AC781x 参考手册

文档版本：　　1.2

发布日期：　　2022-01-06

文档修订信息

版本	日期	作者	描述
1.0	2018-10-16	AutoChips	初版
1.01	2019-03-21	AutoChips	1. 17.3 增加 GPIO 外部中断与中断向量对应关系的详细说明 2. 20.4 章节，增加 DMA 各模式下传输方式的详细说明
1.02	2019-03-29	AutoChips	1. 12.3.10 章节，补充完善内容
1.03	2019-04-08	AutoChips	1. 13 章节，补充完善 PWDT 内容
1.04	2019-04-16	AutoChips	1. 7.4 章节，补充增加 CAN 低通滤波器说明 2. 7.5.9 章节，修改 CAN 部分寄存器说明
1.05	2019-06-17	AutoChips	1. 12.3.14 章节，PWM 模块增加正交解码器模式 2. 15.3 章节，CTU 模块增加对每个功能具体描述
1.06	2019-09-11	AutoChips	1. 2.3.4 章节，修改信息块容量：48byte
1.07	2020-05-15	AutoChips	1. 10.8 章节，补充完善 ADC 部分寄存器说明 2. 18.8 章节，修改 I2C RACK 位属性由 R 改为 WR 3. 22.5 章节，修改 RTC 部分寄存器位属性 R 改为 RW
1.1	2021-06-16	AutoChips	1. 1.2 和 9.1 章节，增加只有 UART6 支持 LIN 的描述 2. 2.1 章节，增加 MPU 的相关描述 3. 2.2.1 章节，更新选项字节地址偏移 0x14 到 0x2F 4. 2.3.6 章节，增加芯片型号信息章节相关描述 5. 2.3.7 章节，增加芯片 UUID 信息章节相关描述 6. 7.4 和 7.5 章节，更新关于 KOER 寄存器的描述 7. 9.12 章节，更新 UART DR 位状态的描述 8. 9.12 章节，修改 UARTn_SLADDR 寄存器的描述 9. 9.12 章节，修改 UART 小数分频器的位宽度 10. 9.12 章节，修改 RS485 控制寄存器延时的描述 11. 9.12 章节，修改 IDLE 标志位的说明 12. 10 章节，增加 ADC 转换公式 13. 10.3 章节，修改 mode5 流程错误 14. 12 章节，修改同步支持描述错误 15. 14 章节，修改“定时器通道”字符为“定时器” 16. 表 17-5，修改 GPIO_ODR 寄存器的描述 17. 17.5 章节，增加上下拉电阻典型值的说明 18. 19.14 章节，修改 SPI_CFG1 寄存器的默认值和 CS_IDLE 位属性 RW 的宽度 19. 19.15 章节，修改 SPI_STATUS 寄存器溢出的描述 20. 表 20-2，修改 Byte 模式的描述 21. 20.4 章节，更新 DMA 操作模式的说明

			22. 20.6 章节，DMA 通道使能，增加在循环模式下，关闭 DMA 的描述 23. 20.6 章节，修改存储器结束地址寄存器的描述 24. 20.2 章节，DMA 删除外设到外设传输的描述 25. 23.1.2 章节，增加最小编程位宽为 32bit，编程地址需 4 字节对齐 26. 23.5 章节，更新 eFLASH_SR 寄存器的 RESET 值为 0x702 27. 23.5 章节，更新 eFLASH_CTRL2 寄存器的 CKDIV 位描述
1.2	2022-01-06	AutoChips	1. 2.3.11 章节，增加 ISP 下载引脚的描述 2. 7.4 章节，更新 Bus off 位的描述 3. 9.2 章节，增加 UART 波特率范围说明 4. 12.3.11 章节，新增章节，增加关于 PWM 写缓存更新寄存器的描述 5. 12.3.22 章节，新增章节，增加 PWM 特性优先级的描述 6. 17.3 章节，新增章节，增加 GPIO 的结构框图 7. 19.2 章节，增加 SPI 最大波特率的说明 8. 19.14 章节，增加关于 SPI TXEIE、RXFIE 的说明 9. 20.5.2 章节，更新 DMA 硬件优先级的描述 10. 20.5.3.3 章节和 20.6 章节，更新 DMA 结束地址的描述

版权声明

本参考手册包含 AutoChips 的机密信息。禁止未经授权使用或披露此手册包含的信息。 对因未经 AutoChips 授权而全部或部分披露此手册内容而给 AutoChips 带来的任何损失或损害，AutoChips 将追究责任。

AutoChips 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。AutoChips 对使用或依赖此处包含的信息不承担任何责任。

本参考手册包含的所有信息均“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。AutoChips 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。AutoChips 对本参考手册可能使用、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。 AutoChips 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

1 简介

1.1 概要

AC781x 是采用 ARM Cortex™-M3 内核的高性能、低功耗 MCU。

- 频率高达 100MHz。
- 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 AC781x 模块

模块	说明
ARM Cortex™-M3 内核	• ARM Cortex™-M3 32位MCU内核 • 高达100 MHz CPU 频率
存储器	• 高达 256 Kbyte的片内Flash存储器 • 高达 64 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16通道 • 2个模拟比较器（ACMP） • 内置1个6位的数模转换器（DAC）
定时器	• 1个全功能6通道脉宽调制（PWM）控制器 • 3个具备基本功能的双通道脉宽调制（PWM）控制器 • 8 个通用定时器（Timer） • 实时时钟（RTC） • 1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个两个内部集成电路（I2C）模块 • 6个通用异步收发器（UART） 模块，其中UART6支持 LIN 复用 • 2个局域网控制器（CAN） • 1个区域互联网络控制器（LIN）
接口	• 通用输入输出控制器（GPIO） • 中断（IRQ）
调试接口	• 联合测试工作组（JTAG）接口 • 串行调试（SWD）接口

2 存储器和总线架构

2.1 系统架构

主系统由以下部分构成：

- 四个主模块单元
  - Cortex™-M3 内核 ICode 总线（I-bus）
  - Cortex™-M3 内核 DCode 总线（D-bus）
  - Cortex™-M3 内核系统总线（S-bus）
  - DMA（通用 DMA）
- 五个从模块单元
  - 片内 SRAM
  - 片内 Flash 存储器
  - 外部串行 Flash 存储器
  - 快速 IO，CRC 和 GPIO
  - AHB 到 APB 的桥（AHB2APBx），它连接所有的 APB 设备。

如图 2-1 所示，这些都是通过一个多级的 AHB 总线架构相互连接。

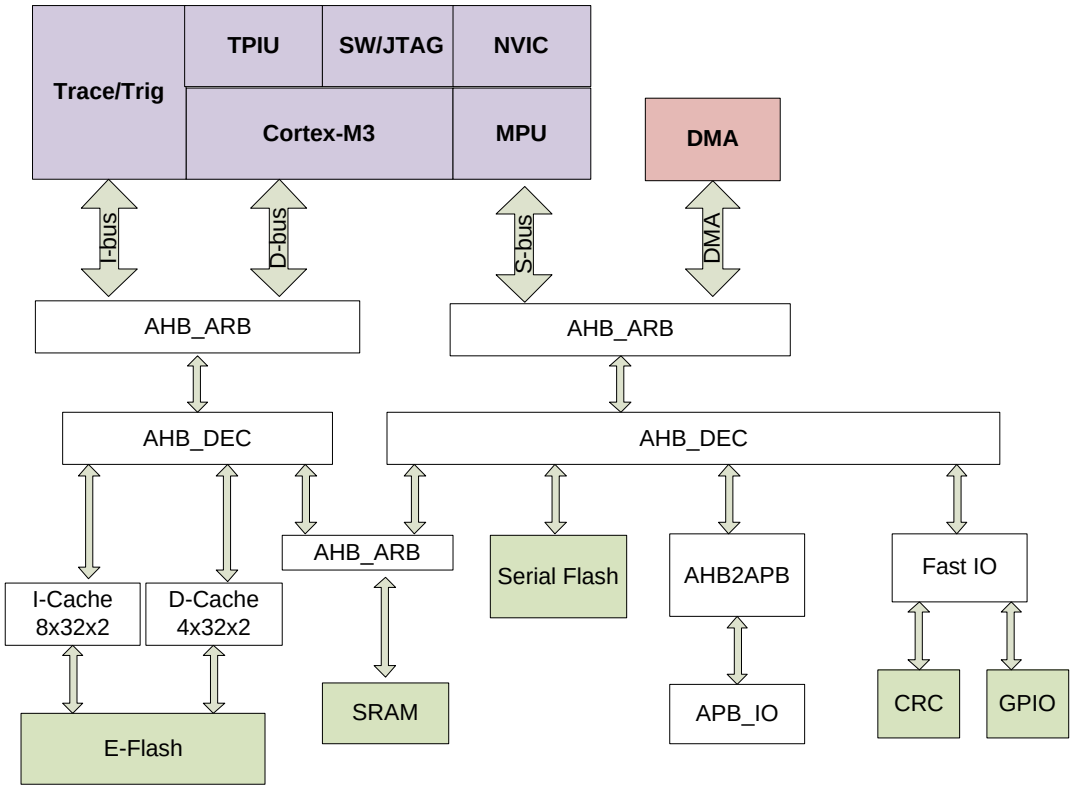

图 2-1 系统架构

ICode 总线

该总线将 Cortex™-M3 内核的指令总线与 Flash 指令接口相连接，指令预取在此总线上完成。

DCode 总线

该总线将 Cortex™-M3 内核的 DCode 总线（用于常量加载和调试访问）和 Flash 存储器数据接口相连接。

System 总线

该总线连接 Cortex™-M3 内核的系统总线（外设总线）到总线矩阵，总线矩阵用于协调内核和 DMA 之间的访问。

MPU

AC781x 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**《Cortex™-M3 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 DCode 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及从模块总线组成。
内部 Flash 只能被 ICode 和 DCode 访问。
ICode 和 DCode 能够访问 SRAM 和内部 Flash，DCode 访问优先级比 ICode 高。
AHB 外设通过系统总线与总线矩阵相连来访问 DMA。

AHB 与 APB 连接桥

AHB 与 APB 连接桥在 AHB 和 APB 总线间提供全同步连接，APB 操作频率限定为 AHB 操作频率的 1/2。

I-Cache

I-Cache 负责 Cortex™-M3 内核 ICode 和内部 Flash 存储器间的访问，包括直接访问 8x32x2 bit SRAM。当缓存匹配时，它提供零等待访问。
I-Cache 架构为 8 路、2 组，使用最近最少使用（LRU） 替换算法。

D-Cache

D-Cache 负责 Cortex™-M3 内核 DCode 和内部 Flash 存储器间的访问，包括直接访问 4x32x2 bit SRAM。当缓存匹配时，它提供零等待访问时间。D-Cache 优先级高于 I-Cache。D-Cache 架构为 2 路、2 组，使用最近最少使用（LRU） 替换算法。

2.2 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，小端指的是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。

2.2.1 存储器映射

表 2-2 为 AC781x 设备存储器映射表，包括四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、SRAM 启动和片外 Nor Flash 存储器启动。
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 设备存储器映射表

0xE010 0000	保留	0xE010 0000	保留	0xE010 0000	保留	0xE010 0000	保留
0xE00F FFFF	Cortex™-M3 内部外设	0xE00F FFFF	Cortex™-M3 内部外设	0xE00F FFFF	Cortex™-M3 内部外设	0xE00F FFFF	Cortex™-M3 内部外设
0xE000 0000		0xE000 0000		0xE000 0000		0xE000 0000	
0xDFFFFFFF	保留	0xDFFFFFFF	保留	0xDFFFFFFF	保留	0xDFFFFFFF	保留
0x6100 0000		0x6100 0000		0x6100 0000		0x6100 0000	
0x60FF FFFF	外部存储器	0x60FF FFFF	外部存储器	0x60FF FFFF	外部存储器	0x60FF FFFF	外部存储器
0x6000 0000		0x6000 0000		0x6000 0000		0x6000 0000	
0x5FFF FFFF	保留	0x5FFF FFFF	保留	0x5FFF FFFF	保留	0x5FFF FFFF	保留
0x4400 0000		0x4400 0000		0x4400 0000		0x4400 0000	
0x43FF FFFF	外设别名区	0x43FF FFFF	外设别名区	0x43FF FFFF	外设别名区	0x43FF FFFF	外设别名区
0x4200 0000		0x4200 0000		0x4200 0000		0x4200 0000	
0x41FF FFFF	保留	0x41FF FFFF	保留	0x41FF FFFF	保留	0x41FF FFFF	保留
0x4010 0000		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	0x400F FFFF	外设 APB 地址
0x4000 0000		0x4000 0000		0x4000 0000		0x4000 0000	
0x3FFF FFFF	保留	0x3FFF FFFF	保留	0x3FFF FFFF	保留	0x3FFF FFFF	保留
0x2400 0000		0x2400 0000		0x2400 0000		0x2400 0000	

0x23FF FFFF 0x2200 0000	AHB SRAM 别名区	0x23FF FFFF 0x2200 0000	AHB SRAM 别名区	0x23FF FFFF 0x2200 0000	AHB SRAM 别名区	0x23FF FFFF 0x2200 0000	AHB SRAM 别名区
0x21FF FFFF 0x2008 2000	保留	0x21FF FFFF 0x2008 2000	保留	0x21FF FFFF 0x2008 2000	保留	0x21FF FFFF 0x2008 2000	保留
0x2008 1FFF 0x2008 0000	AHB 快速 IO	0x2008 1FFF 0x2008 0000	AHB 快速 IO	0x2008 1FFF 0x2008 0000	AHB 快速 IO	0x2008 1FFF 0x2008 0000	AHB 快速 IO
0x2007 FFFF 0x2001 0000	保留	0x2007 FFFF 0x2001 0000	保留	0x2007 FFFF 0x2001 0000	保留	0x2007 FFFF 0x2001 0000	保留
0x2000 FFFF 0x2000 0000	AHB SRAM	0x2000 FFFF 0x2000 0000	AHB SRAM	0x2000 FFFF 0x2000 0000	AHB SRAM	0x2000 FFFF 0x2000 0000	AHB SRAM
0x1FFF FFFF 0x0804 3000	保留	0x1FFF FFFF 0x0804 3000	保留	0x1FFF FFFF 0x0804 3000	保留	0x1FFF FFFF 0x0804 3000	保留
0x0804 27FF 0x0804 2000	保留	0x0804 27FF 0x0804 2000	保留	0x0804 27FF 0x0804 2000	保留	0x0804 27FF 0x0804 2000	保留
0x0804 1FFF 0x0804 0800	ISP Firmware	0x0804 1FFF 0x0804 0800	ISP Firmware	0x0804 1FFF 0x0804 0800	ISP Firmware	0x0804 1FFF 0x0804 0800	ISP Firmware
0x0804 002F 0x0804 0000	选项字节	0x0804 002F 0x0804 0000	选项字节	0x0804 002F 0x0804 0000	选项字节	0x0804 002F 0x0804 0000	选项字节
0x0803_FFFF 0x0800 0000	Flash 存储器	0x0803_FFFF 0x0800 0000	Flash 存储器	0x0803_FFFF 0x0800 0000	Flash 存储器	0x0803_FFFF 0x0800 0000	Flash 存储器
0x07FF FFFF 0x0004 0000	保留	0x07FF FFFF 0x0000 1800	保留	0x07FF FFFF 0x0001 0000	保留	0x07FF FFFF 0x0100 0000	保留
0x0003 FFFF 0x0000 0000	Flash 存储器	0x0000 17FF 0x0000 0000	ISP Firmware	0x0000 FFFF 0x0000 0000	AHB SRAM	0x00FF FFFF 0x0000 0000	外部存储器
Flash 存储器启动		**ISP 启动**		**SRAM 启动**		**串行 Flash 存储器启动**	

2.3 详细功能描述

2.3.1 内置 SRAM

AC781x 内置一个 64Kbyte 的静态 SRAM，它可以以字节、半字（16 位）或全字（32 位）访问。SRAM 的起始地址为： 0x2000 0000。

2.3.2 位段

Cortex™-M3 存储器映射包含两个位段（bit-band）区。这两个位段区将别名存储器区中的每个字 映射到位段存储器区的一个位。在别名存储器区写入一个字具有对位段区的目标位执行读-改-写操作的相同效果。

在 AC781x 中，外设寄存器和 SRAM 都被映射到一个位段区里。这允许执行单一的位段读写操作。

下面的映射公式给出了别名区中的每个字是如何对应到位段区的相应位：

bit_word_addr = *bit_band_base* +（*byte_offset* x 32） +（*bit_number* × 4）

其中：

- *bit_word_addr* 是别名存储器区中字的地址，它映射到某个目标位。
- *bit_band_base* 是别名区的起始地址。
- *byte_offset* 是包含目标位的字节在位段中的序号。
- *bit_number* 是目标位所在位置（0-7） 。

【举例】

如下例子说明如何映射别名区中 SRAM 地址为 0x20000300 的字节中的第 2 位：

0x22006008 = 0x22000000 +（0x300*32） +（2*4）

对 0x22006008 地址的写操作与对 SRAM 中地址为 0x20000300 字节的第 2 位执行读-改-写操作具有相同的效果。

读 0x22006008 地址返回 SRAM 地址为 0x20000300（0x01： 位设置， 0x00： 位重置）字节中位 2 的值（0x01 or 0x00） 。

更多关于位段的信息，请参考**《Cortex™-M3 Technical Reference Manual》14.10 章节。**

2.3.3 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：0x20080000 ~ 0x20080FFF。

CRC 地址范围： 0x20081000 ~ 0x20081FFF。

2.3.4 片内 Flash 存储器

高性能片内 Flash 模块的主要特性如下：

- 高达 256 Kbyte 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 64 K × 32 bit，每个存储块划分为 128 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。

2.3.5 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。I-Cache 模块用于通过 ICode 总线进行指令读取。D-Cache 模块用于通过 DCode 总线进行数据读取。仲裁机制使 DCode 总线拥有更高的优先级。

2.3.6 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。

2.3.7 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）。

2.3.8 I-Cache 和 D-Cache

- I-Cache 容量为 8x32x2 bit， D-Cache 容量为 4x32x2 bit；
- I-Cache 控制器的最大效率为 Cortex™ M3 运行零等待；
- D-Cache 控制器的最大效率为 Cortex™ M3 运行零等待。

2.3.9 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，下面为每个外设的详细地址分配。

2.3.10 嵌套中断向量控制器（NVIC）

【特性】

- 55 个可屏蔽中断通道（不包括 16 个 Cortex™-M3 中断）；
- 128 个可编程优先等级（使用了 7 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™-M3 Technical Reference Manual》**第 5 章 异常及第 8 章 嵌套中断向量控制器（NVIC）。
AC781x 中断表如下：

表 2-3 中断表

	优先级	优先级类型	名称	说明
			保留	保留
	-3	固定	复位（Reset）	系统复位
	-2	固定	不可屏蔽中断（NMI）	不可屏蔽中断
	-1	固定	硬件失效（HardFault）	所有类型的失效
	0	可设置	存储管理（MemManage）	存储器管理
	1	可设置	总线错误（BusFault）	AHB 总线访问错误
	2	可设置	错误应用（UsageFault）	未定义的指令或非法状态
			保留	保留
			保留	保留
			保留	保留
			保留	保留
	3	可设置	SVCall	执行系统服务调用指令（SVC）引发的异常
	4	可设置	调试监控（DebugMonitor）	调试监视器（断点，数据观察点，或者是外部调试请求
			保留	保留
	5	可设置	PendSV	可挂起的系统服务请求
	6	可设置	SysTick	系统嘀嗒定时器
0	7	可设置	PWDT	PWDT 中断
1	8	可设置	UART_1	UART1 中断
2	9	可设置	UART_2	UART2 中断
3	10	可设置	UART_3	UART3 中断
4	11	可设置	UART_4	UART4 中断
5	12	可设置	UART_5	UART5 中断
6	13	可设置	UART_6	UART6 中断
7	14	可设置	EXTI0	EXTI 线 0 中断
8	15	可设置	EXTI1	EXTI 线 1 中断
9	16	可设置	EXTI2	EXTI 线 2 中断
10	17	可设置	EXTI3	EXTI 线 3 中断
11	18	可设置	EXTI4	EXTI 线 4 中断
12	19	可设置	EXTI5	EXTI 线 5 中断
13	20	可设置	SPI_1	SPI0 全局中断
14	21	可设置	SPI_2	SPI1 全局中断
15	22	可设置	IIC_1	IIC0 中断
16	23	可设置	IIC_2	IIC1 中断
17	24	可设置	DMA_1	DMA 通道 0 全局中断
18	25	可设置	DMA_2	DMA 通道 1 全局中断
19	26	可设置	DMA_3	DMA 通道 2 全局中断
20	27	可设置	DMA_4	DMA 通道 3 全局中断
21	28	可设置	DMA_5	DMA 通道 4 全局中断
22	29	可设置	DMA_6	DMA 通道 5 全局中断

	优先级	优先级类型	名称	说明
23	30	可设置	DMA_7	DMA 通道 6 全局中断
24	31	可设置	DMA_8	DMA 通道 7 全局中断
25	32	可设置	DMA_9	DMA 通道 8 全局中断
26	33	可设置	DMA_10	DMA 通道 9 全局中断
27	34	可设置	DMA_11	DMA 通道 10 全局中断
28	35	可设置	DMA_12	DMA 通道 11 全局中断
29	36	可设置	TIMER_0	Timer 0 全局中断
30	37	可设置	TIMER_1	Timer 1 全局中断
31	38	可设置	BKP	备份寄存器 侵入检测中断
32	39	可设置	RTC	RTC 中断
33	40	可设置	WDOG	看门狗定时中断
34	41	可设置	PWM_0	PWM 0 中断
35	42	可设置	PWM _1	PWM 1 中断
36	43	可设置	PWM _2	PWM 2 中断
37	44	可设置	PVD	可编程电压检测中断
38	45	可设置	LIN	LIN 中断
39	46	可设置	EXTI6	EXTI 线 6 中断
40	47	可设置	SPM	系统电源管理中断
41	48		保留	
42	49	可设置	CAN_1	CAN 1 中断
43	50	可设置	CAN_2	CAN 2 中断
44	51	可设置	ADC	ADC 中断
45	52	可设置	ACMP_0	ACMP 0 中断
46	53	可设置	ACMP_1	ACMP 1 中断
47	54	可设置	TIMER_2	TIMER 2 中断
48	55	可设置	TIMER_3	TIMER 3 中断
49	56	可设置	TIMER_4	TIMER 4 中断
50	57	可设置	TIMER_5	TIMER 5 中断
51	58	可设置	TIMER_6	TIMER 6 中断
52	59	可设置	TIMER_7	TIMER 7 中断
53	60	可设置	PWM _3	PWM 3 中断
54	61	可设置	Embedded Flash	片内 Flash 中断

2.3.11 3 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	UART1_CTS	UART1_RTS
eflash boot	0	x	x
ISP boot	1	0	0
SRAM boot	1	1	0
serial flash boot	1	0	1

说明：

- x 表示忽略，不用关注。
- 启动配置的控制与管脚是绑定的，必须是 PC7（UART1_RTS）和 PD0（UART1_CTS）。
- 应用端若只需要 eflash boot，在控制 BOOT=0 之后 UART1_RTS 与 UART1_CTS 可忽略电平状态，可用作其它用途。
- ISP 下载引脚为 UART2 的 PD1（TX）和 PD2（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（通过 ICode/DCode 总线访问）。Cortex™-M3 CPU 始终从 ICode 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）。AC781x 微处理器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 存储器和片外串行 Nor Flash 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。
- **从片外串行 Nor Flash 启动：**片外串行 Nor Flash 被映射到启动存储空间（0x0000 0000），但仍然能够在它原有的地址（0x6000 0000）访问它。

2.4 地址表

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
ADC	0x40003000	4K
ACMP	0x40005000	4K
LIN	0x40007000	2K

APB 存储映射	基地址	大小（字节）
CAN_1	0x40007800	1K
CAN_2	0x40007C00	1K
SPM	0x40008000	1K
RTC	0x40008400	1K
WDOG	0x4000B000	4K
SPI_1	0x4000C000	4K
SPI_2	0x4000D000	4K
IIC_1	0x4000E000	4K
IIC_2	0x4000F000	4K
Cortex™-M3 controller	0x40010000	2K
片外串行 Nor Fash 控制器	0x40010800	2K
TIMER	0x40011000	4K
DMA	0x40012000	4K
PWM_0	0x40013000	4K
PWM _1	0x40014000	4K
PWM _2	0x40015000	4K
CTU	0x40016000	4K
PWDT	0x40017000	4K
UART_1	0x40018000	4K
UART_2	0x40019000	4K
UART_3	0x4001A000	4K
UART_4	0x4001B000	4K
UART_5	0x4001C000	4K
UART_LIN_6	0x4001D000	4K
PWM _3	0x4001E000	4K
CRC	0x20081000	4K

3 复位（RESET）

3.1 简介

支持 8 种复位源产生 IC 复位。

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；
- System Reset：Cortex™-M3 软件复位；
- Lock up Reset： Cortex™-M3 死锁复位 ；
- 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 模块图

模块图如图 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。

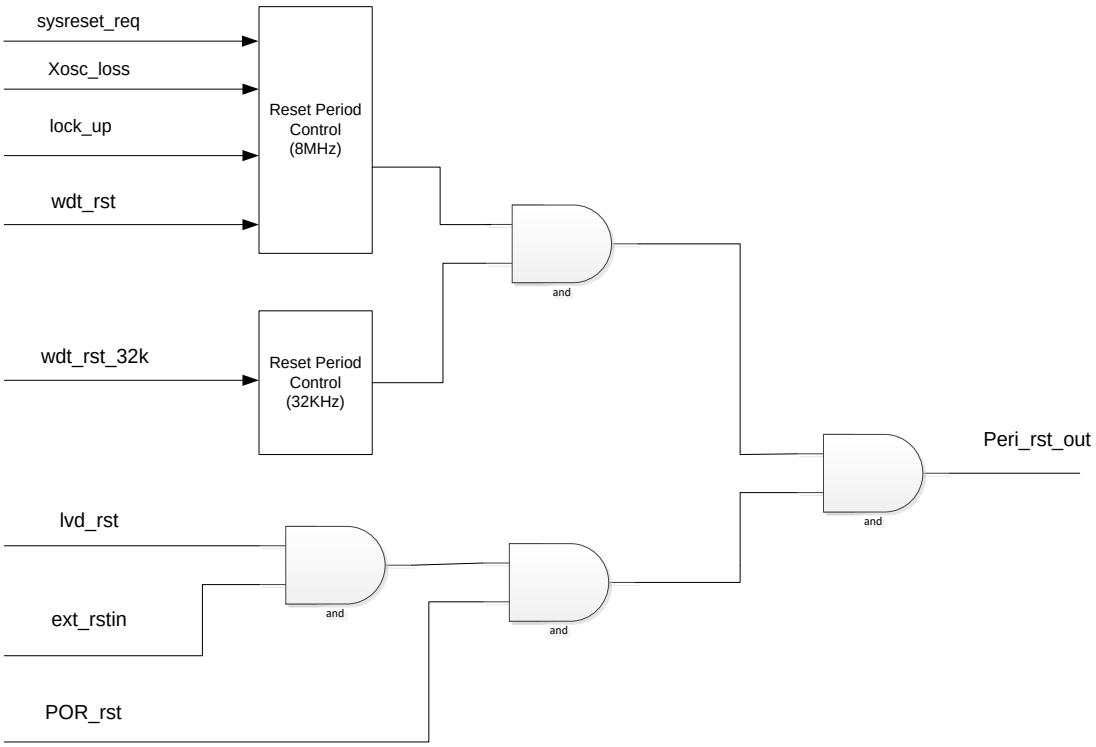

图 3-1 Reset 模块图

3.3 复位（Reset）详细功能描述

3.3.1 上电复位（POR）

产生 POR 的条件：

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），请参考《**ATC_AC781x_Datasheet_CH**》**DC 特性表 8**。

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，用于复位整个 MCU 功能和重启。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_AC781x_Datasheet_CH**》**5.1.1 DC 特性章节**。

3.3.2.3 看门狗定时器复位（Watchdog Reset）

看门狗定时器（WDOG）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，请参考看门狗定时器章节。

3.3.2.4 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，XOSC 失效状态标志置起并且生成 NMI 中断以通知软件故障允许 MCU 执行救援操作。

3.3.2.5 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 复位寄存器映射表

地址	寄存器名称	宽度（位）	寄存器功能
4000000C	**RESET_CTL1**	32	芯片复位控制
40000010	**RESET_STATUS**	32	芯片复位状态

4000000C　RESET_CTL1　芯片复位控制　00000800

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					reset_pulse_32K			
类型						RW	RW	RW					RW			
复位						0	0	1					0	0	0	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

名称	reset_pulse_8M															
类型	RW															
复位	0	0	0	0	1	0	0	0								

位（s）	名称	说明
26	Xosc_loss_rst_en	**XOSC 异常复位使能** 1'b1： 能产生 IC 复位 1'b0： 不能产生 IC 复位
25	cpu_lockup_rst_en	**CPU 死锁复位使能** 1'b1： 能产生 IC 复位 1'b0： 不能产生 IC 复位
24	cpu_sysrst_en	**CPU 系统复位使能** 1'b1：能产生 IC 复位 1'b0：不能产生 IC 复位
19：16	reset_pulse_32K	**提供 32kHz 时钟的可编程复位脉冲配置，包括看门狗定时器 32kHz 复位和停止模式 ACK 错误复位。**
15：8	reset_pulse_8M	**提供 8MHz 时钟的可编程复位脉冲配置，包括 cpu 系统复位，cpu 锁定复位和看门狗定时器复位。**

40000010 **RESET_STATUS** **芯片复位状态** **00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																**clear_reset_status**
类型																RW
复位																0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称							**xosc loss rst**	**stop_ack_error_rst_status**	**CPU_lockup_rst_status**	**CPU_sysreset_status**	**wdt_32k_reset_status**	**wdt_reset_status**	**nowdt_reset_status**	**ext_reset_status**	**LVD_reset_status**	**POR_reset_status**
类型							RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
复位							0	0	0	0	0	0	0	0	0	0

位（s）	名称	说明
16	clear_reset_status	**清除复位状态** 1'b1： 清除所有复位状态 1'b0： 允许复位状态更新
9	xosc loss status	**xosc loss 状态，高电平有效**
7	CPU_lockup_rst_status	**CPU 锁定复位状态，高电平有效**
6	CPU_sysreset_status	**CPU 系统复位状态，高电平有效**
5	wdt_32k_reset_status	**看门狗在 32K 工作模式下复位状态，高电平有效**
4	wdt_reset_status	**看门狗正常模式下复位状态，高电平有效**
2	ext_reset_status	**外部引脚复位状态，高电平有效**
1	LVD_reset_status	**LVD 复位状态，高电平有效**
0	POR_reset_status	**POR 复位状态,，高电平有效**

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 时钟控制图

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 晶体和振荡器。
- 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 100MHz 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，请参阅寄存器控制章节以了解详细地址。

【注意】

- 系统时钟最高可达 100MHz（典型使用值为 96MHz）
- hclk（AHB）最高可达 100MHz
- pclk（APB）最高可达 50MHz。因此当 hclk 为 100MHz 时，APBCLK_DIV 最小只能配置为 2

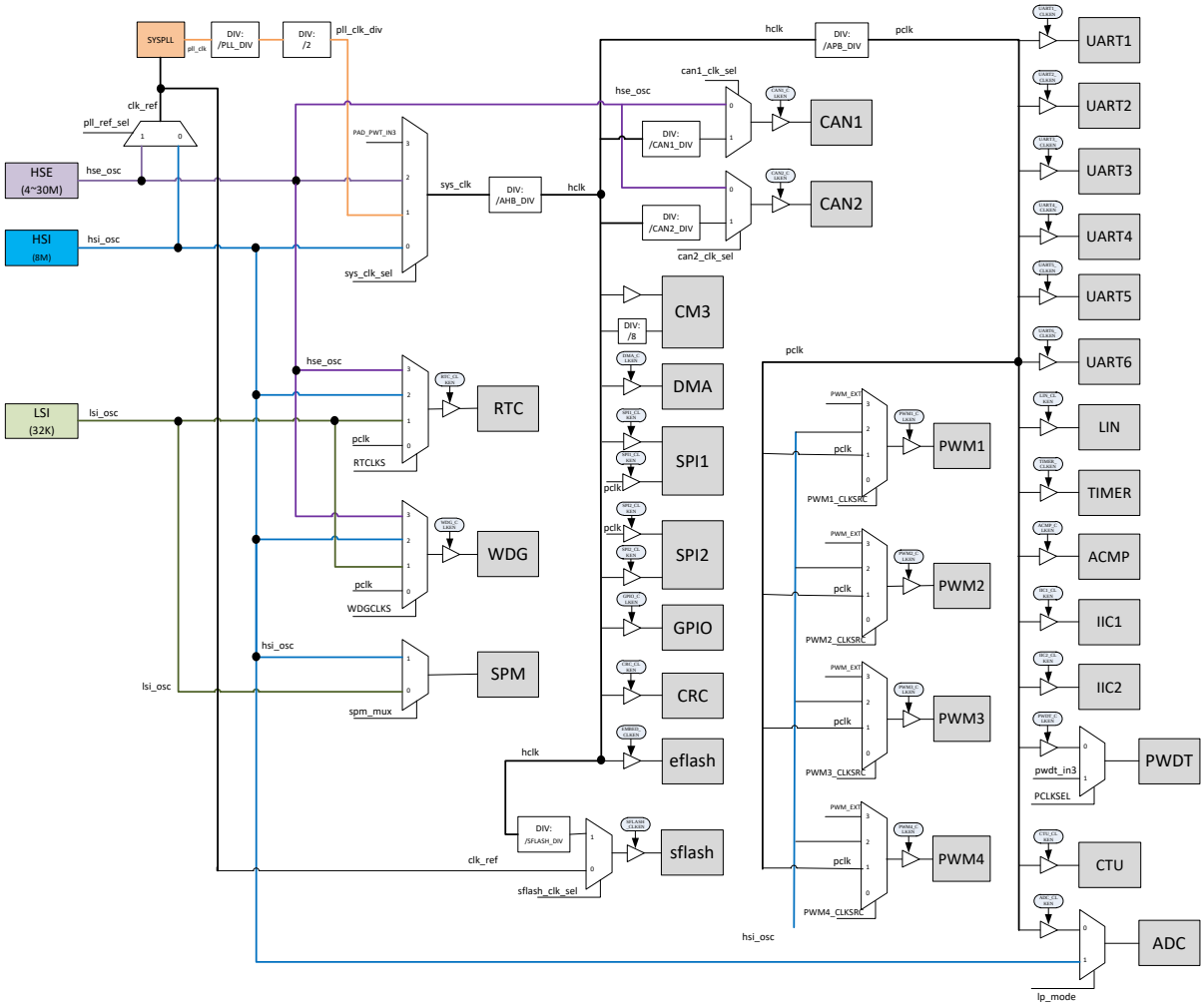

图 4-1 时钟控制图

4.3 系统时钟

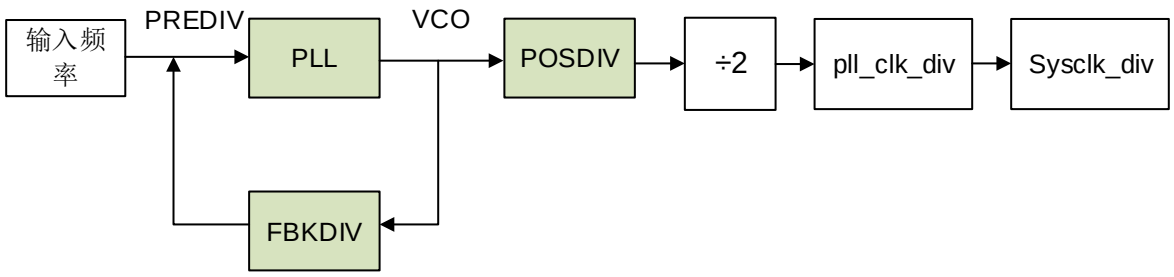

图 4-2 系统时钟示意图

- 输入频率， 支持 4MHz ~ 30MHz；
- VCO = 输入频率* FBKDIV / PREDIV；
- 系统时钟 = VCO / POSDIV / 2/ pll_clk_div/Sysclk_div。

【注意】

1. VCO 频率除以 POSDIV 不能高于 400MHz， VCO 的频率范围是 0.5GHz~1.5GHz；
2. FBKDIV 推荐值为 72, 80, 96, 128, 144, 160, 192；
3. 外部晶振为 4MHz~30MHz，PLL 输入频率推荐值不小于 8MHz。

4.4 寄存器定义

表 4-1 时钟寄存器映射表

地址	寄存器名称	宽度	寄存器功能
40000000	CKGEN_SRC_SEL	32	时钟源选择
40000004	PERI_CLK_EN_0	32	外设时钟使能控制 0
40000008	PERI_CLK_EN_1	32	外设时钟使能控制 1
40000018	PERI_SFT_RST1	32	外设软件复位控制 1
4000001C	PERI_SFT_RST2	32	外设软件复位控制 2
40000020	PLL_CLK_DIV	32	PLL 时钟分频器
40008890	REG_MCU_SYSPLL1_CFG0	32	SYSPLL1 配置寄存器 0

40000000　　CKGEN_SRC_SEL　　时钟源选择　　00000400

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	can2_clk_div		can1_clk_div		can2_clk_sel	can1_clk_sel		sflash_clk_sel				Pll_ref_sel				xosc_mon_enable
类型	RW		RW		RW	RW		RW				RW				RW
复位	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						apbclk_div			sysclk_div				sflash_clk_div		sysclk_sel	
类型						RW			RW				RW		RW	
复位						1	0	0	0	0	0	0	0	0	0	0

位（s）	名称	说明
31：30	can2_clk_div	**CAN2 时钟由 AHB 时钟分频产生（can2_clk <= 50MHz）** 2'h0： 1 分频 2'h1： 2 分频 2'h2： 4 分频 2'h3： 8 分频
29：28	can1_clk_div	**CAN1 时钟由 AHB 时钟分频产生（can1_clk <= 50MHz）** 2'h0： 1 分频 2'h1： 2 分频 2'h2： 4 分频 2'h3： 8 分频
27	can2_clk_sel	**CAN2 时钟源选择** 1'h0： 外部振荡器时钟 1'h1： ahb 分频时钟
26	can1_clk_sel	**CAN1 时钟源选择** 1'h0： 外部振荡器时钟 1'h1： ahb 分频时钟

位（s）	名称	说明
24	sflash_clk_sel	**串行 Flash 时钟源选择** 1'h0： pll 参考时钟 1'h1： ahb 分频时钟
20	Pll_ref_sel	**pll 参考时钟选择** 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_enable	**XOSC 监视器使能** 1： 监视器功能使能 0： 监视器功能禁用
10：8	apbclk_div	**APB 时钟由系统时钟分频产生** 3'b0xx： 1 分频 3'b100： 2 分频 3'b101： 4 分频 3'b110： 8 分频 3'b111： 16 分频
7：4	sysclk_div	**系统时钟分频器** 4'b0xxx ： 1 分频 4'b1000： 2 分频 4'b1001： 4 分频 4'b1010： 8 分频 4'b1011： 16 分频 4'b1100： 64 分频 4'b1101： 128 分频 4'b1110： 256 分频 4'b1111： 512 分频
3：2	sflash_clk_div	**串行 Flash 时钟由 APB 时钟分频得到** 2'h0： 1 分频 2'h1： 2 分频 2'h2： 4 分频 2'h3： 8 分频
1：0	sysclk_sel	**系统时钟源选择** 2'h0： 内部振荡器 2'h1： PLL 输出 2'h2： 外部振荡器 2'h3： 1'b0

40000004　PERI_CLK_EN_0　外设时钟使能控制 0　03400001

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	**CLK_CAN2_CORE**	**CLK_CAN1_CORE**	**CLK_CAN2**	**CLK_CAN1**	**CLK_LIN**	**CLK_CRC**	**CLK_WDT_APB**	**CLK_GPIO_AHB**	**CLK_GPIO_APB**	**CLK_DMA_AHB**	**CLK_DMA_APB**	**CLK_RTC**	**CLK_TIM**	**CLK_PWM3_TIMER**	**CLK_PWM2_TIMER**	**CLK_PWM1_TIMER**
类型	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	1	1	1	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	**CLK_PWM0_TIMER**	**CLK_PWM3_APB**	**CLK_PWM2_APB**	**CLK_PWM1_APB**	**CLK_PWM0_APB**	**CLK_PWDT**	**CLK_I2C2**	**CLK_I2C1**	**CLK_SPI2**	**CLK_SPI1**	**CLK_UART6**	**CLK_UART5**	**CLK_UART4**	**CLK_UART3**	**CLK_UART2**	**CLK_UART1**
类型	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

位（s）	名称	说明
31	CLK_CAN2_CORE	**CAN2 内核时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
30	CLK_CAN1_CORE	**CAN1 内核时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
29	CLK_CAN2	**CAN2 时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
28	CLK_CAN1	**CAN1 时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
27	CLK_LIN	**LIN 时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
26	CLK_CRC	**CRC 时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
25	CLK_WDT_APB	**WDG APB 时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
24	CLK_GPIO_AHB	**GPIO AHB 时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
23	CLK_GPIO_APB	**GPIO APB 时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
22	CLK_DMA_AHB	**DMA AHB 时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
21	CLK_DMA_APB	**DMA APB 时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
20	CLK_RTC	**RTC 时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
19	CLK_TIM	**TIMER 时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
18	CLK_PWM3_TIMER	**PWM3 定时器时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
17	CLK_PWM2_TIMER	**PWM2 定时器时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
16	CLK_PWM1_TIMER	**PWM1 定时器时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
15	CLK_PWM0_TIMER	**PWM0 定时器时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
14	CLK_PWM3_APB	**PWM3 APB 时钟使能** 1'b1： 时钟使能

位（s）	名称	说明
		1'b0： 时钟禁用
13	CLK_PWM2_APB	**PWM2 APB 时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
12	CLK_PWM1_APB	**PWM1 APB 时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
11	CLK_PWM0_APB	**PWM0 APB 时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
10	CLK_PWDT	**PWDT 时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
9	CLK_I2C2	**IIC2 时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
8	CLK_I2C1	**IIC1 时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
7	CLK_SPI2	**SPI2 时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
6	CLK_SPI1	**SPI1 时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
5	CLK_UART6	**UART6 时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
4	CLK_UART5	**UART5 时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
3	CLK_UART4	**UART4 时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
2	CLK_UART3	**UART3 时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
1	CLK_UART2	**UART2 时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
0	CLK_UART1	**UART1 时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用

40000008 **PERI_CLK_EN_1** **外设时钟使能控制 1** **00000001**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型																
复位																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称													**CLK_ACMP**	**CLK_ADC**	**CLK_CTU_APB**	**CLK_SFLASH**
类型													RW	RW	RW	RW
复位													0	0	0	1

位（s）	名称	说明
3	CLK_ACMP	**ACMP 时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
2	CLK_ADC	**ADC core 时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
1	CLK_CTU_APB	**CTU APB 时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用
0	CLK_SFLASH	**serial flash controller 时钟使能** 1'b1： 时钟使能 1'b0： 时钟禁用

40000018 **PERI_SFT_RST1** **外设软件复位控制 1** **03900001**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	**reserved**	**reserved**	**SRST_CAN2**	**SRST_CAN1**	**SRST_LIN**	**SRST_CRC**	**SRST_WDG**	**SRST_GPIO_AHB**	**SRST_GPIO_APB**	**SRST_DMA_AHB**	**SRST_DMA_APB**	**SRST_RTC**	**SRST_TIM**	**SRST_PWM3_TIMER**	**SRST_PWM2_TIMER**	**SRST_PWM1_TIMER**
类型	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	1	1	1	0	0	1	0	0	0	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	**SRST_PWM0_TIMER**	**SRST_PWM3_APB**	**SRST_PWM2_APB**	**SRST_PWM1_APB**	**SRST_PWM0_APB**	**SRST_PWDT**	**SRST_I2C2**	**SRST_I2C1**	**SRST_SPI2**	**SRST_SPI1**	**SRST_UART6**	**SRST_UART5**	**SRST_UART4**	**SRST_UART3**	**SRST_UART2**	**SRST_UART1**
类型	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

位（s）	名称	说明
29	SRST_CAN2	**CAN2 软件复位** 0： 复位有效 1： 复位无效
28	SRST_CAN1	**CAN1 软件复位** 0： 复位有效 1： 复位无效
27	SRST_LIN	**LIN 软件复位** 0： 复位有效 1： 复位无效
26	SRST_CRC	**CRC 软件复位** 0： 复位有效 1： 复位无效
25	SRST_WDG	**Watch dog 定时器软件复位** 0： 复位有效 1： 复位无效
24	SRST_GPIO_AHB	**GPIO AHB 软件复位** 0： 复位有效 1： 复位无效
23	SRST_GPIO_APB	**GPIO APB 软件复位** 0： 复位有效 1： 复位无效
22	SRST_DMA_AHB	**DMA AHB 软件复位** 0： 复位有效 1： 复位无效
21	SRST_DMA_APB	**DMA APB 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_TIM	**TIMER 软件复位** 0： 复位有效 1： 复位无效
18	SRST_PWM3_TIMER	**PWM3 定时器软件复位** 0： 复位有效 1： 复位无效
17	SRST_PWM2_TIMER	**PWM2 定时器软件复位** 0： 复位有效 1： 复位无效
16	SRST_PWM1_TIMER	**PWM1 定时器软件复位** 0： 复位有效 1： 复位无效
15	SRST_PWM0_TIMER	**PWM0 定时器软件复位** 0： 复位有效 1： 复位无效
14	SRST_PWM3_APB	**PWM3 APB 软件复位** 0： 复位有效 1： 复位无效
13	SRST_PWM2_APB	**PWM2 APB 软件复位** 0： 复位有效 1： 复位无效

位（s）	名称	说明
12	SRST_PWM1_APB	**PWM1 APB 软件复位** 0： 复位有效 1： 复位无效
11	SRST_PWM0_APB	**PWM0 APB 软件复位** 0： 复位有效 1： 复位无效
10	SRST_PWDT	**PWDT 软件复位** 0： 复位有效 1： 复位无效
9	SRST_I2C2	**IIC2 软件复位** 0： 复位有效 1： 复位无效
8	SRST_I2C1	**IIC1 软件复位** 0： 复位有效 1： 复位无效
7	SRST_SPI2	**SPI2 软件复位** 0： 复位有效 1： 复位无效
6	SRST_SPI1	**SPI1 软件复位** 0： 复位有效 1： 复位无效
5	SRST_UART6	**UART6 软件复位** 0： 复位有效 1： 复位无效
4	SRST_UART5	**UART5 软件复位** 0： 复位有效 1： 复位无效
3	SRST_UART4	**UART4 软件复位** 0： 复位有效 1： 复位无效
2	SRST_UART3	**UART3 软件复位** 0： 复位有效 1： 复位无效
1	SRST_UART2	**UART2 软件复位** 0： 复位有效 1： 复位无效
0	SRST_UART1	**UART1 软件复位** 0： 复位有效 1： 复位无效

4000001C　　PERI_SFT_RST2　　外设软件复位控制 2　　00000011

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型																
复位																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称												SRST_ANA_REG	SRST_ACMP	SRST_ADC	SRST_CTU	SRST_SFLASH
类型												RW	RW	RW	RW	RW
复位												1	0	0	0	1

位（s）	名称	说明
4	SRST_ANA_REG	**ANA 寄存器软复位**
3	SRST_ACMP	**ACMP 软件复位** 0： 复位有效 1： 复位无效
2	SRST_ADC	**ADC 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	SRST_SFLASH	**Sflash 软件复位** 0： 复位有效 1： 复位无效

40000020　　PLL_CLK_DIV　　pll 时钟分频器　　00000000

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

位（s）	名称	说明
3：0	pll_clk_div	divider = pll_clk_div + 1

40008890 **REG_MCU_SYSPLL1_CFG0** **SYSPLL1 配置寄存器 0** **003240C0**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	RG_MCU_SYSPLL1_PREDIV		RG_MCU_SYSPLL1_POSDIV					RG_MCU_SYSPLL1_FBKDIV								
类型	RW		RW					RW								
复位	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	Reserved					Reserved										
类型																
复位	0	1	0	0	0	0	0	0	1	1	0					

位（s）	名称	说明
31：30	RG_MCU_SYSPLL1_PREDIV	**预分频比** 2'b00： Fref = Fin/1 2'b01： Fref = Fin/2 2'b1X： Fref = Fin/4
29：28	RG_MCU_SYSPLL1_POSDIV	**Post-divider ratio for 单端时钟输出的分频后比** 2'b00： VCO/1 2'b01： VCO/2 2'b1X： VCO/4
24：17	RG_MCU_SYSPLL1_FBKDIV	**反馈分频比** 8'd6： /6 8'd255： /255

5 功耗模式（Power Modes）

5.1 简介

本章介绍芯片电源模式及其各个模块在这些模式下的功能。

5.2 功耗模式

支持运行（Run）、休眠（Sleep）、停止（Stop）、简约停止（StopLite）和待机（Standby）模式。

运行（Run）、休眠（Sleep）、停止（Stop）、简约停止（StopLite）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；
- 简约停止 – CPU 进入浅度休眠模式，多数模块都不会掉电且能唤醒 CPU；
- 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 进入和退出功耗模式

1. 使用 **REG_EN_PERIPH_WKUP** 使能需要的唤醒源；
2. 在 WFI 指令前使用 REG_SLEEP_MODE[1：0] 设置功耗模式，配置如下：
   1) 2’b00： 简约停止模式（支持更多 IP 逻辑状态保持，寄存器设置保持）
   2) 2’b01： 停止模式
   3) 2’b1x： 待机模式
3. 调用 WFI 指令进入功耗模式；

   处理器通过中断退出低功耗模式。

5.4 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	停止模式	简约停止模式	待机
CM3	待机	待机	待机	关闭
SRAM	开启	休眠	休眠	关闭

模块	休眠	停止模式	简约停止模式	待机
片内 Flash	开启	关闭	关闭	关闭
I2C	开启	开启 [3]	开启 [3]	关闭
SPI	开启	开启 [2]	开启 [2]	关闭
ACMP	开启	开启 [6]	开启	关闭
GPIO	开启	开启 [4]	开启 [4]	关闭
WDOG	开启	开启	开启	关闭
PWDT	开启	关闭	开启	关闭
UART	开启	关闭	开启	关闭
DMA	开启	关闭	开启	关闭
TIMER	开启	关闭	开启	关闭
PWM	开启	关闭	开启	关闭
CRC	开启	关闭	开启	关闭
CTU	开启	关闭	开启	关闭
CAN	开启	开启 [1]	开启 [1]	关闭
LIN	开启	开启 [1]	开启 [1]	关闭
RTC	开启	开启	开启	开启
SPM	开启	开启	开启	开启
PLL	开启	关闭	关闭	关闭
XOSC	开启	关闭	关闭	关闭
HSI	开启	关闭	关闭	关闭
LSI	开启	开启	开启	开启
ADC	开启	开启 [5]	开启 [5]	关闭
LVD	可选开启	可选开启	可选开启	可选开启

【注意】

1. 支持停止模式下的边沿唤醒；
2. 支持停止模式下的 从机模式接受和唤醒；
3. 支持停止模式下的地址匹配唤醒；
4. 支持停止模式下的引脚中断唤醒；
5. 支持停止模式下的模拟看门狗唤醒；
6. 支持 ACMP 设定电压比较唤醒。

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 功能列表

- 支持停止（Stop）模式下电源管理；
- 支持简约停止（StopLite）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

AC781x 支持停止（Stop）模式、简约停止（StopLite）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**。

简约停止（StopLite）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、简约停止和待机模式，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 **REG_EN_PERIPH_WKUP**；
3. 使能 SPM 电源控制，REG_PWR_EN；
4. 编程 spm 配置寄存器确定功耗模式， REG_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 REG_PWR_MGR_CFG1，打开或关闭 XOSC/SYSPLL。

SPM 寄存器 REG_PWR_MGR_CFG1：

- **REG_XOSC_ON**：外部高速时钟使能；
- **REG_XOSC_HSEBYP**：外部高速时钟旁路；

- **REG_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 REG_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 寄存器映射表

SPM： 0x40008000

地址	寄存器名称	说明
SPM + 0x000	**REG_PWR_MGR_CFG0**	电源管理器配置寄存器 0
SPM + 0x004	**REG_PWR_MGR_CFG1**	电源管理器配置寄存器 1
SPM + 0x00C	**REG_PERIPH_SLEEP_ACK_STATUS**	外设休眠应答状态
SPM + 0x010	**REG_EN_PERIPH_SLEEP_ACK_WAIT**	外设休眠应答使能寄存器
SPM + 0x014	**REG_EN_PERIPH_WKUP**	外设唤醒使能寄存器
SPM + 0x018	**REG_EN_TRIGGER_SPM_IRQ**	外设触发 SPM IRQ 使能寄存器
SPM + 0x01C	**REG_SPM_WAKEUP_IRQ_STATUS**	SPM 唤醒 IRQ 标志状态寄存器

SPM + 0x000　　REG_PWR_MGR_CFG0　　电源管理器配置寄存器 0

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型																
复位																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称							REG_PWR_MGR_CFG0[9：0]									
类型							RW									
复位							0	1	0	0	0	1	1	0	0	0

位域	名称	说明
31：10	保留	
9：8	REG_SLEEP_MODE	**休眠模式** 2'b00：停止模式 0

位域	名称	说明
		2'b01：停止模式 1 2'b1x：待机模式
7	REG_EN_IO_SUS_STOP_MODE	**在停止/停止精简模式禁用 IO** 1'b1：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。 1'b0：在进入停止（Stop）时，I/O 状态保持
6	REG_EN_CAN2_FILTER	**使能 CAN2 唤醒中断过滤器** 1'b1：使能 当使能 SPM 时，会使用模拟过滤器后的中断作为 CAN2 唤醒中断。 1'b0：禁用
5	REG_EN_CAN1_FILTER	**使能 CAN1 唤醒中断过滤器** 1'b1：使能 在使能 SPM 时，会使用模拟滤波器后的中断作为 CAN1 唤醒中断. 1'b0：禁用
4	REG_EN_LVD	**使能芯片低电压检测** 1'b1：使能 检测芯片 VCC 电压，如果 VCC 欠压，触发 LVD 复位 1'b0：禁用
3	REG_EN_DPWRLVD	**使能低电压检测** **1'b1：使能** 检测芯片内部 LDO 电压，如果 LDO 欠压，同样触发 LVD 复位 **1'b0：禁用**
2	REG_EN_PVD	**使能可编程电压检测** 1'b1：使能 检测芯片 VCC 电压，如果 VCC 欠压，触发 PVD 中断 1'b0：禁用
1	REG_EN_FAST_BOOT	**使能快速唤醒启动模式** 1'b1：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。 1'b0：禁用
0	REG_PWR_EN	**SPM 电源控制使能** 1'b1：使能 SPM 电源控制。 1'b0：禁用

SPM + 0x004　　REG_PWR_MGR_CFG1　　电源管理器配置寄存器 1

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型	R	R	R/W	R/W	R/W											
复位	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									REG_PWR_MGR_CFG1[7：0]							

类型									R/W							
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位域	名称	说明
31	XOSC_RDY	**XOSC 时钟就绪标志** 1'b1：就绪 1'b0：未就续
30	SYSPLL_RDY	**PLL 时钟就绪标志** 1：就绪 1'b0：未就续
29	REG_XOSC_HSEON	**外部高速时钟使能** 1'b1：使能 XOSC 1'b0：禁用 XOSC
28	REG_XOSC_HSEBYP	**外部高速时钟旁路** 1'b1：使用外部时钟旁路振荡器 1'b0：禁用外部时钟旁路振荡器
27	REG_SYSPLL_ON	**SYPLL 使能** 1'b1：使能 SYSPLL 1'b0：禁用 SYSPLL
26：7	保留	
6：4	保留	
3：0	PORLPVD	**PVDLVD 设置** 4'b0000： VLVDL = 2.65V VPVDL_0 = 2.7V 4'b0x01： VLVDL = 2.65V VPVDL_0 = 2.8V 4'b0x10： VLVDL = 2.65V VPVDL_0 = 2.9V 4'b0x11： VLVDL = 2.65V VPVDL_0 = 3.0V 4'b1x00： VLVDL = 4.3V VPVDL_0 = 4.4V 4'b1x01： VLVDL = 4.3V VPVDL_0 = 4.5V 4'b1x10： VLVDL = 4.3V VPVDL_0 = 4.6V 4'b1x11： VLVDL = 4.3V VPVDL_0 = 4.7V

SPM + 0x0c **REG_PERIPH_SLEEP_ACK_STATUS** **外设休眠确认状态寄存器**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称														eflash	gpio	adc
类型														R	R	R
复位														0	0	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	**dma**	**uart6**	**uart5**	**uart4**	**uart3**	**uart2**	**uart1**	**can2**	**can1**	**lin**	**spi2**	**spi1**	**iic2**	**iic1**	**acmp1**	**acmp0**
类型	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位域	名称	说明
31：19	保留	
18	片内 Flash	**片内 Flash 空闲（idle）状态** 1'b1：空闲 1'b0：忙
17	GPIO	**GPIO 休眠确认（sleep ack）状态** 1'b1：已回 ACK 1'b0：未回 ACK
16	ADC	**ADC 休眠确认（sleep ack） 状态** 1'b1：已回 ACK 1'b0：未回 ACK
15	DMA	**DMA 休眠确认（sleep ack） 状态** 1'b1：已回 ACK 1'b0：未回 ACK
14	UART6	**UART6 休眠确认（sleep ack） 状态** 1'b1：已回 ACK 1'b0：未回 ACK
13	UART5	**UART5 休眠确认（sleep ack） 状态** 1'b1：已回 ACK 1'b0：未回 ACK
12	UART4	**UART4 休眠确认（sleep ack） 状态** 1'b1：已回 ACK 1'b0：未回 ACK
11	UART3	**UART3 休眠确认（sleep ack） 状态** 1'b1：已回 ACK 1'b0：未回 ACK
10	UART2	**UART2 休眠确认（sleep ack） 状态** 1'b1：已回 ACK 1'b0：未回 ACK
9	UART1	**UART1 休眠确认（sleep ack） 状态** 1'b1：已回 ACK 1'b0：未回 ACK
8	CAN2	**CAN2 休眠确认（sleep ack） 状态** 1'b1：已回 ACK 1'b0：未回 ACK
7	CAN1	**CAN1 休眠确认（sleep ack） 状态** 1'b1：已回 ACK 1'b0：未回 ACK
6	LIN	**LIN 休眠确认（sleep ack） 状态** 1'b1：已回 ACK（需要 LIN 模块进入 sleep 模式） 1'b0：未回 ACK
5	SPI2	**SPI2 休眠确认（sleep ack） 状态** 1'b1：已回 ACK 1'b0：未回 ACK

4	SPI1	**SPI1 休眠确认（sleep ack） 状态** 1'b1：已回 ACK 1'b0：未回 ACK
3	I2C2	**I2C2 休眠确认（sleep ack） 状态** 1'b1：已回 ACK 1'b0：未回 ACK
2	I2C1	**I2C1 休眠确认（sleep ack） 状态** 1'b1：已回 ACK 1'b0：未回 ACK
1	ACMP1	**ACMP1 休眠确认（sleep ack） 状态** 1'b1：已回 ACK 1'b0：未回 ACK
0	ACMP1	**ACMP0 休眠确认（sleep ack） 状态** 1'b1：已回 ACK 1'b0：未回 ACK

SPM + 0x010　　REG_EN_PERIPH_SLEEP_ACK　　外设休眠确认等待使能寄存器

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称															GPIO	ADC
类型															R/W	R/W
复位															1	1
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	**DMA**	**UART6**	**UART5**	**UART4**	**UART3**	**UART2**	**UART1**	**CAN2**	**CAN1**	**LIN**	**SPI2**	**SPI1**	**I2C2**	**I2C1**	**ACMP1**	**ACMP0**
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

位域	名称	说明
31：18	reserved	
17	GPIO	**使能 GPIO 休眠确认等待（Sleep ACK Waiting）.** 1'b1：使能。在进入休眠时序前，spm 等待 GPIO 休眠确认。 禁用时，在进入休眠时序前，spm 不会检查 GPIO 休眠确认应答。 1'b0：禁用
16	ADC	**使能 ADC 休眠确认等待（Sleep ACK Waiting）.** 1'b1：使能 1'b0：禁用
15	DMA	**使能 DMA 休眠确认等待（Sleep ACK Waiting）.** 1'b1：使能 1'b0：禁用
14	UART6	**使能 UART6 休眠确认等待（Sleep ACK Waiting）.** 1'b1：使能 1'b0：禁用

13	UART5	**使能 UART5 休眠确认等待（Sleep ACK Waiting）.** 1'b1：使能 1'b0：禁用
12	UART4	**使能 UART4 休眠确认等待（Sleep ACK Waiting）.** 1'b1：使能 1'b0：禁用
11	UART3	**使能 UART3 休眠确认等待（Sleep ACK Waiting）.** 1'b1：使能 1'b0：禁用
10	UART2	**使能 UART2 休眠确认等待（Sleep ACK Waiting）.** 1'b1：使能 1'b0：禁用
9	UART1	**使能 UART1 休眠确认等待（Sleep ACK Waiting）.** 1'b1：使能 1'b0：禁用
8	CAN2	**使能 CAN2 休眠确认等待（Sleep ACK Waiting）.** 1'b1：使能 1'b0：禁用
7	CAN1	**使能 CAN1 休眠确认等待（Sleep ACK Waiting）.** 1'b1：使能 1'b0：禁用
6	LIN	**使能 LIN 休眠确认等待（Sleep ACK Waiting）.** 1'b1：使能 1'b0：禁用
5	SPI2	**使能 SPI2 休眠确认等待（Sleep ACK Waiting）.** 1'b1：使能 1'b0：禁用
4	SPI1	**使能 SPI1 休眠确认等待（Sleep ACK Waiting）.** 1'b1：使能 1'b0：禁用
3	I2C2	**使能 I2C2 休眠确认等待（Sleep ACK Waiting）.** 1'b1：使能 1'b0：禁用
2	I2C1	**使能 I2C1 休眠确认等待（Sleep ACK Waiting）.** 1'b1：使能 1'b0：禁用
1	ACMP1	**使能 ACMP1 休眠确认等待（Sleep ACK Waiting）.** 1'b1：使能 1'b0：禁用
0	ACMP0	**使能 ACMP0 休眠确认等待（Sleep ACK Waiting）.** 1'b1：使能 1'b0：禁用

SPM + 0x014　　REG_EN_PERIPH_WAKEUP　　外设唤醒使能寄存器

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称													lvd_warning	nmi	GPIO	ADC
类型													R/W	R/W	R/W	R/W
复位													0	0	1	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	**rtc**	**UART6**	**UART5**	**UART4**	**UART3**	**UART2**	**UART1**	**CAN2**	**CAN1**	**LIN**	**SPI2**	**SPI1**	**I2C2**	**I2C1**	**ACMP1**	**ACMP0**
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位域	名称	说明
31：20	保留	
19	PVD	**使能 PVD 报警唤醒** 1'b1： 使能。 禁用时，不支持 PVD 报警中断唤醒并且该唤醒会被忽略。 1'b0：禁用
18	NMI	**使能 NMI 唤醒** 1'b1：使能 1'b0：禁用
17	GPIO	**使能 GPIO 唤醒** 1'b1：使能 1'b0：禁用
16	ADC	**使能 ADC 唤醒** 1'b1：使能 1'b0：禁用
15	RTC	**使能 RTC 唤醒** 1'b1：唤醒 1'b0：禁用
14	UART6	**使能 UART6 唤醒** 1'b1：使能 1'b0：禁用
13	UART5	**使能 UART5 唤醒** 1'b1：使能 1'b0：禁用
12	UART4	**使能 UART4 唤醒** 1'b1：使能 1'b0：禁用
11	UART3	**使能 UART3 唤醒** 1'b1：使能 1'b0：禁用
10	UART2	**使能 UART2 唤醒**

		1’b1：使能 1’b0：禁用
9	UART1	**使能 UART1 唤醒** 1’b1：使能 1’b0：禁用
8	CAN2	**使能 CAN2 唤醒** 1’b1：使能 1’b0：禁用
7	CAN1	**使能 CAN1 唤醒** 1’b1：使能 1’b0：禁用
6	LIN	**使能 LIN 唤醒** 1’b1：使能 1’b0：禁用
5	SPI2	**使能 SPI2 唤醒** 1’b1：使能 1’b0：禁用
4	SPI1	**使能 SPI1 唤醒** 1’b1：使能 1’b0：禁用
3	I2C2	**使能 I2C2 唤醒** 1’b1：使能 1’b0：禁用
2	I2C1	**使能 I2C1 唤醒** 1’b1：使能 1’b0：禁用
1	ACMP1	**使能 ACMP1 唤醒** 1’b1：使能 1’b0：禁用
0	ACMP0	**使能 ACMP0 唤醒** 1’b1：使能 1’b0：禁用

SPM + 0x018　　REG_EN_TRIGGER_IRQ　　外设触发 SPM 中断或 外设中断配置寄存器

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称												over count	porlpvd warn	nmi	GPIO	ADC
类型												R/W	R/W	R/W	R/W	R/W
复位												1	0	1	0	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	**RTC**	**UART6**	**UART5**	**UART4**	**UART3**	**UART2**	**UART1**	**CAN2**	**CAN1**	**LIN**	**SPI2**	**SPI1**	**I2C2**	**I2C1**	**ACMP1**	**ACMP0**
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0

位域	名称	说明
31：21	保留	
20	spm over count	**使能 SPM over count 中断** 1'b1： 使能。 （1）.计数器溢出：进入休眠模式前，在特定的周期等待所有外设应答休眠确认。否则，定时器计数器会溢出。 （2）.当计数器溢出（在 0x8000 个 32k 时钟周期没有收到模块 ACK 信号），触发 SPM IRQ 唤醒 CM3； （3） 对于外设自身（如： rtc/ADC/GPIO/..） 触发唤醒请求，将相应的位设置为 0 即可。 1'b0： 禁用。
19	PVD	**PVD 唤醒中断配置** 1'b1：PVD 唤醒产生 SPM 中断 1'b0：PVD 唤醒产生 PVD 中断
18	NMI	**NMI 唤醒中断配置** 1'b1：NMI 唤醒产生 SPM 中断 注：正常产生 NMI 中断，也需配置该比特位为 1 1'b0：NMI 唤醒产生 NMI 中断
17	GPIO	**GPIO 唤醒中断配置** 1'b1：GPIO 唤醒产生 SPM 中断 1'b0：GPIO 唤醒产生 GPIO 中断
16	ADC	**ADC 唤醒中断配置** 1'b1：ADC 唤醒产生 SPM 中断 1'b0：ADC 唤醒产生 ADC 中断
15	RTC	**RTC 唤醒中断配置** 1'b1：RTC 唤醒产生 SPM 中断 1'b0：RTC 唤醒产生 RTC 中断
14	UART6	**UART6 唤醒中断配置** 1'b1：UART6 唤醒产生 SPM 中断 1'b0：UART6 唤醒产生 UART6 中断
13	UART5	**UART5 唤醒中断配置** 1'b1：UART5 唤醒产生 SPM 中断 1'b0：UART5 唤醒产生 UART5 中断
12	UART4	**UART4 唤醒中断配置** 1'b1：UART4 唤醒产生 SPM 中断 1'b0：UART4 唤醒产生 UART4 中断
11	UART3	**UART3 唤醒中断配置** 1'b1：UART3 唤醒产生 SPM 中断 1'b0：UART3 唤醒产生 UART3 中断
10	UART2	**UART2 唤醒中断配置** 1'b1：UART2 唤醒产生 SPM 中断 1'b0：UART2 唤醒产生 UART3 中断

9	UART1	**UART1 唤醒中断配置** 1'b1：UART1 唤醒产生 SPM 中断 1'b0：UART1 唤醒产生 UART1 中断
8	CAN2	**CAN2 唤醒中断配置** 1'b1：CAN2 唤醒产生 SPM 中断 1'b0：CAN2 唤醒产生 CAN2 中断
7	CAN1	**CAN1 唤醒中断配置** 1'b1：CAN1 唤醒产生 SPM 中断 1'b0：CAN1 唤醒产生 CAN1
6	LIN	**LIN 唤醒中断配置** 1'b1：LIN 唤醒产生 SPM 中断 1'b0：LIN 唤醒产生 LIN 中断
5	SPI2	**SPI2 唤醒中断配置** 1'b1：SPI2 唤醒产生 SPM 中断 1'b0：SPI2 唤醒产生 SPI2 中断
4	SPI1	**SPI1 唤醒中断配置** 1'b1：SPI1 唤醒产生 SPM 中断 1'b0：SPI1 唤醒产生 SPI1 中断
3	I2C2	**I2C2 唤醒中断配置** 1'b1：I2C2 唤醒产生 SPM 中断 1'b0：I2C2 唤醒产生 I2C2 中断
2	I2C1	**I2C1 唤醒中断配置** 1'b1：I2C1 唤醒产生 SPM 中断 1'b0：I2C1 唤醒产生 I2C1 中断
1	ACMP1	**ACMP1 唤醒中断配置** 1'b1：ACMP1 唤醒产生 SPM 中断 1'b0：ACMP1 唤醒产生 ACMP1 中断
0	ACMP0	**ACMP0 唤醒中断配置** 1'b1：ACMP0 唤醒产生 SPM 中断 1'b0：ACMP0 唤醒产生 ACMP0 中断

SPM + 0x01c　　REG_SPM_WAKEUP_IRQ_STATUS　　SPM 唤醒状态标志寄存器

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称												over count	porlpvd warn	nmi	GPIO	ADC
类型												R	R	R	R	R
复位												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**	**UART6**	**UART5**	**UART4**	**UART3**	**UART2**	**UART1**	**CAN2**	**CAN1**	**LIN**	**SPI2**	**SPI1**	**I2C2**	**I2C1**	**ACMP1**	**ACMP0**
类型	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位域	名称	说明
31：21	保留	
20	spm over count	**spm over count 唤醒标志位** 1'b1：spm 计数器触发此 spm_irq （1）. 当 spm_irq 触发器和 cm3 被唤醒时， cm3 会读取该寄存器以确定哪个外设触发了该唤醒。 （2）. cm3 应该将相应的为置 1 以清除 spm_irq。
19	PVD	**PVD 唤醒标志位** 1'b1：有效 1'b0：无效
18	nmi	**NMI 唤醒标志位** 1'b1：有效 1'b0：无效
17	GPIO	**GPIO 唤醒标志位** 1'b1：有效 1'b0：无效
16	ADC	**ADC 唤醒标志位** 1'b1：有效 1'b0：无效
15	RTC	**RTC 唤醒标志位** 1'b1：有效 1'b0：无效
14	UART6	**UART6 唤醒标志位** 1'b1：有效 1'b0：无效
13	UART5	**UART5 唤醒标志位** 1'b1：有效 1'b0：无效
12	UART4	**UART4 唤醒标志位** 1'b1：有效 1'b0：无效
11	UART3	**UART3 唤醒标志位** 1'b1：有效 1'b0：无效
10	UART2	**UART2 唤醒标志位** 1'b1：有效 1'b0：无效
9	UART1	**UART1 唤醒标志位** 1'b1：有效 1'b0：无效
8	CAN2	**CAN2 唤醒标志位** 1'b1：有效 1'b0：无效

