AC7801x 技术参考手册

文档版本： 1.5

发布日期： 2022-03-17

修订信息

版本	日期	作者	描述
0.1	2020-03-27	AutoChips	文档初版
0.2	2020-03-31	AutoChips	1. 10.5 章节，修改寄存器定义章节 ACMP_CR0 和 ACMP_ANACFG 寄存器内容
0.3	2020-04-17	AutoChips	1. 增加 PLL LD 配置说明 2. 18.6.1 章节，SPI_CFG0 寄存器增加 SCK_LOW,SCK_HIGH 的说明 3. 19.6.1 章节，DMA_TOP_RST 寄存器增加 HARD_RST 和 WARM_RST 的说明 4. 10.5.13 章节，LPF 默认值修改 5. 9.4.6 章节，ADC 校准公式修改 6. 11.4.18 章节，PWM FAULT 说明修改 7. 附录表格 ATC_AC7801x_PINMUX_Table.xlsx
1.0	2020-06-17	AutoChips	1. 4.2.2 章节，系统时钟补充典型 PLL 配置参考表 2. 7.3.1.3 章节，关于 RBUF 的表格修改至 7.3.1.2 章节 3. 7.2 章节，修改可编程的波特率小结内容的描述 4. 7.2 章节，增加 CAN 特性中关于 CAN 时钟源可选择的描述 5. 9.4.7 章节，增加 ADC 采样转换时间说明 6. 22.4.1 章节，修改表 22-1 片内 Flash 存储器组织最大 Page 号为 63 7. 22.6.6 章节，修改 EFLASH_CTRL1 寄存器的 SPEED_LATENCY 位描述默认值改为 0x02 8. 每个章节，增加寄存器名前增加中文描述
1.1	2020-09-25	AutoChips	1. 对文档中所有写“1”清零的寄存器访问属性改为“W1C”，对写“0”清零的寄存器位访问属性改 为“W0C” 2. 4.3.1 章节，修改寄存器位 CAN0_TIMCLK_DIV 的描述 3. 4.3.6 章节，修改 PLL 配置寄存器 0 的 SYSPLL1_MONREF_EN 描述为使能 PLL Monitor 参考时钟 4. 7.4.2 章节，修改寄存器 CAN_TTSx 的描述 5. 12.4.1.2 章节，修改“PWDT RDY”标志为“RDYF” 6. 17.2 章节，修改 overview，将“高速模式”改为“快速+” 7. 22.5.1 章节，修改“EOP_INT”为“EOP”

			8. 22.6.6 章节，修改 bit 8 CKDIV_LOCK 中 EFLASH_CTRL2 为 EFLASH_CTRL1
1.2	2020-12-21	AutoChips	1. 全文中的 WDT 全改为 WDG 2. 图 4-2，修改系统时钟示意图以及相关描述 3. 表 5-1，重新排序低功耗模式下的模块功能 中的注释标号 4. 8.6.15 章节，修改 UART 小数分频器的位宽度 5. 9.4.6 章节，修改 ADC 校准描述 6. 12.4.12 章节，修改 PWDT RDY 标志为 RDYF 7. 16.3.2 章节，增加说明 32pin 没有 I2C1 8. 表 16-7，修改 GPIO_ODR 寄存器的描述 9. 16.5.9 章节，增加 pinmux 寄存器默认值的描述 10. 19.6.11 章节，增加 DMA 在循环模式下，关闭 DMA 的描述 11. 22.5.1 章节，修改 EOP_INT 为 EOP 12. 23.4.2 和 23.4.3 修改 ECC 错误地址描述
1.3	2021-01-28	AutoChips	1. 7.3.2 章节，修改退出 BUSOFF 方式为：模块(SRST_CAN0)复位或者接收到 128 组连续 11 个隐性位（恢复序列） CAN 节点可以返回到主动错误状态 2. 7.3.12 章节，增加 TDC 和 SSPOFF 描述 3. 9 章节， 修改工作模式序列图错误 4. 11.4.18 章节， 故障控制上升沿改为有效边沿 5. 11.5.20 章节， 寄存器输入事件 PSC 增加描述 6. 16.3.2 章节，增加备注：PA12, PA15 配置复用功能为 XOSC 后，不支持切换为其它的复用功能 7. 16.5.6 章节和 16.5.7 章节，增加上下拉电阻典型值为 75KΩ的说明 8. 17.6.3 章节，SAMPLE_CNT_DIV 改为 SAMPLE_CNT 9. 17.6.4 章节，STEP_CNT_DIV 改为 STEP_CNT 10. 21.4.3 章节， RTC_CLKOUT PA9 改为 PA13 11. 21.6.1 章节， RTCO 改为 RTC_CLKOUT
1.4	2021-06-15	AutoChips	1. 7.3.9.3 章节，更新 KOER 的描述 2. 7.4.2 章节，更新寄存器 CAN_CTRL0 的 bit9 TSALL 描述 3. 8.6.9 章节，更新 UART DR 位状态的描述 4. 11.4.20 章节，更新操作步骤中的描述 5. 表 16-2，GPIO 复用功能描述，PA12 增加 PWM0_FLT0 的复用功能 6. 17.3.1 增加对地址操作的详细描述 7. 17.4.1 主机模式，增加了主机同步的描述 8. 17.6.9 增加 BND 行为的详细描述

			9. 19.2 章节，DMA 删除外设到外设传输的描述 10. 20.6.4 章节，寄存器表格中的默认值修改为 0x1F8000，表格右上角的默认值保持不变 11. 22.2 章节，增加最小编程位宽为 32bit，编程地址需 4 字节对齐
1.5	2022-03-17	AutoChips	1. 2.2.11 章节，增加 ISP 下载引脚描述“ISP 下载引脚为 UART 的 PA7（TX）和 PA8（RX）” 2. 7.4.2 章节，更新 BUSOFF 位的描述 3. 8.2 章节，增加 UART 波特率范围说明 4. 9.4.4 章节，增加 AMOHR/AMOLR 寄存器单位描述 5. 9.4.4.2 章节，增加 AMO 边沿模式范围描述 6. 9.5.3 章节，增加 CALEN 使用建议 7. 9.5.4 和 9.5.5 章节，增加 STPx 编号描述 8. 11.4.8.4 章节，修改死区插入波形图 9. 11.4.19 章节，新增章节，增加写缓冲更新的寄存器内容 10. 11.4.21 章节，新增章节，增加特性优先级内容 11. 13.5.1 章节，增加“MDIS 可以实现 4 路 timer 的暂停/重新开始计数” 12. 13.5.3 章节，增加 TIMER_CVAL 寄存器值与时间的换算公式描述 13. 16.2 章节，修改 GPIO 输出状态的描述 14. 16.3 章节，新增章节，增加 GPIO 模块框图 15. 16.6.6 和 16.6.7 章节，增加 GPIO 上下拉电阻不支持同时使能的描述 16. 17.2 章节， I2C 速率更新为 100kHz、400kHz、1MHz 17. 17.6.2 章节，修改 I2C_ADDR1[RAD]位的描述 18. 18.2 章节，增加 SPI 最大波特率说明 19. 18.6.3 章节，增加 SPI_CMD[SPI TXEIE]和 SPI_CMD[RXFIE]位的说明 20. 19.4.3 章节，修改 DMA 硬件优先级的描述 21. 19.5.3 和 19.6.9 章节，修改 DMA 结束地址的描述 22. 22.4.2 章节，增加读写保护的描述 23. 22.6.4 章节，修改 CMD_ST 位的说明 24. 22.6.5 章节，增加清除 EFLASH_SR0 寄存器的错误状态位的操作说明

版权声明

本参考包含杰发科技的机密信息。禁止未经授权使用或披露本手册包含的信息。 对因未经杰发科技授权而全部或部分披露此文档内容而给杰发科技带来的任何损失或损害，杰发科技将追究责任。

杰发科技保留对此处任何信息进行更改的权利，此处的信息如有变更，恕不另行通知。杰发科技对使用或依赖此处包含的信息不承担任何责任。

本参考手册的所有信息均“按原样”提供，不提供任何形式的明示、暗示、法定或其他形式的保证。杰发科技明确拒绝对适销性，非侵权性和针对特定用途的适用性方面的的所有暗示保证。杰发科技对本手册可能使用、包含或提供的任何第三方软件不提供任何担保，并且用户同意仅向该等第三方寻求与此相关的任何担保索赔。杰发科技对于根据用户规格或为符合特定标准或公开论坛而产生的任何交付物，也不承担任何责任。

文档目录

插图目录

表格目录

缩略语

AAI	Auto Address Increment	地址自增模式
AHB	Advanced High Performance Bus	高级高性能总线
APB	Advanced Peripheral Bus	高级外设总线
CKGEN	Clock Generator	时钟产生器
ECC	Error Checking and Correction	错误检测以及纠正
FBKDIV	Feedback Divider	反馈除法器
HSE	High Speed External Clock	高速外部时钟
HSI	High Speed Internal Clock	高速内部时钟
ISP	In-System Programming	在系统编程
LRU	Least Recently Used	最少最近使用
LSI	Low Speed Internal Clock	低速内部时钟
LVD	Low Voltage Detect	低电压检测
MMDIVSQRT	Memory-Mapped Division And Square Root	内存映射除法器和开方根
NMI	Non Maskable Interrupt	不可屏蔽中断
OSC	Oscillator	振荡器
PLL	Phase Locked Loop	锁相环
POR	Power On Reset	上电复位
POSDIV	Post Divider	预置除法器
PREDIV	Previous Divider	前置除法器
PVD	Programmable Voltage Detect	可编程电压检测
SPM	System Power Manager	系统电源管理
SRAM	Static Random-Access Memory	静态随机存取存储器
SYSPLL	System Phase Locked Loop	系统时钟
VCO	Voltage Controlled Oscillator	压控振荡器
XOSC	External Crystal Oscillator	晶体振荡器

1 简介

1.1 概要

AC7801x 是采用 ARM Cortex™-M0+内核的高性能、低功耗 MCU。

- 频率高达 48MHz （部分芯片型号频率高达 72MHz）
- 工作温度范围支持-40℃~ +125℃
- 工作电压支持 2.7V~5.5V

1.2 模块概述

表 1-1 AC7801x 模块

模块	说明
ARM Cortex™-M0+内核	• ARM Cortex™-M0+ 32位MCU 内核 • 高达48 MHz CPU 频率 （部分芯片型号频率高达72MHz）
存储器	• 高达 128 Kbyte的片内Flash存储器 • 高达 20 Kbyte的SRAM
时钟	• 外部晶振或谐振器 – 范围：4MHz ~30MHz • 外部输入时钟 • 内置振荡器 – 8MHz振荡器 – 32kHz振荡器
模拟	• 1个12位模数转换器(ADC)，支持12路外部通道+2路内部通道 • 1个模拟比较器 (ACMP)，内置1个6位的数模转换器 (DAC)
定时器	• 2个全功能8通道脉宽调制（PWM）控制器 • 4个32bit通用定时器 (Timer) • 实时时钟(RTC) • 2个脉宽检测定时器 (PWDT) • 系统滴答定时器 (SysTick)
通信	• 2个串行外设接口(SPI) 模块 • 2个内部集成电路(I2C)模块 • 3个通用异步收发器(UART) 模块，仅UART0和UART1支持 LIN • 1个局域网控制器(CAN)，支持CANFD
接口	• 通用输入输出控制器(GPIO) • 中断(IRQ)
调试接口	• 串行调试（SWD）接口
算法模块	• 1个硬件除法+开方根模块

2 存储器和总线架构

2.1 结构框图

AC7801x 主系统由以下部分构成：

- 两个主模块单元
  - Cortex™-M0+ 内核 AHB-Lite 总线
  - DMA（通用 DMA）
- 四个从模块单元
  - 片内 SRAM
  - 片内 Flash 存储器
  - 快速 IO、 CRC 和 GPIO
  - AHB 到 APB 的桥（AHB_APB），它连接所有的 APB 设备。

如图 2-1 所示，这些模块单元都通过一个多级的 AHB 总线架构相互连接。

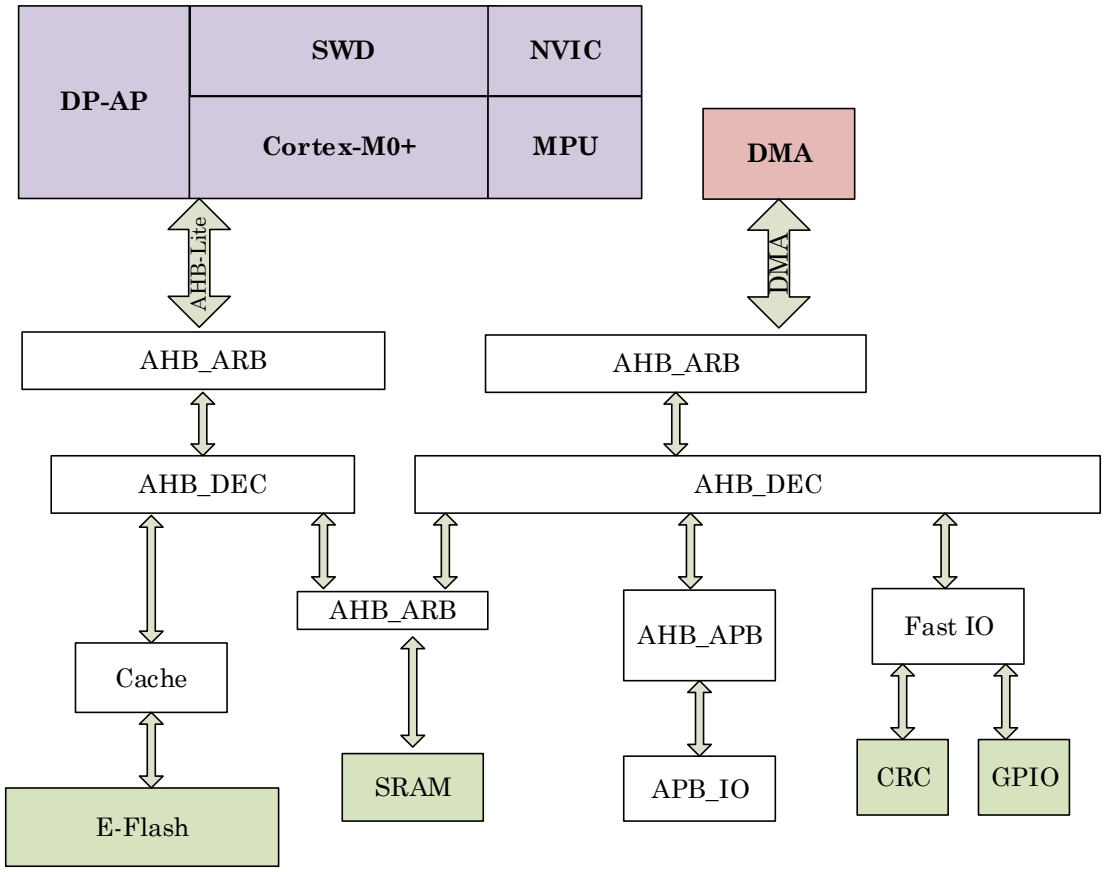

图 2-1 系统架构

AHB-Lite 总线

该总线将 Cortex™-M0+内核的指令和数据请求发出，访问存储器和外设。

AHB_ARB

AHB 仲裁模块（AHB Arbitration），用于对 AHB 总线和 DMA 总线进行仲裁。

AHB_DEC

AHB 译码模块（AHB Decode），仲裁后的地址会根据地址范围进行译码，译码后的地址才可以访问不同的 slaver。

MPU

AC7801x 支持 MPU 功能，默认关闭。使用时请使能相关配置，详细配置方法请参考《ARM Cortex™-M0+ Technical Reference Manual》Memory Protection Unit 章节。

DMA 总线

该总线将 DMA 的 AHB 主控接口和总线矩阵相连接。总线矩阵负责协调 CPU 和 DMA 到 SRAM 和外设的访问。

总线矩阵

总线矩阵负责协调内核系统总线和 DMA 主控总线间的访问仲裁，该仲裁使用轮询算法。

总线矩阵由主模块总线和从模块总线组成。

内部 Flash 只能被 AHB-Lite 总线访问。

AHB 外设通过 AHB-Lite 与总线矩阵相连来访问 DMA。

AHB 与 APB 连接桥

AHB 与 APB 连接桥（图 2-1 的 AHB_APB 单元）在 AHB 和 APB 总线间提供全同步连接，APB 操作频率默认为 AHB 操作频率的 1/2。

DP-AP

DP-AP 即调试访问端口，由 DP（debug port）和 AP（access port）组成。

DP 接口模块（AC7801x 仅支持 SW-DP）是把外部信号转换成 32bit 的调试总线信号。

AP 接口模块相当于一个总线桥，用于把调试总线命令转换成 AHB 总线上的数据，然后进行传送。

通过 DP 和 AP 的配合，实现 SWD 对 AC7801x 的地址访问。

2.2 功能描述

2.2.1 存储器组织

程序存储器、数据存储器、寄存器和输入输出（I/O）接口统一编址在 4G 字节的线性地址空间里。

数据字节在存储器中以小端格式存放。小端是指数据的低位保存在内存的低地址中，而数据的高位保存在内存的高地址中。例如，16bit 宽的数 0x1234 在小端格式 CPU 内存中的存放方式（假设从地址 0x4000 开始存放）为：

表 2-1 小端格式存放方式

内存地址	0x4000	0x4001
存放内容	0x34	0x12

外设寄存器的详细映射，请参照各外设章节。可寻址的存储空间分为 8 块，每块的寻址空间大小为 512Mbyte。

2.2.2 内置 SRAM

AC7801x 内置一个 20Kbyte 的静态 SRAM，它可以以字节、半字（16 位）或全字（32 位）访问。该 SRAM 的起始地址为： 0x2000 0000。

2.2.3 快速 GPIO 存储器映射

快速 IO 桥提供了一条更有效地通过 AHB 访问循环冗余校验（CRC）和 GPIO 的通道。

每个通道有 4Kbyte 的地址空间。

快速 GPIO 地址范围为：0x20080000 ~ 0x20080FFF。

CRC 地址范围为： 0x20081000 ~ 0x20081FFF。

2.2.4 存储器映射

表 2-2 为 AC7801x 设备存储器映射表，包括三种不同的基于不同启动配置的存储器映射表，分别为：片内 Flash 存储器启动、ISP 启动和 SRAM 启动。

所有没有分配片上存储和外设的存储器区域，被称之为“保留”区。

表 2-2 设备存储器映射

0xE010 0000	保留		0xE010 0000	保留		0xE010 0000	保留
0xE00F FFFF	Cortex™-M0+ 内部外设		0xE00F FFFF	Cortex™-M0+ 内部外设		0xE00F FFFF	Cortex™-M0+ 内部外设
0xE000 0000			0xE000 0000			0xE000 0000	
0xDFFF FFFF	保留		0xDFFF FFFF	保留		0xDFFF FFFF	保留
0x6000 0000			0x6000 0000			0x6000 0000	
0x5FFF FFFF	保留		0x5FFF FFFF	保留		0x5FFF FFFF	保留
0x4010 0000			0x4010 0000			0x4010 0000	
0x400F FFFF	外设 APB 地址		0x400F FFFF	外设 APB 地址		0x400F FFFF	外设 APB 地址

0x4000 0000			0x4000 0000			0x4000 0000		
0x3FFF FFFF	保留		0x3FFF FFFF	保留		0x3FFF FFFF	保留	
0x2008 2000			0x2008 2000			0x2008 2000		
0x2008 1FFF	AHB 快速 IO		0x2008 1FFF	AHB 快速 IO		0x2008 1FFF	AHB 快速 IO	
0x2008 0000			0x2008 0000			0x2008 0000		
0x2007 FFFF	保留		0x2007 FFFF	保留		0x2007 FFFF	保留	
0x2001 0000			0x2001 0000			0x2001 0000		
0x2000 FFFF	AHB SRAM		0x2000 FFFF	AHB SRAM		0x2000 FFFF	AHB SRAM	
0x2000 0000			0x2000 0000			0x2000 0000		
0x1FFF FFFF	保留		0x1FFF FFFF	保留		0x1FFF FFFF	保留	
0x0804 3000			0x0804 3000			0x0804 3000		
0x0804 27FF	保留		0x0804 27FF	保留		0x0804 27FF	保留	
0x0804 2000			0x0804 2000			0x0804 2000		
0x0804 1FFF	ISP Firmware		0x0804 1FFF	ISP Firmware		0x0804 1FFF	ISP Firmware	
0x0804 0800			0x0804 0800			0x0804 0800		
0x0804 002F	选项字节		0x0804 002F	选项字节		0x0804 002F	选项字节	
0x0804 0000			0x0804 0000			0x0804 0000		
0x0801 FFFF	Flash 存储器		0x0801 FFFF	Flash 存储器		0x0801 FFFF	Flash 存储器	
0x0800 0000			0x0800 0000			0x0800 0000		
0x07FF FFFF	保留		0x07FF FFFF	保留		0x07FF FFFF	保留	
0x0004 0000			0x0000 1800			0x0001 0000		
0x0003 FFFF	Flash 存储器		0x0000 17FF	ISP Firmware		0x0000 FFFF	AHB SRAM	
0x0000 0000			0x0000 0000			0x0000 0000		
Flash 存储器启动			**ISP 启动**			**SRAM 启动**		

2.2.5 片内 Flash 存储器

高性能片内 Flash 模块的主要特性如下：

- 高达 128Kbyte 的 Flash 存储器结构
- 存储器组织结构：Flash 存储器由主存储块和信息块组成
  - 主存储块容量： 主存储块最大为 32 K × 32 bit，每个存储块划分为 64 个 2Kbyte 的页

– 信息块容量：48byte

Flash 控制器特性为：

- Flash 编程/ 擦除操作
- 读/写保护
- 擦除及空白检查
- 缓存控制器用以提升读效率，最优效率为零等待

2.2.6 片内 Flash 存储器读取

Flash 的指令和数据访问是通过 AHB 总线完成的。

2.2.7 芯片型号信息

芯片型号信息用户可通过 eFlash 读操作接口进行访问。芯片型号信息存放在 0x40002028~0x4000202B 连续的空间(1*32Bit)。其中高 8 位为固定 0XFF，低 24 位为芯片型号信息。

2.2.8 芯片 UUID 信息

UUID 共 128Bit，由随机数产生，可作为芯片唯一的识别标识。用户可通过 eflash 读操作接口进行访问。UUID 信息存放在 0x4000202C~0x40002038 连续的空间(4*32Bit)。

2.2.9 AHB 与 APB 连接桥

AHB 与 APB 连接桥用于将 AHB 协议解析成 APB 协议。大多数外设为 APB 接口，详细信息可参考表 2-5 外设地址分配表。

2.2.10 嵌套中断向量控制器 (NVIC)

【特性】

- 32 个可屏蔽中断通道（不包括 16 个 Cortex™-M0+中断）
- 4 个可编程优先等级 (使用了 2 位 中断优先级)
- 低延迟的异常和中断处理
- 电源管理控制
- 系统控制寄存器的实现

NVIC 和处理器核的接口紧密相连，可以实现低延迟的中断处理和高效地处理新的中断。

NVIC 管理着包括内核异常在内的所有中断。更多关于异常和 NVIC 编程的内容，请参考《ARM Cortex™-M0+ Technical Reference Manual》第 8 章异常和中断。

AC7801x 中断表如下：

表 2-3 中断表

中断向量号	优先级	优先级类型	名称	说明
-15	-3	固定	复位（Reset）	系统复位
-14	-2	固定	不可屏蔽中断（NMI）	不可屏蔽中断
-13	-1	固定	硬件失效（HardFault）	用于处理所有的异常
			保留	保留
			保留	保留
			保留	保留
			保留	保留
			保留	保留
			保留	保留
			保留	保留
-5	0	可设置	SVCall	执行系统服务调用指令（SVC）引发的异常
			保留	保留
			保留	保留
-2	0	可设置	PendSV	可挂起的系统服务请求
-1	0	可设置	SysTick	系统嘀嗒定时器
0	0	可设置	PWDT0	PWDT0 中断
1	0	可设置	PWDT1	PWDT1 中断
2	0	可设置	PWM0	PWM0 中断
3	0	可设置	PWM1	PWM1 中断
4	0	可设置	ACMP0	ACMP0 中断
5	0	可设置	UART0	UART0 中断
6	0	可设置	UART1	UART1 中断
7	0	可设置	UART2	UART2 中断
8	0	可设置	WDG	看门狗中断
9	0	可设置	SPI0	SPI0 全局中断
10	0	可设置	SPI1	SPI1 全局中断
11	0	可设置	I2C0	IIC0 中断
12	0	可设置	I2C1	IIC1 中断
13	0	可设置	DMA0_CHANNEL0	DMA0 通道 0 全局中断
14	0	可设置	DMA0_CHANNEL1	DMA0 通道 1 全局中断
15	0	可设置	DMA0_CHANNEL2	DMA0 通道 2 全局中断
16	0	可设置	DMA0_CHANNEL3	DMA0 通道 3 全局中断
17	0	可设置	TIMER_CHANNEL0	定时器通道 0 中断

中断向量号	优先级	优先级类型	名称	说明
18	0	可设置	TIMER_CHANNEL1	定时器通道 1 中断
19	0	可设置	TIMER_CHANNEL2	定时器通道 2 中断
20	0	可设置	TIMER_CHANNEL3	定时器通道 3 中断
21	0	可设置	RTC	RTC 中断
22	0	可设置	PVD	PVD 中断
23	0	可设置	SPM	SPM 中断
24	0	可设置	CAN0	CAN0 中断
25	0	可设置	ADC0	ADC0 中断
26	0	可设置	ECC_SRAM	ECC SRAM 检错中断
27	0	可设置	EXTI0	EXTI 0 中断
28	0	可设置	EXTI1	EXTI 1 中断
29	0	可设置	EXTI2	EXTI 2 中断
30	0	可设置	EXTI3_8	EXTI 3~8 中断
31	0	可设置	EXTI9_15	EXTI 9~15 中断

2.2.11 启动配置

通过如下配置表，可以选择四种不同的启动模式。

表 2-4 启动配置

PIN 脚名称	BOOT(PA6)	PA1	PA0
eflash boot	0	x	x
ISP boot	1	0	0
SRAM boot	1	1	0

【注意】 x 表示忽略，不用关注，其中 ISP 下载引脚为 UART0 的 PA7（TX）和 PA8（RX）。

在系统复位后，在 8MHz 时钟的第 8 个上升沿，BOOT 配置引脚的值会被锁定。用户需要设置这些引脚来选择需要的启动模式。在锁定前，用户需要保持这些引脚处于稳定状态。

由于固定的存储器映射，代码区始终从地址 0x0000 0000 开始。Cortex™-M0+ CPU 从 AHB-Lite 总线获取复位向量，即启动仅适用于从代码区开始（典型地，从主 Flash 存储器启动）。AC7801x 实现了一种特殊的机制，系统不仅可以从主 Flash 存储器和系统存储器启动，而且可以从 SRAM 启动。

根据选定的启动模式，片内 Flash 存储器、ISP 存储器可以按照如下方式访问：

- **从片内 Flash 存储器启动：** 片内 Flash 存储器被映射到启动存储空间 (0x0000 0000)，但仍然能够在原有的地址(0x800 0000) 访问它。 换言之，片内 Flash 存储器的内容可以在两个地址区域访问：0x0000 0000 或 0x800 0000。
- **从 ISP 启动：**ISP Firmware 被映射到启动存储空间(0x0000 0000)，但仍然能够在它原有的地址(0x08040800)访问它。

- **从片内 SRAM 启动：** SRAM 被映射到启动存储空间 (0x0000 0000)，但仍然能够在它原有的地址 （0x2000 0000）访问它。

2.3 外设地址分配

表 2-5 外设地址分配表

APB 存储映射	基地址	大小（字节）
CKGEN	0x40000000	4K
GPIO	0x40001000 / 0x20080000	4K
片内 Flash 控制器	0x40002000	4K
ADC0	0x40003000	4K
ACMP0	0x40005000	4K
CAN	0x40007800	1K
SPM	0x40008000	1K
RTC	0x40008400	1K
WDG	0x4000b000	4K
SPI_0	0x4000c000	4K
SPI_1	0x4000d000	4K
IIC_0	0x4000e000	4K
IIC_1	0x4000f000	4K
Cortex™-M0+ controller	0x40010000	2K
TIMER	0x40011000	4K
DMA	0x40012000	4K
PWM0	0x40013000	4K
PWM1	0x40014000	4K
CTU	0x40016000	4K
PWDT0	0x40017000	2K
PWDT1	0x40017800	2K
UART_0	0x40018000	4K
UART_1	0x40019000	4K
UART_2	0x4001a000	4K
CRC	0x20081000	4K

3 复位（RESET）

3.1 特性

上电复位（POR）：

- POR_rst： IC 上电复位

系统复位：

- External Reset： 外部复位脚复位，低电平有效
- LVD Reset：低压检测复位，低电平有效
- Software Reset：Cortex™-M0+ 软件复位
- ECC 2 Bit Error Reset：ECC 检测到 2 BIT 错误复位，默认不使能
- Lock up Reset： Cortex™-M0+死锁复位
- PLL Unlock Reset：PLL 失锁复位，默认不使能
- Watchdog Reset Normal Mode：看门狗定时器复位，正常模式有效
- Watchdog Reset Stop Mode：看门狗定时器复位，停止模式有效
- XOSC Lost Reset：当检测到外部 XOSC 时钟异常时，将会产生一个可屏蔽的系统复位

每个系统复位源在 0x40000010 寄存器都有相应的状态位。

3.2 结构框图

模块结构框图如图 3-1 所示。各 Reset 信号默认都是高电平，任意一信号为低电平，即产生复位信号。

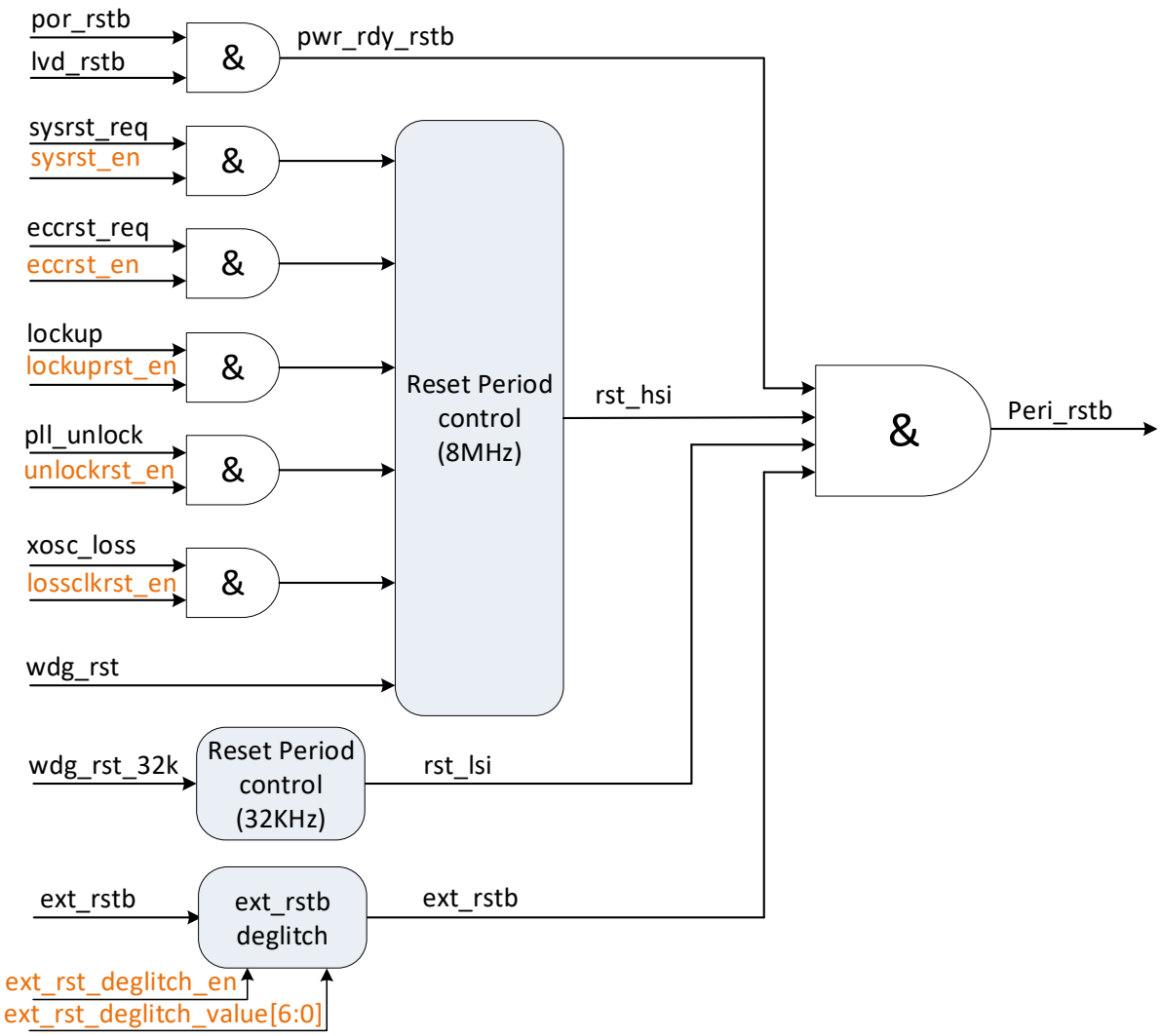

图 3-1 Reset 模块结构框图

3.3 功能描述

3.3.1 上电复位（POR）

产生 POR 的条件为：

1. MCU 初次上电；
2. 电源电压下降到低于上电复位电压电平（VPOR）。

随着电源电压上升，低电压检测（LVD）会将 MCU 保持在复位状态，直到电源电压上升至高于低电压检测（LVD）的低阀值（VLVDL），具体请参考《ATC_AC7801x_Datasheet_CH》表 6-2 LVD/POR/AVDD 电压告警规格。

3.3.2 系统复位（System Reset）

系统复位开始时，片上稳压器处于完全稳压状态，系统时钟来源于内部基准时钟。处理器退出复位状态后，执行下列操作：

- 从向量表偏移 0 处读取 SP(SP_main)初始值；
- 从向量表偏移 4 处读取程序计数器 (PC) 初始值；
- 连接寄存器 (LR) 设置为 0xFFFF_FFFF。

3.3.2.1 外部引脚复位（External Reset）

MCU 专用引脚，用于复位和重启 MCU 所有的模块。由于该引脚是低电平有效，建议在外部 PCB 中添加上拉电阻以防止噪声。

3.3.2.2 低压检测复位（LVD Reset）

集成了一个低压保护系统，以便在电源电压发生变化期间保护存储器内容和控制 MCU 系统状态。该系统由上电复位(POR)电路和 LVD 电路组成，LVD 可以配置为不同的复位基准，可以是高电平(VLVDH)或低电平(VLVDL)。

具体数值，请参考《ATC_AC7801x_Datasheet_CH》5.1.1 DC 特性章节。

3.3.2.3 ECC 2 bit 错误复位（2 Bit Error Reset）

配置 RESET_CTRL[23]=1 将使能 ECC 2 BIT Error Reset。当 ECC 检测到 2 BIT 错误后，将发出系统复位请求以产生系统复位。

3.3.2.4 PLL 失锁复位（PLL Unlock Reset）

配置 RESET_CTRL[22]=1 将使能 PLL Unlock Reset。当 PLL 检测到失锁错误时，将发出系统复位请求以产生系统复位。

3.3.2.5 看门狗定时器复位（Watchdog Reset）

看门狗定时器 (WDG)通过软件定时刷新来对系统进行监控。

如果周期性刷新没有出现，看门狗将发送系统复位。

更多细节的部分，请参考 20 看门狗模块（Watchdog）。

3.3.2.6 晶体振荡器（XOSC）监控器功能

配置 CKGEN_SRC_SEL[16]=1 使能 XOSC 监控系统。使能 XOSC 监控系统后，时钟检测器在 HSE 振荡器启动延迟后启用，并在此振荡器停止时被禁用。如果在 HSE 振荡器时钟上检测到故障，则该振荡器会被自动禁用，同时 XOSC 失效状态标志置起并且生成 NMI 中断以通知 MCU 执行相应的救援操作。

3.3.2.7 死锁复位（Lock up Reset）

死锁（Lock up） 表明内核软件出现严重错误。这是由于激活处理器内置系统状态保护硬件后出现无法恢复的异常情况而导致内核锁定的结果。

死锁（Lock up）出现时，可自动产生复位以恢复系统。

3.4 寄存器定义

表 3-1 复位（Reset）寄存器映射

CKGEN 基地址：0x40000000

地址	名称	宽度（位）	描述
CKGEN 基地址+0x0C	RESET_CTRL	32	芯片复位控制
CKGEN 基地址+0x10	RESET_STATUS	32	芯片复位状态

3.4.1 复位控制寄存器(RESET_CTRL)

表 3-2 RESET_CTRL 寄存器

RESET_CTRL　　芯片复位控制　　Reset:0x01418004

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称						XOSC_LOSS_RST_EN	CPU_LOCKUP_RST_EN	CPU_SYSRST_EN	ECC2_RST_EN	PLL_UNLOCK_RST_EN						
访问						RW	RW	RW	RW	RW						
Reset						0	0	1	0	1						
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									EXT_RST_DEGLITCH_VALUE							EXT_RST_DEGLITCH_EN
访问									RW							RW
Reset									0	0	0	0	0	1	0	0

字段	说明
26 XOSC_LOSS_RST_EN	**XOSC 异常复位使能** 1 : 能产生 IC 复位 0 : 不能产生 IC 复位
25 CPU_LOCKUP_RST_EN	**CPU 死锁复位使能** 1 ： 能产生 IC 复位 0 ： 不能产生 IC 复位
24 CPU_SYSRST_EN	**CPU 系统复位使能** 1：能产生 IC 复位 0：不能产生 IC 复位
23 ECC2_RST_EN	**ECC2 复位使能** 1：能产生 IC 复位 0：不能产生 IC 复位
22 PLL_UNLOCK_RST_EN	**PLL 失锁复位使能** 1：能产生 IC 复位 0：不能产生 IC 复位
7：1 EXT_RST_DEGLITCH_VALUE	**外部复位过滤基数** 以 32K 为计数时钟源的周期为基准
0 EXT_RST_DEGLITCH_EN	**外部复位的毛刺过滤功能使能** 1：使能 0：禁止

3.4.2 复位状态寄存器(RESET_STATUS)

表 3-3 RESET_STATUS 寄存器

RESET_STATUS 芯片复位状态 **Reset:0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																CLEAR_RESET_STATUS
访问																RW
Reset																0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称							XOSC LOSS STATUS	PLL_UNLOCK_RST_STATUS	CPU_LOCKUP_RST_STATUS	CPU_SYSRESET_STATUS	WDG_RESET_STATUS	WDG_32K_RESET_STATUS	ECC_RESET_STATUS	EXT_RESET_STATUS	LVD_RESET_STATUS	POR_RESET_STATUS
访问							R	R	R	R	R	R	R	R	R	R
Reset							0	0	0	0	0	0	0	0	0	0

字段	说明
16 CLEAR_ RESET_STATUS	**清除复位状态** 1：清除所有复位状态 0：允许复位状态更新
XOSC LOSS STATUS	**XOSC LOSS 状态** 1：有效 0：无效
8 PLL_UNLOCK_RST_STATUS	**PLL 失锁复位状态** 1：有效 0：无效
7 CPU_LOCKUP_ RST_STATUS	**CPU 锁定复位状态** 1：有效 0：无效
6 CPU_SYSRESET_STATUS	**CPU 系统复位状态** 1：有效 0：无效
5 WDG_RESET_STATUS	**看门狗正常模式下复位状态，高电平有效** 1：有效 0：无效

字段	说明
4 WDG_32K_RESET_STATUS	**看门狗低功耗模式下复位状态,，高电平有效** 1：有效 0：无效
3 ECC_RESET_STATUS	**ECC 2bit ERROR 复位状态** 1：有效 0：无效
2 EXT_RESET_STATUS	**外部引脚复位状态** 1：有效 0：无效
1 LVD_RESET_STATUS	**LVD 复位状态** 1：有效 0：无效
0 POR_RESET_STATUS	**POR 复位状态**, 1：有效 0：无效

4 时钟（Clock）

4.1 简介

时钟控制模块为 MCU 提供时钟源选择。 该模块包含一个锁相环（PLL）作为时钟源，PLL 可由内部或外部基准时钟作为基准。该模块可以控制此 PLL 时钟或内部/外部基准时钟之一作为 MCU 系统时钟源，进而生成所有模块的时钟和频率。

4.2 结构框图

4.2.1 时钟控制结构框图

该器件包含如下时钟源：

- 高速内部时钟（HSI）：内部 RC 振荡器提供 8MHz 时钟源
- 外部高速时钟（HSE）：外部 OSC 提供 4MHz ~30MHz 晶振
- 低速内部时钟（LSI）： 内部低速 RC OSC 提供 32kHz 时钟源
- 系统时钟 (SYSPLL)： 提供高达 48MHz 的高速时钟

每个外设都有专用的时钟使能信号来控制时钟的开关，请参阅 4.3 寄存器定义章节以了解详细地址。

【注意】

- 系统时钟高达 48MHz，部分型号可达 72MHz(具体型号可参考 MCU 选型手册)
- hclk(AHB)最高可达 48MHz，部分型号可达 72MHz(具体型号可参考 MCU 选型手册)
- pclk(APB)最高可达 48MHz。因此当 hclk 为 48MHz 时，APBCLK_DIV 可以配置为 1；当 hclk 为 72MHz 时，APBCLK_DIV 最小只能配置为 2。

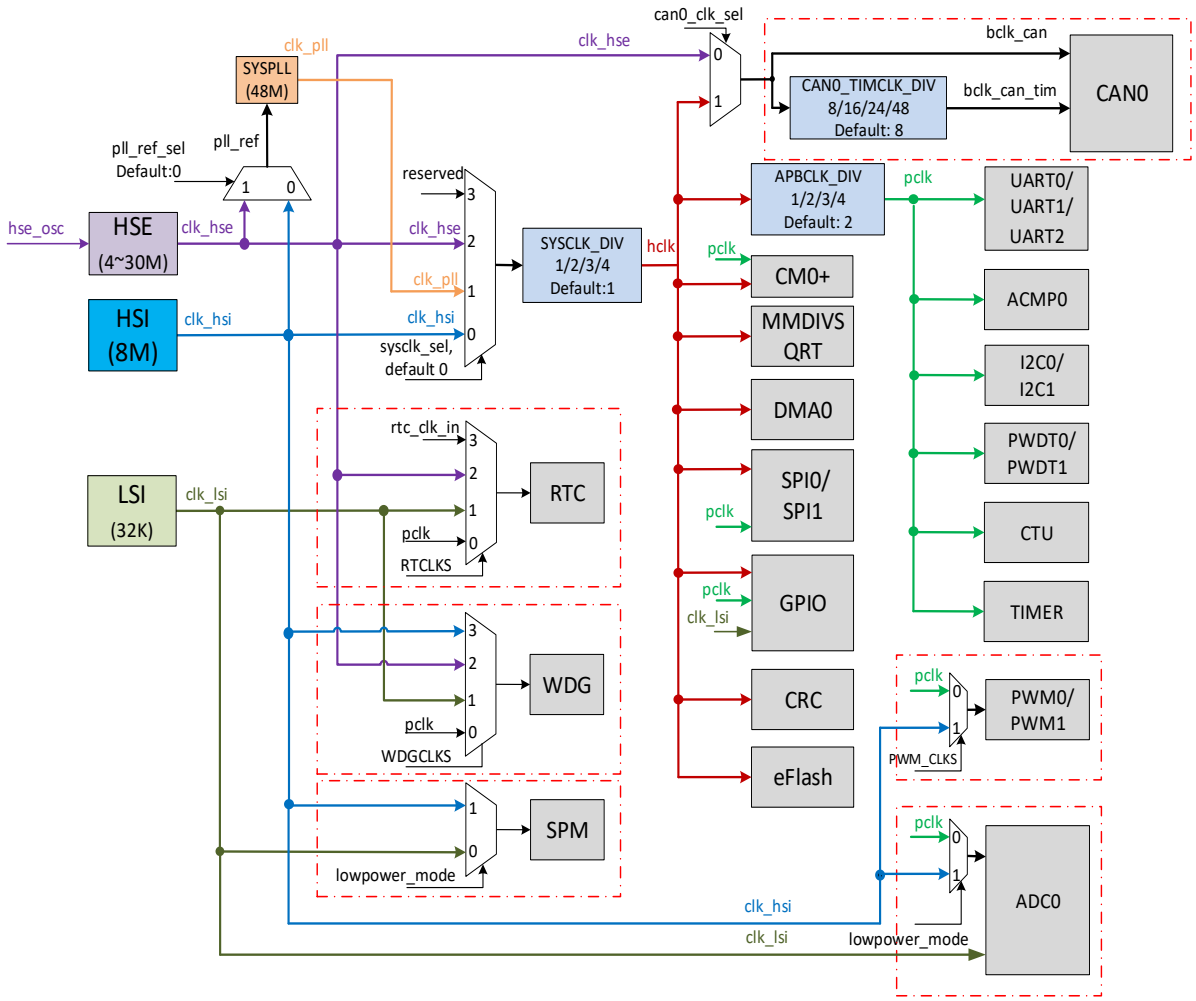

图 4-1 时钟控制结构

4.2.2 系统时钟示意图

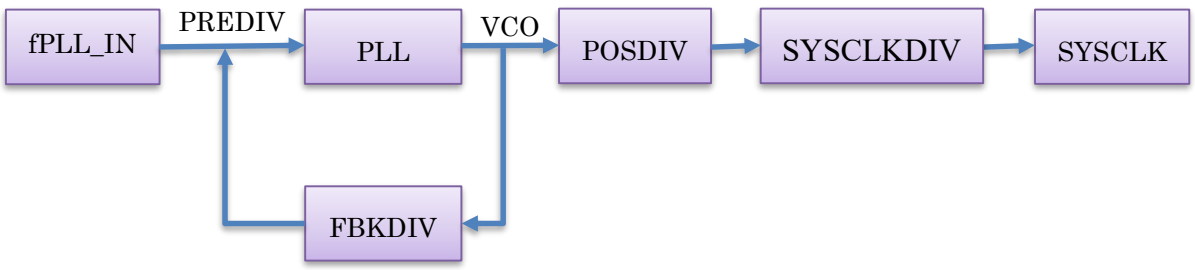

图 4-2 系统时钟示意图

- fPLL_IN，PLL 输入频率，支持 4 MHz ~ 30MHz；
- VCO = fPLL_IN* FBKDIV / PREDIV；
- 系统时钟 SYSCLK = VCO / POSDIV/SYSCLKDIV 。

【注意】

1. VCO 频率除以 POSDIV 不能高于 400MHz， VCO 的频率范围是 0.5~1.5GHz；
2. FBKDIV 推荐值为 48，64, 96, 192；
3. 外部晶振为 4MHz~30MHz，PLL 输入频率推荐值不小于 8MHz ；
4. 表中的 PREDIV,FBKDIV,POSDIV 为分频/倍频值，不是寄存器值。

表 4-1 典型 PLL 配置参考表

fPLL_IN	PREDIV	FBKDIV	POSDIV	Frequency	Sysclk_div	System Clock
8	1	108	6	144	2	72
8	1	96	16	48	1	48
8	1	96	24	32	1	32
8	1	96	16	48	2	24
8	1	96	16	48	3	16
8	1	96	16	48	4	12
4	1	216	12	72	1	72
4	1	192	16	48	1	48
4	1	192	24	32	1	32
4	1	192	16	48	2	24
4	1	192	16	48	3	16
4	1	192	16	48	4	12
30	2	48	10	72	1	72
30	2	64	20	48	1	48
30	2	64	30	32	1	32
30	2	64	20	48	2	24
30	2	64	20	48	3	16
30	2	64	20	48	4	12
12	2	144	12	72	1	72
12	2	96	12	48	1	48
12	2	96	18	32	1	32
12	2	96	24	24	1	24
12	2	96	18	32	2	16
12	2	96	24	24	2	12
16	2	144	16	72	1	72
16	2	96	16	48	1	48
16	2	96	24	32	1	32
16	2	96	16	48	2	24
16	2	96	24	32	2	16
16	2	96	16	48	4	12

【说明】标红的部分表示软件默认采用的配置。

4.3 寄存器定义

表 4-2 时钟寄存器映射

CKGEN 基地址：0x40000000

地址	名称	宽度	描述
CKGEN 基地址+0x00	CKGEN_CTRL	32	时钟控制寄存器
CKGEN 基地址+0x04	CKGEN_PERI_CLK_EN_0	32	外设时钟使能控制 0
CKGEN 基地址+0x08	CKGEN_PERI_CLK_EN_1	32	外设时钟使能控制 1
CKGEN 基地址+0x18	CKGEN_PERI_SFT_RST0	32	外设软件复位控制 1
CKGEN 基地址+0x1C	CKGEN_PERI_SFT_RST1	32	外设软件复位控制 2
CKGEN 基地址+0x8890	CKGEN_SYSPLL1_CFG0	32	SYSPLL1 配置寄存器 0
CKGEN 基地址+0x8894	CKGEN_SYSPLL1_CFG1	32	SYSPLL1 配置寄存器 1

4.3.1 控制寄存器(CKGEN_CTRL)

表 4-3 CKGEN_CTRL 寄存器

CKGEN_CTRL **时钟控制寄存器** **Reset: 0x00000100**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称						CAN0_CLK_SEL				CAN0_TIMCLK_DIV		PLL_REF_SEL				XOSC_MON_EN
访问						RW				RW		RW				RW
Reset						0				0	0	0				0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称							APBCLK_DIV				SYSCLK_DIV				SYSCLK_SEL	
访问							RW				RW				RW	
Reset							0	1			0	0			0	0

字段	说明
26 CAN0_CLK_SEL	**CAN0 时钟源选择** 0：外部振荡器时钟 1：AHB 分频时钟
22:21 CAN0_TIMCLK_DIV	**CAN0 time stamp 外部时钟由 CAN 时钟分频产生** 00 : 8 分频 01 : 16 分频 10 : 24 分频 11 : 48 分频
20	**PLL 参考时钟选择**

字段	说明
PLL_REF_SEL	
	1：参考时钟为外部振荡器
	0：参考时钟为内部振荡器
16 XOSC_MON_EN	**XOSC 监视器使能** 1：监视器功能使能 0：监视器功能禁用
9:8 APBCLK_DIV	**APB 时钟由系统时钟分频产生** 00 : 1 分频 01 : 2 分频 10 : 3 分频 11 : 4 分频
5:4 SYSCLK_DIV	**系统时钟分频器** 00： 1 分频 01： 2 分频 10： 3 分频 11： 4 分频
3:2 保留	保留
1:0 SYSCLK_SEL	**系统时钟源选择** 00： 内部振荡器 01： PLL 输出 10： 外部振荡器 11： 保留

4.3.2 外设时钟使能寄存器 0(CKGEN_PERI_CLK_EN_0)

表 4-4 CKGEN_PERI_CLK_EN_0 寄存器

CKGEN_PERI_CLK_EN_0 外设时钟使能控制 0 **Reset: 0x02800001**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称				CAN0_EN		CRC_EN	WDG_EN		GPIO_EN		DMA0_EN	RTC_EN	TIMER_EN			
访问				RW		RW	RW		RW		RW	RW	RW			
Reset				0		0	1		1		0	0	0			
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称				PWM1_EN	PWM0_EN	PWDT0_EN	I2C1_EN	I2C0_EN	SPI1_EN	SPI0_EN				UART2_EN	UART1_EN	UART0_EN
访问				RW	RW	RW	RW	RW	RW	RW				RW	RW	RW
Reset				0	0	0	0	0	0	0				0	0	1

【说明】写外设寄存器之前需使能对应外设的时钟，否则将产生 Hardfault。

字段	说明
31:29 保留	保留
28 CAN0_EN	**CAN0 时钟使能** 1： 时钟使能 0： 时钟禁用
27 保留	保留
26 CRC_EN	**CRC 时钟使能** 1： 时钟使能 0： 时钟禁用
25 WDG_EN	**WDG APB 时钟使能** 1： 时钟使能 0： 时钟禁用
24 保留	保留
23 GPIO_EN	**GPIO AHB 时钟使能** 1： 时钟使能 0： 时钟禁用
22 保留	保留
21	**DMA0 时钟使能**

字段	说明
DMA0_EN	
	1： 时钟使能
	0： 时钟禁用
20 RTC_EN	**RTC 时钟使能** 1： 时钟使能 0： 时钟禁用
19 TIMER_EN	**TIMER 时钟使能** 1： 时钟使能 0： 时钟禁用
18:13 保留	保留
12 PWM1_EN	**PWM1 定时器时钟使能** 1： 时钟使能 0： 时钟禁用
11 PWM0_EN	**PWM0 定时器时钟使能** 1： 时钟使能 0： 时钟禁用
10 PWDT0_EN	**PWDT0 时钟使能** 1： 时钟使能 0： 时钟禁用
9 I2C1_EN	**IIC1 时钟使能** 1： 时钟使能 0： 时钟禁用
8 I2C0_EN	**IIC0 时钟使能** 1： 时钟使能 0： 时钟禁用
7 SPI1_EN	**SPI1 时钟使能** 1： 时钟使能 0： 时钟禁用
6 SPI0_EN	**SPI10 时钟使能** 1： 时钟使能 0： 时钟禁用
5:3 保留	保留
2	**UART2 时钟使能**

字段	说明
UART2_EN	1： 时钟使能 0： 时钟禁用
1 UART1_EN	**UART1 时钟使能** 1： 时钟使能 0： 时钟禁用
0 UART0_EN	**UART0 时钟使能** 1： 时钟使能 0： 时钟禁用

4.3.3 外设时钟使能寄存器 1(CKGEN_PERI_CLK_EN_1)

表 4-5 CKGEN_PERI_CLK_EN_1 寄存器

CKGEN_PERI_CLK_EN_1 **外设时钟使能控制 1** **Reset:0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称											PWDT1_EN		ACMP0_EN	ADC0_EN	CTU_EN	
访问											RW		RW	RW	RW	
Reset											0		0	0	0	

【说明】写外设寄存器之前需使能对应外设的时钟，否则将产生 Hardfault。

字段	说明
5 PWDT1_EN	**PWDT1 时钟使能** 1： 时钟使能 0： 时钟禁用
4 保留	保留
3 ACMP0_EN	**ACMP0 时钟使能** 1： 时钟使能 0： 时钟禁用
2 ADC0_EN	**ADC0 core 时钟使能** 1： 时钟使能 0： 时钟禁用

字段	说明
1 CTU_EN	**CTU APB 时钟使能** 1： 时钟使能 0： 时钟禁用
0 保留	保留

4.3.4 外设复位寄存器 0(CKGEN_PERI_SFT_RST0)

表 4-6 CKGEN_PERI_SFT_RST0 寄存器

CKGEN_PERI_SFT_RST0 **外设软件复位控制 0** **Reset:0x02900001**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称				SRST_CAN0		SRST_CRC	SRST_WDG		SRST_GPIO		SRST_DMA0	SRST_RTC	SRST_TIMER			
访问				RW		RW	RW		RW		RW	RW	RW			
Reset				0		0	1		1		0	1	0			
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称				SRST_PWM1	SRST_PWM0	SRST_PWDT0	SRST_I2C1	SRST_I2C0	SRST_SPI1	SRST_SPI0				SRST_UART2	SRST_UART1	SRST_UART0
访问				RW	RW	RW	RW	RW	RW	RW				RW	RW	RW
Reset				0	0	0	0	0	0	0				0	0	1

字段	说明
31:29 保留	保留
28 SRST_CAN0	**CAN0 软件复位** 0： 复位有效 1： 复位无效
27 保留	保留
26 SRST_CRC	**CRC 软件复位** 0： 复位有效 1： 复位无效

字段	说明
25 SRST_WDG	**Watch dog 定时器软件复位** 0： 复位有效 1： 复位无效
24 保留	保留
23 SRST_GPIO	**GPIO AHB 软件复位** 0： 复位有效 1： 复位无效
22 保留	保留
21 SRST_DMA0	**DMA0 软件复位** 0： 复位有效 1： 复位无效
20 SRST_RTC	**RTC 软件复位** 0： 复位有效 1： 复位无效
19 SRST_TIMER	**TIMER 软件复位** 0： 复位有效 1： 复位无效
18:13 保留	保留
12 SRST_PWM1	**PWM1 软件复位** 0： 复位有效 1： 复位无效
11 SRST_PWM0	**PWM0 软件复位** 0： 复位有效 1： 复位无效
10 SRST_PWDT0	**PWDT0 软件复位** 0： 复位有效 1： 复位无效
9 SRST_I2C1	**IIC1 软件复位** 0： 复位有效 1： 复位无效

字段	说明
8 SRST_I2C0	**IIC0 软件复位** 0： 复位有效 1： 复位无效
7 SRST_SPI1	**SPI1 软件复位** 0： 复位有效 1： 复位无效
6 SRST_SPI0	**SPI0 软件复位** 0： 复位有效 1： 复位无效
5:3 保留	保留
2 SRST_UART2	**UART2 软件复位** 0： 复位有效 1： 复位无效
1 SRST_UART1	**UART1 软件复位** 0： 复位有效 1： 复位无效
0 SRST_UART0	**UART0 软件复位** 0： 复位有效 1： 复位无效

4.3.5 外设复位寄存器 1(CKGEN_PERI_SFT_RST1)

表 4-7 CKGEN_PERI_SFT_RST1 寄存器

CKGEN_PERI_SFT_RST1 **外设软件复位控制 1** **Reset:0x00000010**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称											SRST_PWDT1	SRST_ANA_REG	SRST_ACMP0	SRST_ADC0	SRST_CTU	
访问											RW	RW	RW	RW	RW	
Reset											0	1	0	0	0	

字段	说明
5 SRST_PWDT1	**PWDT1 软件复位** 0： 复位有效 1： 复位无效
4 SRST_ANA_REG	**ANA 寄存器软复位** 0： 复位有效 1： 复位无效
3 SRST_ACMP0	**ACMP0 软件复位** 0： 复位有效 1： 复位无效
2 SRST_ADC0	**ADC0 软件复位** 0： 复位有效 1： 复位无效
1 SRST_CTU	**CTU 软件复位** 0： 复位有效 1： 复位无效
0	保留

4.3.6 PLL 配置寄存器 0(CKGEN_SYSPLL1_CFG0)

表 4-8 CKGEN_SYSPLL1_CFG0 寄存器

CKGEN_SYSPLL1_CFG0 **SYSPLL1 配置寄存器 0** **Reset: 0x10301030**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	SYSPLL1_PREDIV		SYSPLL1_POSDIV							SYSPLL1_FBKDIV						
访问	RW		RW							RW						
Reset	0	0	0	1	0	0	0			0	1	1	0	0	0	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称							SYSPLL1_MONREF_EN									
访问							RW									
Reset	0						0									

字段	说明
31：30 SYSPLL1_PREDIV	**预分频比** 00： Fref = Fin/1 01： Fref = Fin/2 1X： Fref = Fin/4

字段	说明
29：25 SYSPLL1_POSDIV	**单端时钟输出的分频比** 00000： 无 00001： VCO/2 00010： VCO/4 00011： VCO/6 …… 11110： VCO/60 11111： VCO/62
22：15 SYSPLL1_FBKDIV	**反馈分频比** 8'd6： /6 8'd255： /255
9 SYSPLL1_MONREF_EN	**PLL Monitor 参考时钟使能** 0： 关闭 1： 使能

4.3.7 PLL 配置寄存器 1(CKGEN_SYSPLL1_CFG1)

表 4-9 CKGEN_SYSPLL1_CFG1 寄存器

CKGEN_SYSPLL1_CFG1　　SYSPLL1 配置寄存器 1　　Reset:0x00F000EE

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称								SYSPLL1_LD_EN					SYSPLL1_LD_DLY_SEL			
访问								RW					RW			
Reset								0					1	1	1	

字段	说明
8 SYSPLL1_LD_EN	**PLL Lockup Detector 使能** 0： 关闭 1： 使能
3：1 SYSPLL1_LD_DLY_SEL	**Lockup Detector 比较时间选择** 000： 300ps 001： 600ps 010： 900ps

字段	说明
	011： 1200ps 100： 2200ps 101： 3200ps 110： 4200ps 111： 5200ps

5 电源模式（Power Modes）

5.1 简介

本章介绍 AC7801x 各种电源模式及其各个模块在这些模式下的功能。

5.2 功能描述

支持运行（Run）、休眠（Sleep）、停止（Stop）和待机（Standby）模式。运行（Run）、休眠（Sleep）、停止（Stop）模式下能保持 I/O 状态。待机（Standby）模式下，I/O 处于关闭状态，引脚状态由外部电路决定。建议加上拉/下拉以确定其在 Standby 模式下的状态。

- 运行模式 – CPU 时钟可在全速状态下运行。
- 休眠模式 – CPU 进入休眠模式，系统时钟和总线时钟仍在运行。
- 停止模式 – CPU 进入深度休眠模式，部分模块能够唤醒 CPU。
- 待机模式 – CPU 和各个模块被关闭， RTC 和 NMI 引脚 可以唤醒 CPU。

5.3 应用说明

5.3.1 进入和退出低功耗模式

1. 使用 **SPM_EN_PERIPH_WKUP** 使能需要的唤醒源；
2. 将已使能的模块去能;
3. 在 WFI 指令前使用 **SPM_PWR_MGR_CFG0[SLEEP_MODE]**设置功耗模式，配置如下：
   1) 2’b01： 停止模式
   2) 2’b1x： 待机模式
4. 调用 WFI 指令进入功耗模式;
5. 处理器通过中断退出低功耗模式;
6. 重新将模块使能。

说明：

1. GPIO/RTC/WDG 不需要去能。
2. Stop 模式下，UART/CAN 模块关闭不会影响唤醒。

5.3.2 低功耗模式下的模块操作

下表说明了该芯片处于各低功耗模式时每个模块的功能，表中显示了标准特性及某些例外情况。

表 5-1 低功耗模式下的模块功能

模块	休眠模式	停止模式	待机模式
CM0+	待机	待机	关闭
MMDIVSQRT	开启	关闭	关闭
SRAM	开启	待机	关闭
片内 Flash	开启	关闭	关闭
I2C	开启	可选开启 [1]	关闭
SPI	开启	可选开启 [2]	关闭
WDG	开启	可选开启	关闭
PWDT	开启	关闭	关闭
UART	开启	待机 [3]	关闭
DMA	开启	关闭	关闭
TIMER	开启	关闭	关闭
PWM	开启	关闭	关闭
CRC	开启	关闭	关闭
CTU	开启	关闭	关闭
CAN	开启	待机 [4]	关闭
RTC	开启	可选开启	可选开启
SPM	开启	始终开启	始终开启
PLL	开启	关闭	关闭
XOSC	开启	关闭	关闭
HSI(8MHz)	开启	可选开启 [5]	关闭
LSI(32kHz)	开启	始终开启	始终开启
GPIO	开启	始终开启 [6]	关闭 [6]
ADC	开启	可选开启 [5]	关闭
LVD	可选开启	可选开启	可选开启
PVD	可选开启	可选开启 [7]	关闭
ACMP	可选开启	可选开启 [8]	关闭
DAC	可选开启	可选开启	关闭
T-sensor	可选开启	关闭	关闭
DIGLDO	开启	低功耗	低功耗，CM0 关闭
FLHLDO	开启	关闭	关闭
POR	开启	开启	开启
BG	开启	可选开启 [9]	可选开启

【注意】

1：支持停止模式下的地址匹配唤醒。

2：支持停止模式下的从机模式接收和唤醒。

3：支持停止模式下的边沿唤醒(UART 管脚低电平直接走 SPM)。

4：支持停止模式下的边沿唤醒，可选择开启过滤器。

5：支持停止模式下的模拟监控器唤醒。

6：停止模式 I/O 状态保持，支持所有 GPIO 中断唤醒，待机模式只支持 NMI pin 唤醒。

7：停止模式支持 PVD Warning 中断唤醒。

8：停止模式支持 ACMP 设定电压比较唤醒。

9：在停止模式使能 ADC 唤醒，或者使能 LVD，PVD 则 BG 开启，否则关闭 BG。

【说明】

- 开启：表示模块的 Power 和 Clock 均正常提供。
- 待机：表示模块的 Power 正常，Clock 关闭。
- 关闭：表示模块的 Power 和 Clock 均关闭。

6 系统电源管理（System Power Management）

6.1 简介

系统电源管理（SPM）为软件开发人员提供了灵活的系统管理，包含休眠/唤醒功能、电源域管理和各模块功耗控制。

6.2 特性

- 支持停止（Stop）模式下电源管理
- 支持待机（Standby）模式下电源管理

6.3 应用说明

6.3.1 SPM 电源控制编程指南

AC7801x 支持停止（Stop）模式和待机（Standby）模式。

停止（Stop）模式下，各模块电源工作状态以及唤醒源可参考表 5-1。

待机（Standby）模式下，除 RTC、SPM 本身外，所有数字模块均断电。

WFI 指令调用芯片的停止和待机模式，处理器通过中断指令退出低功耗模式。

编程顺序：

1. 配置唤醒源正常工作，并能正常产生中断；
2. 设置唤醒源：SPM_EN_PERIPH_WKUP；
3. 使能 SPM 电源控制：PWR_EN；
4. 编程 SPM 配置寄存器确定功耗模式：SLEEP_MODE；
5. 执行 WFI 指令。

6.3.2 晶体振荡器（XOSC）/系统时钟（SYSPLL） 电源控制

XOSC/SYSPLL 默认处于关闭状态。需要时，通过配置 SPM_PWR_MGR_CFG1，打开或关闭 XOSC/SYSPLL。

SPM 寄存器 SPM_PWR_MGR_CFG1：

- **XOSC_HSEON**：外部高速时钟使能。
- **XOSC_HSEBYP**：外部高速时钟旁路。

- **SYSPLL_ON**：SYSPLL 使能。

当相应的位设置为 1'b1 时，SPM 将按照上电顺序为 XOSC/PLL 供电，可能需要花上一些时间。因此，在使用之前，软件需要等待 XOSC/SYSPLL 上电完成和时钟就绪。

XOSC/PLL 上电状态可通过读取 SPM 寄存器 SPM_PWR_MGR_CFG1 来确定。

- **XOSC_RDY**：外部高速时钟就绪标志。
- **SYSPLL_RDY**：SYSPLL 时钟就绪标志。

例如，在时钟源切换到 PLL 时钟之前，应首先为 SYSPLL 供电，并等待 SYSPLL 时钟稳定。

当芯片从停止模式唤醒时，SPM 将使 XOSC/SYSPLL 保持打开或关闭状态，与休眠前的状态相同。 但是从待机模式唤醒时，XOSC/SYSPLL 将关闭。

6.4 寄存器定义

表 6-1 SPM 寄存器映射

SPM 基地址： 0x40008000

地址	名称	宽度	描述
SPM 基地址 + 0x00	SPM_PWR_MGR_CFG0	32	电源管理器配置寄存器 0
SPM 基地址 + 0x04	SPM_PWR_MGR_CFG1	32	电源管理器配置寄存器 1
SPM 基地址 + 0x0C	SPM_PERIPH_SLEEP_ACK_STATUS	32	外设休眠应答状态
SPM 基地址 + 0x10	SPM_EN_PERIPH_SLEEP_ACK	32	外设休眠应答使能寄存器
SPM 基地址+ 0x14	SPM_EN_PERIPH_WKUP	32	外设唤醒使能寄存器
SPM 基地址+ 0x1C	SPM_WAKEUP_IRQ_STATUS	32	唤醒状态标志寄存器

6.4.1 电源管理器配置寄存器 0(SPM_PWR_MGR_CFG0)

表 6-2 SPM_PWR_MGR_CFG0 寄存器

SPM_PWR_MGR_CFG0 电源管理配置寄存器 0 **Reset:0x00000118**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称							SLEEP_MODE	EN_IO_SUS			EN_CAN0_FILTER	EN_LVD	EN_DPWRLVD	EN_PVD	EN_FAST_BOOT	PWR_EN
访问							RW		RW		RW	RW	RW	RW	RW	RW
Reset							0	1	0		0	1	1	0	0	0

字段	说明
31：10 保留	保留
9：8 SLEEP_MODE	**休眠模式** 01：停止模式 1x：待机模式
7 EN_IO_SUS	**在停止模式禁用 IO** 1：在进入停止（Stop）时，禁用 I/O 该控制位在 待机模式下并不生效，硬件自动挂起 IO 0：在进入停止（Stop）时，I/O 状态保持
6 保留	保留
5 EN_CAN0_FILTER	**使能 CAN0 唤醒中断过滤器** 1：使能 在使能 spm 时，会使用模拟滤波器后的中断作为 CAN0 唤醒中断 0：禁用
4 EN_LVD	**使能 LVD 低电压检测** 1：使能 检测芯片 VCC 电压，如果 VCC 欠压，触发 LVD 复位 0：禁用
3 EN_DPWRLVD	**使能 LDO 低电压检测** 1：使能 检测芯片内部 LDO 电压，如果 LDO 欠压，同样触发 LVD 复位 0：禁用
2 EN_PVD	**使能可编程电压检测** 1：使能 检测芯片 VCC 电压，如果 VCC 欠压，触发 PVD 中断 0：禁用
1 EN_FAST_BOOT	**使能快速唤醒启动模式** 1：使能 快速唤醒启动模式：在休眠过程中，在接受到唤醒中断时，芯片会停止休眠时序并立即唤醒 0：禁用
0 PWR_EN	**SPM 电源控制使能** 1：使能 SPM 电源控制 0：禁用

6.4.2 电源管理器配置寄存器 1 (SPM_PWR_MGR_CFG1)

表 6-3 SPM_PWR_MGR_CFG1 寄存器

SPM_PWR_MGR_CFG1 电源管理器配置寄存器 1 **Reset:0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	XOSC_RDY	SYSPLL_RDY	XOSC_HSEON	XOSC_HSEBYP	SYSPLL_ON											
访问	R/W	R/W	R/W	R/W	R/W											
Reset	0	0	0	0	0											
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称													LVDPVD			
访问													RW			
Reset													0	0	0	0

字段	说明
31 XOSC_RDY	**XOSC 时钟就绪标志** 1：就绪 0：未就续
30 SYSPLL_RDY	**PLL 时钟就绪标志** 1：就绪 0：未就续
29 XOSC_HSEON	**外部高速时钟使能** 1：使能 XOSC 0：禁用 XOSC
28 XOSC_HSEBYP	**外部高速时钟旁路** 1：使用外部时钟旁路振荡器 0：禁用外部时钟旁路振荡器
27 SYSPLL_ON	**SYSPLL 使能** 1：使能 SYSPLL 0：禁用 SYSPLL
26：7 保留	保留
6：4 保留	保留
3：0 LVDPVD	**LVDPVD 设置** 0000: 保留 0011： VLVDL = 2.60±0.1V VPVDL = 2.95±0.1V 1011： VLVDH = 4.2±0.15V VPVDH = 4.6±0.15V

6.4.3 外设休眠应答状态(SPM_PERIPH_SLEEP_ACK_STATUS)

表 6-4 SPM_PERIPH_SLEEP_ACK_STATUS 寄存器

SPM_PERIPH_SLEEP_ACK_STATUS 外设休眠应答状态 **Reset:0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称														EFLASH		ADC0
访问														R		R
Reset														0		0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	DMA0				UART2	UART1	UART0		CAN0		SPI1	SPI0	I2C1	I2C0		ACMP0
访问	R				R	R	R		R		R	R	R	R		R
Reset	0				0	0	0		0		0	0	0	0		0

字段	说明
31：19 保留	保留
18 EFLASH	**片内 EFLASH 空闲(idle)状态** 1：空闲 0：忙
17 保留	保留
16 ADC0	**ADC0 休眠确认（sleep ack） 状态** 1：已回 ACK 0：未回 ACK
15 DMA0	**DMA0 休眠确认（sleep ack） 状态** 1：已回 ACK 0：未回 ACK
14 保留	保留
13 保留	保留
12 保留	保留
11 UART2	**UART2 休眠确认（sleep ack） 状态** 1：已回 ACK 0：未回 ACK

字段	说明
10 UART1	**UART1 休眠确认（sleep ack） 状态** 1：已回 ACK 0：未回 ACK
9 UART0	**UART0 休眠确认（sleep ack） 状态** 1：已回 ACK 0：未回 ACK
8 保留	保留
7 CAN0	**CAN0 休眠确认（sleep ack） 状态** 1：已回 ACK 0：未回 ACK
6 保留	保留
5 SPI1	**SPI1 休眠确认（sleep ack） 状态** 1：已回 ACK 0：未回 ACK
4 SPI0	**SPI0 休眠确认（sleep ack） 状态** 1：已回 ACK 0：未回 ACK
3 I2C1	**I2C1 休眠确认（sleep ack） 状态** 1：已回 ACK 0：未回 ACK
2 I2C0	**I2C0 休眠确认（sleep ack） 状态** 1：已回 ACK 0：未回 ACK
1 保留	保留
0 ACMP0	**ACMP0 休眠确认（sleep ack） 状态** 1：已回 ACK 0：未回 ACK

6.4.4 外设休眠应答使能(SPM_EN_PERIPH_SLEEP_ACK)

表 6-5 SPM_EN_PERIPH_SLEEP_ACK 寄存器

SPM_EN_PERIPH_SLEEP_ACK 外设休眠应答使能寄存器 **Reset:0x00058EBD**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称														EFLASH		ADC0
访问														R/W		R/W
Reset														1		1
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	DMA0				UART2	UART1	UART0		CAN0		SPI1	SPI0	I2C1	I2C0		ACMP0
访问	R/W				R/W	R/W	R/W		R/W		R/W	R/W	R/W	R/W		R/W
Reset	1				1	1	1		1		1	1	1	1		1

字段	说明
31：18 保留	保留
18 EFLASH	**使能 eflash 休眠确认等待（Sleep ACK Waiting）** 1：使能 0：禁用
17 保留	保留
16 ADC0	**使能 ADC0 休眠确认等待（Sleep ACK Waiting）** 1：使能 0：禁用
15 DMA0	**使能 DMA0 休眠确认等待（Sleep ACK Waiting）** 1：使能 0：禁用
14 保留	保留
13 保留	保留
12 保留	保留
11 UART2	**使能 UART2 休眠确认等待（Sleep ACK Waiting）** 1：使能 0：禁用
10 UART1	**使能 UART1 休眠确认等待（Sleep ACK Waiting）** 1：使能 0：禁用
9 UART0	**使能 UART0 休眠确认等待（Sleep ACK Waiting）**

字段	说明
	1：使能 0：禁用
8 保留	保留
7 CAN0	**使能 CAN0 休眠确认等待（Sleep ACK Waiting）** 1：使能 0：禁用
6 保留	保留
5 SPI1	**使能 SPI1 休眠确认等待（Sleep ACK Waiting）** 1：使能 0：禁用
4 SPI0	**使能 SPI0 休眠确认等待（Sleep ACK Waiting）** 1：使能 0：禁用
3 I2C1	**使能 I2C1 休眠确认等待（Sleep ACK Waiting）** 1：使能 0：禁用
2 I2C0	**使能 I2C0 休眠确认等待（Sleep ACK Waiting）** 1：使能 0：禁用
1 保留	保留
0 ACMP0	**使能 ACMP0 休眠确认等待（Sleep ACK Waiting）** 1：使能 0：禁用

6.4.5 外设唤醒使能寄存器(SPM_EN_PERIPH_WKUP)

表 6-6 SPM_EN_PERIPH_WKUP 寄存器

SPM_EN_PERIPH_WKUP 外设唤醒使能寄存器 **Reset:0x00028000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称													PVD	NMI	GPIO	ADC0
访问													R/W	R/W	R/W	R/W
Reset													0	0	1	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	RTC				UART2	UART1	UART0		CAN0		SPI1	SPI0	I2C1	I2C0		ACMP0
访问	R/W				R/W	R/W	R/W		R/W		R/W	R/W	R/W	R/W		R/W
Reset	1				0	0	0		0		0	0	0	0		0

字段	说明
31：20 保留	保留
19 PVD	**使能 PVD 报警唤醒** 1： 使能 禁用时，不支持 PVD 报警中断唤醒并且该唤醒会被忽略 0：禁用
18 NMI	**使能 NMI 唤醒** 1：使能 0：禁用
17 GPIO	**使能 GPIO 唤醒** 1：使能 0：禁用
16 ADC0	**使能 ADC0 唤醒** 1：使能 0：禁用
15 RTC	**使能 RTC 唤醒** 1：唤醒 0：禁用
14 保留	保留
13 保留	保留
12 保留	保留

字段	说明
11 UART2	**使能 UART2 唤醒** 1：使能 0：禁用
10 UART1	**使能 UART1 唤醒** 1：使能 0：禁用
9 UART0	**使能 UART0 唤醒** 1：使能 0：禁用
8 保留	保留
7 CAN0	**使能 CAN0 唤醒** 1：使能 0：禁用
6 保留	保留
5 SPI1	**使能 SPI1 唤醒** 1：使能 0：禁用
4 SPI0	**使能 SPI0 唤醒** 1：使能 0：禁用
3 I2C1	**使能 I2C1 唤醒** 1：使能 0：禁用
2 I2C0	**使能 I2C0 唤醒** 1：使能 0：禁用
1 保留	保留
0 ACMP0	**使能 ACMP0 唤醒** 1：使能 0：禁用

6.4.6 唤醒状态标志寄存器(SPM_WAKEUP_IRQ_STATUS)

表 6-7 SPM_WAKEUP_IRQ_STATUS 寄存器

SPM_WAKEUP_IRQ_STATUS 唤醒状态标志寄存器 **Reset:0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称												OVER_COUNT	PVD	NMI	GPIO	ADC0
访问												W1C	W1C	W1C	W1C	W1C
Reset												0	0	0	0	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	RTC				UART2	UART1	UART0		CAN0		SPI1	SPI0	I2C1	I2C0		ACMP0
访问	W1C				W1C	W1C	W1C		W1C		W1C	W1C	W1C	W1C		W1C
Reset	0				0	0	0		0		0	0	0	0		0

字段	说明
31：21 保留	保留
20 OVER_COUNT	**SPM over count 唤醒标志位** 1：进入休眠模式前， SPM 会在特定的周期内等待所有外设应答休眠确认。如果有外设 NO ACK，则该标志位置位，并退出休眠模式，产生 SPM 中断。 0：无效 **注意：写 “1” 清零此位**
19 PVD	**PVD 唤醒标志位** 1：有效 0：无效 **注意：写 “1” 清零此位**
18 NMI	**NMI 唤醒标志位** 1：有效 0：无效 **注意：写 “1” 清零此位**
17 GPIO	**GPIO 唤醒标志位** 1：有效 0：无效 **注意：写 “1” 清零此位**
16 ADC	**ADC 唤醒标志位**

字段	说明
	1：有效 0：无效 **注意：写 “1” 清零此位**
15 RTC	**RTC 唤醒标志位** 1：有效 0：无效 **注意：写 “1” 清零此位**
14 保留	保留
13 保留	保留
12 保留	保留
11 UART2	**UART2 唤醒标志位** 1：有效 0：无效 **注意：写 “1” 清零此位**
10 UART1	**UART1 唤醒标志位** 1：有效 0：无效 **注意：写 “1” 清零此位**
9 UART0	**UART0 唤醒标志位** 1：有效 0：无效 **注意：写 “1” 清零此位**
8 保留	保留
7 CAN0	**CAN0 唤醒标志位** 1：有效 0：无效 **注意：写 “1” 清零此位**
6 保留	保留

字段	说明
5 SPI1	**SPI1 唤醒标志位** 1：有效 0：无效 **注意：写 “1” 清零此位**
4 SPI0	**SPI0 唤醒标志位** 1：有效 0：无效 **注意：写 “1” 清零此位**
3 I2C1	**I2C1 唤醒标志位** 1：有效 0：无效 **注意：写 “1” 清零此位**
2 I2C0	**I2C0 唤醒标志位** 1：有效 0：无效 **注意：写 “1” 清零此位**
1 保留	保留
0 ACMP0	**ACMP0 唤醒标志位** 1：有效 0：无效 **注意：写 “1” 清零此位**

7 控制器局域网（CAN）

7.1 简介

7.1.1 CAN-CTRL 内核

CAN-CTRL 内核是一个基于 CAN 协议执行串行通信的串行通信控制器。CAN 总线接口使用基本的 CAN 原理，并符合 CAN 2.0B 规范。此外可以可将 CAN 内核配置为满足 CAN 规范的灵活数据速率的 CAN FD。

CAN 协议使用多主机总线配置来传输网络节点之间的帧（通信对象），并管理错误处理而不会给 CPU 带来任何负担。CAN 控制器使用户能够在各种组件之间建立可靠的链接。CAN 控制器在微控制器中映射为 I/O 设备。CPU 通过系统总线访问 CAN 以控制帧的传输或接收。CAN 总线的连接如图 7-1 所示。

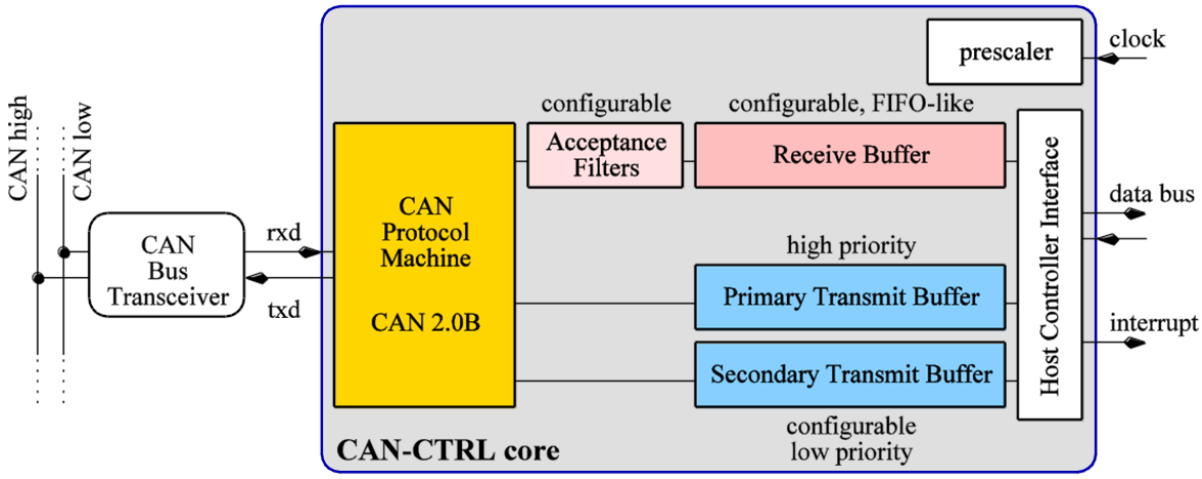

图 7-1 CAN 总线连接及 CAN-CTRL 内核主要功能示意图

7.1.2 CAN 协议

CAN 通信按帧组织。共有两种类型的帧结构：标准帧和扩展帧。对于 CAN 2.0，一帧最大传输数据为 8 个字节，而 CAN FD 一帧最多传输 64 个字节。所有 CAN 节点在总线访问方面都是相同的。

使用消息标识符完成数据寻址。在 CAN 网络中，只有一个节点必须传输具有特定标识符的消息。所有节点都接收所有消息，并且节点主控制器必须确定它是否有适当的消息标识符寻址。为了减少主控制器的负载，CAN 控制器可以使用接收过滤器。这些过滤器将所有接收的消息标识符与用户可选择的位模式进行比较。仅当消息通过接收过滤器时，它才会存储在接收缓冲区中并通知主控制器。

CAN 帧标识符也用于总线仲裁。当具有较高优先级标识符的消息由另一个 CAN 节点发送时，CAN 协议控制器停止发送具有低优先级标识符的消息。CAN 协议控制器自动尝试在下一个可能的发送位置重新发送已停止的消息。

CAN 2.0B 定义了高达 1Mbit/s 的数据比特率。CAN FD 没有固定限制，CAN FD 标准定义了比特率的切换。如果启用，帧有效载荷可以较高的速度传输，而帧头以较低的速度发送。

7.2 特性

- 支持 CAN 规格
  - CAN 2.0A/B (最多 8 个字节的有效载荷，经 Bosch 参考模型验证)
  - 对 CAN FD 的可选支持

    (最多 64 字节的有效载荷，ISO 11898-1:2015 或者 非 ISO Bosch)
- 可编程的比特率
  - CAN 2.0B 支持最高 1Mbit/s
  - CAN FD 支持最高 8Mbit/s(受收发器和所选择的 CAN 控制器时钟频率的限制)
  - 可选择 AHB 分频时钟或者外部振荡器时钟
- 可编程波特率预分频器(1 至 1/256)
- 1 个接收缓冲区，FIFO 深度为 7
- 两个发送缓冲区：
  - 主发送缓冲区 (PTB) FIFO 深度为 1
  - 次发送缓冲区 (STB) FIFO 深度为 3，按 FIFO 或优先级决定数据帧发送的先后顺序
- 16 个独立可编程的内部 29 位接收滤波器
- 扩展特性：
  - 单次发送模式 (PTB 和 STB 都支持)
  - 监听模式
  - 回环模式 (内部、外部)
  - 收发器待机模式
- 扩展状态和错误报告：
  - 捕获最后发生错误的类型和仲裁丢失的位置
  - 可编程错误警告限制
- 可配置的中断资源
- 时间戳
  - CiA 603 时间戳
- 带集成式低通滤波器的唤醒功能

7.3 应用说明

7.3.1 消息缓冲区

7.3.1.1 消息缓冲区的基本概念

消息缓冲区的概念如下图所示，所有缓冲区都足够大到可以存储最大长度的帧。

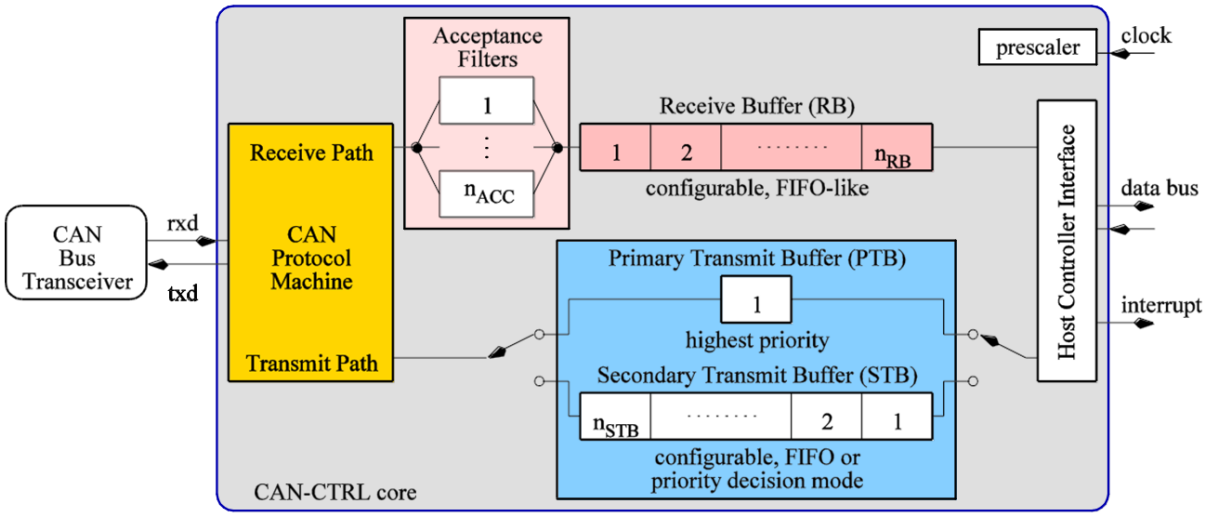

图 7-2 消息缓冲区示意图

7.3.1.2 接收缓冲区

为了减少主控制器接收帧的负载，CAN 控制器使用接收过滤器。CAN 控制器接收数据时，接收过滤器会检查消息标识符。如果接收到的帧与其中一个接收过滤器的过滤条件相匹配，则它将被存储在接收缓冲区（RB）中，该接收缓冲区具有类似 FIFO 的行为。

根据可用消息缓冲区的数量，主控制器不需要立即读取传入消息。CAN 控制器能够在每个接收信息上生成中断。如果用户使能了对应的“满”中断使能寄存器 RFIF 或者”几乎满”寄存器 RAFIE，当接收缓冲区(RB)“已满”或“几乎满”时，CAN 控制器也会产生对应的中断。由于类似 FIFO 的行为，主控制器始终从 RB 读取最旧的消息。

表 7-1 接收缓冲区寄存器 RBUF – 标准格式 (r-0)

地址	位								功能
	7	6	5	4	3	2	1	0	
RBUF	ID(7:0)								标识符
RBUF+1						ID(10:8)			标识符
RBUF+2									标识符
RBUF+3	ESI								标识符

地址	位								功能
	7	6	5	4	3	2	1	0	
RBUF+4	IDE=0	RTR	FDF	BRS	DLC(3:0)				控制位
RBUF+5	KOER			TX					状态位
RBUF+8	d1(7:0)								数据字节 1
RBUF+9	d2(7:0)								数据字节 2
…	…								…
RBUF+71	d64(7:0)								数据字节 64
RBUF+72	RTS(7:0)								CiA 603
…	…								…
RBUF+75	RTS(31:24)								CiA 603

表 7-2 接收缓冲区 寄存器 RBUF – 扩展格式 (r-0)

地址	位								功能
	7	6	5	4	3	2	1	0	
RBUF	ID(7:0)								标识符
RBUF+1	ID(15:8)								标识符
RBUF+2	ID(23:16)								标识符
RBUF+3	ESI			ID(28:24)					标识符
RBUF+4	IDE=1	RTR	FDF	BRS	DLC(3:0)				控制位
RBUF+5	KOER			TX					状态位
RBUF+8	d1(7:0)								数据字节 1
RBUF+9	d2(7:0)								数据字节 2
…	…								…
RBUF+71	d64(7:0)								数据字节 64
RBUF+72	RTS(7:0)								CiA 603
…	…								…
RBUF+75	RTS(31:24)								CiA 603

RBUF 寄存器（0x00 至 0x4f）指向的是接收缓存区 RB 中最早收到的消息。所有 RBUF 寄存器可以以任意顺序读取。

在 RBUF 中的 KOER 位和寄存器 CAN_ERRINFO 中的 KOER 意义相同。如果 RBALL=1，RBUF 中的 KOER 将变得有意义。

如果使能了回环模式，且 CAN 控制器已经接收到了它自己发送的数据帧，则状态位 TX 会被置位为 1。

接收时间戳 RTS 被存储在消息的后面，因此和 TTS 相比较，RTS 存储位置和 RBUF 槽有关。

7.3.1.3 发送缓冲区

出于帧发送的需要，提供了两个发送缓冲器（TB）。主发送缓冲器（PTB）优先级较高，但只能缓冲一帧。次发送缓冲器（STB）优先级较低， 它可以在 FIFO 或优先模式下运行。PTB 和 STB 间的优先级是固定的，完全独立于 CAN 总线仲裁。总线仲裁是基于帧标识符的优先级来决定的。

可以使能 STB 发送一帧或所有存储帧。在 FIFO 模式下，首先发送此缓冲区内最旧的帧。 在优先级模式中，首先发送在该缓冲区内具有最高优先级的帧（基于帧标识符）。

无论帧标识符如何，对于CAN协议机器来说，位于PTB中的帧始终比STB中的帧具有更高的优先级。PTB 发送会停止并延迟 STB 发送。在成功发送 PTB 帧之后，STB 会自动重新发送。

如果 STB 已经启动发送且发送未完成，则 PTB 会在当前 STB 本帧发送完成后停止或者延迟 STB 发送。

在以下情况下，可以使用 PTB 发送打断 STB 发送：

1. 请求 STB 发送所有存储的帧，并且在完成 STB 所有帧发送前，主控制器请求 PTB 发送。
2. 请求 STB 发送单个帧，并且在 STB 使能发送前，主控制器请求 PTB 发送。

如果主控制器等待直到每个命令传输完成，那么它可以很容易地决定哪个缓冲区将传输下一帧。

表 7-3 发送缓冲区寄存器 TBUF – 标准格式 (rw-u)

地址	位								功能
	7	6	5	4	3	2	1	0	
TBUF	ID(7:0)								标识符
TBUF +1						ID(10:8)			标识符
TBUF +2									标识符
TBUF +3	TTSEN								标识符
TBUF+4	IDE=0	RTR	FDF	BRS	DLC(3:0)				控制位
TBUF +8	d1(7:0)								数据字节 1
TBUF +9	d2(7:0)								数据字节 1
…	…								…
TBUF +71	d64(7:0)								数据字节 64

表 7-4 发缓冲区寄存器 TBUF – 扩展格式 (rw-u)

地址	位								功能
	7	6	5	4	3	2	1	0	
TBUF	ID(7:0)								标识符
TBUF +1	ID(15:8)								标识符
TBUF +2	ID(23:15)								标识符
TBUF +3	TTSEN			ID(28:24)					标识符
TBUF+4	IDE=1	RTR	FDF	BRS	DLC(3:0)				控制位
TBUF +8	d1(7:0)								数据字节 1
TBUF +9	d2(7:0)								数据字节 1
…	…								…
TBUF +71	d64(7:0)								数据字节 64

如果 TBSEL=1，TBUF 寄存器（0x50 至 0x97）指向 STB 中的下一个空消息缓冲区，否则指向 PTB。所有 TBUF 寄存器可以以任意顺序写入。对于 STB，需要设置 TSNEXT 以标记填充的缓冲区并跳转到下一个消息缓冲区。

请注意 TBUF 的寻址范围内 TBUF+5 到 TBUF + 7 的间隔，这样做为了更好的地址段对齐。可以读取和写入间隙中的存储单元，但对 CAN 协议没有意义。

