	FMAC 状态寄存器
FMAC_WDATA	0x18	FMAC 写数据寄存器
FMAC_RDATA	0x1C	FMAC 读数据寄存器

12.4.2 寄存器描述

12.4.2.1FMAC_X1BUFCFG FMAC X1 buffer 配置寄存器

地址：0x4001_7000

复位值：0x0

表 12-3 FMAC_X1BUFCFG FMAC X1 buffer 配置寄存器

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X1_BUF_SIZE								X1_BASE							
RW								RW							
0								0							

位置	位名称	说明
[31:26]	未使用	NA
[25:24]	FULL_WM	X1 buffer 满水线阈值，当 X1 buffer 中的空闲位置少于 2^{FULL_WM} 时，X1 buffer 满标志置位。 0：水线阈值为 1 1：水线阈值为 2 2：水线阈值为 4 3：水线阈值为 8 设置水线阈值大于 1 可以在一次中断处理中传输多个数据到 X1 buffer 中，当使用 DMA 写 X1 buffer 时水线阈值应该设置为 1。
[23:16]	未使用	NA
[15:8]	X1_BUF_SIZE	X1 buffer 大小，X1 buffer 中包含的 16bit 有符号数个数，buffer 尺寸最小值为滤波器前馈阶数（+水线阈值-1）
[7:0]	X1_BASE	X1 buffer 基地址

12.4.2.2FMAC_X2BUFCFG FMAC X2 buffer 配置寄存器

地址：0x4001_7004

复位值：0x0

表 12-4 FMAC_X2BUFCFG FMAC X2 buffer 配置寄存器

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X2_BUF_SIZE								X2_BASE							
RW								RW							
0								0							

位置	位名称	说明
[31:16]	未使用	NA
[15:8]	X2_BUF_SIZE	X2 buffer 大小，X2 buffer 中包含的 16bit 有符号数个数，这个位域在操作进行中（START=1）时无法修改
[7:0]	X2_BASE	X2 buffer 基地址，这个位域在操作进行中（START=1）时无法修改

12.4.2.3 FMAC_YBUFCFG FMAC Y buffer 配置寄存器

地址：0x4001_7008

复位值：0x0

表 12-5 FMAC_YBUFCFG FMAC Y buffer 配置寄存器

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

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

位置	位名称	说明
[31:26]	未使用	NA
[25:24]	EMPTY_WM	Y buffer 空水线阈值，当 Y buffer 中的有效数据数量少于 2^{EMPTY_WM} 时，Y buffer 空标志置位。 0：水线阈值为 1 1：水线阈值为 2 2：水线阈值为 4 3：水线阈值为 8 设置水线阈值大于 1 可以在一次中断处理中从 Y buffer 读取多个数据中，当使用 DMA 读取 Y buffer 时水线阈值应该设置为 1。
[23:16]	未使用	NA

[15:8]	Y_BUF_SIZE	Y buffer 大小，Y buffer 中包含的 16bit 有符号数个数。对于 FIR 滤波器，buffer 尺寸最小值为 1（+水线阈值），对于 IIR 滤波器，buffer 尺寸最小值为滤波器反馈阶数（+水线阈值）
[7:0]	Y_BASE	Y buffer 基地址

12.4.2.4FMAC_PARAM FMAC 参数寄存器

地址：0x4001_700C

复位值：0x0

表 12-6 FMAC_PARAM FMAC 参数寄存器

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

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

位置	位名称	说明
[31]	START	0：停止 FMAC 执行 1：开始 FMAC 执行（FMAC 执行过程中） 写入 1 触发 FUNC 指定操作开始执行，写入 0 可以停止执行中的操作，初始化操作完成后此位由硬件自动清零。
[30:28]	未使用	NA
[27:24]	FUNC	0: Reserved 1: Load X1 buffer 2: Load X2 buffer 3: Load Y buffer 4 to 7: Reserved 8: Convolution (FIR filter) 9: IIR filter (direct form 1) 此位域在 START=1 时无法修改
[23:16]	R	输入参数 R，Y 输出前移位幅度，即 Y 输出前乘以 2^R 倍 此位域在 START=1 时无法修改
[15:8]	Q	输入参数 Q，IIR 中反馈系数个数 此位域在 START=1 时无法修改
[7:0]	P	输入参数 P，FIR 及 IIR 中前馈系数个数 此位域在 START=1 时无法修改

12.4.2.5 FMAC_CR FMAC 控制寄存器

地址：0x4001_7010

复位值：0x0

表 12-7 FMAC_CR FMAC 控制寄存器

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLIPEN						DMAWEN	DMAREN				SATIEN	UNFLIEN	OVFLIEN	WIEN	RIEN
RW						RW	RW				RW	RW	RW	RW	RW
0						0	0				0	0	0	0	0

位置	位名称	说明
[31:17]	未使用	NA
[16]	RESET	复位 FMAC 模块，向此位写 1 可以将 FMAC 内部的读写指针，内部逻辑状态机，FMAC_SR 寄存器和 FMAC_PARAM 寄存器（包括 START）进行复位，其他寄存器不受影响。此位写入自动清零。
[15]	CLIPEN	累加器输出饱和使能 1：累加器增益控制后的输出被钳位在 Q1.15 范围内 0：累加器增益控制后的输出超出 Q1.15 范围时丢弃高位
[14:10]	未使用	NA
[9]	DMAWEN	DMA 写 X1 buffer 请求使能 0：禁用，无 DMA 写请求产生 1：使能，当 X1 不满时有 DMA 写请求产生 此位只能在 START=0 时进行改写
[8]	DMAREN	DMA 读请求使能 0：禁用，无 DMA 写请求产生 1：使能，当 Y 不空时有 DMA 读请求产生 此位只能在 START=0 时进行改写
[7:5]	未使用	NA
[4]	SATIEN	累加器饱和错误中断使能 0：禁用 1：使能，当 FMAC_SR.SAT 置位时产生中断请求
[3]	UNFLIEN	Y buffer 下溢出中断使能 0：禁用 1：使能，当 FMAC_SR.UNFL 置位时产生中断请求

[2]	OVFLIEN	X1 buffer 上溢出中断使能 0：禁用 1：使能，当 FMAC_SR.OVFL 置位时产生中断请求
[1]	WIEN	X1 buffer 写请求中断使能 0：禁用 1：使能，当 X1 buffer 不满时，向 CPU 发出写请求中断
[0]	RIEN	Y buffer 读请求使能 0：禁用 1：使能，当 Y buffer 不空时，向 CPU 发出读请求中断

12.4.2.6FMAC_SR FMAC 状态寄存器

地址：0x4001_7014

复位值：0x1

表 12-8 FMAC_SR FMAC 状态寄存器

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					SAT	UNFL	OVFL							X1_FULL	Y_EMPTY
					RO	RO	RO							RO	RO
					0	0	0							0	1

位置	位名称	说明
[31:11]	未使用	NA
[10]	SAT	饱和错误标志 当累加器输出结果超出 Q1.15 表示范围时，此位置位 0：未发生饱和事件 1：监测到饱和事件发生。如果 FMAC_CR.SATIEN 置位，则产生中断请求
[9]	UNFL	Y buffer 下溢出错误标志 当 Y buffer 中无有效数据且软件或 DMA 读取 FMAC_RDATA 时产生下溢出事件 0：未发生下溢出事件 1：监测到下溢出事件发生。如果 FMAC_CR.UNFLIEN 置位，则产生中断请求
[8]	OVFL	X1 buffer 上溢出错误标志 当 X1 buffer 中无空闲空间且软件或 DMA 通过 FMAC_WDATA 向 X1 buffer 写入数据时时产生上溢出事件 0：未发生上溢出事件

		1：监测到上溢出事件发生。如果 FMAC_CR.OVFLIEN 置位，则产生中断请求
[7:2]	未使用	NA
[1]	X1_FULL	X1 buffer 满标志 当X1 buffer中的空闲空间大小小于水线阈值时,X1 buffer被标志位满。空闲空间大小等于写指针与最近一次读指针之差。 0：X1 buffer 不满。如果 FMAC_CR.WIEN 置位则产生中断请求向 X1 buffer 写入数据。如果 FMAC_CR.DMAWEN 置位则请求 DMA 向 X1 buffer 写入数据，直到 X1_FULL 置位。 1：X1 buffer 满 此位由硬件自动置位清零，也可以通过向 FMAC_CR.RESET 写 1 清零
[0]	Y_EMPTY	Y buffer 空标志 当 Y buffer 中的有效数据数量小于水线阈值时，Y buffer 被标志为空。有效数据数量等于读指针与累加器写入数据目的地址之差。 0：Y buffer 不空。如果 FMAC_CR.RIEN 置位则产生中断请求从 Y buffer 读取数据。如果 FMAC_CR.DMAREN 置位，则请求 DMA 从 Y buffer 读取数据直到 Y_EMPTY 置位。 1：Y buffer 空 此位由硬件自动置位清零，也可以通过向 FMAC_CR.RESET 写 1 清零

12.4.2.7FMAC_WDATA FMAC 写入数据寄存器

地址：0x4001_7018

复位值：0x0

表 12-9 FMAC_WDATA FMAC 写入数据寄存器

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

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

位置	位名称	说明
[31:16]	未使用	NA
[15:0]	WDATA	写入数据，向此寄存器写入数据，数据被写入写指针指向的本地存储地址，写后写指针自动加 1

12.4.2.8FMAC_RDATA FMAC 读取数据寄存器

地址：0x4001_701C

复位值：0x0

表 12-10 FMAC_RDATA FMAC 读取数据寄存器

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

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

位置	位名称	说明
[31:16]	未使用	NA
[15:0]	RDATA	读出数据，从寄存器读取数据，数据被从读指针指向的本地存储地址读出，读后读指针自动加 1

13 ADC

13.1 概述

LKS32MC45x 系列芯片集成了 3 个 14BIT SARADC 转换核心， 27 个模拟 IO 输入信号/6 个 OPA 输出/温度传感器输出/模拟地均可作为 ADC 的被采样信号。主要有以下特性：

- 3 个 ADC 数字接口分别对应 3 个 ADC
- 2MSPS 采样率，32MHz 工作频率
- 每个 ADC 接口支持 16 路通道选择
- 每个 ADC 接口配备 2 个模拟看门狗，可同时检测上下阈值
- 支持过采样
- 支持软件、硬件触发功能
- 可以与 MCPWM、Timer 单元联动，触发指示信号可通过 GPIO 送出调试
- 支持自定义采样序列，序列次数和通道号可灵活配置
- 支持左对齐、右对齐模式
- 支持差分输入模拟信号，正负输入通道可配置
- 支持单段采样和两段采样
- 支持硬件过采样，过采样后有效位数可达 18bit 以上

ADC 采样的量词含义约定：

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

1 段采样：可能包含 1 次或若干次采样，若干次采样可以是相同的模拟信号，也可以是不同的模拟信号。采样开始通常由 MCPWM、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。

13.1.1 功能框图

每个 ADC 接口包括 16 个数据寄存器（ADC 16 次采样各个通道模拟量对应的数字量），以及若干控制寄存器。

可以同一时刻进行三个通道的采样。

数据寄存器 ADC_DATx 用于存储 ADC 第 x 次采样得到的数字量。被转换的模拟信号来源由寄存器 ADC_CHNx 中的某 4bit 进行选择（详见 13.2.3）。以 ADC_PCHN0 为例，位[3:0]选择第 0 次采样的模拟信号来源，ADC_CH0~ADC_CH14 任选，若 ADC_PCHN0[3:0]=0，ADC_PCHN0[7:4]=3，则第 0 个采样的模拟量正端信号来自 IO ADC_CH0，第 1 个采样的模拟量正端信号对应 IO ADC_CH3，以此类推。负端信号由 ADC_NCHN 指定，如果不需要差分输入，可以将负端指定为模拟地。

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

控制逻辑根据触发配置寄存器 ADC_TRIG 选择来自 MCPWM 或 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。

一段转换（一段内的所有通道采样转换完毕）完成，触发 ADC 转换完成中断。

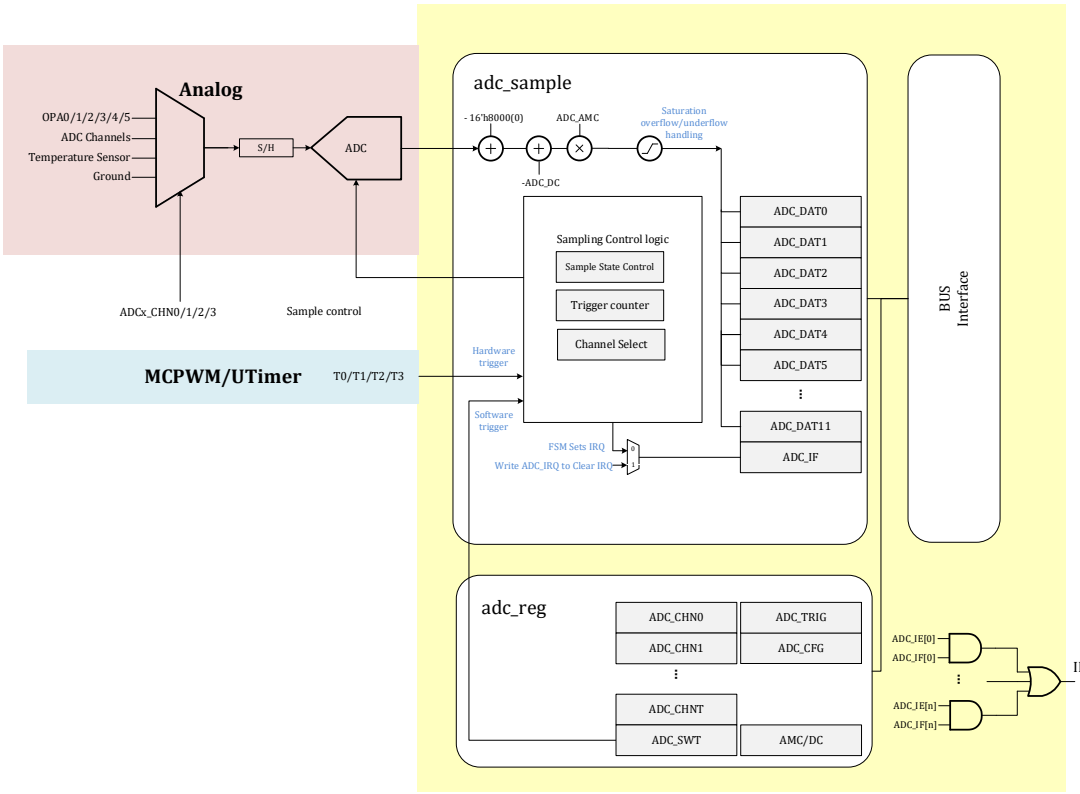


图 13-1 ADC 采集模块功能框图

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

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

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

13.1.2 ADC 触发方式

- 支持单段触发和两段触发
- 可以设置采样触发事件发生次数从而实现间隔触发
- 可以软件触发
- 每段采样触发完成可产生对应中断

- 触发指示信号可以通过 GPIO 送出用于调试

13.1.3 ADC 输出数制

ADC 输出数据为 14bit 补码，输入信号 0 对应 14’b00_0000_0000_0000，以 1 倍增益配置为例，输入信号-2.2V 对应 14’b10_0000_0000_0000，输入信号+2.2V 对应 14’b01_1111_1111_1111。ADC 转换后的 14bit 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 14’b10_0000_0000_1101 为例，如果配置为左对齐，右侧补 2 个 0，存入 ADCx_DAT 的值为 16’b1000_0000_0011_0100；如果配置为右对齐，左侧进行符号扩展，存入 ADCx_DAT 的值为 16’b1110_0000_0000_1101。推荐统一使用左对齐方式。

表 13-1 ADC 输出数字量数制转换

ADC 一倍增益输入/V	ADC 2/3 倍增益输入/V	转为有符号数后的数值
2.2	3.3	14’b01_1111_1111_1111
0	0	14’b00_0000_0000_0000
-2.2	-3.3	14’b10_0000_0000_0000

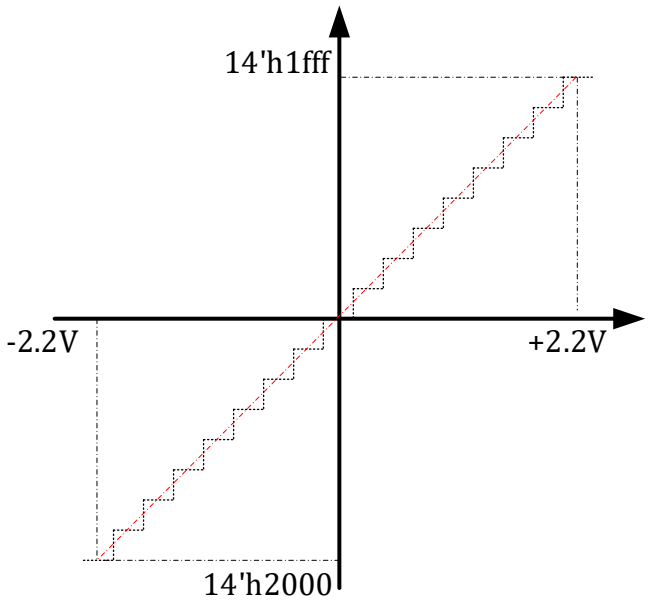


图 13-2 一倍增益设置下 ADC 模数转换数制量程

13.1.4 ADC 量程

ADC 有两种增益模式：高增益（1 倍）和低增益（2/3 倍），针对这两种增益，ADC 的量程也相应有所区别。1 倍增益模式下，对应最大±2.2V 的输入信号幅度，2/3 倍增益模式下，对应最大±3.3V 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~OPA5），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 2/3 倍增益。举例来说，

相线电流最大 100A（正弦波有效值），MOS 内阻（假设为 MOS 内阻采样）为 5mR，则运放的最大输入信号幅值为+/-707mV。此时应该选择运放的放大倍数为 4.5 倍，则放大后的信号约为+/-3.18V。

如果因为客观原因，运放的输出信号经放大后，最大信号仍然小于+/-2.2V，则应将 ADC 的增益配置为 1 倍。

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

13.1.5 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

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

最终硬件会将进行校正后的 D 存入相应的采样数据寄存器。ADC 接口硬件电路会根据每个通道的增益配置(ADC_GAIN0/1)来自动选择 $AMP_{correction}$ 与 DC_{offset}。

13.1.6 ADC 阈值监测(模拟看门狗)

每个 ADC 接口模块配备有 2 组阈值监测电路，同时进行上阈值和下阈值监测，ADC 的数据寄存器可以单独配置是否启用某组阈值监测，1 个数据寄存器可以同时启用多组阈值监测，但通常不这样配置。如果使能阈值监测，当 ADC 完成某次转换，且将经过校正的数据写入 ADCx_DAT 寄存器时，会进行阈值比较，如果写入的数据大于上阈值或小于下阈值则对应的阈值超限中断标志会置 1。

阈值为 14bit 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[15:2]与阈值进行比较，右对齐时，ADCx_DATx[13:0]与阈值进行比较。

13.1.7 过采样

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

$$ADCx\_DAT = \frac{1}{OVSR}\sum_{i=0}^{OVSR-1} ADCx\_DAT\_RAW_i$$

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

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

通过 ADCx_CFG.TROVS，可以配置一次触发即完成多次(ADCx_CFG.OVSR 次)采样并进行数据平均，

如图 13-3 所示，ADCx_CFG.TROVS=0，ADCx_CFG.OVSR=1 即过采样率为 2，触发后，ADC 对每个信号都采样两次平均后才将结果写入对应的 ADCx_DAT 寄存器；如果 ADCx_CFG.TROVS=1，则需要多次(ADCx_CFG.OVSR 次)触发才积累足够数据进行平均，如图 13-4 所示，ADCx_CFG.OVSR=1 即过采样率仍为 2，则每 2 次触发才累计 2 次数据进行平均并写入数据寄存器。当需要多次触发进行累加时，每次触发只能采样一个通道，即要求 ADCx_CHNT=1，如果采样多个不同信号，当通道切换时，会导致不同通道数据累加在一起而得到错误的转换后数据。

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

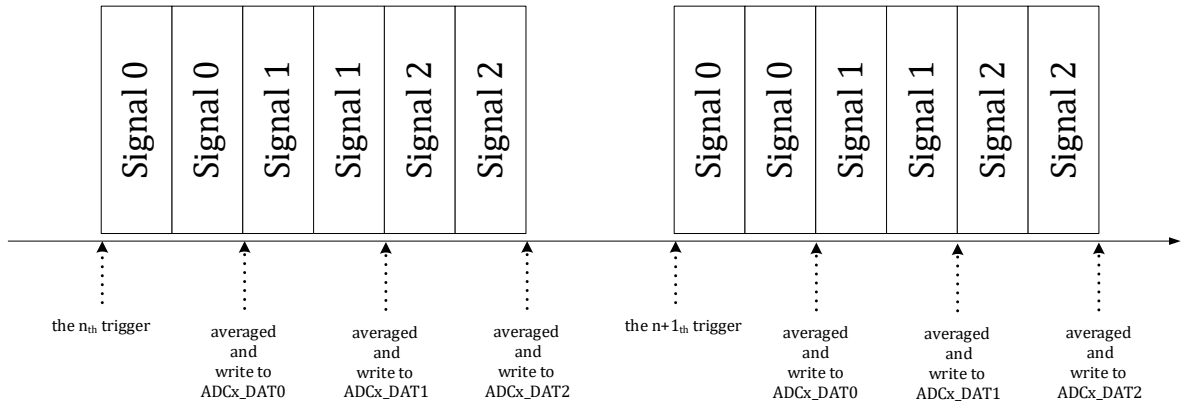


图 13-3 过采样转换时序示例 1

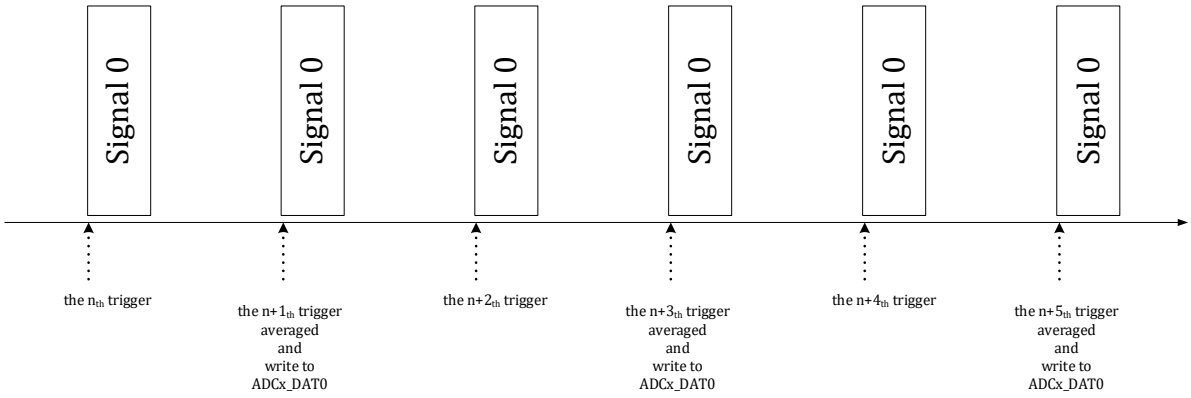


图 13-4 过采样转换时序示例 2

13.2 寄存器

13.2.1 地址分配

ADC0 在芯片中的基地址是 0x4001_7800；

ADC1 在芯片中的基地址是 0x4001_7C00；

ADC2 在芯片中的基地址是 0x4001_8000。

表 13-2 ADCx 寄存器列表

名称	偏移地址	说明
ADCx_DAT0	0x00	ADCx 第 0 次采样数据
ADCx_DAT1	0x04	ADCx 第 1 次采样数据
ADCx_DAT2	0x08	ADCx 第 2 次采样数据
ADCx_DAT3	0x0C	ADCx 第 3 次采样数据
ADCx_DAT4	0x10	ADCx 第 4 次采样数据
ADCx_DAT5	0x14	ADCx 第 5 次采样数据
ADCx_DAT6	0x18	ADCx 第 6 次采样数据
ADCx_DAT7	0x1C	ADCx 第 7 次采样数据
ADCx_DAT8	0x20	ADCx 第 8 次采样数据
ADCx_DAT9	0x24	ADCx 第 9 次采样数据
ADCx_DAT10	0x28	ADCx 第 10 次采样数据
ADCx_DAT11	0x2C	ADCx 第 11 次采样数据
ADCx_DAT12	0x30	ADCx 第 12 次采样数据
ADCx_DAT13	0x34	ADCx 第 13 次采样数据
ADCx_DAT14	0x38	ADCx 第 14 次采样数据
ADCx_DAT15	0x3C	ADCx 第 15 次采样数据
ADCx_PCHN0	0x40	ADCx 第 0~3 次采样正端输入信号选择
ADCx_PCHN1	0x44	ADCx 第 4~7 次采样正端输入信号选择
ADCx_PCHN2	0x48	ADCx 第 8~11 次采样正端输入信号选择
ADCx_PCHN3	0x4C	ADCx 第 12~15 次采样正端输入信号选择
ADCx_NCHN0	0x50	ADCx 第 0~3 次采样负端输入信号选择
ADCx_NCHN1	0x54	ADCx 第 4~7 次采样负端输入信号选择
ADCx_NCHN2	0x58	ADCx 第 8~11 次采样负端输入信号选择
ADCx_NCHN3	0x5C	ADCx 第 12~15 次采样负端输入信号选择
ADCx_CHNT	0x60	ADCx 各种触发模式下采样次数
ADCx_GAIN	0x70	ADCx 增益选择
ADCx_CFG	0x74	ADCx 模式配置
ADCx_TRIG	0x78	ADCx 采样触发配置
ADCx_SWT	0x7C	ADCx 软件触发
ADCx_DC0	0x80	ADCx 增益为 0 时 DC offset
ADCx_AMC0	0x84	ADCx 增益为 0 时增益校正系数
ADCx_DC1	0x88	ADCx 增益为 1 时 DC offset
ADCx_AMC1	0x8C	ADCx 增益为 1 时增益校正系数
ADCx_IE	0x90	ADCx 中断使能
ADCx_IF	0x94	ADCx 中断标志
ADCx_TH0	0x98	ADCx 阈值 0
ADCx_GEN0	0x9C	ADCx 阈值 0 监测使能
ADCx_TH1	0xA0	ADCx 阈值 1
ADCx_GEN1	0xA4	ADCx 阈值 1 监测使能

13.2.2 采样数据寄存器

13.2.2.1 ADCx_DAT0(x = 0,1,2)

地址分别为：0x4001_7800，0x4001_7C00，0x4001_8000

复位值：0x0

表 13-3 采样数据寄存器 ADCx_DAT0

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

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

13.2.2.2 ADCx_DAT1(x = 0,1,2)

地址分别为：0x4001_7804，0x4001_7C04，0x4001_8004

复位值：0x0

表 13-4 采样数据寄存器 ADCx_DAT1

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

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

13.2.2.3 ADCx_DAT2(x = 0,1,2)

地址分别为：0x4001_7808，0x4001_7C08，0x4001_8008

复位值：0x0

表 13-5 采样数据寄存器 ADCx_DAT2

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

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

13.2.2.4 ADCx_DAT3(x = 0,1,2)

地址分别为：0x4001_780C，0x4001_7C0C，0x4001_800C

复位值：0x0

表 13-6 采样数据寄存器 ADCx_DAT3

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

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

13.2.2.5 ADCx_DAT4(x = 0,1,2)

地址分别为：0x4001_7810，0x4001_7C10，0x4001_8010

复位值：0x0

表 13-7 采样数据寄存器 ADCx_DAT4

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

位置	位名称	说明

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

13.2.2.6 ADCx_DAT5(x = 0,1,2)

地址分别为：0x4001_7814，0x4001_7C14，0x4001_8014

复位值：0x0

表 13-8 采样数据寄存器 ADCx_DAT5

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

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

13.2.2.7 ADCx_DAT6(x = 0,1,2)

地址分别为：0x4001_7818，0x4001_7C18，0x4001_8018

复位值：0x0

表 13-9 采样数据寄存器 ADCx_DAT6

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

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

13.2.2.8 ADCx_DAT7(x = 0,1,2)

地址分别为：0x4001_781C，0x4001_7C1C，0x4001_801C

复位值：0x0

表 13-10 采样数据寄存器 ADCx_DAT7

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

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

13.2.2.9 ADCx_DAT8(x = 0,1,2)

地址分别为：0x4001_7820，0x4001_7C20，0x4001_8020

复位值：0x0

表 13-11 采样数据寄存器 ADCx_DAT8

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

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

13.2.2.10 ADCx_DAT9(x = 0,1,2)

地址分别为：0x4001_7824，0x4001_7C24，0x4001_8024

复位值：0x0

表 13-12 采样数据寄存器 ADCx_DAT9

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

位置	位名称	说明

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

13.2.2.11 ADCx_DAT10(x = 0,1,2)

地址分别为：0x4001_7828，0x4001_7C28，0x4001_8028

复位值：0x0

表 13-13 采样数据寄存器 ADCx_DAT10

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

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

13.2.2.12 ADCx_DAT11(x = 0,1,2)

地址分别为：0x4001_782C，0x4001_7C2C，0x4001_802C

复位值：0x0

表 13-14 采样数据寄存器 ADCx_DAT11

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

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

13.2.2.13 ADCx_DAT12(x = 0,1,2)

地址分别为：0x4001_7830，0x4001_7C30，0x4001_8030

复位值：0x0

表 13-15 采样数据寄存器 ADCx_DAT12

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

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

13.2.2.14 ADCx_DAT13(x = 0,1,2)

地址分别为：0x4001_7834，0x4001_7C34，0x4001_8034

复位值：0x0

表 13-16 采样数据寄存器 ADCx_DAT13

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

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

13.2.2.15 ADCx_DAT14(x = 0,1,2)

地址分别为：0x4001_7838，0x4001_7C38，0x4001_8038

复位值：0x0

表 13-17 采样数据寄存器 ADCx_DAT14

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

位置	位名称	说明
[31:16]		未使用
[15:0]	DAT14	ADCx 第 14 次采样数据

13.2.2.16 ADCx_DAT15(x = 0,1,2)

地址分别为：0x4001_783C，0x4001_7C3C，0x4001_803C

复位值：0x0

表 13-18 采样数据寄存器 ADCx_DAT15

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

位置	位名称	说明
[31:16]		未使用
[15:0]	DAT15	ADCx 第 15 次采样数据

13.2.3 信号来源寄存器

13.2.3.1 ADCx_PCHN0(x = 0,1,2)

地址分别为：0x4001_7840，0x4001_7C40，0x4001_8040

复位值：0x0

表 13-19 正端信号来源寄存器 ADCx_PCHN0

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

位置	位名称	说明
[31:16]		未使用
[15:12]	PDS3	ADCx 第 3 次采样正端输入信号选择
[11:8]	PDS2	ADCx 第 2 次采样正端输入信号选择
[7:4]	PDS1	ADCx 第 1 次采样正端输入信号选择
[3:0]	PDS0	ADCx 第 0 次采样正端输入信号选择

13.2.3.2 ADCx_PCHN1(x = 0,1,2)

地址分别为：0x4001_7844，0x4001_7C44，0x4001_8044

复位值：0x0

表 13-20 正端信号来源寄存器 ADCx_PCHN1

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

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

13.2.3.3 ADCx_PCHN2(x = 0,1,2)

地址分别为：0x4001_7848，0x4001_7C48，0x4001_8048

复位值：0x0

表 13-21 正端信号来源寄存器 ADCx_PCHN2

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

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

13.2.3.4 ADCx_PCHN3(x = 0,1,2)

地址分别为：0x4001_784C，0x4001_7C4C，0x4001_804C

复位值：0x0

表 13-22 正端信号来源寄存器 ADCx_PCHN3

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

PDS15	PDS14	PDS13	PDS12
RW	RW	RW	RW
0	0	0	0

位置	位名称	说明
[31:16]		未使用
[15:12]	PDS15	ADCx 第 15 次采样正端输入信号选择
[11:8]	PDS14	ADCx 第 14 次采样正端输入信号选择
[7:4]	PDS13	ADCx 第 13 次采样正端输入信号选择
[3:0]	PDS12	ADCx 第 12 次采样正端输入信号选择

13.2.3.5 ADCx_NCHN0(x = 0,1,2)

地址分别为：0x4001_7850，0x4001_7C50，0x4001_8050

复位值：0x0

表 13-23 负端信号来源寄存器 ADCx_NCHN0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDS3				NDS2				NDS1				NDS0			
RW				RW				RW				RW			
0				0				0				0			

位置	位名称	说明
[31:16]		未使用
[15:12]	NDS3	ADCx 第 3 次采样负端输入信号选择
[11:8]	NDS2	ADCx 第 2 次采样负端输入信号选择
[7:4]	NDS1	ADCx 第 1 次采样负端输入信号选择
[3:0]	NDS0	ADCx 第 0 次采样负端输入信号选择

13.2.3.6 ADCx_NCHN1(x = 0,1,2)

地址分别为：0x4001_7854，0x4001_7C54，0x4001_8054

复位值：0x0

表 13-24 负端信号来源寄存器 ADCx_NCHN1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDS7				NDS6				NDS5				NDS4			
RW				RW				RW				RW			
0				0				0				0			

位置	位名称	说明
[31:16]		未使用
[15:12]	NDS7	ADCx 第 7 次采样负端信号选择
[11:8]	NDS6	ADCx 第 6 次采样负端信号选择
[7:4]	NDS5	ADCx 第 5 次采样负端信号选择
[3:0]	NDS4	ADCx 第 4 次采样负端信号选择

13.2.3.7 ADCx_NCHN2(x = 0,1,2)

地址分别为：0x4001_7858，0x4001_7C58，0x4001_8058

复位值：0x0

表 13-25 负端信号来源寄存器 ADCx_NCHN2

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDS11				NDS10				NDS9				NDS8			
RW				RW				RW				RW			
0				0				0				0			

位置	位名称	说明
[31:16]		未使用
[15:12]	NDS11	ADCx 第 11 次采样负端信号选择
[11:8]	NDS10	ADCx 第 10 次采样负端信号选择
[7:4]	NDS9	ADCx 第 9 次采样负端信号选择
[3:0]	NDS8	ADCx 第 8 次采样负端信号选择

13.2.3.8 ADCx_NCHN3(x = 0,1,2)

地址分别为：0x4001_785C，0x4001_7C5C，0x4001_805C

复位值：0x0

表 13-26 负端信号来源寄存器 ADCx_NCHN3

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDS15				NDS14				NDS13				NDS12			
RW				RW				RW				RW			
0				0				0				0			

位置	位名称	说明
[31:16]		未使用
[15:12]	NDS15	ADCx 第 15 次采样负端信号选择

[11:8]	NDS14	ADCx 第 14 次采样负端信号选择
[7:4]	NDS13	ADCx 第 13 次采样负端信号选择
[3:0]	NDS12	ADCx 第 12 次采样负端信号选择

13.2.3.9 ADC 模拟信号选择

3 个 ADC 接口可以选择的被采集模拟信号不完全相同，具体如表 13-27 所示。

表 13-27 ADC 模拟信号来源分布

寄存器	不同配置值对应模拟信号
ADC0_PCHNx	0000: OPA0 输出 0001: OPA1 输出 0010: OPA2 输出 0011: OPA3 输出 0100: OPA4 输出 0101: OPA5 输出 0110: ADC0_CH6 0111: ADC0_CH7 1000: ADC0_CH8 1001: ADC0_CH9 1010: ADC0_CH10 1011: ADC0_CH11 1100: ADC0_CH12 1101: ADC0_CH13 1110: ADC0_CH14 1111:内部地
ADC0_NCHNx	0000: 内部地 0001: 内部地 0010: 内部地 0011: 内部地 0100: 禁止 0101: 禁止 0110: ADC0_CH6 0111: ADC0_CH7 1000: ADC0_CH8 1001: ADC0_CH9 1010: ADC0_CH10 1011: ADC0_CH11 1100: ADC0_CH12 1101: ADC0_CH13 1110: ADC0_CH14 1111: 连内部地

ADC1_PCHNx	0000: OPA0 输出 0001: OPA1 输出 0010: OPA2 输出 0011: OPA3 输出 0100: OPA4 输出 0101: OPA5 输出 0110: ADC1_CH6 0111: ADC1_CH7 1000: ADC1_CH8 1001: ADC1_CH9 1010: ADC1_CH10 1011: ADC1_CH11 1100: ADC1_CH12 1101: ADC1_CH13 1110: LDO12 1111: 内部地
ADC1_NCHNx	0000: 内部地 0001: 内部地 0010: 内部地 0011: 内部地 0100: 禁止 0101: 禁止 0110: ADC1_CH6 0111: ADC1_CH7 1000: ADC1_CH8 1001: ADC1_CH9 1010: ADC1_CH10 1011: ADC1_CH11 1100: ADC1_CH12 1101: ADC1_CH13 1110: 内部地 1111: 内部地

ADC2_PCHNx	0000: OPA0 输出 0001: OPA1 输出 0010: OPA2 输出 0011: OPA3 输出 0100: ADC2_CH4 0101: ADC2_CH5 0110: ADC2_CH6 0111: ADC2_CH7 1000: ADC2_CH8 1001: ADC2_CH9 1010: ADC2_CH10 1011: ADC2_CH11 1100: ADC2_CH12 1101: ADC2_CH13 1110: 温度传感器 1111: 内部地
ADC2_NCHNx	0000: 内部地 0001: 内部地 0010: 内部地 0011: 内部地 0100: ADC2_CH4 0101: ADC2_CH5 0110: ADC2_CH6 0111: ADC2_CH7 1000: ADC2_CH8 1001: ADC2_CH9 1010: ADC2_CH10 1011: ADC2_CH11 1100: ADC2_CH12 1101: ADC2_CH13 1110: 内部地 1111: 内部地

当 ADCx_PCHN 选择 OPA 输出时，ADCx_NCHN 固定选择 OPA 输出的负端，设置为其他值无效。即 ADC 采样的是 OPA 输出的差分信号。如果需要采集运放的共模电平 Vcm，建议在 OPA 输入短接时通过 OPAx_OUT 将运放放大后的信号输出到 P3.11 或 P4.6，然后再通过 ADC2_CH7 或 ADC1_CH8 进行采样。

13.2.4 采样通道数寄存器

13.2.4.1 ADCx_CHNT(x = 0,1,2)

地址分别为：0x4001_7860，0x4001_7C60，0x4001_8060

复位值：0x0

表 13-28 采样通道数寄存器 ADCx_CHNT

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

位置	位名称	说明
[31:8]		未使用
[7:4]	S2	第二段采样通道数
[3:0]	S1	第一段采样通道数

注意：采样次数不允许配置为 0。1 表示 1 个通道，2 表示 2 个通道，..... ，15 表示 15 个通道。如果使用两段，则两段采样通道数之和不应超过 16。

13.2.5 配置寄存器

13.2.5.1 ADCx_GAIN(x = 0,1,2)

地址分别为：0x4001_7870，0x4001_7C70，0x4001_8070

复位值：0x0

表 13-29 增益选择寄存器 ADCx_GAIN

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G15	G14	G13	G12	G11	G10	G9	G8	G7	G6	G5	G4	G3	G2	G1	G0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:12]		未使用
[15]	G15	DAT15 增益选择
[14]	G14	DAT14 增益选择
[13]	G13	DAT13 增益选择
[12]	G12	DAT12 增益选择
[11]	G11	DAT11 增益选择
[10]	G10	DAT10 增益选择
[9]	G9	DAT9 增益选择
[8]	G8	DAT8 增益选择
[7]	G7	DAT7 增益选择
[6]	G6	DAT6 增益选择

[5]	G5	DAT5 增益选择
[4]	G4	DAT4 增益选择
[3]	G3	DAT3 增益选择
[2]	G2	DAT2 增益选择
[1]	G1	DAT1 增益选择
[0]	G0	DAT0 增益选择

1：1 倍增益模式，对应最大±2.2V 的输入信号幅度；

0：2/3 倍增益模式，对应最大±3.3V 的输入信号幅度；

需要保持 SYS_AFE_REG2.REF2VDD=0

13.2.5.2 ADCx_CFG(x = 0,1,2)

地址分别为：0x4001_7874，0x4001_7C74，0x4001_8074

复位值：0x0

表 13-30 模式配置寄存器 ADCx_CFG

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

位置	位名称	说明
[31:13]		未使用
[12]	NSMP	0:单段采样，1:两段采样
[11]	FSM_RS	状态机复位控制信号。软件写入 1 产生复位，将 ADC 内部状态机回到初始状态，完成后自动回到 0。该复位控制，不影响 ADC 其它寄存器的配置值。 读取该位返回 ADC 当前状态，1 表示 ADC 目前正在采样转换工作中，0 表示空闲。
[10]	DATA_ALIGN	ADC_DAT 对齐方式 0：左对齐，右端补 2'h0， 1：右对齐，左端补 2bit 符号位
[9]		未使用
[8]	CSMP	连续采样模式 0：不开启 1：开启连续采样模式，触发到来后，ADC 即开始采样转换，完成所有信号的转换后，无需等待触发立即从第 0 个信号开始新一轮的采样转换

[7:4]	TCNT	触发一次采样所需的事件数。 0：表示需要发生 1 次事件才能触发一次采样 1：表示需要发生 2 次事件才能触发一次采样 …… 15：表示需要发生 16 次事件才能触发一次采样
[3]	TROVS	过采样触发模式 0：一次触发即连续过采样多次至 OVSR，并将数据平均后存入，可以配置采样多个通道，每个通道都采样 OVSR 次后才进行下一个通道的采样。 1：一次触发只进行一次采样转换，需要 OVSR 次触发才完成足够数据的采集并进行平均，而后存入数据寄存器。受到累加器个数限制，在此种配置下，ADCx_CHNT 只能配置采样一个通道。
[2:0]	OVSR	过采样率 0：1，默认 1 次采样即存入数据 1：2 次采后存入数据 2：4 次采样后存入数据 3：8 次采样后存入数据 4：16 次采样后存入数据 5：32 次采样后存入数据 6：64 次采样后存入数据 7：128 次采样后存入数据

13.2.5.3 ADCx_TRIG(x = 0,1,2)

地址分别为：0x4001_7878，0x4001_7C78，0x4001_8078

复位值：0x0

表 13-31 采样触发配置寄存器 ADCx_TRIG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
						UTIMER4_CMP1_EN	UTIMER4_CMP0_EN	UTIMER3_CMP1_EN	UTIMER3_CMP0_EN	UTIMER2_CMP1_EN	UTIMER2_CMP0_EN	UTIMER1_CMP1_EN	UTIMER1_CMP0_EN	UTIMER0_CMP1_EN	UTIMER0_CMP0_EN
						RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
						0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								MCPWM1_T3_EN	MCPWM1_T2_EN	MCPWM1_T1_EN	MCPWM1_T0_EN	MCPWM0_T3_EN	MCPWM0_T2_EN	MCPWM0_T1_EN	MCPWM0_T0_EN
								RW	RW	RW	RW	RW	RW	RW	RW

	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:26]		未使用
[25]	UTIMER4_CMP1_EN	UTIMER4 比较事件 1 触发 ADC 采样使能，高有效
[24]	UTIMER4_CMP0_EN	UTIMER4 比较事件 0 触发 ADC 采样使能，高有效
[23]	UTIMER3_CMP1_EN	UTIMER3 比较事件 1 触发 ADC 采样使能，高有效
[22]	UTIMER3_CMP0_EN	UTIMER3 比较事件 0 触发 ADC 采样使能，高有效
[21]	UTIMER2_CMP1_EN	UTIMER2 比较事件 1 触发 ADC 采样使能，高有效
[20]	UTIMER2_CMP0_EN	UTIMER2 比较事件 0 触发 ADC 采样使能，高有效
[19]	UTIMER1_CMP1_EN	UTIMER1 比较事件 1 触发 ADC 采样使能，高有效
[18]	UTIMER1_CMP0_EN	UTIMER1 比较事件 0 触发 ADC 采样使能，高有效
[17]	UTIMER0_CMP1_EN	UTIMER0 比较事件 1 触发 ADC 采样使能，高有效
[16]	UTIMER0_CMP0_EN	UTIMER0 比较事件 0 触发 ADC 采样使能，高有效
[15:8]		未使用
[7]	MCPWM1_T3_EN	MCPWM1 T3 事件触发 ADC 采样使能，高有效
[6]	MCPWM1_T2_EN	MCPWM1 T2 事件触发 ADC 采样使能，高有效
[5]	MCPWM1_T1_EN	MCPWM1 T1 事件触发 ADC 采样使能，高有效
[4]	MCPWM1_T0_EN	MCPWM1 T0 事件触发 ADC 采样使能，高有效
[3]	MCPWM0_T3_EN	MCPWM0 T3 事件触发 ADC 采样使能，高有效
[2]	MCPWM0_T2_EN	MCPWM0 T2 事件触发 ADC 采样使能，高有效
[1]	MCPWM0_T1_EN	MCPWM0 T1 事件触发 ADC 采样使能，高有效
[0]	MCPWM0_T0_EN	MCPWM0 T0 事件触发 ADC 采样使能，高有效

ADC 的触发来源为 MCPWM 或 UTimer，通常不会同时使用 MCPWM 和 UTimer 进行触发，只在二者之中选一。

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

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

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

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

13.2.6 软件触发寄存器

13.2.6.1 ADCx_SWT(x = 0,1,2)

地址分别为：0x4001_787C，0x4001_7C7C，0x4001_807C

复位值：0x0

表 13-32 软件触发寄存器 ADCx_SWT

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

位置	位名称	说明
[31:16]		未使用
[15:0]	SWT	写入数据为 0x5AA5 时，产生一次软件触发

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，下一次软件触发则再次写入 0x5AA5。

13.2.7 校正寄存器

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

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

考虑到信号误差，去除 DC offset 的信号可能会发生溢出，对于溢出的数据会做饱和处理，以使信号范围在-1.2V-+1.2V 或-2.2V-+2.2V 之间。

ADC 有两种增益模式：高增益（1 倍）和低增益（2/3 倍），针对这两种增益，ADC 的量程也相应有所区别。1 倍增益模式下，对应最大±2.2V 的输入信号幅度，2/3 倍增益模式下，对应最大±3.3V 的输入信号幅度。两种量程使用不同的 DC offset 和增益校正系数。因此存在两组校正寄存器，分别为 ADCx_DC0，ADCx_AMC0 和 ADCx_DC1，ADCx_AMC1。芯片出厂时已经过工厂标定，标定数据存放在 Flash info 中，芯片上电自动完成校准参数的加载。ADC 模块在初始化的时候，需要根据数据左右对齐模式配置 DC offset, 可以参看芯片供应商提供的库函数。使用 SYS_SFT_RST 软复位 ADC 模块后，ADC 内部的寄存器会被复位。 需要重新使用 ADC 初始化函数从 NVR 中读取 DC/AMC 等校准参数。

13.2.7.1 ADCx_DC0(x = 0,1,2)

地址分别为：0x4001_7880，0x4001_7C80，0x4001_8080

复位值：0x0

表 13-33 0 档增益直流偏置寄存器 ADCx_DC0

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

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

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

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

此处存入的 ADC_DC 值应该对应的是右对齐的 offset 数值,当 ADCx_CFG 寄存器配置为左对齐时，硬件会根据对齐的设置，自动调整 ADCx_DC 参与 ADC 校正。具体来说，右对齐时，ADCx_DC[15:0]直接参与校正运算，左对齐时，ADCx_DC[15:0]会先左移 2 位然后参与 ADC 校正运算。

13.2.7.2 ADCx_AMC0(x = 0,1,2)

地址分别为：0x4001_7884，0x4001_7C84，0x4001_8084

复位值：0x200

表 13-34 0 档增益增益校正寄存器 ADCx_AMC0

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

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

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

13.2.7.3 ADCx_DC1(x = 0,1,2)

地址分别为：0x4001_7888，0x4001_7C88，0x4001_8088

复位值：0x0

表 13-35 1 档增益直流偏置寄存器 ADCx_DC1

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

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

13.2.7.4 ADCx_AMC1(x = 0,1,2)

地址分别为：0x4001_788C，0x4001_7C8C，0x4001_808C

复位值：0x200

表 13-36 1 档增益增益校正寄存器 ADCx_AMC1

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

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

13.2.8 中断寄存器

13.2.8.1 ADCx_IE(x = 0,1,2)

地址分别为：0x4001_7890，0x4001_7C90，0x4001_8090

复位值：0x0

表 13-37 中断使能寄存器 ADCx_IE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	HERR_RE	SERR_RE		AWD1_RE	AWD0_RE	SF2_RE	SF1_RE		HERR_IE	SERR_IE		AWD1_IE	AWD0_IE	SF2_IE	SF1_IE
	RW	RW		RW	RW	RW	RW		RW	RW		RW	RW	RW	RW
	0	0		0	0	0	0		0	0		0	0	0	0

位置	位名称	说明
[31:15]		未使用
[14]	HERR_RE	硬件触发发生在非空闲状态 DMA 请求使能
[13]	SERR_RE	软件触发发生在非空闲状态 DMA 请求使能
[12]		未使用
[11]	AWD1_RE	阈值监测 1 超限 DMA 请求使能
[10]	AWD0_RE	阈值监测 0 超限 DMA 请求使能
[9]	SF2_RE	第二段采样完成 DMA 请求使能
[8]	SF1_RE	第一段采样完成 DMA 请求使能
[7]		未使用
[6]	HERR_IE	硬件触发发生在非空闲状态中断使能
[5]	SERR_IE	软件触发发生在非空闲状态中断使能
[4]		未使用
[3]	AWD1_IE	阈值监测 1 超限中断使能
[2]	AWD0_IE	阈值监测 0 超限中断使能
[1]	SF2_IE	第二段采样完成中断使能
[0]	SF1_IE	第一段采样完成中断使能

13.2.8.2 ADCx_IF(x = 0,1,2)

地址分别为：0x4001_7894，0x4001_7C94，0x4001_8094

复位值：0x0

表 13-38 中断标志寄存器 ADCx_IF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									HERR_IF	SERR_IF		AWD1_IF	AWD0_IF	SF2_IF	SF1_IF
									RW1C	RW1C		RW1C	RW1C	RW1C	RW1C

	0	0		0	0	0	0

位置	位名称	说明
[31:7]		未使用
[6]	HERR_IF	硬件触发发生在非空闲状态中断标志
[5]	SERR_IF	软件触发发生在非空闲状态中断标志
[4]		未使用
[3]	AWD1_IF	阈值监测 1 超限中断标志
[2]	AWD0_IF	阈值监测 0 超限中断标志
[1]	SF2_IF	第二段采样完成中断标志
[0]	SF1_IF	第一段采样完成中断标志

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

13.2.9 模拟看门狗

13.2.9.1 ADCx_TH0 (x = 0,1,2)

地址分别为：0x4001_7898，0x4001_7C98，0x4001_8098

复位值：0x07FF_0000

表 13-39 阈值寄存器 0 ADCx_ TH0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		HTH													
		RW													
		0x1FFF													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		LTH													
		RW													
		0x2000													

位置	位名称	说明
[31:30]		未使用
[29:16]	HTH	ADC 模拟看门狗 0 上阈值
[15:14]		未使用
[13:0]	LTH	ADC 模拟看门狗 0 下阈值

13.2.9.2 ADCx_GEN0 (x = 0,1,2)

地址分别为：0x4001_789C，0x4001_7C9C，0x4001_809C

复位值：0x0

表 13-40 监测使能寄存器 0 ADCx_ GEN0

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

位置	位名称	说明
[31:16]		未使用
[15:0]	GEN	ADC 模拟看门狗 0 对应使能位 BIT0: DAT0 看门狗监测使能 BIT1: DAT1 看门狗监测使能 …… BIT15：DAT15 看门狗监测使能

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

13.2.9.3 ADCx_TH1(x = 0,1,2)

地址分别为：0x4001_78A0，0x4001_7CA0，0x4001_80A0

复位值：0x07FF_0000

表 13-41 阈值寄存器 1 ADCx_ TH1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		HTH													
		RW													
		0x1FFF													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		LTH													
		RW													
		0x2000													

位置	位名称	说明
[31:30]		未使用
[29:16]	HTH	ADC 模拟看门狗 1 上阈值
[15:14]		未使用
[13:0]	LTH	ADC 模拟看门狗 1 下阈值

13.2.9.4 ADCx_GEN1 (x = 0,1,2)

地址分别为：0x4001_78A4，0x4001_7CA4，0x4001_80A4

复位值：0x0

表 13-42 监测使能寄存器 0 ADCx_ GEN1

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

位置	位名称	说明
[31:16]		未使用
[15:0]	GEN	ADC 模拟看门狗 1 对应使能位 BIT0: DAT0 看门狗监测使能 BIT1: DAT1 看门狗监测使能 …… BIT15：DAT15 看门狗监测使能

13.3 应用指南

13.3.1 ADC 采样触发模式

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

第一次触发

来自 MCPWM/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 ADCx_SWT 写入命令字的方式 16'h5AA5 软件触发 ADC 采样。

第一段采样

判断是否为一段采样。

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

否: 采样次数达到预设值 ADCx_CHNT[3:0], ADC 进入空闲状态 1（两段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]，结束本次采样，回到空闲状态 0。

各种硬件触发模式的触发条件汇总如表 13-43 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 13-43 ADC 采样触发模式

<table>
<tr><th></th><th>单段触发</th><th>两段触发</th></tr>
<tr><td rowspan="7">TADC 触发</td><td>None(TADC 触发使能未打开)</td><td rowspan="7">第一段 TADC[0]
第二段 TADC[1]</td></tr>
<tr><td>C 次 TADC[0]</td></tr>
<tr><td>C 次 TADC[1]</td></tr>
<tr><td>C 次 TADC[2]</td></tr>
<tr><td>C 次 TADC[3]</td></tr>
<tr><td>C 次
TADC[0]/TADC[1]/
TADC[2]/TADC[3]</td></tr>
<tr><td>软件触发</td><td>向 ADC_SWT 写入 16‘h5aa5</td><td>第一段向 ADC_SWT 写入 16‘h5aa5
第二段向 ADC_SWT 写入 16‘h5aa5</td></tr>
</table>