7	CAN1	**CAN1 唤醒标志位** 1'b1：有效 1'b0：无效
6	LIN	**LIN 唤醒标志位** 1'b1：有效 1'b0：无效
5	SPI2	**SPI2 唤醒标志位** 1'b1：有效 1'b0：无效
4	SPI1	**SPI1 唤醒标志位** 1'b1：有效 1'b0：无效
3	I2C2	**I2C2 唤醒标志位** 1'b1：有效 1'b0：无效
2	I2C1	**I2C1 唤醒标志位** 1'b1：有效 1'b0：无效
1	ACMP1	**ACMP1 唤醒标志位** 1'b1：有效 1'b0：无效
0	ACMP0	**ACMP0 唤醒标志位** 1'b1：有效 1'b0：无效

7 控制器局域网（CAN）

7.1 简介

7.1.1 CAN-CTRL 内核

CAN-CTRL 控制器是一个基于 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。

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 控制器以控制帧的传输或接收。CAN 总线的连接如图 7-1 所示。

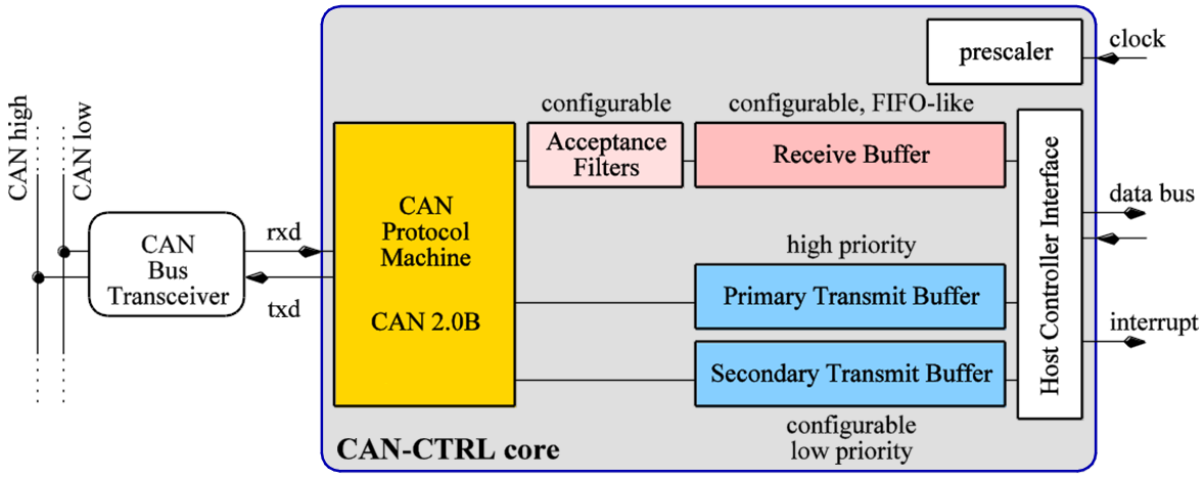

图 7-1 CAN 总线连接及 CAN-CTRL 内核主要功能示意图

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。

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由需要的消息标识符寻址。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配置进行比较。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 帧的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 控制器停止发送具有低优先级标识符的消息。CAN 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。

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)；
- 最高 1Mbit/s 的可编程比特率；
- 可选择外部晶振时钟或总线时钟为时钟源；
- 可编程波特率预分频器(1 至 1/256)；
- 9 个接收缓冲区；
- 两个发送缓冲区：
    - 1 个主发送缓冲区 (PTB)；
    - 5 个次发送缓冲区 (STB)，可按 FIFO 或优先级模式操作；
- 16 个可编程的接收滤波器，每个过滤器可单独设置 ID 和掩码；
- 中止发送功能
- 扩展特性：
    - 单次发送模式 (对于 PTB 和/或 STB)；
    - 监听模式；
    - 环回模式 (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；
- 带集成式低通滤波器的唤醒功能。

7.3 消息缓冲区

7.3.1 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。

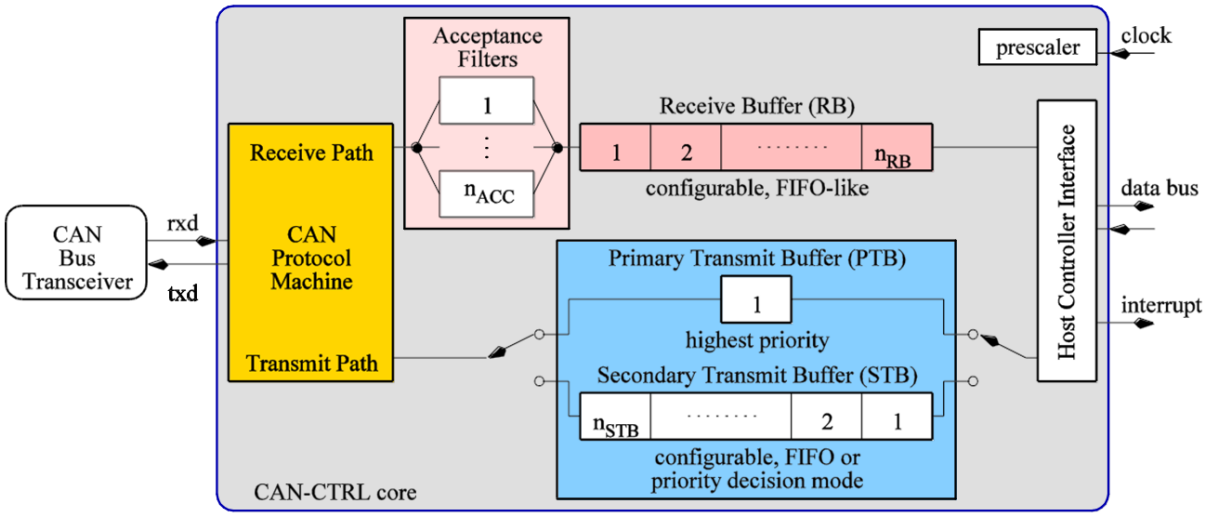

图 7-2 消息缓冲区示意图

7.3.2 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 控制器在验收过滤期间检查消息标识符。如果接收到的帧与其中一个接收过滤器的过滤条件相匹配，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。当 RB 已满或填充到用户可选择的“几乎满”限制时，CAN 控制器能够在每个接收信息上生成中断。由于类似 FIFO 的行为，主控制器始终从 RB 读取最旧的消息。

7.3.3 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 TB（PTB）具有更高的优先级，但只能缓冲一帧。次 TB（STB）具有较低的优先级， 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。

可以命令 STB 发送一个或所有存储帧。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 控制器来说，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 发送会自动重新启动。

PTB 发送在 CAN 协议可能的下一个发送位置开始（在下一个帧间空间之后）。因此，赢得仲裁并正在进行发送的 STB 发送将在 PTB 发送之前完成。

在以下情况下可以使用 PTB 发送中断 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决定请求 PTB 发送。
2. 请求 STB 发送单个帧，并且在完成 STB 发送前主控制器决定请求 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。

7.4 寄存器定义

CAN 控制器是一个 32 位组件，寄存器映射如表 7-1 所示。

表 7-1 CAN-CRTL 寄存器映射表

CAN_1： **0x40007800**

CAN_2： **0x40007C00**

	位								寄存器名称
	7	6	5	4	3	2	1	0	
0x00 ~ 0x4F	接收缓冲区寄存器（Receive Buffer Registers）								RBUF
0x50~ 0x97	发送缓冲区寄存器（Transmit Buffer Registers）								TBUF
0xA0	RESET	LBME	LBMI	TPSS	TSSS	RACTIVE	TACTIVE	BUSOFF	**CFG_STAT**
0xA1	TBSEL	LOM	STBY	TPE	TPA	TSONE	TSALL	TSA	**TCMD**
0xA2	-	TSNEXT	TSMODE	-			TSSTAT(1：0)		**TCTRL**
0xA3	-	ROM	ROV	RREL	-		RSTAT(1：0)		**RCTRL**
0xA4	RIE	ROIE	RFIE	RAFIE	TPIE	TSIE	EIE	TSFF	**RTIE**
0xA5	RIF	ROIF	RFIF	RAFIF	TPIF	TSIF	EIF	AIF	**RTIF**
0xA6	EWARN	EPASS	EPIE	EPIF	ALIE	ALIF	BEIE	BEIF	**ERRINT**
0xA7	AFWL(3:0)				EWL(3:0)				**LIMIT**
0xA8	S_Seg_1(7：0)								**S_Seg_1**
0xA9	-	S_Seg_2(6：0)							**S_Seg_2**
0xAA	-	S_SJW(6：0)							**S_SJW**
0xAB	S_PRESC(7：0)								**S_PRESC**
0xB0	KOER(2:0)			ALC(4:0)					**EALCAP**
0xB2	RECNT								**RECNT**
0xB3	TECNT								**TECNT**
0xB4	-		SELMASK	-	ACFADR				**ACFCTRL**
0xB6	AE_7	AE_6	AE_5	AE_4	AE_3	AE_2	AE_1	AE_0	**ACF_EN_0**
0xB7	AE_15	AE_14	AE_13	AE_12	AE_11	AE_10	AE_9	AE_8	**ACF_EN_1**
0xB8	ACODE_x or AMASK_x (7:0)								**ACF_0**
0xB9	ACODE_x or AMASK_x (15:8)								**ACF_1**
0xBA	ACODE_x or AMASK_x (23:16)								**ACF_2**
0xBB	-	AIDEE	AIDE	ACODE_x or AMASK_x (28:24)					**ACF_3**
0xBC	VERSION(7:0)								**VER_0**
0xBD	VERSION(15:8)								**VER_1**

0x0A0 CFG_STAT **配置和状态寄存器**

位	名称	权限	功能
7	RESET	rw-1	**复位请求** 1 ： 执行 CAN 控制器的本地复位 0 ： CAN 控制器不复位 若 RESET = 1，则只能修改某些寄存器（例如节点配置）。 RESET 位迫使多个组件进入复位状态。 如果节点进入“总线关闭”状态，会自动设置 RESET。 **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。
6	LBME	rw-0	**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。
5	LBMI	rw-0	**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	rw-0	**PTB 单次发送模式** 0 ： 禁用（发送失败后，消息会重发） 1 ： 使能
3	TSSS	rw-0	**STB 单次发送模式** 0 ： 禁用（发送失败后，消息会重发） 1 ： 使能
2	RACTIVE	r-0	**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	r-0	**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	rw-0	**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 ，仅用于调试。

0x0A1 TCMD **命令寄存器**

位	名称	权限	功能
7	TBSEL	rw-0	**发送缓冲区选择** 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。 0 ： PTB (高优先级缓冲区) 1 ： STB
6	LOM	rw-0	**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。

5	STBY	rw-0	**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 如果主控制器将 STBY 设置为 0，则收发器在主控制器请求新发送之前，需要等待时间。
4	TPE	rw-0	**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，主控制器无法设置它。 如果 RESET = 1，BUSOFF = 1，STBY = 1，LOM = 1，该位将复位为硬件复位值。
3	TPA	rw-0	**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，主控制器无法设置它。 如果 RESET = 1，BUSOFF = 1，该位将复位为硬件复位值。 TPA 不应与 TPE 同时设置。
2	TSONE	rw-0	**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，主控制器无法设置它。 如果 RESET = 1，BUSOFF = 1，STBY = 1，LOM = 1，该位将复位为硬件复位值。
1	TSALL	rw-0	**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，主控制器无法设置它。 如果 RESET = 1，BUSOFF = 1，STBY = 1，LOM = 1，该位将复位为硬件复位值。 如果在发送期间 STB 加载了新帧，则新帧也将被发送。换句话说，当 STB 变空时，由 TSALL 启动的发送结束。

0	TSA	rw-0	**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 或 BUSOFF = 1，该位将复位为硬件复位值。 TSA 不应与 TSONE 或 TSALL 同时设置。

不能同时设置 TSONE 和 TSALL。当已经设置了 TSALL，则无法设置 TSONE，反之亦然。如果同时设置 TSONE 和 TSALL，则 TSALL 有效，并且 CAN 控制器清除 TSONE。

0x0A2 TCTRL **发送控制寄存器**

位	名称	权限	功能
6	TSNEXT	rw-0	**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。 它没有任何伤害。 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。
5	TSMODE	rw-0	**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。
4：2	-	r-0	保留
1：0	TSSTAT	r-0	**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。

0x0A3 RCTRL **接收控制（Receive Control）寄存器**

位	名称	权限	功能
7	-	r-0	保留
6	ROM	rw-0	**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。
5	ROV	r-0	**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
4	RREL	rw-0	**接收缓冲区释放** 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。 1 ： 接收缓冲区释放，消息已经被读取； 0 ： 不释放。
3：2	-	r-0	保留
1：0	RSTAT	r-0	**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)

0x0A4 RTIE **接收和发送中断请求寄存器**

位	名称	权限	功能
7	RIE	rw-1	**接收中断使能** 0 ： 禁用 1 ： 使能
6	ROIE	rw-1	**RB 溢出中断使能** 0 ： 禁用 1 ： 使能
5	RFIE	rw-1	**RB 满中断使能** 0： 禁用 1 ： 使能
4	RAFIE	rw-1	**RB 几乎满中断使能** 0 ： 禁用 1 ： 使能
3	TPIE	rw-1	**主发送（Transmission Primary）中断使能** 0 ： 禁用 1 ： 使能
2	TSIE	rw-1	**次发送（Transmission Secondary ）中断使能** 0 ： 禁用 1 ： 使能
1	EIE	rw-1	**错误中断（Error Interrupt） 使能** 0 ： 禁用 1 ： 使能，中断标志对应 EIF
0	TSFF	r-0	**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。

0x0A5 RTIF **接收和发送中断标志寄存器**

位	名称	权限	功能
7	RIF	rw-0	**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。
6	ROIF	rw-0	**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。
5	RFIF	rw-0	**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
4	RAFIF	rw-0	**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
3	TPIF	rw-0	**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
2	TSIF	rw-0	**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
1	EIF	rw-0	**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
0	AIF	rw-0	**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 AIF 的源。 0： 没有执行中止 AIF 没有关联的启用寄存器。

0x0A6 ERRINT **错误中断使能和标志寄存器**

位	名称	权限	功能
7	EWARN	r-0	**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
6	EPASS	r-0	**错误被动模式激活** 0：没有激活 (节点错误激活) 1： 激活 (节点错误被动)
5	EPIE	rw-0	**错误被动中断使能** 0： 禁用 1： 使能
4	EPIF	rw-0	**错误被动中断标志** 1： 使能如果错误状态从主动错误状态变为被动错误状态，或反之 0： 没有被动错误状态改变

位	名称	权限	功能
3	ALIE	rw-0	**仲裁失利中断使能** 0： 禁用 1： 使能
2	ALIF	rw-0	**仲裁失利中断标志** 1： 仲裁失利发生 0： 没有仲裁失利
1	BEIE	rw-0	**总线错误中断使能** 0： 禁用 1： 使能
0	BEIF	rw-0	**总线错误中断标志。**总线错误类型对应 KOER。 1： 总线错误发生 0： 没有总线错误

要清除中断标志，主控制器需要将该标志位写 1，置 0 无效。如果在写访问处于活动状态时发生新的中断事件，则此事件将设置该标志并覆盖此重置。这可确保不会丢失中断事件。

0x0A7 LIMIT **警告限制寄存器**

位	名称	权限	功能
7：4	AFWL(3：0)	rw-0x1	**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 n_{RB}. AFWL_i = n_{RB} 是有效值，但请注意 RFIF 也存在。
3：0	EWL(3：0)	rw-0xB	**可编程错误警告限制** = (EWL+1)*8。可能的限制值为 8, 16, … 128。 EWL 的值控制 EIF

0x0A8 S_Seg_1 **位定时寄存器**

位	名称	权限	功能
7：0	S_Seg_1(7：0)	rw-0x3	**位定时段 1** 位定时开始后，采样点被设置为 $t_{Seg\_1}=(Seg\_1+2).TQ$

0x0A9 S_Seg_2 **位定时寄存器**

位	名称	权限	功能
7	-	r-0	保留
6：0	S_Seg_2(6：0)	rw-0x2	**位定时段 2** 在采样点后定时 $t_{Seg\_2}=(Seg\_2+1).TQ$。

0x0AA S_SJW 位定时寄存器

位	名称	权限	功能
7	-	r-0	保留
6：0	S_SJW(6：0)	rw-0x2	**同步补偿宽度** 同步补偿宽度 t_{SJW} =(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。

0x0AB S_PRESC 预分频器寄存器

位	名称	权限	功能
7：0	S_PRESC	rw-0x01	**预分频器** 预分频器将系统时钟分频以获得时间量子时钟 tq_clk。有效范围 PRESC=[0x00, 0xff] 导致分频器值为 1 ~ 256。

0x0B0 EALCAP 错误和仲裁丢失捕获寄存器

位	名称	权限	功能
7：5	KOER(2：0)	r-0x0	**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） (自己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(4：0)	r-0x0	**仲裁失利捕获** (仲裁失利的帧中的位所处的位置)

0x0B2-B3 RECNT 错误计数器寄存器 (TECNT (0xB3))

位	名称	权限	功能
7：0	RECNT	r-0x00	**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。
7：0	TECNT	r-0x00	**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。

0x0B4 ACFCTRL 接收滤波器控制寄存器

位	名称	权限	功能

7：6	-	r-0	保留
5	SELMASK	rw-0	**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。
4	-	r-0	保留
3：0	ACFADR	rw-0	**接收滤波器地址** ACFADR 指向某一个的接收滤波器 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。

接收过滤器寄存器 ACF_x 提供对接收过滤器编码 ACODE_x 和接收滤波器掩码 AMASK_x 的访问，具体取决于 SELMASK 的设置。只有 RESET = 1 时，才能对 ACF_x 进行读/写访问。 如果 RESET = 0，则从 ACF_x 读取结果为 0。

接收过滤器使用真正的 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。

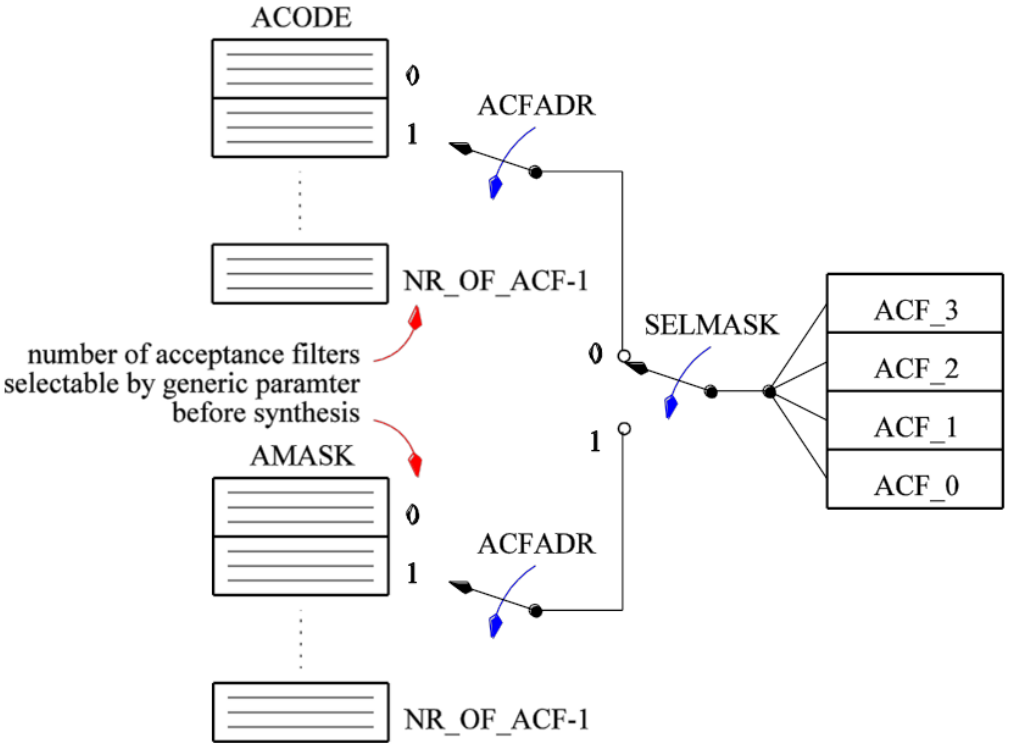

图 7-3 访问接收滤波器

0x0B8-BB ACF_x 接收代码 ACODE_x 寄存器

位	名称	权限	功能
28:0	ACODE_0 ACODE_x	rw-0x00 rw-u	**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。

0x0B8-BB ACF_x 接收代码 AMASK_x 寄存器

位	名称	权限	功能

28:0	AMASK_0 AMASK_x	rw-0xFF rw-u	**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。

AMASK_x 包括寄存器 ACF_3 中的附加位，只有在 SELMASK = 1 时才能访问它。这些位可用于仅接受具有所选 ACODE/AMASK 设置的标准帧或扩展帧，或接受两种帧类型。只有接收过滤器 0 受上电复位的影响，并且配置为在上电后接受两种帧类型。

表 7-2 ACF_3 寄存器位（SELMASK=1）

位	名称	权限	功能
6	AIDEE	rw-0 rw-u	**接受掩码 IDE 位检查使能** 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。
5	AIDE	rw-0 rw-u	**接受掩码 IDE 位的值** 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

0x0B6 ACF_EN_0 **接受过滤器使能寄存器**

位	名称	权限	功能
7：0	AE_x	rw-0x01	**接受过滤器使能 0-7** 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。

0x0B7 ACF_EN_1 **接受过滤器使能寄存器**

位	名称	权限	功能
7：0	AE_x	rw-0x00	**接受过滤器使能 8-15** 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。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。

0x0BC-BD VER_0 – 1 **版本信息寄存器**

位	名称	权限	功能

15：0	VER_0 VER_1	r	**CAN 控制器的版本**，以十进制数表示。 VER_1 代表主版本， VER_0 代表次版本。 例如：VER_1=5 ，VER_0=16。

表 7-3 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	-								标识符
RBUF+4	IDE=0	RTR			DLC(3:0)				控制位
RBUF+5	-								-
RBUF+6	-								-
RBUF+7	-								-
RBUF+8	d1(7:0)								数据字节 1
RBUF+9	d2(7:0)								数据字节 2
.	…								.
RBUF+15	d8(7:0)								数据字节 8

表 7-4 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	-			ID(28:24)					标识符
RBUF+4	IDE=1	RTR	-		DLC(3:0)				控制位
RBUF+5	-								-
RBUF+6	-								-
RBUF+7	-								-
RBUF+8	d1(7:0)								数据字节 1
RBUF+9	d2(7:0)								数据字节 2
.	…								.
RBUF+15	d8(7:0)								数据字节 8

RBUF 寄存器（0x00 至 0x47）以 RB 中最早收到的消息指向消息缓冲区。 所有 RBUF 寄存器可以以任意顺序读取。请注意 RBUF 的寻址范围在 RBUF + 7 的间隔。这是为了更好的地址段对齐，以实现更宽的主控制器接口。

表 7-5 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	-								标识符
TBUF+4	IDE=0	RTR	-		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 2
.	…								.
TBUF+15	d8(7:0)								数据字节 8

表 7-6 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(23:16)								标识符
TBUF+3	-			ID(28:24)					标识符
TBUF+4	IDE=1	RTR	-		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 2
.	…								.
TBUF+15	d8(7：0)								数据字节 8

如果 TBSEL=1，TBUF 寄存器（0x48 至 0x8f）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，这是做为了更好的地址段对齐。

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 7-7** 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 7-7 RBUF 和 TBUF 的控制位

位	描述
IDE	**标识符扩展** 0 –标准格式： ID(10：0) 1 –扩展格式： ID(28：0)
RTR	**远程发送请求** 0 – 数据帧 1 – 远程帧 只有 CAN 2.0 帧可以是远程帧。

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 EDL = 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，则需要小心。在这种情况下，所有发射器都需要使用相同的 DLC，否则会导致无法解决的冲突。

表 7-8 DLC 定义 (基于 CAN 2.0 规格)

DLC (二进制)	帧类型	有效载荷（字节）
0000 ~ 1000	CAN 2.0	0 ~ 8
1001 ~ 1111	CAN 2.0	8

TBUF 寄存器可读写。 因此，如果需要，主控制器可以使用 TBUF 依次逐位地准备消息。

7.5 通用操作

7.5.1 总线关闭状态

使用 CFG_STAT 寄存器中的状态位 BUSOFF 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 节点通过上电复位而复位或者接收到 128 组 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 会回滚，因此可能保持为一个较小的值。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。当 BUSOFF 置位后，则自动设置 RESET。因此，RECNT 和 TECNT 都不受软件复位的影响。

7.5.2 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，则该消息会被接收。如果该消息被接收，则会将其存储到 RB 中，如果启用了 RIE，则 RIF 会被置位。如果该消息未被接收，则不置位 RIF 且 RB FIFO 指针不会增加。未被接收的消息将被丢弃并被下一条消息覆盖。任何未被接收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，并向总线发送应答或错误帧。

接收掩码定义了是否要进行比较的位，而接收代码定义了需要匹配的值。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 的掩码位不进行相应位的接收检查。

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接收消息。与此相反，禁用 AMASK_x 中的掩码位会禁用对此位的检查，这会导致接收消息。

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。

上电复位或模块复位后，默认配置 CAN 控制器会接收所有消息 (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 定义复位值得过滤器， 而所有其他过滤器都有未定义的复位值)。

7.5.3 消息接收

接收数据被存储在 RB 中，如图 7-4 所示，具有类似 FIFO 的行为。如果启用了 RIE，则每个收到的有效和可被接受的消息都会置位 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，则会置位 RAFIF。如果所有缓冲区都已满，则当使能 RFIE 时，置位 RFIF。

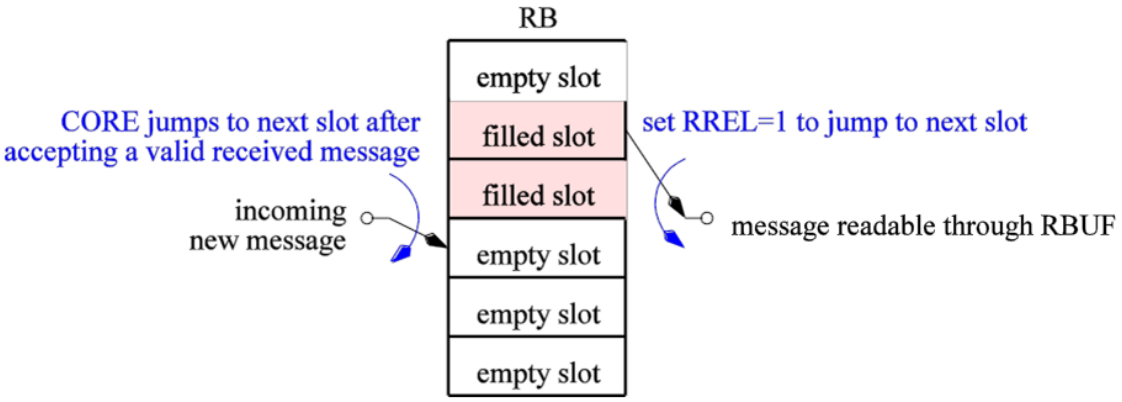

图 7-4 类似 FIFO RB 示意图 （6 slots 示例）

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 slots，并且主控制器在读取完成当前消息后，设置 RREL 以跳转到下一个 RB slot。 可以按任何顺序读取实际 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。

7.5.4 处理消息接收

如果没有接收滤波器，CAN 控制器会对接收到的每帧发出接收信号，并且主控制器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收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 置位为 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，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 包括比最大 slot 数量多一个（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接收过滤器相匹配。

7.5.5 消息发送

在开始任何发送之前，必须向至少一个发送缓冲区（PTB 或 STB）加载消息。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。

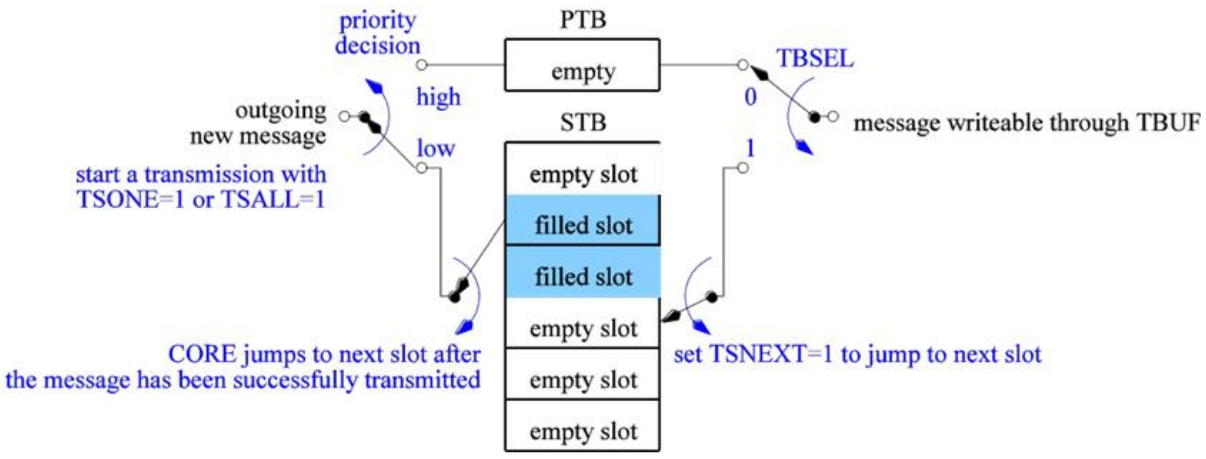

图 7-5 FIFO 模式下 PTB 及 STB 示意图 （空 PTB， 6 STB slots)

CAN 2.0 消息的最大有效载荷长度为 8 个字节。每条消息的长度由 DLC 定义。对于远程帧（RTR 位），DLC 的值没有作用，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 字节都可以按任何顺序写入。

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

使能 TPE 发送 PTB 中的消息。而要发送 STB 中的消息，需使能 TSONE 来发送单个消息或 TSALL 来发送所有消息。

PTB 的优先级始终比 STB 高。如果两个发送缓冲区都具有发送顺序，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。因此，如果主控制器在 TSALL 发送开始后向 STB 写入附加消息，则此附加消息在 TSIF 置位之前也会被发送。

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。

7.5.6 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 只提供单个中断标志 AIF。CAN 控制器只有在不向 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：

- 总线仲裁期间不能中止
  - 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。
- 发送帧时不能中止
  - 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。
  - 在 CAN 节点未接收到应答的发送失败之后，错误计数器递增并且将执行中止。
  - 如果在 STB 中剩余至少一个帧，而主控制器已命令所有要发送的帧（TSALL = 1），则完成发送的帧以及中止的帧都会通知主控制器。

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 的单个消息 slot 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，因为主控制器无法访问它。

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，只有在主控制器没有立即处理 TPIF 时才会发生)；
- TPIF + TSIF + AIF (很少，只有在主控制器没有立即处理 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，主控制器可以使用 ALIF/ALIE。

7.5.7 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控制器请求中止(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。

7.5.8 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。

7.5.8.1 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/发送错误警告事件。可以从 8 到 128 的 8 个错误中选择极限值：

错误计数限制 = (EWL + 1) * 8.

如果在以下条件下由 EIE 使能，则将置位中断 EIF：

- RECENT 或 RECENT 在错误警告限制的边界发生任一方向上的变化；
- BUSOFF 位发生的变化。

7.5.8.2 仲裁失利捕获

CAN 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 中断检测此事件，并且 ALC 可以指示失利在仲裁段中的位置。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。

7.5.8.3 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 中断检测 CAN 总线错误信号。每个新的错误事件都会覆盖以前存储的 KOER 值。因此，主控制器必须对错误事件做出快速反应。

7.5.9 扩展特性

7.5.9.1 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 和通过 TSSS 位对发送缓冲器 STB 分别设置发送消息一次的顺序。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，会发生如下问题：

- 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。因此，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，则主控制器无法检测哪个是成功的帧。

如果总线被另一个帧占用， 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。

7.5.9.2 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。

将在 LOM 中监视错误 (KOER, BEIF) 。

- 在 LOM 中，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，没有应答)。 这是使用如下规则完成的；
- 在 LOM 中，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。只有控制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，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：

- 如果帧由节点发送，则仅当至少一个附加节点连接到不在 LOM 的总线时，才会生成在总线上可见的应答。没有错误，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。

需注意当发送处于活动状态时，不应使能监听模式。如果使能 LOM ，则无法开始发送。

7.5.9.3 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应答)。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应答信息，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）。

7.5.9.4 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)。两种模式都可以接收自己发送的帧，便于自测。

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 控制器接收来自自己的帧。在 LBME 模式下，节点不产生自应答信号。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，并产生应答信号。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，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。

需注意当发送处于活动状态时，不应使能环回模式。

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 0。最后可以禁用 RESET 并且控制器返回至正常操作。与此相反，LBME 每次都可以被禁用。

7.5.9.5 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静默完成的。

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。

7.5.9.6 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 1 写入 BUSOFF 位会复位错误计数器，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。

7.5.9.7 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 2us），可设置 REG_EN_CANx_FILTER。

7.5.10 软件复位

如果 CFG_STAT 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 7-9 软件复位

寄存器	复位 (RESET)	说明
RBUF	(是)	所有 RB slot 被标记为空， RBUF 包含未知数据。
TBUF	(是)	所有 STB slot 被标记为空，因为 TBSEL TBUF 指向 PTB。
TTS	否	-
LBME	是	-
LBMI	是	-
TPSS	是	-
TSSS	是	-
RACTIVE	是	即使接待处于活动状态，接收也会立即取消。 不会生成 ACK 信息。
TACTIVE	是	在 RESET 时，所有发送都立即停止。如果发送有效，则会产生性感帧。其他节点会产生错误帧。
BUSOFF	否	-
TBSEL	是	TBUF 固定指向 PTB。
LOM	是	-
STBY	否	-
TPE	是	-
TPA	是	-
TSONE	是	-
TSALL	是	-
TSA	是	-
TSNEXT	是	-
TSMODE	否	-
TSSTAT	是	所有STB slot被标记为空
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
RIE	否	-
ROIE	否	-
RFIE	否	-

寄存器	复位 (RESET)	说明
RAFIE	否	-
TPIE	否	-
TSIE	否	-
EIE	否	-
TSFF	是	所有STB slot 被标记为空
RIF	是	-
ROIF	是	-
RFIF	是	-
RAFIF	是	-
TPIF	是	-
TSIF	是	-
EIF	否	-
AIF	是	-
EWARN	否	-
EPASS	否	-
EPIE	否	-
EPIF	是	-
ALIE	否	-
ALIF	是	-
BEIE	否	-
BEIF	是	-
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。
S_PRESC	否	如果RESET=1则寄存器可写，为0则寄存器写锁定。
AFWL	否	-
EWL	是	-
KOER	是	-
ALC	是	-
RECNT	否	-
TECNT	否	-
SELMASK	否	-
ACFADR	否	-
AE_x	否	-

寄存器	复位 (RESET)	说明
ACODE_x	否	如果RESET=1则寄存器可写，为0则寄存器写锁定。
AMASK_x	否	如果RESET=1则寄存器可写，为0则寄存器写锁定。

7.5.11 CAN 位时间

CAN 2.0B 定义了高达 1Mbit/s 的数据比特率。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。

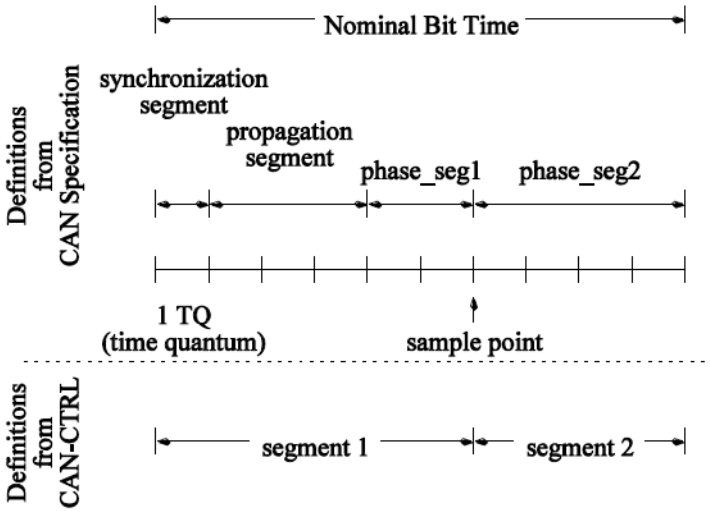

图 7-6 CAN 位定时

CAN 位时间 BT 由若干段（segment）组成，如图 7-6 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：

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

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

CAN 规范要求段长度之间存在若干关系 （如表 7-10 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 7-10 列举了 CAN 规格定义的最小配置范围。

表 7-10 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

段	说明		
PHASE_SEG1	相位缓冲段 1 [1...8] TQ [1...16] TQ	CAN 2.0 比特率 CAN 2.0 比特率	CAN FD 未启用 CAN FD 已启用
PHASE_SEG2	相位缓冲段 2 [2...8] TQ [2...16] TQ	CAN 2.0 比特率 CAN 2.0 比特率	CAN FD 未启用 CAN FD 已启用
SJW	同步跳转宽度 [1...4] TQ [1...16] TQ	CAN 2.0 比特率 CAN 2.0 比特率	CAN FD 未启用 CAN FD 已启用

如**表 7-10** 所示，CAN 控制器将 SYNC_SEG，PROP_SEG 和 PHASE_SEG1 组合为一组，该组的长度可使用 t_{Seg_1} 进行配置。**表 7-11** 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 规范定义的更宽的配置范围。

表 7-11 CAN 控制器定时配置

配置项	需求			
t_{Seg_1}	[2…65] TQ		CAN 2.0 比特率	(慢)
t_{Seg_2}	[1…16] TQ	$t_{Seg\_1} \geq t_{Seg\_2} + 2$	CAN 2.0 比特率	(慢)
t_{SJW}	[1…16] TQ	$t_{Seg\_2} \geq t_{SJW}$	CAN 2.0 比特率	(慢)

对于 CAN 2.0 比特率标称的比特率，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。

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

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

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

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

需要执行如下步骤来初始化 CAN 控制器：

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

   在该示例中，总线数据速率为 1M 波特率，系统时钟为 48 MHz。

   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 7-12 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	2	24	18	6	4
0.8	75	3	20	15	5	4
0.5	75	4	24	18	6	4
0.25	75	8	24	18	6	4
0.125	75	16	24	18	6	4
0.1	75	20	24	18	6	4

表 7-13 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

8 局域互联网（LIN）

8.1 简介

局域互连网络控制器（LINCORE），符合 LIN 协议规范修订版 1.3，2.0，2.1 和 2.2，其工作在主机模式和从机模式，并自动判断错误。

8.2 功能列表

- 符合 LIN 协议规范 1.3，2.0， 2.1 和 2.2；
- 经典的校验和或增强的校验和可选；
- 独立的帧头及应答处理；
- 字段中最多 8 个数据字段；
- 检测显性电平以产生唤醒信号；
- 在从机模式下重新同步时钟；
- 在从机模式下具有 16 个标识符滤波器；
- 小数波特率发生器（Fractional baud rate generator）；
- 3 种省电及配置寄存器模式：
    - 初始模式
    - 正常模式
    - 休眠模式
- 两种测试模式
    - 环回
    - 自测
- 错误检测
    - 误码
    - 校验和错误
    - 帧超时错误
    - 物理总线错误
    - 帧错误
    - Overrun 错误

8.3 框图

LINCORE 架构如图 8-1 所示。

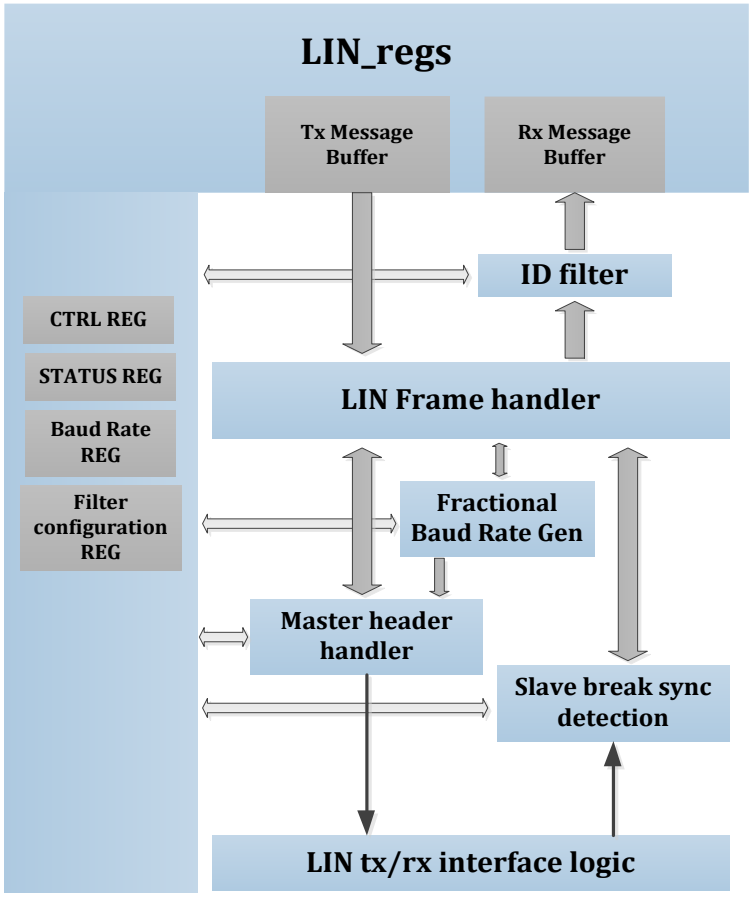

图 8-1 LINCORE 框图

8.4 使用说明

8.4.1 操作模式

LINCORE 主要有三种操作模式：初始化（Initialization）， 正常（Normal）和休眠（Sleep）模式。

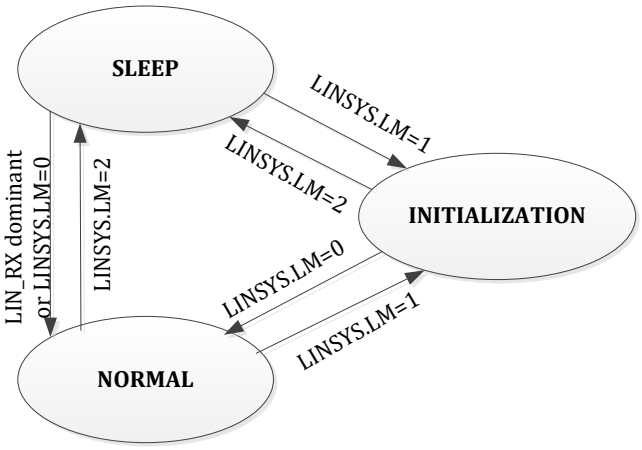

图 8-2 LINCORE 操作模式

8.4.1.1 初始化模式

软件将 LINSYS 中的 LM 置位 0x01 以进入初始化模式，将 LM 位置为其他值（0 或 2）以退出此模式。

在初始化模式下，软件可以设置工作模式（主机或从机模式），波特率和滤波功能。

8.4.1.2 正常模式

初始化完成后，通过将 LINSYS 中的 LM 位置为 0，软件进入正常模式。

8.4.1.3 休眠模式

将 LM 位的值置为 2，LINCORE 进入休眠模式以降低功耗。

如果使能自动唤醒模式，软件对 LM 位编程或 LINCORE 检测显性电平，可以退出睡眠模式。

8.4.2 测试模式

LINCORE 共有两种测试模式：环回模式和自测模式。

8.4.2.1 环回模式

在该模式下，LINCORE 执行一个从其 Tx 输出到其 Rx 输入的内部反馈。RX 输入引脚和 LINRX 引脚断开。在主模式下，LINCORE 可以接收其自身发送的消息，总线可以接收消息。

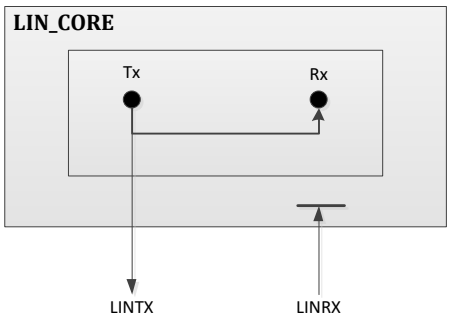

图 8-3 环回模式

8.4.2.2 自测模式

在该模式下，LINCORE 执行一个从其 Tx 输出到其 Rx 输入的内部反馈。RX 输入引脚和 LINRX 引脚断开。TX 输出引脚和 LINRTX 引脚断开。 在主模式下，LINCORE 可以接收其自身发送的消息，但总线不能接收消息。

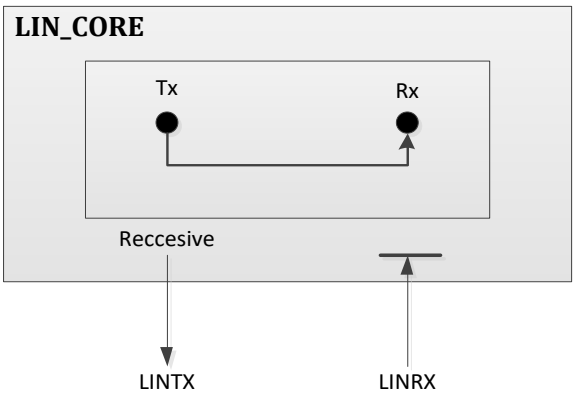

图 8-4 自测模式

8.4.3 波特率生成

使用 Mantissa 和 Fraction 寄存器，LINCORE 可以在多种波特率下工作。

$$\text{波特率} = \frac{F_{bus}}{（16 \times DIV）}$$

DIV 可由上边的公式计算得到，例如：

F_{bus} = 48MHz，若设置波特率 = 19200；

DIV = 156.25；

Mantissa = 156；

Fraction = 0.25 * 16 = 4；

实际波特率 = 19200，错误 = 0%。

8.4.4 错误检测

LINCORE 能够检测和处理 LIN 通信错误。LIN 错误状态寄存器（LINESTS） 出现错误后，触发错误中断。

- **位错误（Bit error）：**在发送过程中，从总线读回的值和发送值不同；
- **帧头错误（Header error）：**仅在从机模式下头部接收期间发生的错误（Break Delimiter,Synch Field, Parity ID 错误）；
- **帧错误（Framing error）：** 已经对当前接收字符的停止位进行采样的显性状态（sync field, identifier field 或 data field）；
- **校验和错误（Checksum error）：**计算的校验和和收到的校验和不匹配；

- **超时错误（Time out error）：** 检查的帧头超时（从同步字节停止位后开始到奇偶校验 ID 停止位结束），最小为 11，最大为 32。在响应时间空间内检查响应空间超时，最大超时值为 36。在包括响应空间（response space），数据字段（data field）和校验和字段（checksum field）的时间空间中检查响应超时，根据数据字段长度设置超时值。

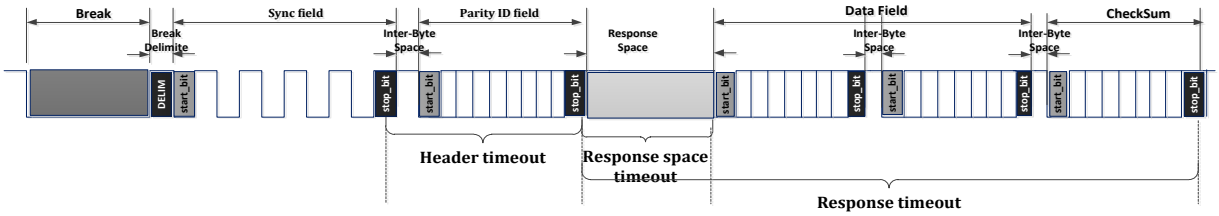

图 8-5 帧结构

8.4.5 中断表

表 8-1 LIN 中断表

中断事件	标志位	使能控制位
帧头发送/接收中断（Header Transmitted/Received interrupt）	HTRF	HTRIE
数据发送中断（Data Transmitted interrupt）	DTF	DTIE
数据接收中断（Data Received interrupt）	DRF	DRIE
唤醒中断（Wake-up interrupt）	WUF	WUIE
缓存溢出中断（Buffer Overrun interrupt）	BOF	BOIE
帧错误中断（Framing Error interrupt）	FEF	FEIE
帧头错误中断（Header Error interrupt）	IPEF, BDEF, SFEF	HEIE
校验和错误（Checksum Error interrupt）	CEF	CEIE
位错误中断（Bit Error interrupt）	BEF	BEIE
超时中断（Time out interrupt）	TOEF	TOIE
长零中断（Long Zero interrupt）	LZEF	LZIE

8.4.6 主机模式

当 LINCR1[SM] 位被清 0 时，选择主机模式。在主机模式下，应用程序处理来自调度表的消息。

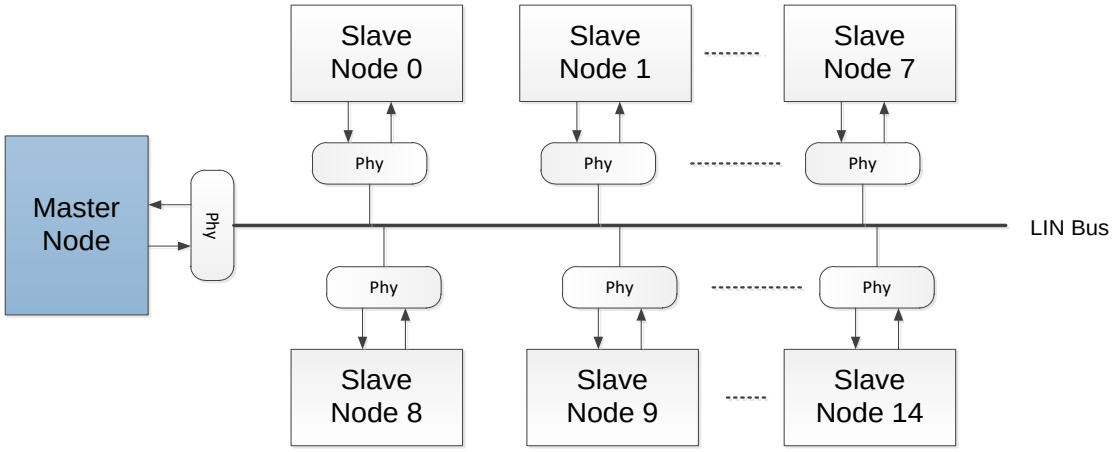

图 8-6 LIN 总线结构

8.4.6.1 帧头发送

在请求帧头发送前，应用程序必须设置标识符、数据字段长度并配置消息（应答方向、校验和类型）。

在发送帧头后，设置帧头发送标志 LINSTS [HRF]，如果 LINIER [HTRIE]使能，则产生中断。

8.4.6.2 应答发送

如果特殊标识符的应答方向是发布者，则应用程序必须在请求帧头传输之前填充数据缓冲区。LINCORE 根据数据字段长度以及校验和类型发送数据以及校验和。如果响应已成功发送，则设置 LINSTS [DTF]位，如果有错误，则不设置 DTF 标志。

8.4.6.3 应答接收

如果应答方向为接收者，在接收到具有相应标识符的帧头后，LINCORE 接收来自从节点的应答。如果应答已成功接收，则设置 LINSTS[DRF]。

8.4.7 从机模式

当设置 LINCR1[SM] 位时，选择从机模式。在从机模式下， LINCORE 接收来自主机模式的帧头，然后发送或接收响应。

8.4.7.1 帧头接收

在从机模式下， LINCORE 检测同步间隔，同步间隔分隔符，同步段和标识符。有效的中断字段比 10 或 11 位显性时间长。在每一个 LIN 中断分隔符后，LINCORE 检测 5 个下降沿，然后在启用重新同步功能时， 测量波特率。此功能使从机频率容差提升 ±14%。

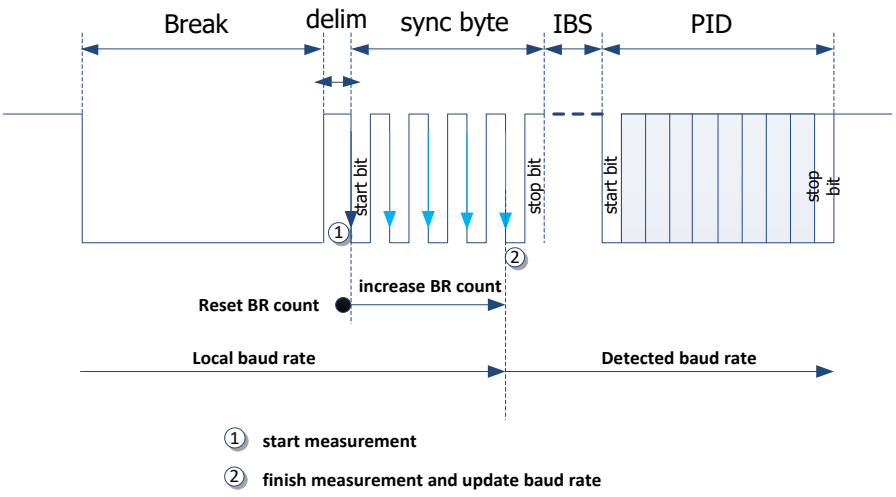

图 8-7 LIN 帧头接收

在同步字段（sync field）后，LINCORE 接收标识符，当禁用手动校验和时，检查奇偶校验数据。

8.4.7.2 应答发送

在接收到帧头后，应用程序检查标识符，如果应答方向为发布者，则通过设置 LINCR2[RTR]位，填充数据缓存，设置数据字段并触发数据传输。如果接收标识符来自标识符滤波器，软件仅需在传输前填充数据缓存。如果响应已经设置成功，则设置 LINSTS[DTF] 位。

8.4.7.3 应答接收

如果应答方向为接收者，则在帧头接收到相应的标识符后， LINCORE 接收其他节点发送的应答。 如果应答已经设置成功，则设置 LINSTS[DRF] 位。

8.4.7.4 滤波器

可以在过滤模式下配置 16 个标识符滤波器，每个滤波器标识一个标识符。可以在掩码模式下配置这些过滤器以处理更多标识符。在该模式下，奇数滤波器寄存器用作标识符寄存器，偶数滤波器寄存器用作掩码寄存器。掩码寄存器设置该位为 1 表示无关紧要，可忽略。

如果在掩码模式下使用屏蔽位配置滤波器，这意味着该滤波器可以接收更多的标识符。请确保这些标识符必须具有相同的响应方向，数据长度和校验和类型。

8.4.8 寄存器定义

表 8-2 LIN 寄存器映射

LIN： 0x40007000

地址	名称	说明
LIN + 0x000	**LINSYS**	LIN 系统控制寄存器
LIN + 0x004	**LINCTRL1**	LIN 控制寄存器 1
LIN + 0x008	**LIN_CTRL2**	LIN 控制寄存器 2
LIN + 0x00C	**LINIEN**	LIN 中断使能寄存器

地址	名称	说明
LIN + 0x010	**LINSTS**	LIN 状态寄存器
LIN + 0x014	**LINESTS**	LIN 错误状态寄存器
LIN + 0x018	**LINTO1**	LIN 超时控制寄存器 1
LIN + 0x01C	**LINTO2**	LIN 超时控制寄存器 2
LIN + 0x020	**LINIBR**	LIN 整数波特率寄存器
LIN + 0x024	**LINFBR**	LIN 小数波特率寄存器
LIN + 0x028	**LINCS**	LIN 校验和字段寄存器
LIN + 0x02C	**LINFRM**	LIN 帧控制寄存器
LIN + 0x030	**BDLR**	LIN 缓冲数据低寄存器
LIN + 0x034	**BDHR**	LIN 缓冲数据高寄存器
LIN + 0x038	**LIN_IFEN**	LIN ID 过滤使能寄存器
LIN + 0x03C	**LIN_IFMR**	LIN ID 过滤模式寄存器
LIN + 0x040	**LIN_IFMI**	LIN ID 过滤匹配索引寄存器
LIN + 0x044	**LIN_IFCR2n**	LIN ID 过滤控制寄存器 偏移量： 0x0044–0x007C（8 个寄存器）
LIN + 0x080	**LIN_IFCR2n+1**	LIN ID 过滤控制寄存器 偏移量： 0x0048–0x0080（8 个寄存器）

0x000 LINSYS LIN 系统控制寄存器

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称															**LM**	
类型															WR	
复位															**0x2**	

位	助记符	名称	说明
[1：0]	**LM**	LIN_MODE	0x0 ： 正常模式 0x1 ： 初始化模式 0x2 ： 休眠模式 0x3 ： 未定义模式

0x004 LINCTRL1 LIN 控制寄存器 1

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称									**DIOB**		**RIL**		**RSL**		**HIL**	
类型									WI_R		WI_R		WI_R		WI_R	
复位									0		0		0		0	
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	**BDL**	**CSM**	**RSE**	**AWU**	**BTL**							**BDT**	**SCM**	**STM**	**LBM**	**SM**
类型	WI_R	WI_R	WI_R	WI_R	WI_R							WI_R	WI_R	WI_R	WI_R	WI_R
复位	0	0	0	0	3							0	0	0	0	0

位	助记符	名称	说明
0	**SM**	从机模式	0： 主机模式 1： 从机模式
1	**LBM**	环回模式	0： 禁用 1： 使能
2	**STM**	自测模式	0： 禁用 1： 使能
3	**SCM**	软件控制模式	0： 禁用 1： 使能（不通过滤波器接收并处理消息）
4	**BLT**	同步间隔长度阀值	**从机模式下同步间隔段接收阈值** 0： 11-位 1： 10-位
[11：8]	**BTL**	同步间隔发送长度	**主机模式下同步间隔段发送长度** 0： 10-位时间 1： 11-位时间 …… 15： 25-位时间
12	**ATWU**	自动唤醒	**休眠模式下，该位控制 LIN 硬件行为** 0： 软件退出休眠模式; 1： LINRX 显性状态检测时， 硬件自动退出休眠模式
13	**RSE**	重新同步使能	**从机自动重新同步使能** 0： 禁用自动重新同步. 1：使能自动重新同步
14	**MCS**	校验和模式	**手动/硬件校验和模式** 0： 硬件自动校验和计算 1： 软件手动加载校验和
15	**BDL**	同步间隔分隔符长度	**同步间隔分隔符发送长度， 仅适用于主机模式** 0： 1-位 1： 4-位
[17：16]	**HIL**	帧头间隔长度	**同步段和 PID 段之间的间隔长度， 仅适用于主机模式** 0： 0-位时间 1： 2-位时间 2： 4-位时间 3： 8-位时间
[19：18]	**RSL**	应答间隔长度	**帧头和应答段之间的间隔长度， 仅适用于主机模式** 0： 0-位时间 1： 1-位时间 2： 4-位时间 3： 8-位时间
[21：20]	**RIL**	应答字节间隔	**应答数据字节之间的间隔长度** 0： 0-位时间 1： 1-位时间 2： 2-位时间 3： 4-位时间
[23]	**DIOB**	位错误时禁用 IDLE	**在检测到位错误时，LIN 状态机直接进入 IDLE 状态** 0： 位错误时使能 IDLE 1： 位错误时禁用 IDLE

0x008 LIN_CTRL2 **LIN 控制寄存器 2**

位	7	6	5	4	3	2	1	0
名称				**WTR**	**RDR**	**RTR**		**HTR**
类型				W_R0	W_R0	W_R0		W_R0
复位				0	0	0		0

位	助记符	名称	说明
0	**HTR**	帧头发送请求	**帧头发送请求（写 1）** 由软件设置来请求 LIN 帧头的发送 当请求完成或中止时，由硬件清零 **注意：当 LIN 内核处于正常模式时，该位可写（LINSYS.LM=0）**
2	**RTR**	应答发送请求	**数据发送请求（写 1）** 在从机模式下由软件设置，已请求传输存储在缓冲区数据寄存器的 LIN 数据字段。仅当 LINSTS 中的 HRF 位置 1 时，才能设置该位。 当请求完成或中止或出现错误时，由硬件清零。 在主机模式下，当 LINFRM[DIR] = 1 且帧头传输完成时，该位由硬件设置。 **注意：当 LIN 内核处于 正常模式时，该位可写（LINSYS.LM=0）**
3	**RDR**	应答中止请求	**应答中止请求（写 1）** 如果该帧与节点无关，则由软件设置以停止数据接收。 一旦 LIN 进入 idle 状态，该位由硬件复位。 在从机模式下，仅当 LINSTS 中的 HRF 位置 1 且标识符不匹配任何滤波器时，可以设置此位。 **说明： 当 LIN 内核处于正常模式时，此位可写（LINSYS.LM=0）**
4	**WUTR**	唤醒发送请求	**唤醒信号发送请求（写 1）** 设置此位会产生一个唤醒脉冲。当唤醒字符已经发送后，由硬件复位。发送的字符从 TX 缓冲区的 DATA0 复制而来。注意：该位不能在休眠模式下设置。在请求唤醒前，软件必须退出休眠模式。在发送唤醒请求时，不检查位错误。 **注意：当 LIN 内核处于正常模式时，该位可写（LINSYS.LM=0）**

0x00C LINIEN **LIN 中断使能寄存器**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	**LZIE**	**TOIE**	**BEIE**	**CEIE**	**HEIE**		**BOIE**	**FEIE**			**WUIE**			**DRIE**	**DTIE**	**HTRIE**
类型	RW	RW	RW	RW	RW		RW	RW			RW			RW	RW	RW
复位	0	0	0	0	0		0	0			0			0	0	0

位	助记符	名称	说明
0	**HRIE**	帧头接收中断使能	**帧头接收中断使能** 0： 收到有效的 LIN 帧头时，没有中断 1： 当接收到有效 LIN 帧头时产生中断，LINSTS 中的 HRF 位置 1。
1	**RTIE**	应答发送中断使能	**应答发送中断使能** 0： 当数据发送完成时没有中断； 1： 当设置 LINSTS 中的数据发送标志（DTF）时产生中断
2	**RRIE**	应答接收中断使能	**应答接收中断使能** 0： 当数据接收完成时，没有中断 1： 当设置 LINSTS 中的数据接收标志（DRF） 时产生中断

位	助记符	名称	说明
5	**WUIE**	唤醒中断使能	**唤醒中断使能** 0：当设置 LINSTS 中的 WUF 位时，没有中断 1：当设置 LINSTS 中的 WUF 位时产生中断
8	**FEIE**	帧错误中断使能	**帧错误中断使能** 0： 当发生帧错误时，没有中断 1： 当发生帧错误时产生中断
9	**BOIE**	缓冲区溢出中断使能	**缓冲区溢出中断使能** 0：当发生缓冲区溢出时，没有中断 1：当发生缓冲区溢出时产生中断
11	**HEIE**	帧头错误中断使能	**帧头错误中断使能** 0：当发生 Break Delimiter 错误、Synch Field 错误、Identifier field 错误时，没有中断 1：当发生 Break Delimiter 错误、Synch Field 错误、Identifier field 错误时，产生中断
12	**CEIE**	校验和错误中断使能	**校验和错误中断使能** 0：发生校验和错误时，没有中断 1：当设置 LINESTS 中的校验和错误标志（CEF） 时，产生中断
13	**BEIE**	位错误中断使能	**位错误中断使能** 0：当设置 LINESTS 中的 BEF 位时，没有中断 1：当设置 LINESTS 中的 BEF 位时，产生中断
14	**TOIE**	超时中断使能	**超时中断使能** 0： 当设置 TOEF 时，没有中断 1：当设置 TOEF 时，产生中断
15	**LZIE**	长零错误中断使能	**长零错误中断使能** 0： 当设置 LZEF 时，没有中断 1： 当设置 LZEF 时，产生中断

0x010 LINSTS **LIN 状态寄存器**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称					**LINS**					**RRIP**	**RTIP**	**RPS**	**WUF**	**DRF**	**DTF**	**HTRF**
类型					R					R	R	R	W1C	W1C	W1C	W1C
复位					0					0	0	0	0	0	0	0

位	助记符	名称	说明
0	**HRF**	帧头传输/接收标志	**帧头传输/接收标志** 该位由硬件设置，表示有效的帧头接收完成。 该位必须由软件清 0。 该位在初始化模式下和完成结束时由硬件复位。
1	**DTF**	数据发送完成标志	**数据发送完成标志** 该位由硬件设置，表示数据发送完成。 该位必须由软件和下一帧的帧头接收事件清零。 它在初始化模式下由硬件复位。
2	**DRF**	数据接收完成标志	**数据接收完成标志** 该位由硬件置位，表示数据接收已完成。 该位必须由软件清 0。 它在初始化模式下由硬件复位。 **注意：如果出现位错误或帧错误，不会设置此标志。**
3	**WUF**	唤醒标志	**唤醒标志**

位	助记符	名称	说明
			该位由硬件置位，且在如下情形下向软件表明 LIN 已经检测到 LINRX 引脚的下降沿： – 从机处于休眠模式； – 主机处于休眠模式或 idle 状态。 该位必须由软件清 0。其在初始化模式下由硬件复位。如果设置了 LINIER 中的 WUIE 位，会产生中断。
4	**PS**	LIN 接收引脚状态	**LIN 接收引脚状态** 处于诊断的目的，该位反映了 LINRX 引脚的当前状态
5	**RTIP**	发送正在进行	**发送正在进行** LIN 正在传输数据
6	**RRIP**	接收正在进行	**接收正在进行** LIN 正在接收数据
7	**RSV**	保留	保留
[11：8]	**STS**	LIN 模式状态	**LIN 模式状态** **0000： Idle** 当发生如下几个事件时，进入该状态： –LIN_SYS_CTRL 中的 SLEEP 位 和 INIT 位已经被软件清 0 – RX 引脚接收到下降沿，AWUM 位已置位 – 前面的帧接收/传输已经完成或中止。 **001： Break** 在从机模式下，检测到一个下降沿，其后为一个显性状态：接收中断（Receiving Break）。 **注意：在从机模式下，如果出现错误，新的 LIN 状态可以为 Idle 或 Break，具体取决于上一位的状态。如果上一位是显性的，则 新的 LIN 状态为 Break，否则为 Idle。** **在 Master 模式下，不断进行 Break 传输。** **0010： Break Delimiter** 在 Slave 模式下，已检测到中断长度配置 configuration（10-位 或 11-位）的有效中断。等待一个上升沿。 在 Master 模式下，已完成中断传输，正在进行 Break Delimiter 传输。 **0011： Synch Field** 在 Slave 模式下，已经检测到一个有效的 Break Delimiter（至少一比特时间隐形状态）， 接收 Synch Field。 在 Master 模式下，正在进行 Synch Field 传输。 **0100： Parity Identifier Field** 在 Slave 模式下，已经完成 有效 Synch Field 的接收，正在接收 Identifier Field。 在 Master 模式下，正在进行标识符传输。 **0101：Header reception/transmission completed** 在 Slave 模式下，已经接收一个有效的帧头，标识符字段可用。 在 Master 模式下，完成帧头发送。 **0110： Data reception/transmission Field** 正在进行应答接收/传输。 **0111： Checksum** 已完成数据接收/传输，正在进行校验和接收/传输 **1111： Wake up transmission field** 正在进行唤醒传输。

0x014 LINESTS **LIN 错误状态寄存器**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称							LZEF	TOEF	BEF	CEF	SFEF	BDEF	IPEF	FEF	BOF	NF
Type							W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset							0	0	0	0	0	0	0	0	0	0

位	助记符	名称	说明
0	**NF**	噪声标志	当在接收到的字符上检测到噪声时， 该位由硬件置位。 **注意： 该位需要由软件清除。**
1	**BOF**	缓冲区溢出标志	当接收到一新的数据字节时，该位由硬件置位。缓冲区满标志未被清除。 **注意：该位需要由软件清除。**
2	**FEF**	帧错误标志	该位由硬件置位，并向软件表明 LINCORE 已经检测到帧错误（无效停止位）。在接收应答字段（Master 模式或 Slave 模式）中的任何数据期间或在 Slave 模式下接收同步段或标识符段期间，可能会发生此错误。 **注意：该位需要由软件清除。**
3	**IPEF**	标识符奇偶校验错误标志	该位由硬件置位，且表明发生标识符奇偶校验错误。 **注意： 当置位 SFEF 或 BDEF 或 IPEF，且 LINIEN 中的 HEIE 位被设置时，触发帧头中断。**
4	**BDEF**	同步间隔分隔符错误标志	该位由硬件置位，且表明接收同步间隔分隔符太短（不足一位时间）。 **注意：该位需要由软件清除。**
5	**SFEF**	同步段错误标志	该位由硬件置位，且表明发生同步错误（不一致的 Synch Field）。 **注意：该位需要由软件清除。**
6	**CEF**	校验和错误标志	该位由硬件置位，且表示接收校验和和硬件计算得出的校验和不匹配。 **注意：该位需要由软件清除。**
7	**BEF**	位错误标志	该位由硬件置位，并向软件表明 LINC 已经检测到位错误。该错误在应答字段发送（Slave 和 Master 模式）期间或在帧头发送（在 Master 模式下）期间发生。 **注意：该位需要由软件清除。**
8	**TOEF**	超时错误标识	超时错误事件发生在帧头超时或应答超时。 **注意：该位需要由软件清除。**
9	**LZEF**	长零错误标识	当总线在超过 256 位时间处于显性状态时，该位由硬件中断。如果显性状态继续，该位由软件清除。

0x018 LINTO1 **LIN 超时控制寄存器**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称											HTO					
类型											WI_R					
复位											0x18					

位	助记符	名称	说明
[5：0]	**HTOV**	帧头超时值	该字段包括帧头超时长度（单位：位时间）。该位不包括 Break、Break Delimiter 和 sync field。 **注意： 如果被写入的值小于或等于 11， 该值将被记为 11.**

0x01C LINTO2 **LIN 超时控制寄存器**

位	7	6	5	4	3	2	1	0
名称								**TOEN**
类型								WI
复位								0

位	助记符	名称	说明
0	**TOEN**	超时使能	**超时使能位** 0： 禁用 1： 使能

0x020 LINIBR **LIN 整数波特率寄存器**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称					**IBR**											
类型					WR											
复位					0											

位	助记符	名称	说明
[11：0]	**IBR**	整数波特率除数值	**整数波特率除数值** 该字段仅在初始化模式下可写。

0x024 LINFBR **LIN 小数波特率寄存器**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称													**FBR**			
类型													WI_R			
复位													0			

位	助记符	名称	说明
[3：0]	**FBR**	小数波特率除数值	**小数波特率除数值** 该字段仅在初始化模式下可写。

0x028 LINCS **LIN 校验和字段寄存器**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									**CS**							
类型									RW							
复位									0							