TBUF 使用真正的 32 位宽存储器构建，因此写访问需要以 32 位写入的方式执行。

RBUF 和 TBUF 都包含一些独立帧控制位 (如表 7-5 所示)。对于 RBUF，这些位表示接收到的 CAN 帧相应 CAN 控制字段位的状态，而对于 TBUF，这些位为必须发送的帧选择适当的 CAN 控制字段位。

与存储每个接收帧的 RTS 相比， TTS 仅存储最后发送的帧。 TTS 与实际选择的 TBUF 缓冲区无关。

表 7-5 RBUF 和 TBUF 的控制位

名称	说明
IDE	**标识符扩展** 0：标准帧：ID(10:0) 1：扩展帧：ID(28:0)
RTR	**远程传输请求** 0：数据帧 1：远程帧 只有 CAN 2.0 帧可以是远程帧。 CAN FD 没有远程帧。 因此如果 TBUF 和 RBUF 中的 FDF = 1，则 RRS 位(对应的 CAN2.0 帧的 RTR 位)强制为 0 当 FDF=1 时即使 RRS = 1，接收节点扔可以正确接收到 CAN FD 帧
FDF	**CAN FD 帧** 0：CAN 2.0 帧(每帧最多 8 个字节) 1：CAN FD 帧(每帧最多 64 个字节)
	位速率切换

名称	说明
BRS	0：整个帧为正常/低速比特率 1：切换到数据/快速比特率，因此如果 FDF=0，BRS 被强制转变为 0
DLC	**帧长度，单位为字节** 说明请参考表 7-6
ESI	**错误状态指示器** 这是 RBUF 的只读状态位，在 TBUF 中不可用 CAN 控制器会自动将正确的 ESI 值嵌入传输的帧中。 ESI 仅包含在 CAN FD 框架中，不存在于 CAN 2.0 框架中 0 – CAN 总线其它节点没有被动错误 1 – CAN 总线其它节点有被动错误 对于 CAN 2.0 帧，RBUF 中的 ESI 位始终保持低 传输错误状态通过寄存器 CAN_CTRL1 中的 EPASS 位显示
TTSEN	**发送时间戳启用** 在 TBUF 中，对于 CiA 603 时间戳 TTS 可选择使能更新 0 – 不获取该帧的发送时间戳(TTS 值无效) 1 – 使能 TTS 更新

RBUF 和 TBUF 中的数据长度代码（DLC）定义了有效载荷的长度 - 帧中有效载荷字节的数量。

远程帧（仅用于 FDF = 0 的 CAN 2.0 帧）总是以 0 个有效载荷字节发送，但 DLC 的内容在帧头中发送。因此，可以将一些信息编码到远程帧的 DLC 位中。但是，如果允许不同的 CAN 节点发送具有相同 ID 的远程帧，则需要注意。在这种情况下，所有发送节点都需要使用相同的 DLC，否则会导致无法解决的冲突。

表 7-6 DLC 定义 (基于 CAN 2.0 和 CAN FD 规格)

DLC(二进制)	帧类型	有效字节数
0000 到 1000	CAN 2.0 和 CAN FD	0 到 8
1001 到 1111	CAN 2.0	8
1001	CAN FD	12
1010	CAN FD	16
1011	CAN FD	20
1100	CAN FD	24
1101	CAN FD	32
1110	CAN FD	48
1111	CAN FD	64

TBUF 寄存器可读写。 因此如果有需要时主控制器可以使用 TBUF 依次逐位地准备消息。

7.3.2 总线关闭状态

CAN_CTRL0 寄存器中的状态位 BUSOFF 用来标识“总线关闭”状态。如果 CAN 节点的发送错误计数器计数超过 255，则该节点自动进入“bus off”状态。然后，在再次进入主动错误激活状态之前，它不会参与进一步的通信。如果 CAN 节点通过模块(SRST_CAN0)复位或者接收到 128 组连续 11 个隐性位（恢复序列），则 CAN 节点返回到主动错误状态。

在“bus off”状态下，RECNT 用于计数恢复序列，而 TECNT 保持不变。请注意，在进入“bus off”状态时，TECNT 会溢出，因此可能保持为一个较小的值 0-7。因此，建议在节点进入“bus off”状态之前使用 TECNT。

如果节点从“bus off”恢复，则 RECNT 和 TECNT 将自动重置为 0。

如果一个帧正在等待传输，但 CAN 节点已进入总线断开状态，则该帧保持挂起状态。如果一个节点在总线关闭后返回主动错误状态，那么将尝试传输挂起的帧。如果不需要，则主机控制器应中止帧。

7.3.3 接收过滤器

为了减少主控制器接收帧的负载，CAN 控制器使用接收过滤器。因此，每个接收过滤器的长度是 29 位。

如果消息通过其中一个过滤器，则该消息会被接受且将其存储到接收缓存区(RB)中，如果用户使能接收中断使能寄存器 RIE，则此时 RIF 会被置位。如果该消息未被接受，则 RIF 不会被置位且 RB FIFO 指针不会增加。未被接受的消息将被丢弃并被下一条消息覆盖。任何未被接受的消息都不会覆盖已存储且有效的消息。

独立于接收过滤的结果，CAN 控制器检查总线上的每条消息，并向总线发送确认或错误帧。

接受掩码定义了要进行比较的位，而接收代码定义了适当的值。将掩码位置为 0 可以将所选择的接受码位与相应的消息标识符位进行比较。设置为 1 的掩码位被禁用以进行接收检查，这会导致接受该消息。

标识符位将与相应的接受代码位 ACODE 进行比较，如下所示：

- 标准：ID(10：0) 和 ACODE(10：0)
- 扩展：ID(28：0) 和 ACODE(28：0)

例如： 如果 AMASK_x(0)=0 且所有其他 AMASK_x 位都为 1， 那么最后一个 ID 位的值必须等于接受消息的 ACODE（0）的值。所有其他 ID 位会被过滤器忽略。

注意

通过设置 AE_x=0 来禁用滤波器会阻止消息。与此相反，禁用 AMASK_x 中的掩码位会禁用对此位的检查，这会导致接受消息。

仅 AMASK 和 ACODE 的定义不区分标准帧或扩展帧。如果位 AIDEE = 1，则接受 AIDE 定义的帧类型值。否则，如果 AIDE = 0，则两种类型都会被接受。

上电复位后，配置 CAN 控制器 接收所有消息 (设置 AE_0=1 使能 Filter 0，AMASK_0 的所有位都被置为 1 同时 AIDEE 置为 0。所有其他过滤器都被禁用。 Filter 0 是唯一为 AMASK/ACODE 定义复位值的过滤器， 而所有其他过滤器都有未定义的复位值)。

7.3.4 消息接收

接收数据被存储在 RB 中，如图 7-3 所示。RB 可由预合成参数配置，并具有类似 FIFO 的行为。如果启用了 RIE，则每个收到有效、可被接受的消息都会设置 RIF=1。根据填充状态设置 RSTAT。当填充缓冲区的数量等于可编程值 AFWL 时，如果 RAFIE 使能，则设置 RAFIF。 如果所有缓冲区都已满，则当使能 RFIE 时，置位 RFIF。

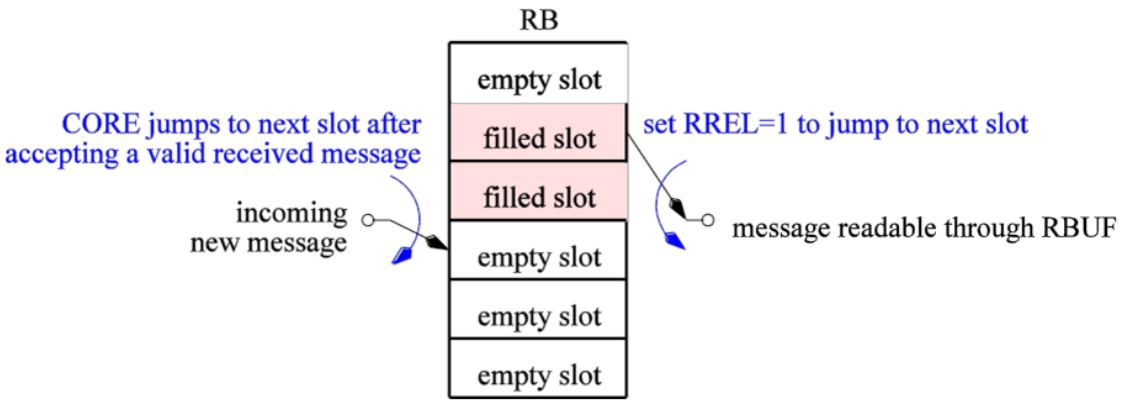

图 7-3 类似 FIFO RB 示意图 (6 slots 示例)

RB 始终将包含最旧消息的消息 slot 映射到 RBUF 寄存器。CAN 2.0 消息的最大有效载荷长度为 8。每条消息的长度由 DLC 定义。因此，RB 为每条消息提供 slot，并且主控制器需要设置 RREL 以跳转到下一个 RB slot。 可以按任意顺序读取实际 slot 的所有 RBUF 字节。

如果 RB 已满，则下一个传入的消息将被临时存储，直到它有效地传递（第 6 个 EOF 位）。如果 ROM 为 0，最旧的消息将被最新的消息所覆盖，或者如 ROM =1，则最新消息将被丢弃。在这两种情况下，如果使能 ROIE，则 ROIF 会被置位。如果主控制器读取最旧的消息并在新传入的消息变为有效之前设置 RREL，则不会丢失任何消息。

7.3.5 处理消息接收

如果没有验收滤波器，CAN 控制器会发出每帧的接收信号，并且主机会被要求决定它是否进行寻址，这会给主控制器带来很大的负担。

可以禁用中断并使用接受滤波器来减少主控制器的负载。一个基本操作为：如果 RIE 已启用且 CAN 控制器已收到有效消息，则 RIF 设置为 1。为了减少接收中断的数量，可以使用 RAFIE/RAFIF（RB 几乎满中断）或RFIE/RFIF（RB满中断）而不是RIE/RIF（接收中断）。 “几乎满限制”可使用AFWL进行编程。

RB 包含多个 RB slot，可在使用通用参数进行合成之前进行选择。读取 RB 应按如下方式进行：

1. 使用 RBUF 寄存器从 RB FIFO 中读取最旧的消息；

2. 用 RREL=1 释放 RB slot。这将选择下一条消息 （下一个 FIFO slot）， RBUF 会被自动更新；
3. 重复以上动作直到 RSTAT 发出空 RB 信号。

如果 RB FIFO 已满并且新接收的消息被识别为有效（第 6 个 EOF 位），则将丢失一条消息（参见 ROM 位）。在此之前，不会丢失任何消息。应该给出足够的时间让主控制器在 RB FIFO 已满并且发生所选中断之后从 RB 读取至少一帧。为了实现这种行为，RB 包括比合成参数 RBUF_SLOTS 指定的 slot 多一个（隐藏）slot。此隐藏 slot 用于接收消息，验证该消息并在溢出发生之前检查它是否与验证过滤器相匹配。

7.3.6 消息发送

开始发送之前，必须向至少一个发送缓冲区（PTB 或 STB）加载消息。如果 PTB 被锁且 TPSTAT 发信号通知 STB 的填充状态，则 TPE 会发出信号。TBUF 寄存器提供对 PTB 和 STB 的访问。

推荐的编程流程如下：

1. 将 TBSEL 设置为需要的值以选择 PTB 或 STB；
2. 将帧写入 TBUF 寄存器；
3. 对于 STB， 设置 TSNEXT=1 以完成该 STB slot 的加载。

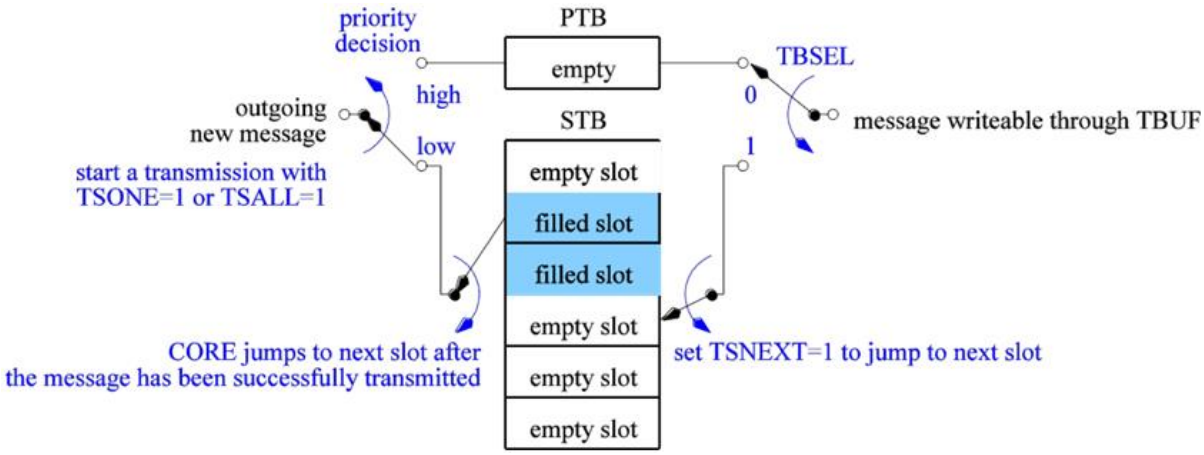

图 7-4 FIFO 模式下 PTB 及 STB 示意图 （空 PTB， 6 STB slots)

CAN 2.0 消息的最大有效载荷长度为 8 个字节，CAN FD 的消息最大为 64 个字节。每条消息的长度由 DLC 定义。对于远程帧（比特 RTR），DLC 变得毫无意义，因为远程帧的数据长度始终为 0 字节。主控制器需要设置 TSNEXT 以跳转到下一个 STB slot。所有 TBUF 字节都可以按任意顺序写入。

如果 TBSEL = 0 时选择 PTB，则设置 TSNEXT = 1 是毫无意义的。在这种情况下，TSNEXT 会被自动清除并且不会有任何影响。

应将 TPE 位设置为在使用 PTB 时启动发送。要使用 STB，必须将 TSONE 设置为开始发送单个消息或 TSALL 发送所有消息。

PTB 的优先级始终比 STB 高。如果 PTB 和 STB 同时使能发送，则无论帧标识符如何，都将始终先发送 PTB 消息。如果来自 STB 的发送已经激活，则在来自 PTB 的消息在下一个可能的发送位置（下一个帧 slot）发送之前完成。在 PTB 发送完成或中止之后，CAN 控制器返回以处理来自 STB 其他暂停的消息。

当发送完成后，会设置如下发送中断：

- 对于 PTB， 如果启用了 TPIE，则将设置 TPIF；
- 对于使用 TSONE 的 STB，如果一条消息已完成且已使能 TSIE，则会置位 TSIF；
- 对于使用 TSALL 的 STB，如果所有消息已完成发送且已使能 TSIE，则会置位 TSIF。换言之，如果 STB 为空，则 TSIE 会被置位。

当 STB 为空时，设置 TSONE 或 TSALL 是毫无意义的。在这种情形下，TSONE 和 TSALL 都将会自动复位。不会设置中断标志，也不会发送帧。

7.3.7 消息发送中止

由于其低优先级而无法发送缓冲区中的消息，这将长时间阻塞缓冲区。为了避免这种情况，如果尚未开始发送，则主控制器可以通过分别设置 TPA 或 TSA 来撤销发送请求。TPA 和 TSA 都提供单个中断标志：AIF。CAN 协议机器只有在不向 CAN 总线发送任何内容时才执行中止。因此，以下行为是有效的：

- 总线仲裁期间没有中止
  - 如果节点失去仲裁，则在此之后将执行中止
  - 如果节点赢得仲裁，则发送该帧
- 发送帧时没有中止
  - 如果成功发送帧，则通知主控制器发送成功。在这种情况下，不会发出中止信号。这是通过适当的中断和状态位完成的
  - 在 CAN 节点未接收到确认的发送失败之后，错误计数器递增并且将执行中止
  - 当主机使能发送所有的帧时（TSALL = 1），如果 STB 中剩余至少一帧，则完成发送的帧以及中止的帧都会通知主机

基于以上事实，中止此发送可能需要一些时间，具体取决于 CAN 通信速度和帧长度。如果执行中止，则会导致以下操作：

- TPA 释放 PTB，导致 TPE=0

  在释放 PTB 之后，帧数据仍存储在 PTB 中。
- TSA 释放 STB 的单个或所有消息 slot，这取决于是使用 TSONE 还是 TSALL 来启动发送。TSSTAT 会相应地进行更新。释放 STB 中的帧会导致丢弃帧数据，因为主机无法访问它。

不建议同时设置 TPA 和 TSA。如果主控制器仍然决定这样做，则将设置 AIF，并且如果可能的话，将中止来自 PTB 和 STB 的所有发送。如前所述，如果在可以执行中止之前完成一次发送，则这将导致信号发送成功。因此，如果使能，可以设置以下中断标志：

- AIF (设置一次，用于 PTB 和 STB 发送中止)
- TPIF + AIF
- TSIF + AIF
- TPIF + TSIF (很少，只有在主机没有立即处理 TPIF 时才会发生)
- TPIF + TSIF + AIF (很少，只有在主机没有立即处理 TPIF 和 TSIF 时才会发生)

要清除整个 STB，需要设置 TSALL 和 TSA。为了检测由于失去仲裁而无法长时间发送消息，主机可以使用 ALIF/ALIE。

7.3.8 STB 满

在向 STB 写入一条消息之后，TSNEXT=1 标记填充的缓冲 slot 并跳转到下一个空闲消息 slot。此操作后，CAN 控制器会自动将 TSNEXT 复位为 0。如果最后一个消息 slot 已被填满，因此所有消息 slot 都被占用，则 TSNEXT 保持置位状态，直到新的消息 slot 空闲为止。当 TSNEXT=1 时，CAN 控制器会阻止写至 TBUF。

当 slot 空闲时，CAN 控制器会自动将 TSNEXT 重置为 0。如果来自 STB 的帧成功发送或者主机请求中止(TSA=1)，则 slot 会变为空闲。如果 TSALL 发送中止，则 TSNEXT 也会复位，但整个 STB 会被标记为空。

7.3.9 扩展状态及错误报告

在 CAN 总线通信时，会发生发送错误。如下特性支持检测和分析发送错误。

7.3.9.1 可编程错误警告限制

接收/发送期间的错误由 RECNT 和 TECNT 来计数。LIMIT 寄存器中的可编程错误警告限制 EWL 可由主控制器灵活配置以用于响应接收/发送错误事件。可以从 8 到 128 中以 8 个错误步进选择极限值：

错误计数限制 = (EWL + 1) * 8.

如果在以下条件下由 EIE 使能，则将设置中断 EIF：

- 错误警告限制的边界已通过 RECENT 或 RECENT 在任一方向交叉
- BUSOFF 位已经在某一方向上变化

7.3.9.2 仲裁失利捕获

控制器能够检测仲裁段中仲裁失利的确切位位置。如果 ALIF 中断被启用，则可以通过 ALIF 中断发出此事件。如果此节点能够赢得此仲裁，则 ALC 的值会保持不变。ALC 保持最后一次仲裁失利的值。

ALC 的值定义如下：一帧以 SOF 位开始，发送 ID 的第 1 位。第一个 ID 位的 ALC 值为 0，第二个 ID 位的 ALC 值为 1，依此类推。仲裁仅允许在仲裁域中进行。因此，ALC 的最大值为 31，是扩展帧的 RTR 位。

如果标准远程帧与扩展帧进行仲裁，则扩展帧会在 IDE 位失去仲裁，ALC 将为 12。发送标准远程帧的节点将不会注意到已经发生了的仲裁，因为该节点已经获胜。在仲裁域之外不可能获得仲裁失利，这样的事件为位错误。

7.3.9.3 错误类型

CAN 控制器识别 CAN 总线上的错误，并将最后一个错误事件存储在 KOER 位中。如果使能 BEIF 中断，则可以通过 BEIF 中断发出 CAN 总线错误信号。每个新的错误事件都会覆盖之前存储的 KOER 值。因此，主控制器必须对错误事件做出快速反应。

7.3.9.4 接收所有的数据帧(RBALL)

如果 RBALL = 1，则所有接收到的数据帧都将被存储，即使有错误的数据帧也会被存储。这也适用于回环模式。仅数据帧存储在 RBUF 中，错误帧或过载帧不会被存储。
如果启用了 CiA 603 时间戳记（TIMEEN = 1）并且为 EOF 配置了时间戳记（TIMEPOS = 1），则在出现错误的情况下，将在错误帧的开始处获取时间戳。
仅当节点是帧的发送方时，大多数可能的错误才会发生。在这种情况下框架如果激活了回环模式，则仅将其存储在 RBUF 中。 根据错误部分的类型，当其他部分未知时，存储在 RBUF 插槽中的帧可能有效。

7.3.10 扩展功能

7.3.10.1 单次发送

有时，不需要自动重传。因此，可以通过 TPSS 位对发送缓冲器 PTB、通过 TSSS 位对发送缓冲器 STB 分别设置发送一次消息的命令。在这种情况下，如果所选发送有效，则在发生错误或仲裁失利的情况下不会执行重传。

在立即发送成功的情况下，与正常发送没有区别。但是在发送失败的情况下，会出现如下问题：

- 如果使能，则 TPIF 会被置位，相应的发送缓冲区 slot 会被清除
- 如果出现错误， 会更新 KOER 和错误计数器。如果使能 BEIE，BEIF 会被置位，其他错误中断标志将相应地起作用
- 如果仲裁失利，当使能 ALIE 时，会置位 ALIF

因此，对于单次发送，TPIF 自身并不指示帧是否已成功发送。单次发送应仅与 BEIF 和 ALIF 一起使用。

如果单次发送使用 TSALL，并且 STB 中存在多个帧，则对于每个帧，进行单次发送。不管是否未成功发送任何帧（例如：由于应答错误），CAN 控制器前行至下一帧，并且如果 STB 为空则停止。

因此，在这种场景下，只有错误计数器指示发生了什么。这将难以评估，因为如果两个帧中的一个出现错误，则主机无法检测哪个是成功的帧。

总线被另一个帧占用情况下， 如果单次发送开始，则 CAN 控制器会一直等待直到总线空闲，然后尝试发送单次帧。

7.3.10.2 监听模式

LOM 提供监控 CAN 总线的能力，而不会对总线产生任何影响。

另一个应用是自动比特率检测，其中主控制器尝试不同的定时设置，直到没有错误发生。

在 LOM 中监视错误 (KOER, BEIF) 。

在 LOM 中，CAN 控制器无法将显性位写入总线 (没有有效的错误标志或过载标志，没有确认)。这是使用如下规则完成的

- 在 LOM 中，协议控制器就好像处于错误被动模式，其中仅生成隐性错误标志。只有协议机器就像处于错误被动模式一样。不触及包括状态寄存器在内的所有其他组件。
- 在 LOM 中，协议控制器不生成显性应答。
- 不管什么错误，错误计数器保持不变。

关于 LOM 的应答：

- 如果帧由一个节点发送，则仅当至少一个不在 LOM 状态的附加节点连接到总线时，才会生成在总线上可见的应答。这样就不会出现错误，所有节点（也包括 LOM 中的节点）都会收到该帧。
- 如果在应答错误之后存在主动或被动错误标志，则 LOM 中的节点能够将其检测为应答错误。

在发送激活时，不应激活 LOM。主控制器必须注意这个问题，没有来自 CAN 控制器额外的保护措施。如果使能 LOM ，则无法开始发送。

7.3.10.3 总线连接测试

要检测节点是否连接至总线，可以采用如下测试步骤：

- 发送一帧。如果节点连接到总线，则其 TX 位在其 RX 输入上是可见的。
- 如果有其他节点连接到 CAN 总线，则预期会发送成功(包括来自其他节点的确认)，不会发出任何错误信号；

- 如果该节点是唯一一个连接到 CAN 总线的节点 (但总线、收发器和 CAN 控制器 之间的连接正常)。由于没有来自其他节点的确认信息，第一个常规错误发生在应答段中。如果使能且 KOER="100"（应答错误），则会产生 BEIF 错误中断。
- 如果与收发器或总线间的连接断开，则在 SOF 位之后立即设置 BEIF 错误中断并且 KOER ="001"（BIT 错误）。

7.3.10.4 环回模式 (LBMI 及 LBME)

CAN 控制器支持两种环回模式：内部(LBMI)和外部(LBME)。两种模式都导致接收自己发送的帧，这对于自测是很有用的。

在 LBMI 中，CAN 控制器与 CAN 总线断开连接，txd 输出设置为隐性。输出数据流在内部反馈到输入。在 LBMI 模式下，节点生成自应答信息以避免应答错误。

在 LBME 模式下，CAN 控制器保持与收发器的连接，并且在总线上可以看到发送的帧。在收发器的帮助下，CAN 控制器接收自己发送的帧。在 LBME 模式下，当 SACK=0 时节点不产生自应答信息。如果 SACK=1 时节点产生应答信息。因此，在 LBME 模式下，帧发送有两种可能的结果：

1. 另外一个节点也接收该帧，并产生确认信息。这会导致一次成功的发送和接收。
2. 没有其他节点连接到总线，这会导致应答错误。为了避免重传并增加错误计数，如果不知道其他节点是否连接到总线，建议使用 TPSS 或 TSSS。

在环回模式下，CAN 控制器接收其自身的消息，将其存储在 RBUF 中，并在使能时设置合适的发送和接收中断标志。

LBMI 可用于芯片内部和软件测试，而 LBME 可以测试收发器及其连接。

当发送处于活动状态时，不应激活两种环回模式，主控制器需要注意此问题。没有来自 CAN 控制器额外的保护措施。

如果节点连接到 CAN 总线，则不能通过简单地将 LBMI 设置为 0 来完成从 LBMI 切换回正常操作，因为这可能只是在另一个 CAN 节点正在发送时发生的情况。在这种情况下，应将 RESET 位设置为 1 来切换回正常操作，这会自动将 LBMI 清零。最后，可以禁用 RESET 并且 CAN 控制器返回至正常操作。与此相反，LBME 每次都可以被禁用。

7.3.10.5 收发器待机模式

使用寄存器位 STBY，可以驱动输出待机信号。它可用于激活收发器的待机模式。

一旦待机模式被激活，无法进行发送，因此无法设置 TPE，TSONE 和 TSALL。另一方面，如果发送已经激活（设置了 TPE，TSONE 或 TSALL），CAN 控制器不允许设置 STBY。

如果已经置位 STBY， 收发器进入低功耗模式。在此模式下，无法以全速接收帧，但会监视 CAN 总线的显性状态。如果显性状态在收发器数据手册中定义的时间内有效，则收发器会将 rxd 信号拉低。如果 rxd 为低电平，CAN 控制器会自动将 STBY 清零，从而禁用收发器的待机模式。这是在没有中断到主控制器的情况下完成的。

从待机模式切换到活动模式对收发器需要一些时间，因此无法成功接收初始唤醒帧。因此，最近处于待机状态的节点不会发送应答信号。如果总线上的 CAN 节点没有应答唤醒帧，这会导致唤醒帧的发送器应答错误。然后发射器将自动重复该帧。在此重复期间，收发器将返回活动模式，CAN 控制器将接收此帧并将做应答。

总之，一个节点发送用于唤醒的帧。如果所有其他节点处于待机模式，发送器会收到应答错误并自动重复该帧。在重复的过程中，节点返回活动模式并将做应答。

STBY 不受 RESET 位的影响。

7.3.10.6 错误计数器复位

根据 CAN 标准，RECNT 对接收错误进行计数，TECNT 对发送错误进行计数。若发送错误太多，CAN 节点必须进入总线关闭状态，这会激活 RESET 位。取消激活该位后，RESET 不会修改错误计数器或总线关闭状态。CAN 规范定义了如何禁用总线关闭状态和减少错误计数的规则。如果只有一个临时错误导致该问题，一个好的节点将自动从所有这些问题中恢复。经典的 CAN 2.0B 规范要求这种自动行为，无需主控制器交互。

将 BUSOFF 位置 1 会复位错误计数器，从而强制节点退出总线关闭状态。这是在不设置 EIF 的情况下完成的。

7.3.10.7 低通滤波器

当 MCU 进入停止模式时，可使能 CAN 唤醒，如需滤除毛刺信号（低于 2μs），可设置寄存器 EN_CAN0_FILTER 等于 1。

7.3.11 软件复位

如果 CAN_CTRL0 寄存器中的 RESET 位被置为 1，则软件复位处于激活状态。如果 RESET = 1，则几个组件被强制置为复位状态，但 RESET 不会触及所有组件。一些组件仅对硬件复位敏感。软件和硬件复位的所有位的复位值始终相同。

表 7-7 软件复位

寄存器	复位 (RESET)	说明
ACFADR	否	-
ACODE_x	否	如果RESET=1则寄存器可写，为0则寄存器写锁定
AE_x	否	-
AFWL	否	-
AIF	是	-
ALC	是	-
ALIE	否	-

寄存器	复位 (RESET)	说明
ALIF	是	-
AMASK_x	否	如果RESET=1则寄存器可写，为0则寄存器写锁定
BEIE	否	-
BEIF	是	-
BUSOFF	否	通过设置BUSOFF=1可以复位BUSOFF及错误计数器
EIE	否	-
EIF	否	-
EPASS	否	-
EPIE	否	-
EPIF	是	-
EWARN	否	-
EWL	是	-
FD_ISO	否	如果RESET=1则寄存器可写，为0则寄存器写锁定
F_PRESC	否	如果RESET=1则寄存器可写，为0则寄存器写锁定
F_SEG_1	否	如果RESET=1则寄存器可写，为0则寄存器写锁定
F_SEG_2	否	如果RESET=1则寄存器可写，为0则寄存器写锁定
F_SJW	否	如果RESET=1则寄存器可写，为0则寄存器写锁定
KOER	是	-
LBME	是	-
LBMI	是	-
LOM	否	-
RACTIVE	是	即使接收处于活动状态，接收也会立即取消，不会生成 ACK 信息
RAFIE	否	-
RAFIF	是	-
RBALL	是	-
RBUF	(是)	所有 RB slot 被标记为空， RBUF 包含未知数据
RECNT	否	-
RFIE	否	-
RFIF	是	-
RIE	否	-
RIF	是	-
ROIE	否	-
ROIF	是	-
ROM	否	-
ROV	是	所有 RB slot 被标记为空
RREL	是	-
RSTAT	是	所有RB slot被标记为空

寄存器	复位 (RESET)	说明
SACK	是	-
SELMASK	否	-
STBY	否	-
S_PRESC	否	如果RESET=1则寄存器可写，为0则寄存器写锁定
S_Seg_1	否	如果RESET=1则寄存器可写，为0则寄存器写锁定
S_Seg_2	否	如果RESET=1则寄存器可写，为0则寄存器写锁定
S_SJW	否	如果RESET=1则寄存器可写，为0则寄存器写锁定
TACTIVE	是	在 RESET 时，所有发送都立即停止。如果发送有效，则会产生不完整的帧。其他节点会产生错误帧
TBSEL	是	TBUF 固定指向 PTB
TBUF	(是)	所有 STB slot 被标记为空，因为 TBSEL TBUF 指向 PTB
TECNT	否	-
TIMEEN	否	-
TIMEPOS	否	-
TPA	是	-
TPE	是	-
TSA	是	-
TSALL	是	-
TSMODE	否	-
TSNEXT	是	-
TSONE	是	-
TPIE	否	-
TPIF	是	-
TPSS	是	-
TSFF	是	所有STB slot 被标记为空
TSIE	否	-
TSIF	是	-
TSSS	是	-
TSSTAT	是	所有STB slot被标记为空
TTS	否	-

7.3.12 CAN 位时间

CAN 2.0B 定义了高达 1Mbit/s 的数据比特率。对于 CAN FD 没有固定的限制，对于实际的系统，数据速率受所使用的收发器和 CAN 控制器可实现的时钟频率的限制，这取决于所使用的目标单元库。

CAN 控制器可以编程为任意选择的数据速率，仅受相应位定时和预分频器寄存器中位设置范围的限制。

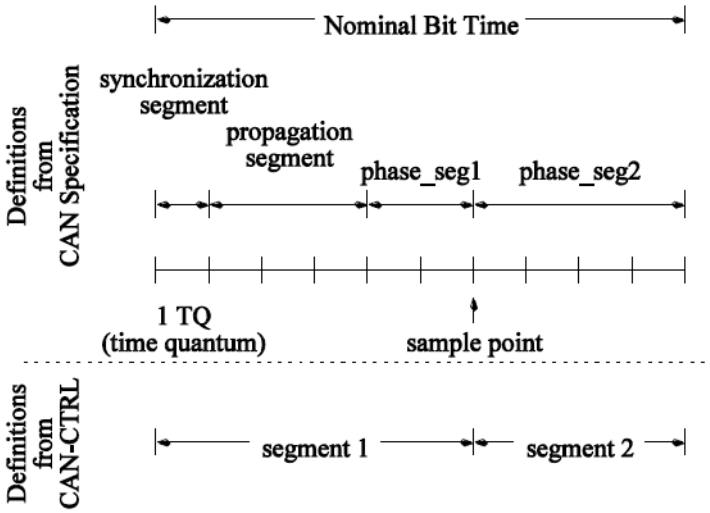

图 7-5 CAN 位定时

CAN 位定时 BT 由若干段（segment）组成，如图 7-5 所示。每个段由许多时间限额单位 n_{TQ} 组成。

时间限额的时长 TQ 为：

$$TQ = n_{prescaler} / f_{CLOCK}$$

$$BT = n_{prescaler} * n_{TQ} / f_{CLOCK} = t_{Seg\_1} + t_{Segt\_2}$$

CAN 规范要求段长度之间存在若干关系（如表 7-8 所示），这会造成 t_{Seg_1}，t_{Segt_2} 和最大同步跳转宽度 t_{SJW} 之间 的关系。请注意：表 7-8 列举了 CAN 规格定义的最小配置范围。

表 7-8 CAN 定时段

段	说明
SYNC_SEG	同步段 = 1 TQ
PROP_SEG	**传播段（Propagation Segment）** [1...8] TQ　CAN 2.0 比特率　CAN FD 未启用 [1...48] TQ　CAN 2.0 比特率　CAN FD 已启用 [1...48] TQ　CAN FD 标称比特率 [0...8] TQ　CAN FD 数据比特率
PHASE_SEG1	**相位缓冲段 1** [1...8] TQ　CAN 2.0 比特率　CAN FD 未启用 [1...16] TQ　CAN 2.0 比特率　CAN FD 已启用 [1...16] TQ　CAN FD 标称比特率 [1...8] TQ　CAN FD 数据比特率

段	说明
PHASE_SEG2	**相位缓冲段 2** [2...8] TQ CAN 2.0 比特率 CAN FD 未启用 [2...16] TQ CAN 2.0 比特率 CAN FD 已启用 [2...16] TQ CAN FD 标称比特率 [2...8] TQ CAN FD 数据比特率
SJW	**同步跳转宽度** [1...4] TQ CAN 2.0 比特率 CAN FD 未启用 [1...16] TQ CAN 2.0 比特率 CAN FD 已启用 [1...16] TQ CAN FD 标称比特率 [1...8] TQ CAN FD 数据比特率
IPT	信息处理时间 = [0…2] TQ PHASE_SEG2 ≥ IPT

如表 7-8 所示，CAN 控制器将 SYNC_SEG，PROP_SEG 和 PHASE_SEG1 等集为一组，该组的长度可使用 t_{Seg_1} 进行配置。表 7-9 列出了可用的配置范围。请注意，CAN 控制器不会检查是否满足所有规则，并提供比 CAN/CAN FD 规范定义的更宽的配置范围。

表 7-9 CAN 控制器 定时配置

配置项	需求			
t_{seg_1}	[2…65]		TQCAN 2.0 比特率	(慢)
	[2…65]		CAN FD 标称比特率	(慢)
	[2…17]		CAN FD 数据比特率	(快)
t_{seg_2}	[1…8] TQ	$t_{Seg\_1} \geq t_{Seg\_2} + 2$	CAN 2.0 比特率	(慢)
	[1…32] TQ	$t_{Seg\_1} \geq t_{Seg\_2} + 2$	CAN FD 标称比特率	(慢)
	[1…8] TQ	$t_{Seg\_1} \geq t_{Seg\_2} + 1$	CAN FD 数据比特率	(快)
t_{SJW}	[1…16] TQ	$t_{Seg\_2} \geq t_{SJW}$	CAN 2.0 比特率	(慢)
	[1…16] TQ	$t_{Seg\_2} \geq t_{SJW}$	CAN FD 标称比特率	(慢)
	[1…8] TQ	$t_{Seg\_2} \geq t_{SJW}$	CAN FD 数据比特率	(快)

对于 CAN 2.0 比特率标称的比特率以及 CAN FD(慢)标称比特率，配置寄存器 S_Seg_1，S_Seg_2，S_SJW 和 S_PRESC 来定义适当的段长度。寄存器 S_Seg_1， S_Seg_2，S_SJW 和 S_PRESC 对于 CAN FD(快)数据比特率是无效的。

$t_{Seg\_1} = (S\_Seg\_1 + 2) * TQ$ $t_{Seg\_1} = (F\_Seg\_1 + 2) * TQ$

$t_{Seg\_2} = (S\_Seg\_2 + 1) * TQ$ $t_{Seg\_2} = (F\_Seg\_2 + 1) * TQ$

$t_{SJW} = (S\_SJW + 1) * TQ$ $t_{SJW} = (F\_SJW + 1) * TQ$

$n_{prescaler} = S\_PRESC + 1$ $n_{prescaler} = F\_PRESC + 1$

CAN 控制器通过设置 BRS=1 就可以从低速标称比特率切换到快速数据比特率，并在 CRC 分隔符位的采样点切回。

对于 CAN-FD 节点，由于收发器的延迟导致 CAN-FD 的通信可能会延迟超过一个 bit 时间，因此，无法使用原始 SP（Sample point）对正确的位值进行采样，CAN FD 规范定义了一个额外的二次采样点(SSP)，此时可以选择启用发射机延迟补偿（TDC）即 TDCEN=1 且配置 SSPOFF 寄存器，建议 SSPOFF 配置等于 t_{Seg_1}，用户需要根据实际情况配置该寄存器。 如果不启用 TDC 发送节点可能就不能正确采样到它所发出的数据，此时发送节点就会检测到一个位错误。

初始化 CAN 控制器 的步骤如下：

1. 设置位 RESET=1；
2. 设置寄存器 S_Seg_1 and S_Seg_2；

   在该示例中，总线数据速率为 1M 波特率，系统时钟为 48MHz。

   所选 n_{TQ} 和 $n_{prescaler}$ 的值需适配 BT。

   在该示例中，所选 $n_{prescaler} = 2$ ， $n_{TQ} = 24$ ，这样可以实现完全匹配：$BT = 24TQ$。

   在时间段定义中，可以选择 $t_{Seg\_1} = 18TQ$; $t_{Seg\_2} = 6TQ$ ，对应寄存器： S_Seg_1=16 ，S_Seg_2 = 5；
3. 加载验证码和掩码寄存器 (可选)；
4. 设置 S_SJW 寄存器；

   当满足 $t_{Seg\_2} \geq t_{SJW}$ ，可以自由选择 $t_{SJW} = 4$，对应寄存器 S_SJW=3。
5. 加载时钟预分频寄存器 S_PRESC: $n_{prescaler}$ = PRESC + 1 ，对应 S_PRESC=1；
6. 设置位 RESET=0；

   如上给出的顺序不是强制的。只需要在开始时设置 RESET=1，否则无法加载位定时，ACODE 和 AMASK 寄存器。配置完成后，需要将 RESET=0 。然后控制器等待 8 个隐形位(帧结束)，然后恢复其正常操作。
7. 继续配置中断其他配置位，并执行指令。

下面是一些适用于 CAN 网络所有节点位定时设置的示例表。

表 7-10 48MHz can_clk 的示例设置

比特率[Mbit/s]	SP[%]	预分频器	位时间[TQ]	Seg1 [TQ]	Seg2 [TQ]	SJW [TQ]
1	75	1	48	36	12	12
0.8	80	2	30	24	6	6
0.5	75	2	48	36	12	12

比特率[Mbit/s]	SP[%]	预分频器	位时间[TQ]	Seg1 [TQ]	Seg2 [TQ]	SJW [TQ]
0.25	75	4	48	36	12	12
0.125	75	8	48	36	12	12
0.1	75	10	48	36	12	12

表 7-11 8MHz can_clk 的示例设置

比特率[Mbit/s]	SP[%]	预分频器	位时间[TQ]	Seg1 [TQ]	Seg2 [TQ]	SJW [TQ]
1	75	1	8	6	2	2
0.8	80	1	10	8	2	2
0.5	75	1	16	12	4	4
0.25	75	2	16	12	4	4
0.125	75	4	16	12	4	4
0.1	75	4	20	15	5	4

表 7-12 48MHz can fd clk 的示例设置

比特率[Mbit/s]	SP[%]	预分频器	位时间[TQ]	Seg1[TQ]	Seg2[TQ]	SJW[TQ]	TDC[clk]
8	83	1	6	5	1	1	5
6	75	1	8	6	2	2	6
4	75	1	12	9	3	3	9
2	75	1	24	18	6	6	18
1	75	2	24	18	6	6	36
0.1	75	20	24	18	6	6	-
0.05	75	40	24	18	6	6	-

7.3.13 时间戳

CAN 控制器中的时间戳包括发送帧时间戳和接收帧时间戳。当接收数据或者发送数据时，CAN 控制器复制计时器值帧时间戳存储在 RTS 和 TTS 中，用户可以读取寄存器 RTS 或者 TTS 获取时间戳值。

CAN 控制器可以在有效帧的 SOF 或 EOF 位的采样点获取时间戳。可以通过配置位 TIMEPOS 进行选择采样点位置。 ACK 界定符之后的七个隐性位形成 CAN / CAN FD 帧的 EOF。在许多系统中广泛使用的基于软件的时间戳依赖于接收和传输中断。因此，建议时间戳采样点配置在 EOF。

CAN 控制器仅支持一个传输帧（TTS）的时间戳，但是对接收帧（RTS）有单独的时间戳。生成传输帧的时间戳可以使用 TBUF 插槽内的 TTSEN 位分别为每个帧启用或禁用。

CAN 控制器内部包含计时器，时间戳机制，存储 TTS 的寄存器以及每帧存储 RTS 的内存。寄存器 TIMEEN 位启用或禁用时间戳。如果禁用，则 TTS 和 RTS 无效。

下面举例说明 CAN 控制器发送时间戳功能使用步骤：

1. 初始化 CAN 控制器，使能发送中断，配置其时钟源为 48MHz；
2. 设置 CAN 控制器 timer clock 时钟分频 48，则 TTS 中每个计数值就是 1μs；
3. 通过配置 TIMEPOS 寄存器设置采样点位置为 EOF；
4. 使能 TIMEEN 寄存器和 TTSEN 寄存器；
5. 在 CAN 控制器发送完成后读取 TTS 值；

下面举例说明 CAN 控制器接收时间戳功能使用步骤：

1. 初始化 CAN 控制器，使能接收中断，配置其时钟源为 48MHz；
2. 设置 CAN 控制器 timer clock 时钟分频 48，则 RTS 每个计数值就是 1μs；
3. 通过配置 TIMEPOS 寄存器设置采样点位置为 EOF；
4. 使能 TIMEEN 寄存器；
5. 在 CAN 控制器接收完成后读取 RTS 值。

7.4 寄存器定义

CAN 控制器是一个 32 位组件，如表 7-13 所示。

表 7-13 CAN-CRTL 寄存器映射

CAN0 基地址：0x40007800

地址	名称	宽度	描述
CAN 基地址 ~ CAN 基地址+0x4F	CAN_RBUF	32*20	接收缓冲区寄存器(Receive Buffer Register)和接收时间戳
CAN 基地址+0x50 ~ CAN 基地址+0x97	CAN_TBUF	32*18	发送缓冲区寄存器(Transmit Buffer Register)
CAN 基地址+0x98 ~ CAN 基地址+0x9F	CAN_TTSx	32*2	发送数据帧时间戳(x=0-1)，TTS1 保留
CAN 基地址+0xA0	CAN_CTRL0	32	控制寄存器 0
CAN 基地址+0xA4	CAN_CTRL1	32	控制寄存器 1
CAN 基地址+0xA8	CAN_SBITRATE	32	普通 CAN 波特率配置寄存器
CAN 基地址+0xAC	CAN_FBITRATE	32	高速 CAN 波特率配置寄存器
CAN 基地址+0xB0	CAN_ERRINFO	32	CAN 错误类型以及错误计数器寄存器
CAN 基地址+0xB4	CAN_ACFCTRL	32	过滤器控制寄存器
CAN 基地址+0xB8	CAN_ACF	32	过滤器使能寄存器
CAN 基地址+0xBC	CAN_VERSIO	32	CAN 控制器版本寄存器

7.4.1 发送时间戳寄存器(CAN_TTSx)

表 7-14 CAN_TTSx 寄存器

CAN_TTSx 发送时间戳寄存器 **RESET：0x0**

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

字段	说明
31: 0 TTS	**传输时间戳** TTS 保留最后发送的帧的时间戳，用于 CiA 603 时间戳 如果 TTSEN = 1，每个新的帧将覆盖 TTS。时间戳可以为 32 位位宽， 其中 TTS1 为为保留位

7.4.2 控制寄存器 0(CAN_CTRL0)

表 7-15 CAN_CTRL0 寄存器

CAN_CTRL0 **CAN 控制寄存器 0** **RESET：0x00900080**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	SACK	ROM	ROV	RREL	RBALL		RSTAT		FDISO	TSNEXT	TSMODE				TSSTAT	
访问	RW	RW	R	RW	RW		R		RW	RW	RW				R	
Reset	0	0	0	0	0		0		1	0	0				0	
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	TBSEL	LOM	STBY	TPE	TPA	TSONE	TSALL	TSA	RESET	LBME	LBMI	TPSS	TSSS	RACTIVE	TACTIVE	BUSOFF
访问	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	R	RW
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

字段	说明
31 SACK	**自应答** 0：没有自应答 1：当 LBME=1 有自应答
30 ROM	**接收缓冲区溢出模式** 如果在收到新消息时出现完整的 RBUF，则 ROM 选择以下内容： 1 ：新消息将不会被存储 0 ：最旧的消息将被覆盖
29 ROV	**接收缓冲区溢出** 1 ： 溢出，至少一个消息丢失 0 ： 不溢出 ROV 由设置 RREL=1 清除
28 RREL	**接收缓冲区释放** 1 ： 释放，主机已经读取 RB 0 ： 不释放 主控制器已读取实际的 RB 缓冲区并将其释放。然后，CAN 控制器指向下一个 RB 缓冲区，RSTAT 得以更新。
27 RBALL	**RB 缓冲区存储所有的数据帧** 0：正常操作 1：RB 缓冲区存储所有接收的数据帧，即使数据帧有误也会被存储，该位设置也适用于回环模式。仅仅存储数据帧而不存储错误帧和过载帧
25：24 RSTAT	**接收缓冲区状态** 00 ： 空 01 ： 大于空，小于几乎满 (AFWL) 10 ： 几乎满 (AFWL 可编程阀值)，但不满不溢出 11 ： 满 (在溢出的情况下保持设置 - 对于溢出信号，请参见 ROV)
23 FDISO	**CAN FD ISO 模式** 0：Bosch CAN FD(non-ISO) 模式 1：ISO CAN FD 模式(ISO 11898-1:2015) ISO CAN FD 模式有一个不同的 CRC 初始值以及一个额外的填位 在同一网络中不能同时混合以上两种模式 该位对 CAN2.0B 没有影响 如果 RESET=1，该位仅可写

字段	说明
22 TSNEXT	**选择下一个次缓冲区** 0： 没有动作 1：填充 STB 缓冲区，选择下一个缓冲区 在将所有帧字节写入 TBUF 寄存器后，主控制器必须设置 TSNEXT 表示此缓冲区已填充。然后 CAN 控制器将 TBUF 寄存器连接到下一个缓冲区。 一旦缓冲区标记为已填充，就可以使用 TSONE 或 TSALL 启动发送。 可以在一次写访问中，将 TSNEXT，TSONE 或 TSALL 一起设置。 TSNEXT 必须由主控制器设置，并在设置后立即由 CAN 控制器自动复位。 如果 TBSEL = 0，则设置 TSNEXT 是没有意义的。在这种情况下，TSNEXT 将被忽略并自动清除且没有任何影响。如果 STB 的所有缓冲区都已填满，则 TSNEXT 将保持设置状态，直到缓冲区空闲为止。
21 TSMODE	**次发送缓冲区模式** 0： FIFO 模式 1： 优先级决定模式 在 FIFO 模式下，帧按照它们写入 STB 的顺序发送。在优先级决策模式中，首先自动发送 STB 中具有最高优先级的帧。帧的 ID 用于优先级决定。较低的 ID 表示一个具有较高优先级的帧。无论 ID 如何，PTB 中的帧始终具有最高优先级。只有当 STB 为空时，才能切换 TSMODE。
20：18 保留	保留
17：16 TSSTAT	**次发送状态位** 00： STB 为空 01： STB 小于或等于半满 10： STB 多于半满 11： STB 满
15 TBSEL	**发送缓冲区选择** 0： PTB (高优先级缓冲区) 1： STB 选择要加载消息的发送缓冲区，使用 TBUF 寄存器进行访问。在写入 TBUF 寄存器和设置 TSNEXT 时，TBSEL 需要始终保持稳定
14 LOM	**监听（Listen Only）模式** 0： 禁用 1： 使能 当发送处于活动状态时，不应启用 LOM。如果启用 LOM，则无法启动发送

字段	说明
13 STBY	**收发器待机（Transceiver Standby） 模式** 0： 禁用 1： 使能 该寄存器位连接到输出信号 standby，可用于控制收发器的待机模式 如果 TPE=1, TSONE=1 或 TSALL=1，STBY 不能被设置成 1 如果主机将 STBY 设置为 0，则收发器在主机请求新发送之前，需要等待时间。
12 TPE	**主发送使能** 1： 发送高优先级 PTB 中的消息 0： 未发送 PTB 如果设置了 TPE，则来自 PTB 的消息将在下一个可能的发送位置发送。来自 STB 的开始发送将在之前完成，但是等待新消息被延迟直到 PTB 消息已被发送。 TPE 保持设置，直到消息成功发送或使用 TPA 中止。 主控制器可以将 TPE 设置为 1 但不能将其重置为 0，这将只能使用 TPA 并中止消息。 在 CAN 控制器重置该位的短时间内，主机无法设置它。 如果 RESET = 1，STBY = 1，(LOM = 1 且 LBME=0)，该位将复位为硬件复位值。
11 TPA	**主发送中止** 1：中止 PTB 的发送，该发送已被 TPE = 1 请求但尚未启动。(消息的数据字节保留在 PTB 中) 0： 没有终止 该位必须由主控制器设置，并由 CAN 控制器复位。 设置 TPA 会自动取消激活 TPE。 主控制器可以将 TPA 设置为 1，但不能将其重置为 0。 在 CAN 控制器重置该位的短时间内，主机无法设置它。 如果 RESET = 1，该位将复位为硬件复位值。 TPA 不应与 TPE 同时设置。
10 TSONE	**次发送一帧使能** 1： STB 中的一个发送使能 在 FIFO 模式下，这是最早的消息。在优先级模式下，这是具有最高优先级的一个。 在优先级模式下，TSONE 难以处理，因为如果同时将新消息写入 STB，则不总是清楚哪个消息将被发送。 一旦总线空闲并且没有 PTB（TPE 位）的请求暂停，控制器就开始发送。 0：没有发送 STB TSONE 保持置位状态，直到消息成功发送或被 TSA 中止。 主控制器可以将 TSONE 设置为 1，但不能将其重置为 0，这只能使用 TSA 并中止消息。 在 CAN 控制器重置该位的短时间内，主机无法设置它。 如果 RESET = 1，STBY = 1，(LOM = 1 且 LBME=0)，该位将复位为硬件复位值。

字段	说明
9 TSALL	**次发送所有帧使能** 1：发送 STB 中的所有消息 一旦总线空闲并且没有 PTB（TPE 位）的请求暂停，控制器就开始发送。 0： STB 没有发送 TSALL 保持设置状态，直到所有消息都已成功发送或被 TSA 中止。 主控制器可以将 TSALL 设置为 1，但不能将其重置为 0，这只能使用 TSA 并中止消息。 在 CAN 控制器重置该位的短时间内，主机无法设置它。 如果 RESET = 1，STBY = 1，(LOM = 1 且 LBME=0)，该位将复位为硬件复位值。
8 TSA	**次发送中止** 1： 从 STB 中止已经请求但尚未启动的发送 对于 TSONE 发送，只有一帧被中止，而对于 TSALL 发送，所有帧都被中止。将释放一个或所有消息缓冲区，用于更新 TSSTAT。 所有中止的消息都将丢失，因为它们不再可访问。 在优先级模式下，如果 TSONE 发送中止，此时如果同时向 STB 写入新帧，则不清楚 哪个帧将被中止。 0： 不终止 该位必须由主控制器设置，并由 CAN 控制器复位。 设置 TSA，分别自动取消 TSONE 或 TSALL。主控制器可以将 TSA 设置为 1，但不能将其重置为 0。 如果 RESET = 1，该位将复位为硬件复位值。 TSA 不应与 TSONE 或 TSALL 同时设置。
7 RESET	**复位请求** 1： 主控制器执行 CAN 控制器的本地复位 0：CAN 控制器的非本地复位 若 RESET = 1，则只能修改某些寄存器（例如节点配置）。 RESET =1 会迫使多个组件进入复位状态。 **注意： RESET 切换为 0，在 11 个 CAN 位时间后，CAN 节点将参与 CAN 通信。 CAN 标准（总线空闲时间）需要此延迟。** 若 RESET 设置为 1 并立即设置为 0，则需要一些时间才能将 RESET 读取为 0 并变为非活动状态。原因是时钟从主机到 CAN 控制器需要转换。根据主机和 CAN 时钟之间的关系，RESET 按需保持活动状态。
6 LBME	**环回模式，外部** 0： 禁用 1： 使能 当发送处于活动状态时，不应启用 LBME。

字段	说明
5 LBMI	**环回模式，内部** 0： 禁用 1： 使能 当发送处于活动状态时，不应启用 LBMI。
4 TPSS	**PTB 单次发送模式** 0： 禁用 1： 使能
3 TSSS	**STB 单次发送模式** 0： 禁用 1： 使能
2 RACTIVE	**接收有效 (接收状态位)** 1： 控制器当前正在接收帧 0： 没有接收活动
1 TACTIVE	**发送有效（发送状态位）** 1： 控制器当前正在发送帧 0： 没有发送活动
0 BUSOFF	**总线关闭 (总线状态位)** 1： 控制器状态为“总线关闭（bus off）” 0： 控制器状态为“总线打开（bus on）” 在 busoff 状态下写 1 退出 bus off 状态且清零 TECNT/RECNT，这仅用于调试。

7.4.3 控制寄存器 1(CAN_CTRL1)

表 7-16 CAN_CTRL1 寄存器

CAN_CTRL1 CAN 控制寄存器 1 RESET：0x1B00007E

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	AFWL				EWL				EWARN	EPASS	EPIE	EPIF	ALIE	ALIF	BEIE	BEIF
访问	RW				RW				R	R	RW	W1C	RW	W1C	RW	W1C
Reset	0	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	RIF	ROIF	RFIF	RAFIF	TPIF	TSIF	EIF	AIF	RIE	ROIE	RFIE	RAFIE	TPIE	TSIE	EIE	TSFF
访问	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	RW	RW	RW	RW	RW	RW	RW	R
Reset	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0

字段	说明
31: 28 AFWL(3：0)	**接收缓冲区几乎满告警限制** AFWL 定义内部警告限制 AFWL_i，其中 ^{n}RB 是可用 RB 缓冲区的数量。 将 AFWL_i 与填充的 RB 缓冲区的数量进行比较，并且如果相等则触发 RAFIF。 AFWL_i 的有效范围： 1…n_{RB}. AFWL_i = 0 无意义，自动被视为 0x1。 **(请注意，AFWL 适用于此规则而非 AFWL_i)** AFWL_i > n_{RB} 无意义，自动被视为 nRB AFWL_i = n_{RB} 是有效值，但 请注意 RFIF 也存在。
27: 24 EWL(3：0)	**可编程错误警告限制值** = (EWL+1)*8。可能的限制值为 8, 16, …128。 EWL 的值控制 EIF。
23 EWARN	**达到错误警告限制** 1 ： 错误计数器 RECNT 或 TECNT 的一个等于或大于 EWL 0 ： 两个计数器的值都小于 EWL
22 EPASS	**错误被动模式激活**

字段	说明
	0： 没有激活 (节点错误激活) 1： 激活 (节点错误被动)
21 EPIE	**错误被动中断使能**
20 EPIF	**错误被动中断标志** 如果错误状态从错误激活变为错误被动或反之，并且如果启用此中断，则 EPIF 将被激活。 写 1 可以清除该标志。
19 ALIE	**仲裁失利中断使能**
18 ALIF	**仲裁失利中断标志** 写 1 可以清除该标志
17 BEIE	**总线错误中断使能**
16 BEIF	**总线错误中断标志** 总线错误类型对应 KOER。 写 1 可以清除该标志
15 RIF	**接收中断标志** 1： 数据或远程帧已接收，并且可在接收缓冲区中使用 0： 没有帧被接收 写 1 可以清除该标志
14 ROIF	**RB 溢出中断标志** 1： RB 中至少有一条接收消息被覆盖 0： 没有 RB 被覆盖 如果发生溢出，ROIF 和 RFIF 都将被设置。 写 1 可以清除该标志
13 RFIF	**RB 满中断标志** 1： 所有 RB 都已满。如果在收到下一个有效消息之前没有释放 RB，则最旧的消息将会丢失 0： RB FIFO 未满 写 1 可以清除该标志
12 RAFIF	**RB 几乎满中断标志** 1： 填充的 RB 缓冲区数 ≥ AFWL 0： 填充的 RB 缓冲区数 < AFWL 写 1 可以清除该标志

字段	说明
11 TPIF	**主发送中断标志** 1：已成功完成所请求的 PTB 发送 0： 没有完成 PTB 的发送 写 1 可以清除该标志
10 TSIF	**次发送中断标志** 1： 已成功完成所请求的 STB 发送 0： 没有完成 STB 的发送 写 1 可以清除该标志
9 EIF	**错误中断标志** 1：错误警告限制的边界已在任一方向上穿越，或者 BUSOFF 位已在任一方向上改变 0： 没有改变
8 AIF	**中止中断标志** 1：在设置 TPA 或 TSA 之后，已经中止了相应的消息 建议不要同时设置 TPA 和 TSA，因为两者都是源 AIF。 0： 没有执行中止。 AIF 没有关联的使能寄存器。 写 1 可以清除该标志
7 RIE	**接收中断使能** 0： 禁用 1： 使能
6 ROIE	**RB 溢出中断使能** 0： 禁用 1： 使能
5 RFIE	**RB 满中断使能** 0： 禁用 1： 使能
4 RAFIE	**RB 几乎满中断使能** 0： 禁用 1： 使能
3 TPIE	**主发送（Transmission Primary）中断使能** 0：禁用 1： 使能

字段	说明
2 TSIE	**次发送（Transmission Secondary ）中断使能** 0： 禁用 1： 使能
1 EIE	**错误中断（Error Interrupt） 使能** 0： 禁用 1： 使能，中断标志对应 EIF
0 TSFF	**次发送缓冲区满标志** 1： STB 填充最大数目的消息 0： STB 没有填充最大数目的消息

7.4.4 低速波特率配置寄存器(CAN_SBITRATE)

表 7-17 CAN_SBITRATE 寄存器

CAN_SBITRATE 低速波特率配置寄存器 RESET: 0x01020203

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	S_PRESC									S_SJW						
访问	RW									RW						
Reset	0x01									0x02						
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称		S_SEG2							S_SEG1							
访问		RW							RW							
Reset		0x02							0x03							

字段	说明
31: 24 S_PRESC	**预分频器** 预分频器将 CAN 时钟分频以获得时间量子时钟 tq_clk。有效范围 S_PRESC=[0x00, 0xff]，导致分频器值为 1 ~ 256。
22: 16 S_SJW	**同步补偿宽度** 同步补偿宽度 t_{SJW} =(S_SJW+1).TQ 是缩短或延长重新同步的位时间的最长时间，其中 TQ 是时间因子。
14: 8 S_Seg_2	**位定时段 2** 在采样点后定时 t_{Seg_2} =(S_Seg_2 +1).TQ。
7: 0 S_Seg_1	**位定时段 1** 位定时开始后，采样点被设置为 t_{Seg_1} =(S_Seg_1+ 2).TQ

7.4.5 快速波特率配置寄存器 (CAN_FBITRATE)

表 7-18 CAN_FBITRATE 寄存器

CAN_FBITRATE 快速 CAN 波特率配置寄存器 **RESET：0x01020203**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	F_PRESC												F_SJW			
访问	RW												RW			
Reset	0x01												0x02			
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称					F_SEG2							F_SEG1				
访问					RW							RW				
Reset					0x02							0x03				

字段	说明
31: 24 F_PRESC	**预分频器** 预分频器将 CAN 时钟分频以获得时间量子时钟 tq_clk。有效范围 F_PRESC=[0x00, 0xff] 导致分频器值为 1 ~ 256。
19: 16 F_SJW	**同步补偿宽度** 同步补偿宽度 t_{SJW} =(F_SJW+1).TQ 是缩短或延长重新同步的位时间的最长时间，其中 TQ 是时间因子。
11: 8 F_Seg_2	**位定时段 2** 在采样点后定时 t_{Seg_2} = (F_Seg_2 +1).TQ。
4: 0 F_Seg_1(6：0)	**位定时段 1** 位定时开始后，采样点被设置为 t_{Seg_1} =(F_Seg_1+ 2).TQ。

7.4.6 错误类型和错误计数寄存器(CAN_ERRINFO)

表 7-19 CAN_ERRINFO 寄存器

CAN_ERRINFO CAN 错误类型和错误计数寄存器 **RESET: 0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	TECNT								RECNT							
访问	R								R							
Reset	0								0							
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	TDCEN	SSPOFF							KOER			ALC				
访问	RW	RW							R			R				
Reset	0	0							0			0				

字段	说明
31: 24 TECNT	**发送错误计数器（发送过程中的错误数）** TECENT 按照 CAN 规范中的定义递增和减少 TECNT 会溢出。 有关 TECNT 和“总线关闭（bus off）”状态的更多详细信息，请参考 7.3.2 节。
23: 16 RECNT	**接收错误计数器 （接收过程中的错误数）** RECENT 按照 CAN 规范中的定义递增和减少 RECNT 不会溢出， RECNT 将 0xff = 255 作为最大值 有关 RECNT 和“总线关闭（bus off）”状态的更多详细信息，请参考 7.3.2 节。
15 TDCEN	**发送器延迟补偿使能** 如果 TDCEN = 1，则在 BRS 处于活动状态时，将在 CAN FD 帧的数据阶段激活 TDC
14: 8 SSPOFF	**二次采样点偏移** 传输延迟 SSPOFF 为 TDC 定义了二次采样点的时间 SSPOFF 是 TQ 的一个数
7: 5 KOER	**错误类型 (错误代码)** 000：没有错误（no error） 001：位错误（BIT ERROR） 010：形式错误（FORM ERROR） 011：填充错误（STUFF ERROR） 100：应答错误（ACKNOWLEDGEMENT ERROR） 101：校验错误（CRC ERROR） 110：其他错误（OTHER ERROR） (错误标志后收到显性电平，接收到主动错误标志太长，ACK 错误后的被动错误标志收到显性电平)。 111： 未使用 KOER 会随着每个新错误而更新。因此当帧被成功地发送或接收时，它保持不变。
4: 0 ALC	**仲裁失利捕获** (仲裁失利的帧中的位所处的位置)

7.4.7 接收滤波器控制寄存器(CAN_ACFCTRL)

表 7-20 CAN_ACFCTRL 寄存器

CAN_ACFCTRL **接收滤波器控制寄存器** **RESET: 0x00010000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	ACFENx															
访问	RW															

Reset	0															1
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称							TIME POS	TIME N			SEL MASK		ACFADR			
访问							RW	RW			RW		RW			
Reset							0	0			0		0			

字段	说明
31: 16 ACFENx	**接受过滤器使能(x=0-15)** 1： 接受过滤器使能 0： 接受过滤器禁用 每个接收过滤器 (AMASK/ACODE)可以单独启用或禁用。硬件复位后，默认情况下仅启用 0 号过滤器。 禁用过滤器拒绝接收消息。如果相应的 AMASK/ACODE 配置匹配，则仅启用的过滤器可以接受消息。 要接受所有消息，必须通过设置 AE_x = 1，AMASK_x = 0xff 和 ACODE_x = 0x00 来启用一个过滤器 x。这是过滤器 x = 0 硬件重置后的默认配置，而所有其他过滤器都被禁用。
9 TIMEPOS	**时间戳标记位置** 0： SOF 1： EOF 仅当 TIMEEN = 0 时才能更改 TIEMPOS，也可同时修改 TIMEPOS 和设置 TIMEEN=1
8 TIMEEN	**时间戳标记使能** 0： 禁用 1： 使能
5 SELMASK	**选择接受掩码（Select acceptance MASK）** 0： ACF_x 寄存器 指向验证码 1： ACF_x 寄存器指向接收掩码 ACFADR 选择一个特定的接收滤波器。
3: 0 ACFADR	**接收滤波器地址** ACFADR 指向某一个接收滤波器 0-15。选择后才能使用寄存器 ACF_x 设置对应滤波器。 SELMASK 位在所选接受过滤器的验证码和掩码之间进行选择。

7.4.8 接收代码 ACODE 寄存器(CAN_ACF)

表 7-21 CAN_ACF 寄存器

CAN_ACF　　接收代码 ACODE 寄存器　　RESET：　0x0

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

字段	说明
28: 0 ACODE	**接受代码（Acceptance CODE），需设置 SELMASK 等于 0** 0/1： 每个位与接收消息的 ID 位进行比较 ACODE_x(10：0)用于标准帧，ACODE_x(28：0)用于扩展帧 只有 0 号过滤器受上电复位的影响，所有其他过滤器保持未初始化状态。

7.4.9　接收代码 AMASK 寄存器(CAN_ACF)

表 7-22 CAN_ACF 寄存器

CAN_ACF　　接收代码 AMASK 寄存器　　RESET：　0x0

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

字段	说明
30 AIDEE	**接受掩码 IDE 位检查使能，需设置 SELMASK 等于 1** 1： 接收过滤器接收 AIDE 定义的标准或扩展 0： 接收过滤器接收标准或扩展帧 只有 0 号过滤器受上电复位的影响，所有其他过滤器保持未初始化状态。
29 AIDE	**接受掩码 IDE 位的值，需设置 SELMASK 等于 1** 若 AIDEE=1， 则： 1： 接受过滤器仅接受扩展帧 0： 接受过滤器仅接受标准帧

字段	说明
28: 0 AMASK	**接收 MASK，需设置 SELMASK 等于 1** 1： 屏蔽对应的接收过滤位 0： 匹配对应的接收过滤位 AMASK_x(10：0)被用做标准帧，AMASK_x(28：0) 被用做扩展帧。 禁用位导致接受该消息。 因此，过滤器 0 重置后的默认配置接受所有消息。 只有 0 号过滤器受上电复位的影响，所有其他过滤器保持未初始化状态。

7.4.10 控制器版本寄存器(CAN_VERSION)

表 7-23 CAN_VERSION 寄存器

CAN_VERSION **CAN 控制器版本寄存器** **RESET： 0x0000708**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	VERSION															
访问	R															
Reset	0x0708															

字段	说明
15：0 VERSION	**CAN 控制器版本** 以十进制数表示。例如：VERSION=0708 表示主版本为 0x7，次版本为 08。

8 通用异步收发器（UART）

8.1 简介

UART (Universal Asynchronous Receiver/Transmitter，通用异步收发器) 是一种基本的用于串行通信的协议。它主要通过发射器和接收器来实现诸多功能。主要功能由寄存器的 3 个 bit: **LINEN, RS485EN, ILEN** 共同组成，如表 8-1 所示。

表 8-1 功能分类和配置

功能	LINEN	RS485EN	ILEN
BASIC UART	0	0	0
RS485	0	1	0
LIN	1	0	0

【说明】

1. 只有 UART0，UART1 支持软件 LIN 功能。
2. 当处于 RS485 模式时，必须禁用硬件流控制（RTS,CTS）功能。
3. LIN 模式仅支持 8 位数据格式及 16 倍过采样。与此同时，如果使能自动波特率功能 (LABAUDEN=1)，则 同步字段数据 (0x55) 将被丢弃。
4. DMA 功能必须在 FIFO 使能时起作用。

8.2 特性

- 全双工，标准不归零（NRZ）格式
- 可编程波特率（16 位分频器）
  - 支持发送或接收波特率范围 600bps~3Mbps，波特率误差不超过 1%
- 轮询或中断方式查询状态
  - 传输数据寄存器为空且传送完成
  - 接收数据寄存器已满
  - 接收溢出错误、帧错误、奇偶错误
  - 空闲线路检测
  - 支持 LIN 的分隔符检测，可选 10 或 11 位 LIN 功能分隔符检测
  - 通过有效的边沿检测将 MCU 从停止（Stop）模式唤醒
- 支持 DMA

- 可编程 8 或 9 位数据， 1 或 2 位停止位，硬件自动生成奇偶校验位
- 可选择传输器输出和接收器输入极性
- 支持硬件流控制
- 可生成 13~28 位分隔符
- 支持 RS485 自动控制方向

8.3 结构框图

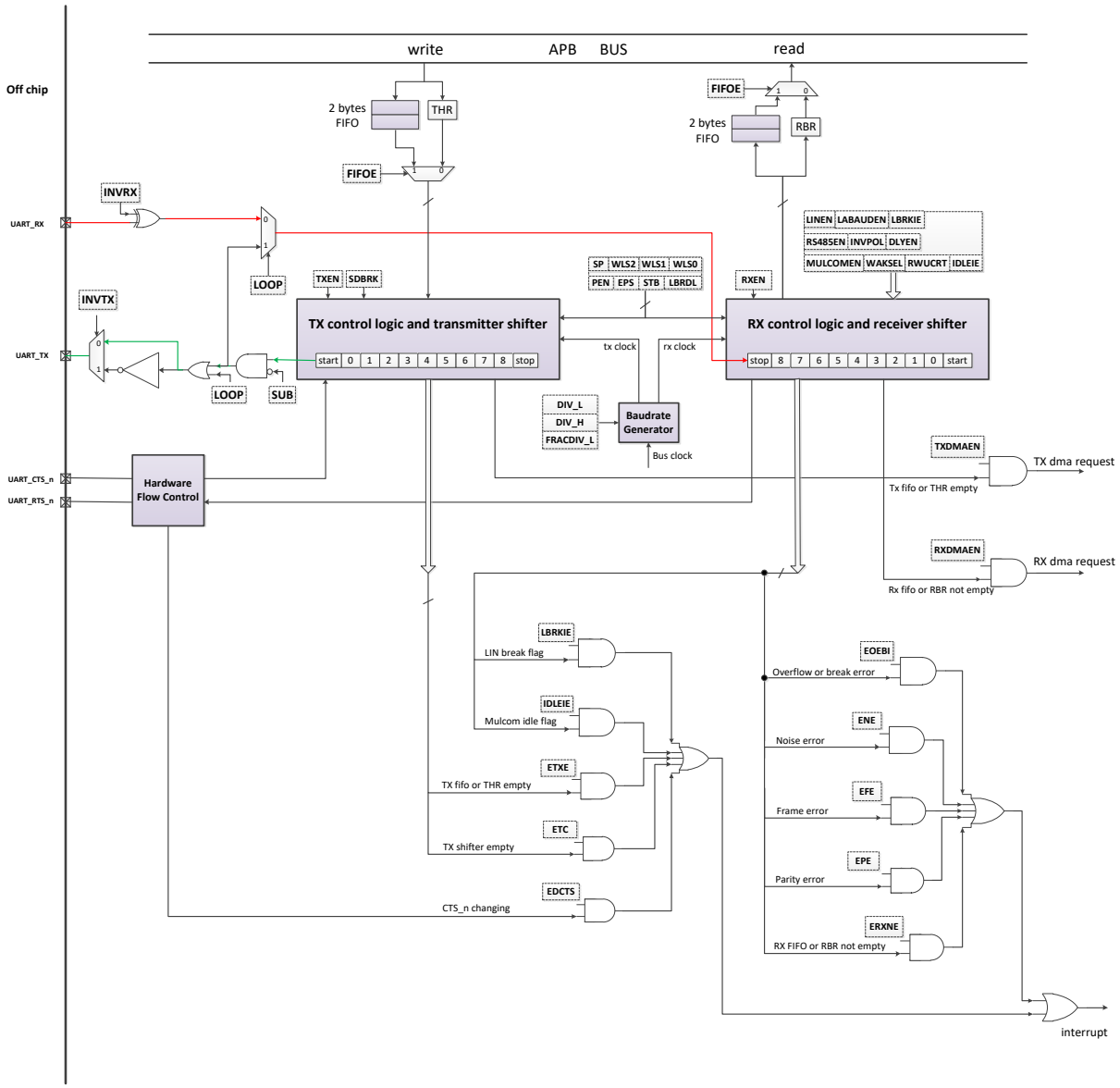

图 8-1 UART 结构框图

8.4 功能描述

UART 功能是逐位发送和接收串行数据。图 8-2 和图 8-3 描述了完整的数据位，包括起始位（start bit）、数据位（data bits）、奇偶位（ parity bit）、停止位（ stop bits）和保护间隔（ guard time）。但 bit6，bit7， bit8， bit9，parity，stop2 位和 guard time 位可由用户配置，详细配置信息请参考 UART_LCR0 和 UART_LCR1 寄存器。一位对应于由波特率控制的一个位时间。

UART 发送和接收时有多种状态。用户最好知道在发送或接收过程中何时产生这些状态。这样，用户可以更好地使用 UART 功能。THRE 和 TC 状态位出现在图 8-2 所示的发送过程中。在全局复位或上电后的初始化状态期间，THRE 和 TC 在 TXEN 设置为 1 后会立即变为 1。但在传输过程中，THRE 会在起始位后立即变为 1，同时 TC 在最后一位后立即变为 1，比如，若 GUARDEN=1，保护时间位也会如此。

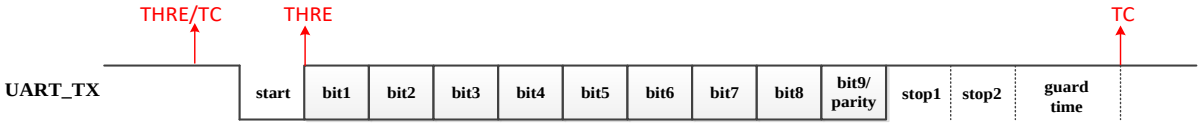

图 8-2 UART 传送器流程

如果在接收过程中发生相关事件，如图 8-3 所示，则状态位 DR， OE， PE， FE， BI 和 NE 会在 stop1 位后立即置为 1。

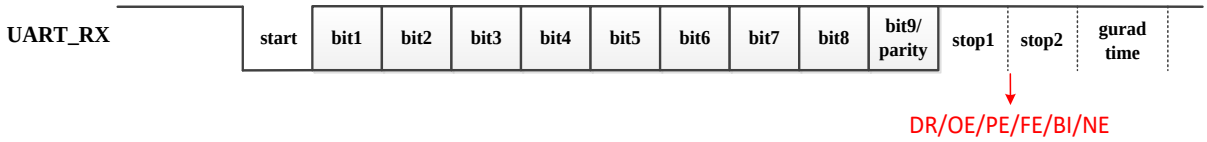

图 8-3 UART 接收器流程

需要指出的是，PE，FE 和 NE 状态仅针对当前接收数据字节，并会在下一个数据接收完成时被自动清除。对于其他状态，如果未通过读取或写入 1 来清除它们，则它们始终保持该值。

8.4.1 输入输出定时

表 8-2 UART 输入输出定时

引脚名	相应信号	宽度	输入/输出	是否上拉	定时限制
UARTx_TX	uart_tx	1	O（输出）	否	无
UARTx_RX	uart_rx	1	I（输入）	否	无
UARTx_RTS	uart_rts_n	1	O（输出）	否	无
UARTx_CTS	uart_cts_n	1	1（输入）	否	无

注意

x= 0~2.。仅 UART0 具有完整的硬件流控制，其四个信号(UART0_TX, UART0_RX, UART0_RTS, UART0_CTS) 都可以在引脚中找到。

8.4.2 噪声检测（Noise Detection）

对于 NE 状态，当 UART_RX 信号中存在噪声时就会产生此信号。为了检测噪声， UART_RX 在中间位置被采样三次，如图 8-4 所示。

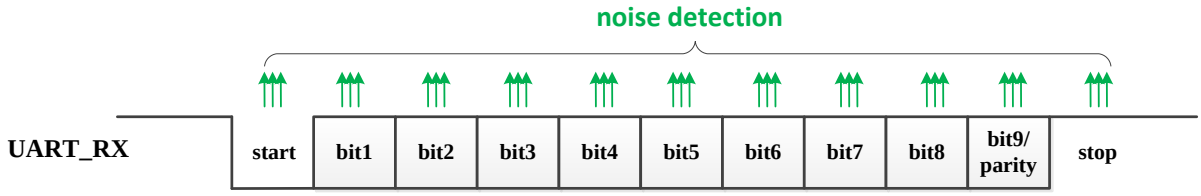

图 8-4 UART 噪声检测

为了更清楚方便地解释，三次采样的值分别叫做 SM1，SM2 和 SM3。 如果 SM1，SM2 和 SM3 中有两个以上的“1”作为起始位，则起始位无效且接收器复位成再次接收。除了起始位的这种情况之外，如果 SM1，SM2 和 SM3 的值互不相同，则在 NE 状态变为 1 的情况下检测噪声。

8.4.3 波特率描述

UART 波特率精度由多个方面确定，包括 UART 时钟、过采样时间等。因此，一些特定的、过高的波特率没有实现或实现起来有大量误差。表 8-3 和表 8-4

表 8-4 分别描述了典型波特率在不同的系统时钟下的配置及相应的误差率。

表 8-3 典型的波特率及误差率@bclock=48MHz

序号	理论值(Kbps)	实际值(bps)	DIV_MAN[15:0]	DIV_FRAC[4:0]	过采样次数	误差率
1	2.4	2399.98	1250	0	16	-0.001%
2	9.6	9599.69	312	16	16	-0.003%
3	19.2	19202.22	156	8	16	0.006%
4	57.6	57603.69	52	3	16	0.006%
5	115.2	115207.38	26	1	16	0.006%
6	230.4	230414.75	13	1	16	0.006%
7	460.8	460829.50	6	16	16	0.006%
8	921.6	917431.19	3	8	16	-0.452%
9	1843.2	1834862.38	3	8	8	-0.452%
10	4200	4166666.75	1	14	8	-0.794%

表 8-4 典型的波特率及误差率@bclock=24MHz

序号	理论值 (Kbps)	实际值 (bps)	DIV_MANTI[15:0]	DIV_FRAC[4:0]	过采样次数	误差率
1	2.4	2400	625	0	16	0
2	9.6	9600	156	8	16	0
3	19.2	19200	78	4	16	0
4	57.6	57600	26	1	16	0
5	115.2	115200	13	1	16	0
6	230.4	230769.23	6	16	16	0.16%
7	460.8	461538.47	3	8	16	0.16%
8	921.6	923076.94	1	20	16	0.16%
9	1843.2	-	-	-	-	-
10	4200.0	-	-	-	-	-

8.4.4 硬件流控制功能

UART 硬件流控制两个 UART 设备 RTS_n 和 CTS_n 间的通信以减少 CPU 负荷。这样，通信双方可以从容地逐一处理数据。实际应用链接如图 8-5 所示。

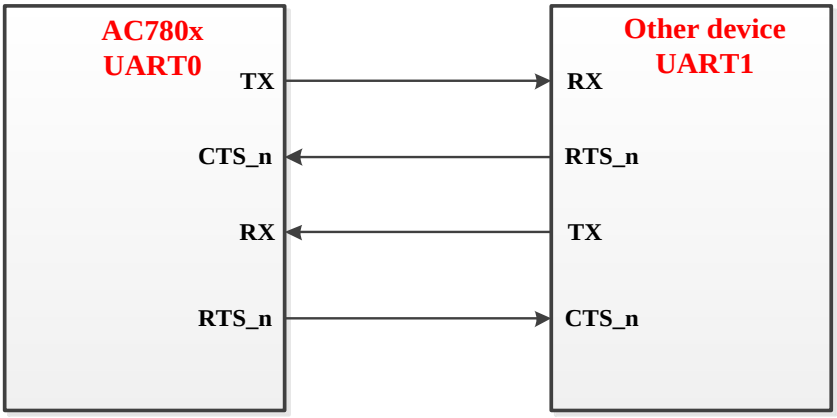

图 8-5 硬件流控制连接

在图 8-5 中，当 UART0 RX 数据寄存器或 FIFO 已满，UART0 的信号 RTS_n 可以通知 UART1 不发送数据。当 UART0 RX 数据寄存器或 FIFO 通过读操作变为未满时， UART0 RTS_n 自动变为低电平。然后 UART1 可以通过检测 UART1 CTS_n 为低电平来传输数据。以类似的方式，UART1 RTS_n 和 UART0 CTS_n 用于控制 UART0 传输。

特别地，如果在 UART1 传输数据期间，UART1 CTS_n 变为高电平，则将完整发送当前数据。因此，用户通常应检查 CTS 或 CTS_n 状态以使用硬件流控制功能和其他状态位。

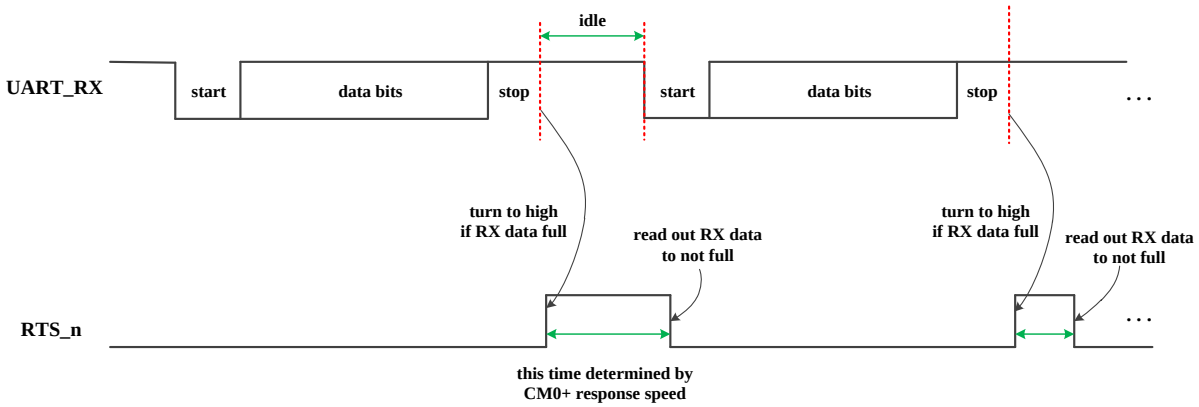

图 8-6 硬件流控制原理

8.4.5 RS485 功能

和 UART 功能相比，RS485 功能有一个自动方向控制信号 UART_RTS，如图 8-7 和图 8-8 所示。其在接收数据时默认为低电平，发送数据时默认为高电平。由于采用半双工，RS485 在同一时刻可以实现发送或接收操作中的一个。在图 8-7 中，有两个延迟（delay），delay1 用于在实际传输数据之前上拉 UART_RTS 信号，而 guard time 位用于在数据位传输已经全部完成后下拉 UART_RTS 信号。实际的 PCB 布线延迟可能导致 UART_RTS 信号在 UART_TX 之后变为高电平，从而可能损坏第一个数据位。因此，延迟有助于保证整个传输过程中 UART_RTS 为高电平。传输完成后，UART_RTS 将自动变为低电平，使 UART 处于接收状态。

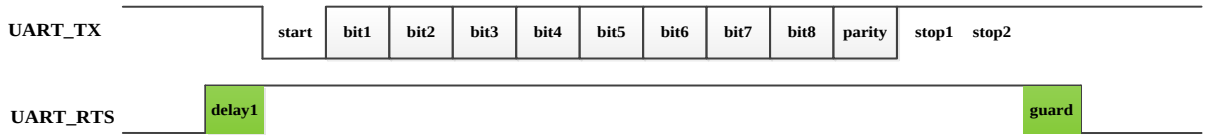

图 8-7 单字节数据传输

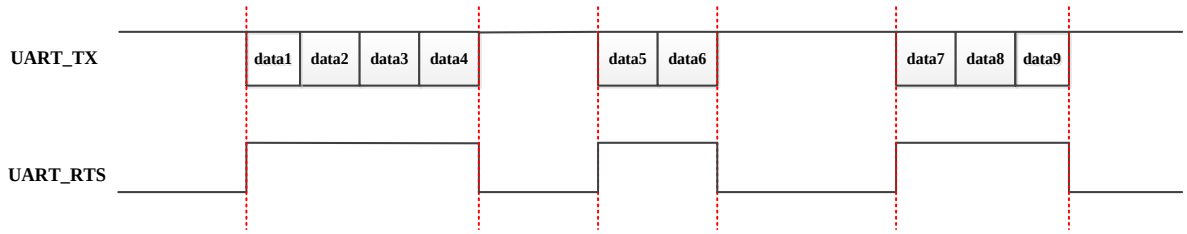

图 8-8 多字节数据传输

图 8-9 描述了用户应用连接的一个典型实例。UART_RTS 充当 MAX485 的方向控制信号。使用此连接方法，UART_RTS 的默认为低电平，因此 MAX485 处于默认接收条件下。当 UART 想要传输数据时，UART_RTS 信号由硬件设置为高电平。

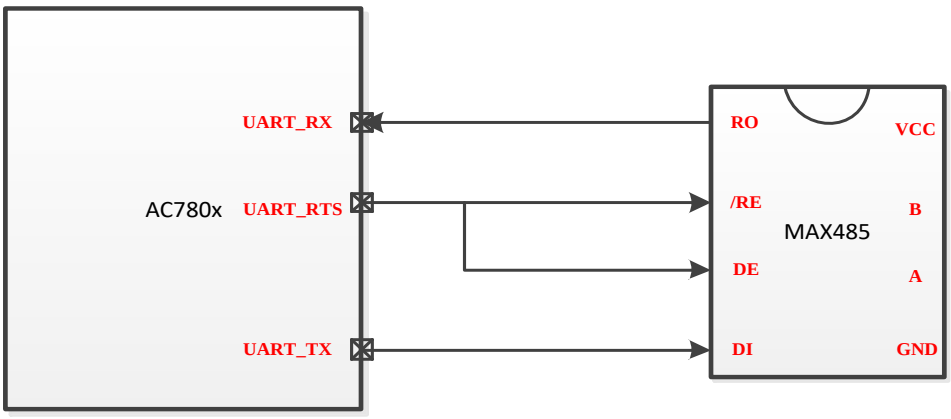

图 8-9 实际电路连接

8.4.6 LIN 功能

如图 8-10 所示，UART LIN 只是一个软件 LIN，具有传输分隔符（break field），同步域（synchronous field）和数据。用户可以在最新的 LIN 协议中了解更多细节。图 8-10 仅描述了一个基本的帧。除基本 UART 寄存器外，用户还应更加注意 UART_LINCR 寄存器。在 UART 模块中，存在一个硬件逻辑单元，用于 LIN 检测，并且当 LINEN 配置为 1 时，就启用 LIN 功能。需要注意的是只有 UART0，UART1 支持 LIN 功能。

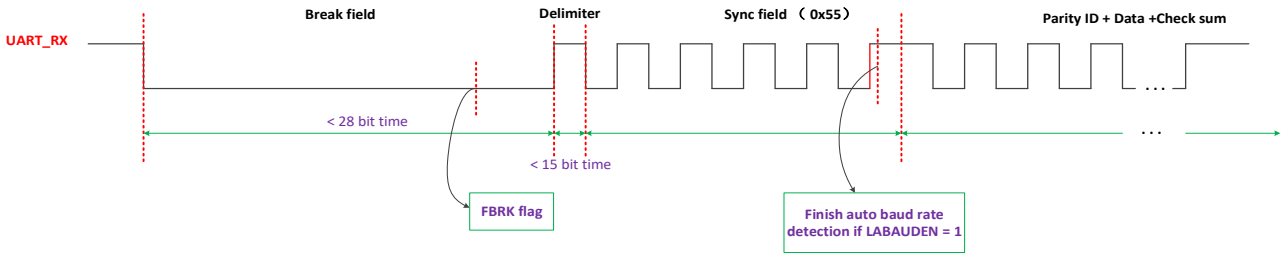

图 8-10 LIN 帧流程

UART-LIN 帧流程如图 8-10 所示, 接收过程参考如下步骤：

1. 当 UART 接收到 10（LBRKDL=0）或 11（LBRKDL=1）位 0 时，UART LIN 检测逻辑单元将其视为 LIN 帧的有效分隔符，LIN 分隔符标志 FBRK 由硬件设置为 1，表示出现了有效的 LIN 分隔符。需要指出的是，LIN 分隔符不是数据，也不存储在 UART 接收数据寄存器或 FIFO 中。
2. 在分隔符位周期之后，同步域即 0x55 当做正常的的数据接收。如果 LABAUDEN 配置为 1，则在 synchronous field 期间自动波特率检测开始运行，并且自动波特率检测操作在图 8-10 所示的第五个上升沿之后完成。但数据 0x55 不会存储到 RX 数据寄存器或 FIFO 中。若 LABAUDEN 配置为 0，则不执行自动波特率检测，数据 0x55 存储在 RX 数据寄存器或 FIFO 中。
3. 在 synchronous field 之后，数据流是用户的有用数据，并由 UART 无差别地接收。

为避免在发生异常情况时模块暂停，引入超时机制，如图 8-10 所示：

1. LIN slave 接收到的 break field 超过 28 位时间，硬件会置位寄存器 UART_LSR1 的 BRKWAK 位，如果使能了寄存器 UART_LINCR 的 BRKWAKIE 位，则会产生唤醒中断。
2. 对于 delimiter 超过 15 位时间会导致模块重置接收器和 LIN 检测逻辑。
3. 如果同步段出错，硬件置位 UART_LSR1 的 SYNERR 位，如果使能 UART_LINCR 的 SYNERRIE 位，当同步出错时候，硬件产生同步段出错中断。

UART-LIN 传输过程参考如下步骤：

1. 作为软件 LIN，如何传输 break field 是关键步骤。当用户想要传输 break field 时，用户应检查 UART_LSR0[THRE]的状态。发送的 break field 长度取决于寄存器 UART_BRKLH 的配置。

   寄存器 UART_BRKLH 的取值为 0-15 分别对应 13-28 bit 的 break field。此时如果 UART_LSR0[THRE]值为 1 时， 用户向寄存器 UART_LINCR[SBRK]写入 1 就可以发送 break field。需要指出的是，当 UART_LINCR[SDBRK]写入 1 时，在当前数据已经完全传输后，或当 UART 处于空闲（idle）状态时，立即传输 break field。UART_LINCR[SBRK]将由硬件自动清除。

2. 将同步域（0x55）和其他后续数据写入 THR 数据寄存器并作为正常数据发送。

8.4.7 两种电源模式

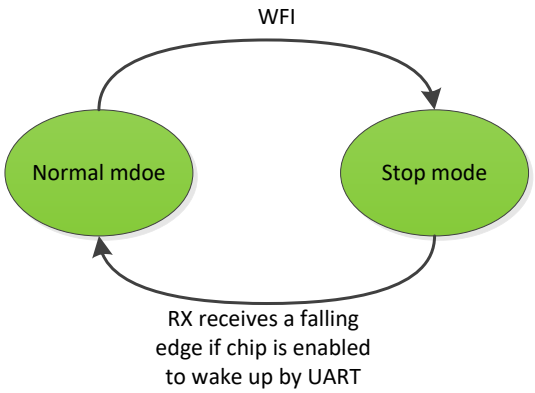

图 8-11 运行模式（Run mode）和停止模式（Stop mode）

本节介绍在两种电源模式下 UART 的状态。如图 8-11 所示，用户可以执行 WFI 指令使芯片进入停止（Stop）模式, 芯片功耗将明显降低，UART 模块的寄存器配置保持，唤醒后不需要重新配置。

在停止模式下，如果芯片被 UART 唤醒，则当 UART 接收下降沿时，芯片可以唤醒至正常模式。由于唤醒流程所需的时间，UART 通常可以在 5ms 的总时间之后立即接收数据。详细而言，TX1 可以发送 0xFF 作为下降沿，实际上其他数据也可以。特别地，如果 TX1 在唤醒流程中向 RX2 发送一些数据，则数据将在 RX2 中丢失。

