南京凌鸥创芯电子有限公司

LKS32MC08x 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。

1.2 缩略词汇表

字：32 位数据/指令。

半字：16 位数据/指令。

字节：8 位数据。

双字：64 位数据。

ADC：Analog-Digital Converter，模数转换器

DAC：Digital-Analog Converter，数模转换器

BGP：Bandgap，带隙基准

WDT：Watch dog，看门狗

LSI：Low Speed Internal Clock，即 32kHZ RC 时钟

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

HSE：High Speed External Clock，即 4~8MHz 外部晶振时钟

PLL：Phase Lock Loop Clock，即 96MHz 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 CPU 配置字节

2 地址空间

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

表 2-1 系统地址空间分配

外设	时钟/软复位	开始地址	结束地址	大小	说明
FLASH	PLL/无	0x0000_0000	0x0000_FFFF	64kB	FLASH 存储空间
RAM	PLL /无	0x2000_0000	0x2000_1FFF	8kB	RAM
SYS	PLL /无	0x4000_0000	0x4000_03FF	1kB	SYSTEM control, Clock / Reset Management
FLSCR	PLL /无	0x4000_0400	0x4000_07FF	1kB	FLASH control registers
SPI	FCLK[8]/sft_rst[0]	0x4001_0000	0x4001_03FF	1kB	SPI interface
I2C	FCLK[0]/sft_rst[0]	0x4001_0400	0x4001_07FF	1kB	I2C interface
CMP	PLL /无	0x4001_0C00	0x4001_0FFF	1kB	Comparator
HALL	FCLK[1]/sft_rst[1]	0x4001_1000	0x4001_13FF	1kB	HALL interface
ADC	ACLK	0x4001_1400	0x4001_17FF	1kB	ADC interface
TIMER	FCLK[2]/sft_rst[2]	0x4001_1800	0x4001_1BFF	1kB	General Purpose Timer
MCPWM	FCLK[3]/sft_rst[3]	0x4001_1C00	0x4001_1FFF	1kB	Motor Control Pulse Width Modulation
GPIO	PLL/无	0x4001_2000	0x4001_23FF	1kB	General Purpose Input / Output
CRC	PLL/无	0x4001_2400	0x4001_27FF	1kB	Cyclic Redundancy Check
UART0	FCLK[4]/ sft_rst[4]	0x4001_2800	0x4001_2BFF	1kB	
UART1	FCLK[5]/ sft_rst[5]	0x4001_2C00	0x4001_2FFF	1kB	
DMA	PLL/无	0x4001_3000	0x4001_33FF	1kB	
CAN	FCLK[7]/无	0x4001_3400	0x4001_37FF	1kB	
SIF	PLL/无	0x4001_3800	0x4001_3BFF	1kB	
DSP	FCLK[6]/无	0x4001_4000	0x4001_5FFF	8kB	

3 内核

3.1 ARM®CortexTM-M0 核心

Cortex-M0 是 Cortex-M 家族中的 M0 系列。最大特点是低功耗的设计。Cortex-M0 为 32 位、3 级流水线 RISC 处理器，其核心仍为冯.诺依曼结构，是指令和数据共享同一总线的架构。作为新一代的处理器，Cortex-M0 的设计进行了许多的改革与创新，如系统存储器地址映像(System Address Map)、改善效率并增强确定性的嵌套向量中断系统(NVIC)与不可屏蔽中断(NMI)、全新的调试单元等等，都带给了使用者全新的体验和更便利、更有效率的操作。

Cortex-M0 其核心架构为 ARMv6M，其运算能力可以达到 0.9 DMIPS/MHz，而与其他的 16 位与 8 位处理器相比，由于 CortexM0 的运算性能大幅提高，所以在同样任务的执行上 Cortex-M0 只需较低的运行速度，而大幅降低了整体的动态功耗。

有关内核的更多信息请参考 ARM 官方技术手册“Cortex-M0 Technical Reference Manual”（汉译版本为“Cortex-M0 技术参考手册”）。

本小节重点描述 NVIC 控制器、SysTick 定时器、LKS32MC08x 中断向量表定义及中断向量表重定义三个功能。

3.2 NVIC 控制器

为了管理中断请求的优先级并处理其他异常，Cortex-M0 处理器内置了嵌套中断控制器（NVIC）。NVIC 的一些可编程控制器控制着中断管理功能，这些寄存器被映射到系统地址空间里，它们所处的区域被称为系统控制空间（SCS）。SCS 的地址空间为：0xE000_E000 – 0xE000_EFFF。

NVIC 有以下特性：

➢ 灵活的中断管理；

Cortex-M0 处理器中，每一个外部中断都可以被使能或者禁止，并且可以被设置为挂起状态或者清除状态。处理器的中断可以是电平信号的（在中断服务程序清除中断请求以前，外设的请求会一直保持），也可以是脉冲信号的（最小一个始终周期），这样中断控制器就可以处理任何中断源。

➢ 支持嵌套中断；

Cortex-M0 处理器的任何中断都有一个固定或者可编程的中断优先级。当外部中断之类的异常发生时，NVIC 将该异常的优先级与当前的优先级进行比较，如果新的优先级更高，当前的任务会被暂停，处理器进行核心寄存器入栈保持，然后开始处理新的异常程序，这个过程也被称为“抢占”。高优先级的中断完成后，异常返回就会执行，处理器自动进行出栈操作恢复刚才寄存器的值，并继续运行刚才的任务。这种机制并没有带来软件开销。

➢ 向量化的异常入口；

异常发生时，处理器需要定位异常对用的程序入口。传统的处理方式需要软件去完成。而 M0 处理器会从存储器的向量表中，自动定位异常的程序入口。从异常到异常的处理事件会被缩减。

➢ 中断屏蔽;

NVIC 通过一组特殊寄存器提供了一种中断屏蔽机制，NVIC 除了硬件错误（HardFault）和 NMI 之外，可以屏蔽所有的异常。有些操作，比如对时间敏感的控制任务或实时多媒体解码任务，不应该被打断，此时中断屏蔽的作用就表现了出来。具体参见 ARM 手册。

LKS32MC08x 支持 ARM 公司推出的 CMSIS（Cortex Microcontroller Software Interface Standard，即 Cortex 微控制器软件接口标准）函数。用户在工程中包含 cm0_core.h 和 cm0_core.c 代码，即可实现对 NVIC 模块的操作。

3.3 异常和中断

异常，会引起程序控制的变化。在异常发生时，处理器停止当前的任务，转而执行异常处理程序，异常处理完成后，会继续执行刚才的任务。异常分为很多种，中断是其中之一。Cortex-M0 处理器最多支持 32 个外部中断（IRQ）和一个不可屏蔽中断（NMI），中断事件的处理叫做中断服务程序（ISR），中断一般由芯片上的硬件资源产生。

每一个异常都对应一个异常编号，异常编号还指明了异常向量的地址。异常编号和中断编号是相互独立的。系统异常使用负数定义，中断使用 0-31 正数定义。复位是一种特殊的异常，数值为-15。除了 NMI，HardFault 和复位，其他所有异常的优先级都是可编程的，NMI 和 HardFault 的优先级是固定的，并且比其他异常的优先级高。

LKS32MC08x 系列芯片共使用了其中的 21 个外部中断，后 11 个保留未使用。NMI 未实现。最多支持 4 个中断优先级可供编程选择。

表 3-1 中断号分布

<table>
<tr><th>异常/中断号</th><th>说明</th><th>异常/中断号</th><th>说明</th></tr>
<tr><td>-14</td><td>NMI</td><td rowspan="15" colspan="2"></td></tr>
<tr><td>-13</td><td>HardFault</td></tr>
<tr><td>-12</td><td rowspan="7">保留</td></tr>
<tr><td>-11</td></tr>
<tr><td>-10</td></tr>
<tr><td>-9</td></tr>
<tr><td>-8</td></tr>
<tr><td>-7</td></tr>
<tr><td>-6</td></tr>
<tr><td>-5</td><td>SVCall</td></tr>
<tr><td>-4</td><td rowspan="2">保留</td></tr>
<tr><td>-3</td></tr>
<tr><td>-2</td><td>PendSV</td></tr>
<tr><td>-1</td><td>SysTick</td></tr>
<tr><td>0</td><td>TIMER0</td><td>16</td><td>WAKEUP，系统唤醒中断</td></tr>
<tr><td>1</td><td>TIMER1</td><td>17</td><td>电源电压过低</td></tr>
<tr><td>2</td><td>TIMER2</td><td>18</td><td>DMA</td></tr>
<tr><td>3</td><td>TIMER3</td><td>19</td><td>CAN</td></tr>
</table>

4	ENCODER0	20	SIF
5	ENCODER1	21	Reserved
6	I2C	22	Reserved
7	GPIO	23	Reserved
8	UART0	24	Reserved
9	HALL	25	Reserved
10	SPI	26	Reserved
11	ADC	27	Reserved
12	DSP	28	Reserved
13	MCPWM	29	Reserved
14	UART1	30	Reserved
15	CMP	31	Reserved

异常发生时，处理器需要定位异常对应的程序入口。标准的 Cortex-M0 处理器不支持中断向量表的重定义，即每个异常的入口地址是固定的。在线升级应用，将擦除 FLASH 部分内容，用户可能擦除/破坏中断向量表。因此，此种应用，一般推荐关闭中断。LKS32MC08x 芯片增加了该功能，实现了中断向量表的重定义功能。即在升级过程中，可开启中断，编程更灵活。具体使用，参见本文档 FLASH 章节--FLASH 在线升级(IAP)部分的描述。

3.4 SysTick 定时器

SysTick 定时器是 Cortex-M0 处理器自带的系统滴答定时器。其存在的主要目的是为嵌入式操作系统提供 100Hz(即 10ms)的定时节拍。当然，也可以做普通定时等其他用途。

- 可编程设置频率的 RTOS 定时器(例如 100 Hz)，调用一个 SysTick 服务程序。
- 简单计数器。软件可使用它测量时间 （如：完成任务所需时间、已使用时间）。

SysTick 定时器使用起来非常简单。它一共有三个寄存器：SYST_CSR、SYST_RVR、SYST_CVR。寄存器的基地址为 0xE000_E000。LKS32MC08x 芯片，针对实际应用，三个寄存器的配置如下。

表 3-2 中断号分布

名称	偏移	描述
SYST_CSR	0x10	SysTick 定时器控制和状态寄存器
SYST_RVR	0x14	SysTick 定时器重载值寄存器
SYST_CVR	0x18	SysTick 定时器当前值寄存器

3.4.1 SYST_CSR 控制和状态寄存器

地址：0xE000_E010

复位值：0x00000000

表 3-3 SYST_CSR 控制和状态寄存器

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

	16
	FLAG
	R
	0

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

位置	位名称	说明
[16]	FLAG	SysTick 定时器计数器回零标志位，读清除。 0：没有回零 1：发生回零
[15:3]	--	保持为 0
[2]	CLKSRC	SysTick 定时器时钟选择信号。默认为 0。 0：CPU 时钟/8 1：CPU 时钟
[1]	TICKINT	SysTick 定时器中断使能信号。默认为 0。 0：关闭 1：使能
[0]	ENABLE	SysTick 定时器使能信号。默认为 0。 0：关闭 1：使能

3.4.2 SYST_RVR 重载值寄存器

地址：0xE000_E014

复位值：0x00000000

表 3-4 SYST_RVR 重载值寄存器

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

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

位置	位名称	说明

[31:24]	--	保持为 0
[23:0]	RELOAD	SysTick 定时器周期寄存器，定时器在 0—RELOAD 之间计数。默认为 0。

3.4.3 SYST_CVR 当前值寄存器

地址：0xE000_E018

复位值：0x00000000

表 3-5 SYST_CVR 当前值寄存器

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

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

位置	位名称	说明
[31:24]	--	保持为 0
[23:0]	CURRENT	读取该寄存器，返回 SysTick 定时器内部计数值，默认为 0。对该寄存器写入任何值，将复位内部计数值，同时清除 SYST_CSR.FLAG。

4 模拟电路

4.1 简述

模拟电路包含以下模块:

- 集成 1 路同步双采样的 12bit SAR ADC，采样及转换速率 3Msps。最多 20 通道
- 集成 4 路运算放大器，可设置为 PGA 模式
- 集成两路比较器，可设置迟滞模式
- 集成 12bit 数模转换器
- 内置±2°C温度传感器
- 内置 1.2V 0.8%精度电压基准源

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。

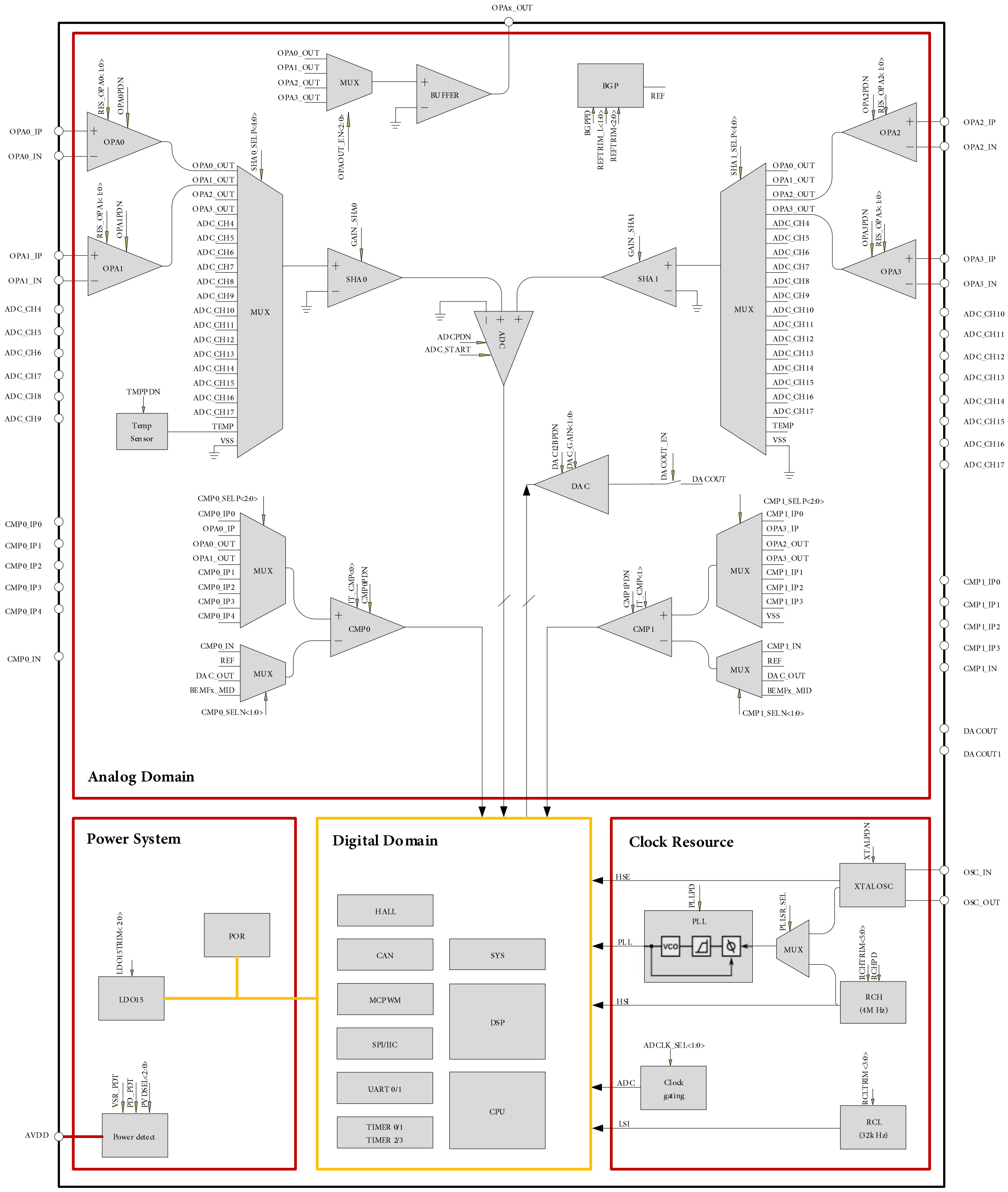


图4-1模拟电路功能框图

4.2 POWER 电源管理系统

电源管理系统由电源检测模块（PVD）、上电/掉电复位模块（POR）组成。

POR 模块监测 AVDD 的电压，在 AVDD 电压低于 3.0V 时（例如上电之初，或者掉电时），为数字电路提供复位信号以避免数字电路工作产生异常。

PVD 模块对 5V 输入电源进行检测，如低于某一设定阈值，则产生报警（中断）信号以提醒 CPU。中断提醒阈值可通过寄存器 PVDSEL[1:0]设置为不同的电压。PVD 模块可通过设置 PD_PDT=1 关闭。

当发生电压过低事件时，会触发产生电源电压过低中断，对应中断号 17，详见第 3 章。

PVDSEL[1:0]/ PD_PDT 的说明见模拟寄存器 SYS_AFE_REG6

电源电压过低标志见模拟前端信息寄存器 SYS_AFE_CMP

4.3 CLOCK 时钟系统

时钟系统包括内部 32kHz RC 时钟、内部 4MHz RC 时钟、外部 4~8MHz 晶体起振电路、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。4MHz RC 时钟可作为 CPU 主时钟使用，CPU 也可以使用 PLL 时钟，PLL 最高可提供 96MHz 的时钟。外部 4~8MHz 晶体起振电路作为备用时钟使用。

32kHz 和 4MHz RC 时钟均带有出厂校正，32kHz RC 时钟在-40~105°C范围内的精度为±50%，4MHz RC 时钟在该温度范围的精度为±1%。

4MHz RC 时钟通过设置 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，4MHz 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 4MHz RC 时钟进行倍频，给 CPU、ADC 等模块提供更高速的工作时钟。CPU 和 PWM 模块的最高时钟为 96MHz，ADC 模块最高时钟 48MHz，通过寄存器 ADCLKSEL[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 6us 的稳定时间来输出稳定时钟。芯片上电的默认状态下，RCH 时钟和 BGP 模块都是开启的，但 PLL 默认是关闭的，需要软件来开启。

晶体起振电路内置了放大器，需在 OSC_IN/OSC_OUT 管脚之间接入一个晶体，且在 OSC_IN/OSC_OUT 引脚各接一个 15pF 电容到地，设置 XTALPDN=1 即可起振。

ADCLKSEL<1:0>的说明见模拟寄存器 SYS_AFE_REG7

BGPPD/RCHPD/XTALPDN/PLLPDN 的说明见模拟寄存器 SYS_AFE_REG5

4.4 BGP 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 2us 达到稳定。BGP 输出电压约 1.2V，精度为±0.8%

BGPPD 的说明见模拟寄存器 SYS_AFE_REG5

4.5 ADC 模数转换器

请参加第 10 章 ADC

4.6 OPA 运算放大器

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

RES_OPAx<1:0>的说明见模拟寄存器 SYS_AFE_REG0

放大器的结构示意图如下所示：

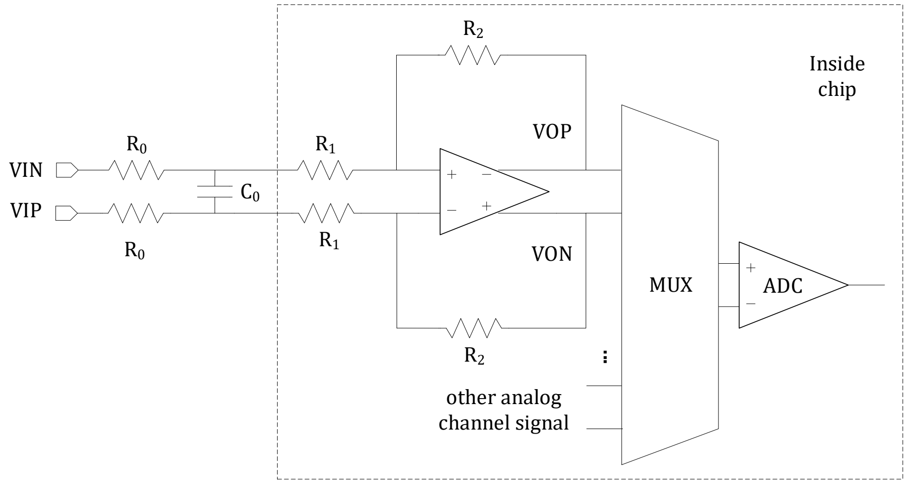


图 4-2 放大器框图

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

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

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

放大器可通过设置 OPAOUT_EN<2:0>选择将 4 路放大器中的某一路输出信号通过 BUFFER 送至 P2.7 管脚口进行测量和应用（对应关系见 datasheet 芯片管脚说明）。因为有 BUFFER 存在，在运放正常工作模式下也可以选择送一路运放输出信号出来。

OPAOUT_EN<2:0>的说明见模拟寄存器 SYS_AFE_REG2

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

OPAxPDN 的说明见模拟寄存器 SYS_AFE_REG5

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

4.7 CMP 比较器

内置 2 路输入轨到轨（rail-to-rail）比较器，比较器比较速度可编程、迟滞电压可编程、信号源可编程。比较器的比较延时可设置为 0.15uS/0.6uS。迟滞电压可设置为 20mV/0mV。支持多种比较器输入信号来源。

4.8 TMP 温度传感器

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

芯片出厂前会经温度校正，校正值保存在 flash info 区。

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

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

温度传感器信号连至 ADC 的通道 18。

ADC 部分的设置参考模数转换器(ADC)章节

TMPPDN 的说明见模拟寄存器 SYS_AFE_REG5

温度传感器的典型曲线如下图所示：

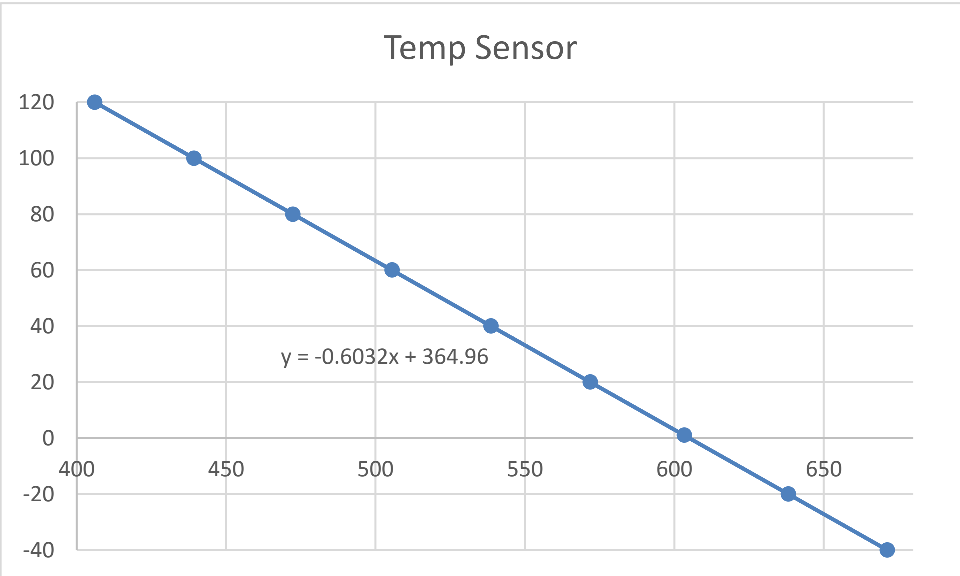


图 4-2 温度传感器曲线

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

y=-0.6032x+364.96

求得的 Y 值即为此时的温度。

公式中有两个系数，a=-0.6032, b=364.96。对于不同的芯片，b 系数的值是不一样的。芯片出厂前会经过温度标定，将每颗芯片所对应的系数 b 写入 flash 的 info 区，地址为 0x0000039C。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 0x00000398。存储时，将 a 系数小数点右移四位（乘 10000）存入 info 区。

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

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

4.9 DAC 数模转换器

芯片内置一路 12bit DAC，输出信号的最大量程可通过寄存器 DAC_GAIN<1:0>设置为 1.2V/3V/4.85V

12bit DAC 可通过配置寄存器 DACOUT_EN=1，将 DAC 输出送至 P0.0 管脚，可驱动>5kΩ 的负载电阻和 50pF 的负载电容。

DAC 最大输出码率为 1MHz。

芯片上电的默认状态下，DAC 模块是关闭的。DAC 可通过设置 DAC12BPDN =1 打开，开启 DAC 模块之前，需要先开启 BGP 模块。

DAC 的输入数字信号寄存器为 SYS_AFE_DAC，低 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 DAC_GAIN<1:0>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_DAC 填入值（理想值对应数字量）。a 来自 SYS_AFE_DAC_AMC 寄存器，b 来自 SYS_AFE_DAC_DC。硬件根据 SYS_AFE_DAC、SYS_AFE_DAC_AMC 和 SYS_AFE_DAC_DC 进行乘加从而求得校正后的数字量，送至 DAC 输入端，使得 DAC 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。

地址 0x00000330，为 3V 量程的 a 参数，地址 0x00000340，为 3V 量程的 b 参数。

地址 0x00000334，为 1.2V 量程的 a 参数，地址 0x00000344，为 1.2V 量程的 b 参数。

地址 0x00000338，为 4.85V 量程的 a 参数，地址 0x00000348，为 4.85V 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。

DACOUT_EN 的说明见模拟寄存器 SYS_AFE_REG3

DAC_GAIN<1:0>的说明见模拟寄存器 SYS_AFE_REG1

DAC12BPDN 的说明见模拟寄存器 SYS_AFE_REG5

SYS_AFE_DAC 的说明见寄存器 SYS_AFE_DAC

5 系统控制及时钟复位

5.1 时钟

5.1.1 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（Low Speed Internal Clock）/内部高速 RC 振荡时钟 HSI（High Speed Internal Clock）不会停振。外部晶振时钟 HSE（High Speed External Clock）在极端工况下可能失效，仅部分应用会使用 HSE。

表 5-1 系统时钟源

时钟源	频率	来源	误差	说明
LSI	32kHz	内部 RC 振荡器	常温 23~42kHz 全温度 16~48kHz	内部系统管理时钟，用于看门狗模块，以及复位信号的滤波和展宽
HSI	4MHz	内部 RC 振荡器	-40~105° 误差<±1% 105~125° 误差<±1.5%	可作为 PLL 源时钟
PLL	96MHz	PLL 时钟	0	PLL 输出时钟，以 HSI/HSE 作为参考时钟，输出是 HSI/HSE 时钟的 24 倍频，作为系统主时钟。
HSE	4~8MHz	外部晶体振荡器	0	外部晶体，在对时钟精度有严格要求(例如 ppm 级别的精度要求)的应用下，可使用 4MHz HSE 作为 PLL 参考时钟来产生 96MHz 的系统主时钟，不支持使用其他频率 HSE 作为 PLL 参考时钟
SWD	1MHz*	调试器	-	SWD 的 JTAG 时钟

*SWD 时钟速率典型值，实际大小与硬件环境有关。

另外，晶振起振电路可以支持 4~8MHz 的外部晶振频率，但如果需要使用 HSE 作为 PLL 参考时钟，则仍需使用 4MHz 晶体，以提供与 HSI 相同频率的 PLL 参考时钟。如果直接使用 HSE 作为系统主时钟，则无此限制。

如

图 5-1 所示，系统可以使用外部晶振时钟 HSE 或内部高速 RC 时钟 HSI 作为 PLL 的参考时钟，使用 SYS_AFE_REG6.PLLSR_SEL 进行选择。同时，芯片内部可以检测 HSE 时钟是否停振，如果 HSE 失效会自动切换至 HSI 作为 PLL 参考时钟。PLL 将 4MHz 的参考时钟 CLK_HS 倍频 24 倍至 96MHz。

PLL 经过 n/8 分频后，可以得到 96MHz×n/8 的高速时钟，SYS_CLK_CFG.CLK_SEL 在此分频后的高速时钟与 4MHz 的 CLK_HS 之间进行二选一，作为系统主时钟 MCLK。系统复位时，PLL 默认关

闭，HSI 默认开启，系统选择 HSI 时钟，即 4MHz 作为系统主时钟进行工作，从而保证系统上电之初功耗处于较低水平。

MCLK 是系统主时钟。可以通过 SYS_CLK_CFG 寄存器 CLK_DIV 位域控制进行 n/8 分频，可以产生 12,24,36,48,60,72,84,96MHz 等频率值。SYS_CLK_CFG.CLK_SEL 表示选择 PLL 或 CLK_HS 作为系统主时钟。当 SYS_CLK_CFG.CLK_SEL 为 1 时，SYS_CLK_CFG.CLK_DIV 作为 PLL 的分频系数。当 SYS_CLK_CFG.CLK_SEL 为 0 时，SYS_CLK_CFG.CLK_DIV 不起任何作用。

表 5-2 PLL 作为 MCLK 时钟时的分频配置

SYS_CLK_CFG	分频系数	频率/MHz	是否均匀
0x0101	1/8	12	是
0x0111	2/8	24	是
0x0115	3/8	36	否
0x0155	4/8	48	是
0x0157	5/8	60	否
0x0177	6/8	72	否
0x017F	7/8	84	否
0x01FF	8/8	96	是

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

PLL 输出的时钟经过 SYS_AFE_REG7.ADCLKSEL 控制的 2/4/8 分频后送至 ADC（典型工作频率 48MHz），即 ACLK。

内部 32kHz RC 产生一路 LSI 时钟 LCLK，主要用于 WDT 工作时钟，以及部分系统控制，复位的滤波展宽等。

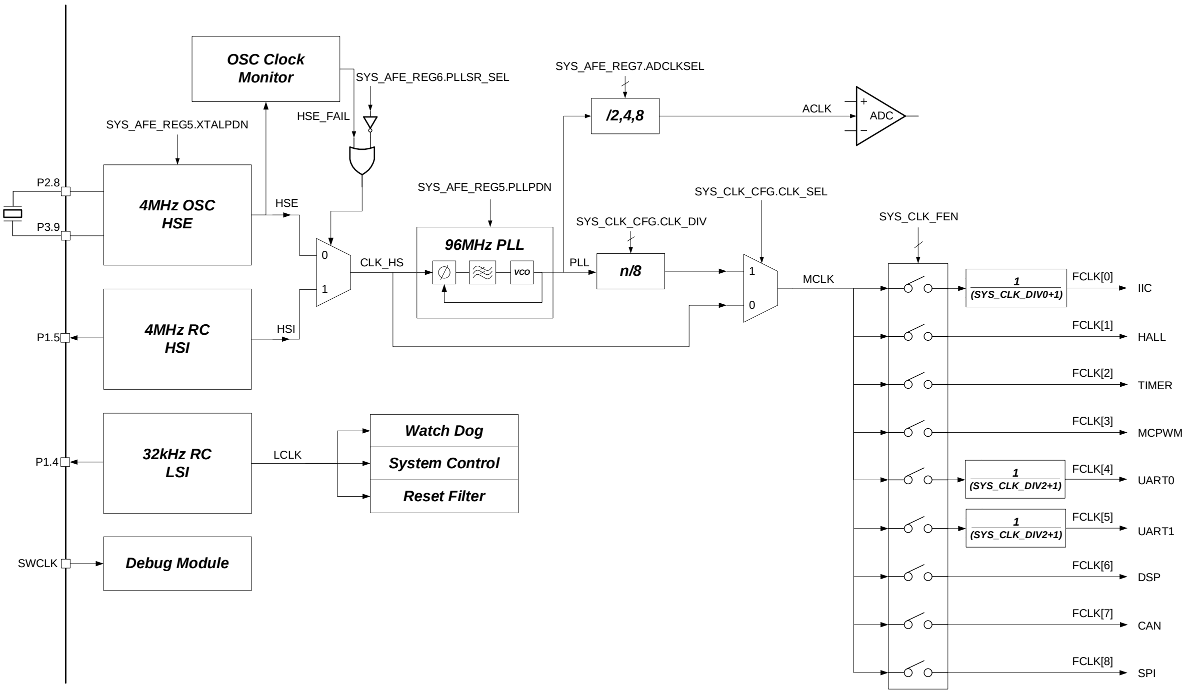


图 5-1 时钟架构

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 或 HSE 无法被软件关闭；当 CLK_HS 作为系统工作主时钟时，作为 CLK_HS 的 HSI 或 HSE 无法被关闭；当 HSE 被使用时，如果系统检测到 HSE 停振会自动切换到使用 HSI 时钟来代替 HSE 时钟。32kHz LSI 时钟上电即工作，且无法关闭。SWCLK 由调试器提供，频率可在调试界面进行选择。

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

5.1.2 功耗管理及休眠唤醒

5.1.2.1 休眠

MCLK 可以通过配置进行门控，从而令包括 CPU 和所有外设在内的大部分数字电路处于休眠状态。门控时，PMU 状态机依次关闭 PLL, HSI/HSE, BGP 等模拟模块，以降低功耗。

需要注意的是，系统的休眠模式仅仅关闭 PLL，HSI，HSE 等高速时钟，LSI 时钟仍然存在。工作于 LSI 时钟的看门狗如果被使能，看门狗复位作为全局复位可以令系统回到初始状态重新开始工作。

向 SYS_CLK_SLP 寄存器写入 0xDEAD 可以令芯片准备进入休眠状态，之后立刻执行__WFI()宏指令使得 CPU 停止取指。

在应用程序编写中请事先配置好唤醒条件。

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

5.1.2.2 唤醒

休眠后，外部 IO 事件、内部唤醒 Timer 均可以作为唤醒源。

内部唤醒 Timer 为独立于 UTimer 模块的独立 Timer，使用 LSI 时钟，不同于系统中的通用 Timer 工作于系统主时钟。唤醒 Timer 可以使用 SYS_RST_CFG.WK_INTV 配置 0.25s,0.5s,1s,2s,4s,8s,16s,32s 共 8 档唤醒时间间隔，具体请参考 5.2.14 章节。

仅有 P0[1:0]、P1[1:0]四个 IO 可以作为外部唤醒 IO 使用，可以配置独立的使能和极性。具体寄存器配置请参考 8.2.13.5 WAKE_POL 和 8.2.13.6 WAKE_EN 章节。需要注意的是，由于外部 IO 唤醒属于电平触发，如果外部 IO 在芯片休眠之前处于唤醒电平，会导致芯片休眠后立刻唤醒。

在应用程序编写中请尽量避免上电即进入休眠状态，如果使用内部唤醒 Timer 作为唤醒源，且唤醒后立即再次睡眠，会导致普通下载器无法进行连接调试，此时需要使用芯片供应商提供的离线下载器进行应用程序擦除改写。

5.1.2.3 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 寄存器门控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，共有 8 路可关闭的外设时钟分别供给不同的外设模块，详见

图 5-1 时钟架构。门控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。

I2C 使用 FCLK[0]

Hall 模块使用 FCLK [1]

Timer 模块使用 FCLK [2]

MCPWM 模块使用 FCLK [3]

UART0/UART1 分别使用 FCLK [4]/ FCLK [5]

DSP 使用 FCLK [6]

CAN 使用 FCLK [7]

SPI 使用 FCLK [8]

5.1.2.4 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_DIV[0]作为分频系数，UART0/1 共享 SYS_CLK_DIV[2]作为分频系数。UART 的波特率在 UART 模块内部还有一个额外的分频器，详见

图 5-1 时钟架构。

5.1.3 复位源

芯片的复位来源包括硬件复位与软件复位。

5.1.3.1 硬件复位

如表 5-3 硬件复位源所示，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 5-3 硬件复位源

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 3.0V 时产生复位
RSTn	外部按键	外部 RC 组成按键复位电路
WDTn	硬件看门狗	如果不进行软件喂狗则定时产生复位，复位间隔可配置

5.1.3.1.1 硬件复位架构

如下图所示，LPORn/HPORn 来自内部模拟电路， RSTn 来自外部按键

经过滤波展宽预处理的复位信号进行与运算得到一个复位信号。

P0.2 引脚持续时间小于 32us 的复位脉冲会被滤除，要求可靠复位宽度大于 200us。

4 个复位信号复位等级和作用域一致，均为全局复位。

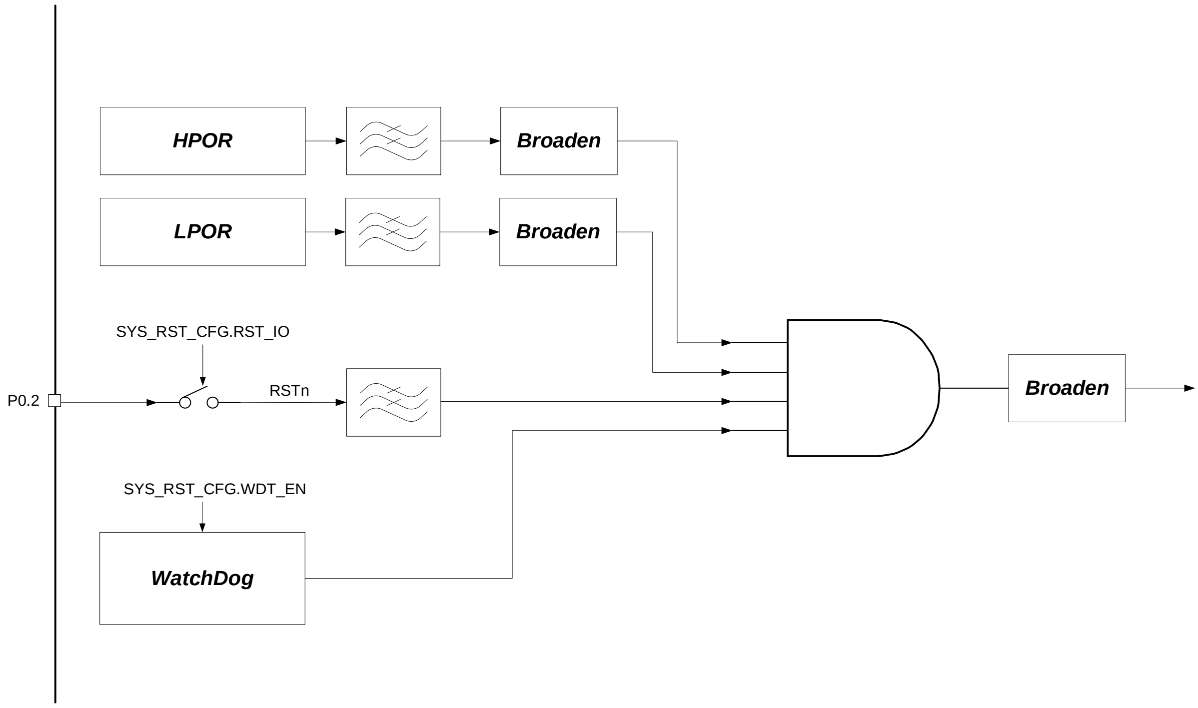


图 5-2 硬件复位架构

5.1.3.1.2 硬件复位记录

SYS_RST_SRC 寄存器用于保存硬件复位事件，当某个硬件复位发生后，SYS_RST_SRC 对应位置位。SYS_RST_SRC 寄存器本身无法被复位信号复位，只能通过向 SYS_CLR_RST 寄存器写入 0xCA40 清空记录，复位记录可以方便地了解是否发生以及发生过何种复位。

5.1.3.2 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 5.2.22。

5.1.4 复位作用域

表 5-4 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
WDTn	硬件看门狗，全局复位，除极少数寄存器

SYS_SFT_RST.SPI_SFT_RST	SPI 模块
SYS_SFT_RST.CAN_SFT_RST	CAN 模块
SYS_SFT_RST.DSP_SFT_RST	DSP 模块
SYS_SFT_RST.UART1_SFT_RST	UART1 模块
SYS_SFT_RST.UART0_SFT_RST	UART0 模块
SYS_SFT_RST.MCPWM_SFT_RST	MCPWM 模块
SYS_SFT_RST.UTIMER_SFT_RST	UTIMER 模块
SYS_SFT_RST.HALL_SFT_RST	HALL 模块
SYS_SFT_RST.I2C_SFT_RST	I2C 模块
NVIC_SystemReset();	CPU 软复位，仅复位 CPU 内核，将 PC 重置为 0，所有外设寄存器值仍然维持。

其中用于控制 P0[2]作为 GPIO 使用还是作为外部复位脚使用的 SYS_RST_CFG.RST_IO，用于控制看门狗使能的 SYS_RST_CFG.WDT_EN，以及复位记录寄存器 SYS_RST_SRC 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。

5.2 寄存器

5.2.1 地址分配

系统模块寄存器基地址为 0x4000_0000，寄存器列表如下：

表 5-5 系统控制寄存器

名称	偏移	说明
	0x00~0x08	保留
	0x10~0x14	保留
SYS_AFE_CMP	0x18	模拟前端信息寄存器
	0x1C	保留
SYS_AFE_REG0	0x20	模拟配置寄存器 0
SYS_AFE_REG1	0x24	模拟配置寄存器 1
SYS_AFE_REG2	0x28	模拟配置寄存器 2
SYS_AFE_REG3	0x2C	模拟配置寄存器 3
SYS_AFE_REG4	0x30	模拟配置寄存器 4
SYS_AFE_REG5	0x34	模拟配置寄存器 5
SYS_AFE_REG6	0x38	模拟配置寄存器 6
SYS_AFE_REG7	0x3C	模拟配置寄存器 7
	0x54~0x78	保留
SYS_AFE_DAC	0x7C	DAC 数字量寄存器

SYS_CLK_CFG	0x80	时钟控制寄存器
SYS_RST_CFG	0x84	复位控制寄存器
SYS_RST_SRC	0x88	复位源记录寄存器
SYS_CLR_RST	0x8C	复位源记录清除寄存器
SYS_CLK_DIV0	0x90	外设时钟分频寄存器 0
	0x94	保留
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_CLK_SLP	0xA0	休眠寄存器
	0xA4	保留
SYS_TRIM	0xA8	校正模式寄存器
SYS_SFT_RST	0xAC	软复位寄存器
SYS_WR_PROTECT	0xB0	写保护寄存器
SYS_DAC_AMC	0xB4	DAC 增益校正寄存器
SYS_DAC_DC	0xB8	DAC 直流偏置寄存器

5.2.2 SYS_AFE_CMP 模拟前端信息寄存器

地址：0x4000_0018

复位值：0x0

表 5-6 模拟前端信息寄存器 SYS_AFE_CMP

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

名称	复位值	偏移	位置	权限	位名称	说明
SYS_AFE_CMP	0x0	0x18	[31:16]	NA		未使用
			[15]	R	CMP1_RESULT	CMP1 输出结果寄存器
			[14]	R	CMP0_RESULT	CMP0 输出结果寄存器
			[13]	R	PWR_WEAK	供电低于掉电监测阈值
			[12:0]	NA		未使用

SYS_AFE_CMP 中的 CMP0/CMP1 为比较器的原始输出，没有经过滤波处理。CMP0/CMP1 也可以通过配置 GPIO 的第二功能（AF1）输出，具体输出引脚请查看器件 DATASHEET。

PWR_WEAK 标志在当供电电压低于掉电监测电路设定的阈值时置位，同时将产生 CPU 掉电中断（中断号 17）。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 4.2 POWER 电源管理系统。

5.2.3 模拟寄存器概述

模拟寄存器 0x40000020~0x4000003C 是开放给用户的寄存器，其中保留寄存器(Res)必须全部配置为 0（芯片上电后会被复位为 0）。其他寄存器根据应用场合需要进行配置。

下面是各个模拟寄存器的详细说明。

5.2.4 SYS_AFE_REG0 模拟配置寄存器 0

地址：0x4000_0020

复位值：0x0

表 5-7 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
Resv.	Resv.	Resv.	Resv.	REF_OPA3	REF_OPA2	REF_OPA1	REF_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
[15:14]	Reserved	保留位，需全部为’0’
[13:12]	Reserved	保留位，需全部为’0’
[11:10]	Reserved	保留位，需全部为’0’
[9:8]	Reserved	保留位，需全部为’0’
[7:6]	REF_OPA3	运放 3 反馈电阻 00: 200k:10.4k 01: 190k:20.4k 10: 180k:30.4k 11: 170k:40.4k
[5:4]	RES_OPA2	运放 2 反馈电阻 00: 200k:10.4k 01: 190k:20.4k 10: 180k:30.4k 11: 170k:40.4k
[3:2]	RES_OPA1	运放 1 反馈电阻 00: 200k:10.4k 01: 190k:20.4k 10: 180k:30.4k 11: 170k:40.4k
[1:0]	RES_OPA0	运放 0 反馈电阻 00: 200k:10.4k

		01: 190k:20.4k 10: 180k:30.4k 11: 170k:40.4k

运放反馈电阻的比例精度为 0.5%，具体精度数据可参考应用笔记《AN8012_运放反馈电阻值精度》

5.2.5 SYS_AFE_REG1 模拟配置寄存器 1

地址：0x4000_0024

复位值：0x0

表 5-8 模拟配置寄存器 1 SYS_AFE_REG1

15–8	7–6	5	4	3	2–1	0
Reserved	DAC_GAIN	REFVDD5	REF2VDD	GAIN_REF	IT_CMP	CMP_FT
RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0

位置	位名称	说明
[31:16]		未使用
[15]	Reserved	保留位，需全部为’0’
[14]	Reserved	保留位，需全部为’0’
[13]	Reserved	保留位，需全部为’0’
[12]	Reserved	保留位，需全部为’0’
[11:8]	Reserved	保留位，需全部为’0’
[7:6]	DAC_GAIN	DAC 输出档位配置 00: DAC 输出档位，满量程为 3V 01: DAC 输出档位，满量程为 1.2V 10: DAC 输出档位，满量程为 4.85V（注意，此时供电电源需为 5V，否则 DAC 输出异常） 11: 错误配置
[5]	REFVDD5	电源作为 ADC REF 且电源为 5V 时，增加运放输出的信号范围 0: 运放最大输出范围为 3.3V 1: 增加运放最大输出范围到 4.8V，需在 REF2VDD 配置为’1’时才可设置为’1’
[4]	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，此时 GAIN_REF 配置无效; 0: 使用默认内部 REF 作为 ADC 基准
[3]	GAIN_REF	ADC 基准电压调节，采用默认配置 0: ×2 1: ×1
[2:1]	IT_CMP	IT_CMP<1>:比较器 1 的比较速度选择 0: 150ns 1: 600ns

		IT_CMP<0>:比较器 0 的比较速度选择 0: 150ns 1: 600ns
[0]	CMP_FT	使能比较器快速比较 1: 在 IT_CMP<1:0>都为默认’00’的时候，比较器比较速度小于 30ns 0: 不使能，比较速度维持 IT_CMP 设置里的参数

5.2.6 SYS_AFE_REG2 模拟配置寄存器 2

地址：0x4000_0028

复位值：0x0

表 5-9 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
Reserved		Reserved				Reserved		Reserved					OPAOUT_EN		
RW		RW				RW		RW					RW		
0		0				0		0					0		

位置	位名称	说明
[31:16]		未使用
[15:14]	Reserved	保留位，需全部为’0’
[13]	Reserved	保留位，需全部为’0’
[12]	Reserved	保留位，需全部为’0’
[11:10]	Reserved	保留位，需全部为’0’
[9:8]	Reserved	保留位，需全部为’0’
[7]	Reserved	保留位，需全部为’0’
[6]	Reserved	保留位，需全部为’0’
[5:4]	Reserved	保留位，需全部为’0’
[3]	Reserved	保留位，需全部为’0’
[2:0]	OPAOUT_EN	使能 OPAx 输出信号送至 IO 口 p2_7 000:不输出; 001:输出 OPA0 信号到 IO 口; 010:输出 OPA1 信号到 IO 口; 011:输出 OPA2 信号到 IO 口; 100:输出 OPA3 信号到 IO 口; 101~111: 禁止

5.2.7 SYS_AFE_REG3 模拟配置寄存器 3

地址：0x4000_002C

复位值：0x0

表 5-10 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.	CMP1_SELP	DACOUT_EN	CMP0_SELP	CMP_HYS	Res.	CMP1_SELN	CMP0_SELN	Res.	LDOOUT_EN
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:16]		未使用
[15]	Reserved	保留位，需全部为'0'
[14:12]	CMP1_SELP	比较器 1 信号正端选择 000: 连 CMP1_IP0 001: 连 OPA3_IP 010: 连 OPA2_OUT 011: 连 OPA3_OUT 100: 连 CMP1_IP1 101: 连 CMP1_IP2 110: 连 CMP1_IP3 111: 连 AVSS 说明:上述除 AVSS/OPA2_OUT/OPA3_OUT 外都为管脚名称，请参看 datasheet 里管脚定义章节
[11]	DACOUT_EN	DAC 输出到 IO 使能 0: 不使能 1:使能输出到 IO P0[0]
[10:8]	CMP0_SELP	比较器 0 信号正端选择 000: 连 CMP0_IP0 001: 连 OPA0_IP 010: 连 OPA0_OUT 011: 连 OPA1_OUT 100: 连 CMP0_IP1 101: 连 CMP0_IP2 110: 连 CMP0_IP3 111: 连 CMP0_IP4 说明:上述除 OPA0_OUT/OPA1_OUT 外都为管脚名称，请参看 datasheet 里管脚定义章节
[7]	CMP_HYS	比较器回差选择，采用默认配置 0: 20mv 1: 0mv
[6]	Reserved	保留位，需全部为'0'
[5:4]	CMP1_SELN	比较器 1 信号负端选择 00: 连 CMP1_IN 01: 连 REF 10: 连 DAC 输出 11: 连 BEMF1_MID

		说明:上述 CMP1_IN 为管脚名称，请参看 datasheet 里管脚定义章节；REF 为芯片内部 1.2V BANDGAP 基准源；DAC 输出即为芯片内部 DAC 模块输出模拟信号；BEMF1_MID 为 CMP1_IP1, CMP1_IP2, CMP1_IP3 信号经电阻星形连接后得到的平均值
[3:2]	CMP0_SELN	比较器 0 信号负端选择 00: 连 CMP0_IN 01: 连 REF 10: 连 DAC 输出 11: 连 BEMF0_MID 说明: 上述 CMP0_IN 为管脚名称，请参看 datasheet 里管脚定义章节；REF 为芯片内部 1.2V BANDGAP 基准源；DAC 输出即为芯片内部 DAC 模块输出模拟信号；BEMF0_MID 为 CMP0_IP1, CMP0_IP2, CMP0_IP3 信号经电阻星形连接后得到的平均值
[1]	Reserved	保留位，需全部为’0’
[0]	LDOOUT_EN	LDO 输出到 IO 使能 0: 不输出 1:使能输出到 IO P2.7

5.2.8 SYS_AFE_REG4 模拟配置寄存器 4

地址：0x4000_0030

复位值：0x0

表 5-11 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
Res.		Res.								Res.					
RW		RW								RW					
0		0								0					

位置	位名称	说明
[31:16]		未使用
[15]	Reserved	保留位，需全部为’0’
[14]		未使用
[13]	Reserved	保留位，需全部为’0’
[12:6]		未使用
[5]	Reserved	保留位，需全部为’0’
[4:0]		未使用

5.2.9 SYS_AFE_REG5 模拟配置寄存器 5

地址：0x4000_0034

复位值：0x0

表 5-12 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
PLLPDN	XTALPDN	TMPPDN	DAC12BPDN	Res.	RCHPD	Res.	BGPPD	CMP1PDN	CMP0PDN	OPA3PDN	OPA2PDN	OPA1PDN	OPA0PDN	Res.	ADCPDN
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:16]		未使用
[15]	PLLPDN	PLL 关闭控制 0:关闭 PLL（默认） 1:开启 PLL
[14]	XTALPDN	晶体起振电路开启使能 0:关闭（默认） 1:开启
[13]	TMPPDN	温度传感器开启使能 0:关闭 1:开启
[12]	DAC12BPDN	12BIT DAC 开启使能 0:关闭 1:开启
[11]	Reserved	保留位，需全部为'0'
[10]	RCHPD	RCH 时钟关闭使能 0:开启 1:关闭
[9]	Reserved	保留位，需全部为'0'
[8]	BGPPD	BGP 开启使能 0:开启 1:关闭
[7]	CMP1PDN	CMP1 开启使能 0:关闭 1:开启
[6]	CMP0PDN	CMP0 开启使能 0:关闭 1:开启
[5]	OPA3PDN	OPA3 开启使能 0:关闭 1:开启
[4]	OPA2PDN	OPA2 开启使能

		0:关闭 1:开启
[3]	OPA1PDN	OPA1 开启使能 0:关闭 1:开启
[2]	OPA0PDN	OPA0 开启使能 0:关闭 1:开启
[1]	Reserved	保留位，需全部为'0'
[0]	ADCPDN	ADC 开启使能 0:关闭 1:开启

如果 SYS_CLK_CFG 选择 PLL 时钟，则 PLLPDN 是被硬件控制的，软件配置 PLLPDN 关闭 PLL 无效。关闭 PLL 需要 PLLPDN=0，且 SYS_CLK_CFG 不选择 PLL 作为芯片主时钟，这两个条件都须满足。

同理，如果 SYS_CLK_CFG 选择了 HRC 时钟，则 RCHPD 是被硬件控制的，软件配置 RCHPD 关闭 RCH 无效。关闭 PLL 需要 RCHPD=1，且芯片进入休眠。

如果芯片主时钟为 PLL 时钟，且 HRC 为 PLL 参考时钟，则 RCH 也是被硬件控制的。

由于 RCH 和 PLL 依赖 BGP（bandgap），所以 BGPPD 也是硬件控制的，在芯片使用了 RCH 或 PLL 时，软件配置 BGPPD 关闭 BGP 无效。关闭 BGP 需要先顺序关闭 PLL 和 RCH，且芯片进入休眠。

5.2.10 SYS_AFE_REG6 模拟配置寄存器 6

地址：0x4000_0038

复位值：0x0

表 5-13 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
PLLSR_SEL	Reserved					PVDSEL		Reserved					Reserved	VSR_PDT	PD_PDT
RW	RW					RW		RW					RW	RW	RW
0	0					0		0					0	0	0

位置	位名称	说明
[31:16]		未使用
[15]	PLLSR_SEL	PLL 时钟源选择 0: 使用 RCH 作为输入时钟源; 1: 使用 XTAL OSC 作为输入时钟源
[14]	Reserved	保留位，需全部为'0'
[13]	Reserved	保留位，需全部为'0'

[12]	Reserved	保留位，需全部为'0'
[11]	Reserved	保留位，需全部为'0'
[10]	Reserved	保留位，需全部为'0'
[9:8]	PVDSEL	电源掉电监测阈值选择 当 VSR_PDT=0 时， 00: 4.5V 01: 4.2V 10: 3.9V 11: 3.6V 当 VSR_PDT=1 时， 4 个档位与 DAC 设定值有关，分别为 00: 4.5/1.35*DAC V 01: 4.2/1.35*DAC V 10: 3.9/1.35*DAC V 11: 3.6/1.35*DAC V
[7]	Reserved	保留位，需全部为'0'
[6:5]	Reserved	保留位，需全部为'0'
[4:3]	Reserved	保留位，需全部为'0'
[2]	Reserved	保留位，需全部为'0'
[1]	VSR_PDT	掉电检测基准源选择，其中低功耗基准源为 1.35V 左右，DAC 输出则可以通过软件配置；低电压检测的阈值有 4 档可选，通过 PVDSEL 选择 0: 选择低功耗基准源; 1: 选择 DAC 输出作为掉电检测基准源
[0]	PD_PDT	关闭掉电检测电路 0: 开启 1: 关闭

5.2.11 SYS_AFE_REG7 模拟配置寄存器 7

地址：0x4000_003C

复位值：0x0

表 5-14 模拟配置寄存器 7 SYS_AFE_REG7

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

位置	位名称	说明
[31:16]		未使用
[15]	Reserved	保留位，需全部为'0'
[14]	Reserved	保留位，需全部为'0'
[13:8]	Reserved	保留位，需全部为'0'

[7:6]	Reserved	保留位，需全部为'0'
[5:4]	ADCLKSEL	ADC 时钟频率选择 00: 48MHz 01: 禁止 10: 12MHz 11: 24MHz
[3:0]	Reserved	保留位，需全部为'0'

5.2.12 SYS_AFE_DAC DAC 数字量寄存器

地址：0x4000_007C

复位值：0x0

表 5-15 DAC 数字量寄存器 SYS_AFE_DAC

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

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

5.2.13 SYS_CLK_CFG 时钟控制寄存器

地址：0x4000_0080

复位值：0x0

表 5-16 时钟控制寄存器 SYS_CLK_CFG

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

位置	位名称	说明
[31:9]		未使用
[8]	CLK_SEL	CLK_HS/PLL 选择，0: CLK_HS, 1:PLL。默认选择 CLK_HS，CLK_HS 根据是否使用外部晶体时钟，可以是 HSI 或 HSE。 PLL 在上电后默认关闭，需要软件来开启
[7:0]	CLK_DIV	PLL 输出分频控制，选择 8 个时钟周期中，哪些周期输出时钟，例如

		8’b00000001，表示 1/8 分频，8’b00010001 表示 1/4 分频，8’b00100101 表示 1/3 分频，但不均匀。

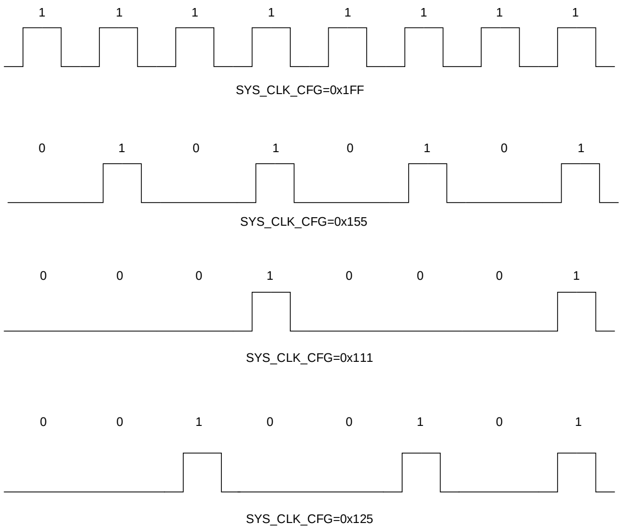


图 5-3 SYS_CLK_CFG 不同配置时的系统主时钟分频波形

当使用 4MHz HSI 时钟作为系统主时钟时，SYS_CLK_CFG[7:0]的分频系数无效，最终输出的系统时钟频率即为 4MHz。

5.2.14 SYS_RST_CFG 复位控制寄存器

地址：0x4000_0084

复位值：0x0

表 5-17 复位控制寄存器 SYS_RST_CFG

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

位置	位名称	说明
[31:6]		未使用
[5]	RST_IO	RSTn/P0[2]复用选择， 0: RSTn, 1:P0[2] 当用作外部复位引脚时，片内上拉默认开启，且不可控。 当用作 GPIO P0[2]时，片内上拉默认关闭，且不可控。
[4:2]	WK_INTV	休眠唤醒间隔设置 000：0.25S　　100：4S 001：0.5S　　101：8S 010：1S　　110：16S 011：2S　　111：32S
[1]		未使用

[0]	WDT_EN	看门狗使能，高有效。

5.2.15 SYS_RST_SRC 复位源记录寄存器

地址：0x4000_0088

复位值：0x0

表 5-18 复位源记录寄存器 SYS_RST_SRC

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

位置	位名称	说明
[31:4]		未使用
[3]	WDT_RST_RCD	看门狗复位发生标志，高有效
[2]	KEY_RST_RCD	按键复位发生标志，高有效
[1]	HPOR_RST_RCD	HPOR 复位发生标志，高有效
[0]	LPOR_RST_RCD	LPOR 复位发生标志，高有效

5.2.16 SYS_CLR_RST 复位源记录清除寄存器

地址：0x4000_008C

复位值：0x0

表 5-19 复位源记录清除寄存器 SYS_CLR_RST

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 0xCA40，清除复位标志记录 请注意由于复位记录工作于低速时钟域，清除执行完成需要一定时间，不应清除后立即读记录状态