位	助记符	名称	说明
[7：0]	**CS**	校验和字段	**校验和字段** 在自动校验和模式下只读。 若置位 LINCTRL1.CSM 位，该位在手动校验和模式下可读写。

0x02C LINFRM **LIN 帧控制寄存器**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称				DFL			DIR	CST			ID					
类型				RW			RW	RW			RW					
复位				0			0	0			0					

位	助记符	名称	说明
[5：0]	**ID**	标识符	identifier 字段的标记符部分，没有标识符奇偶校验。
8	**CST**	校验和类型	该位所应用于目前消息的校验和类型 0：增强校验和，包括标识符和数据字段。这和 LIN 规范 2.0 或更高的版本相兼容。 1：经典校验和，仅包括数据字段。这和 LIN 规范 1.3 或更早的版本相兼容。
9	**DIR**	应答方向	该位控制数据字段的方向。 0： LIN 接收数据，并将其复制到 BDRL 和 BDRH 寄存器。 1：LIN 从 BDRL 和 BDRH 寄存器中传输数据。
[12：10]	**DFL**	数据字段长度	该字段定义帧的应答部分中的数据字节数。取值范围 0-7 表示 1 - 8 个字节长度，主从机都需要配置该寄存器。

0x030 BDLR **LIN 缓冲区数据低寄存器**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	DATA3								DATA2							
类型	RW								RW							
复位	0								0							
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	DATA1								DATA0							
类型	RW								RW							
复位	0								0							

0x034 BDHR **LIN 缓冲区数据高寄存器**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	DATA7								DATA6							
类型	RW								RW							
复位	0								0							
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	DATA5								DATA4							
类型	RW								RW							
复位	0								0							

0x038 LIN_IFEN **LIN ID 滤波器使能寄存器**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	EN															
类型	RW															
复位	0															

位	助记符	名称	说明
[15：0]	**EN**	滤波器编号使能	滤波器编号从 0 到 15. 在 list 模式下，每个 EN 对应一个滤波器 在 mask 模式下， EN [2n+1] 对相应的滤波器没有影响，因

位	助记符	名称	说明
			为它们充当滤波器 2n 的掩码。 0： 滤波器已停用 1： 滤波器已激活

0x03C LIN_IFMR **LIN ID 滤波器模式寄存器**

位	7	6	5	4	3	2	1	0
名称	IFM							
类型	RW							
复位	0							

位	助记符	名称	说明
[7：0]	**IFM**	ID 滤波器模式	0： 滤波器 2n 和 2n + 1 处于标识符滤波模式； 1： 滤波器 2n 和 2n + 1 处于掩码 模式（滤波器 2n + 1 是滤波器 2n 的掩码）。 **注意： “n” 的范围从 0 到 7，对应于 IFM 的位索引。**

0x040 LIN_IFMI **LIN ID 滤波器匹配索引寄存器**

位	7	6	5	4	3	2	1	0
名称				IFMI[4：0]				
类型				RW				
复位				0				

位	助记符	名称	说明
[7：0]	**IFMI**	ID 滤波器匹配索引	此寄存器包含对应于接收标识符的索引。它可用于直接写入或读取 SRAM 中的数据。当没有滤波器匹配时，IFMI[4：0] = 0。当滤波器 n 匹配时， IFMI[4：0] = n + 1。

0x044 LIN_IFCR2n LIN ID 滤波器控制寄存器 **偏移量： 0x0044–0x007C（8 寄存器）**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称				DFL			DIR	CST			ID					
类型				RW			RW	RW			RW					
复位				0			0	0			0					

位	助记符	名称	说明
[5：0]	**ID**	标识符	identifier 字段的标记符部分，没有标识符奇偶校验。
8	**CST**	校验和类型	设置校验和类型 0： 增强校验和，包括标识符和数据字段。这和 LIN 规范 2.0 或更高的版本相兼容。 1： 经典校验和，仅包括数据字段。这和 LIN 规范 1.3 或更早的版本相兼容。
9	**DIR**	应答方向	设置数据字段的方向 0： LIN 接收数据，并将其复制到 BDRL 和 BDRH 寄存器。 1： LIN 从 BDRL 和 BDRH 寄存器中传输数据。
[12：10]	**DFL**	数据字段长度	定义帧的应答部分中的数据字节数。取值范围 0-7 表示 1 - 8 个字节长度，主从机都需要配置该寄存器。

0x080 LIN_IFCR2n+1 LIN ID 滤波器控制寄存器 偏移量： 0x0048–0x0080（8 寄存器）

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称				**DFL**			**DIR**	**CST**			**ID**					
类型				RW			RW	RW			RW					
复位				0			0	0			0					

位	助记符	名称	说明
[5：0]	**ID**	标识符	identifier 字段的标记符部分，没有标识符奇偶校验。
8	**CST**	校验和类型	设置校验和类型 0 ：增强校验和，包括标识符和数据字段。这和 LIN 规范 2.0 或更高的版本相兼容。 1 ：经典校验和，仅包括数据字段。这和 LIN 规范 1.3 或更早的版本相兼容。
9	**DIR**	应答方向	设置数据字段的方向 0 ： LIN 接收数据，并将其复制到 BDRL 和 BDRH 寄存器。 1 ：LIN 从 BDRL 和 BDRH 寄存器中传输数据。
[12：10]	**DFL**	数据字段长度	定义帧的应答部分中的数据字节数。取值范围 0-7 表示 1 - 8 个字节长度，主从机都需要配置该寄存器。

9 通用异步收发器（UART）

9.1 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许多功能。主要功能如表 9-1 所示。

表 9-1 功能分类和配置

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

【说明】

1. 只有 UART1 支持硬件流的控制功能；只有 UART6 支持软件 LIN 的功能。
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。

9.2 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
  - 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 位数据长度， 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 位分隔符，可选 10 或 11 位 LIN 功能分隔符检测；
- 支持 RS485 自动控制方向。

9.3 结构框图

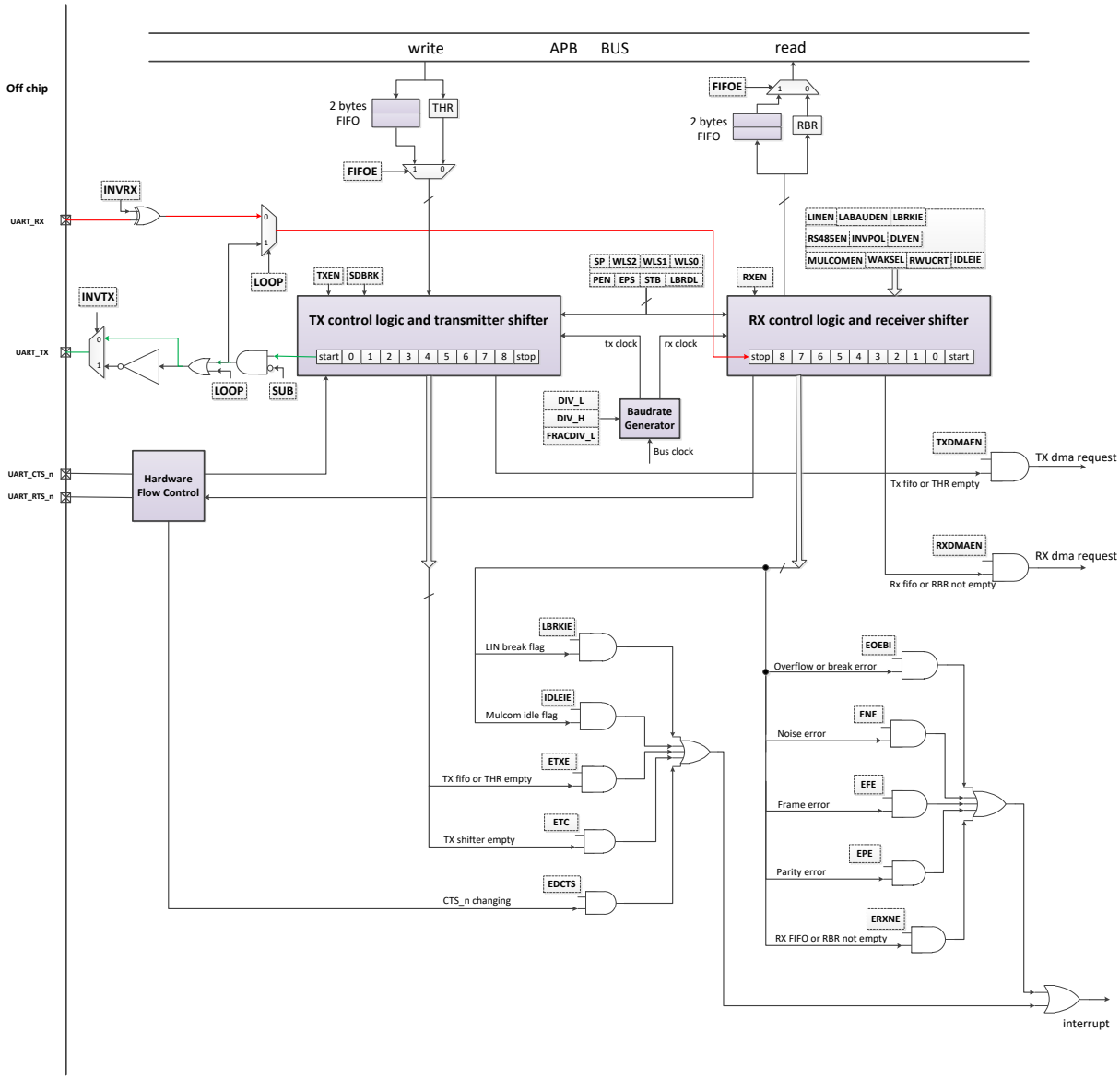

图 9-1 UART 结构框图

9.4 输入输出定时

表 9-2 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

引脚名	相应信号	宽度	输入/输出	是否上拉	定时限制
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= 1~ 6.。仅 UART1 具有完整的硬件流控制，其四个信号（UART1_TX, UART1_RX, UART1_RTS, UART1_CTS） 都可以在引脚中找到。

9.5 UART 功能

UART 功能是逐位发送和接收串行数据。图 9-2 和图 9-3 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 UARTn_LCR0 和 UARTn_LCR1。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 9-2 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。

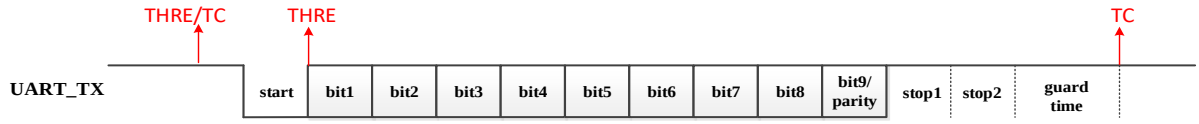

图 9-2 UART 传送器流程

如果在接收过程中发生相关事件，如图 9-3 所示，则 状态位 DR，OE，PE，FE，BI 和 NE 会在 stop1 位后立即置为 1。

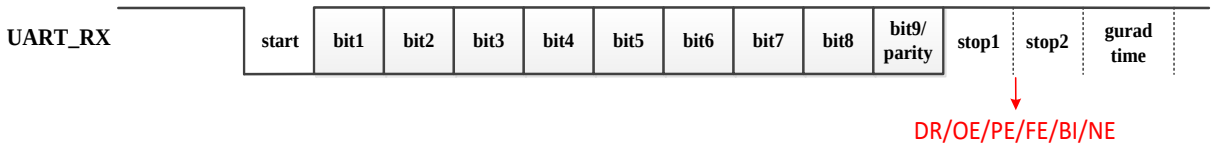

图 9-3 UART 接收器流程

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。

9.5.1 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 9-4 所示。

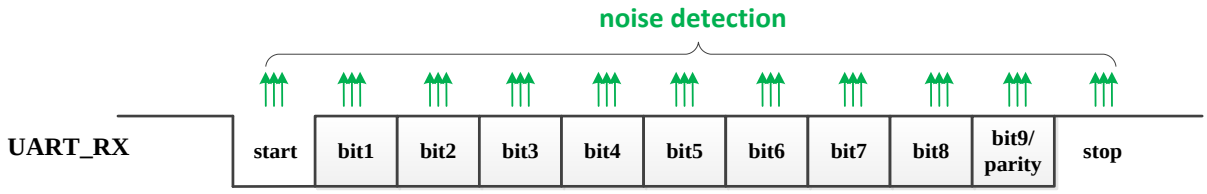

图 9-4 UART 噪声检测

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。

9.5.2 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 9-3 和表 9-4 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 9-3 典型的波特率及误差率@bclock=50MHz

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	1302	3	16	-0.001%
2	9.6	9599.69	325	17	16	-0.003%
3	19.2	19202.22	162	24	16	0.006%
4	57.6	57603.69	54	8	16	0.006%
5	115.2	115207.38	27	4	16	0.006%
6	230.4	230414.75	13	18	16	0.006%
7	460.8	460829.50	6	25	16	0.006%
8	921.6	917431.19	3	13	16	-0.452%
9	1843.2	1834862.38	3	13	8	-0.452%
10	4200	4166666.75	1	16	8	-0.794%

表 9-4 典型的波特率及误差率@bclock=36MHz

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	937	16	16	0
2	9.6	9600	234	12	16	0
3	19.2	19200	117	6	16	0
4	57.6	57600	39	2	16	0
5	115.2	115200	19	17	16	0
6	230.4	230769.23	9	24	16	0.16%
7	460.8	461538.47	4	28	16	0.16%
8	921.6	923076.94	2	14	16	0.16%
9	1843.2	1846153.88	1	7	16	0.16%

10	4200.0	-	-	-	-	-

9.6 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 9-5 所示。

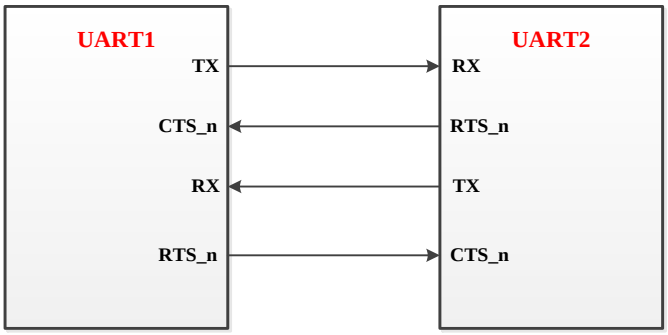

图 9-5 硬件流控制连接

在图 9-5 中，当 UART1 RX 数据寄存器或 FIFO 已满，UART1 的信号 RTS_n 可以通知 UART2 不发送数据。当 UART1 RX 数据寄存器或 FIFO 通过读操作变为未满时， UART1 RTS_n 自动变为低电平。然后 UART2 可以通过检测 UART2 CTS_n 为低电平来传输数据。以类似的方式，UART2 RTS_n 和 UART1 CTS_n 用于控制 UART1 传输。

特别地，如果在 UART2 传输数据期间，UART2 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。

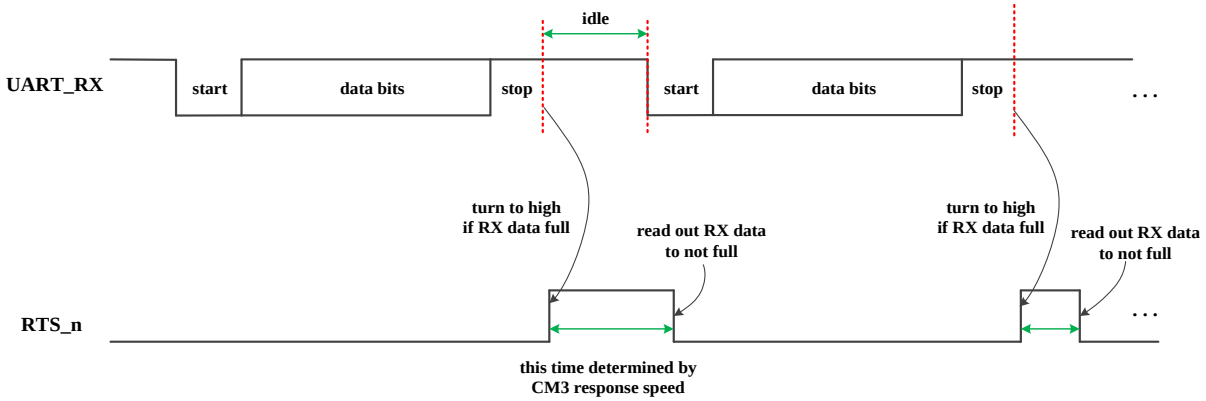

图 9-6 硬件流控制原理

9.7 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 9-7 和图 9-8 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 9-7 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。.

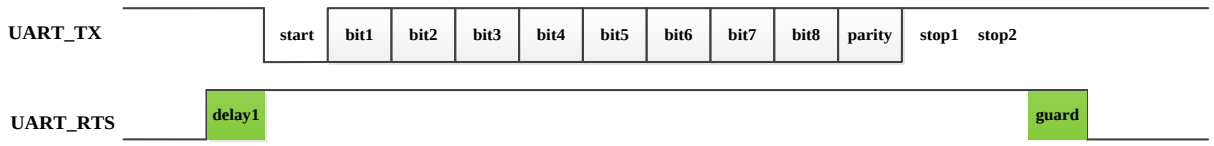

图 9-7 单字节数据传输

图 9-8 多字节数据传输

图 9-9 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 的默认值为低电平，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。

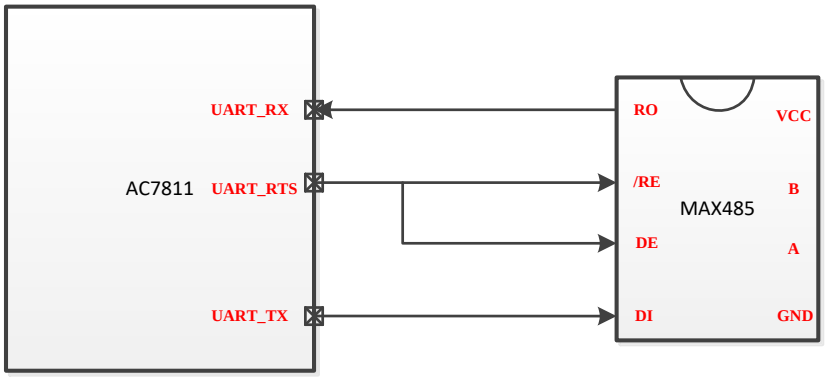

图 9-9 实际电路连接

9.8 LIN 功能

UART LIN 是一个软件 LIN ，只有 UART6 支持软件 LIN 功能。软件 LIN 具有传输分隔符（break field），同步域（synchronous field）和 数据。如图 9-10 所示。用户可以在最新的 LIN 协议中了解更多细节，该图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 UARTn_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。

首先，根据 UART_RX 上出现的串行数据流介绍接收过程，如图 9-10 所示。首先，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 RX 数据寄存器或 FIFO 中。在分隔符位周期之后，同步域即 0x55 当做正常的数据接收。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 9-10 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。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 9-10 所示，对于 break field 超过 25 位时间，对于 delimiter 超过 14 位时间和对于 synchronous field 超过 13 位时间，可以导致模块重置接收器和 LIN 检测逻辑。

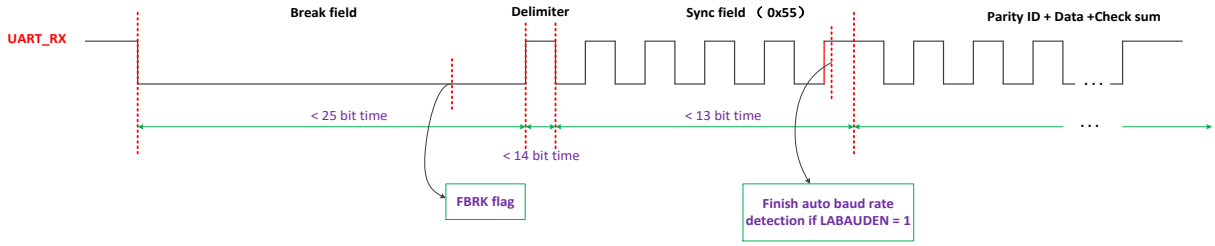

图 9-10 LIN 帧流程

本段描述了传输 LIN 协议。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 UARTn_LSR0 [THRE]的状态。如果 UARTn_LSR0[THRE]值为 1，用户可以向 UARTn_LINCR [SBRK]写入 1 以发送 13 位时间间隔字段，其中 LINCR 中其他控制位中未变化。需要指出的是，当 UARTn_LINCR [SBRK]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。UARTn_LINCR [SBRK]将由硬件自动清除。然后可以将同步域（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。

9.9 两种电源模式

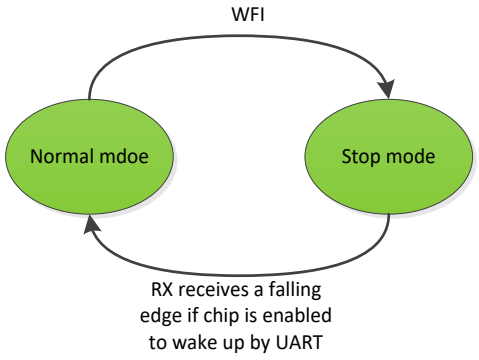

图 9-11 运行模式（Run mode）和停止模式（Stop mode）

本节介绍在两种电源模式下 UART 的状态。如图 9-11 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 模块将关闭并默认重置。当然，所有 UART 配置和其他寄存器都会在停止模式条件下复位。

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，并至少从接收唤醒下降沿重新配置 UART 时间。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。

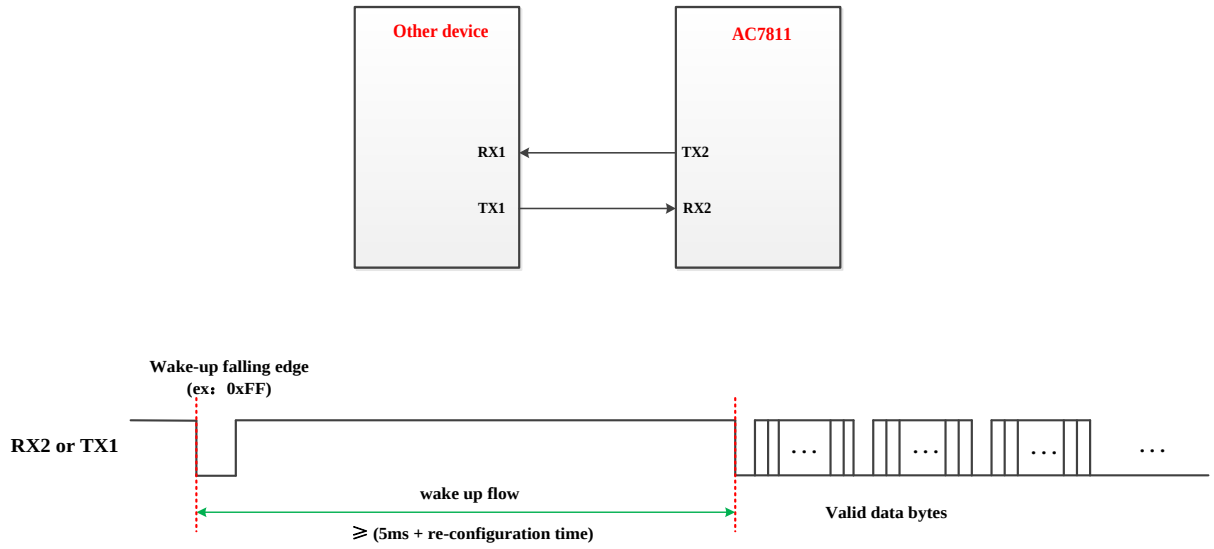

图 9-12 通过 UART 唤醒芯片的典型流程

9.10 波特率配置说明

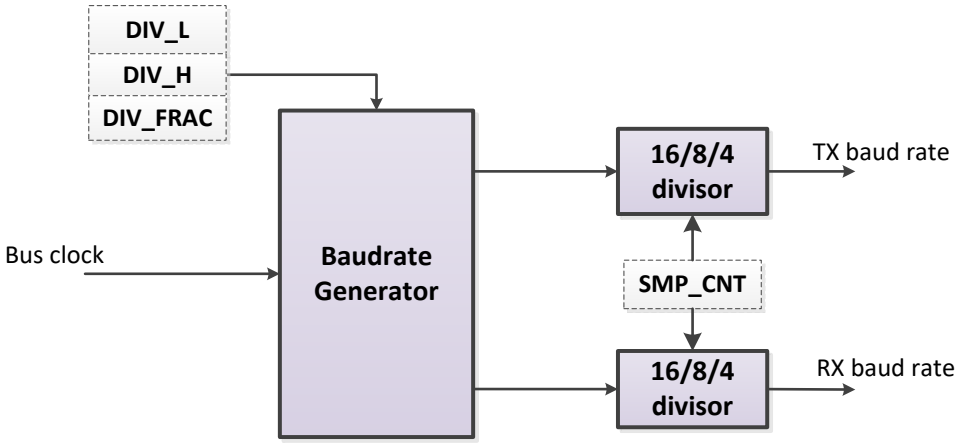

图 9-13 波特率发生器框图

如图 9-13 所示，波特率相关的配置寄存器如下所示：UARTn_SMP_CNT/UARTn_DIV_H/UARTn_DIV_L /UARTn_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。

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

Note： DIV = {UARTn_DIV_H, UARTn_DIV_L} ; SMP_CNT = UARTn_SMP_CNT。

例如：

如果用户希望在 8 倍采样模式下以 50MHz 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， **UARTn_DIV_L=13， UARTn_DIV_H=0， UARTn_DIV_FRAC=32 * 0.5634=18。**

$$DIV = \frac{50000000}{Baudrate * SMP\_CNT} = \frac{50000000}{230400 * 8} = 13.5634$$

结合上面给出的例子， 可以清晰地解释 UARTn_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，UARTn_DIV_FRAC [4：0]配置为表示 DIV 小数部分。由于宽度为 5 位，UARTn_DIV_FRAC 取值范围是 0 至 31。因此，32 乘以 DIV 小数部分生成 UARTn_DIV_FRAC [4：0]的配置值。

9.11 配置说明

配置步骤：

(1) TX/RX 数据存储模式： UARTn_FCR

(2) 波特率： UARTn_DIV_L/UARTn_DIV_H/ UARTn_DIV_FRAC/UARTn_SMP_CNT

(3) DMA： UARTn_DMA_EN

注意： DMA 功能必须在 FIFO 模式下才能起作用。

(4) 数据格式： UARTn_LCR0/UARTn_LCR1

注意： 务必注意 SB 位。

(5) 功能配置： UARTn_RS485CR/UARTn_LINCR/UARTn_MULCOMCR/ UARTn_CNRT / UARTn_SLADDR 等。

注意： 本步骤对不同功能来说是可选的。

(6) 中断使能： UARTn_IER

(7) 收发器使能：UARTn_LCR1[TXEN] / UARTn_LCR1[RXEN]

(8) 发送或接收数据：UARTn_THR/UARTn_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。

9.12 寄存器定义

表 9-5 UART 寄存器映射

Address = 基地址（Base address） + 偏移地址（Offset address）

模块名	基地址	偏移地址	说明
UART1	0x40018000	0x00 ~ 0x58	1. 没有 UARTn_LCR 2. 部分寄存器是相反的
UART2	0x40019000	0x00 ~ 0x58	
UART3	0x4001a000	0x00 ~ 0x58	
UART4	0x4001b000	0x00 ~ 0x58	
UART5	0x4001c000	0x00 ~ 0x58	
UART6 （UART_LIN）	0x4001d000	0x00 ~ 0x5c	1. UARTn_LCR. 2. 部分寄存器是相反的

偏移地址	寄存器名称	宽度	寄存器功能
0x00	**UARTn_RBR/THR**	32	TX 保持寄存器 /RX 缓冲区寄存器
0x04	**UARTn_DIV_L**	32	分频器低 8 位
0x08	**UARTn_DIV_H**	32	分频器高 8 位
0x0C	**UARTn_LCR0**	32	UART 辅助控制寄存器 0
0x10	**UARTn_LCR1**	32	UART 辅助控制寄存器 1
0x14	**UARTn_FCR**	32	FIFO 控制寄存器
0x18	**UARTn_EFR**	32	硬件流程使能寄存器
0x1c	**UARTn_IER**	32	中断使能寄存器
0x20	**UARTn_LSR0**	32	状态寄存器 0
0x24	**UARTn_LSR1**	32	状态寄存器 1
0x28	**UARTn_SMP_CNT**	32	UART 采样计数寄存器
0x34	**UARTn_GUARD**	32	保护时间（Guard time） 添加寄存器
0x38	**Reserved**	32	
0x3c	**UARTn_SLEEP_EN**	32	休眠使能寄存器
0x40	**UARTn_DMA_EN**	32	DMA 使能寄存器
0x44	**UARTn_DIV_FRAC**	32	小数分频器寄存器
0x48	**Reserved**	32	
0x4c	**UARTn_RS485CR**	32	RS485 控制寄存器
0x50	**UARTn_SLADDR**	32	多机通信唤醒地址
0x54	**UARTn_CNTR**	32	RS485 延迟时间

偏移地址	寄存器名称	宽度	寄存器功能
0x58	UARTn_MULCOMCR	32	空闲中断使能寄存器
0x5c	UARTn_LINCR	32	软件 LIN 控制寄存器

0x00　　UARTn_RBR/THR　　RX/TX 数据寄存器　　0000

位	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称								RBR/THR								
类型								RO/WO								
复位								0	0	0	0	0	0	0	0	0

位	助记符	名称	说明
8：0	**RBR/THR**	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。

0x04　　UARTn_DIV_L　　分频器低 8 位寄存器　　0001

位	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									DIV_L							
类型									RW							
复位									0	0	0	0	0	0	0	1

位	助记符	名称	说明
7：0	**DIV_L**	DIV_L	**波特率分频器** 分频器低 8 位。

0x08　　UARTn_DIV_H　　分频器高 8 位寄存器　　0000

位	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									DIV_H							
类型									RW							
复位									0	0	0	0	0	0	0	0

位	助记符	名称	说明
7：0	**DIV_H**	DIV_H	**波特率分频器** 分频器高 8 位。

0x0C **UARTn_LCR0** **控制寄存器 0** **0000**

位	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称										**SUB**	**SP**	**EPS**	**PEN**	**STB**	**WLS1_WLS0**	
类型										RW	RW	RW	RW	RW	RW	
复位										0	0	0	0	0	0	0

注意：一定要将 SUB 位配置位 0，否则 tx 在任何时候都发送 '0'。

位	助记符	名称	说明
6	**SUB**	SUB	**设置 Break** 0： 没有效果 1： SOUT 信号被强制进入 "0" 状态
5	**SP**	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**	EPS	**选择偶校验** 0： 当 EPS = 0，发送并检查奇数个 1 1： 当 EPS = 1, 发送并检查偶数个 1
3	**PEN**	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
2	**STB**	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**	WLS1_WLS0	**选择字长** 0： 5 位 1： 6 位 2： 7 位 3： 8 位 4： 9 位 （wls2）

0x10 **UARTn_LCR1** **控制寄存器 1** **0000**

位	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									**INVTX**	**INVRX**	**WLS2**	**LOOP**			**TXEN**	**RXEN**
类型									RW	RW	RW	RW			RW	RW
复位									**0**	**0**	**0**	**0**			**0**	**0**

位	助记符	名称	说明
7	**INVTX**	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**	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**	WLS2	**确定 9 位数据模式是否可用** 0： 不可用 1： 可用
4	**LOOP**	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**	TXEN	**UART 发射器使能** 0： 禁用 1： 使能
0	**RXEN**	RXEN	**UART 接收器使能** 0： 禁用 1： 使能

0x14 **UARTn_FCR** **FIFO 控制寄存器** **0000**

位	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称																**FIFOE**
类型																RW
复位																0

位	助记符	名称	说明
0	**FIFOE**	FIFOE	**Enables FIFO** 0： 禁用 RX 和 TX FIFO 1： 使能 RX 和 TX FIFO

0x18 **UARTn_EFR** **0000**

位	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									**AUTO_CTS**	**AUTO_RTS**						
类型									RW	RW						
复位									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。

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

位	助记符	名称	说明
			1： 使能

0x1c **UARTn_IER** **中断使能寄存器** **0000**

位	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									**EDCTS**	**EOEBI**	**ENE**	**EFE**	**EPE**	**ETC**	**ETXE**	**ERXNE**
类型									RW	RW	RW	RW	RW	RW	RW	RW
复位									0	0	0	0	0	0	0	0

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

0x20 **UARTn_LSR0** **线路状态寄存器** **0020**

位	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									**NE**	**TC**	**THRE**	**BI**	**FE**	**PE**	**OE**	**DR**
类型									W1C	R	R	W1C	W1C	W1C	W1C	R
复位									0	0	1	0	0	0	0	0

注意： 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。

位	助记符	名称	说明
7	**NE**	NE	**噪声错误标志** **注意：写 1 将此标志清除为 0.** 0：不存在噪声错误 1： 存在噪声错误
6	**TC**	**TC**	**传输完成标志** **注意： 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.** 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。
5	**THRE**	THRE	**TX 保持寄存器或 TX FIFO 空标志** **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.** 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.
4	**BI**	BI	**分隔符错误标志** **注意：写 1 将此标志清除为 0.** 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。
3	**FE**	FE	**帧错误标志** **注意：写 1 将此标志清除为 0.** 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。
2	**PE**	**PE**	**奇偶错误标志** **注意：写 1 将此标志清除为 0.** 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。
1	**OE**	OE	**溢出错误标志** **注意：写 1 将此标志清除为 0.** 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。
0	**DR**	DR	**数据就绪标志** **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.** 0：接收数据未就绪。

位	助记符	名称	说明
			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）。

0x24　　UARTn_LSR1　　线路状态寄存器　　00E0

位	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									RTS	CTS	UART_IDLE		DCTS	FBRK		IDLE
类型									R	R	R		R	R		R
复位									1	1	1		0	0		0

位	助记符	名称	说明
7	**RTS**	RTS	**硬件流程状态 – RTS** **注意： RTS 和 RTS_n 引脚信号相反，它不是一个中断源。** 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。
6	**CTS**	CTS	**硬件流程状态 – CTS** **注意： CTS 和 CTS_n 引脚信号相反，它不是一个中断源。** 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。
5	**UART_IDLE**	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。
3	**DCTS**	DCTS	**CTS_n 引脚信号变化标志** **注意：写 1 将此标志清除为 0.** 0： 未变化。 1：表示 CTS_n 引脚信号从 1 变为 0 或 0 变为 1。
2	**FBRK**	FBRK	**LIN BREAK 发生标志** **注意：写 1 将此标志清除为 0.** 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。
0	**IDLE**	IDLE	**IDLE 标志** 多机通讯功能 MULCOMEN 先使能，接收器已经接收到数据，该数据后面至少保持一个字节数据时间的高电平，IDLE 状态标志位置位。 **注意：写 1 将此标志清除为 0.** 0：尚未检测到空闲线路 1：检测到空闲线路

0x28　　UARTn_SMP_CNT　　采样计数器寄存器　　0000

位	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称															**SMP_CNT**	
类型															RW	
复位															0	0

位	助记符	名称	说明
1：0	**SMP_CNT**	SMP_CNT	**UART 采样计数器** 0： 基于 16*baud_pulse, baud_rate = 系统时钟频率/16/{DLH, DLL} 1： 基于 8*baud_pulse, baud_rate =系统时钟频率/8/{DLH, DLL} 2： 基于 4*baud_pulse, baud_rate =系统时钟频率/4/{DLH, DLL} 3： 基于 sampe_count * baud_pulse, baud_rate =系统时钟频率/16 /{DLM, DLL}

0x34　　UARTn_GUARD　保护时间寄存器　　000F

位	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称												**GUARD_EN**	**GUARD_CNT**			
类型												RW	RW			
复位												0	1	1	1	1

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。

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

0x3C　　UARTn_SLEEP_EN　休眠使能寄存器　　0000

位	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称																**SLEEP_EN**
类型																RW
复位																0

位	助记符	名称	说明
0	**SLEEP_EN**	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。

0x40　　UARTn_DMA_EN　　0000

位	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称															TX_DMA_EN	RX_DMA_EN
类型															RW	RW
复位															0	0

位	助记符	名称	说明
1	**TX_DMA_EN**	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_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

0x44　　UARTn_DIV_FRAC　　小数分频器地址　　0000

位	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称												DIV_FRAC				
类型												RW				
复位												0				

位	助记符	名称	说明
4：0	**DIV_FRAC**	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 。

0x4c　　UARTn_RS485CR　　RS485 控制寄存器　　0000

位	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									RS485EN		INVPOL	DLYEN				
类型									RW		RW	RW				
复位									0		0	0				

注意：

1.　发送前的延迟对应于 UARTn_CNTR ，发送后的延迟可以使用 UARTn_GUARD.

2.　RS485 功能使用 PIN RTS_n 作为发送或接收方向控制引脚。

位	助记符	名称	说明
7	**RS485EN**	RS485EN	0：禁用 rs485 模式 1：使能 rs485 模式
5	**INVPOL**	INVPOL	**INVPOL** 0： 不反转 rosin 的极性 1： 反转 rts_n 的极性
4	**DLYEN**	DLYEN	在 RS485 切换为输出状态到真正开始发送 START 位之间插入

位	助记符	名称	说明
			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 9-7 的 dalay1。 0：禁用延时 1：使能延时

0x50　　UARTn_SLADDR　　多机通讯唤醒地址　　0000

位	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称													**Bit3**	**Bit2**	**Bit1**	**Bit0**
类型													RW	RW	RW	RW
复位													0	0	0	0

位	助记符	名称	说明
3：0	**SLADDR**	ADDR	**SLADDR** 3：0：当 mulcomen =1，且主机发送的帧的最高位为 1，多机通讯的唤醒地址才有效。如：多机通讯的地址 SLADDR = 0x04, 帧为 8 bits 宽度，则主机需要发送 0x84 才能唤醒。地址匹配后，0x84 会作为数据存储在 RX 数据寄存器。

0x54　　UARTn_CNTR　　0000

位	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									**CNTR[7：0]**							
类型									RW	RW	RW	RW	RW	RW	RW	RW
复位													0	0	0	0

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

0x58　　UARTn_MULCOMCR　　空闲中断使能寄存器　　0000

位	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									MULCOMEN			**IDLEIE**				
类型									RW			RW				
复位									**0**			**0**				

位	助记符	名称	说明
7	**MULCOMEN**	MULCOMEN	**多机通信使能** 0：　禁用 1：　使能
4	**IDLEIE**	IDLEIE	**IDLE 中断使能** 0：禁用 1：　使能

注：使能 IDLE 中断必须同时使能多机通信

0x5c **UARTn_LINCR** **LIN 控制寄存器** **0000**

位	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									**LINEN**	**LBRKIE**	**LBRKDL**	**SDBRK**	**LABAUDEN**			
类型									RW	RW	RW	RW	RW			
复位									**0**	**0**	**0**	**0**	**0**			

位	助记符	名称	说明
7	**LINEN**	LINEN	**LIN 模式使能** 0： 禁用 1： 使能
6	**LBRKIE**	LBRKIE	**LIN Break 字符检测中断使能** 0： 禁用 1： 使能
5	**LBRKDL**	LBRKDL	**LIN 模式中断检测长度** 0： 10 位 1： 11 位
4	**SDBRK**	SDBRK	**LIN 模式是否传输 13 个 0** **注意：由软件设置，并在 break 的停止位期间由 MCU 内部硬件清除。** 0： 禁用 1： 立即支持传输 13 ‘0’
3	**LABAUDEN**	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

10 模数转换器（ADC）

10.1 ADC 简介

ADC（Analog-to-Digital Converter），指模/数转换器或者模数转换器。是指将连续变化的模拟信号转换为离散的数字信号的器件。用于信息处理，计算，数据传输和控制。ADC 可工作在运行（Run）模式和停止（Stop）模式下。

10.2 ADC 特性

- ADC 通道输入电压范围： AVSS < Vin < AVDD；
- 最大转换速率： 500K；
- 18 个通道，每个通道可配置采样时间：16 个外部通道， 1 个内部温度传感器（T-Sensor）， 1 个内部带隙基准电压检测（Bandgap）；
- 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 16 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）
    - 单规则组单次转换（mode1）
    - 单规则组通道连续转换（mode2）
    - 多规则组通道单次扫描，带注入触发（mode3）
    - 多规则组通道单次扫描，带自动注入（mode4）.
    - 多规则组通道持续扫描，带注入触发（mode5）.
    - 多规则组通道持续扫描，带自动注入（mode6）.
    - 在间断转换模式下的多规则组通道（mode7）.
    - 在间断转换模式下的多注入组通道（mode8）
- 通过内部软件触发或外部硬件触发启动 ADC；
- 模拟看门狗功能：
    - 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）.

- 模拟看门狗事件（AMO）.

- DMA 访问： 仅用于规则组通道。

10.3 ADC 功能描述

下图为 ADC 模块框图：

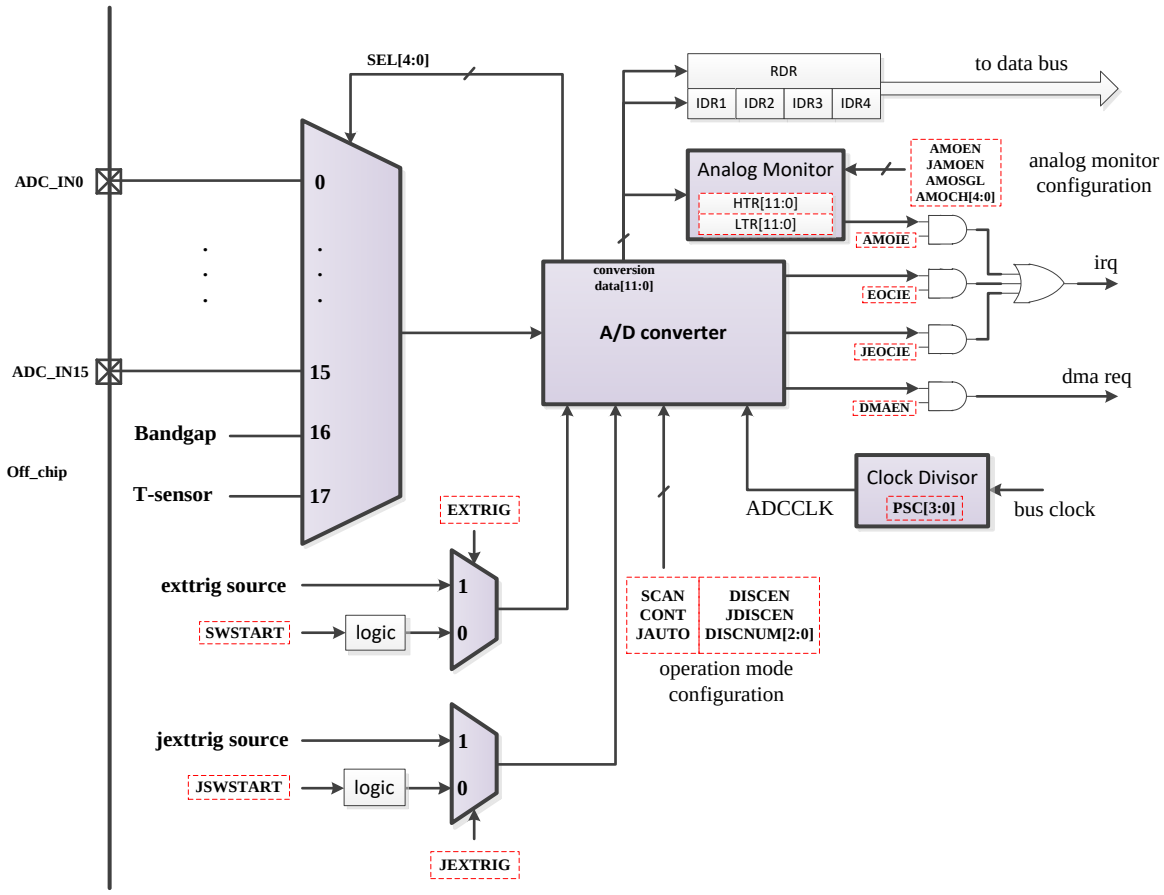

图 10-1 ADC 框图

在上面的框图中， 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）和模拟看门狗（analog monitor）等组成。如图 10-1 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，简称 BCLK 。下面介绍一个典型的操作流程。

首先，ADC 应该上电，这将在 10.3.1 节中进行详细描述。然后，可以通过内部 SWSTART 或外部触发源触发 ADC，该触发来源于 CTU 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。同时，模拟看门狗开始工作，如果发生相应的事件，则会出现相关的状态标志。到目前为止，单通道转换流程结束。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。

10.3.1 ADC 上电时序

在开始所有功能之前，首先应使 ADC 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。上电顺序如下所示。

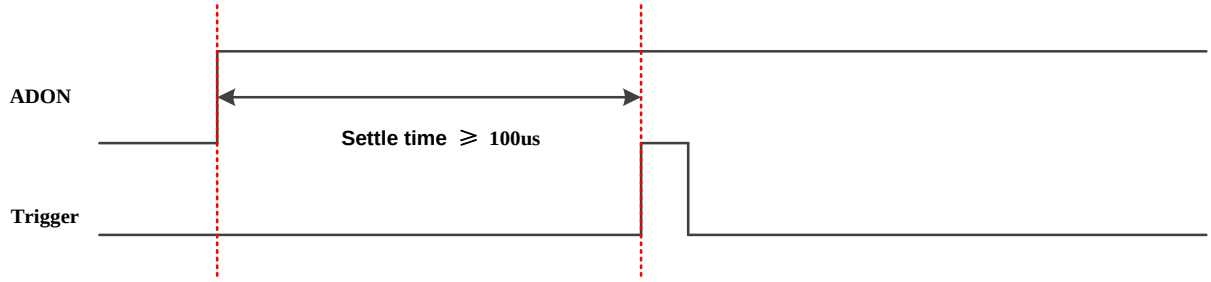

图 10-2 ADC 上电时序

如图 10-2 所示，将 ADC_CTRL2 [ADON]位置为 1 以控制上电过程。在 ADON 设置为 1 后，A/D 转换器单元上电的等待时间不低于 100us，这是基本配置。

10.3.2 ADC 功耗模式

对 ADC 而言，共有两种可用的功耗模式。一种是正常模式，另一种是低功耗模式。如表 10-1 所示，ADC 时钟在低功耗模式下工作频率较低，以降低功耗。当 CPU 通过 WFI 指令进入休眠/停止模式时，可以使 ADC 进入低功耗模式。低功耗模式下的 ADC 模拟看门狗事件可将 CPU 从休眠/停止模式唤醒至正常模式。当 CPU 以正常模式运行并且可以实现所有 ADC 寄存器和功能时，使用 ADC 正常模式。

表 10-1 功耗模式

模式	正常模式	低功耗模式
ADC 时钟 （ADCCLK）	0.1~10 MHz	0.1~1.6 MHz
通道	0 ~ 17	0 ~ 17

为了清楚地解释 ADC 低功耗模式，CPU 工作模式切换的简单流程如图 10-3 所示。例如，如果不需要 CPU 在正常模式下工作并且低功耗有需求，则用户可以通过 WFI 指令让 CPU 进入休眠/停止模式，发生模拟看门狗事件将唤醒 CPU 在正常模式下工作。

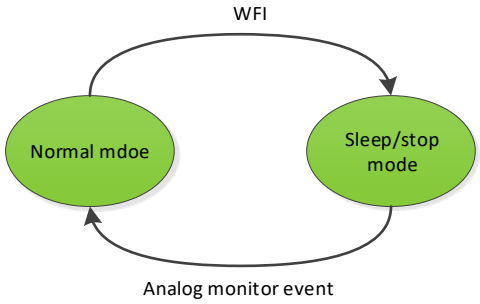

图 10-3 CPU 正常模式和休眠模式

10.3.3 ADC 工作模式

本节为介绍的关键部分，ADC 工作模式细节描述。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 以下列模式之一工作。

表 10-2 ADC 工作模式及对应配置

ADC 工作模式	MODE_BITS	触发源	通道
mode1	5'b0000x	规则触发	单规则组通道
mode2	5'b0100x	规则触发	单规则组通道
mode3	5'b10000	规则触发 （+注入触发）	多规则组通道 （+注入组通道）
mode4	5'b10001	规则触发	多规则组通道 +注入组通道
mode5	5'b11000	规则触发 （+注入触发）	多规则组通道 （+注入组通道）
mode6	5'b11001	规则触发	多规则组通道 +注入组通道
mode7	5'b1x10x	规则触发	规则组通道
mode8	5'b1x01x	注入触发	注入组通道

注意： 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~ch17，其中 ch0~ch15 是外部输入通道，ch16 对应于内部带隙参考电压通道，ch17 代表温度传感器通道。

规则组是按顺序转换的输入通道。基于 ADC_RSQR1，ADC_RSQR2 和 ADC_RSQR3 寄存器，规则组由 RSQ1 至 RSQ16 的最多 16 个通道组成。

例如，如果 RSQ1~RSQ16 分别设置为 9,8,16,1,5,4,7,3,17,2,0,0,0,17,6,15，则将规则组按图 10-4 所示进行排列。

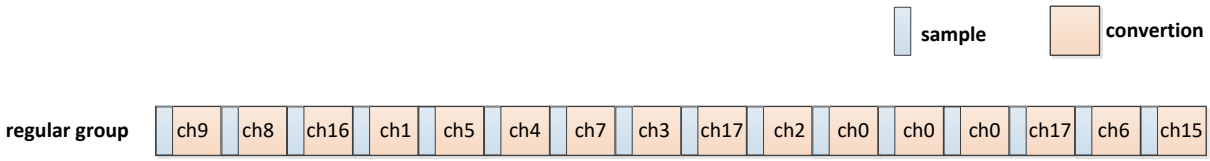

图 10-4 规则组序列

如果 RSQL 设置为 13，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 10-5 所示。

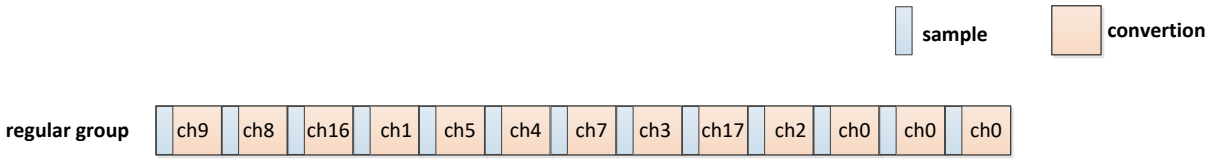

图 10-5 有效规则组序列

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 ISQ1 至 ISQ4。

例如，如果 ISQ1~ISQ4 分别设置为 16,7,17,2，则将注入组按图 10-6 所示进行排列。

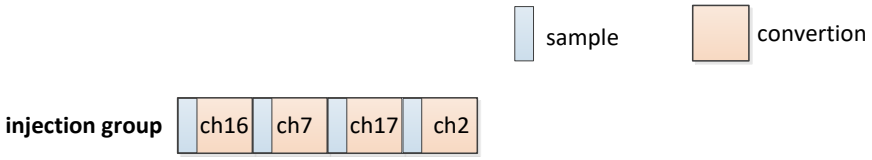

图 10-6 注入组序列

如果 ISQL 设置为 3，则最后 1 个通道将无效并且不会被转换。 因此，有效的注入组序列如图 10-7 所示。

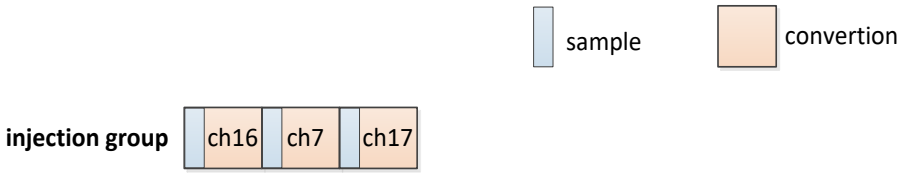

图 10-7 有效注入组序列

显然，规则组触发和注入组触发是开始转换规则和注入组序列的相对应信号。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，每种模式的详细信息描述如下。同时，图 10-4 和图 10-6 用于解释每种模式的操作流程。

10.3.3.1 Mode 1

无论 RSQL 是什么，此模式仅针对规则组中的第一个通道。当模式按表 10-2 进行配置后，有效触发可使 ADC 工作在此模式。

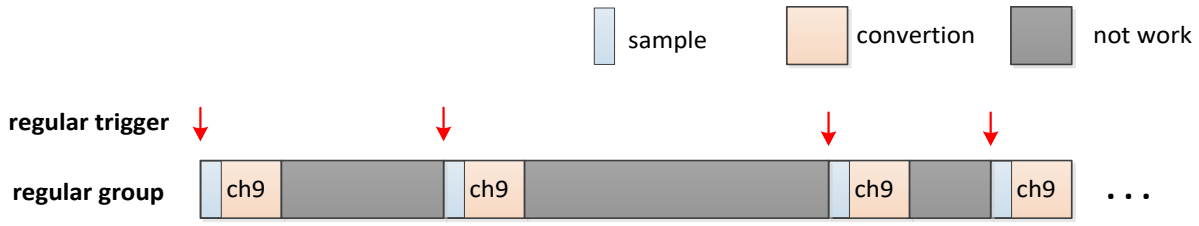

图 10-8 Mode 1 工作流程

如图 10-8 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。

10.3.3.2 Mode 2

无论 RSQL 是什么，这种模式也只针对规则组中的第一个通道。模式按表 10-2 进行配置后，有效触发可以使 ADC 在此模式下工作。

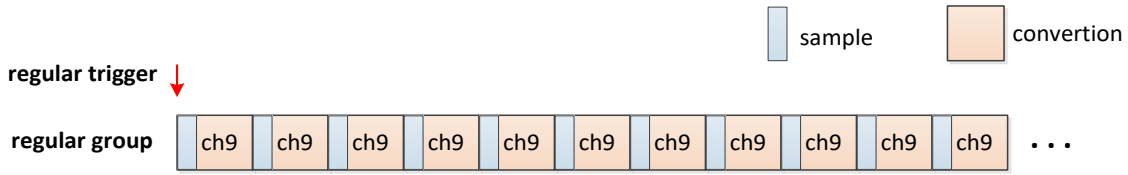

图 10-9 Mode 2 工作流程

如图 10-9 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。

10.3.3.3 Mode 3

此模式针对规则组通道和通过注入触发器触发的注入组通道。有效的规则和注入组通道编号分别由 RSQL 和 ISQL 决定。使用表 10-2 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 7，ISQL 设置为 3。一个典型操作如图 10-10 所示。第一笔规则触发器开始转换组中的前 7 个通道。当 ADC 转换规则组中的通道 1 时，注入触发器会在通道 1 转换结束后将转换切换为 3 个注入组通道。在 3 个注入组通道完成转换后，转换会自动切换回规则组通道，最后 3 个通道开始转换。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。

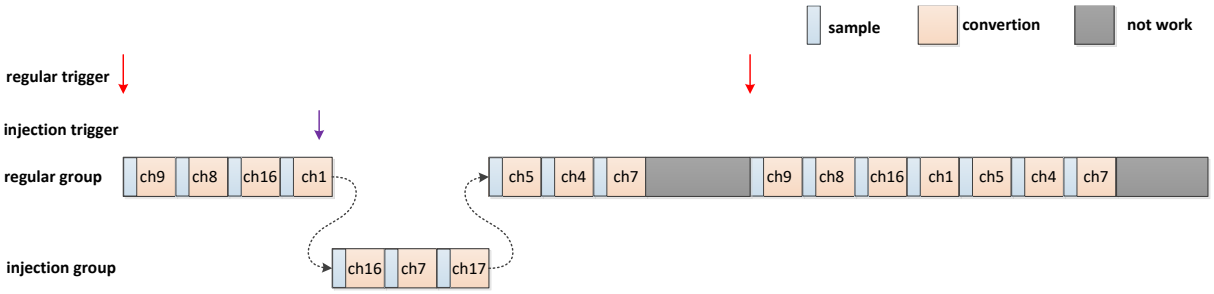

图 10-10 Mode 3 典型工作流程

特别是，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 10-11 所示。

基本上，规则或注入组通道在此模式下由有效触发器转换一次。

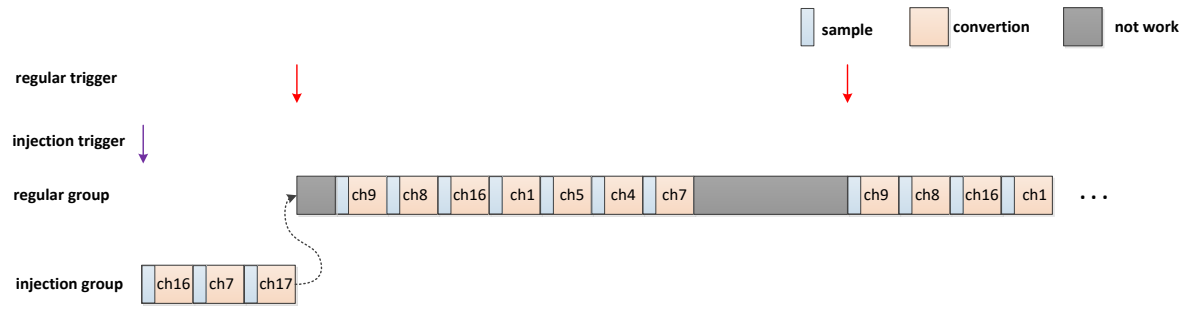

图 10-11 Mode 3 在 ADC 空闲状态下具有注入触发的操作流程

10.3.3.4 Mode 4

此模式旨在对所有规则组通道和注入组通道进行单次转换，这些通道将自动按照规则组通道进行转换。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 10-2 中的模式配置，有效触发可使 ADC 在此模式下工作。例如，SQL 设置为 7，ISQL 设置为 3。典型操作如图 10-12 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。

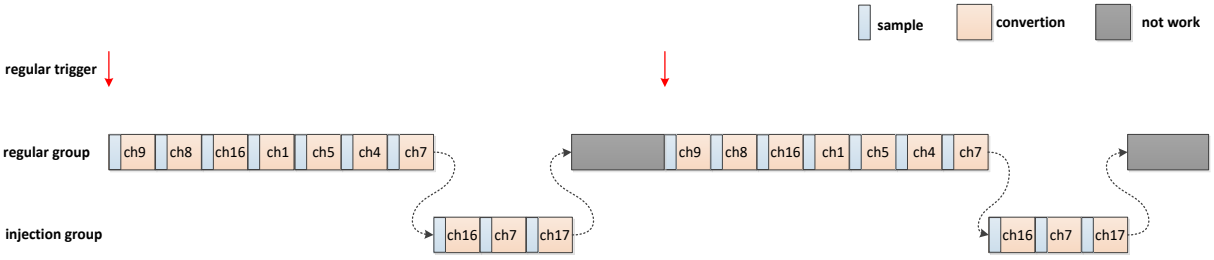

图 10-12 Mode 4 操作流程，注入触发器处于 ADC 空闲状态

10.3.3.5 Mode 5

此模式仅针对规则组通道和通过注入触发器触发的注入组通道。有效的规则通道和注入组通道分别由 SQL 和 ISQL 决定。使用表 10-2 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，顶层复位和模式更改。例如，RSQL 设置为 7，ISQL 设置为 3。一个典型操作如图 10-13 所示。在常规触发后，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。

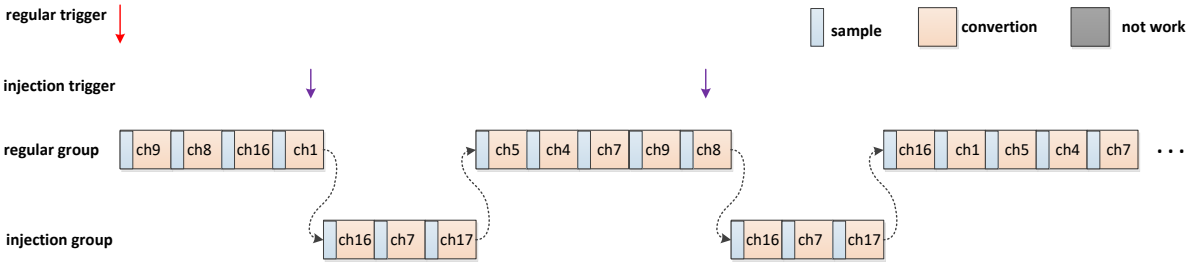

图 10-13 Mode 5 典型操作流程

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 10-14 所示。

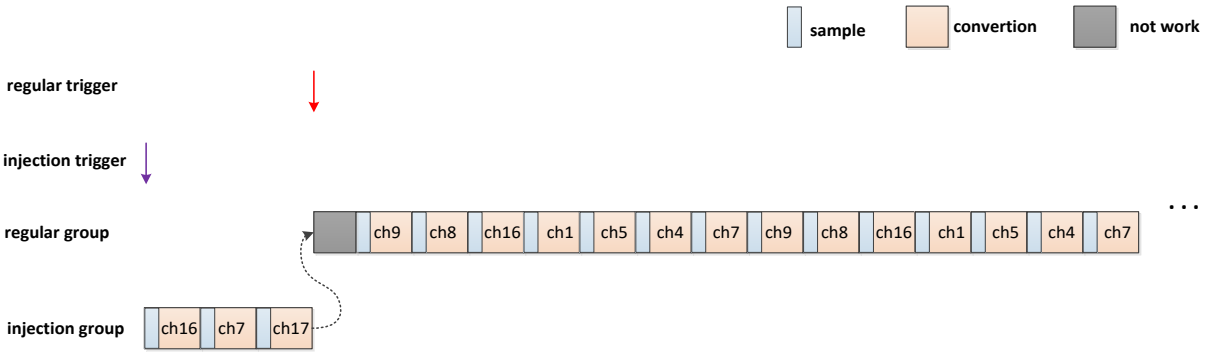

图 10-14 Mode 5 操作流程， 注入触发处于 ADC 空闲状态

10.3.3.6 Mode 6

此模式针对规则组通道和注入组通道。有效的规则通道和注入组通道分别由 RSQL 和 ISQL 决定。使用表 10-2 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，顶层复位和模式更改。例如，RSQL 设置为 7，ISQL 设置为 3，操作流程如图 10-15 所示。ADC 在规则组通道上按顺序工作，然后在规则触发后循环注入组通道。

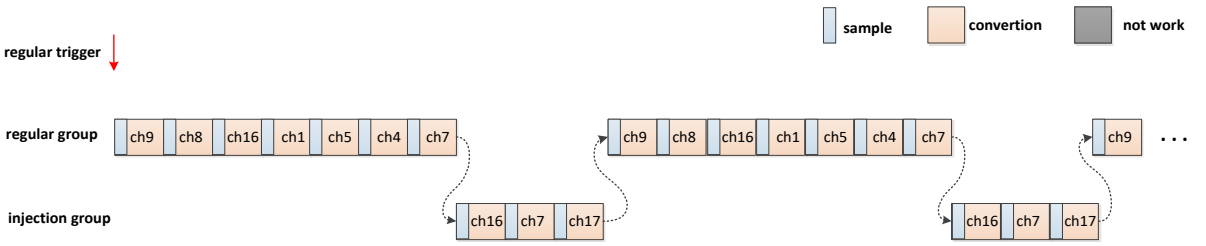

图 10-15 Mode 6 操作流程

10.3.3.7 Mode 7

此模式仅针对规则组通道。有效的规则组通道由 RSQL 决定。使用表 10-2 中的模式配置，ADC 可以在此模式下工作。每个 DISCNUM 信道将有效的规则通道分成几个子组。

例如，RSQL 设置为 7，DISCNUM 设置为 2。

第一次规则触发： ch9, ch8；

第二次规则触发： ch16, ch1；

第三次规则触发： ch5, ch4；

第四次规则触发： ch7，产生 EOC 标志；

第五次规则触发： ch9, ch8；

第六次规则触发： ch16, ch1；

…

因此，实际的转换流程如图 10-16 所示。

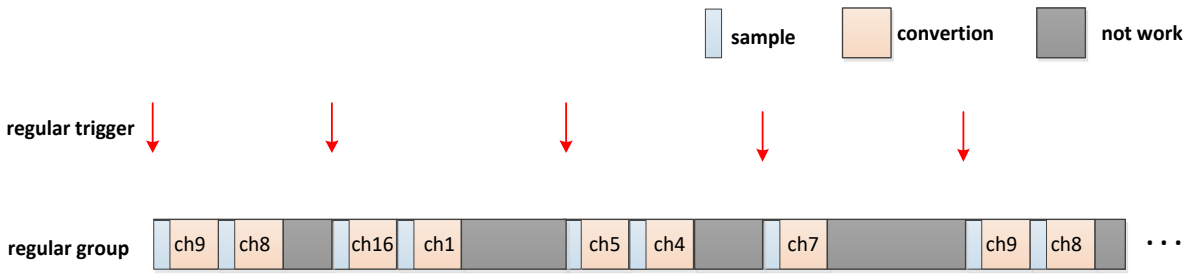

图 10-16 Mode 7 操作流程

10.3.3.8 Mode 8

该模式仅针对注入组通道。有效的注入通道组通道由 ISQL 决定。使用表 10-2 中的模式配置，ADC 可以在此模式下工作。每个通道的有效注入通道分成几个子组。例如，ISQL 设置为 3。

第一次注入触发：ch16；

第二次注入触发：ch7；

第三次注入触发：ch17，产生 EOC 和 IEOC 标志；

第四次注入触发：ch16；

第五次注入触发： ch7；

…

因此，实际的转换流程如图 10-17 所示。

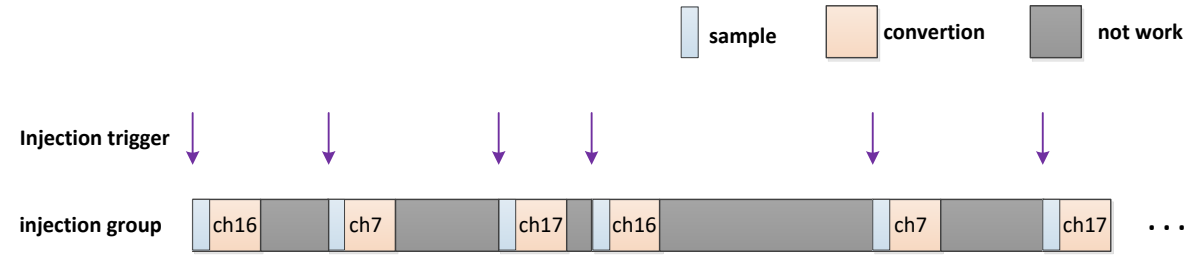

图 10-17 Mode 8 操作流程

10.4 模拟看门狗

如果监控的通道电压高于高阈值或低于低阈值，则模拟看门狗将 AMO 标志设置为 1，通过向其写入 0 来清除该标志。同时，如果 AMO 标志为 1 且 AMOIE 配置为 1，则发生中断。
高阈值和低阈值分别由 AWDH 和 AWDL 决定。模拟看门狗可用于监视基于 AMOEN，IAMOEN，AMOSGL 和 AMOCH 位的无通道，单通道及多通道。

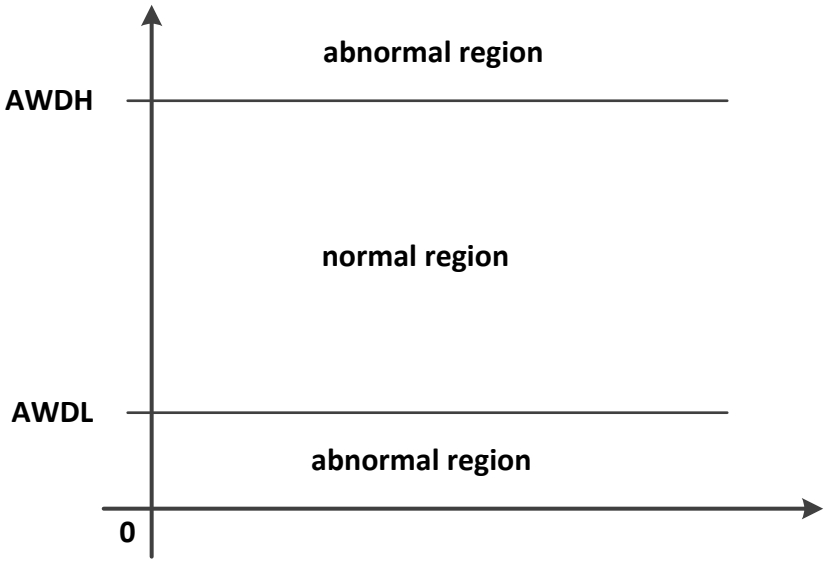

图 10-18 模拟看门狗检测区域

表 10-3 模拟看门狗配置

模拟看门狗通道	{AMOEN,IAMOEN,AMOSGL}	可配置的操作模式	注释
无	3'b00x	所有模式	-
所有注入组通道	3'b010	mode3 ~ mode6, mode8	-
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 ~ mode6, mode8	转换序列必须包含由 AMDCH [4：0]确定的注入通道
单规则组通道	3'b101	mode1 ~ mode7	转换序列必须包含由 AMDCH [4：0]确定的规则通道
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 AMDCH [4：0]确定的规则或注入通道

10.5 状态标志描述

对于 ADC，有三个标志表示转换状态。一个是 EOC，另一个是 IEOC，最后一个是 AMO。EOC 标志表示规则组和注入组通道的转换结束。IEOC 标志标识是否所有注入组通道都转换完成。AMO 标志标识是否发生模拟看门狗事件。模拟看门狗事件是指，基于配置，当前转换结果是高于高阈值还是低于低

阈值。对于不同模式，在不同时间生成 EOC 和 IEOC 标志，可以将其分为三个情形。对于所有模式，同时生成 AMO 标志。假设 ch5 小于 AWDL，ch7 大于 AWDH，并且模拟看门狗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。

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 10-19 所示。

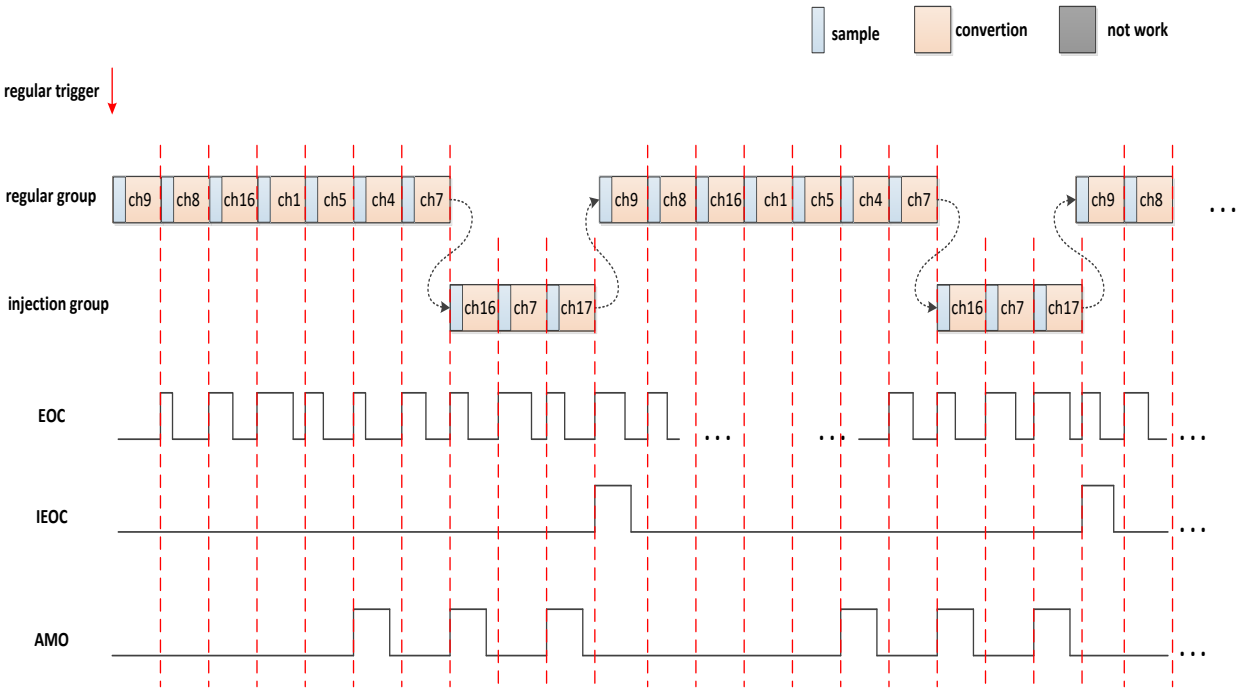

图 10-19 情形 1 下的三个标志

在图 10-19 中，当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看门狗正常区域（例如，ch5，ch7）时，将 AMO 标志设置为 1。需要指出的是，标志保持为 1 的持续时间由 CPU 响应时间决定，该响应时间与此时 CPU 的当前负载有关。

生成标志的时间在 mode7 和 mode 1~mode 6 之间是不同的。有关三个标志的详细信息如图 10-20 所示。

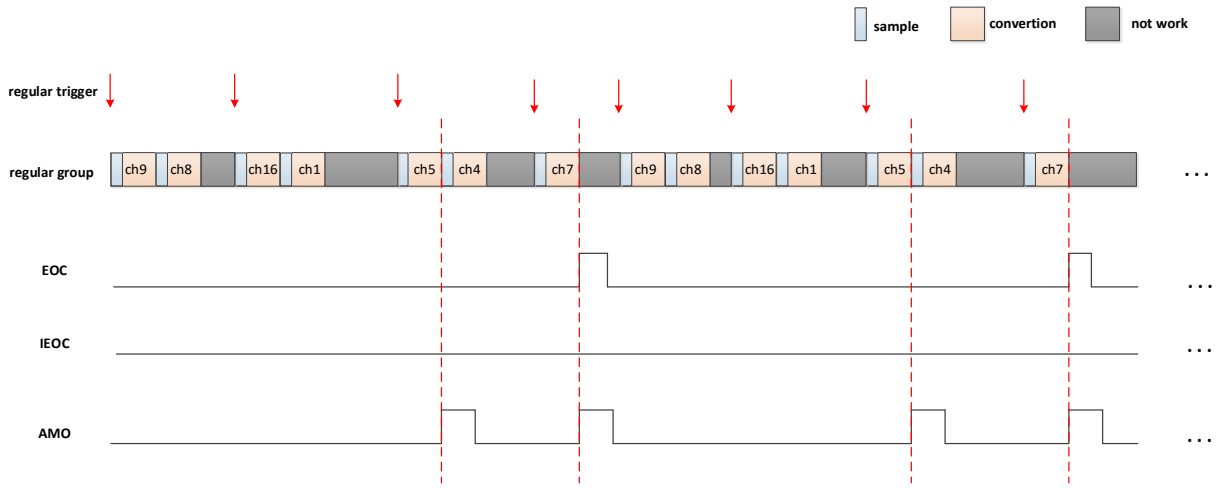

图 10-20 情形 2 下的三个标志

在图 10-20 中，当规则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，此模式始终为 0。当通道电压超出模拟看门狗正常区域（例如 ch5，ch7）时，AMO 标志被设置为 1。需要指出的是，改标志保持为 1 的持续时间由 CPU 响应时间决定，该响应时间与此时 CPU 的当前负载有关。

该标志在 mode 8 下还显示了不同的场景。描述标志信息很容易，因为 ADC 转换只针对注入组通道。

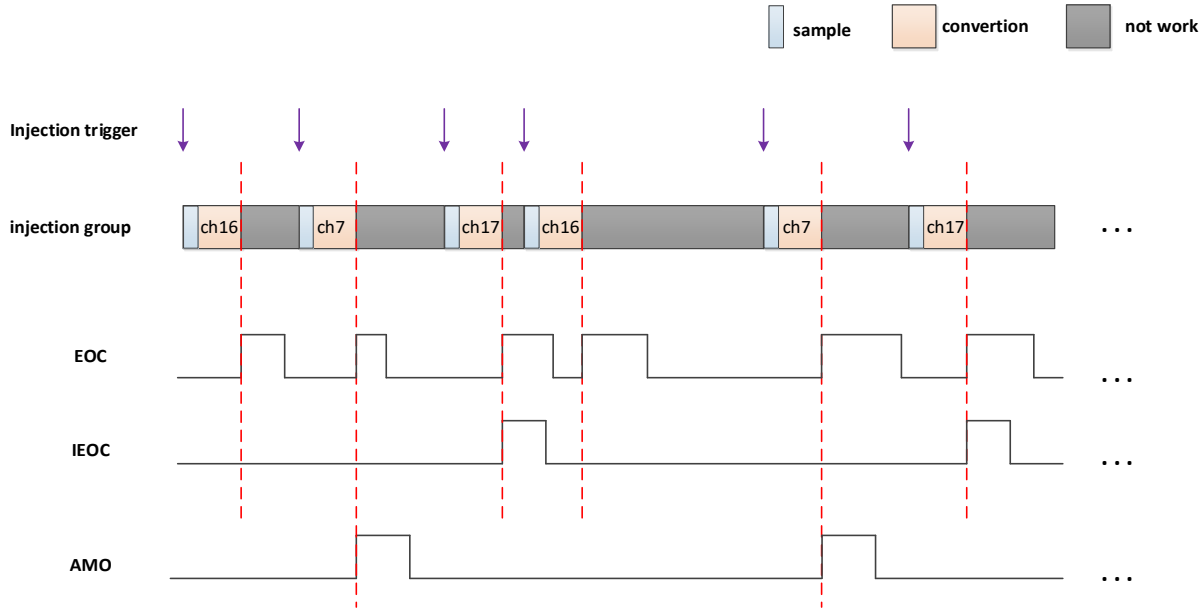

图 10-21 情形 3 下的三个标志

在图 10-21 中，当所有注入组通道的通道转换完成后，EOC 和 IEOC 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，通过将 IEOC 位置为 0 来清除它。当通道电压超出模拟看门狗正常区域（例如 ch7）时，AMO 标志被设置为 1。需要指出的是，此标志保持为 1 的持续时间由 CPU 响应时间决定，该响应时间与此时 CPU 的当前负载有关。

10.6 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。

10.7 采样转换时间

ADC 使用若干个 ADC_CLK 周期对输入电压采样，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=48MHz，ADCCLK=8MHz，SPT=3 ADCCLK，总转换时间=（3+12）/8+（5/48） ≈ 1.98μs。

10.8 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。

10.9 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。因此，DMA 功能专用于规则组通道。

10.10 编程指南

10.10.1 正常功耗模式

基本编程步骤可描述为三个步骤，其简要介绍宏观的配置过程。详细配置可参考如下示例。

宏观的三个基本步骤：

- 第一步： ADC 运行时钟配置和上电。

- 第二步： 模式配置和其他配置。
- 最后一步： 有效触发器。

例如，ADC 使用以下配置：

(1) Mode 5，ADC 工作时钟：8.33MHz @ 总线时钟 = 50MHz.