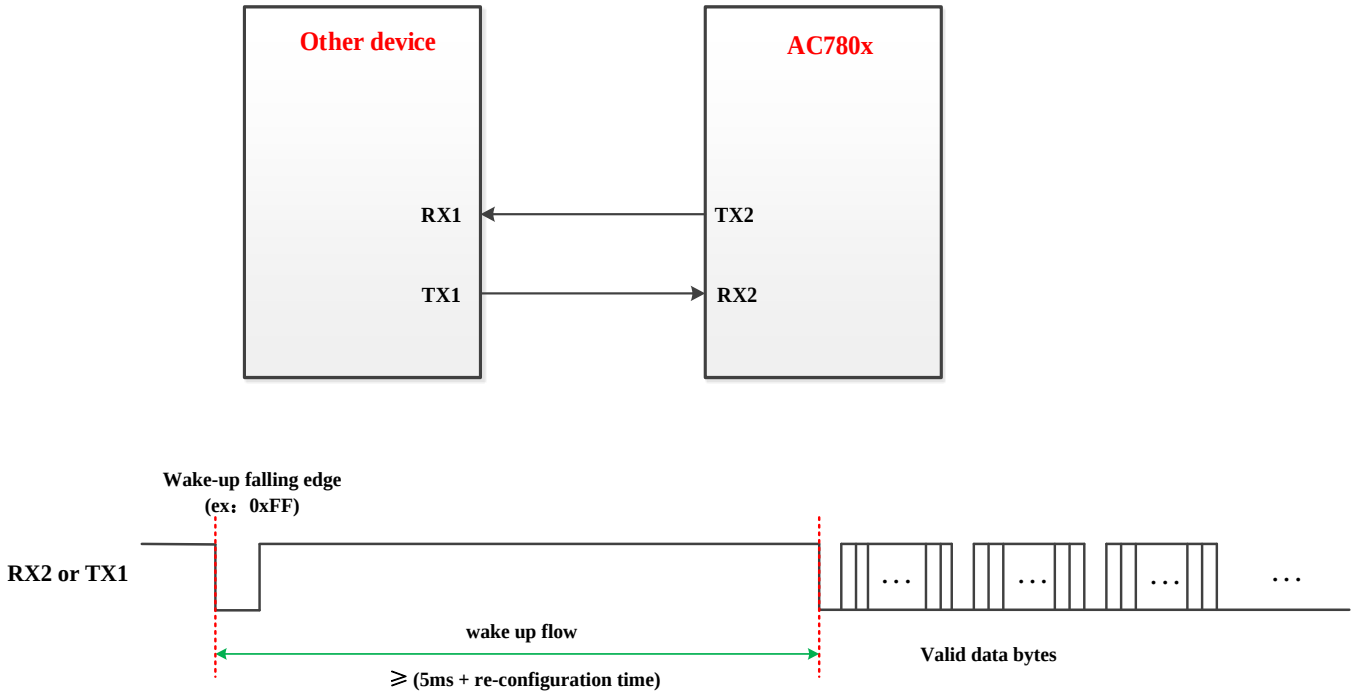

图 8-12 通过 UART 唤醒芯片的典型流程

8.5 应用说明

8.5.1 波特率配置说明

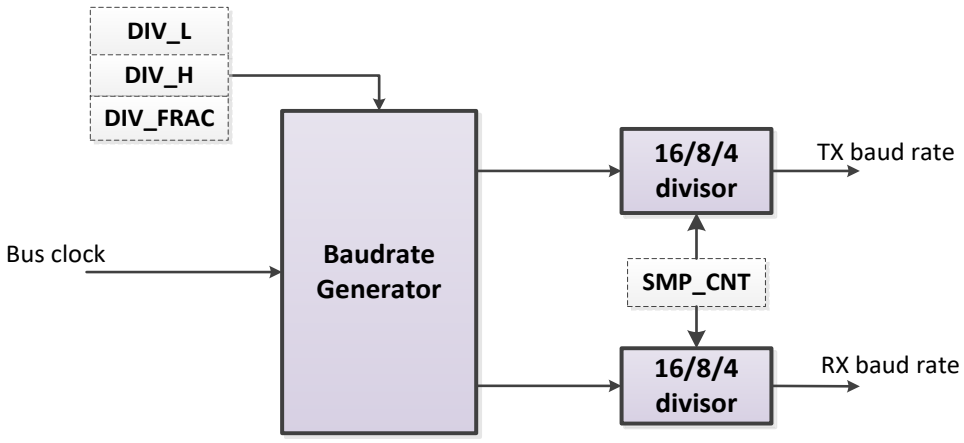

图 8-13 波特率发生器框图

如图 8-13 所示，波特率相关的配置寄存器如下所示：UART_SMP_CNT/UART_DIV_H/ UART_DIV_L /UART_DIV_FRAC。在寄存器映射表中描述了详细的信息。对于配置，如下公式用于波特率配置。

$$Baudrate = \frac{f_{clk}}{DIV * SMP\_CNT}$$

在上面公式中， DIV = {UART_DIV_H, UART_DIV_L} ; SMP_CNT = UART_SMP_CNT。

例如：如果用户希望在 8 倍采样模式下以 24MHz 总线频率获得波特率 230400 bps，则可以按如下方式获得 DIV 配置。因此， **UART_DIV_L=13, UART_DIV_H=0, UART_DIV_FRAC=32 * 0.0208=1。**

$$DIV = \frac{24000000}{Baudrate*SMP\ CNT} = \frac{24000000}{230400*8} = 13.0208$$

结合上面给出的例子，可以清晰地解释 UART_DIV_FRAC[4：0] 。在上面的表达式中， DIV 不是整数。如果用户删除小数部分，波特率的准确度将降低。特别是在大波特率条件下，丢弃 DIV 小数部分可能会将精度降低到较低水平，这样正常的数据传输可能会出错。

为了保持高精度， UART_DIV_FRAC[4： 0] 配置为 DIV 小数部分。由于宽度为 5 位，UART_DIV_FRAC 取值范围是 0 至 31。因此，32 乘以 DIV 小数部分生成 UART_DIV_FRAC[4：0] 的配置值。

8.5.2 UART 配置说明

配置步骤：

1. TX/RX 数据存储模式： UART_FCR
2. 波特率： UART_DIV_L/UART_DIV_H/ UART_DIV_FRAC/UART_SMP_CNT
3. DMA： UART_DMA_EN

⚠ 注意

DMA 功能必须在 FIFO 模式下才能起作用。

---

4. 数据格式： UART_LCR0/UART_LCR1

⚠ 注意

务必注意 SB 位。

---

5. 功能配置： UART_RS485CR/UART_LINCR/UART_IDLE/UART_CNRT 等。

注意

本步骤对不同功能来说是可选的。

6. 中断使能： UART_IER
7. 收发器使能：UART_LCR1[TXEN] / UART_LCR1[RXEN]
8. 发送或接收数据：UART_THR/UART_RBR

注意

此步骤实际上在正常的发送或接收数据过程中。

【说明】

1. 对于 LIN 功能，数据格式必须配置为 8 位，没有奇偶校验，16 次过采样。。
2. 对于 LIN 功能，当 LABAUDEN = 0 时，将接收 sync field 数据（0x55）并将其存储到 FIFO 或 RX 寄存器中，当 LABAUDEN = 1 时，将接收 sync field 数据（0x55），且不存储到 FIFO 或 RX 寄存器中。
3. 对于 RS485 功能， RTS_n PIN 用做发送或接收方向控制信号。

8.6 寄存器定义

表 8-5 UART 寄存器映射

UART0 基地址：0x40018000
UART1 基地址：0x40019000
UART2 基地址：0x4001A000

地址	名称	宽度	描述
UARTx 基地址+0x00	UART_RBR/THR	32	TX 保持寄存器 /RX 缓冲区寄存器
UARTx 基地址+0x04	UART_DIV_L	32	分频器低 8 位
UARTx 基地址+0x08	UART_DIV_H	32	分频器高 8 位
UARTx 基地址+0x0C	UART_LCR0	32	UART 辅助控制寄存器 0
UARTx 基地址+0x10	UART_LCR1	32	UART 辅助控制寄存器 1
UARTx 基地址+0x14	UART_FCR	32	FIFO 控制寄存器
UARTx 基地址+0x18	UART_EFR	32	硬件流使能寄存器
UARTx 基地址+0x1C	UART_IER	32	中断使能寄存器

地址	名称	宽度	描述
UARTx 基地址+0x20	UART_LSR0	32	状态寄存器 0
UARTx 基地址+0x24	UART_LSR1	32	状态寄存器 1
UARTx 基地址+0x28	UART_SMP_CNT	32	UART 采样计数寄存器
UARTx 基地址+0x34	UART_GUARD	32	保护时间（Guard time） 添加寄存器
UARTx 基地址+0x38	Reserved	32	
UARTx 基地址+0x3C	UART_SLEEP_EN	32	休眠使能寄存器
UARTx 基地址+0x40	UART_DMA_EN	32	DMA 使能寄存器
UARTx 基地址+0x44	UART_DIV_FRAC	32	小数分频器寄存器
UARTx 基地址+0x48	Reserved	32	
UARTx 基地址+0x4C	UART_RS485CR	32	RS485 控制寄存器
UARTx 基地址+0x50	Reserved	32	
UARTx 基地址+0x54	UART_CNTR	32	RS485 延迟时间
UARTx 基地址+0x58	UART_IDLE	32	空闲中断使能寄存器
UARTx 基地址+0x5C	UART_LINCR	32	软件 LIN 控制寄存器 **注：UART2 不支持 LIN, 没有该寄存器**
UARTx 基地址+0x60	UART_BRKLH	32	软件 LIN 同步间隔段控制寄存器 **注：UART2 不支持 LIN, 没有该寄存器**

8.6.1 RX/TX 数据寄存器(UART_RBR/THR)

表 8-6 UART_RBR/THR 寄存器

UART_RBR/THR **RX/TX 数据寄存器** **Reset：0x00000000**

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

字段	说明
8：0 RBR/THR	**RX/TX 数据寄存器** 通过访问该寄存器可以读取接收到的数据，且发送数据可以写入该寄存器。数据长度不超过 9 位。

8.6.2 分频器低 8 位寄存器(UART_DIV_L)

表 8-7 UART_DIV_L 寄存器

UART_DIV_L **分频器低 8 位寄存器** **Reset：0x00000001**

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

字段	说明
7：0 DIV_L	**波特率分频器** 分频器低 8 位

8.6.3 分频器高 8 位寄存器(UART_DIV_H)

表 8-8 UART_DIV_H 寄存器

UART_DIV_H **分频器高 8 位寄存器** **Reset：0x00000000**

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

字段	说明
7：0 DIV_H	**波特率分频器** 分频器高 8 位

8.6.4 控制寄存器 0(UART_LCR0)

表 8-9 UART_LCR0 寄存器

UART_LCR0 **控制寄存器 0** **Reset：0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称										SUB	SP	EPS	PEN	STB	WLS1_WLS0	
访问										RW	RW	RW	RW	RW	RW	
Reset										0	0	0	0	0	0	

注意

一定要将 SUB 位配置为 0，否则 tx 在任何时候都发送 ‘0’。

字段	说明
6 SUB	**设置 Break** 0：没有效果 1：SOUT 信号被强制进入 "0" 状态
5 SP	**奇偶校验位** 0：没有效果 1：根据 EPS 和 PEN 的状态，奇偶校验位进入已定义状态 如果 EPS = 1 & PEN = 1，设置奇偶校验位且 checked = 0. 如果 EPS = 0 & PEN = 1，设置奇偶校验位且 checked = 0.
4 EPS	**选择偶校验** 0：发送并检查奇数个 1 1：发送并检查偶数个 1
3 PEN	**使能奇偶校验** 0：不传输和检查奇偶性 1：传输和检查奇偶性

字段	说明
2 STB	**STOP 位个数** 0：始终添加一个 STOP 位 1：每个字符发送后添加两个 STOP 位，除非添加 1 个 STOP 位时，字符长度等于 5
1：0 WLS1_WLS0	**选择字长** 00： 5 位 01： 6 位 10： 7 位 11： 8 位 **注意：如果要设置为 9 位 ，请设置 WLS2 为 1**

8.6.5 控制寄存器 1(UART_LCR1)

表 8-10 UART_LCR1 寄存器

UART_LCR1 **控制寄存器 1** **Reset：0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									INVTX	INVRX	WLS2	LOOP			TXEN	RXEN
访问									RW	RW	RW	RW			RW	RW
Reset									0	0	0	0			0	0

字段	说明
7 INVTX	**确定是否反转 tx 输出，包括 idle， break， data 位，start 位， stop 位** 0：不反转 tx 输出 1：反转 tx 输出
6 INVRX	**确定是否反转 RX 输入，包括 idle， break， data 位，start 位， stop 位** 0： 不反转 rx 输入 1： 反转 rx 输入
5 WLS2	**确定 9 位数据模式是否可用** 0： 不可用 1： 可用

字段	说明
4 LOOP	**循环** 0： 供用户正常使用 1： 控制 uart 进入循环模式(可以用来测试 uart 自身)
1 TXEN	**UART 发射器使能** 0： 禁用 1： 使能
0 RXEN	**UART 接收器使能** 0： 禁用 1： 使能

8.6.6 FIFO 控制寄存器(UART_FCR)

表 8-11 UART_FCR 寄存器

UART_FCR **FIFO 控制寄存器** **Reset：0x00000000**

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

字段	说明
0 FIFOE	**使能 FIFO** 0： 禁用 RX 和 TX FIFO 1： 使能 RX 和 TX FIFO

8.6.7 硬件流使能寄存器(UART_EFR)

表 8-12 UART_EFR 寄存器

UART_EFR 硬件流使能寄存器 **Reset：0x00000000**

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

说明：AUTOCTS=1 表示使能 CTS_n 引脚的硬件流功能，因此如果 AUTOCTS=1，用户必须将 **n_CTS 引脚连接到固定电平，比如其他 MCU 或其他设备的引脚。**如果 AUTOCTS=0，用户不需要关注 CTS_n 引脚。

字段	说明
7 AUTO_CTS	**使能硬件发送流程控制** 0：禁用 1：使能
6 AUTO_RTS	**使能硬件接收流程控制** 0： 禁用 1： 使能

8.6.8 中断使能寄存器(UART_IER)

表 8-13 UART_IER 寄存器

UART_IER 中断使能寄存器 **复位值：0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称								ETXDF	EDCTS	EOEBI	ENE	EFE	EPE	ETC	ETXE	ERXNE
访问								RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset								0	0	0	0	0	0	0	0	0

字段	说明
8 ETXDF	**发送寄存器或发送 FIFO 满中断使能** 0： 禁用 1： 使能
7 EDCTS	**CTS_n 变化中断使能** 0： 禁用 1： 使能
6 EOEBI	**溢出错误或分隔符错误中断使能** 0： 禁用 1： 使能
5 ENE	**噪声错误中断使能** 0： 禁用 1： 使能
4 EFE	**帧错误中断使能** 0： 禁用 1： 使能
3 EPE	**奇偶校验错误中断使能** 0： 禁用 1： 使能
2 ETC	**发送完成中断使能** 0： 禁用 1： 使能
1 ETXE	**发送数据寄存器为空中断使能** 0： 禁用 1： 使能 **注意： fifoe=1 表示 fifo 为空** **fifoe=0 表示数据寄存器为空**
0 ERXNE	**接收数据寄存器非空中断使能** 0： 禁用 1： 使能 **注意：fifoe=1 表示 fifo 非空** **fifoe=0 表示数据寄存器非空**

8.6.9 线路状态寄存器 0(UART_LSR0)

表 8-14 UART_LSR0 寄存器

UART_LSR0 线路状态寄存器 0 **Reset：0x00000020**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称								TXDF	NE	TC	THRE	BI	FE	PE	OE	DR
访问								R	W1C	R	W	W1C	W1C	W1C	W1C	R
Reset								0	0	0	1	0	0	0	0	0

注意

SUB NE/PE/FE 错误仅对目前的字节数据而言。与此同时， OE/BI 将一直存在，直到其被清除。

字段	说明
8 TXDF	**发送数据寄存器或者发送 FIFO 满标志** 0：发送数据寄存器(fifoe=0)或者发送 FIFO (fifoe=1)不满 1：发送数据寄存器(fifoe=0)或者发送 FIFO (fifoe=1)满 **注意：该标志反映了数据和发送状态。**
7 NE	**噪声错误标志** 0：不存在噪声错误 1： 存在噪声错误 **注意：写 1 将此标志清除为 0。**
6 TC	**传输完成标志** 0： TX FIFO(fifoe=1) 或 TX 寄存器(fifoe=0) 非空，或发送端还没有完成数据移位。 1： TX FIFO(fifoe=1) 或 TX 寄存器(fifoe=0)为空，发送端完成数据移位。 **注意： 上电默认值为 0，只有 TXEN 为 1 后，TC 才起作用。将将数据写入 TX FIFO(fifoe=1)/TX 寄存器(fifoe=0) 以将该标志清除为 0。对于 LIN 功能，设置 SBRK 位也可以将此标志清除为 0。**
5 THRE	**TX 保持寄存器或 TX FIFO 空标志** 0：只要 TX FIFO 内容不为空，或 TX 保持寄存器不为空 (废弃 FIFO)，执行复位操作. 1：只要 TX FIFO 内容为空，或 TX 保持寄存器为空 (废弃 FIFO)，执行置位操作.

字段	说明
	注意： 将数据写入 TX FIFO(fifoe=1)/TX 寄存器(fifoe=0)以将此标志清除为 0。
4 BI	**分隔符错误标志** 0：无分隔符错误 1：产生分隔符错误。如果 禁用 FIFO，只要 SIN 保持在 0 状态 超过一个传输时间(START 位 + DATA 位 + PARITY + STOP 位)时，该位被置位。 当中断发生时，只有一个零字符被加载到 FIFO 或 TX 保持寄存器中。 **注意：写 1 将此标志清除为 0.**
3 FE	**帧错误标志** 0：无帧错误 1：产生帧错误。如果接收数据没有一个有效的 STOP 位， 该位被置位。 **注意：写 1 将此标志清除为 0.**
2 PE	**奇偶错误标志** 0： 无奇偶错误 1： 产生奇偶错误。没有接收到有效校验位，该位被置位。 **注意：写 1 将此标志清除为 0.**
1 OE	**溢出错误标志** 0：无接收溢出错误 1：产生接收溢出错误。如果禁用 FIFO，如果在 RX 移位寄存器的新数据覆盖先前内容之前，CPU 未读取 RX 缓冲区，则该位将置 1。如果使能 FIFO，当 RX FIFO 已满且 RX 移位寄存器变满时，会发生溢出错误。一旦发生这种情况，就会设置 OE 。然后移位寄存器中的字符被覆盖，但不会传输到 FIFO。 **注意：写 1 将此标志清除为 0.**
0 DR	**数据就绪标志** 0：接收数据未就绪 1：接收数据已经就绪。由 RX 缓冲区变满或 RX FIFO 非空进行置位(至少有一个字节被传输到 FIFO 中)。 **注意：读数据寄存器 UART_RBR/THR，或如果使能 FIFO, 读完所有 FIFO, 会自动清除此标志位为 0.**

8.6.10 线路状态寄存器 1(UART_LSR1)

表 8-15 UART_LSR1 寄存器

UART_LSR1	线路状态寄存器 1	Reset：0x000000E0

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									RTS	CTS	UART_IDLE	BRKWAK	DCTS	FBRK	SYNERR	IDLE
访问									R	R	R	R	W1C	W1C	W1C	W1C
Reset									1	1	1	0	0	0	0	0

字段	说明
7 RTS	**硬件流程状态 – RTS** 0：在硬件流程控制功能下，它表示 RX FIFO 或 RX 寄存器已满。该信号可以通知其他设备不向 MCU 发送数据。 1：在硬件流程控制功能下，它表示 RX FIFO 或 RX 寄存器未满。 **注意： RTS 和 RTS_n 引脚信号相反，它不是一个中断源。**
6 CTS	**硬件流程状态 – CTS** 0：在硬件流程控制功能下，它表示 RX FIFO 或其他设备的 RX 寄存器已满。该信号可以通知 MCU 不发送下一个数据。 1：在硬件流程控制功能下，它表示 RX FIFO 或其他设备的 RX 寄存器未满。 **注意： CTS 和 CTS_n 引脚信号相反，它不是一个中断源。**
5 UART_IDLE	**UART_IDLE** 0： UART 正在工作中 1： UART 未工作，也就是说，发射器和接收器不工作或已完成数据传输或接收
4 BRKWAK	**LIN BREAK 唤醒标志** 0： 还没接收到 break(超过 29bits) 1：已经接收到 break(超过 29bits)
3 DCTS	**CTS_n 引脚信号变化标志** 0： 未变化 1：表示 CTS_n 引脚信号从 1 变为 0 或 0 变为 1 **注意：写 1 将此标志清除为 0.**

字段	说明
2 FBRK	**LIN BREAK 发生标志** 0：在 LIN 功能中没有检测到 LIN 帧中的 break 字段 1：在 LIN 功能中检测到 LIN 帧中的 break 字段 **注意：写 1 将此标志清除为 0.**
1 SYNERR	**LIN 同步域错误标志** 0： 没有错误 1： 存在错误 **注意：写 1 将此标志清除为 0**
0 IDLE	**IDLE 标志** 0：尚未检测到空闲线路 1：检测到空闲线路 接收器已经接收到数据，该数据后面为一个至少保持一个字节数据时间的高电平。IDLE 状态标志必须先使能总线空闲检测 ILEN 才起作用。 **注意：写 1 将此标志清除为 0.**

8.6.11 采样计数器寄存器(UART_SMP_CNT)

表 8-16 UART_SMP_CNT 寄存器

UART_SMP_CNT **采样计数器寄存器** **Reset：0x00000000**

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

字段	说明
1：0 SMP_CNT	**UART 采样计数器** 00： 基于 16*baud_pulse, baud_rate = 系统时钟频率/16/{DLH, DLL} 01： 基于 8*baud_pulse, baud_rate =系统时钟频率/8/{DLH, DLL} 10： 基于 4*baud_pulse, baud_rate =系统时钟频率/4/{DLH, DLL} 11： 基于 sampe_count * baud_pulse, baud_rate =系统时钟频率/16 /{DLM, DLL}

8.6.12 保护时间寄存器(UART_GUARD)

表 8-17 UART_GUARD 寄存器

UART_GUARD 保护时间寄存器 **Reset：0x0000000F**

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

注意

添加保护时间有助于消除每个字节的累积误差， 因此，通过使用具有保护时间的小数分频器来提高波特率的准确性是很重要的。

字段	说明
4 GUARD_EN	保护间隔时间添加使能信号 0： 禁用 1： 使能
3：0 GUARD_CNT	保护间隔计数值 0~15： 0 ~ 15 位时间

8.6.13 休眠使能寄存器(UART_SLEEP_EN)

表 8-18 UART_SLEEP_EN 寄存器

UART_SLEEP_EN 休眠使能寄存器 **Reset：0x00000000**

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

字段	说明
0 SLEEP_EN	**休眠功能使能** 0：不处理睡眠模式指示信号 1：当芯片进入休眠模式时，根据软件初始设置，激活硬件流程控制。当芯片唤醒时释放硬件流程。

8.6.14 DMA 使能寄存器(UART_DMA_EN)

表 8-19 UART_DMA_EN 寄存器

UART_DMA_EN **DMA 使能寄存器** **Reset：0x00000000**

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

字段	说明
1 TX_DMA_EN	**TX_DMA 机制使能信号** 0： 在 TX 不使用 DMA 1： 当该寄存器使能时，在 TX 使用 DMA
0 RX_DMA_EN	**RX_DMA 机制使能信号** 0：在 RX 不使用 DMA 1：当该寄存器使能时，在 RX 使用 DMA

8.6.15 小数分频器寄存器(UART_DIV_FRAC)

表 8-20 UART_DIV_FRAC 字段

UART_DIV_FRAC	小数分频器寄存器	Reset：0x00000000

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

字段	说明
4：0 DIV_FRAC	**小数分频器** 如果实际的分频器为 135.65，则 DIV_FRAC 为 0.65*32 = [20.8]=21 ，并且 DIV_L=135 。

8.6.16 RS485 控制寄存器(UART_RS485CR)

表 8-21 UART_RS485CR 寄存器

UART_RS485CR	RS485 控制寄存器	复位值：0x00000000

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									RS485EN		INVPOL	DLYEN				
访问									RW		RW	RW				
Reset									0		0	0				

注意

1. **发送前的延迟对应于 UART_CNTR ，发送后的延迟可以使用 UART_GUARD。**
2. **RS485 功能使用 PIN RTS_n 作为发送或接收方向控制引脚。**

字段	说明
7 RS485EN	0：禁用 rs485 模式 1：使能 rs485 模式

字段	说明
5 INVPOL	0： 不反转 rts_n 的极性 1： 反转 rts_n 的极性
4 DLYEN	在 RS485 切换为输出状态到真正开始发送 START 位之间插入 DELAY，具体的延时时间由 RS485 延迟计数器寄存器决定。即对应于图 8-7 的 dalay1。 0：禁用延时 1：使能延时

8.6.17 RS485 延迟时间寄存器(UART_ CNTR)

表 8-22 UART_ CNTR 寄存器

UART_ CNTR **RS485 延迟时间寄存器** **复位值：0x00000000**

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

位	说明
7：0 CNTR	计数器 0~255 位时间作为 RS485 模式下的时间延迟

8.6.18 空闲中断使能寄存器(UART_IDLE)

表 8-23 UART_IDLE 寄存器

UART_IDLE **空闲中断使能寄存器** **Reset：0x00000000**

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

字段	说明
	总线空闲检测使能
7 ILEN	0: 禁用 1: 使能
	IDLE 中断使能
4 IDLEIE	0： 禁用 1： 使能

8.6.19 LIN 控制寄存器 (UART_LINCR)

表 8-24 UART_LINCR 寄存器

UART_LINCR **LIN 控制寄存器** **Reset：0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									LINEN	LBRKIE	LBRKDL	SDBRK	LABAUDEN	SYNERRIE	BRKWAKIE	
访问									RW	RW	RW	RW	RW	RW	RW	
Reset									0	0	0	0	0	0	0	

字段	说明
7 LINEN	**LIN 模式使能** 0： 禁用 1： 使能
6 LBRKIE	**LIN Break 检测中断使能** 0： 禁用 1： 使能
5 LBRKDL	**LIN Break 中断检测长度** 0： 10 位 1： 11 位
4 SDBRK	**LIN 发送 Break 使能** 0： 禁用 1： 使能发送 Break。Break 长度由寄存器 BRKLGH 决定 **注意：由软件设置，并在 break 发送完成后由 MCU 内部硬件清除。**

字段	说明
3 LABAUDEN	**LABAUDEN** 0：0x55 不用于自动波特率检测 1：0x55 用于自动波特率检测
2 SYNERRIE	**SYNERRIE** 0：禁用同步字节错误中断 1：使用同步字节错误中断
1 BRKWAKIE	**BRKWAKIE** 0：禁用 break 唤醒中断 1：使用 break 唤醒中断

8.6.20 LIN 同步间隔段控制寄存器(UART_BRKLGH)

表 8-25 UART_BRKLGH 寄存器

UART_BRKLH **LIN 同步间隔段控制寄存器** **Reset：0x00000000**

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

字段	说明
3：0 BRKLGH	**BRKLGH** 0000：13bits 同步间隔段 0001：14bits 同步间隔段 …… 1111： 28bits 同步间隔段

9 模数转换器（ADC）

9.1 简介

ADC 是一种 12 位逐次逼近型模拟数字转换器，拥有 12 路外部通道和 2 路内部通道，支持单次、连续、扫描或间断转换多种模式。模拟监控器特性允许应用程序监测输入电压是否超出设定的电压范围。

9.2 特性

- 12 位分辨率
- 通道输入电压范围：AVSS < Vin < AVDD
- 最大转换速率：1Msps
- 14 路通道：12 路外部通道， 1 路内部温度传感器（T-Sensor），1 路内部带隙基准电压（Bandgap），每路通道可单独配置采样时间
- 转换序列分为 规则组（regular group）和注入组（injection group）
  - 规则组：最多可配置 12 个通道
  - 注入组：最多可配置 4 个通道
- 8 种操作模式 (方便起见，称为 mode x，x=1~8)
  - 规则组单通道单次转换(mode1)
  - 规则组单通道连续转换(mode2)
  - 规则组扫描+注入组扫描模式多通道单次转换(mode3 注入组扫描模式)
  - 规则组扫描+注入组间隔模式多通道单次转换(mode3 注入组间隔模式)
  - 规则组扫描+自动触发注入组扫描模式多通道单次转换(mode4)
  - 规则组扫描+注入组扫描模式多通道连续转换(mode5 注入组扫描模式)
  - 规则组扫描+注入组间隔模式多通道连续转换(mode5 注入组间隔模式)
  - 规则组扫描+自动触发注入组扫描模式多通道连续转换(mode6)
  - 规则组子组扫描模式转换(mode7)
  - 注入组子组扫描模式转换(mode8)
- 通过内部软件触发或外部硬件触发启动 ADC
- 模拟监控器功能：
  - 配置为单个或所有通道电压检查

- 监控通道电压是否低于低阈值或高于高阈值

- 中断：
  - 规则或注入组转换结束(EOC，End Of Conversion)
  - 注入组转换结束(IEOC)
  - 模拟监控器事件(AMO)
- DMA 访问，仅用于规则组通道

9.3 结构框图

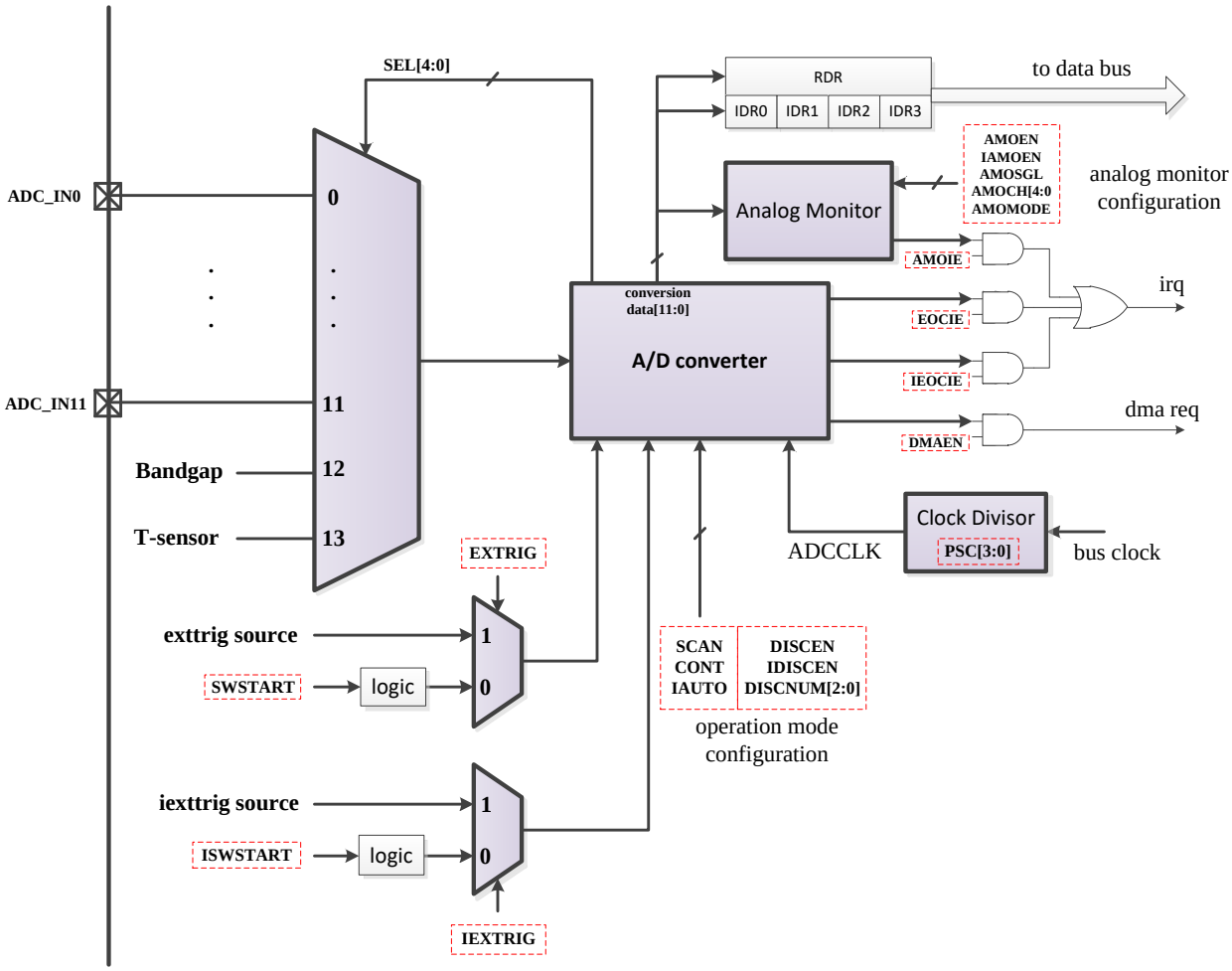

图 9-1 ADC 结构框图

9.4 功能描述

ADC 主要由 ADC 转换器单元（converter unit），输入通道选择器（input channel selector），时钟分频器（clock divisor）和模拟监控器（analog monitor）等组成。如图 9-1 所示，A/D 转换器单元工作在 ADC 时钟，简称 ADCCLK，其他电路单元工作在总线时钟。

下面介绍一个典型的操作流程。

ADC 首先上电，然后可以通过内部 SWSTART 或外部触发源触发 ADC，该触发来源于其它模块。触发后 ADC 转换器单元开始工作，并将选择信号发送至输入通道选择器，根据规则或注入组通道序列逐个选择所需的通道。在一个通道完成转换后，转换结果将根据当前转换通道所属的组存储到 RDR 或 IDRx 中，并且产生相应的 EOC 或 IEOC 标志置位。模拟监控器工作时，如果发生相应的事件则会出现相关的状态标志。需要指出的是，不同的操作模式存在一些差异，详细信息将在后面进行说明。

9.4.1 上电时序

在开始所有功能之前， ADC 首先上电，然后有效的触发器可以启动 ADC 以基于配置的模式工作。上电时序如下图所示。

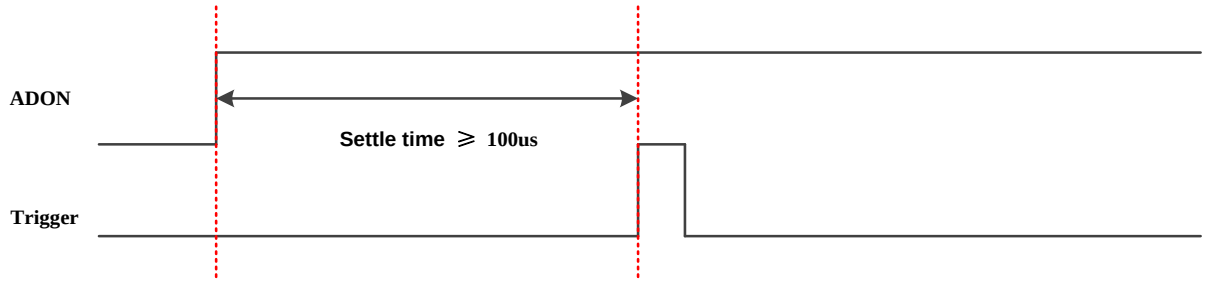

图 9-2 ADC 上电时序

如图 9-2 所示，将 ADC_CTRL1 [ADON]位置为 1 以控制上电过程。在 ADON 置位后，A/D 转换器单元上电等待时间不应低于 100μs。

9.4.2 工作模式

根据实际应用可以灵活使用不同的模式，上电和有效触发后 ADC 工作于以下模式之一。

表 9-1 工作模式配置表

工作模式	MODE_BITS	触发源	转换序列
mode1	5'b0000x	规则触发	规则组单通道单次转换
mode2	5'b0100x	规则触发	规则组单通道连续转换
mode3 (注入组扫描模式)	5'b10000 (INTERVAL=0)	规则/注入触发	规则组扫描+注入组扫描模式 多通道单次转换

工作模式	MODE_BITS	触发源	转换序列
mode3 (注入组间隔模式)	5'b10000 (INTERVAL=1)	规则/注入触发	规则组扫描+注入组间隔模式 多通道单次转换
mode4	5'b10001	规则触发+自动注入触发	规则组扫描+注入组扫描模式 多通道单次转换
mode5 (注入组扫描模式)	5'b11000 (INTERVAL=0)	规则/注入触发	规则组扫描+注入组扫描模式 多通道连续转换
mode5 (注入组间隔模式)	5'b11000 (INTERVAL=1)	规则/注入触发	规则组扫描+注入组间隔模式 多通道连续转换
mode6	5'b11001	规则触发+自动注入触发	规则组扫描+注入组扫描模式 多通道连续转换
mode7	5'b1x10x	规则触发	规则组子组扫描模式转换
mode8	5'b1x01x	注入触发	注入组子组扫描模式转换

注：MODE_BITS = {SCAN, CONT, DISCEN, IDISEN, IAUTO}

在描述每个模式操作流程之前，有必要引入一些术语，例如规则组，注入组等。对于 ADC 输入通道，它们被称为 ch0~ch13，其中 ch0~ch11 是外部输入通道，ch12 对应于内部带隙参考电压通道，ch13 对应于温度传感器通道。

规则组是按顺序转换的输入通道。基于 ADC_RSQR0，ADC_RSQR1 和 ADC_RSQR2 寄存器，规则组由 RSQ0 至 RSQ11 的最多 12 个通道组成。

例如，如果 RSQ0~RSQ11 分别设置为 9,8,12,1,5,4,7,3,13,2,0,0 则将规则组按图 9-3 所示进行排列。

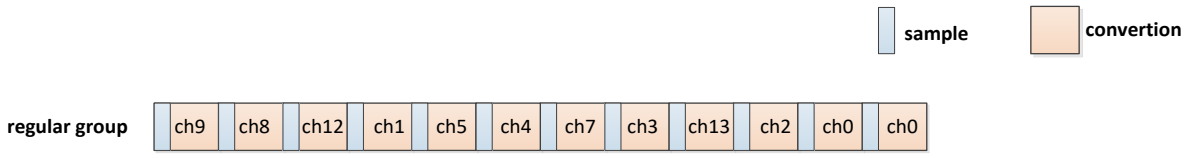

图 9-3 规则组序列

如果 RSQL 设置为 8(Length=9)，则最后 3 个通道将无效且无法转换。因此，有效的规则组序列如图 9-4 所示。

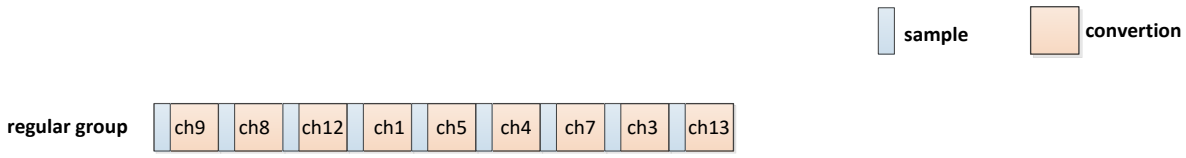

图 9-4 有效规则组序列

以同样的方式，注入组是按顺序转换的输入通道。基于 ADC_ISQR 寄存器，注入组由最多 4 个通道组成，顺序依次为 ISQ0 至 ISQ3。

例如，如果 ISQ0~ISQ3 分别设置为 12,7,13,2，则将注入组按图 9-5 所示进行排列。

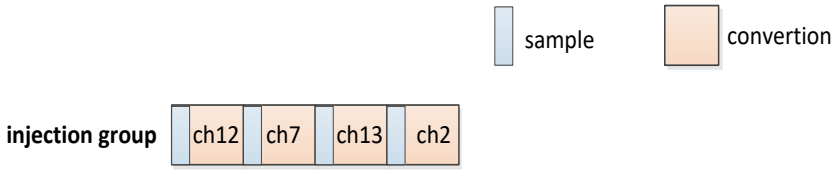

图 9-5 注入组序列

如果 ISQL 设置为 2(Length=3)，则最后 1 个通道将无效并且不会被转换。因此，有效注入组序列如图 9-6 所示。

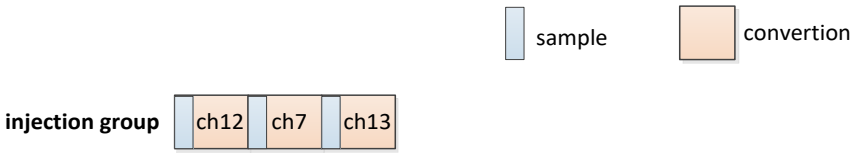

图 9-6 有效注入组序列

显然，规则组触发和注入组触发是开始转换规则组和注入组序列的相应信号。该触发源自 ADC 框图中所示的内部 SWSTART 或外部触发源。当 ADC 处于常规组通道转换过程中时，规则触发无效。基于此基本介绍，每种模式的详细描述如下。

9.4.2.1 Mode 1

此模式仅转换规则组中的第一个通道，无论 RSQL 为任意值。模式按表 9-1 进行配置后，有效触发可使 ADC 工作在此模式。

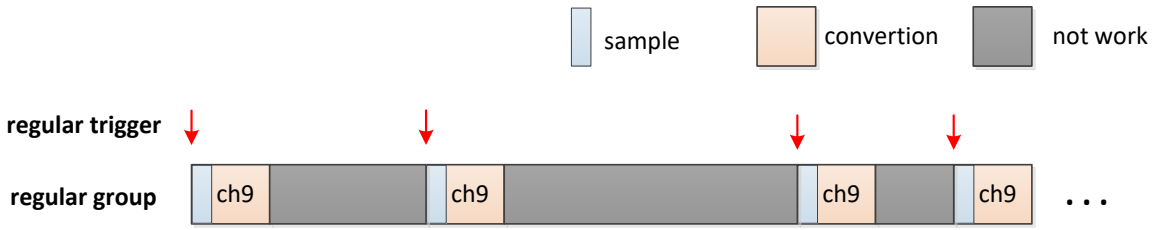

图 9-7 Mode 1 工作流程

如图 9-7 所示，规则组中的第一个通道在有效的规则触发后转换一次。然后 ADC 进入空闲状态，直到下一次有效规则触发带来的下一次转换。

9.4.2.2 Mode 2

此模式连续转换规则组中的第一个通道，无论 RSQL 为任意值。模式按表 9-1 进行配置后，有效触发可以使 ADC 在此模式下工作。

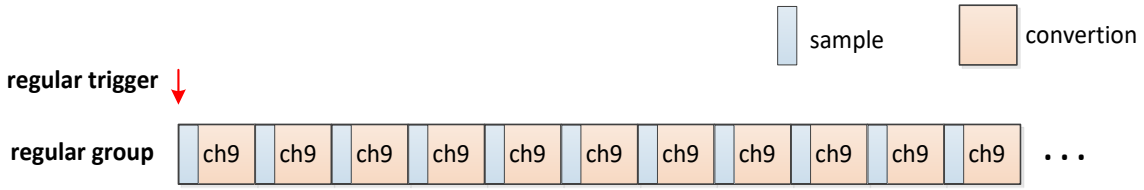

图 9-8 Mode 2 工作流程

如图 9-8 所示，在有效的规则触发后，规则组第一个通道将不断转换，除非，断电/复位或者更改 ADC 工作模式。

9.4.2.3 Mode 3

9.4.2.3.1 interval bit=0，注入组为扫描模式

此模式转换规则组通道和注入组通道。有效的规则和注入组通道长度分别由 RSQL 和 ISQL 决定。使用表 9-1 中的模式配置，有效触发可使 ADC 在此模式下工作。例如，RSQL 设置为 6(Length=7)，ISQL 设置为 2(Length=3)，一个典型操作如图 9-9 所示。第一笔规则触发转换规则组中的 7 个通道。当 ADC 转换规则组中的 ch1 时，此时产生注入触发，在 ch1 转换结束后将切换至转换 3 个注入组通道，在所有注入组通道转换完成后，自动切换回规则组通道 ch5 继续转换，完成有效的规则通道转换后，ADC 将运行至空闲状态，直至下一次触发到来。

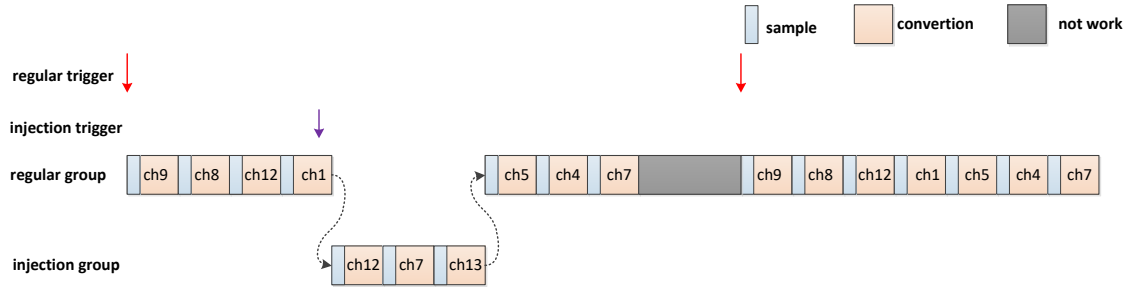

图 9-9 Mode 3 注入组扫描模式工作流程

如果在 ADC 空闲时发生注入触发，ADC 将完成有效注入组通道的转换，如图 9-10 所示。

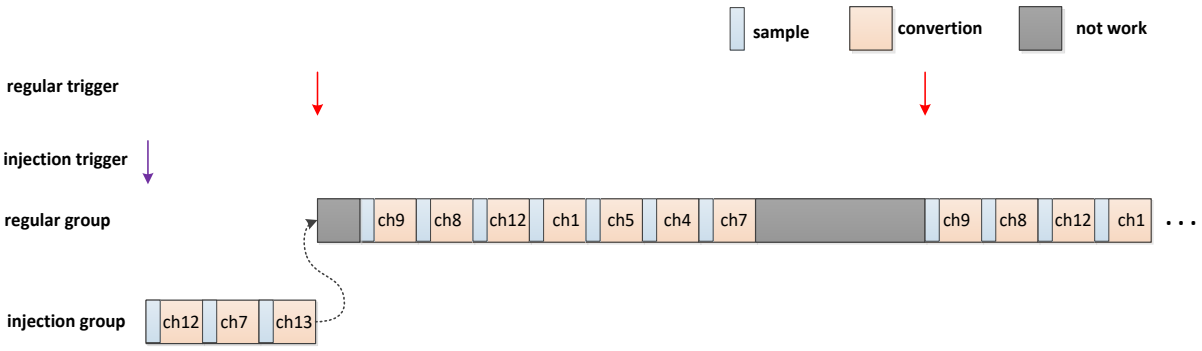

图 9-10 Mode 3 在 ADC 空闲状态下具有注入触发的工作流程

9.4.2.3.2 interval bit=1，注入组为间隔模式

与图 9-9 的区别在于，产生一次注入触发只会转换注入组序列的一个通道，下一次再发生注入触发，注入组序列的下一通道进行转换。

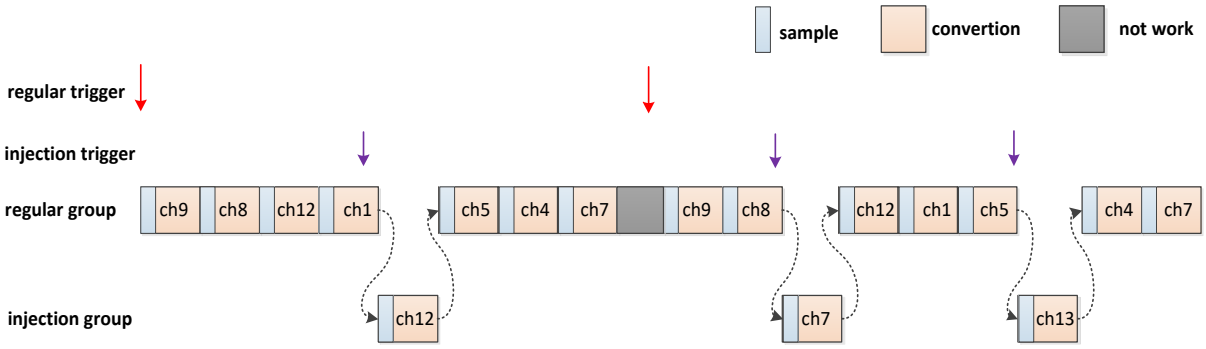

图 9-11 Mode 3 注入组间隔模式工作流程

9.4.2.4 Mode 4

此模式触发后将自动按照规则组通道先转换，后转换注入组通道。有效的规则组通道和注入组通道分别由 RSQL 和 ISQL 决定。使用表 9-1 中的模式配置，有效触发可使 ADC 在此模式下工作。例如，RSQL 设置为 6，ISQL 设置为 2。典型操作如图 9-12 所示。规则触发器开始转换前 7 个规则组通道，然后自动转换 3 个注入组通道。在总共 10 个通道均完成完全转换后，ADC 将运行至空闲状态，直到下一个有效的规则触发。

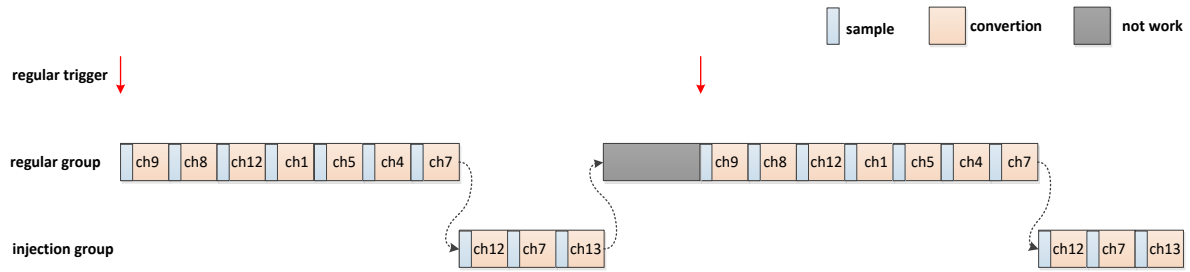

图 9-12 Mode 4 工作流程

9.4.2.5 Mode 5

9.4.2.5.1 interval bit=0，注入组为扫描模式

与 Mode3 区别在于此模式为连续转换。有效的规则和注入组通道长度分别由 RSQL 和 ISQL 决定，但与 Mode 3 不同，该模式下使用连续转换。使用表 9-1 中的模式配置，有效触发可使 ADC 在此模式下工作。此模式的一个关键特性是单个规则触发可以使 ADC 始终工作，除了掉电、复位或模式更改。例如，RSQL 设置为 6，ISQL 设置为 2。一个典型操作如图 9-13 所示。在规则触发后，ADC 按规则组通道顺序工作，如果发生注入触发，则在注入组通道上工作。

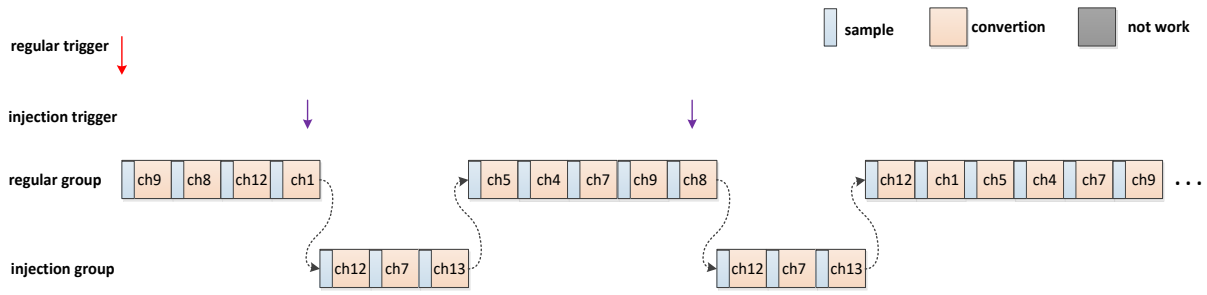

图 9-13 Mode 5 注入组扫描模式工作流程

特别地，如果在ADC空闲时发生注入触发，ADC将首先完成有效注入组通道的转换，如图 9-14所示。

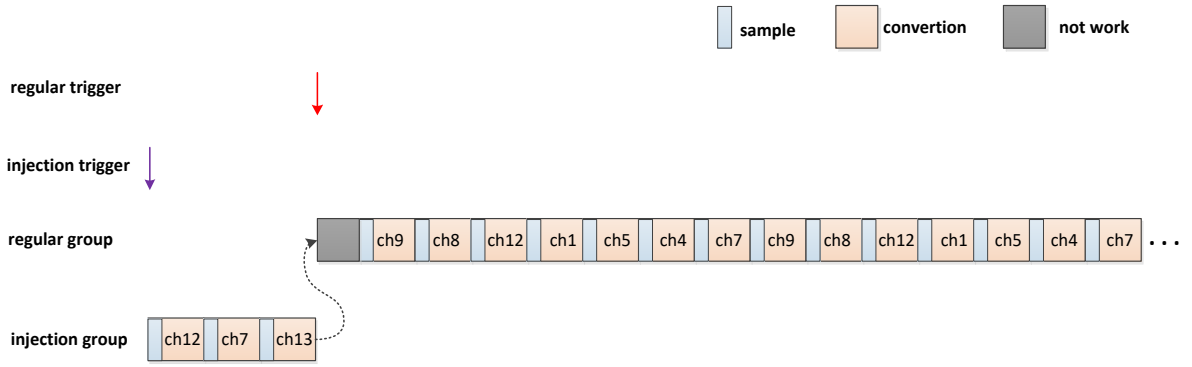

图 9-14 Mode 5 在 ADC 空闲状态下具有注入触发的工作流程

9.4.2.5.2 interval bit=1，注入组为间隔模式

与图 9-13 的区别在于，产生一次注入触发只会转换注入组序列的一个通道，下一次再发生注入触发，注入组序列的下一通道进行转换。

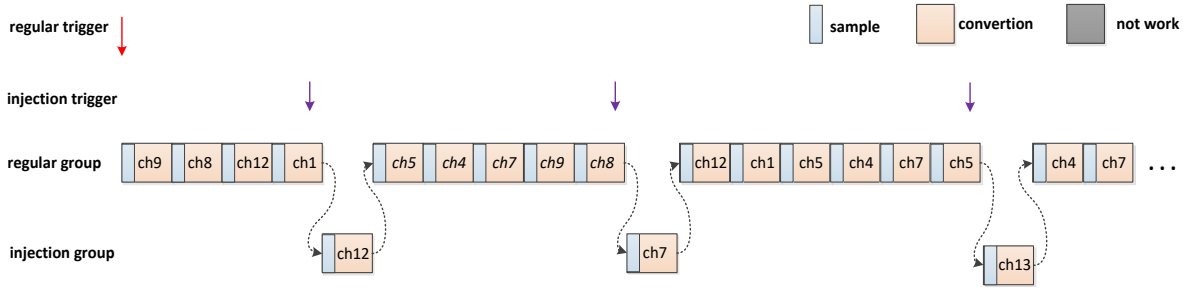

图 9-15 Mode 5 注入组间隔模式工作流程

9.4.2.6 Mode 6

与 Mode4 区别在于此模式为连续转换。有效的规则和注入组通道长度分别由 RSQL 和 ISQL 决定，但与 Mode 4 不同，该模式下使用连续转换。使用表 9-1 中的模式配置，有效的规则触发可以使 ADC 在此模式下工作。此模式的一个关键特性是单个规则触发可以使 ADC 始终工作，除了掉电，复位或模式更改。例如，RSQL 设置为 6，ISQL 设置为 2，操作流程如图 9-16 所示。ADC 在规则组通道上按顺序工作，然后在规则组转换完成后转换注入组通道。

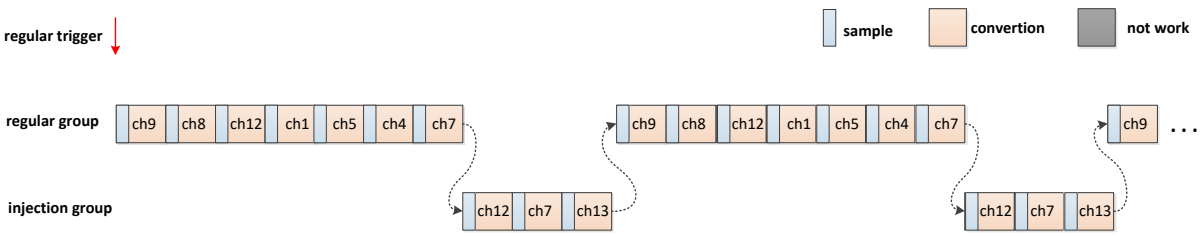

图 9-16 Mode 6 操作流程

9.4.2.7 Mode 7

此模式仅转换规则组通道。有效的规则组通道由 RSQL 决定。使用表 9-1 中的模式配置，ADC 可以在此模式下工作。依据 DISCNUM 将有效的规则通道分成若干子组。

例如，RSQL 设置为 6，DISCNUM 设置为 1。

第一次规则触发： ch9, ch8；

第二次规则触发： ch12, ch1；

第三次规则触发： ch5, ch4；

第四次规则触发： ch7，产生 EOC 标志；

因此，实际的转换流程如图 9-17 所示。

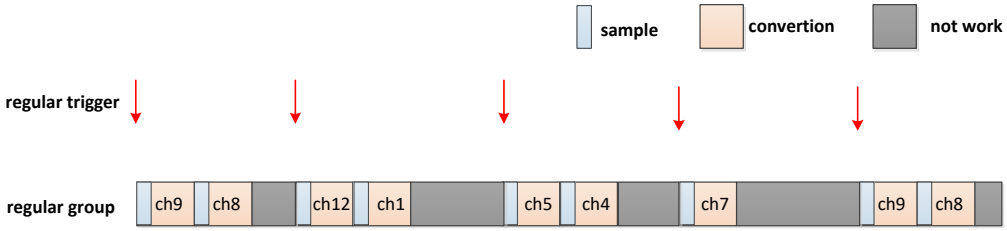

图 9-17 Mode 7 操作流程

9.4.2.8 Mode 8

该模式仅转换注入组通道。有效的注入通道组通道由 ISQL 决定。使用表 9-1 中的模式配置，ADC 可以在此模式下工作。每次触发只转换一个通道。例如，ISQL 设置为 2。

第一次注入触发：ch12；

第二次注入触发：ch7；

第三次注入触发：ch13 ，IEOC 置位；

第四次注入触发：ch12；

…

因此，实际的转换流程如图 9-18 所示。

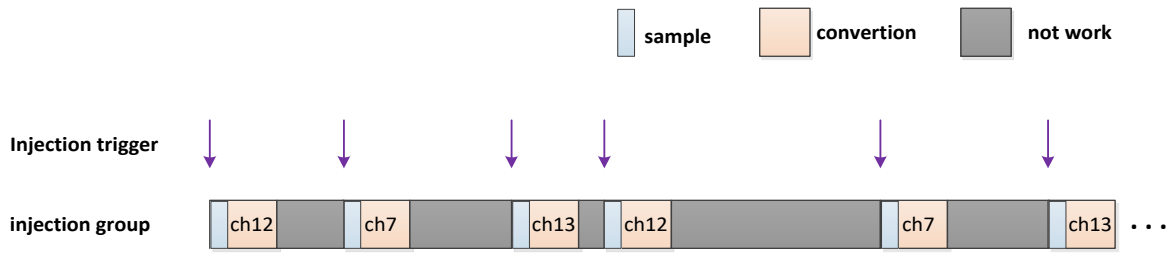

图 9-18 Mode 8 操作流程

9.4.3 触发方式

根据不同的触发方式可以组合出以下 7 种情景。

表 9-2 不同触发方式下的响应行为

触发方式	响应行为
触发规则组	规则组转换
触发注入组	注入组转换
在规则组转换期间产生规则触发	规则组持续转换，第二次触发事件不响应
在规则组转换期间产生注入触发	等待规则组的当前通道转换完才切换到注入组，注入组转换完后切换到原来的规则组继续执行（规则组序列未转换完的情况下）
在注入组转换期间产生注入触发	注入组持续转换，第二次注入触发事件不响应
在注入组转换期间产生规则触发	在注入转换期间产生一规则事件，注入转换不会被打断，但是规则序列将在注入序列结束后被执行
规则触发和注入触发同一时刻产生	注入组先转换，完成后再转换规则组

9.4.4 模拟监控器

模拟监控器支持电平触发监控事件和边沿触发监控事件模式，当监控通道的电压值超出阈值范围时，模拟监控器将产生监控事件。模拟监控器需要监测的通道可通过 AMOEN，IAMOEN，AMOSGL 和 AMOCH 位来配置。

阈值通过 AMOHR 和 AMOLR 寄存器进行配置。这两个寄存器各个域（AMOHT/ AMOLT/ AMOHO/ AMOLO）配置的单位是 LSB，即一个 ADC code 对应的电压。注意，阈值比较时使用的是 ADC 原始数据，设置数据对齐或注入组偏移不影响比较结果。

表 9-3 模拟监控通道配置

模拟监控通道	{AMOEN,IAMOEN,AMOSGL}	工作模式	注释
无	3'b00x	所有模式	-
所有注入组通道	3'b010	mode3/4/5/6/8	-
所有规则组通道	3'b100	除了 mode8	-
所有通道	3'b110	所有模式	-
单注入组通道	3'b011	mode3/4/5/6/8	转换序列必须包含由 AMOCH [4：0]指定的注入通道
单规则组通道	3'b101	mode1 ~ 7	转换序列必须包含由 AMOCH [4：0] 指定的规则通道
单规则组或注入组通道	3'b111	所有通道	转换序列必须包含由 AMOCH [4：0] 指定的规则或注入通道

9.4.4.1 电平触发模式

设置 AMOMODE=0，模拟监控器工作于电平触发模式。

如果被监控通道的电压大于高阈值 AMOHT 或小于低阈值 AMOLT，则模拟监控器将 AMO 标志设置为 1，如果 AMOIE 被配置为 1，则产生中断。

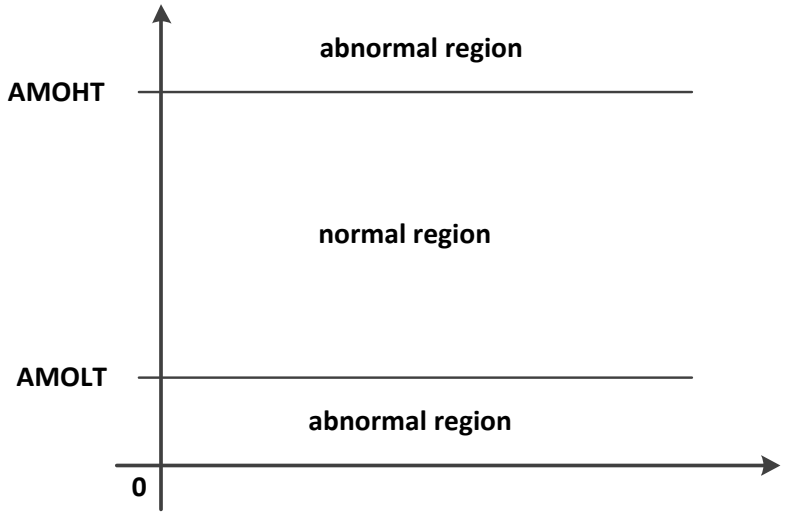

图 9-19 电平触发模式下监控区域

9.4.4.2 边沿触发模式

设置 AMOMODE=1，模拟监控器工作于边沿触发模式。

当监控通道电压从正常区域到异常区域（低于低阈值或高于高阈值）时则产生一次监控异常事件，模拟监控器将 AAMO 标志设置为 1，如果 AMOIE 被配置为 1，则产生监控事件中断。

当监控通道电压从异常区域到正常区域，则产生一次监控恢复事件，模拟监控器将 NAMO 标志设置为 1，如果 AMOIE 被配置为 1，则产生监控事件中断。边界值 = [高阈值-高偏移值，低阈值+低偏移值]，即[AMOHT-AMOHO, AMOLT+AMOLO]。

说明

边沿触发模式只支持在监控单通道时使用，监控多通道无法区分不同通道触发的异常和恢复中断。

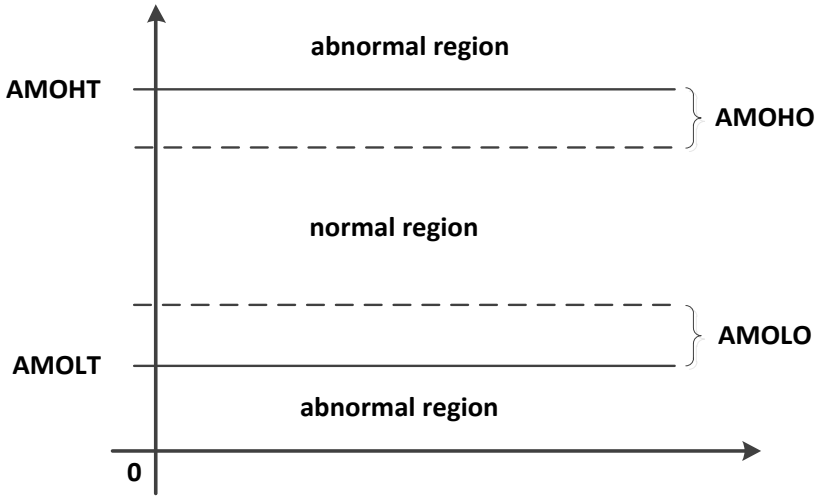

图 9-20 边沿触发模式下监控区域

9.4.5 状态标志

ADC 有三种转换状态标志位：EOC、 IEOC 和 AMO（边沿触发模式使用 AAMO 和 NAMO）。EOC 标志表示规则组或注入组通道的转换结束。IEOC 标志表示所有注入组通道都转换完成。AMO 标志标识是否发生模拟监控器事件。模拟监控器事件表示当前的转换结果是否高于当前配置的高阈值或低于当前配置的低阈值。对于不同模式，根据生成 EOC 和 IEOC 标志的不同时刻可以分为三种情形， AMO 标志在所有模式下产生的时刻相同。假设 ch5 小于 AMOLT，ch7 大于 AMOHT，并且模拟监控器被配置为检查所有通道，如包括规则组通道和注入组通道等，以下描述基于以上假设。

【情形 1】对于 mode1 至 mode6，同时生成 EOC 和 IEOC 标志。对于所有 8 种模式，同时生成 AMO 标志。有关三个标志的详细信息（基于模式 6），如图 9-21 所示。

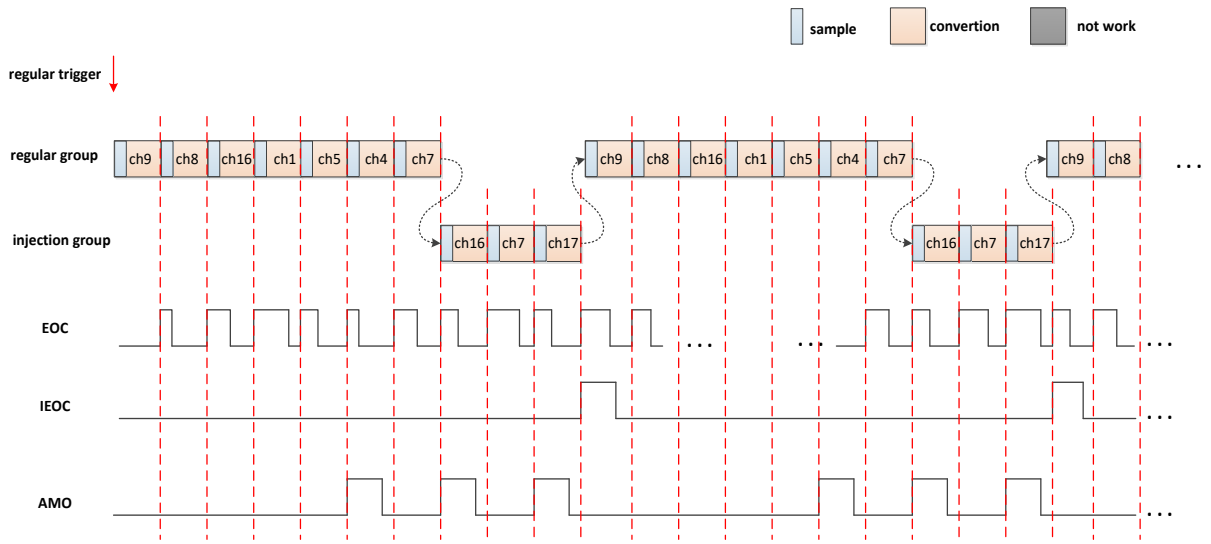

图 9-21 情形 1 下的三个标志行为

当单个规则组和注入组通道的通道转换完成时，EOC 被设置为 1 。通过向其写入 0 或读取 ADC_RDR 寄存器来清除 EOC。对于 IEOC 标志，当所有有效注入组通道已完全转换完成时设置为 1，将 0 写入 IEOC 位进行清除。当通道电压超出模拟监控器正常区域（例如，ch5，ch7）时，将 AMO 标志设置为 1，写 0 清除。

【情形 2】在 mode7 生成标志的时刻与 mode 1~mode 6 之间是不相同的。有关三个标志的详细信息如图 9-22 所示。

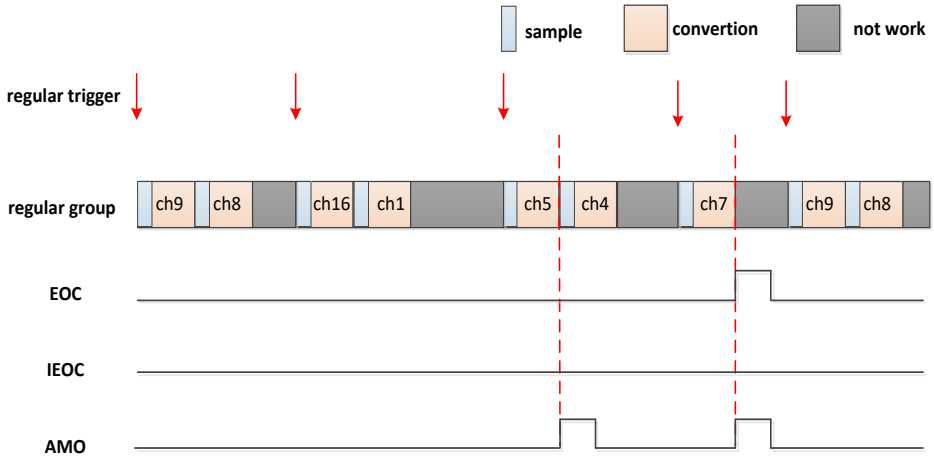

图 9-22 情形 2 下的三个标志行为

当所有规则组通道转换完成时，EOC 才被设置为 1。对于 IEOC 标志，此模式始终为 0。

【情形 3】在 mode8 下的状态标志生成时刻如图 9-23 所示。

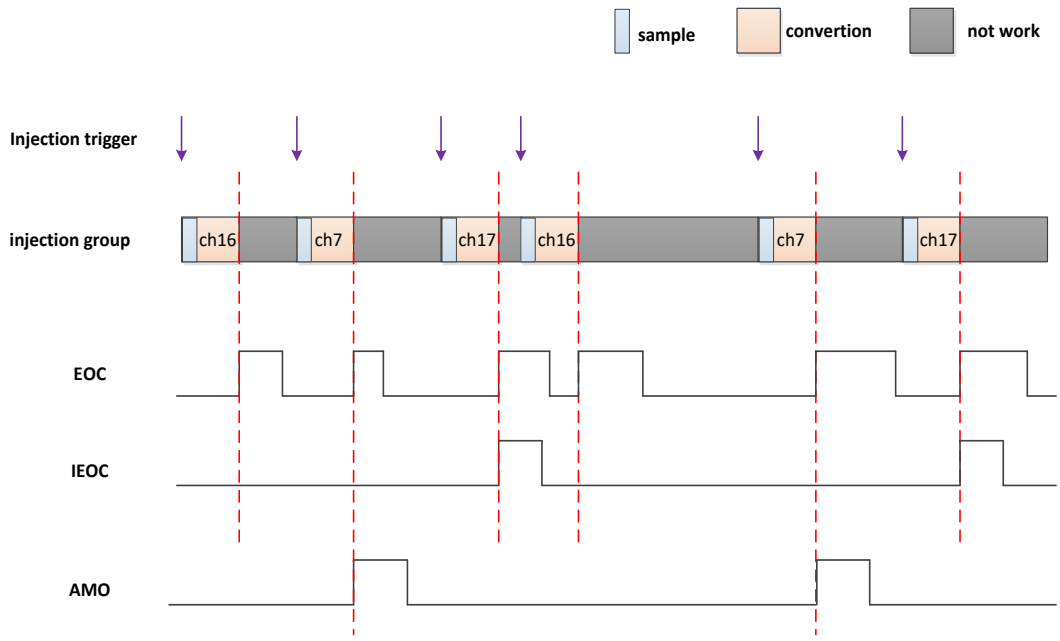

图 9-23 情形 3 下的三个标志行为

当每个注入组通道转换完成时 EOC 置 1，所有注入组通道转换完成后，EOC 和 IEOC 被置 1。

9.4.6 校准

校准功能可以使得 ADC 转换结果更准确，减少增益误差(Gain Error)和偏移误差(Offset Error)，提高精度。

在芯片生产时需要经过机台测试，将测量计算的 GE & OE 系数存储到芯片特定区域。当使能校准功能时 ADC_CTRL1[CALEN]，数据寄存器获取的是使用了 GE & OE 系数进行校准后的结果。

9.4.7 采样转换时间

ADC 需要使用若干个 ADC_CLK 周期对输入电压采样，即对 ADC 内部电路进行充电，使其达到外部输入信号的电平，完成采样之后才能进行模拟到数字的转换。采样周期个数可通过 ADC_SPT 寄存器中的 SPT[2:0]位配置。每个通道可以分别用不同的时间进行采样。

总转换时间公式： (SPT+ 12)*ADC 周期+5 个 APB 周期

例：当 APB=24MHz，ADCCLK=24MHz，SPT=7 ADCCLK，总转换时间= (7+12)/24+(5/24)= 1μs。

9.4.8 温度传感器

温度传感器可以用来测量器件周围的温度(T_A)，与 ADC 内部直接连接，通过 ADC 把传感器输出电压转换成数字量。

内部温度传感器更适合于检测温度的变化，而不是测量绝对的温度。如果需要测量精确的温度，应该使用一个外置的温度传感器。

温度计算公式：温度(℃) = {(V_{TEMP25} - V_{SENSE}) / Slope} + 25

V_{TEMP25}：25℃时的电压数值

V_{SENSE}：当前温度电压数值

Slope：温度传感器的平均斜率(单位为 mV/℃)

具体可参考数据手册的电气特性章节中 V_{TEMP25} 和 Slope 的实际值。

9.4.9 DMA 访问

由于规则组通道只有一个数据寄存器，因此建议使用 DMA 功能，以避免在有多个规则组通道进行转换时，丢失转换结果。DMA 功能专用于规则组通道。

只有规则组通道转换结束标志才会产生 DMA 请求。只有产生了 DMA 请求，DMA 才会将转换数据从 ADC_RDR 搬运到用户指定的目标位置。

9.4.10 低功耗模式

ADC 提供两种功耗模式：一种是正常模式，另一种是低功耗模式。ADC 时钟在低功耗模式下工作频率较低，以降低功耗。当 MCU 进入 stop 模式时，可以使 ADC 进入低功耗模式。低功耗模式下的 ADC 模拟监控器事件可将 MCU 从停止模式唤醒至正常模式。

ADC 功耗模式切换流程如图 9-24 所示。

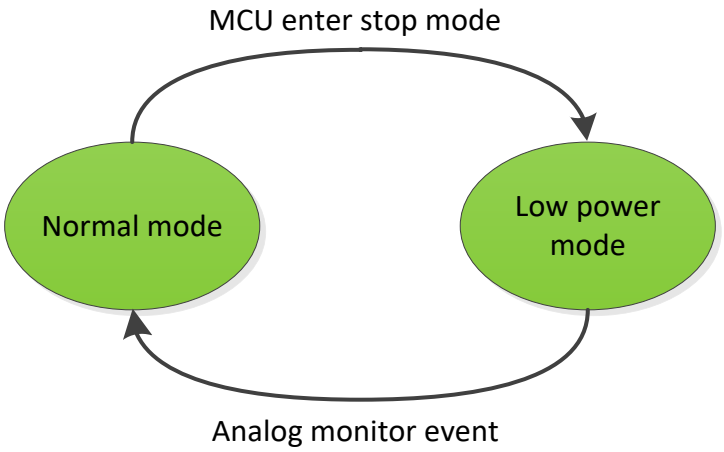

图 9-24 ADC 功耗模式切换流程图

9.5 寄存器定义

表 9-4 ADC 寄存器映射

ADC0 基地址 = 0x40003000

地址	名称	宽度	描述
ADCx 基地址+0x0	ADC_STR	32	状态寄存器
ADCx 基地址+0x4	ADC_CTRL0	32	控制寄存器 0
ADCx 基地址+0x8	ADC_CTRL1	32	控制寄存器 1
ADCx 基地址+0xC	ADC_SPT0	32	采样时间寄存器 0
ADCx 基地址+0x10	ADC_SPT1	32	采样时间寄存器 1
ADCx 基地址+0x14	ADC_IOFR0	32	注入组偏移寄存器 0
ADCx 基地址+0x18	ADC_IOFR1	32	注入组偏移寄存器 1
ADCx 基地址+0x1C	ADC_IOFR2	32	注入组偏移寄存器 2
ADCx 基地址+0x20	ADC_IOFR3	32	注入组偏移寄存器 3
ADCx 基地址+0x24	ADC_AMOHR	32	模拟监控器高阈值寄存器
ADCx 基地址+0x28	ADC_AMOLR	32	模拟监控器低阈值寄存器
ADCx 基地址+0x2C	ADC_RSQR0	32	规则组序列配置寄存器 0
ADCx 基地址+0x30	ADC_RSQR1	32	规则组序列配置寄存器 1
ADCx 基地址+0x34	ADC_RSQR2	32	规则组序列配置寄存器 2
ADCx 基地址+0x38	ADC_ISQR	32	注入组序列配置寄存器
ADCx 基地址+0x3C	ADC_IDR0	32	注入组数据寄存器 0
ADCx 基地址+0x40	ADC_IDR1	32	注入组数据寄存器 1
ADCx 基地址+0x44	ADC_IDR2	32	注入组数据寄存器 2
ADCx 基地址+0x48	ADC_IDR3	32	注入组数据寄存器 3
ADCx 基地址+0x4C	ADC_RDR	32	规则组数据寄存器

9.5.1 状态寄存器(ADC_STR)

表 9-5 ADC_STR 寄存器

ADC_STR **ADC 状态寄存器** **Reset：0x00000010**

位	31~7	6	5	4	3	2	1	0
名称		AAMO	NAMO	IDLE		IEOC	EOC	AMO
访问		W0C	W0C	RO		W0C	W0C	W0C
Reset		0	0	1		0	0	0

字段	说明
6 AAMO	**模拟监控器事件发生(边沿触发模式使用)** 0：没有异常事件 1：发生异常事件，写 0 清除

字段	说明
5 NAMO	**模拟监控恢复事件发生标志(边沿触发模式使用)** 0：没有恢复事件 1：发生恢复事件，写 0 清除
4 IDLE	**ADC 空闲状态标志** 0： ADC 处于非空闲状态 1： ADC 处于空闲状态
2 IEOC	**注入组转换完成标志** 0： 注入组转换未完成 1： 注入组转换完成，写 0 清除
1 EOC	**规则组转换完成标志** 0：规则组转换未完成 1：规则组转换完成，写 0 或读取 ADC_RDR 清除
0 AMO	**模拟监控异常事件发生标志(电平触发模式使用)** 0：没有异常事件 1：发生异常事件，写 0 清除

9.5.2 控制寄存器 0(ADC_CTRL0)

表 9-6 ADC_CTRL0 寄存器

ADC_CTRL0 **ADC 控制寄存器 0** **Reset：0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	SWSTART	ISWSTART						INTERVAL	AMOMODE	ALIGN	IEXTTRIG	EXTTRIG	DMAEN	AMOIE	IEOCIE	EOCIE
访问	RW	RW						RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0						0	0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	SCAN	CONT	DISCEN	IDISCEN	IAUTO	DISCNUM[2: 0]			AMOEN	IAMOEN	AMOSGL	AMOCH[4: 0]				
访问	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

字段	说明
31 SWSTART	**规则通道软件触发** 写 1 触发，读为 0
30 ISWSTART	**注入通道软件触发** 写 1 触发，读为 0
24 INTERVAL	**间隔模式(仅在 Mode3/5 使用)** 0：注入组为扫描模式 1：注入组为间隔模式
23 AMOMODE	**模拟监控器触发模式** 0：电平触发模式 1：边沿触发模式
22 ALIGN	**数据对齐** 0：右对齐 1：左对齐
21 IEXTTRIG	**注入组触发源选择** 0：内部（软件触发） 1：外部
20 EXTTRIG	**规则组触发源选择** 0：内部（软件触发） 1：外部
19 DMAEN	**DMA 功能使能** 0：禁用 1：使能
18：16 AMOIE,IEOCIE,EOCIE	**中断功能使能** 0：禁用 1：使能
15：11 Modes control bits	**ADC 工作模式** 详细配置参考表 9-1
10：8 DISCNUM	**通道的不连续转换长度** 0 ~ 7： mode 7 确定子组长度
7： 5 Analog monitor control bits	**模拟监控通道配置** 详细配置参考表 9-3
4：0 AMOCH	**模拟监控通道**

字段	说明
	当模拟监控器配置为仅检测单个通道时，指定被监测的通道

9.5.3 控制寄存器 1(ADC_CTRL1)

表 9-7 ADC_CTRL1 寄存器

ADC_CTRL1 **ADC 控制寄存器 1** **Reset：0x0000F002**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	PSC[3：0]														CALEN	ADON
访问	RW	RW	RW	RW											RW	RW
Reset	1	1	1	1											1	0

字段	说明
15：12 PSC	**总线时钟预分频为 ADC 工作时钟** 0 ~ 15 ： 1 ~ 16 分频器
1 CALEN	**校准使能** 0：禁用校准 1：使能校准 注意，该功能一般需要保持使能，以保证 ADC 结果的准确性。
0 ADON	**ADC 上电** 写 0： ADC 断电，复位 ADC (配置寄存器不会被重置) 写 1： ADC 上电

9.5.4 采样时间寄存器 0(ADC_SPT0)

表 9-8 ADC_SPT0 寄存器

ADC_SPT0 **ADC 采样时间寄存器 0** **Reset：0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称					SPT13[2：0]			SPT12[2：0]			SPT11[2：0]			SPT10[2：0]		
访问					RW											
Reset					0											

字段	说明
11：0 SPTx(x=10 ~ 13)	**通道采样时间** SPTx 的编号与 ADC_RSQRx 中的通道编号相同： 0~11：外部通道 12：Bandgap 电压 13：温度传感器电压 SPTx 编码含义如下： 000：9 ADCCLK 001：7 ADCCLK 010：15 ADCCLK 011：33 ADCCLK 100：64 ADCCLK 101：140 ADCCLK 110：215 ADCCLK 111：5 ADCCLK

9.5.5 采样时间寄存器 1(ADC_SPT1)

表 9-9 ADC_SPT1 寄存器

ADC_SPT1 **ADC 采样时间寄存器 1** **Reset：0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称			SPT9[2：0]			SPT8[2：0]			SPT7[2：0]			SPT6[2：0]			SPT5[2:0]	
访问			RW													
Reset			0													
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称		SPT4[2：0]			SPT3[2：0]			SPT2[2：0]			SPT1[2：0]			SPT0[2：0]		

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

字段	说明
29：0 SPTx (x=0 ~ 9)	**通道采样时间** SPTx 的编号与 ADC_RSQRx 中的通道编号相同： 0~11：外部通道 12：Bandgap 电压 13：温度传感器电压 SPTx 编码含义如下： 000：9 ADCCLK 001：7 ADCCLK 010：15 ADCCLK 011：33 ADCCLK 100：64 ADCCLK 101：140 ADCCLK 110：215 ADCCLK 111：5 ADCCLK

9.5.6 注入组偏移寄存器(ADC_IOFRx)

表 9-10 ADC_IOFRx（x= 0 ~ 3）寄存器

ADC_IOFRx（x= 0 ~ 3） **ADC 注入组偏移寄存器** **Reset：0x00000000**

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

字段	说明
11：0 IOFR	**注入组偏移值** 注入组通道转换的数据值 IDR 已经减去了在 ADC_IOFR 寄存器中定义的偏移量。

9.5.7 高阈值寄存器(ADC_AMOHR)

表 9-11 ADC_AMOHR 寄存器

ADC_AMOHR **ADC 高阈值寄存器** **Reset：0x00000000**

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

字段	说明
27：16 AMOHO	**模拟监控器高阀值对应的恢复偏移值** 定义高阀值的偏移值
11：0 AMOHT	**模拟监控器的高阀值** 定义高阀值

9.5.8 低阈值寄存器(ADC_AMOLR)

表 9-12 ADC_AMOLR 寄存器

ADC_AMOLR **ADC 低阈值寄存器** **Reset：0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称					AMOLO											
访问					RW											
Reset					0											
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称					AMOLT[11：0]											
访问					RW											
Reset					0											

字段	说明
27：16 AMOLO	**模拟监控器低阀值对应的恢复偏移值** 定义低阀值的偏移值
11：0 AMOLT	**模拟监控器的低阀值** 定义低阀值

9.5.9 规则组序列配置寄存器 0(ADC_RSQR0)

表 9-13 ADC_RSQR0 寄存器

ADC_RSQR0 **ADC 规则组序列配置寄存器 0** **Reset：0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称									RSQL[3：0]							
访问									RW							
Reset									0							
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称																
访问																
Reset																

字段	说明
23：20 RSQL	**规则组的长度** 0 ~ 11：定义规则组长度 1~12 长度必须小于实际有效的规则组序列长度的值

9.5.10 规则组序列配置寄存器 1(ADC_RSQR1)

表 9-14 ADC_RSQR1 寄存器

ADC_RSQR1 **ADC 规则组序列配置寄存器 1** **Reset：0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称			RSQ11[4：0]					RSQ10[4：0]					RSQ9[4：0]			
访问			RW													
Reset			0													
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称		RSQ8[4：0]					RSQ7[4：0]					RSQ6[4：0]				
访问		RW														
Reset		0														

字段	说明
29：0 RSQx(x=6~11)	**规则组通道选择** 0~11：外部通道 12：Bandgap 电压 13：温度传感器电压

9.5.11 规则组序列配置寄存器 2(ADC_RSQR2)

表 9-15 ADC_RSQR2 寄存器

ADC_RSQR2 **ADC 规则组序列配置寄存器 2** **Reset：0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称			RSQ5[4：0]					RSQ4[4：0]					RSQ3[4：0]			
访问			RW													
Reset			0													
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称		RSQ2[4：0]					RSQ1[4：0]					RSQ0[4：0]				
访问		RW														
Reset		0														

字段	说明
29：0 RSQx(x=0~5)	**规则组通道选择** 0~11：外部通道 12：Bandgap 电压 13：温度传感器电压

9.5.12 注入组序列配置寄存器(ADC_ISQR)

表 9-16 ADC_ISQR 寄存器

ADC_ISQR **ADC 注入组序列配置寄存器** **Reset：0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称											ISQL[1：0]		ISQ3[4：0]			
访问											RW					
Reset											0					
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称		ISQ2[4：0]					ISQ1[4：0]					ISQ0[4：0]				
访问		RW														
Reset		0														

字段	说明
21：20 ISQL	**注入组长度** 0 ~ 3： 定义注入组长度 1~4 **注意：该长度必须小于实际有效的注入组序列数。**
19：0 ISQx(x=0 ~ 3)	**注入组通道选择** 0~11：外部通道

字段	说明
	12：Bandgap 电压 13：温度传感器电压

9.5.13 注入组数据寄存器(ADC_IDRx)

表 9-17 ADC_IDRx (x=0 ~ 3)寄存器

ADC_IDRx (x=0 ~ 3) ADC 注入组数据寄存器 Reset：0x00000000

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

字段	说明
15：0 IDR	注入组数据寄存器 注意：ADC_IDR 已经减去了在 ADC_IOFR 寄存器中定义的偏移量。

9.5.14 规则组数据寄存器(ADC_RDR)

表 9-18 ADC_RDR 寄存器

ADC_RDR ADC 规则组数据寄存器 Reset：0x00000000

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

字段	说明
15：0 RDR	规则组数据寄存器 注意：规则组只有一个数据寄存器。如果 ADC 高速工作 CPU 无法及时处理数据，则用户必须使能 ADC 的 DMA 功能，以避免数据丢失。

10 模拟比较器（ACMP）

10.1 简介

模拟比较器提供一个用于比较两个模拟输入电压的电路，模拟多路复用器提供一个用于从 8 路通道中选择模拟输入信号的电路，其中 1 路通道由 6 位数字模拟转换器（DAC）提供，其他通道由外部输入提供。轮询模式和霍尔输出(Hall)功能专为电机应用而设计。

10.2 特性

- 片上 6 位数字模拟转换器（DAC），可从 VDD 或内部带隙基准电压（Bangap）中选择基准电压
- 可配置迟滞，支持 20/40mV
- 可在比较器输出上升沿、下降沿或上升/下降沿时产生中断
- 最多 8 个可选择比较器输入(ADC_IN0~ADC_IN6 以及内部 DAC)
- 支持停止（Stop）模式唤醒
- 支持轮询模式
- 支持霍尔(Hall) 输出

10.3 结构框图

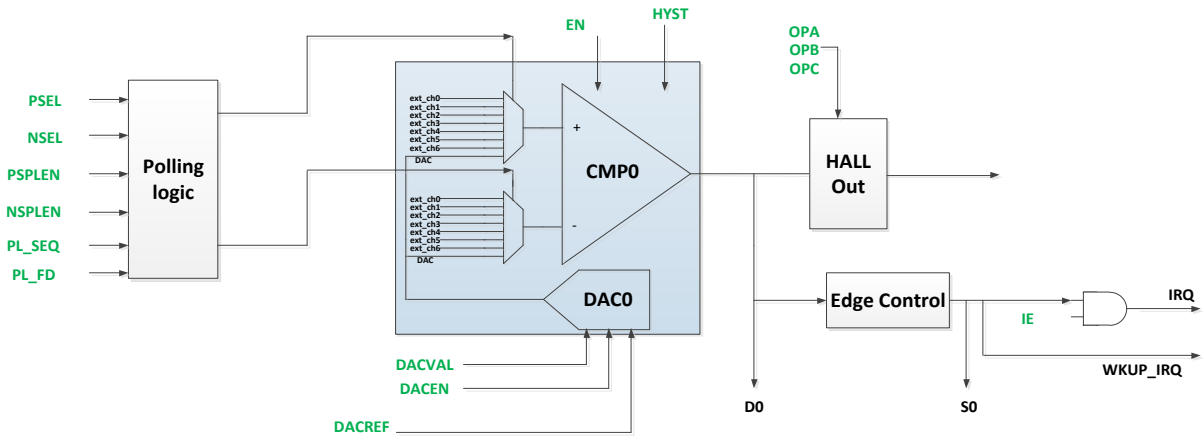

图 10-1 ACMP 结构框图

10.4 功能描述

ACMP 模块就功能而言由两部分组成：数模转换器 (DAC)和比较器 (CMP)。

DAC 包含一个 64 级 DAC （数模转换器）和相关的控制逻辑。通过配置 DACREF，DAC 可选择 Vdd 或片上带隙基准源作为 DAC 输入源。在 DAC 使能后，将 DACVAL 中设置的数据转换为步进式模拟输出，馈入输入端进行比较。

ACMP 可以实现正输入和负输入的模拟比较，然后提供一个数字输出和相关的中断。模拟比较器的正负输入均可从 8 个通用输入中选择。

10.4.1 普通模式

普通模式下，正输入和负输入固定所选中通道进行比较，比较结果以数字输出呈现。只要输出出现设置的有效边沿， SR 的状态位就会变为有效值。如果 IE 置位，则发生中断。ACMP 输出由总线时钟同步生成比较结果，以便 CPU 能读出比较结果。数据寄存器根据比较结果而改变，因此它可以用做一个跟踪标志，连续指示输入的电压变化。

10.4.2 轮询模式

轮询模式下，可通过相关逻辑动态切换比较器正输入或负输入的输入通道。轮询序列在 ACMP_CR4[PLSEQ]中定义，切换频率由 ACMP_FD[PLFD]控制。PSPLEN 和 NSPLEN 是轮询模式的使能位。PSPLEN 和 NSPLEN 无法同时使能。PSPLEN 和 NSPLEN 都使能不会触发轮询模式。因此，软件必须确保只使能上述两个字段其中之一。

这里提供一个有关轮询模式的示例。正输入轮询，轮询频率为 source_clk/100，外部通道 1-4 和 DAC 输出作轮询，负输入选择外部通道 0，下降沿触发中断。

步骤 1： IE = 1'b1,MOD = 2'b00；

步骤 2： DACEN = 1'b1,DACVAL=value；

步骤 3： PSPLEN = 1'b1, NSPLEN = 1'b0；

步骤 4： PLFD = 2'b01, PLSEQ = 8'b10011110, NSEL = 3'b000；

步骤 5： EN = 1'b1。

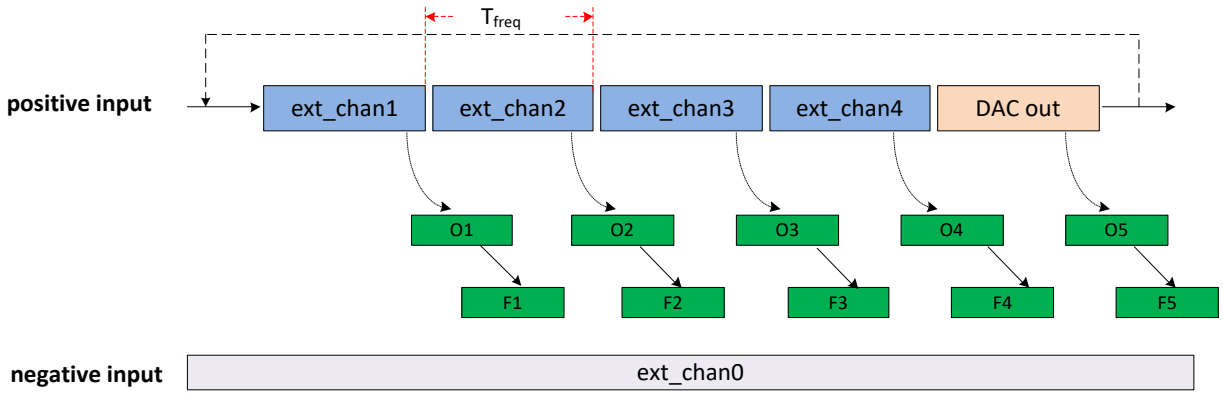

图 10-2 轮询模式工作流程图

10.4.3 轮询模式下霍尔输出

ACMP 有三个 hall 输出：Hall A Output， Hall B Output 和 Hall C Output， 这些信号连接到芯片内部的 PWDT 模块。hall 输出与轮询功能配合，可获取到无传感电机的 hall 位置（通过电机反电动势检测电机转子所在位置）。每个 hall 输出都可以通过轮询模式选择 8 个通道其中之一。