5.2.17 SYS_CLK_DIV0 外设时钟分频寄存器 0

地址：0x4000_0090

复位值：0x0

表 5-20 外设时钟分频寄存器 0 SYS_CLK_DIV0

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

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

5.2.18 SYS_CLK_DIV2 外设时钟分频寄存器 2

地址：0x4000_0098

复位值：0x0

表 5-21 外设时钟分频寄存器 2 SYS_CLK_DIV2

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

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

5.2.19 SYS_CLK_FEN 外设时钟门控寄存器

地址：0x4000_009C

复位值：0x0

表 5-22 外设时钟门控寄存器 SYS_CLK_FEN

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

	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:9]		未使用
[8]	SPI_CKL_EN	SPI 时钟门控，1:使能时钟；0:禁用时钟
[7]	CAN_CLK_EN	CAN 时钟门控，1:使能时钟；0:禁用时钟
[6]	DSP_CLK_EN	DSP 时钟门控，1:使能时钟；0:禁用时钟
[5]	UART1_CLK_EN	UART1 时钟门控，1:使能时钟；0:禁用时钟
[4]	UART0_CLK_EN	UART0 时钟门控，1:使能时钟；0:禁用时钟
[3]	MCPWM_CLK_EN	MCPWM 时钟门控，1:使能时钟；0:禁用时钟
[2]	UTIMER_CLK_EN	UTIMER 时钟门控，1:使能时钟；0:禁用时钟
[1]	HALL_CLK_EN	HALL 时钟门控，1:使能时钟；0:禁用时钟
[0]	I2C_CLK_EN	I2C 时钟门控，1:使能时钟；0:禁用时钟

5.2.20 SYS_CLK_SLP 休眠寄存器

地址：0x4000_00A0

复位值：0x0

表 5-23 休眠寄存器 SYS_CLK_SLP

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	写入密码 0xDEAD，系统关闭高速时钟，进入休眠状态

5.2.21 SYS_TRIM 校正模式寄存器

地址：0x4000_00A8

复位值：0x0

表 5-24 校正模式寄存器 SYS_TRIM

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

位置	位名称	说明
[31:1]		未使用
[0]	TRIM	芯片复位后，进入 TRIM 模式 TRIM 结束后，通过软复位退出 TRIM 模式

5.2.22 SYS_SFT_RST 软复位寄存器

地址：0x4000_00AC

复位值：0x0

表 5-25 软复位寄存器 SYS_SFT_RST

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

位置	位名称	说明
[31:9]		未使用
[8]	SPI_SFT_RST	SPI 软复位，写 1 复位，再写 0 释放
[7]	CAN_SFT_RST	CAN 软复位，写 1 复位，再写 0 释放
[6]	DSP_SFT_RST	DSP 软复位，写 1 复位，再写 0 释放
[5]	UART1_SFT_RST	UART1 软复位，写 1 复位，再写 0 释放
[4]	UART0_SFT_RST	UART0 软复位，写 1 复位，再写 0 释放
[3]	MCPWM_SFT_RST	MCPWM 软复位，写 1 复位，再写 0 释放
[2]	UTIMER_SFT_RST	UTIMER 软复位，写 1 复位，再写 0 释放
[1]	HALL_SFT_RST	HALL 软复位，写 1 复位，再写 0 释放
[0]	I2C_SFT_RST	I2C 软复位，写 1 复位，再写 0 释放

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。

5.2.23 SYS_WR_PROTECT 写保护寄存器

地址：0x4000_00B0

复位值：0x0

表 5-26 写保护寄存器 SYS_WR_PROTECT

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

PSW
WO
0

位置	位名称	说明
[31:16]		未使用
[15:0]	PSW	除 SYS_AFE_REG3、SYS_AFE_DAC、SYS_AFE_DAC_AMC、SYS_AFE_DAC_DC 外，其他系统寄存器受写保护，写入前需先写入密码解除写保护 写入 0x7A83，使能系统寄存器写操作 写入 0xCAFE，使能看门狗 WDT_CLR 寄存器写操作 写入其它值，禁止寄存器写操作

5.2.24 SYS_AFE_DAC_AMC DAC 增益校正寄存器

地址：0x4000_00B4

复位值：0x0

表 5-27 DAC 增益校正寄存器 SYS_AFE_DAC_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

5.2.25 SYS_AFE_DAC_DC DAC 直流偏置寄存器

地址：0x4000_00B8

复位值：0x0

表 5-28 DAC 直流偏置寄存器 SYS_AFE_DAC_DC

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

位置	位名称	说明

[31:8]		未使用
[7:0]	DAC_DC	DAC 直流偏置，8bit 有符号数，B[7]为符号位

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

DAC_cali = DAC_raw * 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 位 DAC 直流偏置，为 8bit 有符号整数。

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

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

需要注意的是，DAC 有三个输出档位，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 FLASH

6.1 概述

FLASH 存储体包含两个部分：NVR 和 MAIN。NVR 大小为 1KB，MAIN 为 32KB 或 64KB（不同型号）。

主闪存存储区（MAIN），包括应用程序和用户数据区

信息存储区（info 区/NVR），预留给用户使用，大小为 1KB，供客户存储私有信息

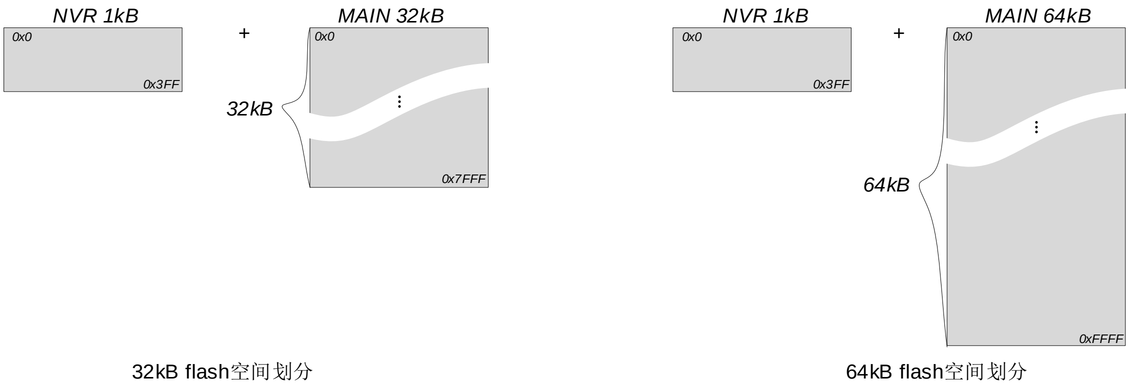


图 6-1 FLASH 存储体空间划分框图

- 可反复擦除写入不低于 2 万次
- 室温数据保持长达 100 年
- 单字节编程时间最长 7.5us，Sector 擦除时间最长 5ms。
- Sector 大小 512 字节，可按 Sector 擦除写入，支持运行时编程，擦写一个 Sector 的同时读取访问另一个 Sector
- Flash 数据防窃取（最后一个 word 须写入非 0xFFFFFFFF 的任意值）

6.2 功能特点

FLASH 控制器模块，主要实现对 FLASH 存储体的相关操作。包括：

- FLASH 读取数据的操作，包括对 NVR 部分的读取和对 MAIN 部分的读取。
- FLASH 写入数据的操作，包括对 NVR 部分的写入和对 MAIN 部分的写入。
- FLASH 擦除操作，包括 CHIP 擦除和 SECTOR 擦除。NVR 部分仅支持 SECTOR 擦除，MAIN 部分支持 CHIP 擦除和 SECTOR 擦除。
- FLASH 深度休眠的操作，以降低芯片的休眠功耗。

- FLASH 存储体内容的加密操作。
- FLASH 的读取加速操作，以提升芯片整体运行效率。
- FLASH 控制寄存器的访问。

6.2.1 功能描述

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

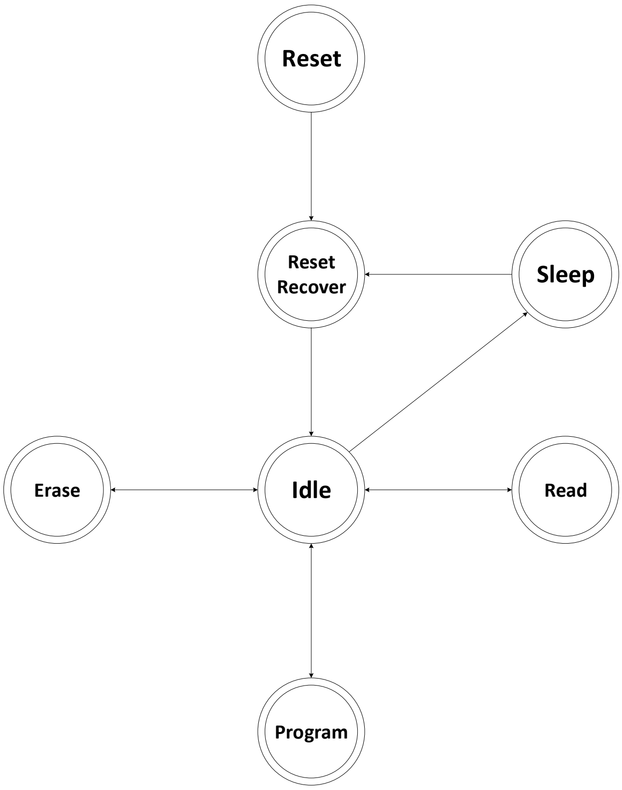


图 6-2 FLASH 控制状态转换图

6.2.1.1 复位操作

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

6.2.1.2 休眠操作

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

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

6.2.1.3 FLASH 读取操作

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

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

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

表 6-1 FLASH 访问空间分配表

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

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

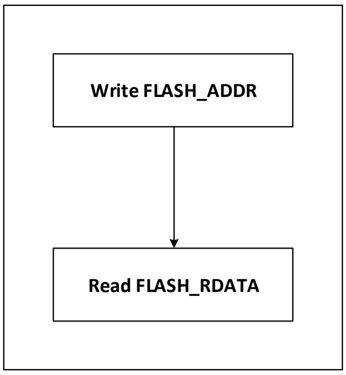


图 6-3 FLASH 间接读取操作流程图

6.2.1.4 FLASH 编程操作

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

- 控制寄存器 CFG，开启编程使能
- 地址寄存器 ADDR，写入编程地址
- 写数据寄存器 WDATA，写入编程数据

访问本控制器的寄存器，实现 FLASH 编程操作的执行流程如下：

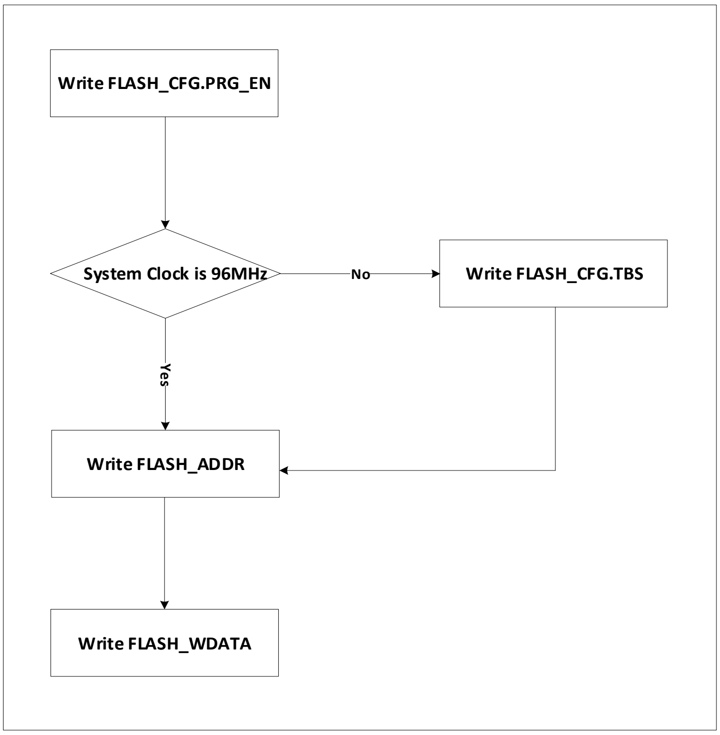


图 6-4 FLASH 模块编程操作流程图

系统工作频率的判断，需要参考 SYS_CLK_CFG 的配置。FLASH 写入/擦除操作的绝对时间是固定的，FLASH 控制器需要保存这些绝对时间对应的计数值。FLASH_CFG.TBS 默认值是 96MHz 时钟频率下的计数值；当芯片工作在其他频率下时，需要配置 FLASH_CFG.TBS 的值，以实现 48MHz/24MHz 和 12MHz 的计数值（其它频率暂不支持）。最终保证计数值的的值×时钟频率等于恒定的时间。不同频率下对应的的 FLASH_CFG.TBS 值请参考 6.3.2。切记，FLASH_CFG.TBS 仅能配置寄存器说明中提供的几组值，不能被写入其它值，否则可能导致 FLASH 编程/擦除失败。建议对 FLASH_CFG 的操作，执行先读回，然后按照或/与的方法操作。另外，在执行 FLASH 的编程/擦除操作时，CPU 将暂停工作直至 FLASH 的编程/擦除操作完毕。

图 6-4 仅展示了一次编程的流程。若执行连续编程时，可以在写入 FLASH_ADDR 寄存器前，配置 FLASH_CFG.ADR_INC，开启地址自动递增模式，后续只需要反复写 FLASH_WDATA 寄存器即可，FLASH_ADDR 每次写入一次数据会自动增加 0x4。连续读操作类似。连续编程的流程如下：

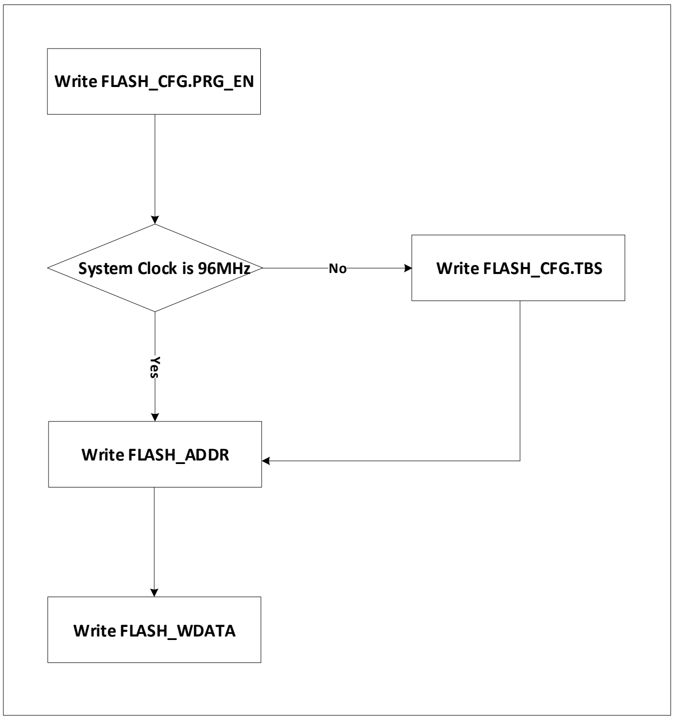


图 6-5 FLASH 模块编程操作流程图

6.2.1.5 FLASH 擦除操作

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

- FLASH 擦除操作使能
- 地址寄存器 ADDR，写入擦除地址
- 写擦除寄存器 ERASE，触发擦除操作

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

下表为 Block 和 Secotor 地址分配空间。

表 6-2 FLASH Sector 地址分配表

Name	Addresses	Size(Bytes)
Sector 0	0x0000 0000 - 0x0000 01FF	512
Sector1	0x0000 0200 - 0x0000 03FF	512
Sector2	0x0000 0400 - 0x0000 05FF	512
...	...	...
Sector127	0x0000 FE00 - 0x0000 FFFF	512

NVR 区域只能实现 Sector 擦除；MAIN 区域可以实现 Sector 擦除和 FULL 擦除。具体表格如下：

NVR (FLASH_CFG.REGION)	Sector Erase	FULL Erase
0	Main 区域	Main 区域

1	NVR 区域	Main 区域

FLASH 擦除操作流程如下所示。

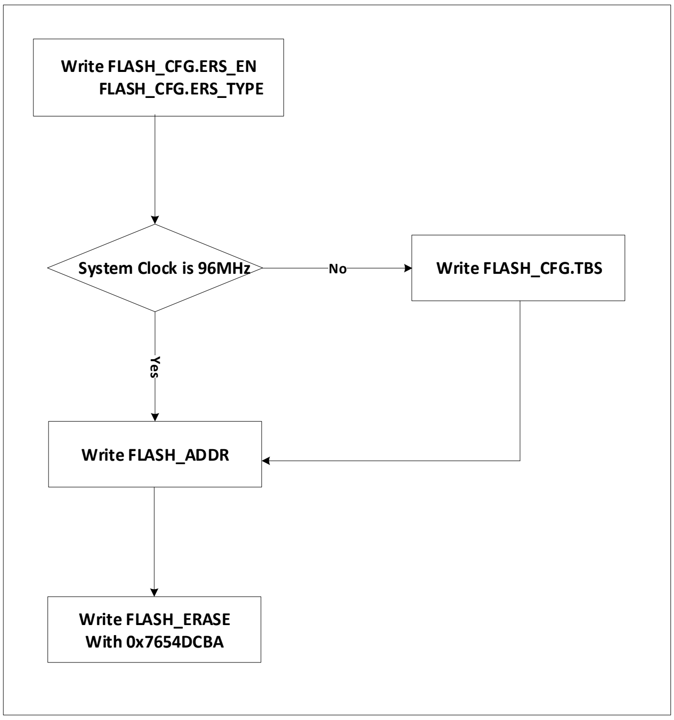


图 6-6 FLASH 模块擦除操作流程图

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

6.2.1.6 FLASH 预取操作

因 FLASH 存储体的速度限制，无法达到 96MHz 的速度。当对 FLASH 进行读取操作时，需要大于 1 个时钟周期才能完成数据的读出。为了加快数据的读出，FLASH 控制器增加了预取功能。当 FLASH 控制器执行完当前读取操作后，在不影响正常程序执行的前提下，顺序预取下一个 WORD 的数据。预取操作的开启和关闭，只需要设置 FLASH_CFG.PREF 即可。

6.2.1.7 FLASH 加密保护

若 FLASH 存储体内的数据处于加密状态，用户可执行解密操作，可对 FLASH 存储体内的数据进行解密。相反，若 FLASH 存储体内的数据处于解密状态，用户可执行加密操作，对 FLASH 存储体内的数据进行加密。默认情况下，FLASH 存储体内的数据处于加密状态。芯片上电复位后，硬件自动执行一次加密状态更新操作，是加密状态的话仍然加密状态，是解密的话状态变成解密状态。

FLASH 存储体共有 32kB/64kB 两个规格。无论哪种规格，相应规格中的最后一个 WORD 设计为加密字。当这个 WORD 内容为全 1 时，表明此时 FLASH 处于解密状态；当这个 WORD 的内容被写为非全 1 时，表明此时 FLASH 处于加密状态。若需要加密，执行最后一个 WORD 的编程，写入非全 1 的值，读取 FLASH_PROTECT 寄存器，即触发一次加密状态更新，完成加密（读取 FLASH_PROTECT 返回值无意义）。

对应的解密流程，分成两种情况。若最后一个 WORD 没有执行过编程写入非全 1 的操作，读取 FLASH_PROTECT 寄存器，即完成解密更新（无需考虑此时返回值）。若已经执行过编程写入非全 1

的操作，那么只能执行擦除操作才能解除。先对 FLASH 执行擦除操作，将最后一个 WORD 恢复为全 1 值，然后读取 FLASH_PROTECT 寄存器，即触发一次加密状态更新，完成解密（读取 FLASH_PROTECT 返回值无意义）。

6.2.1.8 FLASH 在线升级(IAP)

IAP 模式，实现中断向量表的重映射。在 LKS32MC08x 系列芯片中，包含了系统寄存器 VTOR，其地址为 0xE000_ED08。用于重新映射中断向量表入口地址。

表 6-3 IAP VTOR 寄存器描述

名称	复位值	偏移	位置	权限	说明
VTOR	0x0		[31:7]	RW	执行写入操作，写入中断向量表入口地址
			[6:0]	--	默认写 0

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

在 LKS32MC08x 系列芯片中，因为有 VTOR 寄存器。用户可根据自己需求，更新整个 FLASH 的内容。在线升级过程中可以使用中断，也可以关闭中断。

6.2.1.8.1 开启中断的在线升级

推荐软件配置流程：

关闭 CPU 的中断控制器，暂时不接收新的中断响应；

在新的中断入口地址处，放置中断处理函数代码；

将新的中断入口地址写入 VTOR 寄存器；

开启 CPU 的中断控制器，使能中断；

用户跳转至在线升级函数，开始在线升级功能；

完成升级，关闭 CPU 的中断控制器，配置 VTOR 为默认值 0；

执行 CPU 软复位，系统 PC 重新从 0 地址开始执行升级后的程序；

6.2.1.8.2 关闭中断的在线升级

关闭 CPU 的中断控制器，暂时不接收新的中断响应；

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

执行 CPU 软复位，系统 PC 重新从 0 地址开始执行升级后的程序；

6.2.1.8.3 在线升级函数的位置

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

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

6.3 寄存器

6.3.1 地址分配

FLASH 控制器模块寄存器的基地址是 0x4000_0400，寄存器列表如下：

表 6-4 FLASH 控制器模块寄存器列表

名称	偏移	说明
FLASH_CFG	0x00	FLASH 配置寄存器
FLASH_ADDR	0x04	地址寄存器
FLASH_WDATA	0x08	写数据寄存器
FLASH_RDATA	0x0C	读数据寄存器
FLASH_ERASE	0x10	擦除使能寄存器
FLASH_PROTECT	0x14	FLASH 保护状态寄存器
FLASH_READY	0x18	FLASH 闲忙状态寄存器

6.3.2 FLASH_CFG 配置寄存器（推荐先读回，按或/与方式修改）

地址：0x4000_0400

复位值：0x00000060

表 6-5 配置寄存器 FLASH_CFG

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

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

位置	位名称	说明
[31]	ERS_EN	FLASH 擦除使能。默认为 0。 0：关闭擦除

		1：开启擦除
[27]	PRG_EN	FLASH 编程使能。默认为 0。 0：关闭编程 1：开启编程
[23]	ADR_INC	FLASH 地址递增使能。默认为 0。 0：关闭递增使能 1：开启递增使能 当执行 FLASH 连续读写访问时，可以开启此功能减少对地址的操作。
[19]	PREF	FLASH 预取加速使能。默认为 0。 0：关闭加速 1：开启加速
[15]	ERS_TYPE	FLASH 擦除类型选择。默认为 0。 0：Sector 1：FULL
[11]	REGION	访问 FLASH 区域选择。默认为 0。 0：MAIN 1：NVR
[6:0]	TBS	编程/擦除时间基数寄存器,默认值为 0x60。只能配成如下几个值 0x60：96Mhz 系统频率下，FLASH 编程/擦除时间基数配置值。 0x2F：48Mhz 系统频率下，FLASH 编程/擦除时间基数配置值。 0x17：24Mhz 系统频率下，FLASH 编程/擦除时间基数配置值。 0x0B：12Mhz 系统频率下，FLASH 编程/擦除时间基数配置值。

6.3.3 FLASH_ADDR 地址寄存器

地址：0x4000_0404

复位值：0x0

表 6-6 地址寄存器 FLASH_ADDR

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

6.3.4 FLASH_WDATA 写数据寄存器

地址：0x4000_0408

复位值：0x0

表 6-7 写数据寄存器 FLASH_WDATA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WDATA															
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	执行写入操作，写入 FLASH 的值

6.3.5 FLASH_RDATA 读数据寄存器

地址：0x4000_040C

复位值：0x0

表 6-8 读数据寄存器 FLASH_RDATA

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

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

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

6.3.6 FLASH_ERASE 擦除控制寄存器

地址：0x4000_0410

复位值：0x0

表 6-9 擦除控制寄存器 FLASH_ERASE

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

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

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

6.3.7 FLASH_PROTECT 加密状态寄存器

地址：0x4000_0414

复位值：0x0

表 6-10 加密状态寄存器 FLASH_PROTECT

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

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

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

6.3.8 FLASH_READY 工作状态寄存器

地址：0x4000_0418

复位值：0x0

表 6-11 工作状态寄存器 FLASH_READY

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

	READY RO 0

位置	位名称	说明
[31:1]		未使用
[0]	READY	1:FLASH 处于闲状态；0:FLASH 处于忙状态

6.3.9 NVR 校正值地址信息

LKS32MC08x 产品存储芯片校准参数：

校准参数，每颗产品均独立校准，每颗产品不支持校准参数混用；

校准参数，在出厂前进行写入，出厂后不支持编程和擦除，仅支持读取；

校准参数，通过 LKS 公司提供的库函数进行读取访问；

校准参数，推荐关闭系统中断后，执行访问操作；

lks32mc08x_nvr.o 文件包含了校准参数的读取函数：

读取函数：uint32_t Read_Trim(uint32_t adr);

表 6-12 校准参数列表

地址	内容
0x0310	ADC0_DC0 校准值
0x0314	ADC0_DC1 校准值
0x0318	ADC0_AMC0 校准值
0x031C	ADC0_AMC1 校准值
0x0320	ADC1_DC0 校准值
0x0324	ADC1_DC1 校准值
0x0328	ADC1_AMC0 校准值
0x032C	ADC1_AMC1 校准值
0x0330	DAC 选择 3.00V 档位，SYS_AFE_DAC_AMC 校准值（扩大 512 倍结果）
0x0334	DAC 选择 1.20V 档位，SYS_AFE_DAC_AMC 校准值（扩大 512 倍结果）
0x0338	DAC 选择 4.85V 档位，SYS_AFE_DAC_AMC 校准值（扩大 512 倍结果）
0x0340	DAC 选择 3.00V 档位，SYS_AFE_DAC_DC 校准值
0x0344	DAC 选择 1.20V 档位，SYS_AFE_DAC_DC 校准值
0x0348	DAC 选择 4.85V 档位，SYS_AFE_DAC_DC 校准值
0x0350	OPA0 ，200K 欧姆 VS 10.4K 欧姆，GAIN 校准值（扩大 1000 倍结果）,R0 为 0 欧姆
0x0354	OPA0 ，190K 欧姆 VS 20.4K 欧姆，GAIN 校准值（扩大 1000 倍结果）,R0 为 0 欧姆
0x0358	OPA0 ，180K 欧姆 VS 30.4K 欧姆，GAIN 校准值（扩大 1000 倍结果）,R0 为 0 欧姆

0x035C	OPA0 ，170K 欧姆 VS 40.4K 欧姆，GAIN 校准值（扩大 1000 倍结果）,R0 为 0 欧姆
0x0360	OPA1 ，200K 欧姆 VS 10.4K 欧姆，GAIN 校准值（扩大 1000 倍结果）,R0 为 0 欧姆
0x0364	OPA1 ，190K 欧姆 VS 20.4K 欧姆，GAIN 校准值（扩大 1000 倍结果）,R0 为 0 欧姆
0x0368	OPA1 ，180K 欧姆 VS 30.4K 欧姆，GAIN 校准值（扩大 1000 倍结果）,R0 为 0 欧姆
0x036C	OPA1 ，170K 欧姆 VS 40.4K 欧姆，GAIN 校准值（扩大 1000 倍结果）,R0 为 0 欧姆
0x0370	OPA2 ，200K 欧姆 VS 10.4K 欧姆，GAIN 校准值（扩大 1000 倍结果）,R0 为 0 欧姆
0x0374	OPA2 ，190K 欧姆 VS 20.4K 欧姆，GAIN 校准值（扩大 1000 倍结果）,R0 为 0 欧姆
0x0378	OPA2 ，180K 欧姆 VS 30.4K 欧姆，GAIN 校准值（扩大 1000 倍结果）,R0 为 0 欧姆
0x037C	OPA2 ，170K 欧姆 VS 40.4K 欧姆，GAIN 校准值（扩大 1000 倍结果）,R0 为 0 欧姆
0x0380	OPA3 ，200K 欧姆 VS 10.4K 欧姆，GAIN 校准值（扩大 1000 倍结果）,R0 为 0 欧姆
0x0384	OPA3 ，190K 欧姆 VS 20.4K 欧姆，GAIN 校准值（扩大 1000 倍结果）,R0 为 0 欧姆
0x0388	OPA3 ，180K 欧姆 VS 30.4K 欧姆，GAIN 校准值（扩大 1000 倍结果）,R0 为 0 欧姆
0x038C	OPA3 ，170K 欧姆 VS 40.4K 欧姆，GAIN 校准值（扩大 1000 倍结果）,R0 为 0 欧姆
0x0398	温度传感器，斜率校准值
0x039C	温度传感器，斜率校准值
0x03B0	高 16-bit 存放 OPA1 共模电压，低 16-bit 存放 OPA0 共模电压（注意，均是放大 10000 倍存入）
0x03B4	高 16-bit 存放 OPA3 共模电压，低 16-bit 存放 OPA2 共模电压（注意，均是放大 10000 倍存入）
0x02C0	OPA0，200K 欧姆:10.4K 欧姆。高 16 位为 R2 实际电阻值，低 16 位为 R1 实际电阻值（扩大 100 倍结果）
0x02C4	OPA0，190K 欧姆:20.4K 欧姆。高 16 位为 R2 实际电阻值，低 16 位为 R1 实际电阻值（扩大 100 倍结果）
0x02C8	OPA0，180K 欧姆:30.4K 欧姆。高 16 位为 R2 实际电阻值，低 16 位为 R1 实际电阻值（扩大 100 倍结果）
0x02CC	OPA0，170K 欧姆:40.4K 欧姆。高 16 位为 R2 实际电阻值，低 16 位为 R1 实际电阻值（扩大 100 倍结果）
0x02D0	OPA1，200K 欧姆:10.4K 欧姆。高 16 位为 R2 实际电阻值，低 16 位为 R1 实际电阻值（扩大 100 倍结果）
0x02D4	OPA1，190K 欧姆:20.4K 欧姆。高 16 位为 R2 实际电阻值，低 16 位为 R1 实际电阻值（扩大 100 倍结果）
0x02D8	OPA1，180K 欧姆:30.4K 欧姆。高 16 位为 R2 实际电阻值，低 16 位为 R1 实际电阻值（扩大 100 倍结果）
0x02DC	OPA1，170K 欧姆:40.4K 欧姆。高 16 位为 R2 实际电阻值，低 16 位为 R1 实际电阻值（扩大 100 倍结果）
0x02E0	OPA2，200K 欧姆:10.4K 欧姆。高 16 位为 R2 实际电阻值，低 16 位为 R1 实际电阻值（扩大 100 倍结果）
0x02E4	OPA2，190K 欧姆:20.4K 欧姆。高 16 位为 R2 实际电阻值，低 16 位为 R1 实际电阻值（扩大 100 倍结果）
0x02E8	OPA2，180K 欧姆:30.4K 欧姆。高 16 位为 R2 实际电阻值，低 16 位为 R1 实际电阻值（扩大 100 倍结果）
0x02EC	OPA2，170K 欧姆:40.4K 欧姆。高 16 位为 R2 实际电阻值，低 16 位为 R1 实际电阻值（扩大 100 倍结果）

0x02F0	OPA3，200K 欧姆:10.4K 欧姆。高 16 位为 R2 实际电阻值，低 16 位为 R1 实际电阻值（扩大 100 倍结果）
0x02F4	OPA3，190K 欧姆:20.4K 欧姆。高 16 位为 R2 实际电阻值，低 16 位为 R1 实际电阻值（扩大 100 倍结果）
0x02F8	OPA3，180K 欧姆:30.4K 欧姆。高 16 位为 R2 实际电阻值，低 16 位为 R1 实际电阻值（扩大 100 倍结果）
0x02FC	OPA3，170K 欧姆:40.4K 欧姆。高 16 位为 R2 实际电阻值，低 16 位为 R1 实际电阻值（扩大 100 倍结果）

7 DMA

7.1 概述

增加 DMA 后，总线上主设备由 CPU 增加为 CPU 和 DMA，总线架构需要由 AHB lite 演进为 multi-layer AHB lite 架构。如图 7-1 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 AHB bridge 0 上。**包括 ADC/DAC/SPI/I2C/MCPWM/UART/SRAM 在内的设备被 CPU 和 DMA 共享访问，挂载于 AHB bridge 1 上。而 Timer/DSP/GPIO/Hall/Flash 等设备，DMA 无法访问。**

从设备端的仲裁模块本质上相当于一个二选一的多路选择器。只有端口上有仲裁器的从设备才能被 DMA 访问到，否则无法被 DMA 访问，而所有外设设备均可被 CPU 访问。

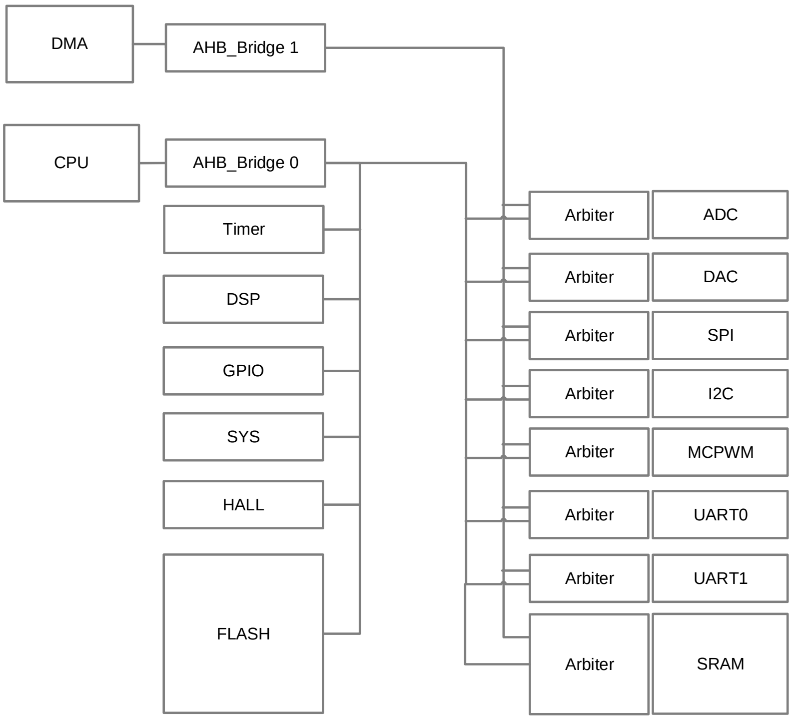


图 7-1 multi-layer AHB lite 总线架构

出于功耗控制考虑，DMA 模块可以通过设置 DMA_CTRL.EN 位为 0 来被禁用（要求关闭 DMA 使能前先关闭 4 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（相当于一个从设备，被 CPU 配置）和数据搬运模块（对于总线为主设备，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.PBTW 和 DMA_CCRx.MBTW 来选择外设和内存访问的位宽，外设访问的位宽与内存访问的位宽可以不同。

DMA 每完成一次传输，地址根据 DMA_CCRx.PINC 和 DMA_CCRx.MINC 决定是否自动递增。所有的外设寄存器地址都是 word 对齐的，所以外设的地址递增永远是 0/4 (根据 DMA_CCRx.PINC=0/1 设置)。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，DMA_CCRx.PINC=0；如要访问 ADC 数据寄存器，则地址通常需要自动加 4，需要设置

DMA_CCRx.PINC=1。而对于内存，如果设置了 DMA_CCRx.MINC=1，每次地址递增的值，根据内存数据位宽(DMA_CCRx.MBTW)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。

需要注意的是，我们将 DMA 的传输分为多个轮次，每轮传输内可以有多次传输。DMA_CCRx.PINC 用于控制外设地址每次传输是否递增。外设地址每轮传输一定重复上一轮的地址，即轮内可以递增，轮间不递增；DMA_CCRx.MINC 用于控制内存地址轮间是否递增，每次传输一定递增。

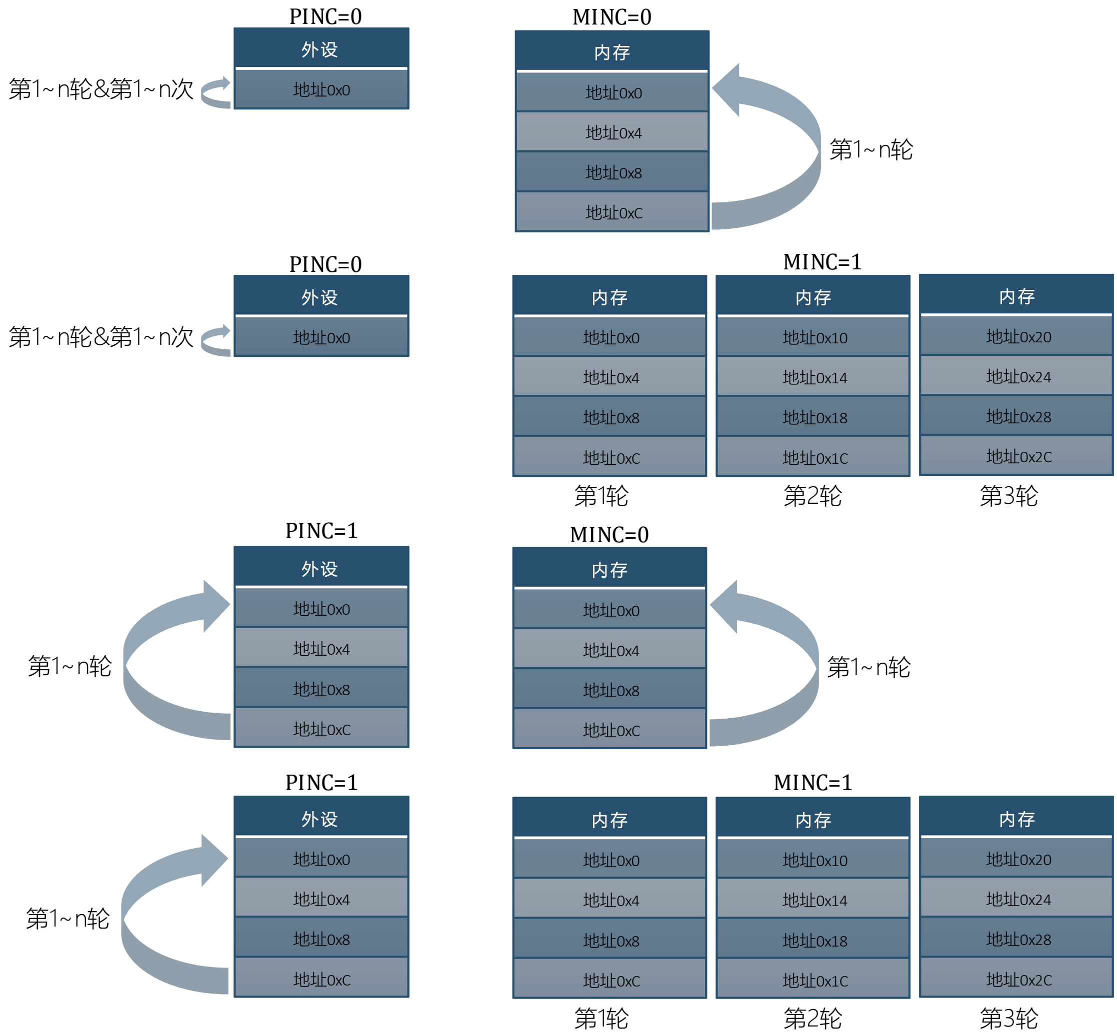


图 7-2 DMA 地址递增控制

DMA 有循环模式和单次模式两种模式，由 DMA_CCRx.CIRC 控制。循环模式下 DMA 完成一定大小的数据块搬运后重新开始下一轮搬运，如果是搬运数据到内存，则覆盖之前搬运至内存的数据；如果是搬运至外设，则重复一轮数据发射。以将 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完成本次 DMA 操作，设置完成 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。

7.2 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_CCRx.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。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 7-1 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_CCRx.SW_TRIG，一旦通道被使能即开始进行 DMA 操作。**此版本 DMA 暂不支持内存至内存搬运!!!**
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，不要软件清零；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。
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。有两种方案可选： 方案 1：若 UARTx_IE.TX_BUF_EMPTY_RE 配置有效。UARTx 模块将预取第一个字节，准备发送；一旦数据进入发射队列，UARTx_IE.UARTx_BUFF 即为空，硬件自动会请求 DMA 搬移下一个字节直至数据搬移完毕。DMA 搬移完毕后，将产生 DMA 完成中断，但是 UARTx 很可能没有发送完毕最后一个字节，若立即操作 UARTx，将可能会产生异常。建议在 DMA 中断处理程序中，开启 UARTx_IE.TX_DONE_IE 中断，UARTx 将最后一个字节发送完毕，产生发送完成中断，在 UARTx 中断处理函数里面，再关闭 UARTx_IE.TX_DONE_IE。 方案 2：若 UARTx_IE.TX_DONE_RE 配置有效。建议 UART 初始化阶段不要清除 TX_DONE 标志。当前传输的数据长度为 Len，DMA 配置传输的字节数为 Len，开启 DMA 中断，DMA 传输完毕后，UART 也发送完毕，软复位 UARTx 模块并重新初始化 UARTx，可开启下一次 UARTx 的发送。
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。
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
Timer	Timer 使用过零/比较/捕获事件作为 DMA 请求，具体 DMA 操作根据配置寄存器设定，通常作为定时事件（如每隔 10ms 触发一次 DMA 操作）
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
CAN	

7.3 优先级

DMA 的优先级采用固定优先级，优先级如图 7-3 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 7-3 所示，优先级由上至下降低。4 个 DMA 通道中，优先级关系为：通道 0>通道 1>通道 2>通道 3 (>号表示优先级高于)。在 DMA 各通道内部，通常有 3 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。3 个硬件请求事件优先级相同，通常应用上面一个 DMA 通道配置一个硬件请求事件使能，多个硬件请求不应在一个通道内同时发生。

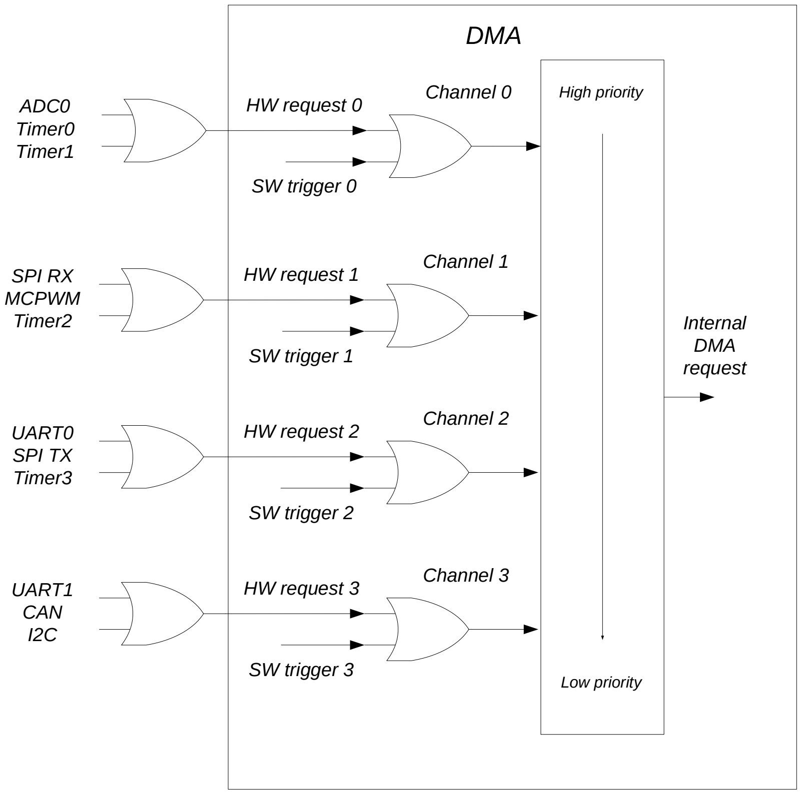


图 7-3 DMA 通道优先级

7.4 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，应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。

CPU 访问 RAM 的优先级始终高于 DMA。

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。

7.5 中断

DMA 的一个通道完成 DMA 操作后或出错则产生 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。

7.6 寄存器

7.6.1 地址分配

DMA 控制器模块寄存器的基地址是 0x4001_3000，寄存器列表如下：

表 7-2 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_CTMS0	0x04	DMA 通道 0 传输次数寄存器
DMA_CPAR0	0x08	DMA 通道 0 外设地址寄存器
DMA_CMAR0	0x0C	DMA 通道 0 内存地址寄存器
DMA_CCR1	0x10	DMA 通道 1 通道配置寄存器
DMA_CTMS1	0x14	DMA 通道 1 传输次数寄存器
DMA_CPAR1	0x18	DMA 通道 1 外设地址寄存器
DMA_CMAR1	0x1C	DMA 通道 1 内存地址寄存器
DMA_CCR2	0x20	DMA 通道 2 通道配置寄存器
DMA_CTMS2	0x24	DMA 通道 2 传输次数寄存器
DMA_CPAR2	0x28	DMA 通道 2 外设地址寄存器
DMA_CMAR2	0x2C	DMA 通道 2 内存地址寄存器
DMA_CCR3	0x30	DMA 通道 3 通道配置寄存器
DMA_CTMS3	0x34	DMA 通道 3 传输次数寄存器
DMA_CPAR3	0x38	DMA 通道 3 外设地址寄存器
DMA_CMAR3	0x3C	DMA 通道 3 内存地址寄存器
DMA_CTRL	0x40	DMA 控制寄存器
DMA_IF	0x44	DMA 中断标志寄存器

7.6.2 DMA_CTRL DMA 控制寄存器

地址：0x4001_3040

复位值：0x0

表 7-3 DMA 控制寄存器 DMA_CTRL

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

	RW	RW
	0	0

位置	位名称	说明
[31:2]		未使用
[1]	PRIORITY	0:CPU 优先级高；1:DMA 优先级高，此版本必须设置 CPU 优先级高
[0]	EN	DMA 使能

7.6.3 DMA_IF DMA 中断标志寄存器

地址：0x4001_3044

复位值：0x0

表 7-4 DMA 中断标志寄存器 DMA_IF

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

位置	位名称	说明
[31:8]		未使用
[7]	CH3_EIF	通道 3 错误中断标志
[6]	CH2_EIF	通道 2 错误中断标志
[5]	CH1_EIF	通道 1 错误中断标志
[4]	CH0_EIF	通道 0 错误中断标志
[3]	CH3_FIF	通道 3 完成中断标志
[2]	CH2_FIF	通道 2 完成中断标志
[1]	CH1_FIF	通道 1 完成中断标志
[0]	CH0_FIF	通道 0 完成中断标志

7.6.4 DMA 通道配置寄存器

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

地址分别是：0x4001_3000，0x4001_3010，0x4001_3020，0x4001_3030

复位值：0x0

表 7-5 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
SW_TRIG	REQ_EN			MBTW		PBTW		MINC	PINC	CIRC	DIR		TEIE	TCIE	EN
RW	RW			RW		RW		RW	RW	RW	RW		RW	RW	RW

0	0	0	0	0	0	0	0		0	0	0

位置	位名称	说明
[31:16]		未使用
[15]	SW_TRIG	软件触发，高有效*2
[14:12]	REQ_EN	通道 x 三个硬件 DMA 请求使能*1，高有效
[11:10]	MBTW	内存访问位宽，0:byte, 1:half-word, 2:word, 3:保留
[9:8]	PBTW	外设访问位宽，0:byte, 1:half-word, 2:word, 3:保留
[7]	MINC	内存地址第二轮是否在第一轮地址的基础上递增（轮内一定递增），高有效
[6]	PINC	指示外设地址每轮内是否递增（外设地址第二轮一定重复第一轮地址），高有效
[5]	CIRC	循环模式，高有效
[4]	DIR	传输方向，0:外设至内存，1:内存至外设
[3]		保留
[2]	TEIE	错误中断使能，高有效
[1]	TCIE	传输完成中断使能，高有效
[0]	EN	通道 x 使能，高有效，在通道操作全部完成后由 DMA 清除此位

*1 以通道 0 为例，DMA_CCR0.REQ_EN[2:0]分别为 Timer1、Timer0、ADC0 的 DMA 请求使能。且 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，即软件写入 DMA_CCRx.SW_TRIG 位即开始一次 DMA 传输。来自外设的硬件请求进入 DMA 后通过或逻辑形成一个请求信号，每一个 DMA 通道同时只应使能一个硬件 DMA 请求。*2 软件触发标志 DMA_CCRx.SW_TRIG 写入 1 后需要软件清 0。

表 7-6 DMA 通道请求信号

DMA 通道	设备请求信号编号	外设
Channel 0	0	ADC0
	1	Timer0
	2	Timer1
Channel 1	0	SPI_RX
	1	MCPWM
	2	Timer2
Channel 2	0	UART0
	1	SPI_TX
	2	Timer3
Channel 3	0	UART1
	1	CAN
	2	I2C

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

地址分别是：0x4001_3004，0x4001_3014，0x4001_3024，0x4001_3034

复位值：0x0

表 7-7 DMA 传输次数寄存器 DMA_CTMSx

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

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

位置	位名称	说明
[31:24]		未使用
[23:16]	ROUND	DMA 通道 x 采样轮数
[15:9]		未使用
[8:0]	TIMES	DMA 通道 x 每轮数据搬运次数，0～512。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。

以外设数据宽度为 16，内存数据宽度为 32，即 CTMS.TIMES=16，CTMS.ROUND=2 为例，DMA 每轮需要读取外设数据 16bit×16=32byte，写入内存数据 32bit×16=64byte。一共需要搬运两轮，即读取外设 64byte，输入内存 128byte；

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，CTMS.ROUND 不再起作用，相当于无限轮；其他情况需要相应设置 CTMS.ROUND，如 CTMS.ROUND=1，用于搬运一轮数据。

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。

通常在开启 DMA 之前需要将触发 DMA 启动的外设接收中断标志位清零，防止之前留下的中断标志位成为 DMA 触发源造成一次错误触发。以 UART 为例，通常 DMA 被配置为多轮搬运，每轮搬运一次，每次搬运一个字节。如果 UART 接收 FIFO 不空，且 UART_IF[1]未清零，则使能 DMA 通道后立即发生一次 DMA 搬运，但此时的 DMA 搬运的数据是 UART 之前遗留的数据，并非实际需要搬运的数据，且此时 DMA 内部的轮次增 1，可能会导致接收的 UART 数据帧发生错位。出现这种情况，可以通过重写 DMA_CTMS 来讲 DMA 内部的轮次计数重置。

而在使用 DMA 通过 UART 发送数据时，通常使用的是 UART TX_FIFO 空标志作为 DMA 请求，外设上电后发送 FIFO 空即有效，无须软件清除这一标志。否则无法向 DMA 产生第一次的发送请求。

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

地址分别是：0x4001_3008，0x4001_3018，0x4001_3028，0x4001_3038

复位值：0x0

表 7-8 DMA 外设地址寄存器 DMA_ CPARx

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

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

位置	位名称	说明
[31:17]		未使用
[16:0]	PERI_ADDR	DMA 通道 x 外设地址

当 DMA_CCRx.PBTW=2’b01 时，即配置为以 16bit 为单位搬运外设数据。CPARx.PERI_ADDR[0]值无效，外设地址会以 2 为单位递增。

当 DMA_CCRx.PBTW=2’b10 时，即配置为以 32bit 为单位搬运外设数据。CPARx.PERI_ADDR[1:0]值无效，外设地址会以 4 为单位递增。

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

由于只进行内存与外设间的数据搬运，因此 DMA_CPAR 只存储外设地址的低 17 位，高 15 位恒为 0x2000，高 20 位可能为 0x40000(对应 SYS 寄存器)或 0x4001*(对应外设寄存器)。

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

地址分别是：0x4001_300C，0x4001_301C，0x4001_302C，0x4001_303C

复位值：0x0

表 7-9 DMA 内存地址寄存器 DMA_CMARx

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

位置	位名称	说明
[31:13]		未使用

[12:0]	MEM_ADDR	DMA 通道 x 内存地址

当 DMA_CCRx.MBTW=2’b01 时，即配置为以 16bit 为单位搬运内存数据。CMARx.MEM_ADDR[0] 值无效，内存地址会以 2 为单位递增。

当 DMA_CCRx.MBTW=2’b10 时，即配置为以 32bit 为单位搬运内存数据。CMARx.MEM_ADDR[1:0] 值无效，内存地址会以 4 为单位递增。

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

由于只进行内存与外设间的数据搬运，因此 DMA_CMAR 只存储地址的低 13 位，对应 SRAM 8kB 地址空间。高 19 位恒为 0x10000。

8 GPIO

8.1 概述

LSK32MC08X 系列芯片共集成了 4 组 16bit 位宽 GPIO。P0.0/P0.1/P1.0/P1.1 4 个 GPIO 可以作为系统的唤醒源。P0.15 ~ P0.0 共 16 个 GPIO 可以用作外部中断源输入。

其中 P0.2 既可以当做外部复位脚，也可以当做 GPIO 使用。可以通过软件设置 SYS_RST_CFG.RST_IO 进行切换。上电后，P0.2 默认用做外部复位脚，因此应用上应该注意上电后 P0.2 的外部信号不能一直为低，否则芯片会一直处于复位状态。复位释放后，软件可以将 SYS_RST_CFG.RST_IO 置 1，将 P0.2 切换为 GPIO 功能，SYS_RST_CFG 受 SYS_WR_PROTECT 保护。

在某些低引脚数封装中，例如 SSOP24，P2.15 与 SWDIO 复用为同一个引脚。此种复用是通过封装直接 bonding 在一起实现的。芯片上电后，P2.15 默认状态下输入输出全部禁用，此时不影响 SWDIO 参与 SWD 通讯。**如果应用上需要将此引脚复用为 GPIO，即使用 P2.15，需要注意保留某种手段再次关闭 P2.15 的输出使能，以将引脚切换成 SWDIO，否则可能导致芯片 SWD 无法使用。**

8.1.1 功能框图

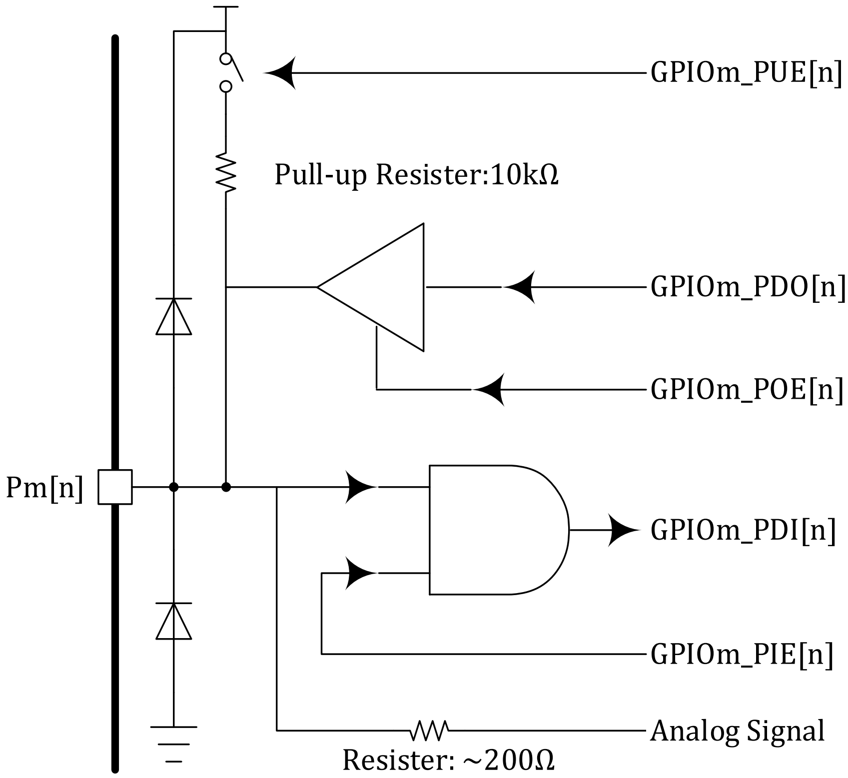
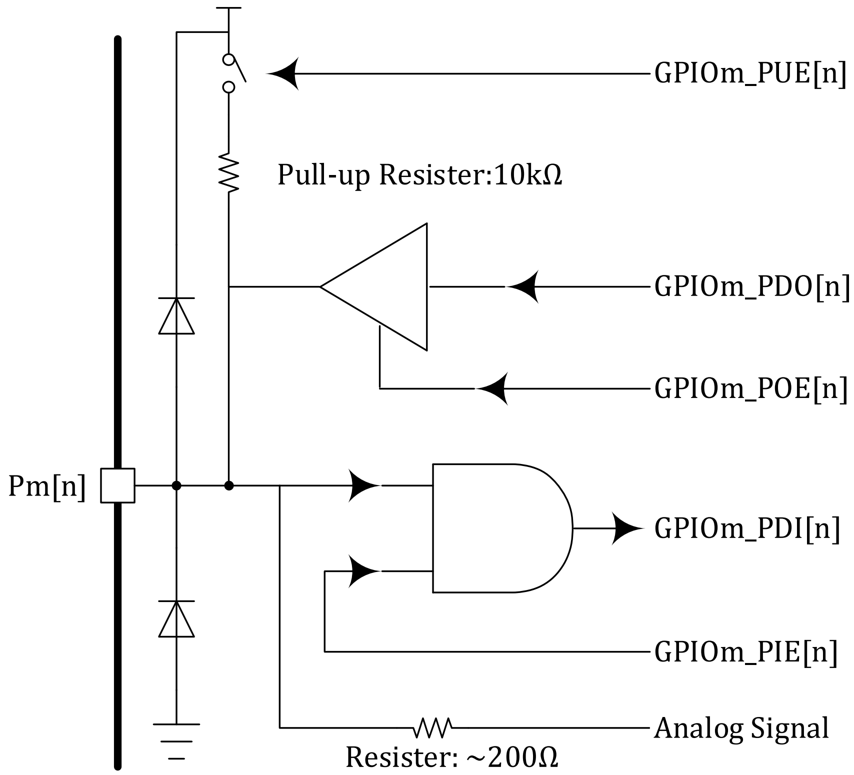


图 8-1 GPIO 功能框图



如图 8-1 所示，Pm[n]为芯片 PAD，m 可以是 0~3，表示 4 组 GPIO 中的任意一组，n 可以是 0~15，表示一组 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]的电平与 Pm[n]电平相同。芯片 PAD 可以配置上拉，P0[2]引脚因为复用为外部复位脚 RSTN 上拉电阻为 100kΩ，其余上拉电阻为 10kΩ，注意，并非所有 PAD 都配备上拉电阻，具体哪些 PAD 具有上拉电阻资源，请参考 8.3.1 上拉实现章节。没有上拉电阻的 PAD 也可以配置 GPIOm_PUE[n]寄存器，但无实际作用。

8.1.2 产品特点

- 4 组 16bit GPIO
- 支持开漏
- 部分 IO 支持上拉
- 支持配置锁定保护
- 支持外部中断
- 支持 GPIO 唤醒

8.2 寄存器

8.2.1 地址分配

GPIO 0 模块在芯片中的基地址是 0x40012000。

GPIO 1 模块在芯片中的基地址是 0x40012040。

GPIO 2 模块在芯片中的基地址是 0x40012080。

GPIO 3 模块在芯片中的基地址是 0x400120C0。

GPIO 0/1/2/3 的寄存器定义完全相同，仅基地址不同，寄存器列表如下：

表 8-1 GPIOx 寄存器列表

名称	偏移地址	说明
GPIOx_PIE	0x00	GPIO x 输入使能
GPIOx_POE	0x04	GPIO x 输出使能
GPIOx_PDI	0x08	GPIO x 输入数据
GPIOx_PDO	0x0C	GPIO x 输出数据
GPIOx_PUE	0x10	GPIO x 上拉使能
GPIOx_PODE	0x18	GPIO x 开漏使能
GPIOx_LCKR	0x1C	GPIO x 配置锁定
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