(2) 规则时序： ch9, ch8, ch16, ch1, ch5, ch4, ch7, ch3, ch17, ch2, ch0, ch0, ch0, ch17, ch6, chf.

RSQL=7;

(3) 注入时序： ch16, ch7, ch17, ch2.

ISQL=3;

(4) 模拟看门狗： 适用于所有通道。

(5) 其他：默认。

10.10.2 低功耗模式

低功耗模式的 ADC 配置与正常模式配置相同，ADC 进入低功耗模式，并从低功耗模式唤醒，如图 10-3 所示。

10.11 ADC 寄存器定义

表 10-4 ADC 寄存器定义

基地址 = 0x40003000.
地址 = 基地址 + 偏移地址

偏移地址	名称	宽度	寄存器功能
00000000	ADC_STR	32	ADC 状态寄存器（status register）
00000004	ADC_CTRL1	32	ADC 控制寄存器 1（control register1）
00000008	ADC_CTRL2	32	ADC 控制寄存器 2（control register2）
0000000C	ADC_SPT1	32	ADC 采样时间选择寄存器 1（sample time selection register 1）
00000010	ADC_SPT2	32	ADC 采样时间选择寄存器 2（sample time selection register 2）
00000014	ADC_IOFR1	32	ADC 注入组偏移寄存器 1（injection group offset register 1）
00000018	ADC_IOFR2	32	ADC 注入组偏移寄存器 2（injection group offset register 2）
0000001C	ADC_IOFR3	32	ADC 注入组偏移寄存器 3（injection group offset register 3）

偏移地址	名称	宽度	寄存器功能
00000020	ADC_IOFR4	32	ADC 注入组偏移寄存器 4（injection group offset register 4）
00000024	AWDH	32	ADC AWD 高阀值寄存器
00000028	AWDL	32	ADC AWD 低阀值寄存器
0000002C	ADC_RSQR1	32	ADC 规则组序列配置寄存器 1（regular group sequence configure register 1）
00000030	ADC_RSQR2	32	ADC 规则组序列配置寄存器 2（regular group sequence configure register 2）
00000034	ADC_RSQR3	32	ADC 规则组序列配置寄存器 3（regular group sequence configure register 3）
00000038	ADC_ISQR	32	ADC 注入组序列配置寄存器（injection group sequence configure register）
0000003C	ADC_IDR1	32	ADC 注入组数据寄存器 1（injection group data register 1）
00000040	ADC_IDR2	32	ADC 注入组数据寄存器 2（injection group data register 2）
00000044	ADC_IDR3	32	ADC 注入组数据寄存器 3（injection group data register 3）
00000048	ADC_IDR4	32	ADC 注入组数据寄存器 4（injection group data register 4）
0000004C	ADC_RDR	32	ADC 规则组数据寄存器（regular group data register）

00000000 ADC_STR ADC 状态寄存器 00000000

位	31~18	7	6	5	4	3	2	1	0
名称					ADC_IDLE		IEOC	EOC	AMO
类型					R		R	R	R

位	助记符	名称	说明
4	ADC_IDLE		ADC 空闲状态标志（用于休眠功能） 0： ADC 非空闲 1： ADC 空闲
2	IEOC		注入组转换完成标志 0： 注入组转换没有完成 1： 注入组转换完成，写 0 清除

位	助记符	名称	说明
1	**EOC**		**规则组转换完成标志** 0：规则组转换没有完成 1：规则组转换完成，写 0 或读取 ADC_RDR 以清除该位
0	**AMO**		**模拟看门狗事件发生** 0： 没有模拟看门狗事件 1： 发生模拟看门狗事件，写 0 以清除该位

00000004 ADC_CTRL1 ADC 控制寄存器 1 00000000

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	**SWSTART**	**ISWSTART**								**ALIGN**	**IEXTTRIG**	**EXTTRIG**	**DMAEN**	**AMOIE**	**IEOCIE**	**EOCIE**
类型	RW	RW								RW	RW	RW	RW	RW	RW	RW
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]**				
类型	RW	RW	RW	RW	RW	RW	RW	RW				RW	RW	RW	RW	RW

位	助记符	名称	说明
31	**SWSTART**	SWSTART	**规则通道软件触发** 写 1 进行触发，硬件自动清除
30	**ISWSTART**	ISWSTART	**注入通道软件触发** 写 1 进行触发，硬件自动清除
22	**ALIGN**	ALIGN	**数据对齐** 0：右对齐 1：左对齐
21	**IEXTTRIG**	IEXTTRIG	**注入组触发源选择** 1：外部 0：内部（软件触发）
20	**EXTTRIG**	EXTTRIG	**规则组触发源选择** 1：外部 0：内部（软件触发）
19	**DMAEN**	DMAEN	**DMA 功能使能** 1：使能 0：禁用
18 ~ 16	**AMOIE,EOCIE,IEOCIE**	AMOIE,EOCIE,IEOCIE	**中断功能使能** 1：使能 0：禁用
15 ~ 11	**Modes control-bits**		**ADC 工作模式** 详细配置在表 10-2 中
10 ~ 8	**DISCNUM**	DISCNUM	**通道的不连续转换长度** 0 ~ 7： 在 mode 7 下确定子组长度

位	助记符	名称	说明
7 ~ 5	**Analog monitor control bits**	-	**模拟看门狗功能配置** 详细配置在**表 10-3** 中
4 ~ 0	**AMOCH**	AMOCH	**模拟看门狗检测通道** 当模拟看门狗配置为仅检测单个通道时，指定被监测的通道

00000008 ADC_CTRL2 **ADC 控制寄存器 2** **0000F002**

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

位	助记符	名称	说明
15 ~ 12	**PSC**	PSC	**总线时钟预分频以获得 ADC 工作时钟** **注意： psc 值必须确保 ADC 工作时钟低于 10 MHz.** 0 ~ 15 ： 1 ~ 16 分频器.
0	**ADON**	ADON	**ADC 上电** 写 1： ADC 上电 写 0： ADC 断电，复位 ADC（但配置寄存器不会被重置）.

0000000C ADC_SPT1 **ADC 采样时间寄存器 1** **00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称									**SPT17[2：0]**			**SPT16[2：0]**			**SPT15[2：0]**	
类型									RW							
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称		**SPT14[2：0]**			**SPT13[2：0]**			**SPT12[2：0]**			**SPT11[2：0]**			**SPT10[2：0]**		
类型	RW															

注意： x=10 ~ 17

位	助记符	名称	说明
23：0	**SPTx**	SPTx	**每个通道采样时间选择** 000 ~ 111： 6/14/29/42/56/72/215/3 ADCCLK.

00000010 ADC_SPT2 **ADC 采样时间寄存器 2** **00000000**

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]	
类型			RW													
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]		
类型	RW															

注意： x=0 ~ 9

位	助记符	名称	说明
23：0	**SPTx**	SPT	**每个通道采样时间选择** 000 ~ 111： 6/14/29/42/56/72/215/3 ADCCLK.

00000014~20 ADC_IOFRx（x=1~4） **ADC 注入组偏移寄存器** **0000000**

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

注意： x= 0 ~ 3

位	助记符	名称	说明
11：0	**IOFRx**	IOFRx	**注入组偏移值** 最终转换结果将减去偏移值

00000024 AWDH **ADC AWD 高阀值寄存器** **00000000**

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

位	助记符	名称	说明
11：0	**AWDH**	AWDH	**模拟看门狗的高阀值** 定义高阀值

00000028 AWDL **ADC AWD 低阀值寄存器** **00000000**

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

位	助记符	名称	说明
11：0	**AWDL**	AWDL	**模拟看门狗的低阀值** 定义低阀值

0000002C ADC_RSQR1 **ADC 规则组序列配置寄存器 1** **00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称									**RSQL[3：0]**				**RSQ16[4：0]**			
类型									RW							
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称		**RSQ15[4：0]**					**RSQ14[4：0]**					**RSQ13[4：0]**				
类型	RW															

位	助记符	名称	说明
23：20	**RSQL**	RSQL	**规则组的长度** **Note**：长度必须小于实际有效的规则组序列长度的值 0 ~ 15：定义规则组长度 1~16
19：0	**RSQx**	RSQx	**规则组通道选择** 0~15：外部通道 16： BG 电压 17： 温度传感器电压

00000030 ADC_RSQR2 **ADC 规则组序列配置寄存器 2** **00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称			**RSQ12[4：0]**					**RSQ11[4：0]**					**RSQ10[4：0]**			
类型			RW													
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称		**RSQ9[4：0]**					**RSQ8[4：0]**					**RSQ7[4：0]**				
类型	RW															

位	助记符	名称	说明
29：0	**RSQx**	RSQx	**规则组通道选择** 0~15：外部通道 16： BG 电压 17： 温度传感器电压

00000034 ADC_RSQR3 **ADC 规则组序列配置寄存器 3** **00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称			RSQ6[4：0]					RSQ5[4：0]					RSQ4[4：0]			
类型			RW													
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称		RSQ3[4：0]					RSQ2[4：0]					RSQ1[4：0]				
类型	RW															

位	助记符	名称	说明
29：0	**RSQx**	RSQx	**规则组通道选择** 0~15： 外部通道 16： BG 电压 17： 温度传感器电压

00000038 ADC_ISQR **ADC 注入组序列配置寄存器** **00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称											ISQL[1：0]		ISQ4[4：0]			
类型											RW					
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称		ISQ3[4：0]					ISQ2[4：0]					ISQ1[4：0]				
类型	RW															

位	助记符	名称	说明
21：20	**ISQL**	ISQL	**注入组长度** **注意：该长度必须小于实际有效的注入组序列数。** 0 ~ 3： 定义注入组长度 1~4
19：0	**ISQx**	ISQx	**注入组通道选择** 0~15： 外部通道 16： BG 电压 17： 温度传感器电压

0000003C~48 ADC_IDRx（x=1~4） **ADC 注入组数据寄存器** **00000000**

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

注意： x=0 ~ 3

位	助记符	名称	说明
11：0	**IDRx**	IDR	**注入组的数据寄存器** **注意：注入组有 4 个数据寄存器**

0000004C ADC_RDR **ADC 规则组数据寄存器** **00000000**

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

位	助记符	名称	说明
11：0	**RDR**	RDR	**规则组数据寄存器** **注意：规则组只有一个寄存器。如果 ADC 正高速工作且 CPU 无法及时处理，则用户必须打开 ADC 的 DMA 功能，以避免数据丢失。**

11 模拟比较器（ACMP）

11.1 简介

ACMP 模块包括 ACMP0 和 ACMP1。ACMP0 和 ACMP1 都包含一个比较器和一个 6 位数字模拟转换器（DAC）。模拟多路复用器（MUX）提供一个用于从六个通道中选择模拟输入信号的电路。一个通道由 6 位数字模拟转换器（DAC）提供，其他通道由外部输入提供。

ACMP0 的轮询模式和霍尔输出（Hall）功能专为电机应用而设计。ACMP1 没有轮询和霍尔（Hall）输出功能。

11.2 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；
- 可在比较器输出的上升沿、下降沿、两个上升沿或下降沿时选择中断；
- 最多 6 个可选择比较器输入（ADC_IN0~ADC_IN4 以及内部 DAC）；
- 支持停止（Stop）模式唤醒；
- 支持 CTU 触发；
- ACMP0 支持轮询模式；
- ACMP0 支持 霍尔（Hall） 输出。

11.2.1 框图

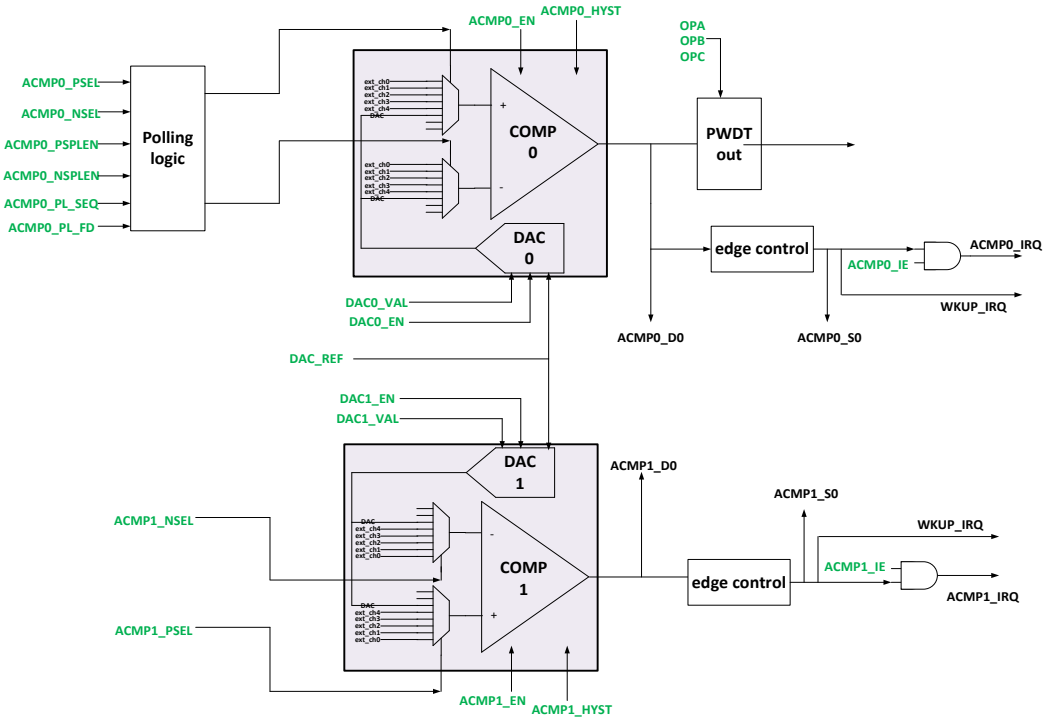

图 11-1 ACMP 框图

11.3 功能描述

ACMP0/ACMP1 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。通过置位 DAC_REF ，DAC 可选择 Vdd 或片上带隙基准源这两个参考输入中的一个 作为 DAC 输入 Vin 。在 DAC 使能后，将 DAC_VAL 中设置的数据转换为步进式模拟输出，然后馈入 ACMP 作为内部参考输入。

ACMP0/ACMP1 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 6 个通用输入中选择： 来自 DAC 输出的三个外部参考输入和一个内部参考输入。

11.3.1 ACMP0

11.3.1.1 正常模式

通过置位 ACMP0_CR0[ACMP0_EN]使能 ACMP0 后，比较结果以数字输出呈现。只要 ACMP0_CR0[ACMP0_MOD]中定义的有效边沿出现，ACMP0_SR[ACMP0_F]的电平就会变为有效值。 如果 ACMP0_CR0[ACMP0_IE] 置位，则会发生 CPU 中断。

ACMP0 输出由总线时钟同步以生成 ACMP0_DR[ACMP0_O] ，以便 CPU 能读出比较结果。ACMP0_DR[ACMP0_O] 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。

11.3.1.2 轮询模式

ACMP0 可以切换比较器正输入或负输入的输入通道。交换序列在 ACMP0_CR4 [ACMP0_PL_SEQ]中定义，频率由 ACMP0_FD[ACMP0_PL_FD]控制。ACMP0_PS_PLEN 和 ACMP0_NS_PLEN 是轮询模式的使能位。ACMP0_PS_PLEN 和 ACMP0_NS_PLEN 无法同时使能。ACMP0_PS_PLEN 和 ACMP0_NS_PLEN 使能都不会触发轮询模式。因此，软件必须确保使能上述两个字段之一。

这里提供一个有关轮询模式的示例。ACMP0 正输入轮询，轮询频率为 source_clk/100，外部通道 1-4 和 DAC 输出动作轮询，ACMP0 负输入选择外部输入 0，下降沿触发中断。

步骤 1： ACMP0_IE = 1'b1, ACMP0_MOD = 2'b00；

步骤 2： 设置 DAC0_VAL, DAC0_EN = 1'b1；

步骤 3： ACMP0_PS_PLEN = 1'b1, ACMP0_NS_PLEN = 1'b0；

步骤 4： ACMP0_PL_FD = 2'b01, ACMP0_PL_SEQ = 6'b111110；

步骤 5： ACMP0_NSEL = 3'b000；

步骤 6： ACMP0_EN = 1'b1。

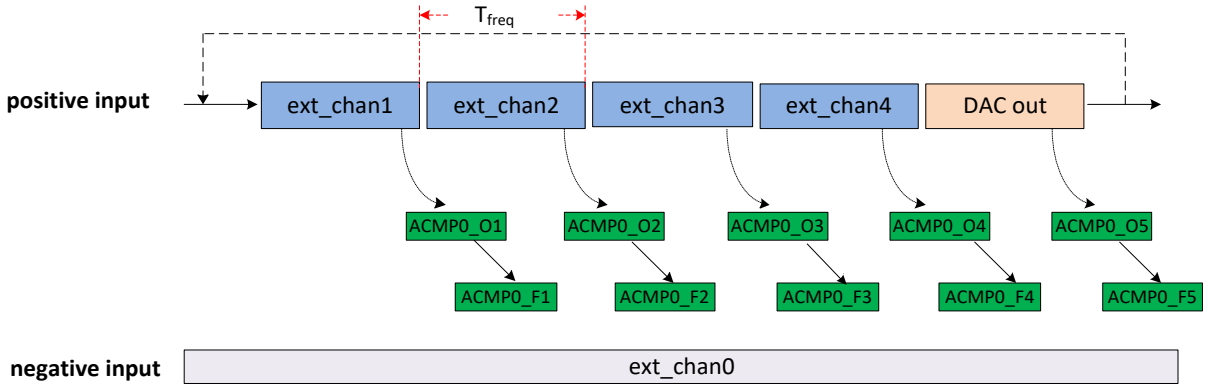

图 11-2 轮询模式

11.3.1.3 HALL 输出模式

ACMP0 有三个 hall 输出：acmp0_out_pwt_a，acmp0_out_pwt_b 和 acmp0_out_pwt_c。 这些信号连接到芯片内部的 PWDT 模块。这些 hall 输出与轮询功能配合，完成电机控制。每个 hall 输出都可以通过轮询模式选择六个通道中的一个。

例如，若轮询模式 ACMP0_PL_SEQ = 6'b001110，则轮询顺序为：外部输入 1- >外部输入 2 -> 外部输入 3（通道 1 -> 通道 2 -> 通道 3）。

设置 ACMP0_OPA[ACMP0_OPA_SEL] = 3'b010，则 acmp0_out_pwt_a 为 ACMP0_DR[ACMP0_O2]。

11.3.2 ACMP1

ACMP1 是一个普通功能模块，没有类似于 ACMP0 的轮询模式以及霍尔输出模式。

11.3.3 低功耗唤醒

在低功耗模式下，ACMP 输出上的有效边沿会产生异步中断，可将 MCU 从低功耗模式唤醒。通过向 ACMP0_WUF/ACMP1_WUF 写 1 来清除该中断。请注意，唤醒功能仅对正常模式有效。

注意：使用 ACMP 唤醒 MCU 时，建议选择上升沿。由于在 MCU 低功耗模式下，ACMP 时钟被禁用以节省功耗，因此它对外部信号敏感，如果在使用下降沿时外部信号存在噪声，MCU 将被错误地唤醒。

11.4 寄存器定义

表 11-1 ACMP 寄存器映射

模块名： ACMP　　基地址：（+40005000h）

地址	名称	宽度	寄存器功能
40005000	ACMP0_CR0	32	ACMP0 配置寄存器 0
40005004	ACMP0_CR1	32	ACMP0 配置寄存器 1
40005008	ACMP0_CR2	32	ACMP0 配置寄存器 2
4000500C	ACMP0_CR3	32	ACMP0 配置寄存器 3
40005010	ACMP0_CR4	32	ACMP0 配置寄存器 4
40005014	ACMP0_DR	32	ACMP0 数据输出寄存器 0
40005018	ACMP0_SR	32	ACMP0 状态寄存器 0
4000501C	ACMP0_FD	32	ACMP0 轮询分频器寄存器
40005020	ACMP0_OPA	32	ACMP0 hall 输出 A 设置寄存器
40005024	ACMP0_OPB	32	ACMP0 hall 输出 B 设置寄存器
40005028	ACMP0_OPC	32	ACMP0 hall 输出 C 设置寄存器
4000502C	ACMP_DACSR	32	ACMP DAC 参考选择寄存器
40005030	ACMP1_CR0	32	ACMP1 配置寄存器 0
40005034	ACMP1_CR1	32	ACMP1 配置寄存器 1
40005038	ACMP1_CR2	32	ACMP1 配置寄存器 2
4000503C	ACMP1_DSR	32	ACMP1 数据和状态寄存器
40008820	ACMPDAC_CFG	32	ACMPDAC CFG0 寄存器

40005000　　ACMP0_CR0　　ACMP0 配置寄存器 0　　00000000

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型																
复位																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									ACMP0_EN	ACMP0_HYST		ACMP0_IE		ACMP0_OPE	ACMP0_MOD	
类型									RW	RW		RW		RW	RW	
复位									0	0		0		0	0	0

位	名称	说明
7	ACMP0_EN	**ACMP0 使能** 0： 禁用 1： 使能

位	名称	说明
6	ACMP0_HYST	**模拟比较器 0 迟滞选择** 0： 10mV 1： 20mV
4	ACMP0_IE	**ACMP0 中断使能** 0： 禁用 1： 使能
2	ACMP0_OPE	**ACMP0 hall 输出使能** 0： 禁用 1： 使能
1：0	ACMP0_MOD	**确定 ACMP0 中断触发的灵敏度模式** 00： ACMP0 输出下降沿中断 01： ACMP0 输出上升沿中断 10： ACMP0 输出下降沿中断 11： ACMP0 输出下降沿或上升沿中断

40005004 **ACMP0_CR1** **ACMP0 配置寄存器 1** **00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型																
复位																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称										**ACMP0_PSEL**				**ACMP0_NSEL**		
类型										RW				RW		
复位										0	0	0		0	0	0

位	名称	说明
6：4	ACMP0_PSEL	**ACMP0 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： DAC0 输出 110： 保留，未使用 111： 保留，未使用
2：0	ACMP0_NSEL	**ACMP0 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： DAC0 输出 110： 保留，未使用 111： 保留，未使用

40005008 ACMP0_CR2 ACMP0 配置寄存器 2 00000000

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型																
复位																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									**DAC0_EN**		**DAC0_VAL**					
类型									RW		RW					
复位									0		0	0	0	0	0	0

位	名称	说明
7	DAC0_EN	**DAC0 使能** 0： 禁用 1： 使能
5：0	DAC0_VAL	**DAC0 输出电平选择**

4000500C ACMP0_CR3 ACMP0 配置寄存器 3 00000000

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型																
复位																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									**ACMP0_PSPLEN**				**ACMP0_NSPLEN**			
类型									RW				RW			
复位									0				0			

位	名称	说明
7	ACMP0_PSPLEN	**ACMP0 正输入轮询模式使能** 0： 禁用 1： 使能
3	ACMP0_NSPLEN	**ACMP0 负输入轮询模式使能** 0： 禁用 1： 使能

40005010 ACMP0_CR4 ACMP0 配置寄存器 4 00000000

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型																
复位																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称											**ACMP0_PL_SEQ**					
类型											RW					
复位											0	0	0	0	0	0

位	名称	说明
5：0	ACMP0_PL_SEQ	**ACMP0 轮询通道序列设置** 0： 禁用相应的通道 1： 使能相应的通道

40005014 **ACMP0_DR** **ACMP0 数据输出寄存器 0** **00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型																
复位																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									**ACMP0_O**		**ACMP0_O5**	**ACMP0_O4**	**ACMP0_O3**	**ACMP0_O2**	**ACMP0_O1**	**ACMP0_O0**
类型									RO		RO	RO	RO	RO	RO	RO
复位									0		0	0	0	0	0	0

位	名称	说明
7	ACMP0_O	**ACMP0 正常模式输出**
5	ACMP0_O5	**ACMP0 轮询模式通道 5 输出**
4	ACMP0_O4	**ACMP0 轮询模式通道 4 输出**
3	ACMP0_O3	**ACMP0 轮询模式通道 3 输出**
2	ACMP0_O2	**ACMP0 轮询模式通道 2 输出**
1	ACMP0_O1	**ACMP0 轮询模式通道 1 输出**
0	ACMP0_O0	**ACMP0 轮询模式通道 0 输出**

40005018 **ACMP0_SR** **ACMP0 状态寄存器 0** **00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型																
复位																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									**ACMP0_F**	**ACMP0_WPF**	**ACMP0_F5**	**ACMP0_F4**	**ACMP0_F3**	**ACMP0_F2**	**ACMP0_F1**	**ACMP0_F0**
类型									RW	RW	RW	RW	RW	RW	RW	RW
复位									0	0	0	0	0	0	0	0

位	名称	说明
7	ACMP0_F	**ACMP0 正常模式中断标志** 写 1 清除该标志
6	ACMP0_WPF	**ACMP0 低功耗模式唤醒中断标志** 写 1 清除该标志
5	ACMP0_F5	**ACMP0 轮询模式通道 5 中断标志** 写 1 清除该标志
4	ACMP0_F4	**ACMP0 轮询模式通道 4 中断标志** 写 1 清除该标志
3	ACMP0_F3	**ACMP0 轮询模式通道 3 中断标志**

位	名称	说明
		写 1 清除该标志
2	ACMP0_F2	**ACMP0 轮询模式通道 2 中断标志** 写 1 清除该标志
1	ACMP0_F1	**ACMP0 轮询模式通道 1 中断标志** 写 1 清除该标志
0	ACMP0_F0	**ACMP0 轮询模式通道 0 中断标志** 写 1 清除该标志

4000501C　ACMP0_FD　ACMP0 轮询分频器寄存器　00000000

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型																
复位																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称															**ACMP0_PL_FD**	
类型															RW	
复位															0	0

位	名称	说明
1：0	ACMP0_PL_FD	**ACMP0 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 10： source_clk/70 11： source_clk/50

40005020　ACMP0_OPA　ACMP0 hall 输出 A 设置寄存器　00000000

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型																
复位																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称														**ACMP0_OPA_SEL**		
类型														RW		
复位														0	0	0

位	名称	说明
2：0	ACMP0_OPA_SEL	**ACMP0 hall 输出 A 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：保留，未使用 111：保留，未使用

40005024 **ACMP0_OPB** **ACMP0 hall 输出 B 设置寄存器** **00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型																
复位																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称														**ACMP0_OPB_SEL**		
类型														RW		
复位														0	0	0

位	名称	说明
2：0	ACMP0_OPB_SEL	**ACMP0 hall 输出 B 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：保留，未使用 111：保留，未使用

40005028 **ACMP0_OPC** **ACMP0 hall 输出 C 设置寄存器** **00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型																
复位																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称														**ACMP0_OPC_SEL**		
类型														RW		
复位														0	0	0

位	名称	说明
2：0	ACMP0_OPC_SEL	**ACMP0 hall 输出 C 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： 保留，未使用 111： 保留，未使用

4000502C **ACMP_DACSR** **ACMP DAC 参考选择寄存器** **00000000**

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

位	名称	说明
0	DAC_REF	**DAC 参考选择** 0： DAC 选择带隙作为参考 1： DAC 选择 Vdd 作为参考

40005030　　ACMP1_CR0　　ACMP1 配置寄存器 0　　00000000

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型																
复位																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									ACMP1_EN	ACMP1_HYST		ACMP1_IE			ACMP1_MOD	
类型									RW	RW		RW			RW	
复位									0	0		0			0	0

位	名称	说明
7	ACMP1_EN	**ACMP1 使能** 0： 禁用 1： 使能
6	ACMP1_HYST	**模拟比较器 1 迟滞选择** 0： 10mV 1： 20mV
4	ACMP1_IE	**ACMP1 中断使能** 0： 禁用 1： 使能
1：0	ACMP1_MOD	**确定 ACMP1 中断触发灵敏度模式** 00： ACMP1 输出下降沿中断. 01： ACMP1 输出上升沿中断. 10： ACMP1 输出下降沿中断. 01： ACMP1 输出下降沿或上升沿中断.

40005034　　ACMP1_CR1　　ACMP1 配置寄存器 1　　00000000

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型																
复位																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称										ACMP1_PSEL				ACMP1_NSEL		
类型										RW				RW		
复位										0	0	0		0	0	0

位	名称	说明
6：4	ACMP1_PSEL	**ACMP1 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： DAC1 输出 110： 保留，未使用

位	名称	说明
		111： 保留，未使用
2：0	ACMP1_NSEL	**ACMP1 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： DAC1 输出 110： 保留，未使用 111： 保留，未使用

40005038 ACMP1_CR2 ACMP1 配置寄存器 2 00000000

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型																
复位																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									**DAC1_EN**		**DAC1_VAL**					
类型									RW		RW					
复位									0		0	0	0	0	0	0

位	名称	说明
7	DAC1_EN	**DAC1 使能** 0： 禁用 1： 使能
5：0	DAC1_VAL	**DAC1 输出电平选择**

4000503C ACMP1_DSR ACMP1 数据和状态寄存器 00000000

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型																
复位																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称										**ACMP1_WUF**		**ACMP1_F**				**ACMP1_O**
类型										RW		RW				RW
复位										0		0				0

位	名称	说明
6	ACMP1_WUF	**ACMP1 低功耗唤醒标志** 写 1 清除该标志位
4	ACMP1_F	**ACMP1 中断标志** 写 1 清除该标志位
0	ACMP1_O	**ACMP1 输出**

40008820 ACMPDAC_CFG0 ACMPDAC 配置寄存器 0 01FFFC00

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

40008820　　ACMPDAC_CFG0　　ACMPDAC 配置寄存器 0　　01FFFC00

类型	RW								RW							
复位	0	0	0	0	0	0	0	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	**MPXSEL**						**Reserved**									
类型	RW						RO									
复位	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0

位	名称	说明
12：10	LPFSEL	**ACMP LPF 选择** **ACMP0 和 ACMP1 共享相同的 LPF 设置，建议选择 1MHz 以获得更好的性能，并选择 200kHz 来过滤外部噪音** 111： 1MHz 110： 1MHz 101： 750kHz 100： 750kHz 011： 500kHz 010： 500kHz 001： 200kHz 000： 200kHz
13	HYST0	**ACMP0 迟滞使能位** 0： 禁用 1： 使能
14	HYST1	**ACMP1 迟滞使能位** 0：禁用 1：使能
15	MODE	**ACMP 模式选择** 0：低功耗模式（为了节省功耗） 1：正常模式

11.5 中断

表 11-2 ACMP 中断表

中断	位字段	使能	清除
ACMP0	ACMP0_SR[ACMP0_F0]	ACMP0_IE	写 1
ACMP0	ACMP0_SR[ACMP0_F1]	ACMP0_IE	写 1
ACMP0	ACMP0_SR[ACMP0_F2]	ACMP0_IE	写 1
ACMP0	ACMP0_SR[ACMP0_F3]	ACMP0_IE	写 1
ACMP0	ACMP0_SR[ACMP0_F4]	ACMP0_IE	写 1
ACMP0	ACMP0_SR[ACMP0_F5]	ACMP0_IE	写 1
ACMP0	ACMP0_SR[ACMP0_WUF]	ACMP0_IE	写 1
ACMP1	ACMP1_DSR[ACMP1_F]	ACMP1_IE	写 1
ACMP1	ACMP1_DSR[ACMP1_WUF]	ACMP1_IE	写 1

12 脉宽调制（PWM）

12.1 简介

PWM 模块是一个双通道到六通道的定时器，支持输入捕获、输出比较和 PWM 信号的生成，以控制电机和电源管理应用。PWM 的计数功能是通过一个 16 位的计数器产生的。

该设备包含四个 PWM 模块，一个具有全功能的 6 通道 PWM 和 3 个基本功能的双通道 PWM。每个 PWM 模块都可以使用独立的外部时钟作为时钟源输入。下表概述了 PWM 模块的配置。

表 12-1 PWM 模块配置

特性	PWM0/PWM1/PWM3	PWM2
通道数	2	6
周期 TOF	是	是
输入捕获模式	是	是
通道输入滤波器	否	通道 0/1/2/3
输出比较模式	是	是
边沿对齐 PWM	是	是
中心对齐 PWM	是	是
组合模式	是	是
互补模式	是	是
同步	是	是
反相	是	是
软件输出控制	是	是
死区时间插入	否	是
输出屏蔽	是	是
故障控制	否	是
故障输入数量	否	4
故障输入滤波器	否	故障输入 0/1/2/3
极性控制	是	是
初始化	是	是
通道匹配触发器	是	是
初始化触发器	是	是
捕获测试模式	是	是
双边沿捕获模式	是	是
正交解码器模式	是	是
正交解码器输入滤波器	是	是

12.1.1 PWM 特性

PWM 特性包括：

- PWM 时钟源可选；

  时钟源可以是系统时钟、内部 RC 时钟或外部时钟；

  选择外部时钟将 PWM 时钟连接到芯片级输入引脚，因此允许 PWM 计数器与片外时钟源同步；

- 16 位预分频器支持 1，2，3 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；
- 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， 可以在匹配时输出 0,1 或者输出反转；
- 所有通道可以配置为中心对齐的 PWM 模式；
- 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，或者作为独立的通道输出信号
- 死区插入可用于每一对互补通道；
- 生成匹配触发器；
- 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 4 个故障输入；
- 每个通道的极性是可配置的；
- 每个通道产生一个中断；
- 当计数器溢出时，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；
- 对关键寄存器的写保护；
- 用于脉冲和 周期宽度测量的双边沿捕获；
- 正交解码器具有输入滤波器、 相对位置计数和位置计数中断，或外部事件位置计数捕获。

  （AB 相输入引脚映射到每个 PWM 模块的 CH0 和 CH1）

12.1.2 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（0-5）。

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。

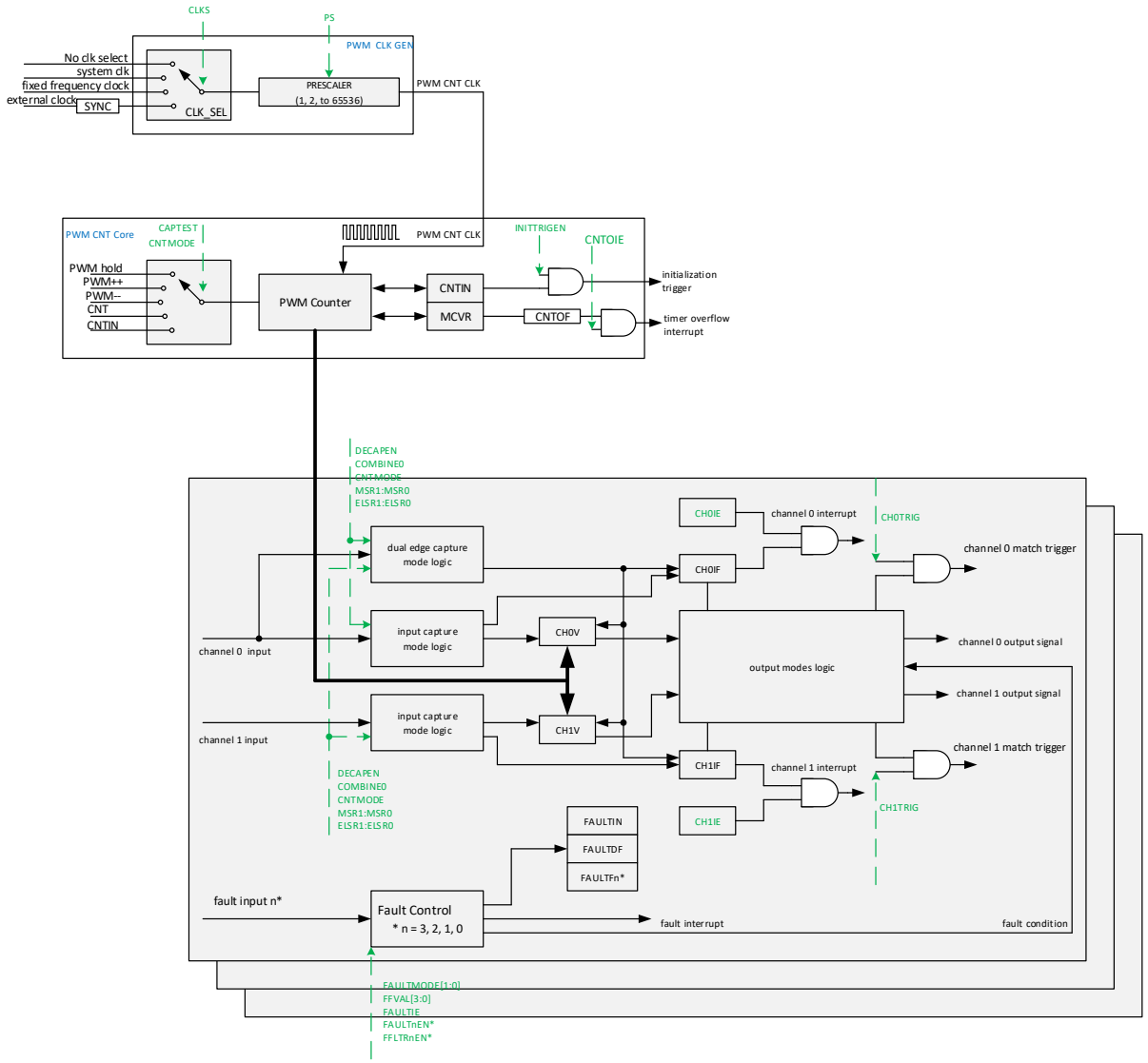

图 12-1 PWM 框图

12.2 寄存器定义

表 12-2 PWM 寄存器映像和复位值

PWM0 基地址：（0x40013000h）
PWM1 基地址：（0x40014000h）
PWM2 基地址：（0x40015000h）
PWM3 基地址：（0x4001e000h）
地址 = 基地址 + 偏移地址

模块名	基地址	偏移地址
PWM0	0x40013000	0x00 ~ 0x94
PWM1	0x40014000	0x00 ~ 0x94
PWM2	0x40015000	0x00 ~ 0x94
PWM3	0x4001e000	0x00 ~ 0x94

偏移地址	名称	宽度	寄存器功能
0x00	PWMx_INIT	32	PWM 初始化寄存器
0x04	PWMx_CNT	32	PWM 计数器寄存器
0x08	PWMx_MCVR	32	最大计数值寄存器
0x0C	PWMx_CH0SCR	32	Channel（0） 状态和控制寄存器
0x10	PWMx_CH0V	32	Channel（0）值
0x14	PWMx_CH1SCR	32	Channel（1） 状态和控制寄存器
0x18	PWMx_CH1V	32	Channel（1） 值
0x1C	PWMx_CH2SCR	32	Channel（2） 状态和控制寄存器
0x20	PWMx_CH2V	32	Channel（2） 值
0x24	PWMx_CH3SCR	32	Channel（3） 状态和控制寄存器
0x28	PWMx_CH3V	32	Channel（3） 值
0x2C	PWMx_CH4SCR	32	Channel（4） 状态和控制寄存器
0x30	PWMx_CH4V	32	Channel（4） 值
0x34	PWMx_CH5SCR	32	Channel（5） 状态和控制寄存器
0x38	PWMx_CH5V	32	Channel（5） 值
0x4C	PWMx_CNTIN	32	计数器初始值寄存器
0x50	PWMx_STR	32	捕获和比较状态寄存器
0x54	PWMx_FUNCSEL	32	功能模式选择寄存器
0x58	PWMx_SYNC	32	同步寄存器
0x5C	PWMx_OUTINIT	32	通道输出的初始状态寄存器
0x60	PWMx_OMCR	32	输出屏蔽控制寄存器
0x64	PWMx_MODESEL	32	模式选择寄存器
0x68	PWMx_DTSET	32	死区设置寄存器
0x6C	PWMx_EXTTRIG	32	PWM 外部触发器
0x70	PWMx_CHOPOLCR	32	通道输出极性控制寄存器
0x74	PWMx_FDSR	32	故障检测状态寄存器
0x78	PWMx_CAPFILTER	32	输入捕获滤波器控制
0x7C	PWMx_FFAFER	32	故障滤波和使能寄存器
0x80	PWMx_QEI	32	正交解码器控制和状态寄存器
0x84	PWMx_CONF	32	配置寄存器
0x88	PWMx_FLTPOL	32	PWM 故障输入极性寄存器

PWM1 基地址：（0x40014000h）
PWM2 基地址：（0x40015000h）
PWM3 基地址：（0x4001e000h）
地址 = 基地址 + 偏移地址

模块名	基地址	偏移地址
PWM0	0x40013000	0x00 ~ 0x94
PWM1	0x40014000	0x00 ~ 0x94
PWM2	0x40015000	0x00 ~ 0x94
PWM3	0x4001e000	0x00 ~ 0x94

0x8C	PWMx_SYNCONF	32	同步配置寄存器
0x90	PWMx_INVCR	32	PWM 反相控制寄存器
0x94	PWMx_CHOSWCR	32	通道软件输出控制寄存器

0x00　　PWMx_INIT　　PWM 初始化寄存器　　00000000

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称									CLKPSC[15：8]							
类型									RW							
复位									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	CLKPSC[7：0]								CNTOF	CNTOIE	CNTMODE	CLKSRC				
类型	RW								RO	RW	RW	RW				
复位	0	0	0	0	0	0	0	0	0	0	0	0	0			

位	名称	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系统时钟周期的时钟源。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** 当 PWM 计数器值超过 MCVR 寄存器中的值时，由硬件设置。当 CNTOF 置 1 时，通过读取 INIT 寄存器来清除 CNTOF 位，然后将 0 写入 CNTOF 位。向 CNTOF 写入 1 不起作用。如果在读和写操作之间发生另一个 PWM 溢出，则写操作不起作用。因此，CNTOF 保持设置状态，表示发生了溢出。在这种情况下，由于先前 CNTOF 的清除顺序，CNTOF 请求中断不会丢失。 0： PWM 计数器没有溢出 1： PWM 计数器溢出
6	CNTOIE	**定时器溢出中断使能** 使能 PWM 溢出中断 0： 禁用 CNTOF 中断，使用软件轮询； 1： 使能 CNTOF 中断，当 CNTOF 等于 1 时，产生中断
5	CNTMODE	**中心对齐 PWM 选择** 选择 CPWM 模式。此模式将 PWM 配置为在上-下计数模式下运行。该字段写保护。它只能在 FUNCSEL[WPDIS] = 1 时写入。 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。
4：3	CLKSRC	**时钟源选择**

位	名称	说明
		选择三个 PWM 计数器时钟源中的一个。该字段为写保护。它仅在 FUNCSEL[WPDIS] = 1 时可写。 00： 没有选择任何时钟。这实际上禁用了 PWM 计数器； 01： 系统时钟 10 ：固定频率时钟 11： 外部时钟

0x04 **PWMx_CNT** **PWM 计数器值** **00000000**

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

位	名称	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。

0x08 **PWMx_MCVR** **最大计数值寄存器** **00000000**

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

位	名称	说明
15：0	MCVR	**最大计数值寄存器** 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。

0x0C **PWMx_CH0SCR** **Channel（0） 状态和控制寄存器** **00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型																
复位																
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**		

0x0C **PWMx_CH0SCR** **Channel（0） 状态和控制寄存器** **00000000**

类型									RO	RW	RW	RW	RW	RW		
复位									0	0	0	0	0	0		

位	名称	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 CSC 寄存器（当 CHnIF 置 1 时），然后将 CHIF 位置 1 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，则写入操作无效。因此，CHIF 保持设置状态，标志着一个事件已经发生。在这种情况下，由于先前 CHIF 的清除序列， CHIF 中断请求不会丢失。 0： 没有发生通道事件 1： 通道事件已经发生
6	CHIE	**通道中断使能** 使能通道中断 0： 禁用通道中断 1： 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。

0x10 **PWMx_CH0V** **Channel（0） 值** **00000000**

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

位	名称	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。

0x4C **PWMx_CNTIN** **计数器初始值** **00000000**

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

0x4C **PWMx_CNTIN** **计数器初始值** **00000000**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	CNTINIT															
类型	RW															
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	名称	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。

0x50 **PWMx_STR** **捕获和比较状态寄存器** **00000000**

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

位	名称	说明
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
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

0x54 **PWMx_FUNCSEL** **功能模式选择** **00000004**

0x54 **PWMx_FUNCSEL** **功能模式选择** **00000004**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型																
复位																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									FAULTIE	FAULTMODE		CAPTEST	PWMSYNC	WPDIS	INIT	PWMEN2
类型									RW	RW		RW	RW	RW	RW	RW
复位									0	0	0	0	0	1	0	0

位	名称	说明
7	FAULTIE	**错误中断使能** 当 PWM 检测到故障，并使能 PWM 故障控制时，允许产生中断。. 0： 禁用故障控制中断 1： 使能故障控制中断
6：5	FAULTMODE	**故障控制模式** 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。 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
4	CAPTEST	**捕获测试模式使能** 使能捕获测试模式。该字段为写保护。它只能在 FUNCSEL[WPDIS] = 1 时可写。 0： 不支持捕获测试模式 1： 支持捕获测试模式
3	PWMSYNC	**PWM 同步模式** 选择 MCVR，CHnV，OMCR 和 PWM 计数器同步可以使用的触发器。 当 SYNCMODE 为 0 时，PWMSYNC 位配置同步。 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。
2	WPDIS	**禁用写保护** 当使能写保护时（WPDIS = 0），写保护位不能被写入。当禁用写保护时（WPDIS = 1），写保护位可以写入。WPDIS 位是 WPEN 位取反。当 WPEN 位写入 1 时，清除 WPDIS。当 WPEN 位读为 1，然后将 1 写入 WPDIS，则置位 WPDIS。 向 WPDIS 写入 1 不起作用。 0： 使能写保护 1： 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	PWMEN2	**PWM 使能** 该字段写保护。它仅能在 FUNCSEL[WPDIS] = 1 时可写。 0：只有 TPM 兼容的寄存器（第一组寄存器） 可以无限制的使用。不要使用 PWM 专用寄存器 1： 所有寄存器包括 PWM 专用寄存器（第二组寄存器） 均可无限制使用。

0x58 **PWMx_SYNC** **同步寄存器** **00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型																
复位																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称					PWM_SYNCPOL	PWM_TRIG2_CLEAR_CNT	PWM_TRIG1_CLEAR_CNT	PWM_TRIG0_CLEAR_CNT	SWSYNC	SWTRIG	PWM0TRIG	ACMPTRIG	OMSYNCP	REINIT	MAXSYNCP	MINSYNCP
类型					RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
复位					0	0	0	0	0	0	0	0	0	0	0	0

位	名称	说明
11	PWM_SYNCPOL	**PWM_SYNCPOL** 选择何时使用缓冲区的值更新 POL 寄存器 0： POL 寄存器在系统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。
10	PWM_TRIG2_CLEAR_CNT	**PWM_TRIG2_CLEAR_CNT** 0： 当 TRIG2 事件发生时，不清除 CNT 计数器的值 1： 当 TRIG2 事件发生时，清除 CNT 计数器的值
9	PWM_TRIG1_CLEAR_CNT	**PWM_TRIG1_CLEAR_CNT** 0：当 TRIG1 事件发生时，不清除 CNT 计数器的值 1：当 TRIG1 事件发生时，清除 CNT 计数器的值
8	PWM_TRIG0_CLEAR_CNT	**PWM_TRIG0_CLEAR_CNT** 0：当 TRIG0 事件发生时，不清除 CNT 计数器的值 1：当 TRIG0 事件发生时，清除 CNT 计数器的值
7	SWSYNC	**PWM 同步软件触发器** 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。 0 ： 未选择软件触发 1 ： 选择软件触发
6	SWTRIG	**PWM 同步硬件触发器 2** 使能 PWM 同步的硬件触发 2， 在触发 2 输入信号中检测到上升沿时，产生硬件触发 2。 0： 触发禁用 1： 触发使能
5	PWM0TRIG	**PWM 同步硬件触发器 1** 使能 PWM 同步的硬件触发 1， 在触发 1 输入信号中检测到上升沿时，产生硬件触发 1。 0： 触发禁用 1： 触发使能
4	ACMPTRIG	**PWM 同步硬件触发器 0** 使能 PWM 同步的硬件触发 0， 在触发 0 输入信号中检测到上升沿时，产生硬件触发 0。 0： 触发禁用 1： 触发使能
3	OMSYNCP	**输出掩码同步** 选择何时使用缓冲区的值更新 OMCR 寄存器 0 ：OMCR 寄存器在系统时钟的所有上升沿使用其缓冲区的值进行更新；

位	名称	说明
		1：OMCR 寄存器仅通过 PWM 同步单元更新其缓冲区的值。
2	REINIT	**PWM 计数器通过同步重新初始化** 在检测到所选同步触发器时，确定是否重新初始化 PWM 计数器 。当 SYNCMODE 为 0 时，REINIT 位配置同步。 0 ：PWM 计数器继续正常计数 1： 当检测到所选触发时，PWM 计数器将更新为其初始值。
1	MAXSYNCP	**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： 禁用最大加载点 1： 使能最大加载点
0	MINSYNCP	**使能最小加载点** 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。 0： 禁用最小加载点 1： 使能最小加载点

0x5C **PWMx_OUTINIT** **通道输出的初始状态** **00000000**

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

位	名称	说明
5	CH5OIV	**通道 5 输出初始值** 进行初始化时，强制进入通道输出的值 0： 初始值为 0. 1： 初始值为 1.
4	CH4OIV	**通道 4 输出初始值** 进行初始化时，强制进入通道输出的值 0： 初始值为 0. 1： 初始值为 1.
3	CH3OIV	**通道 3 输出初始值** 进行初始化时，强制进入通道输出的值 0： 初始值为 0. 1： 初始值为 1.
2	CH2OIV	**通道 2 输出初始值** 进行初始化时，强制进入通道输出的值 0： 初始值为 0. 1： 初始值为 1.
1	CH1OIV	**通道 1 输出初始值**

位	名称	说明
		进行初始化时，强制进入通道输出的值 0： 初始值为 0. 1： 初始值为 1.
0	CH0OIV	**通道 0 输出初始值** 进行初始化时，强制进入通道输出的值 0： 初始值为 0. 1： 初始值为 1.

0x60 **PWMx_OMCR** **输出屏蔽控制寄存器** **00000000**

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

位	名称	说明
5	CH5OMEN	**通道 5 输出屏蔽** 定义通道输出是否被屏蔽或取消屏蔽 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
4	CH4OMEN	**通道 4 输出屏蔽** 定义通道输出是否被屏蔽或取消屏蔽 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
3	CH3OMEN	**通道 3 输出屏蔽** 定义通道输出是否被屏蔽或取消屏蔽 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
2	CH2OMEN	**通道 2 输出屏蔽** 定义通道输出是否被屏蔽或取消屏蔽 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
1	CH1OMEN	**通道 1 输出屏蔽** 定义通道输出是否被屏蔽或取消屏蔽 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
0	CH0OMEN	**通道 0 输出屏蔽** 定义通道输出是否被屏蔽或取消屏蔽 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

0x64 **PWMx_MODESEL** **PWM 功能模式选择** **00000000**

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

0x64 **PWMx_MODESEL** **PWM 功能模式选择** **00000000**

名称										PAIR2FAULTEN	PAIR2SYNCEN	PAIR2DTEN	PAIR2DECAP	PAIR2DECAPEN	PAIR2COMPEN	PAIR2COMBINEN
类型										RW	RW	RW	RW	RW	RW	RW
复位										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
类型		RW	RW	RW	RW	RW	RW	RW		RW	RW	RW	RW	RW	RW	RW
复位		0	0	0	0	0	0	0		0	0	0	0	0	0	0

位	名称	说明
22	PAIR2FAULTEN	**故障控制使能（n = 4）** 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 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
21	PAIR2SYNCEN	**同步使能（n = 4）** 使能寄存器 CH（n）V 和 CH（n+1）V 的 PWM 同步 0 ：禁用此通道对中的 PWM 同步 1： 使能此通道对中的 PWM 同步
20	PAIR2DTEN	**死区使能（n = 4）** 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 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
19	PAIR2DECAP	**双边沿捕获模式捕获（n = 4）** 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 PWMEN2 = 1 且 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。 0： 双边沿捕获无效 1： 双边沿捕获有效
18	PAIR2DECAPEN	**双边沿捕获模式使能（n = 4）** 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。该字段仅在 PWMEN2 = 1 时适用。该字段为写保护。它只能在 FUNCSEL[WPDIS] = 1 时可写入。 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。
17	PAIR2COMPEN	**通道互补（n = 4）** 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 ： 通道（n+1） 输出和通道（n）输出相同

位	名称	说明
		1： 通道（n+1） 输出是通道（n）输出互补
16	PAIR2COMBINEN	**组合通道（n = 4）** 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。 0： 通道（n）和（n+1） 是独立的 1： 通道（n）和（n+1） 是组合的
14	PAIR1FAULTEN	**故障控制使能（n = 2）** 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。 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
13	PAIR1SYNCEN	**同步使能（n = 2）** 使能寄存器 CH（n）V 和 CH（n+1）V 的 PWM 同步 0： 禁用此通道对中的 PWM 同步 1：使能此通道对中的 PWM 同步
12	PAIR1DTEN	**死区使能（n = 2）** 使能通道（n） and（n+1）的死区插入，该字段写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。 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
11	PAIR1DECAP	**双边沿捕获模式捕获（n = 2）** 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 PWMEN2 = 1 且 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。 0： 双边沿捕获无效 1： 双边沿捕获有效
10	PAIR1DECAPEN	**双边沿捕获模式使能（n = 2）** 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。该字段仅在 PWMEN2 = 1 时适用。该字段为写保护。它只能在 FUNCSEL[WPDIS] = 1 时可写入。 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。
9	PAIR1COMPEN	**Channel（n） 互补（n = 2）** 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： 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
8	PAIR1COMBINEN	**组合通道（n = 2）** 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。 0： 通道（n）和（n+1） 是独立的 1： 通道（n）和（n+1） 是组合的
6	PAIR0FAULTEN	**故障控制使能（n = 0）** 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。 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
5	PAIR0SYNCEN	**同步使能（n = 0）** 使能寄存器 CH（n）V 和 CH（n+1）V 的 PWM 同步 0： 禁用此通道对中的 PWM 同步 1： 使能此通道对中的 PWM 同步
4	PAIR0DTEN	**死区使能（n = 0）**

位	名称	说明
		使能通道（n） and（n+1）的死区插入，该字段写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。 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
3	PAIR0DECAP	**双边沿捕获模式捕获（n = 0）** 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 PWMEN2 = 1 且 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。 0： 双边沿捕获无效 1： 双边沿捕获有效
2	PAIR0DECAPEN	**双边沿捕获模式使能（n = 0）** 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。该字段仅在 PWMEN2 = 1 时适用。该字段为写保护。它只能在 FUNCSEL[WPDIS] = 1 时可写入。 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。
1	PAIR0CMPEN	**Channel（n）互补（n = 0）** 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 ： 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
0	PAIR0COMBINEN	**组合通道（n = 0）** 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。 0 ： 通道（n）和（n+1） 是独立的 1 ： 通道（n）和（n+1） 是组合的

0x68 **PWMx_DTSET** **死区插入控制** **00000000**

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

位	名称	说明
7：6	DTPSC	**死区预分频值** 选择系统时钟的分频因子。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。 0x ： 系统时钟 1 分频. 10 ： 系统时钟 4 分频. 11： 系统时钟 16 分频.
5：0	DTVAL	**死区值** 选择死区计数器的死区插入值 。死区计数器由系统时钟的缩放版本计时。请参考 DTPS 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

位	名称	说明
		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。

0x6C **PWMx_EXTTRIG** **PWM 外部触发器** **00000000**

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

位	名称	说明
7	TRIGF	**通道触发标志** 生成通道触发器时由硬件置 1。 通过在设置 TRIGF 时读取 EXTTRIG ，然后将 0 TRIGF 来清除 TRIGF 。向 TRIGF 中写入 1 不起作用。如果在清除序列完成之前生成另一个通道触发，此序列将复位。则在较早的 TRIGF 的清除序列完成后，TRIGF 保持置位。 0： 没有生成通道触发 1： 产生通道触发
6	INITTRIGEN	**使能初始化触发器** 当 PWM 计数器等于 CNTIN 寄存器时，允许产生触发 0： 禁用初始化触发 1： 使能初始化触发
5	CH5TRIG	**通道 5 触发使能** 当 PWM 计数器等于 CHnV 寄存器时， 允许产生通道触发 0：禁用通道触发 1：使能通道触发
4	CH4TRIG	**通道 4 触发使能** 当 PWM 计数器等于 CHnV 寄存器时， 允许产生通道触发 0：禁用通道触发 1：使能通道触发
3	CH3TRIG	**通道 3 触发使能** 当 PWM 计数器等于 CHnV 寄存器时， 允许产生通道触发 0：禁用通道触发 1：使能通道触发
2	CH2TRIG	**通道 2 触发使能** 当 PWM 计数器等于 CHnV 寄存器时， 允许产生通道触发 0：禁用通道触发 1：使能通道触发
1	CH1TRIG	**通道 1 触发使能** 当 PWM 计数器等于 CHnV 寄存器时， 允许产生通道触发 0：禁用通道触发 1：使能通道触发
0	CH0TRIG	**通道 0 触发使能**

位	名称	说明
		当 PWM 计数器等于 CHnV 寄存器时， 允许产生通道触发 0 ： 禁用通道触发 1 ： 使能通道触发

0x70 **PWMx_CHOPOLCR** **通道输出极性控制寄存器** **00000000**

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

位	名称	说明
5	**CH5POL**	**通道 5 极性** 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 ： 通道极性为高电平有效 1： 通道极性为低电平有效
4	**CH4POL**	**通道 4 极性** 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 ： 通道极性为高电平有效 1： 通道极性为低电平有效
3	**CH3POL**	**通道 3 极性** 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 ： 通道极性为高电平有效 1： 通道极性为低电平有效
2	**CH2POL**	**通道 2 极性** 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 ： 通道极性为高电平有效 1： 通道极性为低电平有效
1	**CH1POL**	**通道 1 极性** 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 ： 通道极性为高电平有效 1： 通道极性为低电平有效
0	**CH0POL**	**通道 0 极性** 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 ： 通道极性为高电平有效 1： 通道极性为低电平有效

0x74 **PWMx_FDSR** **故障检测状态寄存器** **00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型																
复位																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									**FAULTDF**	**WPEN**	**FAULTIN**		**FAULTDF3**	**FAULTDF2**	**FAULTDF1**	**FAULTDF0**
类型									RO	RW	RO		RO	RO	RO	RO
复位									0	0	0		0	0	0	0

位	名称	说明
7	FAULTDF	**故障检测标志** 表示各 FAULTDF j 位的逻辑或，其中 j = 3, 2, 1, 0。通过在 FAULTDF 置 1 时读取 FDSR 寄存器来清除 FAULTDF，然后在使能的故障输入处没有现有故障条件时将 FAULTDF 置为 0。将 FAULTDF 置为 1 不起作用。 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问题，则该序列将复位 ，因此在更早的故障条件下当完成序列清除后，FAULTDF 保持置位状态。当 FAULTDF j 位被单独清除后，FAULTDF 也会被清 0. 0： 没有检测到故障状况 1 ： 监测到故障状况
6	WPEN	**写保护使能** WPEN 位和 i WPDIS 位互反。在写入 1 时，WPEN 置位。当 WPEN 位读为 1 是清除 WPEN 位，然后 WPDIS 位置 1。将 WPEN 位置为 0 不起作用。 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
5	FAULTIN	**故障输入** 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 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.
3	FAULTDF3	**故障检测标志 3** 使能故障控制时由硬件置 1，使能相应的故障输入，在故障输入处检测到故障状况。通过在 FAULTDF3 置 1 时读取 FDSR 寄存器来清除 FAULTDF3，然后在相应的故障输入处没有现有故障条件时将 FAULTDF3 置为 0。将 FAULTDF3 置为 1 不起作用。当 FAULTDF 位被清 0 时，FAULTDF3 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问题，则该序列将复位 ，因此在更早的故障条件下当完成序列清除后，FAULTDF3 保持置位状态。 0 ： 在故障输入处没有检测到故障状况 1： 在故障输入处检测到故障状况
2	FAULTDF2	**故障检测标志 2** 使能故障控制时由硬件置 1，使能相应的故障输入，在故障输入处检测到故障状况。通过在 FAULTDF2 置 1 时读取 FDSR 寄存器来清除 FAULTDF2，然后在相应的故障输入处没有现有故障条件时将 FAULTDF2 置为 0。将 FAULTDF2 置为 1 不起作用。当 FAULTDF 位被清 0 时，FAULTDF 2 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问题，则该序列将复位 ，因此在更早的故障条件下当完成序列清除后，FAULTDF2 保持置位状态。 0 ： 在故障输入处没有检测到故障状况 1： 在故障输入处检测到故障状况
1	FAULTDF1	**故障检测标志 1** 使能故障控制时由硬件置 1，使能相应的故障输入，在故障输入处检测到故障状况。通过在 FAULTDF1 置 1 时读取 FDSR 寄存器来清除 FAULTDF1，然后在相应的故障输入处没有现有故障条件时将 FAULTDF1 置为 0。将 FAULTDF 1 置为 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问题，则该序列将复位 ，因此在更早的故障条件下当完成序列清除后，

位	名称	说明
		FAULTDF1 保持置位状态。 0： 在故障输入处没有检测到故障状况 1： 在故障输入处检测到故障状况
0	FAULTDF0	**故障检测标志 0** 使能故障控制时由硬件置 1，使能相应的故障输入，在故障输入处检测到故障状况。通过在 FAULTDF0 置 1 时读取 FDSR 寄存器来清除 FAULTDF0，然后在相应的故障输入处没有现有故障条件时将 FAULTDF0 置为 0。将 FAULTDF 0 置为 1 不起作用。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问题，则该序列将复位 ，因此在更早的故障条件下当完成序列清除后，FAULTDF0 保持置位状态。 0： 在故障输入处没有检测到故障状况 1： 在故障输入处检测到故障状况

0x78 **PWMx_CAPFILTER** **输入捕获滤波器控制** **00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称													**CH3CAPFVAL[4：1]**			
类型													RW			
复位													0	0	0	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	**CH3FVAL[0：0]**	**CH2CAPFVAL**					**CH1CAPFVAL**					**CH0CAPFVAL**				
类型	RW	RW					RW					RW				
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	名称	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

0x7C **PWMx_FFAFER** **故障滤波器和故障使能寄存器** **00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型																
复位																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称					**FFVAL**				**FF3EN**	**FF2EN**	**FF1EN**	**FF0EN**	**FER3EN**	**FER2EN**	**FER1EN**	**FER0EN**
类型					RW				RW	RW	RW	RW	RW	RW	RW	RW
复位					0	0	0	0	0	0	0	0	0	0	0	0

位	名称	说明
11：8	**FFVAL**	**故障输入滤波器**

位	名称	说明
		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
7	**FF3EN**	**故障输入 3 滤波器使能** 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。 0： 禁用故障输入滤波器 1： 使能故障输入滤波器
6	**FF2EN**	**故障输入 2 滤波器使能** 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。 0： 禁用故障输入滤波器 1： 使能故障输入滤波器
5	**FF1EN**	**故障输入 1 滤波器使能** 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。 0： 禁用故障输入滤波器 1： 使能故障输入滤波器
4	**FF0EN**	**故障输入 0 滤波器使能** 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。 0： 禁用故障输入滤波器 1： 使能故障输入滤波器
3	**FER3EN**	**故障输入 3 使能** 使能故障输入。该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。 0：禁用故障输入 1：使能故障输入
2	**FER2EN**	**故障输入 2 使能** 使能故障输入。该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。 0：禁用故障输入 1：使能故障输入
1	**FER1EN**	**故障输入 1 使能** 使能故障输入。该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。 0：禁用故障输入 1：使能故障输入
0	**FER0EN**	**故障输入 0 使能** 使能故障输入。该字段为写保护，它只能在 FUNCSEL[WPDIS] = 1 时可写。 0：禁用故障输入 1：使能故障输入

0x80 **PWMx_QEI** **正交解码器接口配置寄存器** **00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型																
复位																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									**PHAFLTREN**	**PHBFLTREN**	**PHAPOL**	**PHBPOL**	**QUADMODE**	**QUADIR**	**TOFDIR**	**QEIEN**
类型									RW	RW	RW	RW	RW	RO	RO	RW
复位									0	0	0	0	0	0	0	0

位	名称	说明
7	PHAFLTREN	**A 相 输入滤波器使能** 启用正交解码器 A 相输入滤波器。A 相输入的滤波器值由 CAPFILTER 的 CH0CAPFVAL 字段所定义。当 CH0CAPFVAL 为 0 时，A 相滤波器也被禁用。 0： 禁用 A 相输入滤波器 1： 使能 A 相输入滤波器
6	PHBFLTREN	**B 相 输入滤波器使能** 启用正交解码器 B 相输入滤波器。B 相输入的滤波器值由 FILTER 的 CH1CAPFVAL 字段所定义。当 CH1CAPFVAL 为 0 时，B 相滤波器也被禁用。 0： 禁用 B 相输入滤波器 1： 使能 B 相输入滤波器
5	PHAPOL	**A 相 输入极性** 选择正交解码器 A 相输入的极性 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。
4	PHBPOL	**B 相 输入极性** 选择正交解码器 B 相输入的极性 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。
3	QUADMODE	**正交解码器模式** 选择正交解码器模式下使用的编码模式 0： A 相和 B 相编码模式 1： 计数和方向编码模式
2	QUADIR	**正交解码器模式下的 PWM 计数器方向** 表示计数方向。 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）.
1	TOFDIR	**正交解码器模式下的定时器溢出方向** 表示 TOF 位是否设置在计数的顶部或底部 0： TOF 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： TOF 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）
0	QEIEN	**正交解码器模式使能** 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。 0：禁用正交解码器模式 1：使能正交解码器模式

0x84 **PWMx_CONF** **配置寄存器** **00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称					EVENT5_PSC		EVENT4_PSC		EVENT3_PSC		EVENT2_PSC		EVENT1_PSC		EVENT0_PSC	
类型					RW		RW		RW		RW		RW		RW	
复位					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										CNTOFNUM						
类型										RW						
复位										0	0	0	0	0	0	0

位	名称	说明

位	名称	说明
27：26	EVENT5_PSC	**通道 5 输入事件 PSC** 00： 1 01： 2 10： 4 11： 8
25：24	EVENT4_PSC	**通道 4 输入事件 PSC** 00： 1 01： 2 10： 4 11： 8
23：22	EVENT3_PSC	**通道 3 输入事件 PSC** 00： 1 01： 2 10： 4 11： 8
21：20	EVENT2_PSC	**通道 2 输入事件 PSC** 00： 1 01： 2 10： 4 11： 8
19：18	EVENT1_PSC	**通道 1 输入事件 PSC** 00： 1 01： 2 10： 4 11： 8
17：16	EVENT0_PSC	**通道 0 输入事件 PSC** 00： 1 01： 2 10： 4 11： 8
6：0	CNTOFNUM	**TOF 频率** 选择计数器溢出次数与 CNTOF 位置 1 次数之间的比率 NUMTOF = 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

0x88 **PWMx_FLTPOL** **PWM 故障输入极性** **00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型																
复位																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称													**FLT3POL**	**FLT2POL**	**FLT1POL**	**FLT0POL**
类型													RW	RW	RW	RW
复位													0	0	0	0

位	名称	说明
3	FLT3POL	**故障输入 3 极性** 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： 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。
2	FLT2POL	**故障输入 2 极性** 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： 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。
1	FLT1POL	**故障输入 1 极性** 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： 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。
0	FLT0POL	**故障输入 0 极性** 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： 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。

0x8C **PWMx_SYNCONF** **同步配置** **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
类型										RW	RW	RW	RW	RW	RW	RW
复位										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
类型				RW	RW	RW	RW	RW	RW		RW	RW		RW		RW
复位				0	0	0	0	0	0		0	0		0		0

位	名称	说明
22	HWPOL	**通道 CHPOLCR 同步由硬件触发器激活** 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**

位	名称	说明
		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** 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** 选择 PWM 同步模式 0： 选择传统 PWM 同步 1 ： 选择增强 PWM 同步
5	SWOC	**CHOSWCR 寄存器同步** 0 ： 在系统时钟的所有上升沿，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：在系统时钟的所有上升沿，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 ：在系统时钟的所有上升沿，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.