其中，当配置了多次触发事件触发 ADC 开始采样，C 次=ADCx_CFG.TCNT。

13.3.1.1 单段触发模式

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

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

触发事件可以是来自外部的 MCPWM/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_PCHN0/1/2/3 和 ADC_NCHN0/1/2/3 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 ADCx_CFG.TCNT=4，ADCx_TRIG.MCPWM0_T2_EN =1，即 MCPWM0 的 TADC[2]发生 4 次才进行触发，状态转移如图 13-5 所示。

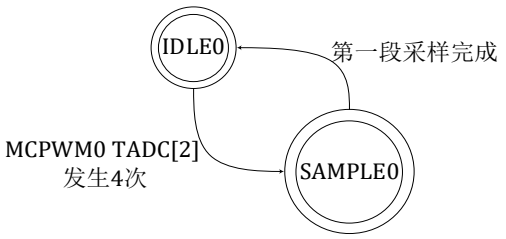


图 13-5 ADC 单段采样状态转移图

13.3.1.2 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 13-6 所示。

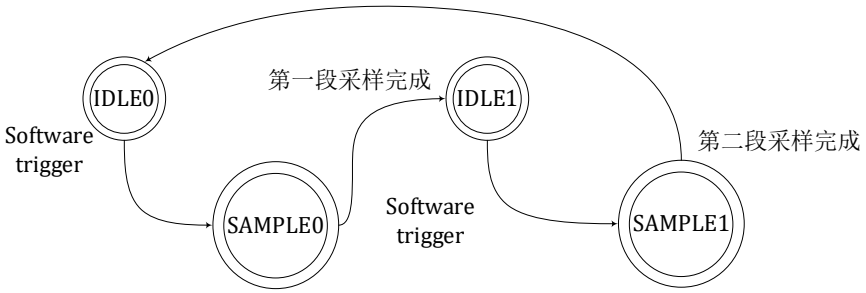


图 13-7 ADC 两段采样状态转移图

13.3.2 中断

13.3.2.1 单段触发采样完成中断

采样完成产生一个中断。

13.3.2.2 两段触发采样完成中断

第一段采样完成产生一个中断，第二段采样完成产生一个中断。

13.3.3 配置修改

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

示例程序如下

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

13.3.4 选择对应的模拟通道

ADC 采样信号来源可以参考表 13-27，也可以参考请查阅 DATASHEET 中表 2.2 引脚功能选择。关闭对应 IO 的 IE 和 OE，即可使用其模拟功能。

14 UTimer 通用定时器

14.1 概述

14.1.1 功能框图

如图 14-1 所示，通用定时器 UTIMER 主要包括 5 个独立的 Timer，以及 4 个正交编码器模块。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。

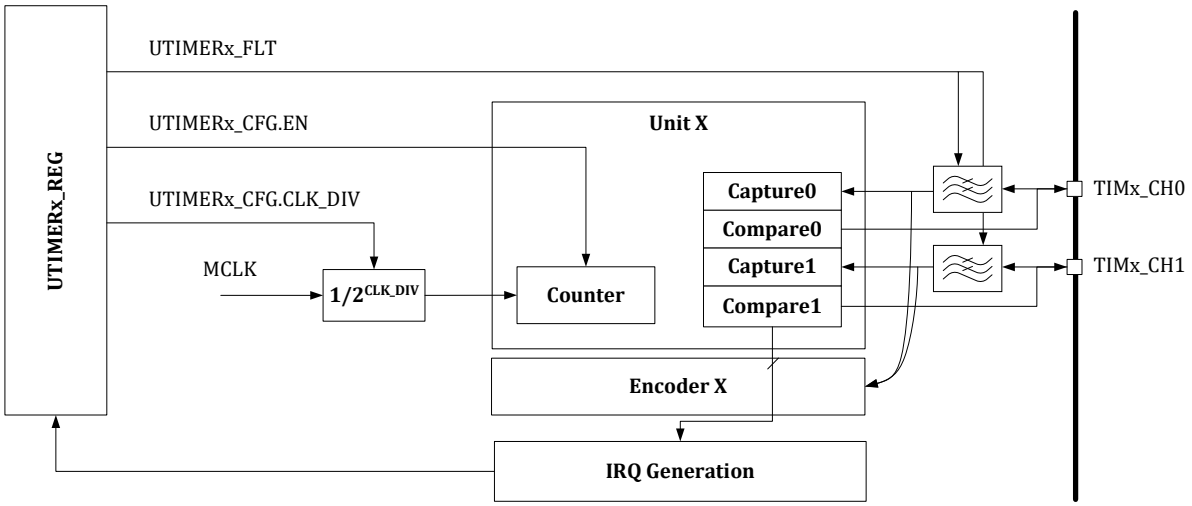


图 14-1 UTimer 模块顶层功能框图

14.1.1.1 总线接口模块

总线接口模块包括:

将来自 AHB 总线的访问信号翻译为寄存器读写信号，控制寄存器模块的时钟，并对寄存器模块发起读写。

CG 时钟门控模块，在 AHB 总线无访问时，将寄存器模块时钟关闭以降低功耗。

14.1.1.2 寄存器模块

UITMERx_REG 模块

实现对各个子模块控制寄存器的读写。

实现对各个子模块状态、结果寄存器的访问。

实现对各个子模块中断信号的处理和中断产生。

14.1.1.3 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。

14.1.1.4 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，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。

14.1.1.5 编码器模块

编码器模块用于对芯片外部送入的编码器编码信号进行计数。时钟分频模块

时钟分频模块用于产生时钟分频的各种信号。

定时器模块中一共包括 5 个独立工作的通用定时器，Timer0/Unit0、Timer1/Unit1、Timer4/Unit4 为 16bit 位宽，Timer2/Unit2、Timer3/Unit3 为 32bit 位宽。每个定时器包含两个通道。

定时器模块中集成了 4 个编码器模块。其中编码器 0/1/2/3 的输入分别来自 Timer0/1/2/3 的通道 0/1。使用编码器不影响 Timer 功能。

14.1.2 功能特点

定时器模块有以下特点:

- ✧ 独立工作计数器，可工作在不同频率下
- ✧ Timer0、Timer1 和 Timer4 为 16bit 通用定时器
- ✧ Timer2 和 Timer3 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）
- ✧ 对每个输入信号进行滤波
- ✧ Timer 事件可作为 DMA 传输触发

14.2 实现说明

14.2.1 时钟分频

为了实现各个 timer 独立分频，且可以方便对中断/计数值进行写操作，采取了各个 timer 均工作在系统主频，但使用分频计数器来降低计数器计数频率。

14.2.2 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。

14.2.3 滤波

定时器模块共有 10 个/5 对通道输入，定时器可以对每个输入进行不同程度的滤波。

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

输入信号滤波使用 Timer 计数时钟，即根据 UTIMER_UNTx_CFG.CLK_DIV 对系统高速时钟分频后的时钟。

如图 14-2 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。

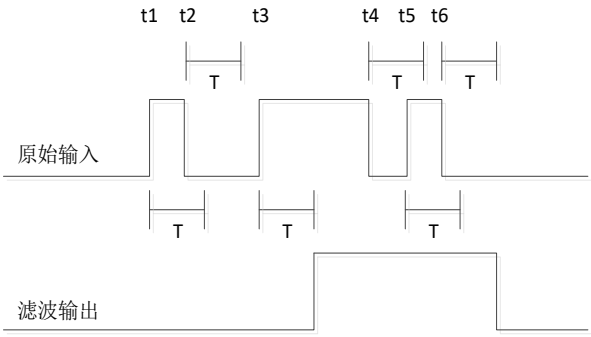


图 14-2 UTimer 滤波示意图

14.2.4 模式

14.2.4.1 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)。

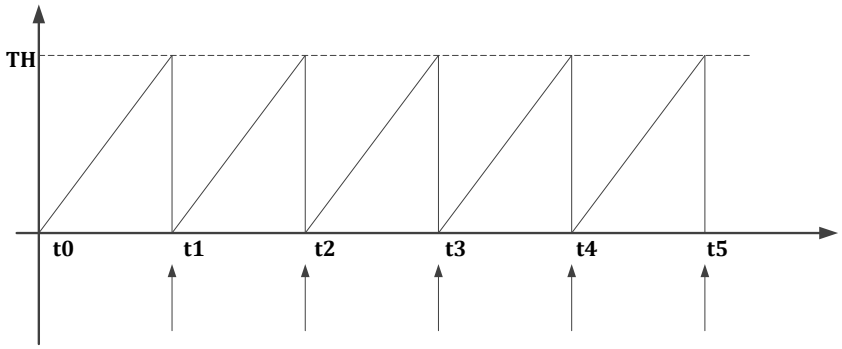


图 14-3 UTimer 通用计数器

14.2.4.2 比较模式

比较模式下，计数器计数到 UTIMERx_CMP0/ UTIMERx_CMP1 时，产生比较中断。比较模式可

以产生一个比较脉冲。

对 Timer 的通道 0 来说，在计数器回零时，输出一个电平（极性可配置）到 TIMERx_CH0，当计数到 UTIMERx_CMP0 即比较事件发生时，电平翻转，输出另一个电平到 IO 口 TIMERx_CH0。

对 Timer 的通道 1 来说，在计数器回零时，输出一个电平（极性可配置）到 TIMERx_CH1，当计数到 UTIMERx_CMP1 即比较事件发生时，电平翻转，输出另一个电平到 IO 口 TIMERx_CH1。

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。

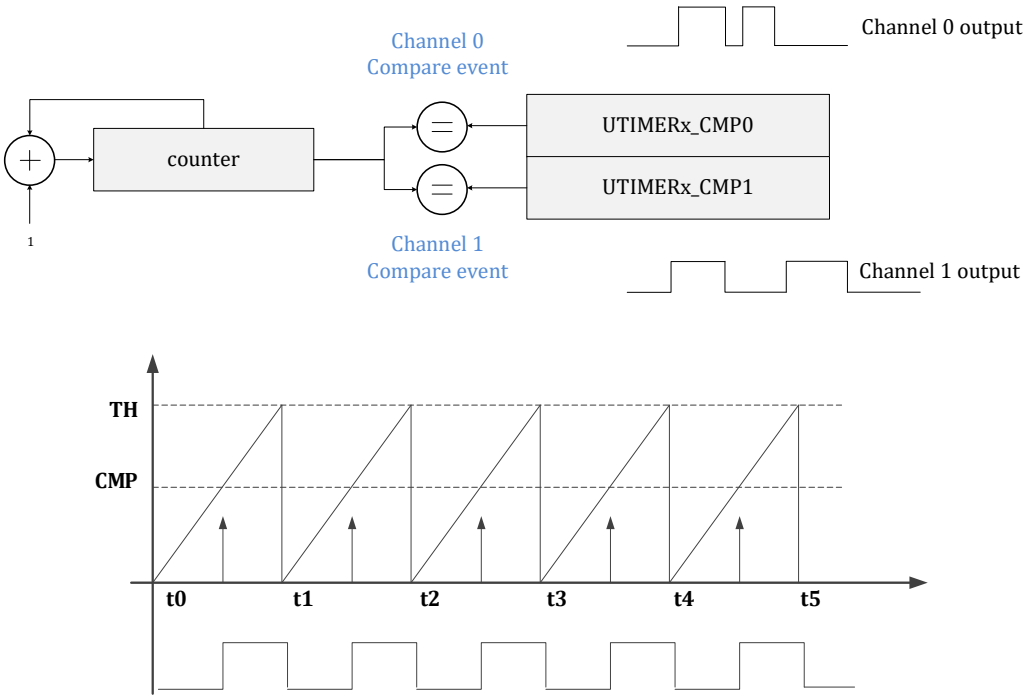


图 14-4 UTimer 比较模式

14.2.4.3 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 UTIMERx_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。

对 Timer 的通道 0 来说，可以配置其检测上升沿、下降沿、或者双沿，当通道 0 输入信号发生特定翻转时，Timer 将此时的计数器的数值写入 UTIMERx_CMP0。

对 Timer 的通道 1 来说，可以配置其检测上升沿、下降沿、或者双沿，当通道 1 输入信号发生特定翻转时，Timer 将此时的计数器的数值写入 UTIMERx_CMP1。

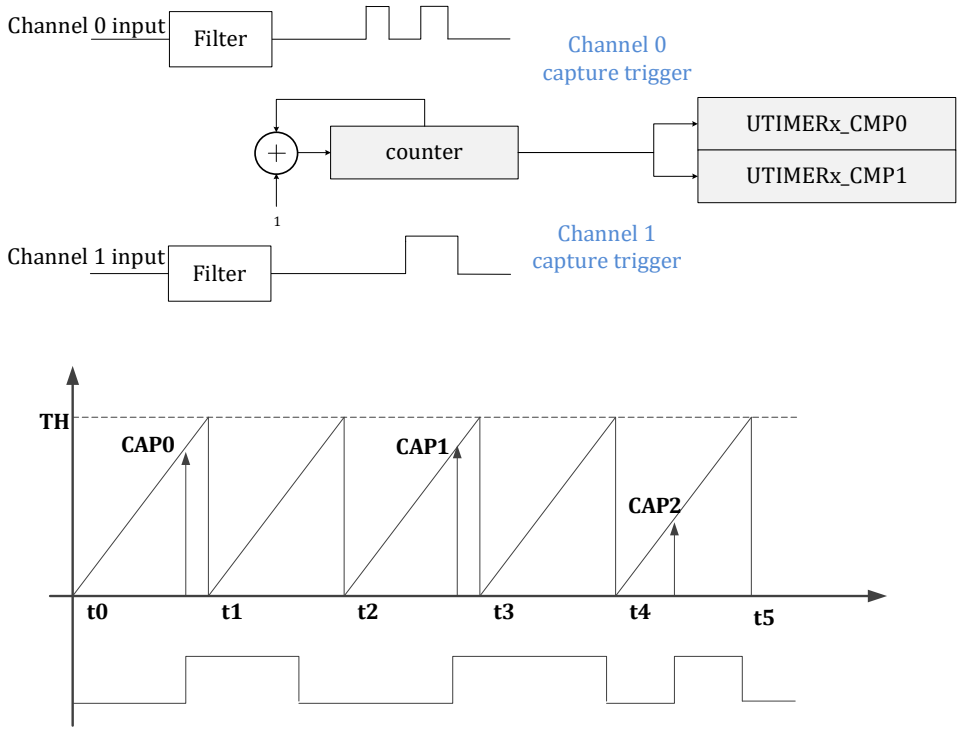


图 14-5 UTimer 捕获模式

如图 14-5 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 UTIMERx_CMP 寄存器中。

14.2.5 ADC 触发

Timer 的比较事件可作为 ADC 采样触发事件，可参考 13.2.5.3 ADCx_TRIG(x = 0,1,2)。

14.2.6 编码器

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

其中 QEP0 的两个输入信号 T1/T2 分别来自 Timer0 Channel0/1 对应的 GPIO 输入，并经过 Timer 内部的滤波；QEP1 的 T1/T2 信号分别来自 Timer1 Channel0/1 对应的 GPIO 输入，并经过 Timer 内部的滤波。QEP2/3，依次类推。开启编码器功能时并不影响 Timer 功能的正常使用。

编码器的输入 T1/T2/Z 信号均受对应的 Timer 滤波系数控制，举例来说，QEP0 的 Z 信号受到 UTIMER0_FLT 滤波时间控制。

使用编码器，需要开启对应的 SYS_CLK_FEN 中对应的 Timer 时钟使能。如使用 QEP1 需要开启 Time1 外设时钟使能。

14.2.6.1 正交编码信号

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

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

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

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

逻辑关系如下公式所示：

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

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

表 14-1 编码器正交编码工作模式

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

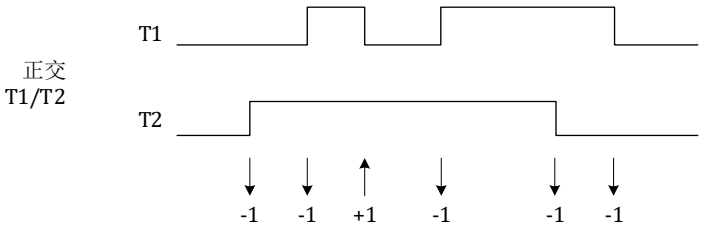


图 14-6 编码器只在 T1 时刻计数的正交编码信号计数情况

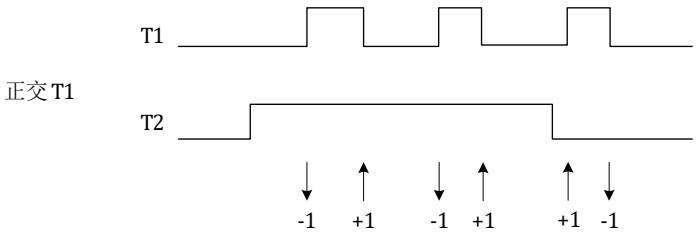


图 14-7 编码器在 T1 或 T2 时刻计数的正交编码信号计数情况

14.2.6.2 符号加脉冲信号

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

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

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

表 14-2 编码器符号加脉冲工作模式

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

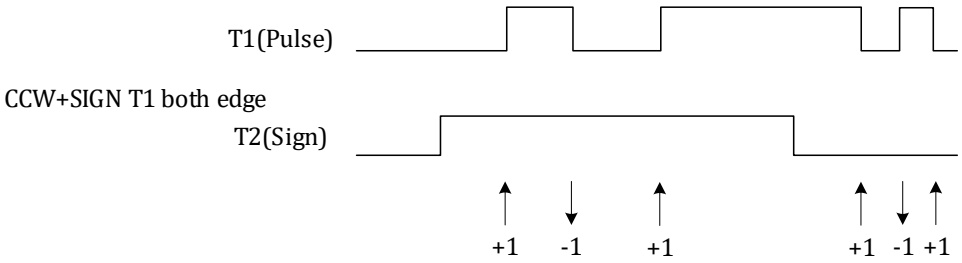


图 14-8 编码器在 T1 上升下降沿都计数的符号加脉冲信号计数情况

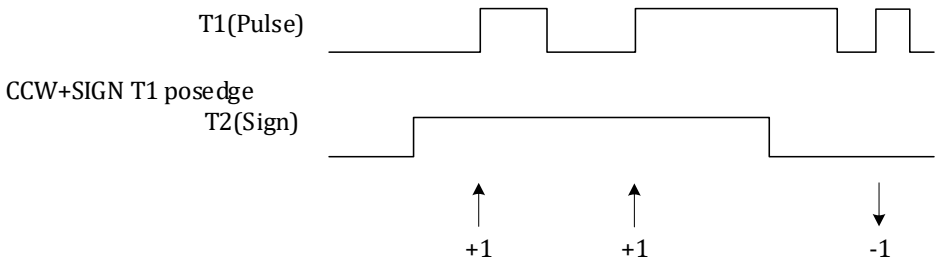


图 14-9 编码器在仅 T1 上升沿计数的符号加脉冲信号计数情况

14.2.6.3 CCW/CW 双脉冲信号

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

Counter Up = 1 @ (T1 triggering edges)

Counter Down = 1 @ (T2 triggering edges)

表 14-3 编码器 CCW/CW 双脉冲工作模式

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

T1/T2 上升下降沿	递增	递增	递减	递减

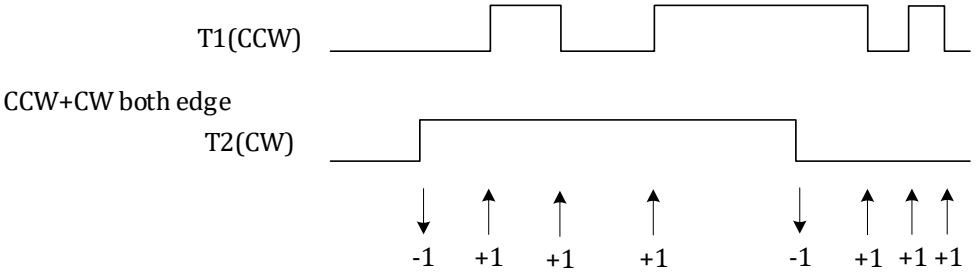


图 14-10 编码器仅在 T1/T2 上升沿计数的 CCW/CW 双脉冲信号计数情况

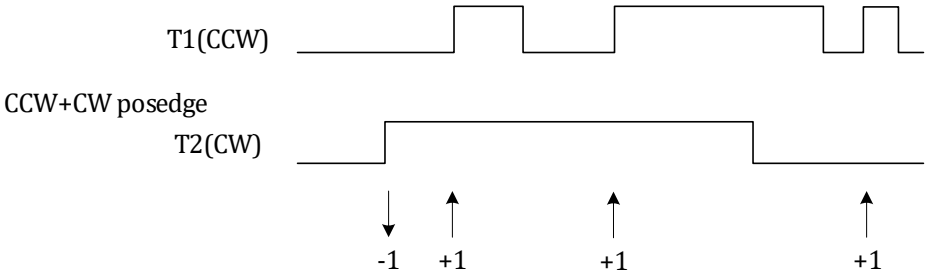


图 14-11 编码器在 T1/T2 上升下降沿计数的 CCW/CW 双脉冲信号计数情况

14.3 寄存器

14.3.1 地址分配

Timer0 在芯片中的基地址是 0x4001_4000

表 14-4 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_4100

表 14-5 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

Timer2 在芯片中的基地址是 0x4001_4200

表 14-6 Timer2 寄存器地址分配

UTIMER2_CFG	0x00	Timer2 配置寄存器
UTIMER2_TH	0x04	Timer2 计数门限寄存器
UTIMER2_CNT	0x08	Timer2 计数值寄存器
UTIMER2_CMP0	0x0C	Timer2 比较/捕获寄存器 0
UTIMER2_CMP1	0x10	Timer2 比较/捕获寄存器 1
UTIMER2_EVT	0x14	Timer2 外部事件选择寄存器
UTIMER2_FLT	0x18	Timer2 滤波控制寄存器
UTIMER2_IE	0x1C	Timer2 中断使能寄存器
UTIMER2_IF	0x20	Timer2 中断标志寄存器

Timer3 在芯片中的基地址是 0x4001_4300

表 14-7 Timer3 寄存器地址分配

UTIMER3_CFG	0x00	Timer3 配置寄存器
UTIMER3_TH	0x04	Timer3 计数门限寄存器
UTIMER3_CNT	0x08	Timer3 计数值寄存器
UTIMER3_CMP0	0x0C	Timer3 比较/捕获寄存器 0
UTIMER3_CMP1	0x10	Timer3 比较/捕获寄存器 1
UTIMER3_EVT	0x14	Timer3 外部事件选择寄存器
UTIMER3_FLT	0x18	Timer3 滤波控制寄存器
UTIMER3_IE	0x1C	Timer3 中断使能寄存器
UTIMER3_IF	0x20	Timer3 中断标志寄存器

Timer4 在芯片中的基地址是 0x4001_4400

表 14-8 Timer4 寄存器地址分配

UTIMER4_CFG	0x00	Timer4 配置寄存器
UTIMER4_TH	0x04	Timer4 计数门限寄存器
UTIMER4_CNT	0x08	Timer4 计数值寄存器
UTIMER4_CMP0	0x0C	Timer4 比较/捕获寄存器 0
UTIMER4_CMP1	0x10	Timer4 比较/捕获寄存器 1
UTIMER4_EVT	0x14	Timer4 外部事件选择寄存器
UTIMER4_FLT	0x18	Timer4 滤波控制寄存器
UTIMER4_IE	0x1C	Timer4 中断使能寄存器
UTIMER4_IF	0x20	Timer4 中断标志寄存器

Timer0/1/4 完全相同。

Timer2/3 完全相同，与 Timer0/1/4 不同之处在于 Timer2/3 计数器相关寄存器为 32 位宽，而 Timer0/1 计数器相关寄存器为 16 位宽。

QEP0 在芯片中的基地址是 0x4001_4500

表 14-9 QEP0 寄存器地址分配

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

QEP1 在芯片中的基地址是 0x4001_4600

表 14-10 QEP1 寄存器地址分配

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

QEP2 在芯片中的基地址是 0x4001_4700

表 14-11 QEP2 寄存器地址分配

QEP2_CFG	0x00	QEP2 配置寄存器
QEP2_TH	0x04	QEP2 计数门限寄存器
QEP2_CNT	0x08	QEP2 计数值寄存器
QEP2_IE	0x0C	QEP2 中断使能寄存器
QEP2_IF	0x10	QEP2 中断标志寄存器

QEP3 在芯片中的基地址是 0x4001_4800

表 14-12 QEP2 寄存器地址分配

QEP3_CFG	0x00	QEP3 配置寄存器
QEP3_TH	0x04	QEP3 计数门限寄存器
QEP3_CNT	0x08	QEP3 计数值寄存器
QEP3_IE	0x0C	QEP3 中断使能寄存器
QEP3_IF	0x10	QEP3 中断标志寄存器

编码器 0/1/2/3 实现完全相同。

14.3.2 UTimer0 寄存器

14.3.2.1 UTIMER0_CFG Timer0 配置寄存器

地址：0x4001_4000

复位值：0x0

表 14-13 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 寄存器进行配置。
[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。 3'h0: 1 分频 3'h1: 2 分频 3'h2: 4 分频 3'h3: 8 分频 3'h4: 16 分频 3'h5: 32 分频 3'h6: 64 分频 3'h7: 128 分频
[19:16]	CLK_SRC	Timer 时钟源 4'h0: 芯片内部时钟 4'h1: 保留 4'h2: Timer0 通道 0 外部时钟信号 4'h3: Timer0 通道 1 外部时钟信号 4'h4: Timer1 通道 0 外部时钟信号 4'h5: Timer1 通道 1 外部时钟信号 4'h6: Timer2 通道 0 外部时钟信号 4'h7: Timer2 通道 1 外部时钟信号 4'h8: Timer3 通道 0 外部时钟信号 4'h9: Timer3 通道 1 外部时钟信号 4'hA: Timer4 通道 0 外部时钟信号 4'hB: Timer4 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 3'h1。 4'h0: Timer 通道 0，来自芯片 GPIO（参见 Datasheet 及应用配置） 4'h1: Timer 通道 1，来自芯片 GPIO（参见 Datasheet 及应用配置） 4'h2: 比较器 0 的输出 4'h3: 比较器 1 的输出 4'h4: 比较器 2 的输出 4'h5: 比较器 3 的输出 4'h6: 比较器 4 的输出 4'h7: 比较器 5 的输出 4'h8: 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。 4'h0: Timer 通道 0，来自芯片 GPIO（参见 Datasheet 及应用配置） 4'h1: Timer 通道 1，来自芯片 GPIO（参见 Datasheet 及应用配置） 4'h2: 比较器 0 的输出 4'h3: 比较器 1 的输出 4'h4: 比较器 2 的输出 4'h5: 比较器 3 的输出 4'h6: 比较器 4 的输出 4'h7: 比较器 5 的输出 4'h8: 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。

14.3.2.2 UTIMER0_TH Timer0 门限寄存器

地址：0x4001_4004

复位值：0x0

表 14-14 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

14.3.2.3 UTIMER0_CNT Timer0 计数寄存器

地址：0x4001_4008

复位值：0x0

表 14-15 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 开启 Timer 时钟。

14.3.2.4 UTIMER0_CMP0 Timer0 通道 0 比较捕获寄存器

地址：0x4001_400C

复位值：0x0

表 14-16 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 为恒 1，设置 CMP0=TH+1，可使得通道 0 为恒 0。

14.3.2.5 UTIMER0_CMP1 Timer0 通道 1 比较捕获寄存器

地址：0x4001_4010

复位值：0x0

表 14-17 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 1 为恒 1，设置 CMP1=TH+1，可使得通道 1 为恒 0。

14.3.2.6 UTIMER0_EVT Timer0 外部事件选择寄存器

地址：0x4001_4014

复位值：0x0

表 14-18 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 4：TIMER2 通道 0 比较事件 5：TIMER2 通道 1 比较事件 6：TIMER3 通道 0 比较事件 7：TIMER3 通道 1 比较事件 8：TIMER4 通道 0 比较事件 9：TIMER4 通道 1 比较事件 10：MCPWM0 TADC[0]比较事件 11：MCPWM0 TADC[1]比较事件 12：MCPWM0 TADC[2]比较事件 13：MCPWM0 TADC[3]比较事件 14：MCPWM1 TADC[0]比较事件 15：MCPWM1 TADC[1]比较事件 16：MCPWM1 TADC[2]比较事件 17：MCPWM1 TADC[3]比较事件

14.3.2.7 UTIMER0_FLT Timer0 滤波控制寄存器

地址：0x4001_4018

复位值：0x0

表 14-19 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.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。

14.3.2.8 UTIMER0_IE Timer0 中断使能寄存器

地址：0x4001_401C

复位值：0x0

表 14-20 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 事件 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。

14.3.2.9 UTIMER0_IF Timer0 中断标志寄存器

地址：0x4001_4020

复位值：0x0

表 14-21 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;

14.3.3 UTimer1 寄存器

14.3.3.1 UTIMER1_CFG Timer1 配置寄存器

地址：0x4001_4100

复位值：0x0

表 14-22 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 UTIMER1_EVT 寄存器进行配置。
[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。 3'h0: 1 分频 3'h1: 2 分频 3'h2: 4 分频 3'h3: 8 分频 3'h4: 16 分频 3'h5: 32 分频 3'h6: 64 分频 3'h7: 128 分频
[19:16]	CLK_SRC	Timer 时钟源 4'h0: 芯片内部时钟 4'h1: 保留 4'h2: Timer0 通道 0 外部时钟信号 4'h3: Timer0 通道 1 外部时钟信号 4'h4: Timer1 通道 0 外部时钟信号 4'h5: Timer1 通道 1 外部时钟信号 4'h6: Timer2 通道 0 外部时钟信号 4'h7: Timer2 通道 1 外部时钟信号 4'h8: Timer3 通道 0 外部时钟信号 4'h9: Timer3 通道 1 外部时钟信号 4'hA: Timer4 通道 0 外部时钟信号 4'hB: Timer4 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 3'h1。 4'h0: Timer 通道 0，来自芯片 GPIO（参见 Datasheet 及应用配置） 4'h1: Timer 通道 1，来自芯片 GPIO（参见 Datasheet 及应用配置） 4'h2: 比较器 0 的输出 4'h3: 比较器 1 的输出

		4'h4: 比较器 2 的输出 4'h5: 比较器 3 的输出 4'h6: 比较器 4 的输出 4'h7: 比较器 5 的输出 4'h8: 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。 4'h0: Timer 通道 0，来自芯片 GPIO（参见 Datasheet 及应用配置） 4'h1: Timer 通道 1，来自芯片 GPIO（参见 Datasheet 及应用配置） 4'h2: 比较器 0 的输出 4'h3: 比较器 1 的输出 4'h4: 比较器 2 的输出 4'h5: 比较器 3 的输出 4'h6: 比较器 4 的输出 4'h7: 比较器 5 的输出 4'h8: 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。

14.3.3.2 UTIMER1_TH Timer1 门限寄存器

地址：0x4001_4104

复位值：0x0

表 14-23 Timer1 门限寄存器 UTIMER1_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

14.3.3.3 UTIMER1_CNT Timer1 计数寄存器

地址：0x4001_4108

复位值：0x0

表 14-24 Timer1 计数寄存器 UTIMER1_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 UTIMER1_CNT 前需要先通过 SYS_CLK_FEN 开启 Timer 时钟。

14.3.3.4 UTIMER1_CMP0 Timer1 通道 0 比较捕获寄存器

地址：0x4001_410C

复位值：0x0

表 14-25 Timer1 通道 0 比较捕获寄存器 UTIMER1_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。

14.3.3.5 UTIMER1_CMP1 Timer1 通道 1 比较捕获寄存器

地址：0x4001_4110

复位值：0x0

表 14-26 Timer1 通道 1 比较捕获寄存器 UTIMER1_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。

14.3.3.6 UTIMER1_EVT Timer1 外部事件选择寄存器

地址：0x4001_4114

复位值：0x0

表 14-27 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
[31:4]		未使用
[3:0]	EVT_SRC	Timer1 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 4：TIMER2 通道 0 比较事件 5：TIMER2 通道 1 比较事件 6：TIMER3 通道 0 比较事件 7：TIMER3 通道 1 比较事件 8：TIMER4 通道 0 比较事件 9：TIMER4 通道 1 比较事件 10：MCPWM0 TADC[0]比较事件 11：MCPWM0 TADC[1]比较事件 12：MCPWM0 TADC[2]比较事件 13：MCPWM0 TADC[3]比较事件 14：MCPWM1 TADC[0]比较事件 15：MCPWM1 TADC[1]比较事件 16：MCPWM1 TADC[2]比较事件 17：MCPWM1 TADC[3]比较事件

14.3.3.7 UTIMER1_FLT Timer1 滤波控制寄存器

地址：0x4001_4118

复位值：0x0

表 14-28 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。

14.3.3.8 UTIMER1_IE Timer1 中断使能寄存器

地址：0x4001_411C

复位值：0x0

表 14-29 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 事件 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。

14.3.3.9 UTIMER1_IF Timer1 中断标志寄存器

地址：0x4001_4120

复位值：0x0

表 14-30 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

14.3.4 UTimer2 寄存器

14.3.4.1 UTIMER2_CFG Timer2 配置寄存器

地址：0x4001_4200

复位值：0x0

表 14-31 Timer2 配置寄存器 UTIMER2_CFG

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

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

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 UTIMER2_EVT 寄存器进行配置。
[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。 3'h0: 1 分频 3'h1: 2 分频 3'h2: 4 分频 3'h3: 8 分频 3'h4: 16 分频 3'h5: 32 分频 3'h6: 64 分频 3'h7: 128 分频
[19:16]	CLK_SRC	Timer 时钟源 4'h0: 芯片内部时钟 4'h1: 保留 4'h2: Timer0 通道 0 外部时钟信号 4'h3: Timer0 通道 1 外部时钟信号 4'h4: Timer1 通道 0 外部时钟信号 4'h5: Timer1 通道 1 外部时钟信号 4'h6: Timer2 通道 0 外部时钟信号 4'h7: Timer2 通道 1 外部时钟信号 4'h8: Timer3 通道 0 外部时钟信号 4'h9: Timer3 通道 1 外部时钟信号 4'hA: Timer4 通道 0 外部时钟信号 4'hB: Timer4 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 3'h1。 4'h0: Timer 通道 0，来自芯片 GPIO（参见 Datasheet 及应用配置） 4'h1: Timer 通道 1，来自芯片 GPIO（参见 Datasheet 及应用配置） 4'h2: 比较器 0 的输出 4'h3: 比较器 1 的输出 4'h4: 比较器 2 的输出 4'h5: 比较器 3 的输出

		4'h6: 比较器 4 的输出 4'h7: 比较器 5 的输出 4'h8: 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。 4'h0: Timer 通道 0，来自芯片 GPIO（参见 Datasheet 及应用配置） 4'h1: Timer 通道 1，来自芯片 GPIO（参见 Datasheet 及应用配置） 4'h2: 比较器 0 的输出 4'h3: 比较器 1 的输出 4'h4: 比较器 2 的输出 4'h5: 比较器 3 的输出 4'h6: 比较器 4 的输出 4'h7: 比较器 5 的输出 4'h8: 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。

14.3.4.2 UTIMER2_TH Timer2 门限寄存器

地址：0x4001_4204

复位值：0x0

表 14-32 Timer2 门限寄存器 UTIMER2_TH

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

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

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

14.3.4.3 UTIMER2_CNT Timer2 计数寄存器

地址：0x4001_4208

复位值：0x0

表 14-33 Timer2 计数寄存器 UTIMER2_CNT

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

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

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

注意：写入 UTIMER2_CNT 前需要先通过 SYS_CLK_FEN 开启 Timer 时钟。

14.3.4.4 UTIMER2_CMP0 Timer2 通道 0 比较捕获寄存器

地址：0x4001_420C

复位值：0x0

表 14-34 Timer2 通道 0 比较捕获寄存器 UTIMER2_CMP0

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

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

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。

14.3.4.5 UTIMER2_CMP1 Timer2 通道 1 比较捕获寄存器

地址：0x4001_4210

复位值：0x0

表 14-35 Timer2 通道 1 比较捕获寄存器 UTIMER2_CMP1

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

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

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。

14.3.4.6 UTIMER2_EVT Timer2 外部事件选择寄存器

地址：0x4001_4214

复位值：0x0

表 14-36 Timer2 外部事件选择寄存器 UTIMER2_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 UTIMER2_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 4：TIMER2 通道 0 比较事件 5：TIMER2 通道 1 比较事件 6：TIMER3 通道 0 比较事件 7：TIMER3 通道 1 比较事件 8：TIMER4 通道 0 比较事件 9：TIMER4 通道 1 比较事件 10：MCPWM0 TADC[0]比较事件 11：MCPWM0 TADC[1]比较事件 12：MCPWM0 TADC[2]比较事件 13：MCPWM0 TADC[3]比较事件 14：MCPWM1 TADC[0]比较事件 15：MCPWM1 TADC[1]比较事件 16：MCPWM1 TADC[2]比较事件 17：MCPWM1 TADC[3]比较事件

14.3.4.7 UTIMER2_FLT Timer2 滤波控制寄存器

地址：0x4001_4218

复位值：0x0

表 14-37 Timer2 滤波控制寄存器 UTIMER2_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。

14.3.4.8 UTIMER2_IE Timer2 中断使能寄存器

地址：0x4001_421C

复位值：0x0

表 14-38 Timer2 中断使能寄存器 UTIMER2_IE

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

位置	位名称	说明
[31:11]		未使用
[10]	ZC_RE	Timer 计数器过 0 事件 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。

14.3.4.9 UTIMER2_IF Timer2 中断标志寄存器

地址：0x4001_4220

复位值：0x0

表 14-39 Timer2 中断标志寄存器 UTIMER2_IF

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

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

14.3.5 UTimer3 寄存器

14.3.5.1 UTIMER3_CFG Timer3 配置寄存器

地址：0x4001_4300

复位值：0x0

表 14-40 Timer3 配置寄存器 UTIMER3_CFG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN				CAP1_CLR_EN	CAP0_CLR_EN	ONE_TRIG	ETON		CLK_DIV			CLK_SRC			
RW				RW	RW	RW	RW		RW			RW			
0				0	0	0	0		0			0			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SRC1	CH1_POL	CH1_MODE	CH1_FE_CAP_EN	CH1_RE_CAP_EN	SRC0	CH0_POL	CH0_MODE	CH0_FE_CAP_EN	CH0_RE_CAP_EN
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0001	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31]	EN	Timer 模块整体使能，高有效
[30:26]		未使用
[25]	CAP1_CLR_EN	当发生 CAP1 捕获事件时，清零 Timer 计数器，高有效
[24]	CAP0_CLR_EN	当发生 CAP0 捕获事件时，清零 Timer 计数器，高有效
[23]	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。
[22]	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 UTIMER3_EVT 寄存器进行配置。
[21:18]	CLK_SRC	Timer 时钟源 4'h0: 芯片内部时钟 4'h1: 保留 4'h2: Timer0 通道 0 外部时钟信号 4'h3: Timer0 通道 1 外部时钟信号 4'h4: Timer1 通道 0 外部时钟信号 4'h5: Timer1 通道 1 外部时钟信号 4'h6: Timer2 通道 0 外部时钟信号 4'h7: Timer2 通道 1 外部时钟信号 4'h8: Timer3 通道 0 外部时钟信号 4'h9: Timer3 通道 1 外部时钟信号 4'hA: Timer4 通道 0 外部时钟信号 4'hB: Timer4 通道 1 外部时钟信号
[17:16]	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。 3'h0: 1 分频 3'h1: 2 分频 3'h2: 4 分频 3'h3: 8 分频 3'h4: 16 分频 3'h5: 32 分频 3'h6: 64 分频 3'h7: 128 分频

[15:12]	SRC1	Timer 捕获模式通道 1 信号来源。默认为 3'h1。 4'h0: Timer 通道 0，来自芯片 GPIO（参见 Datasheet 及应用配置） 4'h1: Timer 通道 1，来自芯片 GPIO（参见 Datasheet 及应用配置） 4'h2: 比较器 0 的输出 4'h3: 比较器 1 的输出 4'h4: 比较器 2 的输出 4'h5: 比较器 3 的输出 4'h6: 比较器 4 的输出 4'h7: 比较器 5 的输出 4'h8: 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。 4'h0: Timer 通道 0，来自芯片 GPIO（参见 Datasheet 及应用配置） 4'h1: Timer 通道 1，来自芯片 GPIO（参见 Datasheet 及应用配置） 4'h2: 比较器 0 的输出 4'h3: 比较器 1 的输出 4'h4: 比较器 2 的输出 4'h5: 比较器 3 的输出 4'h6: 比较器 4 的输出 4'h7: 比较器 5 的输出 4'h8: 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。

14.3.5.2 UTIMER3_TH Timer3 门限寄存器

地址：0x4001_4304

复位值：0x0

表 14-41 Timer3 门限寄存器 UTIMER3_TH

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

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

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

14.3.5.3 UTIMER3_CNT Timer3 计数寄存器

地址：0x4001_4308

复位值：0x0

表 14-42 Timer3 计数寄存器 UTIMER3_CNT

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

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

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

注意：写入 UTIMER3_CNT 前需要先通过 SYS_CLK_FEN 开启 Timer 时钟。

14.3.5.4 UTIMER3_CMP0 Timer3 通道 0 比较捕获寄存器

地址：0x4001_430C

复位值：0x0

表 14-43 Timer3 通道 0 比较捕获寄存器 UTIMER3_CMP0

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

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

位置	位名称	说明
[31:0]	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。

14.3.5.5 UTIMER3_CMP1 Timer3 通道 1 比较捕获寄存器

地址：0x4001_4310

复位值：0x0

表 14-44 Timer3 通道 1 比较捕获寄存器 UTIMER3_CMP1

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

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

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。

14.3.5.6 UTIMER3_EVT Timer3 外部事件选择寄存器

地址：0x4001_4314

复位值：0x0

表 14-45 Timer3 外部事件选择寄存器 UTIMER3_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 UTIMER3_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 4：TIMER2 通道 0 比较事件 5：TIMER2 通道 1 比较事件 6：TIMER3 通道 0 比较事件 7：TIMER3 通道 1 比较事件 8：TIMER4 通道 0 比较事件 9：TIMER4 通道 1 比较事件 10：MCPWM0 TADC[0]比较事件 11：MCPWM0 TADC[1]比较事件 12：MCPWM0 TADC[2]比较事件 13：MCPWM0 TADC[3]比较事件 14：MCPWM1 TADC[0]比较事件 15：MCPWM1 TADC[1]比较事件 16：MCPWM1 TADC[2]比较事件 17：MCPWM1 TADC[3]比较事件

14.3.5.7 UTIMER3_FLT Timer3 滤波控制寄存器

地址：0x4001_4318

复位值：0x0

表 14-46 Timer3 滤波控制寄存器 UTIMER3_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。

14.3.5.8 UTIMER3_IE Timer3 中断使能寄存器

地址：0x4001_431C

复位值：0x0

表 14-47 Timer3 中断使能寄存器 UTIMER3_IE

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

位置	位名称	说明
[31:11]		未使用
[10]	ZC_RE	Timer 计数器过 0 事件 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。

14.3.5.9 UTIMER3_IF Timer3 中断标志寄存器

地址：0x4001_4320

复位值：0x0

表 14-48 Timer3 中断标志寄存器 UTIMER3_IF

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

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

14.3.6 UTimer4 寄存器

14.3.6.1 UTIMER4_CFG Timer4 配置寄存器

地址：0x4001_4400

复位值：0x0

表 14-49 Timer4 配置寄存器 UTIMER4_CFG

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

EN		CAP1_CLR_EN	CAP0_CLR_EN	ONE_TRIG	ETON		CLK_DIV	CLK_SRC
RW		RW	RW	RW	RW		RW	RW
0		0	0	0	0		0	0

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

SRC1				CH1_POL	CH1_MODE	CH1_FE_CAP_EN	CH1_RE_CAP_EN	SRC0				CH0_POL	CH0_MODE	CH0_FE_CAP_EN	CH0_RE_CAP_EN
RW				RW	RW	RW	RW	RW				RW	RW	RW	RW
0001				0	0	0	0	0				0	0	0	0

位置	位名称	说明
[31]	EN	Timer 模块整体使能，高有效
[30:26]		未使用
[25]	CAP1_CLR_EN	当发生 CAP1 捕获事件时，清零 Timer 计数器，高有效
[24]	CAP0_CLR_EN	当发生 CAP0 捕获事件时，清零 Timer 计数器，高有效
[23]	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。
[22]	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 UTIMER4_EVT 寄存器进行配置。
[21:20]	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。 3'h0: 1 分频 3'h1: 2 分频 3'h2: 4 分频 3'h3: 8 分频 3'h4: 16 分频 3'h5: 32 分频 3'h6: 64 分频 3'h7: 128 分频
[19:16]	CLK_SRC	Timer 时钟源 4'h0: 芯片内部时钟 4'h1: 保留 4'h2: Timer0 通道 0 外部时钟信号 4'h3: Timer0 通道 1 外部时钟信号 4'h4: Timer1 通道 0 外部时钟信号

		4'h5: Timer1 通道 1 外部时钟信号 4'h6: Timer2 通道 0 外部时钟信号 4'h7: Timer2 通道 1 外部时钟信号 4'h8: Timer3 通道 0 外部时钟信号 4'h9: Timer3 通道 1 外部时钟信号 4'hA: Timer4 通道 0 外部时钟信号 4'hB: Timer4 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 3'h1。 4'h0: Timer 通道 0，来自芯片 GPIO（参见 Datasheet 及应用配置） 4'h1: Timer 通道 1，来自芯片 GPIO（参见 Datasheet 及应用配置） 4'h2: 比较器 0 的输出 4'h3: 比较器 1 的输出 4'h4: 比较器 2 的输出 4'h5: 比较器 3 的输出 4'h6: 比较器 4 的输出 4'h7: 比较器 5 的输出 4'h8: 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。 4'h0: Timer 通道 0，来自芯片 GPIO（参见 Datasheet 及应用配置） 4'h1: Timer 通道 1，来自芯片 GPIO（参见 Datasheet 及应用配置） 4'h2: 比较器 0 的输出 4'h3: 比较器 1 的输出 4'h4: 比较器 2 的输出 4'h5: 比较器 3 的输出 4'h6: 比较器 4 的输出 4'h7: 比较器 5 的输出 4'h8: Timer 通道 0 和 1 的异或
[3]	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。
[2]	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。
[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。

14.3.6.2 UTIMER4_TH Timer4 门限寄存器

地址：0x4001_4404

复位值：0x0

表 14-50 Timer4 门限寄存器 UTIMER4_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

14.3.6.3 UTIMER4_CNT Timer4 计数寄存器

地址：0x4001_4408

复位值：0x0

表 14-51 Timer4 计数寄存器 UTIMER4_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 UTIMER4_CNT 前需要先通过 SYS_CLK_FEN 开启 Timer 时钟。

14.3.6.4 UTIMER4_CMP0 Timer4 通道 0 比较捕获寄存器

地址：0x4001_440C

复位值：0x0

表 14-52 Timer4 通道 0 比较捕获寄存器 UTIMER4_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。

14.3.6.5 UTIMER4_CMP1 Timer4 通道 1 比较捕获寄存器

地址：0x4001_4410

复位值：0x0

表 14-53 Timer4 通道 1 比较捕获寄存器 UTIMER4_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。

14.3.6.6 UTIMER4_EVT Timer4 外部事件选择寄存器

地址：0x4001_4414

复位值：0x0

表 14-54 Timer4 外部事件选择寄存器 UTIMER4_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 UTIMER4_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 4：TIMER2 通道 0 比较事件 5：TIMER2 通道 1 比较事件 6：TIMER3 通道 0 比较事件 7：TIMER3 通道 1 比较事件 8：TIMER4 通道 0 比较事件 9：TIMER4 通道 1 比较事件 10：MCPWM0 TADC[0]比较事件 11：MCPWM0 TADC[1]比较事件 12：MCPWM0 TADC[2]比较事件 13：MCPWM0 TADC[3]比较事件 14：MCPWM1 TADC[0]比较事件 15：MCPWM1 TADC[1]比较事件 16：MCPWM1 TADC[2]比较事件 17：MCPWM1 TADC[3]比较事件

14.3.6.7 UTIMER4_FLT Timer4 滤波控制寄存器

地址：0x4001_4418

复位值：0x0

表 14-55Timer4 滤波控制寄存器 UTIMER4_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。

14.3.6.8 UTIMER4_IE Timer4 中断使能寄存器

地址：0x4001_441C

复位值：0x0

表 14-56 Timer4 中断使能寄存器 UTIMER4_IE

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

位置	位名称	说明
[31:11]		未使用
[10]	ZC_RE	Timer 计数器过 0 事件 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。

14.3.6.9 UTIMER4_IF Timer4 中断标志寄存器

地址：0x4001_4420

复位值：0x0

表 14-57 Timer4 中断标志寄存器 UTIMER4_IF

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

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

14.3.7 QEP0 寄存器

14.3.7.1 QEP0_CFG QEP0 配置寄存器

QEP0_CFG 地址：0x4001_4500

复位值：0x0

表 14-58 QEP0 配置寄存器 QEP0_CFG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN					FE_CNT_EN	MODE						ZNC	ZPC	ZLC	ZEC
RW					RW	RW						RW	RW	RW	RW
0					0	0						0	0	0	0

位置	位名称	说明
[31:16]		未使用

[15]		QEP 模块整体使能
[14:11]		未使用
[10]	FE_CNT_EN	CCW+SIGN/CCW+CW 两种模式下，是否在下降沿进行计数（上升沿总是计数）
[9:8]	MODE	编码器模式选择 00：counting on T1， 01：counting on T1 & T2 以上两种模式都为正交编码信号计数模式 10：CCW+SIGN，符号加脉冲信号计数模式 11：CCW+CW，CCW+CW 双脉冲信号计数模式
[7:4]		未使用
[3]	ZNC	Z 信号清零极性选择：低电平/下降沿清零使能
[2]	ZPC	Z 信号清零极性选择：高电平/上降沿清零使能
[1]	ZLC	Z 信号电平清零 QEP 计数器使能
[0]	ZEC	Z 信号边沿清零 QEP 计数器使能

当 ZEC=1 时，使用 Z 信号的跳变沿进行 QEP 计数的清零，当 ZNC 同时为 1 时，Z 信号的下降沿清零 QEP 计数器，当 ZPC 同时为 1 时，Z 信号的上升沿清零 QEP 计数器；ZNC 和 ZPC 可以同时为 1，则 Z 信号发生变化则清零 QEP 计数器，清零后立即释放复位，QEP 计数器立即准备好进行后续计数。

当 ZLC=1 时，使用 Z 信号的有效电平进行 QEP 计数的清零，当 ZNC 同时为 1 时，Z 信号低电平时清零 QEP 计数器，当 ZPC 同时为 1 时，Z 信号高电平清零 QEP 计数器；通常使用 Z 信号有效电平清零 QEP 计数器时，ZNC 和 ZPC 不会同时为 1，否则计数器一直处于清零状态。使用有效电平清零时，只有 Z 信号翻转后，QEP 计数器才会恢复计数。

14.3.7.2 QEP0_TH QEP0 计数门限寄存器

QEP0_TH 地址：0x4001_4504

复位值：0x0

表 14-59 QEP0 计数门限寄存器 QEP0_TH

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

位置	位名称	说明
[31:16]		未使用
[15:0]	TH	计数门限 TH。编码器向上计数（增）到 TH 值后，再次向上计数会导致计数器回到 0。编码器向下计数（减）到 0 值后，再次向下计数会导致计数器回到 TH。

14.3.7.3 QEP0_CNT QEP0 计数值寄存器

QEP0_CNT 地址：0x4001_4508

复位值：0x0

表 14-60 QEP0 计数值寄存器 QEP0_CNT

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

位置	位名称	说明
[31:16]		未使用
[15:0]	CNT	QEP0 计数值。

14.3.7.4 QEP0_IE QEP0 中断使能寄存器

地址：0x4001_450C

复位值：0x0

表 14-61 QEP0 中断使能寄存器 QEP0_IE

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

位置	位名称	说明
[31:2]		未使用
[1]	OF_IE	上溢出中断使能，高电平有效。
[0]	UF_IE	下溢出中断使能，高电平有效。

14.3.7.5 QEP0_IF QEP0 中断标志寄存器

地址：0x4001_4510

复位值：0x0

表 14-62 QEP0 中断标志寄存器 QEP0_IF

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

	OF_IF	UF_IF
	RW1C	RW1C
	0	0

位置	位名称	说明
[31:2]		未使用
[1]	OF_IF	上溢出中断标志。高电平有效，写 1 清 0。当计数器计数达到计数门限时，上计数事件触发上溢出中断。
[0]	UF_IF	下溢出中断标志。高电平有效，写 1 清 0。当计数器计数达到 0 时，下计数事件触发下溢出中断。

14.3.8 QEP1 寄存器

14.3.8.1 QEP1_CFG QEP1 配置寄存器

QEP1_CFG 地址：0x4001_4600

复位值：0x0

表 14-63 QEP1 配置寄存器 QEP1_CFG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN					FE_CNT_EN	MODE						ZNC	ZPC	ZLC	ZEC
RW					RW	RW						RW	RW	RW	RW
0					0	0						0	0	0	0

位置	位名称	说明
[31:11]		未使用
[10]	FE_CNT_EN	CCW+SIGN/CCW+CW 两种模式下，是否在下降沿进行计数（上升沿总是计数）
[9:8]	MODE	编码器模式选择 00：counting on T1， 01：counting on T1 & T2 以上两种模式都为正交编码信号计数模式 10：CCW+SIGN，符号加脉冲信号计数模式 11：CCW+CW，CCW+CW 双脉冲信号计数模式
[7:4]		未使用