GPIO 中断/唤醒/配置锁定模块的基地址是 0x40012100，寄存器列表如下：

表 8-2 GPIO 中断/唤醒/配置锁定模块寄存器列表

名称	偏移地址	说明
EXTI_CR0	0x00	GPIO 0[7:0] 中断触发类型
EXTI_CR1	0x04	GPIO 0[15:8]中断触发类型
EXTI_IF	0x08	GPIO 中断标志
LCKR_PRT	0x0C	GPIO 保护锁定配置
WAKE_POL	0x10	GPIO 唤醒信号极性
WAKE_EN	0x14	GPIO 唤醒使能

8.2.2 GPIOx_PIE

地址分别是：0x4001_2000，0x4001_2040，0x4001_2080，0x4001_20C0

复位值：0x0

表 8-3 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

8.2.3 GPIOx_POE

地址分别是：0x4001_2004，0x4001_2044，0x4001_2084，0x4001_20C4

复位值：0x0

表 8-4 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

8.2.4 GPIOx_PDI

地址分别是：0x4001_2008，0x4001_2048，0x4001_2088，0x4001_20C8

复位值：0x0

表 8-5 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。

8.2.5 GPIOx_PDO

地址分别是：0x4001_200C，0x4001_204C，0x4001_208C，0x4001_20CC

复位值：0x0

表 8-6 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

8.2.6 GPIOx_PUE

地址分别是：0x4001_2010，0x4001_2050，0x4001-2090，0x4001_20D0

复位值：0x0

表 8-7 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

8.2.7 GPIOx_PODE

地址分别是：0x4001_2018，0x4001_2058，0x4001_2098，0x4001_20D8

复位值：0x0

表 8-8 GPIOx 开漏使能寄存器 GPIOx_PODE

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

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

8.2.8 GPIOx_LCKR

地址分别是：0x4001_201C，0x4001_205C，0x4001_209C，0x4001_20DC_

复位值：0x0

表 8-9 GPIOx 配置锁定寄存器 GPIOx_LCKR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLCKR15	PLCKR14	PLCKR13	PLCKR12	PLCKR11	PLCKR10	PLCKR9	PLCKR8	PLCKR7	PLCKR6	PLCKR5	PLCKR4	PLCKR3	PLCKR2	PLCKR1	PLCKR0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:16]		未使用
[15]	PLCKR15	GPIO x[15] / Px[15] 配置锁定
[14]	PLCKR14	GPIO x[14] / Px[14] 配置锁定
[13]	PLCKR13	GPIO x[13] / Px[13] 配置锁定
[12]	PLCKR12	GPIO x[12] / Px[12] 配置锁定
[11]	PLCKR11	GPIO x[11] / Px[11] 配置锁定
[10]	PLCKR10	GPIO x[10] / Px[10] 配置锁定
[9]	PLCKR9	GPIO x[9] / Px[9] 配置锁定
[8]	PLCKR8	GPIO x[8] / Px[8] 配置锁定
[7]	PLCKR7	GPIO x[7] / Px[7] 配置锁定
[6]	PLCKR6	GPIO x[6] / Px[6] 配置锁定
[5]	PLCKR5	GPIO x[5] / Px[5] 配置锁定
[4]	PLCKR4	GPIO x[4] / Px[4] 配置锁定
[3]	PLCKR3	GPIO x[3] / Px[3] 配置锁定
[2]	PLCKR2	GPIO x[2] / Px[2] 配置锁定
[1]	PLCKR1	GPIO x[1] / Px[1] 配置锁定
[0]	PLCKR0	GPIO x[0] / Px[0] 配置锁定

配置保护，高有效；有效时 GPIO 输入/输出/上下拉/开漏/功能选择不能被修改；需要注意，只有在 LCKR_PRT 写保护打开时才能改写 LCKR。

8.2.9 GPIOx_F3210

地址分别是：0x4001_2020，0x4001_2060，0x4001_20A0，0x4001_20E0

复位值：0x0

表 8-10 GPIOx 功能选择寄存器 GPIOx_F3210

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

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

GPIO 引脚功能复用如表 8-11 所示。

表 8-11 GPIO 功能复用

GPIOx_Fxxxx 配置值	第二功能代号	功能
0x0	AF0	模拟功能
0x1	AF1	SYS_AF，比较器及时钟等数字信号输出功能
0x2	AF2	HALL
0x3	AF3	MCPWM
0x4	AF4	UART
0x5	AF5	SPI
0x6	AF6	IIC
0x7	AF7	Timer0/Time1
0x8	AF8	Timer2/ Timer3/QEP0/QEP1
0x9	AF9	ADC trigger debug
0xA	AF10	CAN
0xB	AF11	SIF

8.2.10 GPIOx_F7654

地址分别是：0x4001_2024，0x4001_2064，0x4001_20A4，0x4001_20E4

复位值：0x0

表 8-12 GPIOx 功能选择寄存器 GPIOx_F7654

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F7				F6				F5				F4			
RW				RW				RW				RW			
0				0				0				0			

位置	位名称	说明
[31:16]		未使用
[15:12]	F7	GPIO x[7] / Px[7] 功能选择
[11:8]	F6	GPIO x[6] / Px[6] 功能选择
[7:4]	F5	GPIO x[5] / Px[5] 功能选择
[3:0]	F4	GPIO x[4] / Px[4] 功能选择

8.2.11 GPIOx_FBA98

地址分别是：0x4001_2028，0x4001_2068，0x4001_20A8，0x4001_20E8

复位值：0x0

表 8-13 GPIOx 功能选择寄存器 GPIOx_FBA98

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F11				F10				F9				F8			

RW	RW	RW	RW
0	0	0	0

位置	位名称	说明
[31:16]		未使用
[15:12]	F11	GPIO x[11] / Px[11] 功能选择
[11:8]	F10	GPIO x[10] / Px[10] 功能选择
[7:4]	F9	GPIO x[9] / Px[9] 功能选择
[3:0]	F8	GPIO x[8] / Px[8] 功能选择

8.2.12 GPIOx_FFEDC

地址分别是：0x4001_202C，0x4001_206C，0x4001_20AC，0x4001_20EC

复位值：0x0

表 8-14 GPIOx 功能选择寄存器 GPIOx_FFEDC

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F15				F14				F13				F12			
RW				RW				RW				RW			
0				0				0				0			

位置	位名称	说明
[31:16]		未使用
[15:12]	F15	GPIO x[15] / Px[15] 功能选择
[11:8]	F14	GPIO x[14] / Px[14] 功能选择
[7:4]	F13	GPIO x[13] / Px[13] 功能选择
[3:0]	F12	GPIO x[12] / Px[12] 功能选择

GPIO 的功能复用详细列表请见对应产品 DATASHEET 管脚分布章节。

8.2.13 外部中断、唤醒、锁定保护

P0.0/P0.1/P1.0/P1.1 4 个 GPIO 可以作为系统的唤醒源。P0.15 ~ P0.0 共 16 个 GPIO 可以用作外部中断源输入。唤醒功能和外部中断功能使用的是 IO 的 GPIO 功能，GPIO 第二功能可以配置为 0。

8.2.13.1 EXTI_CR0

地址：0x4001_2100

复位值：0x0

表 8-15 外部中断配置寄存器 EXTI_CR0

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

位置	位名称	说明
[31:16]		未使用
[15:14]	T7	GPIO 0[7]/ P0[7]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
[13:12]	T6	GPIO 0[6]/ P0[6]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
[11:10]	T5	GPIO 0[5]/ P0[5]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
[9:8]	T4	GPIO 0[4]/ P0[4]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
[7:6]	T3	GPIO 0[3]/ P0[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 0[2]/ P0[2]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
[3:2]	T1	GPIO 0[1]/ P0[1]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

[1:0]	T0	GPIO 0[0]/ P0[0]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

8.2.13.2 EXTI_CR1

地址：0x4001_2104

复位值：0x0

表 8-16 外部中断配置寄存器 EXTI_CR1

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

位置	位名称	说明
[31:16]		未使用
[15:14]	T15	GPIO 0[15]/P0[15]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
[13:12]	T14	GPIO 0[14]/P0[14]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
[11:10]	T13	GPIO 0[13]/P0[13]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
[9:8]	T12	GPIO 0[12]/P0[12]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
[7:6]	T11	GPIO 0[11]/P0[11]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]	T10	GPIO 0[10]/P0[10]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
[3:2]	T9	GPIO 0[9]/P0[9]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
[1:0]	T8	GPIO 0[8]/P0[8]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

8.2.13.3 EXTI_IF

地址：0x4001_2108

复位值：0x0

表 8-17 外部中断标志寄存器 EXTI_IF

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

位置	位名称	说明
[31:16]		未使用
[15]	IF15	GPIO 0[15] / P0[15] 外部中断标志 中断标志高有效，写 1 清零
[14]	IF14	GPIO 0[14] / P0[14] 外部中断标志 中断标志高有效，写 1 清零
[13]	IF13	GPIO 0[13] / P0[13] 外部中断标志 中断标志高有效，写 1 清零
[12]	IF12	GPIO 0[12] / P0[12] 外部中断标志 中断标志高有效，写 1 清零
[11]	IF11	GPIO 0[11] / P0[11] 外部中断标志

		中断标志高有效，写 1 清零
[10]	IF10	GPIO 0[10] / P0[10] 外部中断标志 中断标志高有效，写 1 清零
[9]	IF9	GPIO 0[9] / P0[9] 外部中断标志 中断标志高有效，写 1 清零
[8]	IF8	GPIO 0[8] / P0[8] 外部中断标志 中断标志高有效，写 1 清零
[7]	IF7	GPIO 0[7] / P0[7] 外部中断标志 中断标志高有效，写 1 清零
[6]	IF6	GPIO 0[6] / P0[6] 外部中断标志 中断标志高有效，写 1 清零
[5]	IF5	GPIO 0[5] / P0[5] 外部中断标志 中断标志高有效，写 1 清零
[4]	IF4	GPIO 0[4] / P0[4] 外部中断标志 中断标志高有效，写 1 清零
[3]	IF3	GPIO 0[3] / P0[3] 外部中断标志 中断标志高有效，写 1 清零
[2]	IF2	GPIO 0[2] / P0[2] 外部中断标志 中断标志高有效，写 1 清零
[1]	IF1	GPIO 0[1] / P0[1] 外部中断标志 中断标志高有效，写 1 清零
[0]	IF0	GPIO 0[0] / P0[0] 外部中断标志 中断标志高有效，写 1 清零

8.2.13.4 LCKR_PRT

地址：0x4001_210C

复位值：0x0

表 8-18 锁定保护寄存器 LCKR_PRT

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

位置	位名称	说明
[31:16]		未使用
[15:0]	PRT	配置锁定写保护；写入 0x5AC4 关闭写保护，然后才能修改 GPIO_LCKR；写入任意其他数据开启写保护；B[0]状态指示当前写保护与否，高电平表示处于写保护，低电平表示处于非写保护。

8.2.13.5 WAKE_POL

地址：0x4001_2110

复位值：0x0

表 8-19 外部唤醒源极性配置寄存器 WAKE_POL

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

位置	位名称	说明
[31:4]		未使用
[3]	GPIO1_1_POL	GPIO 1[1] / P1[1]外部唤醒触发电平选择 1：高电平；0：低电平
[2]	GPIO1_0_POL	GPIO 1[0] / P1[0]外部唤醒触发电平选择 1：高电平；0：低电平
[1]	GPIO0_1_POL	GPIO 0[1] / P0[1]外部唤醒触发电平选择 1：高电平；0：低电平
[0]	GPIO0_0_POL	GPIO 0[0] / P0[0]外部唤醒触发电平选择 1：高电平；0：低电平

8.2.13.6 WAKE_EN

地址：0x4001_2114

复位值：0x0

表 8-20 外部唤醒源使能寄存器 WAKE_EN

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

位置	位名称	说明
[31:4]		未使用
[3]	GPIO1_1_WKEN	GPIO 1[1] / P1[1] 外部唤醒使能 1：使能；0：禁用。
[2]	GPIO1_0_WKEN	GPIO 1[0] / P1[0] 外部唤醒使能 1：使能；0：禁用。
[1]	GPIO0_1_WKEN	GPIO 0[1] / P0[1] 外部唤醒使能 1：使能；0：禁用。

[0]	GPIO0_0_WKEN	GPIO 0[0] / P0[0] 外部唤醒使能 1：使能；0：禁用。

8.3 实现说明

8.3.1 上拉实现

LKS32MC08x 系列芯片，通过内部模拟电路进行上拉功能实现。所有 GPIO 都有上拉控制寄存器 PUE，但不是所有 GPIO 都有上拉电路。配备上拉功能的 GPIO 如下：

表 8-21 GPIO 上拉资源分布表

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

8.4 应用指南

8.4.1 配置锁定

芯片提供对 GPIO 配置的保护功能。当 LCKR_PRT 写保护使能时，3 组 GPIO 的 GPIO_LCKR 不能修改，GPIO_PIE / GPIO_POE / GPIO_PUE / GPIO_PDE / GPIO_ODE / GPIO_F3210 / GPIO_F7654 / GPIO_FBA98 / GPIO_FFEDC 不能修改；

若需要修改 GPIO 配置，应先解除 LCKR_PRT 写保护，然后将对应 GPIO 的 GPIO_LCKR 写 0，解除配置锁定，然后修改 GPIO 配置。

示例如下：

```
    GPIO0_PIE    = 0x1234;
    GPIO1_PIE    = 0x7777;
    GPIO2_PIE    = 0xF000;
//------------------------------------------------------
//  lock specific gpio
    GPIO0_LCKR   = 0x0100;       //lock gpio0 here
    GPIO1_LCKR   = 0xFFFF;       //lock gpio1 here
    GPIO2_LCKR   = 0x8000;       //lock gpio2 here
//------------------------------------------------------
```

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

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

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

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

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

      i=1000;
```

```
while(i--);
PASS;
```

8.4.2 外部中断

示例如下:

```
GPIO0_PIE = 0x0080;          // 使能 P0[7]输入

NVIC_EnableIRQ(GPIO_IRQn); //使能 GPIO 中断
__enable_irq();                    //使能中断
i = 1000;
while(i--);
// P0[7] IO 上外接方波信号
EXTI_CR0 = 0x8000;   // 使能 p0[7]上升沿触发，产生外部中断
while(irq_flag != 2);     // 外部信号翻转两次，产生两次中断，irq_flag 在 GPIO 中断处理
                               程序中递增两次
EXTI_CR0 = 0x4000;   // 使能 p0[7]下降沿触发，产生外部中断
while(irq_flag != 4);
EXTI_CR0 = 0xC000;   // 同时使能 P0[7]上升沿、下降沿触发，产生外部中断
while(irq_flag != 8);
EXTI_CR0 = 0x0000;   // 同时禁用 P[7]上下沿触发，将无法产生外部中断
i = 1000;
while(i--);
if(irq_flag != 8)FAIL;
i = 1000;
while(i--);
PASS;
}
```

8.4.3 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。

9 CRC

9.1 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”。

9.2 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。

9.3 基本概念

9.3.1 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 $C(X)=X^5+X^3+X^2+X+1$。

9.3.2 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。

9.3.3 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。

9.3.4 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
```

9.4 寄存器

9.4.1 地址分配

CRC 的基地址是 0x4001_2400，寄存器列表如下：

表 9-1 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

9.4.2 寄存器描述

9.4.2.1 CRC_DR CRC 信息码寄存器

地址：0x4001_2400

复位值：0x0

表 9-2 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。

9.4.2.2 CRC_CR CRC 控制寄存器

地址：0x4001_2404

复位值：0x0

表 9-3 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。

9.4.2.3 CRC_INIT CRC 初始码寄存器

地址：0x4001_2408

复位值：0x0

表 9-4 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。

9.4.2.4 CRC_POL CRC 生成码寄存器

地址：0x4001_240C

复位值：0x0

表 9-5 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

10 ADC

10.1 概述

LKS32MC08x 系列芯片集成了 1 个 12bit SAR ADC，采用双路同步采样设计，可以同一个时刻采样两个通道的信号，每个采样电路对应 10 路输入通道，共 20 路。主要有以下特性：

- 3Msps 采样及转换速率，48MHz 工作频率。
- 最多支持 20 通道选择。
- 支持软件、硬件触发功能。
- 可以与 MCPWM、UTimer 单元联动，触发指示信号 ADC_TRIGGER 可通过 GPIO 送出用于调试
- 支持单段、双段、四段自定义采样序列采样，序列次数和通道号可灵活配置。
- 支持连续采样模式。
- 支持 DMA 搬移数据。
- 支持 ADC 硬件数值比较功能，可以设置上或下阈值，达到阈值触发 ADC 比较中断。
- 支持左对齐、右对齐模式。

ADC 采样的量词含义约定：

1 次采样：完成对应的一个通道的模拟信号量到数据信号量的采样转换并存储数字量至 ADC_DATx 寄存器；

1 段采样：可能包含 1 次或若干次采样，若干次采样可以是相同的模拟量通道，也可以是不同的模拟量通道。采样开始通常由 MCPWM、UTimer 或软件进行触发，一个触发信号完成一段采样，采样完成后产生相应的段采样完成中断；以 MCPWM 触发的四段采样为例，每段采样 3 次（即完成 3 个模拟量的采样），TADC[0]触发 ADC 开始第一段采样，第一段采样完成后 ADC 进入等待状态，等待 TADC[1]触发事件发生；TADC[1]发生后，触发 ADC 开始第二段采样；同理，TADC[2]/TADC[3]分别触发第三段和第四段采样。

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

10.1.1 功能框图

ADC 接口包括 20 个数据寄存器（ADC 20 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。

ADC 配备双采样电路，同一时刻进行两个通道的采样，然后依次先后完成到数字量的转换。

数据寄存器 ADC_DATx 用于存储 ADC 第 x 个采样得到的数据量。被转换的模拟信号来源由寄存器 ADC_CHNx 中的某 5bit 进行选择（详见 9.2.3 信号来源寄存器章节）。以 ADC_CHN0 为例，位[4:0]选择第 0 个采样的模拟通道号，通道号 CH0~CH19 任选，若 ADC_CHN0[4:0]=0，ADC_CHN0[12:8]=3，

则第 0 个采样的模拟量对应通道 CH0，第 1 个采样的模拟量对应通道 CH3，以此类推。注意，由于双采样电路同步工作，因此第 0 个采样和第 1 个采样实际是同时完成的，即采样电路一次就消费 ADC_CHNx 选择的两个模拟通道。

分段采样次数寄存器 ADC_CHNT0/1 控制每轮采样的个数，1 表示 1 个采样，2 表示 2 个采样，…… ，12 表示 12 个采样，…… ，20 表示 20 个采样。双采样电路一次完成两个采样，如果配置的采样个数是奇数，则最后一次仅完成一个采样的模拟信号的转换。

控制逻辑根据触发配置寄存器 ADC_TRIG 选择来自 MCPWM 或通用定时器 UTimer 的触发信号启动一轮采样或者软件触发启动。MCPWM/ UTimer 会送出定时触发信号 TADC[0]/TADC[1]/TADC[2]/TADC[3]，可选择 TADC[0]/TADC[1]/TADC[2]/TADC[3]作为触发信号。

一段转换（一段内的所有通道采样转换完毕）完成，触发 ADC 转换完成中断。多段触发模式下，每一段转换完成可触发产生一个转换完成中断。

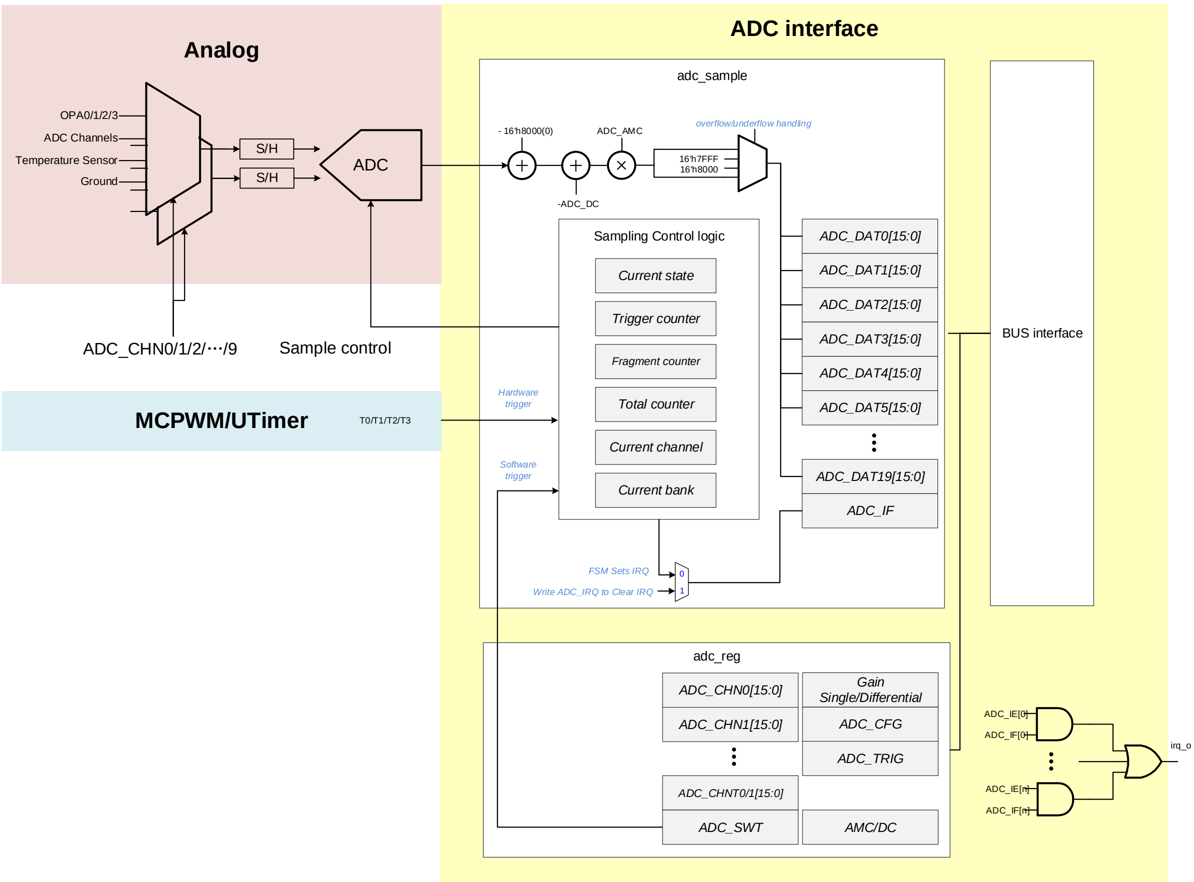


图 10-1 ADC 采集模块功能框图

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。同时每个通道的 ADC 增益也可以通过寄存器配置（两档增益可选）。控制寄存器使得用户可以配置采样个数，提高采样频率或降低采样功耗。

芯片上电的默认状态下，ADC 模块是关闭的。通过将 SYS_AFE_REG5.ADCPDN 设置为 1 来开启 ADC，ADC 开启前，需要先开启 BGP、4M RC 时钟和 PLL 模块，并通过配置 SYS_AFE_REG7.ADCCLKSEL 选择 ADC 工作频率。

ADCPDN 的说明见模拟寄存器 5.2.9 SYS_AFE_REG5

ADCCLKSEL<1:0>的说明见模拟寄存器 5.2.11 SYS_AFE_REG7

ADC 完成一次转换需要 16 个 ADC 时钟周期，其中 13 个为转换周期，3 个为采样周期。在 ADC 时钟设为 48MHz 时，转换速率是 3Msps。

10.1.2 ADC 触发方式

- 支持单段触发、两段触发、四段触发完成采样
- 单段触发可以设置触发事件发生次数，当触发事件发生一定次数后才开始采样
- 两段触发的触发源只能为 MCPWM/UTimer 的定时信号 T0+T1，或两次软件触发
- 四段触发的触发源只能为 MCPWM/UTimer 的定时信号 T0+T1+T2+T3，或四次软件触发
- 每段触发完成均可配置产生中断
- 触发指示信号可以通过 GPIO 送出用于调试

表 10-1 ADC 分段触发方式触发源配置

分段触发方式	当前段	可能的触发源		
		MCPWM	UTIMER	软件触发
单段触发	-	N 次 T0 或 N 次 T1 或 N 次 T2 或 N 次 T3 或 N 次 T0/T1/T2/T3	N 次 T0 或 N 次 T1 或 N 次 T2 或 N 次 T3 或 N 次 T0/T1/T2/T3	一次软件触发
两段触发	第一段	T0	T0	软件触发
	第二段	T1	T1	软件触发
四段触发	第一段	T0	T0	软件触发
	第二段	T1	T1	软件触发
	第三段	T2	T2	软件触发
	第四段	T3	T3	软件触发

其中 N=ADC_TRIG.SINGLE_TNCT。可以同时开启 MCPWM 和 UTIMER 的触发使能，但由于时序控制的原因，通常不进行这样的配置。

各段采样的个数由 ADC_CHNT0/1 进行控制。(1 表示 1 个采样，2 表示 2 个采样，… ，12 表示 12 个采样)。

表 10-2 四段触发分段采样通道数示例

第 n 段	ADC_CHNT0/1	寄存器值	当前段采样个数
1	ADC_CHNT0[4:0]	4	4
2	ADC_CHNT0[12:8]	1	1
3	ADC_CHNT1[4:0]	6	6
4	ADC_CHNT1[12:8]	1	1

10.1.3 ADC 通道选择

每个 ADC 模块有 10 个通道信号来源寄存器，控制采样序列 0~19 的信号选择。ADC_CHN0 控

制采样 0~1， ADC_CHN1 控制采样 2~3，… ， ADC_CHN9 控制序列 18~19。每个序列选择范围都是 0~19，对应通道 0~19， 也就是可以对某一个通道进行多次采样。每一个采样对应一个结果寄存器，转换结束后，可以直接到对应结果寄存器中获取到 ADC 采样结果。

表 10-3 ADC 通道选择与寄存器配置对应关系

ADC 采样序列	对应的采样数据寄存器	对应信号来源寄存器
第 0 个采样	ADC_DAT0	ADC_CHN0[4:0]
第 1 个采样	ADC_DAT1	ADC_CHN0[12:8]
第 2 个采样	ADC_DAT2	ADC_CHN1[4:0]
第 3 个采样	ADC_DAT3	ADC_CHN1[12:8]
	……	
第 18 个采样	ADC_DAT18	ADC_CHN9[4:0]
第 19 个采样	ADC_DAT19	ADC_CHN9[12:8]

10.1.4 ADC 中断

ADC 中断信号在每段采样完成后置高。

软件或硬件触发事件如果在 ADC 工作时发生，则产生异常触发中断

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

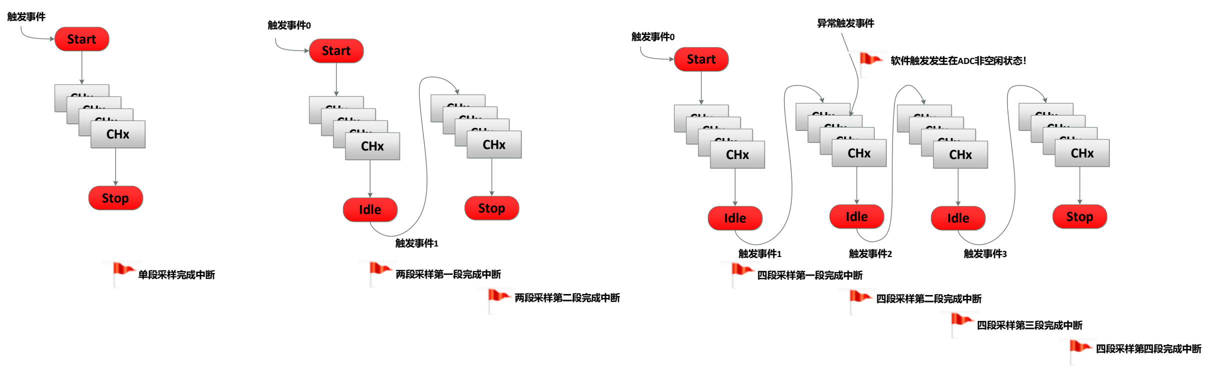


图 10-2 ADC 中断产生

10.1.5 ADC 输出数制

ADC 输出数据为 12bit 补码，输入信号 0 对应 12h’0000_0000_0000，以 1 倍增益配置为例，输入信号-2.4V 对应 12h’1000_0000_0000，输入信号+2.4V 对应 12h’0111_1111_1111。ADC 转换后的 12bit 补码需扩展为 16BIT 存入 16bit 位宽的采样数据寄存器，左对齐/右对齐可根据配置寄存器进行设置。以 12’h1000_0000_1101 为例，如果配置为左对齐，右侧补 4 个 0，存入 ADCx_DAT 的值为 16’h1000_0000_1101_0000；如果配置为右对齐，左侧进行符号扩展，存入 ADCx_DAT 的值为 16’h1111_1000_0000_1101。推荐统一使用左对齐方式。但由于 ADC 存在增益校正系数，在增益校正值不为 1(0x200)时，左对齐后 4 位可能不为全 0。

需要注意的是，由于存在增益校正和直流偏置校正，ADC 最终数据可能会超过 12bit 有符号数的表示范围，比如在右对齐的模式下，ADC 某次转换的数字量可能为 0xF745，此时直接进行低 12bit

的截取取出 0x745，会使得负数被作为正数处理，即发生溢出错误。亦或者 ADC 某次转换的数字量可能为 0x0810，此时直接进行低 12bit 的截取取出 0x810，会使得正数被错误地当做负数处理。因此需要将 ADC 数据作为 16bit 有符号数进行处理。

表 10-4 ADC 输出数字量数制转换

ADC 一倍增益输入模拟量数值/V	ADC 2/3 倍增益输入模拟量数值/V	转为有符号数后的数值
2.4	3.6	12’h0111_1111_1111
0	0	12’h0000_0000_0000
-2.4	-3.6	12’h1000_0000_0000

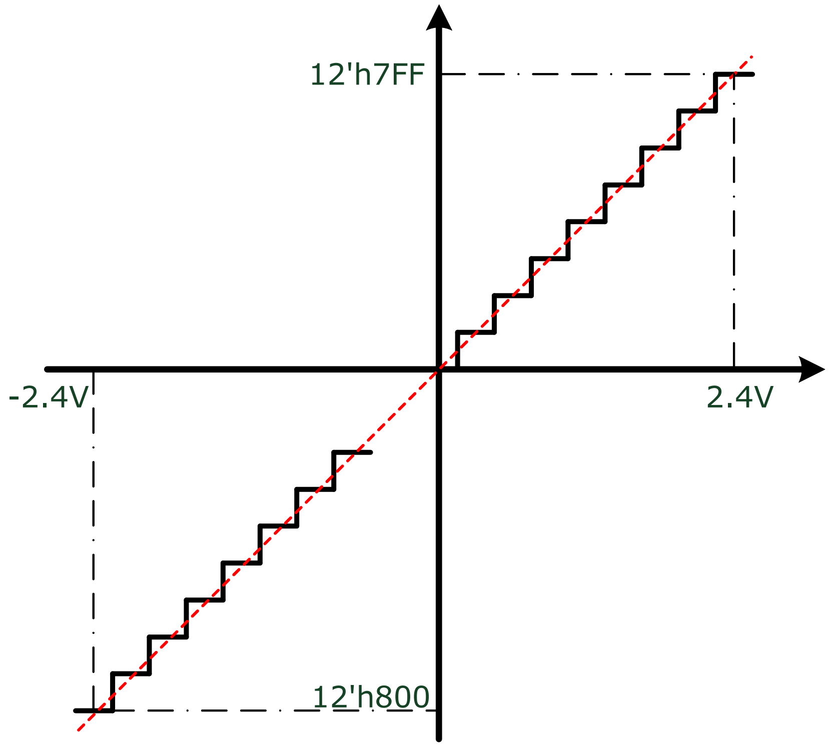


图 10-3 一倍增益设置下 ADC 模数转换数制量程

10.1.6 ADC 基准电压与量程

ADC 有 2.4V/1.2V 两种基准电压配置，通过设置 SYS_AFE_REG1.GAIN_REF 位实现选择，详见 5.2.5。

10.1.6.1 2.4V 基准电压模式

芯片工作在 5V 系统时，推荐使用 2.4V 的默认基准电压配置，此时 ADC 正常工作的最低供电电压为 3.2V。

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±2.4V 的输入信号幅度，2/3 倍增益模式下，对应最大±3.6V 的输入信号幅度。

在 ADC 采样通道配置为运放的输出信号时（即 OPA0~OPA3），应选择合适的运放增益，使得具体应用上的最大信号可被放大到接近+/-3.3V 的水平，同时将 ADC 配置为 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（放大倍数选择方式详见 4.6OPA 运算放大器），则放大后的信号约为+/-3.18V。

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

高低增益选择由 ADC_GAIN0/1 增益寄存器进行控制。

10.1.6.2 1.2V 基准电压模式

芯片工作在 3.3V 系统时，必须使用 1.2V 的默认基准电压配置，此时 ADC 正常工作的最低供电电压为 2.8V。通过设置 SYS_AFE_REG1.GAIN_REF='1'将 ADC 配置为 1.2V 基准电压模式。这一模式下相比 2.4V 模式，ADC 的有效精度(ENOB)基本不变。

这一模式和 2.4V 模式的区别在于：

1）对于最常用的 ADC 2/3 倍增益设置（对应+/-3.6V 量程），没有任何影响。芯片会自动调整增益为 1/3，因此同一个信号的 ADC 采样值不变，量程也不变。

2）对于 ADC 1 倍增益设置，SYS_AFE_REG1.GAIN_REF='1'后，使用该档增益设置的 ADC 通道，其量程变为+/-1.2V，同一个信号对应的 ADC 值也会变大一倍

3）对于内部的温度传感器，ADC 的增益设置不起作用。ADC 采样温度传感器的转换后的数值需要除以 2，再使用 a/b 系数计算温度。温度传感器 a/b 系数的描述详见 4.8TMP 温度传感器。由于温度系数 a/b 是在 ADC 2.4V 基准电压下校正得到的，因此这种工作模式下温度精度会比 2.4V 基准略低，推荐使用 2.4V 基准。

10.1.7 ADC 校正

ADC 硬件接口模块可以进行直流偏置校正与增益校正。

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

芯片出厂时已经过工厂标定，标定数据存放在 NVR 中，芯片上电会自动加载。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。

需要注意的是，ADC 有高增益和低增益两档配置，两种配置对应两套校正参数，每套校正数据分别包含一个 DC offset(以下记为 DC_{offset})和一个增益校正值 $AMP_{correction}$。同时每套校正参数包含两组 DC/AMC 分别对应采样电路 a/b。高增益对应的校正系数为 ADC_DC_A1/ADC_AMC_A1 和 ADC_DC_B1/ADC_AMC_B1，低增益对应的校正系数为 ADC_DC_A0/ADC_AMC_A0 和 ADC_DC_B0/ADC_AMC_B0。

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}。

10.1.8 ADC 配置流程

推荐配置流程：

1. 打开 ADC 模拟开关，选择 ADC 工作频率

   通过配置寄存器 SYS_AFE_REG5.ADCPDN 可以开启 ADC，ADC 开启前，需要先开启 BGP、4M RC 时钟和 PLL 模块。通过配置寄存器 SYS_AFE_REG7.ADCCLKSEL[5:4]设置 ADC 工作频率，00 为 48MHz，10 为 12MHz，11 为 24MHz。

2. 配置 ADC 数据输出格式

   ADC 的输出格式可配置为左对齐或者右对齐，配置的是 ADC0_CFG.DATA_ALIGN 寄存器，0 为左对齐，1 为右对齐。

3. 配置 ADC 采样模式

   ADC 的采样模式可配置为单段、双段、四段采样模式，配置的是 ADC0_TRIG.TRG_MODE[13:12]寄存器，00 为单段采样模式，01 为两段采样模式，11 为四段采样模式。

4. 配置 ADC 触发事件

   ADC 采样的触发事件选择，一共有 8 种采样事件可选择，配置的是 ADC0_TRIG 寄存器。单段采样模式下可以设置触发一次采样所需的事件数，配置的是 ADC0_TRIG.SINGLE_TCNT[11:8]寄存器，设置范围是 0~15，0 表示一次事件即触发，15 表示 16 次事件才触发。

5. 配置 ADC 量程

   ADC 的基准电压可以通过配置寄存器 SYS_AFE_REG1.GAIN_REF 进行选择，0 为 2.4V，1 为 1.2V。ADC 的两种增益模式可以通过配置 ADC_GAIN0/1 增益寄存器进行逐个通道选择，0 为低增益（2/3 倍），1 为高增益（1 倍）。

6. 配置 ADC 通道数，选择采样信号源

   配置各段采样模式下，采样的通道个数，配置的是 ADC0_CHNT0、ADC0_CHNT1 寄存器，设置范围是 1~20，1 代表一个通道。ADC 的采样信号源配置，通过配置 ADC0_CHN0、ADC0_CHN1 等寄存器选择，设置范围是 0~15。

7. 配置 ADC 中断

   ADC 一共有七种中断：第一段~第四段采样完成中断、软件触发发生在非空闲状态下的中断、硬件触发发生在非空闲状态下的中断、ADC0_DTA0 超阈值中断。通过配置 ADC0_IE 寄存器可以使能以上中断。即使未开启中断，中断事件仍能置位 ADC0_IF，但不会提出中断请求。

10.2 寄存器

10.2.1 地址分配

ADC 在芯片中的基地址是 0x4001_1400，寄存器列表如下：

表 10-5 ADC0 寄存器列表

名称	偏移地址	说明
ADC0_DAT0	0x00	ADC 第 0 个采样数据
ADC0_DAT1	0x04	ADC 第 1 个采样数据
ADC0_DAT2	0x08	ADC 第 2 个采样数据
ADC0_DAT3	0x0C	ADC 第 3 个采样数据
ADC0_DAT4	0x10	ADC 第 4 个采样数据
ADC0_DAT5	0x14	ADC 第 5 个采样数据
ADC0_DAT6	0x18	ADC 第 6 个采样数据
ADC0_DAT7	0x1C	ADC 第 7 个采样数据
ADC0_DAT8	0x20	ADC 第 8 个采样数据
ADC0_DAT9	0x24	ADC 第 9 个采样数据
ADC0_DAT10	0x28	ADC 第 10 个采样数据
ADC0_DAT11	0x2C	ADC 第 11 个采样数据
ADC0_DAT12	0x30	ADC 第 12 个采样数据
ADC0_DAT13	0x34	ADC 第 13 个采样数据
ADC0_DAT14	0x38	ADC 第 14 个采样数据
ADC0_DAT15	0x3C	ADC 第 15 个采样数据
ADC0_DAT16	0x40	ADC 第 16 个采样数据
ADC0_DAT17	0x44	ADC 第 7 个采样数据
ADC0_DAT18	0x48	ADC 第 18 个采样数据
ADC0_DAT19	0x4C	ADC 第 19 个采样数据
ADC0_CHN0	0x50	ADC 第 0~1 采样信号选择
ADC0_CHN1	0x54	ADC 第 2~3 采样信号选择
ADC0_CHN2	0x58	ADC 第 4~5 采样信号选择
ADC0_CHN3	0x5C	ADC 第 6~7 采样信号选择
ADC0_CHN4	0x60	ADC 第 8~9 采样信号选择
ADC0_CHN5	0x64	ADC 第 10~11 采样信号选择
ADC0_CHN6	0x68	ADC 第 12~13 采样信号选择
ADC0_CHN7	0x6C	ADC 第 14~15 采样信号选择
ADC0_CHN8	0x70	ADC 第 16~17 采样信号选择
ADC0_CHN9	0x74	ADC 第 18~19 采样信号选择
ADC0_CHNT0	0x78	ADC 各种触发模式下前两段采样通道数
ADC0_CHNT1	0x7C	ADC 各种触发模式下后两段采样通道数
	0x80	保留
	0x84	保留
ADC0_GAIN0	0x88	ADC 第 0~9 采样保持电路增益控制

ADC0_GAIN1	0x8C	ADC 第 10~19 采样保持电路增益控制
ADC0_DC_A0	0x90	ADC 采样保持电路 A 非 1 倍增益 DC offset
ADC0_DC_A1	0x94	ADC 采样保持电路 A 1 倍增益 DC offset
ADC0_AMC_A0	0x98	ADC 采样保持电路 A 非 1 倍增益增益校正
ADC0_AMC_A1	0x9C	ADC 采样保持电路 A 1 倍增益增益校正
ADC0_DC_B0	0xA0	ADC 采样保持电路 B 非 1 倍增益 DC offset
ADC0_DC_B1	0xA4	ADC 采样保持电路 B 1 倍增益 DC offset
ADC0_AMC_B0	0xA8	ADC 采样保持电路 B 非 1 倍增益增益校正
ADC0_AMC_B1	0xAC	ADC 采样保持电路 B 1 倍增益增益校正
ADC0_IE	0xB0	ADC 中断使能
ADC0_IF	0xB4	ADC 中断标志
ADC0_CFG	0xB8	ADC 对齐模式配置
ADC0_TRIG	0xBC	ADC 采样模式配置
ADC0_SWT	0xC0	ADC 软件触发
ADC0_DAT0_TH	0xC4	ADC 通道 0 阈值寄存器

10.2.2 采样数据寄存器

10.2.2.1 ADC0_DAT0

地址：0x4001_1400

复位值：0x0

表 10-6 采样数据寄存器 ADC0_DAT0

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

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

10.2.2.2 ADC0_DAT1

地址：0x4001_1404

复位值：0x0

表 10-7 采样数据寄存器 ADC0_DAT1

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

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

10.2.2.3 ADC0_DAT2

地址：0x4001_1408

复位值：0x0

表 10-8 采样数据寄存器 ADC0_DAT2

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

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

10.2.2.4 ADC0_DAT3

地址：0x4001_140C

复位值：0x0

表 10-9 采样数据寄存器 ADC0_DAT3

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

位置	位名称	说明
[31:16]		未使用

[15:0]	DAT3	ADC 第 3 个采样数据

10.2.2.5 ADC0_DAT4

地址：0x4001_1410

复位值：0x0

表 10-10 采样数据寄存器 ADC0_DAT4

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

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

10.2.2.6 ADC0_DAT5

地址：0x4001_1414

复位值：0x0

表 10-11 采样数据寄存器 ADC0_DAT5

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

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

10.2.2.7 ADC0_DAT6

地址：0x4001_1418

复位值：0x0

表 10-12 采样数据寄存器 ADC0_DAT6

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

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

10.2.2.8 ADC0_DAT7

地址：0x4001_141C

复位值：0x0

表 10-13 采样数据寄存器 ADC0_DAT7

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

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

10.2.2.9 ADC0_DAT8

地址：0x4001_1420

复位值：0x0

表 10-14 采样数据寄存器 ADC0_DAT8

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

位置	位名称	说明
[31:16]		未使用

[15:0]	DAT8	ADC 第 8 个采样数据

10.2.2.10 ADC0_DAT9

地址：0x4001_1424

复位值：0x0

表 10-15 采样数据寄存器 ADC0_DAT9

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

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

10.2.2.11 ADC0_DAT10

地址：0x4001_1428

复位值：0x0

表 10-16 采样数据寄存器 ADC0_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	ADC 第 10 个采样数据

10.2.2.12 ADC0_DAT11

地址：0x4001_142C

复位值：0x0

表 10-17 采样数据寄存器 ADC0_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	ADC 第 11 个采样数据

10.2.2.13 ADC0_DAT12

地址：0x4001_1430

复位值：0x0

表 10-18 采样数据寄存器 ADC0_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	ADC 第 12 个采样数据

10.2.2.14 ADC_DAT13

地址：0x4001_1434

复位值：0x0

表 10-19 采样数据寄存器 ADC0_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	ADC 第 13 个采样数据

10.2.2.15 ADC0_DAT14

地址：0x4001_1438

复位值：0x0

表 10-20 采样数据寄存器 ADC0_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	ADC 第 14 个采样数据

10.2.2.16 ADC0_DAT15

地址：0x4001_143C

复位值：0x0

表 10-21 采样数据寄存器 ADC0_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	ADC 第 15 个采样数据

10.2.2.17 ADC0_DAT16

地址：0x4001_1440

复位值：0x0

表 10-22 采样数据寄存器 ADC0_DAT16

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

位置	位名称	说明
[31:16]		未使用
[15:0]	DAT16	ADC 第 16 个采样数据

10.2.2.18 ADC0_DAT17

地址：0x4001_1444

复位值：0x0

表 10-23 采样数据寄存器 ADC0_DAT17

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

位置	位名称	说明
[31:16]		未使用
[15:0]	DAT17	ADC 第 17 个采样数据

10.2.2.19 ADC0_DAT18

地址：0x4001_1448

复位值：0x0

表 10-24 采样数据寄存器 ADC0_DAT18

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

位置	位名称	说明
[31:16]		未使用

[15:0]	DAT18	ADC 第 18 个采样数据

10.2.2.20 ADC0_DAT19

地址：0x4001_144C

复位值：0x0

表 10-25 采样数据寄存器 ADC0_DAT19

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

位置	位名称	说明
[31:16]		未使用
[15:0]	DAT19	ADC 第 19 个采样数据

10.2.3 信号来源寄存器

10.2.3.1 ADC0_CHN0

地址：0x4001_1450

复位值：0x0

表 10-26 信号来源寄存器 ADC0_CHN0

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

位置	位名称	说明
[31:13]		未使用
[12:8]	DS1	ADC 第 1 个采样信号选择
[7:5]		未使用
[4:0]	DS0	ADC 第 0 个采样信号选择

ADC 第 0/1 次采样是同步采样。

10.2.3.2 ADC0_CHN1

地址：0x4001_1454

复位值：0x0

表 10-27 信号来源寄存器 ADC0_CHN1

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

位置	位名称	说明
[31:13]		未使用
[12:8]	DS3	ADC 第 3 个采样信号选择
[7:5]		未使用
[4:0]	DS2	ADC 第 2 个采样信号选择

ADC 第 2/3 次采样是同步采样。

10.2.3.3 ADC0_CHN2

地址：0x4001_1458

复位值：0x0

表 10-28 信号来源寄存器 ADC0_CHN2

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

位置	位名称	说明
[31:13]		未使用
[12:8]	DS5	ADC 第 5 个采样信号选择
[7:5]		未使用
[4:0]	DS4	ADC 第 4 个采样信号选择

ADC 第 4/5 次采样是同步采样。

10.2.3.4 ADC0_CHN3

地址：0x4001_145C

复位值：0x0

表 10-29 信号来源寄存器 ADC0_CHN3

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

位置	位名称	说明
[31:13]		未使用
[12:8]	DS7	ADC 第 7 个采样信号选择
[7:5]		未使用
[4:0]	DS6	ADC 第 6 个采样信号选择

ADC 第 6/7 次采样是同步采样。

10.2.3.5 ADC0_CHN4

地址：0x4001_1460

复位值：0x0

表 10-30 信号来源寄存器 ADC0_CHN4

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

位置	位名称	说明
[31:13]		未使用
[12:8]	DS9	ADC 第 9 个采样信号选择
[7:5]		未使用
[4:0]	DS8	ADC 第 8 个采样信号选择

ADC 第 8/9 次采样是同步采样。

10.2.3.6 ADC0_CHN5

地址：0x4001_1464

复位值：0x0

表 10-31 信号来源寄存器 ADC0_CHN5

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

位置	位名称	说明
[31:13]		未使用
[12:8]	DS11	ADC 第 11 个采样信号选择
[7:5]		未使用
[4:0]	DS10	ADC 第 10 个采样信号选择

ADC 第 10/11 次采样是同步采样。

10.2.3.7 ADC0_CHN6

地址：0x4001_1468

复位值：0x0

表 10-32 信号来源寄存器 ADC0_CHN6

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

位置	位名称	说明
[31:13]		未使用
[12:8]	DS13	ADC 第 13 个采样信号选择
[7:5]		未使用
[4:0]	DS12	ADC 第 12 个采样信号选择

ADC 第 12/13 次采样是同步采样。

10.2.3.8 ADC0_CHN7

地址：0x4001_146C

复位值：0x0

表 10-33 信号来源寄存器 ADC0_CHN7

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

位置	位名称	说明
[31:13]		未使用
[12:8]	DS15	ADC 第 15 个采样信号选择
[7:5]		未使用
[4:0]	DS14	ADC 第 14 个采样信号选择

ADC 第 14/15 次采样是同步采样。

10.2.3.9 ADC0_CHN8

地址：0x4001_1470

复位值：0x0

表 10-34 信号来源寄存器 ADC0_CHN8

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

位置	位名称	说明
[31:13]		未使用
[12:8]	DS17	ADC 第 17 个采样信号选择
[7:5]		未使用
[4:0]	DS16	ADC 第 16 个采样信号选择

ADC 第 16/17 次采样是同步采样。

10.2.3.10 ADC0_CHN9

地址：0x4001_1474

复位值：0x0

表 10-35 信号来源寄存器 ADC0_CHN9

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

位置	位名称	说明
[31:13]		未使用
[12:8]	DS19	ADC 第 19 个采样信号选择
[7:5]		未使用
[4:0]	DS18	ADC 第 18 个采样信号选择

ADC 第 18/19 次采样是同步采样。

表 10-36 ADC 采样信号通道选择

ADC 采样电路输入信号正端选择	5'h00 OPA0_OUT 5'h01 OPA1_OUT 5'h02 OPA2_OUT 5'h03 OPA3_OUT 5'h04 ADC_CH4 5'h05 ADC_CH5 5'h06 ADC_CH6 5'h07 ADC_CH7 5'h08 ADC_CH8 5'h09 ADC_CH9 5'h0A ADC_CH10 5'h0B ADC_CH11 5'h0C ADC_CH12 5'h0D ADC_CH13 5'h0E ADC_CH14 5'h0F ADC_CH15 5'h10 ADC_CH16 5'h11 ADC_CH17 5'h12 Temp 5'h13 VSS

ADC 采样电路输入信号的负端统一接地。

以单段触发采样 8 个采样为例，ADC0_CHNx 寄存器里设置的第 0/1 个采样是同步采样的，2/3 是同步采样的，4/5 是同步采样的，6/7 是同步采样的。若设置的采样个数是奇数，则最后一次只仍同步采样两个通道，但只转换一个通道的采样值。

10.2.4 分段通道数寄存器

10.2.4.1 ADC0_CHNT0

地址：0x4001_1478

复位值：0x0

表 10-37 分段通道数寄存器 ADC0_CHNT0

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
[31:13]		未使用
[12:8]	S2	两段或四段采样模式下第二段采样个数
[7:5]		未使用
[4:0]	S1	单段、两段或四段采样模式下第一段采样个数

注意，ADC0_CHNT0.S1 在单段、两段、四段模式均不允许设置为 0；ADC0_CHN0.S2 在两段、四段模式下不允许设置为 0，在单段模式下由于没有用到这一字段可以为 0。

10.2.4.2 ADC0_CHNT1

地址：0x40011400_ 0x7C

复位值：0x0

表 10-38 分段通道数寄存器 ADC0_CHNT1

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

位置	位名称	说明
[31:13]		未使用
[12:8]	S4	四段采样模式下第四段采样个数

[7:5]		未使用
[4:0]	S3	四段采样模式下第三段采样个数

注意，ADC0_CHNT1.S3 在四段模式不允许设置为 0；ADC0_CHN1.S4 在四段模式下不允许设置为 0，在单段、两段模式下由于没有用到这两个字段可以为 0。

1 表示 1 个采样，2 表示 2 个采样，…… ，12 表示 12 个采样，…… ，20 表示 20 个采样。多次采样可以采样同一个模拟通道，也可以采样不同的模拟通道。

10.2.5 配置寄存器

10.2.5.1 ADC0_CFG

地址：0x4001_14B8

复位值：0x0

表 10-39 模式配置寄存器 ADC0_CFG

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

位置	位名称	说明
[31:3]		未使用
[2]	FSM_RESET	状态机复位，软件写入后状态机回到 idle 状态，完成后自动清零
[1]	TH_TYPE	ADC0_DAT0_TH 作为上阈值或下阈值 1：上阈值 0：下阈值
[0]	DATA_ALIGN	ADC0_DAT 对齐方式 0:左对齐，右端补 4'h0， 1:右对齐，左端补 4bit 符号位

10.2.5.2 ADC0_TRIG

地址：0x4001_14BC

复位值：0x0

表 10-40 触发控制寄存器 ADC0_TRIG

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

	CON_TRIG	TRG_MODE	SINGLE_TCNT	UTIMER3_CMP1_E	UTIMER3_CMP0_E	UTIMER2_CMP1_E	UTIMER2_CMP0_E	MCPWM_TRG3_E	MCPWM_TRG2_E	MCPWM_TRG1_E	MCPWM_TRG0_E
	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
	0	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:15]		未使用
[14]	CON_TRIG	连续触发使能，高有效 写 1，ADC 进入连续触发模式进行连续采样 写 0，ADC 停止采样
[13:12]	TRG_MODE	触发模式 0:单段触发； 1:两段触发； 2:保留； 3:四段触发
[11:8]	SINGLE_TCNT	单段触发模式下触发一次采样所需的事件数 （0 表示 1 次事件即触发，15 表示 16 次事件才触发）
[7]	UTIMER3_CMP1_E	4’b1xxx: UTimer3 CMP1 被使能
[6]	UTIMER3_CMP0_E	4’bx1xx: UTimer3 CMP0 被使能
[5]	UTIMER2_CMP1_E	4’bxx1x: UTimer2 CMP1 被使能
[4]	UTIMER2_CMP0_E	4’bxxx1: UTimer2 CMP0 被使能
[3]	MCPWM_TRG3_E	4’b1xxx: MCPWM T3 被使能
[2]	MCPWM_TRG2_E	4’bx1xx: MCPWM T2 被使能
[1]	MCPWM_TRG1_E	4’bxx1x: MCPWM T1 被使能
[0]	MCPWM_TRG0_E	4’bxxx1: MCPWM T0 被使能

TADC[3] = MCPWM_T3 | UTimer3_CMP1

TADC[2] = MCPWM_T2 | UTimer3_CMP0

TADC[1] = MCPWM_T1 | UTimer2_CMP1

TADC[0] = MCPWM_T0 | UTimer2_CMP0

来自两个不同设备的四个触发源相或之后送至 ADC 采样模块作为触发事件 TADC[3:0]

在进入连续采样模式之前，须保证 ADC 采样模块处于空闲状态，推荐关闭 ADC 所有硬件触发使能，同时通过写 ADC0_CFG.FSM_RESET=1；使得 ADC 采样状态机回到空闲状态。

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。ADC 触发信号在芯片内部为一个 ADC 时钟周期的窄脉冲。每发生一次 ADC 触发，ADC_TRIGGER 信号翻转一次，以便于输出捕捉。

UTimer 对 ADC 的触发信号可以通过配置 GPIO 为第 8 功能，即 Timer2/3 功能送出用于捕捉调试。

10.2.6 增益选择寄存器

10.2.6.1 ADC0_GAIN0

地址：0x4001_1488

复位值：0x0

表 10-41 增益选择寄存器 ADC0_GAIN0

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

位置	位名称	说明
[31:10]		未使用
[9]	DG9	ADC0_DAT9 采样保持电路增益选择
[8]	DG8	ADC0_DAT8 采样保持电路增益选择
[7]	DG7	ADC0_DAT7 采样保持电路增益选择
[6]	DG6	ADC0_DAT6 采样保持电路增益选择
[5]	DG5	ADC0_DAT5 采样保持电路增益选择
[4]	DG4	ADC0_DAT4 采样保持电路增益选择
[3]	DG3	ADC0_DAT3 采样保持电路增益选择
[2]	DG2	ADC0_DAT2 采样保持电路增益选择
[1]	DG1	ADC0_DAT1 采样保持电路增益选择
[0]	DG0	ADC0_DAT0 采样保持电路增益选择

0: 2/3 增益，1: 1 倍增益。

10.2.6.2 ADC0_GAIN1

地址：0x4001_148C

复位值：0x0

表 10-42 增益选择寄存器 ADC0_GAIN1

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

	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:10]		未使用
[9]	DG19	ADC0_DAT19 采样保持电路增益选择
[8]	DG18	ADC0_DAT18 采样保持电路增益选择
[7]	DG17	ADC0_DAT17 采样保持电路增益选择
[6]	DG16	ADC0_DAT16 采样保持电路增益选择
[5]	DG15	ADC0_DAT15 采样保持电路增益选择
[4]	DG14	ADC0_DAT14 采样保持电路增益选择
[3]	DG13	ADC0_DAT13 采样保持电路增益选择
[2]	DG12	ADC0_DAT12 采样保持电路增益选择
[1]	DG11	ADC0_DAT11 采样保持电路增益选择
[0]	DG10	ADC0_DAT10 采样保持电路增益选择

0: 2/3 增益，1: 1 倍增益。

10.2.7 中断寄存器

10.2.7.1 ADC0_IE

地址：0x4001_14B0

复位值：0x0

表 10-43 中断使能寄存器 ADC0_IE

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

位置	位名称	说明
[31:7]		未使用
[6]	TH_IE	ADC0_DAT0 超阈值中断使能
[5]	HERR_IE	硬件触发发生在非空闲状态中断使能
[4]	SERR_IE	软件触发发生在非空闲状态中断使能
[3]	S4FINISH_IE	第四段采样完成中断使能
[2]	S3FINISH_IE	第三段采样完成中断使能
[1]	S2FINISH_IE	第二段采样完成中断使能
[0]	S1FINISH_IE	第一段采样完成中断使能

10.2.7.2 ADC0_IF

地址：0x4001_14B4

复位值：0x0

表 10-44 中断标志寄存器 ADC0_IF

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

位置	位名称	说明
[31:7]		未使用
[6]	TH_IF	ADC0_DAT0 超阈值中断标志
[5]	HERR_IF	硬件触发发生在非空闲状态中断标志
[4]	SERR_IF	软件触发发生在非空闲状态中断标志
[3]	S4FINISH_IF	第四段采样完成中断标志
[2]	S3FINISH_IF	第三段采样完成中断标志
[1]	S2FINISH_IF	第二段采样完成中断标志
[0]	S1FINISH_IF	第一段采样完成中断标志

以上 ADC0_IF 标志位，0：表示未发生中断，1：表示发生过中断，写 1 清零

10.2.8 软件触发寄存器

10.2.8.1 ADC0_SWT

地址：0x4001_14C0

复位值：0x0

表 10-45 软件触发寄存器 ADC0_SWT

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

位置	位名称	说明
[31:16]		未使用

[15:0]	SWT	写入数据为 0x5AA5 时，产生一次软件触发

注意，软件触发采集寄存器为只写寄存器，且只有写入数据为 0x5AA5 时产生软件触发事件，一次总线的写入产生一次软件触发，数据写入产生一个软件触发后寄存器自动清零，等待后续的软件触发到来。

10.2.9 直流偏置寄存器

芯片出厂时已经过工厂标定，标定数据存放在 Flash info 中，芯片上电会自动加载。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。

10.2.9.1 ADC0_DC_A0

地址：0x4001_1490

复位值：0x0

表 10-46 直流偏置寄存器 ADC0_DC_A0

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	采样电路 A 非一倍增益 ADC DC offset

10.2.9.2 ADC0_DC_A1

地址：0x4001_1494

复位值：0x0

表 10-47 直流偏置寄存器 ADC0_DC_A1

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	采样电路 A 一倍增益 ADC DC offset

10.2.9.3 ADC0_DC_B0

地址：0x4001_14A0

复位值：0x0

表 10-48 直流偏置寄存器 ADC0_DC_B0

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	采样电路 B 非一倍增益 ADC DC offset

10.2.9.4 ADC0_DC_B1

地址：0x4001_14A4

复位值：0x0

表 10-49 直流偏置寄存器 ADC0_DC_B1

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	采样电路 B 一倍增益 ADC DC offset

10.2.10增益校正寄存器

10.2.10.1 ADC0_AMC_A0

地址：0x4001_1498

复位值：0x0

表 10-50 增益校正寄存器 ADC0_AMC_A0

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

位置	位名称	说明
[31:10]		未使用
[9:0]	AM_CALI	采样电路 A 非一倍增益 ADC 增益校正寄存器

10.2.10.2 ADC0_AMC_A1

地址：0x4001_149C

复位值：0x0

表 10-51 增益校正寄存器 ADC0_AMC_A1

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

位置	位名称	说明
[31:10]		未使用
[9:0]	AM_CALI	采样电路 A 一倍增益 ADC 增益校正寄存器

10.2.10.3 ADC0_AMC_B0

地址：0x4001_14A8

复位值：0x0

表 10-52 增益校正寄存器 ADC0_AMC_B0

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

位置	位名称	说明
[31:10]		未使用

[9:0]	AM_CALI	采样电路 B 非一倍增益 ADC 增益校正寄存器

10.2.10.4 ADC0_AMC_B1

地址：0x4001_14AC

复位值：0x0

表 10-53 增益校正寄存器 ADC0_AMC_B1

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

位置	位名称	说明
[31:10]		未使用
[9:0]	AM_CALI	采样电路 B 一倍增益 ADC 增益校正寄存器

ADC0_AMC 存储的为增益校正系数 $AMP_{correction}$，为 10bit 无符号定点数，ADCAMC[9]为整数部分，ADCAMC[8:0]为小数部分。所存值为 1 左右。