0x90 **PWMx_INVCR** **PWM 反相控制寄存器** **00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型																
复位																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称														**PAIR2I**	**PAIR1I**	**PAIR0I**

0x90 **PWMx_INVCR** **PWM 反相控制寄存器** **00000000**

														NVEN	NVEN	NVEN
类型														RW	RW	RW
复位														0	0	0

位	名称	说明
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

0x94 **PWMx_CHOSWCR** **PWM 通道 软件输出控制寄存器** **00000000**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型																
复位																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称			CH5SWCV	CH4SWCV	CH3SWCV	CH2SWCV	CH1SWCV	CH0SWCV			CH5SWEN	CH4SWEN	CH3SWEN	CH2SWEN	CH1SWEN	CH0SWEN
类型			RW	RW	RW	RW	RW	RW			RW	RW	RW	RW	RW	RW
复位			0	0	0	0	0	0			0	0	0	0	0	0

位	名称	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

位	名称	说明
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
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

12.3 功能描述

12.3.1 时钟源

PWM 只有一个时钟域：系统时钟（定时器时钟）。

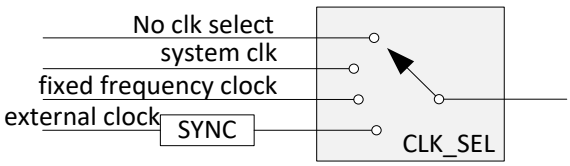

图 12-2 PWM 时钟源

PWMx_INIT 寄存器中的 CLKSRC[1：0]位选择 PWM 计数器的三个可能时钟源之一或禁用 PWM 计数器。在任意 MCU 复位后，CLKSRC[1：0] = 00，因此没有选择时钟源。可以随时读取或写入 CLKSRC [1：0]位。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，这个时钟源的频率为 8MHz 左右；同时也可以选择系统时钟或者外部时钟作为 PWM 计数器的时钟源。

外部时钟通过由系统时钟提供时钟同步器，以确保计数器转换与系统时钟转换正确对齐。因此，为了满足考虑抖动的奈奎斯特标准，外部时钟源的频率不得超过系统时钟频率的 1/4。

12.3.2 预分频器

所选的计数器时钟源通过一个预分频器（是一个 16 位计数器）。预分频器的值由 PWMx_INIT 寄存器 PWM_PSC [15：0]位选择。下图给出了一个预分频计数器和 PWM 计数器的示例。

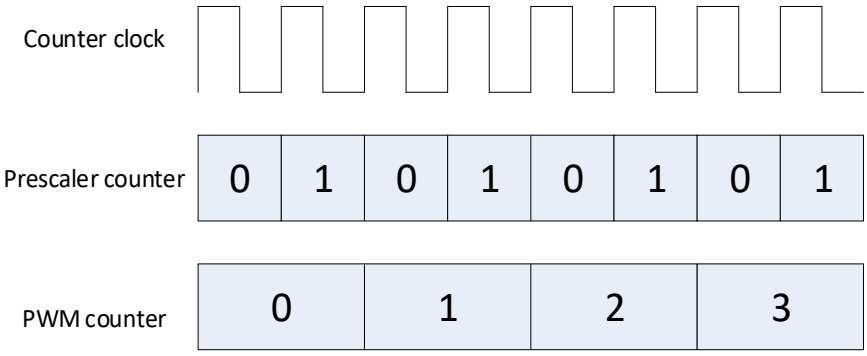

图 12-3 预分频器

12.3.3 计数器

PWM 有一个 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；
- 正交解码器模式。

12.3.3.1 向上计数

当 QEIEN=0 且 CNTMODE = 0 时，选择向上计数。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。使用向上计数时的 PWM 周期是（MCVR-CNTIN + 0x0001）×PWM 计数器时钟的周期。当 PWM 计数器从 MCVR 变为 CNTIN 时，TOF 位置 1。

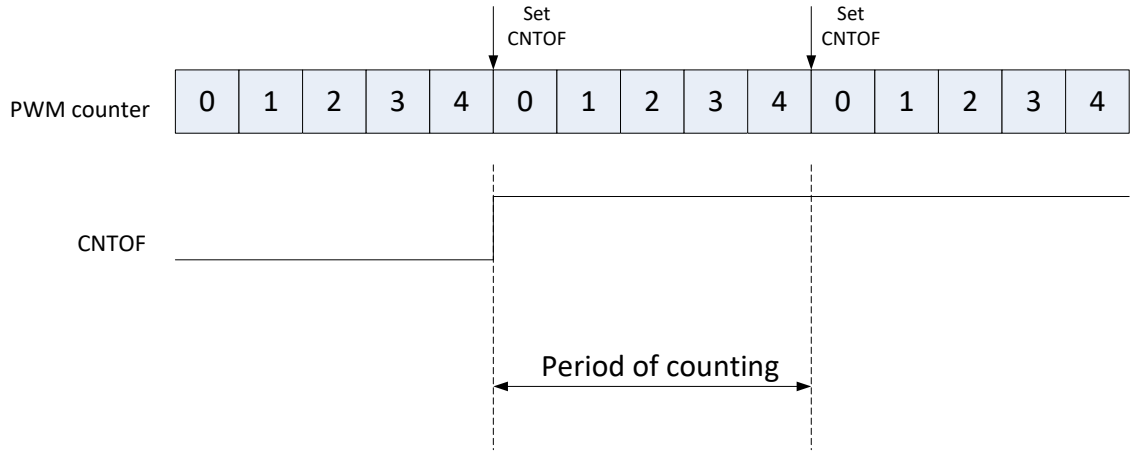

图 12-4 向上计数

12.3.3.2 向上-向下计数

当 QEIEN = 0 且 CNTMODE = 1 时，选择上下计数。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，并且上下计数重新开始。

使用上下计数时的 PWM 周期为 PWM 计数器时钟的 2×（MCVR-CNTIN）×周期。当 PWM 计数器从 MCVR 变为 MCVR - 1 时，TOF 位置 1，如下图所示。

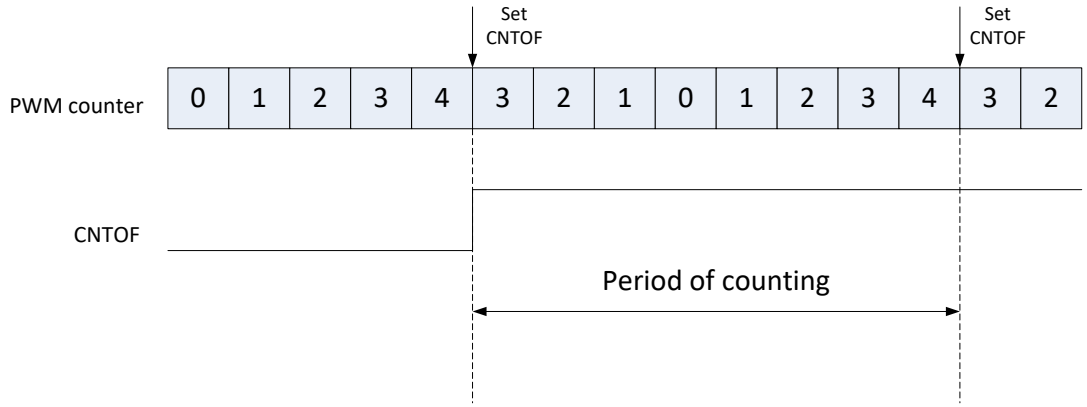

图 12-5 上下计数

12.3.4 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 12-3 工作模式配置

PAIRn DECAPEN	PAIRn COMBINEN	CNTMODE	MSR1：MSR0	ELSR1：ELSR0	工作模式	配置
0	0	0	00	01	输入捕获	仅在上升沿捕获
				10		仅在下降沿捕获
				11		在上升或下降沿捕获
			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	边沿对齐 PWM	High-true 脉冲（匹配时清除输出）
				X1		Low-true 脉冲（匹配时设置输出）
		1	XX	10	中心对齐 PWM	High-true 脉冲（匹配时清除输出）
				X1		Low-true 脉冲（匹配时设置输出）

PAIRn DECAPEN	PAIRn COMBINEN	CNTMODE	MSR1：MSR0	ELSR1：ELSR0	工作模式	配置
	1	0	XX	10	组合 PWM	High-true 脉冲（在 Channel（n） 匹配时置位，Channel（n+1） 匹配时清除）
				X1		Low-true 脉冲（在 Channel（n）匹配时清除，Channel（n+1） 匹配时置位）
1	0	0	X0	参考下表	双边沿单次捕获	单次捕获模式
			X1			持续捕获模式

ELSR1	ELSR0	通道端口使能	检测到的边沿
0	0	禁用	无边沿
0	1	使能	上升沿
1	0	使能	下降沿
1	1	使能	上升沿或下降沿

12.3.5 输入捕获模式

在以下情况下选择输入捕获模式：

- DECAPEN = 0；
- COMBINE = 0；
- CNTMODE = 0；
- MSnR1：MSnR0 = 0：0；
- ELSnR1：ELSnR0 != 0：0。

当通道输入上出现选定的边沿时，PWM 计数器的当前值会被捕获到 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，PWMxCHn 引脚为边沿敏感输入。ELSnR1：ELSnR0 控制位决定哪个边沿（下降沿或上升沿）触发输入捕获事件。注意，要正确检测的通道输入信号的最大频率为系统时钟除以 4，这是满足信号采样的奈奎斯特准则所必需的。在输入捕捉模式下，忽略写入 CHnV 寄存器。

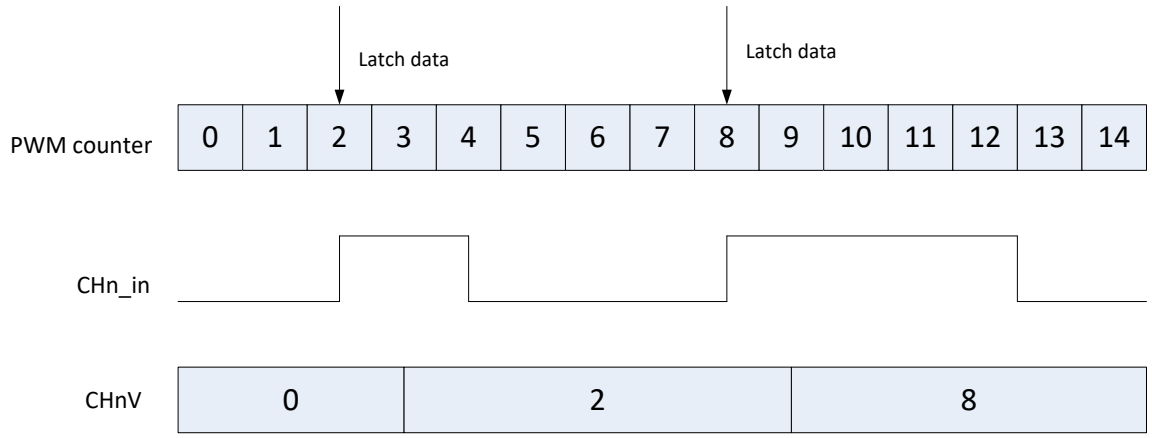

图 12-6 输入捕获模式

对于 PWM2 Channel 0 至 Channel3，存在额外的捕获滤波器来清除输入信号。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 个系统时钟，然后被传送到 输入捕获功能。

请注意，只有 PWM2 存在输入捕获滤波器功能。

12.3.6 输出比较模式

在以下情况下选择输出比较模式：

- DECAPEN = 0；
- COMBINE = 0；
- CNTMODE = 0, 且
- MSnR1：MSnR0 = 0：1。

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 寄存器中的值匹配时，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。

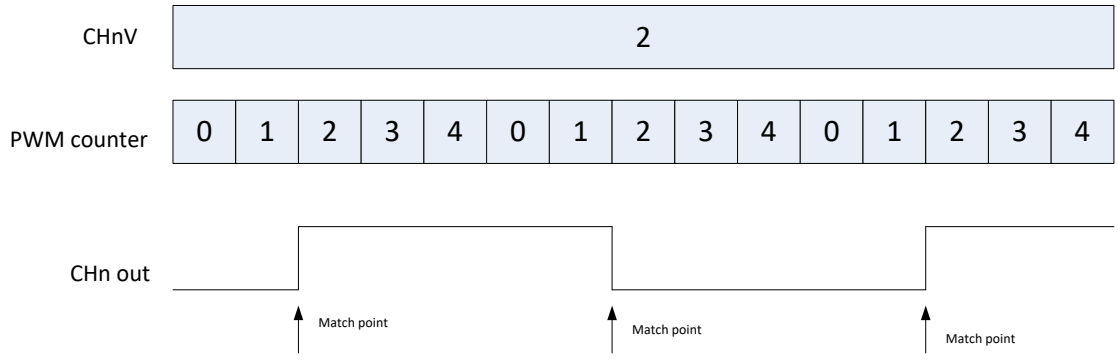

图 12-7 输出比较模式

12.3.7 边沿对齐 PWM（EPWM） 模式

在如下情况下选择边沿对齐模式：

- QEIEN = 0；
- DECAPEN = 0；
- COMBINE = 0；
- CNTMODE = 0, 且
- MSnR1 = 1。

EPWM 周期由 MCVR - CNTIN + 0x0001 确定，脉冲宽度（占空比） 由 CHnV – CNTIN 确定。如果 CHnIE = 1 且通道（n）匹配（PWM 计数器= CHnV），即脉冲宽度结束，则 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，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。

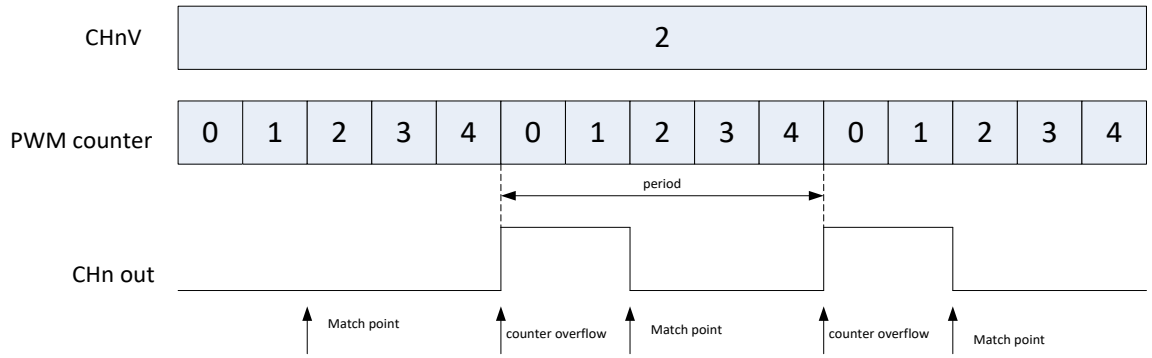

图 12-8 边沿对齐 PWM 模式

12.3.8 中心对齐 PWM（CPWM）模式

在如下情况下选择中心对齐模式：

- QEIEN = 0；
- DECAPEN = 0；
- COMBINE = 0, 且
- CNTMODE = 1。

CPWM 脉冲宽度（占空比） 由 2 ×（CHnV - CNTIN） 确定，周期由 2 ×（MCVR - CNTIN）确定，如下图所示。MCVR 必须保持在 0x0001 至 0x7FFF 的范围内，因为超出此范围的值会产生不确定的结果。在 CPWM 模式下，PWM 计数器向上计数直到达到 MCVR，然后向下计数直到达到 CNTIN。当 PWM 计数减少（脉冲宽度开始处）和当 PWM 计数增加（脉冲宽度结束时），在 channel（n） 匹配（PWM 计数器= CHnV）时，CHnIF 位置 1 且 channel（n）中断产生（如果 CHnIE = 1）。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。

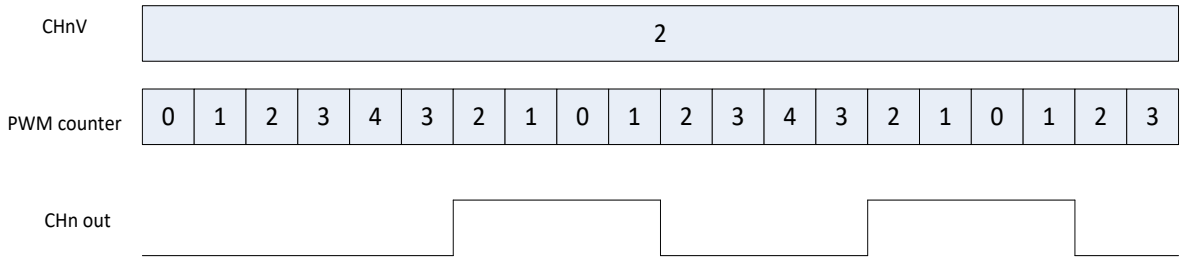

图 12-9 中心对齐 PWM 模式

12.3.9 组合模式

在如下情况下选择组合模式

- PWMEN2 = 1；
- QEIEN = 0；
- DECAPEN = 0；
- COMBINE = 1, 且
- CNTMODE = 0。

在组合模式下，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。在组合模式下，PWM 周期由（MCVR - CNTIN + 0x0001） 确定，且 PWM 脉冲宽度（占空比）由（|CH（n+1）V - CH（n）V|）确定。在 channel（n） 匹配时（PWM 计数器 = CH（n）V），如果 CHnIE = 1，则 CHnIF 位置 1 且产生 channel（n） 中断。在 channel（n+1） 匹配时（PWM 计数器 = CH（n+1）V），如果 CH（n+1）IE = 1，则 CH（n+1）IF 位置 1 且产生 channel（n+1）中断。

若 ELSnR1：ELSnR0 = 1：0，则在 channel（n） 匹配时（PWM 计数器 = CH（n）V），被强制拉至高电平，在 channel（n+1）匹配时候强制拉至低电平。

若 ELSnR1：ELSnR0 = X：1，则在此周期开始（PWM 计数器 = CNTIN）且 channel（n+1） 匹配（PWM 计数器 = CH（n+1）V）时候，输出电平被强制拉高。在 channel（n） 匹配时（PWM 计数器 = CH（n）V），其电平被强制拉低。

在 组合模式下，ELS（n+1）R1 和 ELS（n+1）R0 位不用于产生 channels（n） 和（n+1） 输出。

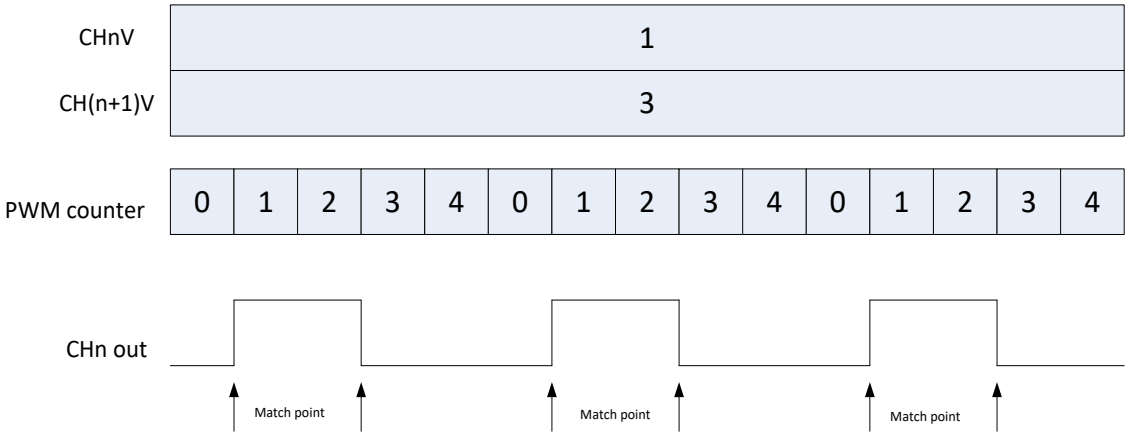

图 12-10 组合模式

12.3.10 互补模式

在如下情况下选择互补模式：

- PWMEN2 = 1；
- QEIEN = 0；
- DECAPEN = 0；
- COMBINE = 1；
- CNTMODE = 0；
- COMP = 1。

在互补模式下， channel（n+1） 输出和 channel（n）输出电平相反。如果（PWMEN2 = 1），（QEIEN = 0），（DECAPEN = 0），（COMBINE = 1），（CNTMODE = 0） 且（COMP = 0），则 channel（n+1） 输出和 channel（n） 输出相同。

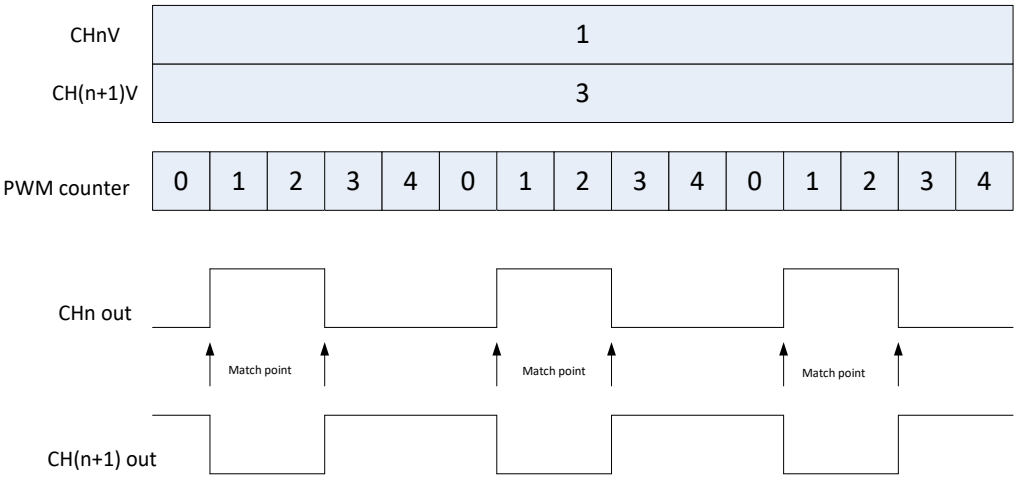

图 12-11 互补模式

12.3.11 写缓存更新的寄存器

12.3.11.1 PWM_CNTIN 寄存器更新缓存

表 12-4 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 PWM_CNTIN 寄存器同步章节

12.3.11.2 PWM_CH（n）V 寄存器更新缓存

表 12-5 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 PWM_CH（n）V 寄存器同步章节

12.3.11.3 PWM_MCVR 寄存器更新缓存

表 12-6 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 PWM_MCVR 寄存器同步章节

12.3.12 PWM 同步

通过 PWM 同步，将有机会以 MCVR，CNTIN，CHnV，OMCR，INVCR 和 CHOSWCR 寄存器各自的缓存值对这些寄存器进行更新，并强制 PWM 计数器为 CNTIN 寄存器值。

注意：

- **PWM 同步只能用于组合模式；**
- **传统 PWM 同步（SYNCMODE=0）是增强型 PWM 同步（SYNCMODE=1） 的一个子集。因此，只能使用增强型 PWM 同步。**

12.3.12.1 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 由系统时钟同步。

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。

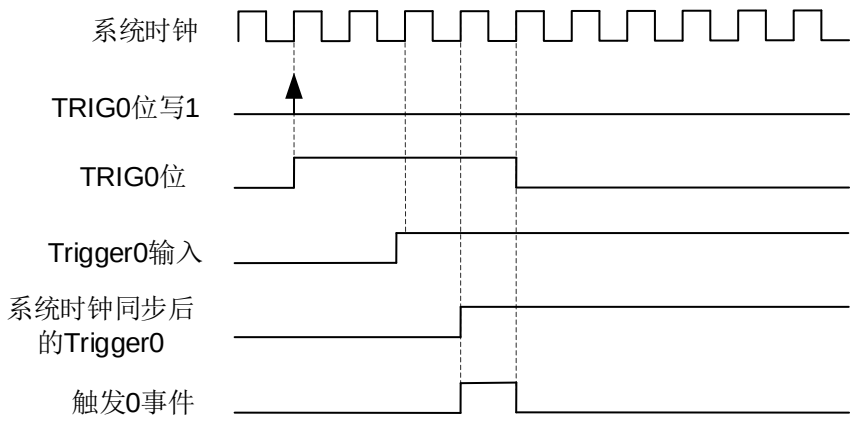

图 12-12 HWTRIGMODESEL = 0 硬件触发事件

12.3.12.2 软件触发器

当 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。

如果 SYNCMODE = 0，则根据 PWMSYNC 和 REINIT 位，PWM 也会清零 SWSYNC 位。 在这种情况下，如果（PWMSYNC = 1） 或（PWMSYNC = 0 同时 REINIT = 0），然后，在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（PWMSYNC = 0） 且（REINIT = 1），则当发生软件触发事件时，SWSYNC 位被清零。

如果 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。

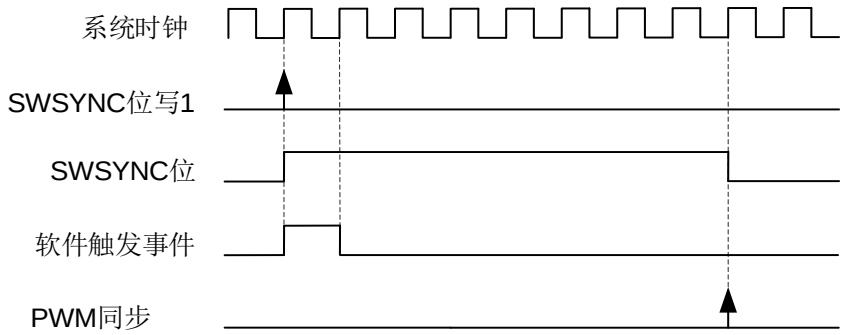

图 12-13 软件触发事件

12.3.12.3 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：

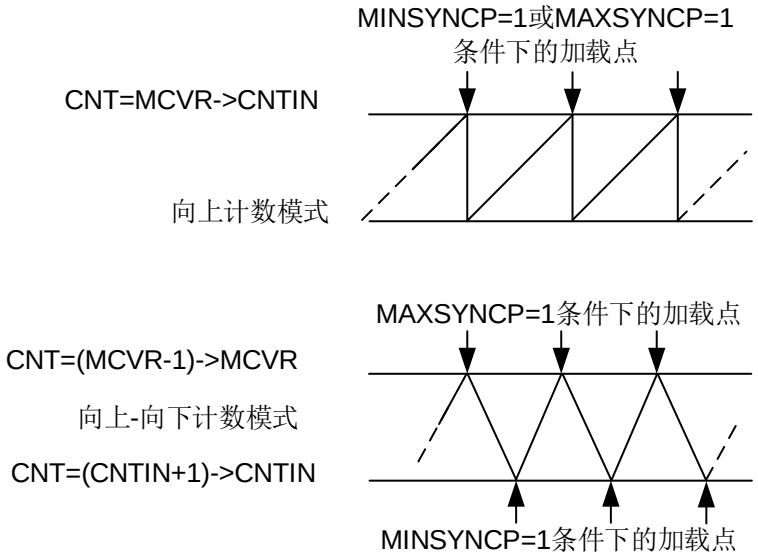

图 12-14 边界周期与加载点

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。

12.3.12.4 MCVR 寄存器同步

PWMEN2 = 1 使能 MCVR 寄存器同步功能，同步时将其缓存值更新至 MCVR 寄存器。
MCVR 寄存器同步可通过增强型 PWM 同步（SYNCMODE=1）或传统 PWM 同步（SYNCMODE=0）完成，推荐使用增强型 PWM 同步功能，流程图如下：

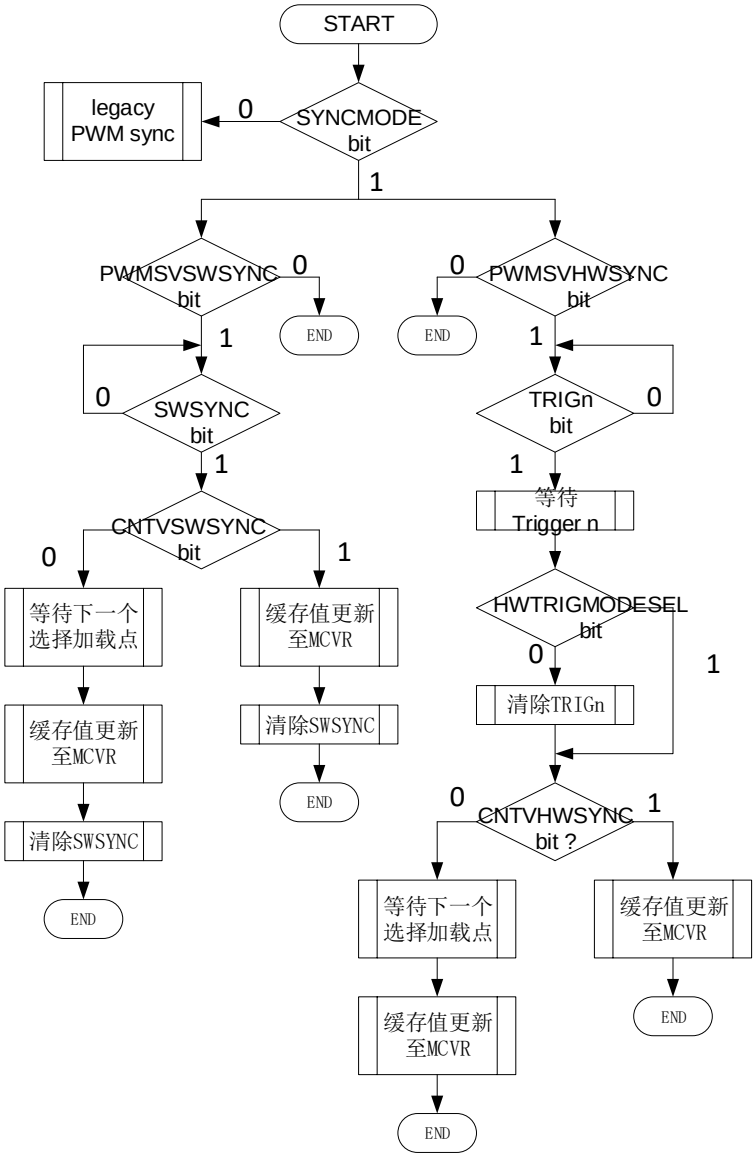

图 12-15 MCVR 寄存器同步流程

12.3.12.5 CNTIN 寄存器同步

PWMEN2 = 1、 SYNCMODE = 1 且 CNTINC = 1，将使能这种同步。只能通过增强型 PWM 同步完成 CNTIN 寄存器同步。同步机制与通过增强型 PWM 同步完成的 MCVR 寄存器同步相同，参照 MCVR 寄存器同步流程图。

12.3.12.6 CH（n）V 和 CH（n+1）V 寄存器同步

PWMEN2 = 1 且 PAIR（n）SYNCEN = 1，则会使能该同步。同步机制与 MCVR 寄存器同步流程相同，推荐使用增强型 PWM 同步功能。

12.3.12.7 OMCR 寄存器同步

OMCR 寄存器同步将其缓存值更新至 OMCR 寄存器。采用增强型 PWM 同步时，OMCR 寄存器同步取决于 OMVSWSYNC 和 OMVHWSYNC 位。参照以下流程图：

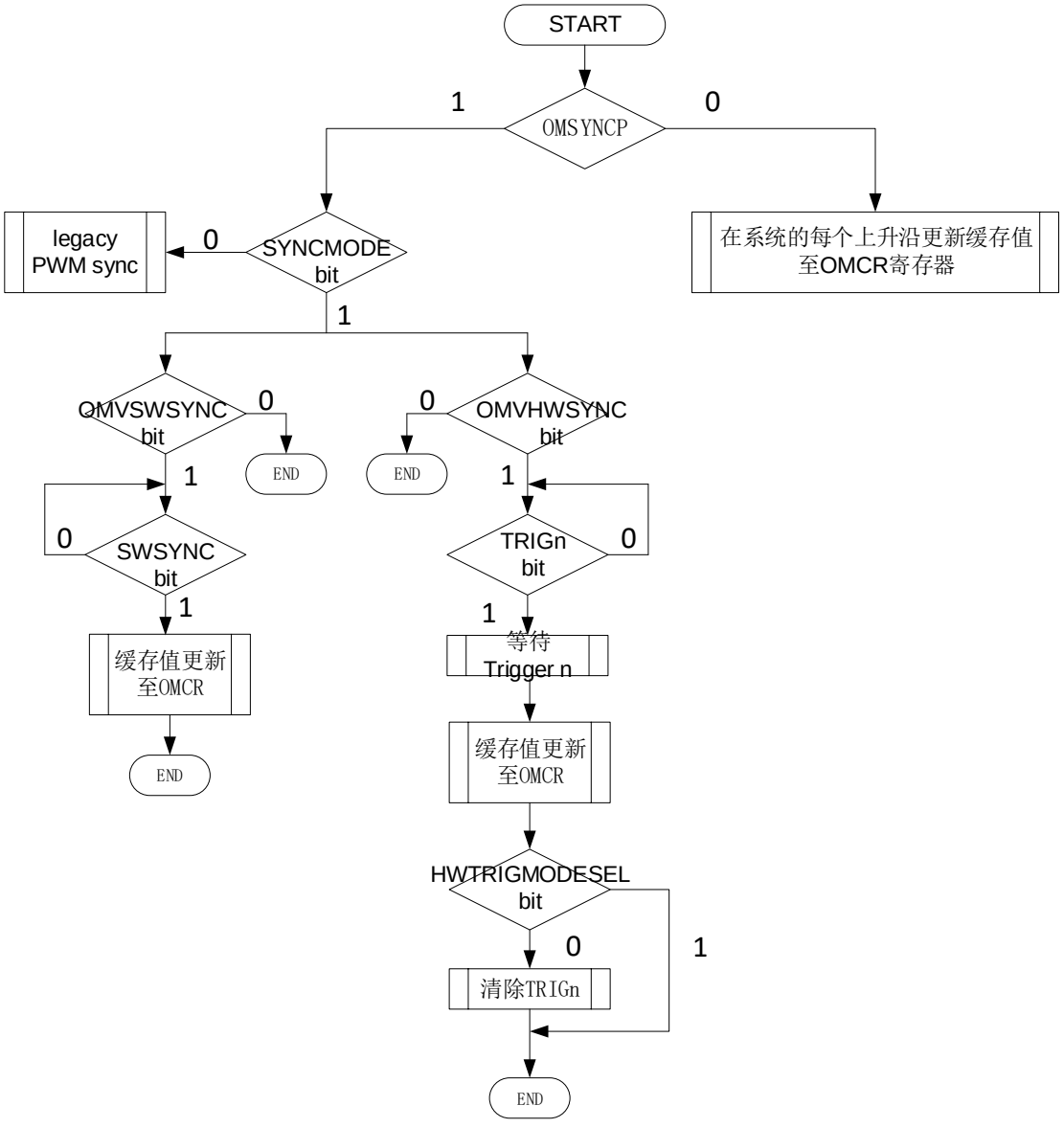

图 12-16 OMCR 寄存器同步流程

12.3.12.8 INVCR 寄存器同步

INVCR 寄存器同步将其缓存值更新至 INVCR 寄存器。采用增强型 PWM 同步时，INVCR 寄存器同步取决于 INVSWSYNC 和 INVHWSYNC 位。参照以下流程图：

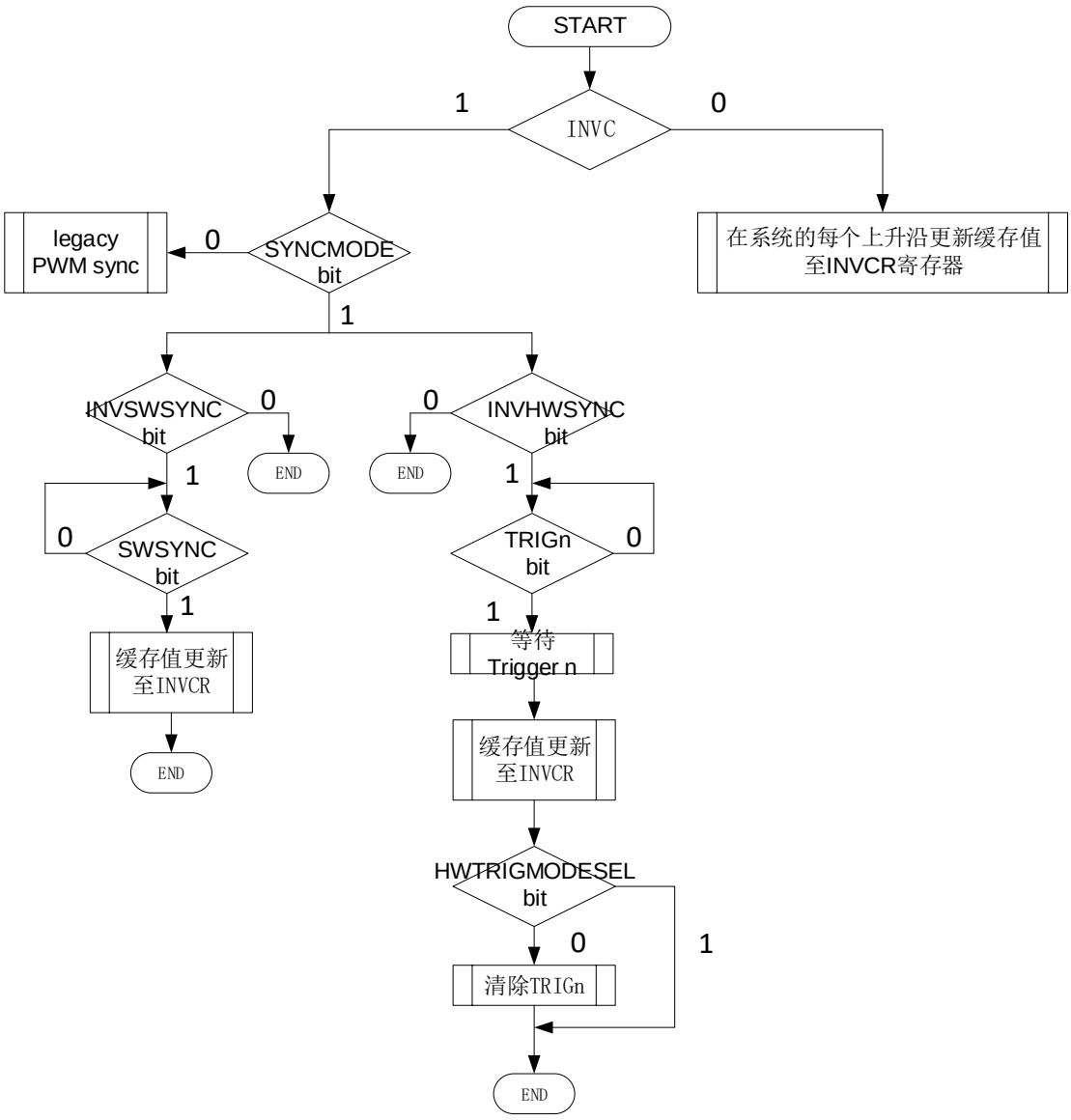

图 12-17 INVCR 寄存器同步流程

12.3.12.9 CHOSWCR 寄存器同步

CHOSWCR 寄存器同步将其缓存值更新至 CHOSWCR 寄存器。采用增强型 PWM 同步时，CHOSWCR 寄存器同步取决于 SWVSWSYNC 和 SWVHWSYNC 位。参照以下流程图：

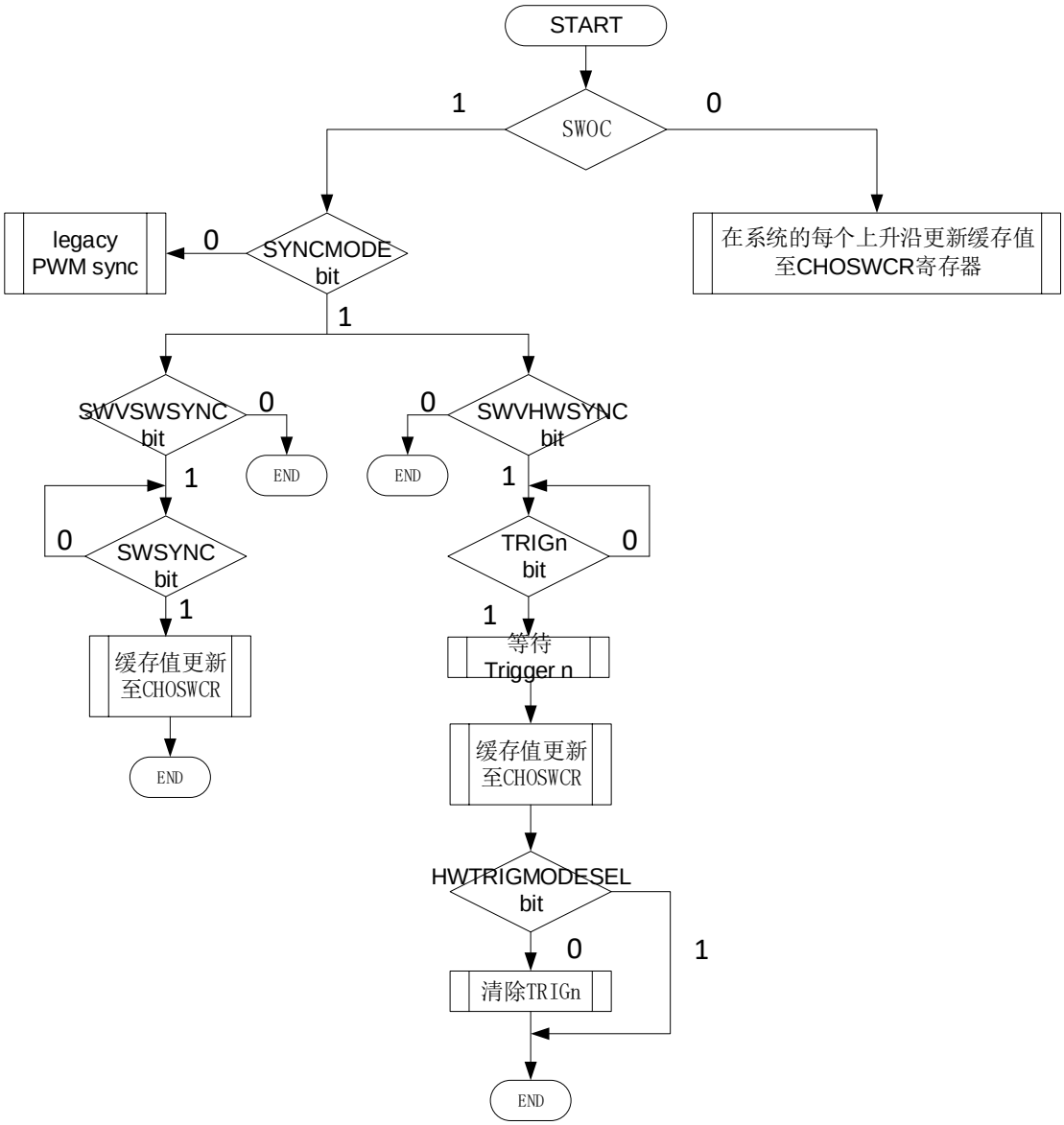

图 12-18 CHOSWCR 寄存器同步流程

12.3.12.10 CNT 寄存器同步

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，CNT 寄存器强制为 CNTIN 寄存器定义的初始计数值。PWM 计数器同步可通过增强型 PWM 同步（SYNCMODE = 1）或传统 PWM 同步（SYNCMODE = 0）完成，推荐使用增强型 PWM 同步功能。CNT 寄存器同步取决于 CNTVSWSYNC 和 CNTVHWSYNC 位，流程图如下：

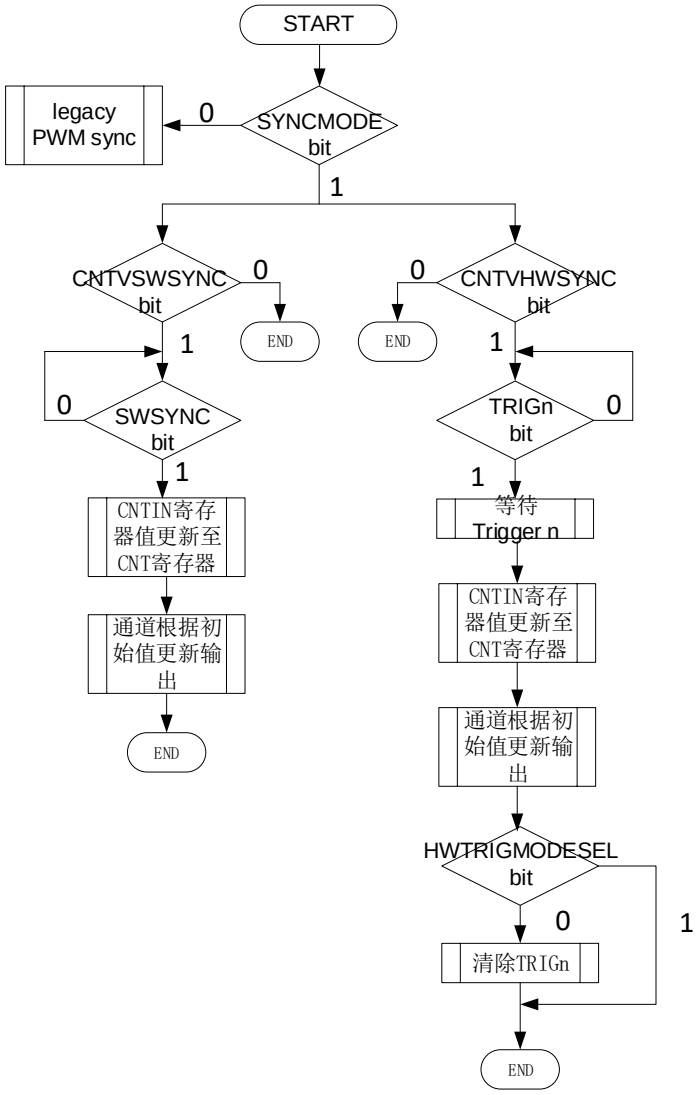

图 12-19 CNT 寄存器同步流程

12.3.13 反相

反转功能在 channel（n）和 channel（n+1）输出之间交换信号。在如下情况下，选择反相操作：

- PWMEN2 = 1；
- QEIEN = 0；
- DECAPEN = 0；
- COMBINE = 1；
- CNTMODE = 0, 且
- PAIR（n）INVEN = 1, n=0,1,2,3。

12.3.14 通道触发输出

如果 CHjTRIG = 1，其中 j = 0, 1, 2, 3, 4, or 5，则当发生 channel（j） 匹配（PWM 计数器 = CH（j）V）.时，PWM 产生触发。通道触发输出提供用于片上模块的触发信号。

功能描述：PWM 能够在一个 PWM 周期内产生多个触发。由于每个触发器是为为特定通道而生成，因此需要多个通道来实现此功能。

12.3.15 双边沿捕获模式

如果 PWMEN2 = 1 且 DECAPEN = 1，则选择双边沿捕捉模式。该模式允许测量通道对 channel（n）输入上的信号的脉冲宽度或周期。在此模式下，仅使用 channel（n）输入，忽略 channel（n+1）输入。当 n 为 0 或 2 时，channel（n）滤波器在此模式下有效。The ELS（n）R1：ELS（n）R0 位选择 channel（n）捕获的边沿，ELS（n+1）R1：ELS（n+1）R0 位选择由 channel（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，则其为周期测量。如果这些位选择不同的边沿，那么它是脉冲宽度测量。

如果在 channel（n）输入处检测到 channel（n）位选择的边沿，则设置 CH（n）IF 位并生成 channel（n）中断（如果 CH（n）IE = 1）。如果在 channel（n） 输入且（CH（n）IF = 1）时检测到 channel（n+1）位选择的边沿，则设置 CH（n+1）位且生成 channel（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 channel（n）输入处检测到 channel（n）选择的边沿时，CH（n）V 寄存器存储 PWM 计数器的值。在 channel（n）输入处检测到 channel（n+1）选择的边沿时，CH（n+1）V 寄存器存储 PWM 计数器的值。在此模式下，当读取 CH（n）V and CH（n+1）V 寄存器时，一致性机制确保相干数据。唯一的要求是必须在 CH（n+1）V 之前读取 CH（n）V。

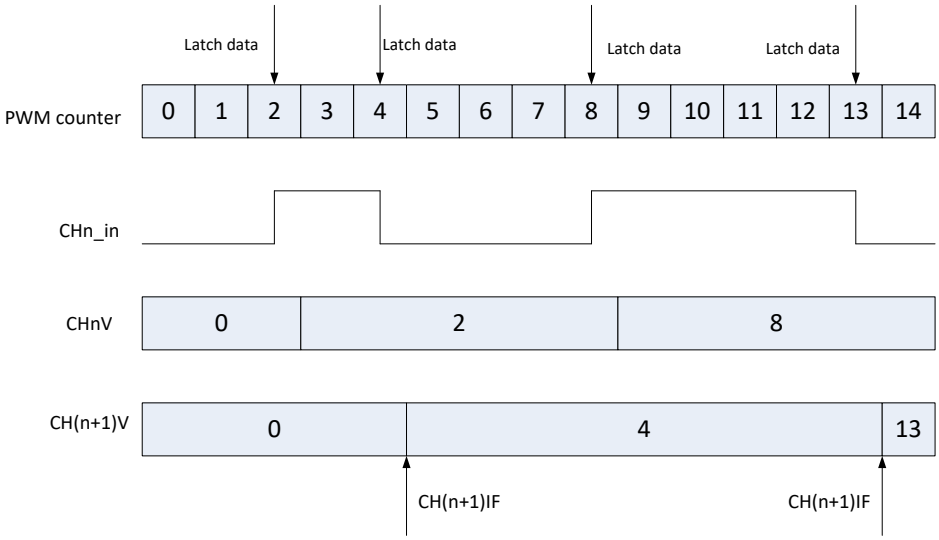

图 12-20 双边沿捕获模式图

12.3.16 正交解码器模式

如果（PWMEN2 = 1）和（QEIEN = 1），则选择正交解码器模式。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 的输入滤波器由 PHAFLTREN 位启用，该滤波器的值由 CH0CAPFVAL[4:0]位定义（CH（n）CAPFVAL[4:0]位在 CAPFILTER 寄存器中）。相位 B 的输入滤波器由 PHBFLTREN 位启用，该滤波器的值由 CH1CAPFVAL [4:0]位定义。

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。

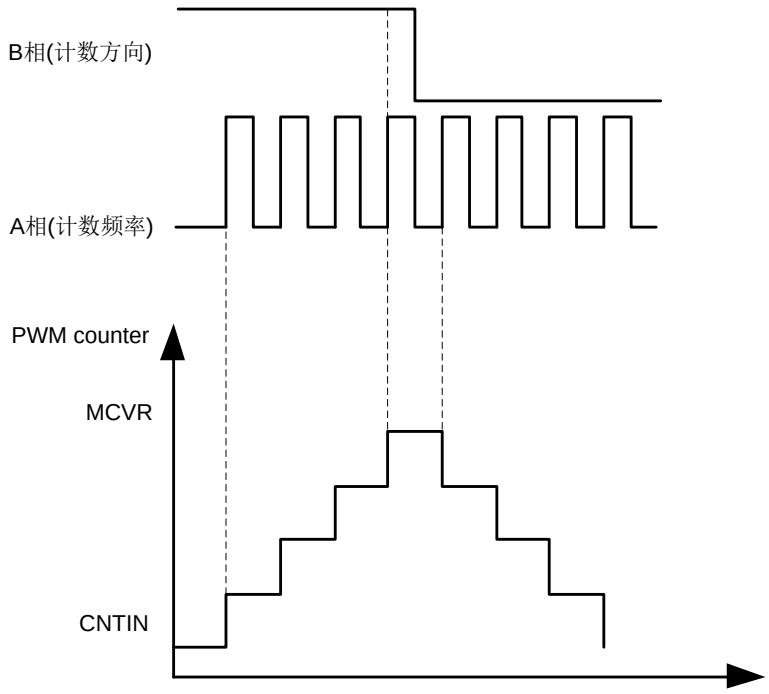

图 12-21 计数和方向编码模式

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；

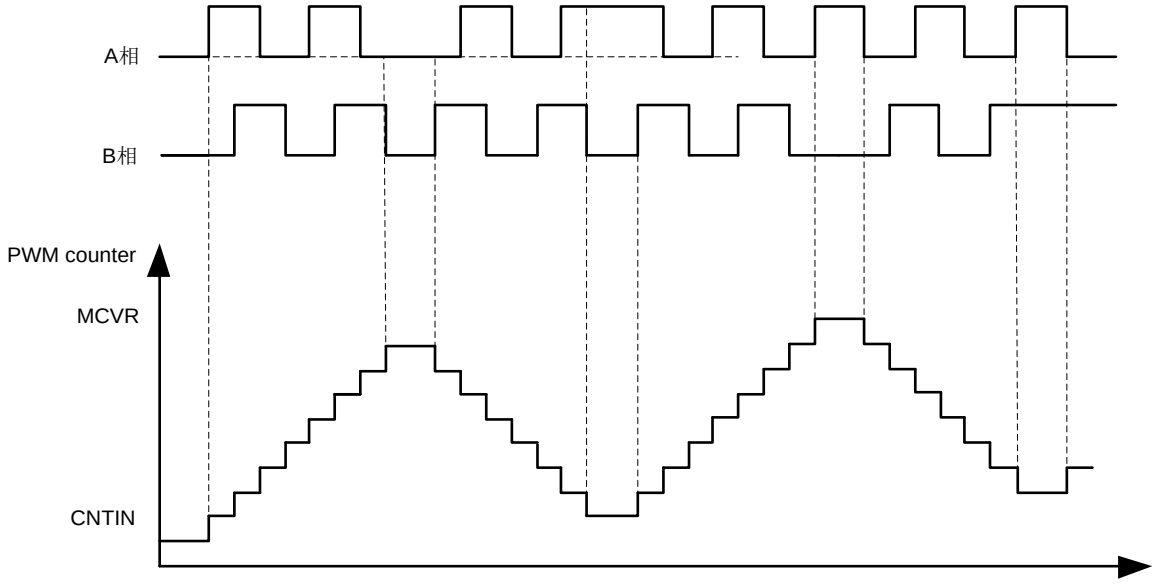

图 12-22 A 相和 B 相编码模式

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 TOF 和 TOFDIR 位。TOF 位表示发生了 PWM 计数器溢出，TOFDIR 指示 PWM 计数器在向上计数时发生溢出。

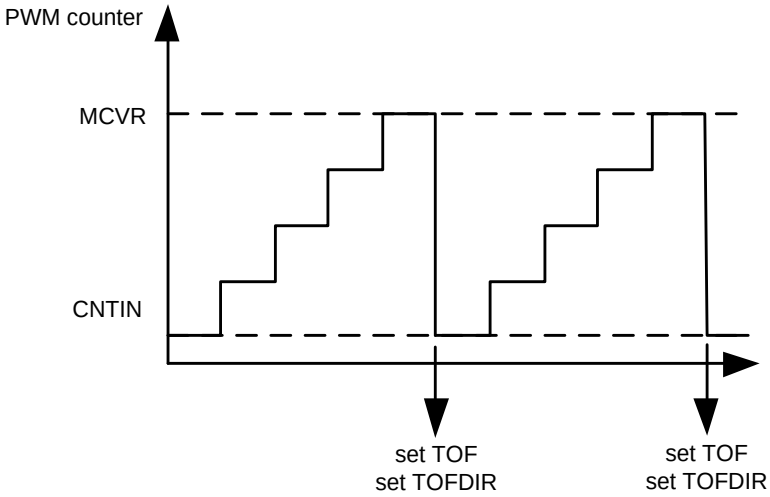

图 12-23 向上计数 PWM counter 溢出

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 TOF 位和清除 TOFDIR 位。TOF 位表示发生了 PWM 计数器溢出，TOFDIR 指示 PWM 计数器在向下计数时发生溢出。

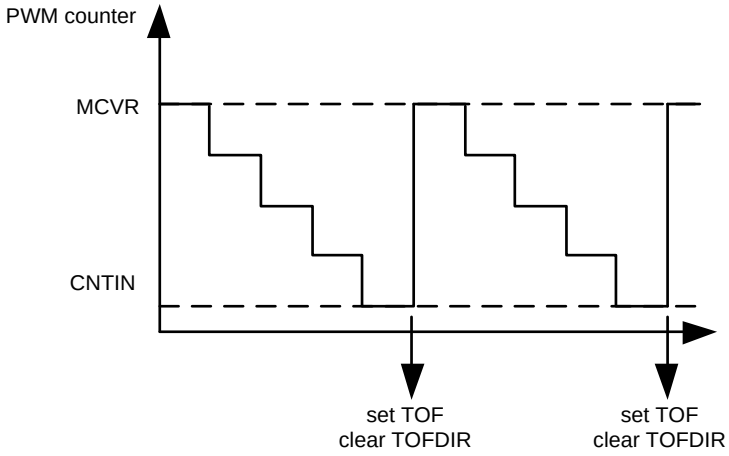

图 12-24 向下计数 PWM counter 溢出

12.3.17 初始化触发器

如果 INITTRIGEN = 1，则在以下情况下使用 CNTIN 寄存器值更新 PWM 计数器时，PWM 会产生触发。

- PWM 计数器通过所选计数模式使用 CNTIN 寄存器值进行自动更新。
- 当对 CNT 寄存器执行写操作时。

- 当存在 PWM 计数器同步时。

12.3.18 软件输出控制

软件输出控制根据 PWM 生成中特定时间的软件定义值强制通道输出。

在如下情形下，选择软件输出控制：

- PWMEN2 = 1；
- DECAPEN = 0；
- COMBINE = 1；
- CNTMODE = 0；
- CH（n）SWEN = 1, n=0,1,2,3,4, 5。

CH（n）SWEN 位使能软件输出控制，CH（n）SWCV 选择强制该通道输出的值。当 COMP 位为零时，软件输出控制在 channs（n） 和（n+1）上强制执行以下值。

表 12-7 软件输出控制（COMP 位为 0）

CH（n）SWEN	CH（n+1）SWEN	CH（n）SWCV	CH（n+1）SWCV	Channel（n） 输出	Channel（n+1） 输出
0	0	X	X	不支持软件控制	不支持软件控制
1	1	0	0	0	0
1	1	0	1	0	1
1	1	1	0	1	0
1	1	1	1	1	1

当 COMP 为 1 时，在 channs（n） 和（n+1）上，软件输出控制强制使用如下值。

表 12-8 软件输出控制（COMP 位为 1）

CH（n）SWEN	CH（n+1）SWEN	CH（n）SWCV	CH（n+1）SWCV	Channel（n） 输出	Channel（n+1）输出
0	0	X	X	不支持软件控制	不支持软件控制
1	1	0	0	0	0
1	1	0	1	0	1
1	1	1	0	1	0
1	1	1	1	1	0

12.3.19 死区插入

当 DTEN = 1 且 DTVAL[5：0]为非零时，使能死区插入。DTSET 寄存器定义了可用于所有 PWM 通道的死区延迟。DTPSC[1：0]位定义系统时钟的预分频器，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（channels（n）和（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 channel（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，channel

（n）输出置位时。类似地，当发生 channel（n+1）匹配（PWM 计数器 = CH（n+1）V）时，channel（n+1）输出保持低电平状态，直到死区延迟结束，channel（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 channel（n）匹配（PWM 计数器 = CH（n）V）时，channel（n）输出保持高电平状态，直到死区延迟结束，channel（n）输出清零时。类似地，当发生 channel（n+1）匹配（PWM 计数器 = CH（n+1）V）时，channel（n+1）输出保持高电平状态，直到死区延迟结束，channel（n+1）输出清零时。

12.3.20 故障控制

存在 4 个故障输入源，2 个内部，2 个外部 ，可以为内部故障输入选择 ACMP0_OUT 和 ACMP1_OUT，而可以为外部故障输入选择外部故障 1 和 故障 2 。

表 12-9 故障源和编号表

故障源	故障输入编号	说明
ACMP0_OUT	FAULT0	内部故障输入
External Fault1	FAULT1	外部故障输入
External Fault2	FAULT2	外部故障输入
ACMP1_OUT	FAULT3	内部故障输入

12.3.20.1 自动故障清除

如果选择自动故障清除（FAULTMODE[1：0]=1：1），则当故障输入信号返回零并且新的 PWM 周期开始时，被故障控制禁用的通道输出再次被启用。

12.3.20.2 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再次被启用。

12.3.20.3 故障输入极性控制

FLTnPOL 位选择故障输入 n 极性，其中 n=0,1,2 ,3。

（1） 如果 FLTnPOL = 0，故障 n 输入极性为高，所以故障输入 n 处的逻辑 1 代表一个故障。

（2） 如果 FLTnPOL = 1, 故障 n 输入极性为低，所以故障输入 n 处的逻辑 0 代表一个故障。

12.3.21 极性控制

CHnPOL 位选择 channel（n） 输出极性，其中 n=0,1,2,3,4,5。

（1） 如果 CHnPOL = 0，channel（n） 输出极性为高，因此逻辑 1 为有效状态，逻辑 0 为无效状态。

（2） 如果 CHnPOL = 1，channel（n） 输出极性为低，因此逻辑 0 为有效状态，逻辑 1 为无效状态。

12.3.22 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。

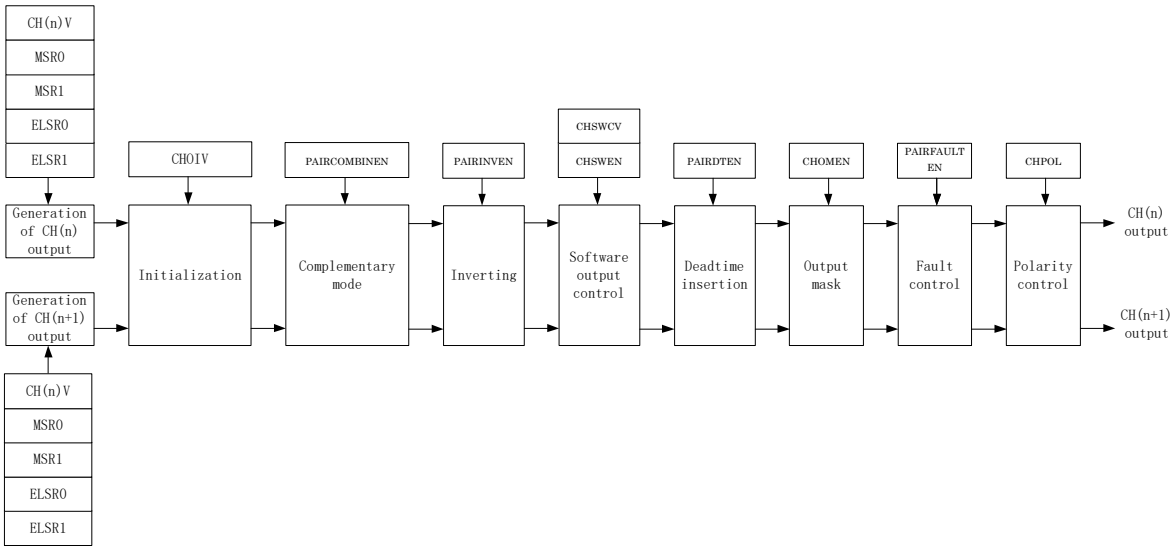

图 12-25 特性优先级

12.4 PWM 中断

12.4.1 计数溢出中断

当 CNTOIE=1 且 CNTOF=1 时，产生计数溢出中断。

12.4.2 通道中断

当 CHnIE=1 且 CHnIF = 1 时，产生 Channel（n） 中断。

12.4.3 故障中断

当 FAULTIE =1 且 FAULTDF = 1 时，产生故障中断。

13 脉冲宽度检测定时器（PWDT）

13.1 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 位定时器。

13.2 特性

- 2 个可选时钟源：总线时钟和备用时钟；
- 4 个可选脉冲时钟；
- 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 ：
        - 可编程起始测量触发沿 .
        - 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
        - 在禁用定时器或在正常操作时，修改定时器加载值
- 16 位计数器，用做脉冲宽度测量或定时器功能；
- 中断：
    - OVF：脉冲宽度测量功能或定时器功能，当 PWDT 计数器溢出时.
    - RDYF： PWDT 脉冲宽度测量值更新。