[3]	ZNC	Z 信号清零极性选择：低电平/下降沿清零使能
[2]	ZPC	Z 信号清零极性选择：高电平/上降沿清零使能
[1]	ZLC	Z 信号电平清零 QEP 计数器使能
[0]	ZEC	Z 信号边沿清零 QEP 计数器使能

当 ZEC=1 时，使用 Z 信号的跳变沿进行 QEP 计数的清零，当 ZNC 同时为 1 时，Z 信号的下降沿清零 QEP 计数器，当 ZPC 同时为 1 时，Z 信号的上升沿清零 QEP 计数器；ZNC 和 ZPC 可以同时为 1，则 Z 信号发生变化则清零 QEP 计数器，清零后立即释放复位，QEP 计数器立即准备好进行后续计数。

当 ZLC=1 时，使用 Z 信号的有效电平进行 QEP 计数的清零，当 ZNC 同时为 1 时，Z 信号低电平时清零 QEP 计数器，当 ZPC 同时为 1 时，Z 信号高电平清零 QEP 计数器；通常使用 Z 信号有效电平清零 QEP 计数器时，ZNC 和 ZPC 不会同时为 1，否则计数器一直处于清零状态。使用有效电平清零时，只有 Z 信号翻转后，QEP 计数器才会恢复计数。

14.3.8.2 QEP1_TH QEP1 计数门限寄存器

QEP1_TH 地址：0x4001_4604

复位值：0x0

表 14-64 QEP1 计数门限寄存器 QEP1_TH

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

位置	位名称	说明
[31:16]		未使用
[15:0]	TH	QEP 计数门限 TH。编码器向上计数（增）到 TH 值后，再次向上计数会导致计数器回到 0。编码器向下计数（减）到 0 值后，再次向下计数会导致计数器回到 TH。

14.3.8.3 QEP1_CNT QEP1 计数值寄存器

QEP1_CNT 地址：0x4001_4608

复位值：0x0

表 14-65 QEP1 计数值寄存器 QEP1_CNT

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

位置	位名称	说明
[31:16]		未使用
[15:0]	CNT	QEP 计数值。

14.3.8.4 QEP1_IE QEP1 中断使能寄存器

地址：0x4001_460C

复位值：0x0

表 14-66 QEP1 中断使能寄存器 QEP1_IE

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

位置	位名称	说明
[31:2]		未使用
[1]	OF_IE	上溢出中断使能，高电平有效。
[0]	UF_IE	下溢出中断使能，高电平有效。

14.3.8.5 QEP1_IF QEP1 中断标志寄存器

地址：0x4001_4610

复位值：0x0

表 14-67 QEP1 中断标志寄存器 QEP1_IF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														OF_IF	UF_IF
														RW1C	RW1C
														0	0

位置	位名称	说明
[31:2]		未使用

[1]	OF_IF	上溢出中断标志。高电平有效，写 1 清 0。当计数器计数达到计数门限时，上计数事件触发上溢出中断。
[0]	UF_IF	下溢出中断标志。高电平有效，写 1 清 0。当计数器计数达到 0 时，下计数事件触发下溢出中断。

14.3.9 QEP2 寄存器

14.3.9.1 QEP2_CFG QEP2 配置寄存器

QEP2_CFG 地址：0x4001_4700

复位值：0x0

表 14-68 QEP2 配置寄存器 QEP2_CFG

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN					FE_CNT_EN	MODE						ZNC	ZPC	ZLC	ZEC
RW					RW	RW						RW	RW	RW	RW
0					0	0						0	0	0	0

位置	位名称	说明
[31:11]		未使用
[10]	FE_CNT_EN	CCW+SIGN/CCW+CW 两种模式下，是否在下降沿进行计数（上升沿总是计数）
[9:8]	MODE	编码器模式选择 00：counting on T1， 01：counting on T1 & T2 以上两种模式都为正交编码信号计数模式 10：CCW+SIGN，符号加脉冲信号计数模式 11：CCW+CW，CCW+CW 双脉冲信号计数模式
[7:4]		未使用
[3]	ZNC	Z 信号清零极性选择：低电平/下降沿清零使能
[2]	ZPC	Z 信号清零极性选择：高电平/上降沿清零使能
[1]	ZLC	Z 信号电平清零 QEP 计数器使能
[0]	ZEC	Z 信号边沿清零 QEP 计数器使能

当 ZEC=1 时，使用 Z 信号的跳变沿进行 QEP 计数的清零，当 ZNC 同时为 1 时，Z 信号的下降沿清零 QEP 计数器，当 ZPC 同时为 1 时，Z 信号的上升沿清零 QEP 计数器；ZNC 和 ZPC 可以同时为 1，则 Z 信号发生变化则清零 QEP 计数器，清零后立即释放复位，QEP 计数器立即准备好进行后续计数。

当 ZLC=1 时，使用 Z 信号的有效电平进行 QEP 计数的清零，当 ZNC 同时为 1 时，Z 信号低电平时清零 QEP 计数器，当 ZPC 同时为 1 时，Z 信号高电平清零 QEP 计数器；通常使用 Z 信号有效电平清零 QEP 计数器时，ZNC 和 ZPC 不会同时为 1，否则计数器一直处于清零状态。使用有效电平清零时，只有 Z 信号翻转后，QEP 计数器才会恢复计数。

14.3.9.2 QEP2_TH QEP2 计数门限寄存器

QEP2_TH 地址：0x4001_4704

复位值：0x0

表 14-69 QEP2 计数门限寄存器 QEP2_TH

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

位置	位名称	说明
[31:16]		未使用
[15:0]	TH	QEP 计数门限 TH。编码器向上计数（增）到 TH 值后，再次向上计数会导致计数器回到 0。编码器向下计数（减）到 0 值后，再次向下计数会导致计数器回到 TH。

14.3.9.3 QEP2_CNT QEP2 计数值寄存器

QEP2_CNT 地址：0x4001_4708

复位值：0x0

表 14-70 QEP2 计数值寄存器 QEP2_CNT

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

位置	位名称	说明
[31:16]		未使用
[15:0]	CNT	QEP 计数值。

14.3.9.4 QEP2_IE QEP2 中断使能寄存器

地址：0x4001_470C

复位值：0x0

表 14-71 QEP2 中断使能寄存器 QEP2_IE

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

位置	位名称	说明
[31:2]		未使用
[1]	OF_IE	上溢出中断使能，高电平有效。
[0]	UF_IE	下溢出中断使能，高电平有效。

14.3.9.5 QEP2_IF QEP2 中断标志寄存器

地址：0x4001_4710

复位值：0x0

表 14-72 QEP2 中断标志寄存器 QEP2_IF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														OF_IF	UF_IF
														RW1C	RW1C
														0	0

位置	位名称	说明
[31:2]		未使用
[1]	OF_IF	上溢出中断标志。高电平有效，写 1 清 0。当计数器计数达到计数门限时，上计数事件触发上溢出中断。
[0]	UF_IF	下溢出中断标志。高电平有效，写 1 清 0。当计数器计数达到 0 时，下计数事件触发下溢出中断。

14.3.10 QEP3 寄存器

14.3.10.1 QEP3_CFG QEP3 配置寄存器

QEP3_CFG 地址：0x4001_4800

复位值：0x0

表 14-73 QEP3 配置寄存器 QEP3_CFG

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

位置	位名称	说明
[31:11]		未使用
[10]	FE_CNT_EN	CCW+SIGN/CCW+CW 两种模式下，是否在下降沿进行计数（上升沿总是计数）
[9:8]	MODE	编码器模式选择 00：counting on T1， 01：counting on T1 & T2 以上两种模式都为正交编码信号计数模式 10：CCW+SIGN，符号加脉冲信号计数模式 11：CCW+CW，CCW+CW 双脉冲信号计数模式
[7:4]		未使用
[3]	ZNC	Z 信号清零极性选择：低电平/下降沿清零使能
[2]	ZPC	Z 信号清零极性选择：高电平/上降沿清零使能
[1]	ZLC	Z 信号电平清零 QEP 计数器使能
[0]	ZEC	Z 信号边沿清零 QEP 计数器使能

当 ZEC=1 时，使用 Z 信号的跳变沿进行 QEP 计数的清零，当 ZNC 同时为 1 时，Z 信号的下降沿清零 QEP 计数器，当 ZPC 同时为 1 时，Z 信号的上升沿清零 QEP 计数器；ZNC 和 ZPC 可以同时为 1，则 Z 信号发生变化则清零 QEP 计数器，清零后立即释放复位，QEP 计数器立即准备好进行后续计数。

当 ZLC=1 时，使用 Z 信号的有效电平进行 QEP 计数的清零，当 ZNC 同时为 1 时，Z 信号低电平时清零 QEP 计数器，当 ZPC 同时为 1 时，Z 信号高电平清零 QEP 计数器；通常使用 Z 信号有效电平清零 QEP 计数器时，ZNC 和 ZPC 不会同时为 1，否则计数器一直处于清零状态。使用有效电平清零时，只有 Z 信号翻转后，QEP 计数器才会恢复计数。

14.3.10.2 QEP3_TH QEP3 计数门限寄存器

QEP3_TH 地址：0x4001_4804

复位值：0x0

表 14-74 QEP3 计数门限寄存器 QEP3_TH

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

位置	位名称	说明
[31:16]		未使用
[15:0]	TH	QEP 计数门限 TH。编码器向上计数（增）到 TH 值后，再次向上计数会导致计数器回到 0。编码器向下计数（减）到 0 值后，再次向下计数会导致计数器回到 TH。

14.3.10.3 QEP3_CNT　QEP3 计数值寄存器

QEP3_CNT 地址：0x4001_4808

复位值：0x0

表 14-75 QEP3 计数值寄存器 QEP3_CNT

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

位置	位名称	说明
[31:16]		未使用
[15:0]	CNT	QEP 计数值。

14.3.10.4 QEP3_IE　QEP3 中断使能寄存器

地址：0x4001_480C

复位值：0x0

表 14-76 QEP3 中断使能寄存器 QEP3_IE

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

位置	位名称	说明
[31:2]		未使用
[1]	OF_IE	上溢出中断使能，高电平有效。
[0]	UF_IE	下溢出中断使能，高电平有效。

14.3.10.5 QEP3_IF　QEP3 中断标志寄存器

地址：0x4001_4810

复位值：0x0

表 14-77 QEP3 中断标志寄存器 QEP3_IF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														OF_IF	UF_IF
														RW1C	RW1C
														0	0

位置	位名称	说明
[31:2]		未使用
[1]	OF_IF	上溢出中断标志。高电平有效，写 1 清 0。当计数器计数达到计数门限时，上计数事件触发上溢出中断。
[0]	UF_IF	下溢出中断标志。高电平有效，写 1 清 0。当计数器计数达到 0 时，下计数事件触发下溢出中断。

15 HALL 信号处理模块

15.1 综述

芯片共集成两个完全相同的 Hall 信号处理模块，以下说明均针对一个 Hall 信号处理模块

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，当 HALL 输入有变化时，记录当前的定时器值，并输出中断

溢出，当 HALL 信号长时间不发生变化导致计数器溢出时，输出中断

15.2 实现说明

15.2.1 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。

15.2.2 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。

15.2.3 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：

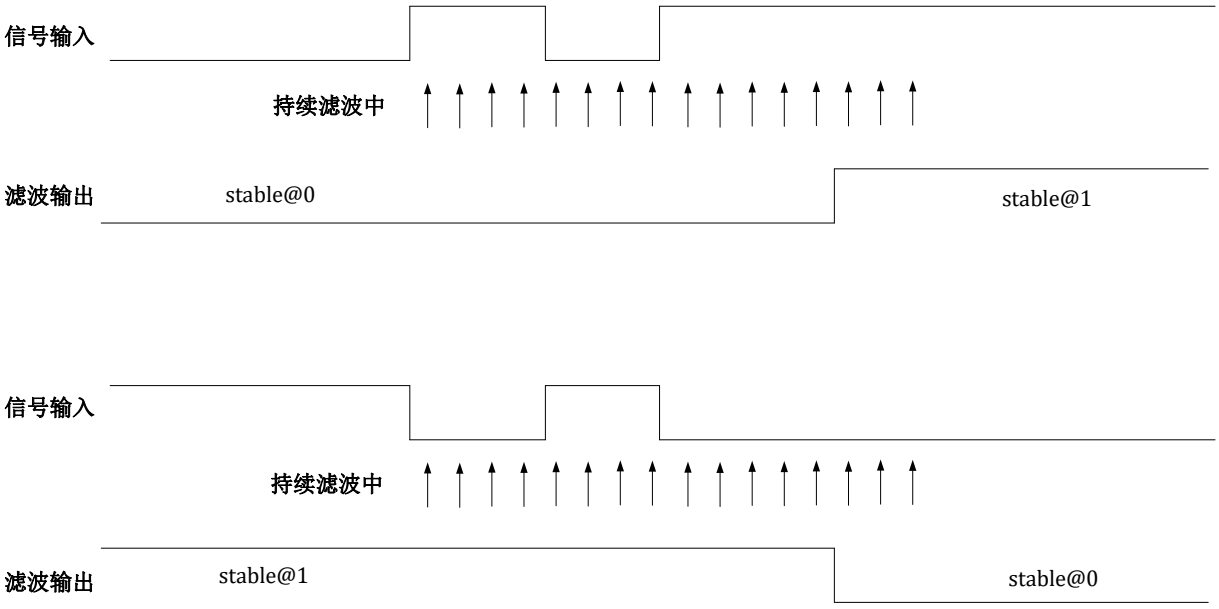


图 15-1 7/5 滤波模块框图

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：

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

举例，在 192MHz 工作频率下，周期 T_{clk} 是 5.2ns，寄存器配置最大为 32767，最长滤波宽度为约 5.2ns×32768≈170us。

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 15.3.3 HALLx_INFO 模块信息寄存器 (x = 0,1)。

15.2.4 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 192MHz 工作频率下，最大可以记录约 0.87 秒的时间宽度，达到 5.2ns 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。

15.2.5 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。中断标志可以通过向 HALL_INFO.CHG_IF 和 HALL_INFO.OV_IF 写 1 清空。

15.2.6 数据流程

HALL 模块的数据流程如下图所示，FCLK 为受 SYS_CLK_FEN 门控控制的系统主时钟，通常为 PLL 时钟。

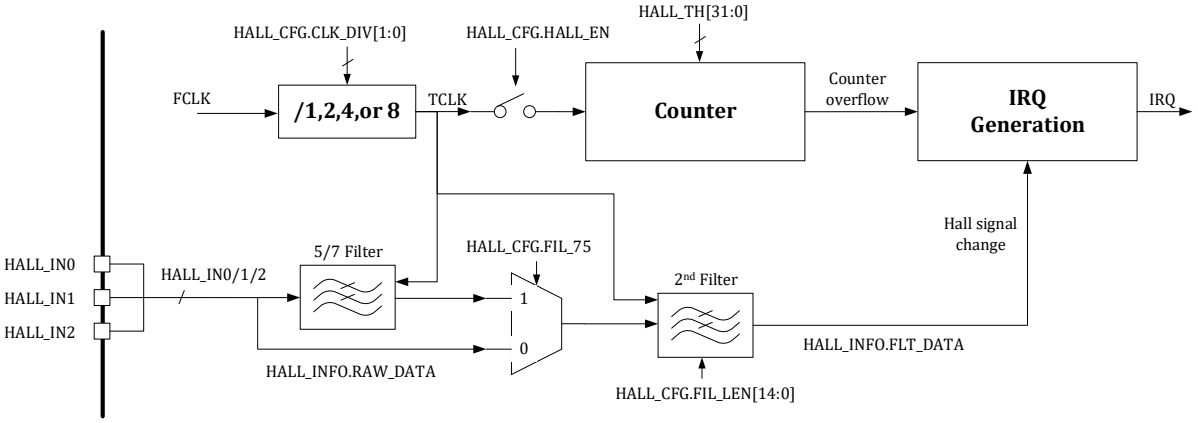


图 15-2 Hall 模块框图

15.3 寄存器

15.3.1 地址分配

HALL0 模块寄存器的基地址是 0x4001_1800，

HALL1 模块寄存器的基地址是 0x4001_1C00，

寄存器列表如下：

表 15-1HALL 模块寄存器地址分配

名称	偏移	描述
HALLx_CFG	0x00	HALL 模块配置寄存器
HALLx_INFO	0x04	HALL 模块信息寄存器
HALLx_WIDTH	0x08	HALL 宽度计数值寄存器
HALLx_TH	0x0C	HALL 模块计数器门限值寄存器
HALLx_CNT	0x10	HALL 计数寄存器

15.3.2 HALLx_CFG 模块配置寄存器(x = 0,1)

HALL0_CFG 地址：0x4001_1800

HALL1_CFG 地址：0x4001_1C00

复位值：0x0

表 15-2 HALL 模块配置寄存器 HALLx_CFG

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

	SW_IE	OV_IE	CHG_IE	SW_RE	OV_RE	CHG_RE	HALL_EN		FIL_75		CLK_DIV
	RW	RW	RW	RW	RW	RW	RW		RW		RW
	0	0	0	0	0	0	0		0		0

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

	FIL_LEN_
	RW
	0

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。有效此位后,INFO[18]写 1，将手动产生 HALL 信号变化中断。
[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]	SW_RE	软件触发 HALL 信号变化 DMA 请求使能，高电平有效。有效此位后，INFO[18]写 1，将手动产生 HALL 信号变化 DMA 请求。
[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。

15.3.3 HALLx_INFO 模块信息寄存器 (x = 0,1)

地址分别是：0x4001_1804,，0x4001_1C04

复位值：0x0

表 15-3 HALL 模块信息寄存器 HALLx_INFO

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

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

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

通常由于 SW_IF 自动清零,软件无法观测到 SW_IF 置位,但是软件触发 HALL 信号变化会使得 CFG_IF 置位，因此只要使能 CHG_IE 或 CFG_RE，并写入软件触发即可产生 HALL 中断或 DMA 请求事件。

15.3.4 HALLx_WIDTH 宽度计数值寄存器(x = 0,1)

地址分别是：0x4001_1808,，0x4001_1C08

复位值：0x0

表 15-4 HALL 宽度计数值寄存器 HALLx_WIDTH

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

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

RO
0

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

15.3.5 HALLx_TH 模块计数器门限值寄存器(x = 0,1)

地址分别是：0x4001_180C，0x4001_1C0C

复位值：0x0

表 15-5 HALL 模块计数器门限值寄存器 HALLx_TH

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

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

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

15.3.6 HALLx_CNT 计数寄存器(x = 0,1)

地址分别是：0x4001_1810,，0x4001_1C10

复位值：0x0

表 15-6 HALL 计数寄存器 HALLx_CNT

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

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

0

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

16 MCPWM

16.1 概述

MCPWM 模块，是一个精确控制电机驱动波形输出的模块。芯片共集成两个完全相同的 MCPWM 模块，以下说明均针对一个 MCPWM 模块。

一个 MCPWM 模块包含两个 16 位递增计数器，用于提供两个基础周期（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 192/96/48/24MHz。通道 0/1/2 固定使用时基 0，通道 3 固定使用时基 1。

包含四组 PWM 生成模块。

- 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 8 路急停信号，其中 4 路来自芯片 IO，4 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 输出信号复位到规定状态，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）。

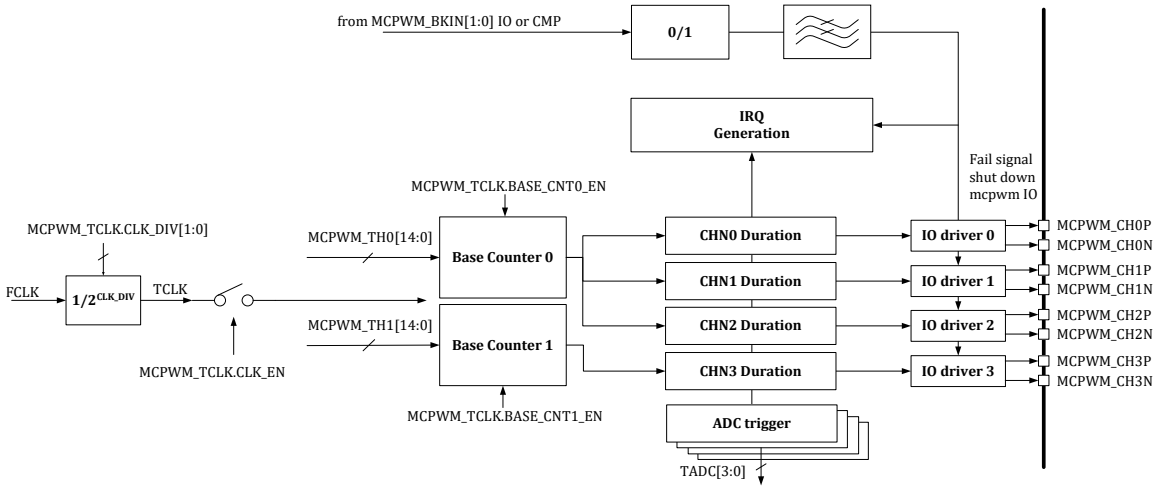


图 16-1 MCPWM 模块框图

为了获得更高的定时精度，通常采用 192MHz 的 PLL 时钟作为 MCPWM 模块工作时钟。

16.1.1 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，计数器在 t0 时刻从-TH 开始递增计数，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_IF.T0_IF 和 MCPWM_IF.T1_IF 将被置位。

可通过寄存器配置 MCPWM_TCLK.BASE_CNT_EN 控制 Base Counter 的启动和停止。

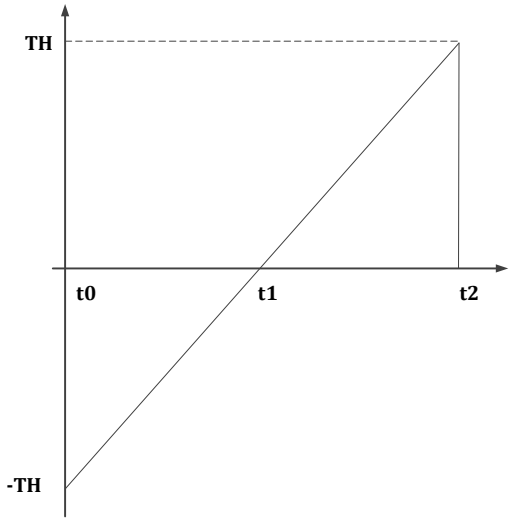


图 16-2 Base Counter t0/t1 时序

在运行 MCPWM 模块前，用户一般需将对应的比较门限值 MCPWM_TH00~MCWM_TH31，死区寄存器 MCPWM_DTH00~MCWM_DTH31 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，硬件把预装载寄存器的值载入到实际运行的影子寄存器中。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 或者二者皆可，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。

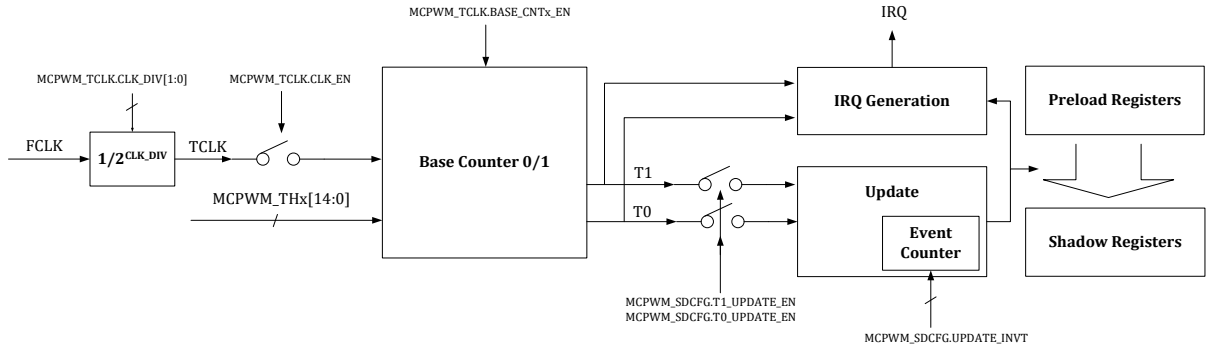


图 16-3 MCPWM 更新机制

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(因为时钟未使能，不进行计数)，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，来将 MCWPM_TCLk.BASE_CNT0_EN/MCWPM_TCLk.BASE_CNT1_EN 置位，从而分别使能时基 0 和时基 1 的 CNT 计数。

16.1.2 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 4 路 fail 信号输入 MCPWM，即 FAIL0~FAIL3，分别可以来自芯片 IO MCPWM_BKIN[3:0]或芯片内部比较器的输出 CMP[5:0]中的 4 路信号。

其中 MCPWM0 的通道 0/1/2 使用 CMP[1:0]，MCPWM0 的通道 3 使用 CMP[5:4]。MCPWM1 的通道 0/1/2 使用 CMP[3:2]，MCPWM1 的通道 3 使用 CMP[5:4]。

表 16-1 MCPWM FAIL 信号分布

	MCPWM0 CH0/1/2		MCPWM0 CH3		MCPWM1 CH0/1/2		MCPWM1 CH3	
	FAIL0	FAIL1	FAIL2	FAIL3	FAIL0	FAIL1	FAIL2	FAIL3
MCPWM0_BKIN0	√							
MCPWM0_BKIN1		√						
MCPWM0_BKIN2			√					
MCPWM0_BKIN3				√				
MCPWM1_BKIN0					√			
MCPWM1_BKIN1						√		
MCPWM1_BKIN2							√	
MCPWM1_BKIN3								√

CMP0	√							
CMP1		√						
CMP2					√			
CMP3						√		
CMP4			√				√	
CMP5				√				√

其中 MCPWMx_FAIL012 用于控制 CH012 的 FAIL 信号选择、极性、使能。MCPWMx_FAIL3 用于控制 CH3 的 FAIL 信号选择、极性、使能。

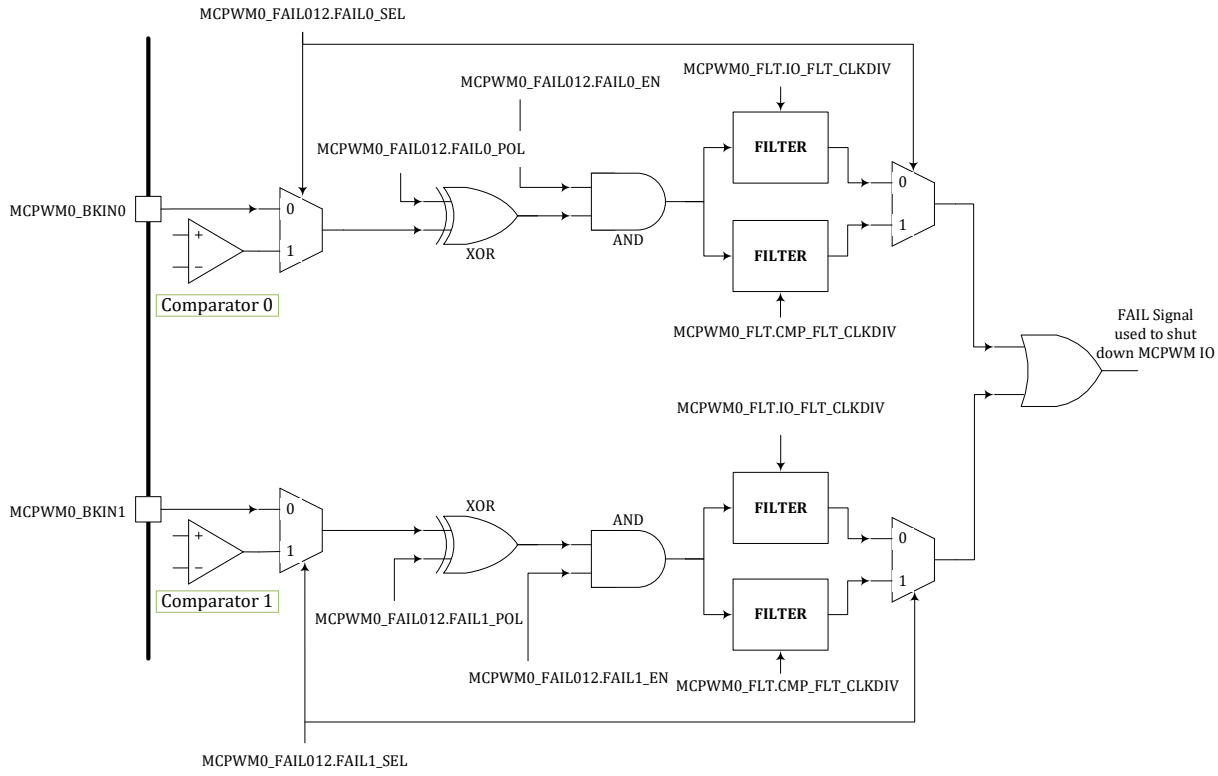


图 16-4 MCPWM0 FAIL0/1 逻辑示意图

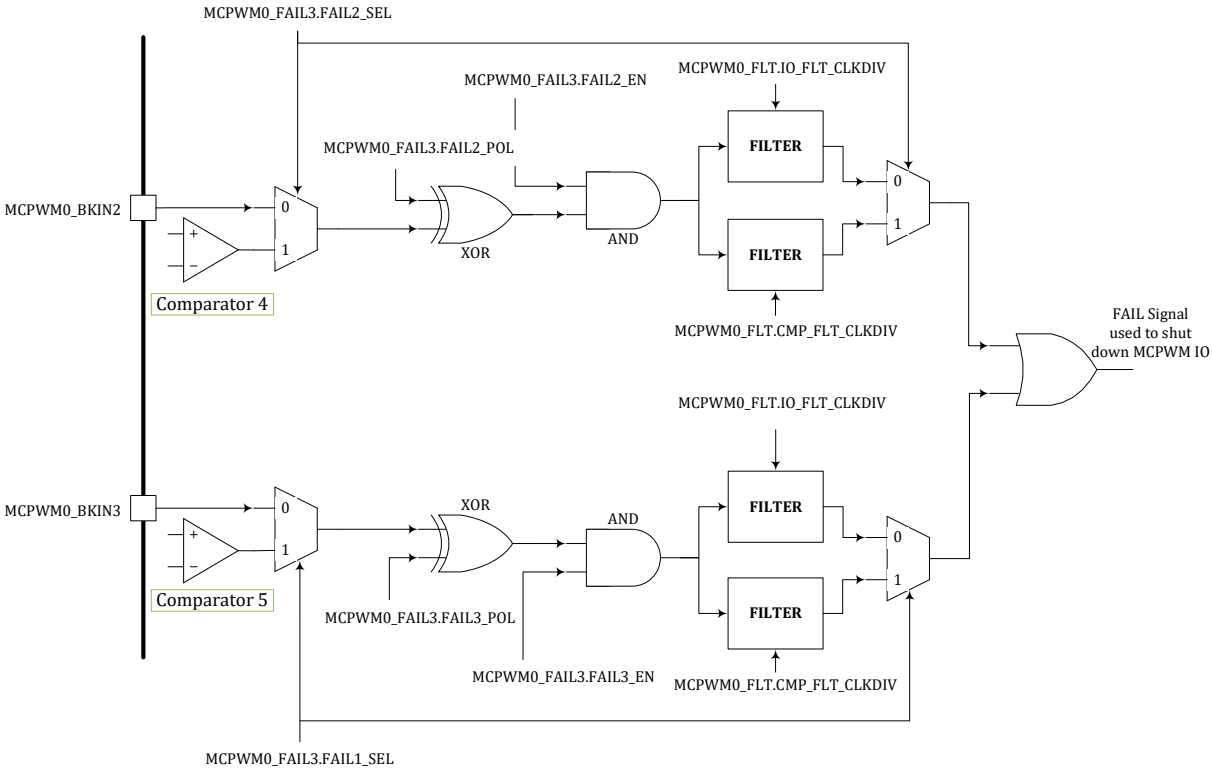


图 16-5 MCPWM0 FAIL2/3 逻辑示意图

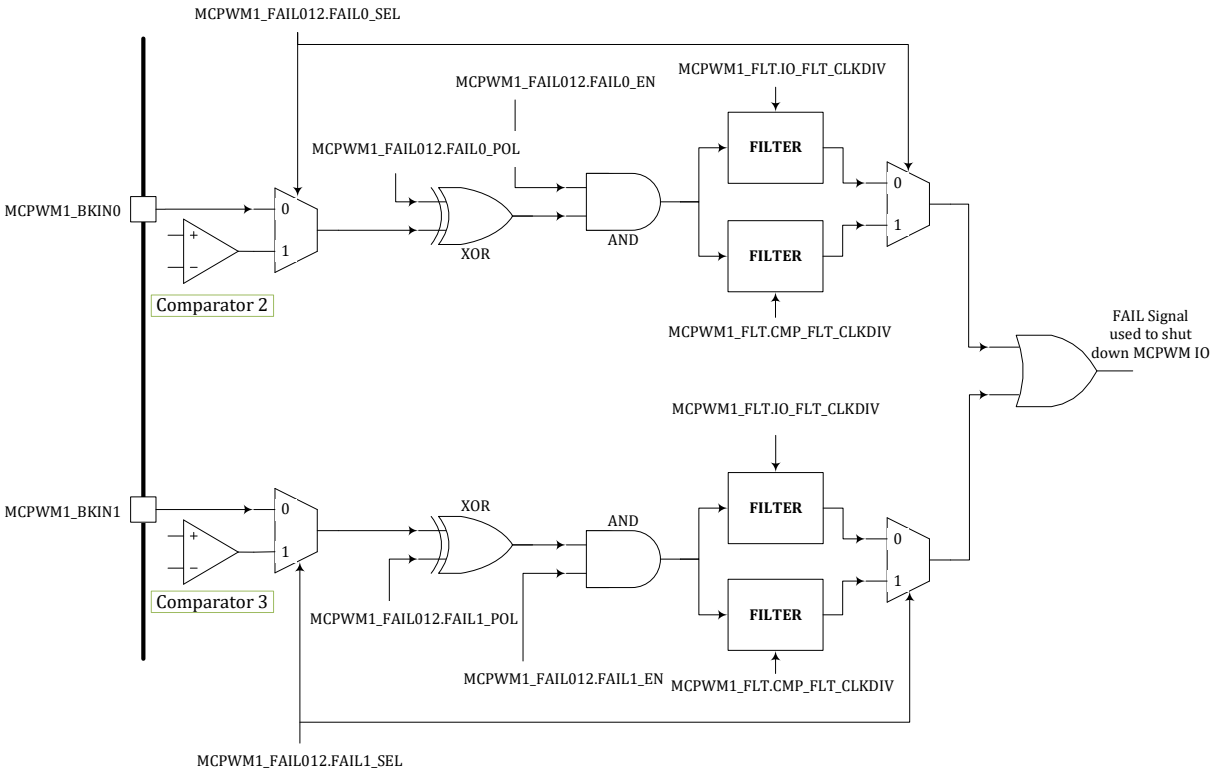


图 16-6 MCPWM1 FAIL0/1 逻辑示意图

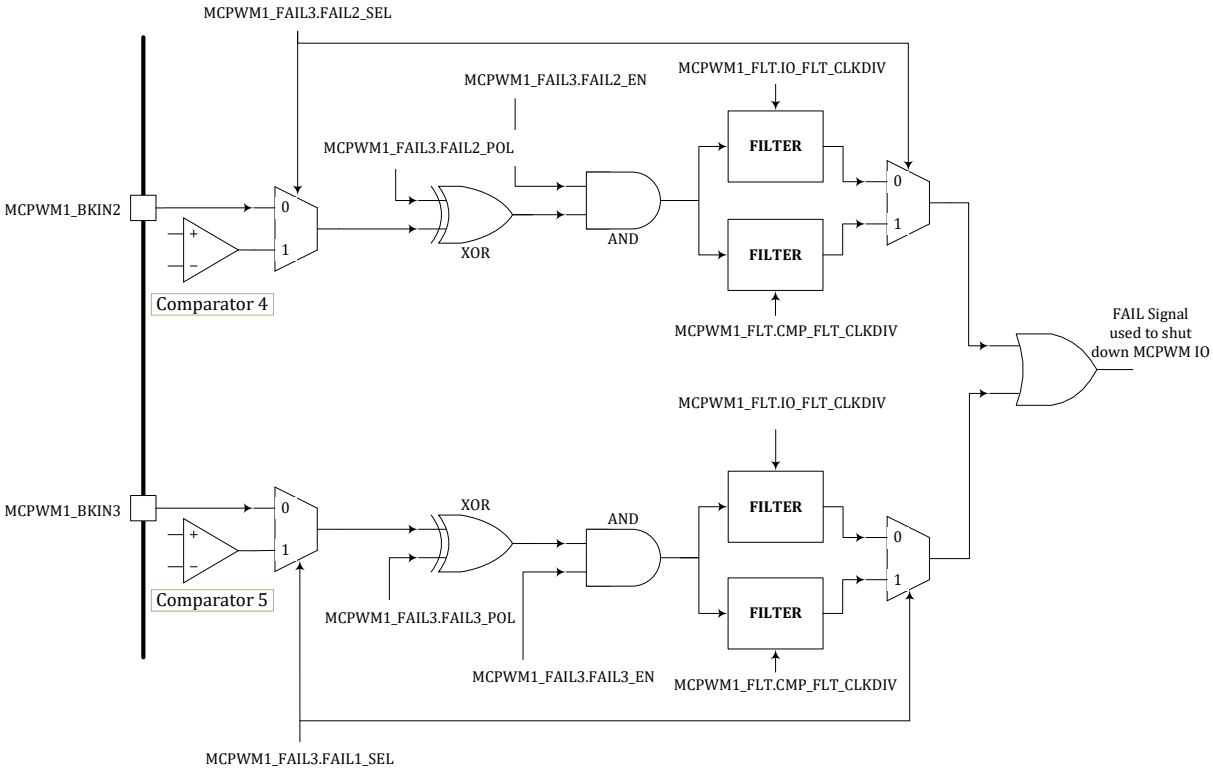


图 16-7 MCPWM1 FAIL2/3 逻辑示意图

Filter 滤波模块的时钟，来自系统主时钟 MCLK 的门控时钟 FCLK[10:9]，并经过两级分频，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 1~16 倍的分频，如果 FAIL 信号来自 MCPWM_BKIN[1:0]则使用 MCPWM_TCLK.IO_FLT_CLKDIV[3:0]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_TCLK.CMP_FLT_CLKDIV[3:0]作为第二级的分频系数，如图 16-8 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 个时钟（两级分频后的时钟）周期稳定后，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 FCLK 的时钟周期，192MHz 对应 5.2ns。MCPWM_TCLK.FLT_CLKDIV 根据配置情况可能是 MCPWM_TCLK.IO_FLT_CLKDIV 或 MCPWM_TCLK.CMP_FLT_CLKDIV。

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

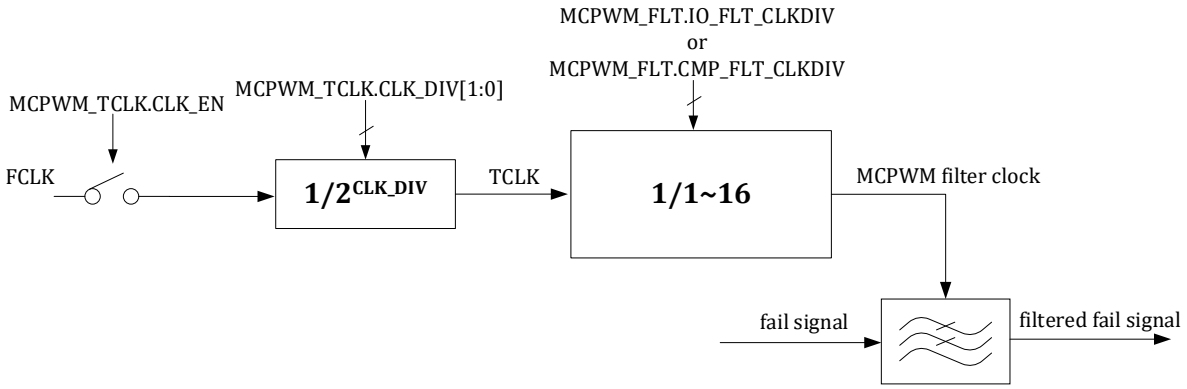


图 16-8 MCPWM Fail 信号滤波时钟生成逻辑

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

16.1.3 MCPWM 特殊输出状态

电机控制中经常会用到全零和全 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

16.1.4 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：

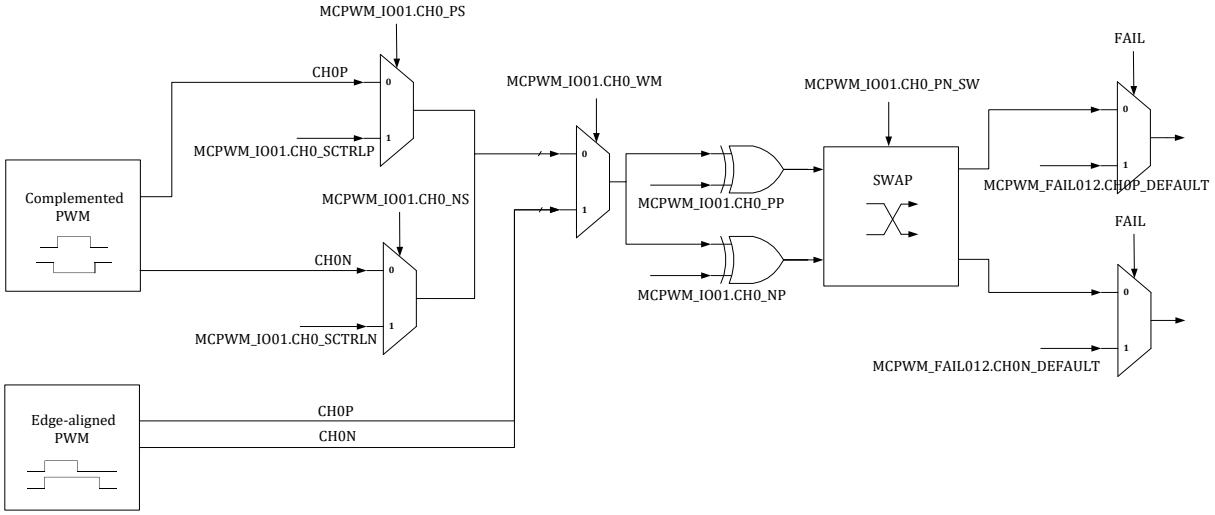


图 16-9 IO Driver 模块数据流程图

16.1.4.1 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 通道 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 DTH<n>0，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 DTH<n>1，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。

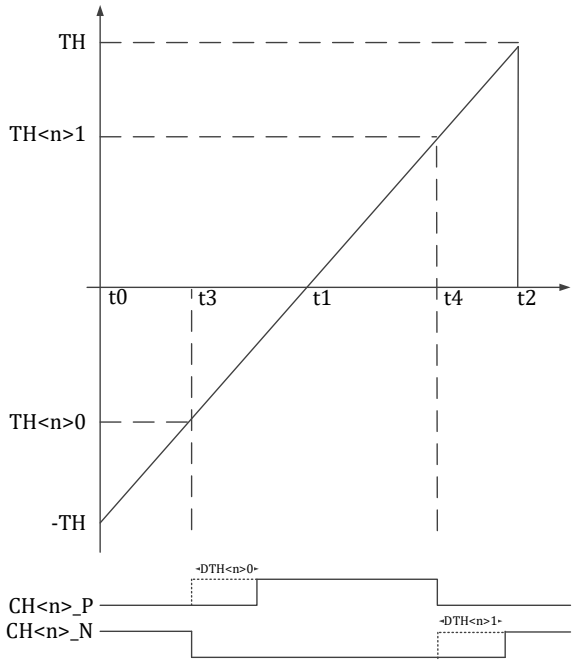


图 16-10 MCPWM 时序 TH<n>0 和 TH<n>1-中心对齐模式

16.1.4.2 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。

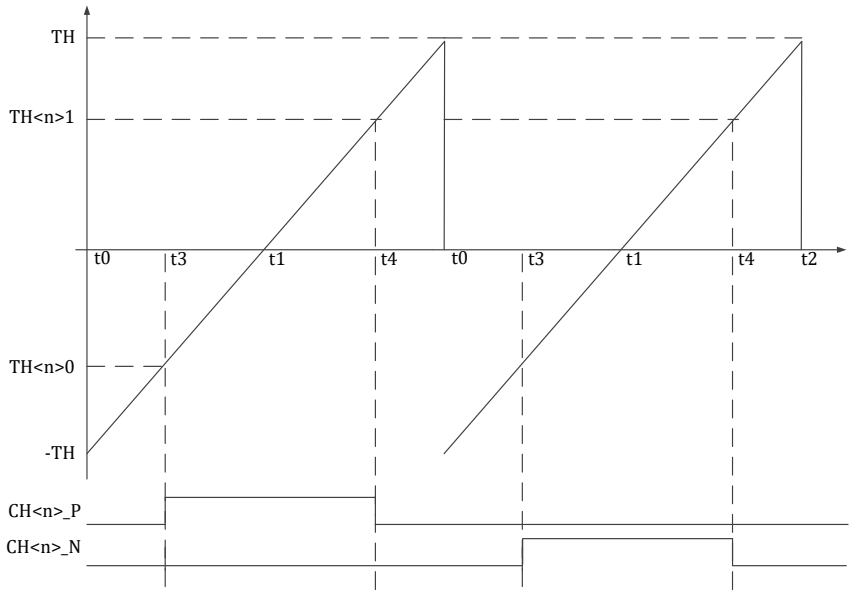


图 16-11 MCPWM 时序 TH<n>0 和 TH<n>1-中心对齐推挽模式

16.1.4.3 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。

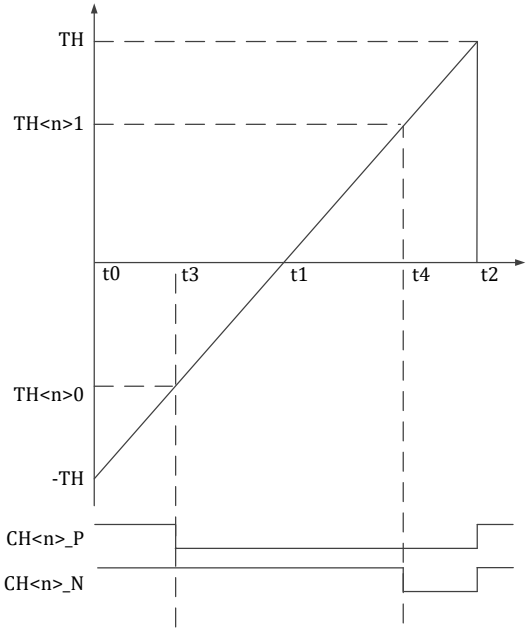


图 16-12 MCPWM 时序边沿对齐模式

16.1.4.4 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。

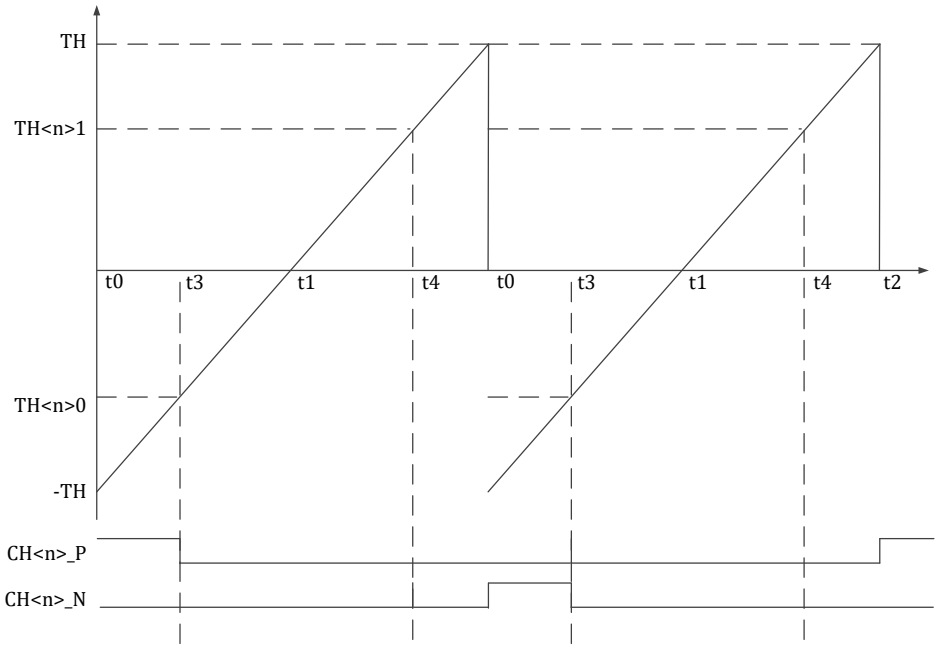


图 16-13MCPWM 时序 TH<n>0 和 TH<n>1 边沿对齐推挽模式

16.1.4.5 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 的死区宽度可独立调整。

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。

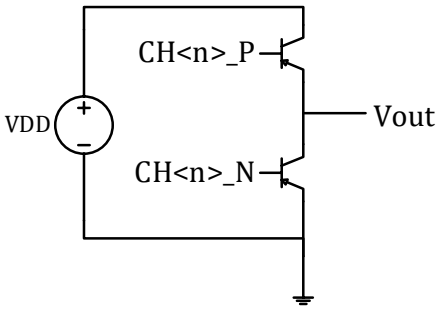


图 16-14MCPWM IO 控制示意图

16.1.4.6 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。

16.1.4.7 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 自动切换到关闭状态。需要注意关闭电平配置（MCPWM_FAIL.CHxN_DEFAULT 和 MCPWM_FAIL.CHxP_DEFAULT 控制默认电平）。其中通道 0/1/2 的缺省电平由 MCPWM_FAIL012 设置，通道 3 的缺省电平由 MCPWM_FAIL3 设置。

- 芯片正常工作后，IO 默认输出的电平是寄存器 MCPWM_FAIL.CHxN_DEFAULT 和 MCPWM_FAIL.CHxP_DEFAULT 指定值，当用户配置完毕，MCPWM 正常工作后，配置 MCPWM_FAIL.MCPWM_OE（即 MOE）为 1，IO 输出电平受到 MCPWM IO 驱动模块控制。
- 当发生 FAIL 短路状况时，硬件立即切换到 IO 缺省电平。
- 当芯片调试中,发生 MCU Halt 时,可配置实现停止 MCPWM 正常输出,转而输出缺省电平。

16.1.5 ADC Trigger Timer 模块

MCPWM 可以为 ADC 提供采样时序控制。当 MCPWM_CNT 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 16.2.33 MCPWM_TCLK。

表 16-2 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

16.1.6 中断

MCPMW_IF0 和 MCPWM_EIF[5:4]标志置位且使能的情况下，产生 MCPWMx_IRQ0 中断；

MCPMW_IF1 和 MCPWM_EIF[7:6]标志置位且使能的情况下，产生 MCPWMx_IRQ1 中断。

详见 16.1.6。

16.2 寄存器

16.2.1 地址分配

MCPWM0 模块寄存器的基地址是 0x4001_3400，

MCPWM1 模块寄存器的基地址是 0x4001_3800，

MCPWM0 和 MCPWM1 的实现完全相同，FAIL 信号来源除外。

寄存器列表如下：

表 16-3 MCPWM 模块寄存器列表

名称	偏移地址	说明
MCPWMx_TH00	0x00	MCPWM CH0_P 比较门限值寄存器
MCPWMx_TH01	0x04	MCPWM CH0_N 比较门限值寄存器
MCPWMx_TH10	0x08	MCPWM CH1_P 比较门限值寄存器
MCPWMx_TH11	0x0C	MCPWM CH1_N 比较门限值寄存器
MCPWMx_TH20	0x10	MCPWM CH2_P 比较门限值寄存器
MCPWMx_TH21	0x14	MCPWM CH2_N 比较门限值寄存器
MCPWMx_TH30	0x18	MCPWM CH3_P 比较门限值寄存器
MCPWMx_TH31	0x1C	MCPWM CH3_N 比较门限值寄存器
MCPWMx_TMR0	0x20	ADC 采样定时器比较门限 0 寄存器
MCPWMx_TMR1	0x24	ADC 采样定时器比较门限 1 寄存器
MCPWMx_TMR2	0x28	ADC 采样定时器比较门限 2 寄存器
MCPWMx_TMR3	0x2C	ADC 采样定时器比较门限 3 寄存器
MCPWMx_TH0	0x30	MCPWM 时基 0 门限值寄存器
MCPWMx_TH1	0x34	MCPWM 时基 1 门限值寄存器
MCPWMx_CNT0	0x38	MCPWM 时基 0 计数器寄存器
MCPWMx_CNT1	0x3C	MCPWM 时基 1 计数器寄存器
MCPWMx_UPDATE	0x40	MCPWM 加载控制寄存器
MCPWMx_FCNT	0x44	MCPWM FAIL 时刻 CNT 值
MCPWMx_EVT0	0x48	MCPWM 时基 0 外部触发
MCPWMx_EVT1	0x4C	MCPWM 时基 1 外部触发
MCPWMx_DTH00	0x50	MCPWM CH0 P 通道死区宽度控制寄存器