ADC 有 1 倍增益和非 1 倍增益两种配置，两种配置对应两套校正参数，每套校正数据分别包含一个 DC offset(以下记为 DC_{offset})和一个增益校正值 $AMP_{correction}$。

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

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

因为有 A/B 两路采样电路，因此有两组 DC 和 AMC 校正值，硬件会在不同通道进行数据校正时自动选择对应的 A/B 路的 DC/AMC。

10.2.11 通道 0 阈值寄存器

10.2.11.1 ADC0_DAT0_TH

地址：0x4001_14C4

复位值：0x0

表 10-54 通道 0 阈值寄存器 ADC0_DAT0_TH

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

位置	位名称	说明
[31:16]		未使用
[15:0]	DAT0_TH	通道 0 阈值寄存器

用来配置 ADC 数值比较中断，当 ADC0_DAT0 大于或小于 ADC0_DAT0_TH 设置的值时，可以触发 ADC 数值比较中断。 根据 ADC0_CFG[1] 配置，ADC0_DAT0_TH 可作为 ADC0_DAT0 的上阈值或下阈值；在一些应用上可以替代硬件比较器。 具体来说，当 ADC0_CFG.TH_TYPE=1 时，即 ADC0_DAT0_TH 为上阈值，如果 ADC0_DAT0> ADC0_DAT0_TH，则产生阈值中断；当 ADC0_CFG.TH_TYPE=0 时，即 ADC0_DAT0_TH 为下阈值，如果 ADC0_DAT0 <ADC0_DAT0_TH，则产生阈值中断。

10.3 实现说明

10.3.1 DMA 请求

只有 ADC_IF.S1FINISHI_IF，即第一段采样完成事件可以作为 DMA 请求事件。因此只有单段触发采样完成，或多段触发采样第一段采样完成时会发起 DMA 请求，且这个请求与 ADC_IE 设置无关。DMA 响应请求后会通过内部的写清零直接将 ADC_IF.S1FINISHI_IF 清零。

10.3.2 连续采样

当配置了连续采样模式时，即 ADC_TRIG.CON_TRIG=1，ADC 会反复采样单段模式下的各个通道，结束后立即开始下一轮单段采样，如果开启了单段采样中断，则会频繁发生中断，大部分情况下 CPU 来不及响应如此频繁的中断。软件写入 ADC_TRIG.CON_TRIG=0，可以关闭连续采样。

连续采样模式可以配合 ADC_DAT0 阈值中断进行使用，只有检测到信号超过阈值时才产生阈值中断通知 CPU。

连续采样可以配置采样多个通道，具体采样通道的配置与单段采样模式相同。

10.4 应用指南

10.4.1 ADC 采样触发模式

ADC 支持一段、两段、四段采样模式，每段采样需要特定的外部事件来触发开始，每段采样支持不同采样次数和采样信号通道配置。ADC 内部的状态转移描述如下，共有 8 个状态分别为采样状态 0~3，空闲状态 0~3。

第一次触发

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

第一段采样

判断是否为一段采样。

是：采样次数达到预设值 ADC_CHNT0.S1 即第一段采样次数后，ADC 回到空闲状态 0；若采样次数尚未达到预设值，继续采样。

否：采样次数达到预设值 ADC_CHNT.S1，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

进入第二段采样

第二段采样

第二段采样次数到达预设值 ADC_CHNT0.S2 即第二段采样次数，判断是否为两段采样。

是：结束本次采样，回到空闲状态 0。

否：进入空闲状态 2，等待第三次触发及第四次触发完成采样。

第三段触发

进入第三段采样，如果到了次状态，则配置一定是四段采样。

第三段采样

第三段采样次数到达预设值 ADC_CHNT1.S3 即第三段采样次数，进入空闲状态 3。

第四段触发

第四段采样

第四段采样通道数到达预设值 ADC_CHNT1.S4 即第四段采样次数，回到空闲状态 0。

各种硬件触发模式的触发条件汇总如表 10-55 所示。其中单段采样模式较为特殊，可以通过 ADC0_CFG 寄存器设置，一次 MCPWM/UTimer 定时事件即触发采样，还是多次 MCPWM/UTimer 定时事件发生到一定次数才触发采样；而两段、四段采样模式仅支持一次 MCPWM/UTimer 事件即触发采样。

此外 ADC 模块也支持通过软件向 ADC0_SWT 写入 0x5AA5 的方式触发采样，软件触发也仅支持写入一次即触发。

表 10-55 ADC 采样触发模式

<table>
<tr><th></th><th>单段触发</th><th>两段触发</th><th>四段触发</th></tr>
<tr><td rowspan="6">Timer 触发</td><td>None
(Timer trigger 使能未打开)</td><td rowspan="6">第一段 TADC[0]
第二段 TADC[1]</td><td rowspan="6">第一段 TADC[0]
第二段 TADC[1]
第三段 TADC[2]
第四段 TADC[3]</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*</td></tr>
</table>

	TADC[0]/TADC[1]/ TADC[2]/TADC[3]		
软件触发	向 ADC_SWT 写入 16'h5aa5	第一段向 ADC0_SWT 写入 0x5AA5 第二段向 ADC0_SWT 写入 0x5AA5	第一段向 ADC0_SWT 写入 0x5AA5 第二段向 ADC0_SWT 写入 0x5AA5 第三段向 ADC0_SWT 写入 0x5AA5 第四段向 ADC0_SWT 写入 0x5AA5

*C 次通过 ADC0_TRIG.SINGLE_TCNT 设置。ADC0_TRIG.SINGLE_TCNT 只在单段触发下使用，如果同时使能了 TADC[3:0]，则 4 个触发源都会被计数，到 SINGLE_TCNT 次触发一次 ADC 采样转换。

10.4.1.1 单段触发模式

单段触发收到一次触发完成一段采样动作，一段采样可能包含多次对模拟信号的采样，次数由分段采样次数寄存器配 ADC_CHNT0[4:0]进行配置，寄存器数值为 1~20 时，对应的采样次数为 1~20。

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 ADC0_DAT0、ADC0_DAT1、ADC0_DAT2、ADC0_DAT3。

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

每个采样的信号源通过信号来源寄存器 ADC_CHN0/1/2.../19 进行配置选定，信号源的选定需在触发前完成，且在一次采样过程完成前不应该改变。

完成一段采样动作后，进入空闲状态，并产生采样完成中断。

以 MCPWM 触发单段采样为例，设置 TADC[2]发生 4 次才进行触发，状态转移如图 10-4 所示。

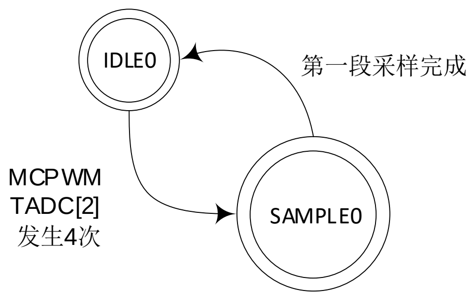


图 10-4 ADC 单段采样状态转移图

10.4.1.2 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。

触发事件可以是来自外部的 MCPWM/UTimer 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_CHNT0[4:0]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_CHNT0[12:8]次

采样。采样次数均通过分段采样次数寄存器 ADC_CHNT0 进行配置。

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_DAT0、ADC0_DAT1，第二段采样转换后的数据会依次填充到 ADC0_DAT2、ADC0_DAT3、ADC0_DAT4。

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 10-5 所示。

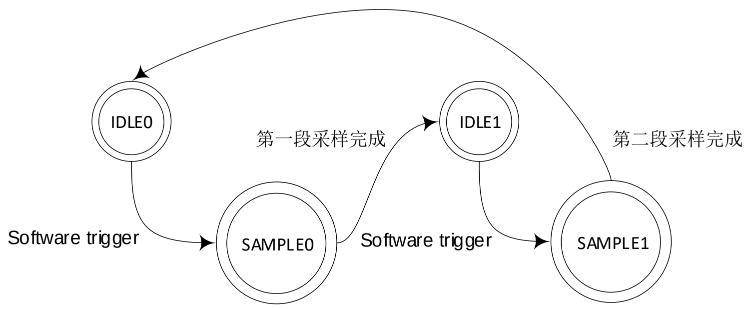


图 10-5 ADC 两段采样状态转移图

10.4.1.3 四段触发模式

与两段触发类似。四段的触发源分别为 TADC[0]、TADC[1]、TADC[2]、TADC[3]，且必须为 MCPWM/UTimer TADC[0]/TADC[1]TADC[2]TADC[3]顺序触发 ADC 的四段采样；或者也可以是 4 次软件触发采样。四段采样的第一、二、三、四段采样次数分别为 ADC_CHNT0[4:0]、ADC_CHNT0[12:8]、ADC_CHNT1[4:0]、ADC_CHNT1[12:8]。

假设四段采样配置通道数目分别为 2、3、1、5，则第一段采样转换后的数据会依次填充到 ADC0_DAT0、ADC0_DAT1，第二段采样转换后的数据会依次填充到 ADC0_DAT2、ADC0_DAT3、ADC0_DAT4，第三段采样转换后的数据会填充到 ADC0_DAT5，第四段采样转换后的数据会依次填充到 ADC0_DAT6、ADC0_DAT7、ADC0_DAT8、ADC0_DAT9、ADC0_DAT10

以 MCPWM TADC[0]/TADC[1]/TADC[2]/TADC[3]触发四段采样为例的状态转移如图 10-6 所示。

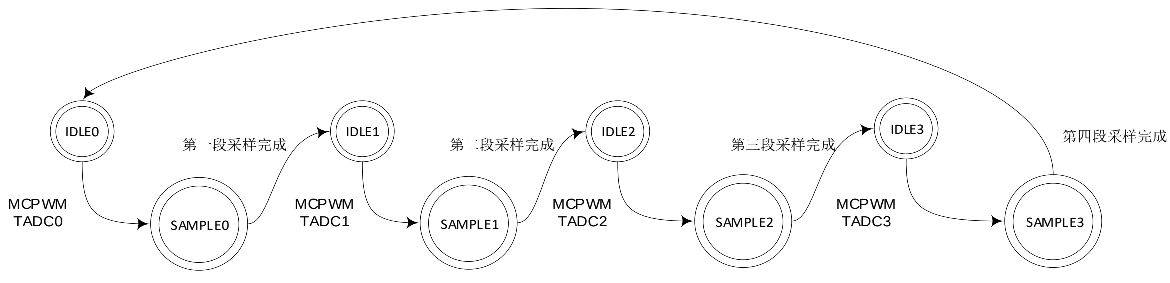


图 10-6 ADC 四段采样状态转移图

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

同理，如果使用 UTimer 对 ADC 进行定时触发，也需要配置 UTIMER_UNT2_CMP0/ UTIMER_UNT2_CMP1/ UTIMER_UNT3_CMP0/ UTIMER_UNT3_CMP1。

10.4.2 中断

10.4.2.1 单段触发采样完成中断

采样完成产生一个中断。

10.4.2.2 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。

10.4.2.3 四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，第三段采样完成产生一个中断，第四段采样完成产生一个中断。

10.4.3 配置修改

建议在 ADC 中断中进行 ADC0_CHNTx 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 ADC0_CHNx 和 ADC0_CHNT 等寄存器，需要先关闭 ADC 触发，并向 ADC0_CFG.FSM_RESET 写入 1，以复位 ADC 接口电路状态机，确保 ADC 不在工作状态。如果 ADC 在运行中配置发生变化会发生不可预判的行为。

示例程序如下

```
ADC0_TRIG_temp = ADC0_TRIG;      //保存 ADC 采样触发配置
ADC0_TRIG = 0x0000;              //关闭 ADC 采样触发
ADC0_CFG |= 0x0004;              //复位 ADC 接口电路状态机
                                 //此处进行的 ADC 采样通道和通道数的修改仅为示例
ADC0_CHNT0 = 0x0005              //修改 ADC 单段采样通道数为 5
ADC0_CHN0 = 0x0305;              //修改 ADC 第 0/1 次采样通道为模拟通道 5 和 3
ADC0_CHN1 = 0x0604;              //修改 ADC 第 2/3 次采样通道为模拟通道 4 和 6


ADC0_TRIG = ADC0_TRIG_temp;      //恢复 ADC 采样触发配置
```

10.4.4 选择对应的模拟通道

ADC 所采样信号对应的通道，请查阅表 10-36 ADC 采样信号通道选择以及 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。

11 通用定时器

11.1 概述

11.1.1 功能框图

如图 11-1 所示，通用定时器 UTIMER 主要包括 4 个独立的 Timer，分别可以独立配置运行计数时钟和滤波常数。每个 Timer 可以用于输出特定周期占空比的波形，也可以捕获外部波形进行周期占空比的检测。

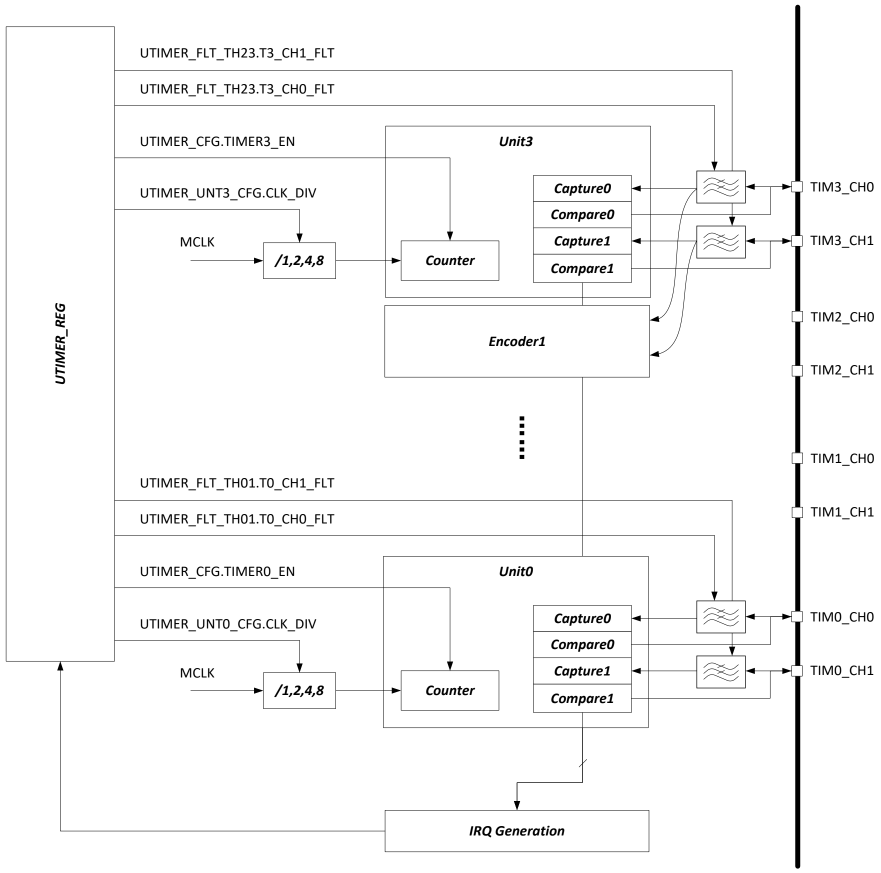


图 11-1 模块顶层功能框图

11.1.1.1 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。

11.1.1.2 寄存器模块

utimer_reg 寄存器模块，实现

对各个子模块控制寄存器的读写。

对各个子模块状态、结果寄存器的访问。

对各个子模块中断信号的处理和中断产生。

11.1.1.3 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。

11.1.1.4 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。定时器模块中一共包括 4 个独立工作的通用定时器，Timer0/Unit0、Timer1/Unit1 为 16bit 位宽，Timer2/Unit2、Timer3/Unit3 为 32bit 位宽。每个定时器包含两个通道。

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。

11.1.1.5 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。定时器模块中集成了 2 个编码器模块。其中编码器的输入分别来自 Timer2 的通道 0/1 和 Timer3 的通道 0/1。使用编码器不影响 Timer 功能。

11.1.1.6 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。

11.1.2 功能特点

定时器模块有以下特点:

- 独立工作，可工作在不同频率下
- Timer0 和 Timer1 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）
- 2 个独立工作计数器
- 对每个输入信号可以进行最大 120 个系统主时钟的滤波，即，当芯片工作在 96MHz 时钟频率下时，可以滤除 1.25uS 宽度以下毛刺
- 支持 DMA 传输

11.2 实现说明

11.2.1 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。

11.2.2 滤波

定时器模块共有 8 个/4 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，0~120 个系统时钟宽度。

输入信号滤波始终使用系统高速时钟，通常为 96MHz PLL 时钟，UTIMER_UNTx_CFG.CLK_DIV 对 Timer 计数时钟的分频对滤波时钟没有影响。

如图 11-2 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。

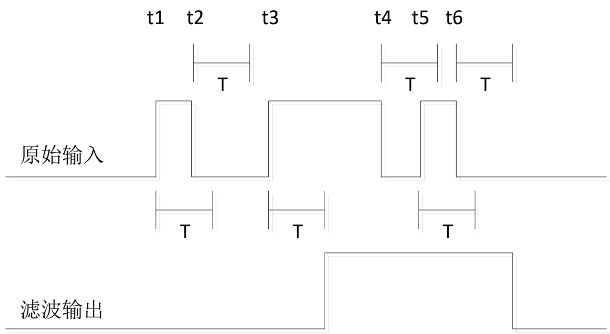


图 11-2 滤波示意图

11.2.3 模式

11.2.3.1 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)。

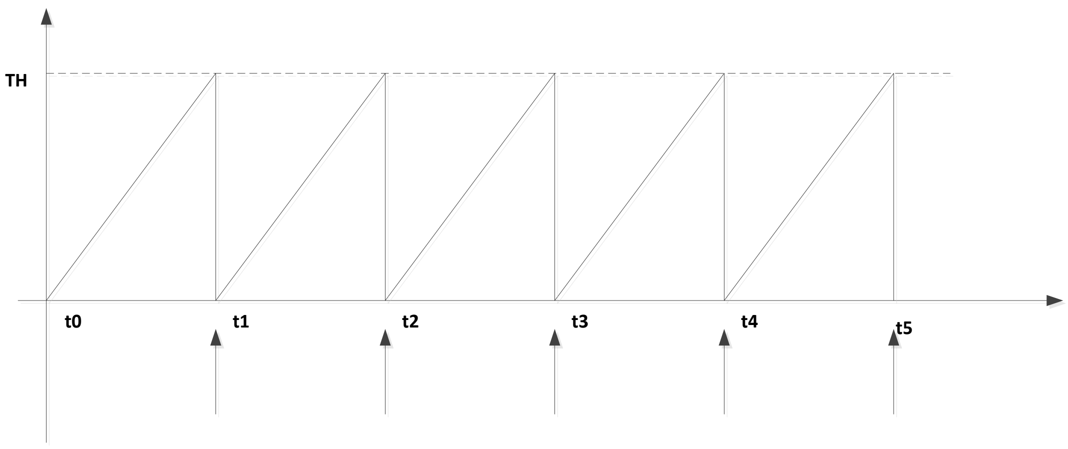


图 11-3 通用计数器

11.2.3.2 比较模式

比较模式下，计数器计数到 UTIMER_UNTx_CMP 值时，产生比较中断。比较模式可以驱动一个比较脉冲发生，在回零时，向 IO 口输出一个电平（极性可配置），在比较事件发生时，电平翻转，向 IO 口输出另一个电平。计数器回零时，仍然会产生回零中断。回零事件优先于比较事件，因此，两者同时发生时候（CMP==0），输出回零的值，而不会输出比较事件的值。

如果要实现某个 Timer 通道 0 输出全 0：

1）可以设置 UTIMER_UNTx_CFG.CH0_POL=0

2）并设置 UTIMER_UNTx_CMP0=UTIMER_UNTx_TH+1，。即 Timer 计数值回 0 时，通道输出 0，且 Timer 计数值全程不命中 CMP0。

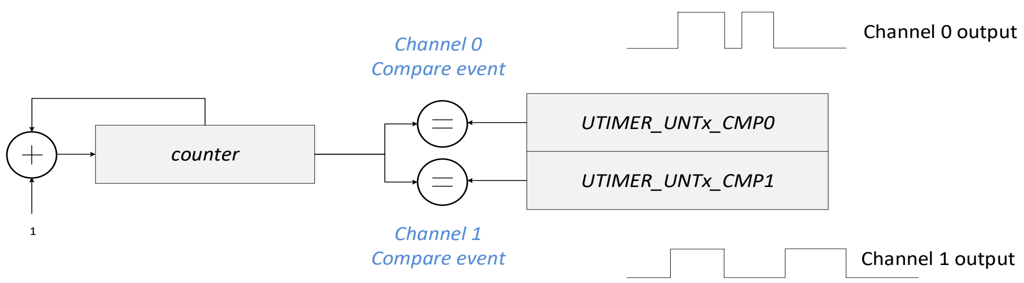
或设置 UTIMER_UNTx_CMP0=0，即 Timer 命中 CMP0 和回零事件为相同事件。

如果要实现某个 Timer 通道 0 输出全 1：

1）可以设置 UTIMER_UNTx_CFG.CH0_POL=1

2）并设置 UTIMER_UNTx_CMP0=UTIMER_UNTx_TH+1，。即 Timer 计数值回 0 时，通道输出 1，且 Timer 计数值全程不命中 CMP0。

或设置 UTIMER_UNTx_CMP0=0，即 Timer 命中 CMP0 和回零事件为相同事件。



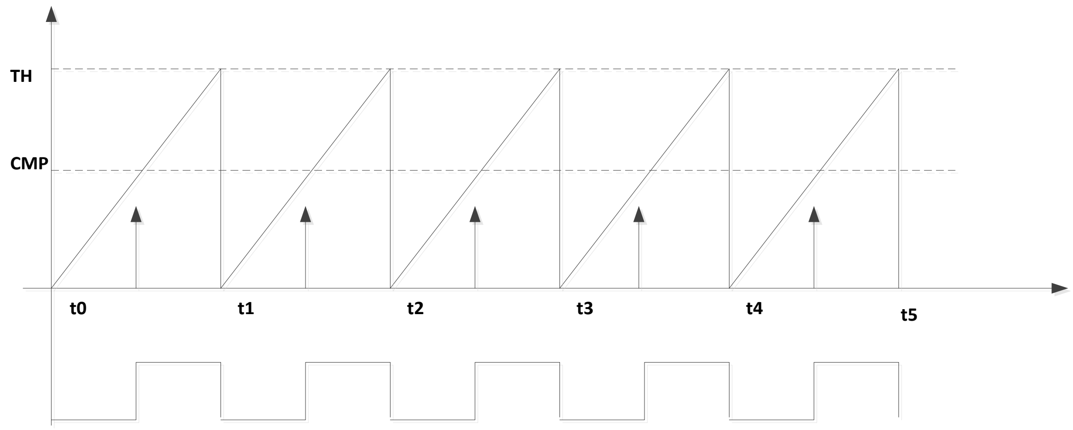
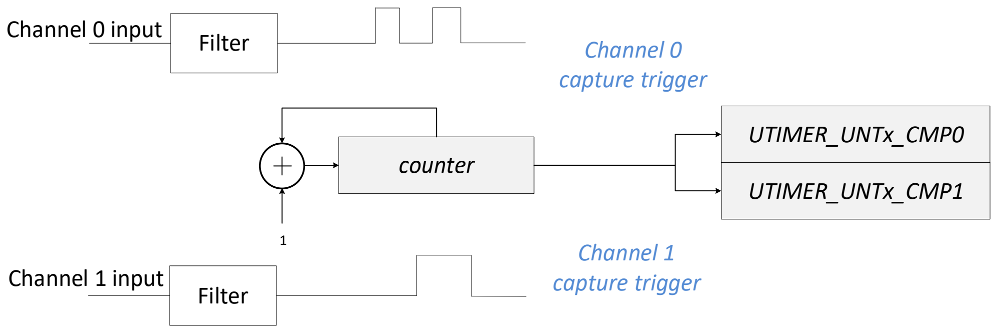


图 11-4 比较模式

11.2.3.3 捕获模式

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



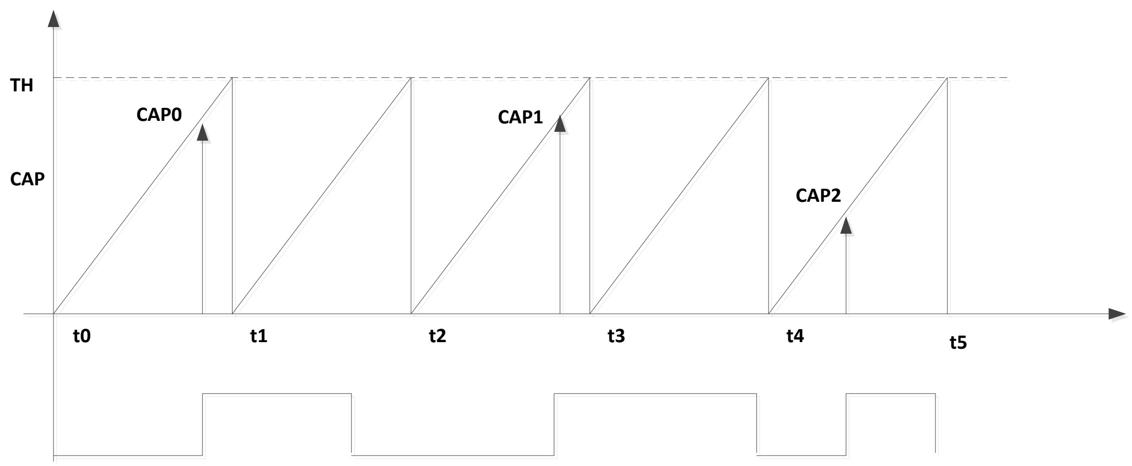


图 11-5 捕获模式

如图 10-5 所示，定时器设置为上升沿捕获。在 CAP0/CAP1/CAP2 三个时刻点，捕获到输入信号发生上升沿变化，对应时刻点的定时器计数值将存入 UTIMER_UNTx_CMP 寄存器中。

11.2.4 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 Encoder0 的两个输入信号 T1/T2 分别来自 Timer2 Channel0/1 对应的 GPIO 输入；Encoder1 的 T1/T2 信号分别来自 Timer3 Channel0/1 对应的 GPIO 输入。开启编码器功能时并不影响 Timer 功能的正常使用。

11.2.4.1 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。

以下式子表示

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 11-1 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

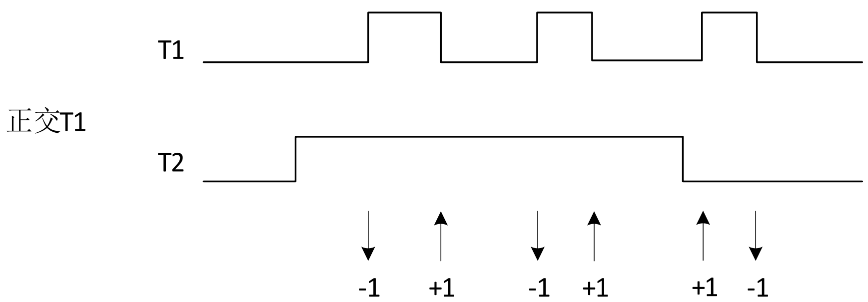


图 11-6 编码器只在 T1 时刻计数的正交编码信号计数情况

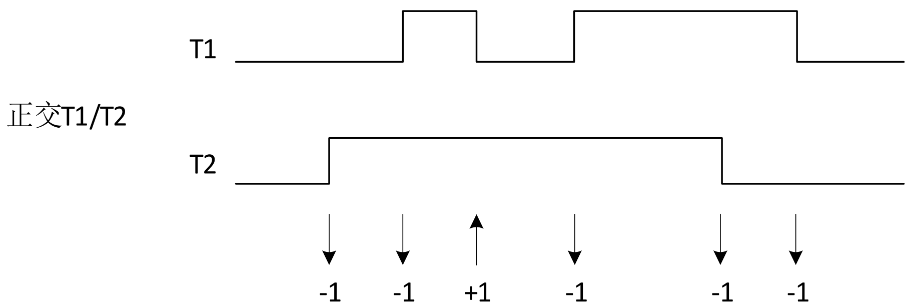


图 11-7 编码器在 T1 或 T2 时刻计数的正交编码信号计数情况

11.2.4.2 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 11-2 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

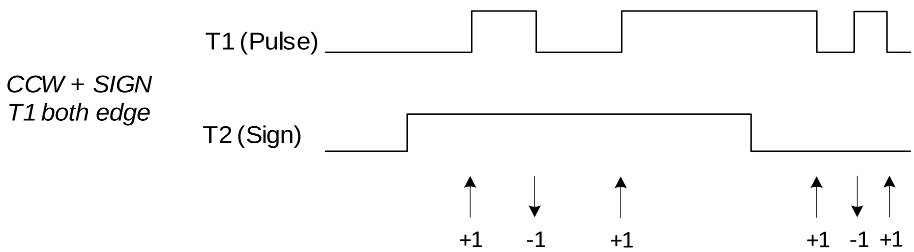


图 11-8 编码器在 T1 上升下降沿都计数的符号加脉冲信号计数情况

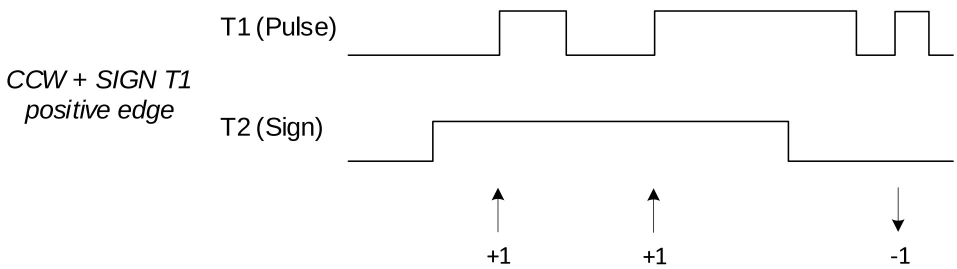


图 11-9 编码器在仅 T1 上升沿计数的符号加脉冲信号计数情况

11.2.4.3 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 11-3 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

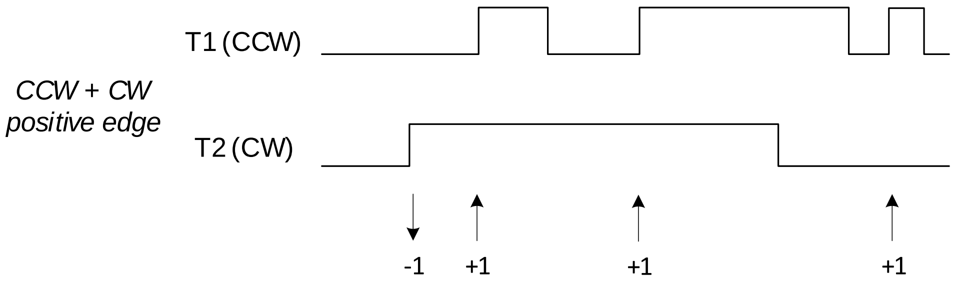


图 11-10 编码器仅在 T1/T2 上升沿计数的 CCW/CW 双脉冲信号计数情况

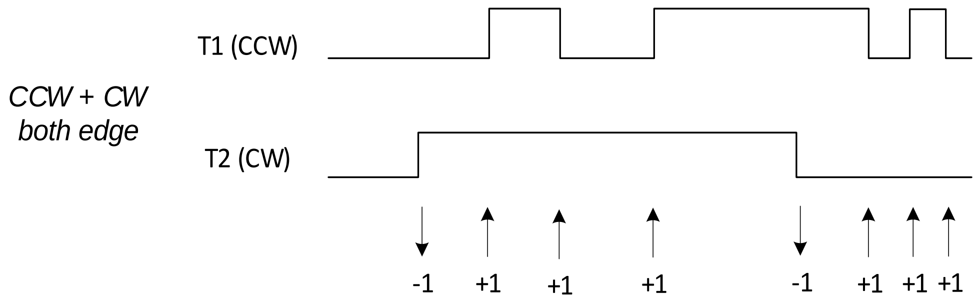


图 11-11 编码器在 T1/T2 上升下降沿计数的 CCW/CW 双脉冲信号计数情况

11.3 寄存器

11.3.1 地址分配

通用定时器模块在芯片中的基地址是 0x4001_1800，寄存器列表如下：

表 11-4 通用定时器配置寄存器地址分配

名称	偏移	描述

UTIMER_UNT0_CFG	0x00	Timer0 配置寄存器
UTIMER_UNT0_TH	0x04	Timer0 计数门限寄存器
UTIMER_UNT0_CNT	0x08	Timer0 计数值寄存器
UTIMER_UNT0_CMP0	0x0C	Timer0 比较/捕获寄存器 0
UTIMER_UNT0_CMP1	0x10	Timer0 比较/捕获寄存器 1
UTIMER_UNT0_EVT	0x14	Timer0 外部事件选择寄存器
UTIMER_UNT1_CFG	0x20	Timer1 配置寄存器
UTIMER_UNT1_TH	0x24	Timer1 计数门限寄存器
UTIMER_UNT1_CNT	0x28	Timer1 计数值寄存器
UTIMER_UNT1_CMP0	0x2C	Timer1 比较/捕获寄存器 0
UTIMER_UNT1_CMP1	0x30	Timer1 比较/捕获寄存器 1
UTIMER_UNT1_EVT	0x34	Timer1 外部事件选择寄存器
UTIMER_UNT2_CFG	0x40	Timer2 配置寄存器
UTIMER_UNT2_TH	0x44	Timer2 计数门限寄存器
UTIMER_UNT2_CNT	0x48	Timer2 计数值寄存器
UTIMER_UNT2_CMP0	0x4C	Timer2 比较/捕获寄存器 0
UTIMER_UNT2_CMP1	0x50	Timer2 比较/捕获寄存器 1
UTIMER_UNT2_EVT	0x54	Timer2 外部事件选择寄存器
UTIMER_UNT3_CFG	0x60	Timer3 配置寄存器
UTIMER_UNT3_TH	0x64	Timer3 计数门限寄存器
UTIMER_UNT3_CNT	0x68	Timer3 计数值寄存器
UTIMER_UNT3_CMP0	0x6C	Timer3 比较/捕获寄存器 0
UTIMER_UNT3_CMP1	0x70	Timer3 比较/捕获寄存器 1
UTIMER_UNT3_EVT	0x74	Timer3 外部事件选择寄存器
UTIMER_ECD0_CFG	0x80	Encoder0 配置寄存器
UTIMER_ECD0_TH	0x84	Encoder0 计数门限寄存器
UTIMER_ECD0_CNT	0x88	Encoder0 计数值寄存器
UTIMER_ECD1_CFG	0x90	Encoder1 配置寄存器
UTIMER_ECD1_TH	0x94	Encoder1 计数门限寄存器
UTIMER_ECD1_CNT	0x98	Encoder1 计数值寄存器
UTIMER_FLT_TH01	0xA0	滤波门限寄存器 01
UTIMER_FLT_TH23	0xA4	滤波门限寄存器 23
UTIMER_CFG	0xF0	通用定时器配置寄存器
UTIMER_IE	0xF4	中断使能寄存器
UTIMER_IF	0xF8	中断标志寄存器
UTIMER_RE	0xFC	DMA 管理寄存器

11.3.2 系统控制寄存器

11.3.2.1 UTIMER_CFG

地址：0x4001_18F0

复位值：0x0

表 11-5 UTIMER 配置寄存器 UTIMER_CFG

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

位置	位名称	说明
[31:10]		未使用
[9]	ENC1_EN	1：启动编码器 1，0：停止编码器 1
[8]	ENC0_EN	1：启动编码器 0，0：停止编码器 0
[7]	TIMER3_EN	Timer3 使能，当 TIMER3_EN 为 0 时，Timer3 停止计数
[6]	TIMER2_EN	Timer2 使能，当 TIMER2_EN 为 0 时，Timer2 停止计数
[5]	TIMER1_EN	Timer1 使能，当 TIMER1_EN 为 0 时，Timer1 停止计数
[4]	TIMER0_EN	Timer0 使能，当 TIMER0_EN 为 0 时，Timer0 停止计数
[3:0]		系统保留，推荐写入 0

当 Timer 停止计数时，Timer 的寄存器配置不受影响，仍然保持；且在停止计数时仍可以通过软件修改 Timer 的寄存器配置。

11.3.3 滤波控制寄存器

11.3.3.1 UTIMER_FLT_TH01

地址：0x4001_18A0

复位值：0x0

表 11-6 滤波控制寄存器 UTIMER_FLT_TH01

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

位置	位名称	说明
[31:16]		未使用
[15:12]	T1_CH1_FLT	TIM1_CH1 信号滤波宽度选择，取值范围 0~15。 [15:12]为 0 时，对 TIM1_CH1 不进行滤波。 [15:12]不为 0 时，对 TIM1_CH1 信号进行滤波：滤波宽度为 T1_CH1_FLT×8。当 TIM1_CH1 电平稳定超过 T1_CH1_FLT×8 个系统时钟周期宽度时，滤波器输出更新到 TIM1_CH1 信号值；否则，滤波器保

		持当前的输出不变。
[11:8]	T1_CH0_FLT	TIM1_CH0 信号滤波宽度选择。含义同 T1_CH1_FLT。
[7:4]	T0_CH1_FLT	TIM0_CH1 信号滤波宽度选择。含义同 T1_CH1_FLT。
[3:0]	T0_CH0_FLT	TIM0_CH0 信号滤波宽度选择。含义同 T1_CH1_FLT。

11.3.3.2 UTIMER_FLT_TH23

地址：0x4001_18A4

复位值：0x0

表 11-7 滤波控制寄存器 UTIMER_FLT_TH23

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

位置	位名称	说明
[31:16]		未使用
[15:12]	T3_CH1_FLT	TIM3_CH1 信号滤波宽度选择，取值范围 0~15。 [15:12]为 0 时，对 TIM3_CH1 不进行滤波。 [15:12]不为 0 时，对 TIM3_CH1 信号进行滤波：滤波宽度为 T3_CH1_FLT×8。当 TIM3_CH1 电平稳定超过 T3_CH1_FLT×8 个系统时钟周期宽度时，滤波器输出更新到 TIM3_CH1 信号值；否则，滤波器保持当前的输出不变。
[11:8]	T3_CH0_FLT	TIM3_CH0 信号滤波宽度选择。含义同 T3_CH1_FLT。
[7:4]	T2_CH1_FLT	TIM2_CH1 信号滤波宽度选择。含义同 T3_CH1_FLT。
[3:0]	T2_CH0_FLT	TIM2_CH0 信号滤波宽度选择。含义同 T3_CH1_FLT。

11.3.4 定时器寄存器

Timer0/1 完全相同，此处仅说明 Timer0 寄存器。

Timer2/3 完全相同，与 Timer0/1 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。

Encoder0 复用了 Timer2 的输入端口，Encoder1 复用了 Timer3 的输入端口；开启 Encoder 功能时，并不影响对应 Timer 的正常使用。

11.3.4.1 UTIMER_UNT0_CFG Timer0 配置寄存器

地址：0x4001_1800

复位值：0x0

表 11-8 Timer 0 配置寄存器 UTIMER_UNT0_CFG

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

位置	位名称	说明
[31:12]		未使用
[11]	ETON	Timer0 计数器计数使能来源。 0，自动运行，持续计数； 1，等待外部事件触发计数一个 Timer 周期后停止。 外部事件通过 UTIMER_UNT0_EVT 寄存器进行配置。
[10]		系统保留，恒 0。
[9:8]	CLK_DIV	Timer0 计数器频率配置，计数器计数频率是系统主时钟频率的 1/2/4/8 分频： 00：1 分频， 01：2 分频，10：4 分频，11：8 分频
[7]	CH1_POL	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。
[6]	CH1_MODE	通道 1 的工作模式： 0，比较模式，输出方波，在通道 1 计数器计数值等于 0 或等于比较捕获寄存器值时发生翻转。 1，捕获模式，当通道 1 输入信号发生捕获事件时，将计数器计数值存入通道 1 比较捕获寄存器。
[5]	CH1_FE_CAP_EN	通道 1 下降沿捕获事件使能。1：使能；0：关闭。 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。
[4]	CH1_RE_CAP_EN	通道 1 上升沿捕获事件使能。1：使能；0：关闭。 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。
[3]	CH0_POL	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	通道 0 的工作模式， 0，比较模式，输出方波，在通道 0 计数器计数值等于 0 或等于比较捕获寄存器值时发生翻转。 1，捕获模式，当通道 0 输入信号发生捕获事件时，将计数器计数值存入通道 0 比较捕获寄存器。
[1]	CH0_FE_CAP_EN	通道 0 下降沿捕获事件使能。1：使能；0：关闭。 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	通道 0 上升沿捕获事件使能。1：使能；0：关闭。

		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。

11.3.4.2 UTIMER_UNT1_CFG Timer1 配置寄存器

地址：0x4001_1820

复位值：0x0

表 11-9 Timer 0 配置寄存器 UTIMER_UNT0_CFG

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

位置	位名称	说明
[31:12]		未使用
[11]	ETON	Timer1 计数器计数使能来源。 0，自动运行，持续计数； 1，等待外部事件触发计数一个 Timer 周期后停止。 外部事件通过 UTIMER_UNT1_EVT 寄存器进行配置。
[10]		系统保留，恒 0。
[9:8]	CLK_DIV	Timer1 计数器频率配置，计数器计数频率是系统主频率的 1/2/4/8 分频： 00：1 分频， 01：2 分频，10：4 分频，11：8 分频
[7]	CH1_POL	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。
[6]	CH1_MODE	通道 1 的工作模式： 0，比较模式，输出方波，在通道 1 计数器计数值等于 0 或等于比较捕获寄存器值时发生翻转。 1，捕获模式，当通道 1 输入信号发生捕获事件时，将计数器计数值存入通道 1 比较捕获寄存器。
[5]	CH1_FE_CAP_EN	通道 1 下降沿捕获事件使能。1：使能；0：关闭。 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。
[4]	CH1_RE_CAP_EN	通道 1 上升沿捕获事件使能。1：使能；0：关闭。 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。
[3]	CH0_POL	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	通道 0 的工作模式， 0，比较模式，输出方波，在通道 0 计数器计数值等于 0 或等于比较捕获寄存器值时发生翻转。 1，捕获模式，当通道 0 输入信号发生捕获事件时，将计数器计数值存入通道 0 比较捕获寄存器。
[1]	CH0_FE_CAP_EN	通道 0 下降沿捕获事件使能。1：使能；0：关闭。 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	通道 0 上升沿捕获事件使能。1：使能；0：关闭。 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。

11.3.4.3 UTIMER_UNT2_CFG Timer2 配置寄存器

地址：0x4001_1840

复位值：0x0

表 11-10 Timer 2 配置寄存器 UTIMER_UNT2_CFG

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

位置	位名称	说明
[31:12]		未使用
[11]	ETON	0，自动运行，持续计数； 1，等待外部事件触发计数一个 Timer 周期后停止。 外部事件通过 UTIMER_UNT2_EVT 寄存器进行配置。
[10]		系统保留，恒 0。
[9:8]	CLK_DIV	Timer2 计数器频率配置，计数器计数频率是系统主频率的 1/2/4/8 分频： 00：1 分频， 01：2 分频，10：4 分频，11：8 分频
[7]	CH1_POL	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。
[6]	CH1_MODE	通道 1 的工作模式： 0，比较模式，输出方波，在通道 1 计数器计数值等于 0 或等于比较捕获寄存器值时发生翻转。 1，捕获模式，当通道 1 输入信号发生捕获事件时，将计数器计数值存入通道 1 比较捕获寄存器。

[5]	CH1_FE_CAP_EN	通道 1 下降沿捕获事件使能。1：使能；0：关闭。 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。
[4]	CH1_RE_CAP_EN	通道 1 上升沿捕获事件使能。1：使能；0：关闭。 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。
[3]	CH0_POL	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	通道 0 的工作模式， 0，比较模式，输出方波，在通道 0 计数器计数值等于 0 或等于比较捕获寄存器值时发生翻转。 1，捕获模式，当通道 0 输入信号发生捕获事件时，将计数器计数值存入通道 0 比较捕获寄存器。
[1]	CH0_FE_CAP_EN	通道 0 下降沿捕获事件使能。1：使能；0：关闭。 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	通道 0 上升沿捕获事件使能。1：使能；0：关闭。 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。

11.3.4.4 UTIMER_UNT3_CFG Timer3 配置寄存器

地址：0x4001_1860

复位值：0x0

表 11-11 Timer 3 配置寄存器 UTIMER_UNT3_CFG

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

位置	位名称	说明
[31:12]		未使用
[11]	ETON	0，自动运行，持续计数； 1，等待外部事件触发计数一个 Timer 周期后停止。 外部事件通过 UTIMER_UNT3_EVT 寄存器进行配置。
[10]		系统保留，恒 0。
[9:8]	CLK_DIV	Timer3 计数器频率配置，计数器计数频率是系统主频率的 1/2/4/8 分频：

		00：1 分频， 01：2 分频，10：4 分频，11：8 分频
[7]	CH1_POL	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。
[6]	CH1_MODE	通道 1 的工作模式： 0，比较模式，输出方波，在通道 1 计数器计数值等于 0 或等于比较捕获寄存器值时发生翻转。 1，捕获模式，当通道 1 输入信号发生捕获事件时，将计数器计数值存入通道 1 比较捕获寄存器。
[5]	CH1_FE_CAP_EN	通道 1 下降沿捕获事件使能。1：使能；0：关闭。 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。
[4]	CH1_RE_CAP_EN	通道 1 上升沿捕获事件使能。1：使能；0：关闭。 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。
[3]	CH0_POL	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	通道 0 的工作模式， 0，比较模式，输出方波，在通道 0 计数器计数值等于 0 或等于比较捕获寄存器值时发生翻转。 1，捕获模式，当通道 0 输入信号发生捕获事件时，将计数器计数值存入通道 0 比较捕获寄存器。
[1]	CH0_FE_CAP_EN	通道 0 下降沿捕获事件使能。1：使能；0：关闭。 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	通道 0 上升沿捕获事件使能。1：使能；0：关闭。 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。

在 LKS32MC08x 系列中，Timer0/1 为 16bit 位宽；Timer2/3 为 32bit 位宽，因此 Timer2/3 下列相关寄存器为 32bit 位宽。

11.3.4.5 UTIMER_UNT0_TH Timer 0 门限寄存器

地址：0x4001_1804

复位值：0x0

表 11-12 Timer 0 门限寄存器 UTIMER_UNT0_TH

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNT0_TH															
RW															
0															

位置	位名称	说明

[31:16]		未使用
[15:0]	UNT0_TH	Timer 0 计数器计数门限。计数器从 0 计数到 UTIMER_UNT0_TH 值后再次回 0 开始计数。

11.3.4.6 UTIMER_UNT1_TH Timer 1 门限寄存器

地址：0x4001_1824

复位值：0x0

表 11-13 Timer 1 门限寄存器 UTIMER_UNT1_TH

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNT1_TH															
RW															
0															

位置	位名称	说明
[31:16]		未使用
[15:0]	UNT1_TH	Timer 1 计数器计数门限。计数器从 0 计数到 UTIMER_UNT1_TH 值后再次回 0 开始计数。

11.3.4.7 UTIMER_UNT2_TH Timer 2 门限寄存器

地址：0x4001_1844

复位值：0x0

表 11-14 Timer 2 门限寄存器 UTIMER_UNT2_TH

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

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

位置	位名称	说明
[31:0]	UNT2_TH	Timer 2 计数器计数门限。计数器从 0 计数到 UTIMER_UNT2_TH 值后再次回 0 开始计数。

11.3.4.8 UTIMER_UNT3_TH Timer 3 门限寄存器

地址：0x4001_1864

复位值：0x0

表 11-15 Timer 3 门限寄存器 UTIMER_UNT3_TH

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

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

位置	位名称	说明
[31:0]	UNT3_TH	Timer 3 计数器计数门限。计数器从 0 计数到 UTIMER_UNT3_TH 值后再次回 0 开始计数。

11.3.4.9 UTIMER_UNT0_CNT Timer 0 计数寄存器

地址：0x4001_1808

复位值：0x0

表 11-16 Timer 0 计数寄存器 UTIMER_UNT0_CNT

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

位置	位名称	说明
[31:16]		未使用
[15:0]	UNT0_CNT	Timer 0 计数器当前计数值。写操作可以写入新的计数值。

11.3.4.10 UTIMER_UNT1_CNT Timer 1 计数寄存器

地址：0x4001_1828

复位值：0x0

表 11-17 Timer 1 计数寄存器 UTIMER_UNT1_CNT

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

位置	位名称	说明
[31:16]		未使用
[15:0]	UNT1_CNT	Timer 1 计数器当前计数值。写操作可以写入新的计数值。

11.3.4.11 UTIMER_UNT2_CNT Timer 2 计数寄存器

地址：0x4001_1848

复位值：0x0

表 11-18 Timer 2 计数寄存器 UTIMER_UNT2_CNT

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

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

位置	位名称	说明
[31:0]	UNT2_CNT	Timer 2 计数器当前计数值。写操作可以写入新的计数值。

11.3.4.12 UTIMER_UNT3_CNT Timer 3 计数寄存器

地址：0x4001_1868

复位值：0x0

表 11-19 Timer 3 计数寄存器 UTIMER_UNT3_CNT

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

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

RW
0

位置	位名称	说明
[31:0]	UNT3_CNT	Timer3 计数器当前计数值。写操作可以写入新的计数值。

11.3.4.13 UTIMER_UNT0_CMP0 Timer 0 通道 0 比较捕获寄存器

地址：0x4001_180C

复位值：0x0

表 11-20 Timer 0 通道 0 比较捕获寄存器 UTIMER_UNT0_CMP0

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

位置	位名称	说明
[31:16]		未使用
[15:0]	UNT0_ CMP0	Timer 0 通道 0 工作在比较模式时，当计数器计数值等于 UTIMER_UNT0_CMP0 时，发生比较事件。 Timer 0 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 UTIMER_UNT0_ CMP0 寄存器。

11.3.4.14 UTIMER_UNT0_CMP1 Timer 0 通道 1 比较捕获寄存器

地址：0x4001_1810

复位值：0x0

表 11-21 Timer 0 通道 1 比较捕获寄存器 UTIMER_UNT0_CMP1

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

位置	位名称	说明
[31:16]		未使用
[15:0]	UNT0_CMP1	Timer 0 通道 1 工作在比较模式时，当计数器计数值等于 UTIMER_UNT0_CMP1 时，发生比较事件。

		Timer 0 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 UTIMER_UNT0_CMP1 寄存器。

11.3.4.15 UTIMER_UNT1_CMP0 Timer 1 通道 0 比较捕获寄存器

地址：0x4001_182C

复位值：0x0

表 11-22 Timer 1 通道 0 比较捕获寄存器 UTIMER_UNT1_CMP0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNT1_ CMP0															
RW															
0															

位置	位名称	说明
[31:16]		未使用
[15:0]	UNT1_ CMP0	Timer 1 通道 0 工作在比较模式时，当计数器计数值等于 UTIMER_UNT1_CMP0 时，发生比较事件。 Timer 1 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 UTIMER_UNT1_ CMP0 寄存器。

11.3.4.16 UTIMER_UNT1_CMP1 Timer 1 通道 1 比较捕获寄存器

地址：0x4001_1830

复位值：0x0

表 11-23 Timer 1 通道 1 比较捕获寄存器 UTIMER_UNT1_CMP1

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

位置	位名称	说明
[31:16]		未使用
[15:0]	UNT1_CMP1	Timer 1 通道 1 工作在比较模式时，当计数器计数值等于 UTIMER_UNT1_CMP1 时，发生比较事件。 Timer 1 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 UTIMER_UNT1_CMP1 寄存器。

11.3.4.17 UTIMER_UNT2_CMP0 Timer 2 通道 0 比较捕获寄存器

地址：0x4001_184C

复位值：0x0

表 11-24 Timer 2 通道 0 比较捕获寄存器 UTIMER_UNT2_CMP0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UNT2_ CMP0															
RW															
0															

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

位置	位名称	说明
[31:0]	UNT2_ CMP0	Timer 2 通道 0 工作在比较模式时，当计数器计数值等于 UTIMER_UNT2_CMP0 时，发生比较事件。 Timer 2 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 UTIMER_UNT2_ CMP0 寄存器。

11.3.4.18 UTIMER_UNT2_CMP1 Timer 2 通道 1 比较捕获寄存器

地址：0x4001_1850

复位值：0x0

表 11-25 Timer 2 通道 1 比较捕获寄存器 UTIMER_UNT2_CMP1

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

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

位置	位名称	说明
[31:0]	UNT2_CMP1	Timer 2 通道 1 工作在比较模式时，当计数器计数值等于 UTIMER_UNT2_CMP1 时，发生比较事件。 Timer 2 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 UTIMER_UNT2_CMP1 寄存器。

11.3.4.19 UTIMER_UNT3_CMP0 Timer 3 通道 0 比较捕获寄存器

地址：0x4001_186C

复位值：0x0

表 11-26 Timer 3 通道 0 比较捕获寄存器 UTIMER_UNT3_CMP0

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

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

位置	位名称	说明
[31:0]	UNT3_ CMP0	Timer 3 通道 0 工作在比较模式时，当计数器计数值等于 UTIMER_UNT3_CMP0 时，发生比较事件。 Timer 3 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 UTIMER_UNT3_ CMP0 寄存器。

11.3.4.20 UTIMER_UNT3_CMP1 Timer 3 通道 1 比较捕获寄存器

地址：0x4001_1870

复位值：0x0

表 11-27 Timer 3 通道 1 比较捕获寄存器 UTIMER_UNT3_CMP1

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

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

位置	位名称	说明
[31:0]	UNT3_CMP1	Timer 3 通道 1 工作在比较模式时，当计数器计数值等于 UTIMER_UNT3_CMP1 时，发生比较事件。

		Timer 3 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 UTIMER_UNT3_CMP1 寄存器。

11.3.4.21 UTIMER_UNT0_EVT Timer0 外部事件选择寄存器

地址：0x4001_1814

复位值：0x0

表 11-28 Timer 0 外部事件选择寄存器 UTIMER_UNT0_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	Timer0 外部事件选择寄存器，本寄存器需要配合 UTIMER_UNT0_CFG[11]使用。U0CFG[11]为高时，根据本寄存器选择触发 Timer0 计数的事件。 0：MCPWM TADC[2]比较事件 1：MCPWM TADC[3]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

11.3.4.22 UTIMER_UNT1_EVT Timer1 外部事件选择寄存器

地址：0x4001_1834

复位值：0x0

表 11-29 Timer 1 外部事件选择寄存器 UTIMER_UNT1_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	Timer1 外部事件选择寄存器，本寄存器需要配合UTIMER_UNT1_CFG[11]使用。UTIMER_UNT1_CFG[11]为高时，根据本寄存器选择触发 Timer1 计数的事件。 0：TIMER0 通道 0 比较事件 1：TIMER0 通道 1 比较事件 2：MCPWM TADC[2]比较事件 3：MCPWM TADC[3]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

11.3.4.23 UTIMER_UNT2_EVT Timer2 外部事件选择寄存器

地址：0x4001_1854

复位值：0x0

表 11-30 Timer 2 外部事件选择寄存器 UTIMER_UNT2_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	Timer2 外部事件选择寄存器，本寄存器需要配合UTIMER_UNT2_CFG[11]使用。UTIMER_UNT2_CFG[11]为高时，根据本寄存器选择触发 Timer2 计数的事件。 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：MCPWM TADC[2]比较事件 5：MCPWM TADC[3]比较事件 6：TIMER3 通道 0 比较事件 7：TIMER3 通道 1 比较事件

11.3.4.24 UTIMER_UNT3_EVT Timer3 外部事件选择寄存器

地址：0x4001_1874

复位值：0x0

表 11-31 Timer 3 外部事件选择寄存器 UTIMER_UNT3_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	Timer3 外部事件选择寄存器，本寄存器需要配合 UTIMER_UNT3_CFG[11]使用。UTIMER_UNT3_CFG[11]为高时，根据本寄存器选择触发 Timer3 计数的事件。 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：MCPWM TADC[2]比较事件 7：MCPWM TADC[3]比较事件

11.3.5 编码器寄存器

Encoder0/1 功能完全相同仅寄存器地址分配不同。

11.3.5.1 UTIMER_ECDx_CFG EncoderX 配置寄存器

UTIMER_ECD0_CFG 地址：0x4001_1880

UTIMER_ECD1_CFG 地址：0x4001_1890

复位值：0x0

表 11-32 EncoderX 配置寄存器 UTIMER_ECDx_CFG

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

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	EncoderX 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
[7:0]		未使用

11.3.5.2 UTIMER_ECDx_TH　EncoderX 计数门限寄存器

UTIMER_ECD0_TH 地址：0x4001_1884

UTIMER_ECD1_TH 地址：0x4001_1894

复位值：0x0

表 11-33 EncoderX 计数门限寄存器 UTIMER_ECDx_TH

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

位置	位名称	说明
[31:16]		未使用
[15:0]	ECDx_TH	EncoderX 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。

11.3.5.3 UTIMER_ECDx_CNT　EncoderX 计数值寄存器

UTIMER_ECD0_CNT 地址：0x4001_1888

UTIMER_ECD1_CNT 地址：0x4001_1898

复位值：0x0

表 11-34 Encoder0 计数值寄存器 UTIMER_ECD0_CNT

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

位置	位名称	说明
[31:16]		未使用

[15:0]	ECDx_CNT	EncoderX 计数值。

11.3.6 中断管理寄存器

中断管理寄存器包括中断标志寄存器 UTIMER_IF 和中断使能寄存器 UTIMER_IE。两个寄存器各个比特对应相同的中断。

11.3.6.1 UTIMER_IE 中断使能寄存器

地址：0x4001_18F4

复位值：0x0

表 11-35 中断使能寄存器 UTIMER_IE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENC1_OF_IE	ENC1_UF_IE	ENC0_OF_IE	ENC0_UF_IE	T3_CH1_IE	T3_CH0_IE	T3_ZC_IE	T2_CH1_IE	T2_CH0_IE	T2_ZC_IE	T1_CH1_IE	T1_CH0_IE	T1_ZC_IE	T0_CH1_IE	T0_CH0_IE	T0_ZC_IE
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:16]		未使用
[15]	ENC1_OF_IE	Encoder1 上溢出中断使能，高电平有效。当 Encoder1 计数器计数达到计数门限时，上计数事件触发上溢出中断。
[14]	ENC1_UF_IE	Encoder1 下溢出中断使能，高电平有效。当 Encoder1 计数器计数达到 0 时，下计数事件触发下溢出中断。
[13]	ENC0_OF_IE	Encoder0 上溢出中断使能，高电平有效。
[12]	ENC0_UF_IE	Encoder0 下溢出中断使能，高电平有效。
[11]	T3_CH1_IE	Timer3 通道 1 比较/捕获中断使能，高电平有效。
[10]	T3_CH0_IE	Timer3 通道 0 比较/捕获中断使能，高电平有效。
[9]	T3_ZC_IE	Timer3 计数器过 0 中断使能，高电平有效。
[8]	T2_CH1_IE	Timer2 通道 1 比较/捕获中断使能，高电平有效。
[7]	T2_CH0_IE	Timer2 通道 0 比较/捕获中断使能，高电平有效。
[6]	T2_ZC_IE	Timer2 计数器过 0 中断使能，高电平有效。
[5]	T1_CH1_IE	Timer1 通道 1 比较/捕获中断使能，高电平有效。
[4]	T1_CH0_IE	Timer1 通道 0 比较/捕获中断使能，高电平有效。
[3]	T1_ZC_IE	Timer1 计数器过 0 中断使能，高电平有效。
[2]	T0_CH1_IE	Timer0 通道 1 比较/捕获中断使能，高电平有效。
[1]	T0_CH0_IE	Timer0 通道 0 比较/捕获中断使能，高电平有效。
[0]	T0_ZC_IE	Timer0 计数器过 0 中断使能，高电平有效。

11.3.6.2 UTIMER_IF 中断标志寄存器

地址：0x4001_18F8

复位值：0x0

表 11-36 中断标志寄存器 UTIMER_IF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENC1_OF_IF	ENC1_UF_IF	ENC0_OF_IF	ENC0_UF_IF	T3_CH1_IF	T3_CH0_IF	T3_ZC_IF	T2_CH1_IF	T2_CH0_IF	T2_ZC_IF	T1_CH1_IF	T1_CH0_IF	T1_ZC_IF	T0_CH1_IF	T0_CH0_IF	T0_ZC_IF
RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:16]		未使用
[15]	ENC1_OF_IF	Encoder1 上溢出中断标志，高电平有效，对此 bit 写 1 可清 0 此 bit。当 Encoder1 计数器计数达到计数门限时，上计数事件触发上溢出中断。
[14]	ENC1_UF_IF	Encoder1 下溢出中断标志。高电平有效，对此 bit 写 1 可清 0 此 bit。当 Encoder1 计数器计数达到 0 时，下计数事件触发下溢出中断。
[13]	ENC0_OF_IF	Encoder0 上溢出中断标志。高电平有效，对此 bit 写 1 可清 0 此 bit。当 Encoder0 计数器计数达到计数门限时，上计数事件触发上溢出中断。
[12]	ENC0_UF_IF	Encoder0 下溢出中断标志。高电平有效，对此 bit 写 1 可清 0 此 bit。当 Encoder0 计数器计数达到 0 时，下计数事件触发下溢出中断。
[11]	T3_CH1_IF	Timer3 通道 1 比较/捕获中断标志。高电平有效，对此 bit 写 1 可清 0 此 bit。
[10]	T3_CH0_IF	Timer3 通道 0 比较/捕获中断标志。高电平有效，对此 bit 写 1 可清 0 此 bit。
[9]	T3_ZC_IF	Timer3 计数器过 0 中断标志。高电平有效，对此 bit 写 1 可清 0 此 bit。
[8]	T2_CH1_IF	Timer2 通道 1 比较/捕获中断标志。高电平有效，对此 bit 写 1 可清 0 此 bit。
[7]	T2_CH0_IF	Timer2 通道 0 比较/捕获中断标志。高电平有效，对此 bit 写 1 可清 0 此 bit。
[6]	T2_ZC_IF	Timer2 计数器过 0 中断标志。高电平有效，对此 bit 写 1 可清 0 此 bit。
[5]	T1_CH1_IF	Timer1 通道 1 比较/捕获中断标志。高电平有效，对此 bit 写 1 可清 0 此 bit。
[4]	T1_CH0_IF	Timer1 通道 0 比较/捕获中断标志。高电平有效，对此 bit 写 1 可清 0 此 bit。
[3]	T1_ZC_IF	Timer1 计数器过 0 中断标志。高电平有效，对此 bit 写 1 可清 0 此 bit。
[2]	T0_CH1_IF	Timer0 通道 1 比较/捕获中断标志。高电平有效，对此 bit 写 1 可清 0

		此 bit。
[1]	T0_CH0_IF	Timer0 通道 0 比较/捕获中断标志。高电平有效，对此 bit 写 1 可清 0 此 bit。
[0]	T0_ZC_IF	Timer0 计数器过 0 中断标志。高电平有效，对此 bit 写 1 可清 0 此 bit。

其中 Timer2 通道 0/1、Timer3 通道 0/1 的比较事件可作为 ADC 采样触发事件 UTimer_T0/ UTimer_T1/ UTimer_T2/ UTimer_T3；

此 4 个事件与 MCPWM 产生的 MCPWM_T0/ MCPWM_T1/ MCPWM_T2/ MCPWM_T3 在 ADC 内部经过使能控制后分别相或得到 4 个 ADC 采样触发事件 TADC[0]/TADC[1]/TADC[2]/TADC[3]。

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 T0_ZC_IF，但如果同时 T0_CH0_IF 在写入执行前置 1 了，则软件先读取回 UTIMER_IF 值为 0x2，然后执行或操作 0x2|0x1=0x3，然后写入，同时对 T0_CH0_IF 和 T0_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|=0x1;

如果希望清零 T0_ZC_IF 标志位，应以如下方式，直接对 BIT0 写 1.