13.3 功能描述

下图为 PWDT 模块的功能框图。

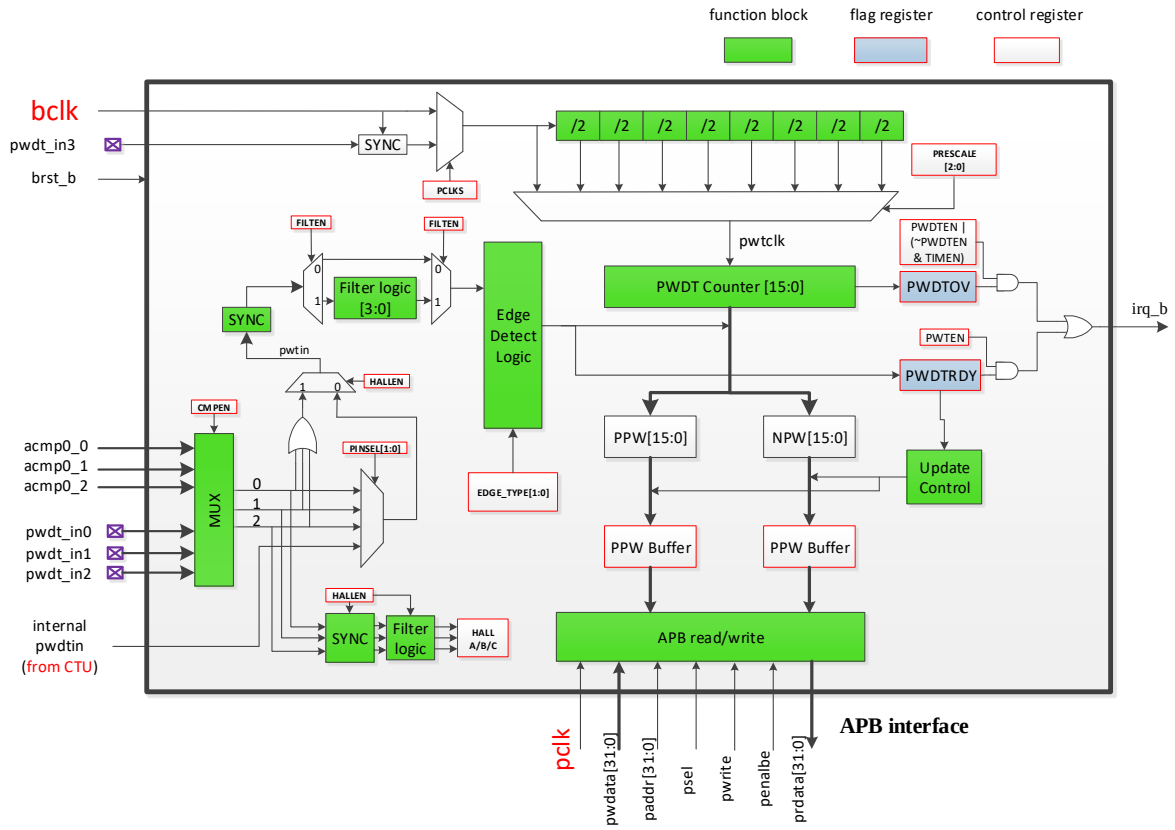

图 13-1 PWDT 功能框图

13.3.1 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 PSC [2：0]从总线时钟分频得到的。因此，为了获得更准确的测量值，用户最好将较小的 PSC[2：0]值用于较窄的脉冲输入，将较大的 PSC[2：0]值用于较宽的脉冲输入。可测量的脉冲宽度如表 13-1 所示。

表 13-1 可测量脉冲宽度范围

项	时钟	时间
可测量脉冲宽度范围	4 bcllk ~ 128*65535 bclk	0.08us ~ 0.168s

首先，对于脉冲宽度测量，用户必须清楚地了解应用场景，主要包括两个条件：一个是用于基本测量的单通道输入，另一个是用于霍尔测量的 3 通道输入。对于基本测量，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。

对于霍尔测量，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 3 个通道比较器输入的配置类似于霍尔测量，只 CMPEN 同时配置为 1'b1。

对于基本测量，可根据实际应用选择 4 种测量模式，如图 13-2 所示。

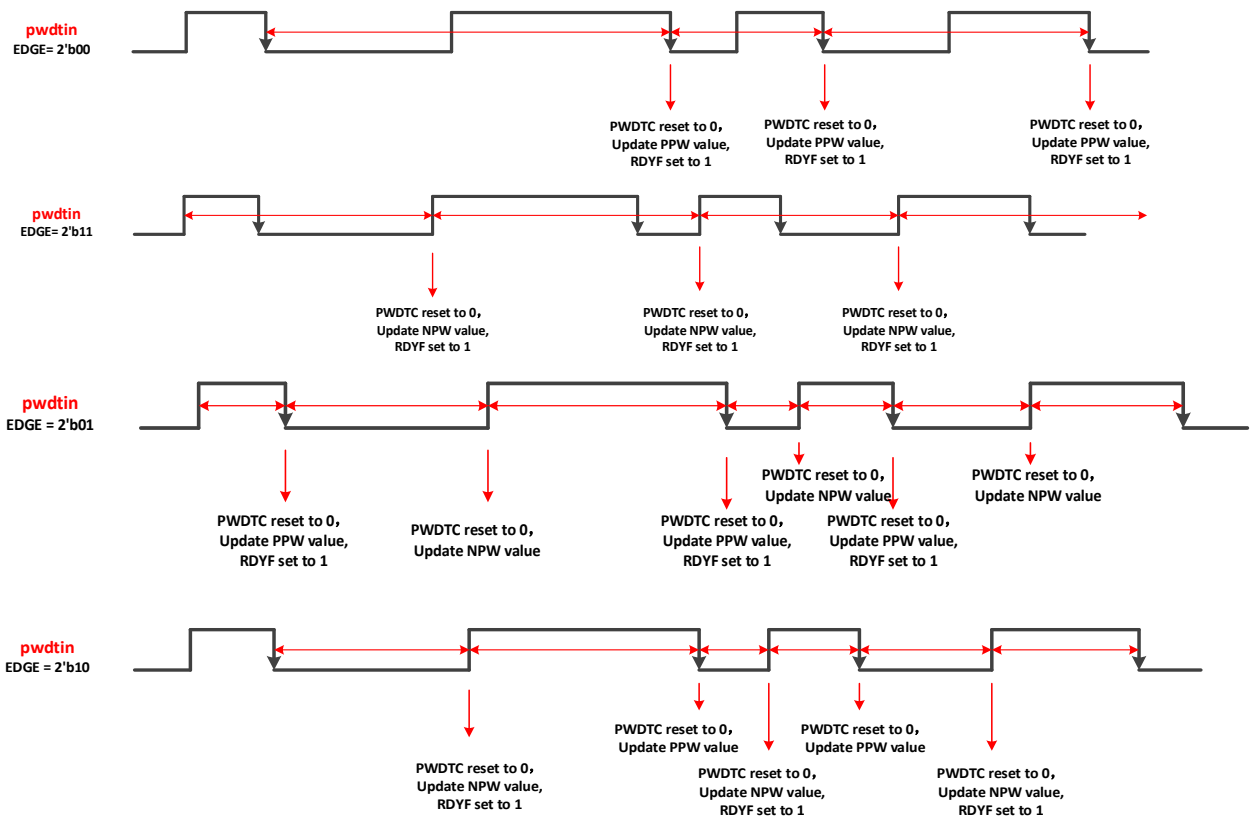

图 13-2 四种基本的测量模式

对于霍尔测量或 3 个内部比较器输入，只需选择一种模式用于电机速度计算或换向，如图 13-3 所示。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向。通常有两个安装，如图 13-4 所示。一个是 120 电度间隔，另一个是 60 电度间隔。

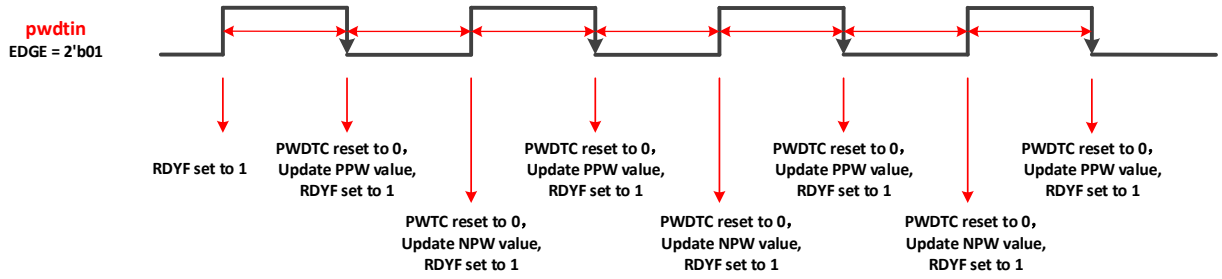

图 13-3 霍尔测量模型

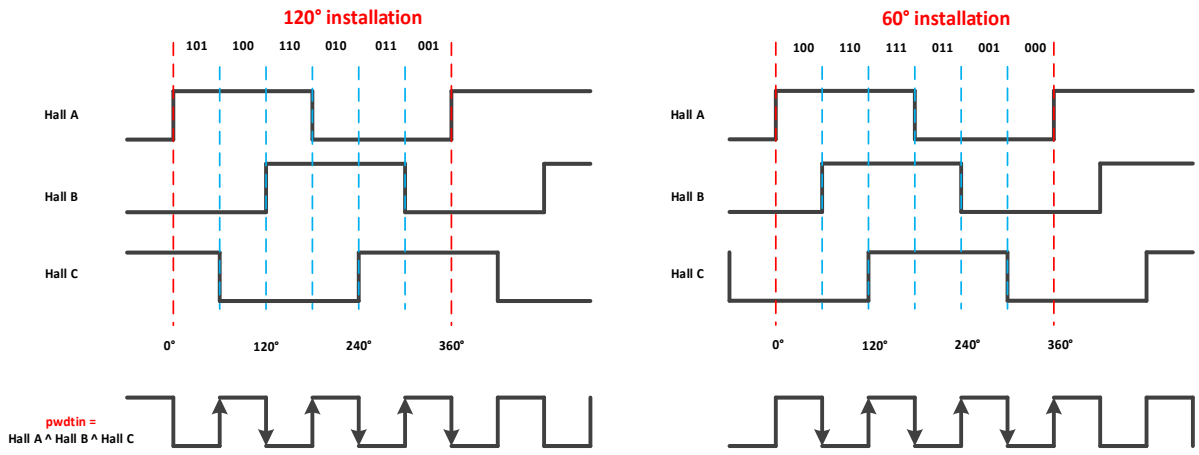

图 13-4 两种常见的安装方式

其次，模块中的滤波器设计用于滤除高或低电平小于本段中描述的特定宽度的噪声信号。FILT_PSC[7：4]和 FILTVAL [3：0]设置确定最大和最小噪声脉冲宽度。图 13-5 和图 13-6 介绍了噪声宽度设置，判断和滤波器。当用户配置 FILTVAL = 15 及 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 13-2 所示。

表 13-2 可滤波脉冲宽度范围

项	时钟	时间
可滤波脉冲宽度范围	4 bcllk ~ 16*4096 bclk	0.08us ~ 1.311ms

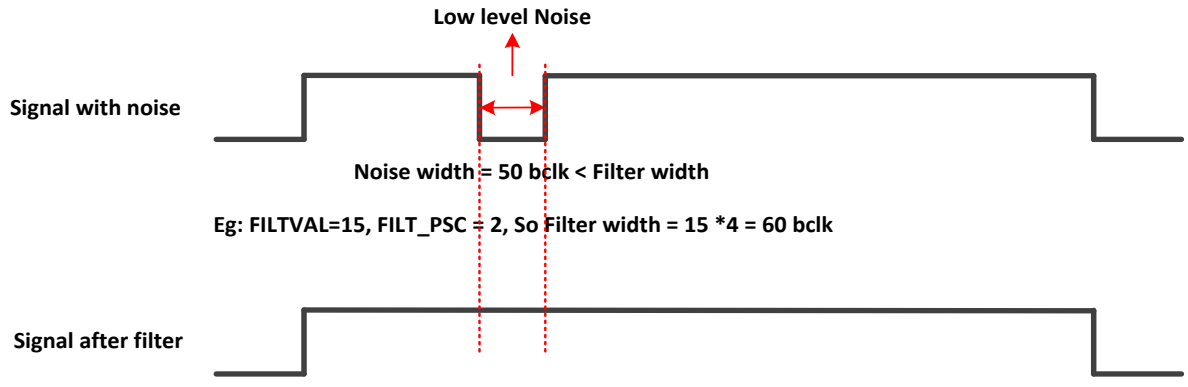

图 13-5 低电平噪音和滤波器示例

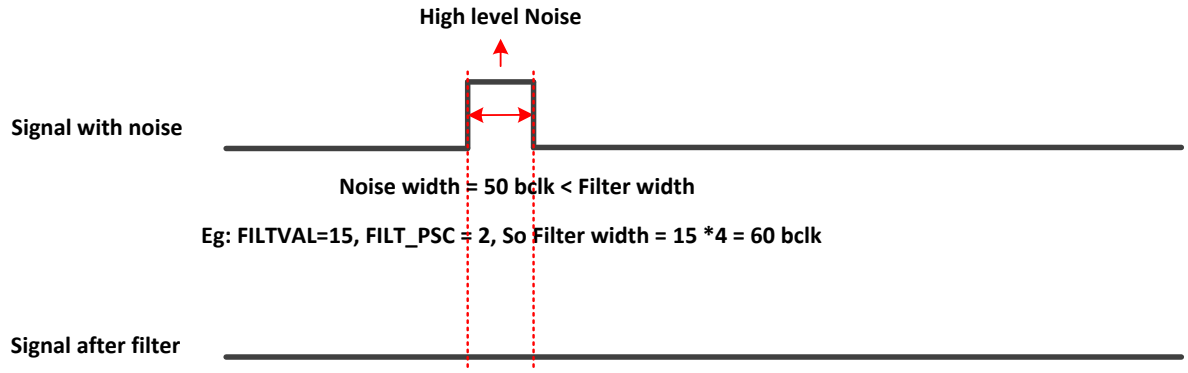

图 13-6 高电平噪音和滤波器示例

再次，当相应的使能为 1'b1 时，RDYF 和 OVF 状态或中断用于脉冲宽度测量功能。RDYF 状态在上述条件下发生。当 PWDTC 计数器溢出时，OVF 状态发生。

最后， 用户应了解脉冲宽度测量的测量精度，配置 PSC[2：0]以适当的值，以达到更准确的测量值。一个基本原则是，使用较小的 PSC[2：0]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 13-7 描述了脉冲宽度测量功能运行时的误差。在图 13-7 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 13-7 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。

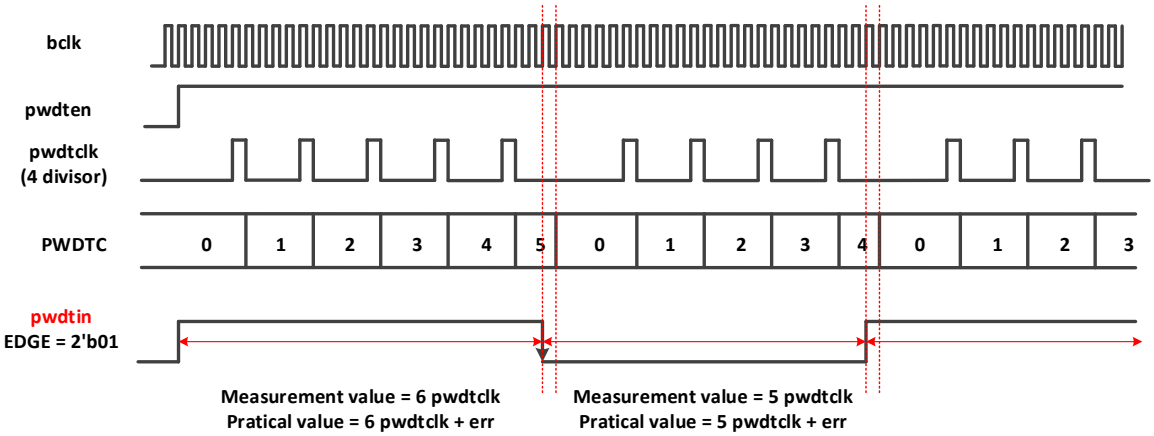

图 13-7 PWDTC 计数器和计数错误

13.3.2 定时器功能

对于定时器功能，只有 OVF 状态有效，并在 PWDTC 计数器溢出时发生。计数器负载值 TIMCNTVAL[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 13-8 和图 13-9 所示。

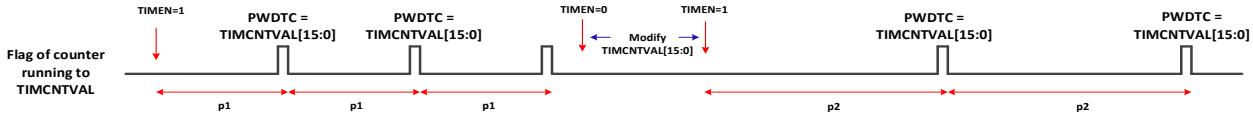

图 13-8 在 TIMEN=0 和 TIMEN=1 之间修改 TIMCNTVAL

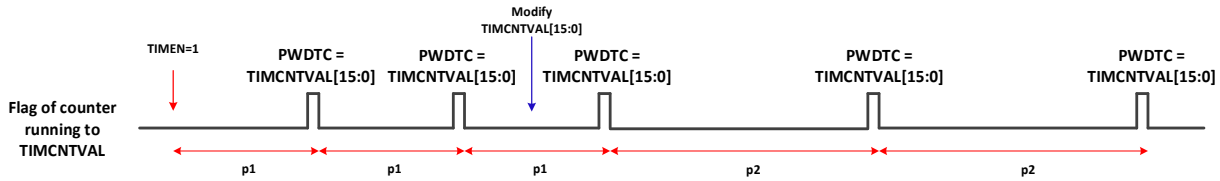

图 13-9 TIMEN=1 期间修改 TIMCNTVAL

13.3.3 复位操作

本节介绍模块软复位操作。用户向 SR 位写入 1 即可触发该模块软复位。一旦触发软复位，则会发生如下操作：

- PWDTC 复位为 0；
- PWDTC 预分频器复位为 0；
- 边沿检测逻辑复位；
- 脉宽寄存器锁存机制复位；
- PPWCV 和 NPWCV 复位为 0；
- RDYF 和 OVF 复位为 0；
- 其他控制位不复位。

同时，将 PWDTEN 配置为 0 也可以达到以上相同的复位效果。

13.4 编程指南

13.4.1 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。特别是，对于内部 3 个比较器输入，HALLEN 和 CMPEN 应配置为 1。

13.4.2 定时器功能编程指南

只需配置 TIMCNTVAL，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。

13.5 寄存器定义

表 13-3 PWDT 寄存器映射及其复位值

基地址： 0x40017000

地址	名称	宽度	寄存器功能
基地址+0x00000000	PWDT_INIT0	32	通用控制和状态位，及正脉宽内容

基地址+0x00000004	PWDT_NPWCV	32	负脉宽内容及 16 位自由运行计数器
基地址+0x00000008	PWDT_INIT1	32	霍尔功能控制和定时器功能控制

00000000 PWDT_INIT0 PWDT 初始化寄存器 0 00000000

位	31~16	15	14	13~12	11	10	9	8	7	6	5	4	3	2	1	0
名称	PPWCV	PCLKSEL		PINSEL	EDGE		PSC			PWDTEN	IE	PRDYIE	OVIE	SR	RDYF	OVF
类型	R	RW		RW	RW		RW			RW	RW	RW	RW	W	R	R
复位	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0

注意：PWDTEN（pwm 功能）使能先于 TIMEN（定时器功能），因此当要使能定时器功能时，必须禁用 PWDTEN。

位	助记符	名称	说明
31：16	**PPWCV**	PPWCV	**正脉宽** 表示正脉宽值
15	**PCLKSEL**	PCLKSEL	**PWDT 时钟源选择** 0： 计数器时钟作为 PWDT 计数器的时钟源. 1： 其他时钟作为 PWDT 计数器的时钟源
13~12	**PINSEL**	PINSEL	**引脚选择** **注意：**internal_pwdtin 来自 CTU 模块内部。 00/01/10/11：分别选择 pwdt_in0/ pwdt_in1/ pwdt_in2/ internal_pwdtin.
11~10	**EDGE**	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	**PSC**	PSC	**PWDT 计数器预分频** 000 ~ 111：分别代表 1/2/4/8/…/128.
6	**PWDTEN**	PWDTEN	**PWDT 模块使能** 0： 禁用 1： 使能
5	**IE**	IE	**PWDT 模块中断使能** 0： 禁用 1： 使能
4	**PRDYIE**	PRDYIE	**PWDT 脉宽数据就绪中断使能** 0： 禁用 1： 使能
3	**OVIE**	OVIE	**PWDT 计数器溢出中断使能** 0： 禁用 1： 使能
2	**SR**	SR	**PWDT 软复位**

位	助记符	名称	说明
			注意： 该字段始终读为 0 0： 未采取任何动作 1： 使能
1	**RDYF**	RDYF	**PWDT 脉宽有效** **注意： 写 0 清除该位** 0： pwdt 脉宽寄存器未更新 1： PWDT 脉宽寄存器已更新
0	**OVF**	OVF	**PWDT 计数器溢出** **注意： 写入 0 清除该位** 0： 无溢出 1： 从 0x 0000 运行到 0x FFFF.

00000004 PWDT_NPWCV PWDT NPWCV 计数值 00000000

位	31~16	15~0
名称	**PWDTC**	**NPWCV**
类型	R	R
复位	0	0

位	助记符	名称	说明
31：16	**PWDTC**	PWDTC	**脉宽计数器** 用于脉宽测量或定时器计数
15：0	**NPWCV**	NPWCV	**负 脉宽计数值** 表示 负脉宽值

00000008 PWDT_INIT1 PWDT 初始化寄存器 1 00000000

位	31	30	29	28	27~12	11	10	9	8	7~4	3~0
名称		**HALLA**	**HALLB**	**HALLC**	**TIMCNTVAL**	**CMPEN**	**TIMEN**	**HALLEN**	**FILTEN**	**FILT_PSC**	**FILTVAL**
类型		R	R	R	RW	RW	RW	RW	RW	RW	R
复位		0	0	0	0	0	0	0	0	0	0

注意：要使滤波器功能起作用， PWDT_INIT1[FILTVAL]设置必须大于 1，否则该功能将不起作用。

位	助记符	名称	说明
30：28	**HALLA/B/C**	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	**TIMCNTVAL**	timcntval	**定时器计数器负载值** 定时器从 0x0000 运行值 TIMCNTVAL.
11	**CMPEN**	cmpen	**切换 pwdt_in0 ~ pwdt_in2 的源** **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**

位	助记符	名称	说明
			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。 **1**：使能来自 acmp0_0 ~ acmp0_2 内部的 pwdt_in0 ~ pwdt_in2 **0**：使能来自 pad PWDT_IN0 ~ PWDT_IN2 外部的 pwdt_in0 ~ pwdt_in2
10	**TIMEN**	timen	**当 PWDTEN = 0 时，使能定时器功能** 0：禁用 1：使能定时器功能
9	**HALLEN**	hallen	**当 PWDTEN = 1 时，使能霍尔传感器信号检测功能** 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**	filten	**当 PWDTEN =1 时，使能 pwdt 输入滤波器功能** 0：禁用. 1： 使能滤波器功能
7 ~ 4	**FILT_PSC**	filt_psc	**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**	filtval	**滤波器值** 0 ~ 15 ： 滤波不同脉宽。

14 周期性中断定时器（TIMER）

14.1 简介

TIMER 模块是可用于定时发起中断和触发的定时器。

14.2 特性

- 定时器能够生成触发脉冲；
- 定时器能够生成中断；
- 可屏蔽中断；
- 每个定时器都具有独立的超时周期；
- 支持最多 8 个定时器；
  - 2 个为 32 位计数器
  - 6 个为 16 位计数器
- 支持链（Chain）模式。

14.3 功能说明

下图为 TIMER 模块的结构框图。

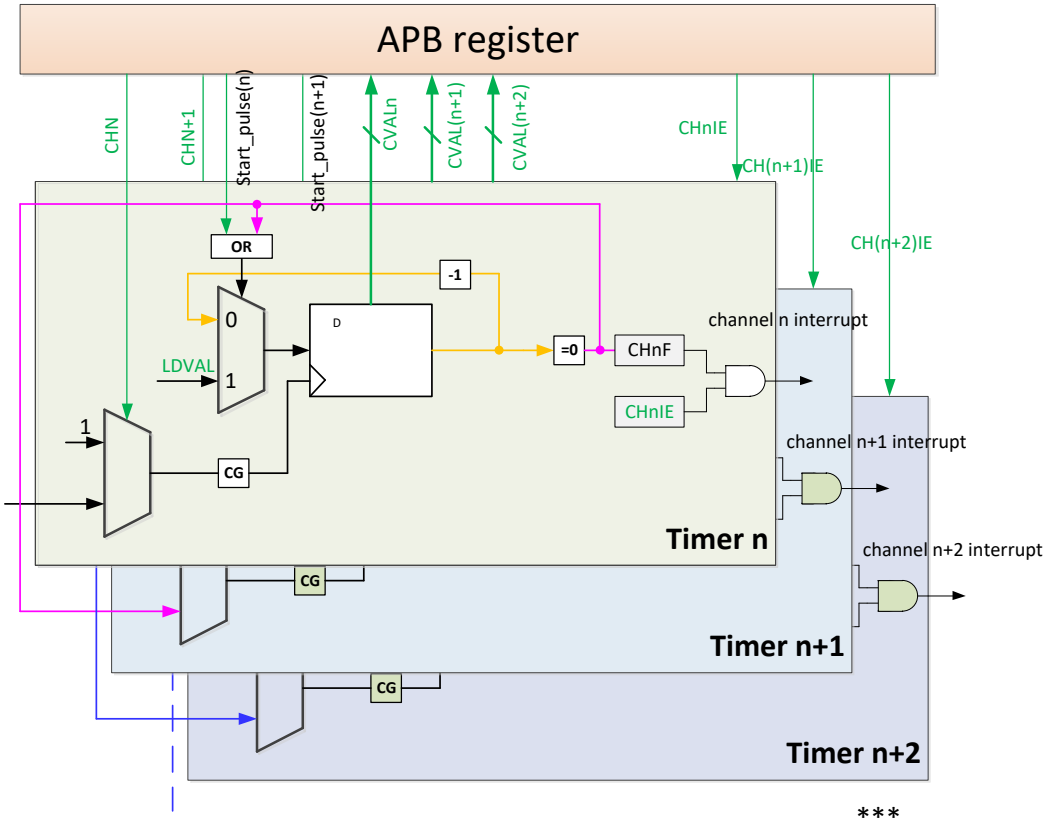

图 14-1 TIMER 结构框图

14.3.1 一般操作

定时器使能时，会定期生成触发器。定时器加 LDVAL 寄存器中指定的起始值，向下计数到 0，然后再次加载相应的起始值。每次定时器达到 0 时，它将生成一个触发脉冲并设置中断标志。

通过设置 INIT[TIE]可以启用或屏蔽所有中断。只有在前一个中断被清除后才能生成新的中断。如果需要，可以通过 CVAL 寄存器读取定时器的当前计数器值。通过首先禁用，然后使用 INIT[TIMEREN]启用计时器，可以重新启动计数器周期。

14.3.2 链接定时器

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。第一个定时器（timer 0） 不能链接至任何其他定时器。.

由于定时器模块具有 8 个定时器，因此链可以多于一个。例如，定时器 0/1/2 在一个链，定时器 3/4/5/6 在另一个链中，其余的定时器在第三个链中。

14.4 寄存器定义

表 14-1 定时器寄存器映像

TIMER： 0x40011000

地址	名称	说明
TIMER+ 0x000	**MCR**	定时器模块控制器（MCR）寄存器
TIMER + 0x100	**INITVAL**	定时器初始值（INITVAL）寄存器 偏移地址 = 0x100+16*x（x=0 至 7）
TIMER + 0x104	**CVAL**	定时器当前定时器值（CVAL）寄存器 偏移地址 = 0x104+16*x（x=0d 至 7d）
TIMER + 0x108	**INIT**	定时器初始（INIT）寄存器 偏移地址 = 0x108+16*x（x=0 至 7）
TIMER + 0x10C	**TF**	定时器标志（TF）寄存器 偏移地址 = 0x10C+16*x（x=0 至 7）

MCR **定时器模块控制寄存器** **偏移地址 = 0x00**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读 写																
缺省	0															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读 写															**MDIS**	
缺省	0														1	

位	名称	说明
1	MDIS	**模块禁用 -（TIMER 部分）** 禁用标准定时器模块。必须在执行任何其他设置前使能该字段。 0： 使能 TIMER 定时器模块 1： 禁用 TIMER 定时器模块

INITVAL **定时器初始值寄存器** **偏移地址 = 0x100+16*x（x=0 至 7）**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读 写	**LDVAL[31：16]**															
缺省																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读 写	LDVAL[15：0]															
缺省值																

位	名称	说明
31：0	LDVAL	**LDVAL：** 加载值寄存器 设置定时器起始值。定时器将倒计时至 0，然后生成一个中断并再次加载该寄存器的值。将新值写入该寄存器不会重启定时器。相反定时器过期后，会加载新值。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。 **注意： 当 x >1 时，LDVAL 位长度为 16。** **当 x = 0,1 时，LDVAL 的长度为 32。**

CVAL **定时器当前值寄存器** **偏移地址 = 0x104+16*x（x=0d 至 7d）**

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

位	名称	说明
31：0	CVALx	**CVALx ：** 当前定时器值寄存器 代表当前定时器值，如果定时器已使能。 **注意：当 x >1 时，CVALx 位长度为 16。** **当 x = 0,1 时，CVALx 的长度为 32。**

INIT **定时器初始寄存器** **偏移地址 = 0x108+16*x（x=0 至 7）**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读 写																
缺省	0															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读														LINKEN	TIE	TIMEREN

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
写																
缺省	0													0	0	0

位	名称	说明
2	LINKEN	**LINKEN： 链/连接模式** 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 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 链接至定时器 1。
1	TIE	**TIE： 定时器中断使能** 当某个中断挂起或 TFLGn[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 TFLGn[TIF] 。 0： 禁用定时器中断请求 1：一旦置位 TIF，请求中断。
0	TIMEREN	**TIMEREN： 定时器使能** 使能或禁用定时器 0： 禁用定时器 n 1： 使能定时器 n

TF　　定时器标志寄存器　　偏移地址 = 0x10C+16*x（x=0 至 7）

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读 写																
缺省	0															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读 写																**TIF**
缺省	0															0

位	名称	说明
0	TIF	**TIF ： 定时器中断标志** 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。 若使能或 TCTRLx[TIE] = 1，TIF 将引发中断请求。 0： 尚未发生超时 1： 超时已经发生

14.5 中断

所有定时器都支持中断生成。

定时器中断可通过置位 INIT[TIE]来使能。当相关定时器发生超时时，TF[TIF] 标志被置位为 1。将 1 写入对应的 TF[TIF]可将其清零。

15 采集传输终端（CTU）

15.1 简介

CTU 模块可用于模块间的互连。

15.2 特性

- ACMP 输出捕获；
- UART1_TX 调制；
- UART1_RX 捕获；
- UART1_RX 滤波；
- RTC 捕获；
- ADC 触发；
- PWM2 软件同步。

15.3 功能描述

下图为 CTU 结构框图。

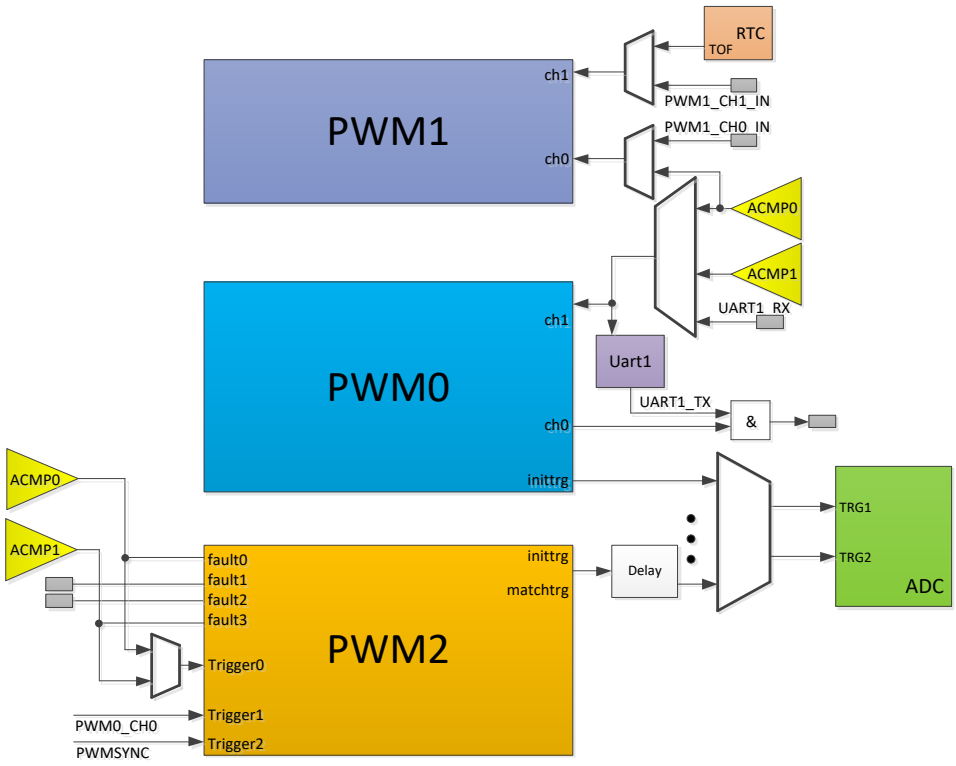

图 15-1 CTU 结构框图

15.3.1 ACMP 输出捕获

CONFIG1[ACIC]位使能 ACMP0 的输出连接到 PWM1_CH0，PWM1_CH0 引脚被释放到给其他复用功能。将 CONFIG1[RXDFE]设置为 01b 时，可以选择 ACMP0 输出连接到 UART1 的接收器通道。将 CONFIG1[RXDFE]设置为 10b 时，可以选择 ACMP1 输出连接到 UART1 的接收器通道。ACMP0 和 ACMP1 输出可连接到 PWDT 输入（用于 BLDC 使用），或可用作 PWM2 触发/故障输入和 ADC 硬件触发。

15.3.2 UART1_TX 调制

UART1_TX 可通过 PWM0_CH0 输出调制。 CONFIG1[TXDME]置位时， UART1_TX 与 PWM_CH0 输出相与，输出结果到 UART1_TX 管脚。将该字段清零后， UART1_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，通过设置所需的 PWM 周期与占空比后，每个通过 UART1_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。

15.3.3 UART1_RX 捕获

CONFIG1[RXDCE]置位时， UART1_RX 引脚连接 UART1 和 PWM0_CH1，PWM0_CH1 引脚被释放给其他复用功能。该字段清零后， UART1_RX 引脚仅连接 UART1。

15.3.4 UART1_RX 滤波器

CONFIG1[RXDFE]置位时，可将 ACMP0/1 输出连接至 UART1 的接收通道。要使能 UART1_RX 滤波
器功能， UART1 和 ACMP 都必须处于有效状态。如果该功能处于有效状态，UART1_RX 引脚被释放给其他复用功能。该字段清零后， UART1_RX 引脚直接连接至 UART1 模块。当 UART1_RX 捕捉功能处于有效状态时， ACMP0/1 输出也将注入 PWM0_CH1。

15.3.5 RTC 捕获

RTC 溢出可通过设置 CONFIG1 [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 引脚被释放给其他复用功能。

15.3.6 ADC 硬件触发

ADC 模块可以通过硬件触发器来启动转换。通过 CONFIG1[ADHWT1]字段设置规则组硬件触发源，CONFIG2[ADHWT2] 字段设置注入组硬件触发源。下表给出了可用的 ADC 规则组硬件触发源

表 15-1 ADC 规则组硬件触发源

ADHWT1	ADC 硬件触发源
000	RTC 溢出
001	PWM0 初始触发器
010	PWM2 初始触发器，带 8 位 可编程延迟
011	PWM2 匹配触发器，带 8 位 可编程延迟

ADHWT1	ADC 硬件触发源
100	TIMER ch0 溢出
101	TIMER ch1 溢出
110	ACMP0 输出
111	ACMP1 输出

当 ADC 硬件触发器选择 PWM2 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 PWM2 的任何触发，计数器的值由 CONFIG1[DELAY]指定。该模块的参考时钟是具有 CONFIG1[BUSREF]指定的可选预分频器的总线时钟。

15.3.7 PWM2 软件同步

PWM2 包含三个同步输入触发器，其中一个触发器通过将 1 写入，CONFIG1[PWMSYNC]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。

15.4 寄存器定义

表 15-2 CTU 寄存器映像

CTU： 0x40016000

地址	名称	说明
CTU + 0x000	CONFIG1	CTU 配置 1 寄存器
CTU + 0x004	CONFIG2	CTU 配置 2 寄存器

CONFIG1　　CTU 配置 1 寄存器　　偏移地址 = 12'h00

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	DELAY								DLYACT	ADHWT1				BUSREF		
写																
复位	0								0	0				0		
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	TXDME	PWMSYNC	0	RXDCE	ACIC	RTCC	RXDFE		0		ACTRG	0				
写																
复位	0	0	0	0	0	0	0		0		0	0				

位	名称	说明
31：24	DELAY	**PWM2 触发延时** 从 PWM2 初始或匹配触发到 ADC 硬件触发的延迟。8 位模数值允许从 0 到 255 的延迟，计数频率由 BUSREF 决定。这是一个单次计数器，当触发到

位	名称	说明
		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	DLYACT	**触发延迟有效** 该只读字段指定 PWM2 初始或匹配延迟有效时的状态。当 PWM2 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。 0： 延迟无效 1： 延迟有效
22：20	ADHWT1	**ADC 规则组硬件触发源** 选择 ADC 硬件触发源，所有触发源都在上升沿启动 ADC 转换 000： RTC 溢出作为 ADC 硬件触发源 001： PWM0 作为 ADC 硬件触发源 010： PWM2 初始化触发，具有 8 位可编程计数器延迟 011： PWM2 匹配触发，具有 8 位可编程计数器延迟 100： TIMER 通道 0 溢出作为 ADC 硬件触发 101： TIMER 通道 1 溢出作为 ADC 硬件触发 110： ACMP0 输出作为 ADC 硬件触发 111： ACMP1 输出作为 ADC 硬件触发
18：16	BUSREF	**总线时钟输出选择** 通过可选的预分频器使能总线时钟输出 000： 总线 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
15	TXDME	**UART1_TX 调制选择** 使能由 PWM0 通道 0 调制的 UART1_TX 输出 0： UART1_TX 输出直接连接到引脚排列 1： 在映射到引脚排列之前，UART1_TX 输出由 PWM0 通道 0 调制
14	PWMSYNC	**PWM 同步选择** 如果向 PWMSYNC 字段中写入 1，则为 PWM 模块生成 PWM 同步触发。注意当设置 PWMSYNC = 1 产生触发行为时，PWMSYNC 位应手动写入 0 0： 没有同步触发 1： 为 PWM 模块生成 PWM 同步触发
12	RXDCE	**UART1_RX 捕获选择** 使能 UART1_RX，由 PWM0 通道 1 捕获 0： UART1_RX 输入信号仅连接到 UART1 模块

位	名称	说明
		1： UART1_RX 输入信号连接到 UART1 模块及 PWM0 通道 1.
11	ACIC	**模拟比较器输入捕获使能** 将 ACMP0 输出连接到 PWM1 输入通道 0. 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.
10	RTCC	**实时计数器捕获** 允许 PWM1 通道 1 捕获实时计数器（RTC） 溢出 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.
9：8	RXDFE	**UART1 Rx 滤波器选择** 使能 UART1 RxD 输入，由 ACMP 滤波。当使能此功能后，任何标记有 ACMP 输入的信号都可视为 UART1。 00： RXD 输入信号直接连接到 UART1 模块 01： RXD 输入信号由 ACMP0 滤波，然后注入 UART1 10： RXD 输入信号由 ACMP1 滤波，然后注入 UART1 11： 保留
5	ACTRG	**ACMP 触发 PWM2 选择** 选择两个 ACMP 输出作为 PWM2 的 触发 0 输入 0： ACMP0 输出 1： ACMP1 输出

CONFIG2　　CTU 配置 2 寄存器　　偏移地址 = 12'h04

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位	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
读	0							ADHWT2			UARTPWDTS		ACPWDTS			
写																
复位	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

位	名称	说明
8：6	ADHWT2	**ADC 注入组硬件触发源** 选择 ADC 硬件触发源。所有触发源在上升沿开始转换。 000： RTC 溢出，作为 ADC 硬件触发源 001： PWM0 作为 ADC 硬件触发源 010： PWM2 初始化触发，具有 8 位可编程计数器延迟

位	名称	说明
		011： PWM2 匹配触发，具有 8 位可编程计数器延迟 100： TIMER 通道 0 溢出，作为 ADC 硬件触发 101： TIMER 通道 1 溢出，作为 ADC 硬件触发 110： ACMP0 输出作为 ADC 硬件触发 111： ACMP1 输出作为 ADC 硬件触发
5：4	UARTPWDTS	**PWDT UART RX 选择** 该字段选择 PWDT Internal_pwdtin 输入信号 00： UART1 RX 连接至 PWDT internal_pwdtin 通道. 01 ： UART2 RX 连接至 PWDT internal_pwdtin 通道. 10： UART3 RX 连接至 PWDT internal_pwdtin 通道）. 11： 保留
3	ACPWDTS	**PWDT ACMP_OUT 选择** 注意：如果想要使用这个功能，将首先设置 UARTPWDTS = 11 该字段选择 PWDT Internal_pwdtin 输入信号 0 ： ACMP1_OUT 连接至 PWDT internal_pwdtin 通道. 1 ： ACMP0_OUT 连接至 PWDT internal_pwdtin 通道.

16 循环冗余校验（CRC）

16.1 简介

循环冗余校验（CRC） 模块用于生成 16/32 位 CRC 校验码以进行错误校验。

CRC 模块提供实现 16 位或 32 位 CRC 标准所需的可编程多项式、写入种子和其他参数。

CRC16 / CRC32 一次计算 32bit 数据。

16.2 特性

- 使用 16 位或 32 位可编程移位寄存器的硬件 CRC 生成器电路。
- 可编程初始种子值和多项式。
- 逐位或逐字节转置输入数据或输出数据（CRC 结果）。某些 CRC 标准要求提供该选项。以 8 位读取操作访问 CRC 数据寄存器时，无法执行逐字节转置操作。在这种情况下，用户软件必须执行逐字节转置功能。
- 提供最终 CRC 结果反转选项。
- 32 位寄存器编程接口。

16.3 结构框图

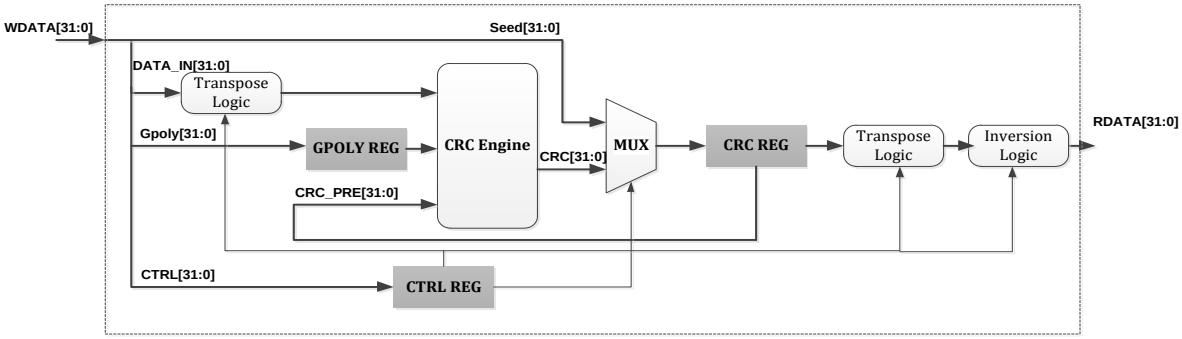

图 16-1 CRC 结构框图

16.4 寄存器定义

表 16-1 CRC 寄存器映像

CRC： 0x20081000

地址	名称	说明
CRC + 0x000	CRC_DATA	CRC 数据寄存器
CRC + 0x004	CRC_POLY	CRC 多项式寄存器

CRC + 0x008	CRC_CTRL	CRC 控制寄存器

0x000 CRC_DATA CRC 数据寄存器 00

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	字节 3								字节 2							
类型	WR								WR							
复位	0x00								0x00							
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	字节 1								字节 0							
类型	WR								WR							
复位	0x00								0x00							

位	助记符	名称	说明
[31：24]	**Byte3**	Byte3	**CRC 数据字节 3** 在 16 位 CRC 模式（CTRL[TCRC] 为 0）下，该字段并不用于种子值的编程。在 32 位 CRC 模式（CTRL[TCRC] 为 1）下，当 CTRL[WAS] 为 1 时，写入该字段的值是种子值的一部分。当 CTRL[WAS] 为 0 时，写入该字段的数据用于在 16 位和 32 位 CRC 模式下生成 CRC 校验和。
[23：16]	**Byte2**	Byte2	**CRC 数据字节 2** 在 16 位 CRC 模式（CTRL[TCRC] 为 0）下，该字段并不用于种子值的编程。在 32 位 CRC 模式（CTRL[TCRC] 为 1）下，当 CTRL[WAS] 为 1 时，写入该字段的值是种子值的一部分。当 CTRL[WAS] 为 0 时，写入该字段的数据用于在 16 位和 32 位 CRC 模式下生成 CRC 校验和。
[15：8]	**Byte1**	Byte1	**CRC 数据字节 1** 当 CTRL[WAS] 为 1 时，写入该字段的值是种子值的一部分。当 CTRL[WAS] 为 0 时，写入该字段的数据用于生成 CRC 校验和。
[7：0]	**Byte0**	Byte0	**CRC 数据字节 0** 当 CTRL[WAS] 为 1 时，写入该字段的值是种子值的一部分。当 CTRL[WAS] 为 0 时，写入该字段的数据用于生成 CRC 校验和。

0x004 CRC_POLY CRC 多项式寄存器 00

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	High															
类型	WR															
复位	0x00															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	Low															
类型	WR															
复位	0x00															

位	助记符	名称	说明
31：16	**High**	高半字	**多项式高半字** 32 位 CRC 模式下可读写（CTRL[TCRC] 为 1）。该字段在 16 位 CRC 模式下不可写（CTRL[TCRC] 为 0）
15：0	**Low**	低半字	**多项式低半字** 在 32 位和 16 位 CRC 模式下都可读写

0x008 CRC_CTRL CRC 控制寄存器 00

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型																
复位																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									**TOTW**		**TOTR**			**FXOR**	**WAS**	**TCRC**
类型									WR		WR			WR	WR	WR
复位									0		0			0	0	0

位	助记符	名称	说明
7：6	**TOTW**	写入的转置类型	写入的转置类型 定义写入 CRC 数据寄存器的数据的转置配置。有关可用的转置选项，请参见转置特性说明。 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**	读取的转置类型	读取的转置类型 识别从 CRC 数据寄存器读取的数据的转置配置，有关可用的转置选项，请参见转置特性说明。 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
3	**RSV**	保留	
2	**FXOR**	CRC 数据寄存器的反码读取	某些 CRC 协议要求最终校验和与 0xFFFFFFFF 或 0xFFFF 进行异或运算。将该 Bit 置 1，对读到的数据取补码 0： 读取时不执行 XOR 运算 1： 对读到的数据执行 XOR 运算
1	**WAS**	写入种子	作为种子写入 CRC 数据寄存器 电平变为有效后，写入 CRC 数据寄存器的值被视为种子值。当电平变为无效后，写入 CRC 数据寄存器的值用做 CRC 计算中的数据。

位	助记符	名称	说明
			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 类型	0： CRC16 1： CRC32

16.5 功能描述

16.5.1 CRC 初始化/重新初始化

要使能 CRC 计算，用户必须在适用的寄存器中对 CRC_CTRL[WAS]、CRC_GPOLY、转置所必须的参数及对 CRC 结果取补码等进行编程。使 CRC_CTRL[WAS]的电平变为有效可实现将种子值写入 CRC_DATA 寄存器。

完成 CRC 计算后，使 CRC_CTRL[WAS]的电平再次变为有效并进行种子值的编程，无论其值是新值还是以前使用过的种子值，都需要重新初始化 CRC 模块以便进行新的 CRC 计算。在进行种子值及后续数据值的编程前必须先设置其他参数。

16.5.2 CRC 计算

在 16 位和 32 位 CRC 模式下，如果所有字节都是连续的，则一次可进行 8 位、16 位或 32 位数据值的编程。非连续字节可能导致 CRC 计算错误。

16.5.3 转置特性

默认情况下，转置特性未使能。然而，某些 CRC 标准要求对输入数据或最终校验和进行转置。用户软件可根据 CRC 标准需要选择单独配置每个转置操作。数据在读写的同时进行转置。

某些协议使用小端格式对数据流来计算 CRC。在这种情况下，转置特性非常有用，可以直接按位翻转。该转置选项是 CRC 模块支持的功能类型之一。

16.5.4 转置类型

CRC 模块提供了可翻转位/字节组合的多种转置功能类型，以便根据使用的 CRC 计算方法分别使用 CTRL[TOTW] 或 CTRL[TOTR]字段写入输入数据，读取 CRC 结果。

下列转置功能类型可用于对 CRC 数据寄存器进行读写操作：

1. CTRL[TOTW] 或 CTRL[TOTR] 为 00

   不发生转置。

2. CTRL[TOTW] 或 CTRL[TOTR] 为 01

   字节中的位反转，而字节不反转。reg[31：0] 变为 {reg[24：31], reg[16：23], reg[8：15], reg[0：7]}。

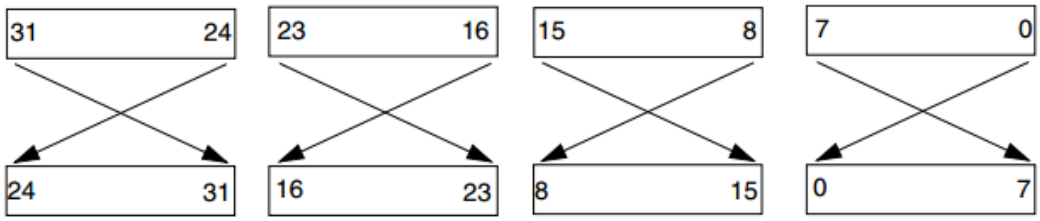

图 16-2 CTRL[TOTW] 或 CTRL[TOTR] 为 01

3. CTRL[TOTW] 或 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]}.

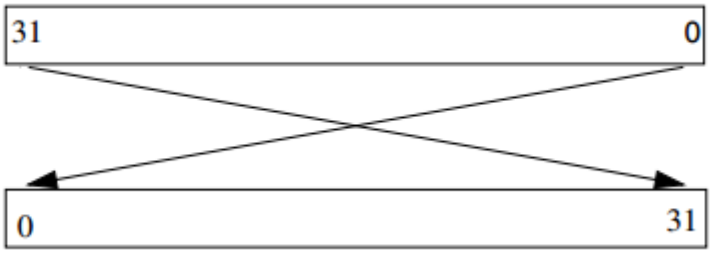

图 16-3 CTRL[TOTW] 或 CTRL[TOTR] 为 10

4. CTRL[TOTW] 或 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]}.

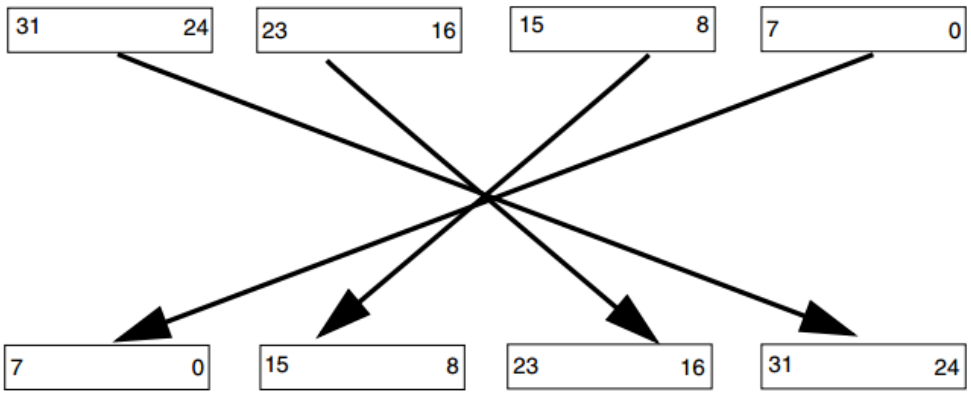

图 16-4 CTRL[TOTW] 或 CTRL[TOTR] 为 11

注意：

为了对 CRC 数据寄存器进行 8 位和 16 位的写访问，对不使用的字节（将 32 位作为一个整体）该数据转置为 0，但 CRC 仅计算有效字节 。当读取 CRC 数据寄存器中的 16 位 CRC 结果并使用转置选

项 10 和 11 时，转置后的结果数值位于 CRC[HU：HL] 字段。在读取 16 位 CRC 结果时，用户软件必须将该情形考虑在内，因此优先读取 32 位。

16.5.5 对 CRC 结果取反码

当 CTRL[FXOR] 置位后，对校验和进行异或操作。若使能对 CRC 结果取反码的功能，可在每次对 CRC 寄存器进行读操作时输出存储在 CRC 数据寄存器中的校验和的反码。当 CTRL[FXOR] 清零后，对 CRC 数据寄存器进行读操作会读到原始校验和。

16.5.6 CRC 数据寄存器（CRC_DATA）

CRC 数据寄存器包含种子、数据和校验和的数值。当 CTRL[WAS] 置位时，则对数据寄存器进行的任何写操作都被视为种子值。当 CTRL[WAS] 清零后， 则对数据寄存器进行的任何写操作都被视为用于一般 CRC 计算的数据。

在 16 位 CRC 模式下，不使用 HU 和 HL 字段来编程种子值，对这些字段进行读操作会返回不确定的值。在 32 位 CRC 模式下，所有字段都用于种子值的编程。

在写入数据时，如果所有字节都是连续的，那么可一次写入 8 位、16 位或 32 位的数值，首先写入的是 MSB 数据数值。

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，LU 字段和 LL 字段提供 CRC 结果。在 32 位 CRC 模式下，所有字段均包含此结果。如果已配置 CRC 模块，则随时对该寄存器进行读操作都会返回当前的 CRC 计算结果。

16.5.7 CRC 多项式寄存器（CRC_GPOLY）

此寄存器包含 CRC 计算所需的多项式值。HIGH 字段包含 CRC 多项式的高 16 位，仅在 32 位 CRC 模式下使用。在 CRC16 模式下会忽略对 HIGH 字段的写操作。LOW 字段包括了 CRC 多项式的低 16 位，在 CRC16 / 32 模式下都会被使用。

16.5.8 CRC 控制寄存器（CRC_CTRL）

该寄存器控制 CRC 模块的配置和操作。在开始新的 CRC 计算前，相应位必须置位。初始化新的 CRC 计算方法为：使 CTRL[WAS] 的电平变为有效，然后将种子写入 CRC 数据寄存器。

16.6 编程指南

16.6.1 16 位 CRC

如需计算 16 位 CRC：

1. 清零 CRC_CTRL[TCRC] 以使能 16 位 CRC 模式；
2. 按 CRC 计算要求对转置相关功能进行编程，并在 CTRL 寄存器中补全该选项位。更多细节，请参见 **16.5.3 转置特性**和 **16.5.5 对 CRC 结果取反码**部分；

3. 将 16 位多项式写入 CRC_GPOLY[LOW] 字段。CRC_GPOLY[HIGH] 字段在 16 位 CRC 模式下不可用；
4. 置位 CRC_CTRL[WAS] 以进行种子值的编程；
5. 将 16 位种子写入 CRC_DATA[15：0]。CRC_DATA[31：16] 未使用；
6. 清零 CRC_CTRL[WAS] 开始写入数据；
7. 将数据写入 CRC_DATA[31：0]。每次执行数据写入操作都会同时进行 CRC 计算，并且 CRC 的中间计算结果都会保存至 CRC_DATA[15：0]；
8. 当所有数据都被写入后，从 CRC_DATA[15：0]读取最终的 CRC 结果。

16.6.2 32 位 CRC

如需计算 32 位 CRC：

1. 置位 CRC_CTRL[TCRC] 以使能 32 位 CRC 模式；
2. 按 CRC 计算要求对转置相关功能进行编程，并在 CTRL 寄存器中补全该选项位。更多细节，请参见 **16.5.3 转置特性**和 **16.5.5 对 CRC 结果取反码**部分；
3. 将 32 位多项式写入 CRC_GPOLY[HIGH：LOW] 字段；
4. 置位 CRC_CTRL[WAS] 以进行种子值的编程；
5. 将 32 位种子写入 CRC_DATA[31：0]；
6. 清零 CRC_CTRL[WAS] 开始写入数据；
7. 将数据写入 CRC_DATA[31：0]。每次执行数据写入操作都会同时进行 CRC 计算，并且 CRC 的中间计算结果都会保存至 CRC_DATA[15：0]；
8. 当所有数据都被写入后，从 CRC_DATA[31：0]读取最终的 CRC 结果。CRC 计算逐字节执行，且需要两个时钟周期完成一次 CRC 计算。

17 通用输入/输出（GPIO）

17.1 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 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 GPIOx_Pmux 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。

17.2 特性

GPIO 引脚支持如下模式：

- 最多可支持 68 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（GPIOx_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（GPIOx_IDR） 或外设（可选功能输入）；
- 位置位和复位寄存器（GPIOx_BSRR） 用于 按位写入访问 GPIOx_ODR；
- 快速切换，能够每两个时钟周期更改一次；
- 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 。

17.3 结构框图

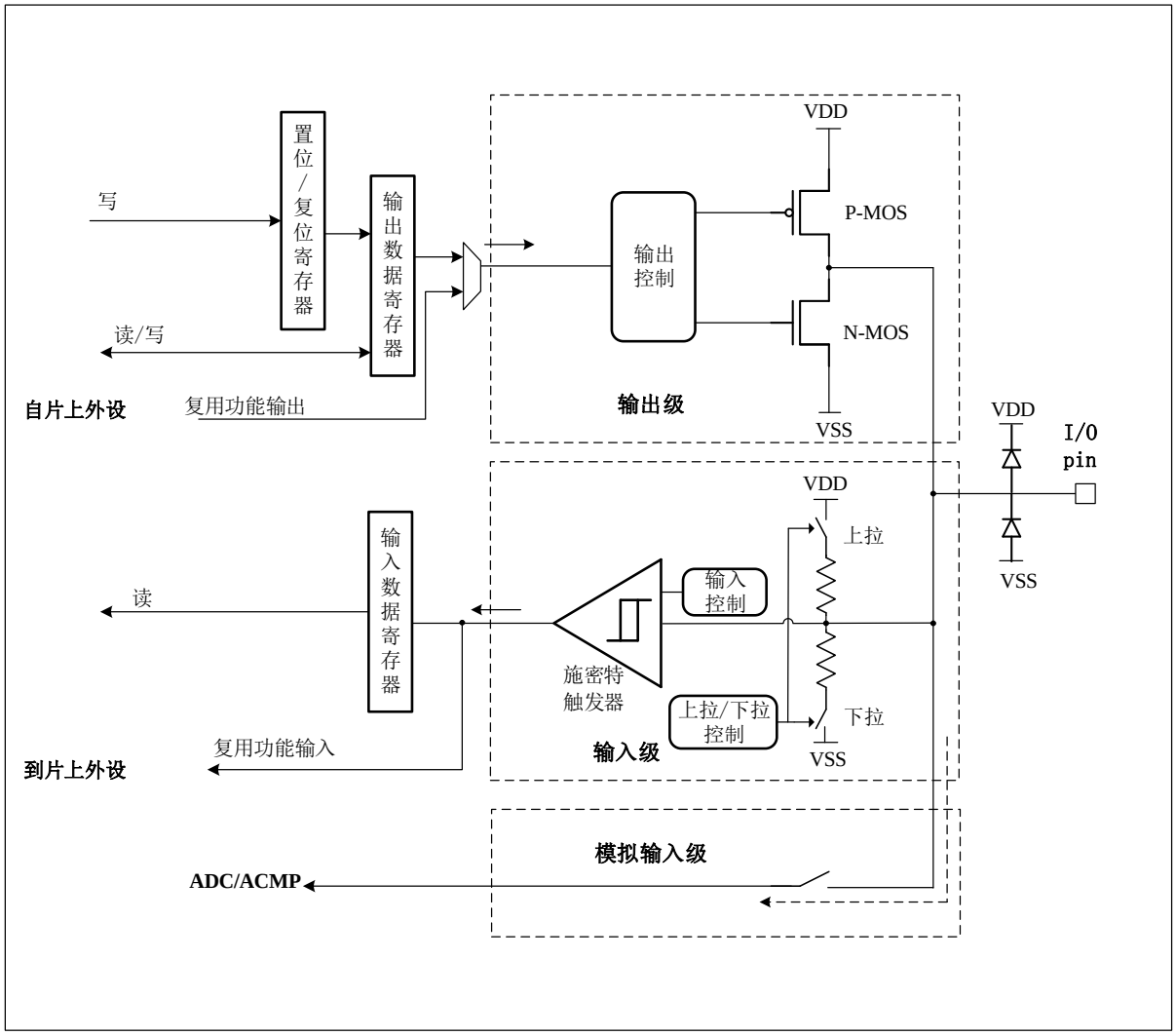

图 17-1 GPIO 结构框图

17.4 操作模式

17.4.1 外部中断

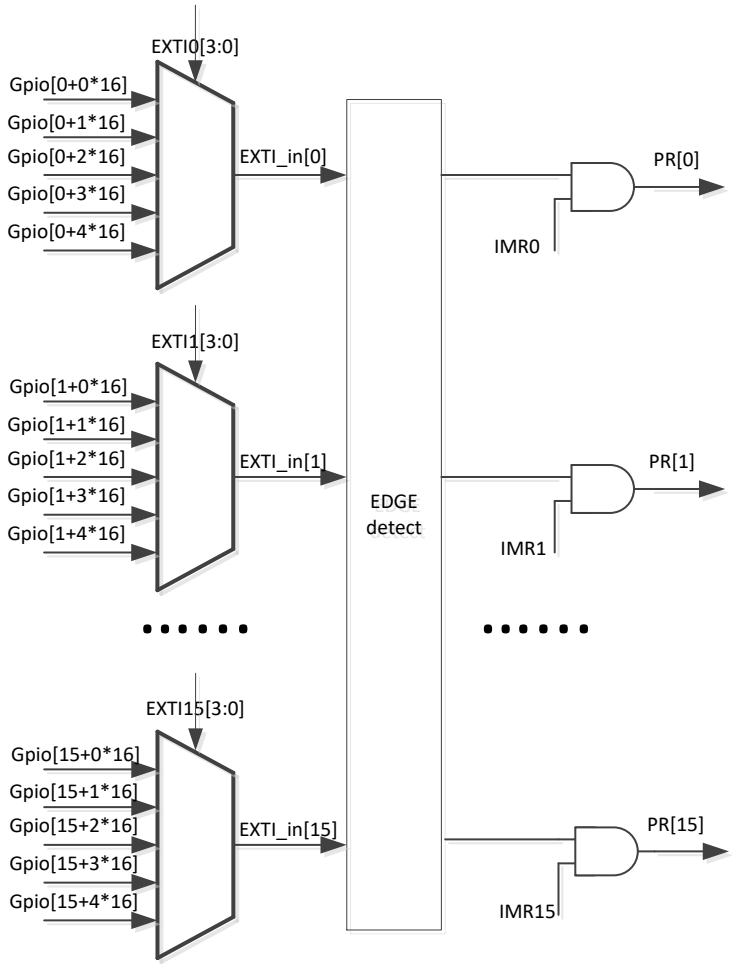

图 17-2 GPIO 外部中断

说明：

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+3*16]表示 GPIOD_PIN_15。

外部中断线与中断向量的对应关系，当 m≤4 时，EXTI_In[m]对应着中断向量 EXTIm_IRQn，当 5≤m≤9 时，EXTI_In[m]对应着中断向量 EXTI5_9_IRQn，当 10≤m≤15 时，EXTI_in[m]对应着中断向量 EXTI10_15_IRQn。

GPIO 外部中断与中断处理函数对应关系如下表示。

表 17-1 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~PE0	EXTI0	EXTI0_IRQHandler
PA1~PE1	EXTI1	EXTI1_IRQHandler
PA2~PE2	EXTI2	EXTI2_IRQHandler
PA3~PE3	EXTI3	EXTI3_IRQHandler
PA4~PE4	EXTI4	EXTI4_IRQHandler
PA5~PE5	EXTI5	EXTI5_9_IRQHandler
PA6~PE6	EXTI6	
PA7~PE7	EXTI7	
PA8~PE8	EXTI8	
PA9~PE9	EXTI9	
PA10~PE10	EXTI10	EXTI10_15_IRQHandler
PA11~PE11	EXTI11	
PA12~PE12	EXTI12	
PA13~PE13	EXTI13	
PA14~PE14	EXTI14	
PA15~PE15	EXTI15	

17.4.2 复用功能

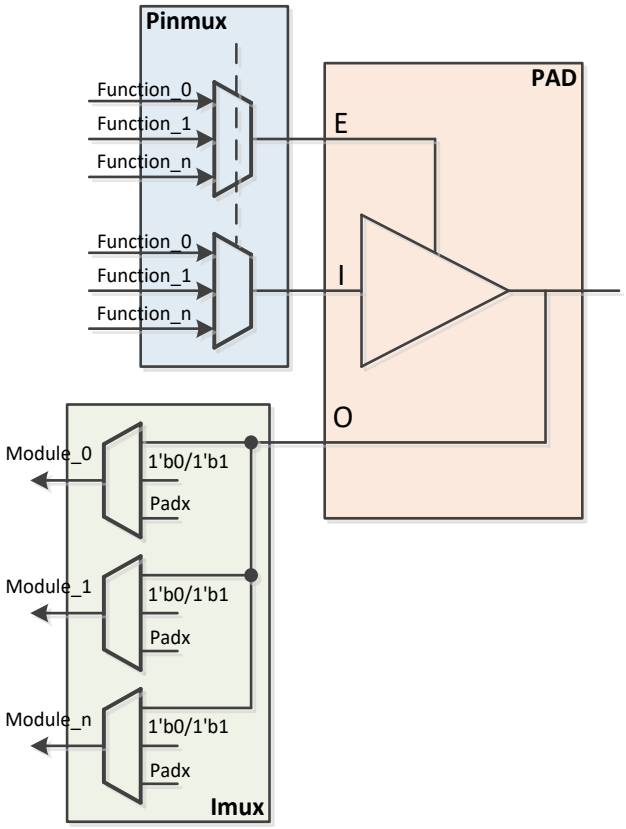

图 17-3 GPIO 复用功能

每个 IO 都有复用功能，如果我们要开始外设通信，我们应该先配置 GPIO 复用功能。每个 GPIO 的相应复用功能描述如下。

表 17-2 GPIO 复用功能描述

80 LQ FP	64 LQ FP	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	1	PA0	PA0	SPI1_NSS	TRACED3	FLASH_NSS	PMUX0[2:0]	0
2	2	PA1	PA1	SPI1_SCK	TRACECLK	FLASH_SCK	PMUX0[5:3]	1
3	3	PA2	PA2	SPI1_MISO	TRACED0	FLASH_DQ1	PMUX0[8:6]	2
4	4	PA3	PA3	SPI1_MOSI	TRACED1	FLASH_DQ0	PMUX0[11:9]	3
5	5	PA4	PA4	UART4_TX	CAN1_RX	FLASH_DQ3	PMUX0[14:12]	4
6	6	PA5	PA5	UART4_RX	CAN1_TX	FLASH_DQ2	PMUX0[17:15]	5
7		PD3	PD3	PWM1_CH0	HWLIN_TX		PMUX5[5:3]	51
8		PD4	PD4	PWM1_CH1	HWLIN_RX		PMUX5[8:6]	52
9		PD5	PD5	FTM_EXT	CAN1_STDBY		PMUX5[11:9]	53

80 LQ FP	64 LQ FP	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）
10	7	OSC_OUT	OSC_OUT					
11	8	OSC_IN	OSC_IN					
12	9	AVSS50	AVSS50					
13	10	AVDD50	AVDD50					
14	11	AVDD50_IO	AVDD50_IO					
15	12	AVSS50_IO	AVSS50_IO					
16	13	PA6	PA6	ADC_IN10	UART1_CTS	UART4_TX	PMUX0[20:18]	6
17		PD6	PD6	ADC_IN12	UART1_RTS	UART4_RX	PMUX5[14:12]	54
18	14	PA7	PA7	ADC_IN0（ACMP_IN0）	UART1_TX	UART5_TX	PMUX0[23:21]	7
19	15	PA8	PA8	ADC_IN1（ACMP_IN1）	UART1_RX	UART5_RX	PMUX0[26:24]	8
20	16	PA9	PA9	ADC_IN2（ACMP_IN2）	SPI2_NSS	HWLIN_TX	PMUX0[29:27]	9
21	17	PA10	PA10	ADC_IN3（ACMP_IN3）	SPI2_SCK	HWLIN_RX	PMUX1[2:0]	10
22	18	PA11	PA11	ADC_IN4（ACMP_IN4）	SPI2_MISO	UART3_RX	PMUX1[5:3]	11
23	19	PA12	PA12	ADC_IN5	SPI2_MOSI	UART3_TX	PMUX1[8:6]	12
24	20	PA13	PA13	ADC_IN6	I2C1_SCL	UART6_RX	PMUX1[11:9]	13
25	21	PA14	PA14	ADC_IN7	I2C1_SDA	UART6_TX	PMUX1[14:12]	14
26	22	PA15	PA15	ADC_IN8	UART2_RTS		PMUX1[17:15]	15
27	23	PB0	PB0	ADC_IN9	UART2_TX	CAN2_STDBY	PMUX1[20:18]	16
28	24	PB1	PB1	ADC_IN11	UART2_RX		PMUX1[23:21]	17
29		PD7	PD7	ADC_IN13	PWDT_EXT	PWM3_CH0	PMUX5[17:15]	55
30		PD8	PD8	ADC_IN14	CAN2_TX	PWM3_CH1	PMUX5[20:18]	56
31		PD9	PD9	ADC_IN15	CAN2_RX	PWDT_IN0	PMUX5[23:21]	57
32	25	PB2	PB2	NMI_B	UART3_TX	CAN1_STDBY	PMUX1[26:24]	18
33	26	NRST	NRST					
34	27	PB3	PB3	PWDT_IN1	UART3_RX		PMUX1[29:27]	19
35	28	PB4	PB4	PWDT_IN2	TRACED2		PMUX2[2:0]	20
36	29	PB5	PB5	UART1_TX	PWDT_IN1		PMUX2[5:3]	21

80 LQ FP	64 LQ FP	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）
37	30	PB6	PB6	UART1_RX	PWDT_IN2	PWM3_CH0	PMUX2[8:6]	22
38	31	PB7	PB7	UART1_RTS	PWM_FAULT1		PMUX2[11:9]	23
39		PD10	PD10	PWM_FAULT1	I2C2_SCL		PMUX5[26:24]	58
40	32	PB8	PB8	PWDT_IN0	I2C2_SDA		PMUX2[14:12]	24
41	33	PB9	PB9	PWM0_CH0	PWM2_CH0	CAN1_RX	PMUX2[17:15]	25
42	34	DVSS50_1	DVSS50_1					
43	35	DVDD50_1	DVDD50_1					
44	36	VPP	VPP					
45	37	PB10	PB10	PWM0_CH1	PWM2_CH1	CAN1_TX	PMUX2[20:18]	26
46	38	PB11	PB11	SPI2_NSS	PWM2_CH2		PMUX2[23:21]	27
47	39	PB12	PB12	SPI2_SCK	PWM2_CH3		PMUX2[26:24]	28
48	40	PB13	PB13	SPI2_MISO	PWM2_CH4	UART5_TX	PMUX2[29:27]	29
49	41	PB14	PB14	SPI2_MOSI	PWM2_CH5	UART5_RX	PMUX3[2:0]	30
50		PD11	PD11	UART5_TX	UART3_RTS		PMUX5[29:27]	59
51		PD12	PD12	UART5_RX	UART4_RTS		PMUX6[2:0]	60
52		PD13	PD13	PWM2_CH4	UART5_RTS		PMUX6[5:3]	61
53		PD14	PD14	PWM2_CH5	UART6_RTS		PMUX6[8:6]	62
54	42	PB15	PB15	PWM2_CH0	PWM0_EXT		PMUX3[5:3]	31
55	43	PC0	PC0	PWM2_CH1	PWM1_EXT		PMUX3[8:6]	32
56	44	PC1	PC1	PWM2_CH2	UART5_RTS		PMUX3[11:9]	33
57	45	PC2	PC2	PWM2_CH3	UART6_RTS		PMUX3[14:12]	34
58	46	PC3	PC3	UARTTX_SFLASH[1]	PWM2_CH4	PWM2_CH0	PMUX3[17:15]	35
59	47	PC4	PC4	UARTRX_SFLASH[1]	PWM2_CH5	PWM2_CH1	PMUX3[20:18]	36
60	48	PC5	PC5	I2C2_SCL	UART1_CTS		PMUX3[23:21]	37
61	49	PC6	PC6	I2C2_SDA	UART1_CTS		PMUX3[26:24]	38
62	50	PC7	PC7	JTCK_SWCLK[1]	UART3_RTS		PMUX3[29:27]	39
63	51	PC8	PC8	JTDO_TRACESWO[1]	UART2_RTS		PMUX4[2:0]	40
64	52	PC9	PC9	JTMS_SWDIO[1]	UART4_RTS	PWM_FAULT1	PMUX4[5:3]	41
65	53	PC10	PC10	CAN2_TX	UART6_TX	PWM_FAULT2	PMUX4[8:6]	42

80 LQ FP	64 LQ FP	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）
66	54	PC11	PC11	CAN2_RX	UART6_RX	PWDT_IN0	PMUX4[11:9]	43
67		PD15	PD15	JTDI[1]	CAN1_STDBY	PWDT_IN1	PMUX6[11:9]	63
68	55	BOOT	PE3	BOOT[1]				67
69	56	PC12	PC12	I2C1_SCL	UART5_TX		PMUX4[14:12]	44
70	57	PC13	PC13	I2C1_SDA	UART5_RX		PMUX4[17:15]	45
71	58	DVSS50_2	DVSS50_2					
72	59	DVDD50_2	DVDD50_2					
73		PE0	PE0	NJTRST[1]	CAN2_STDBY	PWDT_IN2	PMUX6[14:12]	64
74	60	PC14	PC14	CAN1_RX	UART4_RX	PWDT_IN3	PMUX4[20:18]	46
75	61	PC15	PC15	CAN1_TX	UART4_TX	TAMPER_RTC	PMUX4[23:21]	47
76		PE1	PE1	HWLIN_TX	PWM1_CH0		PMUX6[17:15]	65
77		PE2	PE2	HWLIN_RX	PWM1_CH1		PMUX6[20:18]	66
78	62	PD0	PD0	UART1_CTS	PWM_FAULT2		PMUX4[26:24]	48
79	63	PD1	PD1	UART2_TX	PWM0_CH0	I2C2_SCL	PMUX4[29:27]	49
80	64	PD2	PD2	UART2_RX	PWM0_CH1	I2C2_SDA	PMUX5[2:0]	50

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功能 0。**

GPIO 控制寄存器中，除 GPIOx_PINMUX 外，其他如 GPIOx_CR，GPIOx_IDR 等，x=0,1,2,3,4 分别代表 PA,PB,PC,PD,PE，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 PD1 为输出模式 GPIO4_CR[1]=1。**

3、 **复用功能配置举例：如果我们想要将 PIN1 配置为 SPI1_NSS，我们应该设置 PMUX0[2：0]=1。**

4、 **VPP 引脚为测试引脚，产品使用应用时必须保持悬空。**

17.5 应用说明

17.5.1 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 **GPIOx_PR** 寄存器中。

17.5.2 复用功能

为了优化小封装中的功能，引脚通过信号多路复用提供多种可用的功能。PINMUX 寄存器控制外部引脚上的信号。请参考 **GPIOx_PINMUX** 以及 **17.4.2** 节。

17.5.3 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 端口配置为输入模式，RST 和 ARM 调试接口除外。用户可以对 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（GPIOx_ODR）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（GPIOx_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 GPIOx_PU 和 GPIOx_PD 寄存器中的值。

17.5.4 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 PINMUX 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 32kHz 振荡器产生的时钟。

17.6 寄存器定义

表 17-3 GPIO 寄存器映像

GPIO： 0x40001000

地址	名称	说明
GPIO + OFFSET	**GPIOx_CR**	端口配置
GPIO + OFFSET	**GPIOx_IDR**	端口输入数据
GPIO + OFFSET	**GPIOx_ODR**	端口输出数据
GPIO + OFFSET	**GPIOx_BSRR**	端口置位/复位
GPIO + OFFSET	**GPIOx_BRR**	端口复位
GPIO + OFFSET	**GPIOx_PD**	下拉使能
GPIO + OFFSET	**GPIOx_PU**	上拉使能
GPIO + OFFSET	**GPIOx_E4_E2**	驱动能力选择
GPIO + OFFSET	**GPIOx_PINMUX**	复用功能选择
GPIO + OFFSET	**GPIOx_PR**	外部中断标志暂停
GPIO + OFFSET	**GPIOx_IMR**	中断掩码在线
GPIO + OFFSET	**GPIOx_RTSR**	上升沿触发事件配置
GPIO + OFFSET	**GPIOx_FTSR**	下降沿触发事件配置
GPIO + OFFSET	**GPIOx_EXTICR**	外部中断

GPIOx_CR 寄存器（x=0…4）　　偏移地址 =（12'h00, 12'h30, 12'h60, 12'h90, 12'hc0）

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读写	保留															
缺省	0															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	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）

位	名称	说明
[31：16]	保留	**未使用**
[15：0]	Mode	**Mode（y）：** 端口 y 配置位 这些位由软件写入，以配置 I/O 方向模式 0：输入（复位默认状态） 1：输出模式

GPIOx_IDR 寄存器（x=0…4）　　偏移地址 =（12'h04, 12'h34, 12'h64, 12'h94, 12'hc4）

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读写	保留															
缺省	0															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	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）
写																
缺省	0															

位	名称	说明

位	名称	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

GPIOx_ODR 寄存器（x=0…4）　　偏移地址 =（12'h08, 12'h38, 12'h68, 12'h98, 12'hc8）

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读写	保留															
默认	0															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读写	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）
缺省	**0**															

位	名称	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,D） 和 GPIOx_BRR 寄存器（x = A,B,C,D） 来单独设置和复位 ODR 位。**

GPIOx_BSRR 寄存器（x=0…4） **偏移地址 =（12'h0c, 12'h3c, 12'h6c, 12'h9c, 12'hcc）**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写	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）
缺省	0															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0															
写	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）
缺省	0															

位	名称	说明
[31：16]	BR	**BR（y）：** Port（y）复位位 y 这些位只写。 对这些位的读取返回 0x0000 0：对应的 ODRy 位无操作 1：复位对应的 ODRy 位
[15：0]	BS	**BS（y）：** Port（y） 设置位 y 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。 0：对应的 ODRy 位无操作 1：置位对应的 ODRy 位 **注意 ：如果 BS 和 BR 都配置，则 BR 具有更高的优先级。**

GPIOx_BRR 寄存器（x=0…4）　　偏移地址 =（12'h10, 12'h40, 12'h70, 12'ha0, 12'hd0）

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读 写	保留															
缺省	0															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	**0**															
写	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）
缺省	0															

位	名称	说明
[15：0]	BR	**BR（y）：** Port（y） 复位位 y 这些位是只写的，对这些位的读取返回 0x0000 0：对应的 ODRy 位无操作 1：复位对应的 ODRy 位

GPIOx_PD 寄存器（x=0…4）　　偏移地址 =（12'h18, 12'h48, 12'h78, 12'ha8, 12'hd8）

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读 写	保留															
缺省	0															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读 写	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）
缺省																

位	名称	说明
[0] [1] [2]	PD	**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）

位	名称	说明
……		
		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。**

GPIOx_PU 寄存器（x=0…4）　　偏移地址 =（12'h1c, 12'h4c, 12'h7c, 12'hac, 12'hdc）

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读 写	保留															
缺省	0															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读 写	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）
缺省	0															

位	名称	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：上拉和下拉不支持同时使能。**

GPIOx_E4_E2 寄存器（x=0…4）　　偏移地址 =（12'h20, 12'h50, 12'h80, 12'hb0, 12'he0）

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读 写	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）	
缺省	0															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读 写	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（16x+0）	
缺省	0															

位	名称	说明
[1：0] [3：2] [5：4] ……	E4_E2	**E4_E2（y）：** 驱动能力选择 这些位可以通过软件进行读写

E4	E2	Driving
0	0	4mA
0	1	8mA
1	0	12mA
1	1	16mA

GPIOx_PINMUX 寄存器（x=0…6）

偏移地址 =（12'h140, 12'h144, 12'h148, 12'h14C, 12'h150, 12'h154, 12'h158）

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读 写	保留		PINMUXx[29：27]			PINMUXx[26：24]			PINMUXx[23：21]			PINMUXx[20：18]			PINMUXx[17：15]	
缺省	0		0			0			0			0			0	
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读 写		PINMUXx[14：12]			PINMUXx[11：9]			PINMUXx[8：6]			PINMUXx[5：3]			PINMUXx[2：0]		
缺省		**0**			**0**			**0**			**0**			**0**		

位	名称	说明
[2：0] [5：3] [8：6] ……	PINMUXx	**PINMUXx（y）：** 复用功能 这些位由软件写入，以配置复用功能 I/O 000： 功能 0 001：功能 1 010： 功能 2 011： 功能 3

GPIOx_PR 寄存器（x=0） **偏移地址 =（12'h160）**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读 写	保留															
缺省	0															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	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）
写	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
缺省	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	名称	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。

W1C：write '1' clear.