例如，若轮询模式 PLSEQ = 8'b00001110，则轮询顺序为：外部输入 1- >外部输入 2 -> 外部输入 3。

设置 ACMP_OPA[OPASEL] = 3'b010，则 Hall A Output 为 DR[O2]。

10.4.4 迟滞

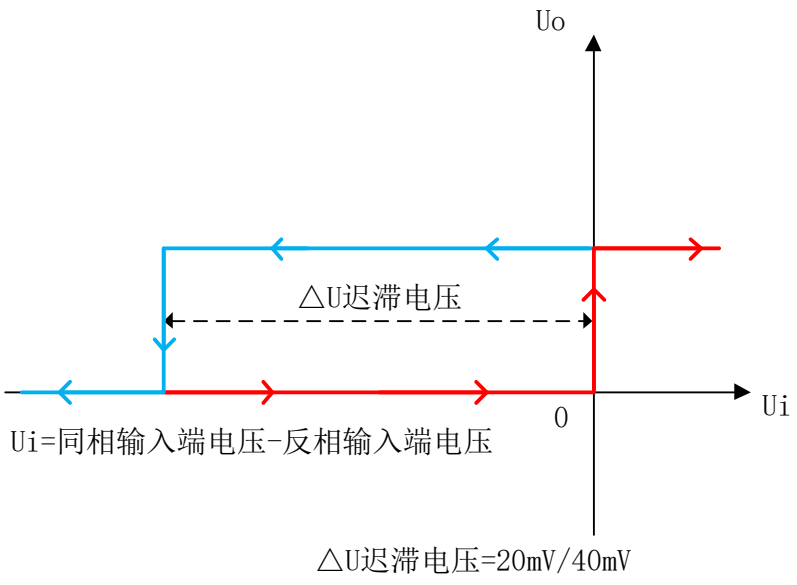

图 10-3 迟滞工作原理

10.4.5 低功耗模式唤醒

在 stop 模式下，ACMP 输出的有效边沿触发中断，可将 MCU 从低功耗模式唤醒。通过向 WPF 写 1 来清除唤醒标志位。

说明

唤醒功能仅作用于普通模式，轮询模式在低功耗模式下不工作。

10.5 寄存器定义

表 10-1 ACMP 寄存器映射

ACMP0 基地址：0x40005000

地址	名称	宽度	描述
ACMPx 基地址+0x0	ACMP_CR0	32	配置寄存器 0
ACMPx 基地址+0x4	ACMP_CR1	32	配置寄存器 1
ACMPx 基地址+0x8	ACMP_CR2	32	配置寄存器 2
ACMPx 基地址+0xC	ACMP_CR3	32	配置寄存器 3
ACMPx 基地址+0x10	ACMP_CR4	32	配置寄存器 4
ACMPx 基地址+0x14	ACMP_DR	32	数据输出寄存器
ACMPx 基地址+0x18	ACMP_SR	32	状态寄存器
ACMPx 基地址+0x1C	ACMP_FD	32	轮询分频器寄存器
ACMPx 基地址+0x20	ACMP_OPA	32	hall A 输出设置寄存器
ACMPx 基地址+0x24	ACMP_OPB	32	hall B 输出设置寄存器
ACMPx 基地址+0x28	ACMP_OPC	32	hall C 输出 设置寄存器
ACMPx 基地址+0x2C	ACMP_DACSR	32	DAC 参考源选择寄存器
0x40008820	ACMP_ANACFG	32	模拟配置寄存器

10.5.1 配置寄存器 0(ACMP_CR0)

表 10-2 ACMP_CR0 寄存器

ACMP_CR0 **配置寄存器 0** **Reset:00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									EN			IE	OUTEN	OPE	MOD	
访问									RW			RW	RW	RW	RW	
Reset									0			0	0	0	0	

字段	说明
7 EN	**ACMP 使能** 0：禁用 1：使能
4 IE	**中断使能** 0：禁用 1：使能
3 OUTEN	**比较结果输出到外部 PIN** 0：禁用 1：使能
2 OPE	**hall 输出使能** 0：禁用 1：使能
1：0 MOD	**中断触发模式** 00：输出下降沿中断 01：输出上升沿中断 10：输出下降沿中断(注：00 和 10 是相同的配置) 11：输出下降沿或上升沿中断

10.5.2 配置寄存器 1(ACMP_CR1)

表 10-3 ACMP_CR1 寄存器

ACMP_CR1 配置寄存器 1 **Reset:00000000**

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

字段	说明
6：4 PSEL	**正输入选择** 000：外部输入 0 001：外部输入 1 010：外部输入 2 011：外部输入 3 100：外部输入 4 101：外部输入 5 110：外部输入 6 111： DAC 输出
2：0 NSEL	**负输入选择** 000：外部输入 0 001：外部输入 1 010：外部输入 2 011：外部输入 3 100：外部输入 4 101：外部输入 5 110：外部输入 6 111： DAC 输出

10.5.3 配置寄存器 2(ACMP_CR2)

表 10-4 ACMP_CR2 寄存器

ACMP_CR2 配置寄存器 2 **Reset:00000000**

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

字段	说明
7 DACEN	**DAC 使能** 0： 禁用 1： 使能
5：0 DACVAL	**DAC 输出电压值**

10.5.4 配置寄存器 3(ACMP_CR3)

表 10-5 ACMP_CR3 寄存器

ACMP_CR3 配置寄存器 3 **Reset:00000000**

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

字段	说明
7 PSPLEN	**使能正输入为轮询模式** 0： 禁用 1： 使能 **注意：**PSPLEN 和 NSPLEN 无法同时使能。PSPLEN 和 NSPLEN 都使能不会触发轮询模式。

字段	说明
3 NSPLEN	**使能负输入为轮询模式** 0： 禁用 1： 使能 **注意：**PSPLEN 和 NSPLEN 无法同时使能。PSPLEN 和 NSPLEN 都使能不会触发轮询模式。

10.5.5 配置寄存器 4(ACMP_CR4)

表 10-6 ACMP_CR4 寄存器

ACMP_CR4 **配置寄存器 4** **Reset:00000000**

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

字段	说明
7：0 PLSEQ	**轮询通道序列设置** 0： 禁用相应的通道 1： 使能相应的通道

10.5.6 数据输出寄存器(ACMP_DR)

表 10-7 ACMP_DR 寄存器

ACMP_DR **数据输出寄存器** **Reset:00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称								O	O7	O6	O5	O4	O3	O2	O1	O0
访问								RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset								0	0	0	0	0	0	0	0	0

字段	说明
8 O	**正常模式输出**
7 O7	**轮询模式通道 7 输出**
6 O6	**轮询模式通道 6 输出**
5 O5	**轮询模式通道 5 输出**
4 O4	**轮询模式通道 4 输出**
3 O3	**轮询模式通道 3 输出**
2 O2	**轮询模式通道 2 输出**
1 O1	**轮询模式通道 1 输出**
0 O0	**轮询模式通道 0 输出**

10.5.7 状态寄存器(ACMP_SR)

表 10-8 ACMP_SR 寄存器

ACMP_SR **状态寄存器** **Reset:00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称							WPF	F	F7	F6	F5	F4	F3	F2	F1	F0
访问							W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset							0	0	0	0	0	0	0	0	0	0

字段	说明
9 WPF	**低功耗模式唤醒中断标志** 写 1 清除该标志
8 F	**正常模式中断标志** 写 1 清除该标志
7 F7	**轮询模式通道 7 中断标志** 写 1 清除该标志
6 F6	**轮询模式通道 6 中断标志**

字段	说明
	写 1 清除该标志
5 F5	**轮询模式通道 5 中断标志** 写 1 清除该标志
4 F4	**轮询模式通道 4 中断标志** 写 1 清除该标志
3 F3	**轮询模式通道 3 中断标志** 写 1 清除该标志
2 F2	**轮询模式通道 2 中断标志** 写 1 清除该标志
1 F1	**轮询模式通道 1 中断标志** 写 1 清除该标志
0 F0	**轮询模式通道 0 中断标志** 写 1 清除该标志

10.5.8 轮询分频器寄存器(ACMP_FD)

表 10-9 ACMP_FD 寄存器

ACMP_FD **轮询分频器寄存器** **Reset:00000000**

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

字段	说明
1：0 PLFD	**轮询模式分频器** 此分频器控制轮询通道的切换频率 00： source_clk/256 01： source_clk/100

字段	说明
	10： source_clk/70
	11： source_clk/50

10.5.9 霍尔输出 A 设置寄存器(ACMP_OPA)

表 10-10 ACMP_OPA 寄存器

ACMP_OPA **Hall A 输出设置寄存器** **Reset:00000000**

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

字段	说明
2：0 OPASEL	**Hall A 输出 设置** 000：轮询通道 0 001：轮询通道 1 010：轮询通道 2 011：轮询通道 3 100：轮询通道 4 101：轮询通道 5 110：轮询通道 6 111：轮询通道 7

10.5.10 霍尔输出 B 设置寄存器(ACMP_OPB)

表 10-11 ACMP_OPB 寄存器

ACMP_OPB **Hall B 输出设置寄存器** **Reset:00000000**

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

字段	说明
2：0 OPBSEL	**Hall B 输出设置** 000：轮询通道 0 001：轮询通道 1 010：轮询通道 2 011：轮询通道 3 100：轮询通道 4 101：轮询通道 5 110：轮询通道 6 111：轮询通道 7

10.5.11 霍尔输出 C 设置寄存器(ACMP_OPC)

表 10-12 ACMP_OPC 寄存器

ACMP_OPC **Hall C 输出设置寄存器** **Reset:00000000**

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

字段	说明
2：0 OPBSEL	**Hall C 输出设置** 000：轮询通道 0 001：轮询通道 1 010：轮询通道 2 011：轮询通道 3 100：轮询通道 4 101：轮询通道 5 110：轮询通道 6 111：轮询通道 7

10.5.12 DAC 参考源选择寄存器(ACMP_DACSR)

表 10-13 ACMP_DACSR 寄存器

ACMP_DACSR DAC 参考源选择寄存器 Reset:00000000

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

字段	说明
0 DACREF	**DAC 参考选择** 0： DAC 选择带隙作为参考 1： DAC 选择 Vdd 作为参考

10.5.13 模拟配置寄存器(ACMP_ANACFG)

表 10-14 ACMP_ANACFG 寄存器

ACMP_ANACFG ANA 配置寄存器 Reset:00000000

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

字段	说明
23 HYST	**模拟比较器 0 迟滞电压选择** 0： 20mV 1： 40mV

字段	说明
22：21 LPFSEL	**低通滤波器选择** **建议选择 1MHz 以获得更好的性能** 00：200kHz 01：500kHz 10：750kHz 11：1MHz

11 脉宽调制（PWM）

11.1 简介

PWM 模块是一个多功能的定时器，支持输入捕获、输出比较、正交解码和 PWM 信号生成。PWM 的计数功能是通过一个 16 位的计数器产生的。该 MCU 设备包含 2 个 PWM 模块，每个 PWM 模块支持 8 通道。

11.2 特性

PWM 特性包括：

- PWM 时钟源可选。时钟源可以是总线时钟、HSI 时钟
- 16 位预分频器支持 1 至 65536 分频
- 16 位计数器
  - 它可以为一个自由运行、没有限制的计数器，或一个有初值和终值的计数器
  - 支持向上、向上-向下两种计数方式
- 每个通道都可以配置为输入捕获、输出比较或边沿对齐 PWM 模式、中心对齐 PWM 模式
- 在输入捕获模式下，捕获可以发生在上升沿、下降沿或上升沿/下降沿
- 输入捕获模式下，可以为某些通道选择输入滤波器
- 在输出比较模式下， 可以在匹配时设置、清除或者反转输出
- 每对通道都可以组合起来生成一个 PWM 信号，并且能够独立控制 PWM 信号的上升沿和下降沿
- PWM 通道可以采用具有同等输出或者互补输出的成对工作方式
- 死区插入可用于每一对互补通道
- 可生成匹配触发
- 软件控制 PWM 输出
- 输出屏蔽可设置通道为无效状态
- 对于故障控制最多有 3 个故障输入
- 每个通道的极性可配置
- 每个通道产生一个中断
- 计数器溢出时产生中断
- 当检测到故障条件时，产生中断

- 同步加载写缓冲 PWM 寄存器
- 关键寄存器支持写保护
- 用于脉冲和周期宽度测量的双边沿捕获
- 支持正交解码（AB 相输入引脚映射到每个 PWM 模块的 CH0 和 CH1）
- 支持全局时基
- 支持 PWM 输出波形相位偏移

11.3 结构框图

PWM 每通道使用一个输入/输出（I / O）引脚、CHn（PWM 通道（n）），其中 n 是通道编号（0-7）。

下图为 PWM 结构图。PWM 的核心部分为 16 位计数器，具有可编程的初始值和最终值，其计数可以是向上或向上-向下。

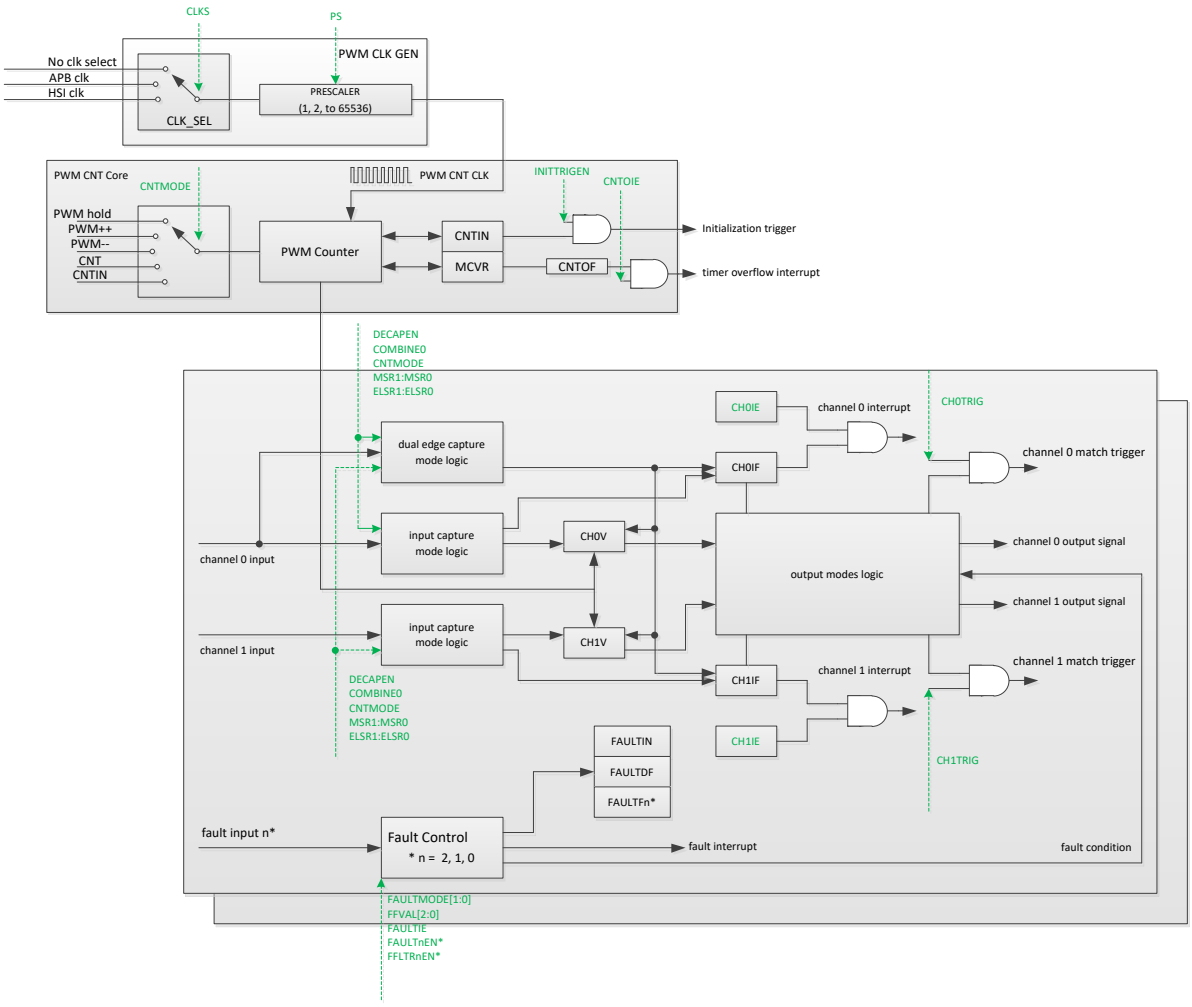

图 11-1 PWM 结构框图

11.4 功能描述

11.4.1 时钟源

PWM_INIT 寄存器中的 CLKSRC[1：0]位选择 PWM 计数器的两个可能时钟源之一或禁用 PWM 计数器。MCU 复位后，CLKSRC[1：0] = 00，因此没有选择时钟源。通过将 CLKSRC [1：0]位写入 00 来禁用 PWM 计数器不会影响 PWM 计数器值或其他寄存器。内部 HSI 时钟源可以作为 PWM 计数器的固定时钟源，时钟源频率为 8MHz。

11.4.2 计数器

PWM 包含一个 16 位计数器，用于通道输入或输出模式。PWM 计数器时钟是由预分频器分频的选定时钟。PWM 计数器具有以下工作模式：

- 向上计数
- 向上-向下计数
- 正交解码模式

11.4.2.1 向上计数

当 QDIEN=0 且 CNTMODE=0 时，为向上计数模式。CNTIN 定义计数的起始值，MCVR 定义计数的终值，如下图所示。CNTIN 的值加载到 PWM 计数器中，计数器递增，直到达到 MCVR 的值，此时计数器重新加载 CNTIN 的值。向上计数模式 PWM 周期为（MCVR － CNTIN + 0x0001）×PWM 计数器时钟周期。当 PWM 计数器从 MCVR 变为 CNTIN 时，CNTOF 位置 1。

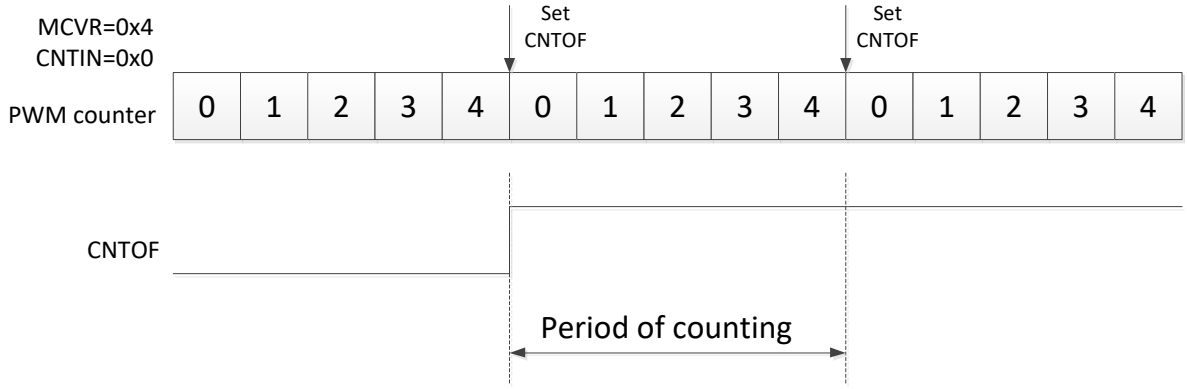

图 11-2 向上计数

11.4.2.2 向上-向下计数

当 QDIEN=0 且 CNTMODE=1 时，选择向上-向下计数。CNTIN 定义计数的起始值，MCVR 定义计数的终值。CNTIN 的值被加载至 PWM 计数器中，并且计数器递增直到达到 MCVR 的值，此时计数器递减，直到它返回到 CNTIN 的值，然后重新开始上下计数。

向上-向下计数时的 PWM 周期为 2×（MCVR-CNTIN）×PWM 计数器时钟周期。当 PWM 计数器从 MCVR 变为 MCVR - 1 时，CNTOF 位置 1，如下图所示。

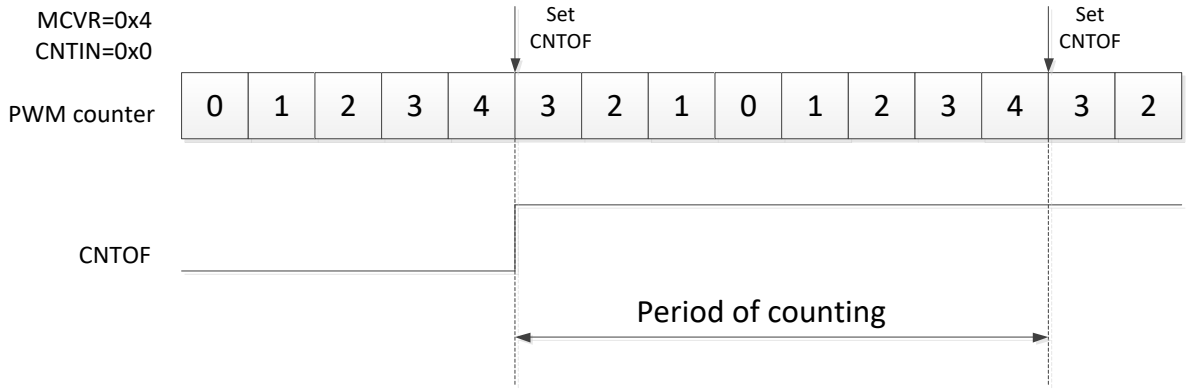

图 11-3 向上-向下计数

11.4.3 工作模式

PWM 可配置为输入捕获、输出比较或边沿对齐 PWM 模式、中心对齐 PWM 模式、组合模式、正交解码模式，详细配置参考以下表格。

表 11-1 工作模式配置

DECAPEN	COMBINE	CNTMODE	MSR1：MSR0	ELSR1：ELSR0	工作模式	配置
0	0	0	00	01	输入捕获	上升沿捕获
				10		下降沿捕获
				11		上升或下降沿捕获
			01	01	输出比较	匹配时切换输出
				10		匹配时清除输出
				11		匹配时设置输出
			1X	10	EPWM	匹配时清除输出
				X1		匹配时设置输出
		1	XX	10	CPWM	向上匹配时清除输出
				X1		向上匹配时设置输出

DECAPEN	COMBINE	CNTMODE	MSR1：MSR0	ELSR1：ELSR0	工作模式	配置
	1	0	XX	10	向上计数组合模式	Channel(n) 匹配时设置输出，Channel(n+1) 匹配时清除输出
				X1		Channel (n)匹配时清除输出，Channel (n+1) 匹配时设置输出
		1		10	向上-向下计数组合模式	Channel(n) 匹配时设置输出，Channel(n+1) 匹配时清除输出
				X1		Channel (n)匹配时清除输出，Channel (n+1) 匹配时设置输出
1	0	0	X0	01	双边沿单次捕获	上升沿
				10		下降沿
				11		上升沿或下降沿
			X1	01	双边沿持续捕获	上升沿
				10		下降沿
				11		上升沿或下降沿
X	X	X	XX	XX	正交解码	QDIEN=1 使能正交解码，正交解码器模式优先于其他模式

11.4.4 输入捕获模式

当通道输入出现选定的边沿时，PWM 计数器的当前值会被捕获到 PWM_CHnV 寄存器中。同时，CHnIF 位置 1。如果由 CHnIE = 1 使能，则产生通道中断。当通道配置为输入捕获时，PWMx_CHn 引脚为边沿敏感输入。ELSnR1:ELSnR0 控制位决定何种边沿（下降沿或上升沿）触发输入捕获事件。注意，可以被正确检测到的输入信号最大频率为 1/4 总线时钟，这是满足信号采样的奈奎斯特准则所必需的条件。在输入捕捉模式下，忽略写入 CHnV 寄存器操作。如下图，通道配置检测上升沿捕获。

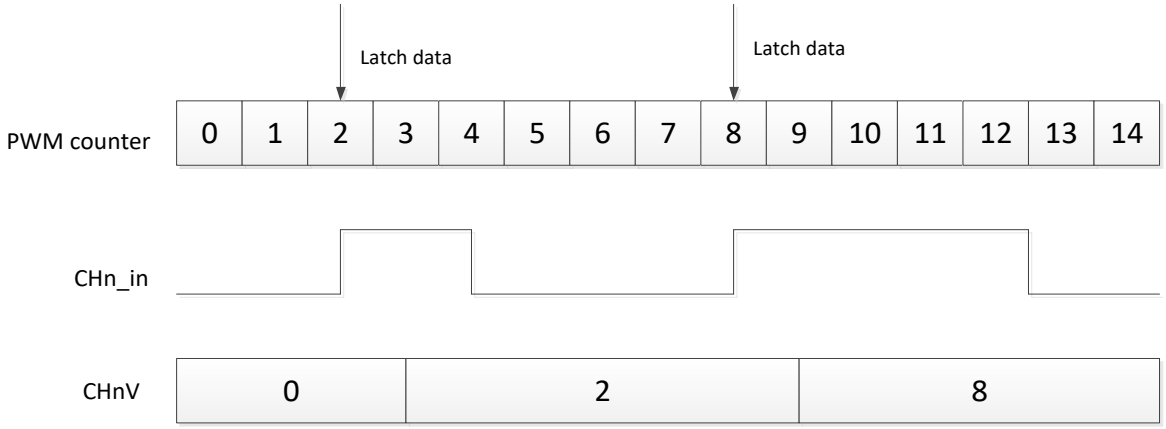

图 11-4 输入捕获模式

对于 PWM CH0~CH3，存在额外的捕获滤波器来过滤输入信号。滤波器为 5 位计数器，可通过寄存器 CHnCAPFVAL（n = 0,1,2,3）进行配置。当 CHnCAPFVAL [4：0] = 0 时，滤波器功能被禁用，如果 CHnCAPFVAL [4：0]≠00000，输入信号将被延迟(CHnCAPFVAL [4：0] x 4)个总线时钟，然后才传输到边沿检测器。

通道输入滤波器中计数器的时钟是总线时钟的 4 分频。

11.4.5 输出比较模式

在输出比较模式下，PWM 可以生成具有可编程位置、极性、持续时间和频率的定时脉冲。当计数器与输出比较通道的 CHnV 值匹配时，可以设置、清除或翻转通道 n 输出。当通道最初配置为翻转（Toggle）模式时，通道保持输出先前的值，直到发生第一个输出比较事件。如果在通道 n 匹配时（PWM 计数器 = CHnV）CHnIE = 1，则 CHnIF 位置 1，产生通道 n 中断。

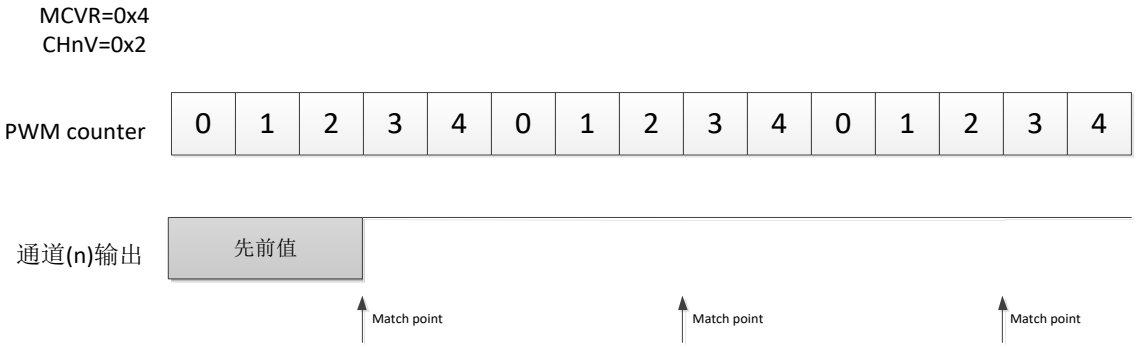

图 11-5 匹配设置输出比较模式

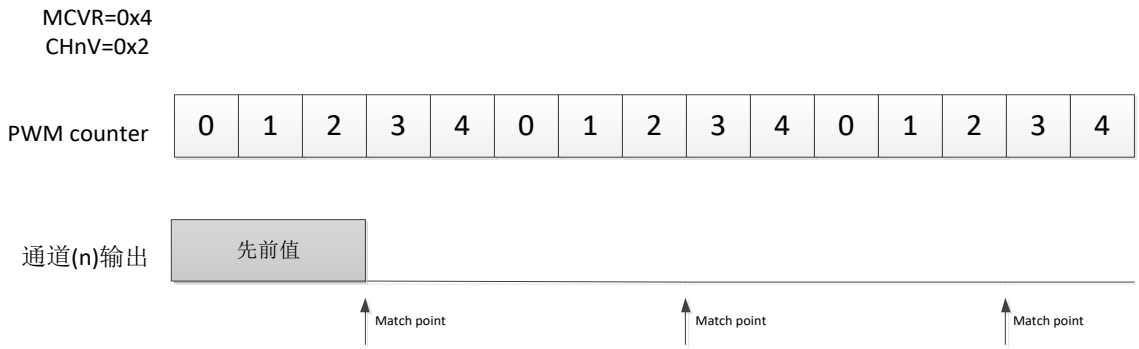

图 11-6 匹配清除输出比较模式

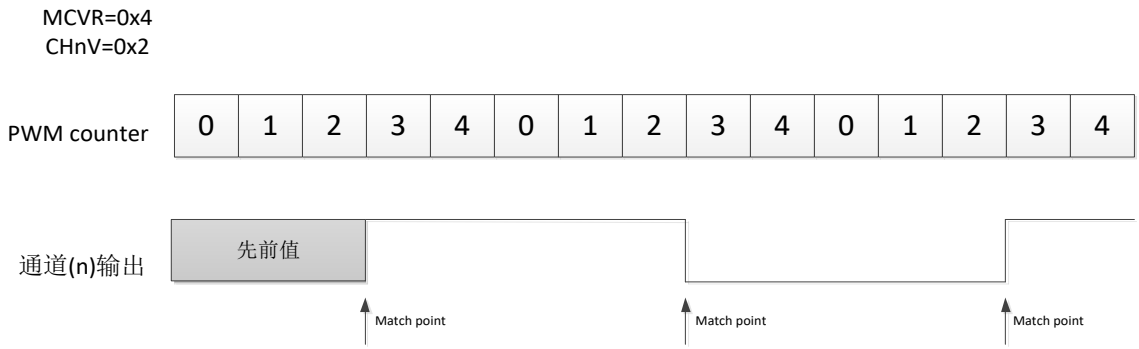

图 11-7 匹配翻转输出比较模式

11.4.6 边沿对齐 PWM 模式(EPWM)

周期=(MCVR-CNTIN + 0x0001)×PWM 计数器时钟周期

脉冲宽度 (占空比)=(CHnV + 0x0001 – CNTIN) ×PWM 计数器时钟周期

ELSnR1：ELSnR0=1：0，通道(n)输出在 CNTIN 值加载到 PWM 计数器时为高电平，在通道(n)匹配（PWM 计数器 = CHnV）时为低电平。

ELSnR1：ELSnR0=X：1，通道(n)输出在 CNTIN 值加载到 PWM 计数器时为低电平，在通道(n)匹配（PWM 计数器 = CHnV）时为高电平。

CHnIE = 1，通道(n)匹配（PWM 计数器= CHnV）时，CHnIF 位置 1 且产生通道(n)中断。

这种类型的 PWM 信号称为边沿对齐，是因为所有 PWM 信号的前沿与周期的开端对齐，这对于 PWM 内的所有通道都是相同的。

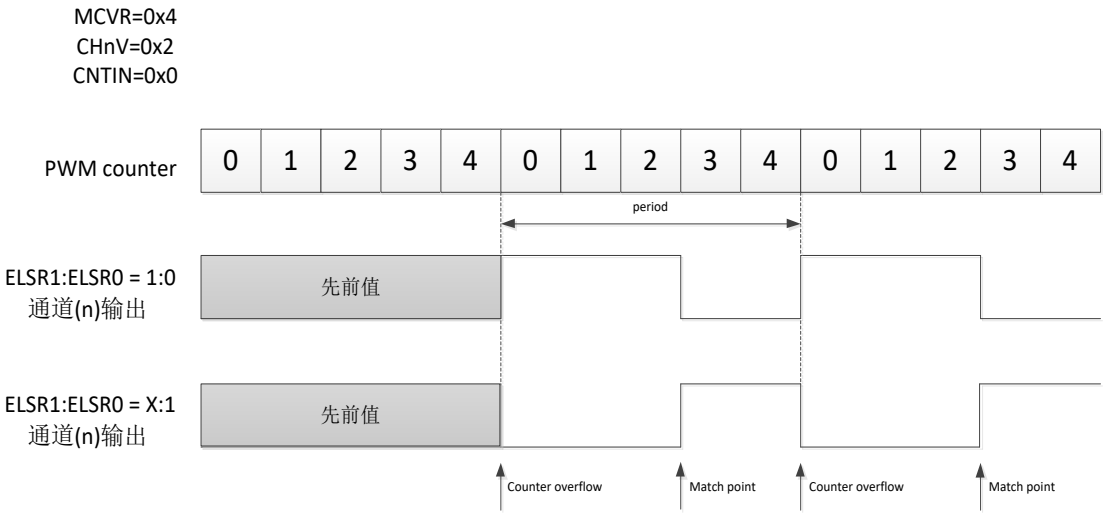

图 11-8 EPWM 波形

ELSnR1：ELSnR0=1：0，如果(CHnV = 0x0000)，则通道(n)输出为 0%占空比 EPWM 信号；如果(CHnV >= MCVR)，则通道(n)输出为 100%占空比 EPWM 信号，ELSnR1：ELSnR0=X：1 则反之。

11.4.7 中心对齐 PWM 模式(CPWM)

周期=2 × (MCVR - CNTIN) ×PWM 计数器时钟周期

脉冲宽度(占空比) =2 × (CHnV - CNTIN) ×PWM 计数器时钟周期

在 CPWM 模式下，PWM 计数器向上计数直到达到 MCVR，然后向下计数直至达到 CNTIN。

ELSnR1：ELSnR0=1：0，通道(n)输出在向下计数时与通道(n)匹配（PWM 计数器 = CHnV）下为高电平，在向上计数时与通道(n)匹配（PWM 计数器 = CHnV）的情况下为低电平。

ELSnR1：ELSnR0=X：1，通道(n)输出在向下计数时与通道(n)匹配（PWM 计数器 = CHnV）时为低电平，在向上计数时与通道(n)匹配（PWM 计数器 = CHnV）的情况下为高电平。

CHnIE=1，当 PWM 计数减少或 PWM 计数增加，在通道(n)匹配（PWM 计数器= CHnV）时，CHnIF 位置 1 且通道(n)中断产生。

这种类型的 PWM 信号被称为中心对齐，因为所有通道的脉冲宽度中心与 CNTIN 的值对齐。

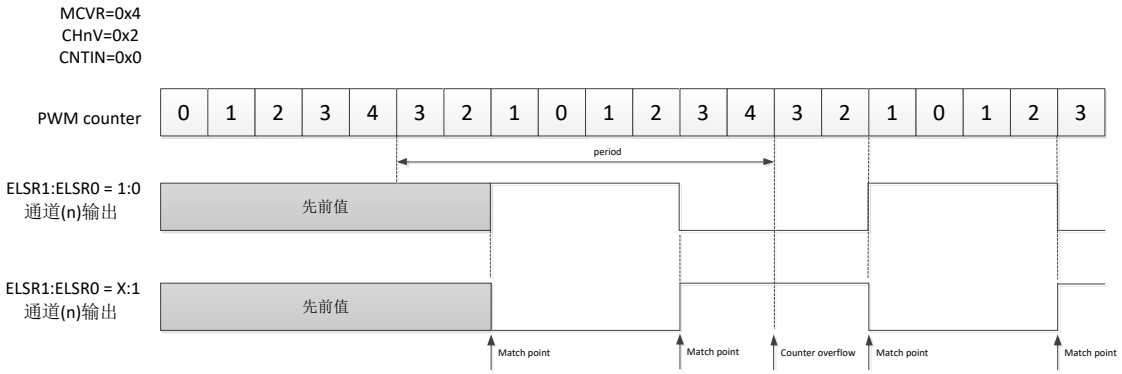

图 11-9 CPWM 波形

ELSnR1：ELSnR0=1：0，如果(CHnV = 0x0000)，则通道(n)输出为 0%占空比 CPWM 信号；如果(CHnV >= MCVR)，则通道(n)输出为 100%占空比 CPWM 信号，ELSnR1：ELSnR0=X：1 则反之。

11.4.8 组合模式

组合模式下，将偶数通道（n）和相邻的奇数通道（n + 1）组合以在通道（n）输出中产生 PWM 信号。

根据计数方式的不同，可分为向上计数组合模式和向上-向下计数组合模式。

ELSnR1：ELSnR0=1：0，在通道(n)匹配时(PWM 计数器=CH(n)V)为高电平；在通道(n+1)匹配时为低电平。

ELSnR1：ELSnR0=X：1，在通道(n)匹配时(PWM 计数器=CH(n)V)为低电平；在通道(n+1)匹配时为高电平。

CH(n)IE=1，通道(n)匹配时(PWM 计数器=CH(n)V)，CH(n)IF 位置 1 且产生通道(n)中断。

CH(n+1)IE=1，通道(n+1) 匹配时(PWM 计数器=CH(n+1)V)，CH(n+1)IF 位置 1 且产生通道(n+1)中断。

说明

ELS(n+1)R1 和 ELS(n+1)R0 位不用于控制 channels (n) 和 (n+1) 输出。

11.4.8.1 向上计数组合模式

周期=(MCVR - CNTIN + 0x0001) ×PWM 计数器时钟周期

脉冲宽度 (占空比) = |CH(n+1)V - CH(n)V| ×PWM 计数器时钟周期

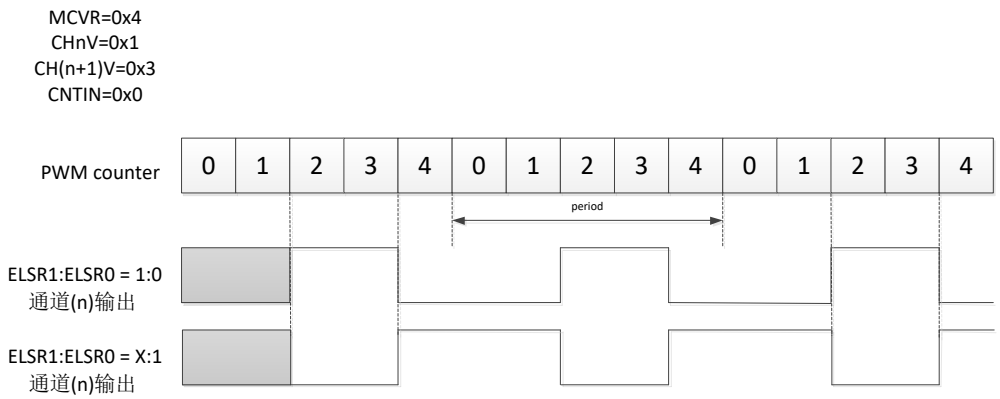

图 11-10 向上计数组合模式输出波形

以下展示了向上计数组合模式的各种条件下 PWM 信号输出波形：

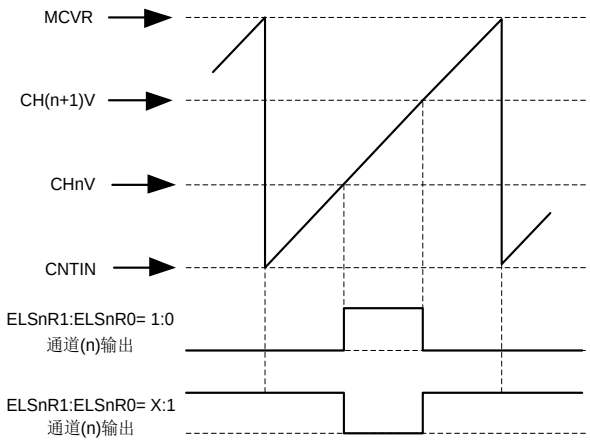

图 11-11 (CNTIN < CHnV/CH(n+1)V < MCVR) &(CHnV < CH(n+1)V)条件下输出波形

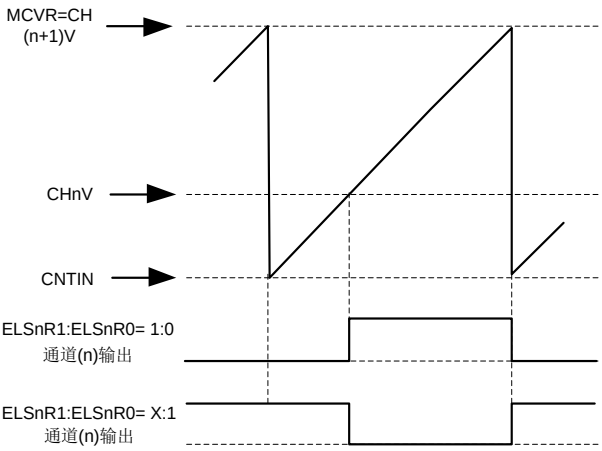

图 11-12 (CNTIN < CHnV < MCVR)&(CH(n+1)V = MCVR)条件下输出波形

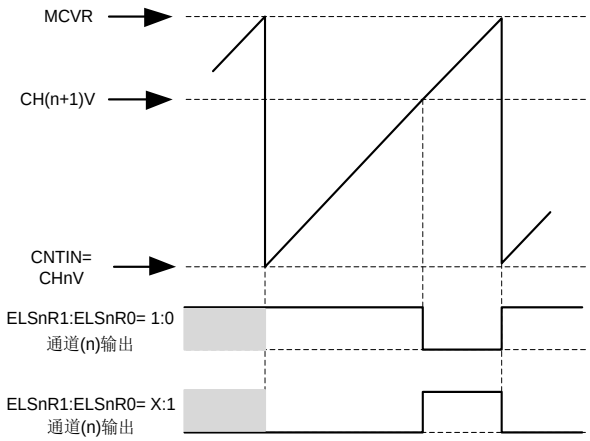

图 11-13 (CHnV = CNTIN)&(CNTIN < CH(n+1)V < MCVR)条件下输出波形

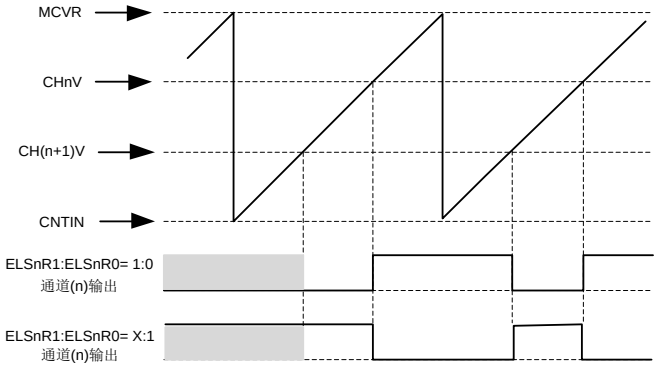

图 11-14 (CNTIN < CHnV/ CH(n+1)V < MCVR)且(CHnV > CH(n+1)V)条件下输出波形

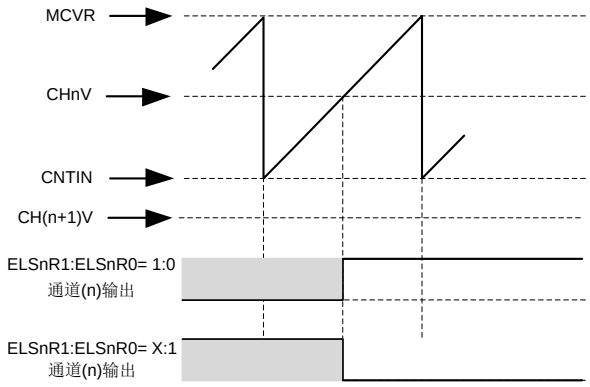

图 11-15 (CH(n+1)V < CNTIN)&(CNTIN < CHnV < MCVR)条件下输出波形

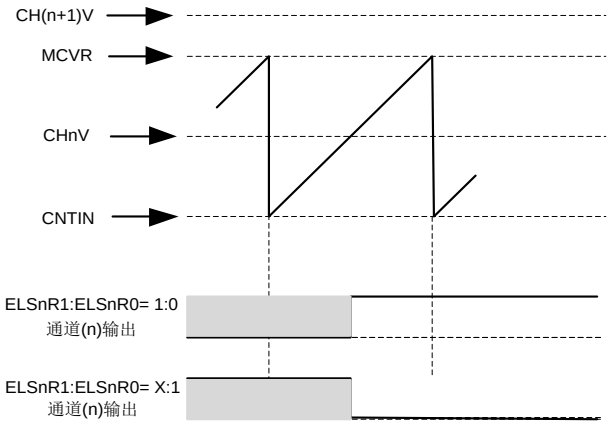

图 11-16 (CH(n+1)V > MCVR)且(CNTIN < CHnV < MCVR)条件下输出波形

11.4.8.2 向上-向下计数组合模式

周期=2 × (MCVR - CNTIN) ×PWM 计数器时钟周期

在向上-向下计数过程中，通道在向上计数产生一次匹配，向下计数也会产生一次匹配。为了便于控制通道输出，提供了匹配生效点设置功能，可通过设置匹配生效点 CHSCR[DIR]是在向上计数或向下计数过程中作用。因为匹配生效点依赖于计数方向，需要清晰定义向上计数和向下计数区间范围，如下图：

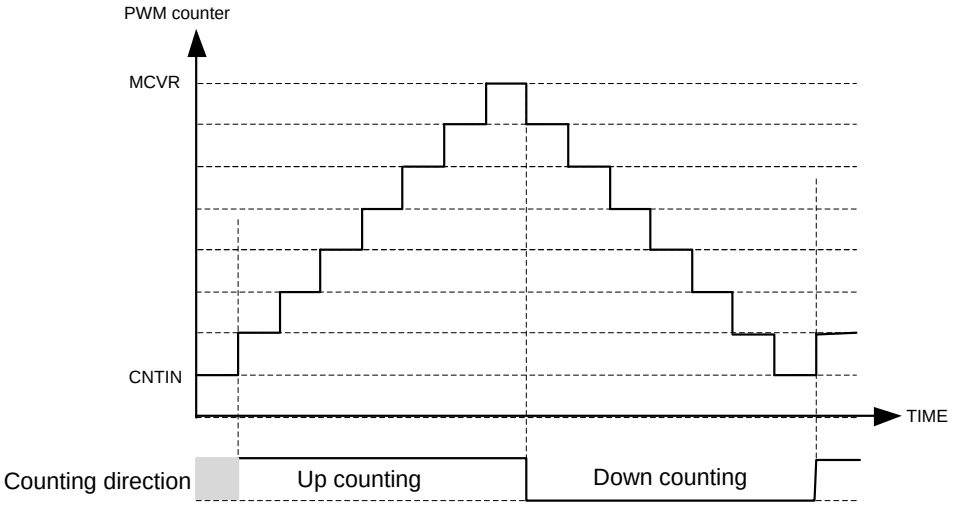

图 11-17 向上-向下计数区间范围

说明

首次计数周期 CNTIN 值为向上计数区间，后续周期 CNTIN 值都为向下计数区间。

2 个匹配点可以组合出 4 种情况：

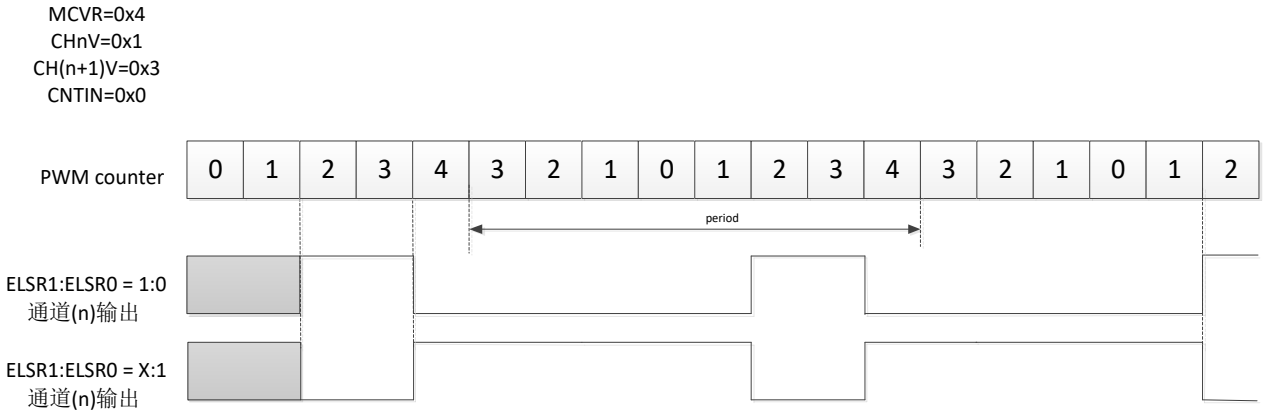

图 11-18 CHn 匹配点 DIR=1(Up)，CH(n+1)匹配点 DIR=1(Up)

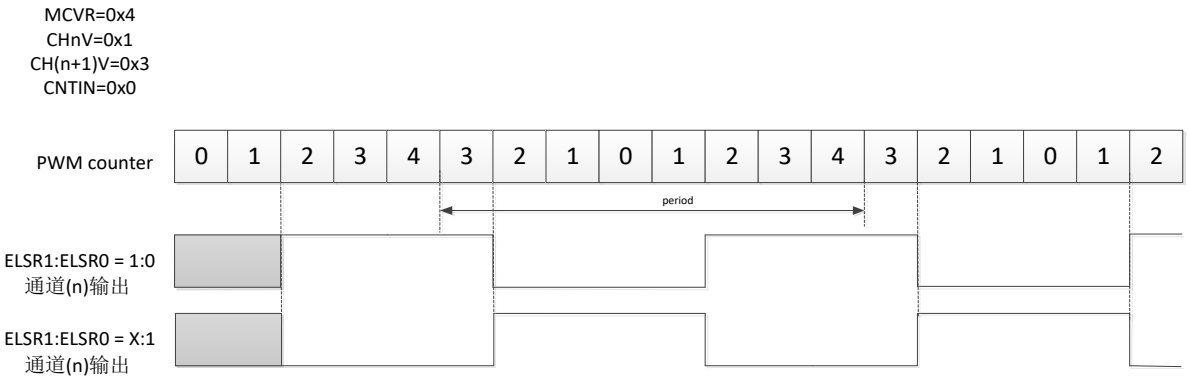

图 11-19 CHn 匹配点 DIR=1(Up)，CH(n+1)匹配点 DIR=0(Down)

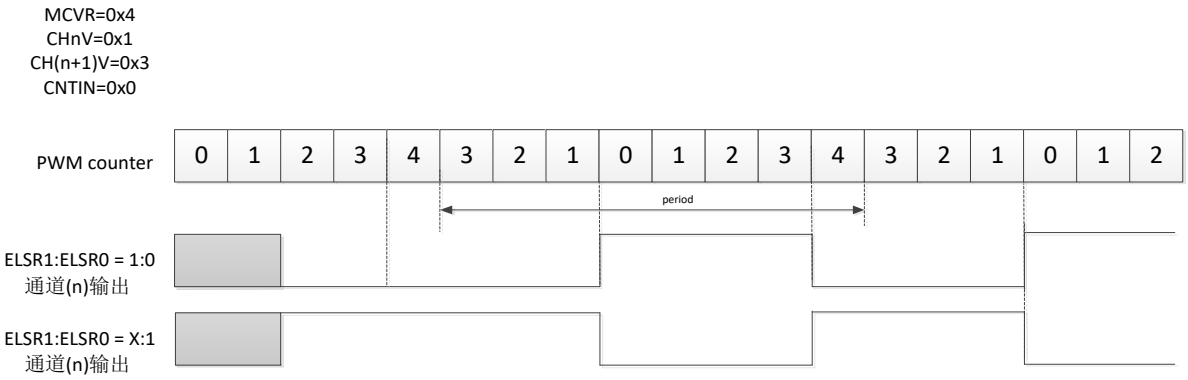

图 11-20 CHn 匹配点 DIR=0(Down)，CH(n+1)匹配点 DIR=1(Up)

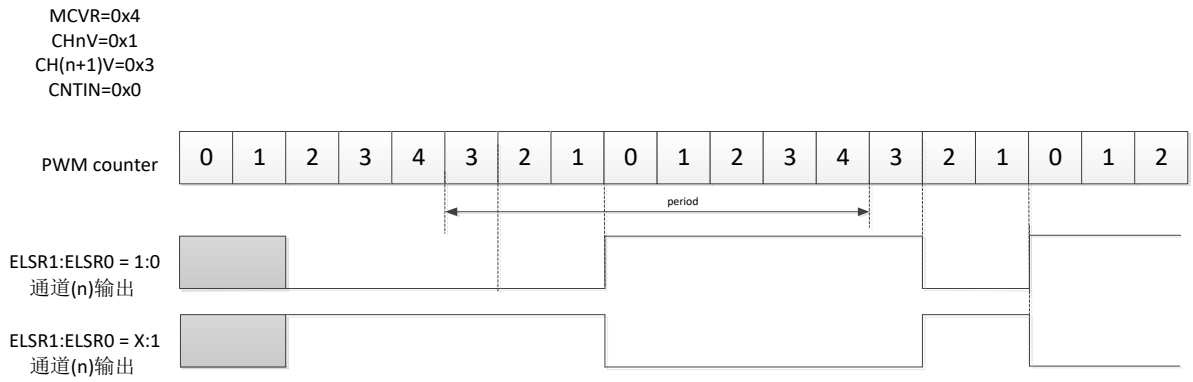

图 11-21 CHnV 匹配点 DIR=0(Down)，CH(n+1)V 匹配点 DIR=0(Down)

11.4.8.3 互补功能

组合模式下支持互补功能。使能互补 PAIRnCOMPEN=1， 通道 (n+1) 输出和通道 (n)输出电平相反。

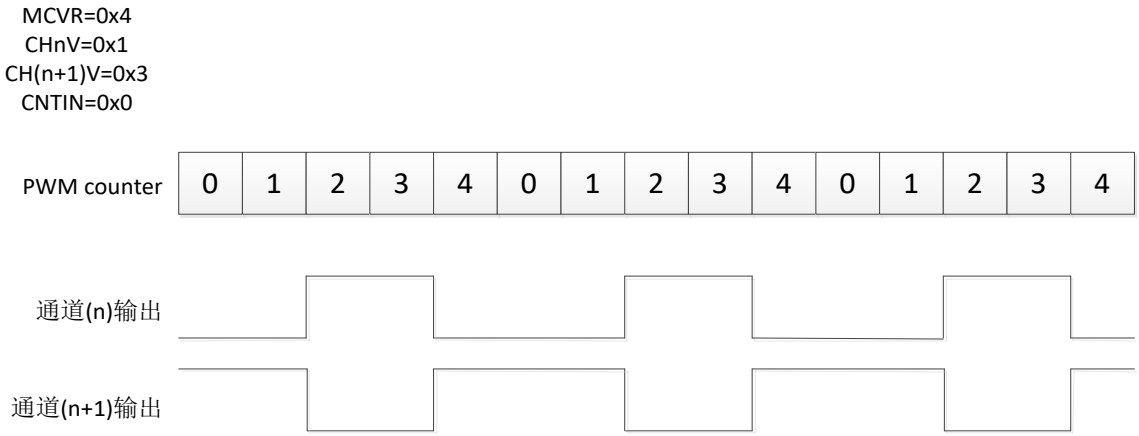

图 11-22 互补模式输出

11.4.8.4 死区时间插入

当 DTEN = 1 且 DTVAL[5：0]为非零时，使能死区插入。PWM_DTSET 寄存器定义了可用于所有 PWM 通道的死区延迟。DTPSC[1：0]位定义总线时钟的预分频器，DTVAL[5：0]位定义死区模数，即死区预分频器时钟数。死区延迟插入确保没有两个互补信号（通道 (n)和 通道(n+1)）同时驱动活动状态。

如果 CH(n)POL = 0， CH(n+1)POL = 0，并且使能死区，那么当出现通道 (n)匹配（PWM 计数器= C(n)V）时，channel (n)输出保持低电平状态，直到死区延迟结束，通道 (n)输出置位时。类似地，当发生通道(n+1)匹配(PWM 计数器 = CH(n+1)V)时，通道 (n+1)输出保持低电平状态，直到死区延迟结束，通道 (n+1)输出置位时。

如果 CH(n)POL = 1， CH(n+1)POL = 1，并且使能死区，则当出现通道(n)匹配(PWM 计数器 = CH(n)V)时，通道 (n)输出保持高电平状态，直到死区延迟结束，通道 (n)输出清零时。类似地，当发生

通道 (n+1)匹配(PWM 计数器 = CH(n+1)V)时，通道 (n+1)输出保持高电平状态，直到死区延迟结束，通道 (n+1)输出清零时。

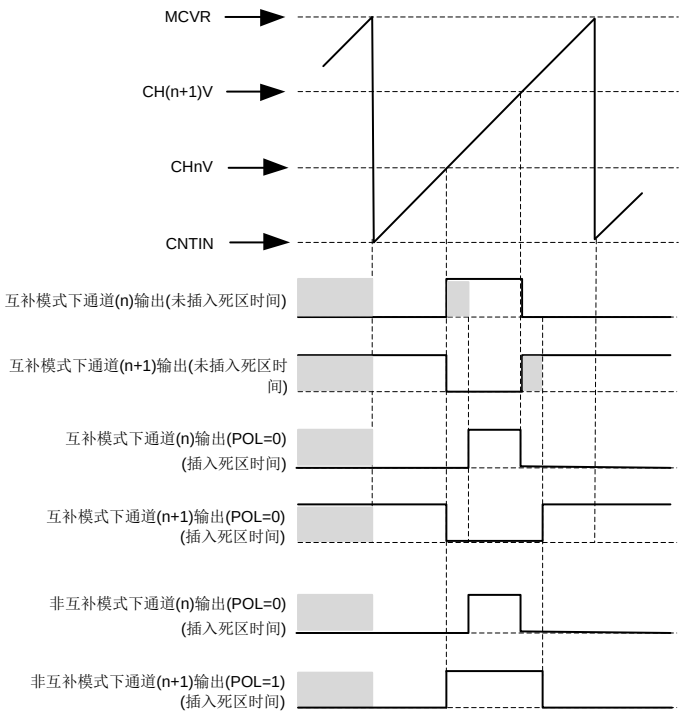

图 11-23 死区时间插入

11.4.8.5 反相

反相功能用于将通道(n)和通道(n+1)输出信号进行交换。在如下情况选择反相操作：

PAIR(n)INVEN = 1， n=0,1,2,3。

11.4.8.6 移相

在某些情况下，通道(n+1)匹配会发生在下一个 PWM 周期。 当(CNTIN < CHnV < MCVR)&(CNTIN < CH(n+1)V < MCVR)时， CHnV > CH(n+1)V 时会发生此种情景。

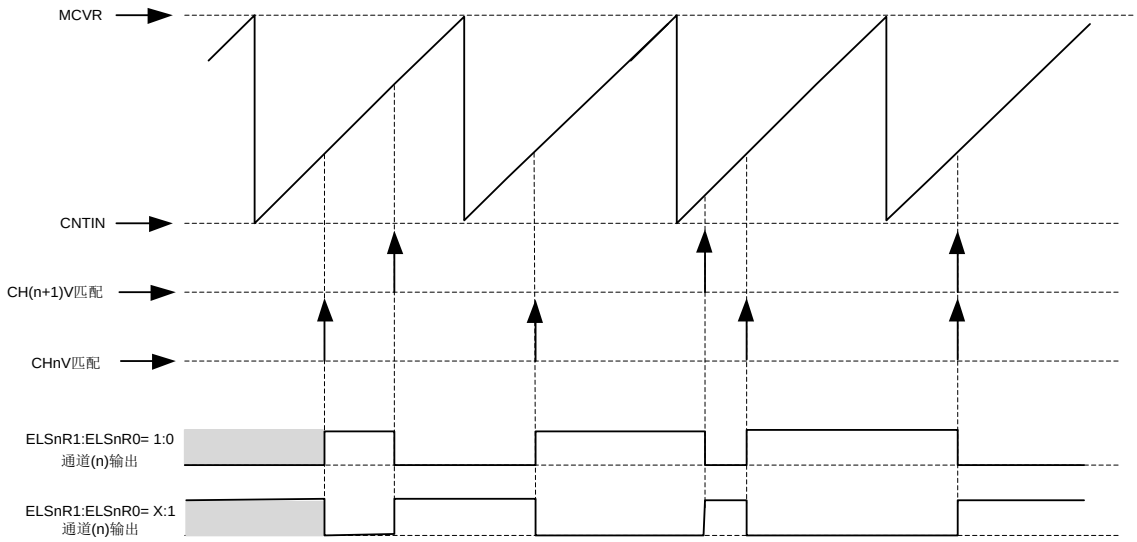

图 11-24 CH(n+1)V 在下一周期匹配的输出波形

基于以上特性，当 PWM 模块多个通道以组合模式输出时，可设定同一个 PWM 模块不同通道对之间的相位偏移量。此种行为有利于产生合适的车灯控制信号，通道输出边沿不重合的信号便于消除噪声。

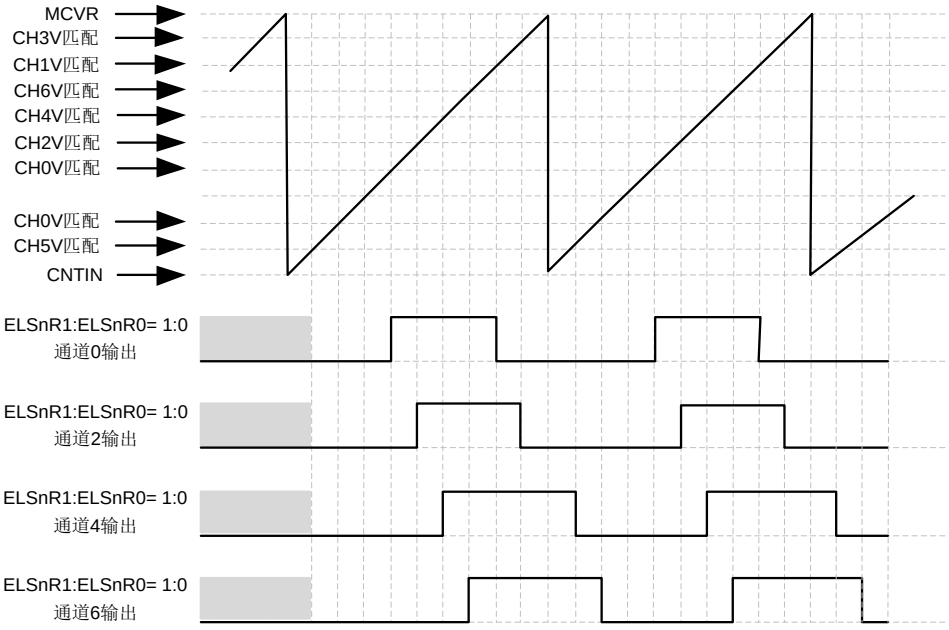

图 11-25 多通道之间相位偏移输出波形

11.4.9 双边沿捕获模式

DECAPEN = 1，则选择双边沿捕捉模式。该模式可用于测量通道(n)输入端脉冲宽度或信号周期。在此模式下，仅使用通道(n)输入，忽略通道(n+1)输入。当 n 为 0 或 2 时，通道(n)滤波器在此模式下有效。ELS(n)R1：ELS(n)R0 位选择通道(n)捕获的边沿，ELS(n+1)R1：ELS(n+1)R0 位选择通道(n+1)捕获

的边沿。如果 ELS(n)R1：ELS(n)R0 和 ELS(n+1)R1：ELS(n+1)R0 位都选择相同的边沿，则为周期测量；如果这些位选择不同的边沿，则为脉冲宽度测量。

如果在通道(n)输入处检测到通道(n)位选择的边沿，则 CH(n)IF 置位并生成通道(n)中断（如果 CH(n)IE = 1）。如果在通道(n) 输入且(CH(n)IF = 1)时检测到通道 (n+1)位选择的边沿，则 CH(n+1)IF 置位且生成通道(n+1) 中断 (如果 CH(n+1)IE = 1)。双边沿模式不支持同时使能通道(n)和通道(n+1)中断。

当在通道 (n)输入处检测到通道(n)选择的边沿时，PWM_CH(n)V 寄存器存储 PWM 计数器的值。在通道(n)输入处检测到通道(n+1)选择的边沿时，PWM_CH(n+1)V 寄存器存储 PWM 计数器的值。在此模式下，当读取 PWM_CH(n)V 和 PWM_CH(n+1)V 寄存器时，唯一的要求是必须在 CH(n+1)V 之前读取 CH(n)V。

如下图，通道 (n)选择上升沿捕获，通道 (n+1)选择下降沿沿捕获，用于测量脉宽。

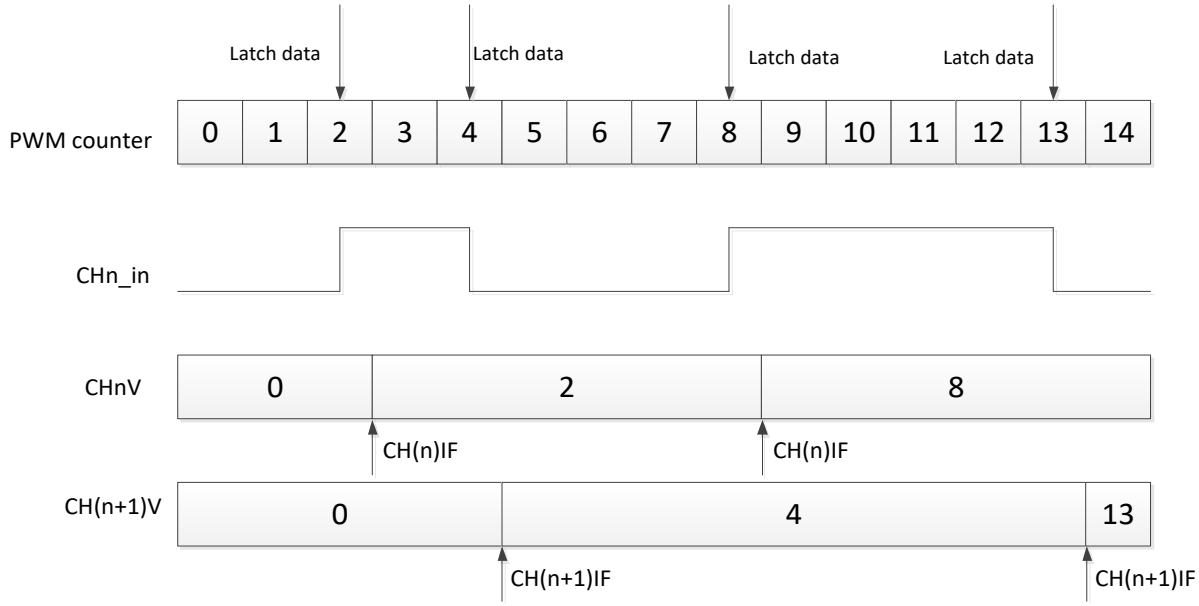

图 11-26 双边沿捕获模式图

11.4.10 正交解码模式

QDIEN = 1 使能正交解码器模式。正交解码器模式采用输入信号相位 A 和相位 B 控制 PWM 计数器递增和递减。每个输入信号相位 A 和相位 B 都有一个滤波器，该滤波器和通道输入中使用的滤波器是相同的，相位 A 的滤波值由 CH0CAPFVAL[4:0]位定义，相位 B 的滤波值由 CH1CAPFVAL [4:0]位定义，值为 0 时禁用滤波器(CH(n)CAPFVAL[4:0]位在 PWM_CAPFILTER 寄存器中)。

PHAPOL 位选择 A 相输入的极性，PHBPOL 位选择 B 相输入的极性。QUADMODE 选择正交解码器使用的编码模式。如果 QUADMODE = 1，则使能计数和方向编码模式，参见下图。在该模式下，相位 B 的输入表示计数方向，相位 A 的输入定义计数频率。当相位 A 输入信号有上升沿时，PWM 计数器将被更新。

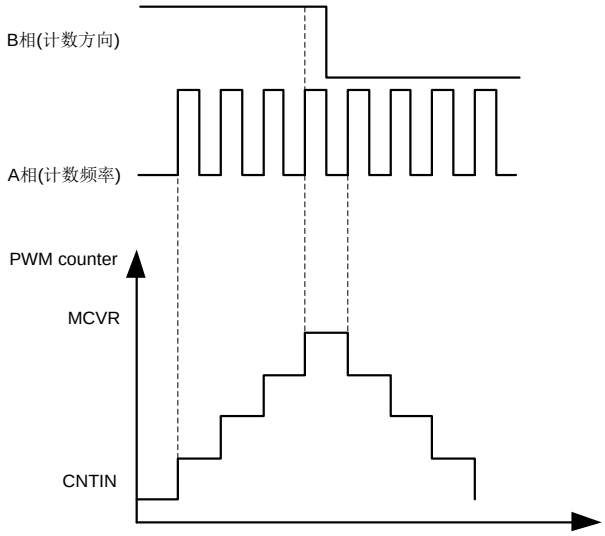

图 11-27 计数和方向编码模式

如果 QUADMODE = 0，则启用相位 A 和相位 B 编码模式，参见下图。在这种模式下，相位 A 和 B 信号之间的关系表示计数方向，相位 A 和 B 信号定义计数频率。当相位 A 或相位 B 信号有边沿时，PWM 计数器将被更新。

如果 PHAPOL = 0 & PHBPOL = 0，则 PWM 计数器递增发生在以下情况：

- A 相信号上升沿时，B 相信号为低电平；
- B 相信号上升沿时，A 相信号为高电平；
- B 相信号下降沿时，A 相信号为低电平；
- A 相信号下降沿时，B 相信号为高电平。

PWM 计数器递减发生在以下情况：

- A 相信号下降沿时，B 相信号为低电平；
- B 相信号下降沿时，A 相信号为高电平；
- B 相信号上升沿时，A 相信号为低电平；
- A 相信号上升沿时，B 相信号为高电平。

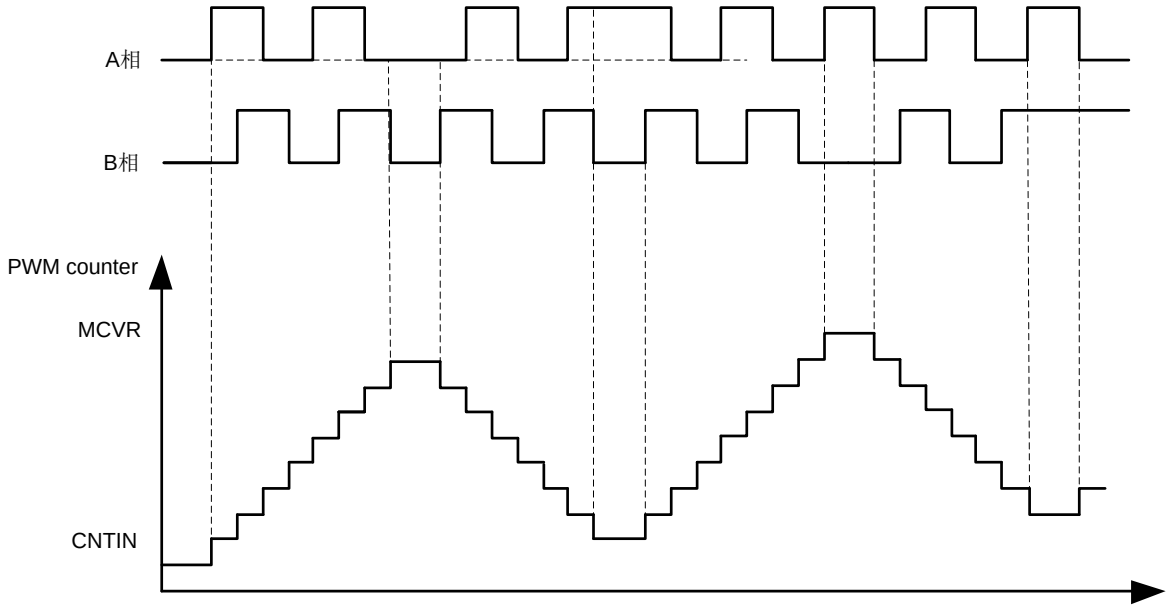

图 11-28 A 相和 B 相编码模式

下图显示了向上计数时 PWM 计数器溢出。当 PWM 计数器从 MCVR 更改为 CNTIN 时，设置 CNTOF 和 CNTOFDIR 位。CNTOF 位表示发生了 PWM 计数器溢出，CNTOFDIR 指示 PWM 计数器在向上计数时发生溢出。

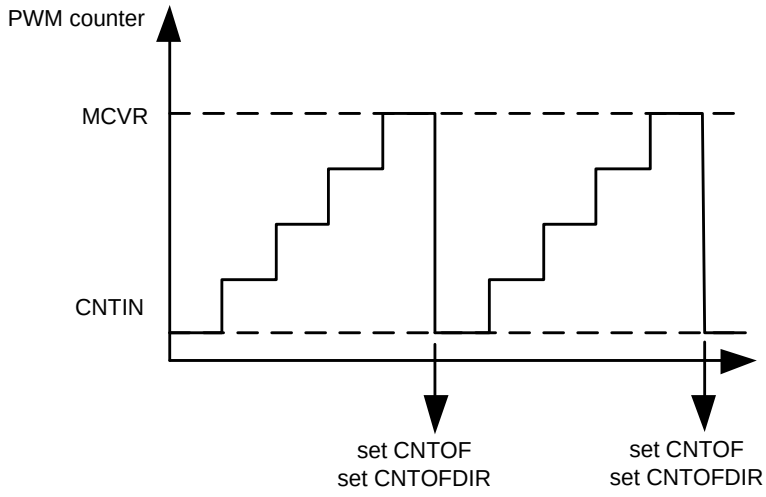

图 11-29 向上计数 PWM counter 溢出

下图显示了向下计数时 PWM 计数器溢出。当 PWM 计数器从 CNTIN 更改为 MCVR 时，将设置 CNTOF 位和清除 CNTOFDIR 位。CNTOF 位表示发生了 PWM 计数器溢出，CNTOFDIR 指示 PWM 计数器在向下计数时发生溢出。

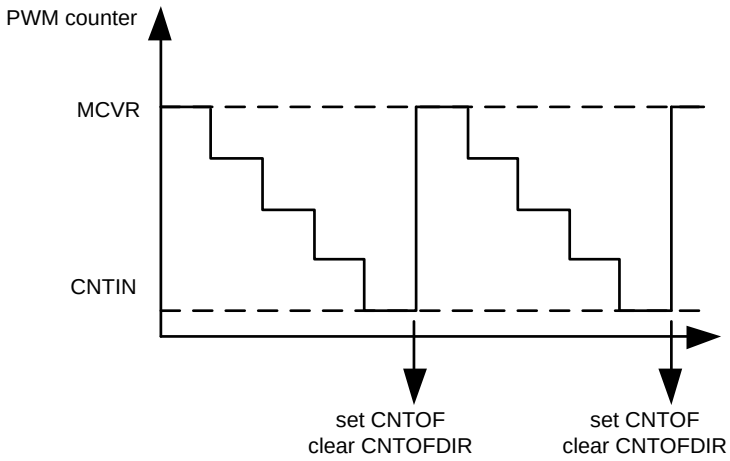

图 11-30 向下计数 PWM counter 溢出

11.4.11 写保护

在一些安全等级要求高的应用场景(如电机控制)，错误修改寄存器配置会引起系统异常，容易导致电机损坏。写保护功能可提供对某些关键寄存器位写入保护。用户在完成初始化配置后，可通过设置 PWM_FDSR[WPEN] = 1 激活写保护功能，避免人为或其它异常引起的对关键寄存器位误操作。设置 PWM_FUNCSEL[WPDIS] = 1 关闭写保护功能，可重新写入。支持写保护功能的位在 11.5 寄存器定义章节中进行说明。

11.4.12 初始化

向 INIT 位写入 1 时，初始化将强制通道(n)输出为 PWM_OUTINIT[CHnOIV]位的值。初始化特性只能在禁用 PWM 计数器时设置使用，当 PWM 计数器开始工作时 INIT 位自动清除，初始化输出一直保持直至 PWM 通道开始接管控制。

11.4.13 极性控制

CHnPOL 位选择通道(n) 输出极性：

- CHPOLCR[CHnPOL] = 0，通道(n) 输出极性为高，逻辑 1 为有效状态，逻辑 0 为无效状态
- CHPOLCR[CHnPOL] = 1，通道(n) 输出极性为低，逻辑 0 为有效状态，逻辑 1 为无效状态

11.4.14 输出屏蔽

输出屏蔽可用于通过软件强制通道输出为各自的无效状态。如果 CHnOMEN = 1，则通道(n)输出强制为通道的无效状态（PWM_CHOPOLCR[CHnPOL]位值）。

11.4.15 软件输出控制

软件输出控制可在 PWM 输出过程中强制通道输出软件定义值。CH(n)SWEN 位使能软件输出控制，CH(n)SWCV 设置通道强制输出值。非组合模式，软件输出功能每个通道可单独设置，组合模式软件输出功能如下表。

表 11-2 组合模式软件输出控制行为

CH(n)SWEN	CH(n+1)SWEN	CH(n)SWCV	CH(n+1)SWCV	Channel(n)输出	Channel(n+1)输出
0	X	X	X	关闭软件控制功能	关闭软件控制功能
1	X	0	0	0	0
1	X	0	1	0	1
1	X	1	0	1	0
1	X	1	1	1	0 或 1（注 2)

注 1：X 为 0 或 1 任意值。

注 2：PAIRnCOMPEN 位为 0 时，输出为 1；PAIRnCOMPEN 位为 1 时，输出为 0。

11.4.16 初始化触发器

如果 INITTRIGEN = 1，则在以下情况中 PWM 计数器更新为 PWM_CNTIN 寄存器值时，生成触发。

- PWM 计数器通过所选计数模式自动更新为 PWM_CNTIN 寄存器值
- 对 PWM_CNT 寄存器执行写操作
- PWM 计数器同步

11.4.17 通道匹配触发器

CHnTRIG = 1，在发生通道(n) 匹配(PWM 计数器 = CH(n)V)时，PWM 产生通道触发。通道触发输出提供用于片上模块的触发信号。

在向上-向下组合模式，通道值在向上计数产生一次匹配，向下计数也会产生一次匹配。为了便于控制通道匹配触发，可通过设置匹配生效点 PWM_CHnSCR[DIR]是在向上计数还是向下计数时作用。

11.4.18 故障控制

PWM 提供 3 个故障输入源：1 个来自芯片内部，2 个来自外部管脚输入。

FERnEN 位使能故障输入 n，FFnEN 位使能故障输入 n 滤波器。 FFVAL 位选择已使能的每个故障输入滤波器的值。

如果故障控制和故障输入 n 已使能，且在故障输入 n 信号上检测到有效边沿，则表明已出现故障状况且 FAULTDFn 位已置位。 FAULTDF 位是 FAULTDFn[2:0]位的逻辑或(OR)。

如果已使能故障控制(FAULTMODE[1:0] ≠ 0:0)、使能故障输入且已检测到故障，则强制 PWM 通道输出为各自安全值：

- 通道(n)输出采用 CHnPOL 的值
- 通道(n+1)采用 CH(n+1)POL 的值

当(FAULTDF = 1)且(FAULTIE = 1)时，生成故障中断。

表 11-3 故障源编号表

故障输入编号	故障源	说明
FAULT0	ACMP0_OUT	内部故障输入
FAULT1	PWMx_FLT0	外部引脚故障输入
FAULT2	PWMx_FLT1	外部引脚故障输入

11.4.18.1 自动故障清除

如果选择自动故障清除模式（FAULTMODE[1：0]=1：1），则当故障输入信号(FAULTIN)为零且新的 PWM 周期开始时，被故障控制禁用的通道输出将恢复正常输出。

11.4.18.2 手动故障清除

如果选择手动故障清除 (FAULTMODE[1：0]=0：1 或 1：0)，则当 FAULTDF 位被清除且新的 PWM 周期开始时，被故障控制禁用的通道输出将恢复正常输出。

11.4.18.3 故障输入极性

FLTnPOL 位选择故障输入 n 的极性：

- FLTnPOL = 0，故障 n 输入极性为高，故障输入 n 处的逻辑 1 代表一个故障
- FLTnPOL = 1，故障 n 输入极性为低，故障输入 n 处的逻辑 0 代表一个故障

11.4.19 写缓冲更新的寄存器

11.4.19.1 PWM_CNTIN 寄存器更新缓存

表 11-4 PWM_CNTIN 寄存器更新缓存

条件	寄存器更新时刻
CLKSRC[1:0] = 0:0	往 PWM_CNTIN 寄存器采取写入操作时
CLKSRC[1:0] ≠ 0:0 & PWMSYNCEN = 0	往 PWM_CNTIN 采取写入操作之后的下一个系统时钟周期
CLKSRC[1:0] ≠ 0:0 & PWMSYNCEN = 1	参考 11.4.20.6 PWM_CNTIN 寄存器同步 章节

11.4.19.2 PWM_CH(n)V 寄存器更新缓存

表 11-5 PWM_CH(n)V 寄存器更新缓存

条件	寄存器更新时刻
CLKSRC[1:0]=0:0	往 PWM_CH(n)V 寄存器采取写入操作时
CLKSRC[1:0] ≠ 0:0 & PWMSYNCEN = 0	• 如果选择的模式为 EPWM，则 PWM 计数器从 MCVR 更改为 CNTIN 之后更新。 • 如果选择的模式为 CPWM，则 PMM 计数器从 MCVR 更改为(MCVR – 0x0001)之后更新。
CLKSRC[1:0] ≠ 0:0 & PWMSYNCEN = 1	参考 11.4.20.7 PWM_CH(n)V 和 PWM_CH(n+1)V 寄存器同步 章节

11.4.19.3 PWM_MCVR 寄存器更新缓存

表 11-6 PWM_MCVR 寄存器更新缓存

条件	寄存器更新时刻
CLKSRC[1:0]=0:0	往 PWM_MCVR 寄存器采取写入操作时
CLKSRC[1:0] ≠ 0:0 & PWMSYNCEN = 0	• 如果选择的模式为 EPWM，则 PWM 计数器从 MCVR 更改为 CNTIN 之后更新。 • 如果选择的模式为 CPWM，则 PMM 计数器从 MCVR 更改为(MCVR – 0x0001)之后更新。
CLKSRC[1:0] ≠ 0:0 & PWMSYNCEN = 1	参考 11.4.20.4 PWM_MCVR 寄存器同步 章节

11.4.20 PWM 同步

PWM 同步功能提供一个机制可将 PWM_MCVR，PWM_CNTIN，PWM_CHnV，PWM_OMCR，PWM_INVCR 和 PWM_CHOSWCR 寄存器对应的缓存值更新到寄存器，并设置 PWM 计数器为 PWM_CNTIN 寄存器值。

11.4.20.1 硬件触发器

当 TRIGn = 1 时，使能 PWM 模块的三个硬件触发信号输入，其中 n （0,1 或 2）分别对应于每个输入信号。硬件触发器输入 n 由总线时钟同步。

如果 HWTRIGMODESEL = 0，在使能的硬件触发输入处检测到上升沿时，将启动与硬件触发的 PWM 同步。当写入 0 或检测到触发 n 事件时，TRIGn 位被清零。

在这种情况下，如果启用了两个或更多硬件触发器（例如，TRIG0 和 TRIG1 = 1），并且仅发生触发 1 事件，则仅清零 TRIG1 位。如果触发事件与写设置 TRIGn 位一起发生，则启动同步，但由于写操作，TRIGn 位会保持置 1。

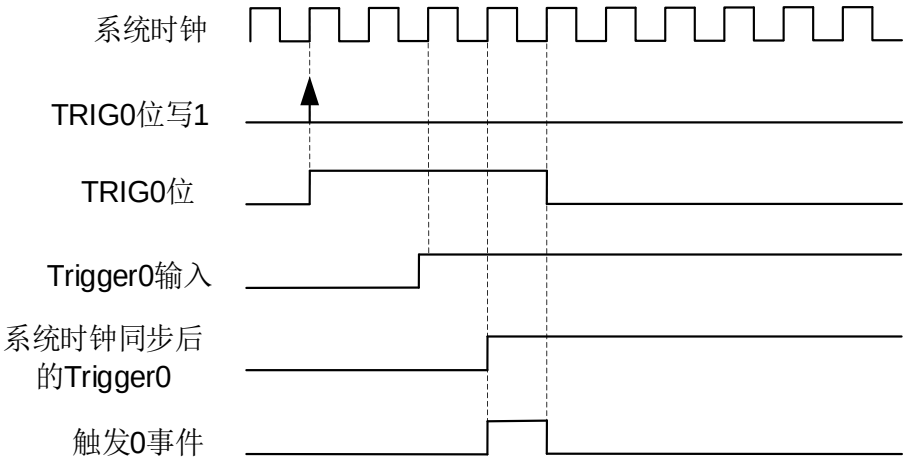

图 11-31 HWTRIGMODESEL = 0 硬件触发事件

11.4.20.2 软件触发器

当 PWM_SYNC[SWSYNC]位写入 1 时，会发生软件触发事件。当 SWSYNC 位写入 0 时，或当由软件事件启动的 PWM 同步完成时，该位被清零。

如果同时发生另一个软件触发事件（通过将另一个 1 写入 SWSYNC 位），则由先前软件触发事件启动的 PWM 同步结束，新的 PWM 同步开始，SWSYNC 位保持为 1。

SYNCMODE = 1，则根据 CNTVSWSYNC 位，SWSYNC 位也会被 PWM 清零。如果 CNTVSWSYNC=0，则软件触发事件发生后，在下一个选定的加载点清除 SWSYNC 位。如果 CNTVSWSYNC = 1，然后在软件触发事件发生时清除 SWSYNC 位。

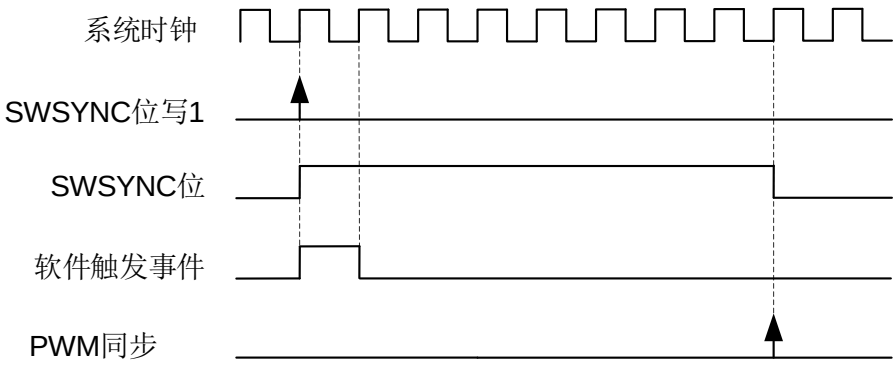

图 11-32 软件触发事件

11.4.20.3 边界周期和加载点

向上计数模式下，边界周期定义为计数器变为其初始值(CNTIN)的时候。向上-向下计数模式下，边界周期则定义为计数器从向下计数变为向上计数的时候以及从向上计数变为向下计数的时候。下图显示了寄存器的边界周期和加载点：

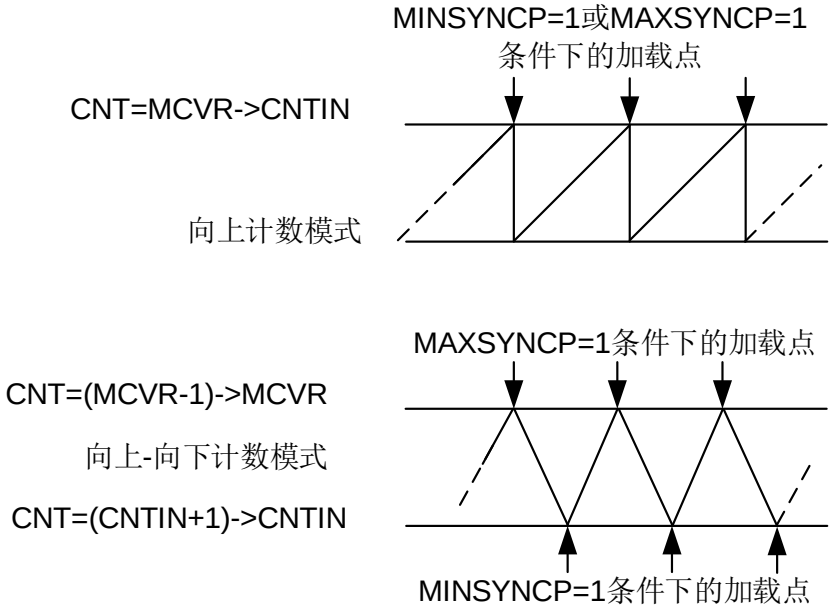

图 11-33 边界周期与加载点

向上计数模式中，MINSYNCP 或 MAXSYNCP 其中一位为 1，则使能加载点。向上-向下计数模式下，由 MINSYNCP 和 MAXSYNCP 位选择加载点。在这两种计数模式中，如果 MINSYNCP 和 MAXSYNCP 都不是 1，则边界周期不用作寄存器更新的加载点，即使有触发信号也不会产生寄存器同步（CNTVSWSYNC=0 条件下）。有关详细信息，请参见以下各节中的寄存器同步说明。

11.4.20.4 PWM_MCVR 寄存器同步

PWMSYNCEN = 1 使能 PWM_MCVR 寄存器同步功能，同步时将其缓存值更新至 PWM_MCVR 寄存器。

MCVR 寄存器同步流程图如下：

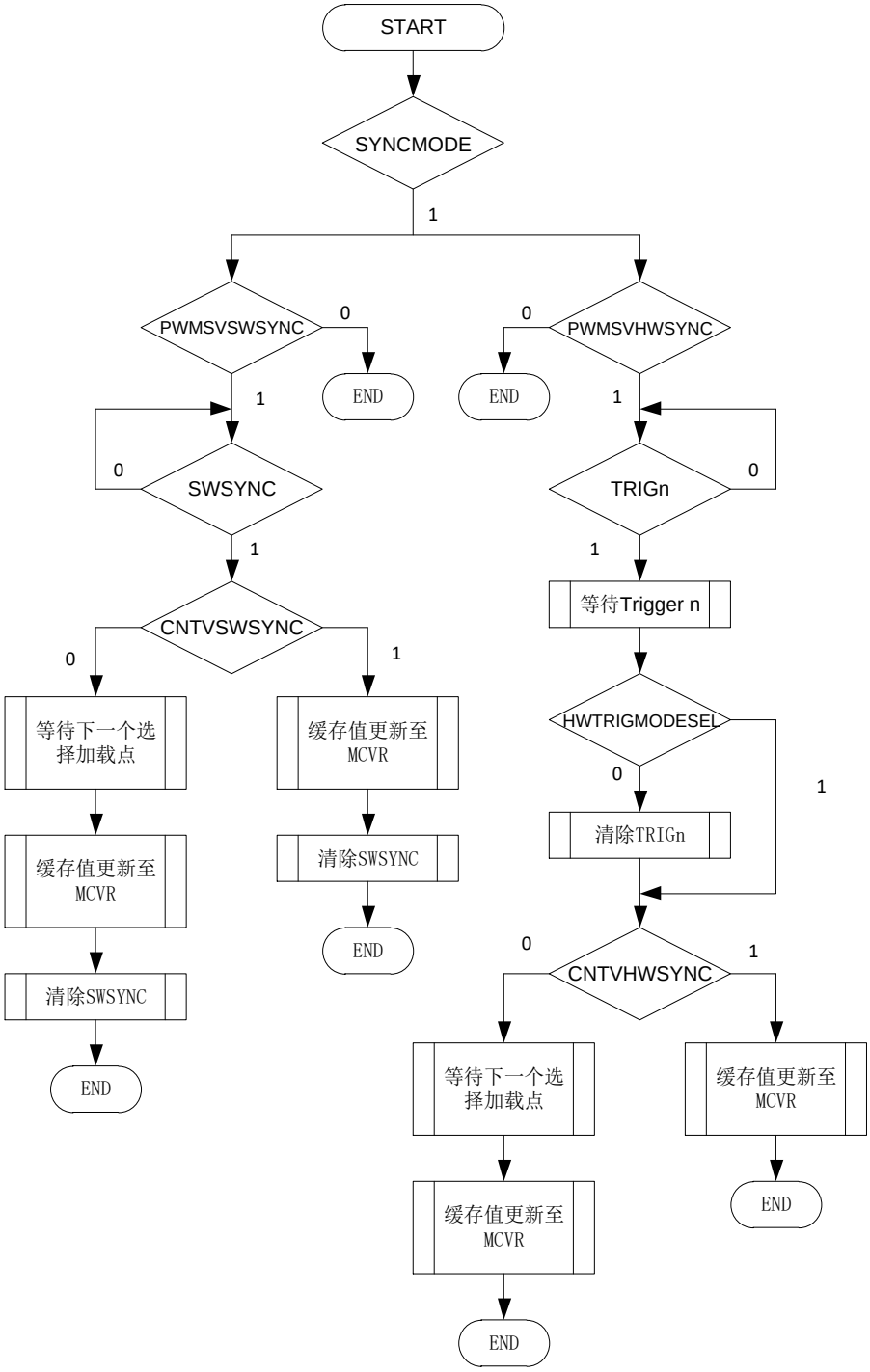

图 11-34 PWM_MCVR 寄存器同步流程

11.4.20.5 PWM_CNT 寄存器同步

PWM_CNT 寄存器同步功能可以在 PWM 周期中的特定点重新开始生成 PWM。通道输出强制为各自的初始值，PWM_CNT 寄存器强制为 PWM_CNTIN 寄存器定义的初始计数值。PWM_CNT 寄存器同步流程图如下：