UTIMER_IF=0x1;

11.3.7 DMA 管理寄存器

中断管理寄存器包括中断标志寄存器 IF 和中断使能寄存器 IE。两个寄存器各个比特对应相同的中断。

11.3.7.1 UTIMER_RE DMA 请求使能寄存器

地址：0x4001_18FC

复位值：0x0

表 11-37 DMA 请求使能寄存器 RE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				T3_CH1_DMA_RE	T3_CH0_DMA_RE	T3_DMA_RE	T2_CH1_DMA_RE	T2_CH0_DMA_RE	T2_DMA_RE	T1_CH1_DMA_RE	T1_CH0_DMA_RE	T1_DMA_RE	T0_CH1_DMA_RE	T0_CH0_DMA_RE	T0_DMA_RE
				RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
				0	0	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:12]		未使用

[11]	T3_CH1_DMA_RE	Timer3 通道 1 比较/捕获 DMA 请求使能，高电平有效。
[10]	T3_CH0_DMA_RE	Timer3 通道 0 比较/捕获 DMA 请求使能，高电平有效。
[9]	T3_DMA_RE	Timer3 计数器过 0 DMA 请求使能，高电平有效。
[8]	T2_CH1_DMA_RE	Timer2 通道 1 比较/捕获 DMA 请求使能，高电平有效。
[7]	T2_CH0_DMA_RE	Timer2 通道 0 比较/捕获 DMA 请求使能，高电平有效。
[6]	T2_DMA_RE	Timer2 计数器过 0 DMA 请求使能，高电平有效。
[5]	T1_CH1_DMA_RE	Timer1 通道 1 比较/捕获 DMA 请求使能，高电平有效。
[4]	T1_CH0_DMA_RE	Timer1 通道 0 比较/捕获 DMA 请求使能，高电平有效。
[3]	T1_DMA_RE	Timer1 计数器过 0 DMA 请求使能，高电平有效。
[2]	T0_CH1_DMA_RE	Timer0 通道 1 比较/捕获 DMA 请求使能，高电平有效。
[1]	T0_CH0_DMA_RE	Timer0 通道 0 比较/捕获 DMA 请求使能，高电平有效。
[0]	T0_DMA_RE	Timer0 计数器过 0 DMA 请求使能，高电平有效。

12 HALL 信号处理模块

12.1 综述

芯片共支持 3 路 HALL 信号输入。

对于输入的 HALL 传感器信号，所进行的处理包括：

滤波，消除 HALL 信号毛刺的影响

捕获，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

12.2 实现说明

12.2.1 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。

12.2.2 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。

12.2.3 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：

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。具体如下图所示：

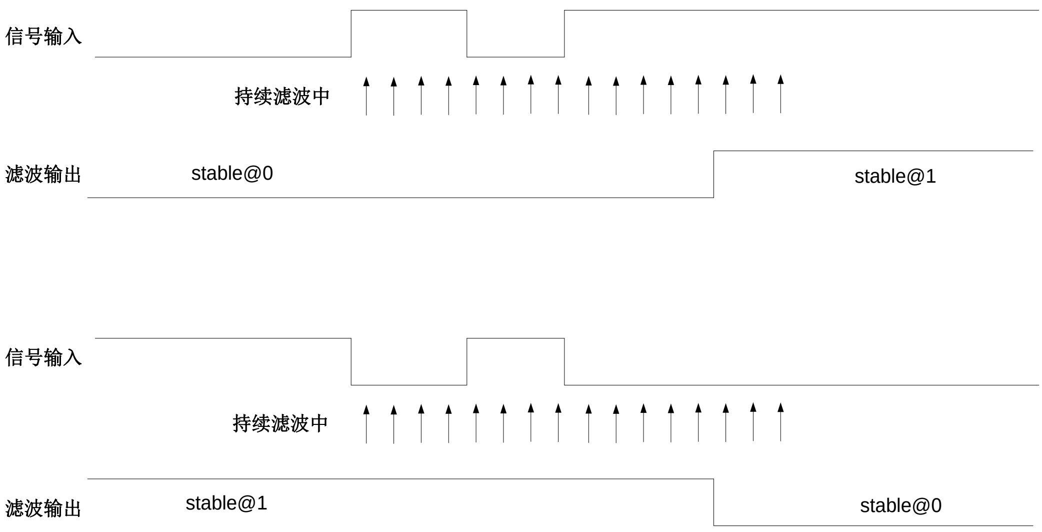


图 12-1 7/5 滤波模块框图

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_75 可以选择是否使能第一级滤波器。

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 96MHz 工作频率下，周期 T_{clk} 是 10.4ns，寄存器配置最大为 32767，最长滤波宽度为约 10.4ns×32768≈340us。

通过访问 HALL_INFO.FIL_DATA[2:0]可以捕捉滤波后的 HALL 信号；HALL_INFO.RAW_DATA[2:0]则是滤波前原始 HALL 输入信号，详见 12.3.3。

12.2.4 捕获

捕获模块用于测量两次 HALL 信号变化之间的时间，其核心为一个 24 位计数器，在 96MHz 工作频率下，如果 Hall 时钟进行 8 分频，最大可以记录约 2^24*8/96e6=1.40s 的时间宽度，达到 10.42ns 的时间分辨率。

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

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

12.2.5 中断

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

12.2.6 数据流程

HALL 模块的数据流程如下图所示，FCLK[1]为受 SYS_CLK_FEN 门控控制的系统主时钟，通常为 96MHz 的 PLL 时钟。

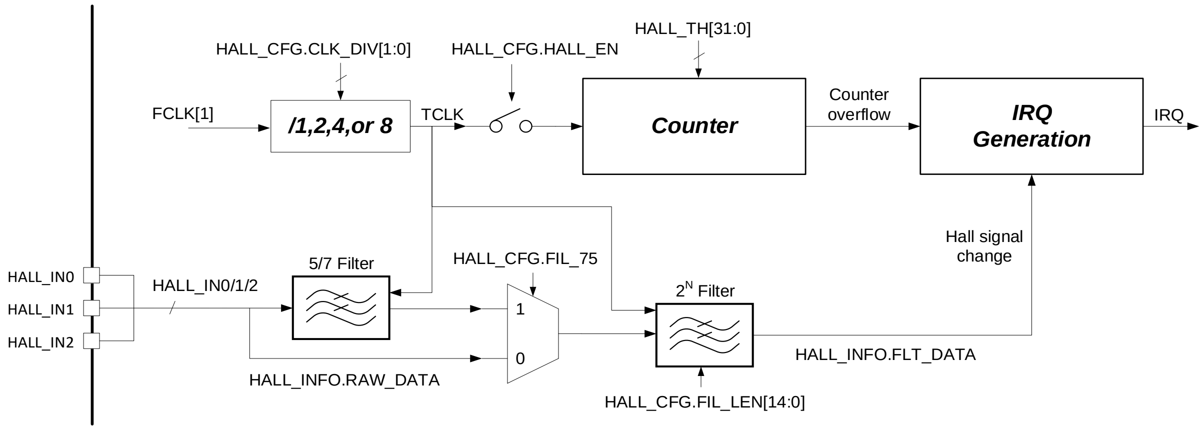


图 12-2 数据流程框图

12.3 寄存器

12.3.1 地址分配

HALL 模块寄存器的基地址是 0x4001_1000，寄存器列表如下：

表 12-1HALL 模块寄存器地址分配

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

12.3.2 HALL_CFG HALL 模块配置寄存器

地址：0x4001_1000

复位值：0x0

表 12-2 HALL 模块配置寄存器 HALL_CFG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	SW_IE	OV_IE	CHG_IE				HALL_EN				FIL_75			CLK_DIV	
	RW	RW	RW				RW				RW			RW	
	0	0	0				0				0			0	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

	FIL_LEN_
	RW
	0

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。 1，使能 0，关闭
[28]	CHG_IE	HALL 信号变化中断使能开关。默认关闭。 1，使能 0，关闭
[27:25]		未使用
[24]	HALL_EN	HALL 模块使能开关。默认关闭。 1，使能 0，关闭
[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。 1，使能 0，关闭
[19:18]		未使用
[17:16]	CLK_DIV	HALL 时钟分频系数 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。

12.3.3 HALL_INFO HALL 模块信息寄存器

地址：0x4001_1004

复位值：0x0

表 12-3 HALL 模块信息寄存器 HALL_INFO

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
													SW_IF	OV_IF	CHG_IF
													WO	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。当 HALL_CFG.SW_IE=1 时，向 SW_IF 写 1 可以触发 CHG_IF 置位。需要使能 HALL_CFG.CHG_IE，才能产生 HALL 中断。
[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

12.3.4 HALL_WIDTH HALL 宽度计数值寄存器

地址：0x4001_1008

复位值：0x0

表 12-4 HALL 宽度计数值寄存器 HALL_WIDTH

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

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

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

12.3.5 HALL_TH HALL 模块计数器门限值寄存器

地址：0x4001_100C

复位值：0x0

表 12-5 HALL 模块计数器门限值寄存器 HALL_TH

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

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

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

12.3.6 HALL_CNT HALL 计数寄存器

地址：0x4001_1010

复位值：0x0

表 12-6 HALL 计数寄存器 HALL_CNT

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

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

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

13 MCPWM

13.1 概述

MCPWM 模块，是一个精确控制电机驱动波形输出的模块。

包含一个 16 位递增计数器，用于提供一个基础周期。计数器的时钟分频有 1/2/4/8 四种选项，产生的的分频时钟频率分别为 96MHz、48MHz、24MHz 和 12MHz。

包含四组 PWM 生成模块。

- 可以产生 4 对（互补信号）不交叠的或 8 路边沿对齐的（边沿模式）PWM 信号；
- 支持边沿对齐 PWM
- 中心对齐 PWM
- 移相 PWM

同时可以产生 4 路与 MCPWM 同时基的定时信息，用于触发 ADC 模块同步采样，进行与 MCPWM 的联动。

包含一组急停保护模块，用于不依赖 CPU 软件的处理快速关断 MCPWM 模块输出。MCPWM 模块可输入 4 路急停信号，其中两路来自芯片 IO，两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）。

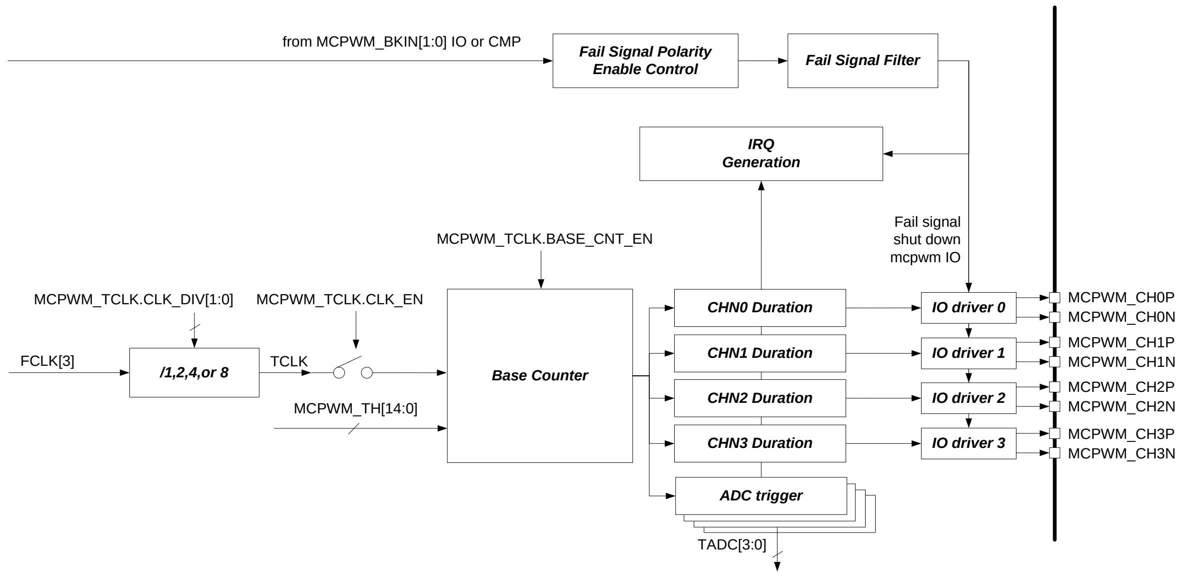


图 13-1 MCPWM 模块框图

为了保证定时精度，通常采用 96MHz 的时钟作为 MCPWM 模块工作频率。

13.1.1 Base Counter 模块

该模块主要是由一个递增计数器组成，其计数门限值为 MCPWM_TH，计数器从 t0 时刻开始从-TH 递增计数，在 t1 时刻过 0 点，在 t2 时刻计数到 TH 完成一次计数循环，回到-TH，重新开始计数。计数周期为(TH×2+1)×计数时钟周期。

在 t0/t1(本次 t0 即上一次 t2)可产生定时事件中断，MCPWM_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。

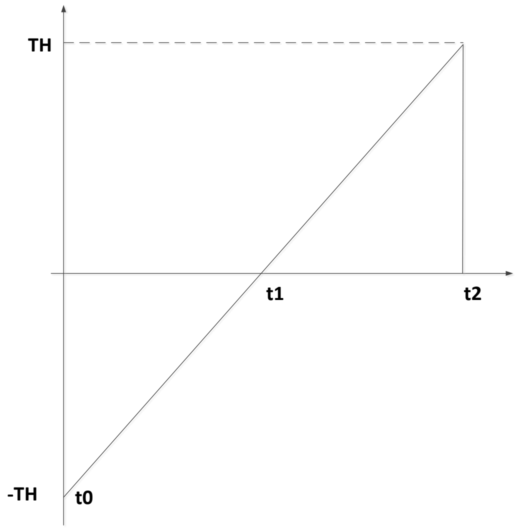


图 13-2 Base Counter t0/t1 时序

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 以及 PWM 周期 MCPWM_TH 配置好。在实际运行过程中，也可动态改变比较门限值和 PWM 周期寄存器，可通过写 MCPWM_UPDATE 寄存器实现手动更新，也可以通过配置 MCPWM_SDCFG.T1_UPDATE_EN 及 MCPWM_SDCFG.T0_UPDATE_EN 进行硬件自动更新。硬件更新，仅在 t0 及 t1 时刻（可配置 t0 或 t1 更新和 t0 t1 时刻都更新）才能产生更新事件，硬件把加载寄存器的值载入到实际运行的寄存器中。而更新事件的发生频率可以配置，即每间隔 N 个 t0 及 t1 时刻才发生更新。无论是否发生更新，t0/t1 时刻均可产生相应的中断。若硬件把加载寄存器的值到载入实际运行的寄存器后，产生装载中断。

通过配置 MCPWM_SDCFG 寄存器选择更新发生在 t0 或者 t1 或者二者皆可，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。

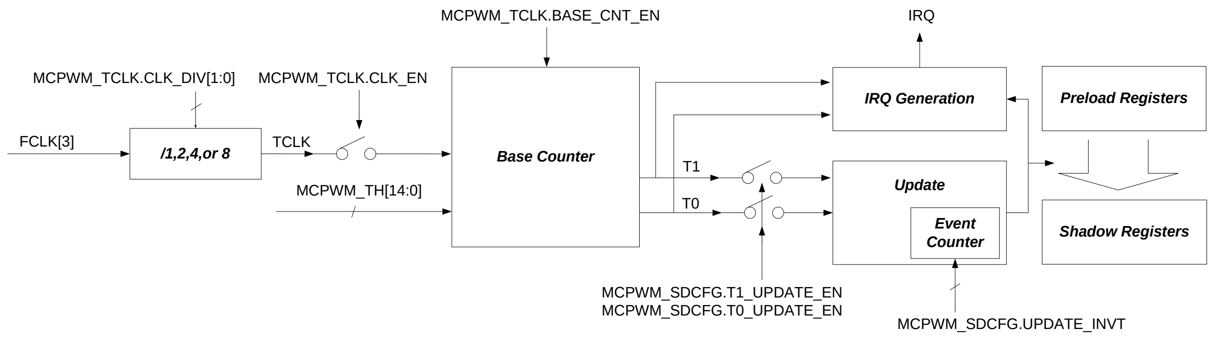


图 13-3 MCPWM 更新机制

13.1.2 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毁。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。有两路 fail 信号输入 MCPWM，即 FAIL0 和 FAIL1，分别可以来自芯片 IO MCPWM_BKIN[1:0]或芯片内部比较器的输出 CMP[1:0]。

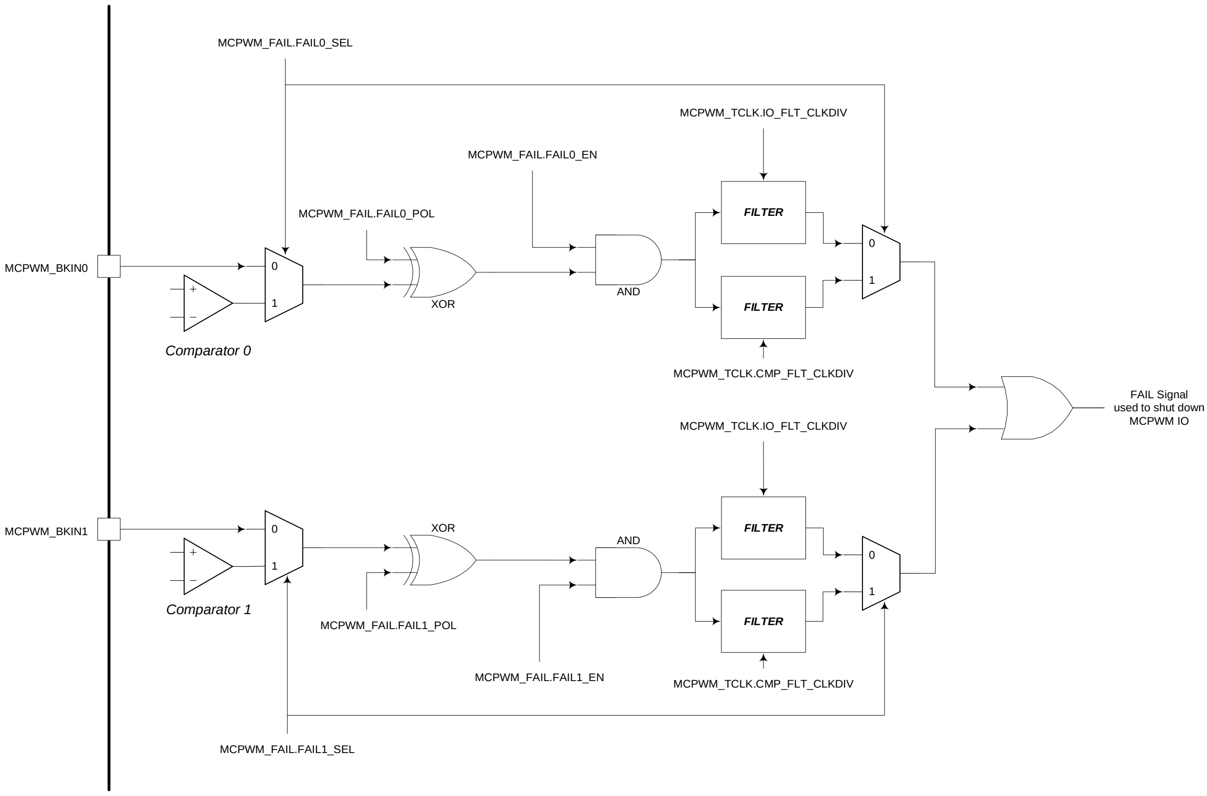


图 13-4 MCPWM FAIL 逻辑示意图

Filter 滤波模块的时钟，来自系统主时钟 MCLK 的门控时钟 FCLK[3]，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 13-5 所示。

MCPWM 模块使用分频后的时钟进行 Fail 信号滤波，滤波宽度固定为 16 个周期，即输入信号必须保持至少 16 个时钟（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 MCLK/FCLK[3] 的时钟周期，96MHz 对应 10.4ns。MCPWM_TCLK.FLT_CLKDIV 根据配置情况可能是 MCPWM_TCLK.IO_FLT_CLKDIV 或 MCPWM_TCLK.CMP_FLT_CLKDIV。

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

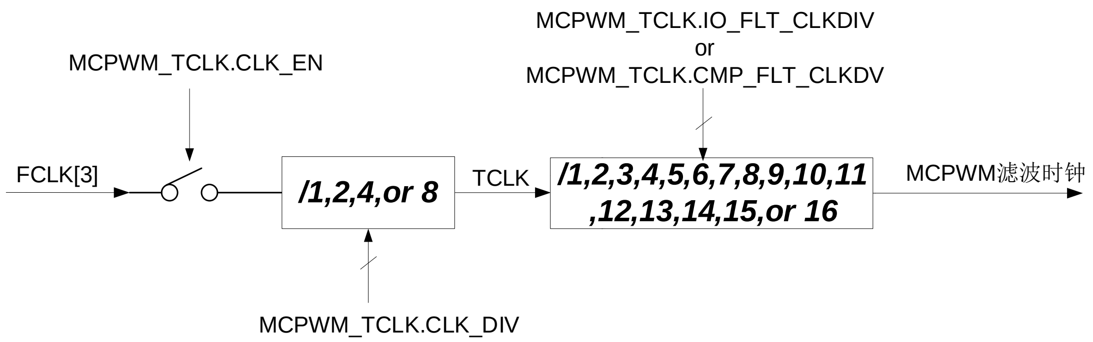


图 13-5 MCPWM Fail 信号滤波时钟生成逻辑

一旦发生 Fail 事件，即

图 13-4 最终输出 Fail 信号，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。

来自比较器的 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。

13.1.3 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

13.1.4 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：

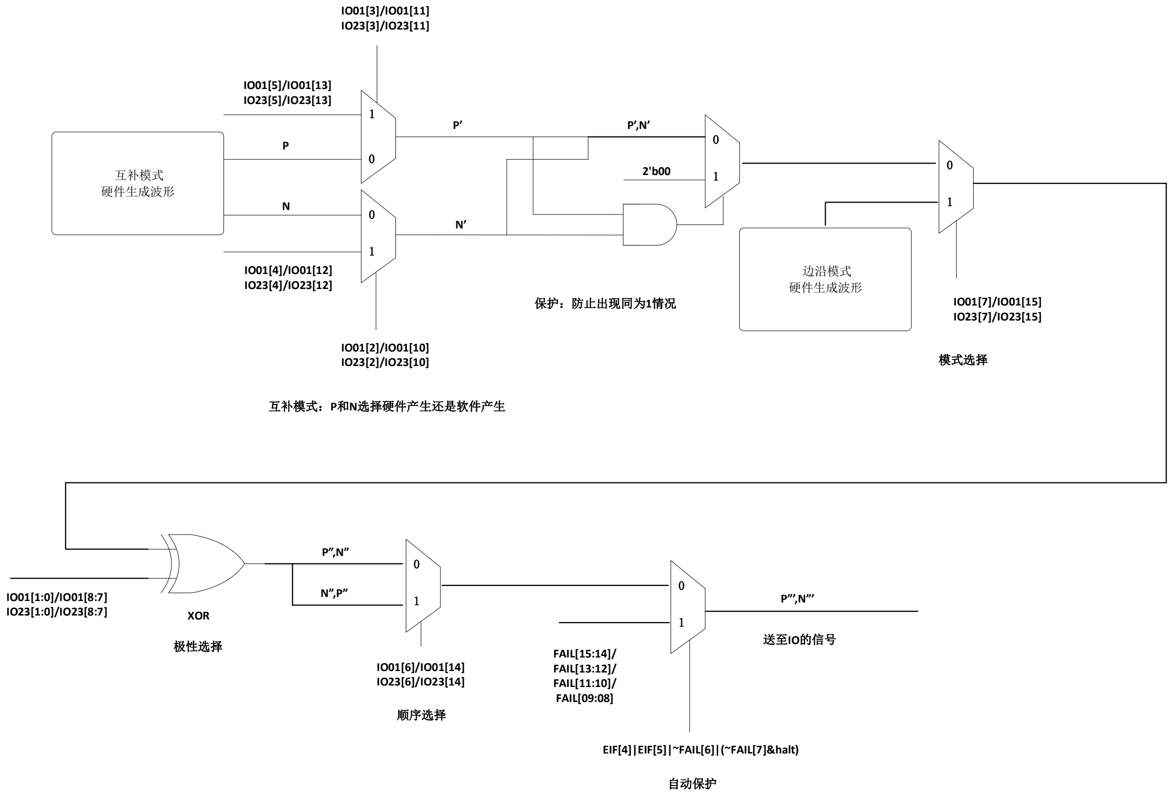


图 13-6 IO Driver 模块数据流程图

13.1.4.1 MCPWM 波形输出-中心对齐模式

4 个 MCPWM IO Driver 采用独立的控制门限，独立死区宽度（每一对互补 IO 的死区需要独立配置，即 4 个死区配置寄存器），共享数据更新事件。

采用 TH<n>0 和 TH<n>1 控制第<n>个 MCPWM IO 的启动、关闭动作，n 为 1/2/3/4。

当计数器 CNT 值向上计数达到 TH<n>0 时，在 t3 时刻关闭 CH<n>N，经过死区延时 DTH0，打开 CH<n>P。

当计数器 CNT 值向上计数达到 TH<n>1 时，在 t4 时刻关闭 CH<n>P，经过死区延时 DTH1，打开 CH<n>N。

采用独立的启动和关闭时间控制，可以提供相位控制的能力。

死区延时保证 CH<n>P/CH<n>N 不会同时为高，避免短路发生。

t3/t4 时刻均会产生相应中断。

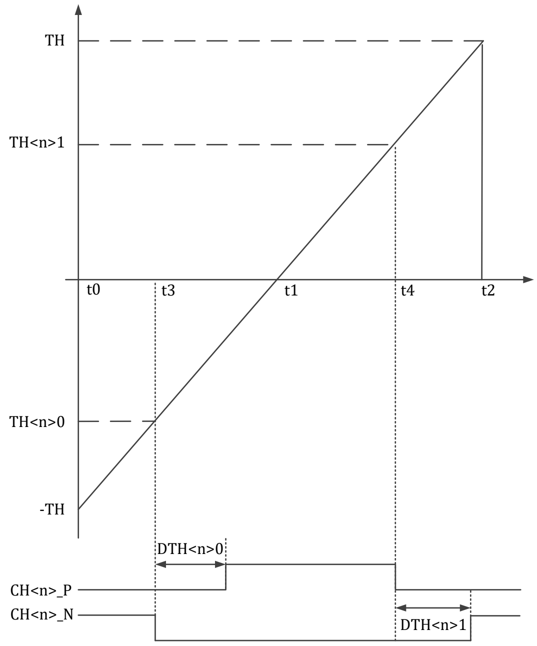


图 13-7 MCPWM 时序 TH<n>0 和 TH<n>1-互补模式

13.1.4.2 MCPWM 波形控制-中心对齐推挽模式

互补推挽模式。第一个周期内，在 t3 时刻 CH<n>P 置 1，在 t4 时刻，CH<n>P 变低。第二个周期内，在 t3 时刻 CH<n>N 置 1，在 t4 时刻，CH<n>N 变低。

t3/t4 均会产生相应中断。

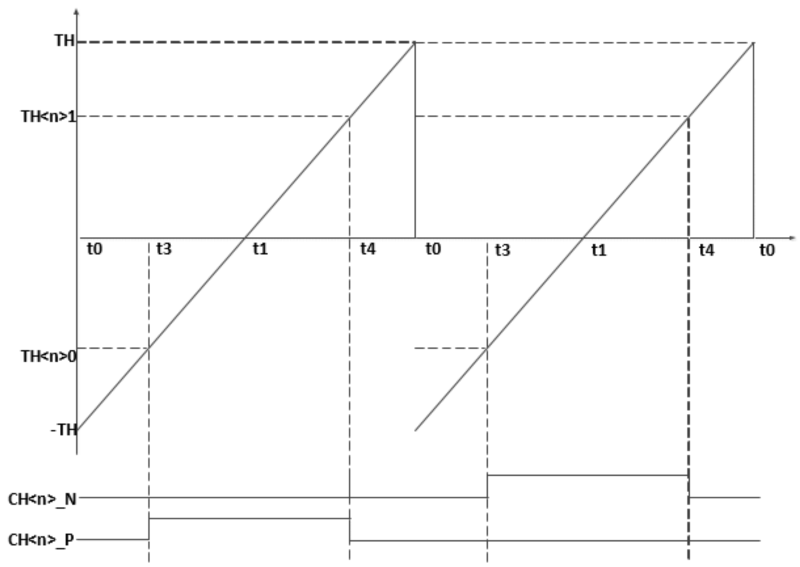


图 13-8 MCPWM 时序 TH<n>0 和 TH<n>1-互补模式

13.1.4.3 MCPWM 波形输出-边沿对齐模式

边沿对齐模式中，在 t0 时刻 CH<n>P/CH<n>N 同时置 0，在 t3 时刻，CH<n>N 变高，在 t4 时刻，CH<n>P 变高。

t3/t4 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。

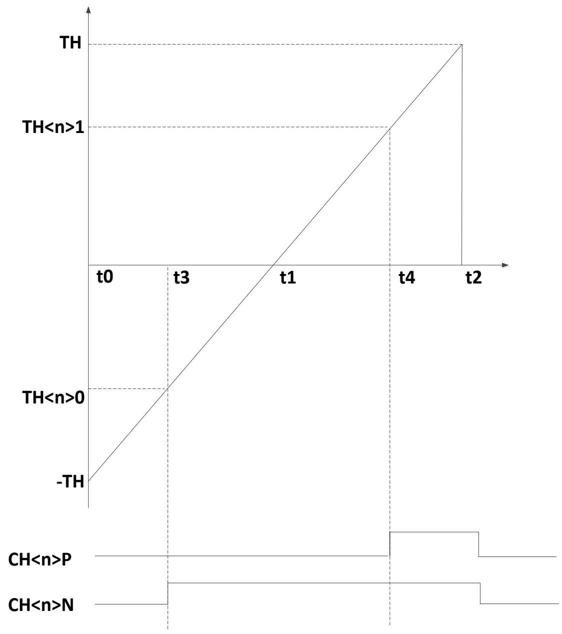


图 13-9 MCPWM 时序边沿对齐模式

13.1.4.4 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 置 1，在 t3 时刻，CH<n>P 变低。第二个周期内，在 t0 时刻 CH<n>N 置 1，在 t3 时刻，CH<n>N 变低。

t0/t3 均会产生相应中断。

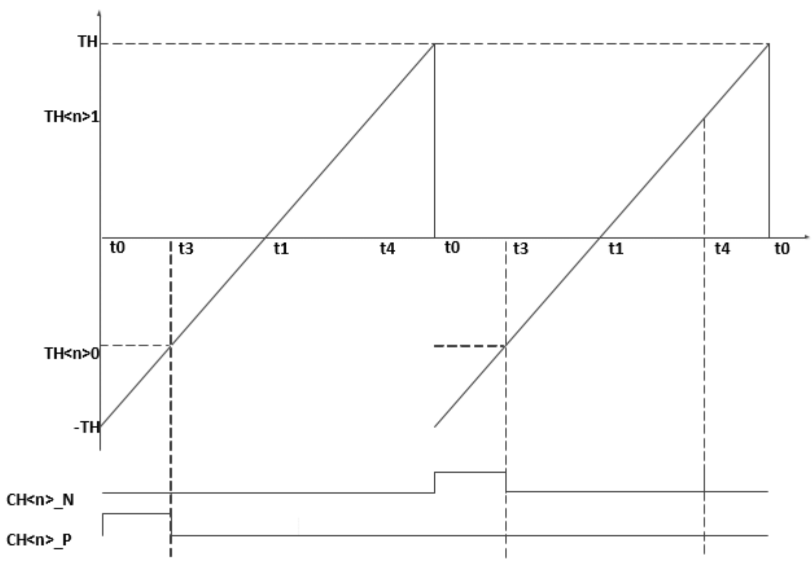


图 13-10 MCPWM 时序 TH<n>0 和 TH<n>1 边沿对齐推挽模式

13.1.4.5 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>_为低时，Vout 输出不确定。

必须避免 CH<n>P/CH<n>N 同时为高的情况，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（中断保持，直到 CPU 写 0）。

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 PWM 模式为互补，仍然由硬件保证不同时为高或者为低。

CH<n>P/CH<n>N，在 IO 上可以互换。

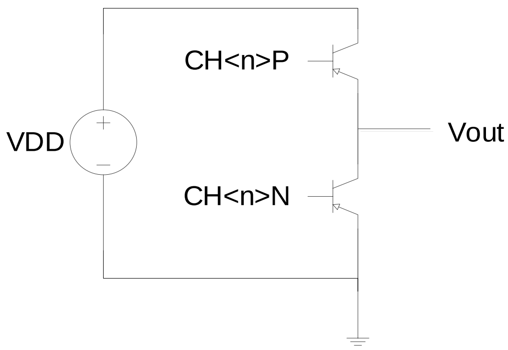


图 13-11 MCPWM IO 控制示意图

13.1.4.6 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。

13.1.4.7 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）。

- 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 模块控制。

- 当发生 Fail 短路状况时，硬件立即切换到 IO 默认输出电平。
- 当芯片调试中，CPU Halt 时，PWM 停止输出，输出 FAIL[15:8]的值。

13.1.5 ADC Trigger Timer 模块

MCPWM 可以提供 ADC 采样控制。当计数器计数到 MCPWM_TMR0/ MCPWM_TMR1/ MCPWM_TMR2/ MCPWM_TMR3 时，可产生定时事件，触发 ADC 采样。该触发信号可同时输出到 GPIO，便于调试之用。输出的具体 GPIO，参见对应器件的 datasheet。

13.1.6 MCPWM 主要事件列表

MCPWM 模块，计数器阈值同对应事件如下表所述。

表 13-1 MCPWM 计数器阈值与事件对应表

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

13.2 寄存器

13.2.1 地址分配

MCPWM 模块寄存器的基地址是 0x4001_1C00，寄存器列表如下：

表 13-2 MCPWM 模块寄存器列表

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

MCPWM_TH21	0x14	MCPWM CH2_N 比较门限值寄存器
MCPWM_TH30	0x18	MCPWM CH3_P 比较门限值寄存器
MCPWM_TH31	0x1C	MCPWM CH3_N 比较门限值寄存器
MCPWM_TMR0	0x20	ADC 采样定时器比较门限 0 寄存器
MCPWM_TMR1	0x24	ADC 采样定时器比较门限 1 寄存器
MCPWM_TMR2	0x28	ADC 采样定时器比较门限 2 寄存器
MCPWM_TMR3	0x2C	ADC 采样定时器比较门限 3 寄存器
MCPWM_TH	0x30	MCPWM 门限值寄存器
MCPWM_UPDATE	0x34	MCPWM 加载控制寄存器
MCPWM_IE	0x38	MCPWM 中断控制寄存器
MCPWM_IF	0x3C	MCPWM 中断标志位寄存器
MCPWM_EIE	0x40	MCPWM 异常中断控制寄存器
MCPWM_EIF	0x44	MCPWM 异常中断标志位寄存器
MCPWM_RE	0x48	MCPWM DMA 请求使能寄存器
MCPWM_PP	0x4C	MCPWM 推挽模式使能寄存器
MCPWM_IO01	0x50	MCPWM IO01 控制寄存器
MCPWM_IO23	0x54	MCPWM IO23 控制寄存器
MCPWM_SDCFG	0x58	MCPWM 加载配置寄存器
MCPWM_TCLK	0x60	MCPWM 时钟分频控制寄存器
MCPWM_FAIL	0x64	MCPWM 短路控制寄存器
MCPWM_PRT	0x74	MCPWM 保护寄存器
MCPWM_CNT	0x78	MCPWM 计数器寄存器
MCPWM_DTH00	0x80	MCPWM CH0 N 通道死区宽度控制寄存器
MCPWM_DTH01	0x84	MCPWM CH0 P 通道死区宽度控制寄存器
MCPWM_DTH10	0x88	MCPWM CH1 N 通道死区宽度控制寄存器
MCPWM_DTH11	0x8C	MCPWM CH1 P 通道死区宽度控制寄存器
MCPWM_DTH20	0x90	MCPWM CH2 N 通道死区宽度控制寄存器
MCPWM_DTH21	0x94	MCPWM CH2 P 通道死区宽度控制寄存器
MCPWM_DTH30	0x98	MCPWM CH3 N 通道死区宽度控制寄存器
MCPWM_DTH31	0x9C	MCPWM CH3 P 通道死区宽度控制寄存器

表 13-3 受 MCPWM_PRT 保护的寄存器

名称	偏移地址	说明
MCPWM_TH	0x30	MCPWM 门限值寄存器
MCPWM_IE	0x38	MCPWM 中断控制寄存器
MCPWM_EIE	0x40	MCPWM 异常中断控制寄存器
MCPWM_RE	0x48	MCPWM DMA 请求使能寄存器
MCPWM_PP	0x4C	MCPWM 推挽模式使能寄存器
MCPWM_IO01	0x50	MCPWM IO01 控制寄存器
MCPWM_IO23	0x54	MCPWM IO23 控制寄存器
MCPWM_SDCFG	0x58	MCPWM 加载配置寄存器
MCPWM_TCLK	0x60	MCPWM 时钟分频控制寄存器

MCPWM_FAIL	0x64	MCPWM 短路控制寄存器
MCPWM_DTH00	0x80	MCPWM CH0 N 通道死区宽度控制寄存器
MCPWM_DTH01	0x84	MCPWM CH0 P 通道死区宽度控制寄存器
MCPWM_DTH10	0x88	MCPWM CH1 N 通道死区宽度控制寄存器
MCPWM_DTH11	0x8C	MCPWM CH1 P 通道死区宽度控制寄存器
MCPWM_DTH20	0x90	MCPWM CH2 N 通道死区宽度控制寄存器
MCPWM_DTH21	0x94	MCPWM CH2 P 通道死区宽度控制寄存器
MCPWM_DTH30	0x98	MCPWM CH3 N 通道死区宽度控制寄存器
MCPWM_DTH31	0x9C	MCPWM CH3 P 通道死区宽度控制寄存器

表 13-4 存在影子寄存器的寄存器

名称	偏移地址	说明
MCPWM_TH00	0x00	MCPWM CH0_P 比较门限值寄存器
MCPWM_TH01	0x04	MCPWM CH0_N 比较门限值寄存器
MCPWM_TH10	0x08	MCPWM CH1_P 比较门限值寄存器
MCPWM_TH11	0x0C	MCPWM CH1_N 比较门限值寄存器
MCPWM_TH20	0x10	MCPWM CH2_P 比较门限值寄存器
MCPWM_TH21	0x14	MCPWM CH2_N 比较门限值寄存器
MCPWM_TH30	0x18	MCPWM CH3_P 比较门限值寄存器
MCPWM_TH31	0x1C	MCPWM CH3_N 比较门限值寄存器
MCPWM_TMR0	0x20	ADC 采样定时器比较门限 0 寄存器
MCPWM_TMR1	0x24	ADC 采样定时器比较门限 1 寄存器
MCPWM_TMR2	0x28	ADC 采样定时器比较门限 2 寄存器
MCPWM_TMR3	0x2C	ADC 采样定时器比较门限 3 寄存器
MCPWM_TH	0x30	MCPWM 门限值寄存器

13.2.2 MCPWM_TH00

无写保护的寄存器

地址：0x4001_1C00

复位值：0x0

表 13-5 MCPWM_TH00 配置寄存器

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

位置	位名称	说明
[31:16]		未使用

[15:0]	TH00	MCPWM CH0_P 比较门限值，16 位有符号数；发生更新事件后，本寄存器加载到 MCPWM 实际运行系统中。

13.2.3 MCPWM_TH01

无写保护的寄存器

地址：0x4001_1C04

复位值：0x0

表 13-6 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；发生更新事件后，本寄存器加载到 MCPWM 实际运行系统中。

13.2.4 MCPWM_TH10

无写保护的寄存器

地址：0x4001_1C08

复位值：0x0

表 13-7 MCPWM_TH10 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	TH10	MCPWM CH1_P 比较门限值，16 位有符号数；发生更新事件后，本寄存器加载到 MCPWM 实际运行系统中。

13.2.5 MCPWM_TH11

无写保护的寄存器

地址：0x4001_1C0C

复位值：0x0

表 13-8 MCPWM_TH11 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	TH11	MCPWM CH1_N 比较门限值，16 位有符号数；发生更新事件后，本寄存器加载到 MCPWM 实际运行系统中。

13.2.6 MCPWM_TH20

无写保护的寄存器

地址：0x4001_1C10

复位值：0x0

表 13-9 MCPWM_TH20 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	TH20	MCPWM CH2_P 比较门限值，16 位有符号数；发生更新事件后，本寄存器加载到 MCPWM 实际运行系统中。

13.2.7 MCPWM_TH21

无写保护的寄存器

地址：0x4001_1C14

复位值：0x0

表 13-10 MCPWM_TH21 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	TH21	MCPWM CH2_N 比较门限值，16 位有符号数；发生更新事件后，本寄存器加载到 MCPWM 实际运行系统中。

13.2.8 MCPWM_TH30

无写保护的寄存器

地址：0x4001_1C18

复位值：0x0

表 13-11 MCPWM_TH30 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	TH30	MCPWM CH3_P 比较门限值，16 位有符号数；发生更新事件后，本寄存器加载到 MCPWM 实际运行系统中。

13.2.9 MCPWM_TH31

无写保护的寄存器

地址：0x4001_1C1C

复位值：0x0

表 13-12 MCPWM_TH31 配置寄存器

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

位置	位名称	说明
[31:16]		未使用
[15:0]	TH31	MCPWM CH3_N 比较门限值，16 位有符号数；发生更新事件后，本寄存器加载到 MCPWM 实际运行系统中。

13.2.10 MCPWM_TMR0

无写保护的寄存器

地址：0x4001_1C20

复位值：0x0

表 13-13 MCPWM_TMR0 配置寄存器

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

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；发生更新事件后，本寄存器加载到 MCPWM 实际运行系统中。

13.2.11 MCPWM_TMR1

无写保护的寄存器

地址：0x4001_1C24

复位值：0x0

表 13-14 MCPWM_TMR1 配置寄存器

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

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；发生更新事件后，本寄存器加载到 MCPWM 实际运行系统中。

13.2.12 MCPWM_TMR2

无写保护的寄存器

地址：0x4001_1C28

复位值：0x0

表 13-15 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															
0															

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。

13.2.13 MCPWM_TMR3

无写保护的寄存器

地址：0x4001_1C2C

复位值：0x0

表 13-16 MCPWM_TMR3 配置寄存器

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

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。。

13.2.14 MCPWM_TH

写保护的寄存器

地址：0x4001_1C30

复位值：0x0

表 13-17 MCPWM_TH 配置寄存器

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 计数器门限值，15 位无符号数，MCPWM 实际运行系统中的计数器从-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。。

13.2.15 MCPWM_UPDATE

无写保护的寄存器

地址：0x4001_1C34

复位值：0x0

表 13-18 MCPWM_UPDATE 配置寄存器

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

位置	位名称	说明
[31:13]		未使用
[12]	TH_UPDATE	手动将加载 MCPWM_TH 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。

13.2.16 MCPWM_IE

写保护的寄存器

地址：0x4001_1C38

复位值：0x0

表 13-19 MCPWM_IE 配置寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SHADE_IE	TMR3_IE	TMR2_IE	TMR1_IE	TMR0_IE	TH31_IE	TH30_IE	TH21_IE	TH20_IE	TH11_IE	TH10_IE	TH01_IE	TH00_IE	T1_IE	T0_IE
	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:15]		未使用
[14]	SHADE_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。

13.2.17 MCPWM_IF

无写保护的寄存器

地址：0x4001_1C3C

复位值：0x0

表 13-20 MCPWM_IF 配置寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SHADE_IF	TMR3_IF	TMR2_IF	TMR1_IF	TMR0_IF	TH31_IF	TH30_IF	TH21_IF	TH20_IF	TH11_IF	TH10_IF	TH01_IF	TH00_IF	T1_IF	T0_IF
	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:15]		未使用
[14]	SHADE_IF	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事件。

		1，发生；0，没发生。写 1 清零。
[13]	TMR3_IF	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TMR3 中断源事件。 1，发生；0，没发生。写 1 清零。
[12]	TMR2_IF	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TMR2 中断源事件。 1，发生；0，没发生。写 1 清零。
[11]	TMR1_IF	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TMR1 中断源事件。 1，发生；0，没发生。写 1 清零。
[10]	TMR0_IF	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TMR0 中断源事件。 1，发生；0，没发生。写 1 清零。
[9]	TH31_IF	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH31 中断源事件。 1，发生；0，没发生。写 1 清零。
[8]	TH30_IF	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH30 中断源事件。 1，发生；0，没发生。写 1 清零。
[7]	TH21_IF	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH21 中断源事件。 1，发生；0，没发生。写 1 清零。
[6]	TH20_IF	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH20 中断源事件。 1，发生；0，没发生。写 1 清零。
[5]	TH11_IF	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH11 中断源事件。 1，发生；0，没发生。写 1 清零。
[4]	TH10_IF	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH10 中断源事件。 1，发生；0，没发生。写 1 清零。
[3]	TH01_IF	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH01 中断源事件。 1，发生；0，没发生。写 1 清零。
[2]	TH00_IF	MCPWM 实际运行系统中计数器的计数值等于 MCPWM_TH00 中断源事件。 1，发生；0，没发生。写 1 清零。
[1]	T1_IF	t1 事件，计数器的计数值到达 0 中断源事件。 1，发生；0，没发生。写 1 清零。
[0]	T0_IF	t0 事件，计数器的计数值回到 MCPWM_TH 中断源事件。 1，发生；0，没发生。写 1 清零。

13.2.18 MCPWM_EIE

写保护的寄存器

地址：0x4001_1C40

复位值：0x0

表 13-21 MCPWM_EIE 配置寄存器

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

	0	0	0	0	0	0

位置	位名称	说明
[31:6]		未使用
[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]	CH3_SHORT_IE	MCPWM CH3_P 和 CH3_N 同时有效，中断源使能。1，使能；0，关闭。
[2]	CH2_SHORT_IE	MCPWM CH2_P 和 CH2_N 同时有效，中断源使能。1，使能；0，关闭。
[1]	CH1_SHORT_IE	MCPWM CH1_P 和 CH1_N 同时有效，中断源使能。1，使能；0，关闭。
[0]	CH0_SHORT_IE	MCPWM CH0_P 和 CH0_N 同时有效，中断源使能。1，使能；0，关闭。

13.2.19 MCPWM_EIF

无写保护的寄存器

地址：0x4001_1C44

复位值：0x0

表 13-22 MCPWM_EIF 配置寄存器

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

位置	位名称	说明
[31:6]		未使用
[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]	CH3_SHORT_IF	MCPWM CH3_P 和 CH3_N 同时有效，中断源事件。 1，发生；0，没发生。写 1 清零。
[2]	CH2_SHORT_IF	MCPWM CH2_P 和 CH2_N 同时有效，中断源事件。 1，发生；0，没发生。写 1 清零。
[1]	CH1_SHORT_IF	MCPWM CH1_P 和 CH1_N 同时有效，中断源事件。 1，发生；0，没发生。写 1 清零。
[0]	CH0_SHORT_IF	MCPWM CH0_P 和 CH0_N 同时有效，中断源事件。 1，发生；0，没发生。写 1 清零。

13.2.20 MCPWM_RE

写保护的寄存器

地址：0x4001_1C48

复位值：0x0

表 13-23 MCPWM_RE 配置寄存器

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

位置	位名称	说明
[31:6]		未使用
[5]	ZC_RE	DMA 请求使能信号。过零事件请求。写 1，使能；写 0，关闭。
[4]	STOV_RE	DMA 请求使能信号。起点事件请求。写 1，使能；写 0，关闭。
[3]	TMR3_RE	DMA 请求使能信号。ADC 通道 3 请求。写 1，使能；写 0，关闭。
[2]	TMR2_RE	DMA 请求使能信号。ADC 通道 2 请求。写 1，使能；写 0，关闭。
[1]	TMR1_RE	DMA 请求使能信号。ADC 通道 1 请求。写 1，使能；写 0，关闭。
[0]	TMR0_RE	DMA 请求使能信号。ADC 通道 0 请求。写 1，使能；写 0，关闭。

13.2.21 MCPWM_PP

写保护的寄存器

地址：0x4001_1C4C

复位值：0x0

表 13-24 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。

13.2.22 MCPWM_IO01

写保护的寄存器

地址：0x4001_1C50

复位值：0x0

表 13-25 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：Edge 模式；0：互补模式。
[14]	CH1_PN_SW	CH1 的 P 和 N 通道输出互换选择。即 P 通道信号最后从 N 通道输出，N 通道的信号最后从 P 通道输出。1：互换；0：不互换。
[13]	CH1_SCTRLP	当 CH1_PS=1 时，输出到 CH1 P 通道的值。
[12]	CH1_SCTRLN	当 CH1_NS=1 时，输出到 CH1 N 通道的值。
[11]	CH1_PS	CH1 P 来源。1：来自 CH1_SCTRLP；0：MCPWM 内部计数器产生。
[10]	CH1_NS	CH1 N 来源。1：来自 CH1_SCTRLN；0：MCPWM 内部计数器产生。
[9]	CH1_PP	CH1 P 极性选择。1：CH1 P 信号取反输出；0：CH1 P 信号正常输出。
[8]	CH1_NP	CH1 N 极性选择。1：CH1 N 信号取反输出；0：CH1 N 信号正常输出。
[7]	CH0_WM	CH0 工作模式选择。1：Edge 模式；0：互补模式。
[6]	CH0_PN_SW	CH0 的 P 和 N 通道输出互换选择。即 P 通道信号最后从 N 通道输出，N 通道的信号最后从 P 通道输出。1：互换；0：不互换。
[5]	CH0_SCTRLP	当 CH0_PS =1 时，输出到 CH0 P 通道的值。
[4]	CH0_SCTRLN	当 CH0_NS =1 时，输出到 CH0 N 通道的值。
[3]	CH0_PS	CH0 P 来源。1：来自 CH0_SCTRLP；0：MCPWM 实际运行系统中计数器产生。
[2]	CH0_NS	CH0 N 来源。1：来自 CH0_SCTRLN；0：MCPWM 实际运行系统中计数器产生。
[1]	CH0_PP	CH0 P 极性选择。1：CH0 P 信号取反输出；0：CH0 P 信号正常输出。
[0]	CH0_NP	CH0 N 极性选择。1：CH0 N 信号取反输出；0：CH0 N 信号正常输出。**极性选择跟随通道交换，例如 CH0 N 选择取反输出，同时选择了通道交换，则交换后的 CH0 N 仍是取反输出。**

13.2.23 MCPWM_IO23

写保护的寄存器

地址：0x4001_1C54

复位值：0x0

表 13-26 MCPWM_IO23 配置寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH3_WM	CH3_PN_SW	CH3_SCTRLP	CH3_SCTRLN	CH3_PS	CH3_NS	CH3_PP	CH3_NP	CH2_WM	CH2_PN_SW	CH2_SCTRLP	CH2_SCTRLN	CH2_PS	CH2_NS	CH2_PP	CH2_NP
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位置	位名称	说明
[31:16]		未使用
[15]	CH3_WM	CH3 工作模式选择。1：Edge 模式；0：互补模式。
[14]	CH3_PN_SW	CH3 的 P 和 N 通道输出互换选择。即 P 通道信号最后从 N 通道输出，N 通道的信号最后从 P 通道输出。1：互换；0：不互换。
[13]	CH3_SCTRLP	当 CH3_PS =1 时，输出到 CH3 P 通道的值。
[12]	CH3_SCTRLN	当 CH3_NS =1 时，输出到 CH3 N 通道的值。
[11]	CH3_PS	CH3 P 来源。1：来自 CH3_SCTRLP；0：MCPWM 实际运行系统中计数器产生。
[10]	CH3_NS	CH3 N 来源。1：来自 CH3_SCTRLN；0：MCPWM 实际运行系统中计数器产生。
[9]	CH3_PP	CH3 P 极性选择。1：CH3 P 信号取反输出；0：CH3 P 信号正常输出。
[8]	CH3_NP	CH3 N 极性选择。1：CH3 N 信号取反输出；0：CH3 N 信号正常输出。
[7]	CH2_WM	CH2 工作模式选择。1：Edge 模式；0：互补模式。
[6]	CH2_PN_SW	CH2 的 P 和 N 通道输出互换选择。即 P 通道信号最后从 N 通道输出，N 通道的信号最后从 P 通道输出。1：互换；0：不互换。
[5]	CH2_SCTRLP	当 CH2_PS =1 时，输出到 CH2 P 通道的值。
[4]	CH2_SCTRLN	当 CH2_NS =1 时，输出到 CH2 N 通道的值。
[3]	CH2_PS	CH2 P 来源。1：来自 CH2_SCTRLP；0：MCPWM 实际运行系统中计数器产生。
[2]	CH2_NS	CH2 N 来源。1：来自 CH2_SCTRLN；0：MCPWM 实际运行系统中计数器产生。
[1]	CH20_PP	CH2 P 极性选择。1：CH2 P 信号取反输出；0：CH2 P 信号正常输出。
[0]	CH2_NP	CH2 N 极性选择。1：CH2 N 信号取反输出；0：CH2 N 信号正常输出。**极性选择跟随通道交换，例如 CH0 N 选择取反输出，同时选择了通道交**

		换，则交换后的 CH0 N 仍是取反输出。

13.2.24 MCPWM_SDCFG

写保护的寄存器

地址：0x4001_1C58

复位值：0x0

表 13-27 MCPWM_SDCFG 配置寄存器

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

位置	位名称	说明
[31:7]		未使用
[6]	AUTO_ERR_CLR	AUTO_ERR_CLR 更新事件是否自动清除 MCPWM_EIF[5:4]并置位 MOE，恢复 MCPWM 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]	T1_UPDATE_EN	t1（过零）事件更新使能。1：使能；0，关闭。
[4]	T0_UPDATE_EN	t0（起点）事件更新使能。1：使能；0，关闭。
[3:0]	UPDATE_INTV	更新间隔。t0/t1 发生次数同 UPDATE_INTV+1 相等时，，MCPWM 系统自动触发 MCPWM_TH（包括 THxx）和 MCPWM_TMR 寄存器加载到 MCPWM 运行系统的操作。若 B[5]和 B[4]均关闭，将不会触发此类型加载，只能手动触发加载。

13.2.25 MCPWM_TCLK

写保护的寄存器

地址：0x4001_1C60

复位值：0x0

表 13-28 MCPWM_TCLK 配置寄存器

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

位置	位名称	说明

[31:16]		未使用
[15:12]	CMP_FLT_CLKDIV	比较器输出的滤波时钟分频寄存器，基于系统时钟分频，影响 MCPWM_FAIL[1:0]。计算公式如下： 系统时钟 / (B[15:12] + 1)。分频范围是 1-16。
[11:8]	IO_FLT_CLKDIV	GPIO 输入的滤波时钟分频寄存器，基于系统时钟分频，影响 MCPWM_FAIL[1:0]。计算公式如下： 系统时钟 / (B[11:8] + 1)。分频范围是 1-16。
[7:4]		未使用
[3]	BASE_CNT_EN	MCPWM 实际运行计数器使能开关。1：使能；0：关闭。
[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