GPIOx_IMR 寄存器（x=0） **偏移地址 =（12'h164）**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读 写	保留															
缺省	0															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读 写	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）
缺省	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	名称	说明
[0] [1]	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

位	名称	说明
[2]		1：来自 y 输入线的中断请求未被屏蔽
……		

GPIOx_RTSR 寄存器（x=0） **偏移地址 =（12'h168）**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读写	保留															
缺省	0															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读写	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）
缺省	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	名称	说明
[0]	RTSR	**RTSR（y）：** y 输入线的上升沿触发事件配置
[1]		0：输入线 y 上升沿触发禁用（对于事件和中断）
[2]		1：输入线 y 上升沿触发使能（对于事件和中断）
……		

GPIOx_FTSR 寄存器（x=0） **偏移地址 =（12'h16C）**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读写	保留															
缺省	0															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读写	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）
缺省	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	名称	说明
[0] [1] [2] ……	FTSR	**FTSR（y）：** y 输入线的下降沿触发事件配置 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）

GPIOx_EXTICR 寄存器（x=0…3）　　偏移地址 =（12'h170, 12'h174, 12'h178, 12'h17C）

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读 写	保留															
缺省	0															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读 写	EXTI（4*x+3）				EXTI（4*x+2）				EXTI（4*x+1）				EXTI（4*x+0）			
缺省	0				0				0				0			

位	名称	说明
[3：0] [7：4] [11：8] ……	EXTI	**EXTI（y）：** EXTI y 配置 这些位由软件写入，用于选择 EXTI（y）外部中断源输入 0000： PA[x] 引脚 0001： PB[x] 引脚 0010： PC[x] 引脚 0011： PD[x] 引脚 0100： PE[x] 引脚

18 I2C 总线模块（I2C）

18.1 简介

I2C（Inter-Integrated Circuit）是内部整合电路的称呼，是一种简单、双向二线制同步串行总线。它只需要两根线（SCL 和 SDA）即可在连接于总线上的器件之间传送信息。SCL 是由主机驱动的时钟信号。SDA 是双向数据信号，可由主设备或从设备驱动。

18.2 特性

- 支持主机和从机模式操作；
- 支持 I2C 标准模式 100kHz 和快速模式 400kHz；
- 当仲裁丢失时，主机模式自动切换至从机模式 ；
- 主机可编程传输比特率；
- 从机地址识别中断；
- 从机支持可扩展 10 位地址；
- 从机支持低功耗模式唤醒；
- 从机支持监测功能；
- 可编程输入毛刺过滤器；
- 支持 SCL 延伸；
- 软件控制应答位；
- 由中断驱动的逐字节数据传输；
- 总线开始/停止信号检测；
- 支持 DMA 发送和接收。

18.3 结构框图

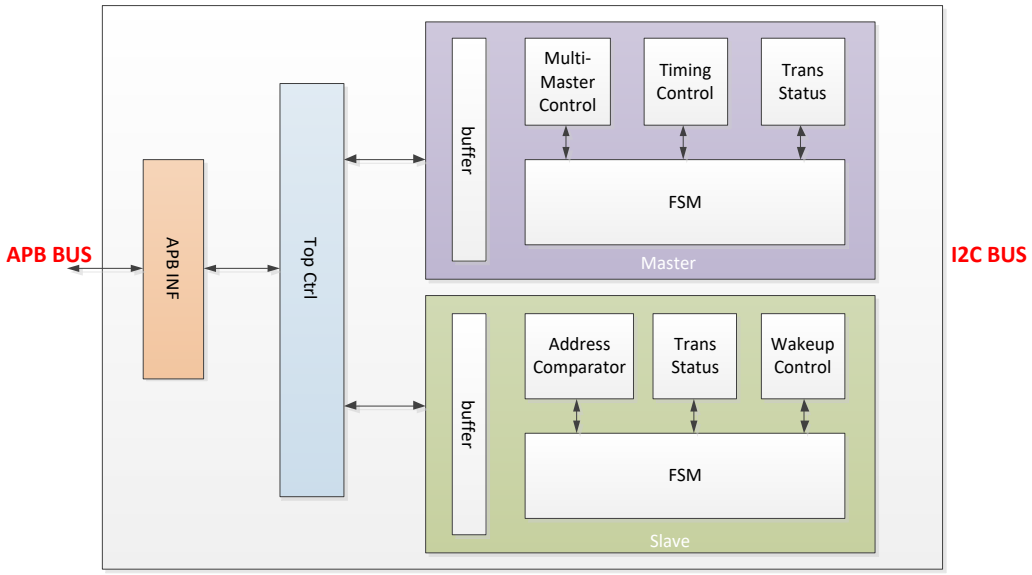

图 18-1 I2C 结构框图

18.4 操作模式

I2C 模块支持唤醒功能，可以将 MCU 从停止（Stop）模式唤醒。当 I2C 配置为从机模式时，用户可以通过设置 CONTROL_1 寄存器的 WUEN 位来使能唤醒功能。

当 MCU 进入 停止（Stop）模式，I2C 从机正在等待地址匹配。如果主机向 I2C 从机发送匹配地址，则在接收到所有匹配地址后，MCU 将从停止（Stop）模式唤醒。

18.5 寄存器定义

表 18-1 I2C 寄存器映像

I2C1 基地址： 0x4000e000
I2C2 基地址： 0x4000f000

地址	名称	宽度	说明
I2Cx +0x000	ADDR_1	8	地址寄存器 1
I2Cx +0x004	ADDR_2	8	地址寄存器 2
I2Cx +0x008	SAMPLE_CNT	8	波特率配置寄存器 1
I2Cx +0x00c	STEP_CNT	8	波特率配置寄存器 2
I2Cx +0x010	CONTROL_1	8	控制寄存器 1
I2Cx +0x014	CONTROL_2	8	控制寄存器 2
I2Cx +0x018	CONTROL_3	8	控制寄存器 3
I2Cx +0x01c	CONTROL_4	8	控制寄存器 4
I2Cx +0x020	STATUS_1	8	状态寄存器 1
I2Cx +0x024	STATUS_2	8	状态寄存器 2
I2Cx +0x028	DGL_CFG	8	毛刺滤波器配置寄存器
I2Cx +0x02c	DATA	9	数据端口寄存器

地址	名称	宽度	说明
I2Cx +0x030	START_STOP	8	主机开始和停止信号控制寄存器

I2Cx+0x000　　ADDR1　　地址寄存器 1

位	31：8	7	6	5	4	3	2	1	0
名称		AD[7：1]							
类型		R/W							
复位		1	1	1	1	1	1	1	

位	助记符	名称	说明
31：8			
7：1	AD[7：1]	从地址	**当 I2Cx 为从机模式时，指定地址** 用于 7 位地址模式以及 10 位地址模式中的低七位
0			

I2Cx+0x004　　ADDR_2　　地址寄存器 2

位	31：8	7	6	5	4	3	2	1	0
名称							AD[10：8]		
类型							R/W		
复位							1	1	1

位	助记符	名称	说明
31：8			
7：3			
2：0	AD[10：8]	从机扩展地址	**当 I2Cx 为从机模式时，指定地址** 包含 10 位地址模式中的高三位，该字段仅在 *ADEXT* 位置 1 时有效。

I2Cx+0x008　　SAMPLE_CNT　　波特率配置寄存器 1

位	31：8	7	6	5	4	3	2	1	0
名称		SAMPLE_CNT_DIV							
类型		R/W							
复位		0	0	0	0	0	1	0	0

位	助记符	名称	说明
31:8			
7:0	**SAMPLE_CNT_DIV**	SAMPLE_CNT_DIV	**调整每个采样点的宽度** 采样宽度 =（*sample_cnt_div* +1） *T_{bclk}

I2Cx+0x00c　　STEP_CNT　　波特率控制寄存器 2

位	31：8	7	6	5	4	3	2	1	0
名称		**STEP_CNT_DIV**							
类型		R/W							
复位		0	0	0	0	0	1	0	0

位	助记符	名称	说明
31:8			
7：0	**STEP_CNT_DIV**	STEP_CNT_DIV	**指定每半脉冲宽度的样本数，比如，每个高或低脉冲** 半波特率宽度=（*step_cnt_div*+1）*采样宽度 **注意：建议 STEP_CNT_DIV 最小为 is 3。**

I2Cx+0x010　　CONTROL_1　　控制寄存器 1

位	31：8	7	6	5	4	3	2	1	0
名称		**I2CEN**	**I2CIE**	**MSTR**	**TX**	**TACK**	**WUEN**		
类型		R/W	R/W	R/W	R/W	R/W	R/W		
复位		0	0	0	0	0	0		

位	助记符	名称	说明
31:8			
7	**I2CEN**	I2C_EN	**I2C 模块使能** 1： 使能 0： 禁用
6	**I2CIE**	I2C_IE	**I2C 中断使能** 1： 使能 IIC 模块中断 0： 禁用
5	**MSTR**	MSTR	**I2C 操作模式选择** 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**	TX	**传输方向选择** 1： 发送（TX） 0： 接收（RX） 选择主机的传输方向。在主机模式下，必须根据所需的传输类型

位	助记符	名称	说明
			将此位置位。因此，对主机地址周期而言，此位始终置位。
3	**TACK**	TACK	**应答控制** 指定在主机和从机接收器的数据应答周期过程中驱动到 SDA 的值。 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 **注意：从机地址字节 ACK 位是 硬件自动响应，软件不关注该字段。如果从机地址匹配，则发送 ACK 信号，否则，发送 NACK 信号。**
2	**WUEN**	WAKEUP_EN	**唤醒功能使能** 当出现从机地址匹配时，I2C 从机模式可将 MCU 从没有外设总线运行的低功耗模式唤醒。 1： 在低功耗模式下使能唤醒功能 0： 禁用唤醒功能 **注意：当 I2C 模块处于从机模式，当发生 7 位地址、2 字节 10 位地址（如果 ADEXT=1），或通用调用广播地址（GCAEN=1）匹配时， I2C 模块将生成从低功耗模式唤醒 MCU 的请求。**
1			
0			

I2Cx+0x014　　CONTROL_2　　控制寄存器 2

位	31：8	7	6	5	4	3	2	1	0
名称		**GCAEN**	**ADEXT**		**SYNCEN**	**ARBEN**			**STREN**
位		R/W	R/W		R/W	R/W			R/W
复位		0	0		0	0			0

位	助记符	名称	说明
31:8			
7	**GCAEN**	GCA_EN	**从机通用广播地址使能** 1： 使能 0： 禁用
6	**ADEXT**	ADEXT	**从机地址扩展** 1： 10 位地址模式 0： 7 位地址模式
5			
4	**SYNCEN**	SYNC_EN	**主机 SCL 同步使能** 使能主机的 SCL 同步功能 1： 使能 0： 禁用
3	**ARBEN**	ARB_EN	**主机仲裁使能**

位	助记符	名称	说明
			使能主机仲裁功能 1： 使能 0： 禁用 **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。** **注意：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。**
2			
1			
0	**STREN**	STR_EN	**从机 SCL 拉伸使能** 使能此位，从机硬件将在每个字节第 9 个 SCL 下降沿后将 SCL 拉低。 1： 使能 0： 禁用 **注意：在 SAMF=1 后，如果 SRW=1， RXFF=1 将造成 scl stretch；否则 SRW=0, TXEF=1 将造成 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。** **注意：在从机模式下，当使能 GCA_EN 或 MNTEN 时，从机不能拉低 SCL。**

I2Cx+0x018 **CONTROL_3** **控制寄存器 3**

位	31：8	7	6	5	4	3	2	1	0
名称		RXOFIE	TXUFIE	RXFIE	TXEMIE			NACKIE	MNTEN
类型		R/W	R/W	R/W	R/W			R/W	R/W
复位		0	0	0	0			0	0

位	助记符	名称	说明
31:8			
7	**RXOFIE**	SLA_RXOF_IE	**从机 RX Buffer 上溢错误中断使能** 1： 使能 0： 禁用
6	**TXUFIE**	SLA_TXUF_IE	**从机 TX Buffer 下溢错误中断使能** 1： 使能 0： 禁用
5	**RXFIE**	SLA_RXF_IE	**从机 RX Buffer 满中断使能** 1： 使能 0： 禁用

位	助记符	名称	说明
4	**TXEIE**	SLA_TXE_IE	**从机 TX Buffer 空中断使能** 1： 使能 0： 禁用
3			
2			
1	**NACKIE**	NACK_IE	**NACK 中断使能** 1： 使能 0： 禁用 注意：在 I2C 从机模式下，此中断不使能此位。
0	**MNTEN**	SLA_MNT_EN	**从机监测功能使能** 1： 使能 0： 禁用

I2Cx+0x01c **CONTROL_4** **控制寄存器 4**

位	31：8	7	6	5	4	3	2	1	0
名称								**DMA RXEN**	**DMA TXEN**
类型								R/W	R/W
复位								0	0

位	助记符	名称	说明
31:8			
7			
6:2			
1	**DMARXEN**	DMA_RX_EN	**DMA RX 使能** 1： 使能 0： 禁用
0	**DMATXEN**	DMA_TX_EN	**DMA TX 使能** 1： 使能 0： 禁用

I2Cx+0x020 **STATUS_1** **状态寄存器 1**

位	31：8	7	6	5	4	3	2	1	0
名称		**BND**	**SAMF**	**BUSY**	**ARB LOST**	**READY**	**SRW**		**RACK**
类型		R/W	RW	R	R/W	R	R		R/W
复位		0	0	0	0	**1**	0		0

位	助记符	名称	说明
31:8			
7	**BND**	Byte End	**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。当 I2C 为 RX 时，读取 DATA 寄存器会清零此位。否则，当 I2C 为 TX 时，写 DATA 寄存器会清零此位。 **注意： 当 I2C 为 从机模式且 MNTEN=1，当 TCF 为 '1'时，读取 DATA 寄存器将清零此位。** **注意：写 '1' 也会清零此位。**
6	**SAMF**	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** **a. 7 位地址，地址匹配** **b. 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。** **c. 通用广播地址匹配** **注意： 写 '1' 清零此位。**
5	**BUSY**	BUSY	**总线忙** 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。 1： 总线忙 0： 总线空闲
4	**ARBLOST**	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。** **注意：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**	READY	**准备好接收新的命令** 该位指示内部硬件状态，仅对主机模式有效。 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 **注意：此位对主机模式有效，也可在主机产生 START/STOP 信号时使用。当 I2C 模块处于主机模式时，写 START_STOP[0]会在总线上产生 START 信号，然后 DGL_CFG[4] 置 1，这种情况必须等待 READY 位为 1，软件可以写 DATA 寄存器来发送地址字节。** **注意：和 START 信号相似，写 TART_STOP[1]位会在总线上产生 STOP 信号，然后 DGL_CFG[6] 置位。这种情况必须等待 READY 位为 1, 软件可以通过写 START_STOP[0] , 发起**

位	助记符	名称	说明
			下一次传输
2	**SRW**	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
1			
0	**RACK**	RACK	**接收应答** 该字段只在发送方向有效（TX） 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 将会设置中断，写 1 清零此位 。**

I2C+0x024　　STATUS_2　　状态寄存器 2

位	31：8	7	6	5	4	3	2	1	0
名称		**IDLE**				**RXOF**	**TXUF**	**RXFF**	**TXEF**
类型		R				R/W	R/W	R/W	R/W
复位		1				0	0	0	1

位	助记符	名称	说明
31:8			
7	**IDLE**	IDLE	I2C 内核硬件状态 1： 空闲 0： 非空闲
3	**RXOF**	RXOF	**从机 RX 缓冲区上溢标志** 1： RX 缓冲区上溢 0： 未溢出 **注意：当 RX 缓冲区溢出时，新接收的数据未存储在 RX 缓冲区中。** **注意：写 '1' 清零此位。**
2	**TXUF**	TXUF	**从机 TX 缓冲区下溢标志** 1： TX 缓冲区下溢 0： 未下溢 **注意：当 TX 缓冲区下溢时，再次发送时发送缓冲区的最后一个 DATA。** **注意：写'1' 清零此位。**
1	**RXFF**	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**	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**

I2C+0x028　　DGL_CFG　　毛刺滤波器控制寄存器

位	31：8	7	6	5	4	3	2	1	0
名称		**DGLEN**	**STOPF**	**SSIE**	**STARTF**	**DGL_CNT**			
类型		R/W	R/W	R/W	R/W	R/W			
复位		0	0	0	0	0	0	0	0

位	助记符	名称	说明
31:8			
7	**DGLEN**	DGL_EN	**毛刺滤波器使能** 1： 使能 0： 禁用
6	**STOPF**	STOP FLAG	**总线 STOP 标志** 当 I2C 总线上检测到 STOP 信号时，由硬件置位此位 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 **注意： 写 “1” 清零此位**
5	**SSIE**	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：在中断服务程序中，写 ‘1 ’ 清零 STARTF 或 STOPF 标志，然后中断自动清零。**
4	**STARTF**	STARTF	**总线 START 标志** 1： 在 I2C 总线上检测到 START 0： 未检测到 START **注意：写 ‘1’ 清零此位**
3:0	**DGL_CNT**	DGL_CNT	**毛刺滤波器计数器** 控制滤波器必须滤除的毛刺宽度（从 I2C 模块时钟周期的角度而言）。对于宽度小于或等于此宽度设置的任意毛刺，滤波器不会允许该毛刺通过。 0h： 无滤波器/旁路 1-Fh：对宽度最多为 1-15 时钟周期的毛刺进行过滤

I2C+0x02c　　DATA　　数据端口寄存器

位	31：9	8	7	6	5	4	3	2	1	0
名称		**MAK**	**DATA**							
类型		R	R/W							
复位		**1**	1	1	1	1	1	1	1	1

位	助记符	名称	说明
31:9			
8	**MAK**	Monitor ACK	**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** **注意：对于监测器，DATA[7：0]是 I2C 总线上传送的数据，DATA[8] 为 ACK 位**
7:0	**DATA**	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，从该寄存器读取数据意味着开始接收下一个字节的数据。 **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。** **注意：读取该寄存器会返回 RX 缓冲区数据。在主机模式下，RX 缓冲区默认为 0xff，在从机模式下， RX 缓冲区默认为 0x1ff。**

I2C+0x030　　START_STOP　　START_STOP 寄存器

位	31：8	7	6	5	4	3	2	1	0
名称								**STOP**	**START**
类型								R/W	R/W
复位								0	0

位	助记符	名称	说明
31:8			
1	**STOP**	STOP	**主机触发停止信号** 写"1" , 主机会发送 STOP 信号 读取此位会始终返回 "0"
0	**START**	START	**主机触发起始信号** 写"1" ，主机将会发送 START（RESTART） 信号 读取此位会始终返回 "0"

注意： 保留位必须保持默认值，不能变化

部分不可用，读取始终返回 0.

18.6 功能描述

18.6.1 数据流程&算法

对于发送器，数据被写入一个 8 位的 TX 缓冲区。 然后按照波特率控制逻辑加载到 TX 移位寄存器。先输出最高有效位，发送器在每个字节的第 9 个 SCL 采样应答位。

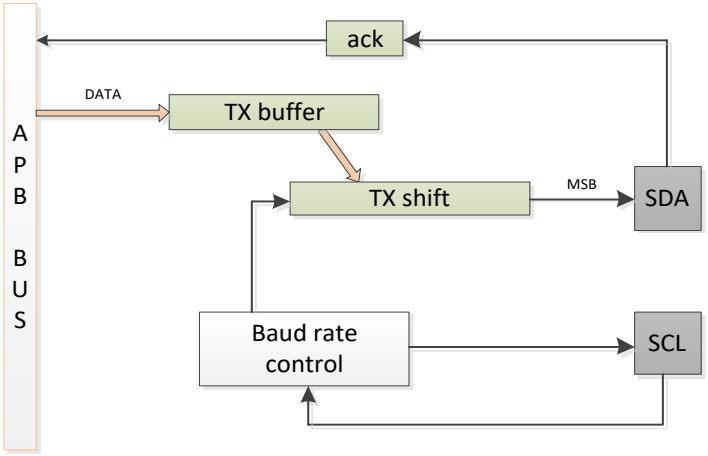

图 18-2 发送器数据流程

对于接收器，RX 移位采样并在 SDA 线输入数据中移位。在每个字节每第 8 个 SCL 之后将接收的数据存储到 8 位 RX 缓冲器中，最高有效位优先。应答位在第 9 个 SCL 脉冲的 SDA 线上。

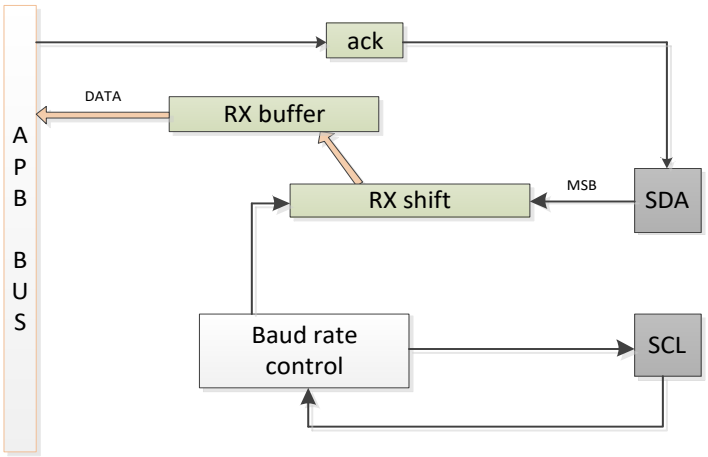

图 18-3 接收器数据流程

18.6.2 输入输出时序

所有操作以 START（S） 开始，STOP（P）位结束。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（见图 18-4）。

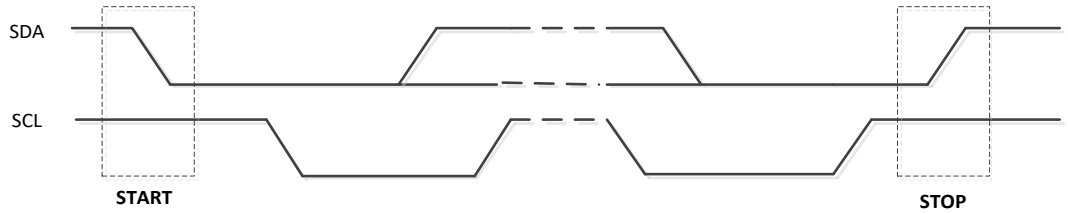

图 18-4 START 和 STOP 条件

SDA 线上的每个字节必须长 8 位。每次传输可以传输的字节数不受限制。每个字节后面必须跟一个 ACK 位。首先使用最高有效位（MSB）传输数据（见图 18-5）。

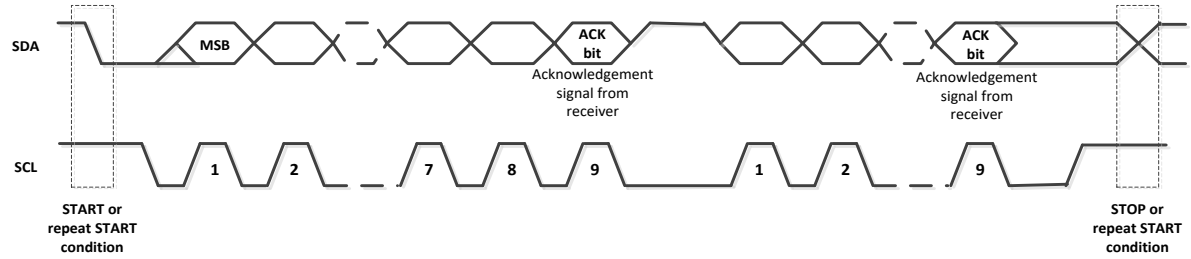

图 18-5 数据传输格式

18.6.3 主机 SCL 输出定时设置

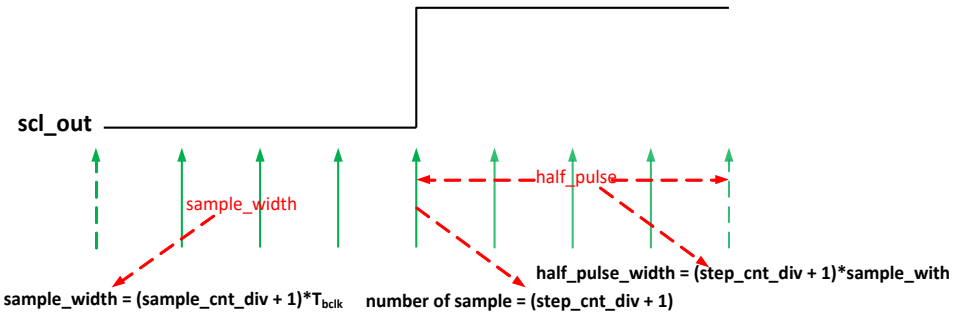

图 18-6 波特率生成

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（50MHz）。

18.6.4 数据传输

I2C 模块是一个逐字节传输的串行接口，需要 CPU 控制每个字节。写 START_STOP [0]'1'触发 START 信号到 I2C 总线，并初始化一个发送。传输最终在 I2C 总线上发出 STOP 信号。STOP_信号由 START_STOP [1]写入'1'来进行触发。每个数据字节完成后，将设置 BND 标志。根据传输方向，在 BND = 1 后软件开始读写数据。

写 BND'1'也可以清除 BND。当最后一个字节在主机写从机模式内完成时，软件可以写入 BND'1'而不是将伪数据写入 DATA。

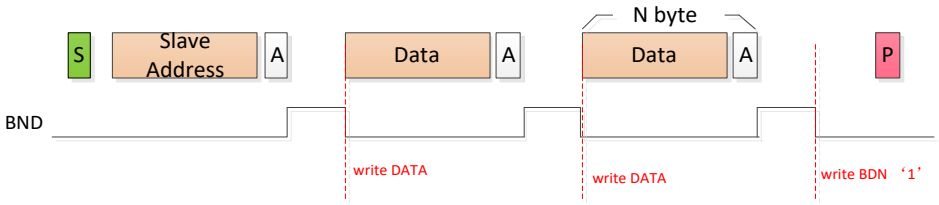

图 18-7 主机写从机模式的 BND 序列

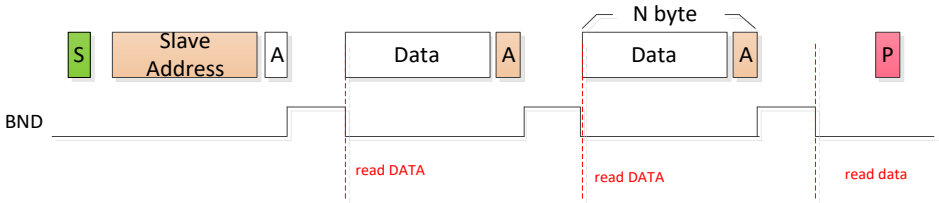

图 18-8 主机读从机模式的 BND 序列

18.6.5 DMA 操作

I2C 模块支持逐字节 DMA TX/RX 请求传输。DMA 请求仅用作数据传输。DMA 仅用于数据传输阶段，并且 START/STOP/RESTART 信号仍然由软件触发。发送器和接收器之间的 DMA 操作是不同的。在接收器的倒数第二个字节完成之后 EOT_1 信号有效。

18.6.5.1 主机发送器

对于主机发送器，在 START 信号之后，I2C 协议规定第一个字节是从机地址。该地址字节由软件写入。DMA TX 请求（DMA_TX_REQ）将在第 1 个字节应答位（第 9 个 SCL）之后置位。请注意，设置 DMA_TX_REQ 需要一个 ACK 条件（从机必须在第 9 个 SCL 上返回低电平）。如果 NACKIE=1，则当发生 NACK 时，I2C 内核将产生 CPU 中断请求。

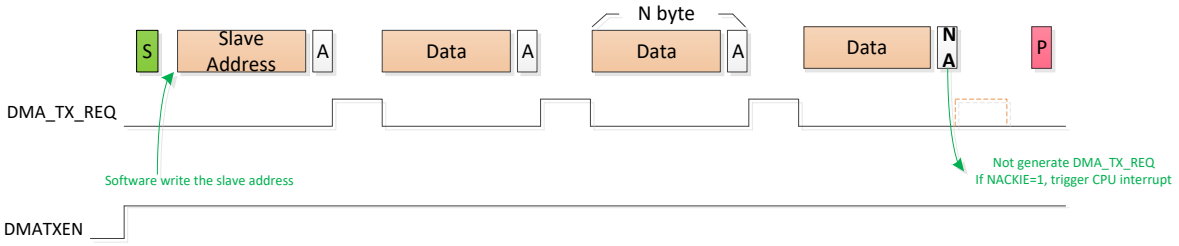

图 18-9 主机发送器情形 1

如果从机所有的数据字节都收到 ACK 回复，并且 DMA 已经传输完成，则 I2C 模块仍会生成 DMA TX 请求，因为 I2C 模块无法判断传输长度。因此软件需要在 DMA 传输完成后写入 BND“1”以清零此位字段。

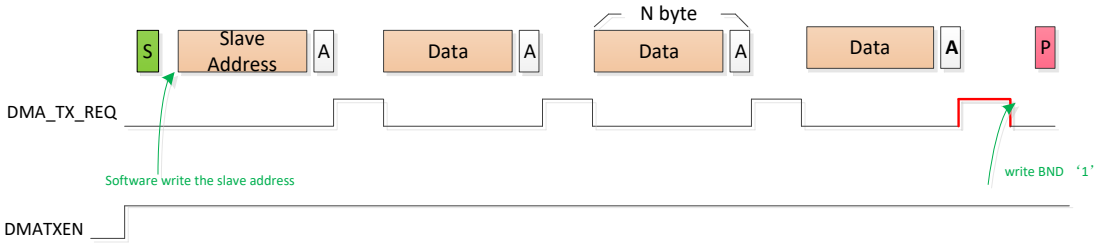

图 18-10 主机发送器情形 2

18.6.5.2 主机接收器

对于主机接收器，START 条件和从机地址字节由软件触发。方向切换和虚拟读数据寄存器动作也是必要的。在第一个数据字节完成后，第一个 DMA RX 请求由硬件自动生成。当要接收的字节数等于或大于 2 时，DMA 控制器发送硬件信号 eto_1=1，对应于最后一个数据字节（number_of_bytes - 1）。I2C 在 eto_1 = 1 之后的下一个字节后自动发送 NACK。如果使能 stop 信号，软件可以在 DMA 传输完成中断程序中生成 STOP 条件。

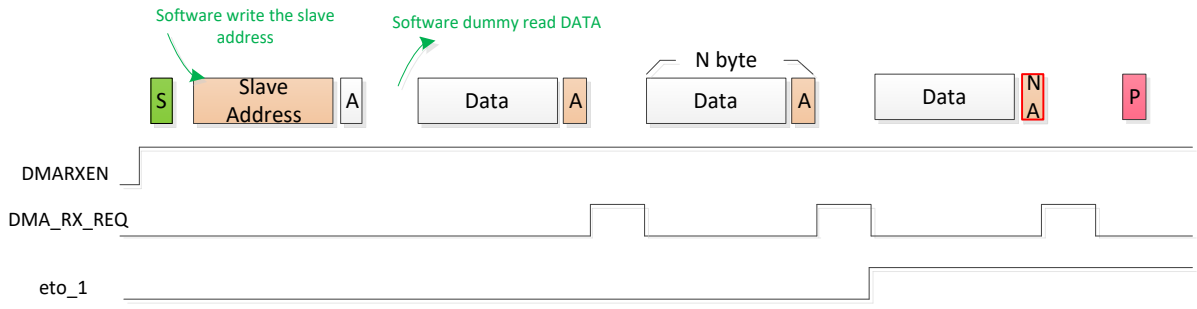

图 18-11 主机接收器

18.6.5.3 从机发送器

如果从机地址匹配且地址字节的 LSB 为“1”，则从机为发送器。I2C 硬件在从机地址字节应答位之后产生第一个 DMA TX 请求。如果 master 给出 NACK，则不会生成 DMA TX 请求。如果使能 NACKIE 和 IICIE，I2C 从机将产生 CPU 中断请求。

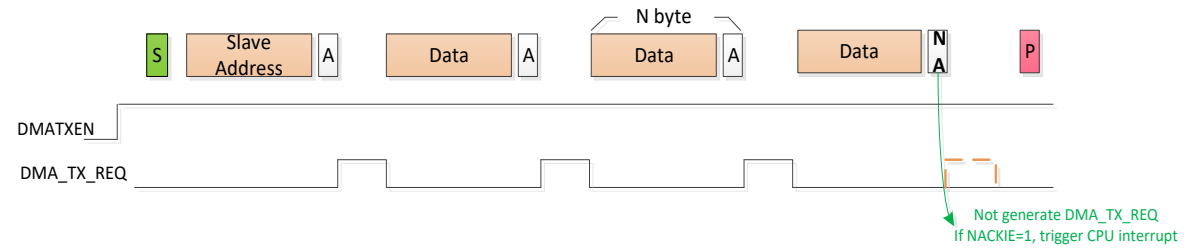

图 18-12 从机发送器

18.6.5.4 从机接收器

在从机接收器的第一个数据字节完成后生成第一个 DMA RX 请求。与主机接收器类似，eto_1 在最后一个字节数据前置位，并且在下一个数据字节回复 NACK。

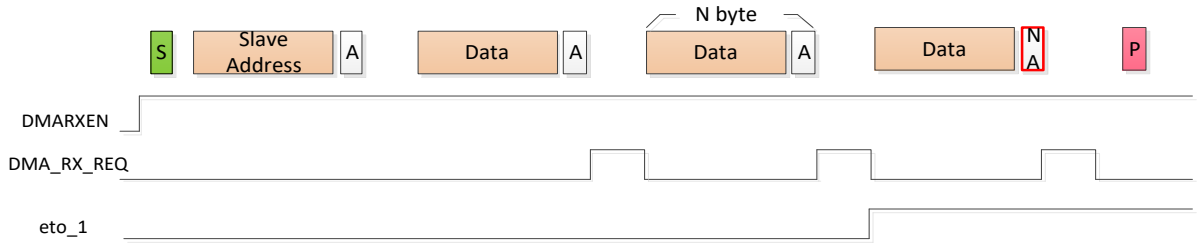

图 18-13 从机接收器

18.6.6 从机低功耗唤醒

I2C 从机可以进入低功耗模式。如果地址匹配，低功耗模式下，I2C 从机将产生唤醒信号以唤醒 MCU。在收到唤醒序列和 Stop 信号之间，I2C 从机不能发送或接收任何数据。因此，如果地址匹配，则从机仅响应（ACK）唤醒序列的地址字节，并且所有后续数据传输都是 NACK。

I2C 从机进入低功耗模式必须满足如下条件：

a. I2C 从机空闲（IDLE）；

b. *WUEN* 位为 '1'；

c. 软件 WFI 指令。

图 18-14 唤醒序列

18.6.7 中断

I2C 共有 10 个中断。

表 18-2 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 缓冲区上溢	RXOF	RXOFIE	IICIE
TX 缓冲区下溢	TXUF	TXUFIE	IICIE
RX 缓冲区满	RXFF	RXFIE	IICIE

中断源	标志	本地使能	全局使能
TX 缓冲区空	TXEF	TXEIE	IICIE
获得应答	RACK	NACKIE	IICIE

DMARXEN=1 或 DMATXEN=1, BDN, RXFF 及 TXEF 不会产生中断，即使相应的使能位使能。

18.7 编程指南

18.7.1 说明

数据传输将按照如下流程进行：

1. 假设微控制器 A 想要向微控制器 B 发送信息：
   - 微控制器 A（主机），寻址微控制器 B（从机）.
   - 微控制器 A（主机发送器），向微控制器 B（从机接收器）发送数据
   - 微控制器 A 终止传输
2. 如果微控制器 A 想要从微控制器 B 接收信息：
   - 微控制器 A（主机），寻址微控制器 B（从机）.
   - 微控制器 A（主机接收器）从微控制器 B（从机发送器器）接收数据
   - 微控制器 A 终止传输

XXX 主机发送至从机

XXX 从机发送至主机

18.7.2 主机发送器

图 18-15 主机写操作序列

① 首先设置传输速度（*SAMPLE_CNT,STEP_CNT*） 和其他功能控制寄存器（*CONTROL_1,CONTROL_2,CONTROL_3,CONTROL_4,*）， *CONTROL_1[4]* 必须为“1”，然后使能 I2C.

② 写 *START_STOP[0]* “1”，触发起始位（START）

③ *STATUS_1[5]*=”1”, *STATUS_1[3]*=”1”，然后写数据（从机地址） 至 *DATA*, DATA[0] 是方向位，这里为 “0” .

④ *STATUS_1[7]*=“1”, *STATUS_1[0]*=”1”，写数据至 *DATA*.

⑤ 如果为多字节传输，则重复步骤 ④

⑥ 写 START_STOP[1] 为“1”。

18.7.3 主机接收器

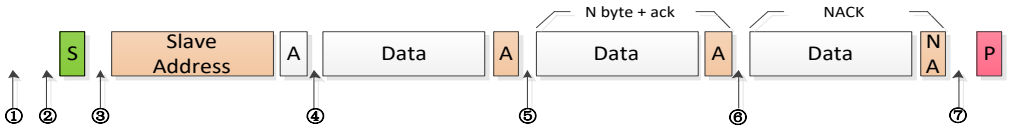

图 18-16 主机接收器操作序列

① 首先设置传输速度（*SAMPLE_CNT,STEP_CNT*） 和其他功能控制寄存器（*CONTROL_1,CONTROL_2,CONTROL_3,CONTROL_4*） ， *CONTROL_1[4]* 必须为 “1”，然后使能 I2C.

② 写 *START_STOP[0]* 为“1”，触发 START.

③ *STATUS_1[5]*=”1”, *STATUS_1[3]*=”1”，然后写数据（从机地址） 至 *DATA*， DATA[0] 是方向位， 这里为 “1” .

④ *STATUS_1[7]*=“1”, *STATUS_1[0]*=”1” , 写 *CONTROL_1[4]* “0”，然后虚拟读出 *DATA*，*触发数据传输*

⑤ *STATUS_1[7]*=“1”, 读取 DATA。如果是多字节传输，则 重复步骤 ⑤.

⑥ *STATUS_1[7]*=“1”, 写 *CONTROL_1[3]* “1”，然后读取 *DATA*.

⑦ *STATUS_1[7]*=“1”，写 CONTROL_1[4] “1”，然后读取 DATA（最后一个字节），然后写 *START_STOP[1]* 触发 STOP。

18.7.4 主机组合格式

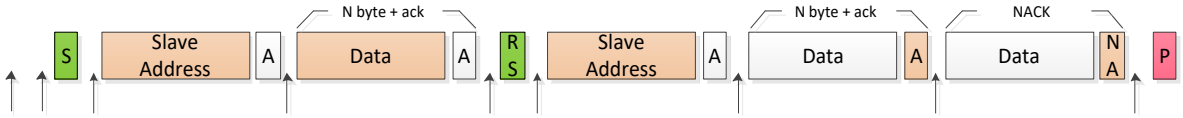

图 18-17 组合格式选项序列

组合格式可以是主机先写入从机，然后读从机，或者首先是主机读从机，然后是写从机。

主机写入从机的所有程序配置和序列，请参考 **18.7.2 主机发送器**，主机读取从机请参考 **18.7.3 主机接收器**。 唯一的区别是‘RS’（RESTART）。I2C 模块没有 RS trig 寄存器，这由 START_STOP [0]实现。

如果 I2C 总线空闲，则 START_STOP [0] 写入“1”将触发 START 信号；如果 I2C 非空闲，则将触发 RESTART 信号。

18.7.5 从机

从机地址匹配将生成 SAMF 标志，并且根据地址字节 LSB 硬件自动控制切换到接收器或发送器。SRW 指示接下来数据传输的方向。在 SAMF 和 SRW = 1（主机读取）之后，软件将数据写入 DATA 寄存器。每个字节（9 SCL）完成后，BND 设置为“1”。如果 SRW = 1，软件将数据写入 DATA 寄存器，如果 SRW = 0，则从 DATA 寄存器读取数据。主机写入时，从机发送 ACK 位。CONTROL_1[3] TACK 位控制下一个字节的 ACK 位。

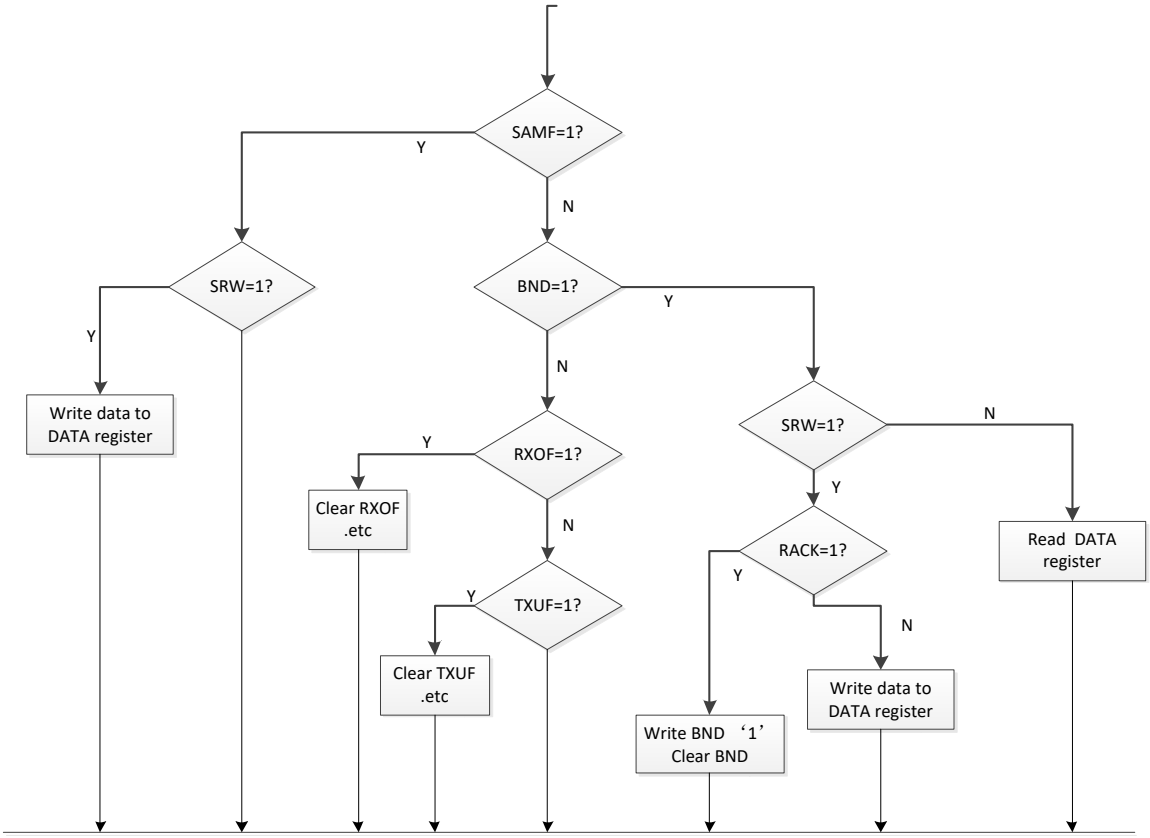

图 18-18 典型 I2C 从机中断程序

19 串行外设接口（SPI）

19.1 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 19-1 给出了 SPI 主机和 SPI 从机之间的连接示例。

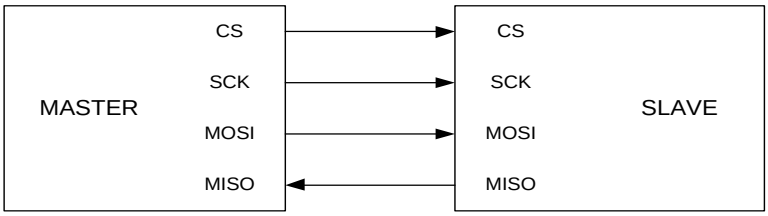

图 19-1 SPI 系统连接

19.2 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 18 MHz
- 全双工模式
- 主机可编程传送比特率
- 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

19.3 结构框图

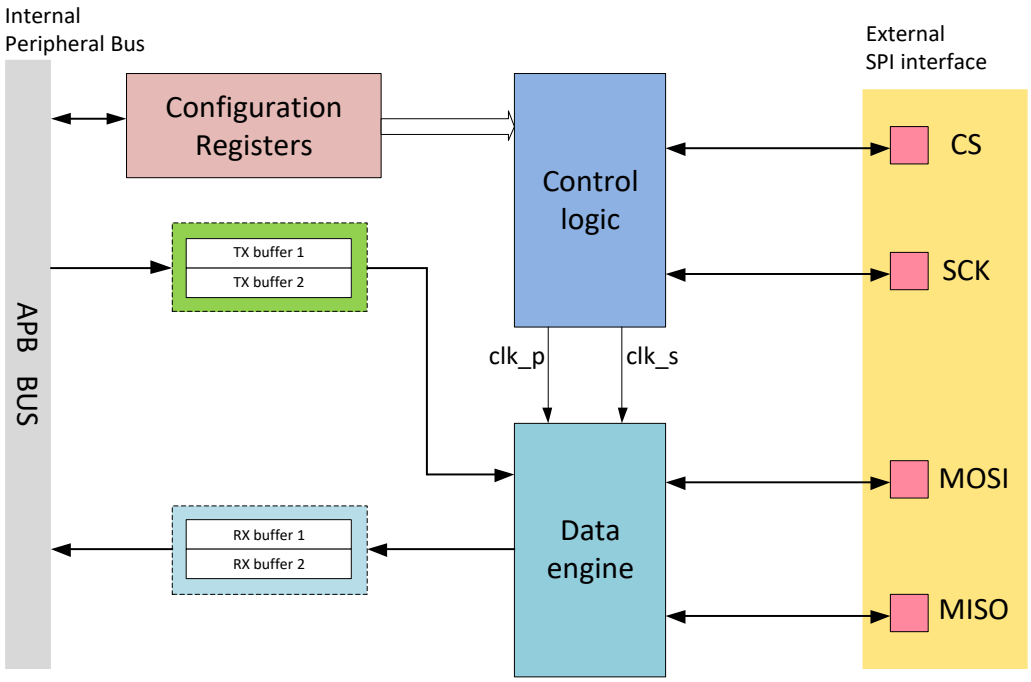

图 19-2 SPI 结构框图

19.4 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。

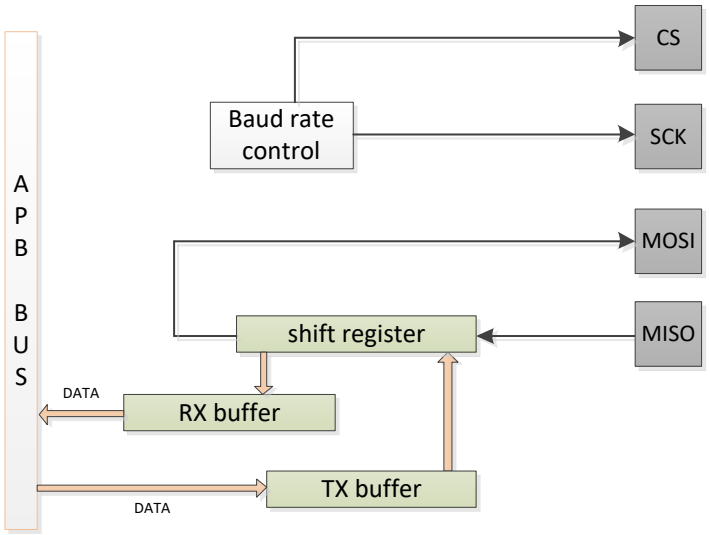

图 19-3 主机数据流

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。

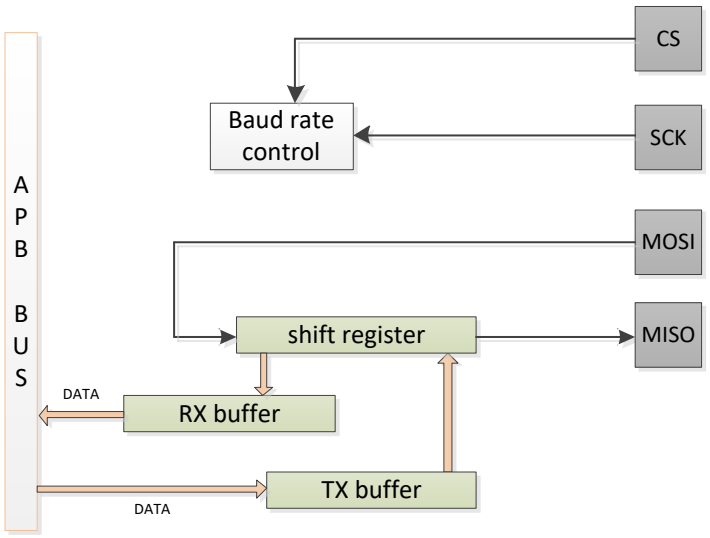

图 19-4 从机数据流

19.5 输入输出时序

19.5.1 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。

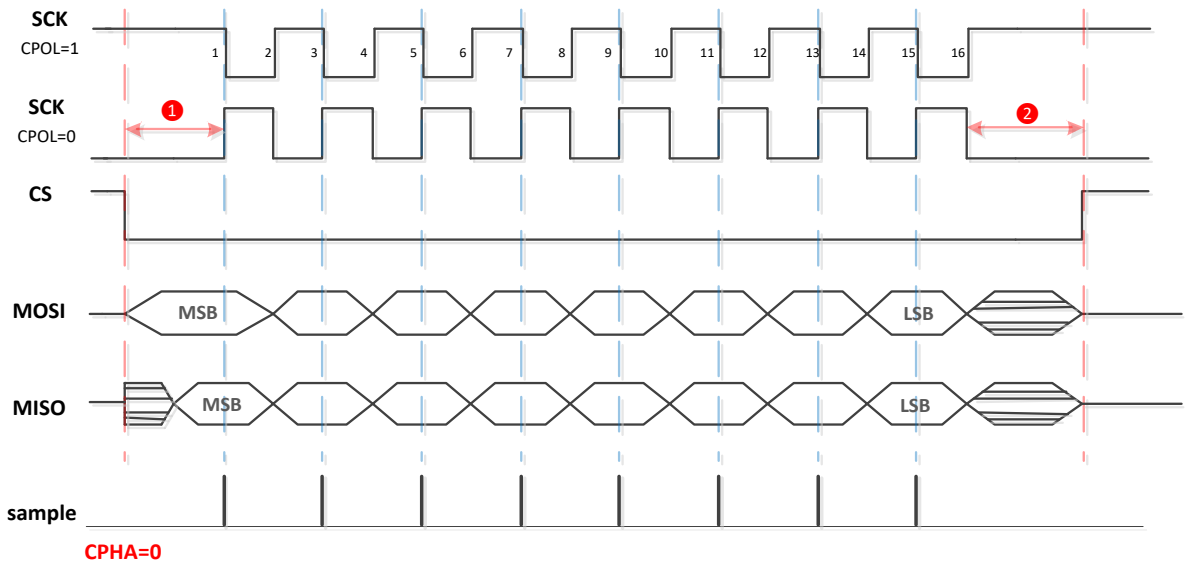

图 19-5 CPHA=0 传输格式

19.5.2 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。

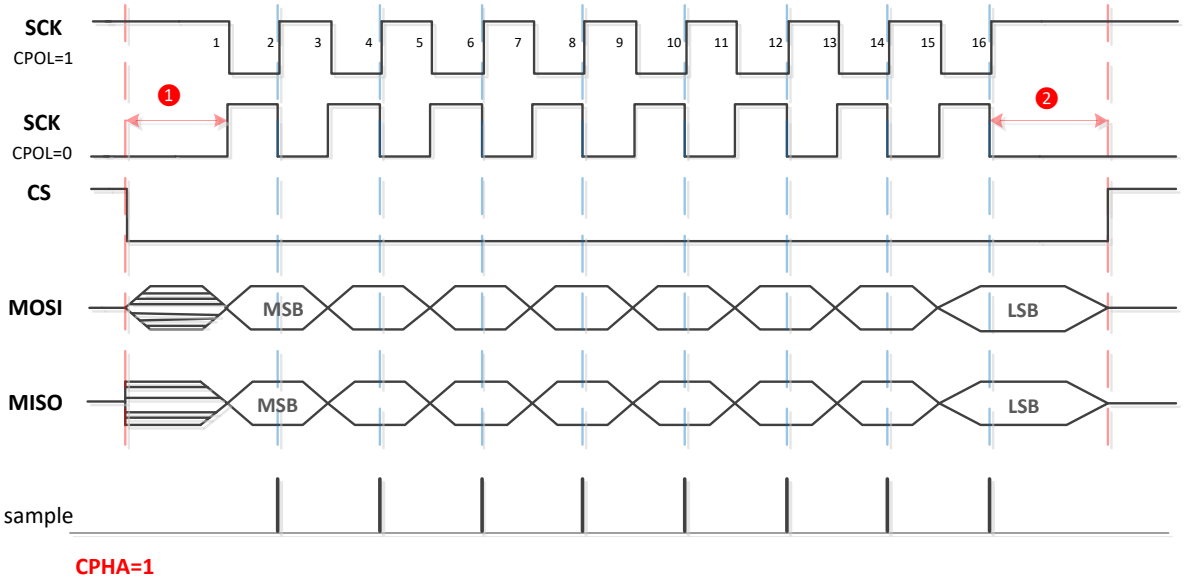

图 19-6 CPHA=1 传输格式

19.6 主机 SCK 输出时序设置

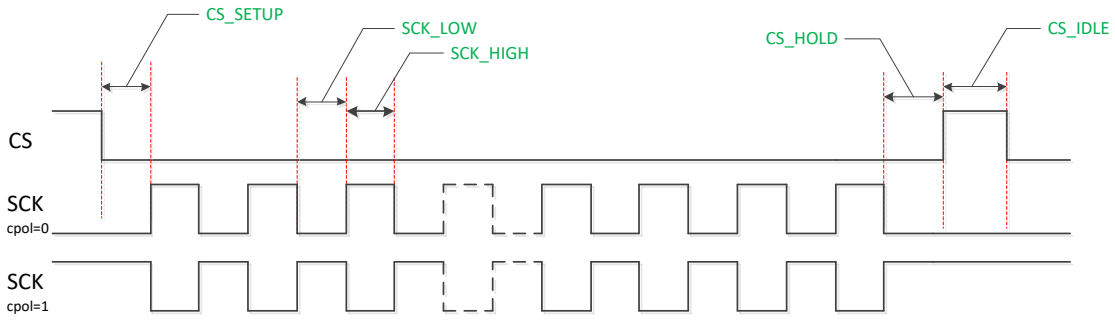

图 19-7 波特率生成

波特率： $f_{SCL}=f_{bclk}/$（SCK_LOW+1+SCK_HIGH+1）, f_{bclk} 是 APB 总线时钟频率。

19.7 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 19-8 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。

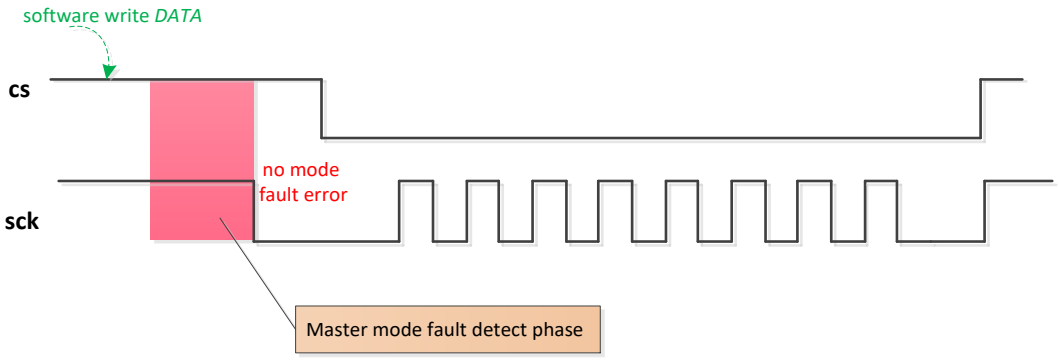

图 19-8 在模式故障检测使能时的 SCK 输出时序

在某些特殊情况下，主机 1 模式故障检测功能可能无法检测到模式故障。如果在图 19-9 中的（1）周期内主机 2 驱动 CS 为低电平，则主机 1 的 SPI 主控制器不会设置 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。

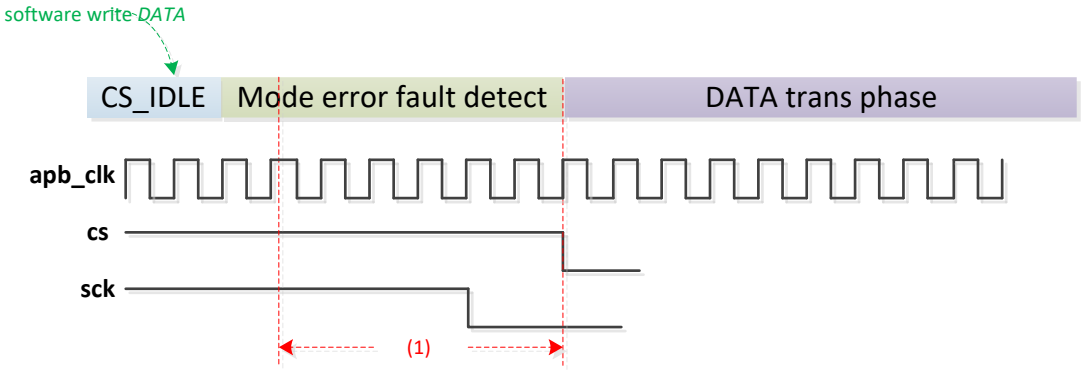

图 19-9 模式故障检测限制

19.8 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 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。

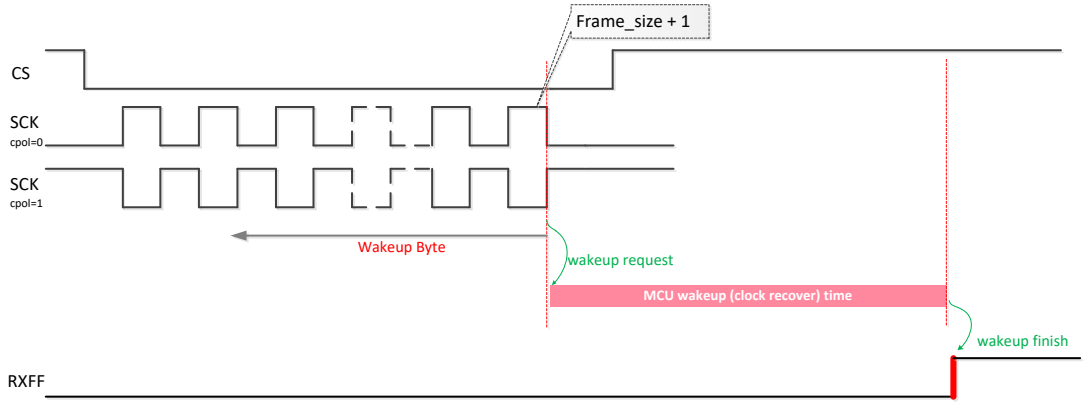

图 19-10 唤醒序列

19.9 中断

SPI 共有 5 个中断。

表 19-1 中断汇总

标志位	本地使能
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

19.10 主机 CS 连续模式

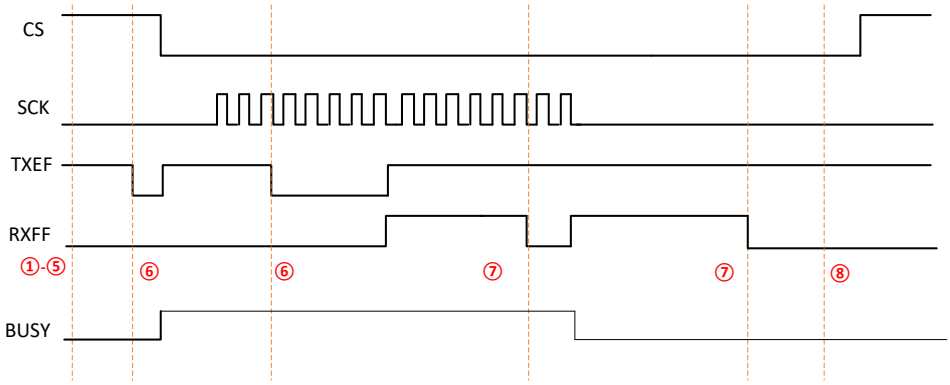

图 19-11 CS 连续模式

CS 连续输出

1. 配置 CFG0：CS_SETUP, CS_HOLD,SCK_LOW,SCK_HIGH；
2. 配置 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 中读数据；
8. 写 CSRLS “1”，释放 CS，然后硬件进入空闲状态。

19.11 主机 CS 非连续输出

1. 配置 CFG0：CS_SETUP,CS_HOLD,SCK_LOW,SCK_HIGH；
2. 配置 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。所以软件并不关注 CSRLS。

19.12 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；

19.13 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。

19.14 寄存器定义

表 19-2 SPI 寄存器映像

SPI1 基地址： 0x4000c000
SPI2 基地址： 0x4000d000

地址	名称	说明
SPIx+0x000	**CFG0**	SPI 配置寄存器 0
SPIx+0x004	**CFG1**	SPI 配置寄存器 1
SPIx+0x008	**CMD**	SPI 命令寄存器
SPIx+0x00c	**STATUS**	SPI 状态寄存器
SPIx+0x010	**DATA**	SPI 数据寄存器
SPIx+0x014	**CFG2**	SPI 配置寄存器 2

SPIx+0x000 CFG0 SPI 配置寄存器 0

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	CS_SETUP								CS_HOLD							
类型	R/W								R/W							
复位	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							
类型	R/W								R/W							
复位	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1

位	名称	说明
31：24	**CS_SETUP**	**CS 建立时间** 片选建立时间 =（CS_SETUP_COUNT+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** 片选保持时间 =（CS_HOLD_COUNT+1）*CLK_PERIOD.
15：8	**SCK_LOW**	**SCK 低电平时间** SCK 时钟低电平时间 =（SCK_LOW_COUNT+1）*CLK_PERIOD.
7：0	**SCK_HIGH**	**SCK 高电平时间** SCK 时钟高电平时间 =（SCK_HIGH_COUNT+1）*CLK_PERIOD.