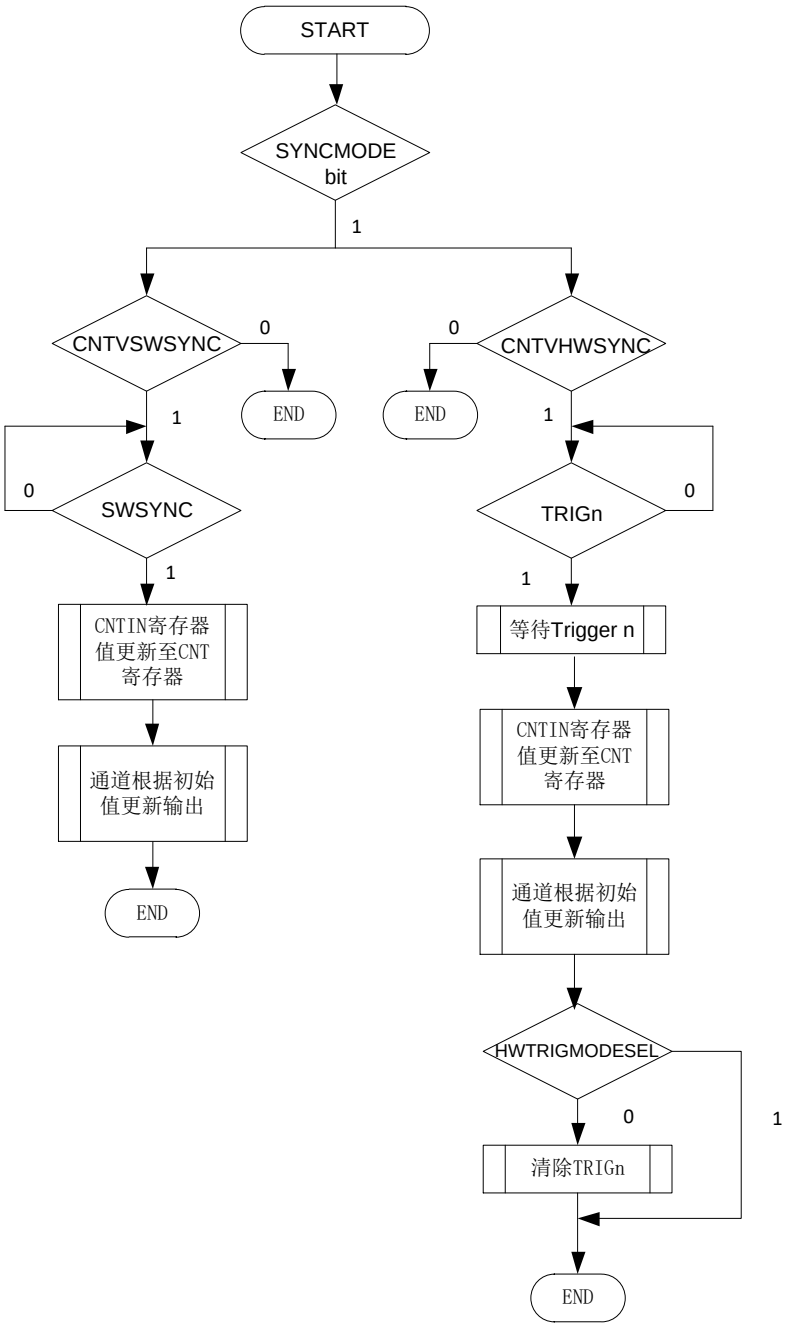

图 11-35 PWM_CNT 寄存器同步流程

11.4.20.6 PWM_CNTIN 寄存器同步

PWMSYNCEN = 1、 SYNCMODE = 1 且 CNTINC = 1，将使能这种同步。同步机制与 PWM_MCVR 寄存器同步相同，参照图 11-34 PWM_MCVR 寄存器同步流程。

11.4.20.7 PWM_CH(n)V 和 PWM_CH(n+1)V 寄存器同步

PWMSYNCEN = 1 且 PAIR(n)SYNCEN = 1，则会使能该同步。同步机制与 PWM_MCVR 寄存器同步流程相同，参照图 11-34 PWM_MCVR 寄存器同步流程。

11.4.20.8 PWM_OMCR 寄存器同步

PWM_OMCR 寄存器同步将其缓存值更新至 PWM_OMCR 寄存器。参照以下流程图：

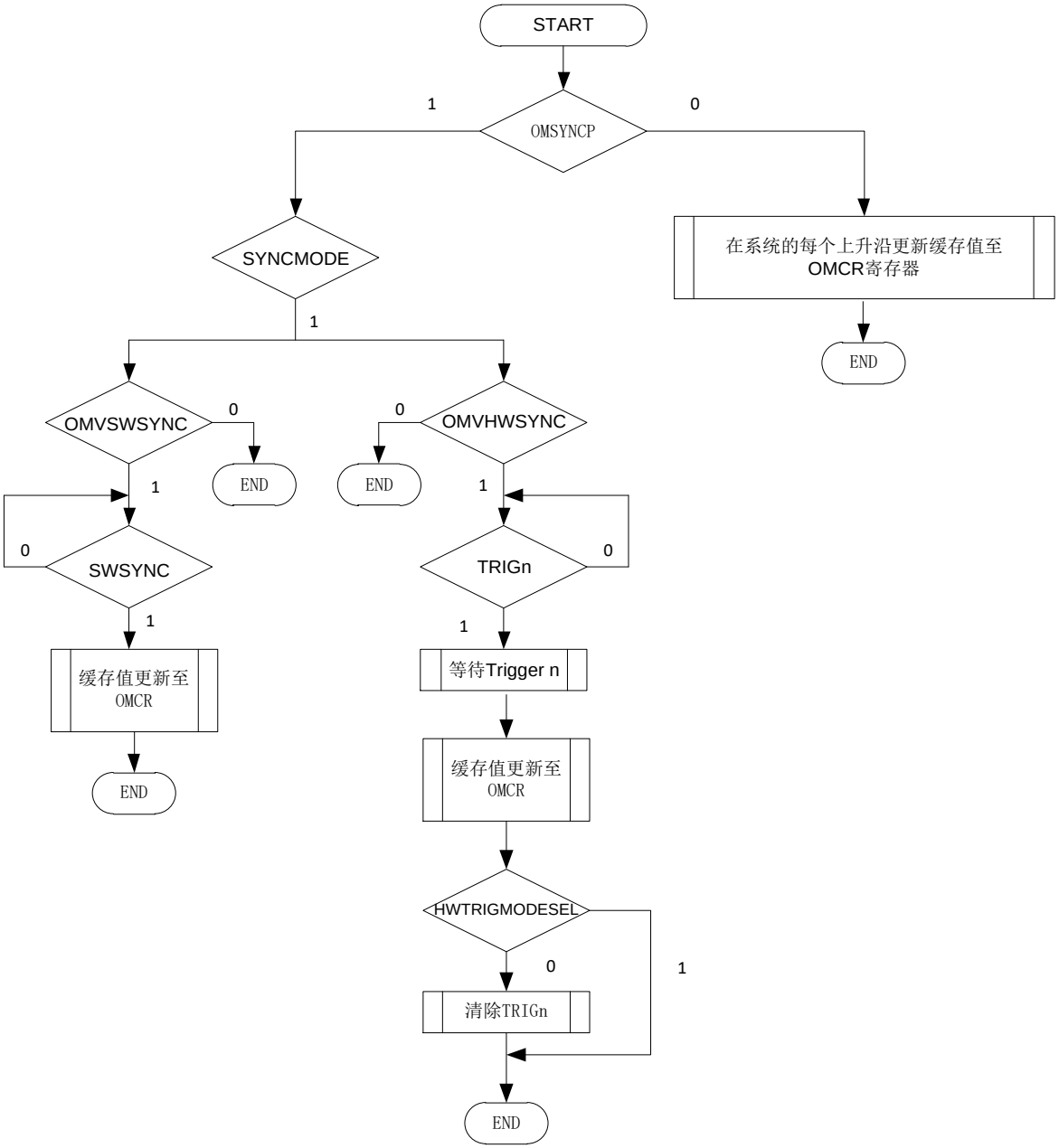

图 11-36 PWM_OMCR 寄存器同步流程

11.4.20.9 PWM_INVCR 寄存器同步

PWM_INVCR 寄存器同步将其缓存值更新至 PWM_INVCR 寄存器。参照以下流程图：

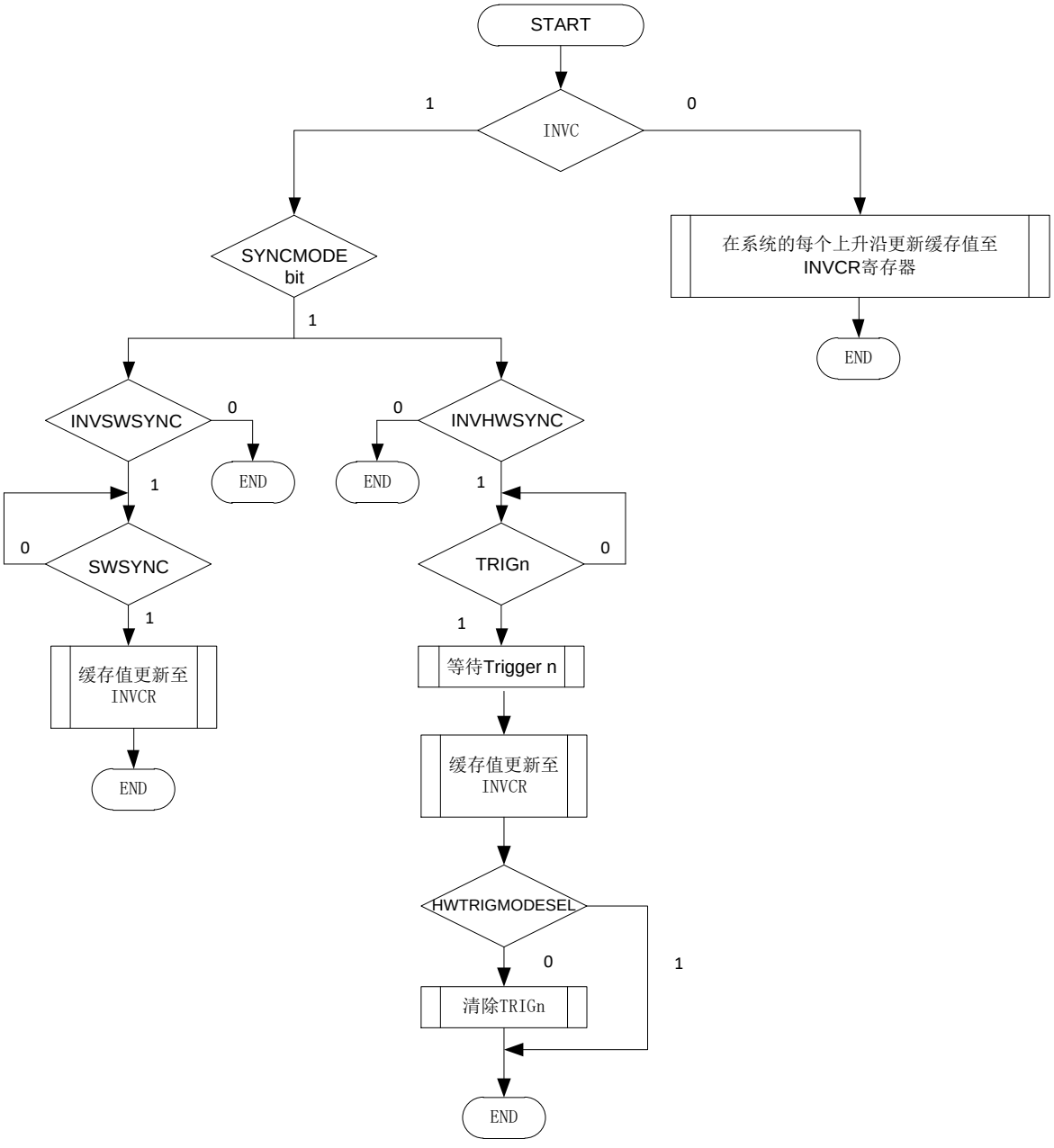

图 11-37 PWM_INVCR 寄存器同步流程

11.4.20.10 PWM_CHOSWCR 寄存器同步

PWM_CHOSWCR 寄存器同步将其缓存值更新至 PWM_CHOSWCR 寄存器。参照以下流程图：

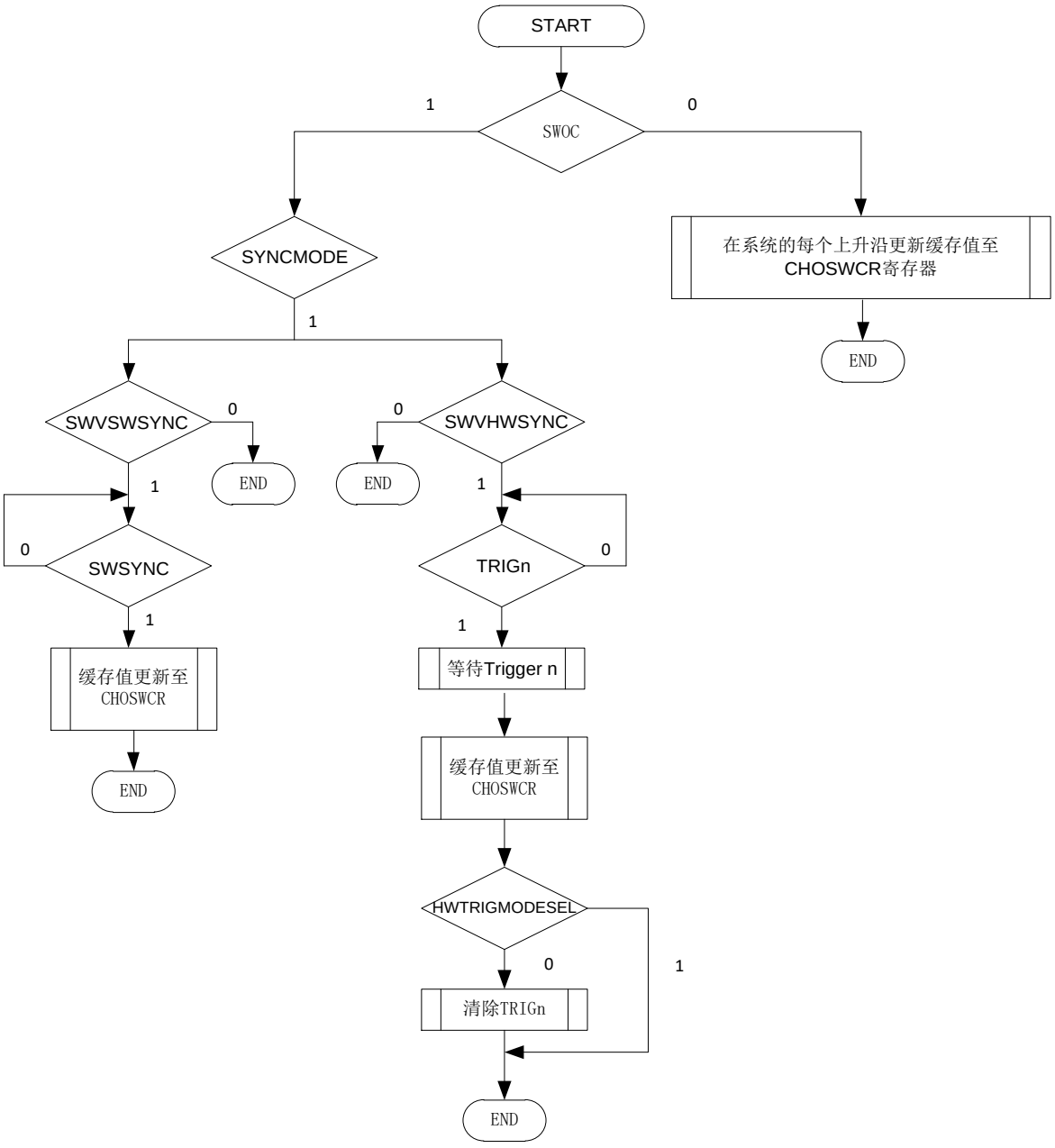

图 11-38 PWM_CHOSWCR 寄存器同步流程

11.4.20.11 PWM_CHOPOLCR 寄存器同步

PWM_CHOPOLCR 寄存器同步将其缓存值更新至 PWM_CHOPOLCR 寄存器。参照以下流程图：

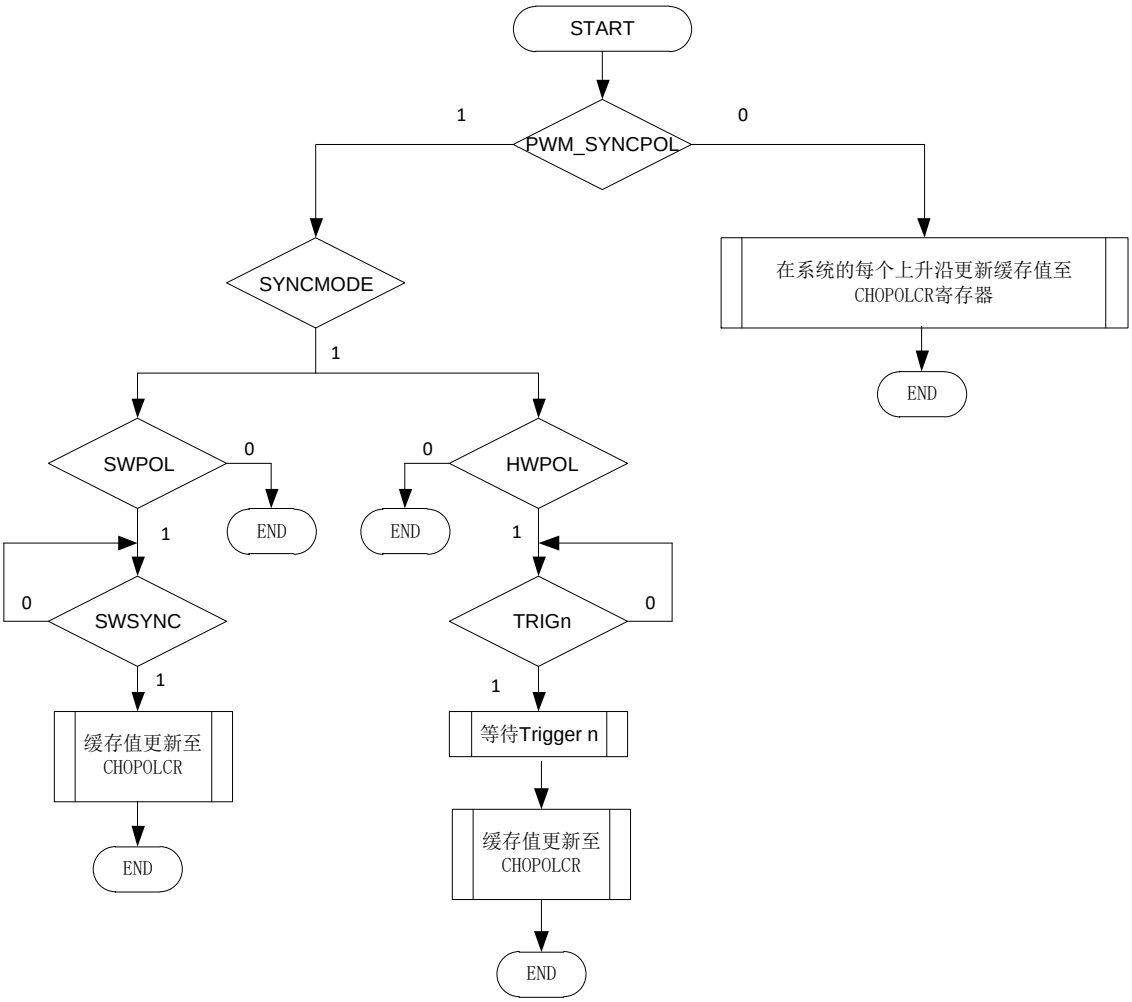

图 11-39 PWM_CHOPOLCR 寄存器同步流程

11.4.21 特性优先级

下图展示了生成通道(n)和(n+1)输出信号时所用特性的优先级。

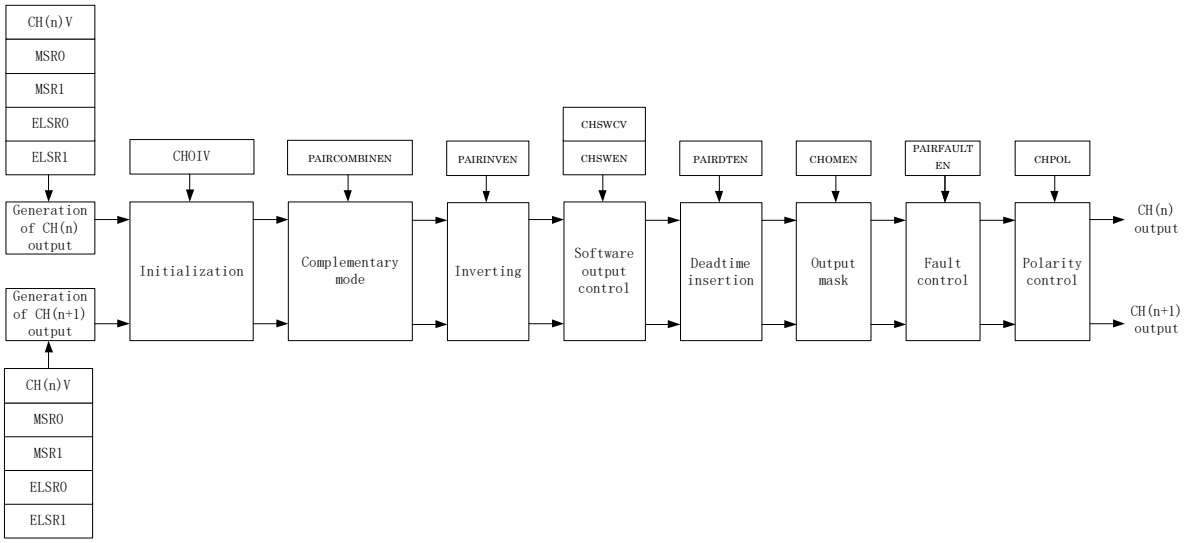

图 11-40 特性优先级

11.4.22 全局时基

全局时基(GTB)允许在同一时刻启动芯片上的多个 PWM 模块同时工作，达到多个 PWM 模块同步开始计数的目的。

GTB 功能可通过 PWM_CONF 寄存器中的 GTBEEN 和 GTBEOUT 位实现。

要使能 GTB 特性，请对每个参与的 PWM 模块执行以下步骤：

1. 停止 PWM 计数器：向 PWM_INIT[CLKSRC]写入 00b；
2. 向 PWM_CONF[GTBEEN]写入 1，同时向 PWM_CONF[GTBEOUT]写入 0；
3. 在 PWM_INIT[CLKSRC]中选择 PWM 计数器时钟源；
4. 向 PWM_CNT 寄存器中写入任意值复位 PWM 计数器；
5. PWM_CONF[GTBEOUT]写入 1，启动多个 PWM 模块同时工作。

11.4.23 PWM 中断

- 当 CNTOIE=1 且 CNTOF=1 时，产生计数溢出中断。
- 当 CHnIE=1 且 CHnIF = 1 时，产生通道(n) 中断
- 当 FAULTIE =1 且 FAULTDF = 1 时，产生故障中断

11.5 寄存器定义

表 11-7 PWM 寄存器映射

PWM0 基地址 = 0x40013000

PWM1 基地址 = 0x40014000

地址	名称	宽度	描述
PWMx 基地址+0x00	PWM_INIT	32	初始化寄存器
PWMx 基地址+0x04	PWM_CNT	32	计数器寄存器
PWMx 基地址+0x08	PWM_MCVR	32	最大计数值寄存器
PWMx 基地址+0x0C	PWM_CH0SCR	32	Channel (0) 状态和控制寄存器
PWMx 基地址+0x10	PWM_CH0V	32	Channel (0)值
PWMx 基地址+0x14	PWM_CH1SCR	32	Channel (1) 状态和控制寄存器
PWMx 基地址+0x18	PWM_CH1V	32	Channel (1) 值
PWMx 基地址+0x1C	PWM_CH2SCR	32	Channel (2) 状态和控制寄存器
PWMx 基地址+0x20	PWM_CH2V	32	Channel (2) 值
PWMx 基地址+0x24	PWM_CH3SCR	32	Channel (3) 状态和控制寄存器
PWMx 基地址+0x28	PWM_CH3V	32	Channel (3) 值
PWMx 基地址+0x2C	PWM_CH4SCR	32	Channel (4) 状态和控制寄存器
PWMx 基地址+0x30	PWM_CH4V	32	Channel (4) 值
PWMx 基地址+0x34	PWM_CH5SCR	32	Channel (5) 状态和控制寄存器
PWMx 基地址+0x38	PWM_CH5V	32	Channel (5) 值
PWMx 基地址+0x3C	PWM_CH6SCR	32	Channel (6) 状态和控制寄存器
PWMx 基地址+0x40	PWM_CH6V	32	Channel (6) 值
PWMx 基地址+0x44	PWM_CH7SCR	32	Channel (7) 状态和控制寄存器
PWMx 基地址+0x48	PWM_CH7V	32	Channel (7) 值
PWMx 基地址+0x4C	PWM_CNTIN	32	计数器初始值寄存器
PWMx 基地址+0x50	PWM_STR	32	捕获和比较状态寄存器
PWMx 基地址+0x54	PWM_FUNCSEL	32	功能选择寄存器
PWMx 基地址+0x58	PWM_SYNC	32	同步寄存器
PWMx 基地址+0x5C	PWM_OUTINIT	32	通道输出的初始状态寄存器
PWMx 基地址+0x60	PWM_OMCR	32	输出屏蔽控制寄存器
PWMx 基地址+0x64	PWM_MODESEL	32	模式选择寄存器
PWMx 基地址+0x68	PWM_DTSET	32	死区设置寄存器
PWMx 基地址+0x6C	PWM_EXTTRIG	32	外部触发器
PWMx 基地址+0x70	PWM_CHOPOLCR	32	通道输出极性控制寄存器
PWMx 基地址+0x74	PWM_FDSR	32	故障检测状态寄存器
PWMx 基地址+0x78	PWM_CAPFILTER	32	输入捕获滤波器控制
PWMx 基地址+0x7C	PWM_FFAFER	32	故障滤波和使能寄存器
PWMx 基地址+0x80	PWM_QDI	32	正交解码控制和状态寄存器

地址	名称	宽度	描述
PWMx 基地址+0x84	PWM_CONF	32	配置寄存器
PWMx 基地址+0x88	PWM_FLTPOL	32	故障输入极性寄存器
PWMx 基地址+0x8C	PWM_SYNCONF	32	同步配置寄存器
PWMx 基地址+0x90	PWM_INVCR	32	反相控制寄存器
PWMx 基地址+0x94	PWM_CHOSWCR	32	通道软件输出控制寄存器

【说明】上表中，x=0,1。

11.5.1 初始化寄存器(PWM_INIT)

表 11-8 PWM_INIT 寄存器

PWM_INIT **PWM 初始化寄存器** **Reset:00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称									CLKPSC[15：8]							
访问									RW							
Reset									0							
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	CLKPSC[7：0]								CNTOF	CNTOIE	CNTMODE	CLKSRC				
访问	RW								W0C	RW	RW	RW				
Reset	0								0							

字段	说明
23：8 CLKPSC	**PWM CLK 预分频器** 在将新值更新为寄存器位后，新的预分频因子会影响下一个总线时钟周期的时钟源。该字段为写保护。它只能在 FUNCSEL[WPDIS] = 1 时写入。
7 CNTOF	**定时器溢出标志** 0： PWM 计数器未溢出 1： PWM 计数器溢出 当 PWM 计数器值达到 MCVR 寄存器中值时置位。当 CNTOF 置 1 时，通过读取 INIT 寄存器，然后将 0 写入 CNTOF 位。如果在读和写操作之间发生另一次 PWM 溢出， 则清除操作无效，CNTOF 仍保持置位。
6 CNTOIE	**定时器溢出中断使能** 0： 禁用 CNTOF 中断，使用软件轮询 1： 使能 CNTOF 中断，当 CNTOF 置位时，产生中断 使能 PWM 溢出中断

字段	说明
5 CNTMODE	**计数器工作模式** 0： PWM 计数器以向上计数模式工作 1： PWM 计数器以向上-向下计数模式工作 选择计数器工作模式。该字段写保护，它只能在 FUNCSEL[WPDIS] = 1 时写入。
4：3 CLKSRC	**时钟源选择** 00：没有选择任何时钟。禁用 PWM 计数器 01：总线时钟 10： HSI 时钟 11：预留 选择 PWM 计数器时钟源。该字段为写保护，它仅在 FUNCSEL[WPDIS] = 1 时可写。

11.5.2 计数器寄存器(PWM_CNT)

表 11-9 PWM_CNT 寄存器

PWM_CNT **PWM 计数器值** **Reset:00000000**

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

字段	说明
15：0 COUNT	**PWM 计数器的值** CNT 寄存器包含 PWM 计数器值。 Reset 清除 CNT 寄存器。将任何值写入 COUNT 都会使用其初始值 CNTIN 更新计数器。

11.5.3 最大计数值寄存器(PWM_MCVR)

表 11-10 PWM_MCVR 寄存器

PWM_MCVR	最大计数值寄存器	Reset:00000000

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

字段	说明
	最大计数值寄存器
15：0 MCVR	MCVR 寄存器包含 PWM 计数器的模数值。当 PWM 计数值达到 MCVR 值后，溢出标志(CNTOF) 在下一个时钟置起，计数器的下一个值 取决于所选的计数方法。写入 MCVR 寄存器会将值锁存到缓冲区中。根据从写缓冲区更新的寄存器， MCVR 寄存器使用其写缓冲区的值进行更新。在写入 MCVR 寄存器之前，通过写入 CNT 来初始化 PWM 计数器。

11.5.4 通道状态和控制寄存器(PWM_CHnSCR)

表 11-11 PWM_CHnSCR 寄存器

PWM_CHnSCR	Channel (n) 状态和控制寄存器	Reset:00000000

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									CHIF	CHIE	MSR1	MSR0	ELSR1	ELSR0	DIR	
访问									W0C	RW	RW	RW	RW	RW	RW	
Reset									0	0	0	0	0	0	o	

字段	说明
7 CHIF	**通道中断标志** 在通道上发生事件时由硬件置位。通过读取 CHSCR 寄存器，然后将 CHIF 位写 0 来清除 CHIF 位。向 CHIF 中写 1 不起作用。如果在读取和写入操作间发生另一个事件，则清除操作无效，CHIF 仍保持置位，此时 CHIF 的中断请求也不会因为写清除操作而丢失。 0： 没有发生通道事件

字段	说明
	1： 通道事件已经发生
6 CHIE	**通道中断使能** 0： 禁用通道中断 1： 使能通道中断 使能通道中断
5 MSR1	**通道模式选择寄存器 1** 用于通道逻辑的进一步选择，其功能取决于通道模式。该字段为写保护。它只能在 FUNCSEL[WPDIS] = 1 时可写入。
4 MSR0	**通道模式选择寄存器 0** 用于进一步选择通道逻辑，其功能取决于通道模式。该字段为写保护。它只能在 FUNCSEL[WPDIS] = 1 时可写入。
3 ELSR1	**边沿或 电平选择寄存器 1** ELSR1 和 ELSR0 的功能依赖于通道模式。该字段写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写入。
2 ELSR0	**边沿或 电平选择寄存器 0** ELSR1 和 ELSR0 的功能依赖于通道模式。该字段写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写入。
1 DIR	**匹配点方向** 0：在向下计数过程中匹配点生效 1：在向上计数过程中匹配点生效 仅用于向上-向下计数组合模式，用于选择匹配生效点方向

11.5.5 通道值寄存器(PWM_CHnV)

表 11-12 PWM_CHnV 寄存器

PWM_CHnV **Channel (n) 值** **Reset:00000000**

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

字段	说明
15：0 CHCVAL	**通道计数值** 这些寄存器包含输入模式的捕获 PWM 计数器值或输出模式的匹配值。在输入捕获、捕获测试和双边沿捕获模式下，忽略对 CHnV 寄存器的任何写入操作。在输出模式下，写入 CHnV 寄存器会将值锁存在缓冲区中。根据从写缓冲区更新的寄存器，使用其写缓冲区的值更新 CHnV 寄存器。

11.5.6 计数器初始值寄存器(PWM_CNTIN)

表 11-13 PWM_CNTIN 寄存器

PWM_CNTIN **计数器初始值** **Reset:00000000**

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

字段	说明
15：0 CNTINIT	**计数器初始值** 计数器初始值寄存器包含 PWM 计数器的初始值。写入 CNTIN 寄存器会将值锁存至缓冲区中。根据从写缓冲区更新的寄存器，使用其写缓冲区的值更新 CNTIN 寄存器。当最初选择 PWM 时钟时，通过 向 CLKS 位写入非 0 值，PWM 计数器以值 0x0000 开始。为避免这种行为，在第一次写入选择 PWM 时钟前，将新值写入 CNTIN 寄存器，然后通过在 CNT 寄存器中写入任意值，来初始化 PWM 计数器。

11.5.7 捕获和比较状态寄存器(PWM_STR)

表 11-14 PWM_STR 寄存器

PWM_STR 捕获和比较状态寄存器 **Reset:00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									CH7SF	CH6SF	CH5SF	CH4SF	CH3SF	CH2SF	CH1SF	CH0SF
访问									W0C	W0C	W0C	W0C	W0C	W0C	W0C	W0C
Reset									0	0	0	0	0	0	0	0

字段	说明
7 CH7SF	**通道 7 状态标志** 0： 没有发生通道事件 1： 已发生通道事件
6 CH6SF	**通道 6 状态标志** 0： 没有发生通道事件 1： 已发生通道事件
5 CH5SF	**通道 5 状态标志** 0： 没有发生通道事件 1： 已发生通道事件
4 CH4SF	**通道 4 状态标志** 0： 没有发生通道事件 1： 已发生通道事件
3 CH3SF	**通道 3 状态标志** 0： 没有发生通道事件 1： 已发生通道事件
2 CH2SF	**通道 2 状态标志** 0： 没有发生通道事件 1： 已发生通道事件
1 CH1SF	**通道 1 状态标志** 0： 没有发生通道事件 1： 已发生通道事件

字段	说明
0 CH0SF	**通道 0 状态标志** 0： 没有发生通道事件 1： 已发生通道事件

11.5.8 功能选择寄存器(PWM_FUNCSEL)

表 11-15 PWM_FUNCSEL 寄存器

PWM_FUNCSEL **功能选择** **Reset:00000004**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									FAULTIE	FAULTMODE			PWMSYNC	WPDIS	INIT	PWMSYNCEN
访问									RW	RW			RW	RW	RW	RW
Reset									0	0			0	1	0	0

字段	说明
7 FAULTIE	**错误中断使能** 0： 禁用故障控制中断 1： 使能故障控制中断 当 PWM 检测到故障，并使能 PWM 故障控制时，允许产生中断。
6：5 FAULTMODE	**故障控制模式** 00： 所有通道禁用故障控制. 01： 仅对偶数通道使能故障控制 (通道： 0, 2, 4 和 6)，所选模式为手动故障清除 10： 所有通道均支持故障控制，所选模式为手动故障清除 11： 所有通道均支持故障控制，所选模式为自动故障清除 定义 PWM 故障控制模式。该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。
3 PWMSYNC	**PWM 同步模式** 0： 没有限制，MCVR, CHnV, OMCR 和 PWM 计数器同步可以使用软件和硬件触发器 1： 软件触发器只能由 MCVR 和 CHnV 同步使用，硬件触发只能由 OMCR 和 PWM 计时器同步使用。 选择 MCVR，CHnV，OMCR 和 PWM 计数器同步可以使用的触发器。

字段	说明
2 WPDIS	**禁用写保护** 0：使能写保护 1：禁用写保护 只允许写 1，写 0 无效。当写保护使能时，具有写保护功能的位不能被写入。当写保护禁用时后，具有写保护功能的位可以写入。WPDIS 位是 WPEN 位取反。当 WPEN 位写入 1 时，可清除 WPDIS，即使能写保护。当 WPDIS 位写入 1 时，可清除 WPEN，即禁用写保护。
1 INIT	**初始化通道输出** 当 INIT 位写入 1 时，根据 OUTINIT 寄存器中相应位的状态，初始化通道输出。将 INIT 位置 0 不起作用。INIT 位始终读为 0。
0 PWMSYNCEN	**PWM 同步功能使能** 0：禁用寄存器同步功能 1：使能寄存器同步功能 该字段写保护，它仅能在 FUNCSEL[WPDIS] = 1 时可写。

11.5.9 同步寄存器(PWM_SYNC)

表 11-16 PWM_SYNC 寄存器

PWM_SYNC **同步寄存器** **Reset:00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称					PWM_SYNCPOL				SWSYNC	TRIG2	TRIG1	TRIG0	OMSYNCP		MAXSYNCP	MINSYNCP
访问					RW				RW	RW	RW	RW	RW		RW	RW
Reset					0				0	0	0	0	0		0	0

字段	说明
11 PWM_SYNCPOL	**PWM_SYNCPOL** 0： POL 寄存器在总线时钟的所有上升沿使用其缓冲区的值进行更新 1： POL 寄存器仅通过 PWM 同步单元更新其缓冲区的值

字段	说明
	选择何时使用缓冲区的值更新 CHOPOLCR 寄存器。
7 SWSYNC	**PWM 同步软件触发器** 0： 未选择软件触发 1： 选择软件触发 选择软件触发器作为 PWM 同步触发器 。将 1 写入 SWSYNC 位时，会发生软件触发。
6 TRIG2	**PWM 同步硬件触发器 2** 0： 触发禁用 1： 触发使能 PWM 通道 0 输出作为 PWM 同步硬件触发源，输入信号检测到上升沿时，产生硬件触发。
5 TRIG1	**PWM 同步硬件触发器 1** 0： 触发禁用 1： 触发使能 CTU 模块的 PWMTRIG 控制位作为 PWM 同步硬件触发源，输入信号中检测到上升沿时，产生硬件触发。
4 TRIG0	**PWM 同步硬件触发器 0** 0： 触发禁用 1： 触发使能 ACMP0 输出作为同步的硬件触发源，输入信号中检测到上升沿时，产生硬件触发。
3 OMSYNCP	**输出掩码同步** 0 ：OMCR 寄存器在总线时钟的所有上升沿使用其缓冲区的值进行更新 1：OMCR 寄存器仅通过 PWM 同步单元更新其缓冲区的值 选择何时使用缓冲区的值更新 OMCR 寄存器。
1 MAXSYNCP	**使能最大加载点** 0： 禁用最大加载点 1： 使能最大加载点 选择 PWM 同步的最大加载点。如果 MAXSYNCP 为 1，则所选加载点为 PWM 计数器达到其最大值（MCVR 寄存器）时。
0 MINSYNCP	**使能最小加载点** 0： 禁用最小加载点 1： 使能最小加载点

字段	说明
	选择 PWM 同步的最小加载点。如果 MINSYNCP 为 1，则所选加载点为 PWM 计数器达到其最小值（CNTIN 寄存器）时。

11.5.10 通道输出初始状态寄存器(PWM_OUTINIT)

表 11-17 PWM_OUTINIT 寄存器

PWM_OUTINIT 通道输出的初始状态 **Reset:00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									CH7OIV	CH6OIV	CH5OIV	CH4OIV	CH3OIV	CH2OIV	CH1OIV	CH0OIV
访问									RW	RW	RW	RW	RW	RW	RW	RW
Reset									0	0	0	0	0	0	0	0

字段	说明
7 CH7OIV	**通道 7 输出初始值** 0： 初始值为 0 1： 初始值为 1 进行初始化时，强制进入通道输出的值
6 CH6OIV	**通道 6 输出初始值** 0： 初始值为 0 1： 初始值为 1 进行初始化时，强制进入通道输出的值
5 CH5OIV	**通道 5 输出初始值** 0： 初始值为 0 1： 初始值为 1 进行初始化时，强制进入通道输出的值
4 CH4OIV	**通道 4 输出初始值** 0： 初始值为 0 1： 初始值为 1

字段	说明
	进行初始化时，强制进入通道输出的值
3 CH3OIV	**通道 3 输出初始值** 0： 初始值为 0 1： 初始值为 1 进行初始化时，强制进入通道输出的值
2 CH2OIV	**通道 2 输出初始值** 0： 初始值为 0 1： 初始值为 1 进行初始化时，强制进入通道输出的值
1 CH1OIV	**通道 1 输出初始值** 0： 初始值为 0 1： 初始值为 1 进行初始化时，强制进入通道输出的值
0 CH0OIV	**通道 0 输出初始值** 0： 初始值为 0 1： 初始值为 1 进行初始化时，强制进入通道输出的值

11.5.11 输出屏蔽控制寄存器(PWM_OMCR)

表 11-18 PWM_OMCR 寄存器

PWM_OMCR **输出屏蔽控制寄存器** **Reset:00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									CH7OMEN	CH6OMEN	CH5OMEN	CH4OMEN	CH3OMEN	CH2OMEN	CH1OMEN	CH0OMEN
访问									RW	RW	RW	RW	RW	RW	RW	RW
Reset									0	0	0	0	0	0	0	0

字段	说明
7 CH7OMEN	**通道 7 输出屏蔽** 0： 通道输出没有被屏蔽，继续正常运行 1： 通道输出被屏蔽，强制进入非活动状态 定义通道输出是否被屏蔽或取消屏蔽
6 CH6OMEN	**通道 6 输出屏蔽** 0： 通道输出没有被屏蔽，继续正常运行 1： 通道输出被屏蔽，强制进入非活动状态 定义通道输出是否被屏蔽或取消屏蔽
5 CH5OMEN	**通道 5 输出屏蔽** 0： 通道输出没有被屏蔽，继续正常运行 1： 通道输出被屏蔽，强制进入非活动状态 定义通道输出是否被屏蔽或取消屏蔽
4 CH4OMEN	**通道 4 输出屏蔽** 0： 通道输出没有被屏蔽，继续正常运行 1： 通道输出被屏蔽，强制进入非活动状态 定义通道输出是否被屏蔽或取消屏蔽
3 CH3OMEN	**通道 3 输出屏蔽** 0： 通道输出没有被屏蔽，继续正常运行 1： 通道输出被屏蔽，强制进入非活动状态 定义通道输出是否被屏蔽或取消屏蔽
2 CH2OMEN	**通道 2 输出屏蔽** 0： 通道输出没有被屏蔽，继续正常运行 1： 通道输出被屏蔽，强制进入非活动状态 定义通道输出是否被屏蔽或取消屏蔽
1 CH1OMEN	**通道 1 输出屏蔽** 0： 通道输出没有被屏蔽，继续正常运行 1： 通道输出被屏蔽，强制进入非活动状态 定义通道输出是否被屏蔽或取消屏蔽
0 CH0OMEN	**通道 0 输出屏蔽** 0： 通道输出没有被屏蔽，继续正常运行

字段	说明
	1： 通道输出被屏蔽，强制进入非活动状态 定义通道输出是否被屏蔽或取消屏蔽

11.5.12 模式选择寄存器(PWM_MODESEL)

表 11-19 PWM_MODESEL 寄存器

PWM_MODESEL **PWM 模式选择** **Reset:00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称		PAIR3FAULTEN	PAIR3SYNCEN	PAIR3DTEN	PAIR3DECAP	PAIR3DECAPEN	PAIR3COMPEN	PAIR3COMBINEN		PAIR2FAULTEN	PAIR2SYNCEN	PAIR2DTEN	PAIR2DECAP	PAIR2DECAPEN	PAIR2COMPEN	PAIR2COMBINEN
访问		RW	RW	RW	RW	RW	RW	RW		RW	RW	RW	RW	RW	RW	RW
Reset		0	0	0	0	0	0	0		0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称		PAIR1FAULTEN	PAIR1SYNCEN	PAIR1DTEN	PAIR1DECAP	PAIR1DECAPEN	PAIR1COMPEN	PAIR1COMBINEN		PAIR0FAULTEN	PAIR0SYNCEN	PAIR0DTEN	PAIR0DECAP	PAIR0DECAPEN	PAIR0COMPEN	PAIR0COMBINEN
访问		RW	RW	RW	RW	RW	RW	RW		RW	RW	RW	RW	RW	RW	RW
Reset		0	0	0	0	0	0	0		0	0	0	0	0	0	0

字段	说明
30 PAIR3FAULTEN	**故障控制使能（n = 6）** 0： 禁用此通道对中的故障控制 1： 使能此通道对中的故障控制 在通道 (n) 和 (n+1)间使能故障控制，该字段写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。

字段	说明
29 PAIR3SYNCEN	**同步使能（ n = 6）** 0：禁用此通道对中的 PWM 同步 1： 使能此通道对中的 PWM 同步 使能寄存器 CH(n)V 和 CH(n+1)V 的 PWM 同步。
28 PAIR3DTEN	**死区使能（ n = 6）** 0： 禁用此通道对中的死区插入 1： 使能此通道对中的死区插入 在 通道 (n) 和 (n+1)之间支持死区插入。该字段写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。
27 PAIR3DECAP	**双边沿捕获模式捕获（n = 6）** 0： 双边沿捕获无效 1： 双边沿捕获有效 根据通道 (n) 输入事件和双边沿捕获位的配置，使能 PWM 计时器值的捕获。该字段仅在 DECAPEN = 1 时适用。如果选择双边沿捕获单发模式，且发生捕获通道(n+1)的事件，则硬件将自动清除 DECAP 位。
26 PAIR3DECAPEN	**双边沿捕获模式使能（ n = 6）** 0：该通道对中的双沿捕捉模式禁用 1 ：该通道对中的双沿捕捉模式使能。 在通道(n) 和 (n+1)中支持双边沿捕获模式。该位在双边沿捕获模式下重新配置 MSnR0，ELSnR1:ELSnR0 和 ELS(n+1) R1:ELS(n+1)R0 位的功能。该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写入。
25 PAIR3COMPEN	**通道互补（n = 6）** 0 ： 通道(n+1) 输出和通道(n)输出相同 1 ： 通道(n+1) 输出是通道(n)输出互补 使能组合通道的互补模式。在互补模式下，通道 (n+1) 输出是通道(n) 输出取反。该字节为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。
24 PAIR3COMBINEN	**组合通道 （ n = 6）** 0 ：通道(n)和(n+1) 是独立的 1 ：通道(n)和(n+1) 是组合的 使能 通道 (n) 和 (n+1)的组合功能。该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。
22 PAIR2FAULTEN	**故障控制使能（n = 4）**

字段	说明
	0： 禁用此通道对中的故障控制 1： 使能此通道对中的故障控制 在通道 (n) 和 (n+1)间使能故障控制，该字段写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。
21 PAIR2SYNCEN	**同步使能（ n = 4）** 0 ：禁用此通道对中的 PWM 同步 1： 使能此通道对中的 PWM 同步 使能寄存器 CH(n)V 和 CH(n+1)V 的 PWM 同步。
20 PAIR2DTEN	**死区使能（ n = 4）** 0： 禁用此通道对中的死区插入 1： 使能此通道对中的死区插入 在 通道 (n) 和 (n+1)之间支持死区插入。该字段写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。
19 PAIR2DECAP	**双边沿捕获模式捕获（n = 4）** 0： 双边沿捕获无效 1： 双边沿捕获有效 根据通道 (n) 输入事件和双边沿捕获位的配置，使能 PWM 计时器值的捕获。该字段仅在 DECAPEN = 1 时适用。如果选择双边沿捕获单发模式，且发生捕获通道(n+1)的事件，则硬件将自动清除 DECAP 位。
18 PAIR2DECAPEN	**双边沿捕获模式使能（ n = 4）** 0：该通道对中的双沿捕捉模式禁用 1 ：该通道对中的双沿捕捉模式使能。 在通道(n) 和 (n+1)中支持双边沿捕获模式。该位在双边沿捕获模式下重新配置 MSnR0，ELSnR1:ELSnR0 和 ELS(n+1) R1:ELS(n+1)R0 位的功能。该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写入。
17 PAIR2COMPEN	**通道互补（n = 4）** 0 ： 通道(n+1) 输出和通道(n)输出相同 1 ： 通道(n+1) 输出是通道(n)输出互补 使能组合通道的互补模式。在互补模式下，通道 (n+1) 输出是通道(n) 输出取反。该字节为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。
16 PAIR2COMBINEN	**组合通道 （ n = 4）**

字段	说明
	0：通道(n)和(n+1) 是独立的 1：通道(n)和(n+1) 是组合的 使能 通道 (n) 和 (n+1)的组合功能。该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。
14 PAIR1FAULTEN	**故障控制使能（ n = 2）** 0：禁用此通道对中的故障控制 1：使能此通道对中的故障控制 使能 通道 (n) 和 (n+1)的故障控制。该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。
13 PAIR1SYNCEN	**同步使能（ n = 2）** 0：禁用此通道对中的 PWM 同步 1：使能此通道对中的 PWM 同步 使能寄存器 CH(n)V 和 CH(n+1)V 的 PWM 同步。
12 PAIR1DTEN	**死区使能（n = 2）** 0：禁用此通道对中的死区插入 1：使能此通道对中的死区插入 使能通道 (n) 和 (n+1)的死区插入，该字段写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。
11 PAIR1DECAP	**双边沿捕获模式捕获（ n = 2）** 0： 双边沿捕获无效 1： 双边沿捕获有效 根据通道 (n) 输入事件和双边沿捕获位的配置，使能 PWM 计时器值的捕获。该字段仅在 DECAPEN = 1 时适用。如果选择双边沿捕获单发模式，且发生捕获通道(n+1)的事件，则硬件将自动清除 DECAP 位。
10 PAIR1DECAPEN	**双边沿捕获模式使能 （ n = 2）** 0：该通道对中的双沿捕捉模式禁用 1：该通道对中的双沿捕捉模式使能。 在通道(n) 和 (n+1)中支持双边沿捕获模式。该位在双边沿捕获模式下重新配置 MSnR0，ELSnR1:ELSnR0 和 ELS(n+1) R1:ELS(n+1)R0 位的功能。它只能在 FUNCSEL[WPDIS] = 1 时可写入。
9 PAIR1COMPEN	**Channel (n) 互补（ n = 2）** 0： 通道(n+1) 输出和通道(n)输出相同

字段	说明
	1： 通道(n+1) 输出是通道(n)输出互补 使能组合通道的互补模式。在互补模式下，通道 (n+1) 输出是通道(n) 输出取反。该字节为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。
8 PAIR1COMBINEN	**组合通道 （ n = 2）** 0：通道(n)和(n+1) 是独立的 1：通道(n)和(n+1) 是组合的 使能通道 (n) 和 (n+1)的组合功能。该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。
6 PAIR0FAULTEN	**故障控制使能（ n = 0）** 0：禁用此通道对中的故障控制 1：使能此通道对中的故障控制 使能通道 (n) 和 (n+1)的故障控制。该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。
5 PAIR0SYNCEN	**同步使能 （ n = 0）** 0：禁用此通道对中的 PWM 同步 1： 使能此通道对中的 PWM 同步 使能寄存器 CH(n)V 和 CH(n+1)V 的 PWM 同步。
4 PAIR0DTEN	**死区使能 （ n = 0）** 0：禁用此通道对中的死区插入 1：使能此通道对中的死区插入 使能通道 (n)和(n+1)的死区插入，该字段写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。
3 PAIR0DECAP	**双边沿捕获模式捕获 （ n = 0）** 0： 双边沿捕获无效 1： 双边沿捕获有效 根据通道 (n) 输入事件和双边沿捕获位的配置，使能 PWM 计时器值的捕获。该字段仅在 DECAPEN = 1 时适用。如果选择双边沿捕获单发模式，且发生捕获通道(n+1)的事件，则硬件将自动清除 DECAP 位。
2 PAIR0DECAPEN	**双边沿捕获模式使能 （n = 0）** 0：该通道对中的双沿捕捉模式禁用 1：该通道对中的双沿捕捉模式使能

字段	说明
	在通道(n) 和 (n+1)中支持双边沿捕获模式。该位在双边沿捕获模式下重新配置 MSnR0，ELSnR1:ELSnR0 和 ELS(n+1) R1:ELS(n+1)R0 位的功能。该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写入。
1 PAIR0COMPEN	**Channel (n)互补（ n = 0）** 0： 通道(n+1) 输出和通道(n)输出相同 1： 通道(n+1) 输出是通道(n)输出互补 使能组合通道的互补模式。在互补模式下，通道 (n+1) 输出是通道(n) 输出取反。该字节为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。
0 PAIR0COMBINEN	**组合通道（ n = 0）** 0： 通道(n)和(n+1) 是独立的 1： 通道(n)和(n+1) 是组合的 使能通道 (n) 和 (n+1)的组合功能。该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。

11.5.13 死区插入控制寄存器(PWM_DTSET)

表 11-20 PWM_DTSET 寄存器

PWM_DTSET 死区插入控制 **Reset:00000000**

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

字段	说明
7：6 DTPSC	**死区预分频值** 0x： 总线时钟 1 分频 10： 总线时钟 4 分频 11： 总线时钟 16 分频 选择总线时钟分频因子。死区计数器使用此预分频时钟，该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。
5：0 DTVAL	**死区值** 选择死区计数器的死区插入值 。死区计数器由总线时钟的缩放版本计时。请参考 DTPS 说明。

字段	说明
	死区插入值 = (DTPSC x DTVAL)。 DTVAL 选择插入的死区计数值，如下所示： 当 DTVAL 为 0 时， 没有插入任何计数 当 DTVAL 为 1 时， 插入 1 个计数值 当 DTVAL 为 2 时，插入 2 个计数 这种模式可能持续到 63 个计数，该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。

11.5.14 外部触发器寄存器(PWM_EXTTRIG)

表 11-21 PWM_EXTTRIG 寄存器

PWM_EXTTRIG **PWM 外部触发器** **Reset:00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称							TRIGF	INITTRIGEN	CH7TRIG	CH6TRIG	CH5TRIG	CH4TRIG	CH3TRIG	CH2TRIG	CH1TRIG	CH0TRIG
访问							W0C	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset							0	0	0	0	0	0	0	0	0	0

字段	说明
9 TRIGF	**通道触发标志** 0： 没有生成通道触发 1： 产生通道触发 任意某个通道触发时置 1。 通过读取 EXTTRIG ， 然后将 0 写入 TRIGF 可清除 。如果在清除序列完成之前生成另一个通道触发，则清除操作无效，TRIGF 仍保持置位。
8 INITTRIGEN	**使能初始化触发器** 0： 禁用初始化触发 1： 使能初始化触发 当 PWM 计数器等于 CNTIN 寄存器时，允许产生触发。
7 CH7TRIG	**通道 7 触发使能** 0：禁用通道触发 1：使能通道触发

字段	说明
	当 PWM 计数器等于 CHnV 寄存器时， 允许产生通道触发。
6 CH6TRIG	**通道 6 触发使能** 0： 禁用通道触发 1： 使能通道触发 当 PWM 计数器等于 CHnV 寄存器时， 允许产生通道触发。
5 CH5TRIG	**通道 5 触发使能** 0： 禁用通道触发 1： 使能通道触发 当 PWM 计数器等于 CHnV 寄存器时， 允许产生通道触发。
4 CH4TRIG	**通道 4 触发使能** 0： 禁用通道触发 1： 使能通道触发 当 PWM 计数器等于 CHnV 寄存器时， 允许产生通道触发。
3 CH3TRIG	**通道 3 触发使能** 0： 禁用通道触发 1： 使能通道触发 当 PWM 计数器等于 CHnV 寄存器时， 允许产生通道触发。
2 CH2TRIG	**通道 2 触发使能** 0： 禁用通道触发 1： 使能通道触发 当 PWM 计数器等于 CHnV 寄存器时， 允许产生通道触发。
1 CH1TRIG	**通道 1 触发使能** 0： 禁用通道触发 1： 使能通道触发 当 PWM 计数器等于 CHnV 寄存器时， 允许产生通道触发。
0 CH0TRIG	**通道 0 触发使能** 0： 禁用通道触发 1： 使能通道触发 当 PWM 计数器等于 CHnV 寄存器时， 允许产生通道触发。

11.5.15 通道输出极性控制寄存器(PWM_CHOPOLCR)

表 11-22 PWM_CHOPOLCR 寄存器

PWM_CHOPOLCR 通道输出极性控制寄存器 Reset:00000000

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									CH7POL	CH6POL	CH5POL	CH4POL	CH3POL	CH2POL	CH1POL	CH0POL
访问									RW	RW	RW	RW	RW	RW	RW	RW
Reset									0	0	0	0	0	0	0	0

字段	说明
7 CH7POL	**通道 7 极性** 0： 通道极性为高电平有效 1： 通道极性为低电平有效 定义通道输出的极性，该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。
6 CH6POL	**通道 6 极性** 0： 通道极性为高电平有效 1： 通道极性为低电平有效 定义通道输出的极性，该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。
5 CH5POL	**通道 5 极性** 0： 通道极性为高电平有效 1： 通道极性为低电平有效 定义通道输出的极性，该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。
4 CH4POL	**通道 4 极性** 0： 通道极性为高电平有效 1： 通道极性为低电平有效 定义通道输出的极性，该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。
3	**通道 3 极性**

字段	说明
CH3POL	0： 通道极性为高电平有效 1： 通道极性为低电平有效 定义通道输出的极性，该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。
2 CH2POL	**通道 2 极性** 0： 通道极性为高电平有效 1： 通道极性为低电平有效 定义通道输出的极性，该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。
1 CH1POL	**通道 1 极性** 0： 通道极性为高电平有效 1： 通道极性为低电平有效 定义通道输出的极性，该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。
0 CH0POL	**通道 0 极性** 0： 通道极性为高电平有效 1： 通道极性为低电平有效 定义通道输出的极性，该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。

11.5.16 故障检测状态寄存器(PWM_FDSR)

表 11-23 PWM_FDSR 寄存器

PWM_FDSR **故障检测状态寄存器** **Reset:00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									FAULTDF	WPEN	FAULTIN			FAULTDF2	FAULTDF1	FAULTDF0
访问									W0C	RW	RO			W0C	W0C	W0C
Reset									0	0	0			0	0	0

字段	说明
7 FAULTDF	**故障检测标志**

字段	说明
	0： 未检测到故障 1： 检测到故障 FAULTDF j 位的逻辑或结果，其中 j = 2, 1, 0。在 FAULTDF 置位的情况下，当故障输入端不存在故障条件时，读取故障状态寄存器，向 FAULTDF 写入 0 可清零 FAULTDF。如果在清除序列完成前在一个已使能的故障输入中检测到另一个故障，则清除操作无效，FAULTDF 仍保持置位状态。当 FAULTDF j 位全部清除后，FAULTDF 也会被清 0。
6 WPEN	**写保护使能** 0： 禁用写保护， 写保护位可写 1： 使能写保护，写保护位不可写 只允许写 1，写 0 无效。当写保护使能时，具有写保护功能的位不能被写入。当写保护禁用时后，具有写保护功能的位可以写入。WPEN 位是 WPDIS 位取反。当 WPDIS 位写入 1 时，可清除 WPEN，即禁用写保护。当 WPENS 位写入 1 时，可清除 WPDIS，即使能写保护。
5 FAULTIN	**故障输入** 0： 使能故障输入的逻辑或为 0 1： 使能故障输入的逻辑或为 1 表示当使能故障控制后，在其过滤器（若已启用过滤器）后使能的故障输入的逻辑或。
2 FAULTDF2	**故障检测标志 2** 0：未检测到故障 1： 检测到故障 在 FAULTDF2 置 1 时，读取 FDSR 寄存器然后往 FAULTDF2 写 0 可清除。向 FAULTDF2 中写 1 不起作用。当 FAULTDF 位被清 0 时，FAULTDF2 位也会被清 0 。如果在清除序列完成前在相应的的故障输入中检测到另一个故障，则清除操作无效， FAULTDF2 仍保持置位状态。
1 FAULTDF1	**故障检测标志 1** 0： 未检测到故障 1： 检测到故障 在 FAULTDF1 置 1 时，读取 FDSR 寄存器然后往 FAULTDF1 写 0 可清除。向 FAULTDF1 中写 1 不起作用。当 FAULTDF 位被清 0 时，FAULTDF1 位也会被清 0 。如果在清除序列完成前在相应的的故障输入中检测到另一个故障，则清除操作无效， FAULTDF1 仍保持置位状态。
0 FAULTDF0	**故障检测标志 0** 0： 未检测到故障 1： 检测到故障

字段	说明
	在 FAULTDF0 置 1 时，读取 FDSR 寄存器然后往 FAULTDF0 写 0 可清除。当 FAULTDF 位被清 0 时，FAULTDF0 位也会被清 0 。如果在清除序列完成前在相应的的故障输入中检测到另一个故障，则清除无效， FAULTDF0 将保持置位状态。

11.5.17 输入捕获滤波器控制(PWM_CAPFILTER)

表 11-24 PWM_CAPFILTER 寄存器

PWM_CAPFILTER 输入捕获滤波器控制 **Reset:00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称													CH3CAPFVAL[4：1]			
访问													RW			
Reset													0			
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	CH3CAPFVAL[0：0]	CH2CAPFVAL					CH1CAPFVAL						CH0CAPFVAL			
访问	RW	RW					RW						RW			
Reset	0	0					0						0			

字段	说明
19：15 CH3CAPFVAL	**通道 3 输入滤波器** 选择用于通道输入的滤波器值，当值为 0 时禁用该滤波器。
14：10 CH2CAPFVAL	**通道 2 输入滤波器** 选择用于通道输入的滤波器值，当值为 0 时禁用该滤波器。
9：5 CH1CAPFVAL	**通道 1 输入滤波器** 选择用于通道输入的滤波器值，当值为 0 时禁用该滤波器。
4：0 CH0CAPFVAL	**通道 0 输入滤波器** 选择用于通道输入的滤波器值，当值为 0 时禁用该滤波器。

11.5.18 故障滤波器和故障使能寄存器(PWM_FFAFER)

表 11-25 PWM_FFAFER 寄存器

PWM_FFAFER 故障滤波器和故障使能寄存器 **Reset:00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	FFVAL									FF2EN	FF1EN	FF0EN		FER2EN	FER1EN	FER0EN
访问	RW									RW	RW	RW		RW	RW	RW
Reset	0									0	0	0		0	0	0

字段	说明
15：8 FFVAL	**故障输入滤波器** 选择用于故障输入的滤波器值，当值为 0 时禁用故障滤波器。
6 FF2EN	**故障输入 2 滤波器使能** 0： 禁用故障输入滤波器 1： 使能故障输入滤波器 启用故障输入滤波器，该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。
5 FF1EN	**故障输入 1 滤波器使能** 0： 禁用故障输入滤波器 1： 使能故障输入滤波器 启用故障输入滤波器，该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。
4 FF0EN	**故障输入 0 滤波器使能** 0： 禁用故障输入滤波器 1： 使能故障输入滤波器 启用故障输入滤波器，该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。
2 FER2EN	**故障输入 2 使能** 0：禁用故障输入 1：使能故障输入 使能故障输入。该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。
1 FER1EN	**故障输入 1 使能**

字段	说明
	0：禁用故障输入 1：使能故障输入 使能故障输入。该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。
0 FER0EN	**故障输入 0 使能** 0：禁用故障输入 1：使能故障输入 使能故障输入。该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。

11.5.19 正交解码器接口配置寄存器(PWM_QDI)

表 11-26 PWM_QDI 寄存器

PWM_QDI **正交解码器接口配置寄存器** **Reset:00000004**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称											PHAPOL	PHBPOL	QUADMODE	QUADIR	CNTOFDIR	QDIEN
访问											RW	RW	RW	RO	RO	RW
Reset											0	0	0	1	0	0

字段	说明
5 PHAPOL	**A 相 输入极性** 0：正常极性。在识别 A 相输入信号的上升沿和下降沿之前，该信号未反相 1：反转极性。在识别 A 相输入信号的上升沿和下降沿之前，该信号反相 选择正交解码器 A 相输入的极性。
4 PHBPOL	**B 相 输入极性** 0：正常极性。在识别 B 相输入信号的上升沿和下降沿之前，该信号未反相 1：反转极性。在识别 B 相输入信号的上升沿和下降沿之前，该信号反相 选择正交解码器 B 相输入的极性
3 QUADMODE	**正交解码器模式** 0： A 相和 B 相编码模式

字段	说明
	1： 计数和方向编码模式 选择正交解码器模式下使用的编码模式。
2 QUADIR	**正交解码器模式下的 PWM 计数器方向** 0： 计数方向为降低 (PWM 计数器递减). 1： 计数方向为增加 (PWM 计数器递增) 表示计数方向。
1 CNTOFDIR	**正交解码器模式下的定时器溢出方向** 0： CNTOF 位设置在计数的底部 。 PWM 计数器递减，PWM 计数器从其最小值（CNTIN 寄存器）变为最大值（MCVR 寄存器） 1： CNTOF 位设置在计数的顶部 。 PWM 计数器递增，PWM 计数器从其最大值（MCVR 寄存器）变为最小值（CNTIN 寄存器） 表示 CNTOF 位是否设置在计数的顶部或底部 。
0 QDIEN	**正交解码器模式使能** 0：禁用正交解码器模式 1：使能正交解码器模式 使能正交解码器模式。在此模式下，A 相和 B 相输入信号控制 PWM 计数器方向。正交解码器模式优先于其他模式。该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。

11.5.20 配置寄存器(PWM_CONF)

表 11-27 PWM_CONF 寄存器

PWM_CONF **配置** **Reset:00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	EVENT7PSC		EVENT6PSC		EVENT5PSC		EVENT4PSC		EVENT3PSC		EVENT2PSC		EVENT1PSC		EVENT0PSC	
访问	RW		RW		RW		RW		RW		RW		RW		RW	
Reset	0		0		0		0		0		0		0		0	
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称						GTBEOUT	GTBEEN		CNTOFNUM							
访问						RW	RW		RW							
Reset						o	o		0							

字段	说明
31：30	**通道 7 输入事件预分频**

字段	说明
EVENT7PSC	
	00: 1 个输入事件触发一次捕获
	01: 2 个输入事件触发一次捕获
	10: 4 个输入事件触发一次捕获
	11: 8 个输入事件触发一次捕获
29：28	**通道 6 输入事件预分频**
EVENT6PSC	
	00: 1 个输入事件触发一次捕获
	01: 2 个输入事件触发一次捕获
	10: 4 个输入事件触发一次捕获
	11: 8 个输入事件触发一次捕获
27：26	**通道 5 输入事件预分频**
EVENT5PSC	
	00: 1 个输入事件触发一次捕获
	01: 2 个输入事件触发一次捕获
	10: 4 个输入事件触发一次捕获
	11: 8 个输入事件触发一次捕获
25：24	**通道 4 输入事件预分频**
EVENT4PSC	
	00: 1 个输入事件触发一次捕获
	01: 2 个输入事件触发一次捕获
	10: 4 个输入事件触发一次捕获
	11: 8 个输入事件触发一次捕获
23：22	**通道 3 输入事件预分频**
EVENT3PSC	
	00: 1 个输入事件触发一次捕获
	01: 2 个输入事件触发一次捕获
	10: 4 个输入事件触发一次捕获
	11: 8 个输入事件触发一次捕获
21：20	**通道 2 输入事件预分频**
EVENT2PSC	
	00: 1 个输入事件触发一次捕获
	01: 2 个输入事件触发一次捕获
	10: 4 个输入事件触发一次捕获
	11: 8 个输入事件触发一次捕获
19：18	**通道 1 输入事件预分频**
EVENT1PSC	
	00: 1 个输入事件触发一次捕获
	01: 2 个输入事件触发一次捕获
	10: 4 个输入事件触发一次捕获
	11: 8 个输入事件触发一次捕获
17：16	**通道 0 输入事件预分频**
EVENT0PSC	
	00: 1 个输入事件触发一次捕获

字段	说明
	01: 2 个输入事件触发一次捕获 10: 4 个输入事件触发一次捕获 11: 8 个输入事件触发一次捕获
10 GTBEOUT	**全局时基输出** 0：禁用全局时基信号生成 1：使能全局时基信号生成 使能到其它 PWM 的全局时基信号生成。
9 GTBEEN	**全局时基使能** 0 ：禁用外部全局时基 1：使能外部全局时基
6：0 CNTOFNUM	**CNTOF 频率** 选择计数器溢出次数与 CNTOF 位置 1 次数之间的比率 CNTOFNUM = 0： 每个计数器溢出都设置 CNTOF 位 CNTOFNUM = 1：CNTOF 位设置为第一次计数器溢出，但不是下一次溢出 CNTOFNUM = 2：CNTOF 位设置为第一次计数器溢出，但不会设置为接下来的两次溢出 CNTOFNUM = 3：CNTOF 位设置为第一次计数器溢出，但不会设置为接下来的三次溢出

11.5.21 故障输入极性寄存器(PWM_FLTPOL)

表 11-28 PWM_FLTPOL 寄存器

PWM_FLTPOL **PWM 故障输入极性** **Reset:00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称														FLT2POL	FLT1POL	FLT0POL
访问														RW	RW	RW
Reset														0	0	0

字段	说明
2 FLT2POL	**故障输入 2 极性** 0： 故障输入极性为高电平有效 。故障输入处的 1 表示一个故障。

字段	说明
	1： 故障输入极性为低电平有效 。故障输入处的 0 表示一个故障。 定义故障输入极性。该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。
1 FLT1POL	**故障输入 1 极性** 0： 故障输入极性为高电平有效 。故障输入处的 1 表示一个故障。 1： 故障输入极性为低电平有效 。故障输入处的 0 表示一个故障。 定义故障输入极性。该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。
0 FLT0POL	**故障输入 0 极性** 0： 故障输入极性为高电平有效 。故障输入处的 1 表示一个故障。 1： 故障输入极性为低电平有效 。故障输入处的 0 表示一个故障。 定义故障输入极性。该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。

11.5.22 同步配置寄存器(PWM_SYNCONF)

表 11-29 PWM_SYNCONF 寄存器

PWM_SYNCONF **同步配置** **Reset:00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称										HWPOL	SWPOL	SWVHWSYNC	INVHWSYNC	OMVHWSYNC	PWMSVHWSYNC	CNTVHWSYNC
访问										RW	RW	RW	RW	RW	RW	RW
Reset										0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称				SWVSWSYNC	INVSWSYNC	OMVSWSYNC	PWMSVSWSYNC	CNTVSWSYNC	SYNCMODE		SWOC	INVC		CNTINC		HWTRIGMODESEL
访问				RW	RW	RW	RW	RW	RW		RW	RW		RW		RW
Reset				0	0	0	0	0	0		0	0		0		0

字段	说明
22 HWPOL	**通道 CHPOLCR 同步由硬件触发器激活**

字段	说明
	0：硬件触发器不会激活 CHPOLCR 寄存器同步 1：硬件触发器激活 CHPOLCR 寄存器同步
21 SWPOL	**通道 CHPOLCR 同步由软件触发器激活** 0： 软件触发器不会激活 CHPOLCR 寄存器同步 1： 软件触发器激活 CHPOLCR 寄存器同步
20 SWVHWSYNC	**软件输出控制同步由硬件触发器激活** 0： 硬件触发器不会激活 CHOSWCR 寄存器同步 1 ：硬件触发器激活 CHOSWCR 寄存器同步
19 INVHWSYNC	**通过硬件触发器激活反相控制同步** 0： 硬件触发器不会激活 INVCR 寄存器同步 1： 硬件触发器激活 INVCR 寄存器同步
18 OMVHWSYNC	**通过硬件触发器激活输出屏蔽同步** 0 ： 硬件触发器不会激活 OMCR 寄存器同步 1： 硬件触发器激活 OMCR 寄存器同步
17 PWMSVHWSYNC	**通过硬件触发器激活 MCVR, CNTIN 和 CHV 寄存器同步** 0 ： 硬件寄存器不会激活 MCVR, CNTIN 和 CHV 寄存器同步 1： 硬件寄存器激活 MCVR, CNTIN 和 CHV 寄存器同步
16 CNTVHWSYNC	**通过硬件触发器激活 PWM 计数器同步** 0 ： 硬件触发器不会激活 PWM 计数器同步 1 ： 硬件触发器激活 PWM 计数器同步
12 SWVSWSYNC	**通过软件触发器激活软件输出控制同步** 0： 软件触发器不会激活 CHOSWCR 寄存器同步. 1： 软件触发器激活 CHOSWCR 寄存器同步
11 INVSWSYNC	**通过软件触发器激活反相控制同步** 0 ： 软件触发器不会激活 INVCR 寄存器同步. 1 ： 软件触发器激活 INVCR 寄存器同步
10 OMVSWSYNC	**通过软件触发器激活输出掩码同步** 0 ： 软件触发器不会激活 OMCR 寄存器同步 1： 软件触发器激活 OMCR 寄存器同步
9 PWMSVSWSYNC	**通过软件触发器激活 MCVR, CNTIN 和 CHV 寄存器同步** 0 : 软件触发器不会激活 MCVR, CNTIN 和 CHV 寄存器同步 1：软件触发器激活 MCVR, CNTIN 和 CHV 寄存器同步.
8 CNTVSWSYNC	**通过软件触发器激活 PWM 寄存器同步**

字段	说明
	0： 软件触发器不会激活 PWM 计数器同步 1： 软件触发器激活 PWM 计数器同步
7 SYNCMODE	**同步模式** 0： 选择传统 PWM 同步 1： 选择增强 PWM 同步 选择 PWM 同步模式
5 SWOC	**CHOSWCR 寄存器同步** 0： 在总线时钟的所有上升沿，使用其缓冲区的值更新 CHOSWCR 寄存器 1： 通过 PWM 同步，使用其缓冲区的值更新 CHOSWCR 寄存器
4 INVC	**INVCR 寄存器同步** 0：在总线时钟的所有上升沿，使用其缓冲区的值更新 INVCR 寄存器 1：通过 PWM 同步，使用其缓冲区的值更新 INVCR 寄存器
2 CNTINC	**CNTIN 寄存器同步** 0：在总线时钟的所有上升沿，使用其缓冲区的值更新 CNTIN 寄存器 1： 通过 PWM 同步，使用其缓冲区的值更新 CNTIN 寄存器
0 HWTRIGMODESEL	**硬件触发模式** 0： 当检测到硬件触发器 j 时，PWM 清零 TRIGj 位，其中 j = 0, 1, 2. 1： 当检测到硬件触发器 j 时，PWM 不会清零 TRIGj 位，其中 j = 0, 1, 2.

11.5.23 反相控制寄存器(PWM_INVCR)

表 11-30 PWM_INVCR 寄存器

PWM_INVCR **PWM 反相控制寄存器** **Reset:00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称													PAIR3INVEN	PAIR2INVEN	PAIR1INVEN	PAIR0INVEN
访问													RW	RW	RW	RW
Reset													0	0	0	0

字段	说明
3 PAIR3INVEN	**配对通道 3 反相使能** 0： 禁用反相 1： 使能反相
2 PAIR2INVEN	**配对通道 2 反相使能** 0： 禁用反相 1： 使能反相
1 PAIR1INVEN	**配对通道 1 反相使能** 0： 禁用反相 1： 使能反相
0 PAIR0INVEN	**配对通道 0 反相使能** 0： 禁用反相 1： 使能反相

11.5.24 通道软件输出控制寄存器(PWM_CHOSWCR)

表 11-31 PWM_CHOSWCR 寄存器

PWM_CHOSWCR **PWM 通道 软件输出控制寄存器** **Reset:00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	CH7SWCV	CH6SWCV	CH5SWCV	CH4SWCV	CH3SWCV	CH2SWCV	CH1SWCV	CH0SWCV	CH7SWEN	CH6SWEN	CH5SWEN	CH4SWEN	CH3SWEN	CH2SWEN	CH1SWEN	CH0SWEN
访问	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

字段	说明
15 CH7SWCV	**通道 7 软件输出控制值** 0： 软件输出控制将通道输出强制置为 0 1： 软件输出控制将通道输出强制置为 1
14 CH6SWCV	**通道 6 软件输出控制值** 0： 软件输出控制将通道输出强制置为 0 1： 软件输出控制将通道输出强制置为 1

字段	说明
13 CH5SWCV	**通道 5 软件输出控制值** 0： 软件输出控制将通道输出强制置为 0 1： 软件输出控制将通道输出强制置为 1
12 CH4SWCV	**通道 4 软件输出控制值** 0： 软件输出控制将通道输出强制置为 0 1： 软件输出控制将通道输出强制置为 1
11 CH3SWCV	**通道 3 软件输出控制值** 0： 软件输出控制将通道输出强制置为 0 1： 软件输出控制将通道输出强制置为 1
10 CH2SWCV	**通道 2 软件输出控制值** 0： 软件输出控制将通道输出强制置为 0 1： 软件输出控制将通道输出强制置为 1
9 CH1SWCV	**通道 1 软件输出控制值** 0： 软件输出控制将通道输出强制置为 0 1： 软件输出控制将通道输出强制置为 1
8 CH0SWCV	**通道 0 软件输出控制值** 0： 软件输出控制将通道输出强制置为 0 1： 软件输出控制将通道输出强制置为 1
7 CH7SWEN	**通道 7 软件输出控制使能** 0： 通道输出不受软件输出控制的影响 1 ：通道输出受软件输出控制的影响
6 CH6SWEN	**通道 6 软件输出控制使能** 0： 通道输出不受软件输出控制的影响 1 ：通道输出受软件输出控制的影响
5 CH5SWEN	**通道 5 软件输出控制使能** 0： 通道输出不受软件输出控制的影响 1 ：通道输出受软件输出控制的影响
4 CH4SWEN	**通道 4 软件输出控制使能** 0： 通道输出不受软件输出控制的影响 1 ：通道输出受软件输出控制的影响
3 CH3SWEN	**通道 3 软件输出控制使能** 0： 通道输出不受软件输出控制的影响 1 ：通道输出受软件输出控制的影响

字段	说明
2 CH2SWEN	**通道 2 软件输出控制使能** 0： 通道输出不受软件输出控制的影响 1 ：通道输出受软件输出控制的影响
1 CH1SWEN	**通道 1 软件输出控制使能** 0： 通道输出不受软件输出控制的影响 1 ：通道输出受软件输出控制的影响
0 CH0SWEN	**通道 0 软件输出控制使能** 0： 通道输出不受软件输出控制的影响 1 ：通道输出受软件输出控制的影响

12 脉冲宽度检测定时器(PWDT)

12.1 简介

脉冲宽度检测定时器（PWDT）被用做测量脉冲宽度的工具或作为 16 位普通定时器使用。该 MCU 设备包含 2 个 PWDT 模块，每个 PWDT 模块支持 3 路外部通道+1 路内部通道输入。

12.2 特性

- 4 个可选脉冲时钟输入
- 支持 2 个功能 ： 脉冲宽度测量功能和定时功能
  - 脉冲宽度测量功能
    - 可编程起始触发沿
    - 4 个可编程测量模式
    - 支持 3 个霍尔传感器的信号输入测量
    - 支持来自模拟比较器的 3 个输入
  - 定时器功能
- 16 位计数器，用做脉冲宽度测量或定时器功能
- 中断
  - OVF：计数器溢出中断
  - RDYF：脉宽数据就绪中断

12.3 结构框图

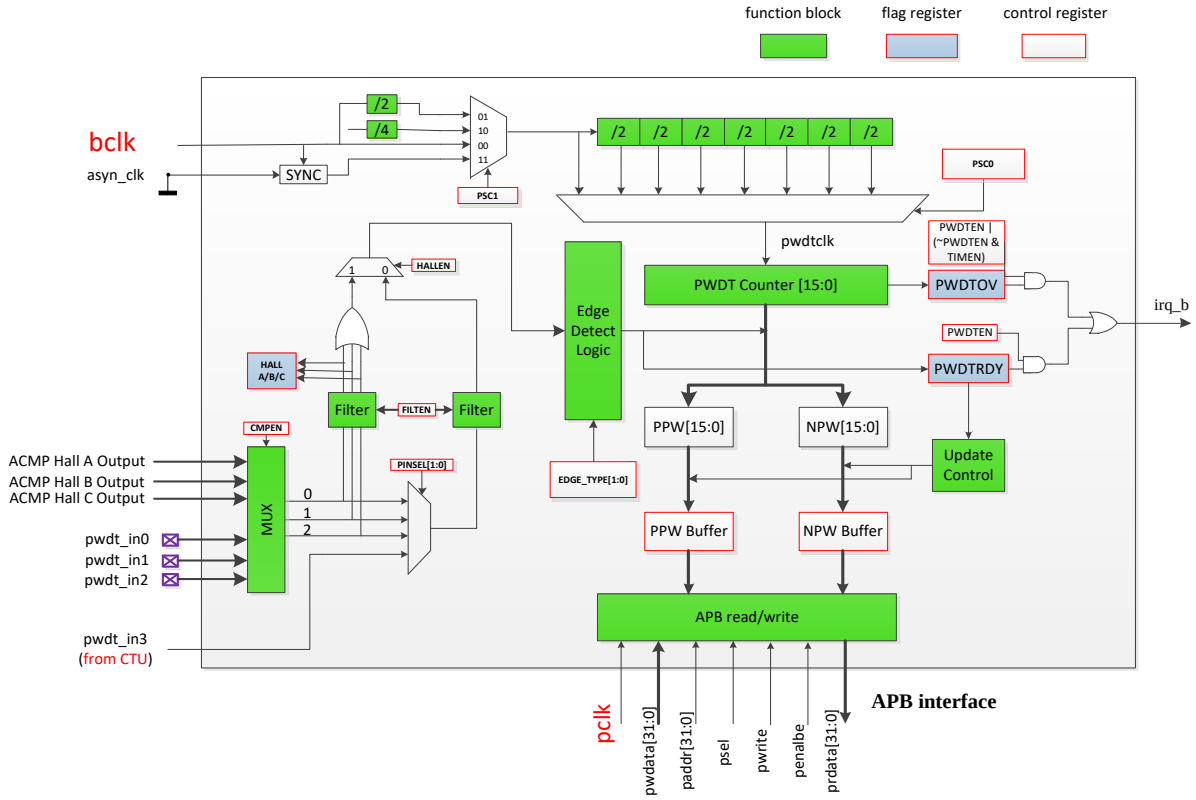

图 12-1 PWDT 结构框图

12.4 功能描述

12.4.1 脉冲宽度测量功能

要得到脉冲宽度测量值，PWDTC 计数器基于 PWDTEN =1 后的框图中描述的 pwdtclk 运行。pwdtclk 是由 PSC0[2：0]和 PSC1[1：0]从总线时钟分频得到, PSC1 为前级预分频，PSC0 为后级预分频，两者相乘组合成总预分频 PSC。为了获得更准确的测量值，用户可设置较小的 PSC 值用于较窄的脉冲输入，设置较大的 PSC 值用于较宽的脉冲输入。

对于脉冲宽度测量，提供了两种可选模式：一种是用于测量单通道输入的基本测量模式，另一个是用于测量 3 通道输入的霍尔测量模式。

12.4.1.1 基本测量模式

用户可以通过设置 PINSEL[1：0]来选择测量特定的通道输入，并可以根据实际应用通过设置 EDGE[1：0]来选择 4 种测量模式中的一种，如图 12-2 所示。

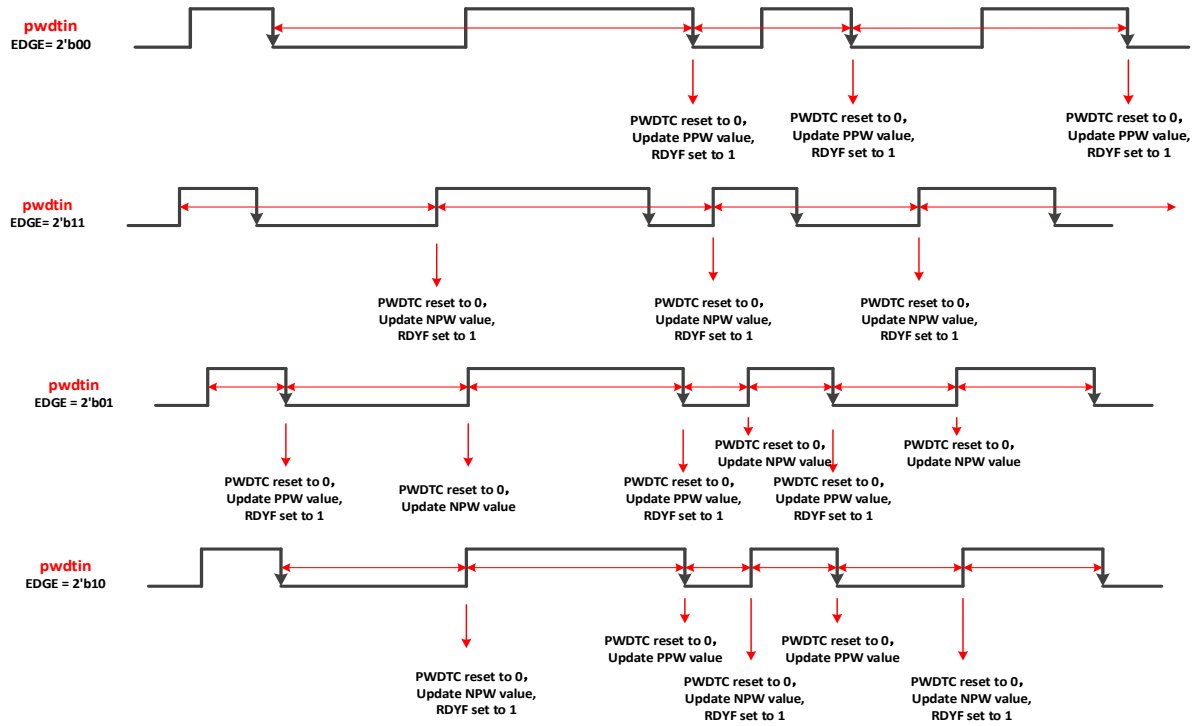

图 12-2 四种基本测量模式(HALLEN=0)

12.4.1.2 霍尔测量模式

模块测量从 3 个通道输入的异或（XOR）得到的脉冲输入，用户应设置 EDGE[1：0]= 2'b01。此外，内部比较器 3 个通道输入的配置类似于霍尔测量，只需 CMPEN 同时配置为 1'b1。

对于霍尔信号测量，只需选择此模式用于电机速度计算或换相，如图 12-3 所示。与上图中的基本测量模式相比， RDYF 标志在上升沿和下降沿均设置为 1。

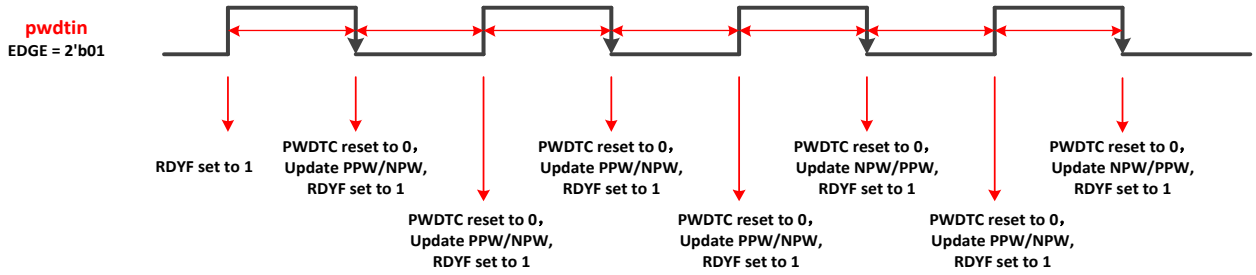

图 12-3 霍尔测量模式(HALLEN=1)

在电动机中，霍尔装置的安装用于检测转子的位置以适当地换相。通常有两种安装方式，如图 12-4 所示。一个是 120 电度间隔，另一个是 60 电度间隔。

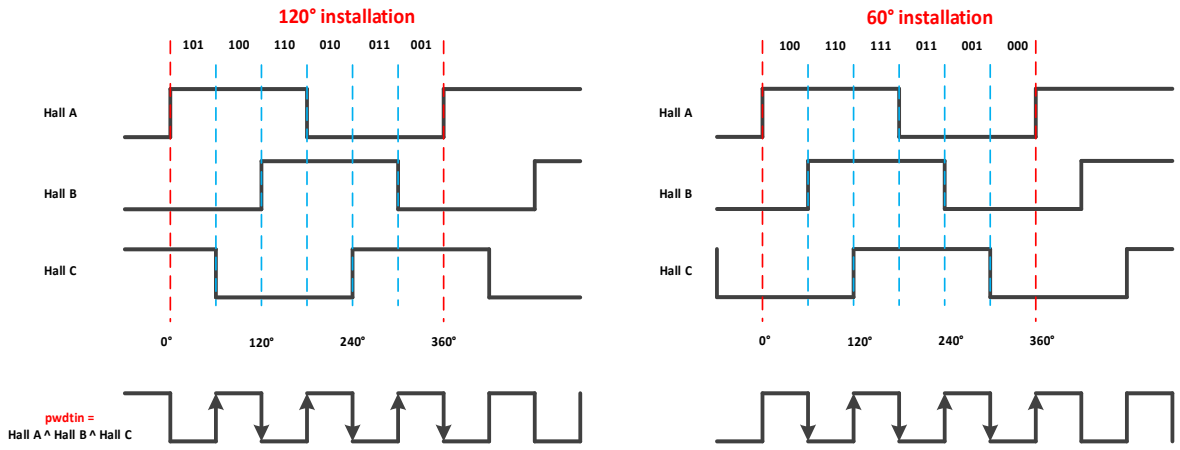

图 12-4 两种常见 Hall 安装方式

12.4.1.3 输入滤波器

输入滤波器用于滤除高/低电平小于特定宽度的噪声信号。FILT_PSC[3：0]和 FILTVAL [3：0]设置确定最大和最小噪声脉冲宽度。图 12-5 和图 12-6 介绍了噪声滤波原理，当用户配置 FILTVAL = 15 和 FILT_PSC = 2 时，滤波器脉冲宽度为 60 bclk，小于 60 bclk 的脉冲被判断为噪声脉冲并将被过滤掉。可滤波的脉冲宽度如表 12-1 所示。

注：使能滤波器功能会导致信号存在延迟（延迟时间为设置的滤波宽度）。因此，如果 PWDT 模块运行在电机应用的霍尔测量模式下，可能需要考虑该延迟导致换相时间偏移的情况，用户可基于滤波延迟做相应的软件补偿。

表 12-1 可滤波脉冲宽度范围

项	时钟	时间
可滤波脉冲宽度范围	2 bcllk ~ 15*4096 bclk	0.04μs ~ 1.229ms

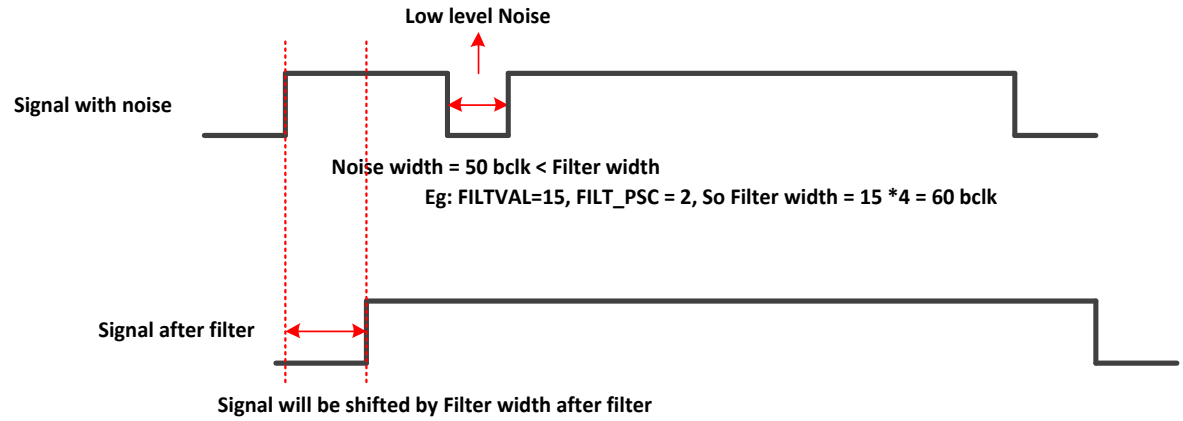

图 12-5 低电平噪音和滤波器示例

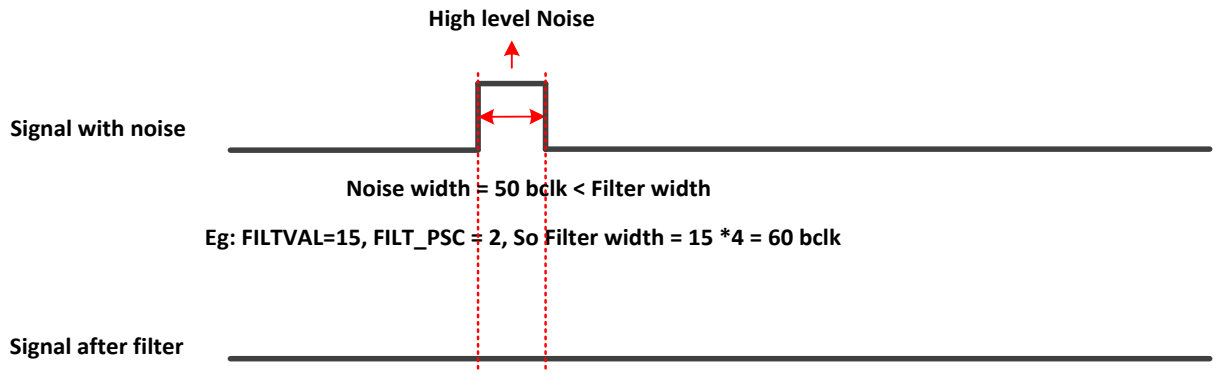

图 12-6 高电平噪音和滤波器示例

12.4.1.4 测量误差

用户应了解脉冲宽度测量的测量精度，配置 PSC 适当的值，以达到更准确的测量值。一个基本原则是，使用较小的 PSC 可以获得更准确的测量值。显然，输入脉冲越窄，相对测量误差越大。图 12-7 描述了脉冲宽度测量功能运行时的误差。在图 12-7 中，当 pwdtin 脉冲从高电平变为低电平或从低电平变为高电平时，PWDTC 计数器和 pwdtclk 除数计数器同时复位为 0，并且恰好在这里发生计数错误，该错误源自图 12-7 中所示的最后计数值。实际宽度值小于测量值不足一个 pwdtclk 周期。