MCPWMx_DTH01	0x54	MCPWM CH0 N 通道死区宽度控制寄存器
MCPWMx_DTH10	0x58	MCPWM CH1 P 通道死区宽度控制寄存器
MCPWMx_DTH11	0x5C	MCPWM CH1 N 通道死区宽度控制寄存器
MCPWMx_DTH20	0x60	MCPWM CH2 P 通道死区宽度控制寄存器
MCPWMx_DTH21	0x64	MCPWM CH2 N 通道死区宽度控制寄存器
MCPWMx_DTH30	0x68	MCPWM CH3 P 通道死区宽度控制寄存器
MCPWMx_DTH31	0x6C	MCPWM CH3 N 通道死区宽度控制寄存器
MCPWMx_FLT	0x70	MCPWM 滤波时钟分频寄存器
MCPWMx_SDCFG	0x74	MCPWM 加载配置寄存器
MCPWMx_AUEN	0x78	MCPWM 自动更新使能寄存器
MCPWMx_TCLK	0x7C	MCPWM 时钟分频控制寄存器
MCPWMx_IE0	0x80	MCPWM 时基 0 中断控制寄存器
MCPWMx_IF0	0x84	MCPWM 时基 0 中断标志位寄存器
MCPWMx_IE1	0x88	MCPWM 时基 1 中断控制寄存器
MCPWMx_IF1	0x8C	MCPWM 时基 1 中断标志位寄存器
MCPWMx_EIE	0x90	MCPWM 异常中断控制寄存器
MCPWMx_EIF	0x94	MCPWM 异常中断标志位寄存器
MCPWMx_RE	0x98	MCPWM DMA 请求控制寄存器
MCPWMx_PP	0x9C	MCPWM 推挽模式使能寄存器
MCPWMx_IO01	0xA0	MCPWM CH0 CH1 IO 控制寄存器
MCPWMx_IO23	0xA4	MCPWM CH2 CH3 IO 控制寄存器
MCPWMx_FAIL012	0xA8	MCPWM CH0 CH1 CH2 短路控制寄存器
MCPWMx_FAIL3	0xAC	MCPWM CH3 短路控制寄存器
MCPWMx_PRT	0xB0	MCPWM 保护寄存器
MCPWMx_CHMSK	0xB8	MCPWM 通道屏蔽位寄存器

表 16-4 受 MCPWM_PRT 保护的寄存器

名称	偏移地址	说明
MCPWMx_TH0	0x30	MCPWM 门限值寄存器
MCPWMx_TH1	0x34	MCPWM 门限值寄存器
MCPWMx_DTH00	0x50	MCPWM CH0 P 通道死区宽度控制寄存器
MCPWMx_DTH01	0x54	MCPWM CH0 N 通道死区宽度控制寄存器
MCPWMx_DTH10	0x58	MCPWM CH1 P 通道死区宽度控制寄存器
MCPWMx_DTH11	0x5C	MCPWM CH1 N 通道死区宽度控制寄存器
MCPWMx_DTH20	0x60	MCPWM CH2 P 通道死区宽度控制寄存器
MCPWMx_DTH21	0x64	MCPWM CH2 N 通道死区宽度控制寄存器
MCPWMx_DTH30	0x68	MCPWM CH3 P 通道死区宽度控制寄存器
MCPWMx_DTH31	0x6C	MCPWM CH3 N 通道死区宽度控制寄存器
MCPWMx_FLT	0x70	MCPWM 滤波时钟分频寄存器
MCPWMx_SDCFG	0x74	MCPWM 加载配置寄存器
MCPWMx_AUEN	0x78	MCPWM 自动加载使能寄存器
MCPWMx_TCLK	0x7C	MCPWM 时钟分频控制寄存器

MCPWMx_IE0	0x80	MCPWM 时基 0 中断控制寄存器
MCPWMx_IE1	0x88	MCPWM 时基 1 中断控制寄存器
MCPWMx_EIE	0x90	MCPWM 异常中断控制寄存器
MCPWMx_RE	0x98	MCPWMDMA 请求控制寄存器
MCPWMx_PP	0x9C	MCPWM 推挽模式使能寄存器
MCPWMx_IO01	0xA0	MCPWM CH0 CH1 IO 控制寄存器
MCPWMx_IO23	0xA4	MCPWM CH2 CH3 IO 控制寄存器
MCPWMx_FAIL012	0xA8	MCPWM CH0 CH1 CH2 短路控制寄存器
MCPWMx_FAIL3	0xAC	MCPWM CH3 短路控制寄存器
MCPWMx_CHMSK	0xB8	MCPWM 通道屏蔽位寄存器

表 16-5 存在影子寄存器的寄存器

名称	偏移地址	说明
MCPWMx_TH00	0x00	MCPWM CH0_P 比较门限值寄存器
MCPWMx_TH01	0x04	MCPWM CH0_N 比较门限值寄存器
MCPWMx_TH10	0x08	MCPWM CH1_P 比较门限值寄存器
MCPWMx_TH11	0x0C	MCPWM CH1_N 比较门限值寄存器
MCPWMx_TH20	0x10	MCPWM CH2_P 比较门限值寄存器
MCPWMx_TH21	0x14	MCPWM CH2_N 比较门限值寄存器
MCPWMx_TH30	0x18	MCPWM CH3_P 比较门限值寄存器
MCPWMx_TH31	0x1C	MCPWM CH3_N 比较门限值寄存器
MCPWMx_TMR0	0x20	ADC 采样定时器比较门限 0 寄存器
MCPWMx_TMR1	0x24	ADC 采样定时器比较门限 1 寄存器
MCPWMx_TMR2	0x28	ADC 采样定时器比较门限 2 寄存器
MCPWMx_TMR3	0x2C	ADC 采样定时器比较门限 3 寄存器
MCPWMx_TH0	0x30	MCPWM 时基 0 门限值寄存器
MCPWMx_TH1	0x34	MCPWM 时基 1 门限值寄存器
MCPWMx_CNT0	0x38	MCPWM 时基 0 计数器寄存器
MCPWMx_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，无论装载前后，读出时总是读出的是影子寄存器的值。

16.2.2 MCPWMx_TH00(x = 0,1)

无写保护的寄存器

地址分别是：0x4001_3400，0x4001_3800

复位值：0x0

表 16-6 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。

16.2.3 MCPWMx_TH01(x = 0,1)

无写保护的寄存器

地址分别是：0x4001_3404，0x4001_3804

复位值：0x0

表 16-7 MCPWM_TH01 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	TH01	MCPWM CH0_N 比较门限值，16 位有符号数；发生更新事件时，本寄存器加载到 MCPWM 实际运行系统中。

16.2.4 MCPWMx_TH10(x = 0,1)

无写保护的寄存器

地址分别是：0x4001_3408，0x4001_3808

复位值：0x0

表 16-8 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。

16.2.5 MCPWMx_TH11(x = 0,1)

无写保护的寄存器

地址分别是：0x4001_340C，0x4001_380C

复位值：0x0

表 16-9 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。

16.2.6 MCPWMx_TH20(x = 0,1)

无写保护的寄存器

地址分别是：0x4001_3410，0x4001_3810

复位值：0x0

表 16-10 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。

16.2.7 MCPWMx_TH21(x = 0,1)

无写保护的寄存器

地址分别是：0x4001_3414，0x4001_3814

复位值：0x0

表 16-11 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。

16.2.8 MCPWMx_TH30(x = 0,1)

无写保护的寄存器

地址分别是：0x4001_3418，0x4001_3818

复位值：0x0

表 16-12 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。

16.2.9 MCPWMx_TH31(x = 0,1)

无写保护的寄存器

地址分别是：0x4001_341C，0x4001_381C

复位值：0x0

表 16-13 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。

16.2.10 MCPWMx_TMR0(x = 0,1)

无写保护的寄存器

地址分别是：0x4001_3420，0x4001_3820

复位值：0x0

表 16-14 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。

16.2.11 MCPWMx_TMR1(x = 0,1)

无写保护的寄存器

地址分别是：0x4001_3424，0x4001_3824

复位值：0x0

表 16-15 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。

16.2.12 MCPWMx_TMR2(x = 0,1)

无写保护的寄存器

地址分别是：0x4001_3428，0x4001_3828

复位值：0x0

表 16-16 MCPWM_TMR2 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	TMR2	ADC 采样定时器比较门限 2 寄存器，16 位有符号数；发生更新事件后，本寄存器加载到 MCPWM 实际运行系统中。

16.2.13 MCPWMx_TMR3(x = 0,1)

无写保护的寄存器

地址分别是：0x4001_342C，0x4001_382C

复位值：0x0

表 16-17 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。。

16.2.14 MCPWMx_TH0(x = 0,1)

写保护的寄存器

地址分别是：0x4001_3430，0x4001_3830

复位值：0x0

表 16-18 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。

16.2.15 MCPWMx_TH1(x = 0,1)

写保护的寄存器

地址分别是：0x4001_3434，0x4001_3834

复位值：0x0

表 16-19 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。。

16.2.16 MCPWMx_CNT0(x = 0,1)

无写保护的寄存器

地址分别是：0x4001_3438，0x4001_3838

复位值：0x0

表 16-20 MCPWM_CNT0 寄存器

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	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

16.2.17 MCPWMx_CNT1(x = 0,1)

无写保护的寄存器

地址分别是：0x4001_343C，0x4001_383C

复位值：0x0

表 16-21 MCPWM_CNT1 寄存器

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	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

16.2.18 MCPWMx_UPDATE(x = 0,1)

无写保护的寄存器

地址分别是：0x4001_3440，0x4001_3840

复位值：0x0

表 16-22 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
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_UPDATE	手动将加载 MCPWM_CNT1 寄存器的内容到 MCPWM 运行系统中。 1：加载；0：不加载。
[14]	CNT0_UPDATE	手动将加载 MCPWM_CNT0 寄存器的内容到 MCPWM 运行系统中。 1：加载；0：不加载。
[13]	TH1_UPDATE	手动将加载 MCPWM_TH1 寄存器的内容到 MCPWM 运行系统中。 1：加载；0：不加载。
[12]	TH0_UPDATE	手动将加载 MCPWM_TH0 寄存器的内容到 MCPWM 运行系统中。 1：加载；0：不加载。
[11]	TMR3_UPDATE	手动将加载 MCPWM_TMR3 寄存器的内容到 MCPWM 运行系统中。 1：加载；0：不加载。
[10]	TMR2_UPDATE	手动将加载 MCPWM_TMR2 寄存器的内容到 MCPWM 运行系统中。 1：加载；0：不加载。
[9]	TMR1_UPDATE	手动将加载 MCPWM_TMR1 寄存器的内容到 MCPWM 运行系统中。 1：加载；0：不加载。
[8]	TMR0_UPDATE	手动将加载 MCPWM_TMR0 寄存器的内容到 MCPWM 运行系统中。 1：加载；0：不加载。
[7]	TH31_UPDATE	手动将加载 MCPWM_TH31 寄存器的内容到 MCPWM 运行系统中。 1：加载；0：不加载。
[6]	TH30_UPDATE	手动将加载 MCPWM_TH30 寄存器的内容到 MCPWM 运行系统中。 1：加载；0：不加载。
[5]	TH21_UPDATE	手动将加载 MCPWM_TH21 寄存器的内容到 MCPWM 运行系统中。 1：加载；0：不加载。
[4]	TH20_UPDATE	手动将加载 MCPWM_TH20 寄存器的内容到 MCPWM 运行系统中。 1：加载；0：不加载。
[3]	TH11_UPDATE	手动将加载 MCPWM_TH11 寄存器的内容到 MCPWM 运行系统中。

		1：加载；0：不加载。
[2]	TH10_UPDATE	手动将加载 MCPWM_TH10 寄存器的内容到 MCPWM 运行系统中。 1：加载；0：不加载。
[1]	TH01_UPDATE	手动将加载 MCPWM_TH01 寄存器的内容到 MCPWM 运行系统中。 1：加载；0：不加载。
[0]	TH00_UPDATE	手动将加载 MCPWM_TH00 寄存器的内容到 MCPWM 运行系统中。 1：加载；0：不加载。

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。每写 1 一次，进行一次软件/手动触发。向 MCPWM_UPDATE[15:14] 写 1 会导致 MCPWM_CNT1/MCPWM_CNT0 更新为预装载值，使用请留意，应用上是否需要更新 CNT 值。

写 0 无左右，但如果写 1 后立即写 0 可能导致影子寄存器更新不成功。不推荐对 MCPWM_UPDATE 寄存器写 0。

16.2.19 MCPWMx_FCNT(x = 0,1)

无写保护的寄存器

地址分别是：0x4001_3444，0x4001_3844

复位值：0x0

表 16-23 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

16.2.20 MCPWMx_EVT0(x = 0,1)

无写保护的寄存器

地址分别是：0x4001_3448，0x4001_3848

复位值：0x0

表 16-24 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

TIMER3_CMP1	TIMER3_CMP0	TIMER2_CMP1	TIMER2_CMP0	TIMER1_CMP1	TIMER1_CMP0	TIMER0_CMP1	TIMER0_CMP0	MCPWM1_TMR3	MCPWM1_TMR2	MCPWM1_TMR1	MCPWM1_TMR0	MCPWM0_TMR3	MCPWM0_TMR2	MCPWM0_TMR1	MCPWM0_TMR0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:16]		未使用
[15]	TIMER3_CMP1	TIMER3 CMP1 事件触发时基 0 开始计数
[14]	TIMER3_CMP0	TIMER3 CMP0 事件触发时基 0 开始计数
[13]	TIMER2_CMP1	TIMER2 CMP1 事件触发时基 0 开始计数
[12]	TIMER2_CMP0	TIMER2 CMP0 事件触发时基 0 开始计数
[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]	MCPWM1_TMR3	MCPWM1 TMR3 事件触发时基 0 开始计数
[6]	MCPWM1_TMR2	MCPWM1 TMR2 事件触发时基 0 开始计数
[5]	MCPWM1_TMR1	MCPWM1 TMR1 事件触发时基 0 开始计数
[4]	MCPWM1_TMR0	MCPWM1 TMR0 事件触发时基 0 开始计数
[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

不支持 Timer4 的比较事件作为 MCPWM 时基 0 的外部触发源。

16.2.21 MCPWMx_EVT1(x = 0,1)

无写保护的寄存器

地址分别是：0x4001_344C，0x4001_384C

复位值：0x0

表 16-25 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
TIMER3_CMP1	TIMER3_CMP0	TIMER2_CMP1	TIMER2_CMP0	TIMER1_CMP1	TIMER1_CMP0	TIMER0_CMP1	TIMER0_CMP0	MCPWM1_TMR3	MCPWM1_TMR2	MCPWM1_TMR1	MCPWM1_TMR0	MCPWM0_TMR3	MCPWM0_TMR2	MCPWM0_TMR1	MCPWM0_TMR0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:16]		未使用
[15]	TIMER3_CMP1	TIMER3 CMP1 事件触发时基 1 开始计数
[14]	TIMER3_CMP0	TIMER3 CMP0 事件触发时基 1 开始计数
[13]	TIMER2_CMP1	TIMER2 CMP1 事件触发时基 1 开始计数
[12]	TIMER2_CMP0	TIMER2 CMP0 事件触发时基 1 开始计数
[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]	MCPWM1_TMR3	MCPWM1 TMR3 事件触发时基 1 开始计数
[6]	MCPWM1_TMR2	MCPWM1 TMR2 事件触发时基 1 开始计数
[5]	MCPWM1_TMR1	MCPWM1 TMR1 事件触发时基 1 开始计数
[4]	MCPWM1_TMR0	MCPWM1 TMR0 事件触发时基 1 开始计数
[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

不支持 Timer4 的比较事件作为 MCPWM 时基 1 的外部触发源。

16.2.22 MCPWMx_DTH00(x = 0,1)

写保护的寄存器

地址分别是：0x4001_3450，0x4001_3850

复位值：0x0

表 16-26 MCPWM_DTH00 配置寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTH00															
RW															
0															

位置	位名称	说明
[31:16]		未使用
[15:0]	DTH00	MCPWM CH0 P 通道死区宽度控制寄存器，10bit 无符号数

16.2.23 MCPWMx_DTH01(x = 0,1)

写保护的寄存器

地址分别是：0x4001_3454，0x4001_3854

复位值：0x0

表 16-27 MCPWM_DTH01 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	DTH01	MCPWM CH0 N 通道死区宽度控制寄存器，10bit 无符号数

16.2.24 MCPWMx_DTH10(x = 0,1)

写保护的寄存器

地址分别是：0x4001_3458，0x4001_3858

复位值：0x0

表 16-28 MCPWM_DTH10 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	DTH10	MCPWM CH1 P 通道死区宽度控制寄存器，10bit 无符号数

16.2.25 MCPWMx_DTH11(x = 0,1)

写保护的寄存器

地址分别是：0x4001_345C，0x4001_385C

复位值：0x0

表 16-29 MCPWM_DTH11 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	DTH11	MCPWM CH1 N 通道死区宽度控制寄存器，10bit 无符号数

16.2.26 MCPWMx_DTH20(x = 0,1)

写保护的寄存器

地址分别是：0x4001_3460，0x4001_3860

复位值：0x0

表 16-30 MCPWM_DTH20 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	DTH20	MCPWM CH2 P 通道死区宽度控制寄存器，10bit 无符号数

16.2.27 MCPWMx_DTH21(x = 0,1)

写保护的寄存器

地址分别是：0x4001_3464，0x4001_3864

复位值：0x0

表 16-31 MCPWM_DTH21 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	DTH21	MCPWM CH2 N 通道死区宽度控制寄存器，10bit 无符号数

16.2.28 MCPWMx_DTH30(x = 0,1)

写保护的寄存器

地址分别是：0x4001_3468，0x4001_3868

复位值：0x0

表 16-32 MCPWM_DTH30 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	DTH30	MCPWM CH3 P 通道死区宽度控制寄存器，10bit 无符号数

16.2.29 MCPWMx_DTH31(x = 0,1)

写保护的寄存器

地址分别是：0x4001_346C，0x4001_386C

复位值：0x0

表 16-33 MCPWM_DTH31 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	DTH31	MCPWM CH3 N 通道死区宽度控制寄存器，10bit 无符号数

16.2.30 MCPWMx_FLT(x = 0,1)

写保护的寄存器

地址分别是：0x4001_3470，0x4001_3870

复位值：0x0

表 16-34 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[3:0]。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[3:0]。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 FCLK 时钟宽度。FCLK 即系统高速时钟，通常为 192MHz PLL 时钟。

16.2.31 MCPWMx_SDCFG(x = 0,1)

写保护的寄存器

地址分别是：0x4001_3474，0x4001_3874

复位值：0x0

表 16-35 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+1 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。

16.2.32 MCPWMx_AUEN(x = 0,1)

写保护的寄存器

地址分别是：0x4001_3478，0x4001_3878

复位值：0x0

表 16-36 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。

16.2.33 MCPWMx_TCLK(x = 0,1)

写保护的寄存器

地址分别是：0x4001_347C，0x4001_387C

复位值：0x0

表 16-37 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。

16.2.34 MCPWMx_IE0(x = 0,1)

写保护的寄存器

地址分别是：0x4001_3480，0x4001_3880

复位值：0x0

表 16-38 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 等寄存器更新事件中断源使能。 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。

16.2.35 MCPWMx_IF0(x = 0,1)

无写保护的寄存器

地址分别是：0x4001_3484，0x4001_3884

复位值：0x0

表 16-39 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，MCPWM_CNT0 回 0 中断源事件。 1，发生；0，没发生。写 1 清零。
[0]	T0_IF	t0 事件，MCPWM_CNT0 回到 MCPWM_TH 中断源事件。 1，发生；0，没发生。写 1 清零。

16.2.36 MCPWMx_IE1(x = 0,1)

写保护的寄存器

地址分别是：0x4001_3488，0x4001_3888

复位值：0x0

表 16-40 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。

16.2.37 MCPWMx_IF1(x = 0,1)

无写保护的寄存器

地址分别是：0x4001_348C，0x4001_388C

复位值：0x0

表 16-41 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 1 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，MCPWM_CNT1 的计数值过 0 中断源事件。 1，发生；0，没发生。写 1 清零。
[0]	T0_IF	T0 事件，MCPWM_CNT1 的计数值回到 MCPWM_TH 中断源事件。 1，发生；0，没发生。写 1 清零。

16.2.38 MCPWMx_EIE(x = 0,1)

写保护的寄存器

地址分别是：0x4001_3490，0x4001_3890

复位值：0x0

表 16-42 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

16.2.39 MCPWMx_RE(x = 0,1)

写保护的寄存器

地址分别是：0x4001_3498，0x4001_3898

复位值：0x0

表 16-43 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。

16.2.40 MCPWMx_EIF(x = 0,1)

无写保护的寄存器

地址分别是：0x4001_3494，0x4001_3894

复位值：0x0

表 16-44 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

16.2.41 MCPWMx_PP(x = 0,1)

写保护的寄存器

地址分别是：0x4001_349C，0x4001_389C

复位值：0x0

表 16-45 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。

16.2.42 MCPWMx_IO01(x = 0,1)

写保护的寄存器

地址分别是：0x4001_34A0，0x4001_38A0

复位值：0x0

表 16-46 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：不互换；0：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：不互换；0：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。**

16.2.43 MCPWMx_IO23(x = 0,1)

写保护的寄存器

地址分别是：0x4001_34A4，0x4001_38A4

复位值：0x0

表 16-47 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：不互换；0：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：不互换；0：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。**

16.2.44 MCPWMx_FAIL012(x = 0,1)

写保护的寄存器

地址分别是：0x4001_34A8，0x4001_38A8

复位值：0x0

表 16-48 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		CH2N_DEFAULT	CH2P_DEFAULT	CH1N_DEFAULT	CH1P_DEFAULT	CH0N_DEFAULT	CH0P_DEFAULT	HALT_PRT	MCPWM_OE	FAIL1_EN	FAIL0_EN	FAIL1_POL	FAIL0_POL	FAIL1_SEL	FAIL0_SEL
RW		RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0		0	0	0	0	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:16]		未使用
[15]	FAIL_0CAP	当发生 fail0/1 事件时，记录 MCPWM_CNT0 值，存入 MCPWM_FCNT
[14]		未使用
[13]	CH2N_DEFAULT	CH2 N 通道默认值
[12]	CH2P_DEFAULT	CH2 P 通道默认值
[11]	CH1N_DEFAULT	CH1 N 通道默认值
[10]	CH1P_DEFAULT	CH1 P 通道默认值
[9]	CH0N_DEFAULT	CH0 N 通道默认值
[8]	CH0P_DEFAULT	CH0 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_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 MCPWM0，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。

对于 MCPWM1，FAIL0 信号来源为比较器 2 或 MCPWM1_BKIN0；FAIL1 信号来源为比较器 3 或 MCPWM1_BKIN1。

FAIL0/1 信号用于工作于时基 0 的 MCPWM 通道 0/1/2。

16.2.45 MCPWMx_FAIL3(x = 0,1)

写保护的寄存器

地址分别是：0x4001_34AC，0x4001_38AC

复位值：0x0

表 16-49 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						CH3N_DEFAULT	CH3P_DEFAULT	HALT_PRT	MOE	FAIL3_EN	FAIL2_EN	FAIL3_POL	FAIL2_POL	FAIL3_SEL	FAIL2_SEL
RW						RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0						0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:16]		未使用
[15]	FAIL_1CAP	当发生 fail2/3 事件时，记录 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；0：来自 GPIO
[0]	FAIL2_SEL	FAIL2 来源选择。1：比较器；0：来自 GPIO

对于 MCPWM0，FAIL2 信号来源为比较器 4 或 MCPWM0_BKIN2；FAIL3 信号来源为比较器 5 或 MCPWM0_BKIN3。

对于 MCPWM1，FAIL2 信号来源为比较器 4 或 MCPWM1_BKIN2；FAIL3 信号来源为比较器 5 或 MCPWM1_BKIN3。

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。

FAIL2/3 信号用于工作于时基 1 的 MCPWM 通道 3。

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。

16.2.46 MCPWMx_PRT(x = 0,1)

无写保护的寄存器

地址分别是：0x4001_34B0，0x4001_38B0

复位值：0x0

表 16-50 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。

16.2.47 MCPWMx_CHMSK(x = 0,1)

写保护的寄存器

地址分别是：0x4001_34B8，0x4001_38B8

表 16-51 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

17 UART

17.1 概述

通用异步收发传输器（Universal Asynchrounous Receiver/Transmitter），通常称作 UART，是一种异步收发传输器。

UART 主要特征如下:

- 支持全双工工作
- 支持单线半双工工作
- 支持 8/9 位数据位
- 支持 1/2 停止位
- 支持奇/偶/无校验模式
- 带 1 字节发送缓存
- 带 1 字节接收缓存
- 支持 LIN 模式 break character 收发
- 支持空闲帧检测
- 支持一主多从的 Multi-drop Slave/Master 模式

17.2 功能说明

17.2.1 发送

UART 包括一个字节发送缓冲区，当发送缓冲区有数据时，UART 将发送缓冲区的数据移入串行移位寄存器，并通过 UART_TXD 端口发送出去。只要 UART 发送缓冲区有数据，UART 就会自动进行发送。

完成加载后，产生发送缓冲区空中断，此时，用户可以往发送缓冲区填入下一个需要发送的字节，这样，发送完成后，UART 将加载这个字节进行发送。

完成发送后，会产生发送完成中断。

发送流程:

设置 SYS_CLK_FEN. UART_CLK_EN = 1 开启 UART 时钟

设置 UART_CTRL.BYTE_LEN，选择 8/9bit 数据字节长度

设置 UART_CTRL.CK_EN，选择是否启用数据字节校验

设置 UART_CTRL.CK_TYPE，选择使用奇校验还是偶校验

设置 UART_CTRL.BIT_ORDER，选择发送时 LSB first 还是 MSB first

设置 UART_CTRL.STOP_LEN，选择停止比特长度为 1bit/2bit

如果使用 DMA 搬运数据，则设置 UART_RE. TX_BUF_EMPTY_RE=1，即 TX buffer 空就触发 DMA 搬运新数据到 UART；如果使用软件轮询或中断的方式，则设置 UART_IE. TX_BUF_EMPTY_IE=1

如果使用 DMA 搬运数据到 UART 发送，需要对 DMA 进行相应设置

如果使用软件搬运数据到 UART 发送，需要软件向 UART_BUFF 写入数据，可以触发 UART 开始通过 UART_TXD 端口发送数据。

17.2.2 接收

UART 包括一个字节的接收缓冲区，当完成一个字节的接收后，会产生接收中断，并将接收到字节存储到接收缓冲区，用户应当在 UART 接收完成下一个字节前完成此字节的读取，否则缓冲区会被写入新接收的字节。接收部分内部使用高速时钟对 UART_RX 信号进行过采样来判定稳态的数据信号，防止噪声干扰造成错误接收。

17.2.3 UART 帧格式

UART 信号上收发的数据格式通常如下:

信号线空闲;

起始比特 (1 bit Start bit: 1 bit Zero)

数据字节 (data word, 8bits or 9bits, LSB first or MSB first)

停止比特 (1/2 bit Stop bit: 1/2bit Ones)

数据字节的长度可以通过 UART_CTRL.BYTE_LEN 进行选择，8bit (UART_CTRL.BYTE_LEN=0)或 9bit (UART_CTRL.BYTE_LEN=1)。起始比特为 1bit 零，TX 信号线为低，停止比特期间 TX 信号线为高。

停止比特的长度由 UART_CTRL.

需要注意的是，当启用校验位时(UART_CTR.CK_EN=1)，校验位会替换掉数据字节的最高 bit，即 MSB，此时 8bit 数据字节实际是 7bit 数据+1bit 校验字，9bit 数据字节实际是 8bit 数据+1bit 校验字。

图 17-1 UART 帧格式

17.2.4 波特率配置

UART 输入时钟为系统主时钟，波特率通过两级分频实现。

波特率=UART 模块时钟/（256*DIVH+DIVL+1)

可以通过 SYS_CLK_DIV2 对 UART 模块时钟进行分频.

UART 模块时钟=系统主时钟/(1+SYS_CLK_DIV2)

表 17-1 UART 波特率配置示例

UART 波特率	SYS_CLK_DIV2	UART_DIVH	UART_DIVL
600	0x0007	0x9C	0x3F
1200	0x0003	0x9C	0x3F
2400	0x0001	0x9C	0x3F
4800	0x0000	0x9C	0x3F
9600	0x0000	0x4E	0x1F
19200	0x0000	0x27	0x0F

38400	0x0000	0x13	0x87
43000	0x0000	0x11	0x71
56000	0x0000	0x0D	0x65
57600	0x0000	0x0D	0x05
115200	0x0000	0x06	0x83

注意，同一波特率，其配置系数可能不唯一。

注意，同一波特率，其配置系数可能不唯一。

17.2.5 收发端口互换(TX/RX 互换)

UART 模块支持 Tx 与 Rx 端口互换。通过将 Tx 对应的 GPIO 配置为输入使能，Rx 对应的 GPIO 配置为输出使能，即可实现 Tx 与 Rx 端口的互换。此时，GPIO 第二功能仍选择为 UART，UART 本身配置无需修改。

此外，如果要使用一个 GPIO 同时作为 Tx 和 Rx，需要将 IO 分时复用为输入或输出，对应 Rx 或 Tx，即可实现单口半双工逻辑。

17.2.6 多机通讯

多机通讯场景通常是一个设备作为主设备 master，另有若干个从设备 slave，主设备的 UARTm_TXD 端口连接到所有从设备的 UARTs_RXD 端口，从设备的 UARTs_TXD 相与连接到主设备的 UARTm_RXD 端口。如图 17-2 所示，为一个主设备和 3 个从设备（地址分别为 0x51,0x52,0x53）的互联情况。

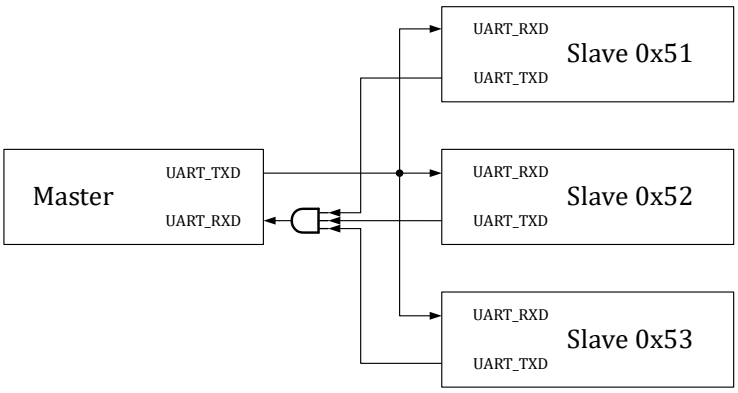


图 17-2 UART 多机通讯互联拓扑

为了降低从机报文处理的负荷，从设备应该只处理发送给他的 UART 数据，而忽略发送给其他从设备的数据。在多机通讯场景中，每一个从机有一个 8bit 的设备地址，即 UART_ADR。从设备设置 UART_CTRL.MD_EN=1 后，开启多机通讯过滤模式，主机无须设置 MD_EN。主机、从机均需要使用 9bit 模式发送数据，即设置 UART_CTRL.BYTE_LEN=1。当发送数据字节 MSB(BIT8)为 1 时，低 8 位(BIT7:BIT0)是主机发出的从机地址，即**地址字节**；当发送数据字节 MSB(BIT8)为 0 时，低 8 位(BIT7:BIT0)是主机发给特定从机的数据，即**数据字节**。

所有从机接收到 MSB(BIT8)=1 的数据字节后，判断低 8 位(BIT7:BIT0)地址是否与自己的 UART_ADR 配置值相等。如果相等，则说明后续数据都是发送给此从机，否则从机进入静默状态，

忽略所有后续主机发出的数据。当从机再次接收到 MSB(BIT8)=1 的数据字节且低 8 位(BIT7:BIT0)地址与自己的 UART_ADR 配置值相等时，表示此从机被选中，退出静默状态。从机设置 UART_CTRL.MD_EN=1 开启多机通讯过滤模式后，立即进入静默状态。如果主机没有发送 MSB=1 的地址字节，直接发送数据字节，则所有从机都处于静默状态，不会接收任何数据。

在多机通讯模式中，主设备无需设置 UART_CTRL.MD_EN=1，从设备需要设置 UART_CTRL.MD_EN=1。如果无需地址过滤，从设备也可以不设置 UART_CTRL.MD_EN=1，另使用软件对接收数据进行判读，此时从设备会接收并处理所有报文数据。

在多机通讯模式中，如果从设备设置了 UART_CTRL.MD_EN=1，则地址字节无论是否命中 UART_ADR，UART_IF.RX_DONE_IF 标志均不置位；如果地址不匹配，即从设备处于静默状态，即使主设备发送数据，UART_IF.RX_DONE_IF 仍不置位。如果地址匹配，当接收到主设备发送的数据字节时，从设备的 UART_IF.RX_DONE_IF 标志置位为 1，表示从设备接收到一个数据字节。

由于检验位需要占用 1bit 数据位，而多机通讯需要使用 9bit 字节长度。因此多机通讯场景下不支持校验位，在设置 UART_CTRL.MD_EN=1 时，请勿设置 UART_CTRL.CK_EN=1。主机、从机均需设置 UART_CTRL.BYTE_LEN=1 使用 9bit 模式，并设置 UART_CTRL.MD_EN。多机通讯支持 MSB first，即 UART_CTRL.BIT_ORDER=1。

如图 17-3 所示，主设备先发送了数据 0x03，但此时没有从设备地址命中，因此 3 个从设备均不接收 0x03。主设备发送地址字节 0x151，从设备 0x51 被选中。但之后主设备没有发送数据，而是直接发送地址字节 0x153，从设备 0x53 被选中，主设备连续发送 3 个数据字节 0x03,0x53,0x04，均被从设备 0x53 接收。之后主设备发送地址字节 0x152，之后发送数据字节 0x07，被从设备 0x52 接收。

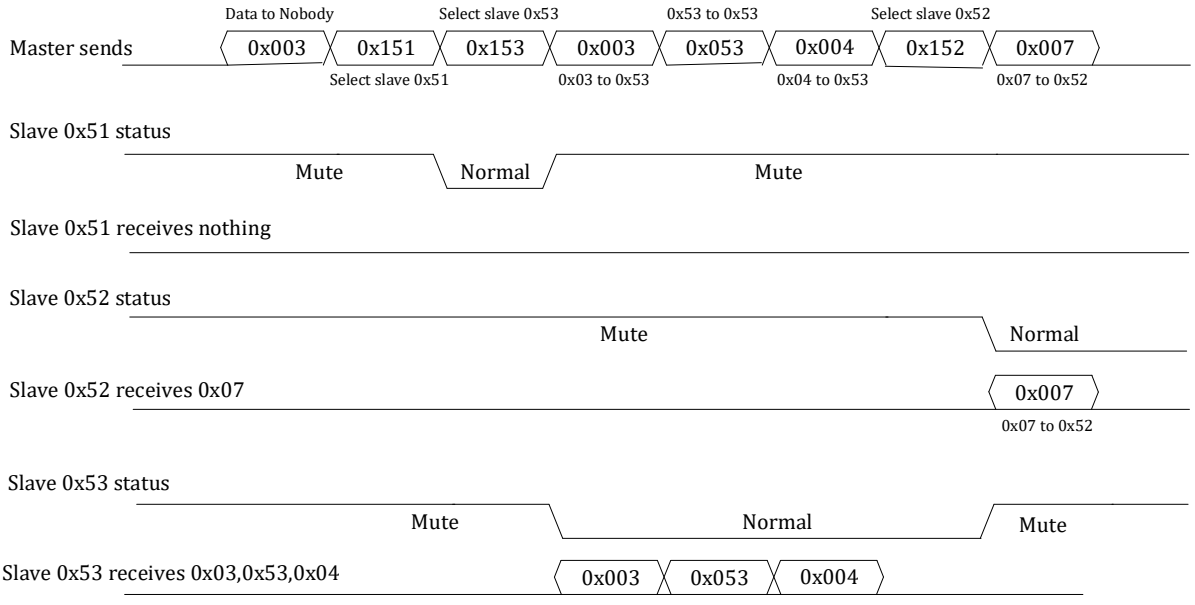


图 17-3 UART 多机通讯示例

17.2.7 校验位

可以通过设置 UART_CTRL.CK_EN=1 来使能校验位。同时根据字节长度 UART_CTRL.BYTE_LEN 不同，UART 的帧格式有 4 种情况。

表 17-2 UART 帧格式

BYTE_LEN	CK_EN	UART frame
0	0	| StartBit | 8bit data | StopBit |
0	1	| StartBit | 7bit data | Parity | StopBit |
1	0	| StartBit | 9bit data | StopBit |
1	1	| StartBit | 8bit data | Parity | StopBit |

17.3 寄存器

17.3.1 地址分配

UART0，UART1 和 UART2 实现完全相同。

UART0 基地址 0x4001_2000。

UART1 基地址 0x4001_2400。

UART2 基地址 0x4001_2800。

表 17-3 UART 地址分配列表

名称	偏移地址	说明
UARTx_CTRL	0x00	UART 控制寄存器
UARTx_DIVH	0x04	UART 波特率设置高字节寄存器
UARTx_DIVL	0x08	UART 波特率设置低字节寄存器
UARTx_BUFF	0x0C	UART 收发缓冲寄存器
UARTx_ADR	0x10	485 通信地址匹配寄存器
UARTx_STT	0x14	UART 状态寄存器
UARTx_RE	0x18	UART DMA 请求使能寄存器
UARTx_IE	0x1C	UART 中断使能寄存器
UARTx_IF	0x20	UART 中断标志寄存器
UARTx_IOC	0x24	UART IO 控制

17.3.2 UARTx_CTRL UARTx 控制寄存器 (x = 0,1,2)

地址分别是：0x4001_2000，0x4001_2400，0x4001_2800

复位值:0x0

表 17-4 UART 控制寄存器 UARTx_CTRL

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

位置	位名称	说明
[31:8]		未使用
[7]	DUPLEX	双工，默认值为 0。 0:全双工；1:半双工
[6]	LIN_EN	LIN 模式使能，默认值为 0。 0:关闭；1:开启
[5]	MD_EN	使能 Multi-drop，默认值为 0。 0:关闭；1:开启
[4]	CK_EN	数据校验开关，默认值为 0。 0:关闭；1:开启
[3]	CK_TYPE	奇偶校验配置，默认值为 0。 0:偶校验（EVEN）；1: 奇校验（ODD）
[2]	BIT_ORDER	数据发送顺序配置，默认值为 0。 0:LSB；1:MSB
[1]	STOP_LEN	停止位长度配置，默认值为 0。 0:1-Bit；1:2-Bit
[0]	BYTE_LEN	数据长度配置，默认值为 0。 0:8-Bit；1:9-Bit

17.3.3 UARTx_DIVH UARTx 波特率设置高字节寄存器 (x = 0,1,2)

地址分别是：0x4001_2004，0x4001_2404，0x4001_2804

复位值:0x0

表 17-5 UART 波特率设置高字节寄存器 UARTx_DIVH

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

位置	位名称	说明

[31:8]		未使用
[7:0]	DIVH	波特率设置高字节 BAUDRATE =主时钟/(1+256* UART_DIVH+UART_DIVL)

17.3.4 UARTx_DIVL UARTx 波特率设置低字节寄存器 (x = 0,1,2)

地址分别是：0x4001_2008，0x4001_2408，0x4001_2808

复位值:0x0

表 17-6 UART 波特率设置低字节寄存器 UART_DIVL

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

位置	位名称	说明
[31:8]		未使用
[7:0]	DIVL	波特率设置低字节 BAUDRATE =主时钟/(1+256* UART_DIVH+UART_DIVL)

17.3.5 UARTx_BUFF UARTx 收发缓冲寄存器 (x = 0,1,2)

地址分别是：0x4001_200C，0x4001_240C，0x4001_280C

表 17-7 UART 收发缓冲寄存器 UART_BUFF

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

位置	位名称	说明
[31:9]		未使用
[8:0]	BUFF	写:发送数据缓存；读:接收数据寄存器

UART 的 Tx_buffer 和 Rx_buffer 共享 UART_BUFF。其中，Tx_buffer 是只写的，Rx_buffer 是只读的。因此,读 UART_BUFF 寄存器是访问 UART_RX_BUFF,写 UART_BUFF 寄存器是访问 UART_TX_BUFF。

当 UARTx_CTRL.BYTE_LEN=0 时，UART 收发只使用 buffer 低 8 位，即 UARTx_BUFF[7:0]；

当 UARTx_CTRL.BYTE_LEN=1 时，UART 收发使用 buffer 全部 9 位，即 UARTx_BUFF[8:0]。

如果使能 UARTx_CTRL.CK_EN，即使用校验位，则 UARTx_BUFF 的最高位（BIT8 when UARTx_CTRL.BYTE_LEN=1 or BIT7 UARTx_CTRL.BYTE_LEN=0）数据无效，发送时会被替换为校验位，接收时会作为校验位，而不会写入 UARTx_BUFF。

17.3.6 UARTx_ADR UARTx 地址匹配寄存器 (x = 0,1,2)

地址分别是：0x4001_2010，0x4001_2410，0x4001_2810

复位值:0x0

表 17-8 UART 地址匹配寄存器 UART_ADR

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

位置	位名称	说明
[31:8]		未使用
[7:0]	ADR	多机通讯时的从机地址

17.3.7 UARTx_STT UARTx 状态寄存器 (x = 0,1,2)

地址分别是：0x4001_2014，0x4001_2414，0x4001_2814

复位值:0x0

表 17-9 UART 状态寄存器 UART_STT

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

位置	位名称	说明
[31:4]		未使用
[3]	RX_BUSY	1:UART 已检测到起始符并正处于接收状态 0:UART 接收端空闲 此状态位只读，置位与清零均由硬件完成
[2]	ADR_MATCH	Multi-drop 模式下，地址匹配标志位。 1:匹配；0:未匹配。
[1]	TX_DONE	发送完成标志位。

		1:完成；0:未完成。
[0]	TX_BUF_EMPTY	发送缓存状态位。 1:空；0:非空。

17.3.8 UARTx_RE UARTx DMA 请求使能寄存器 (x = 0,1,2)

地址分别是：0x4001_2018，0x4001_2418，0x4001_2818

复位值:0x0

表 17-10 UART DMA 请求使能寄存器 UART_RE

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

17.3.9 UARTx_IE UARTx 中断使能寄存器 (x = 0,1,2)

地址分别是：0x4001_201C，0x4001_241C，0x4001_281C

复位值:0x0

表 17-11 UART 中断使能寄存器 UART_IE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							IDLE_IE	LBD_IE	RX_OV_IE	TX_OV_IE	CK_ERR_IE	STOP_ERR_IE	TX_BUF_EMPTY_IE	RX_DONE_IE	TX_DONE_IE
							RW	RW	RW	RW	RW	RW	RW	RW	RW

	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:9]		未使用
[8]	IDLE_IE	空闲帧中断使能 0:关闭；1:开启
[7]	LBD_IE	LIN break character 检测中断使能 0:关闭；1:开启
[6]	RX_OV_IE	接收缓冲区溢出中断使能 0:关闭；1:开启
[5]	TX_OV_IE	发送缓冲区溢出中断使能 0:关闭；1:开启
[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

17.3.10 UARTx_IF UARTx 中断标志寄存器 (x = 0,1,2)

地址分别是：0x4001_2020，0x4001_2420，0x4001_2820

复位值:0x0

表 17-12 UART 中断使能寄存器 UART_IF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							IDLE_IF	LBD_IF	RX_OV_IF	TX_OV_IF	CK_ERR_IF	STOP_ERR_IF	TX_BUF_EMPTY_IF	RX_DONE_IF	TX_DONE_IF
							RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C
							0	0	0	0	0	0	1	0	1