13.2.26 MCPWM_FAIL

写保护的寄存器

地址：0x4001_1C64

复位值：0x0

表 13-29 MCPWM_FAIL 配置寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH3N_DEFAULT	CH3P_DEFAULT	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	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]	CH3N_DEFAULT	CH3 N 通道默认值
[14]	CH3P_DEFAULT	CH3 P 通道默认值
[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	CPU 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]	MCPWM_OE	MOE 控制 MCPWM CH 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 MCPWM_OE 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 1 的结果；0：来自 GPIO 第 1 路。
[0]	FAIL0_SEL	FAIL0 来源选择。1：比较器 0 的结果；0：来自 GPIO 第 0 路。

MCPWM_FAIL 可以用来设置紧急停车事件，封锁 MCPWM 的信号输出。急停事件主要有两个 FAIL0 和 FAIL1。 共有 4 个信号来源，比较器 0 输出和比较器 1 输出以及 MCPWM_BKIN0 和 MCPWM_BKIN0。 FAIL0 可以来自比较器 0 输出或芯片 IO MCPWM_BKIN0，FAIL1 可以来自比较器 1 输出或芯片 IO MCPWM_BKIN1。

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 0 输出和比较器 1 输出的滤波时钟分频由 MCPWM_TCLK.CMP_FLT_CLKDIV 设置；信号来源 MCPWM_BKIN0 和 MCPWM_BKIN0 的滤波时钟分频由 MCPWM_TCLK.IO_FLT_CLKDIV 设置。

最后 FAIL0,FAIL1 会对信号进行 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 13.1.2 Fail 信号处理。

13.2.27 MCPWM_PRT

无写保护的寄存器

地址：0x4001_1C74

复位值：0x0

表 13-30 MCPWM_PRT 配置寄存器

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。

13.2.28 MCPWM_CNT

无写保护的寄存器

地址：0x4001_1C78

复位值：0x0

表 13-31 MCPWM_CNT 配置寄存器

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	MCPWM 实际运行系统中计数器的值。实际读出的计数范围为 0x8000-TH ~ 0x8000+TH，即读出值与 MCPWM 内部 CNT 计数值存在 0x8000 的偏移

13.2.29 MCPWM_DTH00

写保护的寄存器

地址：0x4001_1C80

复位值：0x0

表 13-32 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

13.2.30 MCPWM_DTH01

写保护的寄存器

地址：0x4001_1C84

复位值：0x0

表 13-33 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

13.2.31 MCPWM_DTH10

写保护的寄存器

地址：0x4001_1C88

复位值：0x0

表 13-34 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

13.2.32 MCPWM_DTH11

写保护的寄存器

地址：0x4001_1C8C

复位值：0x0

表 13-35 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

13.2.33 MCPWM_DTH20

写保护的寄存器

地址：0x4001_1C90

复位值：0x0

表 13-36 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

13.2.34 MCPWM_DTH21

写保护的寄存器

地址：0x4001_1C94

复位值：0x0

表 13-37 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

13.2.35 MCPWM_DTH30

写保护的寄存器

地址：0x4001_1C98

复位值：0x0

表 13-38 MCPWM_DTH30 配置寄存器

15–0
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

13.2.36 MCPWM_DTH31

写保护的寄存器

地址：0x4001_1C9C

复位值：0x0

表 13-39 MCPWM_DTH31 配置寄存器

15–0
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

14 UART

14.1 概述

UART 特征如下:

全双工工作

支持 7/8 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 Multi-drop Slave/Master 模式

14.2 功能说明

14.2.1 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加载，并通过 TX 发送出去。

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。

14.2.2 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。

14.2.3 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*UARTx_DIVH+UARTx_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 14-1 UART 波特率配置示例

UART 波特率	SYS_CLK_DIV2	UART_DIVH	UART_DIVL

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

注意，波特率配置系数仅为示例，可能不唯一。

14.2.4 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。

14.2.5 DMA 配置

UART 模块，支持 DMA 操作。实现 DMA 搬移数据，能极大减轻 CPU 的负担。UART 采用 DMA 操作，有以下注意事项。

接收和发送，不能同时使用 DMA。只能接收，或者只能发送使用 DMA。

发送模式，有两种配置 DMA 的方案。

方案 1：若 UARTx_IE.TX_BUF_EMPTY_RE 配置有效。UARTx 模块将预取第一个字节，准备发送；一旦数据进入发射队列，UARTx_IE.UARTx_BUFF 即为空，硬件自动会请求 DMA 搬移下一个字节直至数据搬移完毕。DMA 搬移完毕后，将产生 DMA 完成中断，但是 UARTx 很可能没有发送完毕最后一个字节，若立即操作 UARTx，将可能会产生异常。建议在 DMA 中断处理程序中，开启 UARTx_IE.TX_DONE_IE 中断，UARTx 将最后一个字节发送完毕，产生发送完成中断，在 UARTx 中断处理函数里面，再关闭 UARTx_IE.TX_DONE_IE。

方案 2：若 UARTx_IE.TX_DONE_RE 配置有效。建议 UART 初始化阶段不要清除 TX_DONE 标志。当前传输的数据长度为 Len，DMA 配置传输的字节数为 Len，开启 DMA 中断，DMA 传输完毕后，UART 也发送完毕，软复位 UARTx 模块并重新初始化 UARTx，可开启下一次 UARTx 的发送。

14.3 寄存器

14.3.1 地址分配

UART0 与 UART1 实现完全相同。

UART0 基地址 0x4001_2800。

UART1 基地址 0x4001_2C00。

表 14-2 UARTx 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_IE	0x18	UART 中断使能寄存器
UARTx_IF	0x1C	UART 中断标志寄存器
UARTx_INV	0x20	UART IO 翻转使能

14.3.2 UARTx_CTRL UARTx 控制寄存器

UART0_CTRL 地址：0x4001_2800

UART1_CTRL 地址：0x4001_2C00

复位值：0x0

表 14-3 UARTx 控制寄存器 UARTx_CTRL

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

位置	位名称	说明
[31:8]		未使用
[6]	MDMASTER_BIT9	Multi-drop Master 模式时，第 9 个数据位值
[5]	MD_EN	使能 Multi-drop，0:禁用, 1:使能
[4]	CK_EN	使能校验，0:禁用, 1:使能

[3]	CK_TYPE	奇偶校验，0:EVEN 1:ODD
[2]	BIT_ORDER	先发送的比特，0:LSB , 1:MSB
[1]	STOP_LEN	停止位长度，0:1bit，1:2bit
[0]	DAT_LEN	数据长度，0:8bit，1:7bit

14.3.3 UARTx_DIVH UARTx 波特率设置高字节寄存器

UART0_DIVH 地址：0x4001_2804

UART1_DIVH 地址：0x4001_2C04

复位值：0x0

表 14-4 UARTx 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+DIVL+256*DIVH)

14.3.4 UARTx_DIVL UARTx 波特率设置低字节寄存器

UART0_DIVL 地址：0x4001_2808

UART1_DIVL 地址：0x4001_2C08

复位值：0x0

表 14-5 UARTx 波特率设置低字节寄存器 UARTx_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* UARTx_DIVH+UARTx_DIVL)

14.3.5 UARTx_BUFF UARTx 收发缓冲寄存器

UART0_BUFF 地址：0x4001_280C

UART1_BUFF 地址：0x4001_2C0C

复位值：0x0

表 14-6 UARTx 收发缓冲寄存器 UARTx_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
[31:8]		未使用
[7:0]	BUFF	写:发送数据缓存读：接收数据寄存器

UART 的 Tx_buffer 和 Rx_buffer 共享地址 0x0C 地址。其中，Tx_buffer 是只写的，Rx_buffer 是只读的。因此读访问 UARTx_BUFF 是访问 UARTx_RX_BUFF，写访问 UARTx_BUFF 是访问 UARTx_TX_BUFF。

14.3.6 UARTx_ADR UARTx 地址匹配寄存器

UART0_ADR 地址：0x4001_2810

UART1_ADR 地址：0x4001_2C10

复位值：0x0

表 14-7 UARTx 地址匹配寄存器 UARTx_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	用作 485 通信时的匹配地址

14.3.7 UARTx_STT UARTx 状态寄存器

UART0_STT 地址：0x4001_2814

UART1_STT 地址：0x4001_2C14

复位值：0x0

表 14-8 UARTx 状态寄存器 UARTx_STT

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

位置	位名称	说明
[31:3]		未使用
[2]	ADR_MATCH	Multi-drop 模式下，地址匹配标志位。 1：匹配；0：未匹配。
[1]	TX_DONE	发送完成（此时发送缓存如不为空，则可以继续发送缓存中的数据） 1：完成；0：未完成。
[0]	TX_BUF_EMPTY	发送缓存状态位。 1：空；0：非空。

14.3.8 UARTx_IE UARTx 中断使能寄存器

UART0_IE 地址：0x4001_2818

UART1_IE 地址：0x4001_2C18

复位值：0x0

表 14-9 UARTx 中断使能寄存器 UARTx_IE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								TX_BUF_EMPTY_RE	RX_DONE_RE	TX_DONE_RE	CK_ERR_IE	STOP_ERR_IE	TX_BUF_EMPTY_IE	RX_DONE_IE	TX_DONE_IE
								RW	RW	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

位置	位名称	说明
[31:8]		未使用
[7]	TX_BUF_EMPTY_RE	发送缓冲区空 DMA 请求使能，默认值为 0。 0：关闭；1：开启。
[6]	RX_DONE_RE	接收完成 DMA 请求使能，默认值为 0。

		0：关闭；1：开启。
[5]	TX_DONE_RE	发送完成 DMA 请求使能，默认值为 0。 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。

14.3.9 UARTx_IF UARTx 中断标志寄存器

UART0_IF 地址：0x4001_281C

UART1_IF 地址：0x4001_2C1C

复位值：0x0

表 14-10 UARTx 中断使能寄存器 UARTx_IF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											CK_ERR_IF	STOP_ERR_IF	TX_BUF_EMPTY_IF	RX_DONE_IF	TX_DONE_IF
											RW1C	RW1C	RW1C	RW1C	RW1C
											0	0	1	0	1

位置	位名称	说明
[31:5]		未使用
[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。

上电后发送缓冲区空标志 TX_BUF_EMPTY_IF 和发送完成中断标志 TX_DONE_IF 默认为 1。

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 UARTx_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。

UARTx_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.

UARTx_IF=0x1;

14.3.10 UARTx_INV UARTx IO 翻转寄存器

UART0_INV 地址：0x4001_2820

UART1_INV 地址：0x4001_2C20

复位值：0x0

表 14-11 UARTx 中断使能寄存器 UARTx_INV

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

位置	位名称	说明
[31:2]		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，是指软件发送 1，硬件发送的即是 1；取反输出极性，是指软件发送 1，硬件发送的是 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，是指硬件接收 1，软件接收的即是 1；取反输入极性，是指硬件接收 1，软件接收到的是 0。

15 DSP

15.1 概述

DSP 模块使用自主设计的 DSP 指令集，可以进行加法、乘累加、移位、饱和等单周期算术指令，以及除法、开方、三角函数等多周期算术运算指令；具备 load/store 等访存指令，无条件跳转以及条件跳转等分支指令，还有中断提起等杂项指令。有断点指令和寄存器赋值等伪指令可以在模拟器上用于调试。

DSP 有两种运行模式，自主运行、被动调用。

所谓自主运行即 DSP 读取 CODE MEM 中的指令和 DATA MEM 中的数据进行 DSP 程序执行，独立于 ARM Cortex M0，此时 DSP_SC.PAUSED=0，即 DSP 处于运行状态；CODE MEM 和 DATA MEM 允许 DSP 访问但不允许 CPU 访问改写。

被动调用是指 DSP 作为一个外设模块被 ARM Cortex M0 所调用，CPU 直接访问 DSP 内部的算术运算资源如除法、开方、三角函数等。此时 DSP_SC.PAUSED=1，即 DSP 不运行 DSP 程序，处于暂停状态，CODE MEM 和 DATA MEM 允许 CPU 进行访问改写，对于不进行 DSP 程序开发的用户，推荐使用此种模式，通过 CPU 运行的软件直接调用 DSP 的算术单元。

DSP 配备独立的程序存储器(CODE MEM)和数据存储器(DATA MEM)。在 DSP 暂停即 DSP_SC.PAUSED=1 时可以通过 CPU 访问这两个独立的存储区域,在 DSP 初始化的过程中需要由 CPU 将 DSP 运行的程序和初始数据分别写入 DSP 的 CODE MEM 和 DATA MEM。DSP 具备提起中断的指令，中断置位后，DSP 同时进入暂停状态，此时允许 CPU 通过总线接口访问 DATA MEM 与 DSP 进行数据交互，包括读取 DSP 运算结果，以及写入 DSP 后续运行所需的数据等。

此外，为充分灵活利用 DSP，在 DSP 暂停时允许 CPU 通过 DSP 寄存器接口直接访问 DSP 除法器、开方器、三角函数等运算模块，即允许 CPU 将 DSP 当做简单的运算协处理模块使用。

15.1.1 功能框图

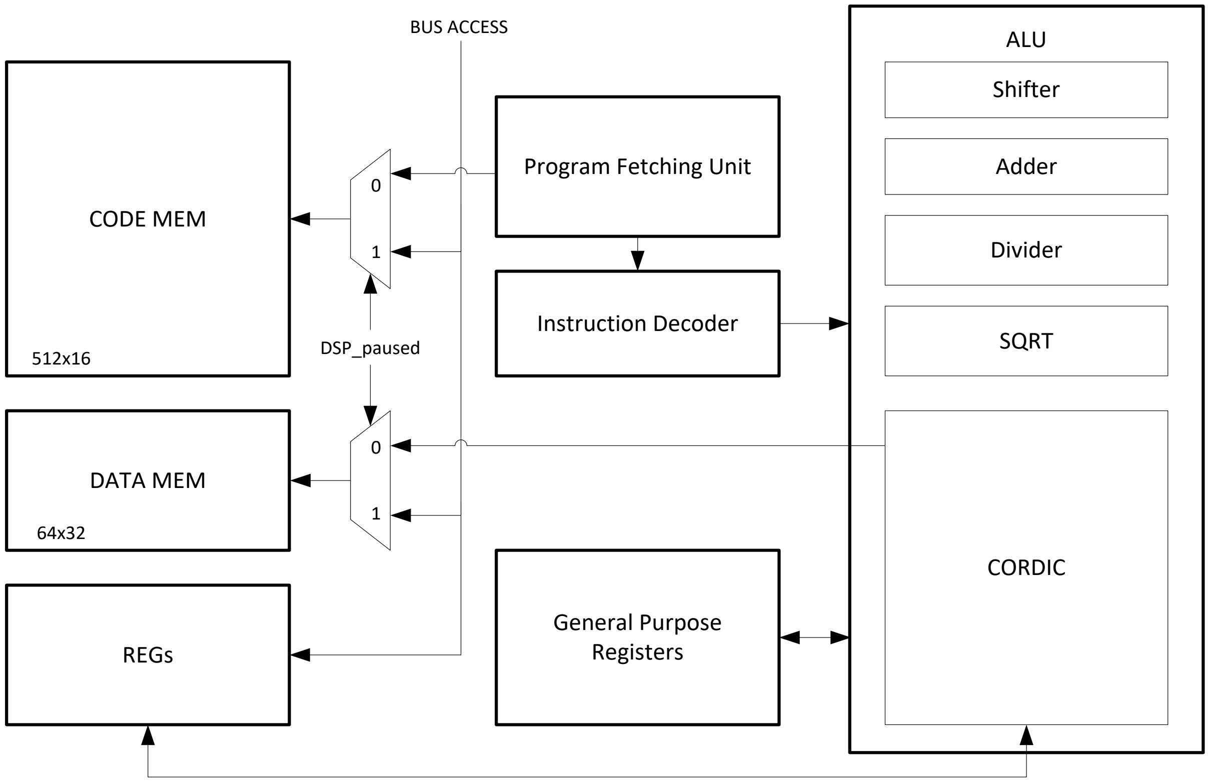


图 15-1 DSP 模块功能框图

15.1.2 DSP 核寄存器

表 15-1 DSP 核心寄存器

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

其中 R0 寄存器为常 0 寄存器，不能写入数据，读回恒为 0。R0 寄存器可作为特殊操作数构造一些伪指令。如

ADD R0 R0 R0 相当于 NOP

MAC R1 R2 R0 相当于 MUL R1 R2，即乘累加中累加的数为 0，变为乘法操作

R6 寄存器在 ARCTAN 指令中固定用于存放向量模值

R7 寄存器在 MAC 指令中固定用于存放计算结果，在 DIV 指令中固定用于存放被除数操作数。

以上约定是由于受定长指令编码限制,在 4 操作数指令中需要固定一个操作数使用约定寄存器。

15.1.3 位宽

除法的被除数和商位宽均为 32 位有符号数，除数和余数为 16 位有符号数。

被开方数为 32 位无符号数，平方根为 16 位无符号数。

乘累加的两个乘数均为 16 位有符号数，加数和结果为 32 位有符号数。

三角函数 CORDIC 模块位宽为 16 位，Q15 定点数格式。

注意：在使用 DSP 算术运算资源时，无论是 CPU 调用还是通过 DSP 算术指令调用，请务必保证操作数不要超过表示范围，否则会出现计算异常。例如乘法的两个操作数为 16 位有符号数，除法的除数为 16 位有符号数。16 位有符号数的表示范围为-32768~32767，如果输入的 32 位数据超过这一范围，比如以 50000 或-40000 作为乘法的操作数，则会因源操作数被截取了低 16 位而出现计算错误。

除法（DIV 指令）的操作数，被除数范围是-(2^31-1) ~ (2^31-1)，除数的范围是-(2^15-1) ~ (2^15-1)；被除数不支持赋值为-2^31，除数不支持赋值为-2^15。

减法（SUB 指令）的操作数，减数范围为-(2^31-1) ~ (2^31-1)，减数不支持赋值为-2^31。

此外，由于 CORDIC 运算基于多次旋转逐次逼近，计算结果存在不超过 0.1%的计算误差。

15.1.4 指令周期

除法指令需要 10 个总线周期（96MHz）完成。

开方指令需要 8 个总线周期（96MHz）完成。

三角函数指令需要 8 个总线周期（96MHz）完成。

其余指令均为单周期指令

15.1.5 地址空间

表 15-2 DSP 地址空间

模块	空间大小	空间区域	实际存储体大小
code_mem	2kB	0x4001_4000 ~ 0x4001_47FF	512 x 16bit
data_mem	2kB	0x4001_4800 ~ 0x4001_4FFF	64 x 32bit
reg	2kB	0x4001_5000 ~ 0x4001_57FF	
Reserved	2kB	0x4001_5800 ~ 0x4001_5FFF	

DSP 的地址空间分为 4 段，分别为程序段、数据段、寄存器段、保留段，各段空间占用 2kB 的地址空间，但实际的 CODE MEM 和 DATA MEM 存储空间不足 2kB。程序段（CODE MEM）用于存放 DSP 运行所需程序代码，为单口 SRAM，单周期完成读写操作；数据段（DATA MEM）用于存放 DSP 运行所需数据，为单口 SRAM，单周期完成读写操作；寄存器段为允许 CPU 通过总线访问的 DSP 寄存器；保留段暂未使用。

CODE MEM 位宽为 16，但仍按 word 寻址，亦即地址每次增 4。

DSP 地址空间需要在适当的时机才能访问。其中 DSP 的状态控制寄存器可以在任何时间访问；DSP 的 CODE MEM、DATA MEM、以及寄存器段中 CODIC 三角函数模块、DIV 除法器、SQRT 开方器只有在 DSP 暂停即 DSP_SC.PAUSED=1 时才能进行寄存器访问。因为在 DSP 运行期间，所有的运算单元可能在被 DSP 使用，CPU 同时通过寄存器接口进行访问会造成访问冲突。因此在 DSP 运行期间，即 DSP_SC.PAUSED=0 时，禁止通过寄存器接口访问 DSP 的运算单元。

DSP 遇到中断指令会提起中断等待 CPU 处理，同时 DSP 进入暂停状态，DSP_SC.PAUSED 会置位为 1,。此外软件也可以在任意时刻通过写 DSP_SC.PAUSED=1 来使 DSP 进入暂停状态，这一机制主要是为了防止 DSP 写入的程序中没有 IRQ 指令，导致 DSP 在不掉电的情况下会永久运行。

15.2 寄存器

15.2.1 地址分配

DSP 模块在芯片中的基地址是 0x4001_4000。DSP 寄存器在芯片中的基地址是 0x4001_5000。

表 15-3 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
DSP_DID	0x20	DSP 除法操作被除数
DSP_DIS	0x24	DSP 除法操作除数
DSP_QUO	0x28	DSP 除法操作商
DSP_REM	0x2C	DSP 除法操作余数
DSP_RAD	0x30	DSP 开方操作被开方数
DSP_SQRT	0x34	DSP 开方操作平方根

15.2.2 DSP 状态控制寄存器

15.2.2.1 DSP_SC

地址：0x4001_5000

复位值：0x2

表 15-4 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
												RESET_PC	CORDIC_MODE	PAUSED	IRQ
												WO	RW	RW	RWIC
												0	0	1	0

位置	位名称	说明
[31:4]		保留
[3]	RESET_PC	当 DSP 暂停时，写 1 重置 DSP PC 到 0 地址
[2]	CORDIC_MODE	CORDIC mode, 0: arctan, 1: sin/cos
[1]	PAUSED	指示 DSP 是否处于暂停状态，当 DSP 执行到 IRQ 指令时此 bit 置 1，软件写可以将此 bit 置 1。软件将此 bit 清零可以启动 DSP 运行 0： DSP 正读取 CODE MEM 和 DATA MEM 自主运行 DSP 程序 1： DSP 暂停取指令，允许 ARM 软件通过寄存器访问 DSP 内部算术单元，向寄存器写入操作数触发运算，读取寄存器取得运算结果 对于不编写 DSP CODE MEM 程序的用户，PASUED 需保持为 1
[0]	IRQ	DSP 中断标志，写 1 清零

注意，复位后 DSP 处于暂停状态，即 DSP_SC.PAUSED=1 时；DSP_SC.CORDIC_MODE 用于 CPU 通过寄存器接口访问 CORDIC 模块时，sin/cos mode 和 arctan mode 的选择，CORDIC 模块计算 sin/cos 或 arctan 使用的是相同的硬件电路。因此在进行某一种计算前，应通过配置 DSP_SC 计算器进行适当模式选择。

只有当 DSP 处于暂停状态时，CPU 才可以通过寄存器接口调用 DSP。当 DSP 自主运行时，CPU 无法通过寄存器接口访问 DSP 内部资源。

DSP_SC.CORDIC_MODE 位仅在 CPU 通过寄存器接口调用 DSP CORDIC 单元时需要设置，DSP 程序可以根据 SIN_COS 或 ARCTAN 指令直接进行模式切换，DSP_SC.CORDIC_MODE 不再起作用。

软件调用 CORDIC 模块计算 sin/cos 时以角度 DSP_THETA 为输入，计算并输出 sin/cos 结果到 DSP_SIN/DSP_COS 寄存器；计算 arctan 时以坐标 DSP_X/DSP_Y 为输入，计算并输出角度 theta= arctan (y/x)和 module= sqrt (x^2+y^2)到 DSP_ARCTAN 和 DSP_MOD 寄存器。

15.2.3 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.CORDIC_MODE 写为 1，使 CORDIC 进入 sin/cos 模式。

15.2.3.1 DSP_THETA

地址：0x4001_5004

复位值：0x0

表 15-5 DSP sin/cos 角度输入寄存器

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

位置	位名称	说明
[31:16]		保留，读出时为符号扩展，即{16{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）。

15.2.3.2 DSP_SIN

地址：0x4001_5010

复位值：0x0

表 15-6 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）。

15.2.3.3 DSP_COS

地址：0x4001_5014

复位值：0x0

表 15-7 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）。

15.2.4 DSP arctan 相关寄存器

15.2.4.1 DSP_X

地址：0x4001_5008

复位值：0x0

表 15-8 DSP arctan/module 坐标 X 输入寄存器

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

位置	位名称	说明
[31:16]		保留，读出时为符号扩展，即{16{DSP_X[15]}}
[15:0]	X	DSP arctan/module 计算输入坐标 X 寄存器

DSP_X 为 16 位有符号定点数，Q15 格式，其中 1bit 符号位，15bit 整数位，表示范围(-32768～32767)。

15.2.4.2 DSP_Y

地址：0x4001_500C

复位值：0x0

表 15-9 DSP arctan/module 计算坐标 Y 输入寄存器

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

位置	位名称	说明
[31:16]		保留，读出时为符号扩展，即{16{DSP_Y[15]}}

[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) 。

15.2.4.3 DSP_MOD

地址：0x4001_5018

复位值：0x0

表 15-10 DSP arctan 向量模结果 sqrt(X^2+Y^2) 寄存器

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

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

DSP_MOD 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)。

需要注意的是，当计算 arctan(Y/X)时，需要保证 sqrt(X^2+Y^2)也不超过 32767，因此一般建议 abs(DSP_X)和 abs(DSP_Y)不超过 2^{14} 为宜。

15.2.4.4 DSP_ARCTAN

地址：0x4001_501C

复位值：0x0

表 15-11 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) 。

需要注意的是，当计算 arctan(Y/X)时，需要保证 sqrt(X^2+Y^2)也不超过 32767，因此一般建议 abs(DSP_X)和 abs(DSP_Y)不超过 2^{14} 为宜。

15.2.5 DSP 除法相关寄存器

15.2.5.1 DSP_DID

地址：0x4001_5020

复位值：0x0

表 15-12 DSP 除法被除数寄存器

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

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

位置	位名称	说明
[31:0]	DID	DSP 除法被除数寄存器

15.2.5.2 DSP_DIS

地址：0x4001_5024

复位值：0x0

表 15-13 DSP 除法除数寄存器

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

位置	位名称	说明
[31:16]		保留，读出时为符号扩展，即{16{DSP_DIS[15]}}
[15:0]	DIS	DSP 除法除数寄存器

15.2.5.3 DSP_QUO

地址：0x4001_5028

复位值：0x0

表 15-14 DSP 除法商寄存器

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

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

位置	位名称	说明
[31:0]	QUO	DSP 商寄存器

15.2.5.4 DSP_REM

地址：0x4001_502C

复位值：0x0

表 15-15 DSP 除法余数寄存器

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

位置	权限	说明
[31:16]		保留，读出时为符号扩展，即{16{DSP_REM[15]}}
[15:0]	REM	DSP 除法余数寄存器

当 CPU 需要使用 DSP 除法器时，应首先保证 DSP 处于暂停状态。先向 DSP 写入被除数，后写入除数；写入除数可以触发一次除法操作，32/16 位除法需要 8 周期完成，期间读取除法结果 DSP_QUO 或 DSP_REM 会使 CPU 进入等待状态，等待除法计算完成并通过总线返回计算结果。

15.2.6 DSP 开方相关寄存器

15.2.6.1 DSP_RAD

地址：0x4001_5030

复位值：0x0

表 15-16 DSP 被开方数寄存器

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

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

位置	位名称	说明
[31:0]	RAD	DSP 被开方数寄存器

15.2.6.2 DSP_SQRT

地址：0x4001_5034

复位值：0x0

表 15-17 DSP 平方根寄存器

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

位置	位名称	说明
[31:16]		保留，读出时恒为 0
[15:0]	SQRT	DSP 平方根寄存器

当 CPU 需要使用 DSP 开方器时，应首先保证 DSP 处于暂停状态。向 DSP 写入被开方数；写入被开方数可以触发一次开方操作，32 位开方需要 8 周期完成，期间读取开方结果 DSP_SQRT 会使 CPU 进入等待状态，等待开方计算完成并通过总线返回计算结果。

15.3 DSP 指令集

15.3.1 Instruction Set Summary

Operation	Description	Assembler	Cycles
Add		ADD Rd1 Rs1 Rs2	1
	5bit Immediate	ADDI Rd1 Rs1 #<Imm>*1	1
Subtract		SUB Rd1 Rs1 Rs2	1
Shift	Arithmetic right shift	ASR Rd1 Rs1 Rs2	1
	5bit Immediate	ASRI Rd1 Rs1 #<Imm>	1
	Logical left shift	LSL Rd1 Rs1 Rs2	1
	5bit Immediate	LSLI Rd1 Rs1 #<Imm>	1
Multiply and accumulation		MAC Rs1 Rs2 Rs3	1
	5bit Immediate	MACI Rs1 Rs2 #<Imm>	1
Divide		DIV Rd1 Rs1 Rs2	10
Saturation		SAT Rd1 Rs1 Rs2	1
	4bit Immediate	SATI Rd1 #<Imm1> #<Imm2>	1
Cordic	SIN/COS	SIN_COS Rd1 Rd2 Rs1	8
	Arctan/Module	ARCTAN Rd1 Rs1 Rs2	8
Square root	Square root	SQRT Rd1 Rs1	8
Memory access	Load word	LDRWI Rd1 #<Imm>	1
	Load double half words	LDRDHI Rd1 Rd2 #<Imm>	1
	Store word	STRWI Rs1 #<Imm>	1
	Store double half words	STRDHI Rs1 Rs2 #<Imm>	1
Branch	Unconditional Jump	JUMP Rs1	2
	Immediate	JUMPI #<Imm>	2
	Jump if less than or equal to	JLE Rs1 Rs2 Rs3	2
	Immediate	JLEI Rs1 Rs2 #<Imm>	2
Miscellaneous	Generate IRQ and Pause DSP	IRQ	1

DSP 使用 16bit 定长编码指令，因为通用寄存器共 8 个，所以寄存器编码需要 3bit。其中大部分指令为 3 操作数指令，包括两个源操作数寄存器和一个目的操作数寄存器；部分指令包含立即数；部分指令会涉及 4 个操作数，以乘加（MAC）操作为例，Rd = Rs1*Rs2+Rs3，由于指令长度不够表示 4 个寄存器，将 Rd 固定为 R7，在指令编码中不显示。其余 4 操作数指令还有 ARCTAN/DIV。具体操作数分配见下文指令的详细解释。

15.3.2 ADD

15.3.2.1编码

15	14	13	12	11	10	9	8 7 6	5 4 3	2 1 0
0	0	0	0	0	0	0	Rs2	Rd1	Rs1

15.3.2.2汇编语法

ADD Rd1 Rs1 Rs2

15.3.2.3伪代码

Rd1 = Rs1 + Rs2

结果防溢出保护，

上溢出后 Rd1 = 0x7FFF_FFFF，

下溢出后 Rd1 = 0x8000_0000

15.3.3 ADDI (reserved)

带立即数的加法指令在此版本 DSP 中保留，未实现。

15.3.3.1指令编码

15	14	13	12	11	10 9 8 7 6	5 4 3	2 1 0
0	0	0	0	1	Imm	Rd1	Rs1

立即数为 5bit 有符号数，表示数值范围是-16~15.

15.3.3.2汇编语法

ADDI Rd1 Rs1 Imm

15.3.3.3伪代码

Rd1 = Rs1 + Imm

结果防溢出保护，

上溢出后 Rd1 = 0x7FFF_FFFF，

下溢出后 Rd1 = 0x8000_0000

15.3.4 SUB

15.3.4.1指令编码

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	0	0	Rs2			Rd1			Rs1		

15.3.4.2汇编语法

SUB Rd1 Rs1 Rs2

15.3.4.3伪代码

Rd1 = Rs1 – Rs2

结果防溢出保护,

上溢出后 Rd1 = 0x7FFF_FFFF,

下溢出后 Rd1 = 0x8000_0000

15.3.5 ASR

15.3.5.1指令编码

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	0	0	Rs2			Rd1			Rs1		

15.3.5.2汇编语法

ASR Rd1 Rs1 Rs2

15.3.5.3伪代码

Rd1 = Rs1 >> Rs2

算术右移指令只支持 0~31 bit 的右移。

15.3.6 ASRI

15.3.6.1指令编码

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	Imm					Rd1			Rs1		

立即数为 5bit 无符号数，表示数值范围是 0~31.

15.3.6.2汇编语法

ASRI Rd1 Rs1 Imm

15.3.6.3伪代码

Rd1 = Rs1 >> Imm

带立即数的算术右移指令只支持 0~31 bit 的右移。

15.3.7 LSL

15.3.7.1指令编码

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	0	0	Rs2			Rd1			Rs1		

15.3.7.2汇编语法

LSL Rd1 Rs1 Rs2

15.3.7.3伪代码

Rd1 = Rs1 << Rs2

逻辑左移只支持 0~31 bit 左移。

15.3.8 LSLI

15.3.8.1指令编码

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

0	0	1	1	1	Imm	Rd1	Rs1

立即数为 5bit 无符号数，表示数值范围是 0~31.

15.3.8.2汇编语法

LSLI Rd1 Rs1 Imm

15.3.8.3伪代码

Rd1 = Rs1 << Imm

带立即数的逻辑左移只支持 0~31 bit 左移。

15.3.9 MAC

15.3.9.1指令编码

15	14	13	12	11	10	9	8 7 6	5 4 3	2 1 0
0	1	0	0	0	0	0	Rs3	Rs2	Rs1

15.3.9.2汇编语法

MAC Rs1 Rs2 Rs3

15.3.9.3伪代码

Rd7 = Rs1 × Rs2 + Rs3

结果防溢出保护，

上溢出后 Rd7 = 0x7FFF_FFFF，

下溢出后 Rd7 = 0x8000_0000

其中 Rs1、Rs2 为 16 位有符号数，Rs3 为 32 位有符号数。注意，作为操作数的 Rs1 和 Rs2 不要超出 16 位有符号数的表达范围。当 Rs3 为 R0 时，MAC 可以当做乘法指令 MUL 使用。

15.3.10 MACI (reserved)

带立即数的乘加指令在此版本 DSP 中保留，未实现。

15.3.10.1指令编码

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	Imm					Rs2			Rs1		

立即数为 5bit 有符号数，表示数值范围是-16~15.

15.3.10.2汇编语法

MACI Rs1 Rs2 Imm

15.3.10.3伪代码

Rd7 = Rs1 × Rs2 + Imm

结果防溢出保护，

上溢出后 Rd7 = 0x7FFF_FFFF，

下溢出后 Rd7 = 0x8000_0000

15.3.11 DIV

15.3.11.1指令编码

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	0	0	Rd2			Rd1			Rs1		

15.3.11.2汇编语法

DIV Rd2 Rd1 Rs1

15.3.11.3伪代码

Rd1 = Rd7/Rs1, Rd2 = Rd7 % Rs1

除法指令需要 10 个周期才会结束提交，在除法进行计算的过程中，DSP 不应该再发射其他的多周期指令，即同一时刻只能由一个多周期指令在 long-pipeline 中。其他多周期指令包括三角函数指令以及开方指令。多周期指令可以后台运行，即多周期指令运算的同时，DSP 可以继续执行其他的单周期指令。但同一时间只能有一条多周期指令在后台运行。在多周期指令计算期间，DSP 仍可以使用多周期指令的目的操作数，但需要注意是当多周期计算完成提交结果的时候，目的操作数寄存器会被改写。

其中 Rd7 为 32 为有符号数，Rs1,为 16 为有符号数，Rd1 为 32 位有符号数，Rd2 为 16 位有符

号数。

15.3.12 SAT

15.3.12.1指令编码

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td colspan="3">Rs2</td><td colspan="3">Rd1</td><td colspan="3">Rs1</td></tr>
</table>

15.3.12.2汇编语法

SAT Rd1 Rs1 Rs2

15.3.12.3伪代码

If (Rd1<Rs1) Rd1=Rs1; else if (Rd1>Rs2) Rd1=Rs2

15.3.13 SATI (reserved)

带立即数的饱和指令在此版本 DSP 中保留，未实现。

15.3.13.1指令编码

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td colspan="4">Imm2</td><td colspan="4">Imm1</td><td colspan="3">Rs1</td></tr>
</table>

15.3.13.2汇编语法

SATI Rd1 Imm1 Imm2

15.3.13.3伪代码

If (Rd1<Imm1) Rd1=Imm1; else if (Rd1>Imm2) Rd1=Imm2

15.3.14 SIN_COS

15.3.14.1指令编码

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td colspan="3">Rd2</td><td colspan="3">Rd1</td><td colspan="3">Rs1</td></tr>
</table>

15.3.14.2汇编语法

SIN_COS Rd1 Rd2 Rs1

15.3.14.3伪代码

Sin/cos 指令 8 周期结束并提交，在其执行期间，DSP 不应该发射其他的多周期指令。

Rd1=cos (Rs1); Rd2=sin (Rs1)

15.3.15 ARCTAN

15.3.15.1指令编码

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	0	0	Rs2			Rd1			Rs1		

15.3.15.2汇编语法

ARCTAN Rd1 Rs1 Rs2

15.3.15.3伪代码

ARCTAN 指令 8 周期结束并提交，在其执行期间，DSP 不应该发射其他的多周期指令。

Rd1= arctan(Rs2/Rs1); R6 = sqrt(Rs1^2+Rs2^2)

15.3.16 SQRT

15.3.16.1指令编码

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	0	0	0	0	0	0	Rd1			Rs1		

15.3.16.2汇编语法

SQRT Rd1 Rs1

15.3.16.3伪代码

开方指令 8 周期结束并提交，在其执行期间，DSP 不应该发射其他的多周期指令。

Rd1 = sqrt(Rs1)

15.3.17 LDRWI

15.3.17.1指令编码

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	Imm						Rd1			0	0	0

15.3.17.2汇编语法

LDRWI Rd1 Imm

15.3.17.3伪代码

Rd1=word(SRAM[imm])

由于 Load 指令都是立即数指令，访问的数据地址可以在译码阶段产生，因此 load 操作可以在一个周期内完成。而类似的 load 指令，CPU 需要访问寄存器来计算地址，因此需要 2 个周期。

立即数 imm 的表达范围是 0~63。因为 DSP data mem 是 64 个 32bit word 构成。

15.3.18 LDRDHI

15.3.18.1指令编码

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	Imm						Rd1			Rd2		

15.3.18.2汇编语法

LDRDHI Rd1 Rd2 Imm

15.3.18.3伪代码

LDRDHI Rd1 Rd2

{Rd1,Rd2}= word(SRAM[Imm]),

从 data mem 取回的 32bit 数据高 16 位会进行符号扩展为 32 位，然后赋值给 Rd1，

低 16 位会进行符号扩展为 32 位，然后赋值给 Rd2。

立即数 imm 的表达范围是 0~63。因为 DSP data mem 是 64 个 32bit word 构成。

15.3.19 STRWI

15.3.19.1指令编码

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	Imm						0	0	0	Rs1		

15.3.19.2汇编语法

STRWI Rs1 Imm

15.3.19.3伪代码

word{SRAM[imm]}=Rs1

Store 指令的数据来自寄存器，因此即使待写入地址在译码阶段产生也无法立即完成 Store 操作。需要将地址锁存一个周期，然后与写入的数据一起送至 data memory 接口。因此 Store 指令后续如果立即连接 load 指令会出现访问冲突。在设计汇编程序时应避免出现此种指令序列。如果 STR 指令后必须是 LDR 指令，可以在两者之间插入 ADD R0 R0 R0 指令作为指令 bubble。

立即数 imm 的表达范围是 0~63。因为 DSP data mem 是 64 个 32bit word 构成。

15.3.20 STRDHI (reserved)

带立即数的存储双半字指令在此版本 DSP 中保留，未实现。

15.3.20.1指令编码

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	Imm						Rs1			Rs2		

15.3.20.2汇编语法

STRDHI Rs1 Rs2 Imm

15.3.20.3伪代码

word{SRAM[imm]}={Rs1, Rs2}

立即数 imm 的表达范围是 0~63。因为 DSP data mem 是 64 个 32bit word 构成。

15.3.21 JUMP (reserved)

基于寄存器的跳转指令在此版本 DSP 中保留，未实现。

15.3.21.1指令编码

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	0	0	0	0	0	0	0	0	Rs1		

15.3.21.2汇编语法

JUMP Rs1

15.3.21.3伪代码

PC = PC + 1 + Rs1

15.3.22 JUMPI

15.3.22.1指令编码

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

15.3.22.2汇编语法

JUMPI Imm

15.3.22.3伪代码

PC = PC + 1 + IMM

15.3.23 JLE

15.3.23.1指令编码

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	0	0	0	Rs3			Rs2			Rs1		

15.3.23.2汇编语法

JLE Rs1 Rs2 Rs3

15.3.23.3伪代码

PC = PC + 1 + Rs3, if (Rs1 <= Rs2)

15.3.24 JLEI

15.3.24.1指令编码

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	1	Imm					Rs2			Rs1		

15.3.24.2汇编语法

JLEI Rs1 Rs2 Imm

15.3.24.3伪代码

PC = PC + 1 + IMM, if (Rs1 <= Rs2)

15.3.25 IRQ

15.3.25.1指令编码

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

15.3.25.2汇编语法

IRQ

15.3.25.3伪代码

IRQ 指令会产生中断，等待 CPU 处理，并将 DSP 暂停运行。

15.3.26 R（仅用于模拟器）

15.3.26.1指令编码

无

15.3.26.2汇编语法

R5 0x5555 # Assign 0x5555 to R5

R1 20 # Assign 20 to R1

15.3.26.3伪代码

调试指令，仅在 DSP Emulator 中使用。可用于在 DSP 程序的任意位置将 R1~R7 置为任意指定数值。

15.3.27 BREAK（仅用于模拟器）

15.3.27.1指令编码

无

15.3.27.2汇编语法

BREAK

15.3.27.3伪代码

调试指令，仅在 DSP Emulator 中使用。可用于在 DSP 程序的任意位置插入断点并打印 R1~R7 寄存器值。断点命中后按回车继续程序运行。

15.3.28 END（仅用于模拟器）

15.3.28.1指令编码

无

15.3.28.2汇编语法

END

15.3.28.3伪代码

类似于断点指令，仅在 DSP Emulator 中使用。在指令模拟中遇到该指令即中断模拟器的运行，直接打印核心寄存器值。

15.4 应用指南

15.4.1 访存寻址

DSP 使用立即数寻址，由于 DATA MEM 地址空间有限，STR 和 LDR 指令中的 6bit 立即数可以直接表示 DATA MEM 全部地址偏移。

以如下 DATA MEM 内容为例,第一行数据 0x01000100 对应 0x0 地址,第二行数据 0x30005000 对应 0x1 地址，第三行数据 0x0003FFF8 对应 0x2 地址。

使用 LDRDHI R1 R2 0x1 可对 R1,R2 进行赋值 R1=0x3000，R2=0x5000

使用 STRWI R3 0x3 可以将 R3 的 32bit 数据写入 0x3 地址并覆盖掉数据 0xFFFFE000。

需要注意的是，虽然 DSP 访存寻址每增加 1 是增加 4Byte，对于 CPU 来说，每增加 4 地址才是增加 4Byte。因此 CPU 访存 DSP DATA MEM 时，应该按照 DSP_DATA_MEM_BASE+offset*4 的方式进行计算。以如下 DATA MEM 内容为例,第二行数据 0x30005000 对应的 CPU 寻址地址为 0x4001_4804，第三行数据 0x0003FFF8 对应的 CPU 寻址地址为 0x4001_4808。

DATA MEM：

0x00100010

0x30005000

0x0003FFF8

0xFFFFE000

0x30004000

0x7FFFFFFF

0x7FFFFFFF

0xF0000003

0x00007FFF   # 8

0x00008000

0x80000000

0x00000000

15.4.2 Load after Store

DSP STR 指令之后不能立即使用 LDR，可在其间插入 “ADD R0 R0 R0” 作为“NOP”指令使用。

15.4.3 多周期指令的的延迟提交

DSP 的多周期算术指令包括除法、开方、三角函数。

除法指令需要 10 个总线周期（96MHz）完成。

开方指令需要 8 个总线周期（96MHz）完成。

三角函数指令需要 8 个总线周期（96MHz）完成。

为了充分利用 DSP 性能，允许多周期指令后台执行，即多周期指令计算期间 DSP 仍可其他指令，而不会阻塞流水线。这需要在编程时对指令序列有所考虑，在多周期指令的发射和结果使用之间插入其他不相关指令。如下指令序列，其中 ADD R0 R0 R0 作用类似 NOP，在实际应用中可以替换为其他指令。

```
SIN_COS R1 R2 R3
ADD R0 R0 R0
ADD R0 R0 R0
ADD R0 R0 R0
ADD R0 R0 R0
ADD R0 R0 R0
ADD R0 R0 R0
ADD R0 R0 R0
LDRWI R4 0x10
MAC R1 R4 R0
```

15.4.4 数据填装

DATA MEM 支持按字、半字、字节写入。为了解决 AHB 总线协议写后立即读问题，SRAM 端口处有一个 word 深度的 cache。DSP CODE MEM 和 DATA MEM 由于允许 CPU 和 DSP 两个主设备访问，就存在一致性问题。举例来说，如图 15-2 所示。CPU 先向 DSP DATA MEM 顺序写入 4 个 word，然后读取最后一个 word。由于 cache 的存在，最后一个 word 实际上留存在 cache 中，而没有真正写入 DSP DATA MEM，但 CPU 读取这一 word 数据时，命中 cache，所以直接从 cache 返回数据，读回的数据是正确的。但 DSP 读取 DATA MEM 时，则只能读取到前 3 个 word，留存在 cache 中的的最后一个 word 是访问不到的。

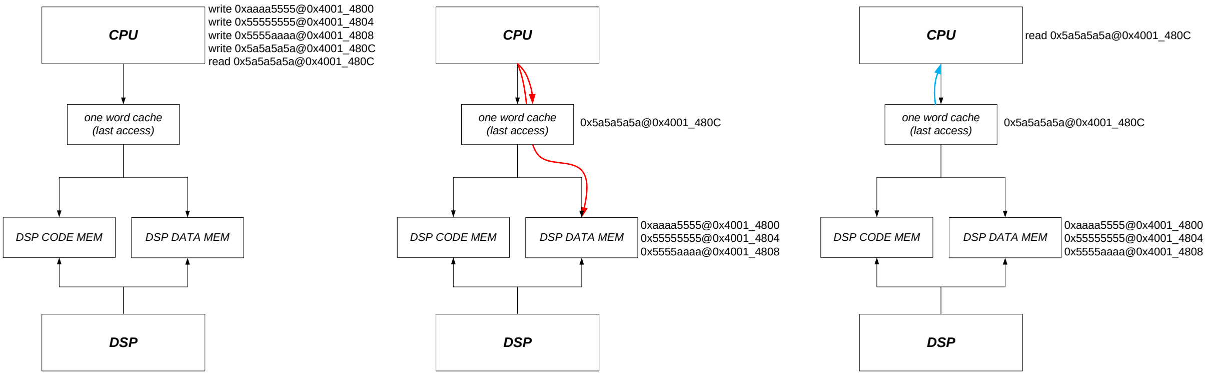


图 15-2 CPU 访问 DSP MEM 一致性问题

因此要求最后一次写入后需要再写任意一个其他地址，从而将停留在 cache 中的数据真实地写入 DSP MEM。对于单一主设备的系统来说，所有的读写操作都通过接口上的 cache，所以不会出现一致性问题，但由于 DSP 中的 MEM 被 CPU 写入后是由 DSP 经由另外的通道进行访问，所以需要通过一次冗余的写入操作实现 cache 数据推送，如图 15-3 所示。

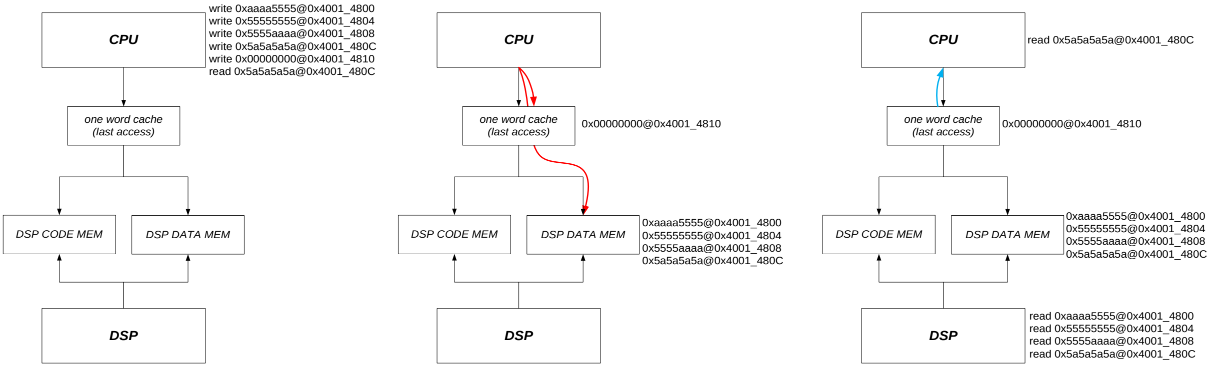


图 15-3 增加 Dummy 写操作解决 DSP MEM 一致性问题

同理，由于 CPU 最后一次读写的地址信号会留存在 Cache 中，如果之后 DSP 在运行中修改了 DATA MEM 数据。而后 CPU 读取 DSP DATA MEM 数据，如果命中 CPU 最后一次读写的地址，CPU 会去 cache 中取数据，而不会从 DSP DATA MEM 取数，即 CPU 认为 cache 数据仍然有效，如。因此应用上应该 CPU 最后一次写入一个 dummy 数据方式来规避此种情况出现。

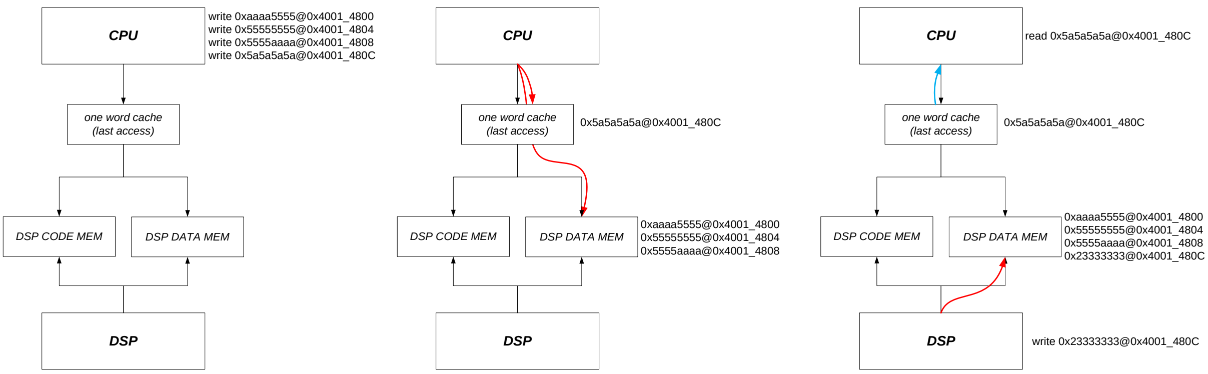


图 15-4 CPU 因命中 cache 无法读取 DSP DATA MEM 真实数据

16 I2C

16.1 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。根据特定设备的需要，可以使用 DMA 以减轻 CPU 的负担。

16.2 主要特性

- 多主机功能：该模块既可做主设备也可做从设备。

  I2C 主设备功能：产生时钟、START 和 STOP 事件。

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

16.3 功能描述

16.3.1 功能框图

本接口采用同步串行设计，实现 CPU 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。

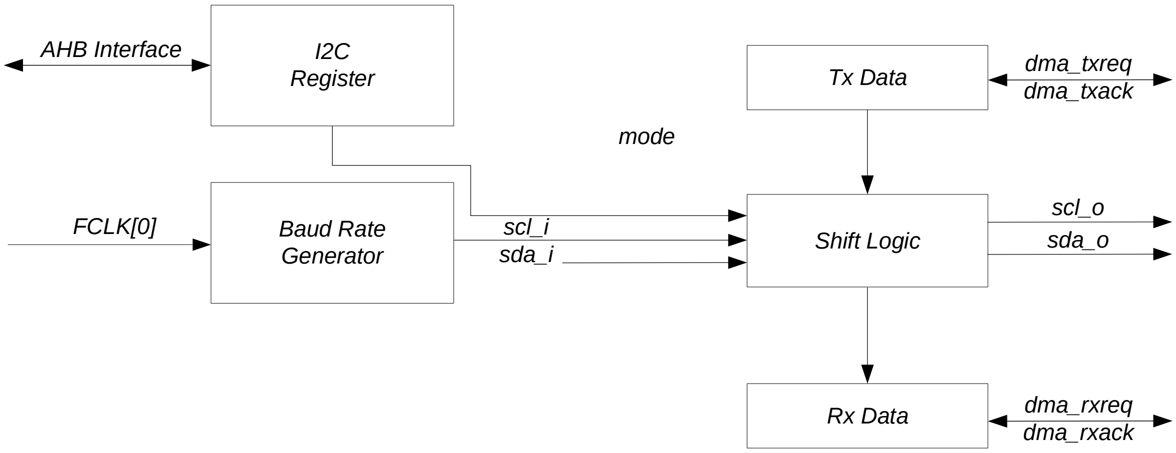


图 16-1 I2C 模块顶层功能框图

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。

16.3.2 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。

16.3.2.1 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 CPU 的负担。只有地址匹配才通知 CPU 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。

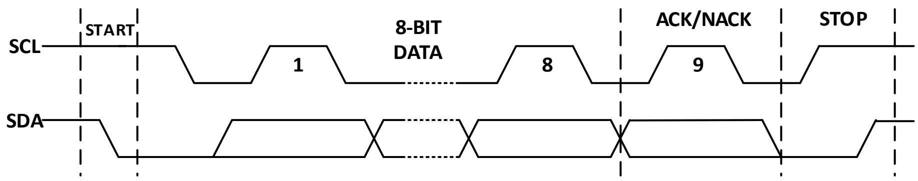


图 16-2 基本 I2C 传输时序图

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

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

一般情况下，非 DMA 方式时，一次传输一个字节（可反复单次传输，需软件介入提供数据）。DMA 方式,一次连续传输可以多字节(最大不超过 16 字节,极端情况一次传输一个字节,因无 FIFO,每次 DMA 请求，仅传输一个字节，多轮完成本次数据传输)。

上述所有模式，遵循如下基本原则:

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

16.3.2.2 I2C 接口从模式

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

- 从模式下，I2C 接口时刻在监控总线上的信号。一旦检测到起始条件，其将保存地址位数据和读写位数据。

- 从模式下，若硬件地址匹配功能开启，只有地址匹配的情况下，才会产生中断，通知 CPU 进行后续处理。若没有开启，每次收到地址及读写位数据，都将产生中断。
- 从模式下，单字节接收模式。每次收到一个字节的数据后，产生中断，此时 I2C 接口可拉低 SCL，直至中断完成，继续后续操作。
- 从模式下，单字节发送模式。每次发送一个字节完毕后且收到响应（ACK/NACK），产生中断，此时 I2C 接口可拉低 SCL，直至中断完成，继续后续操作。
- 从模式下，DMA 接收模式。每次收到 SIZE 约定后的数据，产生中断，此时 I2C 接口可拉低 SCL，直至中断完成。
- 从模式下，DMA 发送模式。每次发送 SIZE 约定后的数据并收到响应（ACK/NACK），产生中断，此时 I2C 接口可拉低 SCL，直至中断完成。

16.3.2.2.1 从模式单字节传输

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

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

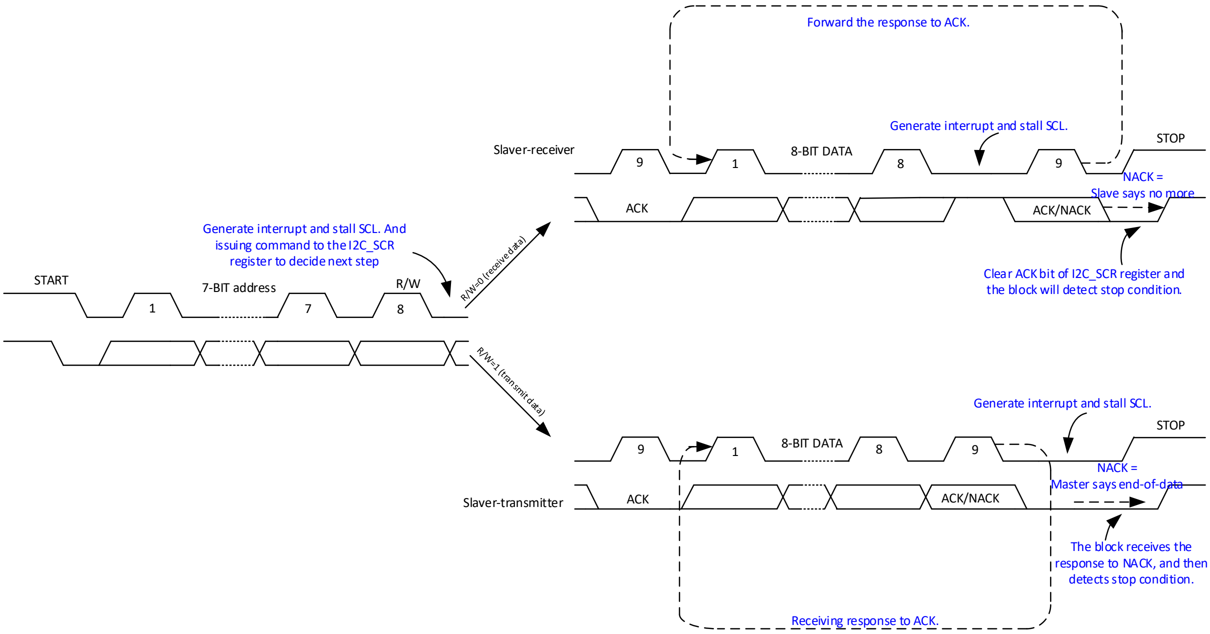


图 16-3 从模式下单字节传输示意图

16.3.2.2.2 从模式单字节发送

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

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

16.3.2.2.3 从模式单字节接收

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

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

16.3.2.3 从模式 DMA 传输

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

- 配置 I2C 从地址，使能 I2C 中断（可使能硬件地址比较中断）。地址匹配，产生 I2C 地址匹配中断，在中断处理函数中，配置 DMA，准备好发送数据或者准备好接收地址。然后写 I2C_SCR，准备开始传输或者停止本次传输。
- 若是接收模式，I2C_BCR.BURST_SIZE 约定的字节接收完毕后，产生中断，软件判断是否继续接收，返回 ACK/NACK 响应。
- 若是发送模式，I2C_BCR.BURST_SIZE 约定的字节发送完毕后，等待响应（ACK/NACK），产生中断，根据响应判断后续操作。
- 获得总线完成标志，本次传输完成。

16.3.2.3.1 从模式 DMA 发送

地址匹配后，配置完毕 DMA。通过发送 DMA 请求，将字节从 RAM 搬移到 I2C_DATA 寄存器，然后经由内部移位寄存器发送到 SDA 线上。在 I2C_DATA 数据没有准备好之前，从设备可拉底 SCL，

直到待发送数据已写入 I2C_DATA 寄存器。从模式下发送数据，需要软件协助触发第一次 DMA 搬移，配置 I2C_BCR.BYTE_CMPLT 为 1 即可。I2C 接口在发送完毕 I2C_BCR.BURST_SIZE 约定的字节数据后都执行下列操作:

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

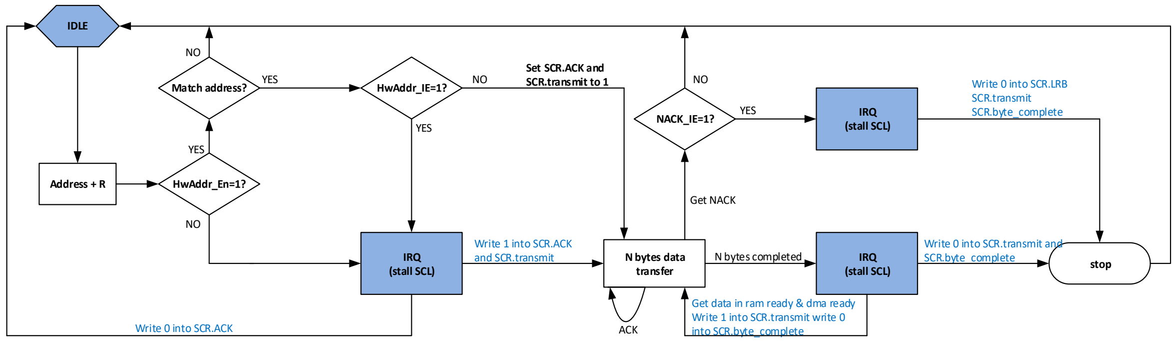


图 16-4 从模式下多字节发送示意图

16.3.2.3.2 从模式 DMA 接收

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

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

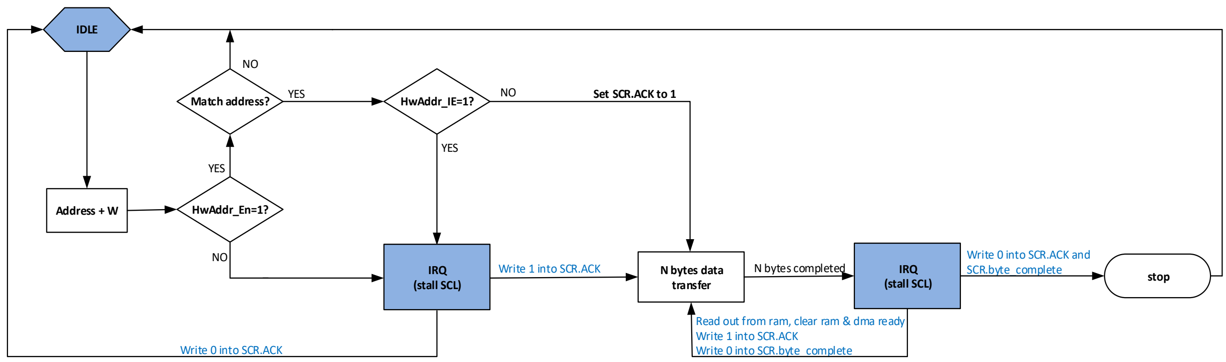


图 16-5 从模式下多字节接收示意图

16.3.2.4 I2C 接口主模式

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。

16.3.2.4.1 主模式单字节传输

单字节传输，并不意味着仅仅传输一个字节数据，其含义为每次传输完一个字节的数据后，将产生中断判断是否还要继续传输。单字节传输的极端情况是仅传输一个字节的数据。图 6 为单字节传输的总线示意图。从图可知，一般单字节传输的流程如下：

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

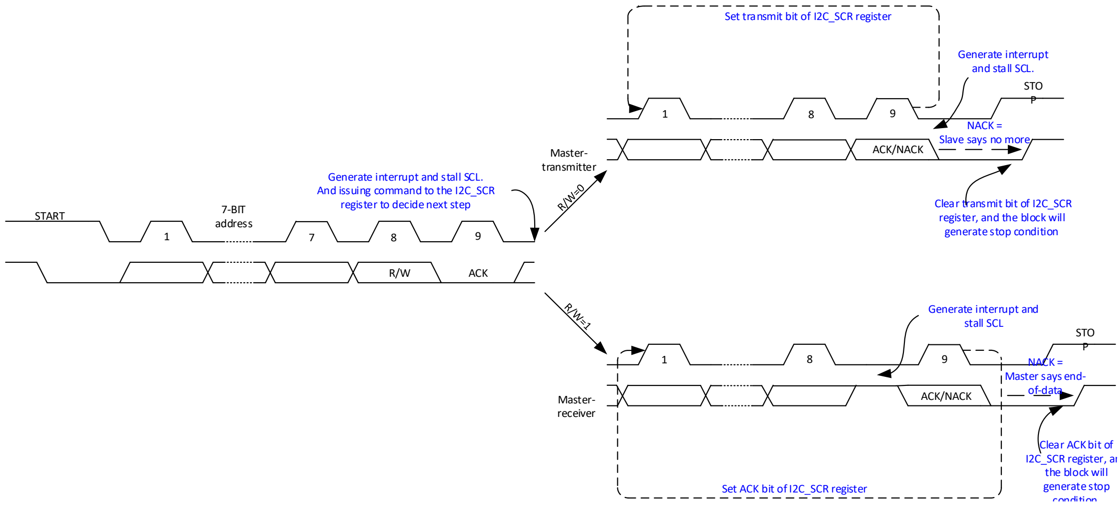


图 16-6 主模式下单字节传输示意图

16.3.2.4.2 主模式单字节发送

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

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

16.3.2.4.3 主模式单字节接收

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

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

16.3.2.4.4 主模式 DMA 传输

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

- 总线空闲，准备开始传输。
- 若是接收模式，I2C_BCR.BURST_SIZE 约定的字节接收完毕后，产生中断，软件判断是否继续接收，返回 ACK/NACK 响应。
- 若是发送模式，I2C_BCR.BURST_SIZE 约定的字节发送完毕后，等待响应（ACK/NACK），产生中断，根据响应判断后续操作。
- 发送 STOP 事件，本次传输完成。

16.3.2.4.5 主模式 DMA 发送

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

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

异常情况是:

- 若从设备地址不匹配或者从设备没有准备好，此时从设备将返回 NACK
- 主设备产生 STOP 事件，停止本次传输。
- 等待一段时间后，重新配置 I2C 寄存器，关闭 DMA 对应的通道使能信号，重新配置 DMA 寄存器，再次发送传输请求。关闭 DMA 对应通道是因为 I2C 有预取，DMA 已经不是初始状态。

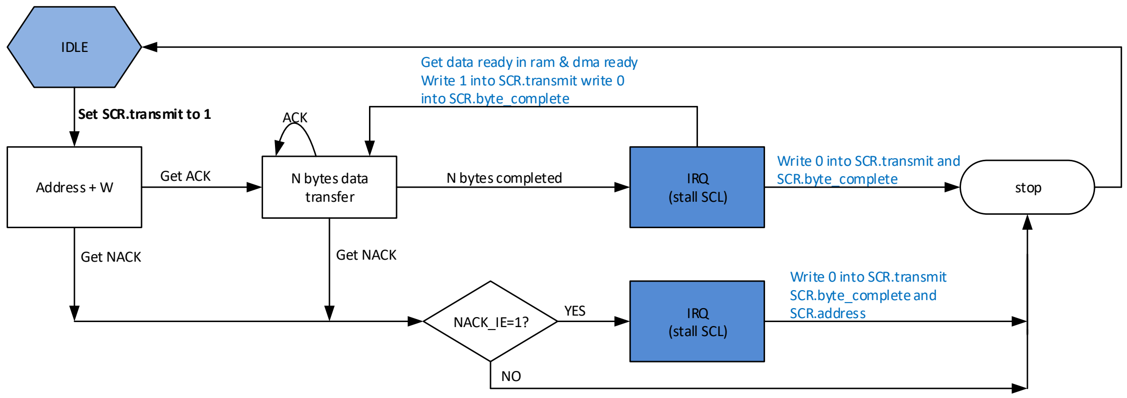


图 16-7 主模式下多字节发送示意图

16.3.2.4.6 主模式 DMA 接收

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

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

异常情况是:

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

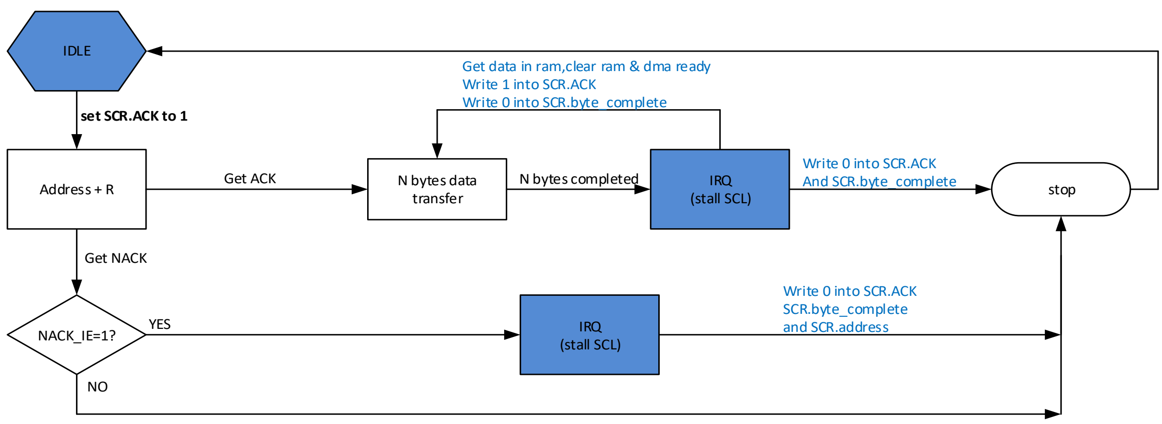


图 16-8 主模式下多字节接收示意图

16.3.2.5 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。

16.3.2.6 DMA 传输

在大容量数据传输应用下，I2C 接口支持 DMA 传输，减轻 CPU 的负担。一次传输，最大传输量为 16 字节，最小传输量为 1 字节。因 I2C 无 FIFO，I2C 发送一次 DMA 请求，DMA 只能搬移一个字节数据。若要实现多字节搬移，DMA 需配置为多轮搬移，每一轮搬移一个字节的方式。

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

DMA 传输需要配置 DMA 模块相应寄存器。

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

DMA 传输，推荐软件配置流程如下:

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

若 I2C 接口为从发送模式，需要考虑预取，I2C_BCR B[4]为预取开关。一般从模式传输的时候，可以开启硬件地址比较(I2C_ADDR B[7])，可选择是否开启硬件地址比较中断使能(I2C_BCR B[6])。

- 开启硬件地址比较中断，从设备收到的地址同自身匹配成功后，会产生中断。通知软件此时主设备请求获得从设备数据，软件判断是否接收，若接收就回 ACK，软件需要 I2C_BCR B[4]置 1，协助硬件预取第一次传输的数据；否则就回 NACK。
- 关闭硬件地址比较中断，一旦总线上出现 START 事件，从设备硬件预取第一次传输数据，无论从设备是否地址匹配成功。匹配成功，也不会产生地址匹配中断，直接开始数据传输。匹配不成功，从设备不会传输数据。此时，若匹配不成功，因为 I2C 有预取的操作，为后续匹配成功后传输正常，需要对 DMA 进行清除操作。

16.3.2.7 CPU 传输

CPU 传输，一次只能发送/接收 1 个字节，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。

CPU 传输，推荐软件配置流程如下：

- 初始化 GPIO 模块，将 I2C 复用的 GPIO 配置完毕。
- 初始化 I2C 接口，IE/CFG 等寄存器配置完毕。
- CPU 触发 I2C 接口进入发送流程，发送的数据来自 CPU 对 I2C_DATA 写入值。

16.3.2.8 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 的 BIT0 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 的 BIT7 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 的 BIT5 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 的 BIT1 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 的 BIT3 写 0 清除。
- 使用 DMA 协助数据传输。若本模块为接收模式，I2C 收到数据后还需要通过 DMA 搬移到 RAM，此时数据最终完成是要看 DMA 是否搬移完成，若使用 I2C 的完成中断作为判断依据的话，推荐在中断处理函数中查询下 DMA 的状态。若本模块为发送模式，无此问题。直接使用 I2C 的完成中断作为判断依据即可。

16.3.2.9 通讯速度设置

I2C 模块的波特率时钟来自系统时钟的分频，分频寄存器为 SYS 模块的 CLK_DIV0。

I2C 接口采用同步设计，需要对外部设备的信号进行同步采样，同步时钟为 I2C 接口工作时钟。数据和时钟信号的时钟频率为接口工作时钟/17。

I2C 模块工作时钟频率=系统频率/(CLK_DIV0 + 1)

I2C 波特率：I2C 波特率 = (系统频率 / (CLK_DIV0 + 1)) / 17

16.4 寄存器

16.4.1 地址分配

I2C 模块寄存器的基地址是 0x4001_0400，寄存器列表如下：

表 16-1 I2C 寄存器地址分配表

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

16.4.2 寄存器说明

16.4.2.1 I2C0_ADDR 地址寄存器

地址：0x4001_0400

复位值：0x0

表 16-2 地址寄存器 I2C0_ADDR

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

位置	位名称	说明
[31:8]		未使用
[7]	ADDR_CMP	I2C 硬件地址比较使能开关，只有在 DMA 模式下开启才有效。默认值为 0。 1：使能 0：关闭
[6:0]	ADDR	从模式下，I2C 设备硬件地址。主模式下，仅在 DMA 模式下，才需填入从设备地址；否则，从设备地址写入 I2C_DATA 寄存器。

16.4.2.2 I2C0_CFG 系统控制寄存器

地址：0x4001_0404

复位值：0x0

表 16-3 系统控制寄存器 I2C0_CFG

15-8	7	6	5	4	3-2	1	0
	IE	TC_IE	BUS_ERR_IE	STOP_IE		MST_MODE	SLV_MODE

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

16.4.2.3 I2C0_SCR 状态控制寄存器

地址：0x4001_0408

复位值：0x0

表 16-4 状态控制寄存器 I2C0_SCR

15-8	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，仅用于主模式下，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，仅用于主模式下，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，主从模式均可使用，写 0 清除。 0：无 STOP 事件 1：有 STOP 事件
[4]	BYTE_CMPLT	ACK 发生控制位，主从模式均可使用。发送前配置好改位。单字节模式下，发送字节完毕后硬件自动清除；多字节模式下，全部发送完毕后硬件自动清除。 0：字节发送完成，返回 NACK 回应 1：字节发送完成，返回 ACK 回应
[3]	ADDR_DATA	Address 数据标志位，主从模式均可使用，写 0 清除。 0：发送或接收到的数据非 Address 数据 1：发送或接收到的数据是 Address 数据
[2]	DATA_DIR	发送或接收控制位，主从模式均可使用，通过此位决定数据传输方向。 总线上 START 事件可清除此位，同时也可写 0 清除此位。 0：接收 1：触发发送
[1]	RX_ACK	接收响应标志位，主从模式均可使用，总线上 START 事件可清除此位，同时也可写 0 清除此位。 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，主从模式均可使用，总线上 START 事件可清除此位，同时也可写 0 清除此位。 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。

16.4.2.4 I2C0_DATA 数据寄存器

地址：0x4001_040C

复位值：0x0

表 16-5 数据寄存器 I2C0_DATA

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

位置	位名称	说明
[31:8]		未使用
[7:0]	DATA	数据寄存器，主从模式均可使用。对寄存器写入，数据进入 I2C 内部发送部分，无法直接读取到刚写入数据。读出操作，获得 I2C 接收到的数据。 主/从接收模式：存放接收的数据。单字节传输，需等待完成中断；多字节传输，数据会被 DMA 搬移到 RAM，仅保留最后一个字节。 从发送模式：存放发送的数据。一般先填好数据，然后触发 I2C_SCR 的 BIT[2]。多字节传输，数据会被 DMA 从 RAM 搬移到 I2C_DATA，然后由 I2C 接口发送。 主发送模式：单字节传输，I2C_DATA 填入地址；多字节传输，I2C_ADDR 填入地址，I2C_DATA 数据由 DMA 传输。

16.4.2.5 I2C0_MSCR 主模式寄存器

地址：0x4001_0410

复位值：0x0

表 16-6 主模式寄存器 I2C0_MSCR

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

位置	位名称	说明
[31:4]		未使用
[3]	BUSY	I2C 总线，闲忙状态。1，检测到 START 事件，忙碌；0，检测到 STOP 事件，空闲。
[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。

16.4.2.6 I2C0_BCR DMA 传输控制寄存器

地址：0x4001_0414

复位值：0x0

表 16-7DMA 传输控制寄存器 I2C0_BCR

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

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

17 SPI

17.1 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。

17.2 主要特性

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

17.3 功能描述

17.3.1 功能框图

本接口采用同步串行设计，实现 CPU 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。

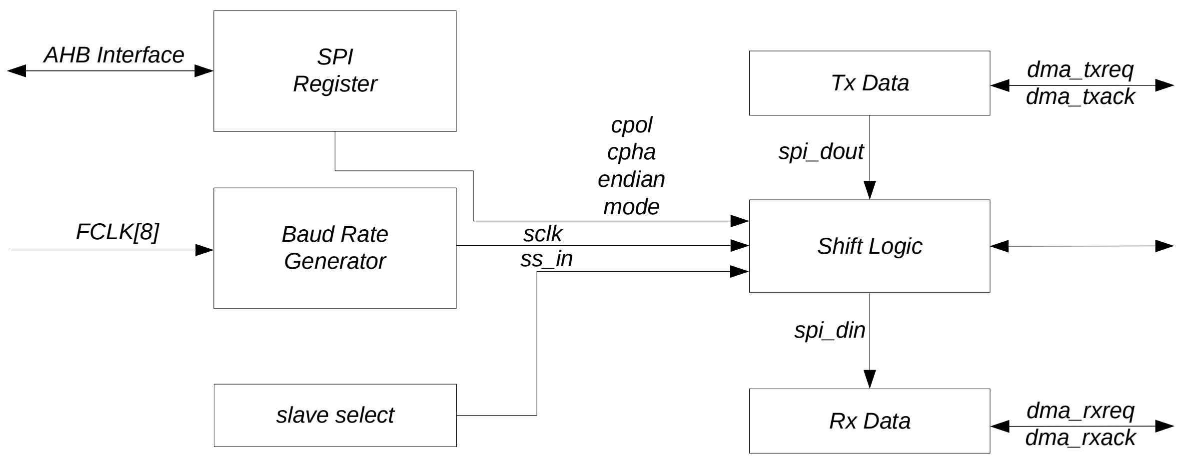


图 17-1 SPI 模块结构框图

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。

17.3.2 功能说明

17.3.2.1 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

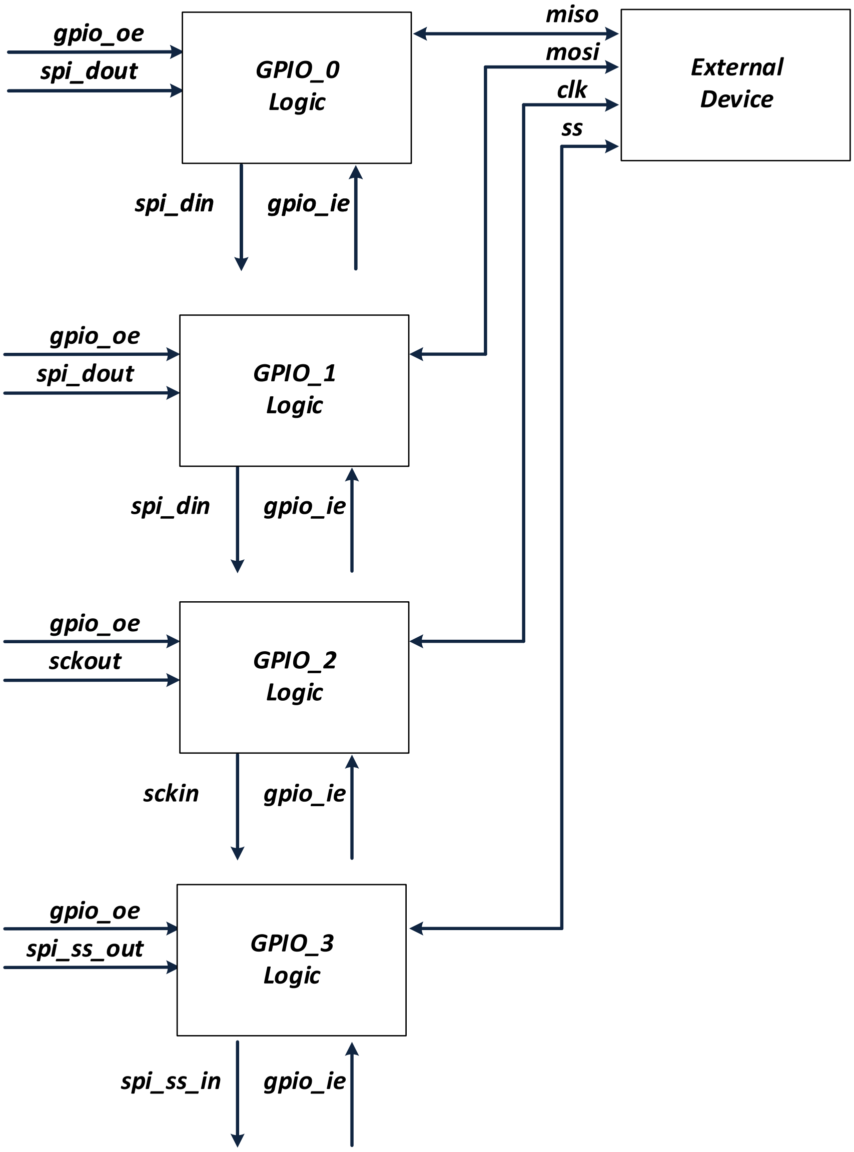


图 17-2 SPI 接口全双工模式互连框图

从上图可知，GPIO 若配置为输出，则 SPI 接口可发送数据；GPIO 若配置为输入，则 SPI 接口可接收数据。亦即 SPI_DI 关闭 GPIO 输入使能且开启 GPIO 输出使能时，即可作为 SPI_DO 使用，SPI_DO 关闭 GPIO 输出使能且开启 GPIO 输入使能时，即可作为 SPI_DI 使用。

17.3.2.2 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。

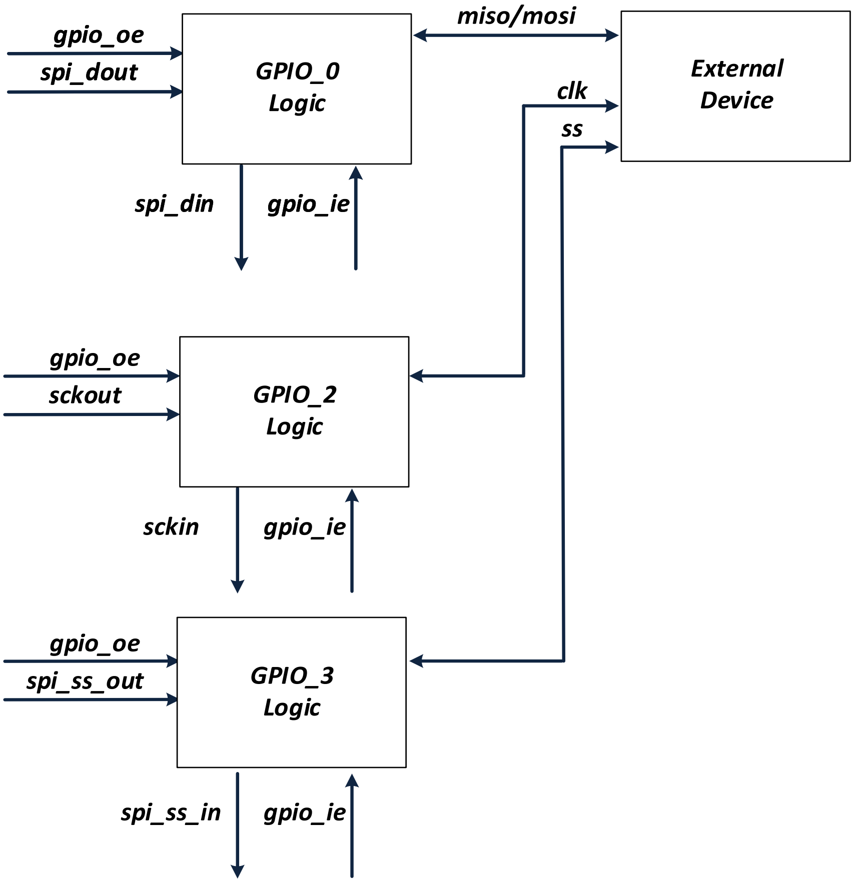


图 17-3 SPI 接口半双工模式互连框图

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

CFG[7:6]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 DMA 传输、支持 Master/Slave 模式下的发送。

仅接收

CFG[7:6]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 DMA 传输、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。

17.3.2.3 片选信号

本接口做 Slave 模式时，片选信号可选，CFG[5]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。

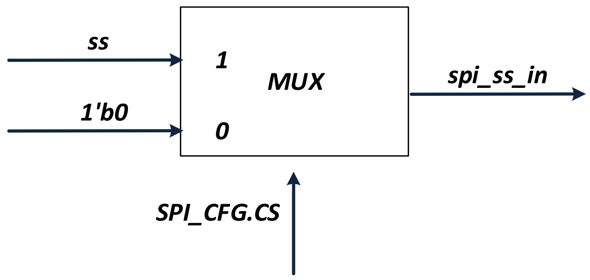


图 17-4 SPI 模块 Slave 模式片选信号选择

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。

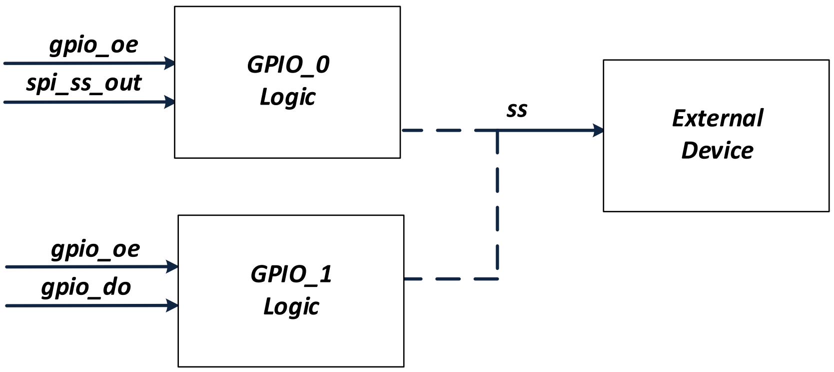


图 17-5 SPI 模块 Master 模式片选信号选择

注意，图 14-5 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。

17.3.2.4 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 CFG[3:2]控制。CFG[3]为 Phase 控制位，CFG[2]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。

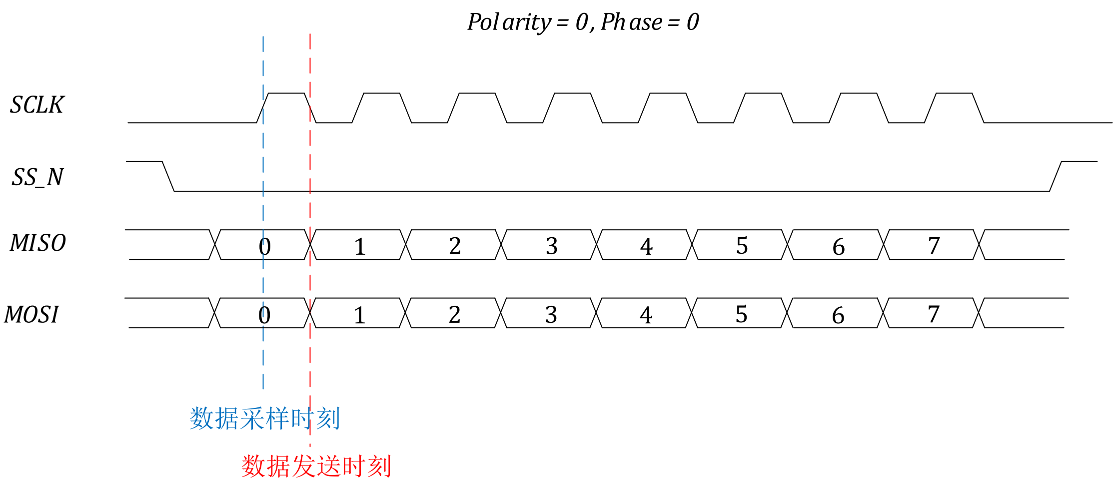


图 17-6 SPI 通讯信号极性相位(Polarity=0, Phase=0)

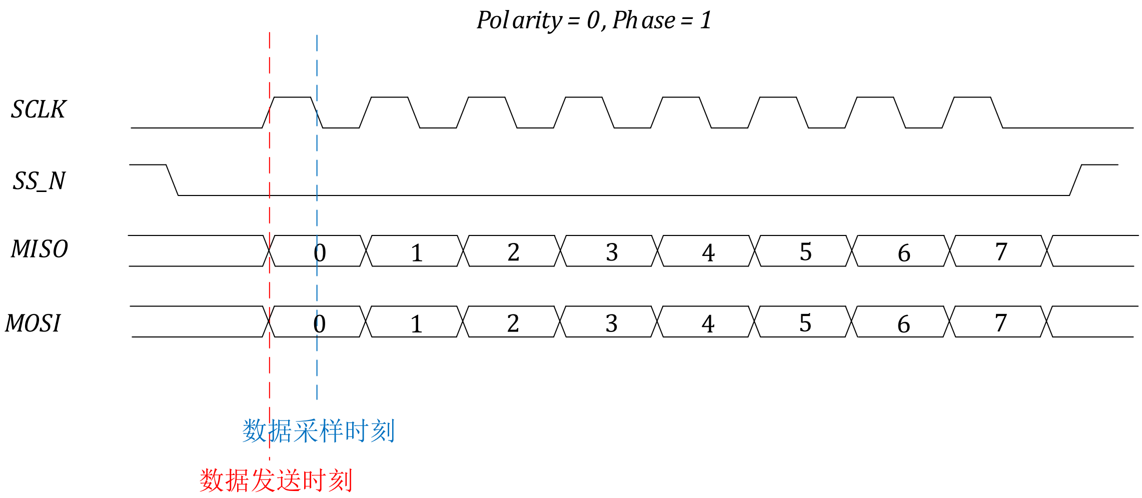


图 17-7 SPI 通讯信号极性相位(Polarity=0, Phase=1)

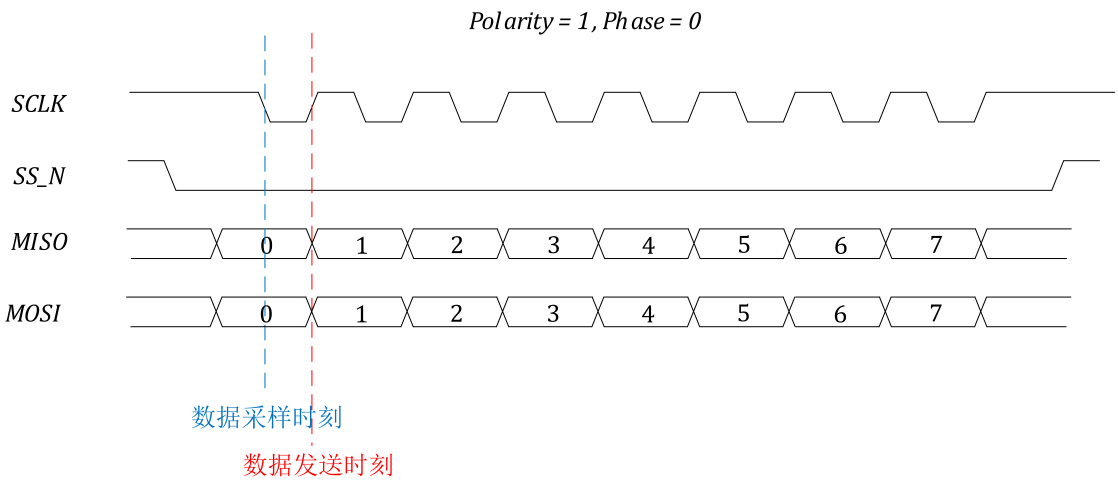


图 17-8 SPI 通讯信号极性相位(Polarity=1, Phase=0)

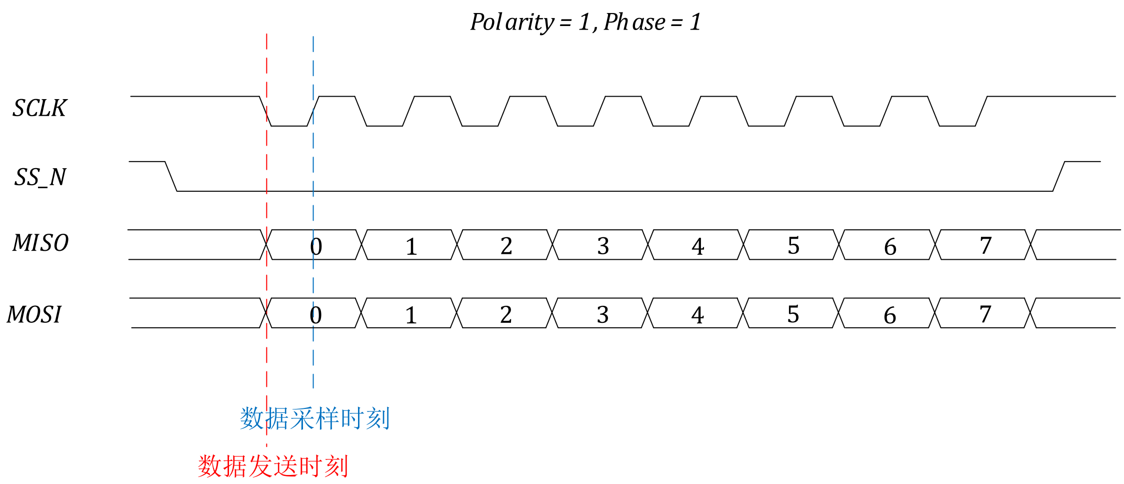


图 17-9 SPI 通讯信号极性相位(Polarity=1, Phase=1)

17.3.2.5 数据格式

SPI 数据传输格式分成两种：MSB 和 LSB。数据传输格式受到 CFG[1]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。

17.3.2.6 DMA 传输

在大容量数据传输应用下，SPI 接口支持 DMA 传输，减轻 CPU 的负担。一次传输，最大传输量为 255 字节，最小传输量为 1 字节。在全双工模式下，接收和发送均可实现 DMA 传输；在半双工模式下，仅接收或发送实现 DMA 传输。

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

DMA 传输需要配置 DMA 模块相应寄存器。

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

DMA 传输，推荐软件配置流程如下：

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

17.3.2.7 CPU 传输

CPU 传输，一次只能发送/接收 1 个字节，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。

SPI 接口支持 DMA 传输，也支持 CPU 传输。CPU 传输，SPI 接口无需触发发送状态，只要 CPU 将数据搬移到 SPI 接口，就可开始发送数据到外界。

CPU 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，IE/CFG/BAUD/SIZE 等寄存器配置完毕。注意 SIZE 只能配置为 1。
- CPU 对 TX_DATA 寄存器执行写操作，触发 SPI 接口进入发送流程，发送的数据来自 CPU 对 TX_DATA 写入值。

注意：若需要连续发送，则需要重新配置 SIZE 和 TX_DATA 寄存器。[1]

17.3.2.8 外部事件传输

在大容量数据传输应用下，SPI 接口支持 DMA 传输。传输过程中，可被打断也可不被打断。所谓打断是指，当前字节传输完成，需等待外部事件才开始下一个字节的传输；不打断是指，当前字节传输完成，直接下一个字节的传输。打断模式需要同其它模块配合使用。例如定时器模块，定时器可触发下一字节的传输。打断模式仅在 Master 模式有效。IE[3]控制了是否使用打断模式。

17.3.2.9 中断处理

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

- 数据完成事件，当前数据传输完成。高电平有效，对 IE[0]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 IE[1]写 1 清除。
- 溢出事件，数据没有及时通过 DMA 返回 RAM，或者从 RAM 获得数据，将产生溢出事件。高电平有效，对 IE[2]写 1 清除。

上述事件，默认是不触发 SPI 中断，可以通过配置 IE[6:4]使能事件产生中断。

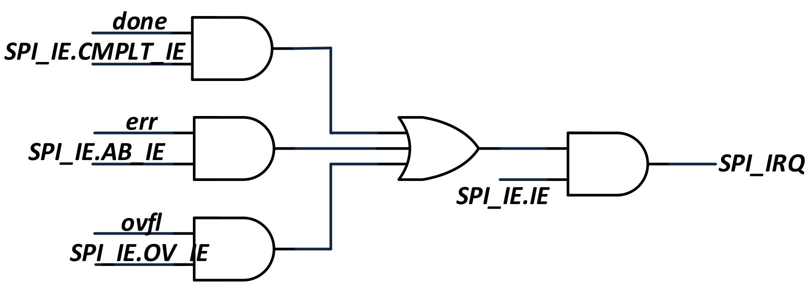


图 17-10 SPI 模块中断选信号产生图

数据传输完毕后，可通过 DMA 中断或者 SPI 自身中断判断结束。

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

1

识。

- 接收模式下，SPI 接收完毕，触发 DMA 搬移。DMA 搬移完成，可作为本次传输完成标识。

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

17.3.2.10波特率设置

SPI 接口时钟通过对系统时钟分频获得，分频系数来自 BAUD[5:0]。分频范围是 1 ～128，对应的 BAUD[5:0]的值为 0 ～63。

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，上升沿采用数据。

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 BAUD 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。

17.4 寄存器

17.4.1 地址分配

SPI 模块寄存器的基地址是 0x4001_0000，寄存器列表如下:

表 17-1 SPI 模块控制寄存器列表

名称	偏移	说明
SPI_CFG	0x00	SPI 配置寄存器
SPI_IE	0x04	SPI 中断寄存器
SPI_DIV	0x08	SPI 波特率寄存器
SPI_TX_DATA	0x0C	SPI 发送数据寄存器
SPI_RX_DATA	0x10	SPI 接收数据寄存器
SPI_SIZE	0x14	SPI 传输数据长度寄存器

17.4.2 SPI_CFG SPI 控制寄存器

地址：0x4001_0000

复位值：0x0

表 17-2 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	SAMPLE	CLK_POL	ENDIAN	EN
								RW		RW	RW	RW	RW	RW	RW
								0		1	0	0	1	0	0

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

SPI_CFG[3:2]对应的通讯波形极性和相位可参考 17.3.2.4。

17.4.3 SPI_IE SPI 中断寄存器

地址：0x4001_0004

复位值：0x0

表 17-3 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。 1：使能 SPI 中断

		0：关闭 SPI 中断
[6]	CMPLT_IE	SPI 传输，完成事件中断使能信号。 1：使能此中断源 0：屏蔽此中断源
[5]	AB_IE	SPI 传输，异常事件中断使能信号。 1：使能此中断源 0：屏蔽此中断源
[4]	OV_IE	SPI 传输，溢出事件中断使能信号。默认值为 0。 1：使能此中断源 0：屏蔽此中断源
[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。包含两种情况： 1. 前次接收的旧数据没有被取得走，本次接收的新数据已经到达。 2. 本次发送的数据已发送完毕，新数据没有准备好。 高电平有效，写 1 清除。

17.4.4 SPI_DIV SPI 波特率寄存器

地址：0x4001_0008

复位值：0x0

表 17-4 SPI_DIV 控制寄存器

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

位置	位名称	说明
[31:8]		未使用
[7]	TRANS_MODE	SPI 数据搬移方式。默认为 0，DMA 方式。 0：SPI 接口支持 DMA 搬移数据到 SPI 接口，完成发送和接收。 1：SPI 接口支持 CPU 搬移数据到 SPI 接口，完成发送和接收。
[6]		未使用
[5:0]	BAUD	SPI 传输波特率配置，SPI 实际传输速度计算公式为： SPI 传输速度 = 系统时钟 / (2*(BAUD + 1)) 切记，BAUD 的配置值不能小于 3。

17.4.5 SPI_TX_DATA SPI 数据发送寄存器

地址：0x4001_000C

复位值：0x0

表 17-5 SPI_TX_DATA 数据发送寄存器

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

位置	位名称	说明
[31:8]		未使用
[7:0]	TX_DATA	SPI 数据发送寄存器。CPU 方式，寄存器写入要发送的数据；DMA 方式，对寄存器执行写入操作，仅仅是触发 DMA 搬移，要发送的数据来自 DMA，非本寄存器。

17.4.6 SPI_RX_DATA SPI 数据接收寄存器

地址：0x4001_0010

复位值：0x0

表 17-6 SPI_RX_DATA 数据接收寄存器

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

位置	位名称	说明
[31:8]		未使用
[7:0]	RX_DATA	SPI 数据接收寄存器

17.4.7 SPI_SIZE SPI 数据传输长度寄存器

地址：0x4001_0014

复位值：0x0

表 17-7 SPI_SIZE 数据传输长度寄存器

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

	0

位置	位名称	说明
[31:8]		未使用
[7:0]	SIZE	SPI 数据传输长度寄存器。CPU 方式，只能写入 1，一次传输一个字节的数据，传输完成后 SIZE 变成 0；DMA 方式，表示本次传输的数据总量，硬件每传输完成一个字节，SIZE 自减 1，传输完成后 SIZE 变成 0。

18 CMP

18.1 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)，用于处理两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。

比较器可用于以下功能:

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

比较器的主要特性如下:

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

需说明的是，CMP负输入端的BEMFx_MID信号，是对正输入端信号CMPx_IP1/ CMPx_IP2/ CMPx_IP3信号的平均，具体连接方式见图18-1，图中电阻R=8.2k欧，图中的开关只有在比较器负输入端信号选择为BEMFx_MID之后才会导通，否则开关都处于断开状态。

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

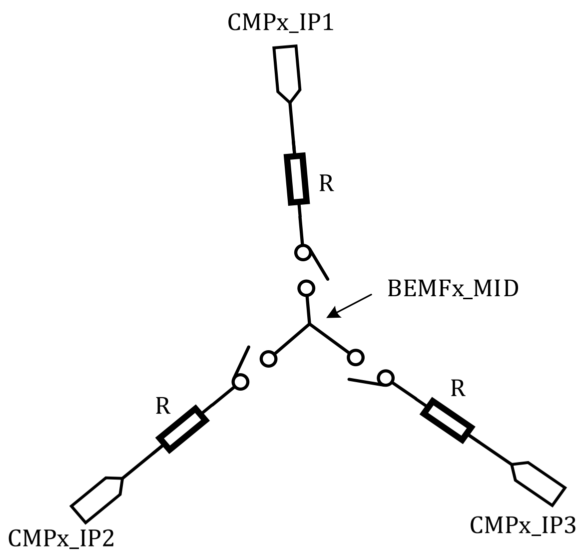


图 18-1 BEMFx_MID 信号

此模块的信号处理时钟由系统主时钟分频得到。模拟比较器未经滤波的原始输出值，可以通过读取寄存器 SYS_AFE_CMP 值获得，同时也可以通过配置 GPIO 的第二功能通过 P0.14 和 P2.3 送出，具体 GPIO 第二功能配置及引进位置，请参考器件 datasheet。

推荐配置流程：

1. 配置 CMP 的模拟开关

   芯片上电的默认状态下，比较器模块是关闭的，通过配置寄存器 SYS_AFE_REG5.CMPxPDN 可以打开比较器(x=0/1,代表 CMP0/CMP1 两个比较器)。开启比较器之前，需要先开启 BGP 模块。

2. 配置 CMP 的数字时钟开关、时钟滤波和信号输入开关

   通过配置寄存器 CMP_TCLK.CLK_EN 可以打开 CMP 的数字时钟开关，1 表示打开，0 表示关闭；通过配置寄存器 CMP_TCLK.FIL_CLK_DIV16[7:4]可以配置滤波时钟，数值设置范围是 0~15，0 表示 1 分频滤波，15 表示 16 分频滤波；通过配置寄存器 CMP_CFG.CMPx_IN_EN 可以打开信号输入开关，1 表示打开，0 表示关闭。

3. 配置 CMP 的比较速度和迟滞电压

   比较器的比较延时可通过配置寄存器 SYS_AFE_REG1.IT_CM[1:0] 设置为 0.15uS/0.6uS。迟滞电压可通过配置寄存器 SYS_AFE_REG3.CMP_HYS 设置为 20mV/0mV。

4. 选择 CMP 的正负端信号来源

   CMP 的正端信号有 8 种信号来源可选择，可以通过配置寄存器 SYS_AFE_REG3.CMPx_SELP[2:0]进行设置，负端有 4 种信号来源可选择，可以通过配置寄存器 SYS_AFE_REG3.CMPx_SELN[1:0]进行设置。

5. 配置 CMP 的中断

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

18.2 寄存器

18.2.1 地址分配

CMP 模块所涉及的寄存器包括两部分，一是 18.1 中提到的模拟配置寄存器，二是 CMP_IE 等 CMP 模块寄存器。

CMP 模块寄存器的基地址是 0x4001_0C00，寄存器列表如下:

表 18-1 比较器模块寄存器列表

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

18.2.2 CMP 模块寄存器描述

18.2.2.1 CMP_IE

地址：0x4001_0C00

复位值：0x0

表 18-2 比较器中断使能寄存器 CMP_IE

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

位置	位名称	说明
[31:2]		未使用
[1]	CMP1_IE	比较器 1 中断使能，高有效
[0]	CMP0_IE	比较器 0 中断使能，高有效

18.2.2.2 CMP_IF

地址：0x4001_0C04

复位值：0x0

表 18-3 比较器中断标志寄存器 CMP_IF

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

	CMP1_IF	CMP0_IF
	RW1C	RW1C
	0	0

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

18.2.2.3 CMP_TCLK

地址：0x4001_0C08

复位值：0x0

表 18-4 比较器分频时钟控制寄存器 CMP_TCLK

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								FIL_CLK_DIV16				CLK_EN		FIL_CLK_DIV1248	
								RW				RW		RW	
								0				0		0	

位置	位名称	说明
[31:8]		未使用
[7:4]	FIL_CLK_DIV16	比较器滤波时钟分频，基于 MCLK 进行 1~16 分频，影响进入比较器中断的时间
[3]	CLK_EN	时钟使能，高有效
[2]		未使用
[1:0]	FIL_CLK_DIV1248	比较器滤波时钟分频，2'b00:1 分频, 2'b01:2 分频, 2'b10:4 分频, 2'b11:8 分频

CMP 滤波时钟频率

Freq(CMP_Filter) = Freq(MCLK)/$2^{\mathrm{CMP\_TCLK.FIL\_CLK\_DIV1248}}$/(CMP_TCLK.FIL_CLK_DIV16+1)，其中 MCLK 为系统的主时钟，通常为 96MHz 全速时钟。需要注意的是，产生 CMP 滤波时钟需要使能 CMP_TCLK.CLK_EN 位。

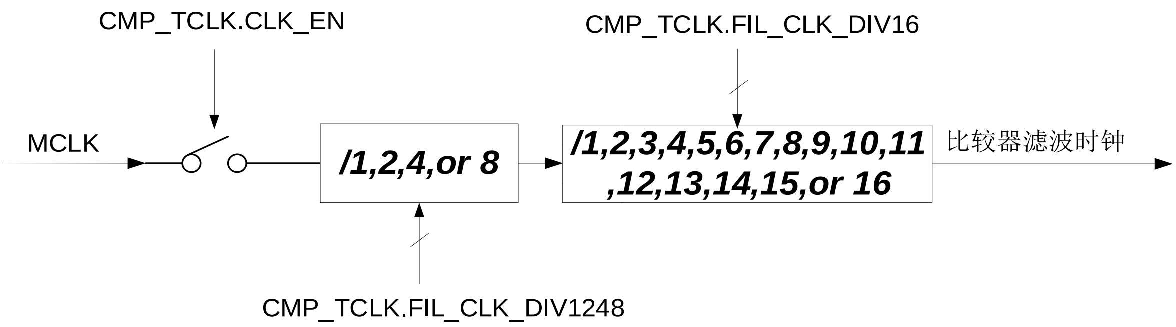


图 18-2 比较器滤波时钟产生

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**.

18.2.2.4 CMP_CFG

地址：0x4001_0C0C

复位值：0x0

表 18-5 比较器控制寄存器 CMP_CFG

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

位置	位名称	说明
[31:8]		未使用
[7]	CMP1_W_PWM_POL	比较器 1 开窗 PWM 信号极性选择，在 CMP_BLCWIN 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_BLCWIN 使能情况下使用
[2]	CMP0_IRQ_TRIG	比较器 0 中断触发类型，0:电平触发，1:边沿触发
[1]	CMP0_IN_EN	比较器 0 信号输入使能
[0]	CMP0_POL	比较器 0 极性选择，0:高电平有效；1:低电平有效

比较器的极性及使能控制如图 18-3 所示。

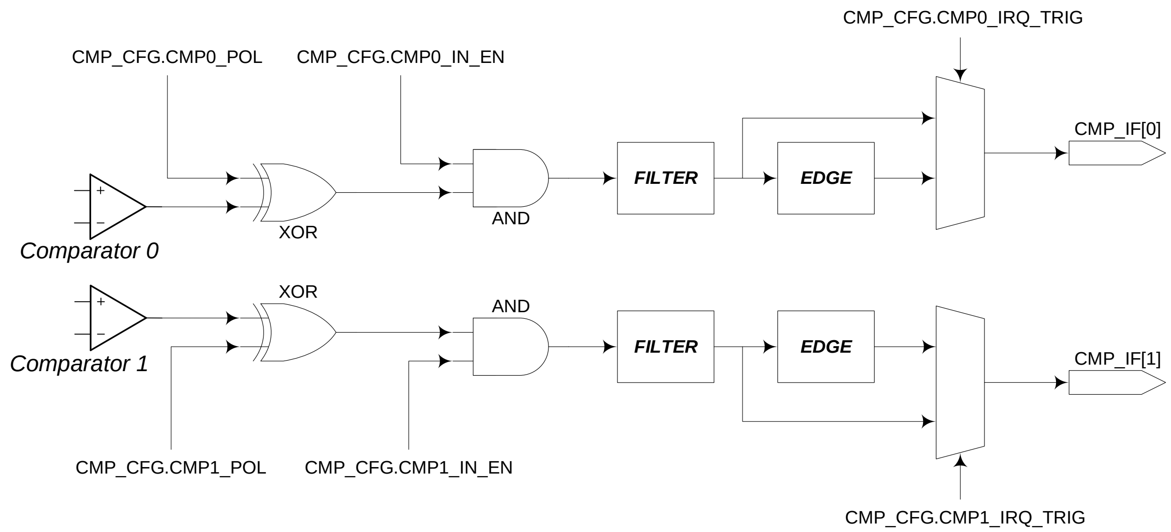


图 18-3 比较器控制及中断产生逻辑

比较器模块与 MCPWM 模块可以联合动作，其中 MCPWM 模块的 P 管控制信号可以作为比较器开窗的控制信号。但比较器自身的中断信号产生于开窗控制无关，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。

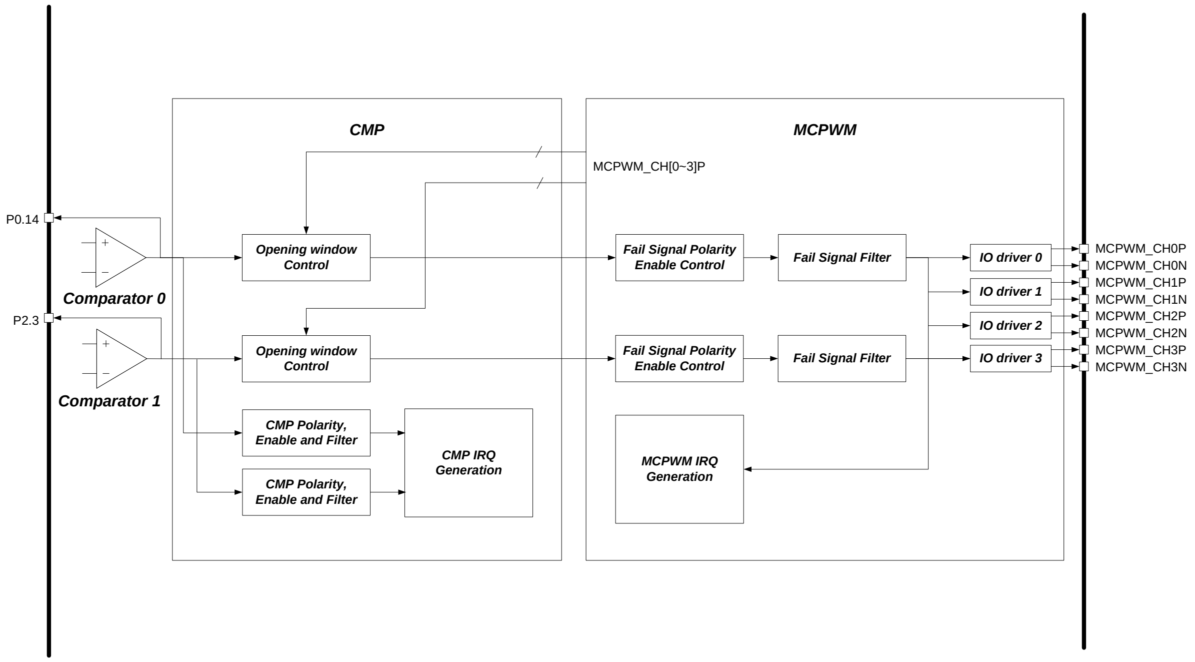


图 18-4 CMP 与 MCPWM 的联动

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_W_PWM_POL 和 CMP_CFG.CMP1_W_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 18-5 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。

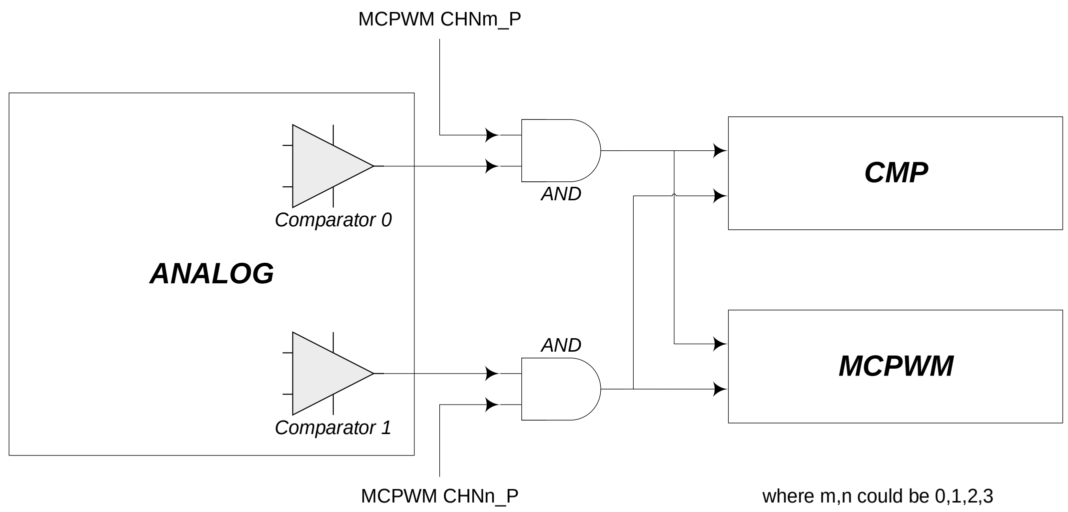


图 18-5 比较器开窗功能图示

18.2.2.5 CMP_BLCWIN

地址：0x4001_0C10

复位值：0x0

表 18-6 比较器开窗控制寄存器 CMP_ BLCWIN

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

位置	位名称	说明
[31:8]		保留
[7]	CMP1_CHN3P_WIN_EN	使用 MCPWM 模块 CHN3_P 通道输出的 P 管开关控制信号作为比较器 1 开窗使能
[6]	CMP1_CHN2P_WIN_EN	使用 MCPWM 模块 CHN2_P 通道输出的 P 管开关控制信号作

		为比较器 1 开窗使能
[5]	CMP1_CHN1P_WIN_EN	使用 MCPWM 模块 CHN1_P 通道输出的 P 管开关控制信号作为比较器 1 开窗使能
[4]	CMP1_CHN0P_WIN_EN	使用 MCPWM 模块 CHN0_P 通道输出的 P 管开关控制信号作为比较器 1 开窗使能
[3]	CMP0_CHN3P_WIN_EN	使用 MCPWM 模块 CHN3_P 通道输出的 P 管开关控制信号作为比较器 0 开窗使能
[2]	CMP0_CHN2P_WIN_EN	使用 MCPWM 模块 CHN2_P 通道输出的 P 管开关控制信号作为比较器 0 开窗使能
[1]	CMP0_CHN1P_WIN_EN	使用 MCPWM 模块 CHN1_P 通道输出的 P 管开关控制信号作为比较器 0 开窗使能
[0]	CMP0_CHN0P_WIN_EN	使用 MCPWM 模块 CHN0_P 通道输出的 P 管开关控制信号作为比较器 0 开窗使能

通常 CMP_ BLCWIN[3:0]或 CMP_ BLCWIN[7:4]中有 1bit 为 1，表明使用对应的 CHNx_P 对比较器 0/1 的信号产生进行控制。如果 CMP_ BLCWIN[3:0]或 CMP_ BLCWIN[7:4]为 4’b0000，则表示比较器 0/1 比较信号的产生与 PWM 信号无关。

19 CAN

19.1 概述

CAN 总线接口连接微控制器和串行 CAN 总线。本 CAN 模块根据特定设备的需要，可以使用 DMA 以减轻 CPU 的负担。使用 CAN 通信时需要用外部晶振作为参考时钟。

19.1 主要特性

支持 BOSCH 2.0A 和 2.0B 协议。2.0A 等效 CAN1.2，包含了 11 位 ID 格式；2.0B 包含了 11 位 ID 和 29 位 ID。

- 支持 SJA1000 大部分功能（部分不支持点在文档后续有说明）。
- 有两种模式：工作模式和复位模式。
- 支持监听和自测试。监听模式下，仅接收外界信号但不会返回任何信号。
- 支持 DMA 功能。

19.2 功能描述

19.2.1 功能框图

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

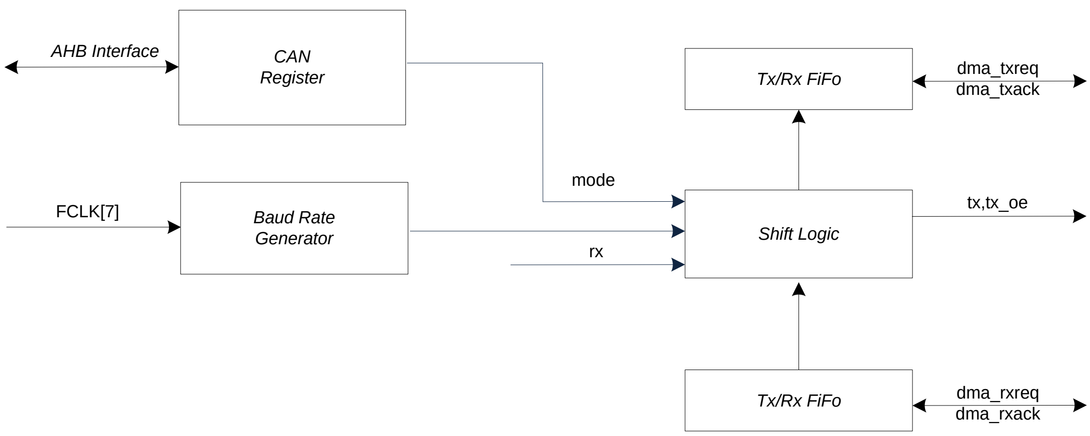


图 19-1 CAN 模块顶层功能框图

CAN 接口同外界通讯只有 tx 和 rx 两根信号线。tx 发送有效数据时候，tx_oe 有效。tx 不发送有效数据的时候，tx_oe 无效。

rx：数据信号。接收来自外界的 CAN 数据。

tx：数据信号。发送 CAN 数据到 CAN 总线。

tx_oe：数据使能信号。当 tx 输出时，tx_oe 有效；当 tx 无数据输出时，tx_oe 无效。

19.2.2 功能说明

CAN 模块接收和发送数据，并将数据从串行转换成并行，或并行转换成串行。可以开启或禁止中断。接口通过数据输出引脚(TX)和数据输入引脚(RX)连接到 CAN 总线的 PHY 芯片上。