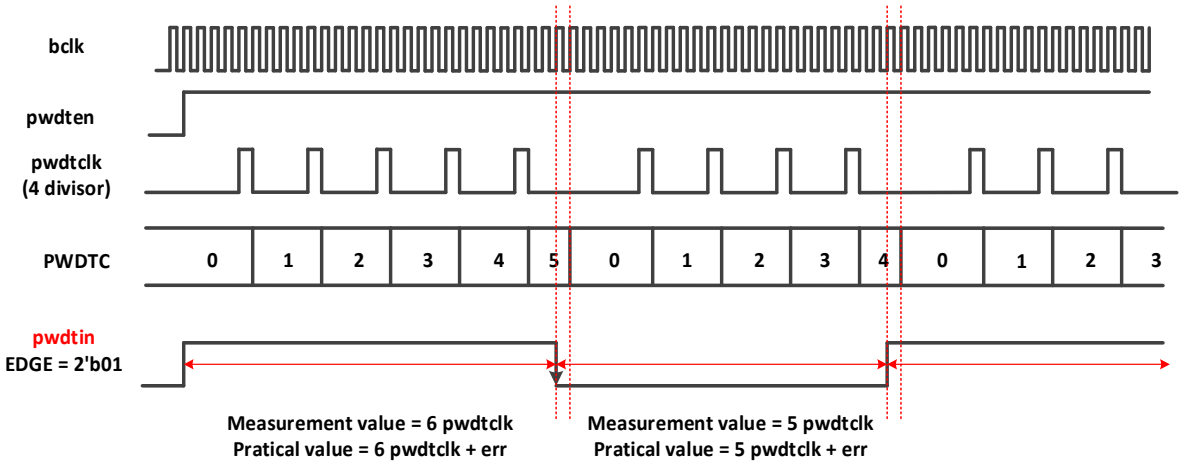

图 12-7 PWDTC 计数器和计数错误

12.4.2 定时器功能

对于定时器功能，只有 OVF 状态有效，在 PWDTC 计数器溢出时置位。计数值 TIMLDVAL[15：0]可以一直修改。但是，在不同的时间点修改计数器负载值会导致 MCU 执行不同的操作，如图 12-8 和图

12-9 所示。

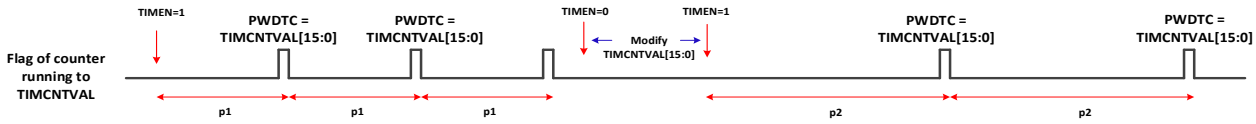

图 12-8 在 TIMEN=0 期间修改 TIMLDVAL

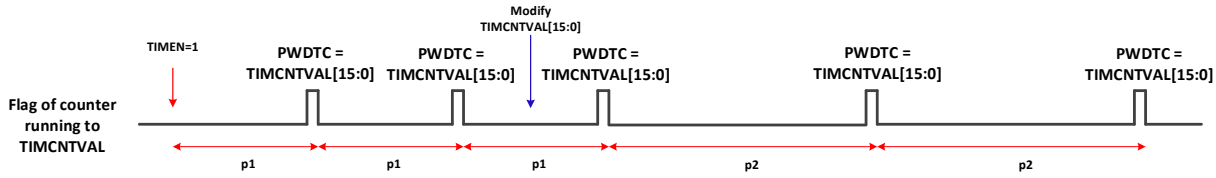

图 12-9 TIMEN=1 期间修改 TIMLDVAL

12.5 应用说明

12.5.1 脉冲宽度测量功能编程指南

用户必须牢记，PWDTEN 应该在所有其他控制位之后配置为 1。否则，可能会出现异常情况。对于内部 3 个比较器输入，HALLEN 和 CMPEN 应配置为 1。

12.5.2 定时器功能编程指南

只需配置 TIMLDVAL，PRESCALE 和 TIMEN 等，即可轻松使用定时器功能。用户应将 TIMEN 设置为 1，并且不能将 PWDTEN 设置为 1，因为脉冲宽度测量功能优先于定时器功能。

12.6 寄存器定义

表 12-2 PWDT 寄存器映射

PWDT0 基地址 = 0x40017000
PWDT1 基地址 = 0x40017800

地址	名称	宽度	描述
PWDTx 基地址+0x00	PWDT_INIT0	32	通用控制和状态位，及正脉宽内容
PWDTx 基地址+0x04	PWDT_NPW	32	负脉宽内容及 16 位自由运行计数器
PWDTx 基地址+0x08	PWDT_INIT1	32	霍尔功能控制和定时器功能控制

【说明】上表中，x=0,1。

12.6.1 初始化寄存器 0(PWDT_INIT0)

表 12-3 PWDT_INIT0 寄存器

PWDT_INIT0 **PWDT 初始化寄存器 0** **Reset:00000000**

位	31~16	15~14	13~12	11~10	9~7	6	5	4	3	2	1	0
名称	PPW	PSC1	PINSEL	EDGE	PSC0	PWDTEN	IE	PRDYIE	OVIE		RDYF	OVF
访问	RO	RW	RW	RW	RW	RW	RW	RW	RW		W0C	W0C
Reset	0	0	0	0	0	0	0	0	0		0	0

字段	说明
31：16 PPW	**正脉宽** 正脉宽值
15：14 PSC1	**PWDT 计数器前级预分频** 00 ~ 11：分别代表 1/2/4/预留
13：12 PINSEL	**引脚选择** 00/01/10/11：分别选择 pwdt_in0/ pwdt_in1/ pwdt_in2/ pwdt_in3. **注意：internal_pwdtin 来自 CTU 模块内部。**
11：10 EDGE	**选择输入边沿触发类型** 00： 第一个下降沿开始，在所有之后的下降沿触发要捕获的脉宽 01：第一个上升沿开始，在所有之后的上升沿和下降沿触发要捕获的脉宽 10：第一个下降沿开始，在所有之后的上升沿和下降沿触发要捕获的脉宽 11：第一个上升沿开始，在所有之后的上升沿触发要捕获的脉宽
9：7 PSC0	**PWDT 计数器后级预分频** 000 ~ 111：分别代表 1/2/4/8/…/128
6 PWDTEN	**PWDT 脉宽测量模式使能** 0： 禁用 1： 使能 **注意：PWDTEN 使能(pwdt 脉宽测量模式)优先于 TIMEN 使能(定时器模式)，因此当要使能定时器模式时，必须禁用 PWDTEN。**
5 IE	**PWDT 模块中断使能** 0： 禁用 1： 使能

字段	说明
4 PRDYIE	**PWDT 脉宽数据就绪中断使能** 0： 禁用 1： 使能
3 OVIE	**PWDT 计数器溢出中断使能** 0： 禁用 1： 使能
1 RDYF	**PWDT 脉宽数据就绪** 0： pwdt 脉宽寄存器未更新 1： pwdt 脉宽寄存器已更新，写 0 清除
0 OVF	**PWDT 计数器溢出** 0： 无溢出 1：pwdt 计数器溢出，写 0 清除

12.6.2 脉宽计数寄存器(PWDT_NPW)

表 12-4 PWDT_NPW 寄存器

PWDT_NPW **PWDT NPW 计数值** **Reset: 00000000**

位	31~16	15~0
名称	PWDTC	NPW
访问	RO	RO
Reset	0	0

字段	说明
31：16 PWDTC	**脉宽计数器** 用于脉宽测量或定时器计数
15：0 NPW	**负脉宽** 负脉宽值

12.6.3 初始化寄存器 1(PWDT_INIT1)

表 12-5 PWDT_INIT1 寄存器

PWDT_INIT1 **PWDT 初始化寄存器 1** **Reset:00000000**

位	31	30	29	28	27~12	11	10	9	8	7~4	3~0
名称		HALLA	HALLB	HALLC	TIMLDVAL	CMPEN	TIMEN	HALLEN	FILTEN	FILTPSC	FILTVAL
访问		RO	RO	RO	RW	RW	RW	RW	RW	RW	RW
Reset		0	0	0	0	0	0	0	0	0	0

字段	说明
30：28 HALLA/B/C	**HALLA/HALLB/HALLC 状态值** 如果 3 个霍尔传感器安装间距为 60 电度： 100 → 110 → 111 → 011 → 001 → 000 否则， 3 个霍尔传感器安装空间为 120 电度： 101 → 100 → 110 → 010 → 011 → 001
27：12 TIMLDVAL	**定时器装载值** 定时器从 0x0000 运行 TIMLDVAL。
11 CMPEN	**比较器输入使能** **0**：使能来自 pad PWDT_IN0 ~ PWDT_IN2 外部的 pwdt_in0 ~ pwdt_in2 **1**：使能来自 acmp0_0 ~ acmp0_2 内部的 pwdt_in0 ~ pwdt_in2 **注意：当 CMPEN=1 时，pwdt_in0 ~ pwdt_in2 来自于 acmp0_0 ~ acmp0_2 内部。然后 HALLEN=1，可以通过 HALL 传感器的行为来测量 acmp0_0 ~ acmp0_2 信号，否则将基于 PINSEL 测量其中一个信号。**
10 TIMEN	**定时器使能** 0：禁用 1：使能定时器功能 **注意：PWDTEN 使能（pwdt 脉宽测量模式）优先于 TIMEN 使能（定时器模式），因此当要使能定时器模式时，必须禁用 PWDTEN。**
9 HALLEN	**霍尔传感器信号检测使能** 0： 禁用霍尔传感器信号检测功能 1：使能霍尔传感器信号检测功能
8 FILTEN	**输入滤波器使能** 0：禁用. 1： 使能滤波器功能

字段	说明
7 ~ 4 FILTPSC	**滤波器预分频器** 1~ 12：分别表示 2/4/8…/4096 分频 0, 13 ~ 15：不分频滤波器时钟
3 ~ 0 FILTVAL	**滤波器值** 0：禁止滤波器 1 ~ 15 ： 滤波值

13 周期性中断定时器（TIMER）

13.1 简介

TIMER 模块是用于定时发起中断和触发的定时器。

13.2 特性

- 定时器能够生成中断
- 定时器能够生成触发脉冲
- 每个定时器都具有独立的超时周期
- 支持 4 个 32bit 定时器
- 支持链（ Link）模式

13.3 结构框图

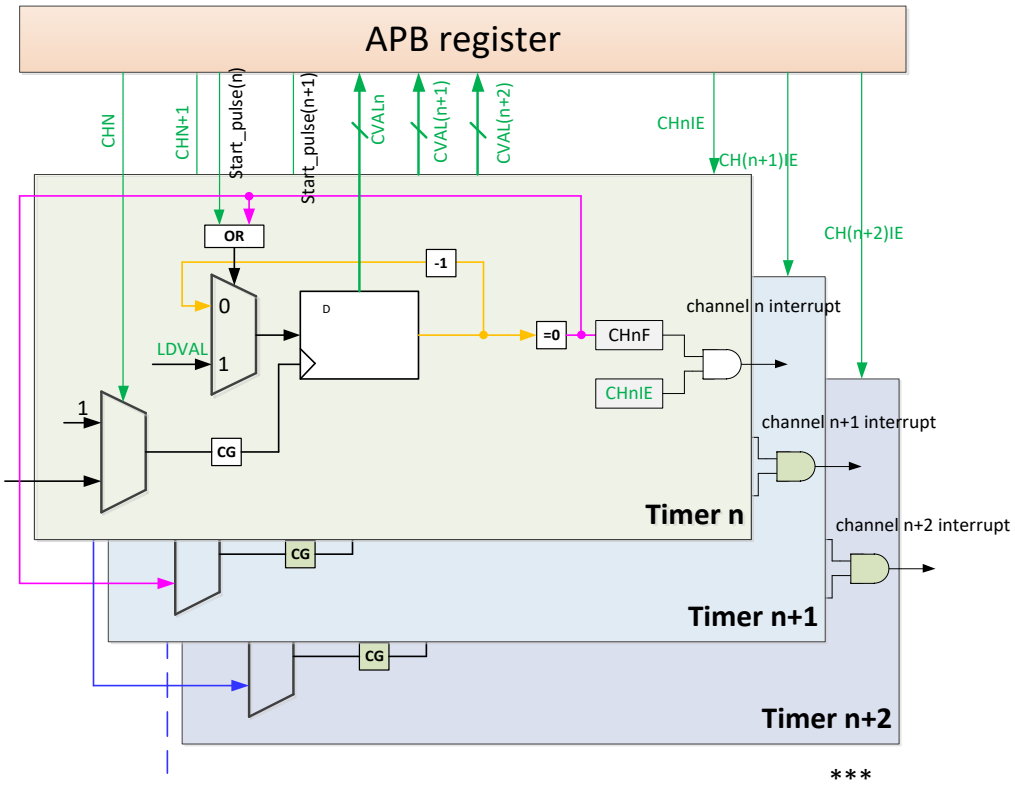

图 13-1 TIMER 结构框图

13.4 功能描述

13.4.1 普通模式

定时器使能时自动加载 TIMER_LDVAL 寄存器指定起始值，倒计数至 0 时，将生成一个触发脉冲并设置超时标志，然后再次装载起始值重新计数。

如有需要，可以通过 TIMER_CVAL 寄存器读取定时器的当前计数值。通过 TIMER_INIT[TEN]先禁用再使能，可以重新启动计数周期。

13.4.2 链接模式

当某个定时器的链接模式处于使能状态，那么只有在上一个定时器溢出后，它才会开始计时。因此，如果定时器 n-1 已倒数至 0，定时器 n 的值将递减 1。这样就能将某些定时器连接起来形成更长的定时器。
注：第一个定时器 (timer 0) 不能链接至其他定时器。

13.4.3 中断

定时器中断可通过置位 TIE 来使能。当相关定时器发生超时， TIF 超时标志置位为 1，写入 1 清零。在使用链接功能时，一般只使能定时器 n 中断，与 n 链接的定时器中断都是关闭状态。

13.5 寄存器定义

表 13-1 定时器寄存器映射

TIMER 基地址 = 0x40011000
TIMER_CH0 基地址=0x40011100
TIMER_CH1 基地址=0x40011110
TIMER_CH2 基地址=0x40011120
TIMER_CH3 基地址=0x40011130

地址	名称	宽度	描述
TIMER 基地址+0x00	TIMER_MCR	32	模块控制器寄存器
TIMER_CHx 基地址+0x00	TIMER_LDVAL	32	初始值寄存器
TIMER_CHx 基地址+0x04	TIMER_CVAL	32	当前值寄存器
TIMER_CHx 基地址+0x08	TIMER_INIT	32	初始化寄存器
TIMER_CHx 基地址+0x0C	TIMER_TF	32	标志寄存器

【说明】上表中，x=0~3。

13.5.1 定时器模块控制寄存器(TIMER_MCR)

表 13-2 TIMER_MCR 寄存器

TIMER_MCR 定时器模块控制寄存器 **Reset：0x00000002**

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

字段	说明
1 MDIS	**模块禁用 - (TIMER 部分)** 0： 使能定时器模块 1： 禁用定时器模块 禁用定时器模块，必须在执行任何其他设置前使能该字段。 **注：**MDIS 可以实现 4 路定时器同时暂停/再次开始计数

13.5.2 定时器装载值寄存器(TIMER_LDVAL)

表 13-3 TIMER_LDVAL 寄存器

TIMER_LDVAL 定时器装载值寄存器 **Reset：0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
字段	LDVAL[31：16]															
访问	RW															
Reset	0															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
字段	LDVAL[15：0]															
访问	RW															
Reset	0															

字段	说明
31：0 LDVAL	**装载值寄存器** 定时器起始值。定时器将倒计时至 0，然后生成一个中断并再次装载该寄存器的值。将新值写入该寄存器不会重启定时器。相反定时器过期后，会装载新值。要中止当前周期并用新值开始一个定时器周期，必须先禁用该定时器然后再将其使能。

13.5.3 定时器当前值寄存器(TIMER_CVAL)

表 13-4 TIMER_CVAL 寄存器

TIMER_CVAL 定时器当前值寄存器 **Reset：0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
字段	CVAL[31：16]															
访问	RO															
Reset	0															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
字段	CVAL[15：0]															
访问	RO															
Reset	0															

字段	说明
31：0 CVAL	**当前定时器值** CVAL 与时间的换算公式如下： the timing period(Unit: second) = (CVAL + 1) / timing clock frequency(Unit: Hz)

13.5.4 定时器初始寄存器 (TIMER_INIT)

表 13-5 TIMER_INIT 寄存器

TIMER_INIT 定时器初始寄存器 **Reset：0x00000000**

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

字段	说明
2 LINKEN	**链接模式** 0：定时器不链接 1：定时器链接至前一定时器。例如，对于定时器 2，若该字段置位，则定时器 2 链接至定时 1。 激活时，定时器 n-1 需先到期，定时器 n 才能递减 1。不能链接定时器 0 。
1 TIE	**定时器中断使能** 0： 禁用定时器中断请求

字段	说明
	1：一旦置位 TIF，请求中断 当某个中断挂起或 TIF 置位时，使能该中断将立即引起中断事件。要避免这种情况，必须先清零相关的 TIF。
0 TEN	**定时器使能** 0： 禁用定时器 n 1： 使能定时器 n 使能或禁用定时器 n

13.5.5 定时器标志寄存器(TIMER_TF)

表 13-6 TIMER_TF 寄存器

TIMER_TF **定时器标志寄存器** **Reset：0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
字段																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
字段																TIF
访问																W1C
Reset																0

字段	说明
0 TIF	**定时器超时标志** 在定时器周期结束时置 1。将 1 写入该标志可将其清零，写入 0 则无效。 若使能 TIE = 1，TIF 将引发中断请求。 0： 尚未发生超时 1： 超时已经发生

14 采集传输终端（CTU）

14.1 简介

CTU 模块可用于模块间的互连，片上不同模块之间传递信号。

14.2 特性

- ACMP0 输出捕获
- UART0_TX 调制
- UART0_RX 捕获
- UART0_RX 滤波
- RTC 捕获
- ADC 触发
- PWM 软件同步

14.3 结构框图

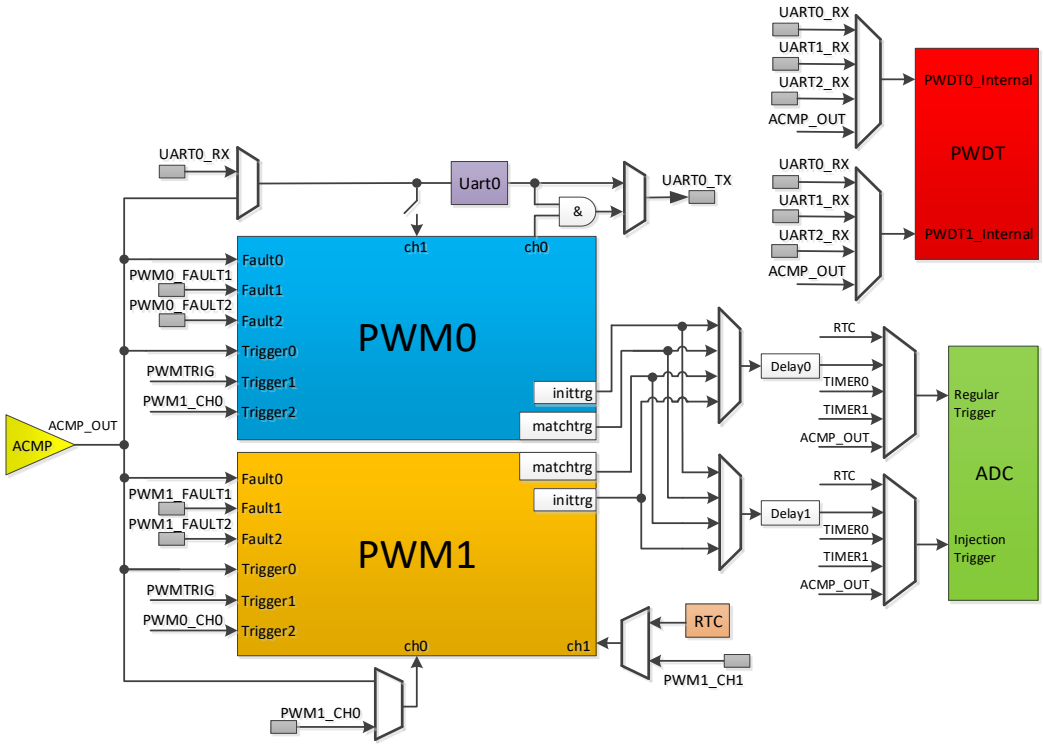

图 14-1 CTU 结构框图

14.4 功能描述

14.4.1 ACMP 输出捕获

CTU_CONFIG0[ACIC]位使能 ACMP0 的输出连接到 PWM1_CH0，PWM1_CH0 对应的外部引脚让与其他复用功能。将 CTU_CONFIG0[RXDFE]设置为 1b 时，可以选择 ACMP0 输出连接到 UART0 的接收器通道。ACMP0 输出可连接到 PWDT 输入（用于 BLDC 使用），或可用作 PWM 触发/故障输入和 ADC 硬件触发。

14.4.2 UART0_TX 调制

UART0_TX 可通过 PWM0_CH0 输出调制。CTU_CONFIG0[TXDME]置位时，UART0_TX 与 PWM_CH0 输出相与，输出结果到 UART0_TX 管脚。将该字段清零后， UART0_TX 会直接映射到管脚上。要使能 IR 调制功能，PWM0_CH0 和 UART 都必须处于有效状态，通过设置所需 PWM 周期与占空比后，每个通过 UART0_TX 传送的数据都通过 PWM0_CH0 输出调制， PWM0_CH0 引脚被释放给其他复用功能。

14.4.3 UART0_RX 捕获

CTU_CONFIG0[RXDCE]置位时， UART0_RX 引脚连接 UART0 和 PWM0_CH1，PWM0_CH1 对应的外部引脚让与其他复用功能。该字段清零后， UART0_RX 引脚仅连接 UART0。

14.4.4 UART0_RX 滤波器

CTU_CONFIG0[RXDFE]置位时，可将 ACMP0 输出连接至 UART0 的接收通道。要使能 UART0_RX 滤波器功能， UART0 和 ACMP 都必须处于有效状态。如果该功能处于有效状态，UART0_RX 对应的外部引脚让与其他复用功能。该字段清零后， UART0_RX 引脚直接连接至 UART0 模块。当 UART0_RX 捕捉功能处于有效状态时， ACMP0 输出也将注入 PWM0_CH1。

14.4.5 RTC 捕获

RTC 溢出信号可通过设置 CTU_CONFIG0 [RTCC]位由 PWM1_CH1 捕获。该字段置位后，RTC 溢出连接到 PWM1_CH1 以便进行捕获，而 PWM1_CH1 对应的外部引脚让与其他复用功能。

14.4.6 ADC 硬件触发

ADC 模块可以通过硬件触发器来启动转换。通过 CTU_CONFIG0[ADHWT0]字段设置规则组硬件触发源， CTU_CONFIG1[ADHWT1] 字段设置注入组硬件触发源。当 ADC 硬件触发器选择 PWM 触发器输出时，将使能一个 8 位延迟模块。该逻辑使用 8 位计数器延迟 PWM 的任何触发，计数器的值由 DELAY 指定。该模块的参考时钟是具有 CTU_CONFIG0 [PSC]指定的可选预分频器的总线时钟。

14.4.7 PWM 软件同步

PWM 包含三个同步输入触发器，CTU 提供其中一个触发器，通过 CTU_CONFIG0[PWMTRIG]写 1 来触发软件，将 0 写入该字段不起任何作用，该字段始终读到的是 0。

14.5 寄存器定义

表 14-1 CTU 寄存器映射

CTU 基地址 = 0x40016000

地址	名称	宽度	描述
CTU 基地址+0x00	CTU_CONFIG0	32	配置 0 寄存器
CTU 基地址+0x04	CTU_CONFIG1	32	配置 1 寄存器

14.5.1 配置寄存器 0(CTU_CONFIG0)

表 14-2 CTU_CONFIG0 寄存器

CTU_CONFIG0 **CTU 配置 0 寄存器** **Reset：0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	DELAY0								DLYACT0	ADHWT0				PSC		
访问	RW								RO	RW				RW		
Reset	0								0	0				0		
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	TXDME	PWMTRIG		RXDCE	ACIC	RTCC	RXDFE									
访问	RW	RW		RW	RW	RW	RW									
Reset	0	0		0	0	0	0									

字段	说明
31：24 DELAY0	**规则触发延时单元** PWM 初始或匹配触发到 ADC 硬件规则触发的延迟。8 位模数值允许从 0 到 255 的延迟，计数频率由 PSC 决定。这是一个单次计数器，当触发到达时开始计数，当计数器值达到定义的模数值时停止计数。
23 DLYACT0	**DELAY0 延迟有效** 0： 延迟无效 1： 延迟有效

字段	说明
	该只读字段指定 PWM 初始或匹配延迟有效时的状态。当 PWM 规则触发到达且延迟计数器正在计数时，该字段置位。否则，该字段会被清除。
22：20 ADHWT0	**ADC 规则组硬件触发源** 000： RTC 溢出作为 ADC 硬件触发源 001： PWM0 初始化触发，具有 8 位可编程计数器延迟 010： PWM0 匹配触发，具有 8 位可编程计数器延迟 011： PWM1 初始化触发，具有 8 位可编程计数器延迟 100： PWM1 匹配触发，具有 8 位可编程计数器延迟 101： TIMER 通道 0 溢出作为 ADC 硬件触发 110： TIMER 通道 1 溢出作为 ADC 硬件触发 111： ACMP0 输出作为 ADC 硬件触发 选择 ADC 硬件规则组触发源，所有触发源都在上升沿启动 ADC 转换。
18：16 PSC	**总线时钟预分频** 000： 总线 1 分频 001： 总线 2 分频 010： 总线 4 分频 011： 总线 8 分频 100： 总线 16 分频 101： 总线 32 分频 110： 总线 64 分频 111： 总线 128 分频 通过可选的预分频器使能总线时钟输出。
15 TXDME	**UART0_TX 调制选择** 0： UART0_TX 输出直接连接到引脚排列 1： 在映射到引脚排列之前，UART0_TX 输出由 PWM0 通道 0 调制 使能由 PWM0 通道 0 调制的 UART0_TX 输出。
14 PWMTRIG	**PWM 同步触发** 0： 没有同步触发 1： 为 PWM 模块生成 PWM 同步触发 如果向 PWMTRIG 字段中写入 1，则为 PWM 模块生成 PWM 同步触发 TRIG1。注意当设置 PWMTRIG = 1 产生触发行为时，PWMTRIG 位应手动写入 0。
12 RXDCE	**UART0_RX 捕获选择** 0： UART0_RX 输入信号仅连接到 UART0 模块 1： UART0_RX 输入信号连接到 UART0 模块及 PWM0 通道 1

字段	说明
	使能 UART0_RX，由 PWM0 通道 1 捕获。
11 ACIC	**模拟比较器输入捕获使能** 0： ACMP0 输出未连接到 PWM1 输入通道 0. 1： ACMP0 输出连接到 PWM1 输入通道 0. 将 ACMP0 输出连接到 PWM1 输入通道 0。
10 RTCC	**实时计数器捕获** 0： RTC 溢出未连接到 PWM1 输入通道 1 1： RTC 溢出 连接到 PWM1 输入通道 1. 允许 PWM1 通道 1 捕获实时计数器 (RTC) 溢出。
9 RXDFE	**UART0 Rx 滤波器选择** 00： RXD 输入信号直接连接到 UART0 模块 01： RXD 输入信号由 ACMP0 滤波，然后注入 UART0 使能 UART0 RxD 输入，由 ACMP0 滤波。当使能此功能后，任何标记有 ACMP0 输入的信号都可视为 UART0。

14.5.2 配置寄存器 1(CTU_CONFIG1)

表 14-3 CTU_CONFIG1 寄存器

CTU_CONFIG1 **CTU 配置 1 寄存器** **Reset:0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称		DELAY1								DLYACT1						
访问		RW								RW						
Reset		0								0						
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称				PWDT1IN3S		PWDT0IN3S		ADHWT1								
访问				RW		RW		RW								
Reset				0		0		0								

字段	说明
30:23 DELAY1	**注入触发延时单元** PWM 初始或匹配触发到 ADC 硬件注入触发的延迟。8 位模数值允许从 0 到 255 的延迟，计数频率由 PSC 决定。这是一个单次计数器，当触发到达时开始计数，当计数器值达到定义的模数值时停止计数。
22 DLYACT1	**DELAY1 延迟有效** 0： 延迟无效

字段	说明
	1： 延迟有效 该只读字段指定 PWM 初始或匹配延迟有效时的状态。当 PWM 注入触发到达且延迟计数器正在计数时，该字段置位。否则，该字段会被清除。
12:11 PWDT1IN3S	**PWDT1 IN3 输入选择** 00：UART0 RX 连接至 pwdt_in3 通道 01 ：UART1 RX 连接至 pwdt_in3 通道 10： UART2 RX 连接至 pwdt_in3 通道 11： ACMP0 OUT 连接至 pwdt_in3 通道 该字段选择 PWDT1_IN3 输入信号。
10:9 PWDT0IN3S	**PWDT0 IN3 输入选择** 00：UART0 RX 连接至 pwdt_in3 通道 01 ：UART1 RX 连接至 pwdt_in3 通道 10： UART2 RX 连接至 pwdt_in3 通道 11： ACMP0 OUT 连接至 pwdt_in3 通道 该字段选择 PWDT0_IN3 输入信号。
8：6 ADHWT1	**ADC 注入组硬件触发源** 000： RTC 溢出作为 ADC 硬件触发源 001： PWM0 初始化触发，具有 8 位可编程计数器延迟 010： PWM0 匹配触发，具有 8 位可编程计数器延迟 011： PWM1 初始化触发，具有 8 位可编程计数器延迟 100： PWM1 匹配触发，具有 8 位可编程计数器延迟 101： TIMER 通道 0 溢出作为 ADC 硬件触发 110 ：TIMER 通道 1 溢出作为 ADC 硬件触发 111 ：ACMP0 输出作为 ADC 硬件触发 选择 ADC 注入硬件触发源。所有触发源在上升沿开始转换。

15 循环冗余校验模块（CRC）

15.1 简介

循环冗余校验模块（CRC）用于生成 16 或 32 位 CRC 校验码，对数据进行校验。

15.2 特性

- 16 / 32 位校验模式
- 可编程多项式
- 输入按位 / 字节转置
- 输出按位 / 字节转置
- 结果按位取反

15.3 结构框图

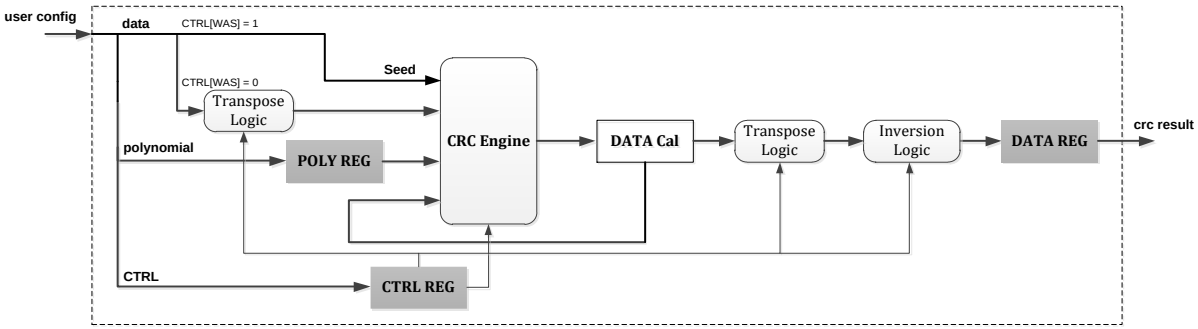

图 15-1 CRC 结构框图

15.4 功能描述

15.4.1 转置特性

转置功能默认不开启，CRC 按照正常的流程进行校验。

由于某些 CRC 标准要求对输入数据或最终校验结果进行转置，为了减少用户软件开销，CRC 模块具备硬件转置能力，用户软件可根据需要，选择配置转置功能。

输入数据转置使能后，写入的数据会先进行转置，后进行 CRC 计算。结果转置使能后，读出的数据是 CRC 校验结果转置后的数据。结果取反功能使能后，读出的数据是 CRC 校验结果转置并取反后的结果。

15.4.2 转置类型

CRC 模块提供了按位/字节翻转的多种转置功能类型，使用 CRC_CTRL[TOTW] 或 CRC_CTRL[TOTR] 进行配置。

转置功能配置选项：

1. CRC_CTRL[TOTW] 或 CRC_CTRL[TOTR] 为 00

   不发生转置。

2. CRC_CTRL[TOTW] 或 CRC_CTRL[TOTR] 为 01

   字节中的位转置，字节不转置：reg[31：0] 变为 {reg[24：31], reg[16：23], reg[8：15], reg[0：7]}。

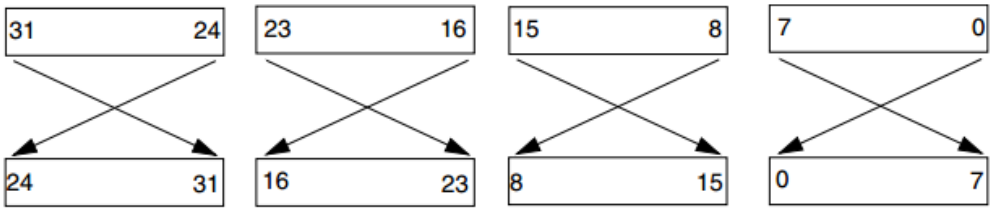

图 15-2 [TOTW] 或 [TOTR]为 01

3. CRC_CTRL[TOTW] 或 CRC_CTRL[TOTR] 为 10

   字节中的位和字节均转置：reg[31：0] 变为 {reg[0：7], reg[8：15],reg[16：23], reg[24：31]}.。

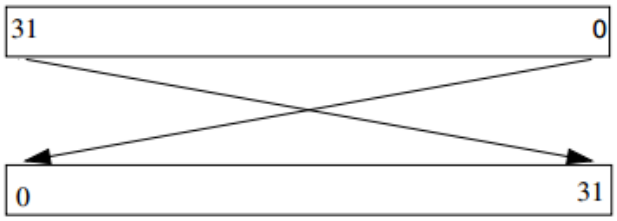

图 15-3 [TOTW] 或 [TOTR]为 10

4. CRC_CTRL[TOTW] 或 CRC_CTRL[TOTR] 为 11

   字节转置，但位不转置：reg[31：0] 变为 {reg[7：0], reg[15：8], reg[23：16], reg[31：24]}。

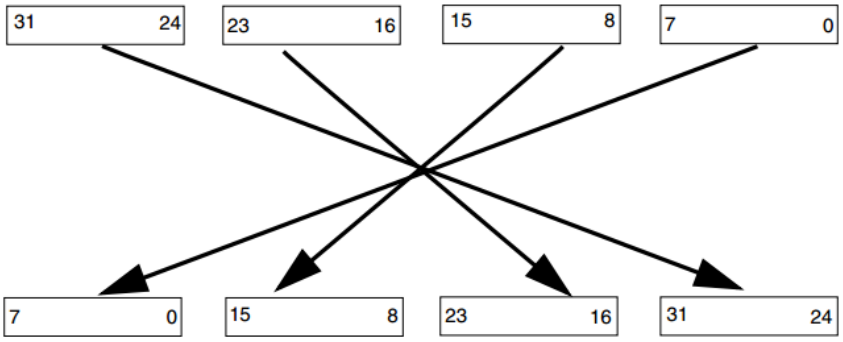

图 15-4 [TOTW] 或 [TOTR]为 11

注意

当向 CRC_DATA 寄存器一次写入 1 个字节时，不会发生按字节转置；

当向 CRC_DATA 寄存器一次写入 2 个字节时，对写入的 2 个字节进行转置；

当向 CRC_DATA 寄存器一次写入 4 个字节时，对写入的 4 个字节进行转置。

15.4.3 结果反码

CRC_CTRL[FXOR]置位时，读取的校验结果会先进行取反。CRC_CTRL[FXOR]复位时，读取的校验结果不会取反，得到原值。

15.5 应用说明

15.5.1 CRC 初始化

在进行对 CRC_POLY、CRC_DATA 寄存器进行操作前，首先必须对 CRC_CTRL 控制寄存器进行配置，选择使用的 CRC 模式 CRC16/32，[TCRC]选择 CRC 模式，[TOTW]使能写入数据转置，[TOTR]使能读取数据转置，[FXOR]使能读取数据取反，[WAS]选择对 CRC_DATA 寄存器写入的是种子还是校验数据。

15.5.2 CRC 多项式配置

此寄存器包含 CRC 校验时的多项式值。高半字段是 CRC 多项式的高 16 位，仅在 32 位 CRC 模式下有效，在 CRC16 模式下会忽略对高半字段的写操作。低半字段是 CRC 多项式的低 16 位，在 CRC16 / 32 模式下都会被使用。

15.5.3 CRC 校验

CRC _DATA 数据寄存器同时拥有种子、数据和校验和这 3 种角色。

当 CRC_CTRL[WAS] 置位时，对数据寄存器进行的任何写操作都被视为种子值。

当 CRC_CTRL[WAS] 清零后， 对数据寄存器进行的任何写操作都被视为写入用于 CRC 计算的数据。在 16 位 CRC 模式下，不使用高半字段来编程种子值；在 32 位 CRC 模式下，所有字段都用于种子值的编程。可以一次写入字节、半字或字，以 MSB 在前的顺序进行校验。

在写入所有数据后， 可以从该数据寄存器中读取 CRC 结果。在 16 位 CRC 模式下，低半字段是 CRC 校验结果。在 32 位 CRC 模式下，所有字段为校验结果，对 CRC_DATA 寄存器的读操作都会返回当前的 CRC 计算结果。

15.5.4 编程指南

16 位 CRC：

1. 清零 CRC_CTRL[TCRC] 使能 16 位 CRC 模式；
2. 按 CRC 计算要求对转置相关功能进行编程；
3. 将 16 位多项式写入 CRC_POLY 低半字段。CRC_POLY 高半字段在 16 位 CRC 模式下不可用；
4. 置位 CRC_CTRL[WAS] 进行种子值的编程；
5. 将 16 位种子写入 CRC_DATA[15：0]，CRC_DATA[31：16] 未使用；
6. 复位 CRC_CTRL[WAS] 开始逐个写入校验数据；
7. 将数据写入 CRC_DATA[31：0]，CRC 的中间计算结果都会保存在 CRC_DATA[15：0]；
8. 当所有数据都被写入后，从 CRC_DATA[15：0]中读取最终的 CRC 结果。

32 位 CRC：

1. 置位 CRC_CTRL[TCRC] 使能 32 位 CRC 模式；
2. 按 CRC 计算要求对转置相关功能进行编程；
3. 将 32 位多项式写入 CRC_POLY 多项式寄存器；
4. 置位 CRC_CTRL[WAS] 进行种子值的编程；
5. 将 32 位种子写入 CRC_DATA 寄存器；
6. 复位 CRC_CTRL[WAS] 开始逐个写入校验数据；
7. 将数据写入 CRC_DATA[31：0]，CRC 的中间计算结果都会保存在 CRC_DATA[31：0]；
8. 当所有数据都被写入后，从 CRC_DATA[31：0]中读取最终的 CRC 结果。

15.6 寄存器定义

表 15-1 CRC 寄存器映射

CRC 基地址：0x20081000

地址	名称	宽度	描述
CRC 基地址+0x00	CRC_DATA	32	数据寄存器
CRC 基地址+0x04	CRC_POLY	32	多项式寄存器
CRC 基地址+0x08	CRC_CTRL	32	控制寄存器

15.6.1 数据寄存器(CRC_DATA)

表 15-2 CRC_DATA 寄存器

CRC_DATA **数据寄存器** **Reset: 0xFFFFFFFF**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	DATA[31: 24]								DATA[23: 16]							
访问	RW								RW							
Reset	0xFF								0xFF							
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	DATA[15: 8]								DATA[7: 0]							
访问	RW								RW							
Reset	0xFF								0xFF							

字段	说明
31: 24 DATA	**数据字节 3** 在 16 位 CRC 模式 (CRC_CTRL[TCRC] 为 0)下，不用于种子值编程。 在 32 位 CRC 模式 (CRC_CTRL[TCRC] 为 1)下，当 CRC_CTRL[WAS] 为 1 时，写入该字段的值是种子值的一部分。当 CRC_CTRL[WAS] 为 0 时，写入该字段的数据参与 16 或 32 位 CRC 校验，生成 CRC 校验和。
23: 16 DATA	**数据字节 2** 在 16 位 CRC 模式 (CRC_CTRL[TCRC] 为 0)下，不用于种子值编程。 在 32 位 CRC 模式 (CRC_CTRL[TCRC] 为 1)下，当 CRC_CTRL[WAS] 为 1 时，写入该字段的值是种子值的一部分。当 CRC_CTRL[WAS] 为 0 时，写入该字段的数据参与 16 或 32 位 CRC 校验，生成 CRC 校验和。
15: 8 DATA	**数据字节 1** 当 CRC_CTRL[WAS] 为 1 时，写入该字段的值是种子值的一部分。当 CRC_CTRL[WAS] 为 0 时，写入该字段的数据参与 16 或 32 位 CRC 校验，生成 CRC 校验和。
7: 0 DATA	**数据字节 0**

字段	说明
	当 CRC_CTRL[WAS] 为 1 时，写入该字段的值是种子值的一部分。当 CRC_CTRL[WAS] 为 0 时，写入该字段的数据参与 16 或 32 位 CRC 校验，生成 CRC 校验和。

15.6.2 多项式寄存器(CRC_POLY)

表 15-3 CRC_POLY 寄存器

CRC_POLY 多项式寄存器 **Reset: 0x00001021**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	POLY[31：16]															
访问	RW															
Reset	0x0000															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	POLY[15：0]															
访问	RW															
Reset	0x1021															

字段	说明
31: 16 POLY	**多项式高半字** 该字段在 32 位 CRC 模式下可读写 (CRC_CTRL[TCRC] 为 1)。 该字段在 16 位 CRC 模式下不可写 (CRC_CTRL[TCRC] 为 0)。
15: 0 POLY	**多项式低半字** 该字段在 32 位与 16 位 CRC 模式下都可读写。

15.6.3 控制寄存器(CRC_CTRL)

表 15-4 CRC_CTRL 寄存器

CRC_CTRL 控制寄存器 **Reset: 0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									TOTW		TOTR			FXOR	WAS	TCRC
访问									RW		RW			RW	RW	RW
Reset									0		0			0	0	0

字段	说明
7: 6 TOTW	**写入转置类型** 定义写入 CRC_DATA 寄存器的数据转置配置。 00： 不转置 01： 字节中的位转置，但字节不转置 10： 字节中的位和字节均转置 11： 仅字节转置，字节中的位不转置
5: 4 TOTR	**读取转置类型** 定义读取 CRC_DATA 寄存器时的数据转置配置。 00： 不转置 01： 字节中的位转置，但字节不转置 10： 字节中的位和字节均转置 11： 仅字节转置，字节中的位不转置
2 FXOR	**读取取反** 某些 CRC 协议要求最终校验和与 0xFFFFFFFF 或 0xFFFF 进行异或运算。 0 ： 读取时不执行 XOR 运算 1 ： 对读到的数据执行 XOR 运算
1 WAS	**写入 CRC_DATA 类型** 定义项 CRC_DATA 寄存器写入时的类型。 0 ：写入 CRC 数据寄存器的是数据 1 ：写入 CRC 数据寄存器的是种子值
0 TCRC	**CRC 校验类型** 定义 CRC 校验的类型。 0：CRC16 1：CRC32

16 通用输入/输出（GPIO）

16.1 简介

通用输入输出 (GPIO) 模块可通过 APB 访问，还能通过 AHB 总线访问，以实现最高的引脚性能。GPIO 寄存器支持 APB 32 位访问，及 AHB 字节访问。

当引脚配置为 GPIO 功能时，端口配置寄存器 GPIO_CR 控制每个引脚的方向。端口输出数据寄存器 GPIO_ODR 控制每个引脚输出数据，也可以通过端口置位/复位寄存器 GPIO_BSRR，端口复位寄存器 GPIO_BRR 置位，控制 GPIO 输出的高低电平。

当引脚配置用于输入功能时，GPIO 输入数据寄存器显示每个引脚上的高低电平（1 代表高电平，0 代表低电平）。

MCU I/O 引脚通过多路复用器连接到外设/模块，多路复用器一次只允许一个外设的复用功能连接到 I/O 引脚。这样，共享同一个 I/O 引脚的外设之间不会发生冲突。每个 I/O 引脚都有一个多路复用器，可通过 GPIO_PINMUX 寄存器进行配置。当某一个外设/模块的功能需要从当前 IO 转移到另一个 IO 时，除了新的 IO 需要将复用功能连接到该外设/模块，原 IO 的复用配置也需要关闭，否则会导致外设/模块在新 IO 管脚工作异常。

16.2 特性

GPIO 引脚支持如下模式：

- 最多支持 42 个 I/O
- 输出状态：推挽或开漏 (与 I2C 有关)
- 输出数据来自输出寄存器 GPIO_ODR 或 外设 (可选功能输出)
- 每个 I/O 的驱动能力选择
- 输入状态：浮空，上拉/下拉，模拟(和 ADC 有关)
- 输入数据至输入数据寄存器 GPIO_IDR 或外设(可选功能输入)
- 位置位和复位寄存器 GPIO_BSRR 用于按位写入访问 GPIO_ODR
- 高灵活度的引脚复用，允许将 I/O 引脚用作 GPIO 或作为多种外设功能之一
- 可配置的上升沿或下降沿中断
- 低功耗模式唤醒中断

16.3 结构框图

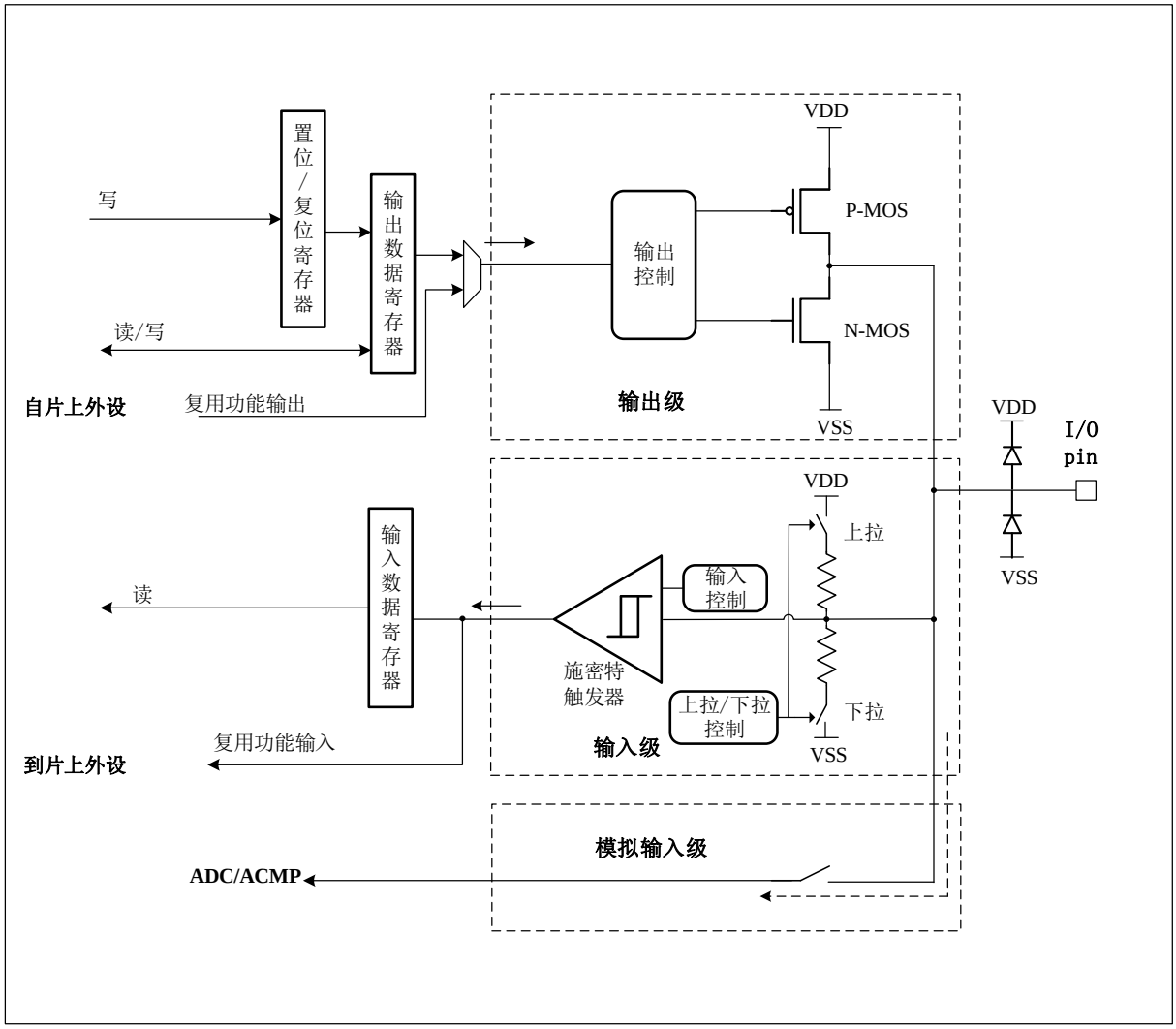

图 16-1 GPIO 结构框图

16.4 功能描述

16.4.1 外部中断

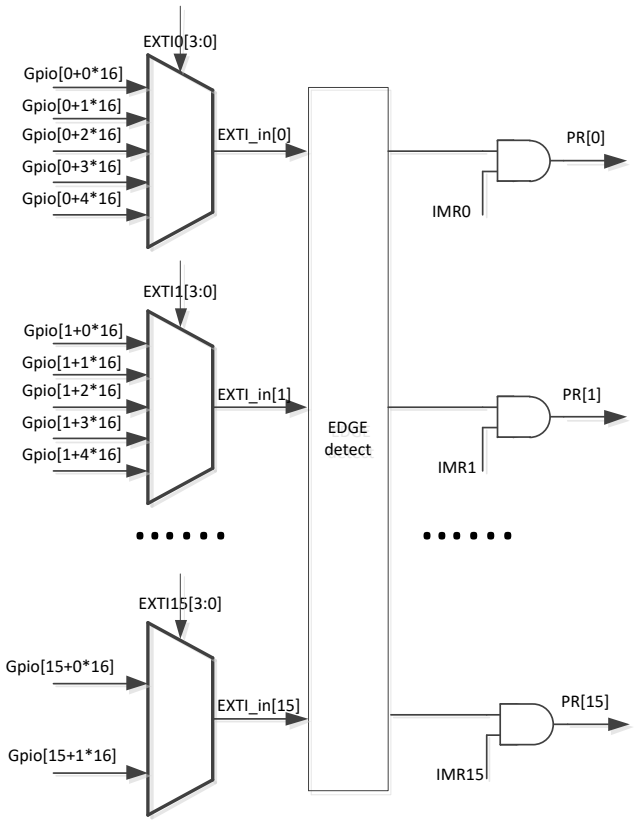

图 16-2 GPIO 外部中断

GPIO 以组的形式划分，每 16 个 IO 构成一组。以图中 GPIO[y+x*16]为例，‘y’表示某一组 IO 中的第 y 个 PIN 脚，x*16 表示第 x 组 GPIO。如 GPIO[1+0*16]表示 GPIOA_PIN_1，GPIO[15+1*16]表示 GPIOB_PIN_15。

外部中断线与中断向量的对应关系：

- 当 m≤2 时，EXTI_In[m]对应着中断向量 EXTIm_IRQn
- 当 3≤m≤8 时，EXTI_In[m]对应着中断向量 EXTI3_8_IRQn
- 当 9≤m≤15 时，EXTI_in[m]对应着中断向量 EXTI9_15_IRQn

GPIO 外部中断与中断处理函数对应关系如下表示。

表 16-1 GPIO 外部中断和中断处理函数对应关系

GPIO 引脚	中断标志位	中断处理函数
PA0~PC0	EXTI0	EXTI0_IRQHandler
PA1~PC1	EXTI1	EXTI1_IRQHandler
PA2~PC2	EXTI2	EXTI2_IRQHandler
PA3~PC3	EXTI3	EXTI3_8_IRQHandler
PA4~PC4	EXTI4	
PA5~PC5	EXTI5	
PA6~PC6	EXTI6	
PA7~PC7	EXTI7	
PA8~PC8	EXTI8	
PA9~PC9	EXTI9	EXTI9_15_IRQHandler
PA10~PB10	EXTI10	
PA11~PB11	EXTI11	
PA12~PB12	EXTI12	
PA13~PB13	EXTI13	
PA14~PB14	EXTI14	
PA15~PB15	EXTI15	

16.4.2 复用功能

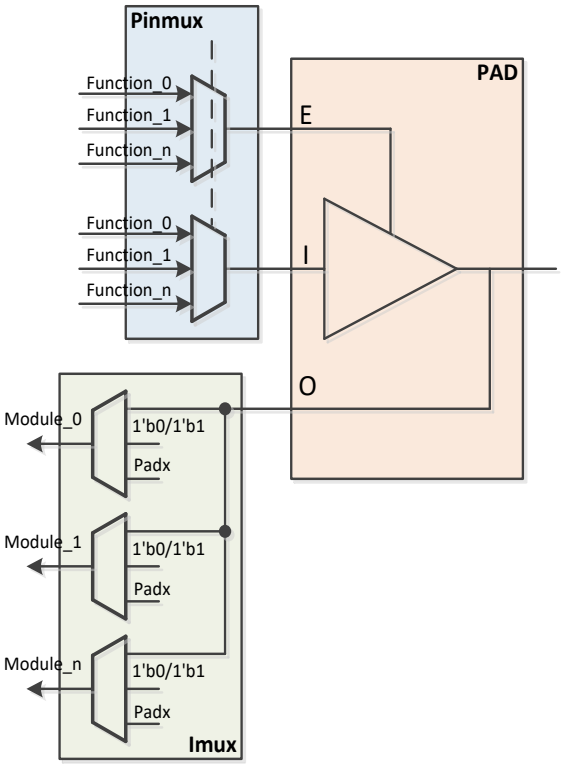

图 16-3 GPIO 复用功能

每个 IO 都有复用功能，如果要开始外设通信，应该先配置 GPIO 复用功能。每个 GPIO 的相应复用功能描述如下。

表 16-2 GPIO 复用功能描述

48 LQFP	32 HVQFN	20 TSSOP	引脚名称	功能 0	功能 1	功能 2	功能 3	复用功能配置	GPIO (序号)
1			PB11	PB11	PWM0_CH3	PB11	SPI1_MOSI	PMUX2[23:21]	27
2			PB12	PB12	PWM0_CH2	PB12	SPI1_SCK	PMUX2[26:24]	28
3	1		PB0	PB0	CAN_TX	PWM0_CH7	SPI1_MISO	PMUX1[20:18]	16
4	2		PB1	PB1	CAN_RX	PWM0_CH6	SPI1_NSS	PMUX1[23:21]	17
5	3	4	VDD1	VDD1					
6			VDDA	VDDA					
7	4	5	VSS1	VSS1					
8	5	6	PA12	PA12	I2C0_SCL	OSC_OUT[1]	PWM0_FLT0	PMUX1[8:6]	12
9	6	7	PA15	PA15	I2C0_SDA	OSC_IN[1]	PWDT0_IN0	PMUX1[17:15]	15
10	7	8	PA0	PA0	PWM0_CH1	UART0_RTS	I2C0_SCL	PMUX0[2:0]	0
11	8	9	PA1	PA1	PWM0_CH0	UART0_CTS	I2C0_SDA	PMUX0[5:3]	1
12			PB13	PB13	PWM0_CH7	PB13	I2C1_SCL	PMUX2[29:27]	29

48 LQFP	32 HVQFN	20 TSSOP	引脚名称	功能 0	功能 1	功能 2	功能 3	复用功能配置	GPIO (序号)
13	9		PB3	PB3	PWM0_CH6	PWM1_CH7	SPI0_MOSI	PMUX1[29:27]	19
14	10	10	PA2	PA2	PWM0_CH5	ADC_IN8	SPI0_MISO	PMUX0[8:6]	2
15	11	11	PA3	PA3	PWM0_CH4	ADC_IN7	SPI0_SCK	PMUX0[11:9]	3
16	12	12	PA4	PA4	PWM0_CH3	ADC_IN6/ ACMP_IN6	UART1_TX	PMUX0[14:12]	4
17	13	13	PA5	PA5	PWM0_CH2	ADC_IN5/ ACMP_IN5	UART1_RX	PMUX0[17:15]	5
18	14	14	PA6	PA6	BOOT[1]	PA6	PA6	PMUX0[20:18]	6
19			PB14	PB14	PWM0_CH1	PB14	SPI1_MOSI	PMUX3[2:0]	30
20			PB15	PB15	PWM1_FLT0	ADC_IN11	SPI1_SCK	PMUX3[5:3]	31
21			PC0	PC0	PWM1_CH3	ADC_IN10	SPI1_MISO	PMUX3[8:6]	32
22			PC1	PC1	PWM1_CH2	ADC_IN9	SPI1_NSS	PMUX3[11:9]	33
23	15		PB4	PB4	PWM1_CH1	ADC_IN8	SPI0_MISO	PMUX2[2:0]	20
24	16		PB5	PB5	PWM1_CH0	ADC_IN7	SPI0_SCK	PMUX2[5:3]	21
25	17	15	PA7	PA7	UART0_TX	ADC_IN4/ ACMP_IN4	SPI0_MOSI	PMUX0[23:21]	7
26	18	16	PA8	PA8	UART0_RX	ADC_IN3/ ACMP_IN3	SPI0_NSS	PMUX0[26:24]	8
27			PC2	PC2	UART1_TX	PWM0_FLT1	UART0_TX	PMUX3[14:12]	34
28			PC3	PC3	UART1_RX	PWM1_FLT1	UART0_RX	PMUX3[17:15]	35
29	19	17	PA9	PA9	PWM0_FLT0	ADC_IN2/ ACMP_IN2	RTC_CLKIN	PMUX0[29:27]	9
30	20		VSS2	VSS2					
31	21	18	VDD2	VDD2					
32			PC4	PC4	PWM0_CH1	PC4	I2C1_SDA	PMUX3[20:18]	36
33	22		PB6	PB6	PWM1_CH6	PWM1_FLT0	CAN_STDBY	PMUX2[8:6]	22
34			PC5	PC5	PC5	PWDT0_IN1	SPI0_NSS	PMUX3[23:21]	37
35	23		PB7	PB7	PWM1_CH3	ACMP_IN3	I2C0_SCL	PMUX2[11:9]	23
36	24		PB8	PB8	PWM1_CH2	PWDT0_IN2	I2C0_SDA	PMUX2[14:12]	24
37	25	19	PA10	PA10	PWM0_CH7	ADC_IN1/ ACMP_IN1	PWDT0_IN2	PMUX1[2:0]	10
38	26	20	PA11	PA11	PWM0_CH6	ADC_IN0/ ACMP_IN0	PWDT0_IN1	PMUX1[5:3]	11
39			PC6	PC6	UART1_TX	PC6	PWDT1_IN2	PMUX3[26:24]	38
40			PC7	PC7	UART1_RX	PC7	PWDT1_IN1	PMUX3[29:27]	39
41			PC8	PC8	PWM1_CH7	CAN_STDBY	PWDT1_IN0	PMUX4[2:0]	40
42			PC9	PC9	PWM1_CH6	PC9	ACMP_OUT	PMUX4[5:3]	41
43	27		PB9	PB9	PWM1_CH5	I2C1_SCL	UART2_TX	PMUX2[17:15]	25
44	28		PB10	PB10	PWM1_CH4	I2C1_SDA	UART2_RX	PMUX2[20:18]	26
45	29		PB2	PB2	NMI_B[1]	PWM0_FLT0	PWDT0_IN0	PMUX1[26:24]	18
46	30	1	PA13	PA13	SWD_CLK[1]		RTC_CLKOUT	PMUX1[11:9]	13

48 LQFP	32 HVQFN	20 TSSOP	引脚名称	功能 0	功能 1	功能 2	功能 3	复用功能配置	GPIO (序号)
47	31	2	RESET_B	RESET_B					
48	32	3	PA14	PA14	SWD_DIO[1]	ACMP_OUT	PWM1_CH0	PMUX1[14:12]	14

注意

1. **除了一些专用引脚外(如标注 [1])，所有引脚在第一次上电时默认为功能 0。**

   GPIO 控制寄存器中，除 GPIO_PINMUX 外，其他如 GPIO_CR，GPIO_IDR 等，x=0,1,2, 分别代表 PA,PB,PC，每个寄存器的低 16 位分别代表 Px0~Px15。

2. **输入/输出配置举例：设置 PC1 为输出模式 GPIO_CR[1] = 1。**
3. **复用功能配置以 48LQFP 举例：如果想要将 PB11 配置为 PWM0_CH3，应该设置 PMUX2[23:21]=1。**
4. **NMI_B, RESET_B 的后缀_B 代表的是低电平有效。**
5. **32 HVQFN 封装不支持 I2C1 和 SPI1 功能，即 PB0, PB1 无复用功能 3，PB9, PB10 无复用功能 2。**
6. **PA12 配置为 OSC_OUT 的复用功能后，不支持再切换为其它的复用功能。PA15 配置为 OSC_IN 的复用功能后，不支持再切换为其它的复用功能。**

16.5 应用说明

16.5.1 外部输入

外部中断/事件控制器由最多 16 个边沿检测器组成，用于生成事件/中断请求。每个输入线可以独立配置，以选择类型（事件或中断）和相应的触发事件（上升沿，下降沿或两者），每条线也可以独立屏蔽。相应中断输入线上的中断请求都记录在外部中断标志暂停寄存器 GPIO_PR 中。

16.5.2 复用功能

为了缩小封装，引脚通过信号多路复用提供多种可用的功能。复用功能选择寄存器 GPIO_PINMUX 控制外部引脚上的信号。请参考复用功能选择寄存器 GPIO_PINMUX 以及 16.4.2 复用功能 章节。

16.5.3 开漏输出

GPIO PIN 没有单独配置为开漏输出的寄存器，但是可以通过寄存器的组合配置以达到开漏输出的功能。

模拟开漏输出高（需要外接上拉电阻）：

1. 配置该 PIN 的 GPIO_CR 寄存器对应 BIT 为 0，输入模式；
2. 输出数据寄存器 GPIO_ODR 对应 BIT 设置为 0；
3. 将 GPIO_PU, GPIO_PD 寄存器对应的 BIT 配置为 0，不带上拉，也不带下拉。

模拟开漏输出低：

1. 配置该 PIN 的 GPIO_CR 寄存器对应 BIT 为 1，输出模式；
2. 输出数据寄存器 GPIO_ODR 对应 BIT 设置为 0；
3. 将 GPIO_PU, GPIO_PD 寄存器对应的 BIT 配置为 0，不带上拉，也不带下拉。

16.5.4 APB/AHB 访问

GPIO 模块除了可以通过 APB 总线访问，也可以通过 AHB 访问以实现最高的引脚性能。因此，GPIO 模块的寄存器可以通过 APB 地址/AHB 地址进行访问，具体地址如下表。

表 16-3 GPIO APB/AHB 地址

GPIO 模块	APB 地址	AHB 地址
GPIOA 首地址	0x40001000	0x20080000
GPIOB 首地址	0x40001030	0x20080030
GPIOC 首地址	0x40001060	0x20080060

注意

GPIO 各寄存器针对首地址的偏移，不管是 APB, 还是 AHB, 偏移地址都是一样。例如，端口输入数据寄存器 GPIO_IDR 的 offset 为 0x04, 因此寄存器 GPIOA_IDR 通过 APB 访问的地址为 0x40001004, 通过 AHB 访问的地址为 0x20080004。

16.5.5 GPIO 功能

在复位期间和刚刚复位后，GPIO 功能处于活动状态，I/O 端口配置为不带上下拉输入模式，RST 和 ARM 调试接口相关引脚除外。当进入 Stop 模式前，如果 IO 在不带上下拉输入的状态下，需要外部给一个高或低的稳定电平，避免 IO 电平不稳定导致漏电。

用户可以对 GPIO_PINMUX 寄存器进行编程，将 I/O 端口从其他功能更改为 GPIO 功能。

当引脚配置为输出时，写入输出数据寄存器的值将输出到 I/O 引脚。可以在推挽模式或开漏模式（当输出为 0 时，仅激活 N-MOS）下使用输出驱动器。输入数据寄存器 GPIO_IDR 在每个 AHB 时钟周期捕获 I/O 引脚上的数据。

所有 GPIO 引脚都具有弱内部上拉和下拉电阻，可以激活或不激活，具体取决于 GPIO_PU 和 GPIO_PD 寄存器中的值。

16.5.6 编程指南

首先，在复位后，除 RST 和 ARM 调试接口外，所有 I/O 端口都处于 GPIO 输入模式。

其次，软件可以编程复用功能选择寄存器 GPIO_PINMUX 寄存器来映射/重映射 I/O 功能。

作为 GPIO 功能的 I/O，软件可以编程外部中断。当 MCU 处于低功耗模式时，外部中断使用内部 32K 振荡器产生的时钟。

16.6 寄存器定义

表 16-4 GPIO 寄存器映射

GPIOA 基地址： 0x40001000
GPIOB 基地址： 0x40001030
GPIOC 基地址： 0x40001060

地址	名称	宽度	描述
GPIOx 基地址 + 0x00	GPIO_CR	32	端口配置寄存器
GPIOx 基地址 + 0x04	GPIO_IDR	32	端口输入数据寄存器
GPIOx 基地址 + 0x08	GPIO_ODR	32	端口输出数据寄存器
GPIOx 基地址 + 0x0C	GPIO_BSRR	32	端口置位/复位寄存器
GPIOx 基地址 + 0x10	GPIO_BRR	32	端口复位寄存器
GPIOx 基地址 + 0x18	GPIO_PD	32	下拉使能寄存器
GPIOx 基地址 + 0x1C	GPIO_PU	32	上拉使能寄存器
GPIOx 基地址 + 0x20	GPIO_E4_E2	32	驱动能力选择寄存器
GPIOA 基地址 + 0x140 GPIOA 基地址 + 0x144 GPIOA 基地址 + 0x148 GPIOA 基地址 + 0x14C GPIOA 基地址 + 0x150	GPIO_PINMUX	32	复用功能选择寄存器：共有 5 个寄存器
GPIOA 基地址 + 0x160	GPIO_PR	32	外部中断标志暂停寄存器
GPIOA 基地址 + 0x164	GPIO_IMR	32	中断掩码寄存器
GPIOA 基地址 + 0x168	GPIO_RTSR	32	上升沿触发事件配置寄存器
GPIOA 基地址 + 0x16C	GPIO_FTSR	32	下降沿触发事件配置寄存器

地址	名称	宽度	描述
GPIOA 基地址 + 0x170 GPIOA 基地址 + 0x174 GPIOA 基地址 + 0x178 GPIOA 基地址 + 0x17C	GPIO_EXTICR	32	外部中断寄存器：共有 4 个寄存器

【说明】上表中，x=A、B、C。

16.6.1 端口配置寄存器(GPIO_CR)

表 16-5 GPIO_CR 寄存器

GPIO_CR **端口配置寄存器** **Reset：0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	MODE(16*x+15)	MODE(16*x+14)	MODE(16*x+13)	MODE(16*x+12)	MODE(16*x+11)	MODE(16*x+10)	MODE(16*x+9)	MODE(16*x+8)	MODE(16*x+7)	MODE(16*x+6)	MODE(16*x+5)	MODE(16*x+4)	MODE(16*x+3)	MODE(16*x+2)	MODE(16*x+1)	MODE(16*x+0)
访问	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

字段	说明
[15：0] MODE	**Mode(y)： 端口 y 配置位** 0： 输入(复位默认状态) 1： 输出模式 这些位由软件写入，以配置 I/O 方向模式。

16.6.2 端口输入数据寄存器 (GPIO_IDR)

表 16-6 GPIO_IDR 寄存器

GPIO_IDR 端口输入数据寄存器 Reset：0x00000000

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	IDR(16*x+15)	IDR(16*x+14)	IDR(16*x+13)	IDR(16*x+12)	IDR(16*x+11)	IDR(16*x+10)	IDR(16*x+9)	IDR(16*x+8)	IDR(16*x+7)	IDR(16*x+6)	IDR(16*x+5)	IDR(16*x+4)	IDR(16*x+3)	IDR(16*x+2)	IDR(16*x+1)	IDR(16*x+0)
访问	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

字段	说明
[15：0] IDR	**IDR(y)： 端口 y 输入数据** 这些位只读，包含相应 I/O 端口的输入值 。

16.6.3 端口输出数据寄存器(GPIO_ODR)

表 16-7 GPIO_ODR 寄存器

GPIO_ODR 端口输出数据寄存器 Reset：0x00000000

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	ODR(16*x+15)	ODR(16*x+14)	ODR(16*x+13)	ODR(16*x+12)	ODR(16*x+11)	ODR(16*x+10)	ODR(16*x+9)	ODR(16*x+8)	ODR(16*x+7)	ODR(16*x+6)	ODR(16*x+5)	ODR(16*x+4)	ODR(16*x+3)	ODR(16*x+2)	ODR(16*x+1)	ODR(16*x+0)
访问	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

字段	说明
[15：0] ODR	**ODR(y)： 端口(y) 输出数据** 这些位可以通过软件读写。 **注意：对于 PIN 置位/复位，可以通过写入 GPIOx_BSRR 寄存器 (x = A, B, C) 和 GPIOx_BRR 寄存器 (x = A, B, C) 来单独设置和复位 ODR 位。**

16.6.4 端口置位/复位寄存器(GPIO_BSRR)

表 16-8 GPIO_BSRR 寄存器

GPIO_BSRR **端口置位/复位寄存器** **复位值：0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	BR(16*x+15)	BR(16*x+14)	BR(16*x+13)	BR(16*x+12)	BR(16*x+11)	BR(16*x+10)	BR(16*x+9)	BR(16*x+8)	BR(16*x+7)	BR(16*x+6)	BR(16*x+5)	BR(16*x+4)	BR(16*x+3)	BR(16*x+2)	BR(16*x+1)	BR(16*x+0)
访问	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	BS(16*x+15)	BS(16*x+14)	BS(16*x+13)	BS(16*x+12)	BS(16*x+11)	BS(16*x+10)	BS(16*x+9)	BS(16*x+8)	BS(16*x+7)	BS(16*x+6)	BS(16*x+5)	BS(16*x+4)	BS(16*x+3)	BS(16*x+2)	BS(16*x+1)	BS(16*x+0)
访问	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

字段	说明
[31:16] BR	**BR(y)： Port(y)复位位 y** 0：对应的 ODRy 位无操作 1：复位对应的 ODRy 位 这些位只写， 对这些位的读取返回 0x0000。该寄存器只支持 APB 访问，不支持 AHB 访问。
[15：0] BS	**BS(y)： Port(y) 设置位 y** 0：对应的 ODRy 位无操作 1：置位对应的 ODRy 位 这些位是只写的，可以在字、半字或字节模式下访问。对这些位的读取返回 0x00000。 **注意 ：如果 BS 和 BR 都配置，则 BR 具有更高的优先级。**

16.6.5 端口复位寄存器(GPIO_BRR)

表 16-9 GPIO_BRR 寄存器

GPIO_BRR 端口复位寄存器 Reset：0x00000000

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	BR(16*x+15)	BR(16*x+14)	BR(16*x+13)	BR(16*x+12)	BR(16*x+11)	BR(16*x+10)	BR(16*x+9)	BR(16*x+8)	BR(16*x+7)	BR(16*x+6)	BR(16*x+5)	BR(16*x+4)	BR(16*x+3)	BR(16*x+2)	BR(16*x+1)	BR(16*x+0)
访问	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

字段	说明
[15：0] BR	**BR(y)： Port(y) 复位位 y** 0：对应的 ODRy 位无操作 1：复位对应的 ODRy 位 这些位是只写的，对这些位的读取返回 0x0000。

16.6.6 下拉使能寄存器(GPIO_PD)

表 16-10 GPIO_PD 寄存器

GPIO_PD 下拉使能寄存器 Reset：0x00000000

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	PD(16*x+15)	PD(16*x+14)	PD(16*x+13)	PD(16*x+12)	PD(16*x+11)	PD(16*x+10)	PD(16*x+9)	PD(16*x+8)	PD(16*x+7)	PD(16*x+6)	PD(16*x+5)	PD(16*x+4)	PD(16*x+3)	PD(16*x+2)	PD(16*x+1)	PD(16*x+0)
访问	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

字段	说明
[0] [1] [2] ……	**PD (y)： 下拉使能** 0：禁用下拉 1：使能下拉（下拉电阻典型值 75 KΩ）

字段	说明
PD	这些位可以通过软件进行读写。 **注意：上拉和下拉不支持同时使能。**

16.6.7 上拉使能寄存器(GPIO_PU)

表 16-11 GPIO_PU 寄存器

GPIO_PU **上拉使能寄存器** **Reset：0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	PU(16*x+15)	PU(16*x+14)	PU(16*x+13)	PU(16*x+12)	PU(16*x+11)	PU(16*x+10)	PU(16*x+9)	PU(16*x+8)	PU(16*x+7)	PU(16*x+6)	PU(16*x+5)	PU(16*x+4)	PU(16*x+3)	PU(16*x+2)	PU(16*x+1)	PU(16*x+0)
访问	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

字段	说明
[0] [1] [2] …… PU	**PU (y)：** 上拉使能 0：禁用上拉 1：使能上拉（上拉电阻典型值 75 KΩ） 这些位可以通过软件进行读写。 **注意：GPIOC 该寄存器的默认值 0x00000400，因为 bit10 是 RESET_b 引脚默认带上拉功能。上拉和下拉不支持同时使能。**

16.6.8 驱动能力选择寄存器 (GPIO_E4_E2)

表 16-12 GPIO_E4_E2 寄存器

GPIO_E4_E2 **驱动能力选择寄存器** **Reset：0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	E4_E2(16*x+15)		E4_E2(16*x+14)		E4_E2(16*x+13)		E4_E2(16*x+12)		E4_E2(16*x+11)		E4_E2(16*x+10)		E4_E2(16*x+9)		E4_E2(16*x+8)	
访问	RW		RW		RW		RW		RW		RW		RW		RW	
Reset	0		0		0		0		0		0		0		0	
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	E4_E2(16*x+7)		E4_E2(16*x+6)		E4_E2(16*x+5)		E4_E2(16*x+4)		E4_E2(16*x+3)		E4_E2(16*x+2)		E4_E2(16*x+1)		E4_E2(16*x+0)	
访问	RW		RW		RW		RW		RW		RW		RW		RW	
Reset	0		0		0		0		0		0		0		0	

字段	说明
[1：0] [3：2] [5：4] …… E4_E2	**E4_E2 (y)：** 驱动能力选择 00：4mA 01：8mA 10：12mA 11： 16mA 这些位可以通过软件进行读写。

16.6.9 复用功能选择寄存器(GPIO_PINMUX)

表 16-13 GPIO_PINMUX 寄存器

GPIO_PINMUX **复用功能选择寄存器** **Reset：0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称			PINMUXx[29：27]			PINMUXx[26：24]			PINMUXx[23：21]			PINMUXx[20：18]			PINMUXx[17：15]	
访问			RW			RW			RW			RW			RW	
Reset			0			0			0			0			0	
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称		PINMUXx[14：12]			PINMUXx[11：9]			PINMUXx[8：6]			PINMUXx[5：3]			PINMUXx[2：0]		
访问		RW			RW			RW			RW			RW		
Reset		0			0			0			0			0		

字段	说明
[2：0] [5：3] [8：6] …… PINMUXx	**PINMUXx (y)：** 复用功能 000：功能 0 001：功能 1 010：功能 2 011：功能 3 这些位由软件写入，以配置复用功能 I/O。 注：GPIO_PINMUX0 寄存器默认值为 0x00040000； GPIO_PINMUX1 寄存器默认值为 0x01011280； GPIO_PINMUX2 寄存器默认值为 0x00000000； GPIO_PINMUX3 寄存器默认值为 0x00000000； GPIO_PINMUX4 寄存器默认值为 0x00000000.

16.6.10 外部中断标志暂停寄存器(GPIO_PR)

表 16-14 GPIO_PR 寄存器

GPIO_PR 外部中断标志暂停寄存器 Reset：0x00000000

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	PR(15)	PR(14)	PR(13)	PR(12)	PR(11)	PR(10)	PR(9)	PR(8)	PR(7)	PR(6)	PR(5)	PR(4)	PR(3)	PR(2)	PR(1)	PR(0)
访问	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

字段	说明
[0] [1] [2] …… PR	**PR (y)：** 外部中断标志暂停位 0： 没有发生触发请求 1： 发生所选的触发请求 当所选边沿事件达到外部中断线时，该位置 1。通过向该位写 1 来清除该位。

16.6.11 中断掩码寄存器(GPIO_IMR)

表 16-15 GPIO_IMR 寄存器

GPIO_IMR 中断掩码寄存器 Reset：0x00000000

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

字段	说明
[0] [1] [2] …… IMR	**IMR (y)：** y 输入线的中断掩码 0：来自 y 输入线的中断请求被屏蔽 1：来自 y 输入线的中断请求未被屏蔽

16.6.12 上升沿触发事件配置寄存器(GPIO_RTSR)

表 16-16 GPIO_RTSR 寄存器

GPIO_RTSR 上升沿触发事件配置寄存器 Reset：0x00000000

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

字段	说明
[0]	**RTSR (y)： y 输入线的上升沿触发事件配置**
[1]	
[2]	0：输入线 y 上升沿触发禁用（对于事件和中断）
……	1：输入线 y 上升沿触发使能（对于事件和中断）
RTSR	

16.6.13 下降沿触发事件配置寄存器(GPIO_FTSR)

表 16-17 GPIO_FTSR 寄存器

GPIO_FTSR 下降沿触发事件配置寄存器 Reset：0x00000000

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

字段	说明
[0]	**FTSR (y)：y 输入线的下降沿触发事件配置**
[1]	
[2]	0：输入线 y 下降沿触发禁用（对于事件和中断）
……	1：输入线 y 下降沿触发使能（对于事件和中断）
FTSR	

16.6.14 外部中断寄存器(GPIO_EXTICR)

表 16-18 GPIO_EXTICR 寄存器

GPIO_EXTICR 外部中断寄存器 **Reset：0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	EXTI(4*x+3)				EXTI(4*x+2)				EXTI(4*x+1)				EXTI(4*x+0)			
访问	RW				RW				RW				RW			
Reset	0				0				0				0			

字段	说明
[3：0]	**EXTI (y)： EXTI y 配置**
[7：4]	
[11：8]	0000： PA[x] 引脚
……	0001： PB[x] 引脚
EXTI	0010： PC[x] 引脚
	这些位由软件写入，用于选择 EXTI(y)外部中断源输入。

17 I2C 总线模块（I2C）

17.1 简介

I2C 总线是一种简单、双向二线制同步串行总线，通过时钟线（SCL）与数据线（SDA）进行数据传输。SCL 是由主机驱动产生的时钟信号，SDA 是双向数据信号，既可由主设备产生，也可由从设备产生。

17.2 特性

- 主从一体模式
- 标准 100kHz、快速 400kHz、快速+ 1MHz 模式
- 支持 7bit 范围地址
- 支持 10bit 扩展地址
- 支持从机 Stretch
- 支持从机低功耗模式唤醒
- 支持从机监测功能
- 支持多主机仲裁
- 主机仲裁丢失切换为从机
- 可编程毛刺滤波器
- 总线开始/停止信号检测
- 软件控制应答信号
- DMA 发送/接收
- 地址匹配中断
- 字节传输中断
- 无应答中断
- 发送/接收溢出中断

17.3 结构框图

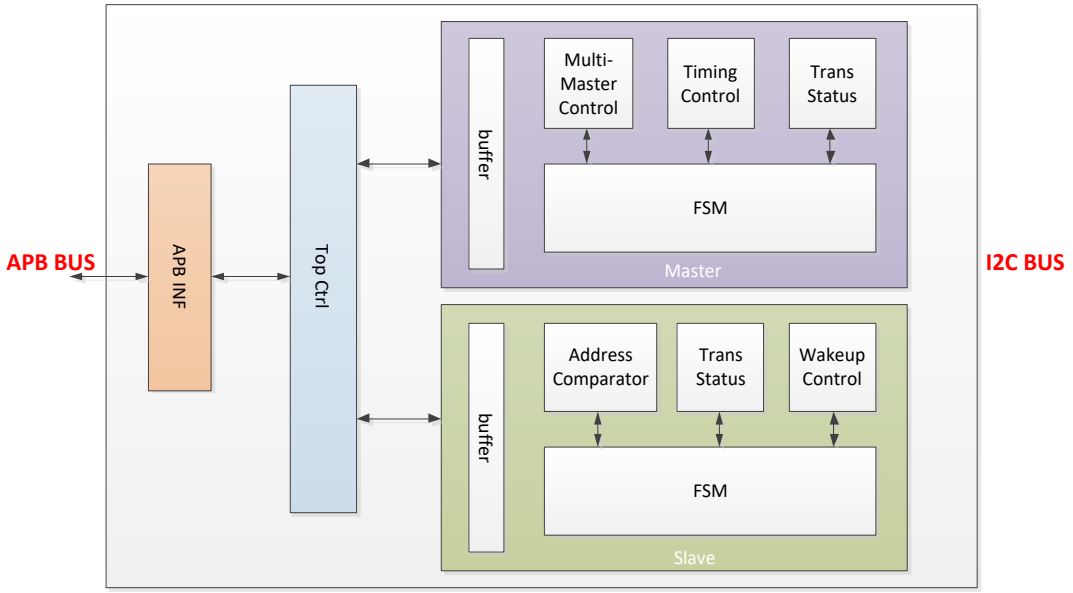

图 17-1 I2C 结构框图

17.3.1 I2C 信号组成

I2C 通讯以起始信号开始，以停止信号结束。当 SCL 为高电平时，SDA 线上高电平到低电平的跳变定义了 START 条件。当 SCL 为高电平时，SDA 线上低电平到高电平的跳变定义了 STOP 条件。

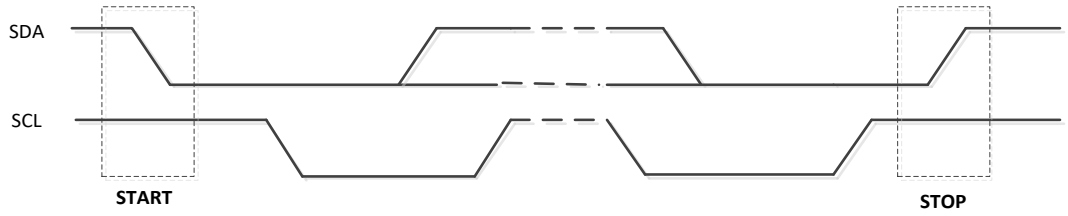

图 17-2 START 和 STOP 条件

每帧数据由 9 位组成，8 位数据（MSB 在前）和 1 位应答信号，传输次数不受限制。

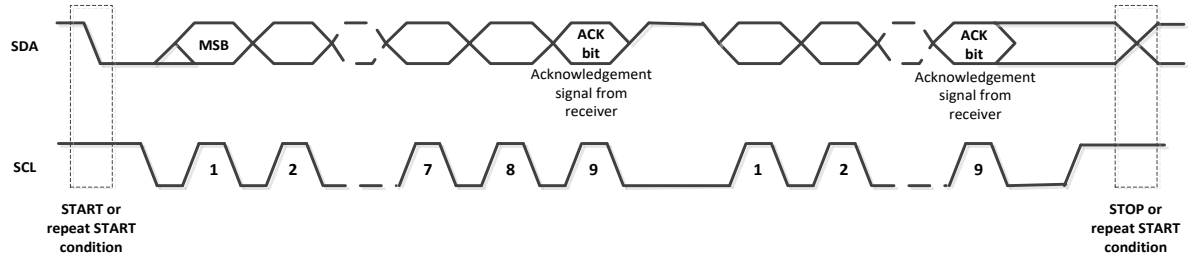

图 17-3 数据传输格式

起始信号后的第一个数据是地址字节，地址字节后的继续传输的数据都是数据字节。
7bit 地址模式下，I2C 协议规定地址字节的高 7 位是从机的地址，最低位为 0，表示写入从机，最高位为 1，表示读取从机。
10bit 地址模式下，地址由两个字节组成。
写从机时，I2C 协议规定：第一个字节发送 11110XX0，高 5 位固定为 11110，bit2，bit1 是 10bit 地址中的高 2 位，bit0 是方向位，值为 0，表示方向为写；第二个字节是 10bit 地址的低 8 位。
读从机时，I2C 规定：先发写的地址（两个字节），再发读的地址（读的地址只需要发送一个字节）。具体流程如下：先发写的地址，第一个字节发送 11110XX0，高 5 位固定为 11110，bit2、bit1 是 10bit 地址的高 2 位，bit0 位方向位，值为 0，表示方向位写，第二个字节是 10bit 地址的低 8 位；再发写的地址，发送 11110XX1，bit2、bit1 是 10bit 地址中的高 2 位，bit0 位方向位，值为 1，表示方向位读。

17.3.2 波特率组成

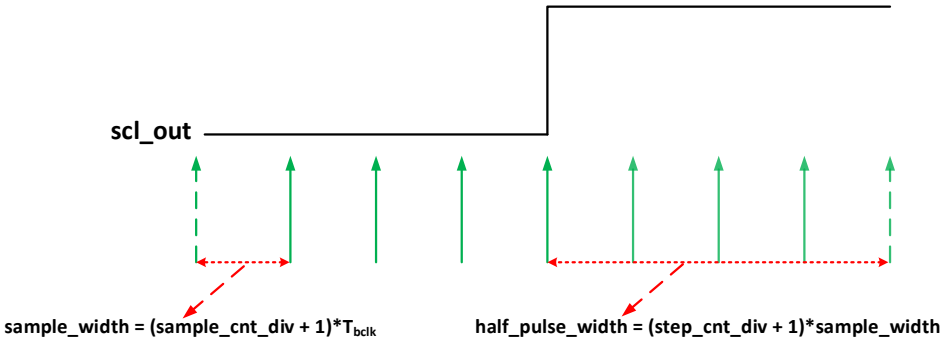

图 17-4 波特率生成

波特率： $f_{SCL}=f_{bclk}/(((SAMPLE\_CNT\_DIV+1)*(STEP\_CNT\_DIV+1))*2)$

f_{bclk} 是 APB 总线时钟频率。

17.3.3 数据流程

作为发送器，数据被写入一个 8 位的发送缓冲区。然后由波特率控制器加载到发送移位寄存器。MSB 在前，发送器会在每个字节的第 9 个 SCL 采样应答位。

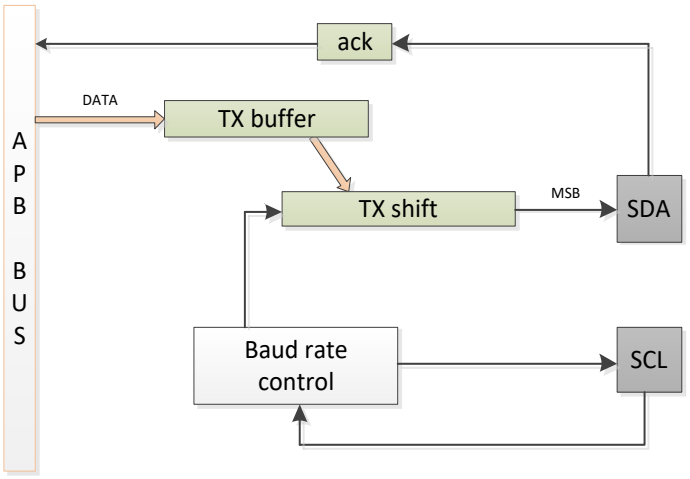

图 17-5 发送器数据流程

对于接收器，接收移位寄存接收每个比特位并依次传入移位寄存器，数据在每第 8 个 SCL 之后将被存储到 8 位接收缓冲器中，MSB 在前。应答位在第 9 个 SCL 驱动到 SDA 线上。

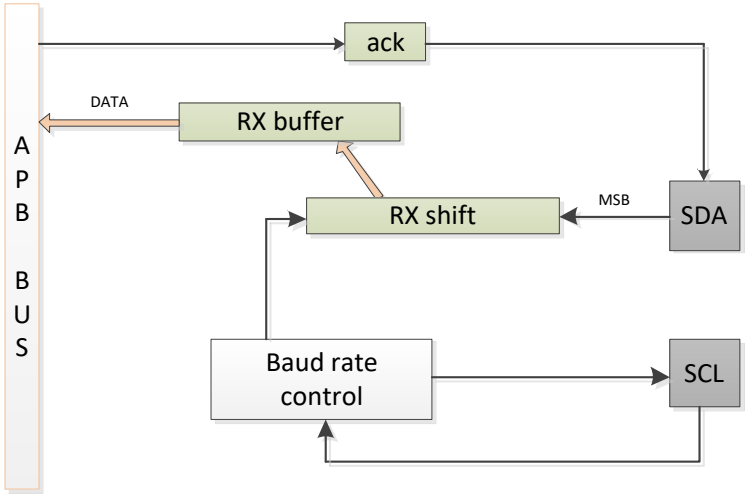

图 17-6 接收器数据流程

17.4 功能描述

I2C 模块支持标准、快速与高速通信模式，I2C 外设同一时刻只能当做主机或从机模式使用。一旦配置为从机模式，除非模块复位，则会一直处于从机模式下。

17.4.1 主机模式

在主机模式下，主机发送起始信号后，每发送完一字节数据，BND 标志位都会置位。
主机可发送重复起始信号，在不发送 STOP 信号前提下，重新对从机寻址并写入或读取数据，如图 17-7 所示。

主机支持时钟同步，在多主从机通讯或者有从机 stretch 总线时，能与其它主从机的时钟进行同步：当主机 SCL 输出高时，不会开始它的高电平时间计数，而是等待总线的 SCL 被所有主机拉高后开始计数，这样实现了多主机之间的时钟同步。此时实际的 SCL 时钟频率会变慢，因为 SCL 拉高是通过外部上拉电阻充电拉高，此时的频率取决于外部总线电容和上拉电阻的取值。

支持主机仲裁，在多主机通讯情况下，I2C 模块对总线上传输的数据进行逐 bit 比较。当主机发送 1 而检测到总线数据为 0 时，仲裁丢失，置位仲裁丢失标志，可产生中断请求，并将自己切换为从机模式，因为此时其它主机可能在寻址自己。该功能的前提条件是时钟同步功能先打开。

在 DMA 模式下，发送或者接收时，DMA 的使能信号会屏蔽 BND 中断产生。具体是：DMA 发射时，BND 标志都不会置位，不会产生 BND 中断，当最后一个字节发送完成时，等最后一字节传输完后，BND 标志置起，关闭 DMA，可以产生 BND 中断。

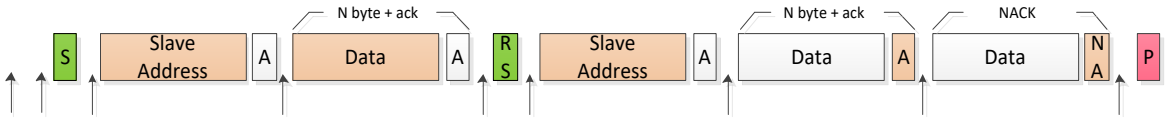

图 17-7 主机组合模式

17.4.2 从机模式

在从机模式下，从机支持 7bit 地址、10bit 地址、7bit 范围地址、广播地址四种地址匹配模式。

从机在以上四种模式下地址得到匹配后，SAMF 标志位置位，SRW 标志位根据主机的写/读信号置 0/1。地址匹配成功后，主机在地址字节后传输的每一个字节数据，从机的 BND 标志位都会置位，当主机在读取从机时，发送了 NACK 信号，后续若主机继续传输，从机不会产生 BND 标志，需要重新被寻址，才能与从机通讯。

从机具备 stretch 功能，使能后，从机收到一字节数据而软件没有读取，或者主机读取时从机还没有将数据准备好，从机将主动拉低 SCL 总线，防止主机发出下一帧数据的时钟信号，当然主机需要使能时钟同步功能，以等待从机释放 SCL 总线。

从机具有四种状态标志：

- **发送缓存区空标志 TXEF**，当从机发送寄存器未载入发送数据或一字节发送完成时，此标志置位，可以产生中断请求。
- **发送缓存区溢出标志 TXUF**，当从机发送完一字节数据，但没有写入下一字节数据，而此时主机再次发完一字节时钟信号后，此标志置位，可以产生中断请求。
- **接收缓存区满标志 RXFF**，当从机数据寄存器收到一字节数据后，此标志置位，可以产生中断请求。
- **接收缓存区溢出标志 RXOF**，当从机接收完一字节数据后，若软件没有及时读取数据，而此时主机再次发完一字节时钟信号后，此标志置位，可以产生中断请求。