位置	位名称	说明
[31:9]		未使用

[8]	IDLE_IF	空闲帧中断标志，高有效，写 1 清零 当 UART 检测到空闲帧后，硬件置位这一标志，如果 UARTx_IE.IDLE_IE=1，则产生中断请求；写 1 清零后，只有待 RX_DONE_IF 标志再次置位后，IDLE_IF 标志才会置位。在多从机（Multi-drop）场景中，只有从机地址命中时，即 UARTx_STT.ADR_MATCH=1 时，表示主机此时欲与本从机通讯，IDLE_IF 才会置位，否则即使 UARTx_RX 信号为常高，IDLE_IF 也不会置位。
[7]	LBD_IF	LIN break character 检测中断标志，高有效，写 1 清零
[6]	RX_OV_IF	接收缓冲区溢出中断标志，高有效，写 1 清零
[5]	TX_OV_IF	发送缓冲区溢出中断标志，高有效，写 1 清零
[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;
```

17.3.11 UARTx_IOC UARTx IO 控制寄存器 (x = 0,1,2)

地址分别是：0x4001_2024，0x4001_2424，0x4001_2824

复位值:0x0

表 17-13 UARTIO 控制寄存器 UART_IOC

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

位置	位名称	说明

[31:8]		未使用
[7]	SBK	LIN 模式下，写 1 发送一次 break character，即连续 13 个 0，完成后自动清零，未完成时读回为 1，向此位写 1 之前需要先配置 UARTx_CTRL.LIN_EN=1，否则无法进行 break character 的发送
[6]	LBDL	LIN break character 检测长度，10/11 个 0 0:10bits，1:11bits
[5]		未使用
[4]	AUTO	波特率自适应 IO 端口使能开关。 0:关闭；1:开启
[3:2]		未使用
[1]	TXD_INV	TXD 输出极性使能开关，默认值为 0。 0:正常输出；1:取反输出。 正常输出极性，即软件发送 1，硬件发送 1；取反输出极性，即应用发送 1，硬件发送 0。
[0]	RXD_INV	RXD 输入极性使能开关，默认值为 0。 0:正常输入；1:取反输入。 正常输入极性，即硬件接收到 1，软件接收的是 1；取反输入极性，即硬件接收 1，软件接收的是 0。

18 I2C

18.1 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。

18.2 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。

18.3 功能描述

18.3.1 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。

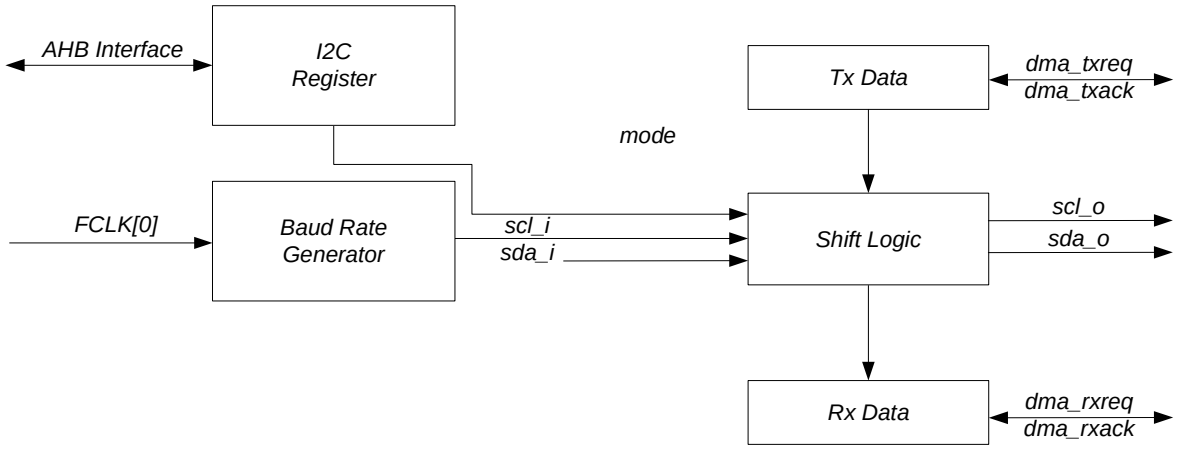


图 18-1 I2C 模块顶层功能框图

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。

18.3.2 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。

18.3.2.1 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。

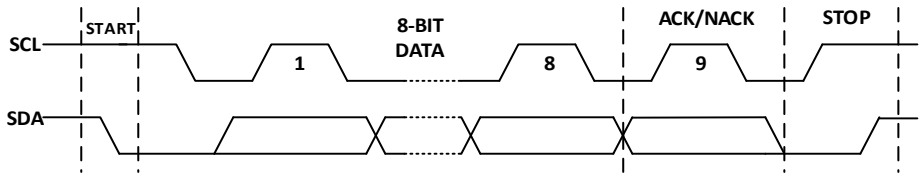


图 18-2 基本 I2C 传输时序图

一般情况下，一次传输一个字节（可反复单次传输，需软件介入提供数据）。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。
- 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。

18.3.2.2 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，必须在系统寄存器 CLK_DIV0 中设定 I2C 接口的工作时钟。

- 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。
- 从模式接收。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。
- 从模式发送。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。

18.3.2.2.1 从模式传输

每次传输完一个字节的数据后，硬件将产生中断，软件判断是否还要继续传输。图 15-3 为从模式传输的总线示意图。从图可知，流程如下：

- 地址匹配，产生地址匹配中断，准备开始传输。
- 若是从接收，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。
- 若是从发送，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。

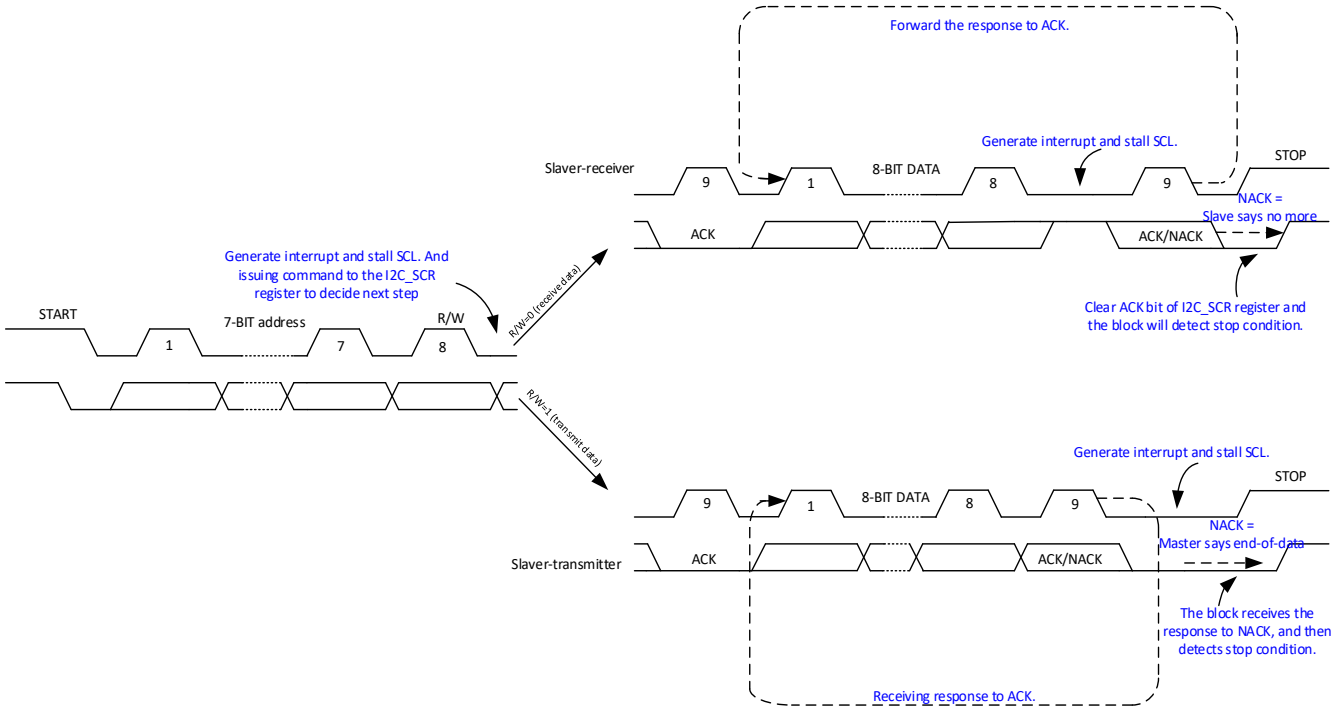


图 18-3 从模式传输示意图

18.3.2.2.2 从模式发送

地址匹配后，将数据从 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。

18.3.2.2.3 从模式单字节接收

地址匹配后，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。

18.3.2.3 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。

18.3.2.3.1 主模式传输

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。下图为主模式传输的总线示意图。从图可知，流程如下：

- 判断总线是否空闲，若空闲，准备开始传输。
- 首先，发送从地址，若地址匹配，才继续后续传输，否则停止。
- 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。

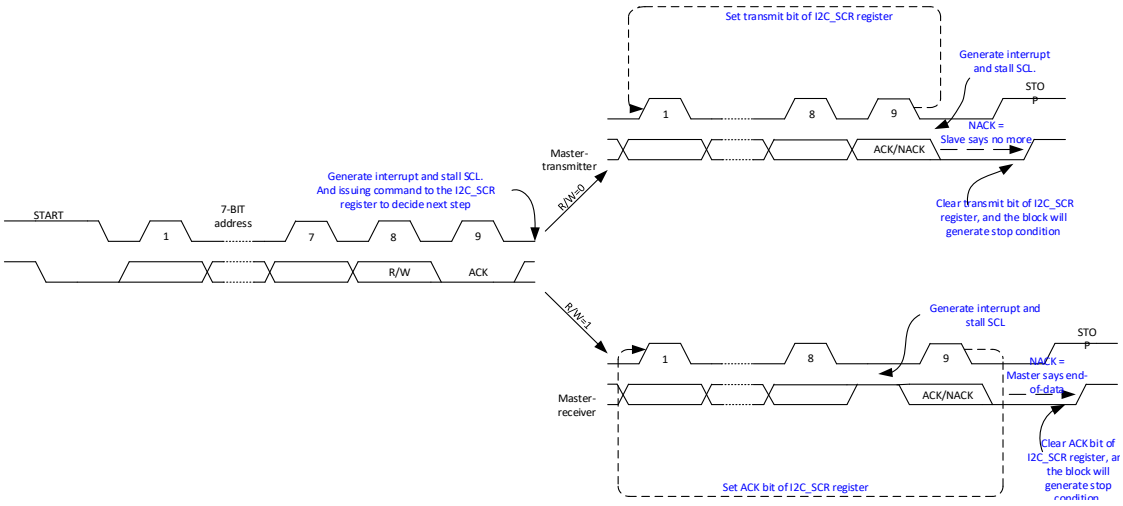


图 18-4 主模式下单字节传输示意图

18.3.2.3.2 主模式发送

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，从地址也通过 I2C_DATA 发送出去。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。
- 若主设备，完成传输。无论接收的是 ACK/NACK，都停止后续传输。
- 产生 STOP 事件，结束本次传输。

18.3.2.3.3 主模式接收

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，从地址也通过 I2C_DATA 发送出去。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。
- 若主设备，完成传输。无论发送的是 ACK/NACK，都停止后续传输。
- 产生 STOP 事件，结束本次传输。

18.3.2.4 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。

18.3.2.5 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。

18.3.2.6 通讯速度设置

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

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

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

18.4 寄存器

18.4.1 地址分配

I2Cx 模块寄存器的基地址分别是是 0x4001_1000，0x4001_1400，寄存器列表如下：

表 18-1 I2C 寄存器地址分配表

名称	偏移	说明
I2Cx_ADDR	0x00	I2C 地址寄存器
I2Cx_CFG	0x04	I2C 配置寄存器
I2Cx_SCR	0x08	I2C 状态寄存器
I2Cx_DATA	0x0C	I2C 数据寄存器
I2Cx_MSCR	0x10	I2C 主模式寄存器
I2Cx_BCR	0x14	I2C 传输控制寄存器

18.4.2 I2Cx_ADDR 地址寄存器(x = 0,1)

地址分别是：0x4001_1000，0x4001_1400

复位值：0x0

表 18-2 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。

18.4.3 I2Cx_CFG 系统控制寄存器(x = 0,1)

地址分别是：0x4001_1004，0x4001_1404

复位值：0x0

表 18-3 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

18.4.4 I2Cx_SCR 状态控制寄存器(x = 0,1)

地址分别是：0x4001_1008，0x4001_1408

复位值：0x0

表 18-4 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。

18.4.5 I2C_DATA 数据寄存器(x = 0,1)

地址分别是：0x4001_100C，0x4001_140C

复位值：0x0

表 18-5 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。

18.4.6 I2Cx_MSCR 主模式寄存器(x = 0,1)

地址分别是：0x4001_1010，0x4001_1410

复位值：0x0

表 18-6 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。

18.4.7 I2Cx_BCR I2C 传输控制寄存器(x = 0,1)

地址分别是：0x4001_1004，0x4001_1404

复位值：0x0

表 18-7DMA 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
								NACK	ADDR_CMP	BURST_EN					
								RW	RW	RW					
								0	0	0					

位置	位名称	说明
[31:8]		未使用
[7]	NACK_IE	I2C 传输，NACK 事件中断使能信号。 0：屏蔽此中断源 1：使能此中断源
[6]	ADDR_CMP_IE	I2C 传输，硬件地址匹配中断使能信号。 0：屏蔽此中断源 1：使能此中断源
[5]	BUSRT_EN	I2C 多数据传输使能，需要采用 DMA 方式。 1：使能 0：关闭
[4:0]	--	保留

19 SPI

19.1 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。

19.2 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。包含溢出检测和片选信号异常检测

19.3 功能描述

19.3.1 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。

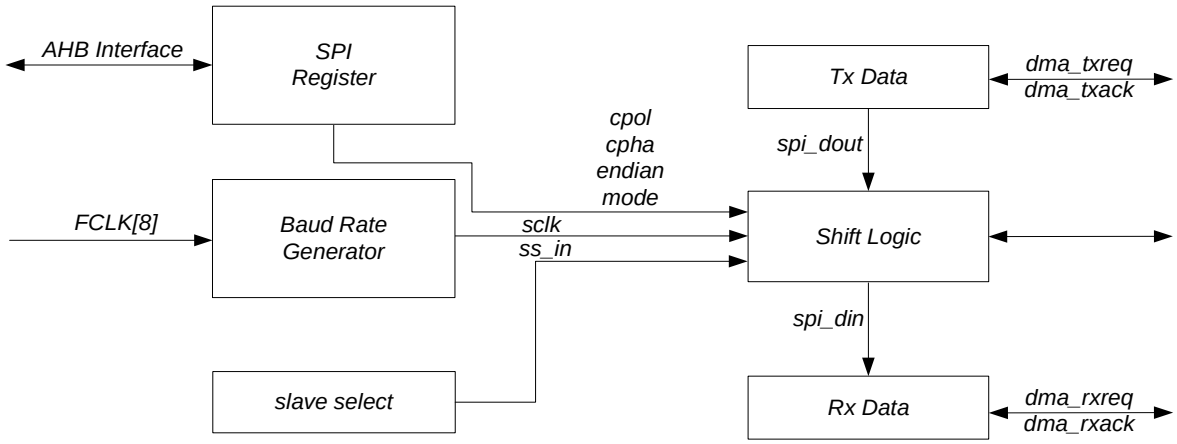


图 19-1 SPI 模块结构框图

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。

19.3.2 功能说明

19.3.2.1 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

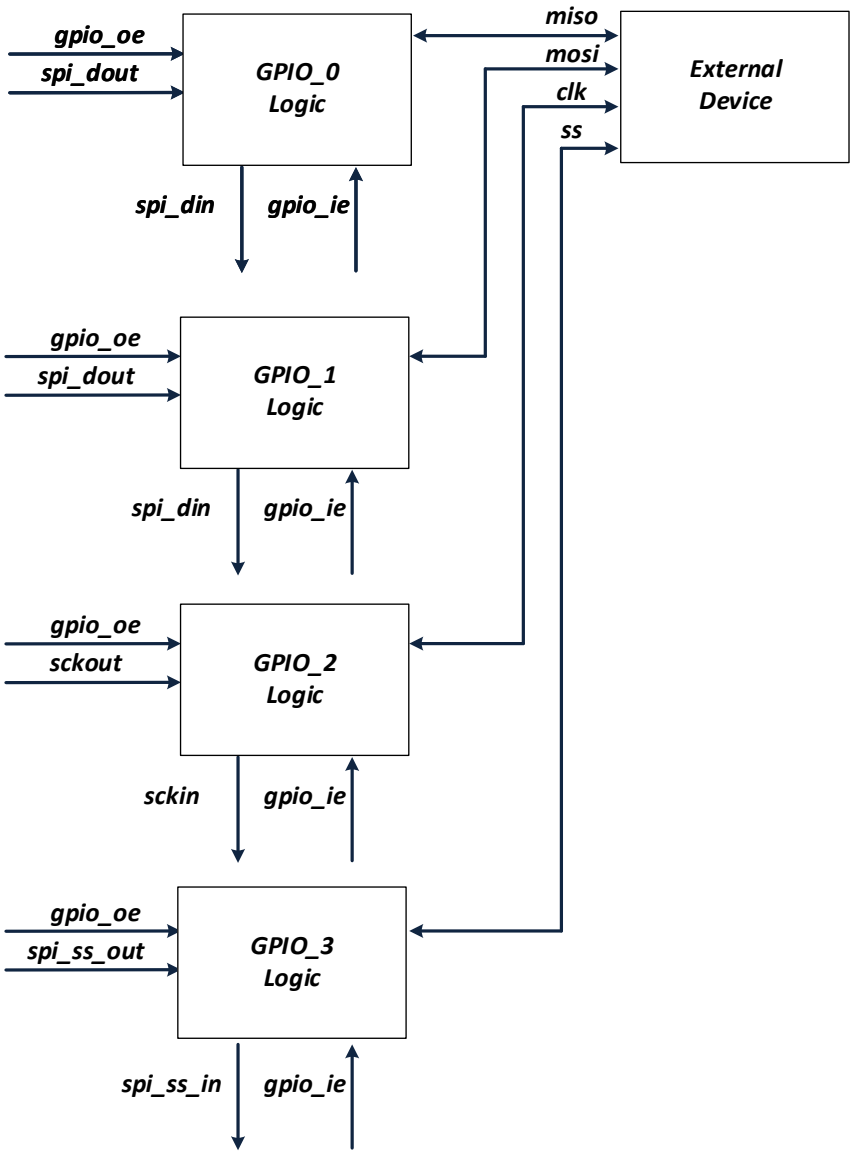


图 19-2SPI 接口全双工模式互连框图

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。

19.3.2.2 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。

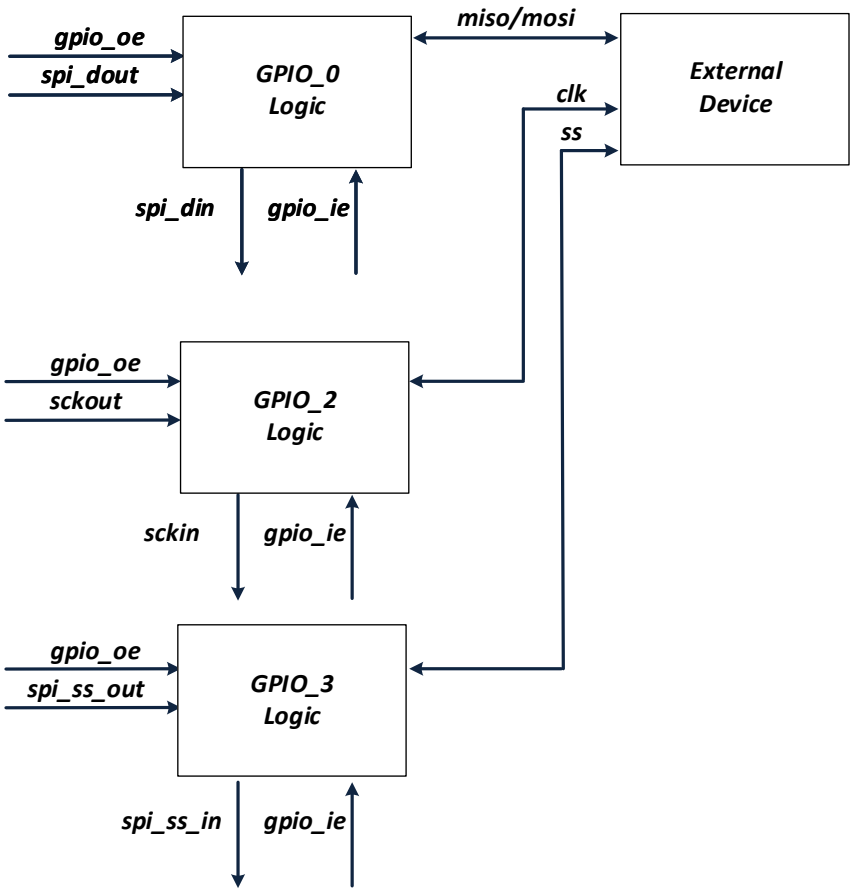


图 19-3 SPI 接口半双工模式互连框图

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。

19.3.2.3 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。

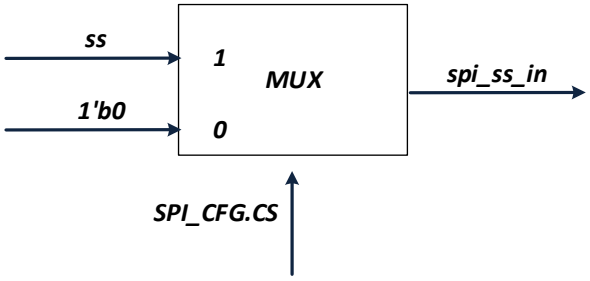


图 19-4 SPI 模块 Slave 模式片选信号选择

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。

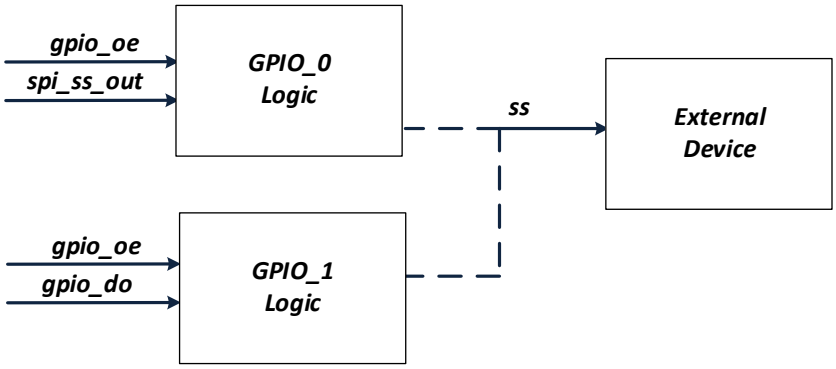


图 19-5 SPI 模块 Master 模式片选信号选择

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。

19.3.2.4 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。

19.3.2.5 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，无需软件转换。

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

19.3.2.6 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_TX_DATA 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_TX_DATA 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。

支持连续发送，由 SPI_BAUD.TRANS_MODE 控制。主要针对主模式。

非连续模式下，一次完整的输出传输为：片选信号有效，SPI_TX_DATA 写入发送值，触发一个 SPI_SIZE.BITSIZE 长度的数据传输，完毕后，片选信号失效。

连续模式下，一次完整的输出传输为：片选信号有效，SPI_TX_DATA 写入发送值，一个 SPI_SIZE.BITSIZE 长度的数据传输完毕，SPI_TX_DATA 写入新值，触发下一个 SPI_SIZE.BITSIZE 长度的数据传输，片选信号保持有效，直至应用将本批次数据发送完毕，片选信号失效。

19.3.2.7 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。

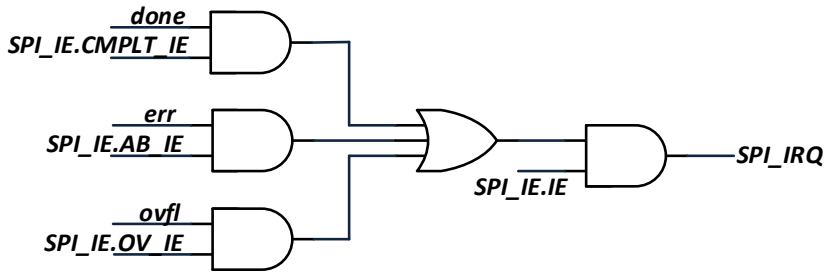


图 19-6 SPI 模块中断选信号产生图

19.3.2.8 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。

19.4 寄存器

19.4.1 地址分配

SPI0 模块寄存器的基地址是 0x4001_A800，SPI1 模块寄存器的基地址是 0x4001_AC00。列表如下。

表 19-1 SPI 模块控制寄存器列表

名称	偏移	说明
SPIx_CFG	0x00	SPI 配置寄存器
SPIx_IE	0x04	SPI 中断寄存器
SPIx_BAUD	0x08	SPI 波特率寄存器
SPIx_TXDATA	0x0C	SPI 发送数据寄存器
SPIx_RXDATA	0x10	SPI 接收数据寄存器
SPIx_BITSIZE	0x14	SPI 传输数据字节长度寄存器

19.4.2 SPIx_CFG SPI 控制寄存器

地址分别是：0x4001_A800，0x4001_AC00

复位值：0x0

表 19-2 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	0	0	0

位置	位名称	说明
[31:8]		未使用
[7:6]	DUPLEX	半双工模式设置 0X：关闭半双工模式 10：开启半双工模式，仅发送

		11：开启半双工模式，仅接收
[5]	CS	SPI 从设备下，片选信号来源。默认值为 1。 0：Slave 模式下，片选信号恒为有效值--0 1：Slave 模式下，片选信号来自 Master 设备
[4]	MS	SPI 主从模式选择。默认值为 0。 0：Slave 模式 1：Master 模式
[3]	CPHA	SPI 相位选择。默认值为 0。 0：Phase 为 0 1：Phase 为 1
[2]	CPOL	SPI 极性选择。默认值为 0。 0：Polarity 为 0 1：Polarity 为 1
[1]	ENDIAN	SPI 模块传输顺序。默认值为 0。 0：MSB，高位先传输 1：LSB，低位先传输
[0]	EN	SPI 模块使能信号。默认值为 0。 0：关闭 SPI 模块 1：开启 SPI 模块

19.4.3 SPI_IE SPI 中断寄存器

地址分别是：0x4001_A804，0x4001_AC04

复位值：0x0

表 19-3 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走，本次接收的新数据已经到达。 高电平有效，写 1 清除。

19.4.4 SPI_BAUD SPI 波特率寄存器

地址分别是：0x4001_A808，0x4001_AC08

复位值：0x0

表 19-4 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
[15]	TRANS_MODE	SPI 数据搬移方式。默认为 0，DMA 方式。 0：SPI 接口支持 DMA 搬移数据到 SPI 接口，完成发送和接收。 1：SPI 接口支持 MCU 搬移数据到 SPI 接口，完成发送和接收。
[14:12]	--	--
[11:0]	BAUD	SPI 传输波特率配置，SPI 实际传输速度计算公式为: SPI 传输速度 = 系统时钟 / (2*(BAUD + 1)) 切记，BAUD 的配置值不能小于 3。

19.4.5 SPI_TXDATA SPI 数据发送寄存器

地址分别是：0x4001_A80C，0x4001_AC0C

复位值：0x0

表 19-5 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

19.4.6 SPI_RXDATA SPI 数据接收寄存器

地址分别是：0x4001_A810，0x4001_AC10

复位值：0x0

表 19-6 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

19.4.7 SPI_SIZE SPI 数据字节长度寄存器

地址分别是：0x4001_A814，0x4001_AC14

复位值：0x0

表 19-7 SPI_BITSIZE 数据字节长度寄存器

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
[31:8]		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

20 CMP 比较器

20.1 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 CMP 模块使用 CMP 表示)，用于处理 6 个模拟轨到轨比较器产生的输出信号，由一系列使能、极性控制、滤波等数字电路组成。信号处理时钟由系统主时钟分频得到，此模块也用于向 CPU 产生比较器中断。

CMP0/1 可以使用 MCPWM0 的 4 路 P 通道进行开窗控制，CMP2/3 可以使用 MCPWM1 的 4 路 P 通道信号进行开窗控制，CMP4/5 不支持开窗控制。

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。

20.2 寄存器

20.2.1 地址分配

CMP 模块寄存器的基地址是 0x4001_3000，寄存器列表如下:

表 20-1 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_BLCWIN0	0x10	比较器开窗控制寄存器 0
CMP_BLCWIN1	0x14	比较器开窗控制寄存器 1
CMP_DATA	0x18	比较器输出数值寄存器

20.2.2 CMP_IE 中断使能寄存器

地址：0x4001_3000

复位值：0x0

表 20-2 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

	CMP5_RE	CMP4_RE	CMP3_RE	CMP2_RE	CMP1_RE	CMP0_RE		CMP5_IE	CMP4_IE	CMP3_IE	CMP2_IE	CMP1_IE	CMP0_IE
	RW	RW	RW	RW	RW	RW		RW	RW	RW	RW	RW	RW
	0	0	0	0	0	0		0	0	0	0	0	0

位置	位名称	说明
[31:14]		未使用
[13]	CMP5_RE	比较器 5 DMA 请求使能，高有效
[12]	CMP4_RE	比较器 4 DMA 请求使能，高有效
[11]	CMP3_RE	比较器 3 DMA 请求使能，高有效
[10]	CMP2_RE	比较器 2 DMA 请求使能，高有效
[9]	CMP1_RE	比较器 1 DMA 请求使能，高有效
[8]	CMP0_RE	比较器 0 DMA 请求使能，高有效
[7:6]		未使用
[5]	CMP5_IE	比较器 5 中断使能，高有效
[4]	CMP4_IE	比较器 4 中断使能，高有效
[3]	CMP3_IE	比较器 3 中断使能，高有效
[2]	CMP2_IE	比较器 2 中断使能，高有效
[1]	CMP1_IE	比较器 1 中断使能，高有效
[0]	CMP0_IE	比较器 0 中断使能，高有效

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。

20.2.3 CMP_IF 中断标志寄存器

地址：0x4001_3004

复位值：0x0

表 20-3 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
										CMP5_IF	CMP4_IF	CMP3_IF	CMP2_IF	CMP1_IF	CMP0_IF
										RW	RW	RW	RW	RW	RW
										0	0	0	0	0	0

位置	位名称	说明
[31:6]		未使用
[5]	CMP5_IF	比较器 5 中断标志，高有效，写 1 清零

[4]	CMP4_IF	比较器 4 中断标志，高有效，写 1 清零
[3]	CMP3_IF	比较器 3 中断标志，高有效，写 1 清零
[2]	CMP2_IF	比较器 2 中断标志，高有效，写 1 清零
[1]	CMP1_IF	比较器 1 中断标志，高有效，写 1 清零
[0]	CMP0_IF	比较器 0 中断标志，高有效，写 1 清零

20.2.4 CMP_TCLK 分频时钟控制寄存器

地址：0x4001_3008

复位值：0x0

表 20-4 比较器分频时钟控制寄存器 CMP_TCLK

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								FIL_CLK54_DIV16				CLK54_EN	FIL_CLK54_DIV2		
								RW				RW	RW		
								0				0	0		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIL_CLK32_DIV16				CLK32_EN	FIL_CLK32_DIV2			FIL_CLK10_DIV16				CLK10_EN	FIL_CLK10_DIV2		
RW				RW	RW			RW				RW	RW		
0				0	0			0				0	0		

位置	位名称	说明
[31:24]		未使用
[23:20]	FIL_CLK54_DIV16	比较器 5/4 滤波时钟分频，基于 MCLK 进行 1~16 分频，影响进入比较器中断的时间
[19]	CLK54_EN	比较器 5/4 时钟使能，高有效
[18:16]	FIL_CLK54_DIV2	比较器 5/4 滤波时钟分频，3'h0:1 分频, 3'h1:2 分频, 3'h2:4 分频, 3'h3:8 分频, 3'h4:16 分频, 3'h5:32 分频, 3'h6:64 分频, 3'h7:128 分频
[15:12]	FIL_CLK32_DIV16	比较器 3/2 滤波时钟分频，基于 MCLK 进行 1~16 分频，影响进入比较器中断的时间
[11]	CLK32_EN	比较器 3/2 时钟使能，高有效
[10:8]	FIL_CLK32_DIV2	比较器 3/2 滤波时钟分频，3'h0:1 分频, 3'h1:2 分频, 3'h2:4 分频,

		3'h3:8 分频, 3'h4:16 分频, 3'h5:32 分频, 3'h6:64 分频, 3'h7:128 分频
[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，3'h0:1 分频, 3'h1:2 分频, 3'h2:4 分频, 3'h3:8 分频, 3'h4:16 分频, 3'h5:32 分频, 3'h6:64 分频, 3'h7:128 分频

CMP 滤波时钟频率

Freq(CMP_Filter) = Freq(MCLK)/$2^{CMP\_TCLK.FIL\_CLK\_DIV2}$/(CMP_TCLK.FIL_CLK_DIV16+1)，其中 MCLK 为系统的主时钟。需要注意的是，产生 CMP 滤波时钟需要使能 CMP_TCLK.CLK_EN 位。

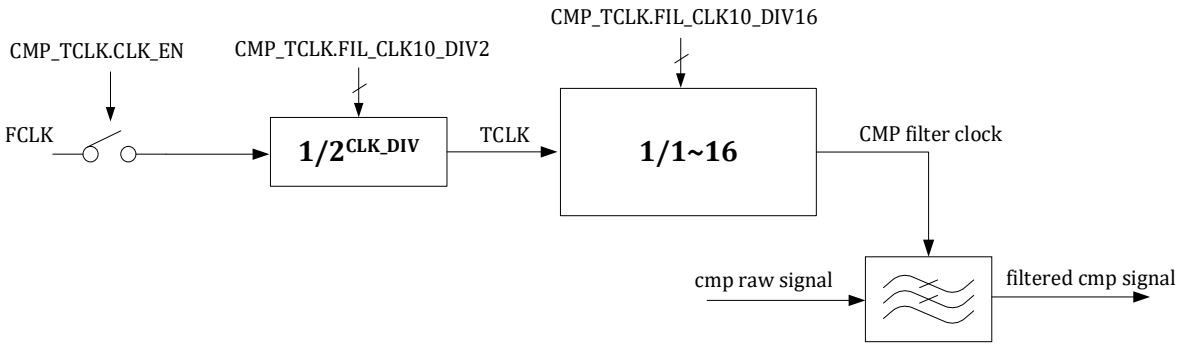


图 20-1 比较器滤波时钟产生

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 个总线周期。

20.2.5 CMP_CFG 控制寄存器

地址：0x4001_300C

复位值：0x0

表 20-5 比较器控制寄存器 CMP_CFG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								CMP5_W_PWM_POL	CMP5_IRQ_TRIG	CMP5_IN_EN	CMP5_POL	CMP4_W_PWM_POL	CMP4_IRQ_TRIG	CMP4_IN_EN	CMP4_POL
								RW	RW	RW	RW	RW	RW	RW	RW

								0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP3_W_PWM_POL	CMP3_IRQ_TRIG	CMP3_IN_EN	CMP3_POL	CMP2_W_PWM_POL	CMP2_IRQ_TRIG	CMP2_IN_EN	CMP2_POL	CMP1_W_PWM_POL	CMP1_IRQ_TRIG	CMP1_IN_EN	CMP1_POL	CMP0_W_PWM_POL	CMP0_IRQ_TRIG	CMP0_IN_EN	CMP0_POL
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:24]		未使用
[23]	CMP5_W_PWM_POL	比较器 5 开窗 PWM 信号极性选择，在 CMP_BLCWIN1 使能情况下使用
[22]	CMP5_IRQ_TRIG	比较器 5 中断触发类型，0:电平触发，1:边沿触发
[21]	CMP5_IN_EN	比较器 5 信号输入使能
[20]	CMP5_POL	比较器 5 极性选择，0:高电平有效；1:低电平有效
[19]	CMP4_W_PWM_POL	比较器 4 开窗 PWM 信号极性选择，在 CMP_BLCWIN1 使能情况下使用
[18]	CMP4_IRQ_TRIG	比较器 4 中断触发类型，0:电平触发，1:边沿触发
[17]	CMP4_IN_EN	比较器 4 信号输入使能
[16]	CMP4_POL	比较器 4 极性选择，0:高电平有效；1:低电平有效
[15]	CMP3_W_PWM_POL	比较器 3 开窗 PWM 信号极性选择，在 CMP_BLCWIN0 使能情况下使用
[14]	CMP3_IRQ_TRIG	比较器 3 中断触发类型，0:电平触发，1:边沿触发
[13]	CMP3_IN_EN	比较器 3 信号输入使能
[12]	CMP3_POL	比较器 3 极性选择，0:高电平有效；1:低电平有效
[11]	CMP2_W_PWM_POL	比较器 2 开窗 PWM 信号极性选择，在 CMP_BLCWIN0 使能情况下使用
[10]	CMP2_IRQ_TRIG	比较器 2 中断触发类型，0:电平触发，1:边沿触发
[9]	CMP2_IN_EN	比较器 2 信号输入使能
[8]	CMP2_POL	比较器 2 极性选择，0:高电平有效；1:低电平有效
[7]	CMP1_W_PWM_POL	比较器 1 开窗 PWM 信号极性选择，在 CMP_BLCWIN0 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

以比较器 0/1 为例，其极性及使能控制如图 20-2 所示。

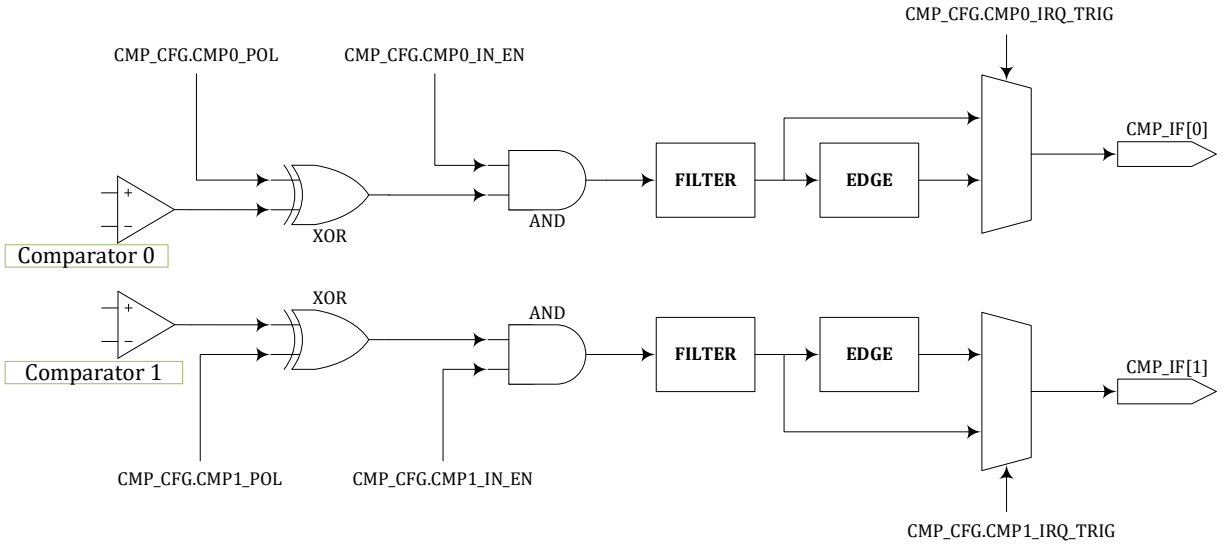


图 20-2 比较器控制及中断产生逻辑(以 CMP0/1 为例)

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于开窗控制无关，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。

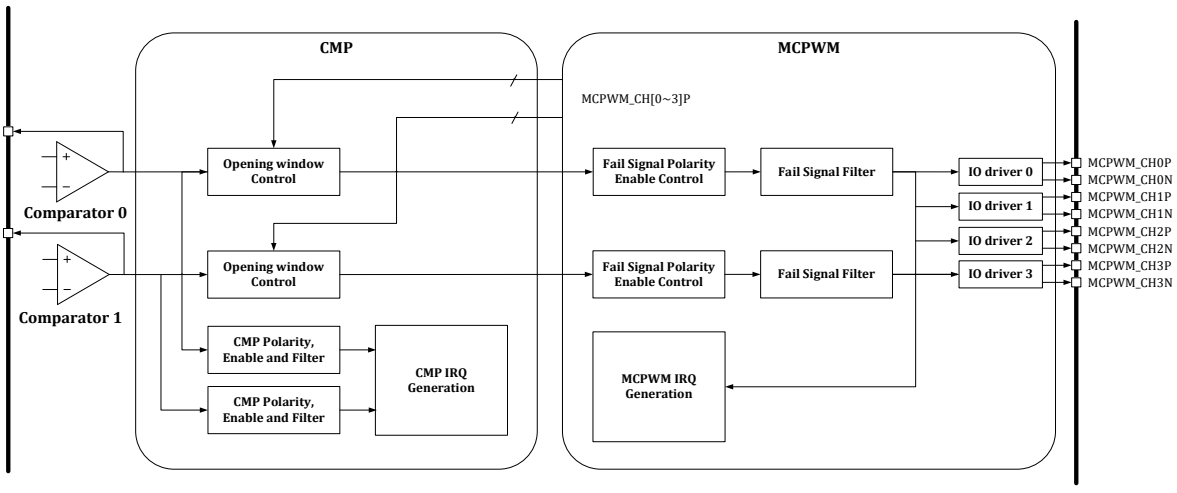


图 20-3 CMP 与 MCPWM 的联动

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 20-4 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。

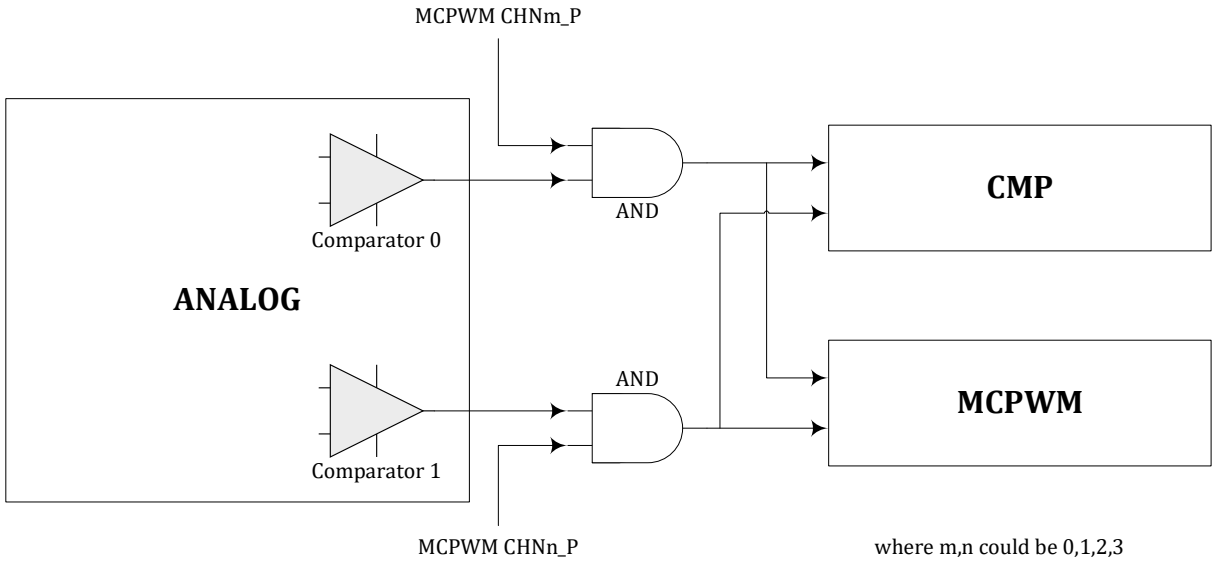


图 20-4 比较器开窗功能图示

20.2.6 CMP_BLCWIN0 开窗控制寄存器 0

地址：0x4001_3010

复位值：0x0

表 20-6 比较器开窗控制寄存器 0 CMP_ BLCWIN0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP3_CHN3P_WIN_EN	CMP3_CHN2P_WIN_EN	CMP3_CHN1P_WIN_EN	CMP3_CHN0P_WIN_EN	CMP2_CHN3P_WIN_EN	CMP2_CHN2P_WIN_EN	CMP2_CHN1P_WIN_EN	CMP2_CHN0P_WIN_EN	CMP1_CHN3P_WIN_EN	CMP1_CHN2P_WIN_EN	CMP1_CHN1P_WIN_EN	CMP1_CHN0P_WIN_EN	CMP0_CHN3P_WIN_EN	CMP0_CHN2P_WIN_EN	CMP0_CHN1P_WIN_EN	CMP0_CHN0P_WIN_EN
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:16]		保留
[15]	CMP3_CHN3P_WIN_EN	使用 MCPWM1 模块 CHN3_P 通道输出的 P 管开关控制信号作为比较器 3 开窗使能

[14]	CMP3_CHN2P_WIN_EN	使用 MCPWM1 模块 CHN2_P 通道输出的 P 管开关控制信号作为比较器 3 开窗使能
[13]	CMP3_CHN1P_WIN_EN	使用 MCPWM1 模块 CHN1_P 通道输出的 P 管开关控制信号作为比较器 3 开窗使能
[12]	CMP3_CHN0P_WIN_EN	使用 MCPWM1 模块 CHN0_P 通道输出的 P 管开关控制信号作为比较器 3 开窗使能
[11]	CMP2_CHN3P_WIN_EN	使用 MCPWM1 模块 CHN3_P 通道输出的 P 管开关控制信号作为比较器 2 开窗使能
[10]	CMP2_CHN2P_WIN_EN	使用 MCPWM1 模块 CHN2_P 通道输出的 P 管开关控制信号作为比较器 2 开窗使能
[9]	CMP2_CHN1P_WIN_EN	使用 MCPWM1 模块 CHN1_P 通道输出的 P 管开关控制信号作为比较器 2 开窗使能
[8]	CMP2_CHN0P_WIN_EN	使用 MCPWM1 模块 CHN0_P 通道输出的 P 管开关控制信号作为比较器 2 开窗使能
[7]	CMP1_CHN3P_WIN_EN	使用 MCPWM0 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 MCPWM0 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 MCPWM0 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 MCPWM0 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 MCPWM0 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 MCPWM0 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 MCPWM0 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 MCPWM0 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

20.2.7 CMP_BLCWIN1 开窗控制寄存器 1

地址：0x4001_3014

复位值：0x0

表 20-7 比较器开窗控制寄存器 1 CMP_ BLCWIN0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP5_1CHN3P_WIN_EN	CMP5_1CHN2P_WIN_EN	CMP5_1CHN1P_WIN_EN	CMP5_1CHN0P_WIN_EN	CMP5_0CHN3P_WIN_EN	CMP5_0CHN2P_WIN_EN	CMP5_0CHN1P_WIN_EN	CMP5_0CHN0P_WIN_EN	CMP4_1CHN3P_WIN_EN	CMP4_1CHN2P_WIN_EN	CMP4_1CHN1P_WIN_EN	CMP4_1CHN0P_WIN_EN	CMP4_0CHN3P_WIN_EN	CMP4_0CHN2P_WIN_EN	CMP4_0CHN1P_WIN_EN	CMP4_0CHN0P_WIN_EN

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

位置	位名称	说明
[31:16]		保留
[15]	CMP5_1CHN3P_WIN_EN	使用 MCPWM1 模块 CHN3_P 通道输出的 P 管开关控制信号作为比较器 5 开窗使能
[14]	CMP5_1CHN2P_WIN_EN	使用 MCPWM1 模块 CHN2_P 通道输出的 P 管开关控制信号作为比较器 5 开窗使能
[13]	CMP5_1CHN1P_WIN_EN	使用 MCPWM1 模块 CHN1_P 通道输出的 P 管开关控制信号作为比较器 5 开窗使能
[12]	CMP5_1CHN0P_WIN_EN	使用 MCPWM1 模块 CHN0_P 通道输出的 P 管开关控制信号作为比较器 5 开窗使能
[11]	CMP5_0CHN3P_WIN_EN	使用 MCPWM0 模块 CHN3_P 通道输出的 P 管开关控制信号作为比较器 5 开窗使能
[10]	CMP5_0CHN2P_WIN_EN	使用 MCPWM0 模块 CHN2_P 通道输出的 P 管开关控制信号作为比较器 5 开窗使能
[9]	CMP5_0CHN1P_WIN_EN	使用 MCPWM0 模块 CHN1_P 通道输出的 P 管开关控制信号作为比较器 5 开窗使能
[8]	CMP5_0CHN0P_WIN_EN	使用 MCPWM0 模块 CHN0_P 通道输出的 P 管开关控制信号作为比较器 5 开窗使能
[7]	CMP4_1CHN3P_WIN_EN	使用 MCPWM1 模块 CHN3_P 通道输出的 P 管开关控制信号 1 作为比较器 4 开窗使能
[6]	CMP4_1CHN2P_WIN_EN	使用 MCPWM1 模块 CHN2_P 通道输出的 P 管开关控制信号作为比较器 4 开窗使能
[5]	CMP4_1CHN1P_WIN_EN	使用 MCPWM1 模块 CHN1_P 通道输出的 P 管开关控制信号作为比较器 4 开窗使能
[4]	CMP4_1CHN0P_WIN_EN	使用 MCPWM1 模块 CHN0_P 通道输出的 P 管开关控制信号作为比较器 4 开窗使能
[3]	CMP4_0CHN3P_WIN_EN	使用 MCPWM0 模块 CHN3_P 通道输出的 P 管开关控制信号作为比较器 4 开窗使能
[2]	CMP4_0CHN2P_WIN_EN	使用 MCPWM0 模块 CHN2_P 通道输出的 P 管开关控制信号作为比较器 4 开窗使能
[1]	CMP4_0CHN1P_WIN_EN	使用 MCPWM0 模块 CHN1_P 通道输出的 P 管开关控制信号作为比较器 4 开窗使能
[0]	CMP4_0CHN0P_WIN_EN	使用 MCPWM0 模块 CHN0_P 通道输出的 P 管开关控制信号作为比较器 4 开窗使能

通常 CMP_ BLCWIN[3:0]中有 1bit 为 1,表明使用对应的 CHNx_P 对比较器的信号产生进行控制。如果 CMP_ BLCWIN[3:0]为 4'b0000，则表示比较器比较信号的产生与 MCPWM 信号无关。

CMP0/1 可以使用 MCPWM0 的 4 路 P 通道进行开窗控制，CMP2/3 可以使用 MCPWM1 的 4 路

P 通道信号进行开窗控制，CMP4/5 既可以使用 MCPWM0，也可以使用 MCPWM1 的 4 路 P 通道信号进行开窗控制。

20.2.8 CMP_DATA 输出数据寄存器

地址：0x4001_3018

复位值：0x0

表 20-8 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
		CMP5_FLT_DATA	CMP4_FLT_DATA	CMP3_FLT_DATA	CMP2_FLT_DATA	CMP1_FLT_DATA	CMP0_FLT_DATA			CMP5_RAW_DATA	CMP4_RAW_DATA	CMP3_RAW_DATA	CMP2_RAW_DATA	CMP1_RAW_DATA	CMP0_RAW_DATA
		RO	RO	RO	RO	RO	RO			RO	RO	RO	RO	RO	RO
		0	0	0	0	0	0			0	0	0	0	0	0

位置	位名称	说明
[31:14]		未使用
[13]	CMP5_FLT_DATA	比较器 5 经过滤波后的信号
[12]	CMP4_FLT_DATA	比较器 4 经过滤波后的信号
[11]	CMP3_FLT_DATA	比较器 3 经过滤波后的信号
[10]	CMP2_FLT_DATA	比较器 2 经过滤波后的信号
[9]	CMP1_FLT_DATA	比较器 1 经过滤波后的信号
[8]	CMP0_FLT_DATA	比较器 0 经过滤波后的信号
[7:6]		未使用
[5]	CMP5_RAW_DATA	比较器 5 原始输出信号，直接来自模拟比较器 5
[4]	CMP4_RAW_DATA	比较器 4 原始输出信号，直接来自模拟比较器 4
[3]	CMP3_RAW_DATA	比较器 3 原始输出信号，直接来自模拟比较器 3
[2]	CMP2_RAW_DATA	比较器 2 原始输出信号，直接来自模拟比较器 2
[1]	CMP1_RAW_DATA	比较器 1 原始输出信号，直接来自模拟比较器 1
[0]	CMP0_RAW_DATA	比较器 0 原始输出信号，直接来自模拟比较器 0

21 CAN-FD

21.1 介绍

CAN 总线控制器是符合 CAN 协议的串行通信控制器，满足 CAN 2.0B 规范的所有约束，同时支持 CAN-FD。

CAN 协议使用多主总线配置在网络节点之间传输帧（通信对象）并进行错误管理，以减少给 CPU 带来额外负担。CAN 总线控制器使用户能够在各种组件之间建立经济可靠的链接。 CAN 总线控制器对微控制器来说是一个内存映射 I/O 设备。 CPU 访问 CAN 总线控制器以通过两线 CAN 总线系统控制帧的发送或接收。与 CAN 总线的连接如下所示。

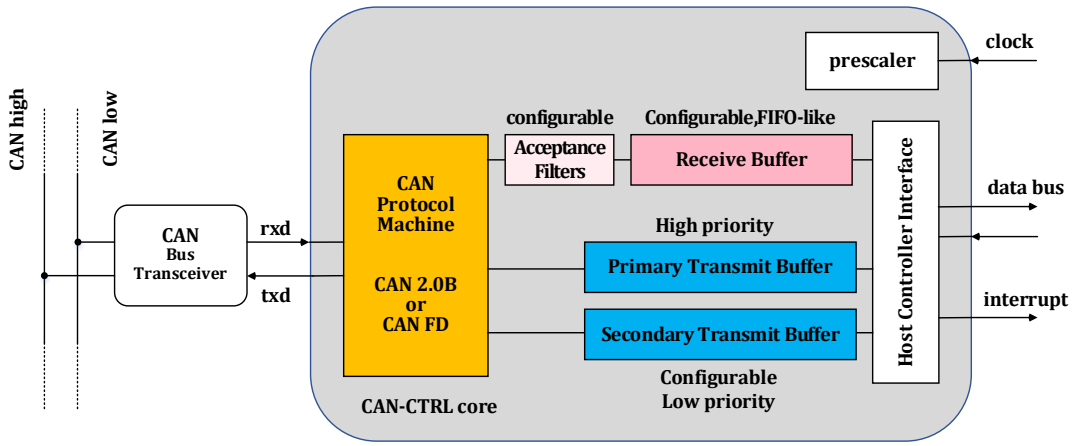


图 21-1 CAN 总线的连接和 CAN 总线控制器的主要特性

21.1.1 CAN 协议

CAN 通信是逐帧进行的。CAN 协议存在两种类型的帧：标准帧和扩展帧。对于 CAN 2.0，最大数据有效负载最多为 8 个字节，而对于 CAN FD，一帧最多可以传输 64 个字节。

所有 CAN 节点在总线上都是平等的。CAN 是多主总线，不存在超级节点。

数据寻址是使用消息标识符完成的。在 CAN 网络中，只有一个节点可以传输带有特定标识符的消息。所有节点都接收所有发出的消息。为了减少 CPU 的负载，CAN 控制器可以使用接收过滤器。这些过滤器将所有接收到的消息标识符与用户可选择的位掩码进行比较。只有当消息通过接收过滤器时，它才会存储在接收缓冲区中并发送给 CPU。

帧的标识符也用于总线仲裁。当另一个 CAN 节点传输具有较高优先级标识符的消息时，CAN 协议状态机停止传输具有低优先级标识符的消息，并自动尝试在下一个可能的时机重新传输停止的消息。小的标识符优先级更高。

CAN 2.0B 定义了高达 1Mbit/s 的数据波特率。对于 CAN FD，没有特定限制。CAN FD 标准定义

了一种波特率切换的机制。如果启用波特率切换，帧有效载荷的传输可以以更高的速度完成，而帧头以较低的速度传输。

21.1.2 CAN 2.0B 以及如何启用 CAN FD

CAN FD 是 CAN 2.0B 协议的向下兼容扩展。每个 CAN FD 节点都能够接收和传输经典的 CAN 帧。

主机软件可以在运行时，使用发送缓冲区中的配置位，为每个帧选择是作为经典 CAN 或 CAN FD 帧进行传输。对于接收，接收帧是经典 CAN 或 CAN FD 帧，由接收缓冲区中的状态位进行指示。

21.1.3 时间戳

21.1.3.1 TTCAN（Time Trigger CAN）

根据 ISO 11898-4 标准，CAN 控制器可选择使用时间触发 CAN 通信机制 (TTCAN)。CAN 控制器提供部分硬件支持，在此模式下需要主机软件与 CAN 控制器进行实时交互。

TTCAN 的基本概念是存在一个定时器，用于为接收到的消息加上时间戳，并触发消息进行传输。CAN 网络中的一个节点是时间主机。时间主机发送参考帧。随着参考帧发送，循环时间开始。两个参考帧之间的时间是一个基本周期。基本周期内的消息可以在特定时间窗口中传输。 TTCAN 系统管理节点在离线设置期间设定好每个时间窗口的开始和持续时间。

共有三种时间窗口类型：

- 独家时间窗口 （只允许一个节点发送一个具有特定ID的帧）
- 自由时间窗口 （未使用的时间窗口，用于未来扩展网络）
- 仲裁时间窗口 （几个节点可能会同时传输，可能导致仲裁发生）

如果接收到一帧，它会获取实际的循环时间作为时间戳。对于传输，CAN 控制器提供了一个硬件触发机制，可以在预定义的时机开始传输事先准备好的消息。

CAN 控制器在接收时自动检测参考帧并开始循环时间。硬件定时器是一个 16 位定时器，按照 ISO 11898-4 Level 1 定义以 CAN bit 时间为单位运行。

除了用于消息传输的硬件触发之外，CAN 控制器还提供了一个超时监测触发来检测参考帧丢失的情况。

主控制器需要为每个时间窗口配置节点的动作。例如，主机需要为下一次传输准备消息数据并为其定义触发时间。换句话说：主机需要对传输时空进行维护。

21.1.3.2 CiA 603 时间戳

CAN in Automation (CiA) 在规范 603 中定义了一种时间戳产生方法，CAN 控制器支持该特性。CiA603 可以与 TTCAN 一起使用。

CiA 603 的基本概念是有一个独立运行的定时器，以系统时钟周期而不是 CAN bit 时间进行计数。在 CAN/CAN FD 帧的 SOF 或 EOF 处获取时间戳。

CiA 603 支持 AUTOSAR 的时间戳和时间同步。对于 AUTOSAR，CAN 网络中的一个节点是时间主机。时间主机发送同步消息（SYNC 消息）。 SYNC 消息的时间戳由时间主机和所有时间从机获

取。指示 SYNC 消息的发送与实际发送 SYNC 消息之间的时间差将由时间主机在 follow-up (FUP) 消息中传输。

21.2 特性

21.2.1 功能列表

- CAN规格
  - 支持2.0B（最多8个字节有效载荷，符合博世参考模型验证标准）
  - 支持FD（最多64字节有效载荷，ISO 11898-1:2015 或非ISO/博世）
- 可编程数据速度:
  - CAN 2.0B 定义数据速率高达 1Mbit/s
  - CAN FD受收发器限制
- 可编程波特率预分频器（1至1/256)
- 接收缓冲器 (RB)
  - 可容纳12帧(slot)
  - 类FIFO行为
  - “被滤除”或“不正确”的接收消息不会覆盖已存储的消息
- 两个发送缓冲器
  - 主发送缓冲器 (PTB)，可容纳1帧(slot)
  - 第二发送缓冲器 (STB)，可容纳4帧(slot)
- 独立可编程29bit接收过滤器
  - 共16个接收过滤器
- 扩展特征
  - 单次发传输模式（用于PTB、STB）
  - 监听模式
  - 环回模式（内部回环和外部回环）
- 扩展状态和错误报告
  - 捕获最后发生的错误和仲裁丢失位置
  - 可编程错误警告门限
- 可配置中断来源
- 时间戳:
  - ISO 11898-4 TTCAN
  - CiA 603 时间戳
- 兼容AUTOSAR

21.2.2 向上兼容

CAN 协议中存在用于协议扩展的保留位，已用于在 CAN 2.0B 规范之上构建 CAN FD 规范。如果不使用，保留位被传输为低（显性）。但 CAN 2.0B 规范定义了保留位为高（隐性）的行为。因此，如果使用此行为的 CAN 2.0B 节点接收到 CAN FD 帧（与 CAN 2.0B 帧相比具有不同帧格式），则这将导致 CAN 2.0B 节点产生错误帧。这种行为称为“CAN FD 不兼容”。

为了向上兼容新的协议规范，如果节点检测到保留位为高，则应发生所谓的协议异常事件。这适用于 CAN 2.0B 以及 CAN FD 节点。协议异常事件不会导致接收方采取任何行动。接收方只是忽略此帧，不回复 ACK，等待总线空闲并可以发送或接收下一帧。对于 CAN 2.0B 节点，这称为“CAN FD 兼容”，可在一个网络中实现 CAN 2.0B 和 FD 帧的共存。

21.2.3 消息缓冲区

21.2.3.1 消息缓冲区概念

消息缓冲区的概念如下图所示。单个缓冲区消息槽（slot）应足以容纳最大帧长。

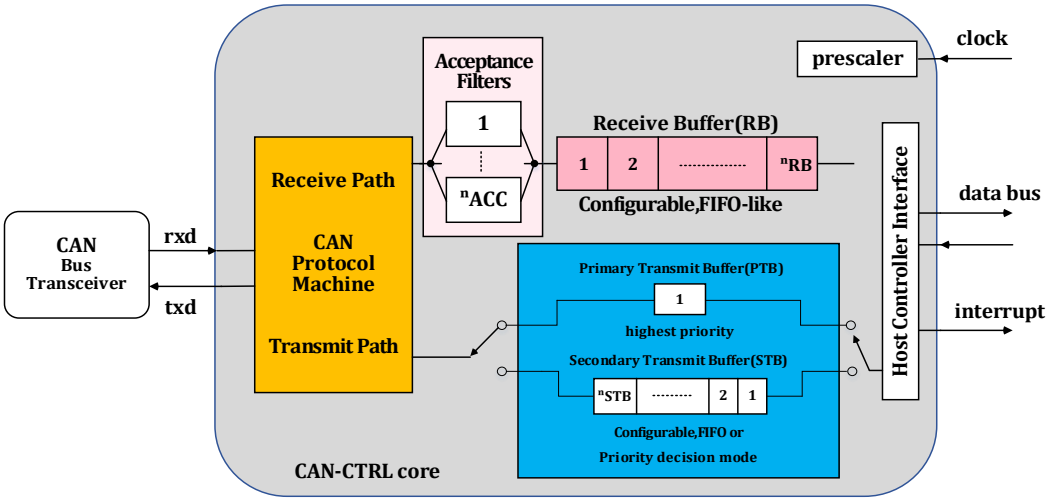


图 21-2 消息缓冲区

21.2.3.2 接收缓冲区

为了减少 CPU 的接收负载，CAN 控制可以使用接收过滤器。 CAN 总线控制器在接收过滤期间检查消息标识符。如果接收到的帧与其中一个接收过滤器的过滤标准匹配，则它将存储在接收缓冲区 (RB) 中，该缓冲区具有类似 FIFO 的行为。

根据可用消息槽的数量，主机控制器不需要立即读取接收到的消息。当 RB 已满或填充到软件选择的“几乎已满”门限时，CAN 总线控制器能够在后续每次收到消息时生成中断。由于 RB 具有类似 FIFO 的特性，主机控制器总是从 RB 读取最早的消息。

21.2.3.3 发送缓冲器

CAN 控制器配备了两个发送缓冲器 (TB)。Primary TB (PTB)具有更高的优先级，但只能存储一帧。Secondary TB (STB) 的优先级较低。它可以在 FIFO 或优先级模式下运行。 PTB 和 STB 之间的优先级决定是固定的，完全独立于 CAN 总线仲裁。总线仲裁基于帧标识符的优先级决定。

STB 可以被配置发送一帧或所有存储的帧。在 FIFO 模式下，每次发送都会先发送此缓冲区中最早的帧。在优先级模式中，缓冲区内具有最高优先级的帧（基于帧标识符）首先传输。

对于 CAN 协议状态机来说，PTB 中的帧总是比 STB 中的帧具有更高的优先级，而不管帧标识符

如何。PTB 传输可以停止并延迟 STB 传输。PTB 帧传输成功后，STB 传输会自动重启。

PTB 传输从下一个可能的传输时机开始（在下一个帧间隔之后）。但已取得仲裁并在实际进行传输的 STB 帧不会被打断，将先完成传输。

在以下情况下，可能会发生使用 PTB 传输中断 STB 传输：

1. STB被命令输出所有存储的帧，并且主机控制器决定在所有STB传输完成之前配置PTB进行一帧的传输。
2. STB被命令输出单个帧，并且主机控制器决定命令在STB传输完成之前配置PTB进行一帧的传输。