19.2.2.1 工作模式

CAN 模块主要包含两个工作模式：正常工作模式和复位模式。

复位模式，时序参数、ID 配置、错误统计值等均在此模式下设定好。CAN_MOD.0 为 1，是复位模式。硬件复位结束后，CAN 模块处于复位模式下。

正常工作模式，CAN_MOD.0 为 0。此时，可以正常响应 CAN 总线请求。

在上述两个模式下，扩展了监听模式（Listen Only）和自测试模式（Self Test）。前者，类似一个搜集器，只接收 CAN 总线上的数据，不发送任何数据。后者，是内部自测试，发送的数据同时被自己接收，检测内部功能是否正确。

19.2.2.2 DMA 传输

在大容量数据传输应用下，CAN 接口支持 DMA 传输，减轻 CPU 的负担。一次传输，SFF 为 11 个字节，EFF 为 13 个字节。

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

DMA 传输需要配置 DMA 模块相应寄存器。

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

DMA 传输，推荐软件配置流程如下：

- 初始化 DMA 模块，将本次发送的数据来源，接收的数据去向配置好，传输长度配置完毕。
- 初始化 GPIO 模块，将 CAN 复用的 GPIO 配置完毕。
- 初始化 CAN 接口，控制寄存器配置完毕。
- 触发 CAN 接口，进入发送状态。

本 CAN 模块设计的 DMA 不同于其它模块的 DMA 搬移操作，需要 CPU 介入部分搬移操作。假定当前配置 CAN 模块发送 N 帧数据，那么第一帧数据需要 CPU 搬移到 CAN 模块寄存器中，后续帧（N-1）的数据可由 DMA 实现搬移。

19.2.2.3 CPU 传输

CPU 传输，根据 SFF 还是 EFF，搬移 11 个字节或者 13 个字节；完成后需要通过中断或者轮询的方式判断传输是否完成。

CPU 传输，推荐软件配置流程如下:

- 初始化 GPIO 模块，将 CAN 复用的 GPIO 配置完毕。
- 初始化 CAN 接口，控制寄存器配置完毕。
- CPU 触发 CAN 接口进入发送流程，发送的数据来自 CPU 对 CAN_TXDATA 写入值。

19.2.2.4 中断处理

CAN 接口包含中断事件比较多，在 CAN_IR 和 CAN_EIR 寄存器描述中有相应说明。根据实际使用情况，开启对应的中断事件使能开关。

CAN 控制器提供了以下七种中断:

- 接收中断
- 发送中断
- 错误报警中断
- 数据溢出中断
- 被动错误中断
- 仲裁丢失中断
- 总线错误中断

只要在 CAN_IR 一个或多个中断位为 1，CAN 控制器中的中断信号即为有效，当 CAN_IR 中的所有位都被清除时，CAN 控制器中的中断信号则失效。寄存器 CAN_IR 被读取后，其中的大多数中断位将自动清除。但是，接收中断不包括在内，直到通过 CAN_CMR 的 bit2 写 1，读取所有接收报文后，接收中断位才能被清除。

19.2.2.4.1 接收中断

当 CAN 接收 FIFO 中有待读取帧时（CAN_RMC > 0），都会触发接收中断。CAN_RMC 中记录的帧数量包括接收 FIFO 中的有效帧和溢出帧。直到通过给 CAN_CMR 的 bit2 写 1，清除所有挂起接收帧后，接收中断才会失效。

19.2.2.4.2 发送中断

每当发送缓冲器空闲，将其他帧加载到发送缓冲器中等待发送时，都会触发发送中断。以下情况下，发送缓冲器将变为空闲，同时发送中断将失效:

- 帧发送已成功完成（如，应答未发现错误）。任何发送失败将自动重发。
- 单次发送已完成（CAN_SR bit3 指示发送成功与否）。
- CAN_CMR 的 bit1 位，写 1，终止帧发送。

19.2.2.4.3 错误报警中断

每当寄存器 CAN_IR 中 ON_BUS 或 ERR_OV 的位值改变时（如，从 0 变为 1 或反之），都会触发

此中断。根据触发时 ON_BUS 或 ERR_OV 的值分成以下几种情况：

- 如果 ERR_OV = 0 且 ON_BUS = 0：

– 如果 CAN 控制器处于主动错误状态，则表示接收错误计数和发送错误计数的值都返回到了 CAN_EWLR 所设的阈值之下。

– 如果 CAN 控制器此前正处于总线恢复状态，则表示此时总线恢复已成功完成。

- 如果 ERR_OV = 1 且 ON_BUS = 0：则表示接收错误计数或发送错误计数的值超过了 CAN_EWLR 所设的阈值。

- 如果 ERR_OV = 1 且 ON_BUS = 1：表示 CAN 控制器已进入 BUS_OFF 状态（因发送错误计数 >= 256）。

- 如果 ERR_OV = 0 且 ON_BUS = 1：表示 BUS_OFF 恢复期间，CAN 控制器的发送错误计数的数值已低于 CAN_EWLR 所设的阈值。

19.2.2.4.4　　数据溢出中断

接收 FIFO 满，若还收到有效帧，将触发数据溢出中断。

19.2.2.4.5　　仲裁丢失中断

每当 CAN 控制器尝试发送帧且丢失仲裁时，都会触发此中断。CAN 控制器丢失仲裁的 bit 位置将自动记录在仲裁丢失捕捉寄存器 CAN_ALC 中。仲裁丢失捕捉寄存器被清除（通过 CPU 读取该寄存器）之前，将不会再记录新发生的仲裁失败时的 bit 位置。

19.2.2.4.6　　总线错误中断

每当 CAN 控制器在 CAN 总线上检测到错误时，都会触发此中断。发生总线错误时，总线错误的类型和发生错误时的 bit 位置都将自动记录在错误捕捉寄存器 CAN_ECC 中。错误捕捉寄存器被清除（通过 CPU 的读取）之前，将不会再记录新的总线错误信息。

19.2.2.5 通讯采样设置

CAN 的波特率设定，主要依靠 CAN_BTR0 和 CAN_BTR1 两个寄存器配合完成。CAN_BTR0 主要是配置 TQ 参数（TQ 的计算见 BTR0 寄存器说明），CAN_BTR1 主要处理 1-bit 数据的采样点、采样次数以及宽度信息。

CAN_BTR0 设置传输基本时间单元参数 TQ：TQ = 2 * Tclk * (CAN_BTR0.BAUDRATE + 1)

LKS08x 时钟最快为 96M，对应的 Tclk 为 10.4ns，TQ 最大值为 1.3312us。

CAN_BTR1 设置波特率；同时，可调节 BIT 信息中各个部分的宽度（TSEG1、TSEG2 和 Sync.Seg），找到理想采样点。

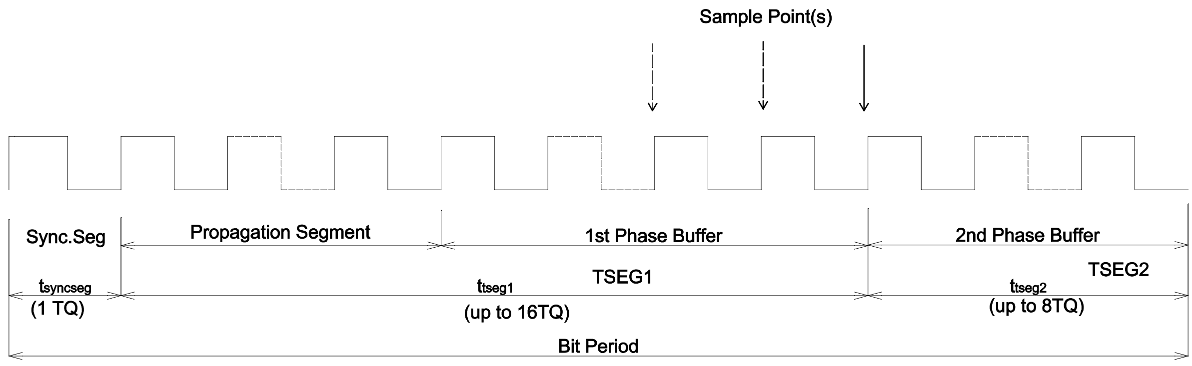


图 19-2 CAN 模块 Bit 周期介绍图

SEG1 段时间计算公式：Tseg1 = TQ*(CAN_BTR1.SEG1 + 1)

SEG2 段时间计算公式：Tseg2 = TQ*(CAN_BTR1.SEG2 + 1)

波特率计算公式为：**Can Baudrate =1/(TQ * (1 + Tseg1 + Tseg2))**

CAN_BTR0.SJW 是 SJW 位，其为容差范围的配置寄存器。在一定波特率下通讯的各个设备，允许多大的通讯时间误差存在。

总线下界容差 < 总线波特率 < 总线上限容差

容差计算公式为：TQ * (SJW + 1)

CAN_BTR1.SAM 是 SAM 位，其位采样次数配置寄存器。0：一次；1：三次。根据实际使用情况配置即可，此位不参与波特率计算。

基于 LKS08x 芯片，常规波特率索引值如下：

Can 波特率	BTR0	BTR1
1Mbps	0x05	0x14
800Kbps	0x05	0x16
666Kbps	0x85	0xB6
500Kbps	0x05	0x1C
400Kbps	0x05	0xFA
250Kbps	0x0B	0x1C
200Kbps	0x0B	0xFA
125Kbps	0x17	0x1C
100Kbps	0x1D	0x1C
83.33Kbps	0x17	0x6F
80Kbps	0x97	0xFF
66.66Kbps	0x1D	0x6F
50Kbps	0x3B	0x1C
40Kbps	0xAF	0xFF

理论上，可以通过配置寄存器实现 1-bit 数据的时间宽度在 3TQ 到 25TQ 之间。实际应用中，

如果遵循 BOSCH 标准，1-bit 数据的时间宽度在 8TQ 到 25TQ 之间。若不遵循 BOSCH 标准，实际可达的范围在 4TQ 到 25TQ 之间。

19.2.2.6 ID 滤波

CAN 总线，可以挂载很多设备。通过 ID 号，不同设备可以知道当前总线上发送的帧是否需要被自己接收，或者发送出去的帧，是否有响应。

接收滤波器允许 CAN 控制器根据帧 ID 过滤接收帧（有时可以过滤帧的第一个数据字节和帧类型）。只有通过过滤的帧才能存储到接收 FIFO 中。接收滤波器的使用可以一定程度地减轻 CAN 控制器的运行负荷（如，可减少使用接收 FIFO 和发生接收中断的次数），因为 CAN 控制器将只需要操作一小部分过滤后的帧。只有当 CAN 控制器处于复位模式时，才可以访问接收滤波器的配置寄存器。

CAN 帧的 ID 号码，有两种长度--11 位和 29 位。前者对应的 SFF（standard frame format），后者对应 EFF（extended frame format）。通过 CAN_ACR 和 CAN_AMR 判断当前 CAN 模块可接收的 ID 范围。CAN_ACR 列出了一个特定的 ID，CAN_AMR 为 MASK 寄存器，标识 CAN_ACR 中对应位数据同接收到的 ID 对应位数据需完全匹配，哪些位可以不用。极端情况就是 CAN_ACR 上每一位都要匹配，或者每一位都不要匹配。CAN_AMR 对应位是 0，表示接收到的 ID 对应位需要同 CAN_ACR 对应位匹配；CAN_AMR 对应位是 1，表示接收到的 ID 对应位不需要同 CAN_ACR 对应位匹配。

CAN_MOD.3 决定了 CAN_ACR 包含两个滤波 ID 还是一个滤波 ID。为 1 的话，CAN_ACR 包含一个长的滤波 ID；为 0 的话，CAN_ACR 包含了两个短的滤波 ID。两个滤波 ID 情况下，只要接收帧的 ID 同其中一个匹配，就会被 CAN 接收。

SFF，单滤波 ID 数据格式

CAN_TXRX1	CAN_TXRX2	CAN_TXRX3	CAN_TXRX4
ID.28..ID21	ID.20..ID.18 RTR X X X X	Data Byte 1	Data Byte 2

滤波 ID 设置

ACR0[7:0]	ACR1[7:4] (ACR1[3:0] unused)	ACR2[7:0]	ACR3[7:0]
AMR0[7:0]	AMR1[7:4] (AMR1[3:0] unused)	AMR2[7:0]	AMR3[7:0]

此时， ACR2 和 ACR3 若没有用到，对应的 AMR2 和 AMR3 需配置为全 1。RTR 是否需要匹配，结合实际情况配置 AMR。

SFF，双滤波 ID 数据格式

CAN_TXRX1	CAN_TXRX2	CAN_TXRX3	CAN_TXRX4
ID.28..ID21	ID.20..ID.18 RTR X X X X	Data Byte 1	Data Byte 2

滤波 ID1

ACR0[7:0]	ACR1[7:0]	ACR3[3:0]
AMR0[7:0]	AMR1[7:0]	AMR3[3:0]

滤波 ID2

ACR2[7:0]	ACR3[7:4]
AMR2[7:0]	AMR3[7:4]

此时， ACR 若没有用到，对应的 AMR 需配置为全 1。例如滤波 ID1 的 ACR1[3:0]和 ACR[3:0]。

RTR 是否需要匹配，结合实际情况配置 AMR。

EFF，单滤波 ID 数据格式

CAN_TXRX1	CAN_TXRX2	CAN_TXRX3	CAN_TXRX4
ID.28..ID21	ID.20..ID.13	ID.12..ID.5	ID.4..ID.0 RTR X X

滤波 ID

ACR0[7:0]	ACR1[7:0]	ACR2[7:0]	ACR3[7:2]
AMR0[7:0]	AMR1[7:0]	AMR2[7:0]	AMR3[7:2]

此时， ACR3[1:0]没有用到，对应的 AMR 需配置为全 1。

EFF，双滤波 ID 数据格式

CAN_TXRX1	CAN_TXRX2	CAN_TXRX3	CAN_TXRX4
ID.28..ID21	ID.20..ID.13	ID.12..ID.5(not matched)	ID.4..ID.0 RTR X X(not matched)

滤波 ID1

ACR0[7:0]	ACR1[7:0]
AMR0[7:0]	AMR1[7:0]

滤波 ID2

ACR2[7:0]	ACR3[7:0]
AMR2[7:0]	AMR3[7:0]

此时， 实际接收的 ID12—ID5，不同 ACR 比较。

ACR 和 AMR 的关系如下：

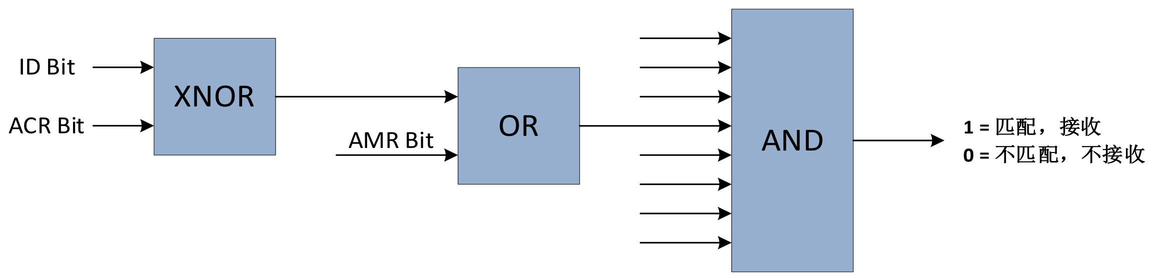


图 19-3 CAN 模块 ID 滤波逻辑关系

19.2.2.7 发送帧格式

发送帧分成 ID 部分和数据部分。第一个字节包含了帧分类等信息，确定是 SFF（标准）帧还是 EFF（扩展）帧；确定是远程帧，还是数据帧；以及，确定数据长度。SFF 的 ID 长度为 2 个字节，EFF 的 ID 长度为 4 个字节。数据长度为最大 8 个字节。

表 19-1 发送帧结构

SFF		EFF	
地址	域	地址	域
0x40	TX 帧信息	0x40	TX 帧信息
0x44	TX ID0	0x44	TX ID0
0x48	TX ID1	0x48	TX ID1
0x4C	TX DATA0	0x4C	TX ID2
0x50	TX DATA1	0x50	TX ID3
0x54	TX DATA2	0x54	TX DATA0
0x58	TX DATA3	0x58	TX DATA1
0x5C	TX DATA4	0x5C	TX DATA2
0x60	TX DATA5	0x60	TX DATA3
0x64	TX DATA6	0x64	TX DATA4
0x68	TX DATA7	0x68	TX DATA5
0x6C	unused	0x6C	TX DATA6
0x70	unused	0x70	TX DATA7

SFF 帧非数据信息

表 19-2 发送 SFF 头信息

CAN Address	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
0x40	FF	RTR	X1	X1	DLC.3	DLC.2	DLC.1	DLC.0
0x44	ID.28	ID.27	ID.26	ID.25	ID.24	ID.23	ID.22	ID.21

0x48	ID.20	ID.19	ID.18	X2	X1	X1	X1	X1

EFF 帧非数据信息

表 19-3 发送 EFF 头信息

CAN Address	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
0x40	FF	RTR	X1	X1	DLC.3	DLC.2	DLC.1	DLC.0
0x44	ID.28	ID.27	ID.26	ID.25	ID.24	ID.23	ID.22	ID.21
0x48	ID.20	ID.19	ID.18	ID.17	ID.16	ID.15	ID.14	ID.13
0x4C	ID.12	ID.11	ID.10	ID.9	ID.8	ID.7	ID.6	ID.5
0x50	ID.4	ID.3	ID.2	ID.1	ID.0	X2	X1	X1

- FF：1 表示是 EFF（扩展）帧，0 表示是 SFF（标准）帧。
- RTR：1 标识是 remote（远程）帧，0 表示是 data（数据）帧。
- DLC：表示此帧要发送数据的长度。最大为 8 个字节，最小为 0 个字节。
- ID：帧标识号。SFF 帧的 ID 长度为 11 位（ID.28 到 ID.18）。EFF 帧的 ID 长度为 29 位（ID.28 到 ID.0）。高位优先发送。
- DATA：数据。字节间顺序，从大到小，即 TX DATA7 先发送。字节内顺序，从高位到低位。
- X2：最好同 RTR 值一致
- X1：1 或 0 均可

19.2.2.8 接收帧格式

接收帧分成 ID 部分和数据部分。第一个字节包含了帧分类等信息，确定是 SFF（标准）帧还是 EFF（扩展）帧；确定是远程帧，还是数据帧；以及，确定数据长度。SFF 的 ID 长度为 2 个字节，EFF 的 ID 长度为 4 个字节。数据长度为最大 8 个字节。

表 19-4 接收帧结构

SFF		EFF	
地址	域	地址	域
0x40	RX 帧信息	0x40	RX 帧信息
0x44	RX ID0	0x44	RX ID0
0x48	RX ID1	0x48	RX ID1
0x4C	RX DATA0	0x4C	RX ID2
0x50	RX DATA1	0x50	RX ID3
0x54	RX DATA2	0x54	RX DATA0
0x58	RX DATA3	0x58	RX DATA1
0x5C	RX DATA4	0x5C	RX DATA2
0x60	RX DATA5	0x60	RX DATA3
0x64	RX DATA6	0x64	RX DATA4

0x68	RX DATA7	0x68	RX DATA5
0x6C	unused	0x6C	RX DATA6
0x70	unused	0x70	RX DATA7

SFF 帧非数据信息

表 19-5 接收 SFF 头信息

CAN Address	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
0x40	FF	RTR	0	0	DLC.3	DLC.2	DLC.1	DLC.0
0x44	ID.28	ID.27	ID.26	ID.25	ID.24	ID.23	ID.22	ID.21
0x48	ID.20	ID.19	ID.18	RTR	0	0	0	0

EFF 帧非数据信息

表 19-6 接收 EFF 头信息

CAN Address	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
0x40	FF	RTR	0	0	DLC.3	DLC.2	DLC.1	DLC.0
0x44	ID.28	ID.27	ID.26	ID.25	ID.24	ID.23	ID.22	ID.21
0x48	ID.20	ID.19	ID.18	ID.17	ID.16	ID.15	ID.14	ID.13
0x4C	ID.12	ID.11	ID.10	ID.9	ID.8	ID.7	ID.6	ID.5
0x50	ID.4	ID.3	ID.2	ID.1	ID.0	RTR	0	0

19.2.2.9 发送

CAN 模块要发送数据，必须在正常工作模式下。

执行发送操作，一般推荐开启发送中断源和错误警告数量中断源，而仲裁丢失和总线错误中断源头不强迫开启，CAN 模块有自动重发机制。

发送的数据，存放在 TX FIFO 中，TX FIFO 为 32-Byte，及一次可以存几帧数据，且 TX FIFO 中的数据一旦被成功发送出去，就被 CAN 模块释放掉供新的数据写入。一帧数据长度为 13 个字节，其考虑到最大数据量为 8 个字节。在发送之前，需确保此时模块处于空闲状态。

发送帧的格式，分成标准帧（SFF）和扩展帧（EFF）。CAN_CMR 的 BIT0 置 1，触发 CAN 模块发送数据。一旦 CAN 总线空闲，数据就会发出。传输完毕与否，一方面可以检查 CAN_SR 的 BIT2 是否变成 1（1 表示空闲），也可以通过中断方式检查是否发送完毕。

在数据发送到总线前，设置 CAN_CMR 的 BIT1，可以停止发送，产生相应中断；通过 CAN_SR 的 BIT3，可知当前数据是否发送完毕。

19.2.2.10接收

CAN 模块要接收数据，必须在正常工作模式下。

接收 FIFO 中存储的帧大小可以不同（3 ~ 13 byte 范围之间）。当接 FIFO 为满时（或剩余的空间不足以完全存储一帧），将触发溢出中断，后续的接收帧将丢失，直到接收 FIFO 中清除出足够的存储空间。接收 FIFO 中的第一条帧将被映射到 13-byte 的接收缓冲器中，直到该帧被清除（通过释放接收缓冲器指令）。清除后，接收缓冲器将继续映射接收 FIFO 中的下一条帧，接收 FIFO 中上一条已清除帧的空间将被释放。

接收，开始于侦测到 Start 帧。接收到的数据，执行 ID 匹配操作:

匹配成功，存入 RX FIFO，RMC 递增。RX FIFO 为 32-Byte，即一次可以存几帧数据，且 RX FIFO 中的数据一旦被 DMA 成功取走，就被 CAN 模块释放掉供新的数据写入（非 DMA 模式，需要对 CAN_CMR BIT2 写 1 释放 RX FIFO）。一帧数据长度为 13 个字节，其考虑到最大数据量为 8 个字节。RX FIFO 中有几个有效帧，通过 CAN_AMC 寄存器可知。通过读取 CAN_TXRX 寄存器可以获得最先被接收到的数据帧。若 RX FIFO 满了，抛弃此帧，RX FIFO 不接收，同时产生溢出中断（使能前提下）。

匹配失败，存入 RX FIFO，RMC 不会递增。因此，数据也会被后续帧覆盖。若 RX FIFO 满了，抛弃此帧，RX FIFO 不接收，不产生溢出中断（使能前提下）。

19.2.2.11错误管理

CAN 协议要求每个节点中都包含发送错误计数和接收错误计数。这两个错误计数的数值决定了控制器当前的错误状态（如主动错误、被动错误、离线）。控制器将数值分别存储在 CAN_RXERR 和 CAN_TXERR 中，CPU 可随时进行读取。除了错误状态之外，控制器还提供错误报警限制的功能，这个功能可在 CAN 控制器进入被动错误状态之前，提醒用户，当前发生的严重总线错误。

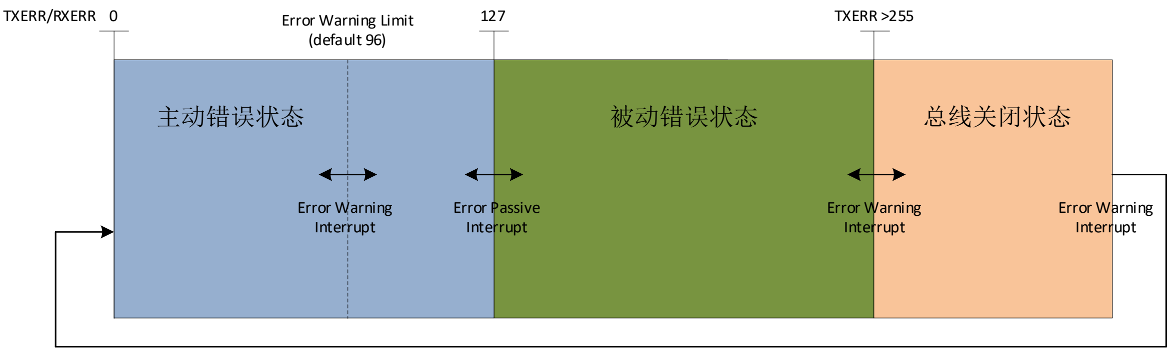


图 19-4 CAN 模块错误管理

若发生总线错误，总线错误中断标志位（CAN_IR. BUS_ERR_IF）置位，对应中断使能的话，将产生对应中断。当总线错误积累到一定数值，总线错误报警中断位（CAN_IR.W_ERR_IF）置位，对应中断使能的话，将产生对应中断；CAN_SR.ERR_OV 置位，表明错误超过警戒线。若继续增加，超过 127，进入被动错误状态，被动错误中断位（CAN_IR.M_ERR_IF）置位，对应中断使能的话，将产生对应中断。若总线错误继续增加（每次增加总线错误中断标志位都会置位），发送错误超过 255，CAN_SR.ON_BUS 置位进入总线关闭状态。此时，开始离线恢复，从 127 开始递减，减到 0 完成恢复，CAN_SR.ON_BUS 和 CAN_SR.ERR_OV 自动清除。

只要发生总线错误，总线错误中断标志位（CAN_IR.BUS_ERR_IF）均置位。

19.2.2.12错误计数

发送错误计数和接收错误计数根据以下规则递增/递减。请注意，一帧传输中可应用多个规则。

1. 当接收器检测到错误时，接收错误计数数值将增加 1。当检测到的错误为发送主动错误标志或过载标志期间的位错误除外。

2. 发送错误标志后，当接收器第一个检测到的位是显性位时，接收错误计数数值将增加 8。

3. 当发送器发送错误标志时，发送错误计数数值增加 8。但是，以下情况不适用于该规则：

- 发送器为被动错误状态，因为在应答槽未检测到显性位而产生应答错误，且在发送被动错误标志时检测到显性位时，则发送错误计数数值不应增加。

- 发送器在仲裁期间因填充错误而发送错误标志，且填充位本该是隐性位但是检测到显性位，则发送错误计数数值不应增加。

4. 若发送器在发送主动错误标志和过载标志时检测到位错误，则发送错误计数数值增加 8。

5. 若接收器在发送主动错误标志和过载标志时检测到位错误，则接收错误计数数值增加 8。

6. 任意节点在发送主动/被动错误标志或过载标志后，节点仅能承载最多 7 个连续显性位。在（发送主动错误标志或过载标志时）检测到第 14 个连续显性位，或在被动错误标志后检测到第 8 个连续显性位后，发送器将使其发送错误计数数值增加 8，而接收器将使其接收错误计数数值增加 8。每增加 8 个连续显性位的同时，（发送器的）发送错误计数和（接收器的）接收错误计数数值也将增加 8。

7. 每当发送器成功发送帧后（接收到 ACK，且直到 EOF 完成未发生错误），则发送错误计数数值将减小 1，除非发送错误计数的数值已经为 0。

8. 当接收器成功接收帧后（确认槽前未检测到错误，且成功发送 ACK），则接收错误计数数值将相应减小。

- 若接收错误计数数值位于 1～127 之间，则其值减小 1。

- 若接收错误计数数值大于 127，则其值减小到 127。

- 若接收错误计数数值为 0，则仍保持为 0。

9. 当一个节点的发送错误计数 和/或 接收错误计数数值大于等于 128 时，该节点变为被动错误节点。导致节点发生上述状态切换的错误，该节点仍发送主动错误标志。请注意，一旦接收错误计数数值到达 128，后续任何增加该值的动作都是无效的，直到接收错误计数数值返回到 128 以下。

10. 当某节点的发送错误计数数值大于等于 256 时，该节点将变为离线节点。

11. 当某被动错误节点的发送错误计数和接收错误计数数值都小于等于 127，则该节点将变为主动错误节点。

12. 当离线节点在总线上检测到 128 次 11 个连续隐性位后，该节点可变为主动错误节点（发送错误计数和接收错误计数数值都重设为 0）。

19.2.2.13错误报警限制

错误报警限制（CAN_EWLR）为发送和接收的可配置阈值，若错误计数数值超过该阈值，将触发 CAN 总线错误中断，作为一个报警功能提示当前发生的严重 CAN 总线错误，且在 CAN 控制器进入被动错误状态之前被触发。CAN_EWLR 寄存器存储相应阈值，此寄存器的配置必须处于复位模式下。CAN_EWLR 寄存器默认数值为 96。

当发送和/或接收错误的数值大于等于 CAN_EWLR 寄存器对应的数值时，CAN_IR.W_ERR_IF 位将立即被置 1。同理，当发送和/或接收错误的数值都小于 CAN_EWLR 寄存器对应的数值时，CAN_IR.W_ERR_IF 位将立即复位为 0。只要 CAN_IR.W_ERR_IF（CAN_EIR.W_ERR_IE 使能）位值变成 1，便会触发错误报警中断。

19.2.2.14被动错误

当发送或接收错误的数值大于 127 时，CAN 控制器处于被动错误状态，CAN_IR.M_ERR_IF 位将立即被置 1。同理，当发送或接收错误的数值都小于等于 127 时，CAN 控制器进入主动错误状态。每当 CAN 控制器从主动错误状态变为被动错误状态，或反之时，CAN_IR.M_ERR_IF 位都将立即被置 1。

19.2.2.15离线状态与离线恢复

当发送错误的数值大于 255 时，CAN 控制器将进入离线状态。进入离线状态后，CAN 控制器将自动进行以下动作：

- 接收错误计数器的数值置为 0
- 发送错误计数器的数值置为 127
- CAN_IR.BUS_ERR_IF 位置 1
- 进入 CAN 的复位模式

每当 CAN_IR.BUS_ERR_IF/CAN_IR.W_ERR_IF 的数值发生变化时，都将触发错误报警中断。

为了返回主动错误状态，CAN 控制器必须进行离线恢复。要启动离线恢复，首先需要退出复位模式，进入操作模式。然后要求 CAN 控制器在总线上检测到 128 次 11 个连续隐性位。每一次 CAN 控制器检测到 11 个连续隐性位时，发送错误计数器的数值都将减小，以追踪离线恢复进程。当离线恢复完成后（发送错误的数值从 127 减小到 0），CAN_IR.BUS_ERR_IF 位将自动复位为 0，从而触发错误报警中断。

19.2.3 寄存器

19.2.3.1 地址分配

CAN 模块寄存器的基地址是 0x4001_3400，寄存器列表如下。

表 19-7 CAN 寄存器地址分配

名称	偏移	说明
CAN_MOD	0x000	CAN 模式寄存器
CAN_CMR	0x004	CAN 命令寄存器

CAN_SR	0x008	CAN 状态寄存器
CAN_IR	0x00C	CAN 中断状态寄存器
CAN_IER	0x010	CAN 中断控制寄存器
CAN_BTR0	0x018	CAN 总线时序控制寄存器 0
CAN_BTR1	0x01C	CAN 总线时序控制寄存器 1
CAN_ALC	0x02C	CAN 仲裁丢失捕捉寄存器
CAN_ECC	0x030	CAN 错误码捕捉寄存器
CAN_EWLR	0x034	CAN 错误&警告门限值设置寄存器
CAN_RXERR	0x038	CAN 接收错误计数器
CAN_TXERR	0x03C	CAN 发送错误计数器
CAN_TXRX0	0x040	正常工作模式下，CAN 发送帧格式寄存器/CAN 接收帧格式寄存器
CAN_TXRX1	0x044	正常工作模式下，CAN 发送数据寄存器 0/CAN 接收数据寄存器 0
CAN_TXRX2	0x048	正常工作模式下，CAN 发送数据寄存器 1/CAN 接收数据寄存器 1
CAN_TXRX3	0x04C	正常工作模式下，CAN 发送数据寄存器 2/CAN 接收数据寄存器 2
CAN_TXRX4	0x050	正常工作模式下，CAN 发送数据寄存器 3/CAN 接收数据寄存器 3
CAN_TXRX5	0x054	正常工作模式下，CAN 发送数据寄存器 4/CAN 接收数据寄存器 4
CAN_TXRX6	0x058	正常工作模式下，CAN 发送数据寄存器 5/CAN 接收数据寄存器 5
CAN_TXRX7	0x05C	正常工作模式下，CAN 发送数据寄存器 6/CAN 接收数据寄存器 6
CAN_TXRX8	0x060	正常工作模式下，CAN 发送数据寄存器 7/CAN 接收数据寄存器 7
CAN_TXRX9	0x064	正常工作模式下，CAN 发送数据寄存器 8/CAN 接收数据寄存器 8
CAN_TXRXA	0x068	正常工作模式下，CAN 发送数据寄存器 9/CAN 接收数据寄存器 9
CAN_TXRXB	0x06C	正常工作模式下，CAN 发送数据寄存器 10/CAN 接收数据寄存器 10
CAN_TXRXC	0x070	正常工作模式下，CAN 发送数据寄存器 11/CAN 接收数据寄存器 11
CAN_ACR0	0x040	复位模式下，CAN ID 码寄存器 0
CAN_ACR1	0x044	复位模式下，CAN ID 码寄存器 1
CAN_ACR2	0x048	复位模式下，CAN ID 码寄存器 2
CAN_ACR3	0x04C	复位模式下，CAN ID 码寄存器 3
CAN_AMR0	0x050	复位模式下，CAN ID 掩码寄存器 0
CAN_AMR1	0x054	复位模式下，CAN ID 掩码寄存器 1
CAN_AMR2	0x058	复位模式下，CAN ID 掩码寄存器 2
CAN_AMR3	0x05C	复位模式下，CAN ID 掩码寄存器 3
CAN_RMC	0x074	CAN FIFO 有效接收信息计数器
CAN_RBSA	0x078	CAN 第一条有效接收信息在 FIFO 中的地址寄存器
CAN_RFIFO0~CAN_RFIFO31	0x080~0x144	CAN RX FIFO 地址寄存器
CAN_TFIFO0~CAN_TFIFO31	0x180~0x1B0	CAN TX FIFO 地址寄存器

19.2.3.2 寄存器说明

19.2.3.2.1 CAN_MOD 模式寄存器

地址：0x4001_3400

复位值：0x0

表 19-8 模式寄存器 CAN_MOD

15-5	4	3	2	1	0
	SLEEP	FLT_ID	TEST_MODE	FUNC_MODE	MODE
	RW	RW	RW	RW	RW
	0	0	0	0	1

位置	位名称	说明
[31:5]		未使用
[4]	SLEEP	正常工作模式下写 1，触发休眠操作，进入休眠模式。
[3]	FLT_ID	CAN 滤波 ID 选择。默认值为 0。 1：单滤波 ID。一个 32 位长的滤波 ID 0：双滤波 ID。两个短滤波 ID
[2]	TEST_MODE	CAN 工作模式选择。默认值为 0。 1：自测试模式 0：正常工作模式
[1]	FUNC_MODE	CAN 工作模式选择。默认值为 0。 1：监听模式 0：正常工作模式
[0]	MODE	CAN 工作模式选择。默认值为 1。 1：复位模式 0：正常工作模式

19.2.3.2.2 CAN_CMR 命令寄存器

地址：0x4001_3404

复位值：0x0

表 19-9 命令寄存器 CAN_CMR

15-6	5	4	3	2	1	0
	DMA_EN	RX_DUR_TX	CLR_OV	RELEASE_FIFO	INTR_TRANS	TRANS_REQ
	WO	WO	WO	WO	WO	WO
	0	0	0	0	0	0

位置	位名称	说明

[31:6]		未使用
[5]	DMA_EN	写 1，使能 DMA 功能
[4]	RX_DUR_TX	写 1，发送数据的同时也将数据接收回来
[3]	CLR_OV	写 1，清除数据溢出标志位
[2]	RELEASE_FIFO	写 1，释放 RFIFO
[1]	INTR_TRANS	写 1，将中断取消未执行的发送传输
[0]	TRANS_REQ	写 1，产生 CAN 发送传输请求

19.2.3.2.3 CAN_SR 状态寄存器

地址：0x4001_3408

复位值：0x0

表 19-10 状态寄存器 CAN_SR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								ON_BUS	ERR_OV	TXING	RXING	TRANS_DONE	TFIFO_EMPTY	RFIFO_EMPTY	DATA_AVAIL
								RO	RO	RO	RO	RO	RO	RO	RO
								0	0	0	0	0	0	0	0

位置	位名称	说明
[31:8]		未使用
[7]	ON_BUS	1：CAN 模块，处于 BUS OFF 状态，无数据发送接收动作 0：CAN 模块，可以发送数据到 CAN 总线，或接收 CAN 总线数据
[6]	ERR_OV	1：CAN 传输产生的错误总数达到或超过 CAN_EWL 规定值 0：CAN 传输产生的错误总数低于 CAN_EWL 规定值 错误值低于 CAN_EWLR，标志自动拉低
[5]	TXING	1：CAN 模块正在发送一帧数据 0：CAN 模块没有发送数据的动作
[4]	RXING	1：CAN 模块正在接收一帧数据 0：CAN 模块没有接收数据的动作
[3]	TRANS_DONE	1：最近一次传输已完成 0：最近一次传输未完成
[2]	TFIFO_EMPTY	1：TFIFO 空，可以写入发送数据 0：TFIFO 非空，内部数据没有发送完毕
[1]	RFIFO_EMPTY	1：RFIFO 存入的帧太多，已满，导致数据丢失 0：RFIFO 未满 对 CAN_CMR BIT[2]写 1，可清除此位（表示用户已经取走一帧数据，暂时 RFIFO 非满了）

[0]	DATA_AVAIL	1：RFIFO 存有一帧或多帧数据，可以通过 RFIFO 寄存器读取 0：RFIFO 没有有效帧数据

19.2.3.2.4 CAN_IR 中断状态寄存器

地址：0x4001_340C

复位值：0x0

表 19-11 中断状态寄存器 CAN_IR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								BUS_ERR_IF	LOST_ARB_IF	M_ERR_IF	WK_IF	RFIFO_OV_IF	W_ERR_IF	TX_DONE_IF	RFIFO_N_EMPTY_IF
								RO	RO	RO	RO	RO	RO	RO	RO
								0	0	0	0	0	0	0	0

位置	位名称	说明
[31:8]		未使用
[7]	BUS_ERR_IF	1：总线错误中断（CAN_EIR.7 为有效前提下）
[6]	LOST_ARB_IF	1：丢失仲裁，改成接收模式（CAN_EIR.6 为有效前提下）
[5]	M_ERR_IF	被动错误中断。若值为 1，表明节点由于错误计数数值的变化，在主动错误状态与被动错误状态间发生了切换。
[4]	WK_IF	1：CAN 模块从休眠中唤醒（CAN_EIR.4 为有效前提下）
[3]	RFIFO_OV_IF	1：RFIFO 数据发生溢出（CAN_EIR.3 为有效前提下）
[2]	W_ERR_IF	错误报警中断。若值为 1，表明 SR 寄存器中 CAN_SR.7 和 CAN_SR.6 发生变化（0 变为 1 或 1 变为 0）。 1：CAN_SR.7 或者 CAN_SR.6 有变化（CAN_EIR.2 为有效前提下）
[1]	TX_DONE_IF	1：发送完毕当前帧（CAN_EIR.1 为有效前提下），可以执行新的数据发送任务
[0]	RFIFO_N_EMPTY_IF	1：RFIFO 有数据（CAN_EIR.0 为有效前提下）

CAN_IR 寄存器，为读清除寄存器。只有 BIT0--RFIFO_N_EMPTY_IF 无法读清除，此位只能通过对 CAN_CMR BIT2 写 1 清除，若 RX FIFO 有多帧接收的数据，CAN_CMR BIT2 写 1 也无法清除，除非 RX FIFO 全部读完，即读一帧数据写 CAN_CMR BIT2，反复迭代。

19.2.3.2.5 CAN_EIR 中断控制寄存器

地址：0x4001_3410

复位值：0x0

表 19-12 中断控制寄存器 CAN_EIR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								BUS_ERR_IE	LOST_ARB_IE	M_ERR_IE	WK_IE	RFIFO_OV_IE	CAN_SR67_IE	TX_DONE_IE	RFIFO_N_EMPTY_IE
								RW	RW	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

位置	位名称	说明
[31:8]		未使用
[7]	BUS_ERR_IE	1：总线错误中断，中断源使能 0：该中断源关闭
[6]	LOST_ARB_IE	1：丢失仲裁，改成接收模式，中断源使能 0：该中断源关闭
[5]	M_ERR_IE	1：被动错误中断源使能 0：该中断源关闭
[4]	WK_IE	1：CAN 模块从休眠中唤醒，中断源使能 0：该中断源头关闭
[3]	RFIFO_OV_IE	1：RFIFO 数据发生溢出，中断源使能 0：该中断源头关闭
[2]	W_ERR_IE	1：错误报警中断，CAN_SR.7 或者 CAN_SR.6 有变化，中断源使能 0：该中断源头关闭
[1]	TX_DONE_IE	1：发送完毕当前帧，中断源使能 0：该中断源头关闭
[0]	RFIFO_N_EMPTY_IE	1：RFIFO 有新的数据被接收到，中断源使能 0：该中断源头关闭

19.2.3.2.6 CAN_BTR0 波特率 0 控制寄存器

地址：0x4001_3418

复位值：0x0

表 19-13 波特率 0 控制寄存器 CAN_BTR0

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

	SJW	BAUDRATE
	RW	RW
	0	0

位置	位名称	说明
[31:8]		未使用
[7:6]	SJW	同步跳转宽度配置。公式如下： Tsjw = TQ*(SJW + 1)
[5:0]	BAUDRATE	波特率配置，计算 CAN 传输基本时间单元参数 TQ,Tclk 为 CAN 模块的系统时钟频率 TQ = 2*Tclk*(BAUDRATE+1)

19.2.3.2.7 CAN_BTR1 波特率 1 控制寄存器

地址：0x4001_341C

复位值：0x0

表 19-14 波特率 1 控制寄存器 CAN_BTR1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								OSR	SEG2			SEG1			
								RW	RW			RW			
								0	0			0			

位置	位名称	说明
[31:8]		未使用
[7]	OSR	1：1-bit 数据被采样 3 次 0：1-bit 数据被采样 1 次
[6:4]	SEG2	SEG2 段时间 Tseg2 = TQ*(SEG2+1)
[3:0]	SEG1	SEG1 段时间 Tseg1 = TQ*(SEG1+1)

19.2.3.2.8 CAN_ALC 仲裁丢失捕捉寄存器

地址：0x4001_342C

复位值：0x0

表 19-15 仲裁丢失捕捉寄存器 CAN_ALC

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

	LOST_ARB
	RW
	0

位置	位名称	说明
[31:5]		未使用
[4:0]	LOST_ARB	记录总线仲裁丢失的具体位置。 0：ID 的第一位 1：ID 的第二位 2：ID 的第三位 3：ID 的第四位 4：ID 的第五位 5：ID 的第六位 6：ID 的第七位 7：ID 的第八位 8：ID 的第九位 9：ID 的第十位 A：ID 的第十一位 B：SRTR 位 C：IDE 位

19.2.3.2.9 CAN_ECC 错误码捕捉寄存器

地址：0x4001_3430

复位值：0x0

表 19-16 错误码捕捉寄存器 CAN_ECC

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

位置	位名称	说明
[31:8]		未使用
[7:6]	BUS_ERR_TYPE	总线错误类型 00：bit 错误 01：格式错误 10：填充错误 11：其它类型错误
[5]	ERR_TIMING	1：总线错误发生在接收过程中

		0：总线错误发生在发送过程中
[4:0]	ERR_POSITION	错误发生的位置 02：ID.28 到 ID.21 03：起始帧 04：SRTR 位 05：IDE 位 06：ID.20 到 ID.18 07：ID.17 到 ID.13 08：CRC 数据段 09：保留位，固定为 0 0A：Data Field 0B：Data Length Code 0C：RTR 位 0D：保留位，固定为 1 0E：ID.4 到 ID.0 0F：ID.12 到 ID.5 11：active error flag 12：Intermission 13：Tolerate dominant 位 16：passive error flag 17：Error delimiter 18：CRC delimiter 19：确认位 1A：帧结束 1B：确认 delimiter 1C：Overload flag

19.2.3.2.10 CAN_EWLR 错误&警告门限值寄存器

地址：0x4001_3434

复位值：0x0

表 19-17 错误&警告门限值寄存器 CAN_EWLR

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

位置	位名称	说明
[31:8]		未使用
[7:0]	EWLR	正常工作模式下只读，复位模式下可读写。写入新值后，在正常工作模式下生效。

19.2.3.2.11 CAN_RXERR 接收错误计数器寄存器

地址：0x4001_3438

复位值：0x0

表 19-18 接收错误计数器寄存器 CAN_RXERR

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

位置	位名称	说明
[31:8]		未使用
[7:0]	RXERR	正常工作模式下只读，复位模式下可读写。 错误数超过 CAN_EWLR，CAN_SR.ERR_OV 被拉高；数值超过 127，CAN 模块进入被动错误状态。复位或者 BUS OFF 事件，硬件清零本寄存器。本寄存器自动增减，具体增减规则见本章 19.2.2.12。

19.2.3.2.12 CAN_TXERR 发送错误计数器寄存器

地址：0x4001_343C

复位值：0x0

表 19-19 发送错误计数器寄存器 CAN_TXERR

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

位置	位名称	说明
[31:8]		未使用
[7:0]	TXERR	正常工作模式下只读，复位模式下可读写。 错误数超过 CAN_EWLR，CAN_SR.ERR_OV 被拉高；数值超过 127，CAN 模块进入被动错误状态。数值超过 255，进入 BUS OFF 状态，CAN_MOD.MODE 被硬件拉高，处于复位模式。CAN_SR.ON_BUS 被硬件自动拉高，本寄存器被硬件赋值为 127；复位，硬件清零本寄存器。具体增减规则见本章 19.2.2.12。

19.2.3.2.13 CAN_TXRX0~ CAN_TXRXC 发送接收寄存器

地址：0x4001_3440~0x4001_3470

复位值：0x0

表 19-20 发送接收寄存器 CAN_TXRX

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

位置	位名称	说明
[31:8]		未使用
[7:0]	TXRX	发送或者接收数据寄存器。 正常工作模式下，该寄存器映射的是 RFIFO 中第一个有效帧数据； 正常工作模式下，对该寄存器写入数据，直接存入 TFIFO 中。

19.2.3.2.14 CAN_ACR ID 寄存器

地址：0x4001_3440~0x4001_344C

复位值：0x0

表 19-21 ID 寄存器 CAN_ACR

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

位置	位名称	说明
[31:8]		未使用
[7:0]	ACR	仅复位模式下可用；复位模式下，可读写； 可接收的 ID 寄存器，当输入帧的 ID 同此匹配，则被 CAN 模块接受。配合 AMR 寄存器，可同一类 ID 匹配。

19.2.3.2.15 CAN_AMR ID 掩码寄存器

地址：0x4001_3450~0x4001_345C

复位值：0x0

表 19-22 ID 掩码寄存器 CAN_AMR

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

位置	位名称	说明
[31:8]		未使用
[7:0]	AMR	仅复位模式下可用；复位模式下，可读写； 配合 ACR 寄存器，同输入 ID 进行匹配。AMR 寄存器某一位为 0，标识 ID 对应位需要同 ACR 匹配；为 1，标识 ID 对应位不需要同 ACR 匹配。

19.2.3.2.16 CAN_RMC FIFO 有效接收数据寄存器

地址：0x4001_3474

复位值：0x0

表 19-23 FIFO 有效接收数据寄存器 CAN_RMC

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

位置	位名称	说明
[31:5]		未使用
[4:0]	FRAME_CNT	正常工作模式和复位模式，此寄存器均只可读。 CAN 收到一帧数据，计数器自动累加 1；DMA 方式读取完毕一帧数据后，计数器自动减 1；CPU 方式读取完毕一帧数据后，软件需将 CAN_CMR[2]写 1，手动实现计数器减 1。

19.2.3.2.17 CAN_RBSA 有效接收数据地址寄存器

地址：0x4001_3478

复位值：0x0

表 19-24 有效接收数据地址寄存器 CAN_RBSA

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

	0

位置	位名称	说明
[31:5]		未使用
[4:0]	RBSA	正常工作模式，只可读。复位模式，可读写 RFIFO 的大小为 32 字节，本寄存器指明第一帧有效接收数据在 RFIFO 的位置。

19.2.3.2.18 CAN_ RFIFO0~CAN_RFIFO31 RX FIFO 寄存器

地址：0x4001_3480~0x4001_34FC

复位值：0x0

表 19-25 RX FIFO 寄存器 CAN_ RFIFO0~CAN_RFIFO31

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
[31:8]		未使用
[7:0]	RX_DATA	RX FIFO 地址 正常工作模式下，只可读；复位模式下，可读写。

19.2.3.2.19 CAN_TFIFO0~CAN_TFIFO12 TX FIFO 寄存器

地址：0x4001_3580~0x4001_35B0

复位值：0x0

表 19-26 TX FIFO 寄存器 CAN_ TFIFO0~CAN_TFIFO12

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
[31:8]		未使用
[7:0]	TX_DATA	TX FIFO 地址 正常工作模式下和复位模式下，均只可读。

20 SIF

20.1 概述

SIF 总线接口，支持 SIF 协议，液晶显示器与电动车控制器之间的数据传输，实现其对运行状态和故障的检测。

20.2 主要特性

- 采用国际标准 SIF 通信协议，接口通用方便。
- 主从方式采用单线单向传输，即只需要一根传输线路，电动车控制器为发送，仪表为接收方。
- 传输线与电动车控制故障运行灯共用 I/O 口，不占用额外资源。
- 传输波特率自适应范围宽，主机可以利用空闲时间发送数据。
- 时间基准单位范围广，32us <Tosc<320us。
- 数据的电平遵守 TTL 规范。
- 一个中断向量。

20.3 功能描述

20.3.1 功能框图

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

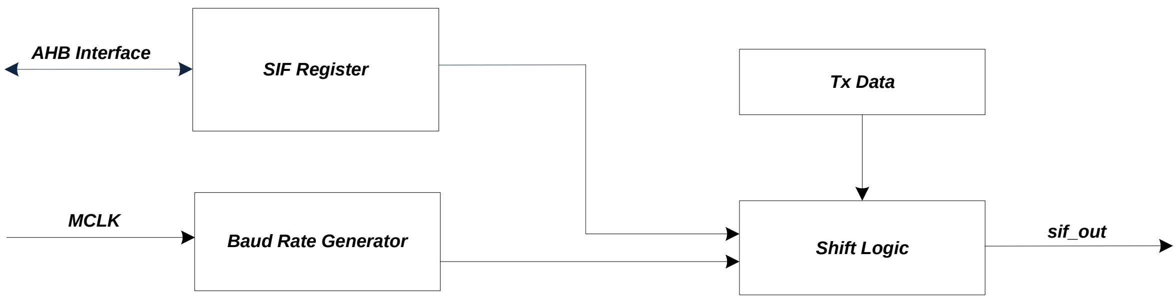


图 20-1 SIF 模块顶层功能框图

SIF 接口同外界通讯只有 sif_out 一根信号线。

20.3.2 功能说明

SIF 接口仅支持主发送，模块发送数据，将数据从并行转换成串行。可以开启或禁止中断。接口通过数据引脚 sif_out 连接到 SIF 总线。

20.3.3 模式选择

SIF 接口仅有一种运行模式：主发送模式。因为 SIF 接口的传输速度很慢，不支持 DMA 传输。每次传输一个字节完毕后，产生中断信号。

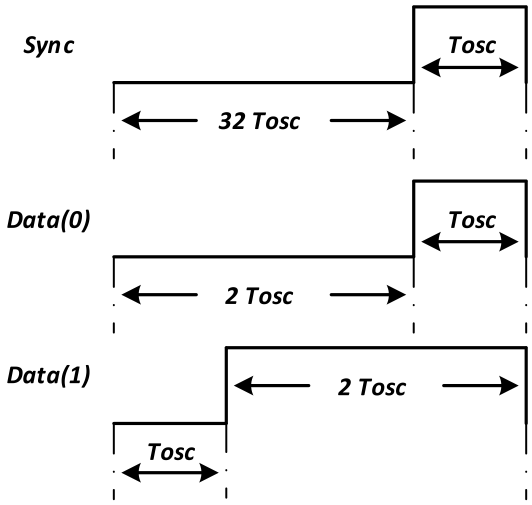


图 20-2 SIF 基本传输时序图

注：图 17-2 占空比为 2:1，可选配为 3:1。SIF_FREQ[0]控制占空比的选择。

20.3.4 SIF 接口传输

默认情况下，SIF 接口关闭。若需要发送数据，需要使能 SIF 接口。为了产生正确的时序，必须在 SFI_CFG 中设定 SIF 接口的工作时钟。

20.3.5 中断处理

SIF 接口仅含一种类型的中断事件--数据传输完成事件。

20.3.6 通讯速度设置

SIF 接口的基本时间单位是 32us--Tosc。Tosc 的产生来自系统时钟。在不同系统频率下，通过配置 SIF_FREQ 寄存器，获得我们需要的基准时间。

20.4 寄存器

20.4.1 地址分配

SIF 模块寄存器的基地址是 0x4001_3800，寄存器列表如下:

表 20-1 SIF 模块控制寄存器列表

名称	偏移	说明
SIF_CFG	0x00	SIF 配置寄存器
SIF_FREQ	0x04	SIF 波特率配置寄存器
SIF_IRQ	0x08	SIF 中断寄存器
SIF_WDATA	0x0C	SIF 发送数据寄存器

20.4.2 寄存器说明

20.4.2.1 SIF_CFG 配置寄存器

地址：0x4001_3800

复位值：0x0

表 20-2 地址寄存器 SIF_CFG

15-5	4	3-2	1	0
	LAST_BYTE		ENDIAN	EN
	RW		RW	RW
	0		0	0

位置	位名称	说明
[31:5]		未使用
[4]	LAST_BYTE	当前字节传输完毕后，结束本次传输。SIF 接口传输当前字节，字节传输完成后，产生中断。在中断处理程序中对该位写 1，让 SIF 接口恢复默认状态。若不操作此位，表明还有数据要发送。该位读回永远是 0。
[3:2]		未使用
[1]	ENDIAN	SIF 大小端设置。1，MSB；0，LSB。默认为 0。
[0]	EN	SIF 模块使能。默认值为 0。 1，使能 0，关闭

20.4.2.2 SIF_FREQ 波特率寄存器

地址：0x4001_3804

复位值：0x0

表 20-3 系统控制寄存器 SIF_FREQ

15-8	7-4	3-1	0
	TOSC		DUTY
	RW		RW
	0		0

位置	位名称	说明
[31:8]		未使用
[7:4]	TOSC	96Mhz 系统时钟下，基准时间单位 Tosc 设置。 01: 32us; 02: 64us; 03: 96us;04:128us 05:160us;06:192us;07:220us;08:256us 09:288us;10:320us;其它均为 32us。

[3:1]		未使用
[0]	DUTY	SIF 传输 Data 占空比 1: 占空比为 3:1 0: 占空比为 2:1

20.4.2.3 SIF_IRQ 状态控制寄存器

地址：0x4001_3808

复位值：0x0

表 20-4 状态控制寄存器 SIF_IRQ

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

位置	位名称	说明
[31:5]		未使用
[4]	IF	SIF 中断事件，高电平有效。写 1 清除。
[3:1]		未使用
[0]	IE	SIF 中断使能信号，高电平有效。默认值为 0。

20.4.2.4 SIF_WDATA 数据寄存器

地址：0x4001_380C

复位值：0x0

表 20-5 数据寄存器 SIF_WDATA

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

位置	位名称	说明
[31:8]		未使用
[7:0]	WDATA	数据寄存器，SIF 接口使能后，对该寄存器写入发送数据将触发 SIF 的数据传输。

21 看门狗

21.1 概述

看门狗工作于低速 RC 时钟域 LSI，即使用 32kHz 进行计数。支持 2s、4s、8s、64s 四档复位时间可选。

复位控制寄存器 SYS_RST_CFG.WDT_EN 可用于使能或禁用看门狗，SYS_RST_CFG.WDT_EN=1 则使能看门狗模块。复位源记录寄存器 SYS_RST_SRC.WDT_RST_RCD 记录了看门狗复位事件，SYS_RST_SRC.WDT_RST_RCD 为高表示发生过看门狗复位。

看门狗复位是硬件全局复位，其作用域等同于外部引脚复位以及内部的上电复位。

21.2 寄存器

21.2.1 地址分配

看门狗模块寄存器基地址为 0x4000_0000。

表 21-1 看门狗模块寄存器

名称	偏移	说明
SYS_WDT_CLR	0x38	看门狗清零寄存器

21.2.2 寄存器说明

21.2.2.1 SYS_WDT_CLR 看门狗清零寄存器

表 21-2 看门狗清零寄存器 SYS_WDT_CLR

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

位置	位名称	说明
[31:16]		未使用
[15:0]	WDT_CLR	写入字节 16’b0111_1001_1000_1$B_2B_1B_0$，高 13 位为密码，密码正确时，B[2:0]才能写入 其中， B[2:1]为 MODE 2’b00：64 秒复位 2’b01：8 秒复位

		2’b10：4 秒复位 2’b11：2 秒复位 B[0]为 CLR，写入 1，则复位 WDT 计数器

看门狗清零前，需要向 SYS_WR_PROTECT 写入 0xCAFE，开启 WatchDog SYS_WDT_CLR 寄存器写入。

22 版本历史

表 22-1 文档版本历史

时间	版本号	说明
2024.06.14	1.37	增加 NVR 校正值地址信息
2024.01.15	1.36	增加关于休眠模式的说明 增加关于 I2C 模块通讯速度的说明
2023.12.06	1.35	CAN 常规波特率索引值修正，计算公式修正
2023.09.07	1.34	修订关于 ADC TIMER 触发的说明 增加 ADC 在 3.3V 供电时必须使用 1.2V 基准的说明
2023.08.15	1.33	GPIO 功能框图增加 ESD 二极管
2023.03.18	1.32	修改时钟精度的描述
2023.02.28	1.31	修改 PVDSEL 的描述
2023.02.18	1.3	修改 MCPWM_SDCFG 的描述
2023.02.11	1.29	增加关于关闭 PLL 等操作的说明 增加 CAN 通讯需要使用外部晶振的描述 修改 SYS_AFE_REG5.BGPPD 的描述
2023.01.03	1.28	修改编码器寄存器描述
2022.11.07	1.27	增加 IO 与内部模拟电路间连接电阻描述
2022.10.29	1.26	修改 MCPWM_DTHx0/MCPWM_DTHx1 寄存器描述
2022.10.17	1.25	修订 DAC 校准输出公式
2022.09.23	1.24	修改 ADC 1.2V 基准时温度传感器使用说明
2022.08.18	1.23	增加 ADC/CMP 配置流程，简化 SYS_AFE_REG 部分描述
2022.04.08	1.22	增加 MCU 内核描述
2021.10.18	1.21	修订 FLASH 擦写次数的描述
2021.08.03	1.20	修订 CAN 模块部分描述
2021.06.07	1.19	修订 CAN 模块部分描述，修订 FLASH NVR 空间描述
2021.05.20	1.18	修订比较器 BEMFx_MID 的描述
2020.06.12	1.17	增加了复位源的说明，修订 UTimer ETON 的说明，SYS_AFE_CMP 修改为 SYS_AFE_INFO，增加了 PWR_WEAK 标志位的说明
2020.02.18	1.16	修订部分说明
2019.12.03	1.15	修订部分图示
2019.09.12	1.14	修订部分说明
2019.06.01	1.10	时钟部分章节修订
2019.03.18	1.0	针对发布的修订
2018.11.29	0.1	初始版本

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。