17.4.3 中断请求

I2C 模块共有 10 个中断。

表 17-1 I2C 中断汇总

中断源	标志	本地使能	全局使能
完成 1 个字节的传输	BND		IICIE
从机地址匹配	SAMF		IICIE
仲裁丢失	ARBLOST		IICIE
总线起始信号检测	START	SSIE	IICIE
总线停止信号检测	STOP	SSIE	IICIE
RX 缓冲区溢出	RXOF	RXOFIE	IICIE
TX 缓冲区溢出	TXUF	TXUFIE	IICIE
RX 缓冲区满	RXFF	RXFIE	IICIE
TX 缓冲区空	TXEF	TXEIE	IICIE
获得应答	RACK	NACKIE	IICIE

在 DMA 传输方式下，不会响应 BND, RXFF 及 TXEF 的中断请求。

17.4.4 DMA 发送/接收

I2C 模块支持逐字节 DMA TX/RX 请求传输。DMA 仅用于数据传输阶段， START、STOP、RESTART 信号以及发地址信号仍需由软件操作完成。

17.4.4.1 主机发送器

在 Master DMA 发送模式下，地址匹配后，DMA 会自动搬运内存的数据至 I2C 数据寄存器。后续每传输完一个字节数据时检测 ACK 信号，当 ACK 时，产生 DMA 请求，搬运下一字节数据，当检测为 NACK 时，DMA 不会进行后续的传输，如果 NACKIE=1，将产生中断请求。在最后一字节传输完成后， BND 标志会置起，若 DMA 保持使能，不会产生 BND 中断，DMA Disable 后，产生 BND 中断。

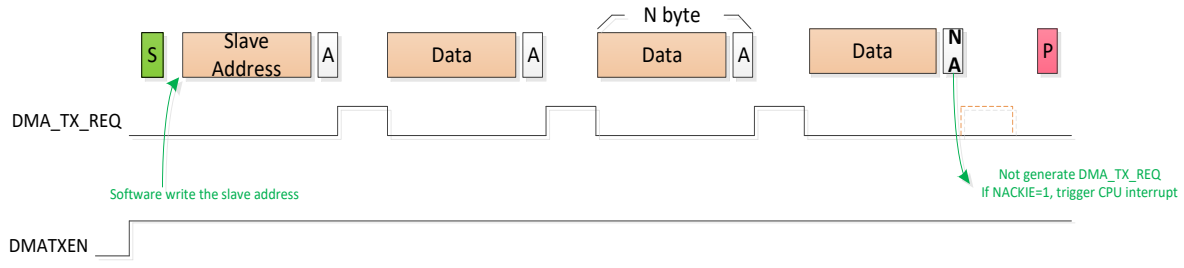

图 17-8 主机发送器情形 1

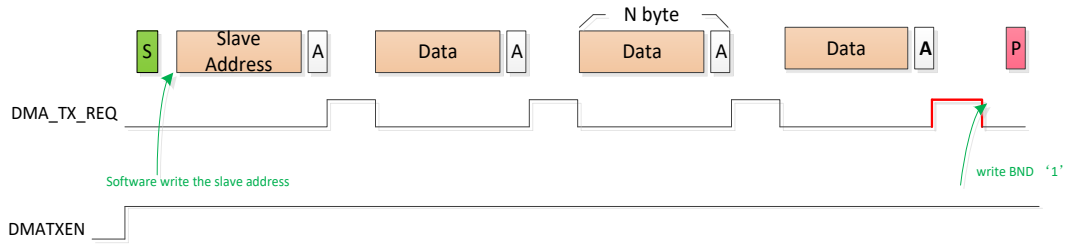

图 17-9 主机发送器情形 2

17.4.4.2 主机接收器

在 Master DMA 接收模式下，**地址匹配后，必须先手动清除 BND 标志**，然后将数据传输方向转为读取，并执行空读操作产生时钟信号，一字节传输完毕后 BND 标志置起会请求 DMA 将数据搬运到内存。DMA 会根据设定的传输长度自动在最后一字节回 NACK。

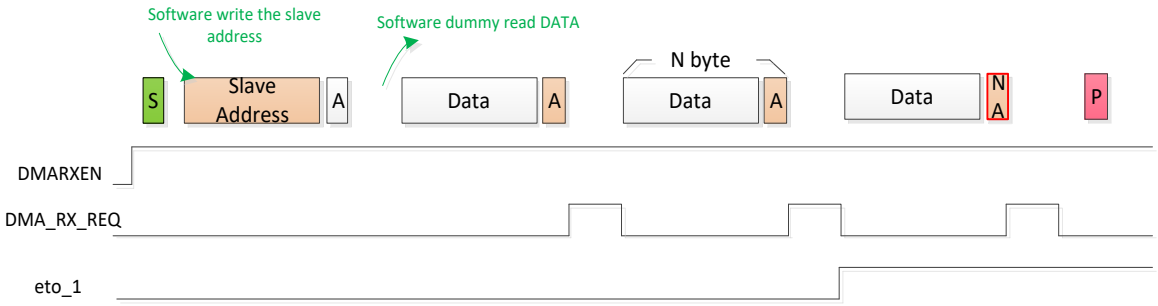

图 17-10 主机接收器

17.4.4.3 从机发送器

在 Slave DMA 发送模式下，如果地址匹配且为读信号，则触发 DMA 将数据搬运至数据寄存器。并检测后续每一字节的 ACK 信号，收到 ACK 信号时会产生 DMA 搬运请求，收到 NACK 信号时，DMA 会停止搬运，若 NACKIE=1，产生中断请求。在最后一字节传输完成后，BND 标志置位，若 DMA 保持使能，不会产生 BND 中断，DMA Disable 后，产生 BND 中断请求。

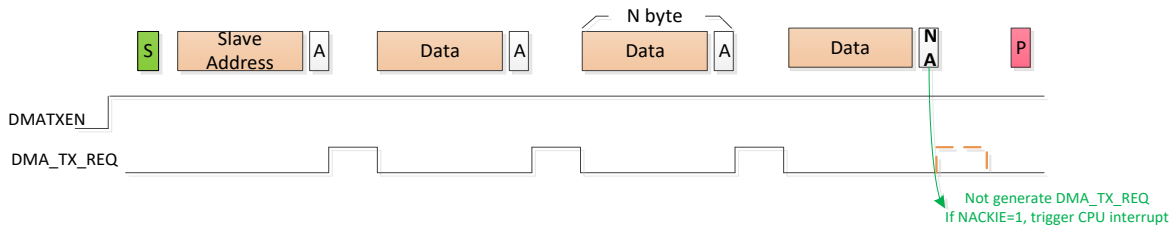

图 17-11 从机发送器

17.4.4.4 从机接收器

在 Slave DMA 接收模式下，**地址匹配后，必须先手动清除地址匹配标志**，然后每当接收到一字节数据产生的 BND 标志都将触发 DMA 将数据搬运到内存。DMA 会根据设定的传输长度自动在最后一字节回 NACK。

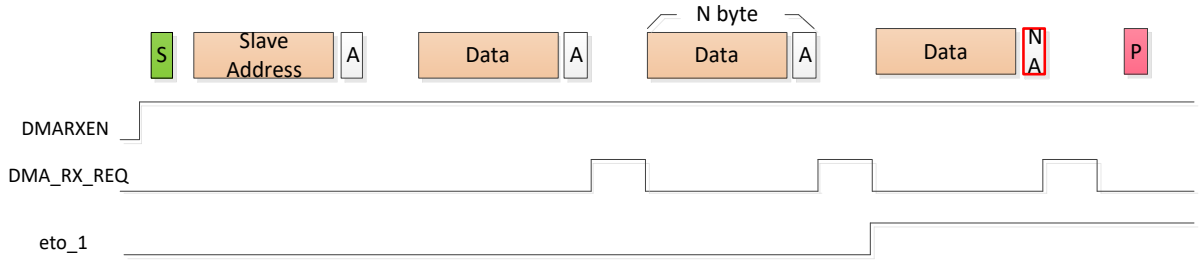

图 17-12 从机接收器

17.4.5 从机低功耗唤醒

MCU 在低功耗模式下，若使能唤醒功能，则地址匹配时，将在第 9 个 SCL 处产生 ACK 低电平信号，并产生唤醒信号唤醒 MCU，紧跟在地址后的数据 I2C 都不回 ACK，I2C 重新初始化或重新收到起始信号后可以正常接收数据。

17.5 应用说明

17.5.1 数据传输

写 STARTSTOP [START]为 '1' 发送 START 信号到 I2C 总线。传输结束写 STARTSTOP [STOP]为 '1' 发送 STOP 信号到 I2C 总线。

主机发送时，TX 控制置位 1，对数据寄存器写入后，波特率控制器自动开始向 I2C 总线传输数据，一字节传输完成后，BND 标志置位，表明可以再次写入数据，写入数据或向 BND 位写 1 可清除 BND 标志。

主机接收时，TX 控制位置 0，对数据寄存器的读操作，会触发波特率控制器自动向 I2C 总线发出时钟信号，并根据 RACK 控制应答，收到的数据载入数据寄存器。BND 标志位置位，对数据寄存器进行读取或向 BND 位写 1 可清除 BND 标志。

从机发送时，先将待发送数据写入数据寄存器，当收到主机读取的完整时钟信号后，数据被传出，TXEF 标志位置位，BND 标志位置位，对数据寄存器的写入操作可清除 TXEF 和 BND 标志位，或向 BND 位写 1 也可清除 BND 标志。

从机接收时，主机发送一字节完整数据后，数据被载入从机数据寄存器，RXFF 标志位置位，BND 标志位置位，读取数据寄存器可清除 RXFF 和 BND 标志位，或向 BND 位写 1 也可清除 BND 标志。

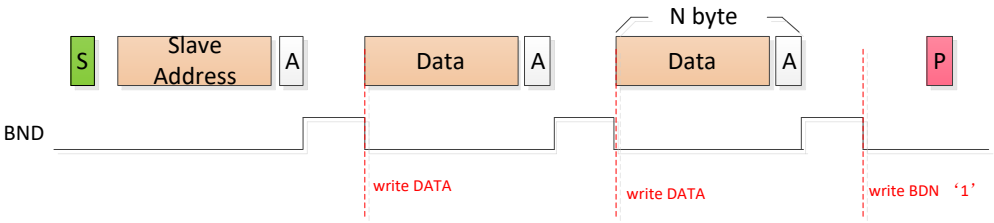

图 17-13 主机写从机模式的 BND 序列

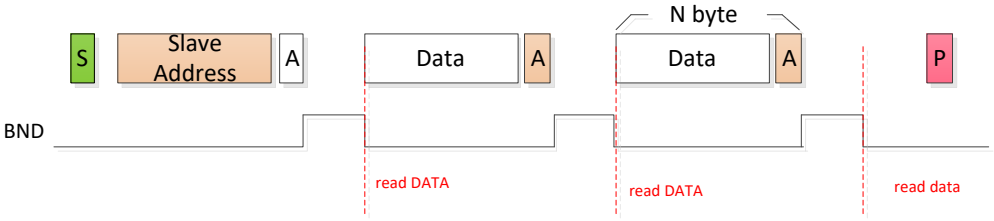

图 17-14 主机读从机模式的 BND 序列

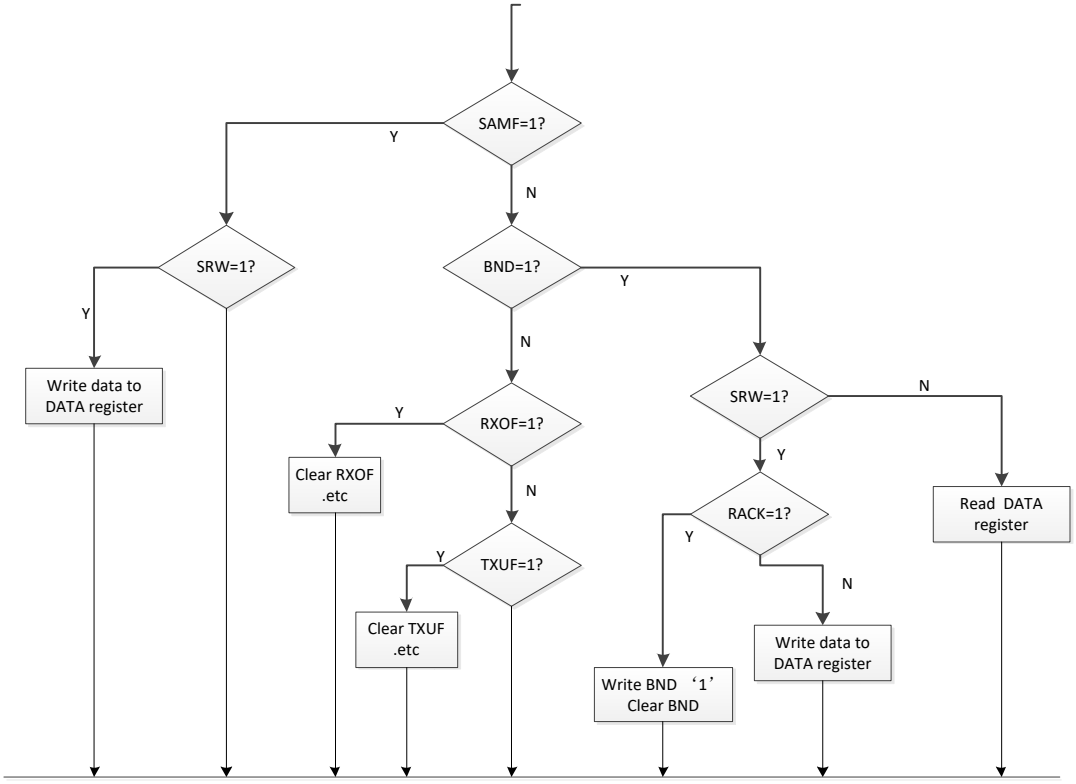

图 17-15 典型 I2C 从机中断程序

17.5.2 应答控制

I2C 模块通过软件方式控制应答信号。

在主机接收数据时，TACK 位的值决定下一字节读取完后，在第 9 个时钟信号上的应答值。

在从机接收数据时，TACK 位的值决定下一字节读取完后，在第 9 个时钟信号上的应答值。

17.6 寄存器定义

表 17-2 I2C 寄存器映射

I2C1 基地址：0x4000e000

I2C2 基地址：0x4000f000

地址	名称	宽度	描述
I2Cx 基地址+0x00	I2C_ADDR0	32	地址寄存器 0
I2Cx 基地址+0x04	I2C_ADDR1	32	地址寄存器 1
I2Cx 基地址+0x08	I2C_SAMPLE_CNT	32	波特率配置寄存器 0
I2Cx 基地址+0x0C	I2C_STEP_CNT	32	波特率配置寄存器 1
I2Cx 基地址+0x10	I2C_CTRL0	32	控制寄存器 0
I2Cx 基地址+0x14	I2C_CTRL1	32	控制寄存器 1
I2Cx 基地址+0x18	I2C_CTRL2	32	控制寄存器 2
I2Cx 基地址+0x1C	I2C_CTRL3	32	控制寄存器 3
I2Cx 基地址+0x20	I2C_STATUS0	32	状态寄存器 0
I2Cx 基地址+0x24	I2C_STATUS1	32	状态寄存器 1
I2Cx 基地址+0x28	I2C_DGLCFG	32	毛刺滤波配置寄存器
I2Cx 基地址+0x2C	I2C_DATA	32	数据寄存器
I2Cx 基地址+0x30	I2C_STARTSTOP	32	起始与停止信号控制寄存器

【说明】上表中，x=1~2。

17.6.1 地址寄存器 0(I2C_ADDR0)

表 17-3 I2C_ADDR0 寄存器

I2C_ADDR0 **地址寄存器 0** **Reset: 0x000000FE**

位	31：8	7	6	5	4	3	2	1	0
名称		AD [6: 0]							
访问		RW							
Reset		1							

字段	说明
7: 1 AD[6: 0]	**I2C 从机 7Bit 地址** 在从机模式下设置 7 位地址或 10 位地址中的低 7 位。

17.6.2 地址寄存器 1(I2C_ADDR1)

表 17-4 I2C_ADDR1 寄存器

I2C_ADDR1 地址寄存器 1 **Reset: 0x000007F7**

位	31：13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称		RMEN		RAD								AD[9：7]		
访问		RW		RW								RW		
Reset		0		1								1		

字段	说明
12 RMEN	**从机模式时，范围地址使能位** 1： 使能 0： 禁用 范围地址使能控制位。
10: 4 RAD	**从机模式时，范围地址值** 从机 7bit 范围地址值，范围地址使能时，从机接收到的地址大于 AD[6: 0]且小于等于 RAD[6:0]时，将得到匹配。
2: 0 AD [9: 7]	**I2C 从机 10Bit 地址** 在从机模式下设置 10 位地址的高 3 位。

17.6.3 波特率配置寄存器 0(I2C_SAMPLE_CNT)

表 17-5 I2C_SAMPLE_CNT 寄存器

I2C_SAMPLE_CNT 波特率配置寄存器 0 **Reset: 0x00000004**

位	31：8	7	6	5	4	3	2	1	0
名称		SAMPLE_CNT							
访问		RW							
Reset		0							

字段	说明
7: 0 SAMPLE_CNT	**调整每个采样点的宽度** 采样周期 = (SAMPLE_CNT +1) *总线时钟周期

17.6.4 波特率寄存器 1(I2C_STEP_CNT)

表 17-6 I2C_STEP_CNT 寄存器

I2C_STEP_CNT 波特率配置寄存器 1 **Reset: 0x00000004**

位	31：8	7	6	5	4	3	2	1	0
名称		STEP_CNT							
访问		RW							
Reset		0	0	0	0	0	1	0	0

字段	说明
7: 0 STEP_CNT	**调整半脉冲宽度** 半波特率周期= (STEP_CNT +1)*采样宽度 **注意：STEP_CNT 最小为 3。**

17.6.5 控制寄存器 0(I2C_CTRL0)

表 17-7 I2C_CTRL0 寄存器

I2C_CTRL0 CTRL0 **Reset: 0x00000000**

位	31：8	7	6	5	4	3	2	1	0
名称		IICEN	IICIE	MSTR	TX	TACK	WUEN		
访问		RW	RW	RW	RW	RW	RW		
Reset		0	0	0	0	0	0		

字段	说明
7 IICEN	**I2C 模块使能** 1： 使能 0： 禁用
6 IICIE	**I2C 全局中断使能** 1： 使能 I2C 模块全局中断 0： 禁用
5 MSTR	**I2C 模式选择** 1： 主机模式 0： 从机模式 **注意： 如果 ARB_LOST 置位时，该位会自动清零。**
4 TX	**传输方向选择** 1： 发送(TX) 0： 接收(RX)

字段	说明
	主要作用在主机模式下的读取：TX=0，读取 DATA 寄存器会触发读时钟； TX=1，读取 DATA 寄存器不会触发读时钟。
3 TACK	**应答控制** 1： 向接下来的接收字节上的总线发送 NACK 信号 0： 向接下来的接收字节上的总线发送 ACK 信号 指定在主机或从机的数据应答周期过程中驱动到 SDA 的值。
2 WUEN	**唤醒功能使能** 1： 在低功耗模式下使能唤醒功能 0： 禁用唤醒功能 当出现从机地址匹配时，I2C 模块可将 MCU 从低功耗模式唤醒。 **注意：当 I2C 模块处于从机模式并进入低功耗模式时，发生 7bit、10bit（ADEXT=1），或通用调用广播地址（GCAEN=1）匹配，或范围地址匹配（RMEN=1）时， I2C 模块将唤醒 MCU。**

17.6.6 控制寄存器 1(I2C_CTRL1)

表 17-8 I2C_CTRL1 寄存器

I2C_CTRL1 **CTRL1** **Reset: 0x00000000**

位	31：8	7	6	5	4	3	2	1	0
名称		GCAEN	ADEXT		SYNCEN	ARBEN			STREN
访问		RW	RW		RW	RW			RW
Reset		0	0		0	0			0

字段	说明
7 GCAEN	**从机通用广播地址使能** 1： 使能 0： 禁用
6 ADEXT	**从机地址扩展使能** 1： 10 位地址模式 0： 7 位地址模式
4 SYNCEN	**主机 SCL 同步使能** 1： 使能 0： 禁用 使能主机的 SCL 同步功能。

字段	说明
3 ARBEN	**主机仲裁使能** 1： 使能 0： 禁用 使能主机仲裁功能。 **注意：如果在多主机系统中使用 I2C，则多主机功能必须同时设置 SYNC_EN 和 ARB_EN。** **如果 I2C 在单主机系统中使用，且从机具有 SCL 拉伸功能，则可以只设置 SYNC_EN。**
0 STREN	**从机 SCL 拉伸使能** 1： 使能 0： 禁用 使能此位，从机将在每个字节第 9 个 SCL 下降沿后将 SCL 拉低。 **注意：** **在 SAMF=1 后，如果 SRW=1， RXFF=1 将产生 scl stretch，如果否则 SRW=0, TXEF=1 将产生 scl stretch。因此当使能 STR_EN 且从机为 发送器时(TX)，在每个字节（包括地址字节）的第 9 个 SCL 下降沿后，从机会拉低 SCL，直到写 DATA 寄存器。类似地，当从机为发送器时(RX),在每个字节的第 9 个 SCL 下降沿后，从机会拉低 SCL，直到读取数据寄存器。** **在从机模式下，当使能 GCA_EN 或 MNTEN 时，从机不会拉低 SCL。**

17.6.7 控制寄存器 2(I2C_CTRL2)

表 17-9 I2C_CTRL2 寄存器

I2C_CTRL2 **CTRL2** **Reset: 0x00000000**

位	31：8	7	6	5	4	3	2	1	0
名称		RXOFIE	TXUFIE	RXFIE	TXEMIE			NACKIE	MNTEN
访问		RW	RW	RW	RW			RW	RW
Reset		0	0	0	0			0	0

字段	说明
7 RXOFIE	**从机接收缓存区溢出中断使能** 1： 使能 0： 禁用
6 TXUFIE	**从机发送缓存溢出中断使能** 1： 使能 0： 禁用
5	**从机接收缓存满中断使能**

字段	说明
RXFIE	1： 使能 0： 禁用
4 TXEMIE	**主机发送缓存空中断使能** 1： 使能 0： 禁用
1 NACKIE	**NACK 中断使能** 1： 使能 0： 禁用
0 MNTEN	**从机监测功能使能** 1： 使能 0： 禁用

17.6.8 控制寄存器 3(I2C_CTRL3)

表 17-10 I2C_CTRL3 寄存器

I2C_CTRL3 **CTRL3** **Reset: 0x00000000**

位	31：2	1	0
名称		DMA RXEN	DMA TXEN
访问		RW	RW
Reset		0	0

字段	说明
1 DMARXEN	**DMA 接收使能** 1： 使能 0： 禁用
0 DMATXEN	**DMA 发送使能** 1： 使能 0： 禁用

17.6.9 状态寄存器 0(I2C_STATUS0)

表 17-11 I2C_STATUS0 寄存器

I2C_STATUS0 STATUS0 Reset: 0x00000008

位	31：8	7	6	5	4	3	2	1	0
名称		BND	SAMF	BUSY	ARBLOST	READY	SRW		RACK
访问		W1C	W1C	R	W1C	R	R		W1C
Reset		0	0	0	0	1	0		0

字段	说明
7 BND	**字节结束标志** 1： 一个字节传输完成(包括 ACK 位，共 9 个 SCL) 0： 传输进行中，一个字节传输未结束 复位后，*BND* 为 '0'。*BND* 仅在总线上 START 和 STOP 信号之间的数据传输期间设置。BND 会在每 9 个 SCL 下降沿之后设置。 在 Master 模式下，发送数据时，软件写 DATA 寄存器会清零此位，并发出 DATA；读取数据时，一字节传输完成，软件读取 DATA 寄存器会清零此位，并发出下一字节数据时钟。 在 Slave 模式下，地址匹配后置位 SAMF，此时 BND 标志不会置位，BND 在地址匹配后的每次数据传输置位。具体为：地址匹配后，主机写数据，从机置位 BND，当从机主动 NACK 后，主机若继续写数据，从机不再置位 BND；主机读取数据，从机置位 BND，若收到主机的 NACK，主机继续读数据，从机不再置位 BND。 在 Master/Slave DMA 发送或接收模式下，会屏蔽 BND 中断，当发送完最后一字节时，DMA 发送完成，但 I2C 还会继续传输完最后这一字节，可以通过轮询 BND 来判断是否发送完最后一字节，或者 Disable DMA 后，等待 BND 中断产生。 **注意：** **当 I2C 为 从机模式且 MNTEN=1，当 BND 为 '1'时，读取 DATA 寄存器将清零此位。写 '1' 也可以清零此位。**
6 SAMF	**从机地址匹配标志** 1： 地址匹配 0： 地址不匹配 **注意：** **该位在满足如下条件之一时置位 ：：** **1. 7 位地址匹配** **2. 10 位地址，第 1st 字节和第 2nd 字节匹配。并且在第 2nd 字节匹配后置位。** **3. 通用广播地址匹配。** **4. 范围地址匹配** **写 '1' 清零此位。**
5	**总线忙**

字段	说明
BUSY	
	1： 总线忙 0： 总线空闲 指示总线的状态，对从机模式和主机模式都有效。当硬件在总线上检测到 START 信号时，该位置位。检测到 STOP 信号时，该位清零。
4 ARBLOST	**仲裁丢失标志** 1： 仲裁丢失 0： 仲裁未丢失 **注意：** **写 '1'清零此位。** **当仲裁丢失时， I2C 主机将切换至从机模式，MSTR 位由硬件清零。**
3 READY	**准备好接收新的命令** 1： 内部硬件准备好接收软件的新命令 0： 内部硬件未准备好 该位指示内部硬件状态，仅对主机模式有效。 **注意：此位对主机模式有效，在主机产生 START/STOP 信号时使用。当 I2C 模块处于主机模式时，写 STARTSTOP 寄存器产生 START/STOP 信号后必须等待 READY 位为 1。**
2 SRW	**从机读取/写入方向** 1： 从机发送(TX)，主机从从机读取 0： 从机接收(RX)，主机向从机写入
0 RACK	**接收应答** 该字段只在发送方向有效，主机发送或从机发送。 1： 未检测到应答信号 0： 在总线上完成一个字节的数据传送后，接收到应答信号 **注意：若 NACKIE=1, RACK=1 将会产生中断请求，写 1 清零此位 。**

17.6.10 状态寄存器 1(I2C_STATUS1)

表 17-12 I2C_STATUS1 寄存器

I2C_STATUS1 STATUS1 Reset: 0x00000081

位	31：4	3	2	1	0
名称		RXOF	TXUF	RXFF	TXEF
访问		W1C	W1C	W1C	W1C
Reset		0	0	0	1

字段	说明
3 RXOF	**从机 RX 缓冲区溢出标志** 1： RX 缓冲区溢出 0： 未溢出 **注意：** **当 RX 缓冲区溢出时，新接收的数据不会存储在 RX 缓冲区中。** **写 '1' 清零此位。**
2 TXUF	**从机 TX 缓冲区溢出标志** 1： TX 缓冲区溢出 0： 未溢出 **注意：** **当 TX 缓冲区溢出时，再次发送时发送缓冲区的最后一个 DATA。** **写'1' 清零此位。**
1 RXFF	**从机 RX 缓冲区满标志** 1： RX 缓冲区满 0： 未满 **注意：读取 *DATA* 寄存器将清零此位**
0 TXEF	**从机 TX 缓冲区空标志** 1： TX 缓冲区为空 0： 非空 **注意：写 DATA 寄存器会清零此位**

17.6.11 毛刺滤波配置寄存器(I2C_DGLCFG)

表 17-13 I2C_DGLCFG 寄存器

I2C_DGLCFG DGLCFG Reset: 0x00000000

位	31：8	7	6	5	4	3	2	1	0
名称		DGLEN	STOPF	SSIE	STARTF	DGL_CNT			
访问		RW	W1C	RW	W1C	RW			
Reset		0	0	0	0	0			

字段	说明
7 DGLEN	**毛刺滤波器使能** 1： 使能 0： 禁用
6 STOPF	**总线 STOP 标志** 1： 在 I2C 总线上检测到 STOP 信号 0： 在 I2C 总线上未检测到 STOP 信号 当 I2C 总线上检测到 STOP 信号时，由硬件置位此位。 **注意： 写 “1” 清零此位。**
5 SSIE	**总线 STOP 或 START 中断使能** 1： 使能 STRAT 或 STOP 检测中断 0： 禁用 **注意：写 ‘1 ’ 清除 STARTF 或 STOPF 标志。**
4 STARTF	**总线 START 标志** 1： 在 I2C 总线上检测到 START 0： 未检测到 START 当 I2C 总线上检测到 START 信号时，由硬件置位此位。 **注意：写 ‘1’ 清零此位**
3: 0 DGL_CNT	**毛刺滤波器计数器。** 0h： 无滤波器 / 旁路 1-Fh：对宽度最多为 1-15 个总线时钟周期的毛刺进行过滤 选择毛刺滤波宽度。对于毛刺宽度大于所设置的滤波宽度，滤波器不会允许该毛刺通过。

17.6.12 数据寄存器(I2C_DATA)

表 17-14 I2C_DATA 寄存器

I2C_DATA DATA Reset: 0x000001FF

位	31：9	8	7	6	5	4	3	2	1	0
名称		MAK	DATA							
访问		R	RW							
Reset		1	1							

字段	说明
8 MAK	**从机监控功能 ACK 位** 对于从机监测器，该字段为 I2C 总线的 ACK 位。 **注意：MAK='1', NACK** **MAK='0', ACK** **对于监测器，DATA 是 I2C 总线上传送的数据， 第 8 位 为 ACK 位。**
7: 0 DATA	**数据** 在主机传送(TX)模式下，当向该寄存器中写入数据时，会启动数据传输。首先发送最高有效位。在主机接收(RX) 模式下，读该寄存器会触发主机发出读时钟。 **注意： 当从主机接收模式转换时，在读取 DATA 寄存器之前需要先切换 I2C 模式，以防意外启动主机数据接收传输流程。**

17.6.13 起始与停止信号控制寄存器(I2C_STARTSTOP)

表 17-15 I2C_STARTSTOP 寄存器

I2C_STARTSTOP 起始停止信号控制寄存器 Reset: 0x00000000

位	31：2	1	0
名称		STOP	START
访问		RW	RW
Reset		0	0

字段	说明
1 STOP	**主机发送停止信号** 写"1"，主机会发送 STOP 信号 读取此位会始终返回 "0"
0 START	**主机发送起始信号** 写"1"， 主机将会发送 START 或 RESTART 信号 读取此位会始终返回 "0"

18 串行外设接口（SPI）

18.1 简介

SPI(Serial Peripheral Interface--串行外设接口)总线系统是一种同步串行外设接口，支持串行、同步、全双工协议。SPI 模块包含主机和从机并以 4 线方式进行通信。

图 18-1 给出了 SPI 主机和 SPI 从机之间的连接示例，如下图所示。

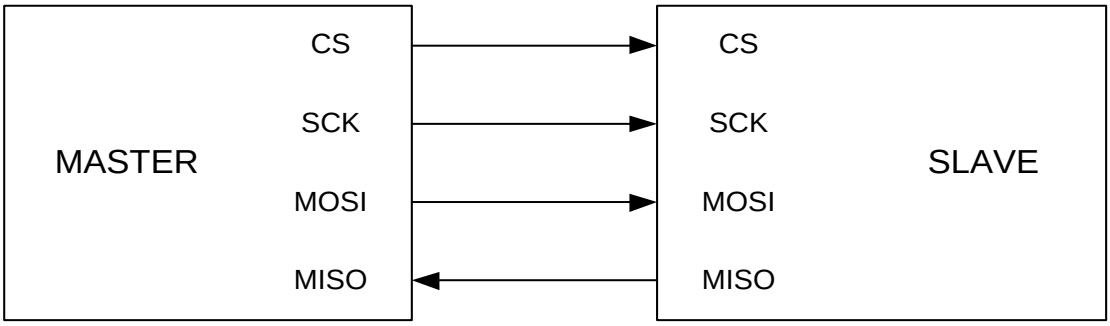

图 18-1 SPI 系统连接

18.2 特性

- 主机模式或从机模式操作
  - 作为主机，波特率最高支持 $f_{bus}/2$ Hz (f_{bus} 是 APB 总线时钟)
  - 作为从机，波特率最高支持 12 MHz
- 全双工模式
- 主机可编程波特率
- 串行时钟相位和极性选择
- 可配置连续或不连续 CS (从机选择) 输出
- 带 CPU 中断功能的模式错误标志位
- 可供选择的最高有效位（MSB）优先或 最低有效位（LSB）优先移位
- 可配置的 CS 建立时间，保持时间和空闲时间
- 可配置的 SCK 高和低周期
- 4-16 位传输帧格式选择
- DMA 模式
- 带中断功能的 TX 缓冲区下溢及 RX 缓冲区溢出标志位

- 从机支持停止（Stop）模式唤醒功能

18.3 结构框图

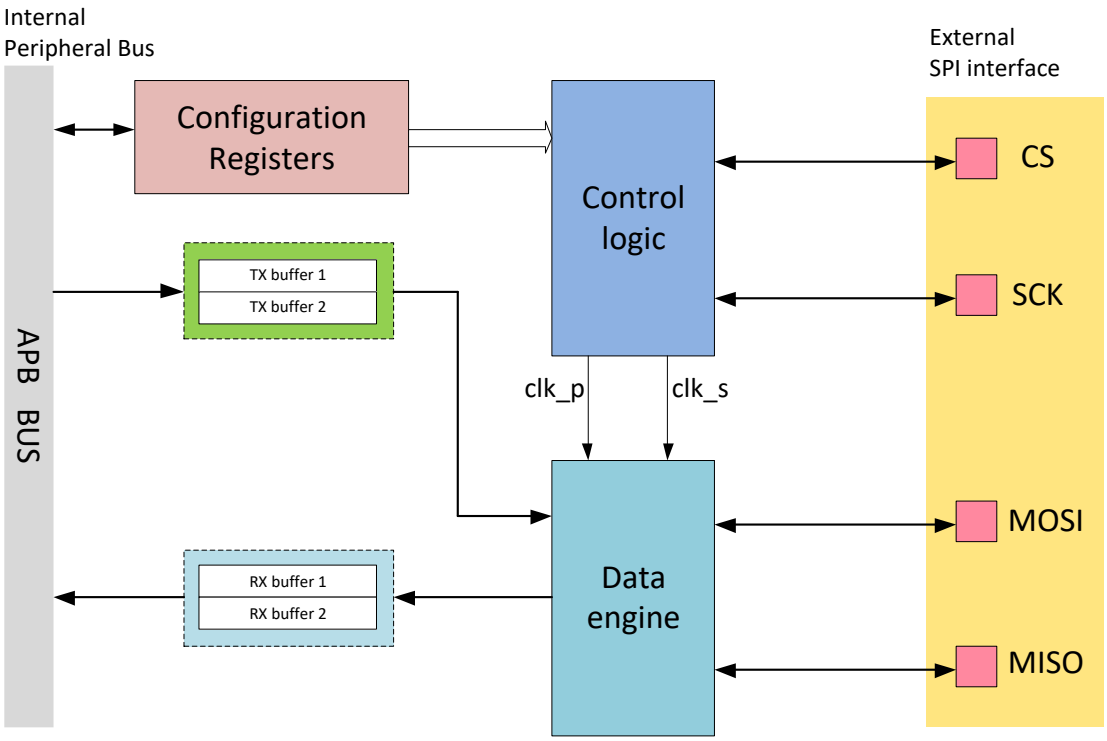

图 18-2 SPI 结构框图

18.4 功能描述

18.4.1 数据流 & 算法

对于主机模式，数据被写入一个 16 位 TX 缓冲区，然后参考波特率控制逻辑，加载到移位寄存器。由 TMSBF 控制输出数据是高位先发还是低位先发。在 FRMSIZE 指定的 SCK 周期数之后，移位寄存器从 MISO 引脚移入一个字节的数据。接收的数据存储在 16 位 RX 缓冲区中。

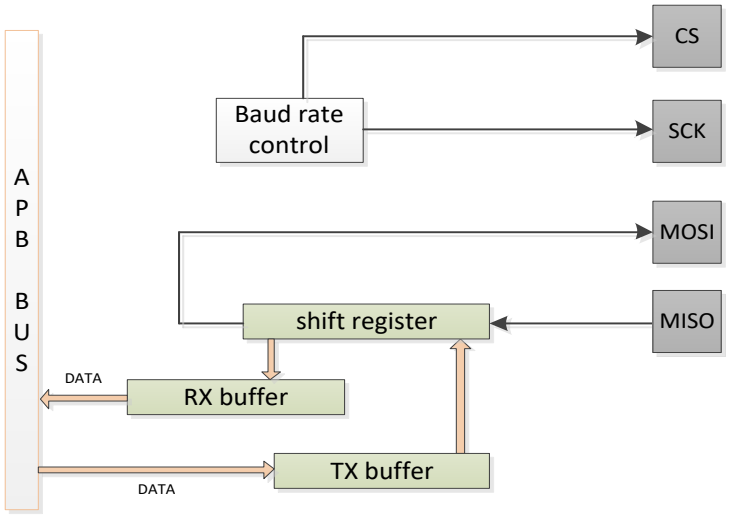

图 18-3 主机数据流

对于从机模式，数据流类似于主机模式。但 CS 引脚是从机选择输入，SCK 是来自主机的 SPI 时钟输入。在发生数据传输之前，从机 SPI 的 CS 引脚必须为低电平。MOSI 是从机数据输入引脚，MISO 是数据输出引脚。

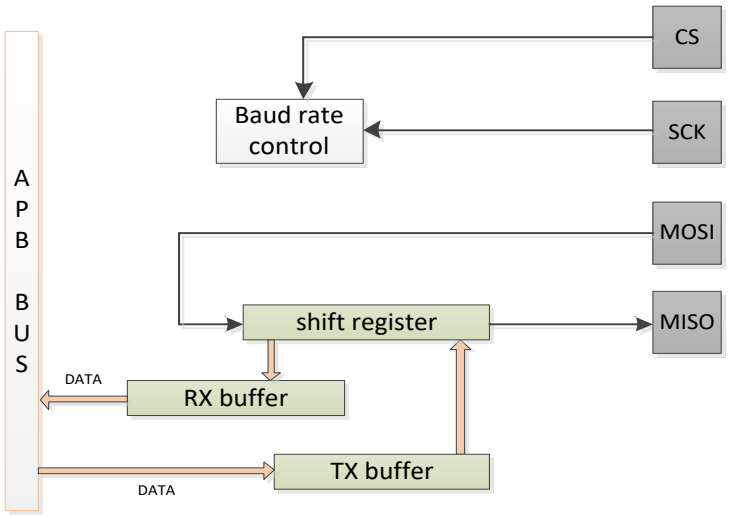

图 18-4 从机数据流

18.4.2 输入输出时序

18.4.3 CPHA = 0 传输格式

SCK 线上的第一个边沿用于将从机的第一个数据位计时到主机，将主机的第一个数据位计时到从机。在某些外设中，只要选择了从机，从机的数据输出引脚就会提供从机数据的第一位。在这种格式中，在 CS 变低之后，第一个 SCK 边沿发出半个周期。

半个 SCK 周期后，第二个边沿出现在 SCK 线上。当第二个边沿出现时，先前从串行数据输入引脚锁存的值被移入移位寄存器。

在第二个边沿之后，SPI 主机的下一位从主机的串行数据输出引脚传输到从机的串行输入引脚。该过程在 SCK 线上总共持续 16 个边沿，数据被锁存在奇数边沿上并在偶数边缘上移位。

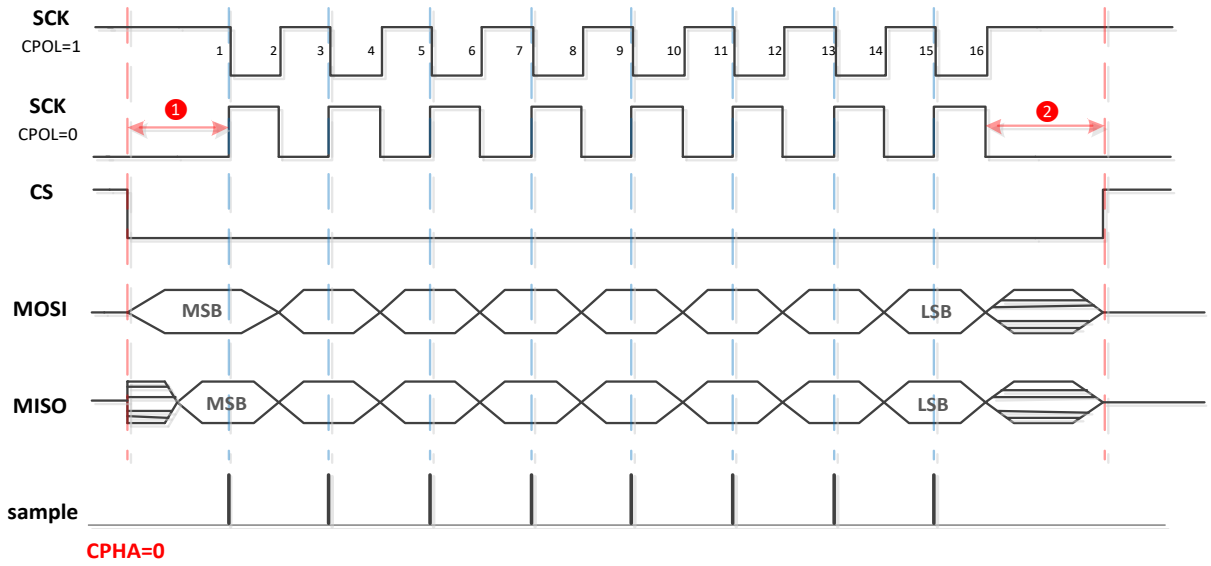

图 18-5 CPHA=0 传输格式

18.4.4 CPHA = 1 传输格式

某些外设在第一个数据位在数据输出引脚变得可用之前需要第一个 SCK 边沿，第二个边沿将数据计入系统。

SCK 的第一个边沿在半个 SCK 时钟周期同步延迟之后立即发生。第一个边沿命令从机将其第一个数据位传输到主机的串行数据输入引脚。半个 SCK 周期后，第二个边沿出现在 SCK 引脚上。这是主机和从机的锁存边沿。

当第三个边沿出现时，先前从串行数据输入引脚锁存的值被移入移位寄存器。在此边沿之后，主机数据的下一位从主机的串行数据输出引脚耦合到从机上的串行输入引脚。

该过程在 SCK 线上总共持续 16 个边沿，数据被锁存在偶数边沿上并且移位发生在奇数边沿上。

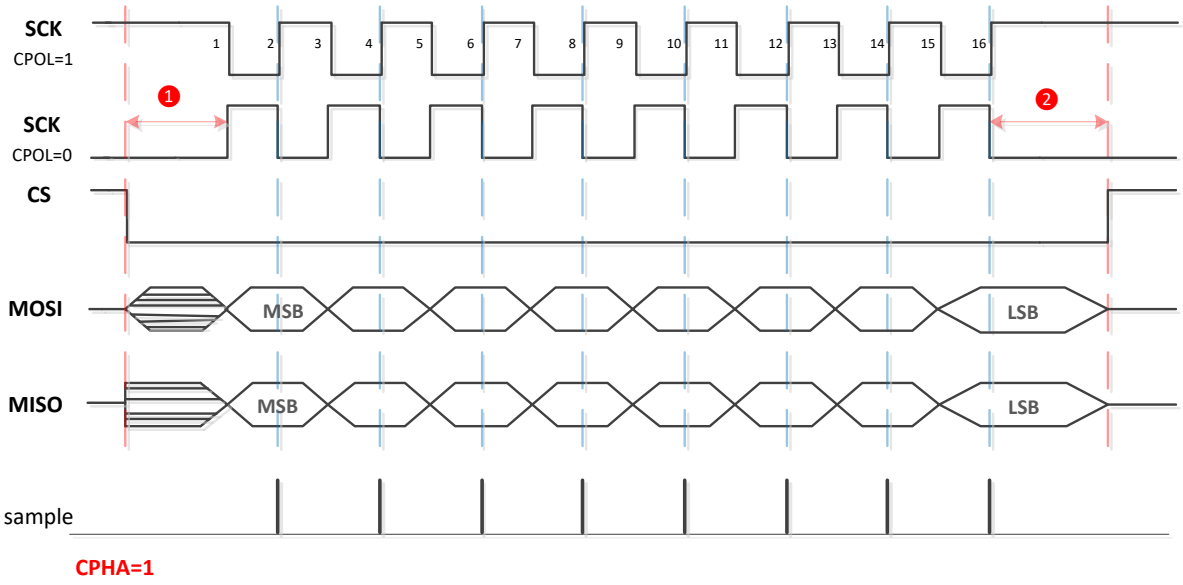

图 18-6 CPHA=1 传输格式

18.4.5 主机 SCK 输出时序设置

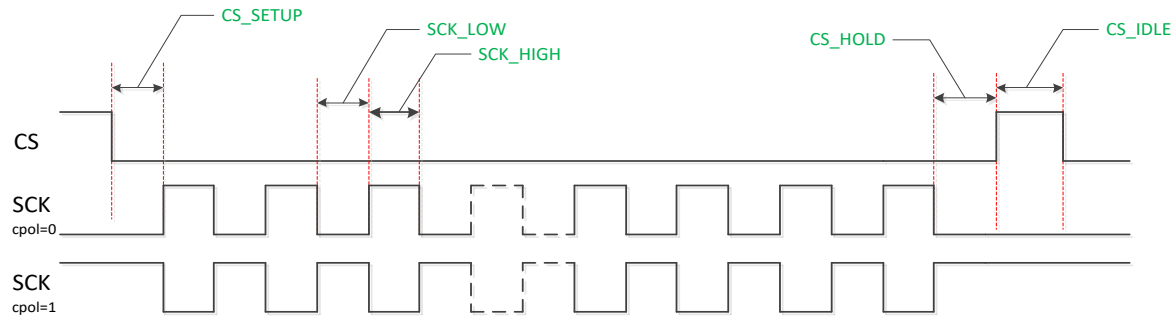

图 18-7 波特率生成

波特率： $f_{SCL}=f_{bclk}/(SCK\_LOW+1+SCK\_HIGH+1)$, f_{bclk} 是 APB 总线时钟频率。

18.4.6 主机模式故障检测

如果在 SPI 主机初始化传输之前 CS 引脚已被驱动为低电平，则置位 MODEF。主模式故障检测功能仅在 MSTR=1, MODFEN=1, CSOE=1 时有效。

如果在 CPOL = 0 时,使能主机模式故障检测功能，则 SCK 与正常情况不同。当处于空闲状态时，SCK 为高电平，检查主机模式故障错误后，且没有发生故障，才变为低电平，图 18-8 的红色部分为主机检测故障的时间段。由于 CS 拉低前，SCK 也变为低了，即依旧是 CPOL = 0 的通讯模式。传输完成后 SCK 保持高电平。

如果 CPOL = 1 时，使能主机模式故障检测功能，SCK 的波形与正常情况相同。

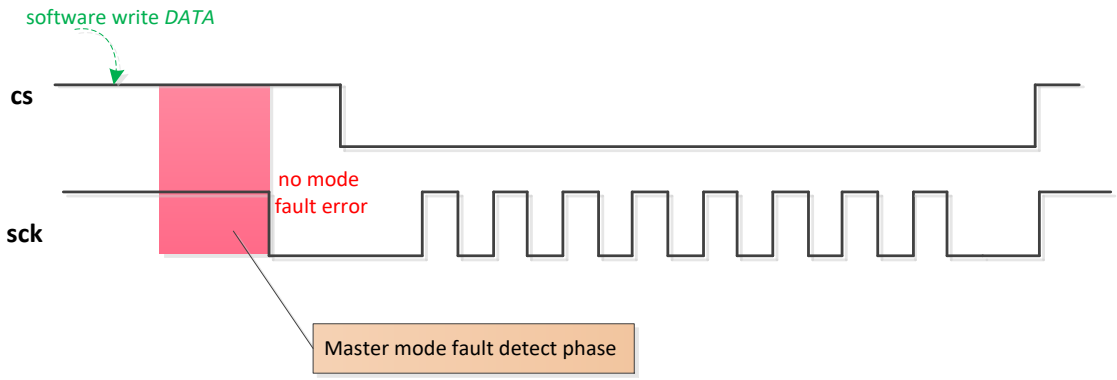

图 18-8 在模式故障检测使能时的 SCK 输出时序

在某些特殊情况下，主机 1 的主机模式故障检测功能可能无法检测到模式故障。如果在图 18-9 中的（1）周期内，主机 2 驱动 CS 为低电平，则主机 1 的控制器不会设置 MODEF。只有主机 2 在（1）周期之前驱动 CS 为低电平，主机 1 才可以正常设置 MODEF。

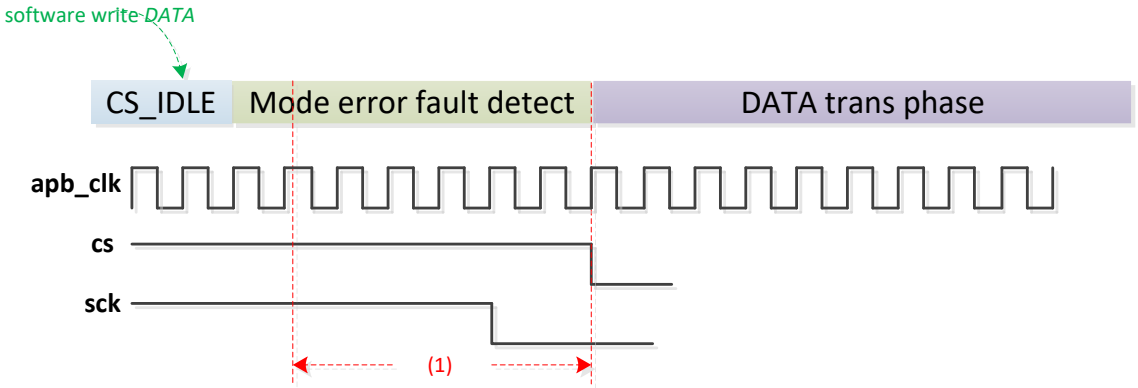

图 18-9 模式故障检测限制

18.4.7 从机低功耗唤醒

处于停止模式的 SPI 从机可以产生异步中断，以在接收到数据时将 CPU 从低功耗模式唤醒。为确保 SPI 模块低功耗唤醒功能正确，系统必须遵循一些规定。在 CPU 进入低功耗模式之前，系统必须确认 SPI 模块处于空闲状态。软件可以检查状态寄存器 SPI_STATUS[8] IDLEF 位的状态。对于主模式，tx 缓冲区为空，rx 缓冲区为空，内部硬件空闲，IDLEF 为“1”。对于从机模式，tx 缓冲区为空，rx 缓冲区为空，CS 为无效（CS 为高电平），IDLEF 为“1”。如果当 SPI 模块忙时，CPU 进入低功耗模式，则 SPI 模块无法确保数据有效或唤醒功能正确。

从机仅在如下条件全部得到满足时生成异步唤醒中断：

a. SPI 模块处于从机模式；

b. SPI 从机处于空闲状态；

c. WUEN 位 为‘1’；

d. 单字节唤醒序列传输结束。

CS 从高到低会初始化唤醒阶段，从机在 FRMSIZE 指定的 SCK 周期后生成异步唤醒请求。SPI 从机可以接收唤醒相位字节的数据。芯片唤醒结束（时钟恢复）后，RXFF 标志位会置位，读取数据寄存器将返回主机发送的唤醒相位字节的数据。唤醒阶段只能传输一个字节。在唤醒阶段，来自主机的连续传输会破坏接收到的数据，并可能导致无法正确设置 RXFF 标志。

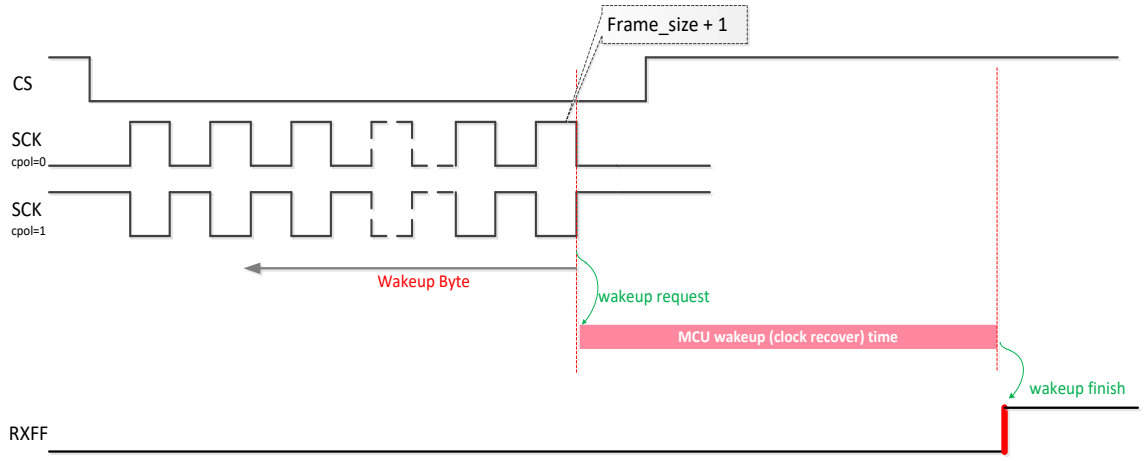

图 18-10 唤醒序列

18.4.8 中断

SPI 共有 5 个中断。

表 18-1 中断汇总

状态标志位	中断使能位
发送缓冲区空标志（TXEF）	TXEIE
接收缓冲区非空标志（RXFF）	RXFIE
发送缓冲区下溢（TXUF）	TXUIE
接收缓冲区溢出（RXOF）	RXOIE
主模式失效事件（MODEF）	MODFIE

18.5 应用说明

18.5.1 主机 CS 连续模式

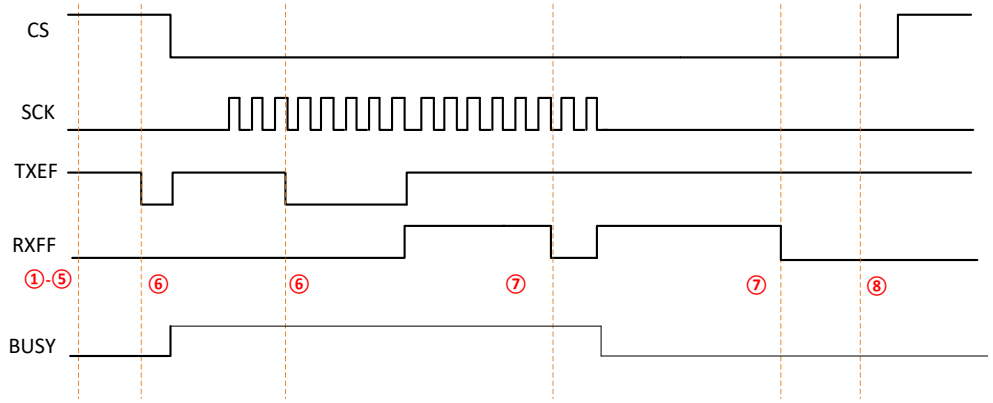

图 18-11 CS 连续模式

CS 连续输出

1. 配置寄存器 SPI_CFG0：CS_SETUP, CS_HOLD, SCK_LOW, SCK_HIGH；
2. 配置寄存器 SPI_CFG1：CS_IDLE；
3. 配置 FRMSIZE, CPHA, CPOLRMSBF, TMSBF 等；
4. 配置 CSOE, CONT_CS,MSTR；
5. SPIEN=1；
6. TXEF=1, 写数据至 DATA；
7. RXFF=1, 从 DATA 中读数据；
8. 写 CSRLS'1'，释放 CS，然后硬件进入空闲状态。即 CSOE = 1, CONT_CS = 1 时， CS 由硬件自动拉低，但是，数据发送完成后，需要用户软件写 CSRLS'1'，拉高 CS。

18.5.2 主机 CS 非连续输出

1. 配置寄存器 SPI_CFG0：CS_SETUP,CS_HOLD,SCK_LOW,SCK_HIGH；
2. 配置寄存器 SPI_CFG1：CS_IDLE；
3. 配置 FRMSIZE,CPHA,CPOL,RMSBF,TMSBF 等；
4. 配置 CSOE, CONT_CS, MSTR；
5. SPIEN=1；
6. TXEF=1，写数据至 DATA；

7. RXFF=1，从 DATA 读取数据。

当 CS 处于不连续模式时，CS 通过硬件变为低或高电平，所以软件不需要配置 CSRLS。

18.5.3 从机模式

1. 配置 FRMSIZE,CPHA,CPOL.RMSBF,TMSBF 等；
2. 配置 MSTR；
3. SPIEN=1；
4. TXEF=1, 写数据至 DATA；
5. RXFF=1，从 DATA 读取数据。

18.5.4 DMA 模式

当 TXEF=1 时，会产生 DMA TX 请求。

当 RXFF=1 时，会产生 DMA RX 请求。

1. 初始化 DMA；
2. 初始化 SPI 模块，并使能 *DMATXEN*, *DMARXEN*；
3. 等待 DMA 完成；
4. 其他选项类似于 CS 连续或不连续模式。

18.6 寄存器定义

表 18-2 SPI 寄存器映射

SPI0 基地址： 0x4000c000
SPI1 基地址： 0x4000d000

地址	名称	宽度	描述
SPIx 基地址+0x00	SPI_CFG0	32	SPI 配置寄存器 0
SPIx 基地址+0x04	SPI_CFG1	32	SPI 配置寄存器 1
SPIx 基地址+0x08	SPI_CMD	32	SPI 命令寄存器
SPIx 基地址+0x0c	SPI_STATUS	32	SPI 状态寄存器
SPIx 基地址+0x10	SPI_DATA	32	SPI 数据寄存器
SPIx 基地址+0x14	SPI_CFG2	32	SPI 配置寄存器 2

【说明】上表中，x=0~1。

18.6.1 配置寄存器 0(SPI_CFG0)

表 18-3 SPI_CFG0 寄存器

SPI_CFG0 **配置寄存器 0** **Reset：0x05050505**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	CS_SETUP								CS_HOLD							
访问	RW								RW							
Reset	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	SCK_LOW								SCK_HIGH							
访问	RW								RW							
Reset	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1

字段	说明
31: 24 CS_SETUP	**CS 建立时间配置** 片选建立时间 = (CS_SETUP +1)*CLK_PERIOD，其中 CLK_PERIOD 是 SPI 引擎采用的时钟周期时间
23: 16 CS_HOLD	**CS 保持时间配置** 片选保持时间 = (CS_HOLD +1)*CLK_PERIOD.
15:8 SCK_LOW	**SCK 低电平时间配置** **注意：CPOL 为 0 时，该位配置的是 SCK_LOW 的时间。CPOL 为 1 时，配置的是 SCK_HIGH 的时间。** SCK 时钟低电平时间 = (SCK_LOW +1)*CLK_PERIOD.
7：0 SCK_HIGH	**SCK 高电平时间配置** **注意：CPOL 为 0 时，该位配置的是 SCK_HIGH 的时间。CPOL 为 1 时，配置的是 SCK_LOW 的时间。** SCK 时钟高电平时间 = (SCK_HIGH +1)*CLK_PERIOD.

18.6.2 配置寄存器 1(SPI_CFG1)

表 18-4 SPI_CFG1 寄存器

SPI_CFG1 **配置寄存器 1** **Reset：0x027C0005**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称		WKUEN		CONT_CS		MODFEN	CSOE		FRMSIZE				RMSBF	MSBF	CPHA	CPOL
访问		RW		RW		RW	RW		RW				RW	RW	RW	RW
Reset		0		0		0	1		0	1	1	1	1	1	0	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	DMARXEN	DMATXEN	MODFIE	MSTR	RXOIE	TXUIE	RXFIE	TXEIE	CS_IDLE							
访问	RW	RW	RW	RW	RW	RW	RW	RW	RW							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

字段	说明
30 WKUEN	**从机唤醒功能使能** 0： 从机唤醒功能禁用 1： 从机唤醒功能使能
28 CONT_CS	**CS 连续输出使能** 0： CS 输出不连续 1： CS 输出连续
26 MODFEN	**主机模式故障检测使能** 0： 禁用主机模式故障检测功能 1： 使能主机模式故障检测功能
25 CSOE	**CS 硬件输出使能** 0： 禁用 CS 硬件输出 1： 使能 CS 硬件输出
23：20 FRMSIZE	**帧大小** 0000： 4bit 0001： 4bit 0010： 4bit 0011： 4bit ``` 1110： 15bit 1111： 16bit
19	**RX 最高有效位优先**

字段	说明
RMSBF	
	0：移位器移位的第一位是输入数据的 LSB
	1：移位器移位的第一位是输入数据的 MSB
18 MSBF	**TX 最高有效位优先** 0： TX LSB 优先(LSB 位首先移出) 1： TX MSB 优先(MSB 位首先移出)
17 CPHA	**时钟相位** 0： 第一个 SCK 过渡边沿为数据移出边沿 1： 第一个 SCK 过渡边沿为数据捕获边沿
16 CPOL	**时钟极性** 0： 空闲时，SCK 为 0 1： 空闲时，SCK 为 1
15 DMARXEN	**DMA RX 请求使能** 0： 禁用 DMA RX 请求 1： 使能 DMA RX 请求
14 DMATXEN	**DMA TX 通道使能** 0：禁用 DMA RX 请求 1：使能 DMA RX 请求
13 MODFIE	**模式故障中断使能** 0： 禁用 1： 使能
12 MSTR	**主机或从机模式选择** 0： 从机模式 1： 主机模式
11 RXOIE	**RX 缓冲区溢出中断使能** 0： 禁用 1： 使能
10 TXUIE	**TX 缓冲区下溢中断使能** 0： 禁用 1： 使能
9 RXFIE	**RX 缓冲区满中断使能** 0： 禁用 1： 使能 **注意： RX 缓冲区非空即可以产生中断**
8	**TX 缓冲区空中断使能**

字段	说明
TXEIE	0： 禁用 1： 使能 **注意： TX 缓冲区非满即可以产生中断**
7：0 CS_IDLE	**CS 空闲时间** CS 空闲时间 = (CS_IDLEA_COUNT+1)*CLK_PERIOD

18.6.3 命令寄存器 (SPI_CMD)

表 18-5 SPI_CMD 寄存器

SPI_CMD **命令寄存器** **Reset：0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称										RO TRIG	CS RLS	SW RST				SPI EN
访问										RW	RW	RW				RW
Reset										0	0	0				0

字段	说明
6 ROTRIG	**主机只读模式触发** **注意： 当 CFG2 中的 ROEN=1 时，将此位写入 ‘1’会触发读序列。读取此位将始终返回 ‘0’ 。**
5 CSRLS	**CS 释放** 0： 不起作用 1： 释放 CS 软件将此位写入‘1’，然后 CS 将变为高电平。读取此位始终返回 “0”，该位对 CS 连续输出有效(CONT_CS=1, CSOE=1)。
4 SWRST	**软件复位** 0： 不起作用 1： 复位 **注意：本复位位只能复位主机引擎/缓冲区/标志位逻辑，从机缓冲区 /标志位逻辑。CFG0/CFG1/CFG2/CMD 控制位不会复位。**

字段	说明
0 SPIEN	**SPI 使能** 0： 禁用 1： 使能

18.6.4 状态寄存器(SPI_STATUS)

表 18-6 SPI_STATUS 寄存器

SPI_STATUS **状态寄存器** **Reset：0x00000101**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称								IDLEF	MEBY			MODEF	RXOF	TXUF	RXFF	TXEF
访问								R	R			R	W1C	W1C	R	R
Reset								1	0			0	0	0	0	1

字段	说明
8 IDLEF	**SPI 空闲标志** 0： SPI 模块硬件非空闲 1： SPI 模块硬件空闲 **注意：对于主机，TX 缓冲区为空, RX 缓冲区为空，内部硬件空闲，该位可以为 '1'。** **对于从机， TX 缓冲区为空， RX 缓冲区为空, CS 无效(没有没选中) , 该位可以为 '1' 。**
7 MEBY	**SPI 主机引擎忙标志** 0： SPI 主机硬件一个字节传输完成 1： SPI 主机硬件一个字节传输未完成
4 MODEF	**主机模式故障检测标志** 0： 未检测到主机模式故障 1： 检测到主机模式故障 当 SPI 配置为主机时， 它会在驱动 CS 低电平之前检测 CS 线路状态。如果 CS 已经为低电平，表明另一个主机已经发起一个传输，置起模式故障检测错误标志。 **注： 发生该标志置时，需要配置 SWRST=1，SPI 进行软复位恢复。**
3 RXOF	**RX 缓冲区溢出标志** 0： 没有溢出

字段	说明
	1： RX 缓冲区溢出 接收缓冲区有两个 FIFO，如果接收到两个字节后，RX 缓冲区数据都没有被用户及时读取，下一个接收的数据过来，会产生 RX 溢出。 **注意：写 “1” 清零此位。产生溢出时，需要及时清除，以免影响 RXFF 标志和 DMA 接收。如果溢出，则 DATA 寄存器中仅能读出一个字节有效数据。**
2 TXUF	**TX 缓冲区下溢标志** 0： 没有下溢 1： TX 缓冲区下溢 在 slave 模式下，如果没有往 TX 数据寄存器写数据，master 端就开始通讯，会产生 TX 下溢。 **注意：写 “1” 清零此位。产生下溢时，需要及时清除。**
1 RXFF	**RX 缓冲区满标志** 0： 不满 1： 满 **注意：只要 rx 缓冲区中有有效的接收数据 (1 字节或 2 字节)，此位将为‘1’。因此该位 ‘1’ 不表示在 rx 缓冲区中接收到 2 个字节的数据。读取 DATA 寄存器将会自动清零该位 。此位称为 RX 缓冲区非空更为合理。**
0 TXEF	**TX 缓冲区空标志** 0： 非空 1： 空 **注意：只要 tx 缓冲区不满(2 字节数据)，该位将为‘1’，写 DATA 寄存器将自动清零此硬件位 。此位称为 TX 缓冲区非满更为合理。**

18.6.5 数据寄存器(SPI_DATA)

表 18-7 SPI_DATA 寄存器

SPI_DATA **数据寄存器** **Reset：0x00000000**

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

字段	说明
15:0 DATA	**SPI 数据端口寄存器** 读： 读操作将返回接收的 DATA 值 写： 写入要发送的 DATA

18.6.6 配置寄存器 2(SPI_CFG2)

表 18-8 SPI_CFG2 寄存器

SPI_CFG2 **配置寄存器 2** **Reset：0x00000000**

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

字段	说明
3 ROEN	**仅 RX 模式使能** 0：禁用 1：使能 **注意： 仅 RX 模式,，TXEF 将保持 0。即使 TXEIE=1 也不触发 TXEF IRQ。**
2 TOEN	**仅 TX 模式使能** 0： 禁用 1： 使能 **注意： 仅 TX 模式，在一个字节传输完成后，RXFF 不置位。即使 RXFIE=1，也不触发 RXFF IRQ。**
1 MNOV	**主机非溢出模式** 0： 禁用 1： 使能 如果 rxbuff 为满，则写入 txbuff 不会触发新的发送操作。

19 直接存储器访问（DMA）

19.1 简介

直接存储器访问控制器 (DMA) 用于在外设和存储器之间，或存储器与存储器之间的高速数据传输，无需 CPU 干预， 数据可以通过 DMA 快速地移动，提升了微处理器的性能。

共有 4 个通用通道用于不同类型的外设， 支持 UART, I2C, SPI, ADC。 还有一个轮询仲裁器来协调通道间的优先级。

19.2 特性

- 4 个独立的可配置通道
- 每个通道都直接连接至 UART, I2C, SPI, ADC 的硬件 DMA 请求，每个通道上都同样支持软件触发(存储器到存储器)。 该配置可通过软件完成
- 在同一个 DMA 模块上，多个请求间的优先级可以通过软件编程设置 (4 个等级分别为：很高，高，中等和低)，优先级设置相等时由硬件决定 (通道 0 优先级最高，通道 3 优先级最低，依次类推)
- 独立数据源和目标数据区的传输大小 (字节，半字，字)，源/目标地址必须按数据传输宽度对齐
- 支持循环的缓冲器管理
- 每个通道都有 3 个事件标志 (DMA 半传输， DMA 传输完成和 DMA 传输错误) ，这 3 个事件标志逻辑或成为一个单独的中断请求
- 存储器和存储器间的传输
- 外设和存储器 、存储器和外设间的传输
- SRAM 和 APB 外设均可作为 访问的源和目标
- 可编程的数据传输数目：最大为 32767

19.3 结构框图

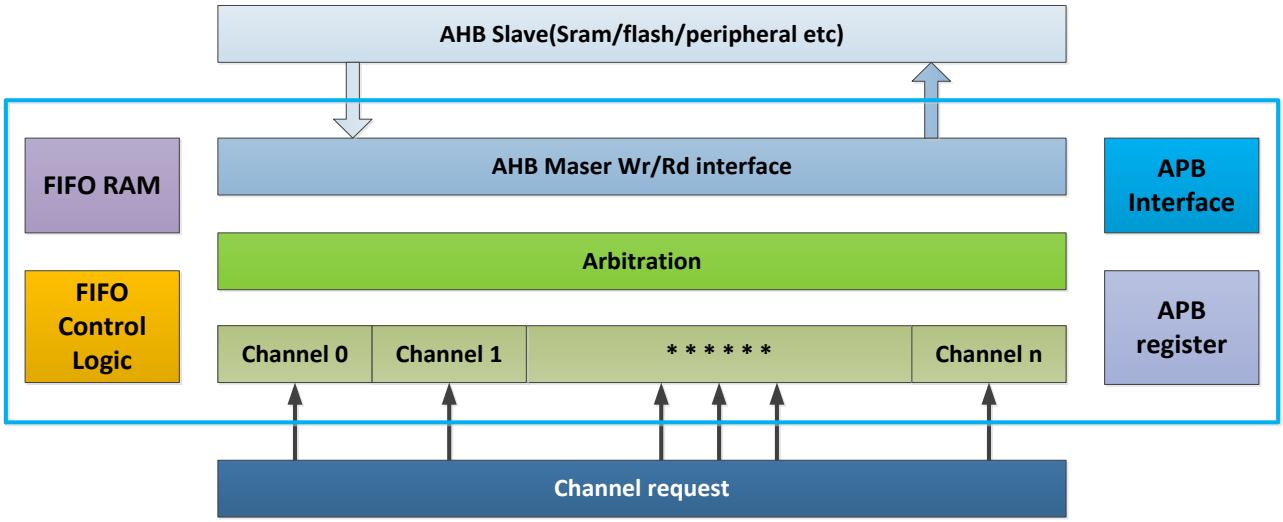

图 19-1 DMA 结构框图

19.4 功能描述

19.4.1 操作模式

DMA 模块不支持停止模式（Stop），也不支持待机模式（Standby）。在进入停止模式（Stop）或待机模式（Standby）前，必须关闭所有的 DMA 通道。

19.4.2 DMA 请求列表

表 19-1 DMA 请求列表

外设	DMA 通道 0, 1, 2, 3 请求
ADC	ADC
SPI	SPI0_TX/ SPI0_RX/ SPI1_TX/ SPI1_RX
UART	UART0_TX/ UART0_RX/ UART1_TX/ UART1_RX/ UART2_TX/ UART2_RX
I2C	I2C0_TX/

外设	DMA 通道 0, 1, 2, 3 请求
	I2C0_RX/ I2C1_TX/ I2C1_RX

19.4.3 仲裁器

仲裁器根据通道请求的优先级来管理通道请求，并启动外设/存储器访问序列。

优先级管理分如下 2 个阶段：

- 软件：每个通道的优先级可在通道配置寄存器 DMA_CONFIG 中配置，共有 4 个等级：
  - 最高优先级
  - 高优先级
  - 中等优先级
  - 低优先级
- 当软件优先级相等时，优先级根据硬件通道号来决定。如通道 0 优先级最高，通道 3 优先级最低，依次类推。

19.5 应用说明

在用户配置通道使能寄存器 DMA_CHAN_ENABLE 之前，应编程除寄存器 DMA_CHAN_ENABLE 之外的所有寄存器。DMA 配置参数在 CHAN_ENABLE=1 生效。当 CHAN_ENABLE 为 1 时，请不要修改其他配置参数。

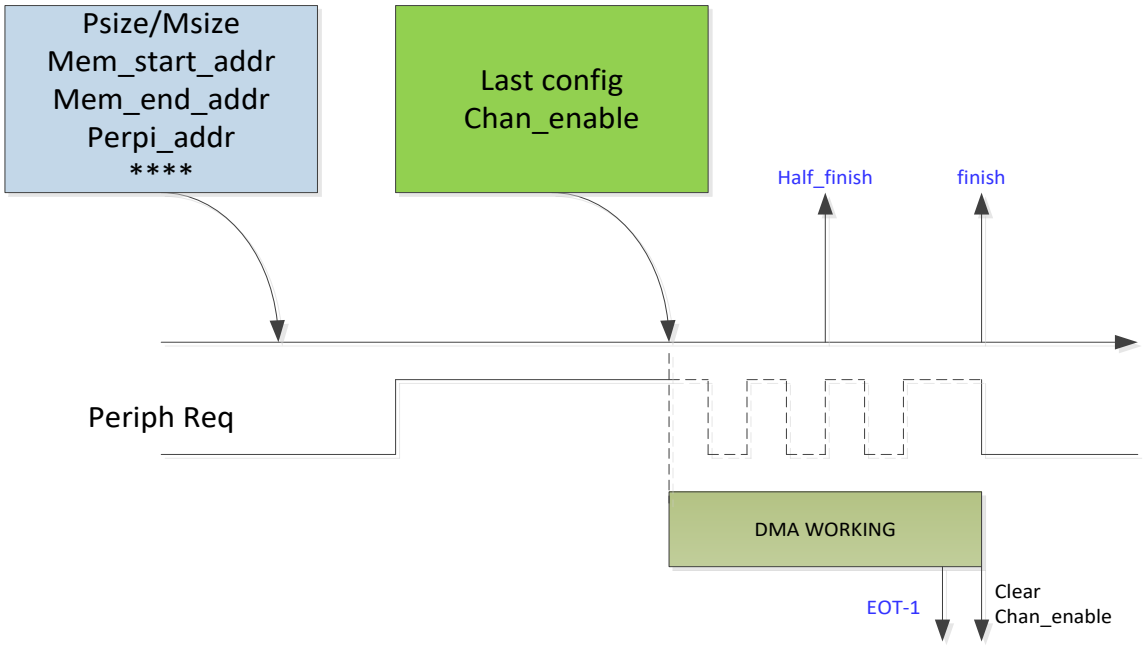

图 19-2 DMA 配置指南

19.5.1 软复位和硬复位

DMA 全局控制和每个 DMA 引擎中都存在软复位和硬复位。设置软复位后，在当前事务完成后引擎将复位，因此软复位不会导致任何总线挂起。相反，当设置硬复位时，引擎将立即复位，因此总线可能由于未完成（并且永远不会完成）事务而停机。

全局软复位的机制为：当软件想要重启所有引擎或重新清除 DMA 中的所有引擎时，它可以将全局软复位器设置为 1。然后 HW 将 WARM_RST 设置为 0 以完成全局软复位。

全局硬复位的机制为：当软件想要重启所有引擎或重新清除 DMA 中的所有引擎而立即执行不需任何时间等待，用户可以将全局 HARD_RST 设置为 1 后，再设置为 0 以完成全局硬复位。请注意，这可能会破坏总线协议并导致系统挂起。

19.5.2 暂停和恢复

DMA 有暂停和恢复功能，机制如下：

1. 开始 DMA（程序根据需要设置后，然后设置 CHAN_ENABLE = 1）；
2. 暂停 DMA（设置 STOP = 1）；
3. 恢复 DMA（设置 STOP = 0）；
4. 等待 DMA 完成 （CHAN_ENABLE 将变为 0，中断标志将设置为 1）。

DMA 运行时，软件可以多次重复步骤 2 和 3。DMA 不会立即暂停，并等待最后一个事务完成。

19.5.3 通道循环

循环模式用于处理循环缓冲区和连续的数据传输（如 ADC 扫描模式）。通道配置寄存器 DMA_CONFIG 的 CHAN_CIRCULAR 位用于使能该特性。当启动了循环模式，数据传输的数目完成时，将会自动地被恢复成通道配置时设置的初值，DMA 操作将会继续进行。

需要注意的是，DMA 主机搬运完 MEM_END_ADDR-0x01 地址的数据后，回到 MEM_START_ADDR 地址继续搬运。

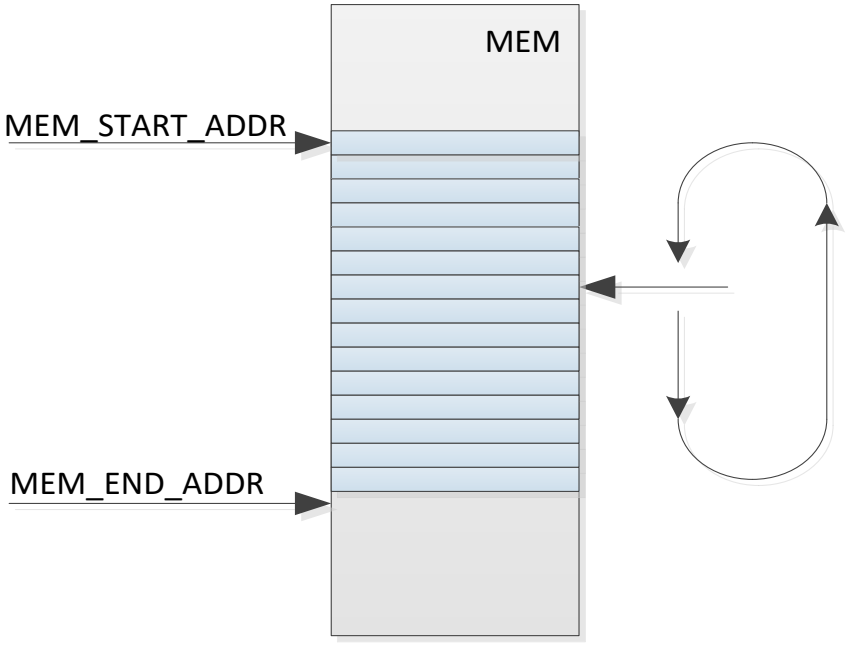

图 19-3 DMA 通道循环

19.5.4 I2C 使用 DMA

当达到为相应 DMA 编程的数据传输次数时，DMA 控制器将一个传输结束 EOT-1 信号发送到 I2C 模块，I2C 硬引擎收到信号并决定是否发送 ACK/NACK 信号。

19.5.5 可编程数据传输宽度和数据对齐

表 19-2 可编程数据传输宽度 &数据对齐

内存宽度(MSIZE)	外设宽度(PSIZE)	传输数目(Length)	Byte 模式(MEM_BYTE_MODE)	源：地址/数据	传输方向(Dir)	传输操作	目标：地址/数据
32	8	4	00	0x00/03020100 0x04/07060504 0x08/0B0A0908 0x0C/0F0E0D0C	1(TX)	1. 在 0x00 读 03020100，往 0x00 写入 00 2. 在 0x04 读 07060504，往 0x01 写入 04 3. 在 0x08 读 0B0A0908，往 0x02 写入 08 4. 在 0x0C 读 0F0E0D0C，往 0x03 写入 0C	0x00/00 0x01/04 0x02/08 0x03/0C
32	16	4	00	0x00/03020100 0x04/07060504 0x08/0B0A0908 0x0C/0F0E0D0C	1(TX)	1. 在 0x00 读 03020100，往 0x00 写入 0100 2. 在 0x04 读 07060504，往 0x02 写入 0504 3. 在 0x08 读 0B0A0908，往 0x04 写入 0908 4. 在 0x0C 读 0F0E0D0C，往 0x06 写入 0D0C	0x00/0100 0x02/0504 0x04/0908 0x06/0D0C
32	32	4	00	0x00/03020100 0x04/07060504 0x08/0B0A0908 0x0C/0F0E0D0C	1(TX)	1. 在 0x00 读 03020100，往 0x00 写入 03020100 2. 在 0x04 读 07060504，往 0x04 写入 07060504。 3. 在 0x08 读 0B0A0908，往 0x08 写入 0B0A0908 4. 在 0x0C 读 0F0E0D0C，	0x00/03020100 0x04/07060504 0x08/0B0A0908 0x0C/0F0E0D0C

内存宽度(MSIZE)	外设宽度(PSIZE)	传输数目(Length)	Byte 模式(MEM_BYTE_MODE)	源：地址/数据	传输方向(Dir)	传输操作	目标：地址/数据
						往 0x0C 写入 0F0E0D0C	
32	8	4	01	0x00/03020100 0x04/07060504	1(TX)	1. 在 0x00 读 03020100, 再拆分成 0302 和 0100 往 0x00 写入 00 往 0x01 写入 02 2. 在 0x04 读 07060504, 再拆分成 0706 和 0504 往 0x02 写入 04 往 0x03 写入 06	0x00/00 0x01/02 0x02/04 0x03/06
32	16	4	01	0x00/03020100 0x04/07060504	1(TX)	1. 在 0x00 读 03020100, 再拆分成 0302 和 0100 往 0x00 写入 0100 往 0x02 写入 0302 2. 在 0x04 读 07060504, 再拆分成 0706 和 0504 往 0x04 写入 0504 往 0x06 写入 0706	0x00/0100 0x02/0302 0x04/0504 0x06/0706
32	32	4	01	0x00/03020100 0x04/07060504	1(TX)	1. 在 0x00 读 03020100， 再拆分成 0302 和 0100 往 0x00 写入 **0000**0100 往 0x04 写入 **0000**0302 2. 在 0x04 读 07060504, 再拆分成 0706 和 0504 往 0x08 写入 **0000**0504 往 0x0C 写入 **0000**0706	0x00/00000100 0x04/00000302 0x08/00000504 0x0C/00000706

内存宽度(MSIZE)	外设宽度(PSIZE)	传输数目(Length)	Byte 模式(MEM_BYTE_MODE)	源：地址/数据	传输方向(Dir）	传输操作	目标：地址/数据
32	8	4	11	0x00/03020100	1(TX)	1. 在 0x00 读 03020100, 再拆分成 03、02、01、00 共 4byte 往 0x00 写入 00 往 0x01 写入 01 往 0x02 写入 02 往 0x03 写入 03	0x00/00 0x01/01 0x02/02 0x03/03
32	16	4	11	0x00/03020100	1(TX)	1. 在 0x00 读 03020100, 再拆分成 03、02、01、00 共 4byte 往 0x00 写入 **00**00 往 0x02 写入 **00**01 往 0x04 写入 **00**02 往 0x06 写入 **00**03	0x00/0000 0x02/0001 0x04/0002 0x06/0003
32	32	4	11	0x00/03020100	1(TX)	1. 在 0x00 读 03020100, 再拆分成 03、02、01、00 共 4byte 往 0x00 写入 **000000**00 往 0x04 写入 **000000**01 往 0x08 写入 **000000**02 往 0x0C 写入 **000000**03	0x00/00000000 0x04/00000001 0x08/00000002 0x0C/00000003
8	32	4	00	0x00/03020100 0x04/07060504 0x08/0B0A0908 0x0C/0F0E0D0C	0(RX)	1. 在 0x00 读 03020100， 往 0x00 写入 00 2. 在 0x04 读 07060504， 往 0x01 写入 04 3. 在 0x08 读 0B0A0908， 往 0x02 写入 08 4. 在 0x0C 读 0F0E0D0C, 往 0x03 写入 0C	0x00/00 0x01/04 0x02/08 0x03/0C

内存宽度(MSIZE)	外设宽度(PSIZE)	传输数目(Length)	Byte 模式(MEM_BYTE_MODE)	源：地址/数据	传输方向(Dir）	传输操作	目标：地址/数据
16	32	4	00	0x00/03020100 0x04/07060504 0x08/0B0A0908 0x0C/0F0E0D0C	0(RX)	1. 在 0x00 读 03020100, 往 0x00 写入 0100 2. 在 0x04 读 07060504, 往 0x02 写入 0504 3. 在 0x08 读 0B0A0908, 往 0x04 写入 0908。 4. 在 0x0C 读 0F0E0D0C, 往 0x06 写入 0D0C	0x00/0100 0x02/0504 0x04/0908 0x06/0D0C
32	32	4	00	0x00/03020100 0x04/07060504 0x08/0B0A0908 0x0C/0F0E0D0C	0(RX)	1. 在 0x00 读 03020100, 往 0x00 写入 03020100。 2. 在 0x04 读 07060504, 往 0x04 写入 07060504 3. 在 0x08 读 0B0A0908, 往 0x08 写入 0B0A0908 4.在 0x0C 读 0F0E0D0C, 往 0x0C 写入 0F0E0D0C	0x00/03020100 0x04/07060504 0x08/0B0A0908 0x0C/0F0E0D0C
32	32	4	01	0x00/03020100 0x04/07060504 0x08/0B0A0908 0x0C/0F0E0D0C	0(RX)	1. 在 0x00 读 03020100, 将 0100 保存在 DMA 内部缓存的 BIT[15:0]， 0302 丢弃;	0x00/05040100 0x04/0D0C0908

内存宽度(MSIZE)	外设宽度(PSIZE)	传输数目(Length)	Byte 模式(MEM_BYTE_MODE)	源：地址/数据	传输方向(Dir）	传输操作	目标：地址/数据
						在 0x04 读 07060504,将 0504 保存在 DMA 内部缓存的 BIT[31:16]，0706 丢弃，重新拼成的 32bit 数据再放入 0x00 处 2. 在 0x08 读 0B0A0908, 将 0908 保存在 DMA 内部缓存的 BIT[15:0]，0B0A 丢弃; 在 0x0C 读 0F0E0D0C, 将 0D0C 保存在 DMA 内部缓存的 BIT[31:16]，0F0E 丢弃，重新拼成的 32bit 数据再放入 0x04 处	
32	32	4	11	0x00/03020100 0x04/07060504 0x08/0B0A0908 0x0C/0F0E0D0C	0(RX)	1. 在 0x00 读 03020100, 将 00 保存在 DMA 内部缓存的 BIT[7:0] 2. 在 0x04 读 07060504,将 04 保存在 DMA 内部缓存的 BIT[15:8] 3. 在 0x08 读 0B0A0908，将 08 保存在 DMA 内部缓存的 BIT[23:16] 4. 在 0x0C 读 0F0E0D0C, 将 0C 保存在 DMA 内部缓存的 BIT[31:24] 重新拼成的 32bit 数据再放入 0x00 处	0x00/0c080400

【说明】

1. 当 dir=1(TX)，数据从 MEM 往 PERIPH 搬运；

   当 dir=0(RX)，数据从 PERIPH 往 MEM 搬运。

2. 部分 0 加粗表示是 DMA 为匹配 DEST SIZE 额外补充的，并不是从 SRC 读到的。

3. 当 dir=1(TX)时，MSIZE 只有 32，表示 DMA 从 MEM 恒定读 32bit 数据。

   当 dir=0(RX)时，PSIZE 只有 32，表示 DMA 从 PERIPH 恒定读 32bit 数据，即使外设寄存器有效位只有 8bit。

4. 使用 MEM2MEM 的方式搬运数据时，需将 MSIZE 与 PSIZE 设置成一样的大小。

19.5.6 通道配置过程

1. 配置该 DMA 通道 MEM 的起始地址，结束地址：DMA_MEM_START_ADDR，DMA_MEM_END_ADDR；
2. 配置该 DMA 通道的 PERIPH 的地址：DMA_PERIPH_ADDR；
3. 配置该 DMA 通道的中断使能：DMA_INTEN，可以配置为半完成，完成， 传输错误中断；
4. 配置该 DMA 通道的 DMA_CONFIG 寄存器：PERIPH_SEL，CHAN_PRIORITY，CHAN_CIRCULAR，CHAN_DIR，MEM2MEM，MEM_INCREMENT，PERIPH_INCREMENT，MEM_BYTE_MODE，MEM_SIZE，PERIPH_SIZE。

   ① 配置为 MEM 到 PERIPH

   配置 CHAN_DIR=1（从存储器读取），MEM2MEM=0（在非存储器和存储器间传输），MEM_INCREMENT=1（MEM 地址增加），PERIPH_INCREMENT=0（外设地址固定），PERIPH_SEL 选择为 UARTx_TX/ SPIx_TX/ I2Cx_TX。

   ② 配置为 PERIPH 到 MEM

   配置 CHAN_DIR=0（从外设读取），MEM2MEM=0（在非存储器和存储器间传输），MEM_INCREMENT=1（MEM 地址增加），PERIPH_INCREMENT=0（外设地址固定），PERIPH_SEL 选择为 UARTx_RX/ SPIx_RX/ I2Cx_RX/ ADC。

   ③ 配置为 MEM 到 MEM

   配置 CHAN_DIR=1（从存储器读取），MEM2MEM=1（在存储器和存储器间传输），MEM_INCREMENT=1（MEM 地址增加），PERIPH_INCREMENT=1（外设地址增加）。

5. 配置该 DMA 通道的 DMA_CHAN_LENGTH 寄存器，该寄存器不是指的是总字节长度，而是 DMA 通道搬运的次数。
6. 使能该 DMA 通道，即配置 DMA_CHAN_ENABLE 寄存器的 CHAN_ENABLE=1。

19.6 寄存器定义

表 19-3 DMA 寄存器映射

DMA0 基地址： 0x40012000
DMA0_Channel0 基地址： 0x40012040
DMA0_Channel1 基地址： 0x40012080
DMA0_Channel2 基地址： 0x400120c0
DMA0_Channel3 基地址： 0x40012100

地址	名称	宽度	描述
DMA0 基地址+0x00	DMA_TOP_RST	32	通用 DMA 复位寄存器
DMA0_Channelx 基地址+0x00	DMA_STATUS	32	状态寄存器
DMA0_Channelx 基地址+0x04	DMA_INTEN	32	中断使能寄存器
DMA0_Channelx 基地址+0x08	DMA_RST	32	通道复位寄存器
DMA0_Channelx 基地址+0x0C	DMA_STOP	32	通道停止寄存器
DMA0_Channelx 基地址+0x10	DMA_CONFIG	32	通道配置寄存器
DMA0_Channelx 基地址+0x14	DMA_CHAN_LENGTH	32	通道长度寄存器
DMA0_Channelx 基地址+0x18	DMA_MEM_START_ADDR	32	存储器起始地址寄存器
DMA0_Channelx 基地址+0x1C	DMA_MEM_END_ADDR	32	存储器结束地址寄存器
DMA0_Channelx 基地址+0x20	DMA_PERIPH_ADDR	32	通道外设地址寄存器
DMA0_Channelx 基地址+0x24	DMA_CHAN_ENABLE	32	通道使能寄存器
DMA0_Channelx 基地址+0x28	DMA_DATA_TRANS_NUM	32	数据传输数目寄存器
DMA0_Channelx 基地址+0x2C	DMA_INTER_FIFO_DATA_LEFT_NUM	32	FIFO 数据剩余数目寄存器

19.6.1 通用 DMA 复位寄存器(DMA_TOP_RST)

表 19-4 DMA_TOP_RST 寄存器

DMA_TOP_RST 通用 DMA 复位寄存器 **Reset：0x00000000**

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

字段	说明
1 HARD_RST	**通用 DMA 硬复位(不论当前事务如何，都复位)** 0： 禁用 1： 使能 软件设置'HARD_RST'为 1，然后设置 ' HARD_RST ' 回 0，则该复位机制完成。HARD_RST 会使所有 DMA Channel 寄存器恢复默认值。
0 WARM_RST	**通用 DMA 软复位 (在当前事务完成后复位)** 0： 禁用 1： 使能 软件设置 'WARM_RST' 为 1 ，然后硬件清零 ' WARM_RST ' 位回 0，则该复位机制完成。WARM_RST 会使所有 DMA Channel 寄存器恢复默认值(除了 DMA_INTEN 寄存器)。

19.6.2 状态寄存器(DMA_STATUS)

表 19-5 DMA_STATUS 寄存器

DMA_STATUS 状态寄存器 Reset：0x00000000

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称														TRANS_ERROR	HALF_FINISH	FINISH
访问														W0C	W0C	W0C
Reset														0	0	0

字段	说明
2 TRANS_ERROR	**传输错误标志;** 0： 错误没有发生 1： 错误发生 指示当通道 n 读或写时，是否出现错误标志。 对该位置“0”将会清除此状态。
1 HALF_FINISH	**半完成标志** 0： 数据传输未完成一半 1： 数据传输完成一半 指示当通道 n 读或写时，是否完成半数数据传输。对该位置“0”将会清除此状态。
0 FINISH	**完成标志** 0： 数据传输未完成 1： 数据传输完成 指示当通道 n 读或写时，是否完成 DMA 通道数据传输。对该位置“0”将会清除此状态。

19.6.3 中断使能寄存器(DMA_INTEN)

表 19-6 DMA_INTEN 寄存器

DMA_INTEN 中断使能寄存器 **Reset：0x00000000**

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

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称														TRANS_ERROR INTERRUPT ENABLE	HALF_FINISH INTERRUPT ENABLE	FINISH INTERRUPT ENABLE
访问														RW	RW	RW
Reset														0	0	0

字段	说明
2 TRANS_ERROR INTERRUPT ENABLE	**TRANS_ERROR 中断使能** 0： 中断禁用 1： 中断使能
1 HALF_FINISH INTERRUPT ENABLE	**HALF_FIISH 中断使能** 0： 中断禁用 1： 中断使能 只有 DMA_CHAN_LENGTH≥8 时，半完成中断才生效。
0 FINISH INTERRUPT ENABLE	**FINISH 中断使能** 0： 中断禁用 1： 中断使能