如果主机控制器一直等到每个命令的传输完成，那么它可以很容易地决定哪个缓冲区应该传输下一帧。但带有低优先级标识符的消息可能会阻塞更重要的消息。主机可以选择取消消息传输。

21.2.3.4 FIFO 模式下的发送缓冲器应用示例

在此示例中，CAN 节点用于传感器测量。 CAN 系统软件决定以低优先级处理这些传感器数据。

主控制器自动获取测量结果并将它们作为 CAN 帧放置在 STB 中。由于 CAN 总线上的高流量，可能会出现无法立即传输传感器数据帧并且在 STB 中保留若干帧的情况。稍后，如果 CAN 总线上的流量较少时，它们将被传输。

如果 STB 中已经保存了若干帧，主机控制器可能会有发送重要的高优先级帧的需求。在这种情况下，主机可以使用 PTB。 PTB 中的帧总是在 STB 之前传输。

拥有两个传输缓冲区的优点是可以选择保留所有消息。在高优先级事件的需要发送情况下，不必中止（丢弃）任何消息。

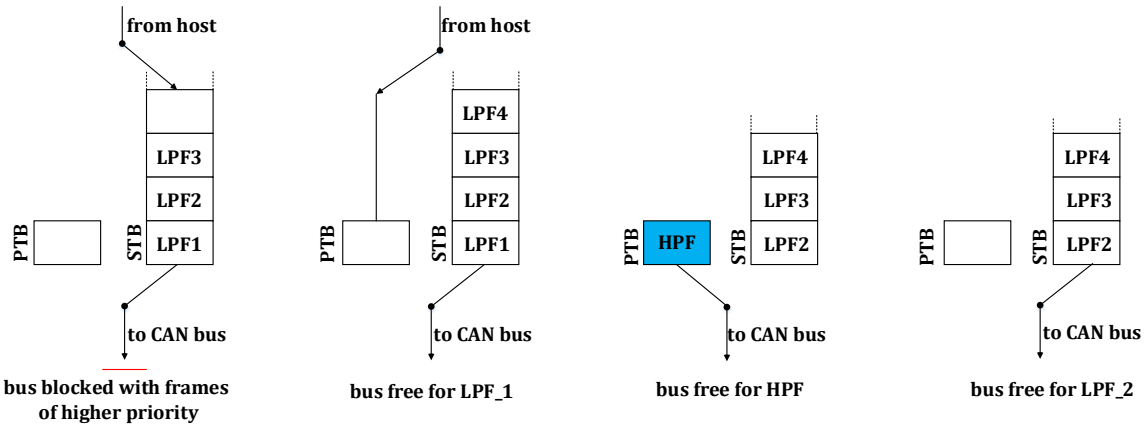


图 21-3 发送缓冲器应用示例（TSALL=1）

如图，LPF_x 是“低优先级帧”的缩写，HPF 表示“高优先级帧”。可以看出，LPF_1 被来自其他 CAN 节点的其他更高优先级帧阻塞。主控制器将第四帧放入 STB (LPF_4)，然后决定输出 HPF。此后，CAN 总线可以自由传输 LPF_1，然后是 HPF、LPF_2 等。

PTB 和 STB 之间的优先级决策由 CAN 协议状态机完成，但 CAN 协议本身使用另一种独立的优先级决策机制，称为总线仲裁。对于总线仲裁，帧标识符定义了优先级。

在上面给出的示例中，可以将具有非常低优先级标识符的帧放置在 PTB 中，而较高优先级的帧保留在 STB 中。无论帧标识符如何，PTB 总是优先发送。主机控制器的任务是将具有有意义的标识符优先级的帧放置在合适的缓冲区中。

21.2.3.5 优先模式下的发送缓冲器应用示例

在 STB 的优先模式下，CAN 控制器会自动更改 STB 内的帧顺序。具有最高优先级的帧首先被发送。**优先级决定与总线仲裁相同，因此具有较低值的帧标识符具有较高的优先级。**但 PTB 始终具有最高优先级。

一旦新帧准备好传输并写入 STB，就会自动重新排序 STB 内的帧，选择具有最高优先级的帧进行传输。主机控制器只需要将新帧写入 STB 即可自动加入重排。

通常，具有较高优先级的帧永远不会中断活动传输。以图 21-3 为例。在最左边的部分，帧 LPF_1 当前正在传输中，而 LPF_4 被写入 STB。即使 LPF_4 的优先级高于 LPF_1，LPF_1 仍然先会被传输完成，除非它在总线上失去仲裁。然后在节点失去仲裁后，将选择 LPF_4 进行传输，因为它具有更高优先级。

优先级重新排序在后台自动完成，但重排需要一些时间。当大量不同优先级的新帧写入 STB，节点失去仲裁或总线出现错误时，有可能不是优先级最高的帧被选中。如果在决定下一帧的截止日期之前写入该帧，可能会发生这种情况。但这是一种非常罕见的情况，通常不需关心这一点。

如果将具有相同优先级的两个帧写入 STB，则将首先传输最早的帧。

21.2.3.6 中止传输

如果出现这种情况，传输缓冲区中的消息由于其低优先级而无法发送，将阻塞缓冲区很长时间。为了避免这种情况，主控制器可以撤回传输请求并中止消息传输。 PTB 和 STB 的单帧或所有帧传输都可以中止。

21.2.3.7 TTCAN 模式下的发送缓冲器

在 TTCAN 模式下，STB 中的每个消息槽都可以由主机寻址。每个消息槽都可以定义为已填充或空。因此，每个消息槽可以专用于一个特定的消息。

传输由硬件触发启动。每个触发都包含一个指向消息槽的指针。 PTB 在 TTCAN 模式下没有特殊优先级，和 STB 消息槽进行相同处理。

主机应用程序处于方便考虑，可以将一条消息专用于一个缓冲槽。然后主机中的任务可以随时更新消息，然后如果该消息的触发时刻到达，则可以开始传输。但是为所有消息提供足够的消息槽需要大量硬件。因此可以在一个基本周期中为多个消息使用一个消息缓冲槽。因此，即使 STB 被禁用并且内核仅包含单个 PTB 消息槽，也可以支持 ISO 11898-4 协议。

21.2.4 CAN 2.0 和 CAN FD 帧

CAN FD 是 CAN 2.0 的协议扩展。主要区别在于：

- 数据负载：CAN 2.0最多8个字节，CAN-FD最多64个字节
- CAN 2.0有一种可配置的波特率，而CAN FD有2种：仲裁阶段使用低速率，数据阶段使用高速率。

CAN 规范中对缩写进行了说明。对于经典 CAN 帧(CAN 2.0)，一些位名称根据 CAN FD ISO 规范进行了重命名，但这里仍然使用旧名称以便于向后引用兼容。

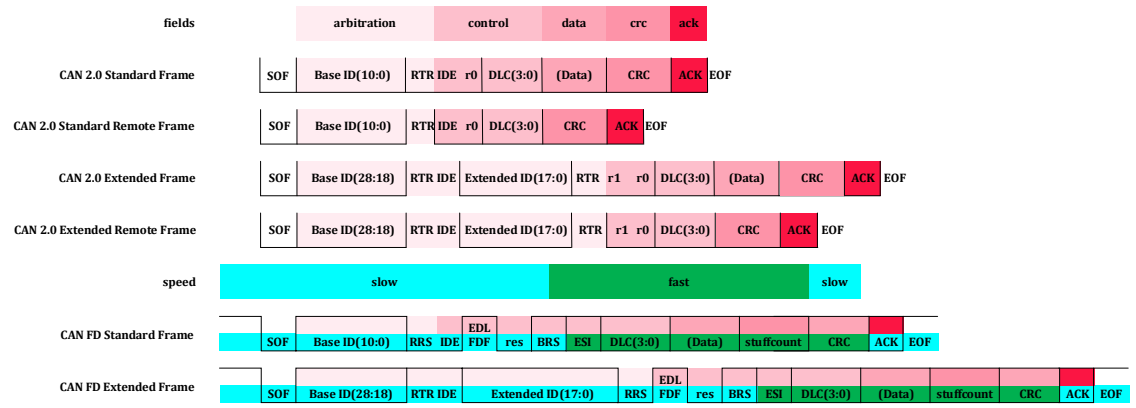


图 21-4 CAN 2.0 和 CAN FD 帧类型

表 21-1 CAN 位缩写

缩写	描述		评论
ID	IDentifier	标识符	

RTR	Remote Transmission Request	远程传输请求	远程或数据帧
SRR	Substitute Remote Request	替代远程请求	
RRS	Remote Request Substitution	远程请求替换	
IDE	IDentifier Extension	标识符扩展	标准或扩展框架
DLC	Data Length Code	数据长度代码	有效负载字节数
EDL	Extended Data Length	扩展数据长度	CAN 2.0 或 CAN FD 帧
FDF	FD Format indicator (=EDL)	FD 格式指示符 (=EDL)	CAN 2.0 或 CAN FD 帧
BRS	Bit Rate Switch	波特率开关	
ESI	Error State Indicator	错误状态指示器	
r1, r0, res	Reserved bits	保留位	

Bosch (非 ISO)的 CAN FD 规范使用 EDL，而 CAN FD ISO 规范使用 FDF 来表示同一位。本文档使用 FDF。

对于 CAN FD ISO 帧，填充计数作为 CRC 字段的一部分传输。对于 CAN FD 非 ISO 帧，填充计数不是帧的一部分。此外，CRC 对非 ISO 和 ISO 帧有不同的初始化系数。因此，ISO 和非 ISO 帧是不兼容的。

CAN 控制中的 CAN 协议状态机自动发送和接收帧并嵌入适当的控制和状态位。软件需要选择所需的帧类型（IDE、RTR、FDF），选择标识符并设置有效的数据载荷。

21.2.5 AUTOSAR 和 SAE J1939

AUTOSAR 和 SAE J1939 软件协议栈可与 CAN 控制器一起使用。

对于 AUTOSAR，传输缓冲区的数量应设置为至少 3 个消息槽，并且可以中止传输（寄存器 TCMD 中的 TPA 和 TSA）。符合 CiA 603 协议，支持时间戳和时间同步。

SAE J1939 使用扩展 ID（29 位）。为了便于处理，可以将接收过滤器配置为仅接收扩展 ID（寄存器 CAN_ACF）。

表 21-2 CAN 名称定义

缩写	描述
ACF	接收过滤器
PTB	主发送缓冲器（高优先级）
RB	接收缓冲区
RDC	接收器延迟补偿（与 TDC 相关）
RTS	接收时间戳
SP	采样点
SSP	二级采样点 (CAN FD)
STB	辅助发送缓冲器（低优先级）
TDC	发送器延迟补偿（CAN FD 规范）
TTCAN	时间触发 CAN (ISO 11898-4)
TTS	传输时间戳
TQ	时间量子（CAN 规范）

21.3 寄存器（软件接口）

CAN-FD 基地址为 0x40018800。

21.3.1 地址分配

CAN-FD 模块寄存器的基地址是 0x4001_1880，寄存器列表如下。

表 21-3 CAN 寄存器地址分配

名称	偏移	说明
CAN_RBUF_00	0x00	CAN Rx Buffer 寄存器 0
CAN_RBUF_01	0x04	CAN Rx Buffer 寄存器 1
CAN_RBUF_02	0x08	CAN Rx Buffer 寄存器 2
CAN_RBUF_03	0x0C	CAN Rx Buffer 寄存器 3
CAN_RBUF_04	0x10	CAN Rx Buffer 寄存器 4
CAN_RBUF_05	0x14	CAN Rx Buffer 寄存器 5
CAN_RBUF_06	0x18	CAN Rx Buffer 寄存器 6
CAN_RBUF_07	0x1C	CAN Rx Buffer 寄存器 7
CAN_RBUF_08	0x20	CAN Rx Buffer 寄存器 8
CAN_RBUF_09	0x24	CAN Rx Buffer 寄存器 9
CAN_RBUF_10	0x28	CAN Rx Buffer 寄存器 10
CAN_RBUF_11	0x2C	CAN Rx Buffer 寄存器 11
CAN_RBUF_12	0x30	CAN Rx Buffer 寄存器 12
CAN_RBUF_13	0x34	CAN Rx Buffer 寄存器 13
CAN_RBUF_14	0x38	CAN Rx Buffer 寄存器 14
CAN_RBUF_15	0x3C	CAN Rx Buffer 寄存器 15
CAN_RBUF_16	0x40	CAN Rx Buffer 寄存器 16
CAN_RBUF_17	0x44	CAN Rx Buffer 寄存器 17
CAN_RBUF_18	0x48	CAN Rx Buffer 寄存器 18
CAN_RBUF_19	0x4C	CAN Rx Buffer 寄存器 19
CAN_TBUF_00	0x50	CAN Tx Buffer 寄存器 0
CAN_TBUF_01	0x54	CAN Tx Buffer 寄存器 1
CAN_TBUF_02	0x58	CAN Tx Buffer 寄存器 2
CAN_TBUF_03	0x5C	CAN Tx Buffer 寄存器 3
CAN_TBUF_04	0x60	CAN Tx Buffer 寄存器 4
CAN_TBUF_05	0x64	CAN Tx Buffer 寄存器 5
CAN_TBUF_06	0x68	CAN Tx Buffer 寄存器 6
CAN_TBUF_07	0x6C	CAN Tx Buffer 寄存器 7
CAN_TBUF_08	0x70	CAN Tx Buffer 寄存器 8
CAN_TBUF_09	0x74	CAN Tx Buffer 寄存器 9
CAN_TBUF_10	0x78	CAN Tx Buffer 寄存器 10

CAN_TBUF_11	0x7C	CAN Tx Buffer 寄存器 11
CAN_TBUF_12	0x80	CAN Tx Buffer 寄存器 12
CAN_TBUF_13	0x84	CAN Tx Buffer 寄存器 13
CAN_TBUF_14	0x88	CAN Tx Buffer 寄存器 14
CAN_TBUF_15	0x8C	CAN Tx Buffer 寄存器 15
CAN_TBUF_16	0x90	CAN Tx Buffer 寄存器 16
CAN_TBUF_17	0x94	CAN Tx Buffer 寄存器 17
CAN_TTS_0	0x98	CAN 传输时间戳寄存器
CAN_TTS_1	0x9C	CAN 传输时间戳寄存器
CAN_CFG_STAT	0xA0	CAN 配置和状态寄存器
CAN_TCMD	0xA1	CAN 发送命令寄存器
CAN_TCTRL	0xA2	CAN 发送控制寄存器
CAN_RCTRL	0xA3	CAN 接收控制寄存器
CAN_RTIE	0xA4	CAN 发送接收中断控制寄存器
CAN_RTIF	0xA5	CAN 发送接收中断标志寄存器
CAN_ERRINT	0xA6	CAN 错误中断使能和标志寄存器
CAN_LIMIT	0xA7	CAN 警告寄存器
CAN_SBAUD	0xA8	CAN 慢速波特率配置寄存器
CAN_FBAUD	0xAC	CAN 快速波特率配置寄存器
CAN_EALCAP	0xB0	CAN 错误信息和丢失仲裁信息记录寄存器
CAN_TDC	0xB1	CAN 发送延时补偿寄存器
CAN_RECNT	0xB2	CAN 接收错误计数器寄存器
CAN_TECNT	0xB3	CAN 发送错误计数器寄存器
CAN_ACFCTRL	0xB4	CAN ID 过滤器控制寄存器
CAN_TIMECFG	0xB5	CAN IDCiA603 时间戳配置寄存器
CAN_ACFEN	0xB6	CAN ID 过滤器使能寄存器
CAN_ACF	0xB8	CAN ID 过滤器选择寄存器
CAN_TBSLOT	0xBE	TTCAN 发送缓冲区指针
CAN_TTCFG	0xBF	TTCAN 配置寄存器
CAN_REF_MSG	0xC0	TTCAN 参考帧寄存器
CAN_TRG_CFG	0xC4	TTCAN 触发配置寄存器
CAN_TT_TRIG	0xC6	TTCAN 触发时刻寄存器
CAN_TT_WTRIG	0xC8	TTCAN 超时监测寄存器
CAN_CIAWDAT0	0xD0	CiA603 时间戳低 32 位写入值
CAN_CIAWDAT1	0xD4	CiA603 时间戳高 32 位写入值

21.3.2 寄存器位分配表

访问定义使用 <access>-<reset_value> 形式的缩写给出。 <access> 属性：“r”表示读，“w”表示写，“rw”表示读/写访问。对于未初始化的寄存器，<reset_value >属性可以是“0”、“1”和“u”。示例：“rw-0”表示“可读可写并重置为 0”，“ru”表示“可读且未初始化”。未使用的位总是 r-0。

表 21-4 CAN 寄存器位分配表

偏移	位置								名称
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
0x00 ~ 0x4f	接收缓冲寄存器和接收时间戳								RBUF（和 RTS）
0x50 ~ 0x97	发送缓冲寄存器								TBUF
0x98 ~ 0x9f	传输时间戳								TTS
0xa0	RESET	LBME	LBMI	TPSS	TSSS	RACTIVE	TACTIVE	BUSOFF	CFG_STAT
0xa1	TBSEL	LOM	STBY	TPE	TPA	TSONE	TSALL	TSA	TCMD
0xa2	FD_ISO *	TSNEXT	TSMODE	TTTBM	-		TSSTAT(1:0)		TCTRL
0xa3	SACK	ROM	ROV	RREL	RBALL	-	RSTAT(1:0)		RCTRL
0xa4	RIE	ROIE	RFIE	RAFIE	TPIE	TSIE	EIE	TSFF	RTIE
0xa5	RIF	ROIF	RFIF	RAFIF	TPIF	TSIF	EIF	AIF	RTIF
0xa6	EWARN	EPASS	EPIE	EPIF	ALIE	ALIF	BEIE	BEIF	ERRINT
0xa7	AFWL(3:0)				EWL(3:0)				LIMIT
0xa8	S_Seg_1(7:0)								S_Seg_1 *
0xa9	-	S_Seg_2(6:0)							S_Seg_2 *
0xaa	-	S_SJW(6:0)							S_SJW *
0xab	S_PRESC(7:0)								S_PRESC *
0xac	-			F_Seg_1(4:0)					F_Seg_1 *
0xad	-				F_Seg_2(3:0)				F_Seg_2 *
0xae	-				F_SJW(3:0)				F_SJW *
0xaf	F_PRESC(7:0)								F_PRESC *
0xb0	KOER(2:0)			ALC(4:0)					EALCAP
0xb1	TDCEN	SSPOFF(6:0)							TDC *
0xb2	RECNT								RECNT
0xb3	TECNT								TECNT
0xb4	-		SELMASK	-	ACFADR				ACFCTRL
0xb5	-						TIMEPOS	TIMEEN	TIMECFG
0xb6	AE_7	AE_6	AE_5	AE_4	AE_3	AE_2	AE_1	AE_0	ACF_EN_0
0xb7	AE_15	AE_14	AE_13	AE_12	AE_11	AE_10	AE_9	AE_8	ACF_EN_1
0xb8	ACODE_x or AMASK_x (7:0)								ACF_0 *
0xb9	ACODE_x or AMASK_x (15:8)								ACF_1 *
0xba	ACODE_x or AMASK_x (23:16)								ACF_2 *
0xbb	-	IDCMP	IDMASK	ACODE_x or AMASK_x (28:24)					ACF_3 *
0xbc	VERSION(7:0)								VER_0
0xbd	VERSION(15:8)								VER_1
0xbe	TBE	TBF	TBPTR(5:0)						TBSLOT

偏移	位置							名称
0xbf	WTIE	WTIF	TEIF	TTIE	TTIF	T_PRESC(1:0)	TTEN	TTCFG
0xc0	REF_ID(7:0)							REF_MSG_0
0xc1	REF_ID(15:8)							REF_MSG_1
0xc2	REF_ID(23:16)							REF_MSG_2
0xc3	REF_IDE	-		REF_ID(28:24)				REF_MSG_3
0xc4	-		TTPTR(5:0)					TRIG_CFG_0
0xc5	TEW(3:0)				-	TTYPE(2:0)		TRIG_CFG_1
0xc6	TT_TRIG(7:0)							TT_TRIG_0
0xc7	TT_TRIG(15:8)							TT_TRIG_1
0xc8	TT_WTRIG(7:0)							TT_WTRIG_0
0xc9	TT_WTRIG(15:8)							TT_WTRIG_1
0xca	-							-
0xcb	-							-

* 只有在寄存器 CAN_CFG_STAT.RESET=1 时，才能写入寄存器。

对寄存器映射中未显示的可寻址位置的写访问不会导致任何操作，读回为 0x00。

21.3.3 寄存器说明

21.3.3.1 CAN_RBUF0~CAN_RBU19 寄存器

地址：0x4001_1880~0x4001_18CC

复位值：x

表 21-5 接收缓冲寄存器 CAN_RBUF0~CAN_RBUF19

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

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

位置	位名称	说明
[31:0]	RDATA	接收缓冲区寄存器，从 Rx Buffer Slot 中读出 CAN 帧的数据

不同帧的类型及数据长度不同，CAN_RBUF0~CAN_RBUF19 的数据内容不同。

表 21-6 接收缓冲寄存器 RBUF – 标准格式 (r-0)

地址	位								功能
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
RBUF	ID[7:0]								Identifier
RBUF+1	-					ID[10:8]			Identifier
RBUF+2	-								Identifier
RBUF+3	ESI								Identifier
RBUF+4	IDE=0	RTR	FDF	BRS	DLC[3:0]				Control
RBUF+5	KOER			TX					Status
RBUF+6	CYCLE_TIME[7:0]								TTCAN
RBUF+7	CYCLE_TIME[15:8]								TTCAN
RBUF+8	d1[7:0]								Data byte 1
RBUF+9	d2[7:0]								Data byte 2
…	…								…
RBUF+71	d64[7:0]								Data byte 64
RBUF+72	RTS[7:0]								CiA 603
…	…								…
RBUF+79	RTS[63:56]								CiA 603

表 21-7 接收缓冲寄存器 RBUF – 扩展格式 (r-0)

地址	位								功能
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
RBUF	ID[7:0]								Identifier
RBUF+1	ID[15:8]								Identifier
RBUF+2	ID[23:16]								Identifier
RBUF+3	ESI			ID[28:24]					Identifier
RBUF+4	IDE=0	RTR	FDF	BRS	DLC[3:0]				Control
RBUF+5	KOER			TX					Status
RBUF+6	CYCLE_TIME[7:0]								TTCAN
RBUF+7	CYCLE_TIME[15:8]								TTCAN
RBUF+8	d1[7:0]								Data byte 1
RBUF+9	d2[7:0]								Data byte 2
…	…								…
RBUF+71	d64[7:0]								Data byte 64
RBUF+72	RTS[7:0]								CiA 603
…	…								…
RBUF+79	RTS[63:56]								CiA 603

RBUF 寄存器（偏移 0x00 到 0x4f）指向 RB 中接收到最早消息所在的消息槽。所有 RBUF 寄存器都可以按任何顺序读取。

RBUF 中的 KOER 与寄存器 EALCAP 中的 KOER 位具有相同的含义。如果 RBALL=1，RBUF 中的 KOER 才有意义。

如果 CAN 控制器处于回环模式并且内核已接收到自己的传输帧，则 RBUF 中的状态位 TX 置 1。

这一特性在当 LBME=1 并且网络中的其他节点也进行传输时很有用。

时间戳 CYCLE_TIME 将仅在 TTCAN 模式下存储在 RBUF 中。参考帧的循环时间始终为 0。

CiA 603 时间戳的接收时间戳 (RTS) 储在 RBUF 地址范围末尾的每个接收到的消息中。因此，与 TTS 相比，RTS 与实际选择的 RBUF 消息槽有关。

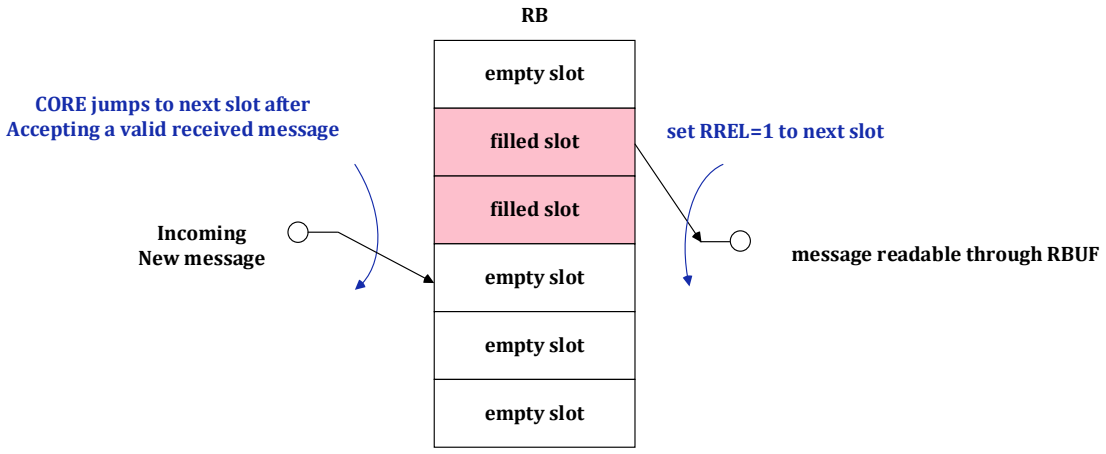


图 21-5 类似 FIFO 的 RB 示意图（6 个消息槽示例）

21.3.3.2 CAN_TBUF0~CAN_TBUF17 寄存器器

地址：0x4001_18D0~0x4001_1914

复位值：x

表 21-8 写入寄存器 CAN_TBUF0/1/2/3

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

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

位置	位名称	说明
[31:0]	WDATA	发送数据寄存器，写入数据到 Tx Buffer Slot 中（PTB 或者 STB）

不同帧的类型及数据长度不同，CAN_TBUF 的数据内容也不同。根据文档对发送帧的结构解析即可。

表 21-9 发送缓冲寄存器 TBUF 标准格式 (rw-u)

地址	位	功能

	7	6	5	4	3	2	1	0	
TBUF	ID[7:0]								Identifier
TBUF+1	-					ID[10:8]			Identifier
TBUF+2	-								Identifier
TBUF+3	TTSEN								Identifier
TBUF+4	IDE=0	Control	FDF	BRS	DLC[3:0]				Control
TBUF+8	d1[7:0]								Data byte 1
TBUF+9	d2[7:0]								Data byte 2
…	…								…
TBUF+71	d64[7:0]								Data byte 64

表 21-10 发送缓冲寄存器 TBUF 扩展格式 (rw-u)

地址	位置								功能
	7		5	4	3	2	1	0	
TBUF	ID[7:0]								Identifier
TBUF+1	ID[15:8]								Identifier
TBUF+2	ID[23:16]								Identifier
TBUF+3	TTSEN			ID[28:24]					Identifier
TBUF+4	IDE=1	Control	FDF	BRS	DLC[3:0]				
TBUF+8	d1[7:0]								Data byte 1
TBUF+9	d2[7:0]								Data byte 2
…	…								…
TBUF+71	d64[7:0]								Data byte 64

如果 TBSEL=1，TBUF 寄存器指向 STB 中的下一个空消息槽或指向 PTB。 TBUF 寄存器可以任意顺序访问。 对于 STB，需要设置 TSNEXT 来指示一个消息槽已满并令指针跳转到下一条消息槽。

为了更好地对齐地址段，TBUF 在 TBUF+5 到 TBUF+7 的寻址范围内留有地址间隙。间隙中的存储单元可以读写，但无实际意义。

TBUF 写入访问需要作为 32 位写入执行。

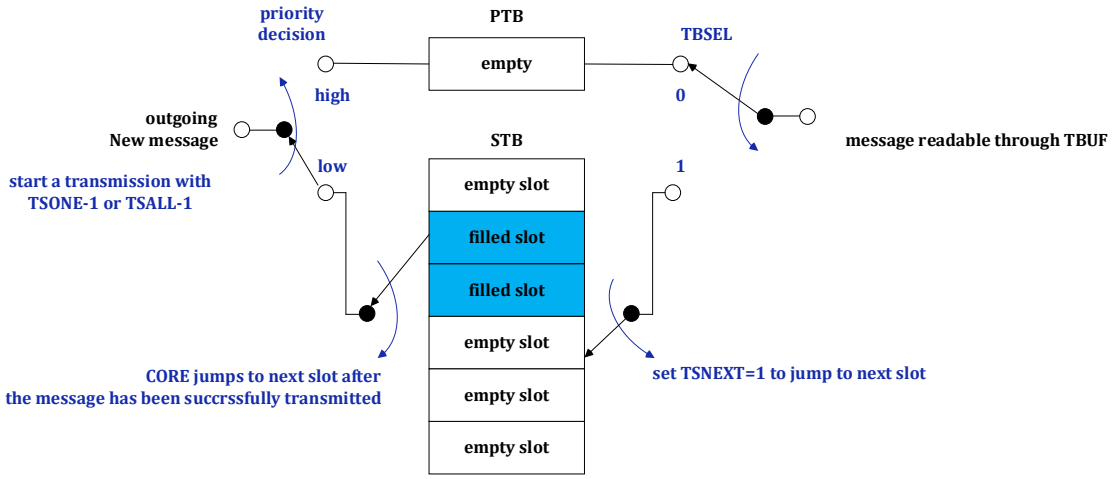


图 21-6 FIFO 模式下的 PTB 和 STB 示意图（6 个 STB 消息槽）

RBUF 和 TBUF 都包括一些单独的帧控制位。对于 RBUF，这些位指示的是接收帧的状态。对于 TBUF 这些位是待发送帧的控制位。

与 RTS 不同，如果 TTSEN=1，TTS 储的是最后一个发送帧的时间戳。 TTS 与实际选择的 TBUF 消息槽无关。

表 21-11 RBUF 和 TBUF 中的控制位

名称	描述
IDE	标识符扩展 0 – 标准格式：ID(10:0) 1 – 扩展格式：ID(28:0)
RTR	远程传输请求 0 – 数据帧 1 – 远程帧 只有 CAN 2.0 帧可以是远程帧。CAN FD 没有远程帧。因此，如果在 TBUF 和 RBUF 中 FDF =1，则 RTR 被强制为 0。 如果接收到 RRS=1 的 CAN FD 帧，则忽略该帧， RBUF 中的 RTR 被覆盖，但帧的 CRC 是用 RRS=1 计算的。
FDF	CAN FD 帧 0 – CAN 2.0 帧（最多 8 字节有效负载） 1 – CAN FD 帧（最多 64 字节有效载荷）
BRS	波特率开关 0 – 标准/慢速波特率帧 1 – 数据/快速波特率的数据有效载荷和 CRC 只有 CAN FD 帧可以切换波特率。因此，如果 FDF =0，BRS 被强制为 0。

ESI	错误状态指示 在 RBUF 中微只读状态位，在 TBUF 中不可用。 协议状态机自动将正确的 ESI 值嵌入到传输的帧中。ESI 仅包含在 CAN FD 帧中，不存在于 CAN 2.0 帧中。 0 – CAN 节点 error active 1 – CAN 节点 error passive 对于 CAN 2.0 帧，RBUF 中的 ESI 始终为低。 传输的错误状态由寄存器 ERRINT 中的位 EPASS 显示。
TTSEN	发送时间戳启用 对于 CiA 603 时间戳，可以在 TBUF 中选择传输时间戳 TTS： 0 – 未获取此帧的传输时间戳 1 – 启用 TTS 更新

RBUF 和 TBUF 中的数据长度代码 (DLC) 定义了有效载荷的长度——帧中的有效载荷字节数。有关详细信息，请参见表 21-12。

远程帧（仅适用于 FDF = 0 的 CAN 2.0 帧）始终以 0 有效载荷字节传输，但 DLC 的内容在帧头中传输。因此，可以将一些信息编码到远程帧的 DLC 位中。但是，如果允许不同的 CAN 节点传输具有相同 ID 的远程帧，则需要注意。在这种情况下，所有发射器都需要使用相同的 DLC，否则会产生无法解决的冲突。

表 21-12 DLC 的定义（根据 CAN 2.0 / FD 规范）

DLC（二进制）	框架类型	有效负载（以字节为单位）
0000~1000	CAN 2.0/CAN FD	0~8
1001~1111	CAN 2.0	8
1001	CAN FD	12
1010	CAN FD	16
1011	CAN FD	20
1100	CAN FD	24
1101	CAN FD	32
1110	CAN FD	48
1111	CAN FD	64

21.3.3.3 CAN_TTS0 发送时间戳寄存器 0

地址：0x4001_1918

复位值：0x0

表 21-13 发送时间戳寄存器 0 CAN_TTS0

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

R
0

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

位置	位名称	说明
[31:0]	TTS	64 位传输时间戳低 32 位 TTS 保存最后传输帧的时间戳。 如果 TTSEN=1，每个新帧都会覆盖 TTS。TTS 旨在供时间主机用于获取 SYNC 消息的时间戳

21.3.3.4 CAN_TTS1 发送时间戳寄存器 1

地址：0x4001_191C

复位值：0x0

表 21-14 发送时间戳寄存器 1 CAN_TTS1

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

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

位置	位名称	说明
[31:0]	TTS	64 位传输时间戳高 32 位 TTS 保存最后传输帧的时间戳。 如果 TTSEN=1，每个新帧都会覆盖 TTS。TTS 旨在供时间主机用于获取 SYNC 消息的时间戳

21.3.3.5 CAN_CFG_STAT 配置和状态寄存器

地址：0x4001_1920

复位值：0x80

表 21-15 配置和状态寄存器 CAN_CFG_STAT

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

	RESET	LBME	LBMI	TPSS	TSSS	RACTIVE	TACTIVE	BUSOFF
	RW	RW	RW	RW	RW	RO	RO	RW
	1	0	0	0	0	0	0	0

位置	位名称	说明
[7]	RESET	1：CAN 模块复位模式 0：CAN 模块正常工作模式 复位模式下，CAN 模块部分电路处于复位状态，部分寄存器只能在复位模式下配置；正常工作模式下，可以收发 CAN 帧。 注意，从复位模式切换到正常工作模式，需要等待 11 个 CAN Bit Time 的时间（例如波特率为 500K，一个 Bit Time 就是 2us，需要等待 22us 才能发送或者接收 CAN 帧）
[6]	LBME	外部回环模式使能位 1：使能 0：关闭 注意：正常通信中，禁止使能该位
[5]	LBMI	内部回环模式使能位 1：使能 0：关闭 注意：正常通信中，禁止使能该位
[4]	TPSS	PTB 单次传输模式使能位 1：使能 0：关闭
[3]	TSSS	STB 单次传输模式使能位 1：使能 0：关闭
[2]	RACTIVE	接收状态标志位 0：本 CAN 设备，接收空闲，总线空闲 1：本 CAN 设备，正在接收 CAN 帧
[1]	TACTIVE	发送状态标志位 0：本 CAN 设备，发送空闲 1：本 CAN 设备，正在发送 CAN 帧
[0]	BUSOFF	本 CAN 设备总线关闭状态位（Bus Off） 0：本 CAN 设备，不在总线关闭状态位 1：本 CAN 设备，处于总线关闭状态位 注意：BUSOFF 写 1 可以清零 TECNT 和 RECNT 寄存器，但仅限于项目调试

21.3.3.6 CAN_TCMD 发送命令寄存器

地址：0x4001_1921

复位值：0x0

表 21-16 发送命令寄存器 CAN_TCMD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								TBSEL	LOM	--	TPE	TPA	TSONE	TSALL	TSA
								RW	RW	--	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

位置	位名称	说明
[7]	TBSEL	Tx Buffer 选择位，通过 CAN_TBUF 写入到对应 Buffer 中 1：STB 0：PTB 注意在写入 CAN_TBUF 过程中，TBSEL 的值不能改变；同时，在写入 STB 的时候，完成一帧的写入，需要更新 TSNEXT。
[6]	LOM	监听模式使能位 1：使能 0：关闭 如果设置了 TPE、TSONE 或 TSALL，则无法设置 LOM。如果 LOM 启用且 LBME 禁用，则无法启动传输。 LOM=1 和 LBME=0 禁止所有传输。 LOM=1 和 LBME=1 禁止应答相应接收到的帧和错误帧，但可以发送本设备的帧 注意：正常通信中，禁止使能该位
[5]	--	保持默认值
[4]	TPE	PTB 发送使能位 1：触发 PTB 发送 0：不触发 PTB 发送 配置 TPE，在下一个可以发送的窗口，优先 STB 发送 PTB 的数据（除非当前已经在发送 STB，否则先发 PTB）。 该位写 1 后将保持为 1 直到 PTB 发送完成或者通过 TPA 取消发送。 软件不能通过写 0 清除该位。TPE 可以被清除的另外几种情况： 1. CAN_CFG_STAT.RESET=1 2. CAN_TCMD.LOM=1 同时 CAN_CFG_STAT.LBME=0
[3]	TPA	PTB 发送取消位 1：取消已经通过 TPE 置 1 请求但还未开始的 PTB 发送 0：不取消 软件置位，硬件自动清零。设置 TPA 会自动清零 TPE，一般 TPA 和

		TPE 不同时置位。CAN_CFG_STAT.RESET=1 可清零 TPA。
[2]	TSONE	发送一帧 STB 数据控制位（Transmit Secondary ONE Frame） 1：发送一帧 STB 数据 0：不发送 TSONE 配置为 1，触发发送，成功发送或者 TSA 终止发送可实现硬件清零，否则一直保持 1。CAN_CFG_STAT.RESET=1，也可清零。
[1]	TSALL	发送多帧 STB 数据控制位（Transmit Secondary All Frames） 1：发送多帧 STB 数据 0：不发送 TSALL 配置为 1，触发发送，成功发送或者 TSA 终止发送可实现硬件清零，否则一直保持 1。CAN_CFG_STAT.RESET=1，也可清零。
[0]	TSA	STB 发送取消位 1：取消已经通过 TSONE 或者 TSALL 置 1 请求但还未开始的 STB 发送，软件写 1，硬件清零。写 1 可以清零 TSONE 或者 TSALL 位 0：不取消 一般 TSA 和 TSONE/TSALL，不同时置位。CAN_CFG_STAT.RESET=1 可清零 TSA

21.3.3.7 CAN_TCTRL 发送控制寄存器

地址：0x4001_1922

复位值：0x10

表 21-17 发送控制寄存器 CAN_TCTRL

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								FD_ISO	TSNEXT	TSMODE	TTTBM	Recv.		TSSTAT	
								RW	RW	RW	RW	R		R	
								0	0	0	1	0		0	

位置	位名称	说明
[7]	FD_ISO	CAN FD ISO 模式 0 – 博世 CAN FD（非 ISO）模式 1 – ISO CAN FD 模式 (ISO 11898-1:2015) ISO CAN FD 模式具有不同的 CRC 初始化值和额外的填充位计数。两种模式不兼容，不能在一个 CAN 网络中混合使用。 该位对 CAN 2.0B 没有影响。该位仅在 RESET=1 时可写。
[6]	TSNEXT	更新 STB Slot 位置 1：更新

		0：不更新 当前的 STB Slot 填满数据后，软件置位 TSNEXT，硬件自动指向下一个 STB Slot，此时，TSONE/TSALL 置位，即可发送被填满的 STB Slot。可以同时设置 TSNEXT 和 TSONE 或 TSALL。如果 TBSEL=0，则设置 TSNEXT 没有意义，TSNEXT 的配置将被忽略并自动清除。如果 STB 的所有 Slot 都已填满，则 TSNEXT 将保持置位状态，直到 Slot 有空闲，才会被硬件清零。
[5]	TSMODE	发送缓冲器 STB 工作模式 0 - FIFO 模式 1 - 优先级模式 在 FIFO 模式下，帧按照写入 STB 的顺序传输。 在优先级模式下，STB 中优先级最高的帧会先发送。 帧的 ID 用于优先级判定。较小的 ID 意味着帧的优先级更高。PTB 中的帧始终具有最高优先级，与 ID 无关。 只有当 STB 为空时，才可以切换 TSMODE。
[4]	TTTBM	TTCAN 发送缓冲模式 如果 TTEN=0，则忽略 TTTBM，如果 TTEN=1，根据 TTTBM 不同值： 0 - 单独的 PTB 和 STB，由 TSMODE 定义缓冲区行为 1 - 完整的 TTCAN 支持：消息槽可由 TBPTR 和 TTPTR 选择 对于事件驱动的 CAN 通信 (TTEN=0)，系统提供 PTB 和 STB，STB 的行为由 TSMODE 定义，忽略 TTTBM 设定。 对于支持时间触发传输在内的所有功能的 TTCAN 通信 (TTEN=1)，需要选择 TTTBM=1。 此时使用 TTPTR 和 TBPTR 可寻址所有 TB 消息槽。 对于仅支持接收时间戳的时间触发 CAN 通信 (TTEN=1)，可以选择 TTTBM=0。发送缓冲器在事件驱动模式下工作，并且可以通过 TSMODE 选择缓冲区行为。 只有当 TBUF 为空时，才可以切换 TTTBM。
[3:2]	Resv.	
[1:0]	TSSTAT	STB 状态位 3：STB 满 2：STB 大于半满 1：STB 小于等于半满 0：STB 空

21.3.3.8 CAN_RCTRL 接收控制寄存器

地址：0x4001_1923

复位值：0x0

表 21-18 接收控制寄存器 CAN_RCTRL

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

	SACK	ROM	ROV	RREL	RBALL		RSTAT
	RW	RW	RW	RW	RW		RO
	0	0	0	0	0		0

位置	位名称	说明
[7]	SACK	自应答 1：LBME=1 时，使能自应答功能 0：无自应答
[6]	ROM	Rx Buffer 满，溢出控制位 1：新接收到的数据不被存储 0：最早接收到的数据被覆盖
[5]	ROV	Rx Buffer 溢出标志位 1：溢出，最少有一帧数据丢失。通过写 RREL 为 1 清零 0：无溢出
[4]	RREL	释放 Rx Buffer Slot 1：软件已经取走当前 Rx Buffer Slot 数据，可释放当前 Rx Buffer Slot，硬件指向下一个 Rx Buffer Slot 0：无释放操作
[3]	RBALL	接收 Rx Buffer 存储通过 ID 滤波的帧 1：存储所有的 CAN 帧，包括错误的帧 0：正常模式，仅存储正确的 CAN 帧
[2]	--	保持默认值
[1:0]	RSTAT	Rx Buffer 状态位 3：满（溢出状态下，保持此值） 2：Rx Buffer 已经接收的帧的数量大于等于 AFWL 的值，但未满 1：Rx Buffer 已经接收的帧的数量小于 AFWL 的值 0：Rx Buffer 已经接收的帧的数量为 0，空

21.3.3.9 CAN_RTIE 发送接收中断控制寄存器

地址：0x4001_1924

复位值：0x0

表 21-19 发送接收中断控制寄存器 CAN_RTIE

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

	RIE	ROIE	RFIE	RAFIE	TPIE	TSIE	EIE	TSFF
	RW	RW	RW	RW	RW	RW	RW	RW
	0	0	0	0	0	0	0	0

位置	位名称	说明
[7]	RIE	接收中断使能（Receive Interrupt Enable） 1：使能 0：禁止
[6]	ROIE	接收溢出中断使能（Receive Overrun Interrupt Enable） 1：使能 0：禁止
[5]	RFIE	Rx Buffer 满中断使能（Rx Buffer Full Interrupt Enable） 1：使能 0：禁止
[4]	RAFIE	Rx Buffer 将满中断使能（Rx Buffer Almost Full Interrupt Enable） 1：使能 0：禁止
[3]	TPIE	PTB 发送中断使能（Transmission Primary Interrupt Enable） 1：使能 0：禁止
[2]	TSIE	STB 发送中断使能（Transmission Secondary Interrupt Enable） 1：使能 0：禁止
[1]	EIE	错误中断使能（Error Interrupt Enable） 1：使能 0：禁止
[0]	TSFF	Tx Buffer 标志位 1：STB Slot 被全部填满 0：STB Slot 没有被全部填满

21.3.3.10 CAN_RTIF 发送接收中断标志寄存器

地址：0x4001_1925

复位值：0x0

表 21-20 发送接收中断标志寄存器 CAN_RTIF

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

	RIF	ROIF	RFIF	RAFIF	TPIF	TSIF	EIF	AIF
	RW	RW	RW	RW	RW	RW	RW	RW
	0	0	0	0	0	0	0	0

位置	位名称	说明
[7]	RIF	接收中断标志位（Receive Interrupt Flag） 1：接收到有效帧（数据帧或者远程帧），写 1 清零 0：没有接收到有效帧
[6]	ROIF	接收溢出中断标志位（Receive Overrun Interrupt Flag） 1：Rx Buffer 至少有一帧未读走的数据被覆盖 0：Rx Buffer 无覆盖 溢出时 ROIF 和 RFIF 同时置 1，写 1 清零
[5]	RFIF	Rx Buffer 满中断标志位（Rx Buffer Full Interrupt Flag） 1：Rx Buffer 满，写 1 清零 0：Rx Buffer 未满
[4]	RAFIF	Rx Buffer 将满中断标志位（Rx Buffer Almost Full Interrupt Flag） 1：被填充的 Rx Buffer Slot 数目大于等于 AFWL 设定值 0：被填充的 Rx Buffer Slot 数目小于 AFWL 设定值
[3]	TPIF	PTB 发送中断标志（Transmission Primary Interrupt Flag） 1：触发 PTB 发送，发送成功完成。写 1 清零 0：无 PTB 发送请求，无完成标志
[2]	TSIF	STB 发送中断标志（Transmission Secondary Interrupt Flag） 1：触发 STB 发送，发送成功完成。写 1 清零 0：无 STB 发送请求，无完成标志
[1]	EIF	错误中断标志（Error Interrupt Flag） 1：错误计数器的值发生变化，大于或者小于错误警告寄存器的设定值。写 1 清零 0：无 错误计数器的值从小于错误警告寄存器的设定值变为大于设定值，或者从大于设定值变为小于设定值，均会触发中断标志。另外，进入 Bus Off 或从 Bus Off 退出，也会触发。
[0]	AIF	取消发送中断标志（Abort Interrupt Flag） 1：通过 TPA 和 TSA 请求的发送消息被成功取消。写 1 清零 0：未取消发送数据 AIF 没有对应的使能寄存器

注：对应使能寄存器使能后，中断标志位才会置位；AIF 例外。

要重置中断标志，软件需要将 1 写入该标志。 写入 0 无效。如果在写访问处于活动状态时发生新的中断事件，则该事件将设置标志并覆盖复位。这确保不会丢失任何中断事件。只有在设置了相关的中断使能位时，才会设置中断标志。

21.3.3.11 CAN_ERRINT 错误中断使能和标志寄存器

地址：0x4001_1926

复位值：0x0

表 21-21 错误中断使能和标志寄存器 CAN_ERRINT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								EWARN	EPASS	EPIE	EPIF	ALIE	ALIF	BEIE	BEIF
								RW	R	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

位置	位名称	说明
[7]	EWARN	错误计数值超过门限中断标志位 1：RECNT 或者 TECNT 大于等于 EWL 设定值，写 1 清零 0：RECNT 或者 TECNT 小于 EWL 设定值
[6]	EPASS	CAN 设备处于被动错误状态 1：CAN 设备处于被动错误状态 0：CAN 设备处于主动错误状态
[5]	EPIE	被动错误中断使能（Error Passive Interrupt Enable） 1：使能 0：禁止
[4]	EPIF	被动错误中断标志（Error Passive Interrupt Flag） 1：发生主动错误到被动错误（或者相反）的改变，写 1 清零 0：未发生
[3]	ALIE	仲裁失败中断使能（Arbitration Lost Interrupt Enable） 1：使能 0：禁止
[2]	ALIF	仲裁失败中断标志位（Arbitration Lost Interrupt Flag） 1：仲裁失败，写 1 清零 0：仲裁成功
[1]	BEIE	总线错误中断使能（Bus Error Interrupt Enable） 1：使能 0：禁止
[0]	BEIF	总线错误中断标志（Bus Error Interrupt Flag） 1：总线错误，写 1 清零 0：无总线错误

要重置中断标志，软件需要向对应位写 1。 写入 0 无效。如果在写访问处于活动状态时发生新的中断事件，则该事件将令中断标志置位（写清零被覆盖）。这确保不会丢失任何中断事件。只有

在设置了相关的中断使能位时，才会令中断标志置位。

21.3.3.12 CAN_LIMIT 错误&警告门限值寄存器

地址：0x4001_1927

复位值：0x1B

表 21-22 错误&警告门限值寄存器 CAN_LIMIT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								AFWL				EWL			
								RW				RW			
								0x1				0xB			

位置	位名称	说明
[7:4]	AFWL	Rx Buffer 快满警告配置值 AFWL 同 Rx Buffer 实际接收到的消息数量进行比对，实际数量超过 AFWL，触发 RAFIF。 AFWL=0，无意义，硬件强制配置为 1；AFWL 超过 Rx Buffer 实际容量，无意义，硬件强制配置为 Rx Buffer 实际容量
[3:0]	EWL	可编程错误警告限值=(EWL+1)×8。可能的极限值：8，16，...128。EWL 的值控制 EIF。

21.3.3.13 CAN_SBAUD 慢速波特率配置寄存器

地址：0x4001_1928

复位值：0x1000

表 21-23 慢速波特率配置寄存器 CAN_ SBAUD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
S_PRESC									S_SJW						
RW									RW						
1									0						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	S_SEG_2							S_SEG_1							
	RW							RW							
	0							0							

位置	位名称	说明

[31:24]	S_PRESC	慢速时间量子（Time Quanta）TQ 预分频设定值（S_Prescaler），主要配置 TQ 的大小。 慢速 TQ = system clock period *(S_PRESC+1)
[23]	--	保持默认值
[22:16]	S_SJW	再同步补偿宽度时间设定（Bit Timing Segment 2） 再同步补偿宽度时间=(S_SJW+1)*TQ
[15]	--	保持默认值
[14:8]	S_SEG_2	慢速 bit 时间段 2 时间单元设定（Bit Timing Segment 2） 慢速 bit 时间段 2 时间=(S_SEG_2+1)*TQ
[7:0]	S_SEG_1	慢速 bit 时间段 1 时间单元设定（Bit Timing Segment 1） 慢速 bit 时间段 1 时间=(S_SEG_1+2)*TQ

21.3.3.14CAN_FBAUD 快速波特率配置寄存器

地址：0x4001_192C

复位值：0x1000

表 21-24 快速波特率配置寄存器 CAN_ FBAUD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
F_PRESC												F_SJW			
RW												RW			
1												0			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				F_SEG_2								F_SEG_1			
				RW								RW			
				0								0			

位置	位名称	说明
[31:24]	F_PRESC	快速时间量子 TQ 预分频率设定值（F_Prescaler），主要配置 TQ 的大小。 快速 TQ = system clock period *(F_PRESC+1)