19.6.4 通道复位寄存器(DMA_RST)

表 19-7 DMA_RST 寄存器

DMA_RST 通道复位寄存器 **Reset：0x00000000**

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

字段	说明
2 FLUSH	**DMA 通道刷新** 0： 禁用 1： 使能 设置 FLUSH = 1 将停止 DMA 并允许 DMA 将其内部缓冲区残留数据刷新至存储器。刷新完成后，DMA 将通道使能设置为 0 并停止 DMA。软件设置 FLUSH = 1 并等待 HW 清除为 0。
1 HARD_RST	**DMA 通道硬复位 (不论当前事务如何，都复位)** 0： 禁用 1： 使能 软件设置 ' HARD_RST ' 为 1，然后将 ' HARD_RST ' 设置回 0， 硬复位机制完成。
0 WARM_RST	**DMA 通道软复位(在当前事务后复位)** 0： 禁用 1： 使能 软件设置 ' WARM_RST ' 为 1，然后硬件自动将 ' WARM_RST ' 设置回 0， 软复位机制完成

19.6.5 通道停止寄存器(DMA_STOP)

表 19-8 DMA_STOP 寄存器

DMA_STOP 通道停止寄存器 **Reset：0x00000000**

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

字段	说明
0 STOP	**DMA 通道暂停 (在当前事务后暂停)** 0： 没有停止通道传输 1： 暂停通道传输 软件设置' STOP ' 为 1，然后在当前事务后，传输暂停，软件设置 ' STOP ' to 为 0，然后继续数据传输。从 STOP 模式到继续 DMA 运行， MEM 到 MEM 或 MEM 到外设是不会丢数据的，外设到 MEM 可能会丢数据，取决于 STOP 之后，外设接收数据 FIFO 是否有溢出。

19.6.6 通道配置寄存器(DMA_CONFIG)

表 19-9 DMA_CONFIG 寄存器

DMA_CONFIG 通道配置寄存器 **Reset：0x00000050**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称													PERIPH_SEL			
访问													RW			
Reset													0			
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称				MEM_BYTE_MODE		CHAN_DIR	CHAN_CIRCULAR	PERIPH_INCREMENT	MEM_INCREMENT	PERIPH_SIZE		MEM_SIZE		CHAN_PRIORITY		MEM2MEM
访问				RW		RW	RW	RW	RW	RW		RW		RW		RW
Reset				0		0	0	0	0	10		10		0		0

字段	说明
19:16 PERIPH_SEL	**外设选择** 0000：UART0_TX 0001：UART0_RX 0010：UART1_TX 0011：UART1_RX 0100：UART2_TX 0101：UART2_RX 0110：SPI0_TX 0111：SPI0_RX 1000：SPI1_TX 1001：SPI1_RX 1010：I2C0_TX 1011：I2C0_RX 1100：I2C1_TX 1101：I2C1_RX 1110：ADC
12：11 MEM_BYTE_MODE	**标识 MEM 字分割传输数** 00： 1 01： 2 10： 2 11： 4
10 CHAN_DIR	**标识数据传输方向** 0： 从外设读取 1： 从存储器读取
9 CHAN_CIRCULAR	**循环模式** 0： 循环模式禁用 1： 循环模式使能
8 PERIPH_INCREMENT	**外设地址增量模式** 0： 外设地址固定 1： 外设地址增加
7 MEM_INCREMENT	**存储器地址增量模式** 0： MEM 地址固定 1： MEM 地址增加
6：5 PERIPH_SIZE	**外设数据宽度** 00： 8 位 01：16 位 10：32 位

字段	说明
	11： 保留
4：3 MEM_SIZE	**存储器数据宽度** 00： 8 位 01：16 位 10：32 位 11： 保留
2：1 CHAN_PRIORITY	**通道优先级** 00： 低 01： 中等 10： 高 11： 很高
0 MEM2MEM	**存储器到存储器间模式** 0： 在非存储器和存储器间传输 1： 在存储器和存储器间传输

19.6.7 通道长度寄存器(DMA_CHAN_LENGTH)

表 19-10 DMA_CHAN_LENGTH 寄存器

DMA_CHAN_LENGTH **通道长度寄存器** **Reset：0x00000000**

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

字段	说明
15：0 CHAN_LENGTH	**DMA 通道传送次数** 0~32767 第 15 位应该为 0 **注意：I2C 使用 DMA 传送时，需要配置 CHAN_LENGTH >= 2。**

19.6.8 存储器起始地址寄存器(DMA_MEM_START_ADDR)

表 19-11 DMA_MEM_START_ADDR 寄存器

DMA_MEM_START_ADDR 存储器起始地址寄存器 **Reset：0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	MEM_START_ADDR[31：16]															
访问	RW															
Reset	0															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	MEM_START_ADDR [15：0]															
访问	RW															
Reset	0															

字段	说明
31：0 MEM_START_ADDR	**MEM_START_ADDR** Memory 起始地址

19.6.9 存储器结束地址寄存器(DMA_MEM_END_ADDR)

表 19-12 DMA_MEM_END_ADDR 寄存器

DMA_MEM_END_ADDR 存储器结束地址寄存器 **Reset：0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	MEM_END_ADDR[31：16]															
访问	RW															
Reset	0															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	MEM_END_ADDR [15：0]															
访问	RW															
Reset	0															

字段	说明
31：0 MEM_END_ADDR	**MEM_END_ADDR** DMA 主机搬运完 MEM_END_ADDR-0x01 地址数据后，回到 MEM_START_ADDR 地址继续搬运。

19.6.10 通道外设地址寄存器(DMA_PERIPH_ADDR)

表 19-13 DMA_PERIPH_ADDR 寄存器

DMA_PERIPH_ADDR 通道外设地址寄存器 **Reset：0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	PERIPH_ADDR[31：16]															
访问	RW															
Reset	0															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	PERIPH_ADDR[15：0]															
访问	RW															
Reset	0															

字段	说明
31：0 PERIPH_ADDR	**PERIPH_ADDR** 外设地址

19.6.11 通道使能寄存器(DMA_CHAN_ENABLE)

表 19-14 DMA_CHAN_ENABLE 寄存器

DMA_CHAN_ENABLE 通道使能寄存器 **Reset：0x00000000**

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

字段	说明
0 CHAN_ENABLE	**通道使能** 0： 禁用通道 1： 使能通道 在非通道循环模式下，软件设置 CHAN_ENABLE 为 1, 当传输完成时， 硬件将 CHAN_ENABLE 清零。 在通道循环模式下，软件设置 CHAN_ENABLE 为 1, 当传输完成时， 硬件使用相同的配置重新开始新的传输。如果需要关闭 DMA 通道，必须先关闭循环模式。

19.6.12 数据传输数目寄存器 (DMA_DATA_TRANS_NUM)

表 19-15 DMA_DATA_TRANS_NUM 寄存器

DMA_DATA_TRANS_NUM 数据传输数目寄存器 Reset：0x00000000

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

字段	说明
15：0 DATA_TRANS_NUM	**DATA_TRANS_NUM** 标识已传输了多少次

19.6.13 FIFO 数据剩余数目寄存器(DMA_INTER_FIFO_DATA_LEFT_NUM)

表 19-16 DMA_INTER_FIFO_DATA_LEFT_NUM 寄存器

DMA_INTER_FIFO_DATA_LEFT_NUM FIFO 数据剩余数目寄存器 Reset：0x00000000

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

字段	说明
5：0 INTER_FIFO_DATA_LEFT_NUM	**INTER_FIFO_DATA_LEFT_NUM** 标识 fifo 中剩余了多少字节数据，通常与 FLUSH 一起使用。当超时且 INTER_FIFO_DATA_LEFT_NUM 不等于 0 时，软件可以启动刷新过程。

20 看门狗模块（Watchdog）

20.1 简介

看门狗是一个独立定时器，一般用来检测系统软件程序是否按预期运行。如果看门狗模块没有被按时刷新，看门狗会产生系统复位，比如程序中存在死循环，但因某些原因没有跳出，或者使用多任务操作系统，喂狗任务没有按时得到执行，都会由于看门狗模块没有得到被刷新而产生系统复位。一般用于高安全性场合。

20.2 特性

- 4 种时钟源
- 可编程超时时间
- 窗口模式
- 超时中断

20.3 结构框图

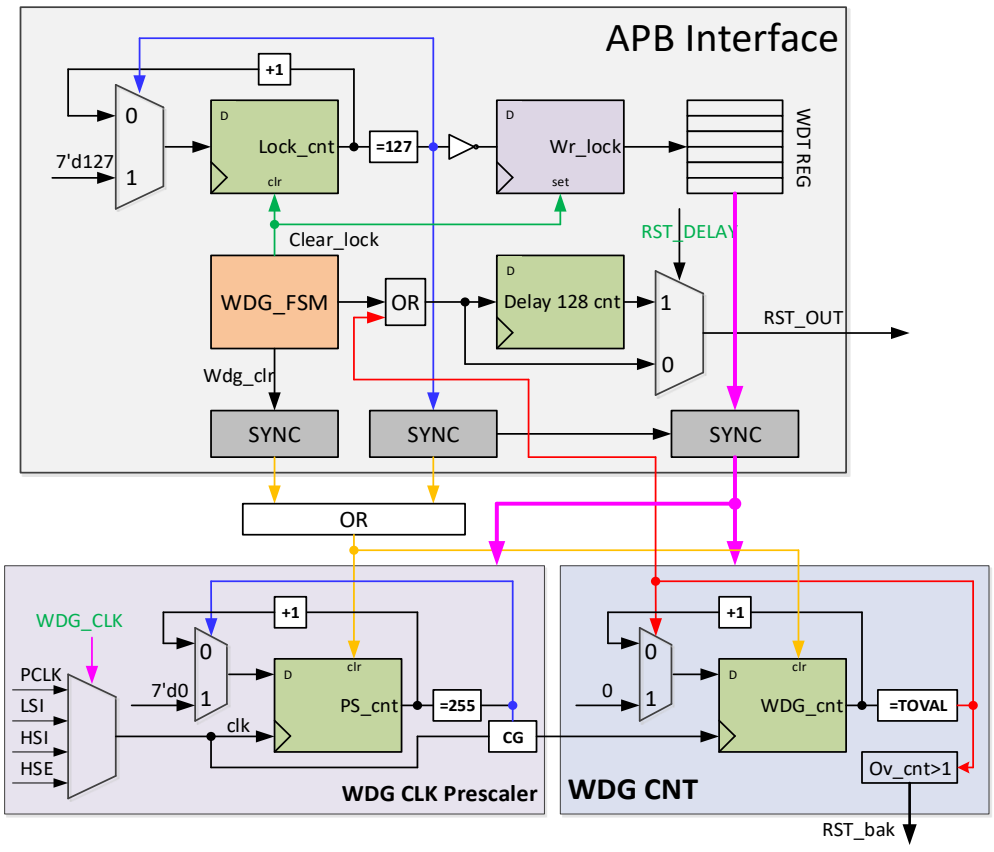

图 20-1 WDG 结构框图

20.4 功能描述

20.4.1 基本看门狗

看门狗具有四种时钟源：总线时钟、内部 32kHz 时钟、内部 8MHz 时钟、外部 XOSC 时钟。看门狗定时器使用一个 32 位的可编程向上计数器以及一个可选的固定 256 分频器。

看门狗使能后，开始计数，若计数到 TOVAL 值，会产生系统复位，在计数达到 TOVAL 值前，刷新看门狗会使计数器复位并重新开始计数。

20.4.2 窗口看门狗

看门狗具有窗口模式，在该模式下，在计数器达到窗口值 WIN 前刷新看门狗或者计数器达到 TOVAL 值前未刷新看门狗都会导致复系统复位，在计数器大于 WIN 值，小于 TOVAL 值之间刷新看门狗会使计数器复位并重新开始计数。

20.4.3 低功耗行为

MCU 在 Stop 模式下，看门狗可以保持运行，但需要以内部 32kHz 时钟作为时钟源，此时的复位时间是正常的两倍。在 Standby 模式下，看门狗模块不会起作用。

20.5 应用说明

20.5.1 配置看门狗

对看门狗的所有寄存器进行配置的条件是更新位 WDG_CS0[UPDATE]为 1 且看门狗解锁。解锁后，只能在 128 个总线时钟内对看门狗的任意寄存器进行配置，然后所有寄存器自动上锁，再次配置需要再次满足上述两个条件。

在 WDG_CS0[UPDATE]为 0 的情况下解锁会引起系统复位。

解锁序列：向 WDG_CNT 寄存器先后写入 0xE064D987、0x868A8478，写入的值不正确或顺序颠倒会引起系统复位。

20.5.2 刷新看门狗

在基本看门狗与窗口看门狗模式下，为了保证看门狗不复位系统，需要软件在规定时间内刷新看门狗。看门狗刷新后，看门狗计数器重新从 0 开始计数，软件需要再次刷新看门狗。这种机制使得软件必须定时刷新看门狗，在一定程度上反映了程序的正常运行，当程序意外跑飞后，看门狗会复位系统。

刷新序列：向 WDG_CNT 寄存器先后写入 0x7908AD15、0x5AD5A879，写入的值不对或顺序颠倒会引起系统复位。

20.5.3 看门狗中断

看门狗具有中断功能，中断使能后，若看门狗计数器超时，看门狗不会立即复位系统，而是延迟 128 个总线时钟后复位系统。这 128 个总线时钟是留给中断服务程序响应的时间，用户软件可以在中断程序中进行简单的程序处理，但不建议在此时刷新看门狗。

20.6 寄存器定义

表 20-1 WDG 寄存器映射

WDG 基地址：0x4000b000

地址	名称	宽度	描述
WDG 基地址+0x00	WDG_CS0	32	配置寄存器 CS0
WDG 基地址+0x04	WDG_CS1	32	配置寄存器 CS1
WDG 基地址+0x08	WDG_CNT	32	计数器寄存器
WDG 基地址+0x0C	WDG_TOVAL	32	超时值寄存器
WDG 基地址+0x10	WDG_WIN	32	窗口值寄存器

20.6.1 配置寄存器 0(WDG_CS0)

表 20-2 WDG_CS0 寄存器

WDG_CS0 **配置寄存器 CS0** **Reset: 0x00000020**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									EN	INT	UPDATE					
访问									RW	RW	RW					
Reset									0/1	0	1					

字段	说明
7 EN	**看门狗使能** 0：看门狗禁用 1： 看门狗使能 使能看门狗计数器开始计数。 若选项字节关闭，使能位默认为 0，若选项字节打开，使能位默认为 1。

字段	说明
6 INT	**看门狗中断** 0：看门狗中断禁用，看门狗复位系统无延迟 1：看门狗中断使能，看门狗复位系统延迟 128 个总线时钟 在看门狗超时或非法写入时，若使能了看门狗中断，会触发看门狗中断，并在中断响应 128 个总线时钟后再发生系统复位。
5 UPDATE	**看门狗配置更新位** 0：不允许更新。完成初始配置后，除非强制复位，否则无法修改看门狗寄存器配置 1：允许更新。执行解锁写入序列后，软件可以在 128 个总线时钟内修改看门狗配置寄存器

20.6.2 配置寄存器 1(WDG_CS1)

表 20-3 WDG_CS1 寄存器

WDG_CS1 **配置寄存器 CS1** **Reset: 0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									WIN	FLG		PRES			CLK	
访问									RW	R		RW			RW	
Reset									0	0		0			0	

字段	说明
7 WIN	**看门狗窗口** 0： 窗口模式禁用 1 ： 窗口模式使能 使能看门狗窗口模式
6 FLG	**看门狗中断标志** 0： 未发生中断 1 ： 发生中断 该位指示中断产生，写入 1 将其清零

字段	说明
4 PRES	**看门狗预分频器** 0：256 预分频器禁用 1：256 预分频器使能 使能看门狗计数器 256 倍预分频
1: 0 CLK	**看门狗时钟** 00： 总线时钟 01：32kHz 内部 RC 振荡器 10：8MHz 内部 RC 振荡器 11： 外部 XOSC 选择看门狗计数器时钟源

20.6.3 计数器寄存器(WDG_CNT)

表 20-4 WDG_CNT 寄存器

WDG_CNT **计数器寄存器** **Reset: 0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	CNT[31: 16]															
访问	R															
Reset	0															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	CNT[15: 0]															
访问	R															
Reset	0															

字段	说明
31: 16 CNT	**看门狗计数值** 看门狗计数器寄存器指示当前看门狗计数器的数值，软件可在任意时刻对计数器寄存器进行读操作。软件无法直接更改计数器的值。 但对这个寄存器的写操作具有特殊功能：刷新序列和解锁序列。

20.6.4 模值寄存器(WDG_TOVAL)

表 20-5 WDG_TOVAL 寄存器

WDG_TOVAL 超时值寄存器 **Reset: 0x001F8000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	TOVAL[31: 16]															
访问	RW															
Reset	0x1F															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	TOVAL[15: 0]															
访问	RW															
Reset	0x8000															

字段	说明
31: 0 TOVAL	**看门狗超时值寄存器** 看门狗计数器与超时数值连续作比较。若计数器达到超时值，则看门狗复位系统。计时长度等于 WDG_TOVAL+1，默认值为 0x1F8000。

20.6.5 窗口寄存器(WDG_WIN)

表 20-6 WDG_WIN 寄存器

WDG_WIN 窗口值寄存器 **Reset: 0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	WIN[31: 16]															
访问	RW															
Reset	0															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	WIN[15: 0]															
访问	RW															
Reset	0															

字段	说明
31: 0 WIN	**看门狗窗口寄存器** 窗口模式使能后（WDG_CS2[WIN]置位），WIN 值 决定看门狗允许刷新的最早时间，早于窗口值刷新会引起系统复位。

21 实时计数器模块（Real Time Clock）

21.1 简介

实时计数器模块（RTC），主要功能是实时计数。在 Standby 低功耗模式下，RTC 具有保持运行并唤醒 MCU 的功能。

21.2 特性

- 32 位向上计数器
- 可编程 20 位预分频器
- 计数溢出翻转 GPIO

21.3 结构框图

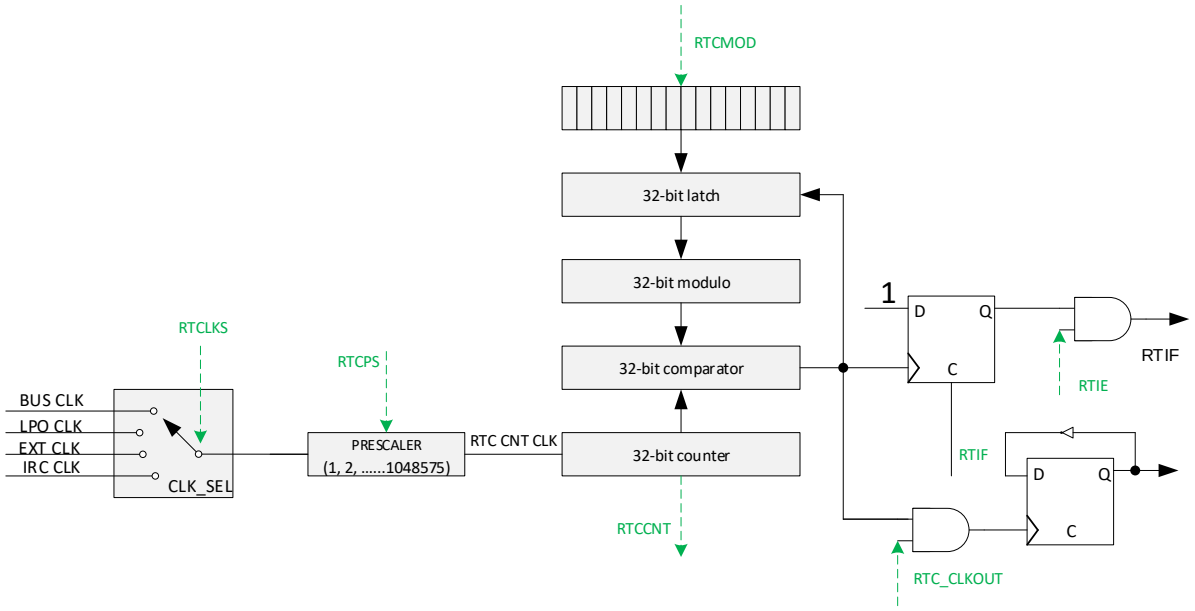

图 21-1 RTC 结构框图

21.4 功能描述

21.4.1 时钟源选择

RTC 模块有四种时钟源可选择使用：总线时钟、内部 32kHz RC 振荡器时钟、外部 XOSC 时钟以及外部 RTCIN 引脚输入时钟。

21.4.2 计时特性

RTC 是一个向上增加的计数器，计数达到预设的模值后，产生 RTC 溢出标志。如果使能了 RTC 溢出中断，则产生一次 RTC 中断请求，溢出后 RTC 计数器将从 0 开始新的计数。RTC 同时内置一个向上计数的预分频器，当达到预设的预分频值后，产生预分频器溢出标志，如果使能了预分频器中断，则产生一次预分频器中断请求，预分频器计数溢出后将从 0 开始新的计数。

21.4.3 RTC 计时信号输出

PA13 可以复用为 RTC_CLKOUT 功能，复用后，RTC 计数溢出能翻转 PA13。

21.4.4 低功耗唤醒

MCU 在低功耗模式 Stop、Standby 下，RTC 可作为唤醒源唤醒 MCU，需要在进入低功耗模式前选择内部 32K 作为时钟源，并启动模块。Stop 模式下，RTC 溢出后产生 RTC 中断，唤醒 MCU。Standby 模式下，RTC 溢出后直接唤醒 MCU。从低功耗模式唤醒后，RTC 计数器会继续保持计数。

21.5 应用说明

21.5.1 RTC 基本使用

使用前，配置 RTC_SC 寄存器，初始化 RTC 模块的时钟、模值、中断等，在最后配置预分频器为非 0 值时， RTC 计数器与预分频计数器开始计数。

RTC 预分频计数器溢出时，标志位 RPIF 置位，写 1 清除标志。RTC 计数器溢出时，标志位 RTIF 置位，写 1 清除标志。在计数器运行时，对 RTC_CLK 或 RTC_PS 的更改将清 0 计数器。

RTCO 位为 1，使能 RTC 计数器溢出翻转指定 GPIO，GPIO 同时需要复用为 RTC_CLKOUT 功能。

21.5.2 RTC 低功耗唤醒

使用 RTC 唤醒低功耗状态 Stop 或 Standby 下的 MCU，需要先在 SPM 模块中使能 RTC 唤醒，同时选择内部 32K 时钟作为 RTC 时钟源并启动计时。

21.6 寄存器定义

表 21-1 RTC 寄存器映射

RTC 基地址：0x40008400

地址	名称	宽度	描述
RTC 基地址+0x00	RTC_SC	32	状态与控制寄存器
RTC 基地址+0x04	RTC_MOD	32	模值寄存器
RTC 基地址+0x08	RTC_CNT	32	计数器寄存器
RTC 基地址+0x0C	RTC_PS	32	预分频器寄存器
RTC 基地址+0x10	RTC_PSCNT	32	预分频器计数器寄存器

21.6.1 控制与状态寄存器(RTC_SC)

表 21-2 RTC_SC 寄存器

RTC_SC 控制与状态寄存器 **Reset: : 0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称															RPIF	RPIE
访问															R	RW
Reset															0	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	RTCLK								RTIF	RTIE		RTCO				
访问	RW								R	RW		RW				
Reset	0								0	0		0				

字段	说明
17 RPIF	**实时预分频器中断标志** 0： RTC 预分频器计数器没有达到 RTC 预分频器寄存器的值 1： RTC 预分频器计数器达到 RTC 预分频器寄存器的值 该状态位表示 RTC 预分频器计数器达到了 RTC 预分频寄存器中的值。写入逻辑 0 无效。写入逻辑 1 会清除此位和实时中断请求。复位将 RPIF 清零。
16 RPIE	**实时预分频器中断使能** 0： 禁用实时预分频器中断请求 1 ： 使能实时预分频器中断请求 使能实时预分频器中断。如果 RPIE 置位，当 RPIF 置位时产生中断请求。
15：14 RTCLKS	**实时时钟源选择** 00 ： 总线时钟 01： 内部 32kHz RC 振荡器

字段	说明
	10： 外部 XOSC 11： 外部 RTCIN 引脚输入时钟 RTC 模块时钟源选择位。切换时钟源会清零预分频器与 RTC 计数器。 复位清除 RTCLKS 为 0。RTC 溢出时间为(((MOD +1) * (RTCPS+1)) / RTCLKS)
7 RTIF	**实时中断标志** 0： RTC 计数器没有达到 RTC 模数寄存器的值 1： RTC 计数器达到 RTC 模数寄存器的值 该状态位表示 RTC 计数器达到了 RTC 模数寄存器中的值。写入逻辑 0 无效。写入逻辑 1 会清除该位和实时中断请求。
6 RTIE	**实时中断使能** 0： 禁用实时中断请求 1：使能实时中断请求 使能实时中断。如果 RTIE 置位，则在 RTIF 置位时会产生中断请求。
4 RTCO	**实时计数器输出** 0： 实时计数器输出禁用. 1： 实时计数器输出使能 使能在 RTC 计数溢出时指定引脚输出电平信号翻转。如果该位置 1，RTC 计数器溢出时将 RTC_CLKOUT 引脚电平翻转。

21.6.2 模值寄存器(RTC_MOD)

表 21-3 RTC_MOD 寄存器

RTC_MOD **模值寄存器** **Reset: : 0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	MOD[31：16]															
访问	RW															
Reset	0															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	MOD[15：0]															
访问	RW															
Reset	0															

字段	说明
31：0 MOD	**RTC 模值** 用于与当前计数值(RTC_CNT)进行比较的模数值，计数值等于模数时计数将重置为 0x0，置位 SC[RTIF]。

21.6.3 计数器寄存器(RTC_CNT)

表 21-4 RTC_CNT 寄存器

RTC_CNT **计数器寄存器** **Reset: 0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	CNT[31：16]															
访问	R															
Reset	0															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	CNT[15：0]															
访问	R															
Reset	0															

字段	说明
31：0 CNT	**RTC 计数器** 指示 RTC 计数器当前计数值。对此寄存器写无效，将不同的值写入 SC [RTCLKS]或 PS[RTCPS]重新开始计数

21.6.4 预分频器寄存器(RTC_PS)

表 21-5 RTC_PS 寄存器

RTC_PS **预分频器** **Reset: 0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称													RTCPS[19：16]			
访问													RW			
Reset													0			
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	RTCPS[15：0]															
访问	RW															
Reset	0															

字段	说明
19：0 RTCPS	**RTC 预分频器** 更改预分频器值会清除预分频器计数器和 RTC CNT 计数器。 RTCPS 为 0 时，RTC 计数器停止计数，RTCPS 为非 0 时，开始计数。

21.6.5 预分频器计数寄存器(RTC_PSCNT)

表 21-6 RTC_PSCNT 寄存器

RTC_PSCNT **预分频器计数器** **Reset: 0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称													PSCNT[19：16]			
访问													RW			
Reset													0			
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	PSCNT[15：0]															
访问	RW															
Reset	0															

字段	说明
19：0 PSCNT	**RTC 预分频器计数器** 指示当前预分频计数器的值。写入对此寄存器无效，复位或将不同的值写入SC[RTCLKS]和 RTCPS 将计数清除为0x0。

22 片内 Flash（Embedded Flash）

22.1 简介

片内 Flash 控制器是 Cortex™-M0+和片内 Flash 之间的桥梁，在实际应用中，用户代码存放于片内 Flash 中，片内 Flash 启动作为主要启动模式。

片内 Flash 以下简称 eflash。

22.2 特性

- eflash 存储器
  - 最大支持 128K 字节
  - 寿命： ≥ 10000 周期
  - 页容量： 每页 2048 字节
- eflash 控制器
  - 操作列表：
    - 擦除： 页擦除，整片擦除，选项字节页擦除
    - 编程： 页编程，选项字节页编程，最小编程位宽为 32bit，编程地址需 4 字节对齐
    - 读：可以按照 8bit/16bit/32bit 位宽读数据
    - 验证：页擦除验证，整片擦除验证
  - 包含预取缓冲区

22.3 结构框图

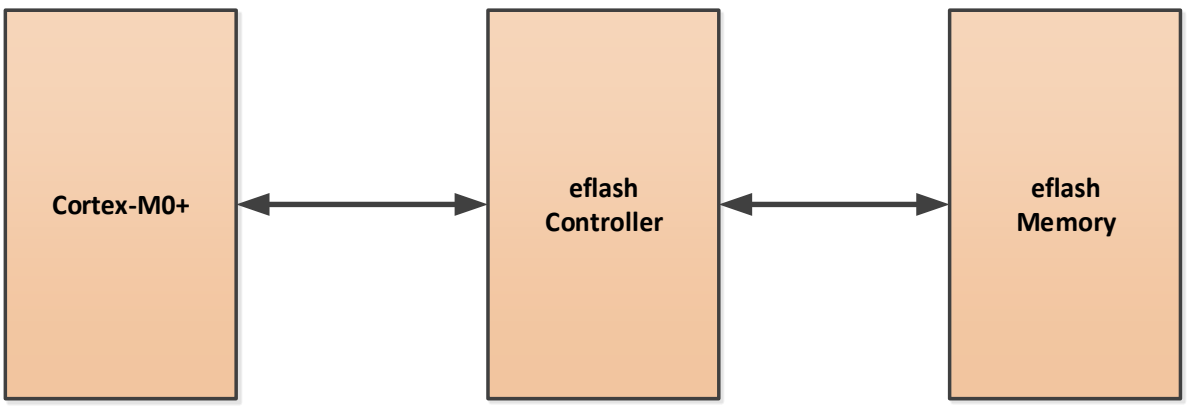

图 22-1 eflash 和 eflash 控制器结构框图

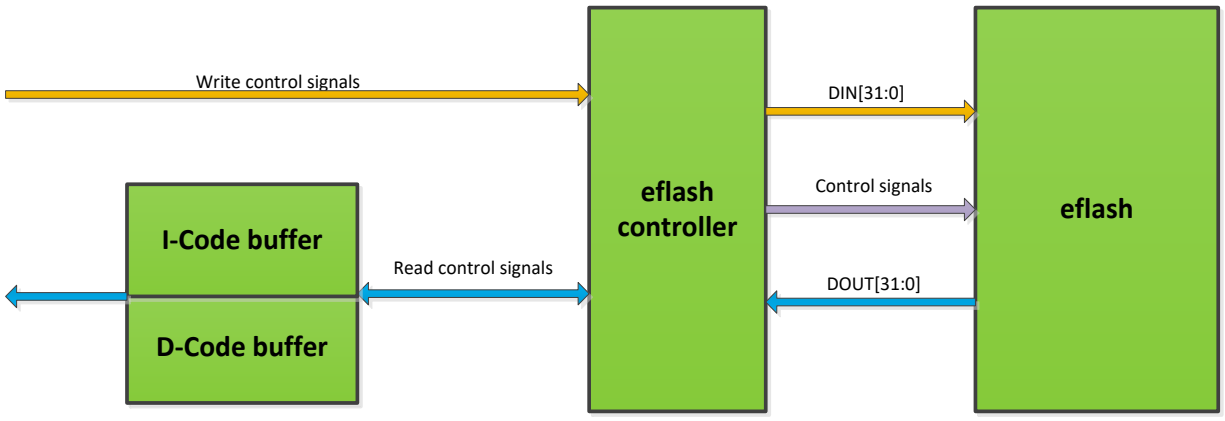

图 22-2 eflash 和 eflash 控制器数据流

22.4 功能描述

22.4.1 片内 Flash 组织

在介绍命令之前，先了解表 22-1 中的片内 Flash 组织。整个片内 Flash 由两部分组成：一部分是主存储器，另一部分是信息存储器。如表 22-1 所示，每个 2K 字节在片内 Flash 中称为一页。

- 主存储器用于存储用户代码和数据，用户代码可以对主存储器进行擦除、编程、验证和读取命令操作。
- 信息存储器可分为 ISP Firmware 和选项字节两部分。
  - ISP Firmware 部分用于固化 ISP 升级的代码，用户无法擦除和编程。
  - 选项字节（Option byte）部分中的前 24 个字节是主存储器的写和读保护信息，第 41 字节到第 48 字节总共 8 个字节用于存放用户特殊的数据。对于选项字节部分，用户可以擦除、编程、验证和读取。

表 22-1 片内 Flash 存储器组织

片内 Flash 存储器	名称	地址	尺寸 (字节)	用户权限 (说明)
主存储器 (用户页)	Page 0	0x0800 0000 ~ 0x0800 07FF	2K	擦除/编程/读取/验证
	Page 1	0x0800 0800 ~ 0x0800 0FFF	2K	
	Page 2	0x0800 1000 ~ 0x0800 17FF	2K	
	Page 3	0x0800 1800 ~ 0x0800 1FFF	2K	
	…	…	2K	
	Page 63	0x0801 F800 ~ 0x0801 FFFF	2K	
信息存储器	Option byte	0x0804 0000 ~ 0x0804 002F	48	
		0x0804 0030 ~ 0x0804 07FF (Reserved)	2000	
	ISP Firmware	0x0804 0800 ~ 0x0804 1fff	6K	读

22.4.2 片内 Flash 保护

在选项字节页中保存的主要内容有读保护，写保护，看门狗默认工作状态等。为了避免非法访问 eflash，控制器对主存储器的写入和读取具有保护功能。相关信息存储在以下选项字节中，其中写保护信息也被加载到 EFLASH_WPRT_EN0 ~ EFLASH_WPRT_EN1 寄存器。修改选项字节中内容后，需要复位或重新上电后才生效，其中补码部分(如 nRDP / nWDGEN / nWPRT_EN / nDATAx)由硬件自动实现。

当写保护生效后，不支持对有写保护页的 eflash 区域执行擦除(页擦除/整块擦除)以及编程操作，但不影响正常读取 eflash 数据。

当失能写保护生效后，对应的 eflash 区域就可以正确擦除和编程。

当读保护生效后，无法通过 JTAG/SWD 方式正确读取 eflash 主存储区数据，当失能读保护生效后 eflash 主存储区的数据将被擦除。

使能和失能读写保护都是通过编程选项字节地址且都是需要复位后生效，详细设置参考下文所述。

表 22-2 选项字节内容列表

地址	[31：24]	[23：16]	[15：8]	[7：0]	默认值	注释
0x0804 0000	0xFF	nRDP	0xFF	RDP	0xFF53FFAC	读保护默认状态
0x0804 0004	0xFF	nWDGEN	0xFF	WDGEN	0xFFFFFFFF	看门狗默认状态
0x0804 0008	nWPRT_EN[15：0]		WPRT_EN[15：0]		0xFFFFFFFF	page 15 ~ 0
0x0804 000C	nWPRT_EN[31：16]		WPRT_EN[31：16]		0xFFFFFFFF	page 31 ~ 16
0x0804 0010	nWPRT_EN[47：32]		WPRT_EN[47：32]		0xFFFFFFFF	page 47 ~ 32
0x0804 0014	nWPRT_EN[63：48]		WPRT_EN[63：48]		0xFFFFFFFF	page 63 ~ 48
0x0804 0028	nDATA0		DATA0		0xFFFFFFFF	用户数据
0x0804 002C	nDATA1		DATA1		0xFFFFFFFF	用户数据

22.4.2.1 读写保护

参考表 22-2 可知：主存储器读保护所在的选项字节地址为 0x0804 0000 ~ 0x0804 0003，写保护所在的选项字节地址为 0x0804 0008 ~ 0x0804 0017。读写保护设置如表 22-3 和表 22-4 所示。

表 22-3 读保护设置

条件	RDP	nRDP	读保护状态
Case1	0xFF	0xFF	受保护
Case2	0xAC	0x53	不受保护
其他 case	除了 case1 和 case2 以外任意值		受保护

表 22-4 写保护设置

条件	WPRT_EN[x]	nWPRT_EN[x]	写保护状态
Case1	0	1	受保护
其他 case	除了 case1 以外任意值		不受保护

说明

在写保护值中，一个比特位对应一页。

22.4.2.2 看门狗

参考表 22-2 可知：看门狗默认状态所在的选项字节地址为 0x0804 0004 ~ 0x0804 0007。如果 WDGEN 编程为 0xCC，nWDGEN 编程为 0x33 则看门狗默认为使能状态。否则看门狗默认为禁用状态。

表 22-5 看门狗默认状态设置

条件	WDGEN	nWDGEN	状态
case1	0xCC	0x33	默认为使能状态
其他	除了以上任意值		默认为禁用状态

22.4.2.3 用户数据

参考表 22-2 可知：用户数据 DATAx/nDATAx 选项字节地址为 0x0804 0028 ~ 0x0804 002F。其中低 16 位 DATAx 用于用户自由存储数据，高 16 位 nDATAx 为用户数据补码，由硬件自动计算。

22.5 应用说明

在本节中，将通过流程图介绍页擦除，整片擦除，页编程，选项字节擦除，选项字节编程，页擦除验证和整片擦除验证。所有命令操作主要涉及以下寄存器：EFLASH_CTRL0，EFLASH_CTRL1，EFLASH_ADR_CMD，EFLASH_SR0。对于读操作，用户可以按照 8 bit/16 bit/32 bit 访问方式直接读取 eflash 存储所需地址，因此本文档中将省略相关描述。

以下流程图仅说明了单个命令操作。在多个命令操作之前只需要进行一次解锁。对于片内 Flash 存储器，共有 68 页，由 1 个选项字节页、3 个 ISP Firmware 页和 64 个用户页组成。

以下重点介绍页擦除，页擦除验证和页编程流程，其他命令操作可以参考这些流程。

22.5.1 页擦除

页擦除操作仅作用于 eflash 中的主存储器，页擦除操作不能用于信息存储器。以下详细描述页擦除流程，其他命令操作可参考。

1. 在配置 EFLASH_CTRL0 和 EFLASH_CTRL1 之前，必须通过读取 LOCK 的状态来检查控制器是否被锁定。如果控制器处于锁定状态，必须顺序写入 0xac7811 和 0x01234567 至 EFlASH_KEY，然后才能解锁。 如果控制器不处于锁定状态，可以直接进入下一步；
2. 解锁后，先通过读出 CMD_BSY 的状态来检查是否有正在进行的命令操作。CMD_BSY 等于 1 表示某些命令操作未完成，必须等到 CMD_BSY 等于 0。事实上，解锁过程可以在检查 CMD_BSY 之前或之后完成，下面的图表只是说明了“之前”的情况；
3. 当 CMD_BSY 变为 0 时，可以配置如下两个寄存器：EFLASH_CTRL0 和 EFLASH_CTRL1。对于所有擦除和编程命令操作，必须按比例调整 EFLASH_CTRL1 中 CKDIV 值，调整公式为 CKDIV= eflash 控制器时钟频率/1,比如 eflash 控制器时钟频率为 48MHz, CKDIV= 48/1 = 48= 0x30, 但是建议配置的这个值最好是比 0x30 大一点，比如 0x31；
4. 在 EFLASH_ADR_CMD 中配置页擦除起始地址，该值是所需擦除页的起始地址；
5. 通过控制 EFLASH_CTRL0 来启动命令并触发它。应保证在 EFLASH_CTRL0 中配置其他位后，CMD_ST 位必须从 0 变为 1 才能获得有效触发。EOPIE 配置为 1，使能命令完成时状态发生变化。CMD_CTL 配置为 0x1 以确定传入命令操作是页擦除。 其他位应为 0，并将用于其他命令操作；
6. 在有效触发之后，命令操作开始。应通过读出 CMD_BSY 和 EOP 位来检查命令操作是否完成。当 CMD_BSY 等于 0 且 EOP 等于 1 时，表示命令操作已完成，通过向该位写 0 来清除 EOP。实际上，只需使用 CMD_BSY 来指示操作是否已完成所有命令操作；
7. 清除 EFLASH_CTRL0；
8. 读出状态以检查是否存在某些错误，例如 OPR_ERR，特别是对于写保护情况，用户无法擦除已经被写保护的页。

页擦除流程如下图所示。其他命令操作的过程类似于页擦除，区别的地方将在后续章节进行描述。

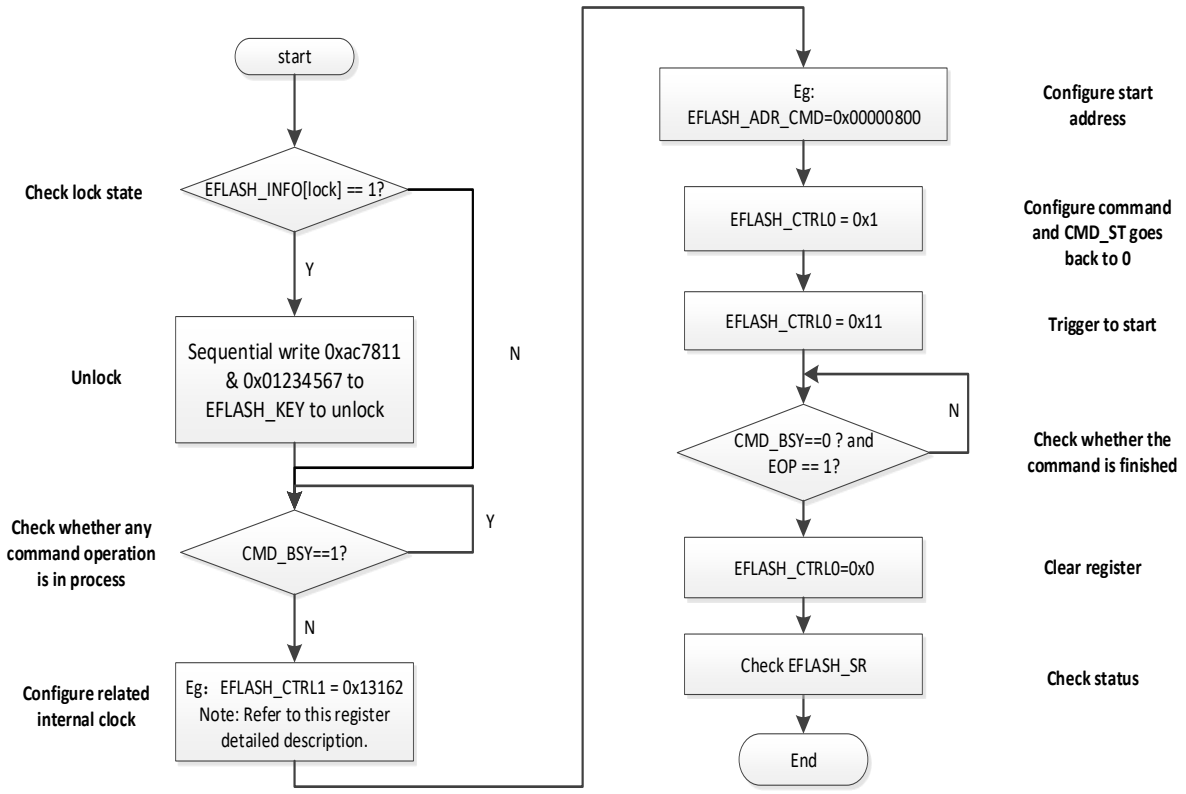

图 22-3 页擦除命令操作流程

22.5.2 整片擦除

整片擦除可以擦除整个用户 64 页 eflash，流程如图 22-4 所示。与页擦除命令流程相比，整片擦除流程有两个不同：一是 EFLASH_CTRL0 寄存器 CMD_CTL 位设置值不同，二是不需要在 EFLASH_ADR_CMD 寄存器中指定擦除地址。

特别地，当主存储器属性从读保护改变为非读保护时，将自动执行整片擦除以保护用户代码不被非法读取。例如，当用户将选项字节中的 RDP 从默认的 0xFF 编程为 0xAC 时，整片擦除会自动执行。

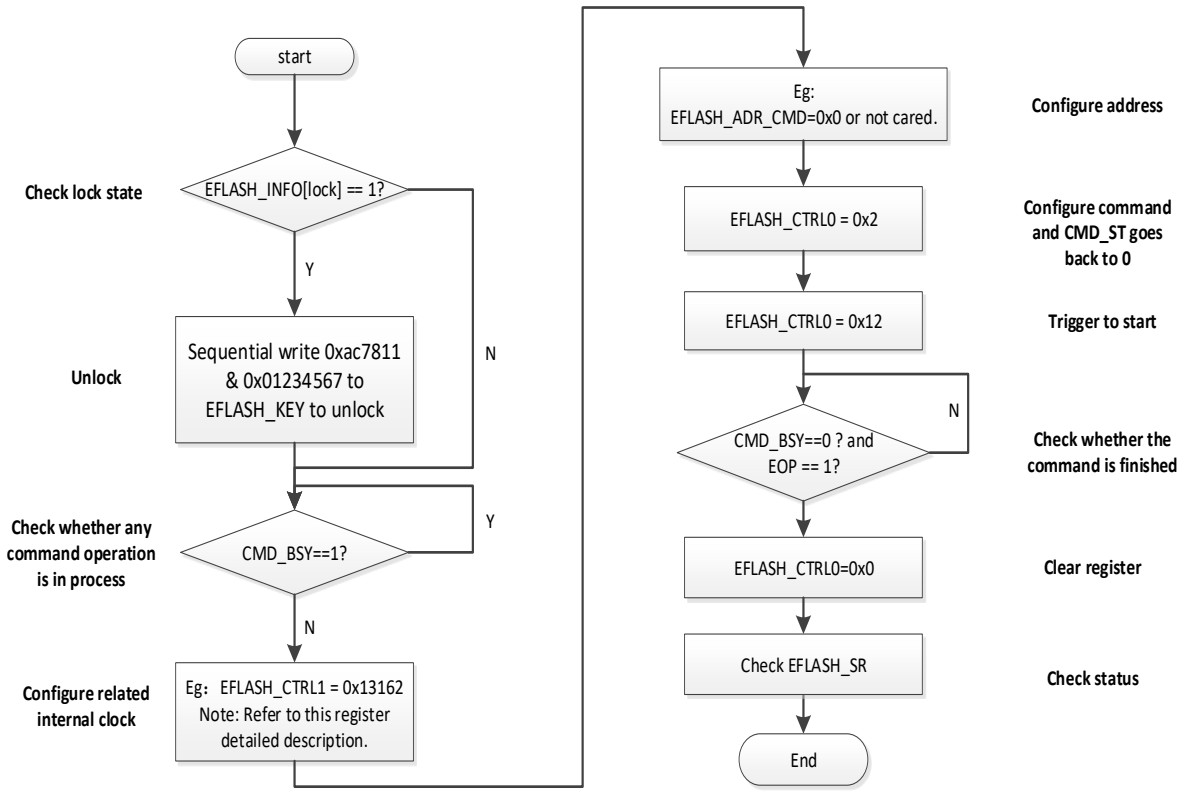

图 22-4 整片擦除命令操作流程

22.5.3 页编程

页编程命令可用在整个用户 64 页 eflash，流程如图 22-5 所示。页编程命令流程也类似于页擦除，但是有两个不同之处：一个是 EFLASH_CTRL0 寄存器 CMD_CTL 位设置值不同，另一个是需配置 EFLASH_CTRL0 寄存器 PROG_LENGTH [9：0]位值来配置页编程命令写入的字长度。PROG_LENGTH [9：0]位配置为指定值，以字为单位来指定编程长度。例如，如果 PROG_LENGTH [9：0]配置为 150，则用户应写入不超过 150 个字。如果用户写入超过 150 字，例如 180 字，则最后 30 字实际上不会写入 eflash。同时，如果用户写入少于 150 字，如 110 字，则写操作将会验证正常，但编程过程还不会结束，直到数据数目等于长度或使用 flush 命令强制编程终止。

关于页编程命令，有一些自动预检机制可以进行内容保护，如写保护检查，空白内容检查和地址边界检查等。因此，用户应在每个程序命令后仔细检查 EFLASH_SR0 寄存器，以确认写操作是否成功。

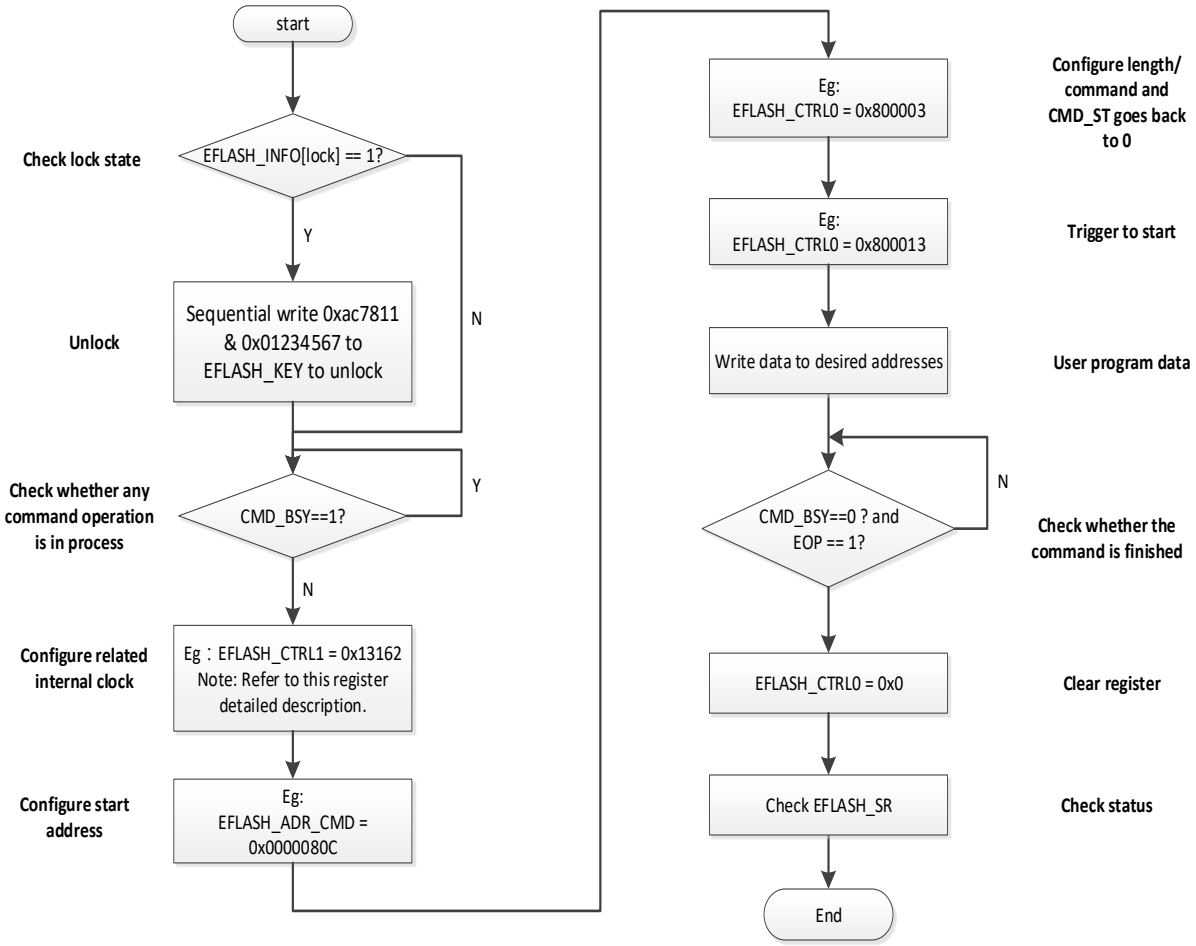

图 22-5 页编程命令操作流程

22.5.4 页擦除验证

页擦除验证命令用于整个用户 64 页 eflash。该操作通常在擦除操作之后执行，以验证擦除操作是否成功执行，流程如图 22-6 所示。与页擦除命令流程相比，页擦除验证流程只有一个区别：EFLASH_CTRL0 寄存器 CMD_CTL 位设置值不同。

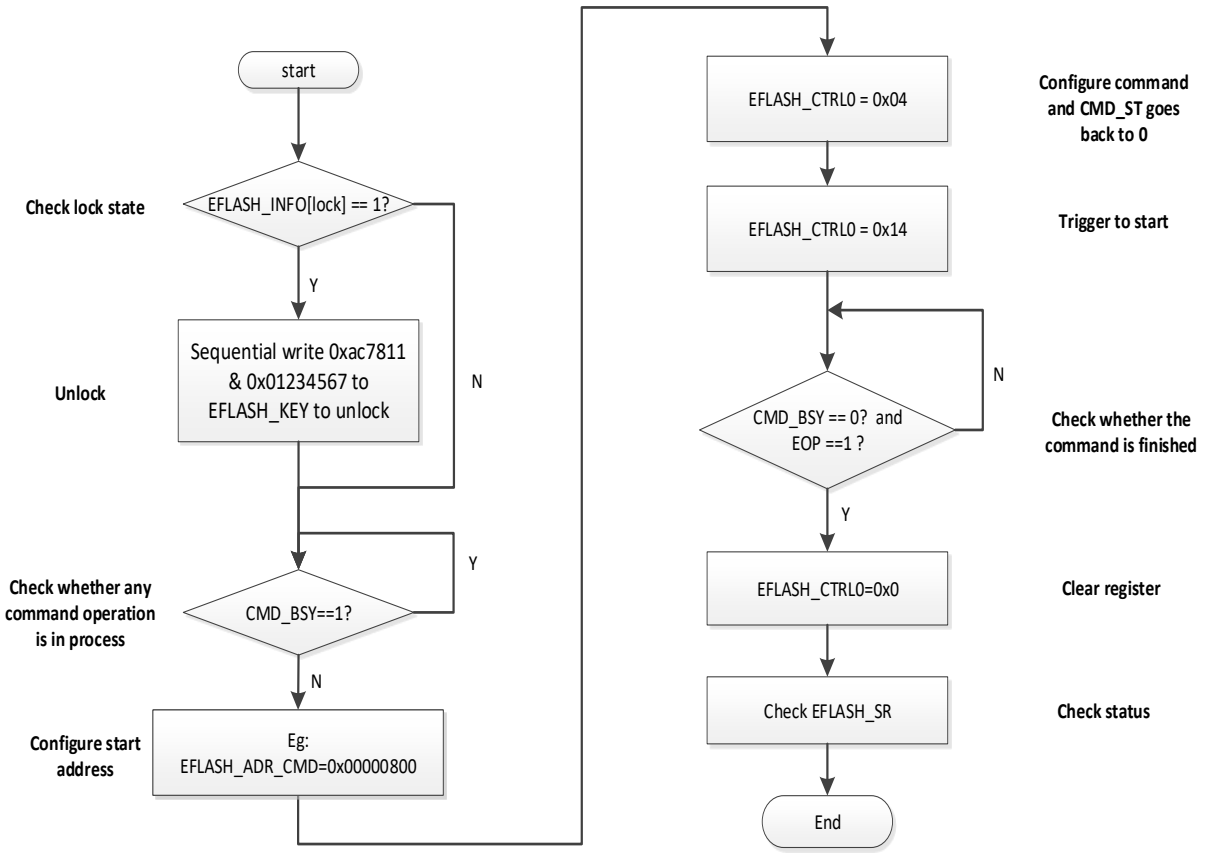

图 22-6 页擦除验证命令操作流程

22.5.5 整片擦除验证

整片擦除验证命令可用于整个用户 64 页 eflash。该操作通常在整片擦除操作之后执行，以便验证整片擦除操作是否成功完成，流程如图 22-7 所示。与页擦除命令流程相比，整片擦除验证流程有两个不同：一是 EFLASH_CTRL0 寄存器 CMD_CTL 位设置值不同，二是不需在 EFLASH_ADR_CMD 寄存器指定擦除地址。由于达到了整个 64 页的存储，因此无需为此命令操作指定 EFLASH_ADR_CMD。

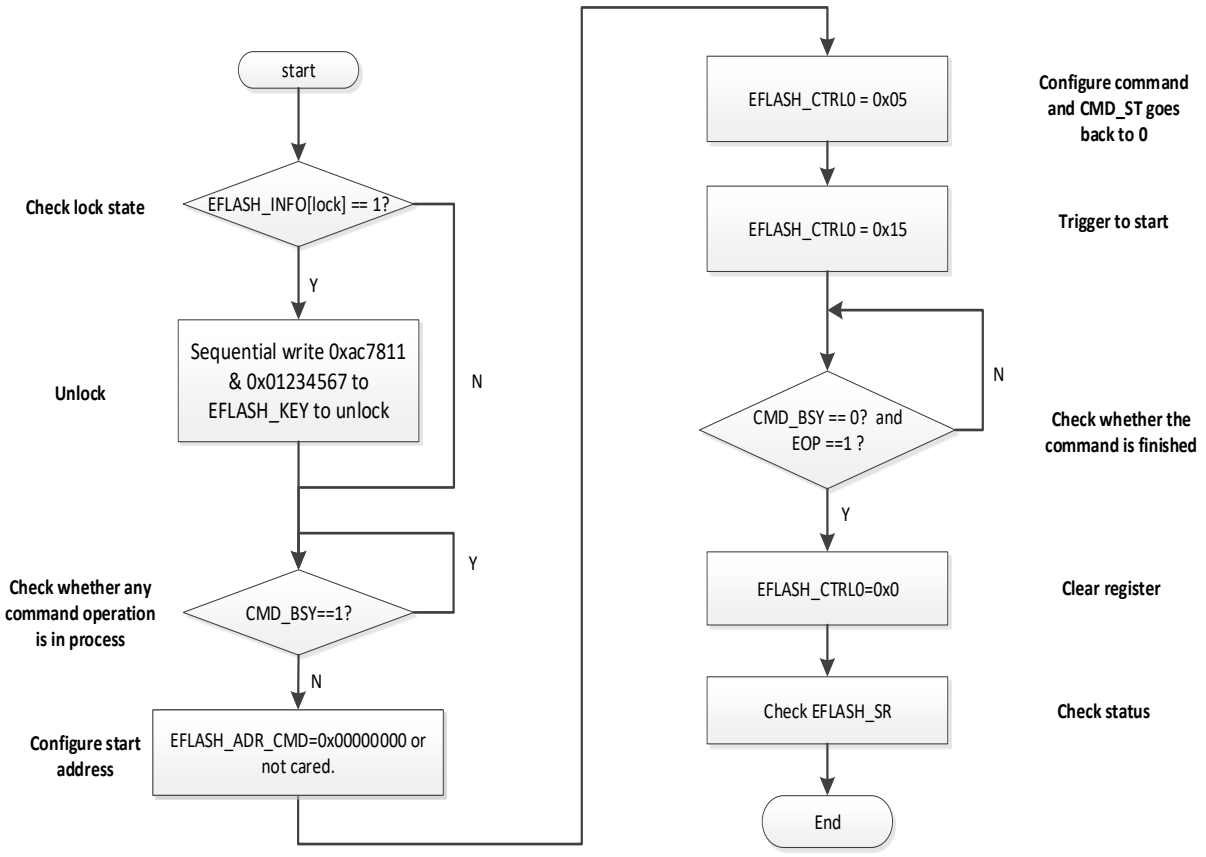

图 22-7 整片擦除验证命令操作流程

22.5.6 选项字节擦除

选项字节擦除命令用于整个选项字节，流程如图 22-8 所示。与页擦除命令流程相比，选项字节擦除流程有两点区别：一是 EFLASH_CTRL0 寄存器 CMD_CTL 位设置值不同，二是在 EFLASH_ADR_CMD 寄存器中指定地址为选项字节地址。

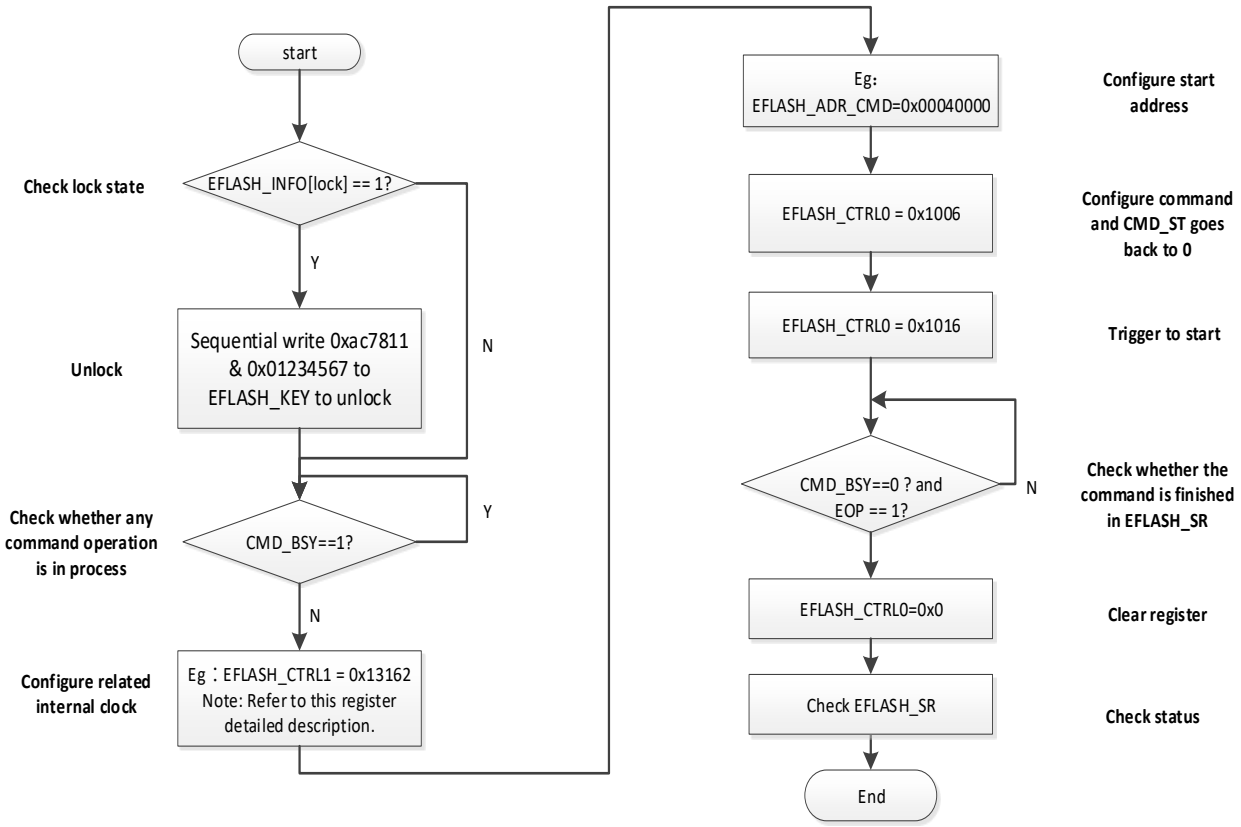

图 22-8 选项字节擦除命令操作流程

22.5.7 选项字节编程

选项字节擦除命令用于整个选项字节，流程如图 22-9 所示。与页编程命令流程相比，选项字节编程流程有两点区别：一是 EFLASH_CTRL0 寄存器 CMD_CTL 位设置值不同，二是在 EFLASH_ADR_CMD 寄存器中指定地址为选项字节地址。

在进行选项字节编程时，用户应高度重视读取保护字符的更改，如果将读取保护字符从保护更改为不保护，eflash 控制器将自动执行整片量擦除命令，此时将会擦除整个用户页的内容。

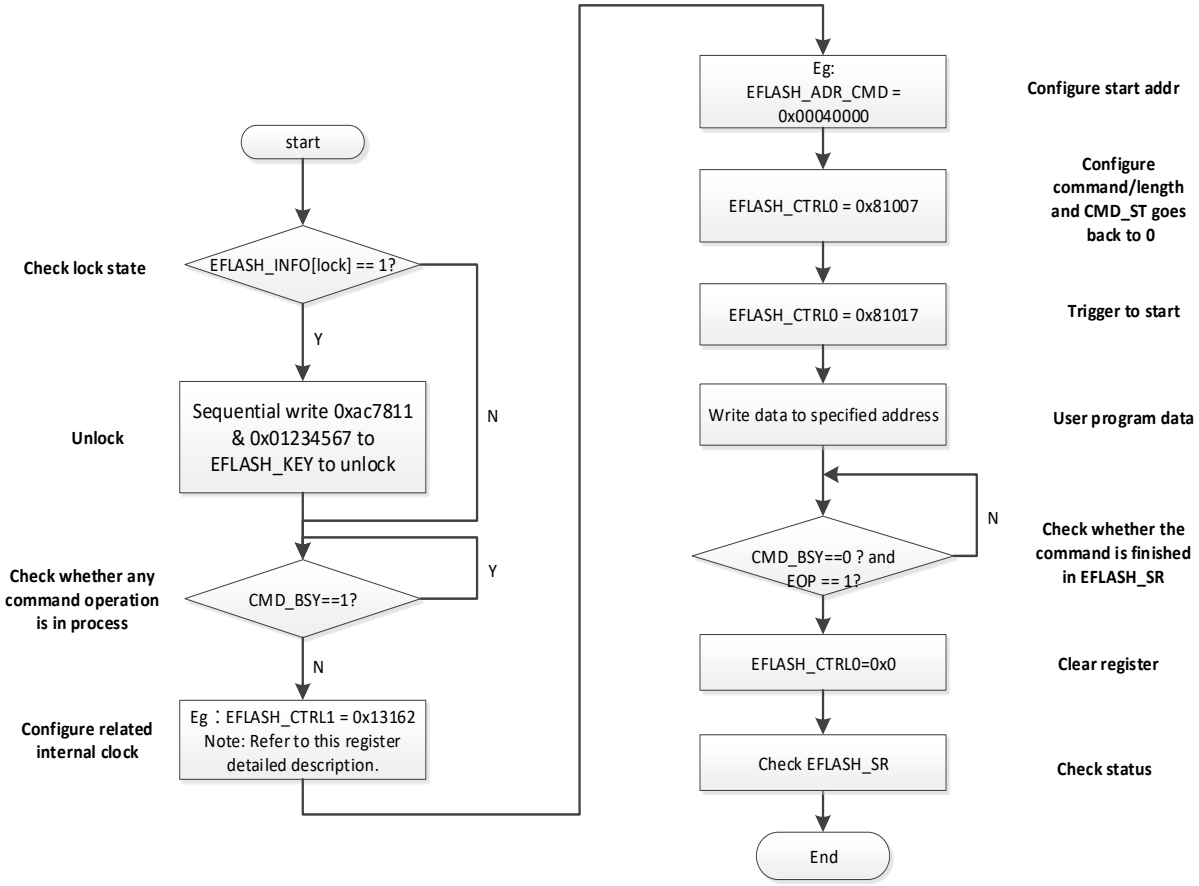

图 22-9 选项字节编程命令操作流程

22.6 寄存器定义

表 22-6 片内 Flash 寄存器映射

EFLASH 基地址: 0x40002000

地址	名称	宽度	描述
EFLASH 基地址+0x0	EFLASH_KEY	32	解锁序列寄存器
EFLASH 基地址+0x4	EFLASH_INFO	32	保护信息寄存器
EFLASH 基地址+0x8	EFLASH_ADR_CMD	32	擦除起始地址
EFLASH 基地址+0xC	EFLASH_CTRL0	32	控制寄存器 0
EFLASH 基地址+0x10	EFLASH_SR0	32	状态寄存器
EFLASH 基地址+0x14	EFLASH_CTRL1	32	控制寄存器 1：时钟设置
EFLASH 基地址+0x18 ~ EFLASH 基地址+0x1C	EFLASH_WPRT_ENx	32	写保护使能位[63：0]
EFLASH 基地址+0x40	EFLASH_CTRL2	32	控制寄存器 2

22.6.1 Key 序列寄存器(EFLASH_KEY)

表 22-7 EFLASH_KEY 寄存器

EFLASH_KEY **Key 序列寄存器** **RESET：0x00000000**

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

字段	说明
31：0 KEY	**解锁 eflash 控制寄存器的写保护** 当 EFLASH_INFO[LOCK]位为 1 时，表示片内 Flash 控制器寄存器被锁定时，顺序写入 0xac7811 和 0x01234567 以解锁。

22.6.2 保护信息寄存器(EFLASH_INFO)

表 22-8 EFLASH_INFO 寄存器

EFLASH_INFO **保护信息寄存器** **RESET：当前保护状态**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	LOCK															
访问	RW															
Reset	1															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称															WDG_EN	RD_PRT
访问															R	R
Reset																

字段	说明
31 LOCK	**片内 Flash 控制器锁定寄存器** 0: 表示片内 Flash 控制器寄存器未锁定，此时直接写 1 会锁定片内 Flash 控制器 1: 表示片内 Flash 控制器寄存器锁定，不能直接写 0 来解锁，只能通过操作寄存器 EFLASH_KEY 来解锁

字段	说明
1 WDG_EN	**选项字节中 WDG 使能状态** 0: WDG 失能 1: WDG 使能 **注意：该位复位值取决于选项字节中看门狗使能状态。**
0 RD_PRT	**片内 Flash 读保护状态** 0: 读保护失能 1: 读保护使能 **注意：该位复位值取决于当前读保护使能状态。**

22.6.3 擦除/编程起始地址(EFLASH_ADR_CMD)

表 22-9 EFLASH_ADR_CMD 寄存器

EFLASH_ADR_CMD **擦除/编程起始地址** **RESET：0x00000000**

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

字段	说明
31：0 ADR_CMD	**起始地址命令** (1) 页擦除 ADR_CMD[19：0]是 eflash 地址的低 20 位，就在这个页中。例如，如果用户想要擦除从 0x0801 F000 到 0x0801 F7FF 的 63 页，那么，用户可以将 ADR_CMD [31：0]配置为从 0x0001 F000 到 0x0001 F7FF 的任何值。 (2) 编程 ADR_CMD[19：0]为编程起始地址，为 eflash 地址的低 20 位。 ADR_CMD[19：0]和 PROG_LENGTH[9：0]共同确定可编程地址范围。 (3) 页验证 ADR_CMD[19：0] 是该页第一个 eflash 地址的低 20 位。 (4) 整片擦除/验证 ADR_CMD[31：0] 未关注。

字段	说明
	注意：在使用擦除/编程/验证命令时，必须配置起始地址。ADR_CMD[31：20] 必须为 12'b0。

22.6.4 控制寄存器 0(EFLASH_CTRL0)

表 22-10 EFLASH_ CTRL0 寄存器

EFLASH_CTRL0 控制寄存器 0 RESET：0x00000000

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	HDFEN						PROG_LENGTH									
访问	RW						RW									
Reset	0						0									
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称				OPT_CMD_EN								CMD_ST	CMD_CTL			
访问				RW								RW	RW			
Reset				0								0	0			

字段	说明
31 HDFEN	**违反读写保护规则时是否产生 HardFault 中断** 0： 禁用，当违反读写保护规则时候不会产生 HardFault 中断 1： 使能，当违反读写保护规则时候会产生 HardFault 中断
25：16 PROG_LENGTH	**编程操作长度** 编程长度. **注意：单位是字。**
12 OPT_CMD_EN	**使能选项字节区域相关的命令操作** 0：寄存器 CMD_CTL 取值 0110 或 0111 无效，取值 0001、 0010 或 0011 有效 1：寄存器 CMD_CTL 取值 0110 或 0111 有效，取值 0001、0010 或 0011 无效 **注意：参考 CMD_CTL。**
4 CMD_ST	**控制起始命令操作** 写 1 以触发起始命令且会清除 EFLASH_SR0 寄存器的错误状态位 Bit[11：2]。
3：0 CMD_CTL	**命令** 0000： 空闲 0001： 页擦除 0010： 整片擦除 0011： 页编程 0100： 页擦除验证 0101： 整片擦除验证 0110： 选项字节擦除 0111： 选项字节编程

字段	说明
	注意：CMD_CTL 取值的有效性请参考 OPT_CMD_EN 的配置。其中当 CMD_CTL 取值 0100 或 0101 时与 OPT_CMD_EN 的配置无关。

22.6.5 状态寄存器(EFLASH_SR0)

表 22-11 EFLASH_SR0 寄存器

EFLASH_SR0 状态寄存器 RESET： 0x00800002

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称				FLUSH	OPTER				VRER	ERAER	PPADRER	PPERER	RDER	WRER	EOP	CMD_BSY
访问				RW	R				R	R	R	R	W1C	W1C	W1C	R
Reset				0	0				0	0	0	0	0	0	1	0

说明

EFLASH_ CTRL0[CMD_ST]位写 1 会清除 EFLASH_SR0 寄存器的错误状态位 Bit[11：2]。

字段	说明
12 FLUSH	写 1 以完成程序命令操作 注意：当用户按照本应用说明执行程序命令操作时，不会使用该位。但在某些情况下，剩余内 存和 PROG_LENGTH[9：0] 的值不匹配，或实际程序内存数小于 PROG_LENGTH[9：0]。因此， 程序无法完成。对于这种情况，将 1 写入 FLUSH 将强制完成程序命令操作。.
11：8 OPTER	表示选项字节的错误状态 OPTER [3]： 0：没有错误，即： DATAx=~nDATAx 1：存在错误，即 DATAx != ~nDATAx, 除了 DATAx = nDATAx=0xff OPTER [2]： 0： 没有错误, 即 WPRT_EN[x] = ~nWPRT_EN[x] 1：存在错误，即 WPRT_EN[x] != ~nWPRT_EN[x], 除了 WPRT_EN[x] = ~nWPRT_EN[x] = 0x1 OPTER [1]： 0：没有错误, 即 WDGEN = ~nWDGEN

字段	说明
	1：存在错误，即 WDGEN != ~nWDGEN, 除了 WDGEN = nWDGEN=0xff OPTER [0]： 0：没有错误, 即 RDP = ~nRDP 1：存在错误，即 RDP != ~nRDP, 除了 RDP = nRDP=0xff
7 VRER	**表示验证命令操作中是否存在错误** 0：没有错误 1： 数据验证存在问题
6 ERAER	**表示擦除命令操作中是否存在错误** 0：没有错误 1： 存在错误，因为 EFLASH_ADR_CMD 中的地址是非法的
5 PPADRER	**表示页编程命令操作中是否存在错误** 0： 没有错误 1： 存在错误，因为程序地址非法或在程序命令操作不正常之前验证操作
4 PPERER	**表示命令操作的权限错误** 0： 没有错误 1：当页/整片擦除或编程作用在受保护的主存储器时，发生错误
3 RDER	**表示操作违反了读保护规则** 0：不违反读保护规则 1：违反读保护规则 写 1 将 RDER 清除为 0.
2 WRER	**表示操作违反了写保护规则** 0：不违反写保护规则 1：违反写保护规则 写 1 将 WRER 清除为 0.
1 EOP	**表示命令操作是否完成** 0： 没有完成 1： 已完成 **注意： 该位对用户不重要。** 写 1 将 EOP 清除为 0。
0 CMD_BSY	**表示是否正在执行任何命令操作** 0：所有操作都没有进行中 1：至少有一个操作在进行中

22.6.6 控制寄存器 1(EFLASH_CTRL1)

表 22-12 EFLASH_CTRL1 寄存器

EFLASH_CTRL1	控制寄存器 1	RESET:0x00002009

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

字段	说明
13:12 SPEED_LATENCY	**FLASH 初始化基准分频值** **注意: 上电系统时钟初始化完后，必须写为 0x2。**
8 CKDIV_LOCK	**锁定对于 EFLASH_CTRL1 的配置** 0： 可以配置该寄存器 1： 不能配置该寄存器 **注意：如果在上电后该位写入 1，则在芯片断电之前， 不能再次配置 EFlASH_CTRL1 寄存器。**
6：0 CKDIV	**时钟分频用以产生 1μs 脉冲** 必须根据 eflash 控制器的速度配置合理的值才能在以下操作之前获得 1μs 周期：页编程，页擦除，基于命令操作流程的整片擦除。例如，如果 eflash 控制器时钟频率为 48MHz, CKDIV= 48/1 = 48= 0x30，但是建议配置的这个值最好是比 0x30 大一点，比如 0x31。

22.6.7 写保护使能寄存器 x (EFLASH_WPRT_ENx)

表 22-13 EFLASH_WPRT_ENx 寄存器

EFLASH_WPRT_ENx 写保护使能寄存器 x RESET:

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

说明

上表中 Reset 值会受因选项字节寄存器配置而发生变化。

注意

x=0~1，四个 EFLASH_WPRT_ENx 寄存器组成 64 个使能位，分别决定每个主存储器页的写保护属性。

字段	说明
31：0 WPRT_ENx	表示是否存在对相应主存储器页的写保护 0： 没有写保护 1： 存在写保护

22.6.8 控制寄存器 2(EFLASH_CTRL2)

表 22-14 EFLASH_CTRL2 寄存器

EFLASH_CTRL2 控制寄存器 2 **RESET: 0x00000000**

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

字段	说明
31 PVD_WRN	**编程电压检测器状态** 默认：1'b0 **注意：该位总是作为状态标记使用，如果写操作，则只能写 0。如果使能 PVD 模块且当前 IC 电压低于 PVD 模块中的编程电压，在电源电压超过编程电压前该状态位会被置位为 1。**
28 PVDMON_EN	**编程电压检测器，可以避免在擦除或者编程时候因为突然掉电造 eflash 的毁坏** 1'b1：使能监视功能 1'b0：禁止监视功能 默认：1'b0

23 SRAM 错误检测纠正（ECC_SRAM）

23.1 简介

ECC_SRAM 的全称是 SRAM Error Correcting Code，是用于 SRAM 的差错检测和修正。SRAM 出错的时候一般不会造成整个 SRAM 不能读取或是全部出错，而是整个 SRAM 中只有一个或几个 bits 出错。ECC_SRAM 采用汉明码 ECC 单 bit 纠错、两 bits 检测算法，计算速度很快，对 1 bit 以上的错误无法纠正，对 2 bits 以上的错误不保证能检测。

23.2 特性

- 支持 1 bit 和 2 bits 检错
  - 可以检出 1 bit 和 2 bits 错误状态
  - 软件可以通过寄存器读取出错的地址以及错误的状态
- 支持 1bit 错误状态纠错
  - 当检测出 1 bit 错误状态时，硬件自动纠错
  - 2 bits 错误状态硬件不可以纠错
- 支持 2bit 错误状态中断
  - 软件可以配置当 ECC_SRAM 检测到 2 bits 错误时产生中断
  - 软件可以配置在 ECC_SRAM 检测到 2 bits 错误时是否使能系统复位
- 2 bits 以上错误状态不能保证检测

23.3 功能描述

1. ECC_SRAM 默认使能且不可禁用。
2. 当使能 ERR2_IRQEN=1 且 ECC_SRAM 检测到 2 bits 错误时将会产生中断。当发生中断时，
   - 中断标志寄存器 ERR2_STATUS=1
   - ECC_SRAM 状态寄存器 ERR_STATUS=1
   - ECC_SRAM 错误地址寄存器 ERR2_ADDR 会自动填充当前产生错误的地址
   - 通过向中断标志寄存器 ERR2_STATUS 写 1 清除该标记位
3. 软件可以查询寄存器 ERR_STATUS 的值来判断当前 ECC_SRAM 错误的状态。
   - 如果 ERR_STATUS=2'b00，表明 ECC_SRAM 没有检测到错误。

- 如果 ERR_STATUS=2'b01，表明 ECC_SRAM 检测到 2 bits 错误，此时软件通过读取寄存器 ERR2_ADDR 获取当前发生 2 bits 错误的地址。
- 如果 ERR_STATUS=2'b10 或者 2b'11，表明 ECC_SRAM 检测到 1 bit 错误，此时软件通过读取寄存器 ERR1_ADDR 获取当前发生 1 bit 错误的地址。
- ECC_SRAM 错误状态寄存器 ERR_STATUS 和错误地址寄存器 ERR_ADDRx(x=1,2)在系统不复位情况下会保留最近一次 ECC_SRAM 检查到的错误状态以及错误地址，软件通过向 ERR_STATUS 写入 2'b11 清除该寄存器以及错误地址寄存器 ERR_ADDRx(x=1,2)。
- 当 RESET_CRTL(0x4000000C)寄存器的 bit 23(ecc2_rst_en)被编程为 1 且 ECC_SRAM 检测到 2 bits 错误时系统将会产生复位。

23.4 寄存器定义

表 23-1 ECC_SRAM 寄存器映射

ECC_SRAM 基地址：0x40000020

地址	名称	宽度	描述
ECC_SRAM 基地址 +0x0	ECC_SRAM_CTRL	32	ECC 控制以及状态寄存器
ECC_SRAM 基地址 +0x4	ECC_SRAM_ERR1_ADDR	32	ECC 1 bit 错误地址寄存器
ECC_SRAM 基地址 +0x8	ECC_SRAM_ERR2_ADDR	32	ECC 2 bits 错误地址寄存器

23.4.1 控制以及状态寄存器(ECC_SRAM_CTRL)

表 23-2 ECC_SRAM_CTRL 寄存器

ECC_SRAM_CTRL　　ECC_SRAM 控制以及状态寄存器　　RESET： 0x00000001

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称											ERR _STATUS			ERR2 _STATUS	ERR2 _IRQEN	
访问											W1C			W1C	RW	
Reset											0			0	0	

字段	说明
5：4 ERR_STATUS	**ECC_SRAM 错误状态**

字段	说明
	00：没有错误 01：2 bits 错误且不能纠正 10/11：1bit 错误且可以纠正 **注意：向这两位写入 2'b11 可以清除这两位以及 ERRx_ADDR(x=1,2)**
2 ERR2_STATUS	**ECC_SRAM 2 bits 错误状态** **注意：** **向该位写 1 可以清除该位** **如果使能 ERR2_IRQEN 且产生了 2 bits 错误，此时必须写 1 清除该位，否则该模块会一直产生 2 bits 错误中断**
1 ERR2_IRQEN	**ECC 2 bits 错误中断使能** 1：使能 0：禁止

23.4.2 1 bit 错误地址寄存器(ECC_SRAM_ERR1_ADDR)

表 23-3 ECC_SRAM_ERR1_ADDR 寄存器

ECC_SRAM_ ERR1_ADDR　　ECC_SRAM 1bit 错误地址寄存器　　RESET：0x00000000

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称				ERR1_ADDR [12：0]												
访问				R												
Reset				0												

字段	说明
12：0 ERR1_ADDR	**1 bit 错误地址** **注意：使用 ERR1_ADDR 的值乘以 4 加上 0x20000000 得到发生 1bit 错误的 SRAM 地址。**

23.4.3 2 bits 错误地址寄存器(ECC_SRAM_ERR2_ADDR)

表 23-4 ECC_SRAM_ERR2_ADDR 寄存器

ECC_SRAM_ERR2_ADDR **ECC_SRAM 2bit 错误地址寄存器** **RESET： 0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
访问																
Reset																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称				ERR2_ADDR [12：0]												
访问				R												
Reset				0												

字段	说明
12：0 ERR2_ADDR	**2 bits 错误地址** **注意：使用 ERR2_ADDR 的值乘以 4 加上 0x20000000 得到发生 2 bits 错误的 SRAM 地址。**

24 协处理器单元（MMDIVSQRT）

24.1 简介

MMDIVSQRT 模块是用于计算除法和开方根的，其计算公式的原型如下：

除法： (x << z) / y

开方根：$\sqrt[2]{x^2+y^2}$

被除数 x，除数 y 都是 32 bits, 移位 z 是 5 bits, 范围为 0~31。MMDIVSQRT 模块支持有符号整数、无符号整数的计算。

24.2 特性

- 支持 32 bits 有符号 /32 bits 无符号除法（或求余）的计算
- 支持 32 bits 有符号开方根的计算
- 模块编程配置简单，只有输入数据寄存器，计算结果寄存器和控制状态寄存器
- 除法运算支持计算的结果为商或者余数
- 除法使用的是 srt 算法，开方使用的是 cordic 算法

24.3 结构框图

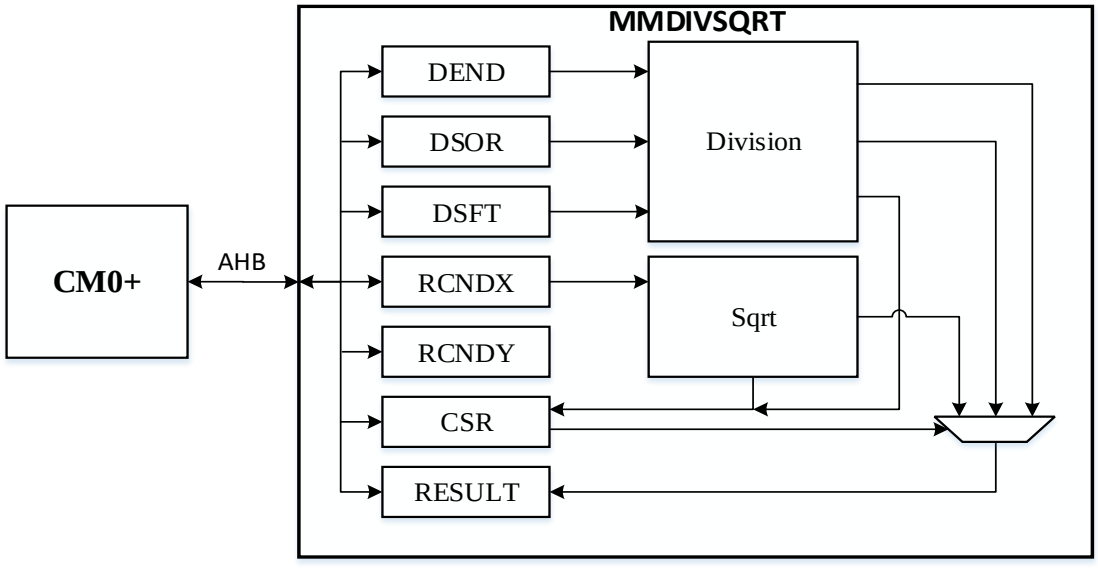

图 24-1 MMDIVSQRT 结构框图

24.4 应用说明

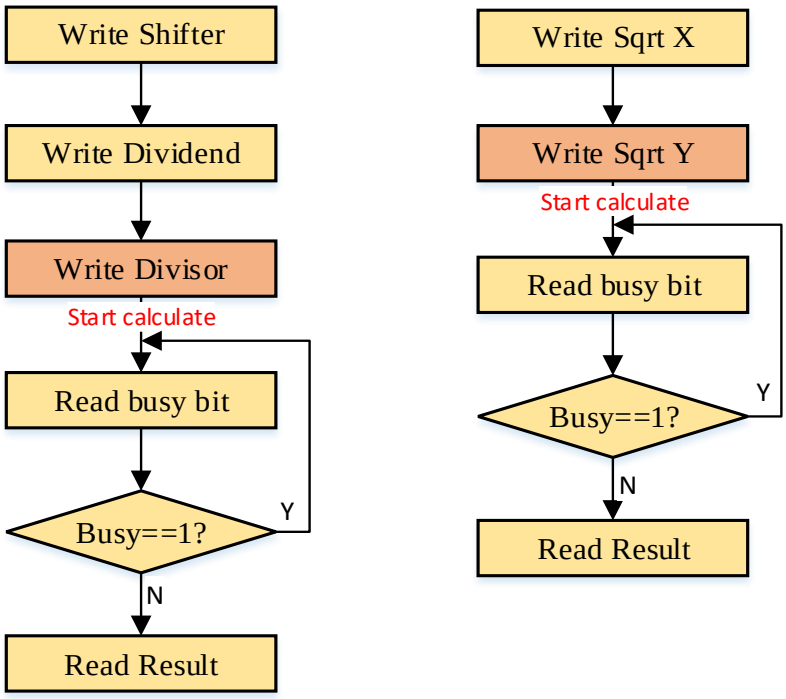

图 24-2 MMDIVSQRT 编程步骤图

除法运算配置步骤：

1. 默认为有符号的除法运算，如果需要配置为无符号的除法运算，请先设置 **USIGN** 位为 1。
2. 默认除法计算的结果为商，如果想要得到的是余数，请配置 **MEM** 位为 1。
3. 配置移位寄存器 MMDIVSQRT_DSFT，其范围为 0~31。
4. 配置被除数寄存器 MMDIVSQRT_DEND。
5. 配置除数寄存器 MMDIVSQRT_DSOR，配置完后，自动启动计算。
6. 读 **busy** 位，如果为 1，表示除法运算正在进行，需要继续等待。当 busy 位为 0，表示除法运算计算完毕，可以从 MMDIVSQRT_RESULT 寄存器读取计算的结果。
   MMDIVSQRT_RESULT 寄存器存放的结果为商或者余数，取决于 **MEM** 位。

开方根运算配置步骤：

1. 配置开方数 x 寄存器 MMDIVSQRT_RCNDX。
2. 配置开方数 y 寄存器 MMDIVSQRT_RCNDY，配置完后，自动启动计算。
3. 读 **busy** 位，如果为 1，表示开方根运算正在进行，需要继续等待。当 busy 位为 0，表示开方根运算计算完毕，可以从输出结果寄存器（MMDIVSQRT_RESULT）读取计算的结果。

注意

1. 开方根运算是有符号的运算，因此配置 USING 位对开方根运算无作用。
2. 开方根的精度: 8。即 RCNDX, RCNDY 的输入范围为-8192~8192，计算结果的误差保证在 8 以内。

24.5 寄存器定义

表 24-1 MMDIVSQRT 寄存器映射

MMDIVSQRT 基地址： 0x20081800

地址	名称	宽度	描述
MMDIVSQRT 基地址 + 0x000	MMDIVSQRT_DEND	32	被除数寄存器
MMDIVSQRT 基地址 + 0x004	MMDIVSQRT_DSOR	32	除数寄存器
MMDIVSQRT 基地址 + 0x008	MMDIVSQRT_DSFT	32	除法移位寄存器
MMDIVSQRT 基地址 + 0x00C	MMDIVSQRT_RCNDX	32	开方数 x 寄存器
MMDIVSQRT 基地址 + 0x010	MMDIVSQRT_RCNDY	32	开方数 y 寄存器
MMDIVSQRT 基地址+ 0x014	MMDIVSQRT_CSR	32	控制状态寄存器
MMDIVSQRT 基地址+ 0x018	MMDIVSQRT_RESULT	32	输出结果寄存器

24.5.1 被除数寄存器(MMDIVSQRT_DEND)

表 24-2 MMDIVSQRT_DEND 寄存器

MMDIVSQRT_DEND 被除数寄存器 **Reset：0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	DEND[31:16]															
访问	RW															
Reset	0															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	DEND[15:0]															
访问	RW															
Reset	0															

字段	说明
31：0 DEND	**DEND** 被除数

24.5.2 除数寄存器(MMDIVSQRT_DSOR)

表 24-3 MMDIVSQRT_DSOR 寄存器

MMDIVSQRT_DSOR 除数寄存器 **Reset：0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	DSOR[31:16]															
访问	RW															
Reset	0															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	DSOR[15:0]															
访问	RW															
Reset	0															

字段	说明
31：0 DSOR	**DSOR** 除数

24.5.3 除法移位寄存器(MMDIVSQRT_DSFT)

表 24-4 MMDIVSQRT_DSFT 寄存器

MMDIVSQRT_DSFT 除法移位寄存器 **Reset：0x00000000**

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

字段	说明
4：0 DSFT	**DSFT** 除数的移位（范围 0~31）

24.5.4 开方数 x 寄存器(MMDIVSQRT_RCNDX)

表 24-5 MMDIVSQRT_RCNDX 寄存器

MMDIVSQRT_RCNDX 开方数 x 寄存器 **Reset：0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	RCNDX[31:16]															
访问	RW															
Reset	0															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	RCNDX[15:0]															
访问	RW															
Reset	0															

字段	说明
31：0 RCNDX	**RCNDX** 开方数 x 范围为：-8192 ~ +8192

24.5.5 开方数 y 寄存器(MMDIVSQRT_RCNDY)

表 24-6 MMDIVSQRT_RCNDY 寄存器

MMDIVSQRT_RCNDY 开方数 y 寄存器 **Reset：0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	RCNDY[31:16]															
访问	RW															
Reset	0															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	RCNDY[15:0]															
访问	RW															
Reset	0															

字段	说明
31：0 RCNDY	**RCNDY** 开方数 y 范围为：-8192 ~ +8192

24.5.6 控制状态寄存器(MMDIVSQRT_CSR)

表 24-7 MMDIVSQRT_CSR 寄存器

MMDIVSQRT_CSR 控制状态寄存器 **Reset：0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	BUSY	DIV	SQRT													
访问	R	R	R													
Reset	0	0	0													
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称														REM	USIGN	
访问														RW	RW	
Reset														0	0	

字段	说明
31 BUSY	**Busy** 0：已经完成 1：正在执行 指示除法或开方根运算是否已经完成，由硬件自动设置
30 DIV	**Div** 0：是除法 1：不是除法 指示当前或上一次执行的运算是否为除法，由硬件自动设置
29 SQRT	**SQRT** 0：是开方根 1：不是开方根 指示当前或上一次执行的运算是否为开方根，由硬件自动设置
2 REM	**REM** 0：商 1：余数 选择除法运算结果类型，可以为商或余数

字段	说明
1 USIGN	**USIGN** 0：有符号 1：无符号 选择除法运算为有符号或无符号类型，即 DEND 和 DSOR 寄存器的值为有符号或无符号类型

24.5.7 输出结果寄存器(MMDIVSQRT_RESULT)

表 24-8 MMDIVSQRT_RESULT 寄存器

MMDIVSQRT_RESULT **输出结果寄存器** **Reset：0x00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	RESULT[31:16]															
访问	R															
Reset	0															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	RESULT[15:0]															
访问	R															
Reset	0															

字段	说明
31：0 RESULT	**RESULT** 计算结果输出，除法（商/余数）和开方根运算共用

25 调试

25.1 简介

该器件基于 ARM CoreSight 架构进行调试。外部调试器通过调试接口访问寄存器和存储器，控制程序运行/停止/复位等操作。该器件仅支持一个调试器接口，即串行线调试(SWD)。

25.2 特性

- 4 个硬件断点
- 2 个数据观察点
- SWD 接口访问