[23]	--	保持默认值
[22:16]	F_SJW	再同步补偿宽度时间设定（Bit Timing Segment 2） 再同步补偿宽度时间=(F_SJW+1)*TQ
[15]	--	保持默认值
[3:0]	F_SEG_2	快速 bit 时间段 2 时间单元设定（Bit Timing Segment 2） 快速 bit 时间段 2 时间=(F_SEG_2+1)*TQ
[4:0]	F_SEG_1	快速 bit 时间段 1 时间单元设定（Bit Timing Segment 1） 快速 bit 时间段 1 时间=(F_SEG_1+2)*TQ

21.3.3.15 CAN_EALCAP 错误信息和丢失仲裁信息记录寄存器

地址：0x4001_1930

复位值：0x0

表 21-25 错误信息和丢失仲裁信息记录寄存器 CAN_EALCAP

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

位置	位名称	说明
[7:5]	KOER	错误类别（Kind Of Error） 0：无错误 1：位错误 2：形式错误 3：填充错误 4：应答错误 5：CRC 错误 6：其他错误 7：保留 有错误时 KOER 位更新，正常发送接收时 KOER 位保持不变。
[4:0]	ALC	仲裁失败位置捕捉（Arbitration Lost Capture） 仲裁失败时 ALC 记录一帧数据仲裁失败时的 bit 位置

21.3.3.16 CAN_TDC 发送延迟补偿寄存器

地址：0x4001_1931

复位值：0x0

表 21-26 发送延迟补偿寄存器 CAN_TDC

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

位置	位名称	说明
[7]	TDCEN	发送延迟补偿使能 如果 TDCEN=1，BRS 处于活动状态，则 TDC 将在 CAN FD 帧的数据阶

		段被激活。有关 TDC 的更多详细信息，请参见 21.7.6。
[6:0]	SSPOFF	次采样点偏移 发送延迟加上 SSPOFF 定义了 TDC 的辅助采样时间点。SSPOFF 的计数单位为 TQ。

仅当 CAN_CFG_STAT.RESET=1 时，才能写入 S_Seg_1、S_Seg_2、S_SJW、S_PRESC、F_Seg_1、F_Seg_2、F_SJW、F_PRESC 和 TDC。

其中的所有时序参数均针对慢速（前缀“S_”）和快速（前缀“F_”）给出。慢速用于 CAN 2.0 和 CAN FD 仲裁阶段。快速用于 CAN FD 数据阶段。

21.3.3.17 CAN_RECNT 接收错误计数器寄存器

地址：0x4001_1932

复位值：0x0

表 21-27 接收错误计数器寄存器 CAN_ RECNT

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

位置	位名称	说明
[7:0]	RECNT	接收错误计数器（Receive Error Count） 接收错误计数器根据 CAN 协议规定的错误计数增加或者减少。该计数器不存在上溢，255 为最大值

21.3.3.18 CAN_TECNT 发送错误计数器寄存器

地址：0x4001_1933

复位值：0x0

表 21-28 发送错误计数器寄存器 CAN_ TECNT

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

位置	位名称	说明
[7:0]	TECNT	发送错误计数器（Transmit Error Count） 发送错误计数器根据 CAN 协议规定的错误计数增加或者减少。该计数器不存在上溢，255 为最大值

21.3.3.19 CAN_ACFCTRL ID 过滤器控制寄存器

地址：0x4001_1934

复位值：0x0

表 21-29 ID 过滤器控制寄存器 CAN_ ACFCTRL

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

位置	位名称	说明
[7:6]	--	保持默认值
[5]	SELMASK	选择 ID 过滤器的屏蔽寄存器（Select Acceptance MASK） 1：ACF 指向 ID 过滤器掩码 AMR 寄存器（MASK） 0：ACF 指向 ID 过滤器 ACR 寄存器 通过 ACFADR 选择具体的筛选寄存器组
[4]	--	保持默认值
[3:0]	ACFADR	ID 过滤器地址（Acceptance Filter Address） ACFADR 指向具体的 ID 过滤器器，通过 SELMASK 去区分 ACR 和 AMR。 0：指向 ACF_0 1：指向 ACF_1 2：指向 ACF_2 3：指向 ACF_3 4：指向 ACF_4 5：指向 ACF_5 6：指向 ACF_6 7：指向 ACF_7 8：指向 ACF_8 9：指向 ACF_9 10：指向 ACF_10 11：指向 ACF_11 12：指向 ACF_12 13：指向 ACF_13 14：指向 ACF_14

		15：指向 ACF_15

通过接收过滤器寄存器 CAN_ACF 访问接收过滤器选通 ACODE_x 和接收过滤器掩码 AMASK_x 时，取决于 SELMASK 的设置。仅当 CAN_CFG_STAT.RESET=1 时才能对 CAN_ACF 进行写访问。

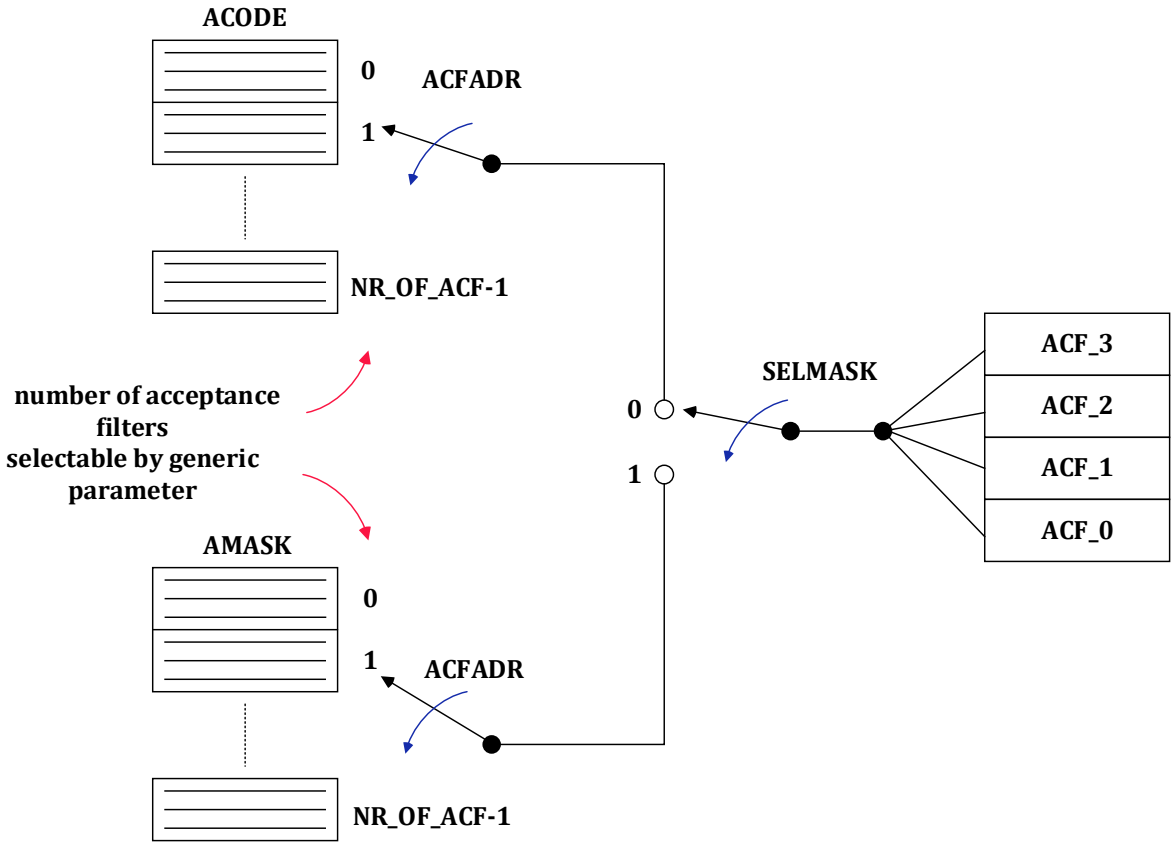


图 21-7 接收过滤器

21.3.3.20 CAN_TIMECFG CiA603 时间戳配置寄存器

地址：0x4001_1935

复位值：0x2

表 21-30 CiA603 时间戳配置寄存器 CAN_TIMECFG

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

位置	位名称	说明

[7:2]	--	保持默认值
[1]	TIMEPOS	时间戳位置 0 – SOF 1 – EOF TIMEPOS 只能在 TIMEEN=0 时更改，但可以在设置 TIMEEN=1 的同时修改 TIMPOS。
[0]	TIMEEN	时间戳使能 0 – 禁用 1 – 启用

21.3.3.21 CAN_ACFEN ID 过滤器使能寄存器

地址：0x4001_1936

复位值：0x1

表 21-31 ID 过滤器使能寄存器 CAN_ACFEN

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AE_15	AE_14	AE_13	AE_12	AE_11	AE_10	AE_9	AE_8	AE_7	AE_6	AE_5	AE_4	AE_3	AE_2	AE_1	AE_0
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	1

位置	位名称	说明
[31:16]		未使用
[15]	AE_15	ACF_15 使能（Acceptance Filter 15 Enable） 1：使能 0：禁止
[14]	AE_14	ACF_14 使能（Acceptance Filter 14 Enable） 1：使能 0：禁止
[13]	AE_13	ACF_13 使能（Acceptance Filter 13 Enable） 1：使能 0：禁止
[12]	AE_12	ACF_12 使能（Acceptance Filter 12 Enable） 1：使能 0：禁止
[11]	AE_11	ACF_11 使能（Acceptance Filter 11 Enable） 1：使能 0：禁止
[10]	AE_10	ACF_10 使能（Acceptance Filter 10 Enable） 1：使能

		0：禁止
[9]	AE_9	ACF_9 使能（Acceptance Filter 9 Enable） 1：使能 0：禁止
[8]	AE_8	ACF_8 使能（Acceptance Filter 8 Enable） 1：使能 0：禁止
[7]	AE_7	ACF_7 使能（Acceptance Filter 7 Enable） 1：使能 0：禁止
[6]	AE_6	ACF_6 使能（Acceptance Filter 6 Enable） 1：使能 0：禁止
[5]	AE_5	ACF_5 使能（Acceptance Filter 5 Enable） 1：使能 0：禁止
[4]	AE_4	ACF_4 使能（Acceptance Filter 4 Enable） 1：使能 0：禁止
[3]	AE_3	ACF_3 使能（Acceptance Filter 3 Enable） 1：使能 0：禁止
[2]	AE_2	ACF_2 使能（Acceptance Filter 2 Enable） 1：使能 0：禁止
[1]	AE_1	ACF_1 使能（Acceptance Filter 1 Enable） 1：使能 0：禁止
[0]	AE_0	ACF_0 使能（Acceptance Filter 0 Enable） 1：使能 0：禁止 Filter 0 默认开启

每个接收过滤器 (AMASK / ACODE) 可以单独启用或禁用。硬件复位后，默认仅启用 Filter 0。禁用的过滤器拒绝接收任何消息。只有启用的过滤器进行适当的 AMASK/ACODE 配置才能接收消息。

要接收所有消息，须设置 AE_x=1 来启用一个过滤器 x，同时 AMASK_x=0xFF 和 ACODE_x=0x00。这是 Filter 0 硬件复位后的默认配置，而所有其他过滤器都被禁用。

21.3.3.22 CAN_ACF ID 过滤器寄存器

地址：0x4001_1938

复位值：0x0

表 21-32 ID 过滤器寄存器 CAN_ACF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	IDCMP	IDMASK	ID ACR [28:16] or AMR [28:16]												
	RW	RW	RW												
	0	0	0												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID ACR [15:00] or AMR [15:00]															
RW															
0															

位置	位名称	说明
[31]	--	保持默认值
[30]	IDCMP	SELMASK=1 时有效，ID AMR（MASK）选择作用范围 1：ID 过滤器只接收标准帧或者扩展帧，由 IDMASK 进一步指定 0：ID 过滤器接收标准帧和扩展帧两种格式
[29]	IDMASK	IDCMP 位为 1 时，MASK 选择哪种帧格式 1：ID 过滤器仅接收扩展帧格式 0：ID 过滤器仅接收标准帧格式
[28:0]	ID CODE/MASK	CAN_ACFCTRL.ACFADR 选择具体的 ID 过滤器组 SELMASK=0，表示 ID 过滤器选择的是 ID ACR SELMASK=1，表示 ID 过滤器选择的是 ID AMR 标准帧格式时 ID ACR/AMR 的范围是 Bit10-Bit0，扩展帧格式时的范围是 Bit28-Bit0

21.3.3.23 CAN_TBSLOT TTCAN 发送缓冲区指针寄存器

地址：0x4001_193e

复位值：0x0

表 21-33 TTCAN 发送缓冲区指针 CAN_TBSLOT

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

位置	位名称	说明
[7]		将 PTB/STB 消息槽设置为“空”

	TBE	1 - 将 TBPTR 选择的消息标记为“空的” 0 - 无动作 一旦消息槽被标记为空且 TSFF=0，TBE 就会自动重置为 0。如果消息正在被发送，则只要发送完成或在传输错误或仲裁丢失之后传输不再进行，TBE 就会保持置位状态。 如果 TBF 和 TBE 同时被设置为 1，则 TBE 优先。
[6]	TBF	将 TB 消息槽设置为“已填充” 1 - TBPTR 选择的消息应标记为 “填充” 0 - 无动作 一旦消息槽被标记为已满且 TSFF=1，TBF 就会自动重置为 0。如果 TBF 和 TBE 同时被设置为 1，则 TBE 优先。
[5:0]	TBPTR	指向 TB 消息的指针。 0x00 - 指向 PTB 其他 - 指向 STB 中的消息槽 TBPTR 指向的消息槽可以使用 TBUF 寄存器读取/写入。只有在 TSFF=0 时才能进行写访问。将 TBF 设置为 1 将所选消息槽标记为已填充，将 TBE 设置为 1 将所选消息槽标记为空。 TBSEL 和 TSNEXT 在 TTCAN 模式下无意义。 TBPTR 仅限于 PTB 和 STB 消息槽。如果 TBPTR 太大并指向一个不可用的消息槽，则 TBF 和 TBE 将自动重置并且不执行任何操作。

21.3.3.24 CAN_TTCFG TTCAN 配置寄存器

地址：0x4001_193f

复位值：0x0

表 21-34 TTCAN 配置寄存器 CAN_TTCFG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								WTIE	WTIF	TEIF	TTIE	TTIF	T_PRESC		TTEN
								RW	RW	RW	RW	RW	RW		RW
								1	0	0	1	0	0		0

位置	位名称	说明
[7]	WTIE	超时监测触发中断使能
[6]	WTIF	超时监测触发中断标志 如果循环计数达到由 TT_WTRIG 定义的门限值并且 WTIE=1，则 WTIF 置位
[5]	TEIF	触发错误中断标志

		TEIF 始终使能
[4]	TTIE	时间触发中断使能 如果设置 TTIE=1，则如果循环计数等于触发时间 TT_TRIG，则 TTIF 置位。
[3]	TTIF	时间触发中断标志 如果设置 TTIE=1 并且循环计数等于触发时间 TT_TRIG，则 TTIF 置位。向 TTIF 写入 1 清零。写 0 无影响。 TTIF 将只设置一次。如果 TT_TRIG 没有更新，则在下一个基本周期中不会再次设置 TTIF。
[2:1]	T_PRESC	TTCAN 定时器预分频设置 00 - 1 01 - 2 10 - 4 11 - 8 TTCAN 时基是由 S_PRES、S_SEG_1 和 S_SEG_2 定义的 CAN bit 时间。使用 T_PRESC 定义了额外的预分频因子 1、2、4 或 8。 T_PRESC 只能在 TTEN=0 时修改，但可以在修改 T_PRESC 的同时设置 TTEN=1。
[0]	TTEN	TTCAN 启用 1 - TTCAN 启用，循环计数定时器开启计数 0 - 禁用

要重置中断标志，软件需要写入 1。写入 0 无效。如果在写访问处于活动状态时发生新的中断事件，则该事件将置位中断标志（写清零被覆盖）。这确保了不会丢失任何中断事件。

在设置了相关的中断使能位时，才会设置中断标志。

21.3.3.25 CAN_REF_MSG TTCAN 参考帧寄存器

地址：0x4001_1940

复位值：0x0

表 21-35 TTCAN 参考帧寄存器 CAN_REF_MSG

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

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

位置	位名称	说明
[31]	REF_IDE	参考帧 IDE 位

[30:29]		
[28:0]	REF_ID	参考帧标识符。如果 REF_IDE 是 1 - REF_ID(28:0) 有效（扩展 ID） 0 - REF_ID(10:0) 有效（标准 ID） REF_ID 在 TTCAN 模式下用于检测参考帧。这适用于时间从站（接收）以及时间主站（传输）。如果检测到参考帧并且没有错误，则该帧的 Sync_Mark 将成为 Ref_Mark。 REF_ID(2:0)强制为 0。 CAN 控制器仅通过 ID 识别参考帧。有效载荷未经测试。附加说明：时间主机发送参考帧的方式和正常帧相同。 REF_ID 用于检测参考帧的成功传输。

表 21-4 给出了 REF_MSG_0 到 REF_MSG_2 内 REF_ID 和 REF_IDE 的 bit 位置定义。

对 CAN_REF_MSG 的写访问将参考帧同步到 CAN 时钟域。在此期间对 CAN_REF_MSG 的再次写操作被禁用。

21.3.3.26 CAN_TRG_CFG TTCAN 触发配置寄存器

地址：0x4001_1944

复位值：0x0

表 21-36 TTCAN 触发配置寄存器 CAN_TRG_CFG

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

位置	位名称	说明
[15:12]	TEW	发送使能窗口 对于单次发射触发，允许帧开始的循环时间最多有 16 个 cycle。TWE+1 定义了 cycle 数。见章节 TTCAN Timin。TEW=0 将传输启用窗口缩短到 1 个 cycle。
[11]		保留位
[10:8]	TTYPE	触发类型 000 - 立即触发发送 001 - 定时触发 010 - 单次发射发送，针对独家时间窗口 011 - 传输开始触发，针对合并仲裁时间窗口 100 - 传输停止触发，针对合并仲裁时间窗口 其他- 无动作 触发时间由 TT_TRIG 定义。TTPTR 选择 TB 中的消息。
[7:6]		保留位

[5:0]	TT_PTR	TB 发送消息指针 如果 TTPTR 太大并指向不可用的消息槽，则 TEIF 置位，并且在对 TT_TRIG 进行写访问后无法产生新的触发。 如果 TTPTR 指向一个空槽，那么 TEIF 将在触发发送时被设置。

21.3.3.27 CAN_TT_TRIG TTCAN 触发时刻寄存器

地址：0x4001_1946

复位值：0x0

表 21-37 TTCAN 触发时刻寄存器 CAN_TT_TRIG

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

位置	位名称	说明
[15:0]	TT_TRIG	触发时间 TT_TRIG 定义了触发时间。对于发送触发，相应帧的 SOF 的最早传输点将是 TT_TRIG+1。

对 TT_TRIG 的写操作会将触发相关配置同步到 CAN 时钟域。如果触发正处于活动状态，则所有寄存器 TRIG_CFG、TT_TRIG 的写入都被禁止。当达到触发时间（TTIF 置位）或检测到错误（TEIF 置位）时，写锁定解除。

21.3.3.28 CAN_TT_WTRIG TTCAN 超时寄存器

地址：0x4001_1948

复位值：0x0

表 21-38 TTCAN 超时寄存器 CAN_TT_WTRIG

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

位置	位名称	说明
[15:0]	TT_WTRIG	超时触发时间 TT_WTRIG 定义了超时触发周期。初始值是最大周期时间 0xFFFF。

对 TT_WTRIG 的写操作触发寄存器值同步到 CAN 时钟域，耗时 6 个 CAN 时钟。在此期间，寄存器 TT_WTRIG 禁止再次写入。

21.3.3.29 CAN_CIAWDAT0 CiA603 时间戳低 32 位写入值寄存器

地址：0x4001_1950

复位值：0

表 21-39 CiA603 时间戳低 32 位写入值寄存器 CAN_CIAWDAT0

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

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

位置	位名称	说明
[31:0]	WDATA	CiA603 时间戳低 32 位写入值

21.3.3.30 CAN_CIAWDAT1 CiA603 时间戳高 32 位写入值寄存器

地址：0x4001_1954

复位值：0

表 21-40 CiA603 时间戳高 32 位写入值寄存器 CAN_CIAWDAT1

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

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

位置	位名称	说明
[31:0]	WDATA	CiA603 时间戳高 32 位写入值

21.4 一般操作

本章介绍 CAN 通信操作。在进行通信之前，须配置 CAN 控制器以匹配 CAN 总线时序。

21.4.1 接收过滤器

为了减少 CPU 接收帧的负载，可以配置 CAN 控制器使用接收过滤器。 CAN 控制器在接收过滤期间检查消息标识符。因此，每个接收过滤器的长度为 29bit。

如果消息通过其中一个过滤器，那么它将被接收，并将存储到 RB 中，如果 RIE=1，则 RIF 被置位。如果消息未被接收，则不设置 RIF 并且不增加 RB FIFO 指针。未被接收的消息将被丢弃并被下一条消息覆盖。没有存储的有效消息将被后续消息覆盖。

独立于接收过滤的结果，CAN 控制器检查总线上的每条消息并向总线发送确认或错误帧。

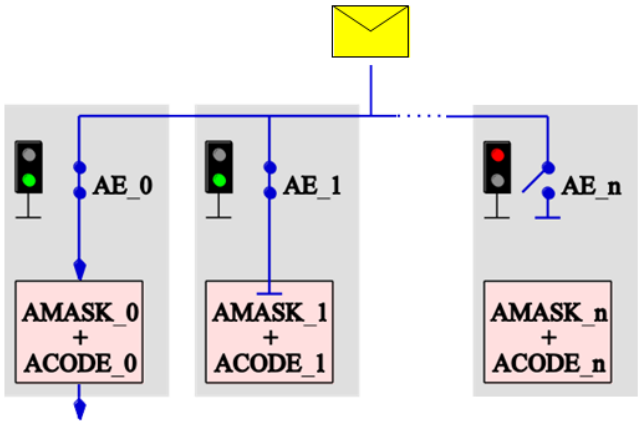


图 21-8 接收过滤示例

接收掩码定义了需要将过滤器的哪些位和接收消息 ID 进行比较判断。掩码位设置为 0，则接收 ID 与相应的消息标识符位进行比较。掩码位设置为 1 则不进行接收检查，从而接收消息。

ID 与相应的 ACODE 进行比较，如下所示：

- 标准帧： ID(10:0) 与 ACODE(10:0)
- 扩展帧： ID(28:0) 与 ACODE(28:0)

示例：如果 AMASK_x(0)=0 且所有其他 AMASK_x 位为 1，则最后一个 ID 位的值必须等于 ACODE(0) 才能接收消息。过滤器忽略所有其他 ID 位。

图 21-8 给出了一个使用多个过滤器的接收过滤示例。在此示例中，过滤器 0 和 1 由 ACF_EN_x 寄存器中的相应 AE_x 位启用。所有其他过滤器都被禁用，因此不接收任何消息。对于启用的过滤器，AMASX_x 和 ACODE_x 的组合定义了消息是被接收（如过滤器 0）还是不被接收（如过滤器 1）。

注意：通过设置 AE_x=0，被禁用过滤器会阻止消息被接收。相反，设置 AMASK_x 中的掩码位 =1 会禁用对该位的检查，从而允许接收消息。

AMASK 和 ACODE 的定义不区分标准帧或扩展帧。如果 CAN_ACF.IDCMP=1，则接收帧类型由 CAN_ACF.IDMASK 决定。如果 CAN_ACF.IDCMP=0 则两种类型帧都被接收。

上电复位后，CAN 控制器被配置为接收所有消息。上电后复位后，过滤器 0 由 AE_0=1 启用，AMASK_0 中的所有位都设置为 1 并且 CAN_ACF.IDCMP=0。所有其他过滤器都被禁用。过滤器 0 是唯一为 AMASK / ACODE 定义了复位值的过滤器，而所有其他过滤器都未定义复位值。

21.4.2 消息接收

接收到的数据将存储在 RB 中。RB 具有类似 FIFO 的行为。如果启用 RIE，则每条接收到的有效消息都会设置 RIF=1。 RSTAT 根据填充状态设置。当填充缓冲区的数量等于可编程值 AFWL 时，如果设置了 RAFIE=1，则 RAFIF 置位。如果所有缓冲区都已满，若设置了 RFIE=1，则 RFIF 置位。

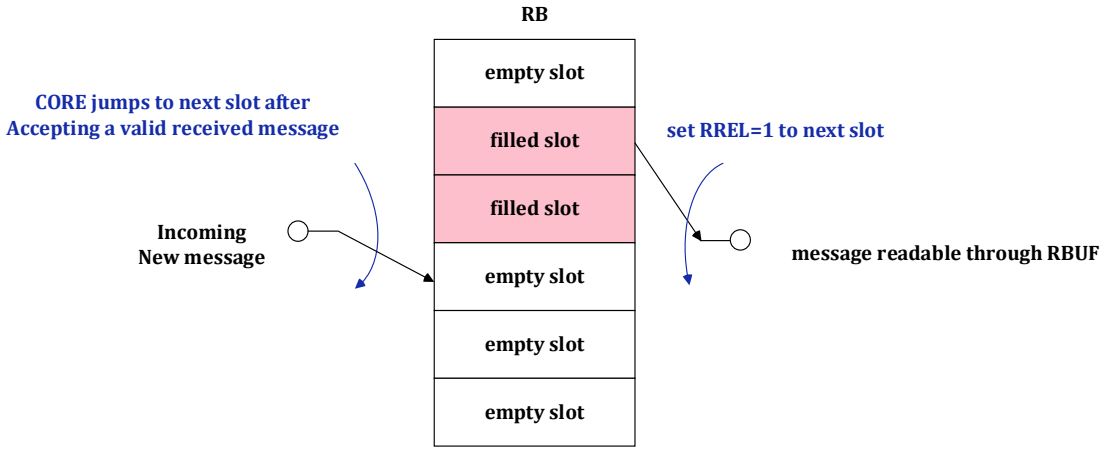


图 21-9 类似 FIFO 的 RB 示意图（6 个消息槽的示例）

RB 总是将最早消息映射到 RBUF 寄存器。

CAN 2.0 消息的最大有效负载长度为 8 个字节，CAN FD 消息为 64 个字节。每条消息的数据长度由 DLC 定义。RB 为每个消息提供了消息槽，并且主机软件需要设置 RREL 以跳转到下一个 RB 消息槽。RBUF 的所有字节都可以按任何顺序读取。

如果 RB 已满，则将临时存储下一个传入的消息。然后如果 ROM=0，最早的消息将被最新的消息覆盖；如果 ROM=1，则最新的消息将被丢弃。如果启用 ROIE，则在这两种情况下都会导致 ROIF 置位。如果主机软件读取了一条消息，并在新消息传入之前设置 RREL，则不会有消息被覆盖丢失。

如果没有接收过滤，CAN 控制器每接收到每一帧，都会告知 CPU。这将导致 CPU 的负载相当大。

可以通过禁用中断并使用接收过滤器来减少主机控制器的负载。如果启用 RIE 并且 CAN 控制器已接收到有效消息，则 RIF 被设置为 1。为了减少接收中断的数量，可以使用 RAIE/RAIF（RB 几乎满中断）或 RFIE / RFIF（RB 满中断）代替 RIE / RIF（接收中断）。 “RB 几乎满门限”可使用 AFWL 进行编程。

读取 RB 的步骤如下：

1. 使用RBUF寄存器从RB FIFO中读取最早的消息。
2. 设置RREL=1以释放RB消息槽。硬件会选择下一条消息（下一个FIFO槽）。 RBUF 将会被自动更新。
3. 重复以上操作，直到RSTAT指示RB空。

如果 RB FIFO 已满并且新接收到的消息被识别为有效（6th EOF），则将丢失一个消息。为了让主机控制器有足够的时间，能够在 RB FIFO 被填满且发生中断之后从 RB 读取至少一帧，RB 包含一个（隐藏）消息槽用于接收消息，验证它并在溢出发生之前检查它是否与接收过滤器匹配。

21.4.3 消息发送

在开始任何发送之前，至少有一个发送缓冲区（PTB 或 STB）已加载消息。TPE 指示 PTB FIFO 空满，TSSTAT 指示 STB 的填充状态。TBUF 寄存器提供对 PTB 和 STB 的访问。以下是推荐的编程流程：

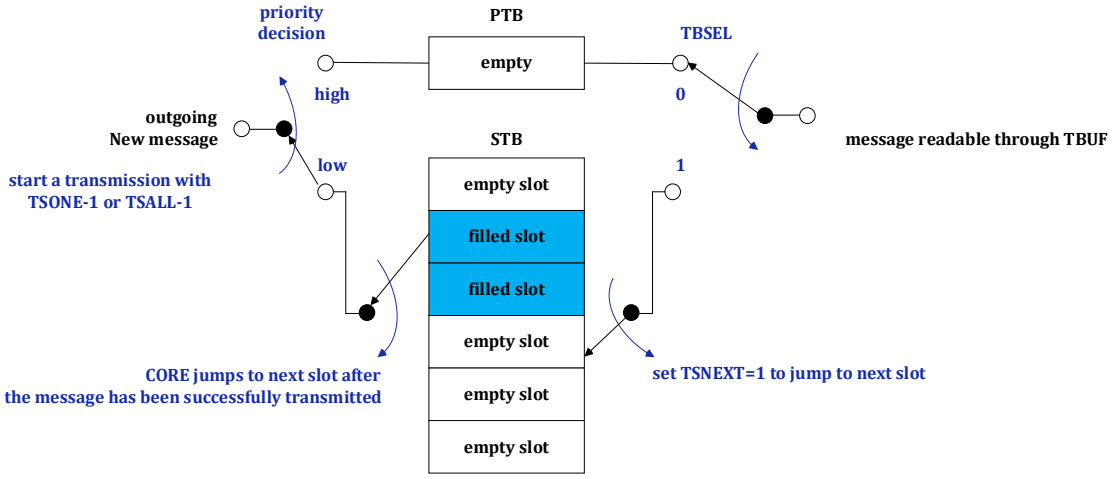


图 21-10 FIFO 模式下的 PTB 和 STB 示意图（空 PTB 和 6 个 STB 消息槽）

CAN 2.0 消息的最大有效负载长度为 8 个字节，CAN FD 消息为 64 个字节。每条消息的长度由 DLC 定义。对于远程帧（RTR=1），DLC 无意义，因为远程帧的数据长度始终为 0 字节。主机控制器需要设置 TSNEXT 以跳转到下一个 STB 消息槽。所有 TBUF 字节都可以按任何顺序写入。

如果 TBSEL=0 选择 PTB，则设置 TSNEXT=1 没有意义。在这种情况下，TSNEXT 会自动清零无其他影响。

使用 PTB 时应设置 TPE=1 启动传输。要使用 STB，须设置 TSONE=1 开始传输单个消息或设置 TSALL=1 以传输 STB 中的所有消息。

PTB 的优先级总是高于 STB。无论帧标识符如何，PTB 消息总是首先发送。如果来自 STB 的传输已经处于活动状态，它将在来自 PTB 的消息发送之前完成。在 PTB 传输完成或中止后，CAN 控制器返回处理来自 STB 的其他消息。

发送完成后，设置以下传输中断：

对于 PTB，如果 TPIE=1，则 TPIF 置位

对于 STB，如果 TSONE=1，一条消息已完成且 TSIE=1，则 TSIF 置位。

对于 STB，如果 TSONE=1，所有消息都已完成且 TSIE=1，则 TSIF 置位。换句话说：如果 STB 为空，则设置 TSIE。如果主机控制器在 TSALL 传输开始后向 STB 写入附加消息，则附加消息也将在 TSIF 设置之前进行传输。

21.4.4 消息发送中止

实际应用中存在这种情况，传输缓冲区中的消息由于其低优先级而很久无法发送。为了避免这种情况发送，如果传输尚未开始，主控制器可以通过分别设置 TPA=1 或 TSA=1 来撤回发送请求。

设置 TPA 和 TSA 都将产生一个中断标志： AIF。 CAN 协议状态机只有在没有向 CAN 总线传输任何内容时才会执行中止。

- 总线仲裁期间不能进行发送中止。
  - 如果节点仲裁失败，则后续执行中止。
  - 如果节点赢得仲裁，则该帧将被发送。
- 帧传输时不会进行发送中止。
  - 如果一个帧被成功传输，则发送成功中断标志将通知CPU。在这种情况下，不会发生发送中止。
  - 在CAN节点没有收到ACK的情况下，属于传输不成功，错误计数器会增加，可以执行传输中止。
  - 如果STB中至少还剩一帧，而主机已下令发送所有帧（TSALL=1），则将产生帧完成事件和中止事件。

中止传输可能需要一些时间，具体取决于 CAN 通信速度和帧长度。如果执行中止，则会导致以下动作：

- TPA释放PTB导致TPE=0。
- 释放PTB后，帧数据仍保存在PTB中。
- TSA释放STB的单个消息槽或所有消息槽。这取决于是使用TSONE还是TSALL来启动传输。TSSTAT将被更新。 释放STB中的消息将导致消息丢失，因为主机软件无法再访问此消息。

不建议同时设置 TPA 和 TSA。如果主机软件这样做，则 AIF 置位，来自 PTB 和 STB 的传输都将被中止。如前所述，如果在执行中止之前完成一次传输，这将导致发送成功传输标志置位。因此，如果启用，则以下中断标志置位：

- AIF（PTB和STB传输中止）
- TPIF + AIF
- TSIF + AIF
- TPIF + TSIF（非常少见，只有在主机不立即处理TPIF时才会发生）
- TPIF + TSIF + AIF （非常很少，只有在主机不立即处理TPIF和TSIF时才会发生）

要清空整个 STB，可以设置 TSALL 和 TSA。为了检测消息是否因为失去仲裁而无法长时间发送，主机可以使用 ALIF/ALIE。

21.4.5 STB 空满

写入 STB 后，设置 TSNEXT=1 可以将当前缓冲区消息槽标记为已满，并跳转到下一个空闲消息槽。 TSNEXT 在此操作后由 CAN 控制器自动复位为 0。

如果最后一个消息槽已被填满，所有消息槽都被占用，则 TSNEXT 保持为 1，直到有新的消息槽空闲。当 TSNEXT=1 时，写入 TBUF 被 CAN 控制器禁止。

当消息槽空闲时，CAN 控制器自动将 TSNEXT 重置为 0。如果 STB 的帧成功发送或主机请求中止 (TSA=1)，则消息槽变为空闲。如果 TSALL 传输被中止，则 TSNEXT 也被复位，并将整个 STB 标记为空。

21.4.6 错误处理

通常，CAN 控制器可以自动进行错误处理，这意味着在大多数情况下主机控制器无须关心错误，包括自动重传消息和自动删除收到的错误消息。另一方面，如果主机需要，CAN 控制器可以选择性地提供有关错误的详细信息，并通过中断向主机发送每个错误信号。使得能够在主机上运行诸如 CAN 总线监视器之类的应用程序。

每个 CAN 节点都有 3 种错误处理状态:

1. Error-Active：节点在检测到错误时自动发送活动错误帧。
2. Error-Passive：节点在检测到错误时传输被动错误帧，即它不会将显性值传输到总线。
3. 总线关闭：在出现太多错误后，节点进入“总线关闭”状态，停止访问CAN总线。

为了处理这 3 种错误状态，每个 CAN 节点都有两个错误计数器：发送错误计数器和接收错误计数器。两者都按照 CAN 规范的定义递增和递减，并且在节点达到 CAN 规范定义的计数值时进入适当的错误状态。

如果有错误，错误计数器会增加。由该节点引起的严重错误将导致计数器加 8，由其他节点引起的错误将导致计数器加 1。有效帧发送或有效接收会令计数器减 1。所有这些都由 CAN 规范定义并由 CAN 控制器自动处理。

错误帧（或者更贴切的名称为：错误标志）与数据帧不同。它是至少 6 个连续显性位的显性脉冲，这对其他节点来说是填充位违规。主机控制器软件不能主动发送错误帧。它由 CAN 控制器全自动完成。

如果 CAN 控制器收到指令需要传输一帧，那么它会尝试传输并反复地自动重传，直到该帧被正确传输或节点关闭总线。如果接收到帧并且 CAN 控制器检测到错误，则丢弃接收到的数据，并发送错误帧，通知帧的发送者将重新传输该帧。 RBUF 中的帧永远不会被有错误的帧覆盖。只有有效的接收帧可能导致 RBUF 溢出。

21.4.7 BUS Off

“总线关闭”状态由寄存器 CAN_CFG_STAT 中的 BUSOFF 来指示。如果 CAN 节点的发送错误计数>255，则 CAN 节点自动进入“总线关闭”状态。然后它不会参与进一步的通信，直到它再次返回到错误激活状态。如果 EIE 使能，则 BUSOFF 置位的同时也会置位 EIF 中断。如果 CAN 节点被上电复位或接收到 128 个 11 个隐性位序列（恢复序列），则 CAN 节点返回到错误激活状态。请注意，在每个恢复序列之间，总线可能具有显性状态。

在“总线关闭”状态下，RECNT 和 TECNT 保持不变。需要注意的是，在进入总线关闭状态时，TECNT 可能仍在递增计数，因此在进入总线关闭状态后可能会保持较小的值。因此，建议在节点进入总线关闭状态之前使用 TECNT，之后使用状态位 BUSOFF 来判断总线状态。

如果节点从“总线关闭”状态恢复，则 RECNT 和 TECNT 将自动重置为 0。

如果一帧正在等待传输但 CAN 节点已进入总线关闭状态，则该帧保持等待状态。如果节点在总线关闭后返回到错误激活状态，则将尝试传输此帧。如果不希望这样做，则应由主机控制器中止该帧。

21.4.8 扩展状态和错误报告

在 CAN 总线通信期间，可能会出现传输错误。以下功能支持对其进行检测和分析。

21.4.8.1 可编程错误警告门限值

接收/发送期间的错误由 RECNT 和 TECNT 计数。主控制器可以使用 CAN_LIMIT 中的可编程错误警告限制 EWL 对这些事件做出灵活反应。可以从 8 到 128 以 8 个错误为步长选择门限值：

错误计数门限值= (EWL+1)×8

如果在以下条件下 EIE 使能，则 EIF 中断将置位：

- RECNT 或 TECNT 超过错误警告门限，或超过后再递减至小于警告门限
- BUSOFF 位被更改。

21.4.8.2 仲裁丢失捕获 (ALC)

内核能够检测仲裁字段中仲裁丢失的确切位位置。如果启用了 ALIF 中断，则该事件被记录并且 ALIF 置位。如果节点能够赢得仲裁，则 ALC 的值保持不变，保存上次仲裁失败的旧值。

ALC 值定义如下：一帧以 SOF 开始，然后开始传输 ID 的第 1bit。ID 的第 1bit 对应 ALC 的值 为 0，ID 的第 2bit 对应 ALC 值为 1，以此类推。

仲裁只允许在仲裁域中进行。因此，ALC 的最大值为 31，即扩展帧中的 RTR 位。

附加提示：如果标准远程帧与扩展帧进行仲裁，扩展帧在 IDE 位失去仲裁，ALC 将为 12。发送标准远程帧的节点不会注意到已发生仲裁，因为这节点赢得了仲裁。

仲裁域之外的仲裁损失是不可能发生的。这样的事件实际是位传输错误。

21.4.8.3 错误类型 (KOER)

CAN 控制器识别 CAN 总线上的错误并将最后一个错误事件存储在 KOER 位中。如果启用了 BEIF 中断，则 CAN 总线错误可以发出中断信号。每个新的错误事件都会覆盖之前存储的 KOER 值。因此，主机控制器必须对错误事件做出快速反应。错误代码见表 21-41。

每个新错误都会更新 KOER。因此，当成功发送或接收帧时，它保持不变。以便于事后进行错误检查。

21.4.8.4 接收所有数据帧 (RBALL)

如果 RBALL=1，那么所有接收到的数据帧都会被存储，即使是有错误的。这也适用于环回模式。只有数据帧存储在 RBUF 中。不存储错误帧或过载帧。

如果启用 CiA 603 时间戳 (TIMEEN=1) 功能并且采用 EOF 时刻为时间戳 (TIMEPOS=1)，则在出现错误的情况下，将在错误帧开始时获取时间戳。

大多数可能的错误只有在节点是帧的发送者时才会发生。在这种情况下，如果激活了环回模式，则帧将仅存储在 RBUF 中。根据错误的类型，存储在 RBUF 时隙中的帧的部分可能是有效的，而其他部分是未知的。

表 21-41 RBALL 和 KOER

KOER	**状况**	**描述**
NO ERROR	All	接收成功。
BIT ERROR	Receiver	只能发生在 ACK 中。所有存储的数据都是有效的。
	Transmitter	有效载荷始终无效。包括 ID 的帧头可能是有效的。在仲裁阶段检测到错误位属于仲裁授权失败，因此不是位错误。但是如果在仲裁阶段的填充比特被帧的发送器检测到错误，那么则是一个比特错误，帧头是无效的。
FORM ERROR	All	仅涵盖数据帧中的 FORM ERROR。这包括 CRC 定界符、ACK 定界符或 EOF 位中的错误。所有存储的数据都是有效的。
STUFF ERROR	Receiver	STUFF ERROR 的位置未知。所有存储的数据均无效。
	Transmitter	只能在仲裁期间发生。所有存储的数据均无效。
ACK ERROR	Receiver	仅当节点在 LOM 中时才会发生。所有存储的数据都是有效的。
	Transmitter	只能在没有自 ACK 的环回模式下发生。所有存储的数据都是有效的。
CRC ERROR	Receiver	所有存储的数据均无效。

请注意，即使（部分）数据在被标志位指示为有效的，也存在可能是包含接收错误的消息数据。例如：如果节点的位时序配置错误，即没有正确同步，也可能收到无效数据。因此，数据（部分）有效的声明仅表明它可能是有效的。

21.4.9 扩展功能

21.4.9.1 单次传输

有时不需要自动重新传输。因此，可以通过设置 TPSS=1 令发送缓冲器 PTB 发送一帧，或通过设置 TSSS=1 令发送缓冲器 STB 发送一次消息。在这种情况下，在传输发生错误或仲裁丢失时不会执行重新传输。

在立即成功传输的情况下，与正常传输没有区别。但在传输不成功的情况下，会发生以下情况：

- 如果TPIF=1，则设置相应的发送缓冲区消息槽被清空
- 如果出现错误，KOER 和错误计数器会被更新。如果BEI=1，则BEIF会被置位，其他相应错误中断标志将置位。
- 在仲裁失败的情况下，如果ALIE=1，则ALIF置位。

因此，对于单次传输，TPIF 本身并不能指示帧是否已成功传输。单发传输只能与 BEIF 和 ALIF 一起使用。

如果单次传输与 TSALL 一起使用，并且 STB 中存在多于一帧，则对每一帧进行单次传输。无论是否有未成功传输（例如， ACK 错误），CAN 控制器都会前进到下一帧并在 STB 为空时停止。因此，在这种情况下，只有错误计数器可以指示成功发送了多少帧。如果两个帧之一出错，主机无法获知哪一帧被成功发送了。

如果总线被另一帧占用，如果开始单发传输，则 CAN 控制器等待总线空闲，然后尝试传输单发帧。

21.4.9.2 监听模式 (LOM)

LOM 模式提供了在不影响总线的情况下监控 CAN 总线的能力。CAN 控制器的 LOM 与 CAN FD 规范中定义的总线监控功能兼容。

另一个应用是自动波特率检测，主机控制器尝试不同的时序设置，直到没有错误发生。

在 LOM 模式中，错误（KOER、BEIF）情况仍被检测。

在 LOM 模式中，CAN 控制器无法将显性位写入总线（没有活动的错误标志或过载标志，没有 ACK）。遵从以下规则：

- 在LOM中，协议状态机就像处于错误被动模式一样，其中只有隐性错误标志被生成。状态寄存器可以被修改。
- 在LOM中，协议状态机不会生成显性ACK。
- 无论发生任何错误，错误计数器都保持不变。
- 如果一个节点传输一帧，那么只有在至少一个附加节点连接到总线上。且不处于LOM模式时，才会在总线上生成一个可见的 ACK。此时，就不会有 错误，所有节点（也包括 LOM 中的节点）都将收到该ACK。
- 如果在ACK错误后主动或被动错误标志置位，则节点会检测到ACK错误。

当节点正在发送时，不应设置成 LOM 模式。

如果启用 LOM，则无法进行发送。

环回模式（外部）LBME 对 LOM 的行为起着重要作用。如果禁用 LBME，则节点无法将任何显性位写入总线。但是如果 LBME 被激活，那么节点就被允许发送 self-ACK (SACK) 帧，但节点不会以 ACK 响应来自其他节点的帧，也不会生成错误或过载帧。总而言之，LOM 和 LBME 的组合是“一个静默的接收器，可以在必要时进行传输”。

21.4.9.3 总线连接测试

要检查节点是否连接到总线，可以执行以下步骤：

- 传送一帧。如果节点连接到总线，则在RX线路上可以观测到TX发送的消息。
- 如果有其他节点连接到CAN总线，则可以预期有其他节点发送ACK。
- 如果本节点是唯一连接到CAN总线的节点(但总线、收发器和 CAN控制器内核之间的连接良好)，则在ACK位发生错误，因为其他节点没有发送ACK。BEIF错误中断将 置位并且KOER= “100”（ACK错误）。
- 如果与收发器或总线的连接断开，则在SOF位之后BEIF错误中断将立即置位并且设置KOER= “001” (BIT错误）。

21.4.9.4 环回模式（LBMI 和 LBME）

CAN 控制器支持两种环回模式：内部回环(LBMI)和外部回环(LBME)。这两种模式都会导致接收到自己发送的帧，便于自检。

在 LBMI 中，CAN 控制器与 CAN 总线断开连接，并且 txd 输出设置为隐性电平。输出数据流在内部反馈到输入端。在 LBMI 中，节点生成自 ACK 以避免 ACK 错误。

在 LBME 中，CAN 控制器保持与收发器的连接，传输的帧将在总线上可见。借助收发器，CAN 控制器也可以接收到自己发出的帧。在 LBME 中，节点在 SACK=0 时不生成 Self-ACK，但在 SACK=1 时生成 Self-ACK。因此，在 SACK=0 的 LBME 中，帧传输有两种可能的结果：

1. 其他节点收到帧后发出ACK，从而完成一次成功的传输。
2. 总线上无其他节点，导致出现ACK错误。 如果不了解总线上是否有其他节点，为避免重传和错误计数器递增，建议使用TPSS或TSSS进行发送。

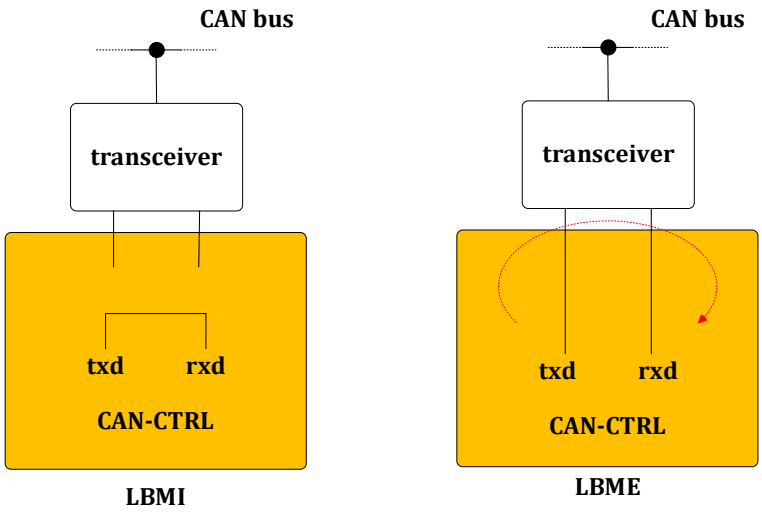


图 21-11 环回模式：内部和外部

在环回模式下，CAN 控制会接收自己的消息，将其存储在 RBUF 中，并设置适当的接收和发送中断标志（如果启用）。

LBMI 可用于芯片内部和软件测试，而 LBME 可以测试收发器及总线连接。

当传输处于活动状态时，不应设置 LBMI=1 或 LBME=1。主机软件应避免如此配置。 CAN 控制器硬件没有额外的保护。

如果节点连接到 CAN 总线，则不能通过简单地设置 LBMI=0 来切换回正常操作，因为此时另一个 CAN 节点可能正在传输。在这种情况下，应通过将位 RESET 设置为 1 来切换回正常操作，自动将 LBMI 清除为 0。最后设置 RESET=0，CAN 控制器恢复正常操作。与此相反，LBME 每次都可以禁用。

LBME 可以与 LOM 结合使用。

21.4.9.5 错误计数器复位

根据 CAN/CAN FD 标准 RECNT 计数接收错误，TECNT 计数发送错误。在过多的传输错误之后，CAN 节点必须进入总线关闭状态。RESET 不会修改错误计数器或总线关闭状态。CAN/CAN FD 规范定义了如何禁用总线关闭状态和减少错误计数器的规则。如果只是发生一个临时错误，节点会自动从错误中恢复。经典的 CAN 2.0B 规范定义了这种无需主机控制器交互的自动行为。

CAN FD 规范更为灵活，允许主机控制器接管这种自动错误计数器处理机制。但这应该格外小心，建议仅用于调试目的。

向 CAN_CFG_STAT.BUSOFF 写入 1 会重置错误计数器，同时强制节点离开总线关闭状态，动作期间 EIF 不置位。

21.4.10 软件复位

CAN_CFG_STAT.RESET=1 时，CAN 控制器处于件复位状态。但不是 CAN 控制器内所有模块都被 RESET 复位。部分模块只会被硬件复位。对于软件和硬件复位，复位值相同。

表 21-42 软件复位

寄存器	Reset	描述
ACFADR	No	-
ACODE_x	No	如果 RESET=1，寄存器可写，如果为 0，则写锁定。
AE_x	No	-
AFWL	No	-
AIF	Yes	-
ALC	Yes	-
ALIE	No	-
ALIF	Yes	-
AMASK_x	No	如果 RESET=1，寄存器可写，如果为 0，则写锁定。
BEIE	No	-
BEIF	Yes	-

寄存器	Reset	描述
BUSOFF	(No)	写 1 清零 BUSOFF，同时复位的错误计数器
EIE	No	-
EIF	No	-
EPASS	No	-
EPIE	No	-
EPIF	Yes	-
EWARN	No	-
EWL	Yes	-
FD_ISO	No	如果 RESET=1，寄存器可写，如果为 0，则写锁定。
F_PRESC	No	如果 RESET=1，寄存器可写，如果为 0，则写锁定。
F_Seg_1	No	如果 RESET=1，寄存器可写，如果为 0，则写锁定。
F_Seg_2	No	如果 RESET=1，寄存器可写，如果为 0，则写锁定。
F_SJW	No	如果 RESET=1，寄存器可写，如果为 0，则写锁定。
KOER	Yes	-
LBME	Yes	-
LBMI	Yes	-
LOM	No	-
RACTIVE	Yes	即使正处于接收状态，接收也会立即取消。不会产生 ACK。
RAFIE	No	-
RAFIF	Yes	-
RBALL	Yes	-
RBUF	(Yes)	所有 RB 消息槽都标记为空。RBUF 包含未知数据。
RECNT	No	通过设置 CAN_CFG_STAT.BUSOFF=1 可以复位错误计数器。
REF_ID	No	-
REF_IDE	No	-
RFIE	No	-
RFIF	Yes	-
RIE	No	-
RIF	Yes	-
ROIE	No	-
ROIF	Yes	-
ROM	No	-
ROV	Yes	所有 RB 消息槽都标记为空。
RREL	Yes	-
RSTAT	Yes	所有 RB 消息槽都标记为空。
SACK	Yes	-
SELMASK	No	-
STBY	No	-
S_PRESC	No	如果 RESET=1，寄存器可写，如果为 0，则写锁定。
S_Seg_1	No	如果 RESET=1，寄存器可写，如果为 0，则写锁定。
S_Seg_2	No	如果 RESET=1，寄存器可写，如果为 0，则写锁定。
S_SJW	No	如果 RESET=1，寄存器可写，如果为 0，则写锁定。

寄存器	Reset	描述
TACTIVE	Yes	所有传输立即被 RESET 终止。如果传输正在进行，会引起其他节点会产生错误帧。
TBE	Yes	-
TBF	Yes	-
TBPTR	No	-
TBSEL	Yes	TBUF 固定指向 PTB。
TBUF	(Yes)	所有 STB 消息槽都标记为空。因为 TBSEL TBUF 指向 PTB。
TECNT	No	通过设置 CAN_CFG_STAT.BUSOFF=1 可以复位错误计数器。
TEIF	Yes	-
TIMEEN	No	-
TIMEPOS	No	-
TPA	Yes	-
TPE	Yes	-
TSA	Yes	-
TSALL	Yes	-
TSMODE	No	-
TSNEXT	Yes	-
TSONE	Yes	-
TPIE	No	-
TPIF	Yes	-
TPSS	Yes	-
TSFF	Yes	所有 STB 消息槽都标记为空。
TSIE	No	-
TSIF	Yes	-
TSSS	Yes	-
TSSTAT	Yes	所有 STB 消息槽都标记为空。
TTEN	Yes	-
TTIF	Yes	-
TTIE	No	-
TTPTR	No	-
TTS	No	-
TTTBM	No	-
TTYPE	No	-
TT_TRIG	No	-
TT_WTRIG	No	-
T_PRESC	No	-
WTIE	No	-
WTIF	Yes	-

21.5 时间触发 CAN

时间触发 CAN (TTCAN)符合 ISO 11898-4 标准，帧仅在预定义的时间窗口传输。共有三种时间窗口类型：

- 独家时间窗口 （只允许一个节点传输特定ID的一帧）
- 空闲时间窗口 （未使用的时间窗口，用于进一步扩展网络）
- 仲裁时间窗口 （几个节点可以传输帧，可能发生总线仲裁）

发送时机由 TTCAN 系统管理节点离线定义。如图 21-12 所示，一行称为一个基本循环，它以参考帧开始。参考帧由时间主机发送。包括时间主机在内的任何节点都可以传输其他消息。

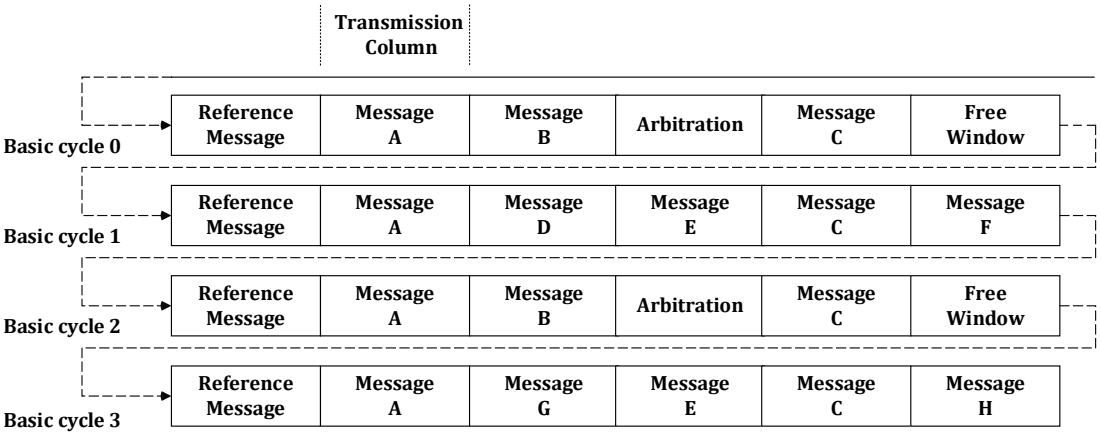


图 21-12 TTCAN 时空矩阵示例

消息的 SOF 时间是 Sync_Mark。参考帧的 Sync_Mark 是 Ref_Mark。使用一个 16 位定时器进行计时。计时器与 Ref_Mark 的值之间的差异就是循环时间。每个基本循环都在 Ref_Mark 处开始一次循环。循环时间作为时间戳存储在 RBUF 中。

时间窗口的长度由 TTCAN 管理节点定义，其长度必须足够传输一条消息。消息必须以单发模式传输。该规则只有一个例外：如果合并了多个仲裁时间窗口，则可以启用重传。但需要尽早禁用，以免影响接下来的时间窗口。

为了在时间窗口中传输帧，CAN 控制器提供了一个硬件触发机制。此触发时机需要由主机配置。所需的帧需要存储在 TBUF 消息槽中，等待触发。如果循环时间达到软件定义的触发时间，CAN 控制器会自动以单次模式传输帧。

如果到达触发时间，将产生中断信号通知 CPU 为下一个时间窗口配置触发时间。因此需要 CPU 实时响应，在一个时间窗口内完成后续触发时机的配置和消息帧的填充。

如 CAN 制器需要作为时间主机运行，则参考帧需要像所有其他消息一样放置在 TBUF 消息槽中，在到达硬件触发时间时发送。

参考帧的检测由所有时间从机和时间主机自动完成。通过在 CAN_REF_MSG 寄存器中设置参考帧的正确 REF_ID 和 REF_IDE 位来完成的。检测到参考帧之后，CAN 控制器后会自动更新 Ref_Mark，

开启一个新的循环周期。

此外，对于消息的触发，CAN 控制器提供了一种超时机制，用于检查自上次参考帧以来是否已经过了太长时间。如果超时，这将产生触发超时中断。

21.5.1 TTCAN 模式下的 TBUF

21.5.1.1 TTTBM=1

在 TTCAN 模式下（TTTBM=1），STB 被用作消息槽数组。每个消息槽可以由 TBPTR 寻址。主机可以使用 TBF 和 TBE 将消息槽标记为满或空。已填充的消息槽是无法再次写入的。TBSEL 和 TSNEXT 在 TTCAN 模式下无意义。

PTB 可以通过 TBPTR=0 来寻址。TTCAN 模式下，PTB 没有特殊属性，与任何其他 STB 消息槽作用相同。PTB 和 STB 消息帧的成功发送都由 TSIF 标志来指示。

TTCAN 模式下 TBUF 没有 FIFO 或优先级行为。每次传输选择一帧。

消息触发时间定义了需要传输消息的时间（时间窗口的开始），并使用指针 TTPTR 选择消息。如果触发事件发生，则开始传输选择的消息。最后，触发中断被设置为向主机发出需要准备下一个动作的信号。

所有传输只能使用触发启动。TPE、TSONE、TSALL、TPSS 和 TPA 固定为 0，在 TTCAN 模式下无意义。

21.5.1.2 TTTBM=0

在这种模式下，PTB 和 STB 与 TTEN=0 时行为相同。PTB 具有比 STB 更高的优先级，并且 STB 可以在 FIFO 模式(TSMODE=0)或优先级模式(TSMODE=1)下运行。

21.5.2 TTCAN 操作

上电后，时间主机需要按照 ISO 11898-4 中的定义进行初始化。 CAN 网络中可能有多达 8 个潜在的时间主机。每个主机都有自己的参考帧 ID（ID 的最后 3 位）。潜在的时间主机可能会尝试根据他们的优先级来传输他们各自的参考帧。优先级较低的时间主机后稍后尝试传输其参考帧。

如果设置了 TTEN=1，则 16 位定时器开启运行。如果检测到其他主机发送的参考帧或时间主机成功发送其参考帧,则 CAN 控制器将该消息的 Sync_Mark 复制到 Ref_Mark,将循环时间设置为 0 。参考帧成功接收后，CAN_RTIF.RIF 将置位，参考帧成功传输后 CAN_RTIF.TPIF 或 CAN_RTIF.TSIF 置位。主机需要为下一个动作配置触发时间。

触发事件可以是接收触发。只触发中断产生，可以用来检测是否收到预期的消息。这样的触发事件也可以用于其他操作，取决于主机应用程序。

另一种触发事件是传输触发。TTPTR 指向待发送消息。如果 TBUF 槽标记为空，则不开始传输，但会置位触发中断。

按通用管理，一般一个消息槽专门用于一条消息，但如果没有足够的 TBUF 消息槽可用，则一个消息槽也可以由不同的消息共享。

ISO 11989-4 中的大多数操作都需要单次传输。

21.5.3 TTCAN 时序

CAN 控制器支持 ISO 11898-4 Level 1。这包括一个以 CAN bit 时间为基本时间单位运行的 16 位定时器（CAN bit 时间由 S_PRESC、S_SEG_1、S_SEG_2 定义）。CAN bit 时间预分频系数由 T_PRESC 定义。如果 TTEN=1，则计时器开启连续计数。

在消息的 SOF 位置，定时器值为 Sync_Mark。如果该消息是参考帧，则将该值复制到 Ref_Mark。循环时间是定时器值减去 Ref_Mark，用作接收消息的时间戳或用作待传输消息的触发时间。由于存在溢出保护，因此循环时间在基本循环周期内始终是单调递增的。

ISO 11898-4 不包括 CAN FD 波特率切换。因此，CAN 控制器始终以慢速波特率运行定时器。定时器不受同步或波特率切换的影响。

由于触发事件、接收或传输而产生的中断需要跨时钟域（从 CAN 控制器时钟域到 MCU 高速时钟域），因此存在一定延迟。在一般情况下，主机应用程序都有足够的时间为下一个触发事件（即下一个时间窗口）准备所有操作。

一个节点的周期时间可能与其他节点的周期时间不同。在许多情况下，是±1 个刻度的差别。随着参考帧的传输开始新的基本周期将重新同步节点。

如果传输触发变为活动状态，则只能从下一个 CAN 位开始传输。因此，帧的 SOF 的最早传输点是 TT_TRIG+1。

21.5.4 TTCAN 触发类型

触发类型由 TTYPE 定义。TTPTR 是一个指向 TB 消息槽的指针，TT_TRIG 定义了触发时间。

触发相关动作应在达到最大循环时间 0xFFFF 之前完成，因为这是基本循环周期的最大长度。

除了立即触发，其他情况下，如果 TTIE=1，所有触发都会令 TTIF 置位。

如果 TTTBM=1，则仅支持时间触发。在此模式下使用其他触发器将导致 TEIF 置位。

对 TT_TRIG 进行写操作后，在达到触发时间（如果设置了 TTIE，则 TTIF 置位）或检测到错误（TEIF 置位）之前，对 TT_CFG、TT_TRIG 的写访问将被锁定（禁止再次写入）。如果 TTEN=0，写锁定解除。

21.5.4.1 立即触发

立即传输 TTPTR 指向的帧，TTIF 不置位。向 TT_TRIG 写入任意值立即触发。

在 TTCAN 模式下，不能使用 TPE、TSONE、TSALL。立即触发只能传输一帧。在第一次传输成功或未完成之前，主机不得发起第二次立即触发。

对于立即触发，可以通过 TSSS 选择单次触发模式。可以使用 TSA 中止传输。（TPSS 和 TPA 没

有意义。）

如果立即触发的 TTPTR 指向一个空的消息槽，则 TEIF 置位。

21.5.4.2 时间触发

时间触发会设置 TTIF=1，并产生一个中断事件，不执行其他操作。

时间触发可用作接收触发。如果节点希望在某个时间窗口内接收到消息，那么如果该消息未到达并且 RIF 未设置，则可以使用接收触发指示节点没有收到该消息。接收触发在预期成功接收消息的最后时刻之后置位。

如果 TT_TRIG 低于实际循环时间，则 TEIF 置位。

如果 TTTBM=1，时间触发是唯一可用的触发类型。

21.5.4.3 单发发送触发

单发传输触发用于专用时间窗口，其中需要以单发模式传输消息。所选消息由 TTPTR 定义。

在此模式下，无论 TSSS 的状态如何，都会自动使用单发模式。寄存器位 TSSS 被忽略并保持不变。

单次发射触发用于独家时间窗口。ISO 11898-4 定义了最多 16 个时钟周期的发送使能窗口。寄存器位 TEW(3:0)+1 定义了时钟周期数。如果总线被另一个帧占用，则不能启动发送。如果发送使能窗口结束并且帧无法启动发送,则帧传输被中止,帧的 TB 消息槽将被标记为空,如果设置了 AIE=1，则 AIF 置位。TB 消息槽中的帧数据不会被修改，因此如果下次尝试传输相同的数据，则只需再次将消息槽标记为已填充。

如果 TT_TRIG 低于实际循环时间，则 TEIF 置位，且不执行任何操作。

21.5.4.4 发送开始触发

发送开始触发器旨在用于合并的仲裁时间窗口，其中几个节点可以发送消息并进行 CAN 总线仲裁。所选消息由 TTPTR 定义。

此模式下，TSSS 定义是使用重传还是单发模式。

如果无法传输所选帧（仲裁丢失，错误后多次传输），则可以使用传输停止触发来停止传输。

如果 TT_TRIG 低于实际循环时间，则 TEIF 置位，且不执行任何操作。

21.5.4.5 发送停止触发

发送停止触发配合发送开始触发使用，用于停止发送开始触发的帧传输。

如果传输停止，则该帧被中止，如果 AIE=1，则 AIF 置位，并且帧的消息槽被标记为空。 TB 消息槽中的帧数据不会被修改，因此如果下次尝试传输相同的数据，则只需再次将消息槽标记为已填充。

发送停止触发的行为类似于单次发送触发使用的发送使能窗口的结束。如果传输停止触发指向一个空消息槽（这意味着消息已经传输），则不执行任何操作，并且 TEIF 不会置位。

如果 TT_TRIG 低于实际循环时间，则 TEIF 置位，但传输仍然会停止。

21.5.5 TTCAN 超时监测触发

与通用触发类型不同。超时触发有一个专用的中断标志 WTIF。如果循环计数等于 TT_WTRIG 定义的值，且 WTIE=1，则 WTIF 置位。

如果上一次收到有效参考帧已经经过很长时间，则可能产生超时监测触发。参考帧可以在周期性循环中或在事件之后被接收。主机应用程序需要对此进行处理，并且相应地调整超时触发时间。

WTIE 的默认值为 1， watch trigger 默认值为 0xffff, 超时触发默认开启, 且超时时间为最大值。要关闭超时监测触发，WTIE 需要设置为 0。

如果 TT_WTRIG 已更新且低于实际循环时间，则 TEIF 置位。如果 TTTBM=1，也可以使用超时监测触发。

21.6 CiA 603 时间戳

21.6.1 时间戳

CAN in Automation (CiA) 在规范 603 中定义了一个至少 16 位的时间戳，CAN 控制器支持该特性，并且可以与 TTCAN 一起使用，也可以单独使用。

CiA 603 的基本概念是有一个独立运行的定时器，它使用硬件主时钟进行计数而不是 CAN bit 时间（TTCAN 使用 CAN bit 时间）。精度高于 1μs（64 位定时器）。CAN 控制器配备了 64 位独立运行计数器。

事件发生后，将获取计时器的计数值，即时间戳。该时间戳分别存储在 RTS 或 TTS 中。

通常在 SOF 的采样点或帧被视为有效的 EOF 位处获取时间戳。可以通过配置 TIMEPOS 来选择时间戳位置。ACK 定界符后的七个隐性位构成 CAN/CAN FD 帧的 EOF。对接收来说，帧在 EOF 的倒数第二位时对接收器有效；但对发送来说，在 EOF 的最后一位时发送完成。

在许多系统中广泛使用的基于软件的时间戳依赖于接收和发送中断。因此，建议在 EOF 处添加时间戳。

CiA 603 支持 AUTOSAR 的时间戳和时间同步。对于 AUTOSAR，CAN 网络中的一个节点是时间主机。时间主机发送同步消息（SYNC 消息）。 SYNC 消息的时间戳由时间主机和所有时间从机获取。SYNC 消息的发送指令与实际传输 SYNC 消息时间之间的时间差将由时间主机在后续(FUP)消息中传

输。因此 CAN 控制器支持发送帧(TTS)使用一个时间戳，但支持所有接收帧(RTS)使用不同单独时间戳。可以使用 TBUF 消息槽内的 TTSEN 位为每个帧单独启用或禁用为传输帧生成时间戳。

寄存器位 TIMEEN 可以启用或禁用时间戳。如果禁用时间戳，则 TTS 和 RTS 无效。

21.6.2 CiA603 时间戳计时器

CiA 603 时间戳的计时器为 64bit 位宽，在 CAN 控制内部实现。计时器从零开始，不断向上计数。可以通过软件读写该计时器。定时器可以通过向 CAN_CIAWDAT0/1 进行写入，由于时间戳计时器为 64 位宽，因此需要进行两次 32 位写入，推荐先将计时器暂停，写入后 64 位时间戳后再启动计时器。

21.7 CAN 位时间

21.7.1 数据波特率

CAN 2.0B 定义了高达 1Mbit/s 的数据波特率。对于 CAN FD，没有固定限制。对于实际应用系统，数据速率受限于使用的收发器和 CAN 控制器可实现的时钟频率。

CAN 控制器可被编程为任意选择的数据速率，仅受相应位时序和预分频器寄存器中的位设置范围限制。

21.7.2 定义

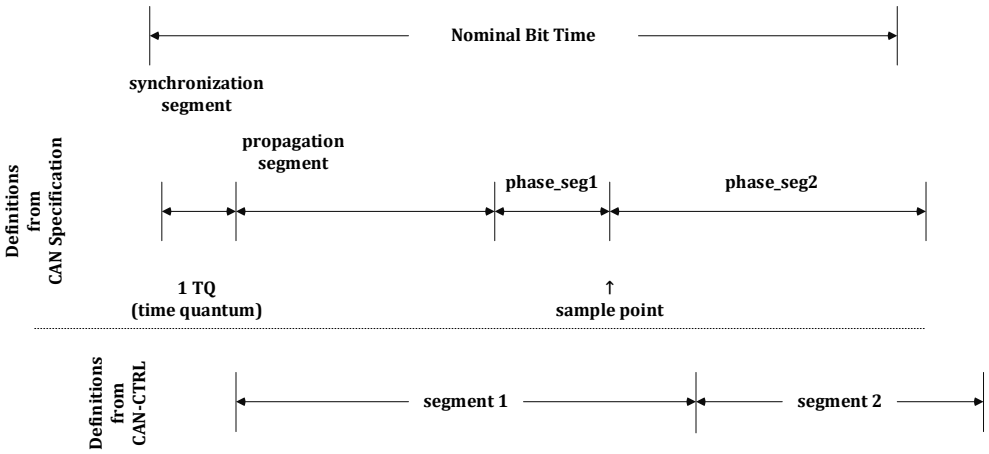


图 21-13 CAN 位时序规范

CAN 位时间 *BT* 由几个部分组成，如图 21-13 所示。每个段由多个时间量子单元 n_{TQ} 组成。时间量子 *TQ* 的持续时间为：

$$TQ = \frac{n_{prescaler}}{f_{clock}}$$

n_{TQ}和$n_{prescaler}$的值必须根据系统时钟频率匹配BT_{real}尽可能接近$BT_{Ideal} = 1/BR$,其中BR是 CAN 总线波特率:

$$BT_{ideal} \approx BT_{real} = \frac{n_{prescaler} \cdot n_{TQ}}{f_{clock}} = t_{Seq\_1} + t_{Seg\_2}$$

表 21-43 列出了 CAN/CAN FD 规范定义的最小配置范围。

表 21-43 CAN 时序段（最小配置范围）

部分	描述
SYNC_SEG	同步段 = 1 TQ
PROP_SEG	传播段 [1…8] TQ CAN 2.0 慢速速率 CAN FD 不启用 [1…48] TQ CAN 2.0 慢速速率 CAN FD 启用 [1…48] TQ CAN FD 帧头速度 [0…8] TQ CAN FD 数据速度
PHASE_SEG1	相位缓冲段 1 [1…8] TQ CAN 2.0 慢速速率 CAN FD 不启用 [1…16] TQ CAN 2.0 慢速速率 CAN FD 启用 [1…16] TQ CAN FD 帧头速度 [1…8] TQ CAN FD 数据速度
PHASE_SEG2	相位缓冲段 2 [2…8] TQ CAN 2.0 慢速速率 CAN FD 不启用 [2…16] TQ CAN 2.0 波特率 CAN FD 不启用 [2…16] TQ CAN FD 帧头速度 [2…8] TQ CAN FD 数据速度
SJW	同步跳转宽度 [1…4] TQ CAN 2.0 慢速速率 CAN FD 不启用 [1…16] TQ CAN 2.0 波特率 CAN FD 不启用 [1…16] TQ CAN FD 帧头速度 [1…8] TQ CAN FD 数据速度
IPT	信息处理时间 = [0…2] TQ PHASE_SEG2 ≥ IPT

CAN 控制器将 SYNC_SEG、PROP_SEG 和 PHASE_SEG1 收集到一个组中，并且该组的长度可通过 t_{Seg_1} 进行配置。表 21-44 列出了可用的配置范围。请注意，CAN 控制器硬件并对配置进行合法性检查，可以提供比 CAN/CAN FD 规范定义的更宽的配置范围。

表 21-44 CAN CAN 控制器时序设置（可用配置范围）

Setting	Requirements
t_{Seg_1}	[2…65] TQ CAN 2.0bit rate (slow) [2…65] TQ CAN FD nominal bit rate rate(slow) [2…17] TQ CAN FD data bit (fast)

t_{Seg_2}	[1...8] TQ	$t_{Seg\_1} \geq t_{Seg\_2}+2$	CAN	2.0bit rate	(slow)
	[1...32] TQ	$t_{Seg\_1} \geq t_{Seg\_2}+2$	CAN	FD nominal bit rate rate	(slow)
	[1...8] TQ	$t_{Seg\_1} \geq t_{Seg\_2}+1$	CAN	FD data bit	(fast)
t_{SWJ}	[1...16] TQ	$t_{Seg\_2} \geq t_{SWJ}$	CAN	2.0bit rate	(slow)
	[1...16] TQ	$t_{Seg\_2} \geq t_{SWJ}$	CAN	FD nominal bit rate rate	(slow)
	[1...8] TQ	$t_{Seg\_2} \geq t_{SWJ}$	CAN	FD data bit	(fast)

对于 CAN 2.0 波特率和 CAN FD 慢速波特率，由寄存器 S_Seg_1、S_Seg_2、S_SJW 和 S_PRESC 定义段长度。对于 CAN FD（快速）数据波特率，由寄存器 F_Seg_1、F_Seg_2、F_SJW 和 F_PRESC 定义段长度。

$$t_{Seg_1} = (S_{Seg_1} + 2) \cdot TQ$$
$$t_{Seg_2} = (S_{Seg_2} + 1) \cdot TQ$$
$$t_{SJW} = (S_{SWJ} + 1) \cdot TQ$$
$$n_{prescaler} = (S\_PRESC + 1)$$

$$t_{Seg_1} = (F_{Seg_1} + 2) \cdot TQ$$
$$t_{Seg_2} = (F_{Seg_2} + 2) \cdot TQ$$
$$t_{SJW} = (F_{SJW} + 2) \cdot TQ$$
$$n_{prescaler} = (F\_PRESC + 1)$$

CAN FD 在 BRS 位的采样点从慢速波特率切换到快速数据波特率，并在 CRC 定界符位的采样点切换回。

图 21-14 给出了一个合适的波特率配置的说明，其中$f_{tq_clk} = \frac{f_{clock}}{n_{prescaler}}$

位时间 BT_{real}、采样点和同步跳转宽度 SJW 都以 tq_clk 为单位。

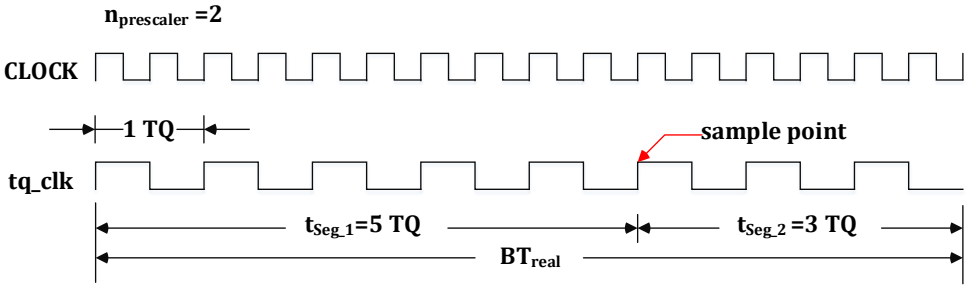


图 21-14 位采样时钟分频示例

主机控制器必须定义段 1、段 2 的长度和慢波特率的同步跳转宽度，如果 CAN FD 需要使用快速波特率，则还需要进行快速波特率段长度定义。

一些设置建议：

- 段1须比段2略大。那么采样点晚于位时间的中点。
- 同步跳转宽度不能大于段2。如果SJW太小，则 CAN节点可能太慢而无法重新同步，如果SJW 太大，则CAN节点可能会过于频繁地重新同步。SJW为第2段的一半似乎是较合适的设定。
- 连接到CAN总线的所有CAN节点应尽可能选择类似的时间单位设置。

可以使用最小时序值配置最快的 CAN 总线速度。但需要考虑：

如果预分频器大于 1，所有其他时序参数都可以设置为零，但这违反了规则 $t_{Seg\_1} \geq t_{Seg\_2} + 2$（慢速）和 $t_{Seg\_1} \geq t_{Seg\_2} + 1$（快速）。 因此，S_Seg_1 和 F_Seg_1 至少应设置为 1。但是这种时序参数的选

择仅在理论上是可行的，但对于实际网络可能不够稳妥。

如果预分频器设置为 1，则同步将变得困难。一般来说，对于 CAN 2.0 和 CAN FD 慢速波特率，CAN 规范要求 1bit 时间至少为 8 TQ 长度。对于 CAN 控制器，如果预分频器设置为 1，快速数据波特率的一位时间也需要至少为 8TQ 长度。

总之，最快的 CAN 总线频率是 can_clk 频率除以 8：预分频器设置为 1，即 8 TQ 位时间。

21.7.3 示例配置

以下示例是 CAN 2.0 慢速波特率配置的示例。CAN 控制器需要进行以下步骤：

1. 设置RESET=1。
2. 设置寄存器 S_Seg_1 和 S_Seg_2：

在本例中，总线 f_{bus} = 1Mbaud 上的数据速率，系统时钟为 16MHz 。

选择 n_{TQ} 和 $n_{prescaler}$ 的值以使 BT_{real} 尽可能接近 BT_{Ideal}

在本例中， $n_{prescaler}$ = 2，n_{TQ} = 8：从而使得

BT_{ideal} = BT_{real} = 8TQ。

BT_{real} = t_{Seg_1} + t_{Seg_2}

t_{Seg_1} = 5TQ，t_{Seg_2} = 5TQ，推导可得 S_Seg_1=3 和 S_Seg_2=2。

3. 设置接收过滤器（可选的）。
4. 设置寄存器S_SJW：
   $t_{Seg\_2} \geq t_{SJW}$ 可以选择 t_{SJW} = 2。
5. 设置时钟预分频寄存器S_PRESC: $n_{prescaler}$ = PRESC + 1即 S_PRESC=1。
6. 设置RESET=0。

以上给定的顺序不是强制性的。只需在开始时设置 RESET=1，否则无法对位时序、ACODE 和 AMASK 寄存器进行写入。完成配置后需要设置 RESET=0。然后控制器等待 8 个隐性位（帧结束）时间后恢复其正常操作。

7. 继续配置中断其他配置位并执行收发指令。

21.7.4 波特率切换和采样点

在 CAN 网络中，当使用具有波特率切换的 CAN FD 帧时，采样点的确切位置很重要，并且所有节点的采样点都需要尽可能接近。

CAN 点在 BRS 位的采样点和 CRC 定界符的采样点之后切换波特率。如图 21-15 所示，采样点的位置对 BRS 位的绝对长度有很大的影响。慢速帧头波特率的 1 个 TQ 可能比快数据率的几个 TQ 还大得多。如果数据波特率远高于帧头波特率，则采样点时机不合适可能会导致快速数据的采样错误。

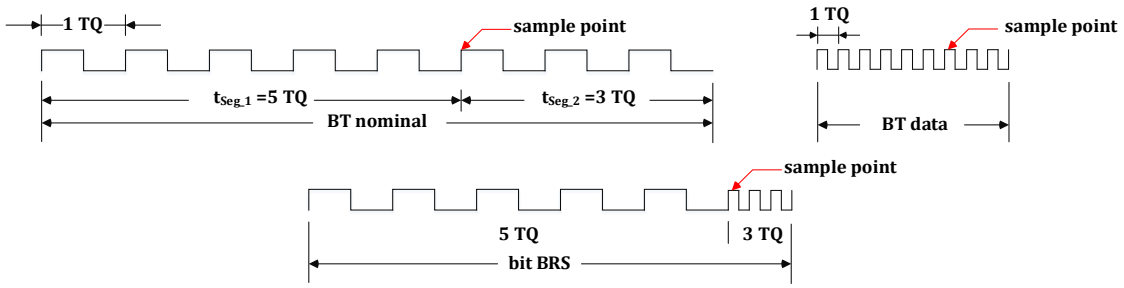


图 21-15 比特 BRS 的波特率切换 *S_PRESC≠F_PRESC*

21.7.5 CAN FD 节点的位时序配置

采样点的位置对于波特率切换非常重要。因此，一般建议所有 CAN FD 节点在 CAN 网络中使用完全相同的时序参数。与只需要所有节点的数据速率相同的经典 CAN 相比，CAN FD 要求采样点位于同一位置。换句话说，段 1 和段 2 需要为所有节点配置为相同的长度。此外，建议也使用相同长度的一个 TQ 以使所有节点同步。

只有当所有节点都以相同的基本时钟频率（CAN 控制器时钟）或至少以兼容的时钟频率运行时，才能实现此要求。

21.7.6 TDC 和 RDC

对于 CAN FD 节点，可以选择启用发射器延迟补偿(TDC)，同时自动使能接收器延迟补偿(RDC)。这些特性有以下背景：对于 CAN FD 数据波特率的通信，当发送收发器的延迟或总线的延迟大于比特时间。这时就需要对采样点进行补偿的。在没有 TDC 的情况下，FD 帧数据阶段的波特率受到以下限制：如果发送节点在该比特的采样点处不能接收到自己发送的最新比特，则它会检测到比特错误。

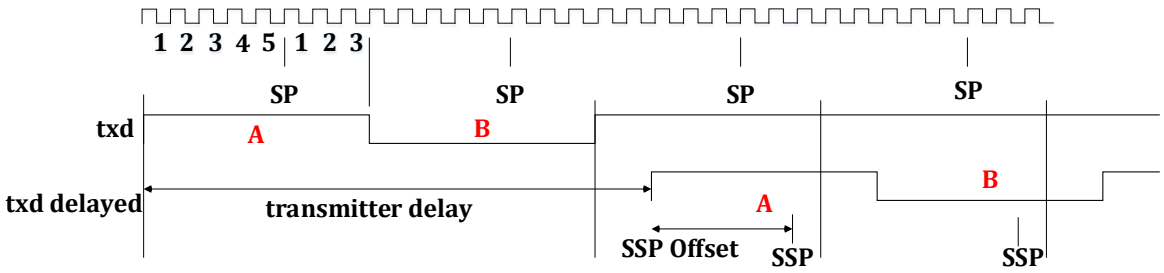


图 21-16 发射机延迟

图 21-16 给出了一个发射节点延迟比较大的例子。在该图中，显示了一个 txd 数据流，从位 A 和 B 开始。1bit 时间由 t_{Seg_1} = 5 TQ 的采样点 (SP) 之前的时间和 t_{Segt_2} = 3 TQ 的采样点之后的时间组成。在本例中，发送节点延迟大于 2 bit 时间。因此，使用原始采样点 SP 不能进行正确的采样。CAN FD 规范定义了一个附加的辅助采样点 (SSP)。如果 TDCEN=1 启用 TDC， CAN 控制器自动确定发送延迟。SSP 的位置是发送延迟加上 SSPOFF 定义的偏移量。 SSPOFF 以 TQ 为单位，建议设置 t_{Seg_1} 等于 SSPOFF。 F_SEG_1 定义了数据波特率的 t_{Seg_1}。在图 21-16 中 F_SEG_1=3，选择 SSPOFF=5

如果使用 TDC，ISO 11898-1:2015 要求仅可使用 F_PRESC=0 或 1。根据这一要求，CAN 控制器

能够自动确定 3bit 以内的快速数据波特率发送延迟。

接收节点也需要考虑延迟大于 1 个比特时间的情况。CAN FD 在位 FDF =1 和随后的 r0 位之间的下降沿定义了额外的硬件同步。（此时为慢速波特率）CAN 和 CAN FD 的同步通常定义为以 TQ 为步长进行。但是对于快速数据波特率的同步来说，慢速波特率的一个 TQ 的步长可能太大了。因此，在 FDF 位的附加硬同步需要违反此规则，需要尽可能精确地同步，并且仅受系统时钟频率的限制。CAN 控制器使用这种特殊的同步，称为 RDC。如果 FDF =1，无论是否启用 TDC，RDC 都会在接收期间自动完成。

21.7.7 位时序建议

CAN FD 时序配置要求 CAN 网络中的所有节点使用相同的数据速率和相同的采样点。

请注意，如果预分频器设置为 1，则位时间至少应为 8TQ，以实现稳定的通信。较小的位时间设置也可能进行通讯，但不稳定。

是否使用 TDC 取决于 CAN 网络实际负载情况，可以根据需要启用或禁用 TDC。

表 21-45 CAN 相关名词缩写

缩写	描述
PSP	初级采样点
SSP	二级采样点
Seg 1	第 1 段
Seg 2	第 2 段
TDC	发射延迟补偿（见 SSPOFF）

表 21-46 20MHz can_clk 的建议

Bit Rate [Mbit/s]	PSP [%]	SSP [%]	Prescaler	Bit Time [TQ]	Seg 1 [TQ]	Seg 2 [TQ]	SJW [TQ]	TDC[clk]
0.25（仲裁）	80	-	1	80	64	16	16	-
0.5（仲裁）	80	-	1	40	32	8	8	-
0.5	80	（禁用）	1	40	32	8	8	-
0.833	79	（禁用）	1	24	19	5	5	-
1	80	80	1	20	16	4	4	16
1.538	77	77	1	13	10	3	3	10
2	80	80	1	10	8	2	2	8
4	80	80	1	5	4	1	1	4
5	75	75	1	4	3	1	1	3

表 21-47 40MHz can_clk 的建议

Bit Rate [Mbit/s]	PSP [%]	SSP [%]	Prescaler	Bit Time [TQ]	Seg 1 [TQ]	Seg 2 [TQ]	SJW [TQ]	TDC[clk]

0.25（仲裁）	80	-	2	80	64	16	16	-
0.5（仲裁）	80	-	1	80	64	16	16	-
0.5	80	（禁用）	2	40	32	8	8	-
0.833	79	（禁用）	2	24	19	5	5	-
1	80	80	1	40	32	8	8	32
1.538	77	77	1	26	20	6	6	20
2	80	80	1	20	16	4	4	16
3.077	77	77	1	13	10	3	3	10
4	80	80	1	10	8	2	2	8
5	75	75	1	8	6	2	2	6
6.667	83	83	1	6	5	1	1	5
8	80	80	1	5	4	1	1	4
10	75	75	1	4	3	1	1	3

表 21-48 80MHz can_clk 的建议

Bit Rate [Mbit/s]	PSP [%]	SSP [%]	Prescaler	Bit Time [TQ]	Seg 1 [TQ]	Seg 2 [TQ]	SJW [TQ]	TDC[clk]
0.25（仲裁）	80	-	4	80	64	16	16	-
0.5（仲裁）	80	-	2	80	64	16	16	-
0.5	80	（禁用）	4	40	32	8	8	-
0.833	79	（禁用）	4	24	19	5	5	-
1	80	80	2	40	32	8	8	64
1.538	77	77	2	26	20	6	6	40
2	80	80	2	20	16	4	4	32
3.077	77	77	2	13	10	3	3	20
4	80	80	1	20	16	4	4	16
5	75	75	1	16	12	4	4	12
6.667	83	83	1	12	10	2	2	10
8	80	80	1	10	8	2	2	8
10	75	75	1	8	6	2	2	6

22 WWDG 窗口看门狗

22.1 概述

窗口看门狗用于检测由于外界干扰或未预见的逻辑情况而产生的软件运行异常，当异常发生时，软件通常并不按照正常时序执行。

窗口看门狗使用系统高速主时钟的分频时钟进行工作，通常为 PLL 时钟的分频时钟，具有较高的时钟精度，适用于对看门狗超时时间有精确要求的应用场景。

窗口看门狗包含一个 7bit 递减计数器 WWDG_CR.T[6:0]，当计数器最高位 T[6]清零后，看门狗会产生全芯片复位，如果看门狗计数器 WWDG_CR.T[6:0]尚未计数至窗口值 WWDG_CFG.W[6:0]就被软件进行刷新重置，同样会产生复位。即看门狗必须在有效的窗口内进行刷新重置操作。

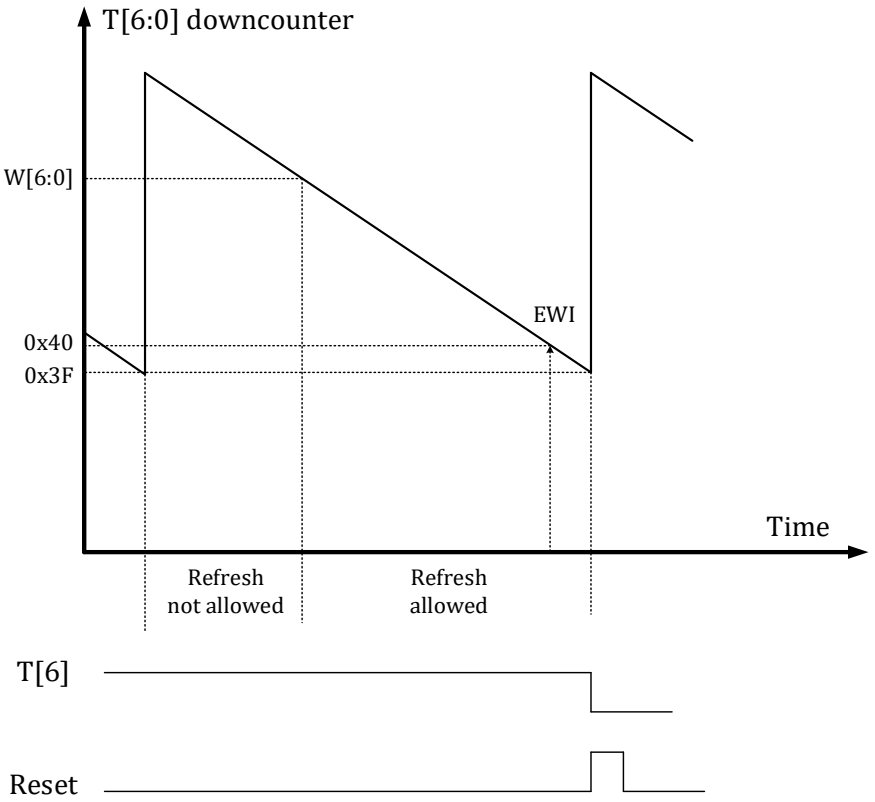


图 22-1 窗口看门狗计数及刷新机制

可编程的独立运行的递减计数器

可控的复位

当看门狗计数值小于 0x40 时产生复位(如果看门狗被使能)

当看门狗计数值在窗口之外被刷新时产生复位(如果看门狗被使能)

超前提醒中断(Early Wakeup Interrupt: EWI)，当 EWI 位使能且看门狗使能，且计数器计数至 0x40 时产生超前提醒中断。

在某些应用中，超前中断可用于在不产生复位的前提下进行软件系统检查或系统恢复。在这种应用中，在超前中断的中断处理函数中可以将看门狗计数值刷新重置。

如果超前中断无法得到及时响应，则看门狗计数器会递减计数至 0x3F 以下，从而产生看门狗复位。

看门狗超时时间由如下公式计算

$$Timeout_{WWDG}=t_{MCLK}\times 32768\times 2^{WDGTB}\times(T[5:0]+1)$$

其中 $Timeout_{WWDG}$ 为看门狗超时时间，

t_{MCLK} 为系统主时钟

32768 为看门狗内部固定时钟分频系数

WDGTB 为看门狗计数时钟预分频系数

T[5:0]为看门狗计数器的装载值的低 6 位。

由于看门狗通常使用高速时钟的分频时钟进行计数，因此从 0x40 计数至 0x3F 有 32768×2^{WDGTB} 个高速时钟周期，足够软件进行重置看门狗的操作。

表 22-1 窗口看门狗超时时间(与系统主时钟有关)

WDGTB	Prescaler(2^{WDGTB})	最短超时时间	最长超时时间
0	1	0.17ms	10.92ms
1	2	0.34ms	21.84ms
2	4	0.68ms	43.69ms
3	8	1.36ms	87.38ms

在调试模式下，当 CPU 暂停运行时，可以根据 SYS_DBG_CFG.DBG_WWDG_STOP 位控制窗口看门狗是否继续计数。

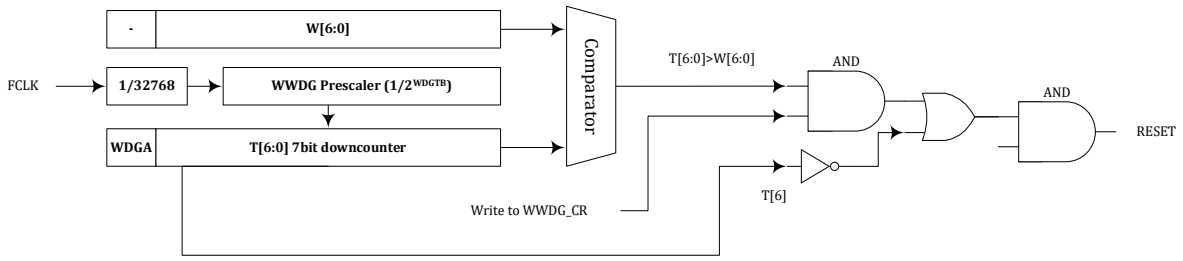


图 22-2 窗口看门狗复位产生逻辑

应用软件需要在适当的时机写 WWDG_CR 以刷新看门狗计数器，防止 MCU 复位，这个刷新操作需要在计数值 T[6:0]<窗口寄存器值 W[6:0]时进行，否则在窗口之外写 WWDG_CR 也会产生复位动作。T[6]每次写入强制为 1，因此 WWDG_CR.T[6:0]可以写入的数据范围是 0xFF~0xC。

22.2 寄存器

22.2.1 地址分配

WWDG 基地址为 0x4001B400

表 22-2 窗口看门狗寄存器列表

名称	偏移地址	说明
WWDG_CR	0x00	窗口看门狗控制寄存器
WWDG_CFG	0x04	窗口看门狗配置寄存器
WWDG_IF	0x08	窗口看门狗中断标志寄存器

22.2.2 WWDG_CR 窗口看门狗控制寄存器

地址：0x4001_B400

复位值：0x0

表 22-3 WWDG_CR 窗口看门狗控制寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								WDG_EN	T						
								RW	RW						
								0	0x7F						

位置	位名称	说明
[31:8]		未使用
[7]	WDG_EN	窗口看门狗使能，1：使能；0：禁用。上电默认禁用。软件写 1 可使能看门狗，软件写 0 且同时向[15:8]写入 0x3C 时，可禁用看门狗。禁用后看门狗不再产生复位，但仍可计数并产生超前中断
[6:0]	T	窗口看门狗计数器，使用看门狗内部的分频时钟进行递减计数，当计数值小于 0x40(T[6]为 0)时产生复位。为避免写入 T[6]为 0 立即产生复位， T[6]在写入时强制置 1，软件只能写入 T[5:0]。

22.2.3 WWDG_CFG 窗口看门狗配置寄存器

地址：0x4001_B404

复位值：0x0

表 22-4 WWDG_CFG 窗口看门狗配置寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						WDGTB		EWI	W						
						RW		RS	RW						
						0		0	0x7F						

位置	位名称	说明
[31:10]		未使用
[9:8]	WDGTB	窗口看门狗时钟预分频系数 2'h0：系统主时钟/32768/1 2'h1：系统主时钟/32768/2 2'h2：系统主时钟/32768/4 2'h3：系统主时钟/32768/8
[7]	EWI	超前中断使能，1：使能；0：禁用。上电默认禁用。软件写 1 可使能中断，软件写 0 无效，此位一旦置 1 只能通过硬件外部复位或上电复位清零。看门狗计数至 0x40 时产生超前中断。
[6:0]	W	窗口看门狗喂狗窗口，当 T>W 时喂狗也会产生复位，即合法的复位窗口为 W>=T>0x40

22.2.4 WWDG_IF 窗口看门狗中断标志寄存器

地址：0x4001_B408

复位值：0x0

表 22-5 WWDG_CFG 窗口看门狗中断标志寄存器

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

位置	位名称	说明
[31:2]		未使用
[0]	EWIF	窗口看门狗超前中断标志,当计数器计数至 0x40 时 EWIF 置位为 1，写 1 清零，写 0 无效果。即使看门狗不使能，EWIF 位仍可以置位，因为看门狗计数器 T[6:0]在看门狗禁用时仍在计数。但只有 WDGA 和 EWI 均为 1 时，看门狗才会向 CPU 提起中断请求。

23 IWDG 独立看门狗

23.1 概述

独立看门狗使用 32kHz 低速 RC(LRC)时钟进行工作，在系统主时钟故障的情况下仍可保证正常工作。由于 LRC 时钟精度不如系统高速时钟，独立看门狗可用于对超时时间精确性要求不高的应用场景。

独立看门狗内部包含一个 21bit 递减计数器。独立看门狗可以配置产生系统深度休眠（时钟关闭或电源关闭）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 是一直存在的，所以独立看门狗可以继续正常工作；甚至在系统进入 Standby 模式大部分电路进入掉电状态，LRC 和 IWDG 仍是正常工作的。

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 进行深度休眠定时唤醒，在进入休眠前将 IWDG 刷新重置，IWDG 从 IWDG_RTH 开始递减计数，在经过相对准确的休眠时间后被 IWDG 唤醒。如果不使能 IWDG 深度休眠定时唤醒，当 IWDG 产生全芯片复位时也可以将芯片从深度休眠中解除，但同时会复位 MCU 所有寄存器配置。

独立看门狗超时复位时间由如下公式计算

$Timeout_{IWDG}=t_{LRC}\times(IWDG\_RTH-7)$

独立看门狗唤醒时间由如下公式计算

$Wakeup_{IWDG}=t_{LRC}\times(IWDG\_RTH-IWDG\_WTH)$

其中 $Timeout_{IWDG}$ 为独立看门狗超时复位时间，$Wakeup_{IWDG}$ 为独立看门狗唤醒时间，t_{LRC} 为 LRC 时钟周期，1/32kHz=31.25uS，IWDG_RTH 为独立看门狗超时复位门限值，IWDG_WTH 为独立看门狗唤醒门限值。

IWDG_RTH=0x001000 对应独立看门狗最小复位时间间隔为 4096/32kHz = 0.128s。

IWDG_RTH=0x1FF000 对应独立看门狗最大复位时间间隔为 511×4096/32kHz = 65.408s。

关于 IWDG 喂狗跨时钟同步问题:

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。

23.2 寄存器

23.2.1 地址分配

IWDG 基地址为 0x4001B8C0

表 23-1 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

23.2.2 IWDG_PSW 独立看门狗密码寄存器

地址：0x4001_B8C0

复位值：0x0

表 23-2 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，因此每次对看门狗进行写操作前都需要写入密码

23.2.3 IWDG_CFG 独立看门狗配置寄存器

地址：0x4001_B8C4

复位值：0x0

表 23-3 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
[31:16]		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。

23.2.4 IWDG_CLR 独立看门狗清零寄存器

地址：0x4001_B8C8

复位值：0x0

表 23-4 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	权限	说明
[31:16]	NA	未使用
[15:0]	WO	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。

23.2.5 IWDG_WTH 独立看门狗定时唤醒门限寄存器

地址：0x4001_B8CC

复位值：0x001F_F000

表 23-5 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]		未使用
[20:12]	WTH	看门狗定时唤醒门限值，看门狗使用 32kHz LRC 时钟从 IWDG_RTH 开始计数，递减计数至 IWDG_WTH 产生唤醒信号。
[11:0]		恒为 0

IWDG_WTH 的写入无需向 IWDG_PSW 写入密码。

23.2.6 IWDG_RTH 独立看门狗超时复位门限寄存器

地址：0x4001_B8D0

复位值：0x001F_F000

表 23-6 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]		未使用
[20:12]	RTH	看门狗超时复位门限值，也是重新装载值。看门狗使用 32kHz LRC 时钟从 IWDG_RTH 开始计数，计数至 0x7 产生复位。向该寄存器写入 0x0，会被硬件强制改写为 0x1000。先向 IWDG_PSW 写入正确密码才能改写 IWDG_RTH 寄存器。改写 IWDG_RTH 同样具有重置看门狗计数器的作用，看门狗会从新的 IWDG_RTH 开始计数。
[11:0]		恒为 0

为防止 IWDG_RTH 被写入为 0，当软件写入值为全 0 时，硬件会强制改写为 0x1000，且寄存器低 12 位恒为 0。IWDG_RTH 的写入需要先向 IWDG_PSW 写入密码。

23.2.7 IWDG_CNT 独立看门狗当前计数值寄存器

地址：0x4001_B8D4

复位值：0x0000_0000

表 23-7 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				
											RO				
											0x00				

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

位置	位名称	说明
[31:21]		未使用
[20:0]	CNT	看门狗当前计数值，此数值≤IWDG_RTH

24 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，关断部分电路供电来达到降低功耗的目的。

24.1 外设时钟门控

外设时钟由系统高速时钟 MCLK 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 6.3.8。**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。**

24.2 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时钟频率上。

其中 I2C 使用 SYS_CLK_DIV0 作为分频系数，UART0/1 共享 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。

24.3 低功耗模式

表 24-1 低功耗模式汇总

模式	模式进入	模式退出	PLL/HSI/HSE	LSI	内核供电
休眠 Sleep	WFI/WFE	任意中断 Debug 操作 外部复位 IWDG 复位	PLL/HSI/HSE On，CPU 时钟 Off，NVIC/外设时钟 On	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/HSE Off，CPU/外设时钟 Off		
停机(内核断电) Standby	PWR_CFG.PG + SLEEPDEEP + WFI/WFE	IWDG 定时唤醒 IO 唤醒 外部复位 IWDG 复位			除功耗管理、复位管理、后备存储器、IWDG 之外大部分电路均断电

24.4 Sleep Mode 休眠模式

休眠模式下，CPU 时钟被关闭，但 NVIC 模块仍继续工作，所有的外设模块以及 IO 正常工作。

休眠模式对电路供电没有影响，所有寄存器状态和存储器数据在休眠过程中保持正常供电。

可参考休眠例程，进入休眠前关闭数字部分各个模块的时钟，关闭模拟 ADC/OPA/CMP/DAC 等模块

24.4.1 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

24.4.2 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。

24.5 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/HSE。32kHz RC 时钟 LSI 仍正常工作。同时 LDO 会进入低功耗模式，BGP 模块被关闭。

相比休眠模式，深度休眠模式可以进一步降低功耗。

深度休眠模式对电路供电没有影响，所有寄存器状态和存储器数据在深度休眠模式中保持正常供电。

如果有外设需要使用高速时钟完成正在进行中的操作，例如 flash 的擦除写入，则深度休眠会推迟等待操作完成后再进入。

24.5.1 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。

24.5.2 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。

24.6 Standby Mode 停机模式

停机模式可使芯片功耗降至最低。停机模式在深度休眠模式的基础上，进一步将大部分数字电路断电。但 LSI 时钟、功耗管理模块 PMU、复位管理模块 RMU、独立看门狗模块 IWDG 仍正常工作，后备存储器 BRAM 在进入停机模式后不断电，可以提供 128Bytes 大小的存储空间用于保持停机模式前的系统关键数据。

此外，IO 唤醒使能极性选择，系统复位、休眠、唤醒事件记录寄存器不断电。

停机模式下所有 IO 处于高阻态，IO 唤醒信号从 PAD 直接送入芯片内部唤醒逻辑。

24.6.1 模式进入

设置 PWR_CFG.PG 为 1，设置 CPU 内核 System Control Register SLEEPDEEP 为 1，通过执行 WFI/WFE 指令进入停机模式。

如果只配置了 PWR_CFG.PG 为 1，而 SLEEPDEEP 为 0，则执行 WFI/WFE 指令后芯片进入休眠模式而不是停机模式。

24.6.2 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，也可以解除深度休眠状态。

注意：此时 Debug 操作无法将芯片从停机模式中唤醒。

在应用程序编写中请尽量避免上电即进入休眠状态，且进入休眠模式前应设置合理的唤醒源。

24.7 寄存器

24.7.1 地址分配

PMU 基地址为 0x4001B800

表 24-2 功耗管理模块地址空间

名称	偏移	说明
BRAM	0x00~0x7F	后备存储器
PWR_CFG	0x80	功耗管理配置寄存器
EVT_RCD	0x84	事件记录寄存器

IO_WAKE_POL	0x88	IO 唤醒极性寄存器
IO_WAKE_EN	0x8C	IO 唤醒使能寄存器

24.7.2 BRAM 后备存储器

后备存储器共 128bytes，在停机模式下不断电，用于在芯片进入停机模式前保持内存或寄存器中的关键数据,以免断电后数据遗失。BRAM 仅允许按 word 写入和读出,可以连续写入或连续读出，但应避免写入数据后立即读出，BRAM 内部需要至少一个总线周期的写入时间。

24.7.3 AON_PWR_CFG 功耗管理配置寄存器

地址：0x4001_B880

复位值：0x0

表 24-3 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	PG
														RW	RW
														1	0

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]	PG	停机模式配置位

24.7.4 AON_EVT_RCD 事件记录寄存器

地址：0x4001_B884

复位值：0x0

表 24-4 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
	STANDBY	DEEPSLEEP	SLEEP			IWDG_WK	IO_WK			CPOR_RST	WWDG_RST	IWDG_RST	KEY_RST	HPOR_RST	LPOR_RST
	RO	RO	RO			RO	RO			RO	RO	RO	RO	RO	RO
	0	0	0			0	0			0	0	0	0	1	1

位置	位名称	说明
[31:15]		未使用
[14]	STANDBY	停机(掉电)模式记录，高表示发生过
[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,高表示休眠被 IWDG 定时唤醒,包括 Deep Sleep 以及 Standby 模式的唤醒
[8]	IO_WK	IO 唤醒记录，高表示休眠被 IO 唤醒，包括 Deep Sleep 以及 Standby 模式的唤醒
[7:6]		未使用
[5]	CPOR_RST	CPU 掉电后上电复位发生记录，高表示发生过
[4]	WWDG_RST	窗口看门狗复位发生记录，高表示发生过
[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]	HPOR_RST_RCD	HPOR 复位发生记录，高表示发生过
[0]	LPOR_RST_RCD	LPOR 复位发生记录，高表示发生过

向 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一定时间。

24.7.5 AON_IO_WAKE_POL IO 唤醒极性寄存器

地址：0x4001_B888

复位值：0x0

表 24-5 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，唤醒 IO 开启下拉 0：低电平，唤醒 IO 开启上拉

24.7.6 AON_IO_WAKE_EN IO 唤醒使能寄存器

地址：0x4001_B88C

复位值：0x0

表 24-6 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
								P3_2_EN	P2_14_EN	P1_11_EN	P0_13_EN	P0_5_EN	P0_4_EN	P0_3_EN	P0_2_EN
								RW	RW	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

位置	位名称	说明
[31:8]		未使用
[7]	P3_2_EN	P3[2]外部唤醒触发使能
[6]	P2_14_EN	P2[14]外部唤醒触发使能
[5]	P1_11_EN	P1[11]外部唤醒触发使能
[4]	P0_13_EN	P0[13]外部唤醒触发使能
[3]	P0_5_EN	P0[5]外部唤醒触发使能
[2]	P0_4_EN	P0[4]外部唤醒触发使能
[1]	P0_3_EN	P0[3]外部唤醒触发使能
[0]	P0_2_EN	P0[2]外部唤醒触发使能

25 版本历史

表 25-1 文档版本历史

时间	版本号	说明
2024.01.19	1.47	添加关于休眠的说明，修正关于 IIC 时钟分频的说明
2023.10.22	1.46	增加芯片版本信息寄存器
2023.09.25	1.45	补充关于 IWDG 喂狗跨时钟问题和 GPIO PDI 的说明
2023.08.10	1.44	修改了 DAC 校准值地址和 GPIO 部分的笔误
2023.07.09	1.43	修改 IWDG 地址和图 3-4 SRAM0 笔误部分
2023.05.07	1.42	更新 SYS_FLSE 和 SYS_FLSP 的说明
2023.04.07	1.41	增加 ADC 采集共模电压的说明
2023.02.22	1.40	修改低速时钟精度范围
2023.02.18	1.39	修改 MCPWM_SDCFG 的描述
2023.02.10	1.38	增加关于使能寄存器 PLLPDN，RCHPD，BGPPD，LDO2_CTL 的描述
2023.02.07	1.37	修订 SYS_AFE_REG3.ADCCLKSEL 的描述
2023.01.29	1.36	修改 MCPWMx_EIF、MCPWMx_RE 寄存器地址有误部分
2022.11.23	1.35	增加软复位 ADC 模块描述
2022.11.10	1.34	增加 IO 与内部模拟电路间连接电阻描述，修改 ADC 范围
2022.09.05	1.33	修改 MCPWMx_IO01/IO23，将 P/N 通道默认不互换修改为默认互换
2022.08.21	1.32	更新 CAN-FD 章节
2022.07.29	1.31	简化 SYS_AFE_REG 的部分描述
2022.06.24	1.3	修订 CMPx_SELN/CMPx_SELP
2021.09.15	1.2	修订部分文档格式
2021.07.04	1.1	修订 CAN 章节
2021.05.24	1.0	Release
2021.04.29	0.9	修改 FMAC 图片
2021.04.05	0.8	更新 FSMC，即片内 flash 章节
2021.04.02	0.7	更新 SPI
2021.04.02	0.6	更新 I2C
2021.03.12	0.5	更新 DMA
2021.03.11	0.4	更新 ADC
2021.03.09	0.3	更新 SYS_AFE_REGx，核对所有文字表述，更新各模块说明框图
2021.06.16	0.2	增加 FMAC 章节
2020.02.08	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。