SPIx+0x004 **CFG1** **SPI 配置寄存器 1**

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	TMSBF	CPHA	CPOL
类型		R/W		R/W		R/W	R/W		R/W				R/W	R/W	R/W	R/W
复位		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							
类型	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W							
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

位	名称	说明
31		
30	**WKUEN**	**从机唤醒功能使能** 1： 从机唤醒功能使能 0： 从机唤醒功能禁用
29		
28	**CONT_CS**	**CS 连续输出使能** 1： CS 输出连续 0： CS 输出不连续
27		
26	**MODFEN**	**主机模式故障检测使能** 1： 使能多主机检测功能 0： 禁用多主机检测功能
25	**CSOE**	**CS 硬件输出使能** 1： 使能 CS 硬件输出 0： 禁用 CS 硬件输出
24		
23:20	**FRMSIZE**	**帧大小** 1111： 16bit 1110： 15bit ``` 0011： 4bit 0010： 4bit 0001： 4bit 0000： 4bit
19	**RMSBF**	**RX 最高有效位优先** 1：移位器移位的第一位是输入数据的 MSB 0：移位器移位的第一位是输入数据的 LSB
18	**MSBF**	**TX 最高有效位优先** 1： TX MSB 优先（MSB 位首先移出） 0： TX LSB 优先（LSB 位首先移出）
17	**CPHA**	**时钟相位** 1： 第一个 SCK 过渡边沿为数据捕获边沿 0： 第一个 SCK 过渡边沿为数据移出边沿
16	**CPOL**	**时钟极性** 1： 空闲时，SCK 为 1

位	名称	说明
		0： 空闲时，SCK 为 0
15	**DMARXEN**	**DMA RX 请求使能** 1： 使能 DMA RX 请求 0： 禁用 DMA RX 请求
14	**DMATXEN**	**DMA TX 通道使能** 1：使能 DMA RX 请求 0：禁用 DMA RX 请求
13	**MODFIE**	**模式故障中断使能** 1： 使能 0： 禁用
12	**MSTR**	**主机或 从机模式选择** 1： 主机模式 0： 从机模式
11	**RXOIE**	**RX 缓冲区溢出中断使能** 1： 使能 0： 禁用
10	**TXUIE**	**TX 缓冲区下溢中断使能** 1： 使能 0： 禁用
9	**RXFIE**	**RX 缓冲区满中断使能** 1： 使能 0： 禁用 **注意： RX 缓冲区非空即可以产生中断**
8	**TXEIE**	**TX 缓冲区空中断使能** 1： 使能 0： 禁用 **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

SPIx+0x008　　CMD　　SPI 命令寄存器

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型																
复位																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称										**ROTRIG**	**CSRLS**	**SWRST**				**SPIEN**
类型										R/W	R/W	R/W				R/W
复位										0	0	0				0

位	名称	说明
31：6		
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** 1： 释放 CS 0： 不起作用

位	名称	说明
		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** 1： 复位 0： 不起作用 **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。**
0	**SPIEN**	**SPI 使能** 1： 使能 0： 禁用

SPIx+0x00C **STATUS** **SPI 状态寄存器**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型																
复位																
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
类型								R/W	R/W			R/W	R/W	R/W	R/W	R/W
复位								1	0			0	0	0	0	1

位	名称	说明
31：8		
8	**IDLEF**	**SPI 空闲标志** 1： SPI 模块硬件空闲 0： SPI 模块硬件非空闲 **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** 1： SPI 主机硬件一个字节传输未完成 0： SPI 主机硬件一个字节传输完成
4	**MODEF**	**模式故障检测标志** 1： 检测到主机模式故障 0： 未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。 仅当 *CSOE=1,MODEEN=1*，引脚充当多主机故障检测输入。 **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** 1： RX 缓冲区溢出 0： 没有溢出 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及时读走，下一个接收的数据过来，会产生 RX 溢出。 写 “1” 清零此位。 **注意：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** 1： TX 缓冲区下溢

		0： 没有下溢 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。**
1	**RXFF**	**RX 缓冲区满标志** 1： 满 0： 不满 **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 。** **注意：此位称为 RX 缓冲区非空更为合理。**
0	**TXFF**	**TX 缓冲区非空标志** 1： 空 0： 非空 **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 。** **注意：此位称为 TX 缓冲区非满更为合理。**

SPIx+0x010　　DATA　　SPI 数据寄存器

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型																
复位																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	**DATA**															
类型	R/W															
复位	0															

位	名称	说明
31：16		
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

SPIx+0x014　　CFG2　　SPI 配置寄存器 2

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称																
类型																
复位																
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称													**ROEN**	**TOEN**	**MNDC**	
类型													R/W	R/W	R/W	
复位													0	0	0	

位	名称	说明
31：4		
3	**ROEN**	**仅 RX 模式使能** 1：使能 0：禁用 **注意： 仅 RX 模式,，TXEF 将保持 0。即使 TXEIE=1 也不触发 RXEF IRQ。**
2	**TOEN**	**仅 TX 模式使能** 1： 使能 0： 禁用 **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** 1： 使能 0： 禁用 如果 rxbuff 为满，则写入 txbuff 不会触发新的发送操作。
0		

20 直接存储器访问（DMA）

20.1 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加速存储器传输，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 12 个专用通道用于不同类型的外围设备， 比如 UART, I2C, SPI, ADC 等。 还有一个轮询仲裁器来处理通道间的优先级。

20.2 特性

- 12 个独立的可配置通道 。
- 每个通道都直接连接至专用的硬件 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 15 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 个事件标志逻辑对应于单一的中断请求。
- 存储器和存储器间的传输。
- 外设到存储器 、存储器到外设的传输。
- 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。

20.3 结构框图

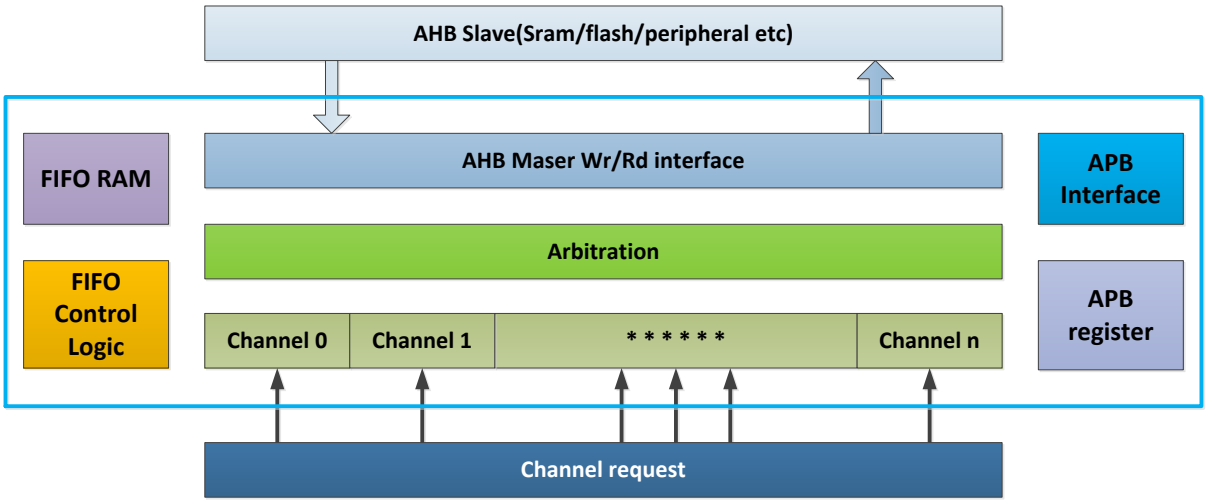

图 20-1 DMA 结构框图

20.4 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，即配置所有 DMA 通道的 CHANNELx_CHAN_ENABLE 寄存器的 CHAN_ENABLE 位都等于 0。

20.5 功能描述

20.5.1 DMA 请求列表

表 20-1 DMA 请求列表

外设	通道 1	通道 2	通道 3	通道 4	通道 5	通道 6
ADC	ADC					
SPI		SPI1_RX	SPI1_TX	SPI2_RX	SPI12_TX	
UART	UART6_RX	UART3_TX	UART3_RX	UART1_TX	UART1_RX	UART2_TX
I2C						
外设	**通道 7**	**通道 8**	**通道 9**	**通道 10**	**通道 11**	**通道 12**
ADC						
SPI						
UART	UART2_RX	UART4_TX	UART4_RX	UART5_TX	UART5_RX	UART6_TX
I2C			I2C2_TX	I2C2_RX	I2C1_TX	I2C1_RX

20.5.2 仲裁

仲裁器根据通道请求的优先级来管理信道请求，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 CHANNELx_CONFIG 寄存器中配置，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 15 优先级最低，依次类推。

20.5.3 配置指南

在用户配置 **CHANNELx_CHAN_ENABLE** 寄存器之前，应编程除 **CHANNELx_CHAN_ENABLE** 寄存器之外的所有寄存器。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。

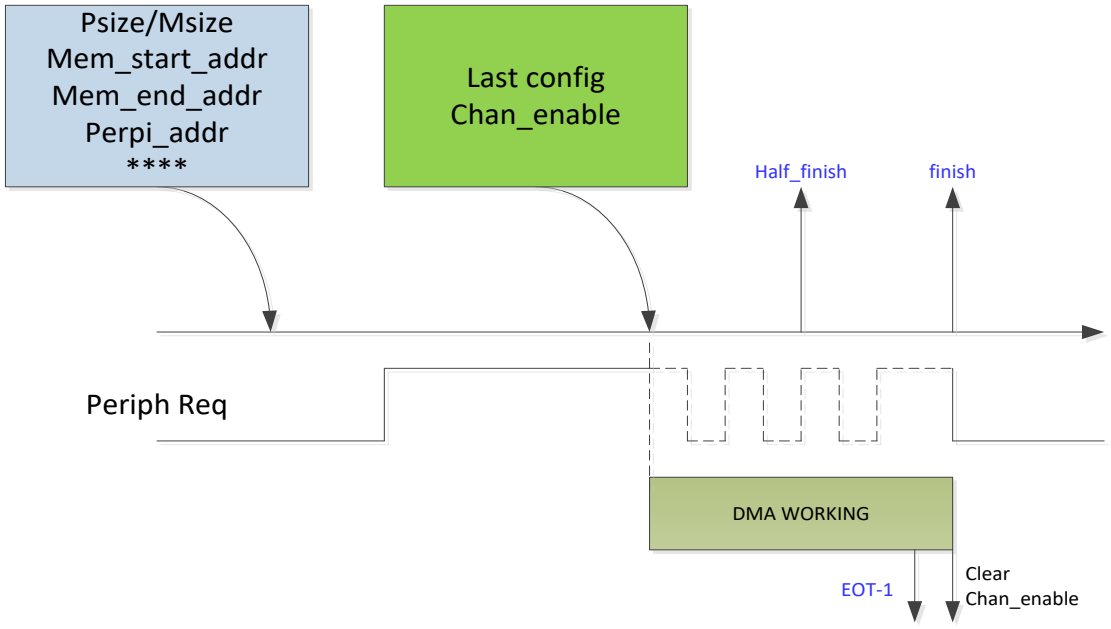

图 20-2 DMA 配置指南

20.5.3.1 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。

全局软复位的机制描述如下。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。

全局硬复位的机制描述如下。当软件想要重启所有引擎或重新清除 DMA 中的所有引擎而不等待任何时间段时，它可以将全局 HARD_RST 设置为 1 然后返回 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。

20.5.3.2 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.）；

当 DMA 运行时，软件可以多次重复步骤 2 和 3。DMA 不会立即暂停，并将等待最后一个事务完成。

20.5.3.3 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）。**CHANNELx_CONFIG 寄存器**的 CHAN_CIRCULAR 位用于使能该特性。当启动了循环模式，数据传输的数目变为 0 时，将会自动地被恢复成通道配置阶段设置的初值，DMA 请求将会继续进行。

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。

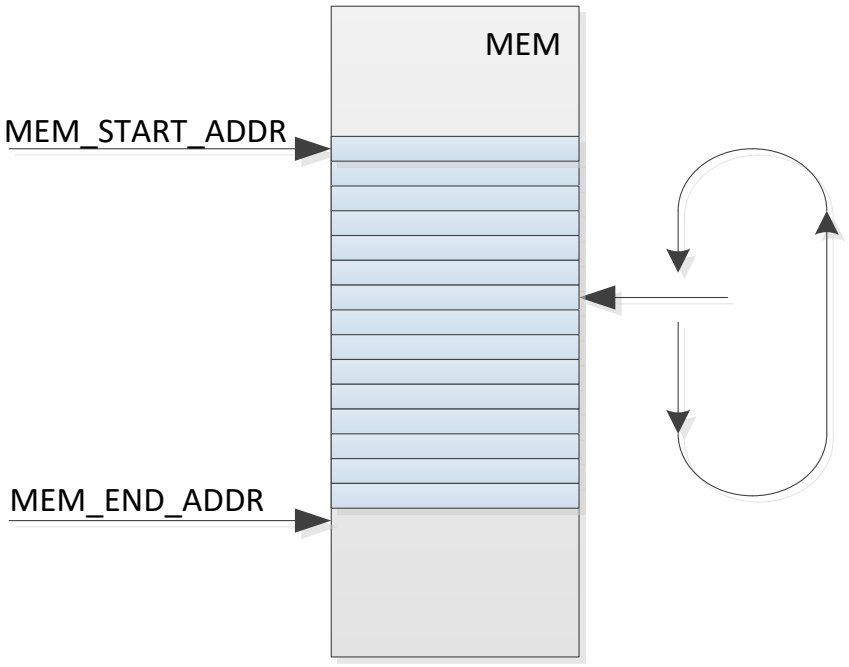

图 20-3 DMA 通道循环

20.5.3.4 I2C 使用 DMA

当达到为相应 DMA 流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 接口。I2C 硬引擎收到信号并决定是否发送 ACK/NACK 信号。

20.5.3.5 可编程数据宽度、数据对齐

表 20-2 可编程数据宽度 &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，往 0x0C 写入 0F0E0D0C。	0x00/03020100 0x04/07060504 0x08/0B0A0908 0x0C/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

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	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
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 **00000000**; 往 0x04 写入 **00000001**; 往 0x08 写入 **00000002**; 往 0x0C 写入 **00000003**;	0x00/00000000 0x04/00000001 0x08/00000002 0x0C/00000003

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
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
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;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。	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
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

注：

1、 当 dir=1（TX），表述数据从 MEM 往 PERIPH 搬运；

当 dir=0（RX），表述数据从 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=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。

20.6 寄存器定义

表 20-3 DMA 寄存器映像

DMA 基地址： 0x40012000

地址	名称	说明
DMA+0x00	**DMA_TOP_RST**	通用 DMA 复位
DMA+0x40	**CHANNELx_STATUS**	状态标志
DMA+0x44	**CHANNELx_INTEN**	中断使能
DMA+0x48	**CHANNELx_RST**	通道使能
DMA+0x4C	**CHANNELx_STOP**	DMA 通道停止
DMA+0x50	**CHANNELx_CONFIG**	DMA 通道配置
DMA+0x54	**CHANNELx_CHAN_LENGTH**	DMA 通道长度
DMA+0x58	**CHANNELx_MEM_START_ADDR**	存储器起始地址
DMA+0x5C	**CHANNELx_MEM_END_ADDR**	存储器结束地址
DMA+0x60	**CHANNELx_PERIPH_ADDR**	通道外设地址

地址	名称	说明
DMA+0x64	**CHANNELx_CHAN_ENABLE**	通道使能
DMA+0x68	**CHANNELx_DATA_TRANS_NUM**	数据传输数目
DMA+0x6C	**CHANNELx_INTER_FIFO_DATA_LEFT_NUM**	fifo 中剩余的数据

DMA_TOP_RST 寄存器 **偏移地址 = 12'h00**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位	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
读	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**HARD_RST**	**WARM_RST**
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	名称	说明
1	HARD_RST	**通用 DMA 硬复位（不论当前事务如何，都复位）** 软件设置'HARD_RST'为 1，然后设置 ' HARD_RST ' 回 0，则该复位机制完成。 0： 禁用 1： 使能
0	WARM_RST	**通用 DMA 软复位（在当前事务完成后复位）** 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。 0： 禁用 1： 使能

CHANNELx_STATUS 寄存器 **偏移地址 = 12'h40**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位	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
读	0													**TRANS_ERROR**	**HALF_FINISH**	**FINISH**
写														W0C	W0C	W0C

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

位	名称	说明
2	TRANS_ERROR	**传输错误标志** 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 0： 错误没有发生 1： 错误发生
1	HALF_FINISH	**半完成标志** 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： 数据传输未完成一半 1： 数据传输完成一半
0	FINISH	**完成标志** 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。 0： 数据传输未完成 1： 数据传输完成

CHANNELx_INTEN 寄存器 **偏移地址 = 12’h44**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位	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
读	0													**TRANS_ERROR interrupt enable**	**HALF_FINISH interrupt enable**	**FINISH interrupt enable**
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	名称	说明
2	TRANS_ERROR interrupt enable	**TRANS_ERROR 中断使能** 此位由软件置位和清零 0： 中断禁用 1： 中断使能
1	HALF_FINISH	**HALF_FIISH 中断使能**

位	名称	说明
	interrupt enable	此位由软件置位和清零 0： 中断禁用 1： 中断使能
0	FINISH interrupt enable	**FINISH 中断使能** 此位由软件置位和清零 0： 中断禁用 1： 中断使能

CHANNELx_RST 寄存器 **偏移地址 = 12'h48**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位	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
读	0	0	0	0	0	0	0	0	0	0	0	0	0	**FLUSH**	**HARD_RST**	**WARM_RST**
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	名称	说明
2	FLUSH	**DMA 通道刷新** 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. 0： 禁用 1： 使能
1	HARD_RST	**DMA 通道硬复位（不论当前事务如何，都复位）** 软件设置 HARD_RST 为 1，然后将 HARD_RST'设置回 0， 软件机制完成 0： 禁用 1： 使能
0	WARM_RST	**DMA 通道软复位（在当前事务后复位）** 软件设置 WARM_RST 为 1，然后将 WARM_RST 设置回 0， 软件机制完成 0： 禁用 1： 使能

CHANNELx_STOP 寄存器 **偏移地址 = 12'h4C**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位	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
读	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	STOP
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	名称	说明
0	STOP	**DMA 通道暂停（在当前事务后暂停）** 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 0： 没有停止通道传输 1： 暂停通道传输

CHANNELx_CONFIG 寄存器 **偏移地址 = 12'h50**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
写	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
复位	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
读	0	0	0	MEM_BYTE_MODE		CH_DIR	CHAN_CIRCULAR	PERIPH_INCREMENT	MEM_INCREMENT	PERIPH_SIZE		MEM_SIZE		CHAN_PRIORITY		MEM2MEM
写	0	0	0													
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	名称	说明
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

位	名称	说明
9	CHAN_CIRCULAR	0： 循环模式禁用 1： 循环模式使能
8	PERIPH_INCREMENT	0： 外设地址固定 1： 外设地址增加
7	MEM_INCREMENT	0： MEM 地址固定 1： MEM 地址增加
6：5	PERIPH _SIZE	00： 8 位 01：16 位 10：32 位 11： 保留
4：3	MEM _SIZE	00： 8 位 01：16 位 10：32 位 11： 保留
2：1	CHAN_PRIORITY	00： 低 01： 中等 10： 高 11： 很高
0	MEM2MEM	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

CHANNELx_CHAN_LENGTH 寄存器 **偏移地址 = 12'h54**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位	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
读 写	**CHAN_LENGTH**															
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	名称	说明
15：0	CHAN_LENGTH	**DMA 通道传送长度** 0~32767 第 15 位应该为 0

CHANNELx_MEM_START_ADDR 寄存器 **偏移地址 = 12'h58**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读写	MEM_START_ADDR[31：16]															
复位	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
读写	MEM_START_ADDR [15：0]															
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	名称	说明
31：0	MEM_START_ADDR	MEM_START_ADDR Memory 起始地址

CHANNELx_MEM_END_ADDR 寄存器 **偏移地址 = 12'h5C**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读写	MEM_END_ADDR[31：16]															
复位	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
读写	MEM_END_ADDR [15：0]															
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	名称	说明
31：0	MEM_END_ADDR	MEM_END_ADDR 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。

CHANNELx_PERIPH_ADDR 寄存器 **偏移地址 = 12'h60**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读写	PERIPH_ADDR[31：16]															
复位	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

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读 写	**PERIPH_ADDR[15：0]**															
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	名称	说明
31：0	PERIPH_ADDR	PERIPH_ADDR

CHANNELx_CHAN_ENABLE 寄存器 **偏移地址 = 12'h64**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位	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
读	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**CHAN_ENABLE**
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	名称	说明
0	CHAN_ENABLE	**通道使能** 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。 0： 禁用通道 1： 使能通道

CHANNELx_DATA_TRANS_NUM 寄存器 **偏移地址 = 12'h68**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位	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
读	**DATA_TRANS_NUM**															
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	名称	说明

位	名称	说明
15：0	DATA_TRANS_NUM	**DATA_TRANS_NUM** 标识已传输了多少数据

CHANNELx_INTER_FIFO_DATA_LEFT_NUM 寄存器　　偏移地址 = 12'h6C

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位	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
读	0										**INTER_FIFO_DATA_LEFT_NUM**					
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	名称	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。

21 看门狗定时器（WDOG）

21.1 简介

看门狗定时器（WDOG）模块是一个可供系统使用的独立定时器。可确保软件按计划执行，且 CPU 不会陷入死循环中或执行错误的代码。若 WDOG 模块在规定时间内未得到刷新， 它会复位 MCU。

21.2 特性

- 独立于总线时钟的可配置时钟源输入；
  - 内部 32kHz RC 振荡器
  - 内部 8MHz RC 振荡器
  - 外部 XOSC 时钟源
- 可编程超时周期；
  - 可编程的 32 位超时值
  - 需要更长超时周期时，可选的固定 256 倍时钟预分频器
- 可实现计数器刷新的鲁棒性写入序列；
  - 刷新序列为，先写 0x02A602A6，然后写入 0x80B480B4。
- 可选的窗口模式刷新机制；
  - 可编程 32 位窗口值
  - 可以通过鲁棒性检查判断执行刷新序列的时间是否早于预期
  - 试图提前刷新会触发复位。
- 允许后期诊断处理的可选超时中断；
  - CPU 中断请求，产生相应的中断向量，并可执行相应的中断服务例程（ISR）
  - 在中断向量产生 128 个总线时钟后，强制复位。
- 看门狗配置只允许在复位后写入一次，确保无法误更改看门狗的配置；
- 用于解锁只能写入一次的配置位的鲁棒性写入序列。
  - 先写入 0x20C520C5 后写入 0x28D928D9 的解锁序列，允许更新只能写入一次的配置位
  - 软件必须在解锁后以及 WDOG 关闭解锁窗口前的 128 个总线时钟内执行更新操作。

21.3 结构框图

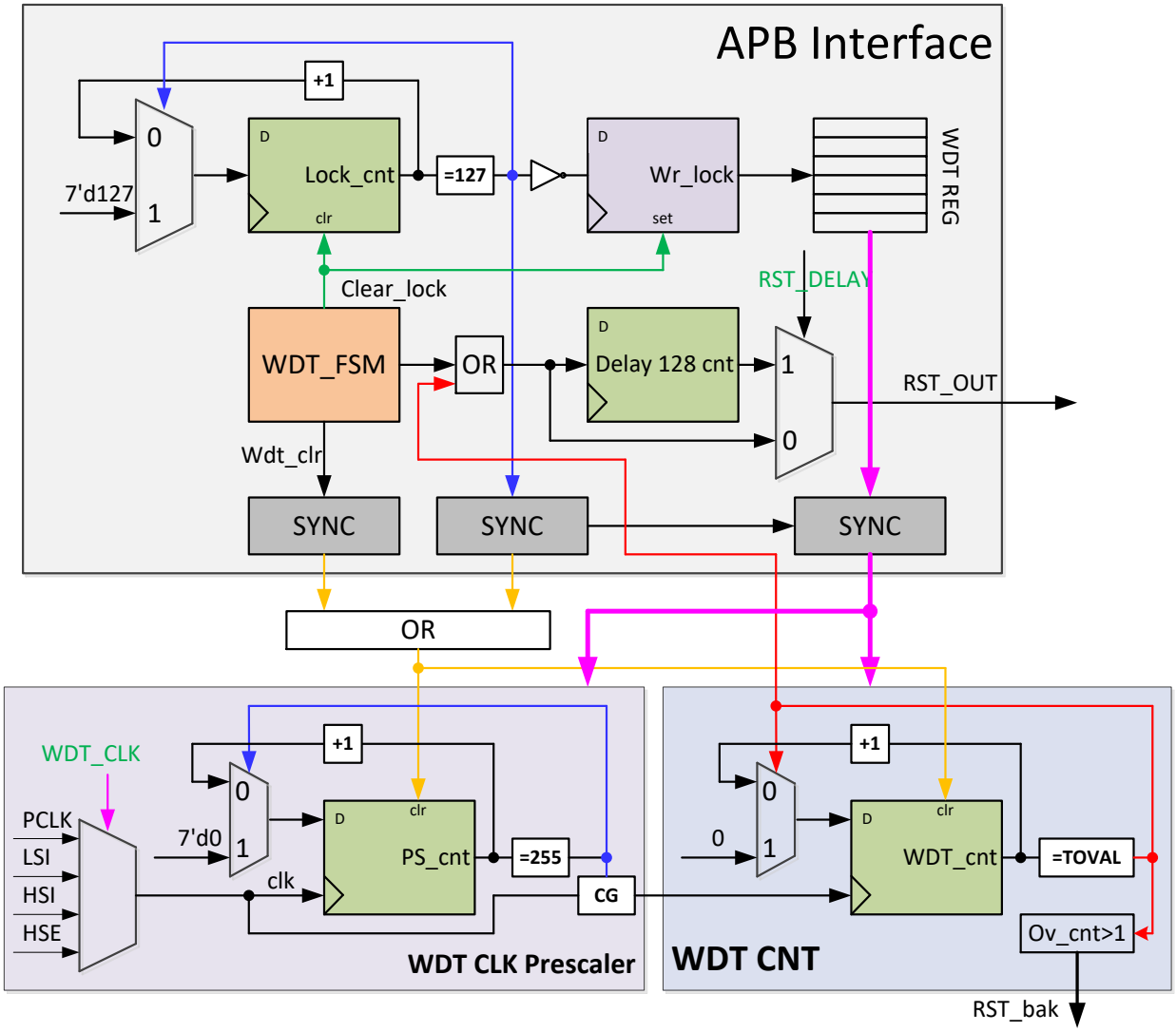

图 21-1 WDG 结构框图

21.4 操作模式

即使微控制器已进入停止（Stop）模式，WDOG 也可以使用内部 32 kHz RC 振荡器运行。因此，如果在 STOP 模式下没有为 WDOG 供电，用户应确保禁用 WDOG，以避免意外的系统复位。

21.5 寄存器定义

表 21-1 WDOG 寄存器映像

WDOG 基地址： 0x4000b000

地址	名称	说明
WDOG+0x00	WDT_CS1	看门狗状态

地址	名称	说明
WDOG+0x04	WDT_CS2	看门狗状态
WDOG+0x08	WDT_CNT	看门狗计时器
WDOG+0x0C	WDT_TOVAL	看门狗超时值
WDOG+0x10	WDT_WIN	看门狗窗口

WDT_CS1 寄存器 **偏移地址 = 12'h00**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位	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
读	0								EN	INT	UPDATE	0			0	
写																
复位	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

位	名称	说明
7	EN	**看门狗使能** 该一次写入位使能看门狗计数器以开始计数。 如果选项字节被关闭，则使能位默认为 0；如果选项字节打开，则使能位默认为 1。详细内容，参考表 23-5。 0 ：看门狗禁用. 1： 看门狗使能
6	INT	**看门狗中断** 该一次写入位将看门狗配置为在发生复位触发事件（超时或看门狗非法写入）时，首先生成一个中断请求，然后再强制进行复位。在中断响应后，延迟 128 个总线时钟后发生强制复位。 0：看门狗中断禁用，看门狗复位无延迟。 1：看门狗中断使能。看门狗复位有被 128 个总线时钟延迟。
5	UPDATE	**允许更新** 该一次写入位使系统无需复位即能重新配置看门狗。 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。

WDT_CS2 寄存器 **偏移地址 = 12'h04**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位	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
读	0								WIN	FLG	0	PRES	0		CLK	
写										w1c						
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	名称	说明
7	WIN	**看门狗窗口** 该一次写入位用来使能窗口模式 0： 窗口模式禁用 1 ： 窗口模式使能
6	FLG	**看门狗中断标志** 当 INT 在控制和状态寄存器 1 中置位后，该位可表示中断的产生。写入 1 将其清零。 0： 未发生中断 1 ： 发生一次中断
4	PRES	**看门狗预分频器** 一次写入位用来使能看门狗计数器参考时钟的 256 倍预分频器。（功能框图显示了该时钟分频选择） 0 ： 256 预分频器禁用 1 ： 256 预分频器使能
1：0	CLK	**看门狗时钟** 一次写入字段表示看门狗计数器的时钟源。 00： 总线时钟 01 ： 32 kHz 内部低功耗振荡器（LSI）. 10 ： 8 MHz 内部振荡器（HSI）. 11： 外部时钟源（XOSC）.

WDT_CNT 寄存器 **偏移地址 = 12'h08**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	**WDT_CNT[31：16]**															
写																
复位	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
读	**WDT_CNT [15：0]**															
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	名称	说明
31：0	WDT_CNT [31：0]	**看门狗计时器寄存器** 看门狗计数器寄存器提供对看门狗计数器数值的访问。软件可在任意时刻对计数器寄存器进行读操作。软件无法直接写入看门狗计数器。然而，针对这个寄存器的两个写入序列具有特殊功能：刷新序列和解锁序列。

WDT_TOVAL 寄存器 **偏移地址 = 12'h0C**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读 写	**WDT_TOVAL[31：16]**															
复位	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读 写	**WDT_TOVAL [15：0]**															
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	名称	说明
31：0	WDT_TOVAL [31：0]	**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，则看门狗会强制进行复位。 计数器长度等于WDT_TOVAL+1 默认值为0x3f0000

WDT_WIN 寄存器 **偏移地址 = 12'h10**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读 写	WDT_WIN[31：16]															
复位	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
读 写	WDT_WIN [15：0]															
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	名称	说明
		看门狗窗口寄存器
31：0	WDT_WIN [31：0]	看门狗窗口寄存器：窗口模式使能后（WDOG_CS2[WIN]置位），WDOG_WIN 决定刷新序列被视为有效的最早时间。

21.6 功能描述

21.6.1 看门狗刷新机制

若看门狗计数器未得到刷新，则看门狗会复位 MCU。鲁棒性刷新机制使跑飞的代码不太可能刷新看门狗。要刷新看门狗计数器，软件必须在超时周期结束前执行刷新写入序列。此外，如果使用了窗口模式，那么只有在 WDOG_WIN 寄存器的时间值设置之后，软件才能开始执行刷新序列。

窗口模式使能时，必须在计数器达到最小期望时间值后刷新看门狗。否则，看门狗会复位 MCU。最小期望时间值在 WDOG_WIN 寄存器中指定。置位 CS2[WIN] 会使能窗口模式。

刷新写入序列为：先将 0x02A602A6 写入 WDOG_CNT 寄存器，然后再将 0x80B480B4 写入。写入 0x80B480B4 的操作必须在写入 0x02A602A6 后执行；否则，看门狗会复位 MCU。

注意： timeout =（PRES *（WDT_CNT+1））/ CLK

21.6.2 配置看门狗

如果 CS1[UPDATE]为 0，则所有看门狗控制位、超时值和窗口值都只能在系统或者看门狗模块复位后一次写入。这表示，除非系统复位或模块复位，否则它们的内容在写操作后无法更改。这就为配置看门狗提供了可靠的机制，确保软件跑飞情况无法误禁用或误修改已配置过的看门狗。

用户首先配置窗口和超时值，然后再配置其他控制位并确保 CS1[UPDATE]也置 0，从而实现上述功能。该新配置仅在复位后一次写入除 WDOG_CNT 外的所有寄存器后生效。否则，WDOG 默认使用复位值。如果未使用窗口模式（CS2[WIN]为 0），那么不需要将数值写入 WDOG_WIN 来使新配置生效。

某些情况下（比如支持 bootloader 功能时），用户可能想要在不先强制复位的情况下 重新配置或禁用看门狗。这时可以通过复位后进行看门狗初始配置时将 CS1[UPDATE]置 1，从而使用户可采用执行解锁序列的方法，在任意时刻重新配置看门狗。（相反，如果 CS1[UPDATE]保持 0，重新配置看门狗的唯一方法是发起复位。）解锁序列与刷新序列类似，但使用不同的数值。

解锁序列是指在配置完看门狗之后，先对 WDOG_CNT 寄存器写入 0x20C520C5，然后再写入 0x28D928D9。完成解锁序列后，用户必须在 128 个总线时钟内重新配置看门狗。

21.6.3 使用中断延迟复位

中断使能（**WDT_CS1**[INT] = 1）时，看门狗在发生复位触发事件（如计数器超时或无效刷新尝试）时首先生成中断请求。看门狗将强制复位信号延迟 128 个总线时钟执行，从而允许中断服务例程（ISR）执行任务，如分析堆栈以调试代码。中断禁用（**WDT_CS1**[INT] = 0）时，看门狗不会延迟强制复位信号。

21.6.4 备用复位

必须用 LSI 的时钟源作计数器的参考时钟，否则，备用复位功能不可用。备用复位功能是一种安全保护特性，在 WDOG 主逻辑丢失其时钟（总线时钟）而无法再监控计数器时，独立生成复位。如果看门狗计数器连续两次溢出（没有强制复位），那么备用复位功能生效并生成复位。

22 实时计数器（RTC）/备份寄存器（BKP）

22.1 简介

实时计数器（RTC）由一个 32 位计数器、一个 32 位比较器、若干个基于二进制和基于十进制的预分频器、四个时钟源、一个可编程周期性中断和一个可编程外部脉冲输出组成。此模块可用于计时、日历或任何任务调度功能。它还能充当循环唤醒，将器件从停止（Stop）模式和等待（Standby）模式中唤醒且无需外部组件。

22.2 特性

RTC 模块特性包括：

- 32 位向上计数器；
  - 32 位模数寄存器
  - 软件可控制的周期性定时中断
- 可编程 20 位预分频器，可由软件选择时钟源；
  - 总线时钟
  - LSI（32 kHz）
  - 外部 XOSC 时钟
  - 内部 HSI 时钟（8 MHz）
- 备用寄存器入侵检测；
  - 1 个边沿触发的入侵事件检测
  - 2 个备用寄存器（8 个字节）

22.3 结构框图

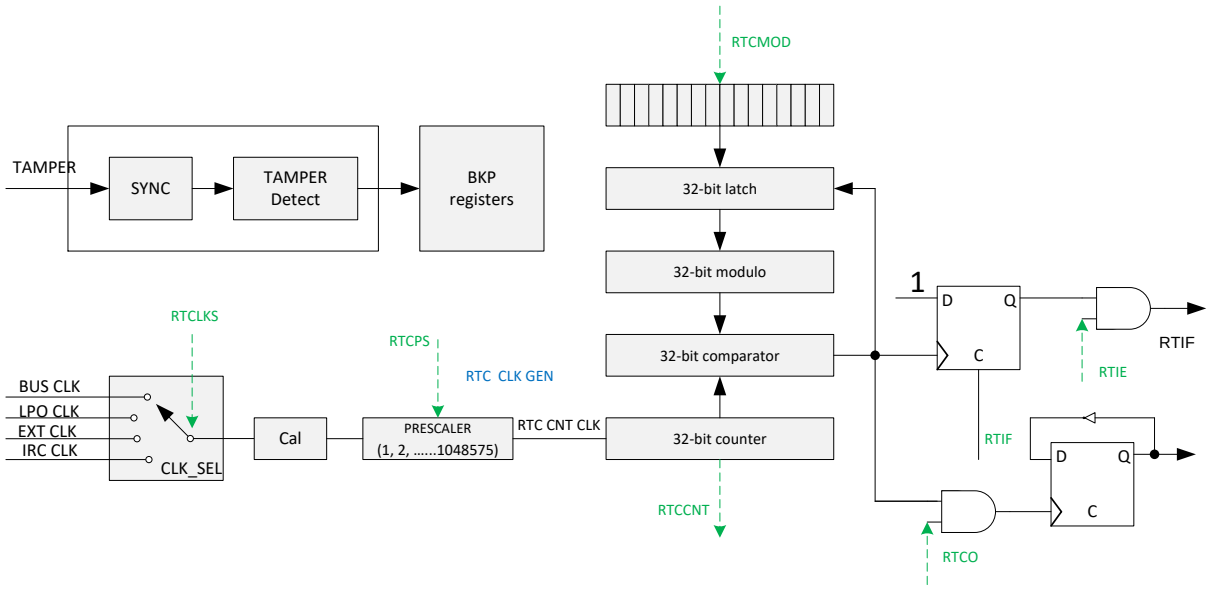

图 22-1 RTC 结构框图

22.4 操作模式

使用 LSI（32kHz）作为时钟源时，即使微控制器进入停止（Stop）模式，RTC 也会运行。如果使能 RTC 中断，当 RTC 达到定时时间时，它将触发中断并从停止（Stop）模式唤醒 MCU。

22.5 寄存器定义

表 22-1 RTC 寄存器映像

RTC 基地址： 0x40008400

地址	名称	描述
RTC+0x00	**RTC_SC**	RTC 状态
RTC+0x04	**RTC_MOD**	RTC 模数
RTC+0x08	**RTC_Counter**	RTC 计数器
RTC+0x0C	**RTC Clock Prescaler**	RTC 时钟预分频器
RTC+0x10	**RTC_Prescaler Counter**	RTC 预分频器计数器
RTC+0x20	**BKP_CR**	TAMPER 引脚有效电平/使能
RTC+0x24	**BKP_CSR**	TAMPER 中断
RTC+0x28,0x2c	**BKP_DRx**	备用数据

RTC_SC 寄存器 **偏移地址 = 12'h00**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0												Pclk_eclk_sel	CLK_CHG_OK	RPIF	RPIE
写																
复位	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
读	RTCLKS		0		0				RTIF	RTIE	0	RTCO	0			
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	名称	说明
18	**CLK_CHG_OK**	**CLK_CHG_OK： 时钟变化完成标志** 该状态位指示 RTC 计数器时钟变化状态并且只读 0： RTC 计数器时钟变化成功 1： RTC 计数器时钟正在变化
17	**RPIF**	**RPIF： 实时预分频器中断标志** 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。 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
16	**RPIE**	**RPIE：实时预分频器中断使能** 该可读,可写位用来使能实时预分频器中断。如果 RPIE 置位，则当 RPIF 置位时产生中断 。复位清零 RPIE 位。 0：禁用实时预分频器中断请求。使用软件轮询 1：使能实时预分频器中断请求
15：14	**RTCLKS**	**RTCLKS： 实时时钟源选择** 该可读,可写字段用来选择输入到 RTC 预分频器的时钟源。切换时钟源可以清零预分频器 RTCCNT 寄存器。复位清除 RTCLKS 为 00。Freq = RTCLKS /（（MOD +1） *（RTCPS+1）） 00： 总线时钟 01： 内部 32 kHz 振荡器（LPOCLK）. 10： 外部振荡器（XOSC）. 11： 内部 8 MHz 振荡器（HSI）.
7	**RTIF**	**RTIF： 实时中断标志** 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

位	名称	说明
		效。写入逻辑 1 会清除该位和实时中断请求。重置将 RTIF 清除为 0。 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
6	**RTIE**	**RTIE：实时中断使能** 该可读可、写位用来使能实时中断。如果 RTIE 置位，则在 RTIF 置位时会产生中断。 复位会将 RTIE 清除为 0。 0： 禁用实时中断请求，使用软件轮询 1：使能实时中断请求
4	**RTCO**	**RTCO： 实时计数器输出** 该可读可、写位用来使能在引脚上输出电平翻转信号。如果该位置 1，RTC 计数器溢出时将 RTCO 引脚电平翻转 0： 实时计数器输出禁用. 1：实时计数器输出使能

RTC_MOD 寄存器 **偏移地址 = 12'h04**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读 写	**MOD[31：16]**															
复位	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
读 写	**MOD[15：0]**															
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	名称	说明
31：0	**MOD[31：0]**	**RTC 模数** 该可读/写字段是用于与当前计数值（RTC_COUNTER）进行比较匹配的模数值，计数值等于模数时将计数重置为 0x0,设置 SC[RTIF]状态字段。在正常使用中，MOD 不应等于 0x0，Reset 将模数设置为 0x0。

RTC_Counter 寄存器 **偏移地址 = 12'h08**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	**CNT[31：16]**															
写																
复位	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
读	**CNT[15：0]**															
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	名称	说明
31：0	**CNT[31：0]**	**RTC 计数** 该只读字段包含 32 位计数器的当前值。 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。

RTC 时钟预分频器 **偏移地址 = 12'h0C**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0												**RTCPS[19：16]**			
写																
复位	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
读 写	**RTCPS[15：0]**															
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	名称	说明
19：0	**RTCPS**	**RTCPS： 实时时钟预分频器选择** 该可读/写字段为时钟源选择基于二进制或基于十进制的除数值。 更改预分频器值会清除预分频器和 RTCCNT 计数器。复位将 RTCPS 清除为 0000。 当 RTCPS 等于 0000 时, RTC 计数器关闭。

RTC_Prescaler 计数寄存器 **偏移地址 = 12'h10**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读													PSCNT[19：16]			
写																
复位	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
读	PSCNT[15：0]															
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	名称	说明
19：0	**PSCNT[31：0]**	**RTC PSCount** 该只读字段包含20位计数器的当前值。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。

BKP_CR 寄存器 **偏移地址 = 12'h20**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位	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
读	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TPAL	TPE
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	名称	说明
1	**TPAL**	**TPAL： TAMPER 引脚有效电平** 0： TAMPER 引脚上出现高电平会复位所有数据备份寄存器（如果 TPE 位置位）。 1： TAMPER 引脚上出现低电平会复位所有数据备份寄存器（如果 TPE 位置位）。
0	**TPE**	**TPE： TAMPER 引脚使能** 0： TAMPER 引脚可用于 RTCO 功能 1： Tamper I/O 功能被激活 **注：需配置引脚 PC15 为 RTC_TAMPER 功能。**

BKP_CSR 寄存器 **偏移地址 = 12'h24**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读 写	RESERVED															
复位	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
读	RESERVED						TIF	TEF	RESERVED					TPIE	0	0
写															CTI	CTE
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	名称	说明
9	**TIF**	**TIF： Tamper 中断标志** 当检测到入侵事件且 TPIE 位置 1 时，该位由硬件置 1。 通过将 1 写入 CTI 位清除（也会清除中断）。 如果 TPIE 位复位，它也会被清零。 0： 没有入侵中断 1： 发生入侵中断
8	**TEF**	**TEF： Tamper 事件标志** 当检测到入侵事件时，该位由硬件置位。通过将 1 写入 CTE 位来清除它。 0： 没有入侵事件 1： 发生入侵事件
2	**TPIE**	**TPIE： TAMPER 引脚中断使能** 0： 禁用入侵中断 1： 入侵中断使能（还必须在 BKP_CR 寄存器中设置 TPE 位）
1	**CTI**	**CTI： 清除入侵中断** 该位只写，读始终为 0. 0： 不起作用 1： 清除入侵中断和 TIF 入侵中断标志
0	**CTE**	**CTE： 清除入侵事件** 该位只写，读始终为 0. 0： 无效 1： 复位 TEF Tamper 事件标志（和 the Tamper 检测器）

BKP_DRx 寄存器 **偏移地址 = 12'h28, 12'h2C**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读 写	BKP_DATA[31：16]															
复位	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
读 写	BKP_DATA[15：0]															
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	名称	说明
31：0	BKP_DATA[31：0]	**备用数据** 这些位可以由用户数据写入。 **注意： 系统复位或设备从待机模式唤醒时，BKP_DRx 寄存器不会复位。它们通过备份域复位或 TAMPER 引脚事件（如果激活 TAMPER 引脚功能）复位。**

22.6 功能描述

22.6.1 低功耗模式操作

如果在执行 WFI 指令且时钟源为 LSI 时钟之前使能 RTC，则 RTC 继续以低功耗模式运行。

因此，RTC 模块也可以在停止（Stop）模式和待机（Standby）模式下工作。因此可以使用 RTC 唤醒 MCU 退出停止（Stop）模式或待机（Standby）且无需其他外部元件。

22.6.2 备份寄存器

备份寄存器（BKP）是两个 32 位寄存器，用于存储 8 字节的用户应用程序数据。 它们在 VDD 域中实现。系统复位或设备从待机（Standby）模式唤醒时，它们不会复位。它们会被上电复位（POR）复位。

入侵检测事件会复位所有备份寄存器（BKP）。通过将 BKP_CSR 寄存器中的 TPIE 位置 1，可在发生入侵检测事件时产生中断。

22.6.3 配置序列

所有寄存器应在 RTC_CLK 和 RTC_RTPTS 之前编程。当 RTC 模块正常工作时，RTC_CLK 和 RTC_RTPTS 的任何更改都将清除 RTC 计数器。

23 片内 Flash（Embedded Flash）

23.1 片内 Flash 功能概述

23.1.1 简介

片内 Flash 控制器是 Cortex™-M3 和片内 Flash 之间的桥梁，在实际应该中，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。

23.1.2 特性列表

- eflash 存储器：
  - 256K 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：读
    - 验证：页擦除验证，整片擦除验证
  - 包含预取缓冲区。

23.1.3 结构框图

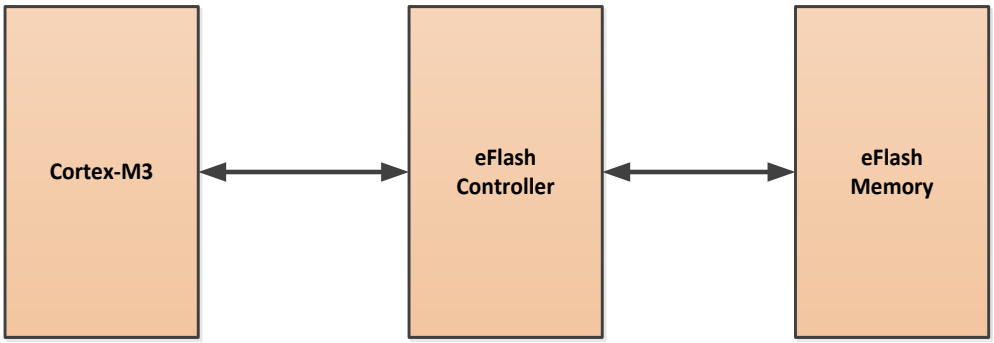

图 23-1 eflash 和 eflash 控制器结构框图

23.1.4 数据流 & 算法

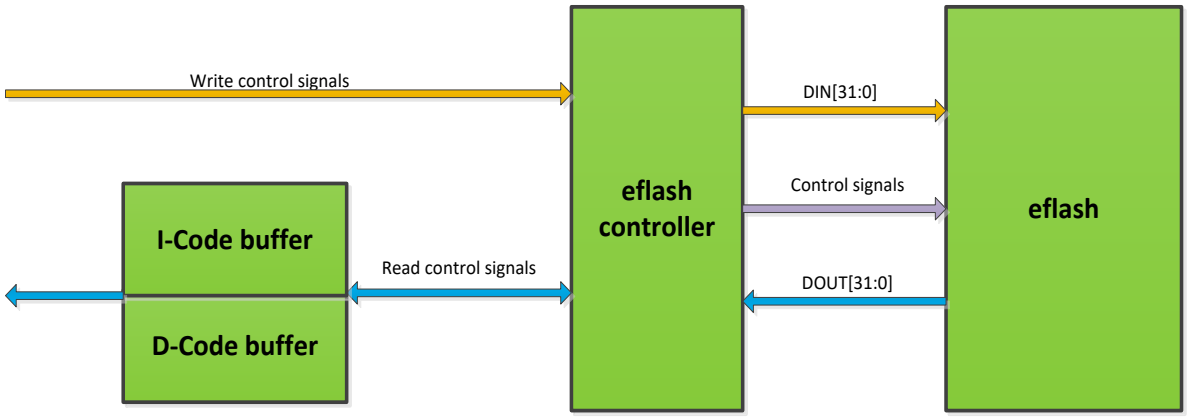

图 23-2 eflash 和 eflash 控制器数据流和算法

23.2 片内 Flash 组织

在介绍命令之前，先了解表 23-1 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。每个 2K 字节在片内 Flash 中称为一页，如表 23-1 所示。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上进行擦除、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。选项字节部分中的前 40 个字节是主存储器的写和读保护信息，后 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 23-1 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 127	0x0803 F800 ~ 0x0803 FFFF	2K	
信息存储器	Option byte	0x0804 0000 ~ 0x0804 002F	48	

片内 Flash 存储器	名称	地址	尺寸（字节）	用户权限（说明）
		0x0804 0030 ~ 0x0804 07FF（Reserved）	2000	
	ISP Firmware	0x0804 0800 ~ 0x0804 1fff	6K	读

23.3 片内 Flash 保护

在选项字节页中，内容主要有读保护，写保护，看门狗使能等。eflash 控制器具有对主存储器的写入和读取保护功能，以避免非法访问。相关信息存储在以下选项字节中，写保护信息也被加载到 **eFLASH_WPRT_EN1** ~ **eFLASH_WPRT_EN4** 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，其中补码部分由硬件自动实现，如 nRDP / nWDTEN 等。

表 23-2 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	0xFF53 FFAC	
0x0804 0004	0xFF	nWDTEN	0xFF	WDTEN	0xFFFFFFFF	
0x0804 0008	nWPRT_EN[15：0]		WPRT_EN[15：0]		0xFFFFFFFF	page 16 ~ 1
0x0804 000c	nWPRT_EN[31：16]		WPRT_EN[31：16]		0xFFFFFFFF	page 32 ~ 17
0x0804 0010	nWPRT_EN[47：32]		WPRT_EN[47：32]		0xFFFFFFFF	page 48 ~ 33
0x0804 0014	nWPRT_EN[63：48]		WPRT_EN[63：48]		0xFFFFFFFF	page 64 ~ 49
0x0804 0018	nWPRT_EN[79：64]		WPRT_EN[79：64]		0xFFFFFFFF	page 80 ~ 65
0x0804 001c	nWPRT_EN[95：80]		WPRT_EN[95：80]		0xFFFFFFFF	page 96 ~ 81
0x0804 0020	nWPRT_EN[111：96]		WPRT_EN[111：96]		0xFFFFFFFF	page 112 ~ 97
0x0804 0024	nWPRT_EN[127：112]		WPRT_EN[127：112]		0xFFFFFFFF	page 128 ~ 113
0x0804 0028	nDATA0		DATA0		0xFFFFFFFF	用户数据
0x0804 002C	nDATA1		DATA1		0xFFFFFFFF	用户数据

23.3.1 读写保护

主存储器读写保护设置如下表所示。

表 23-3 读保护

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 23-4 写保护

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

注：在写保护值中，一个比特位对应一页。

23.3.2 其他

对于 WDOG 使能设置，WDOG 必须编程为 0xCC，nWDT 必须编程为 0x33。否则，WDT 功能将被禁用。对于 DATAx/nDATAx，用户可以自由使用这两个存储单元，无需特别注意。

表 23-5 看门狗使能描述

条件	WDTEN	nWDTEN	状态
case1	0xCC	0x33	使能看门狗
其他	除了以上任意值		禁用看门狗

23.4 应用说明

23.4.1 简介

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_CTRL1，eFLASH_CTRL2，eFLASH_ADR_CMD，eFlASH_SR1。对于读操作，用户可以直接读取 eflash 存储区中表 23-1 所示的所需地址，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 132 页，由 1 个选项字节页、3 个 ISP Firmware 页和 128 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。

23.4.2 命令操作描述

23.4.2.1 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_CTRL1 和 eFlASH_CTRL2 之前，必须通过观察 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_CTRL1 和 eFlASH_CTRL2。对于所有擦除和编程命令操作，必须按比例调整 eFLASH_CTRL2 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 96MHz, CKDIV= 96/1 = 96= 7'h60；

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_CTRL1 来启动命令并触发它。应保证在 eFLASH_CTRL1 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、 在有效触发器之后，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_CTRL1；

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。

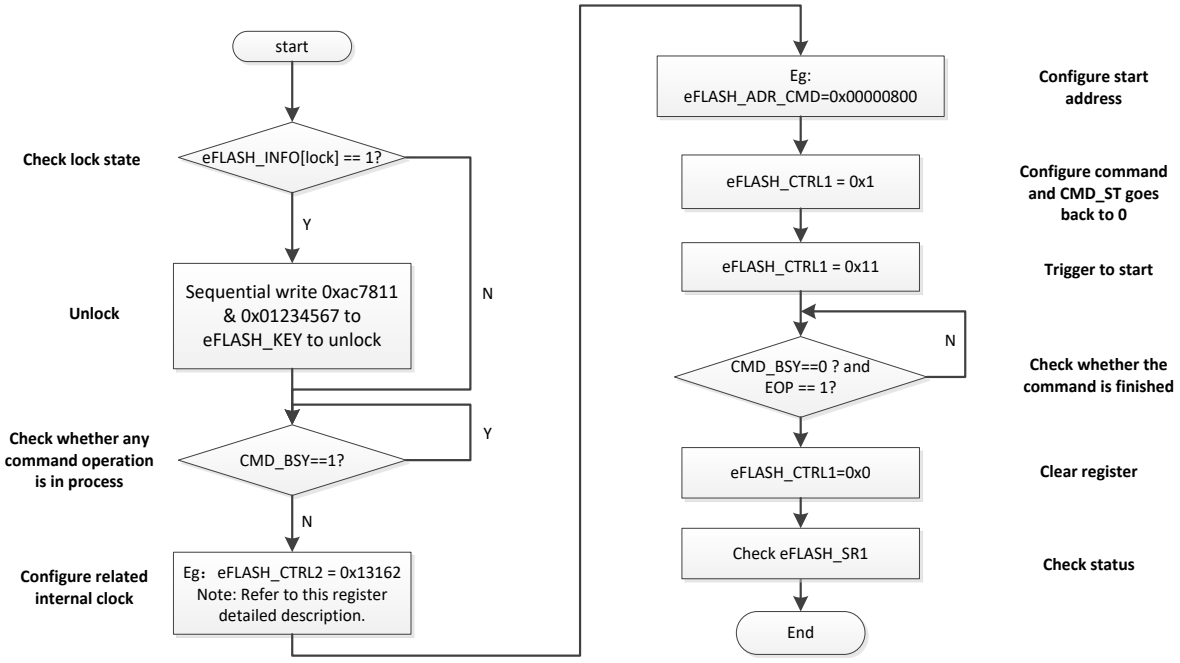

图 23-3 页擦除命令操作流程

23.4.2.2 整片擦除

整片擦除可以擦除整个用户 128 页 eflash。流程如图 23-4 所示。与页擦除相比，不需要在 eFLASH_ADR_CMD 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。

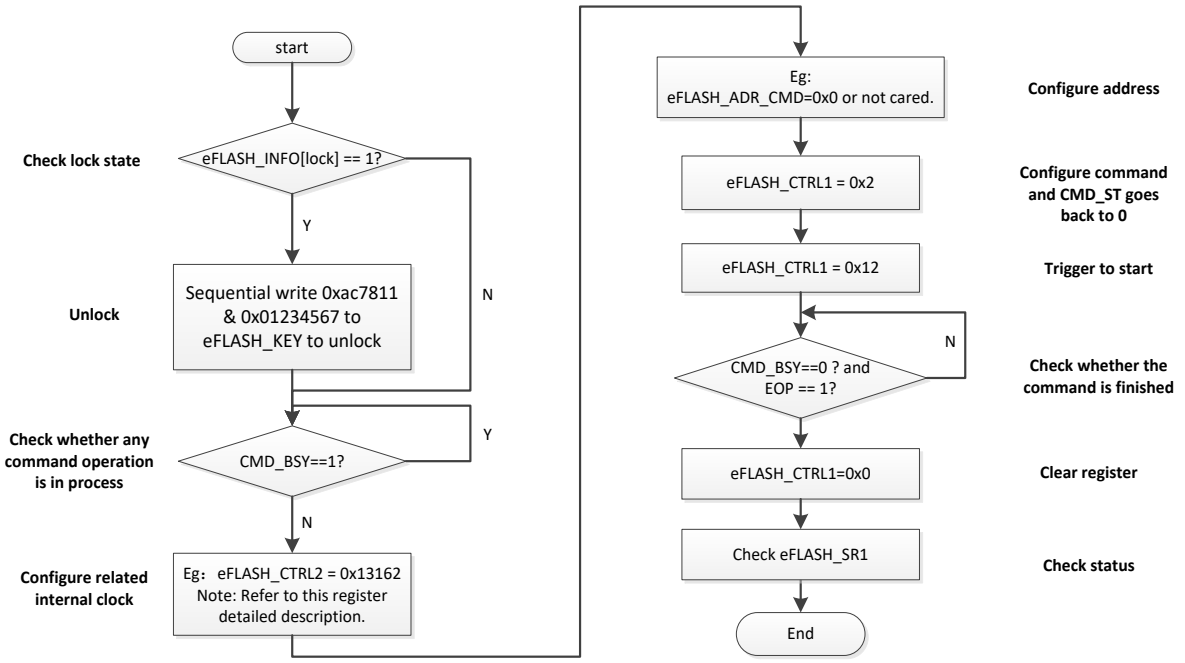

图 23-4 整片擦除命令操作流程

23.4.2.3 页编程

页编程命令可用在整个用户 128 页 eflash，流程如

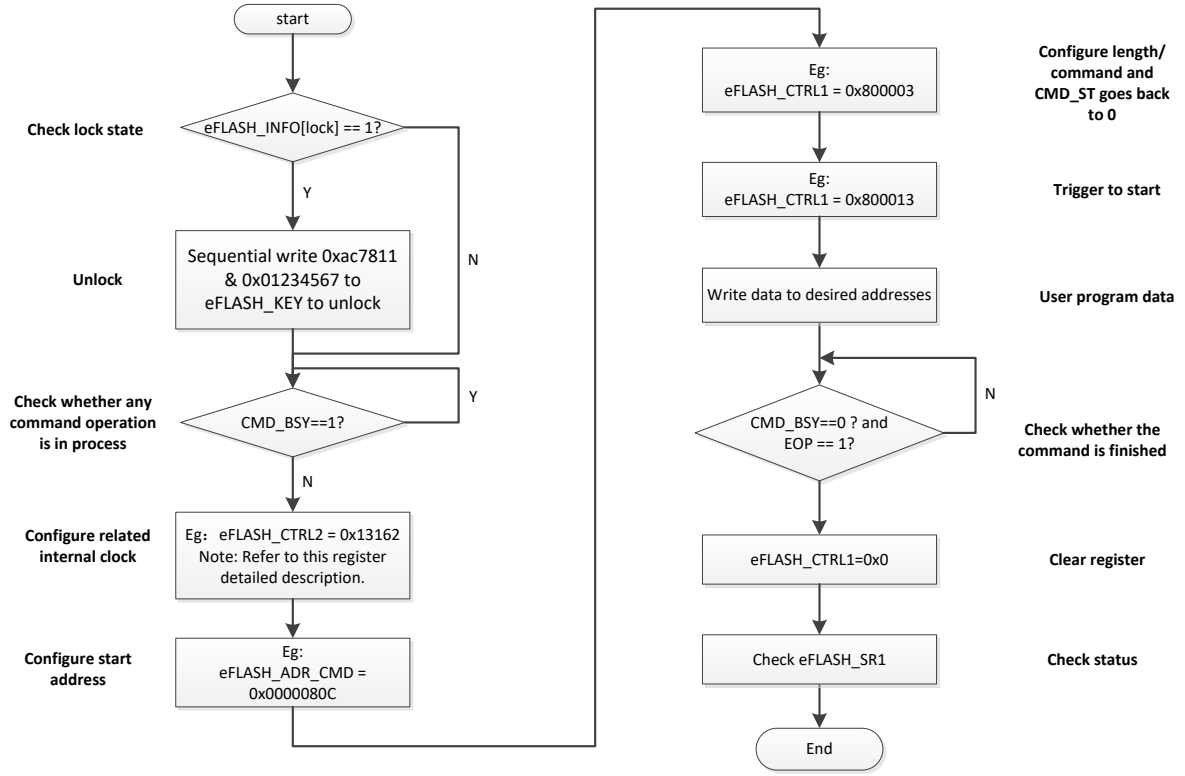

图 *23-5* 所示。页编程命令流程也类似于页擦除，但是有两个不同之处：一个是 CMD_CTL，另一个是 PROG_LENGTH [9：0]。PROG_LENGTH [9：0]位配置为指定值，以字为单位来指定编程长度。例

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，则写操作将正常验证，但编程过程还不结束，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_SR1 寄存器，以确认写操作是否成功。

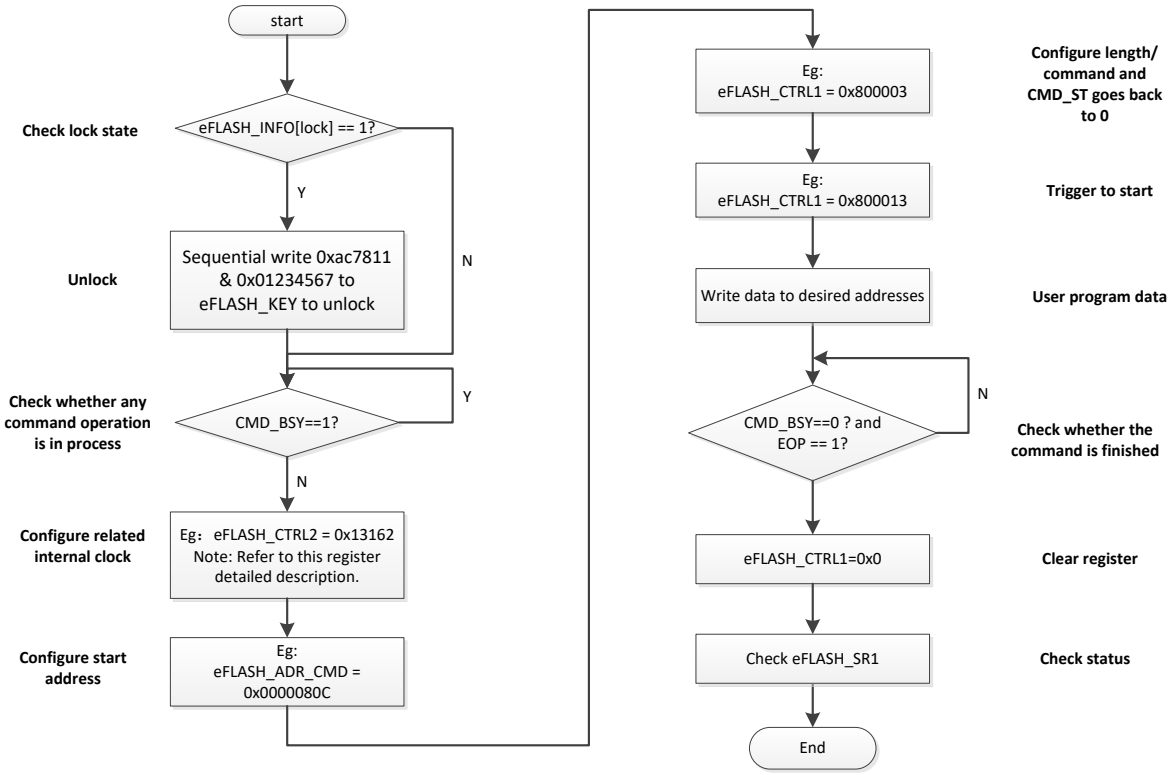

图 23-5 页编程命令操作流程

23.4.2.4 页擦除验证

页擦除验证命令用于整个用户 128 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 23-6 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：CMD_CTL。

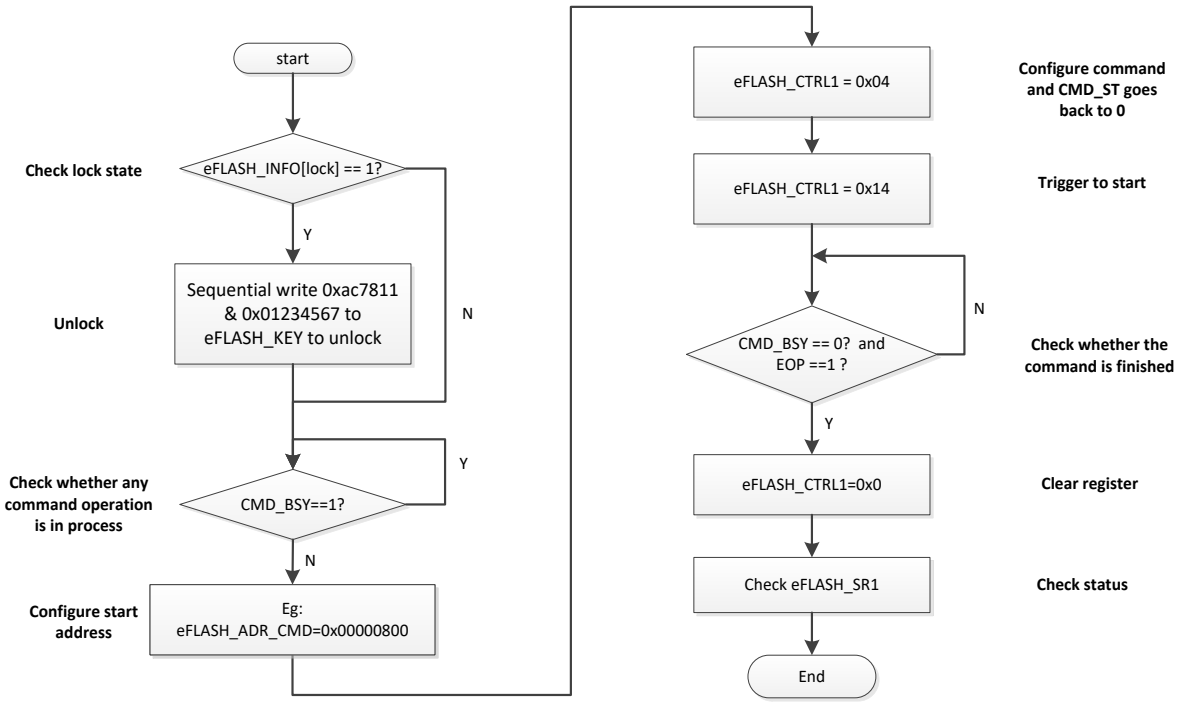

图 23-6 页擦除验证命令操作流程

23.4.2.5 整片擦除验证

整片擦除验证命令可用于整个用户 128 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 23-7 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：CMD_CTL 和 eFLASH_ADR_CMD。由于达到了整个 128 页的存储，因此无需为此命令操作指定 eFLASH_ADR_CMD。

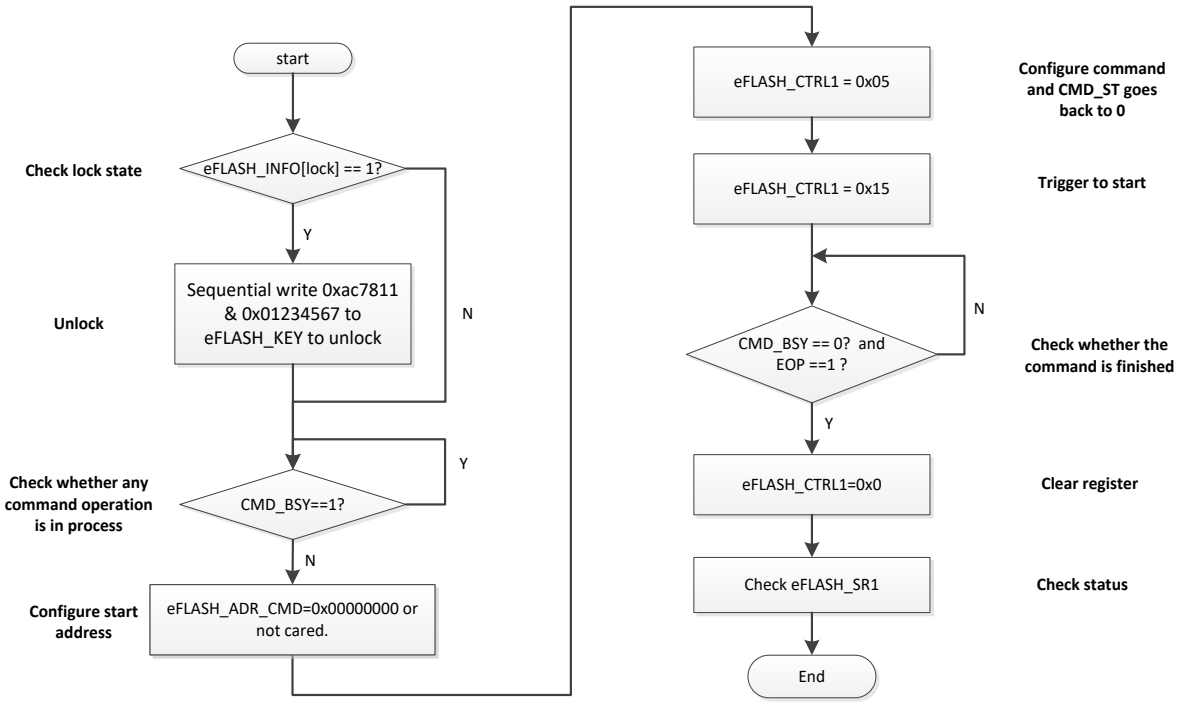

图 23-7 整片擦除验证命令操作流程

23.4.2.6 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 23-8 所示。与页擦除命令流程相比，选项字节擦除流程只有一个区别：CMD_CTL。寄存器映像中很容易描述不同之处。

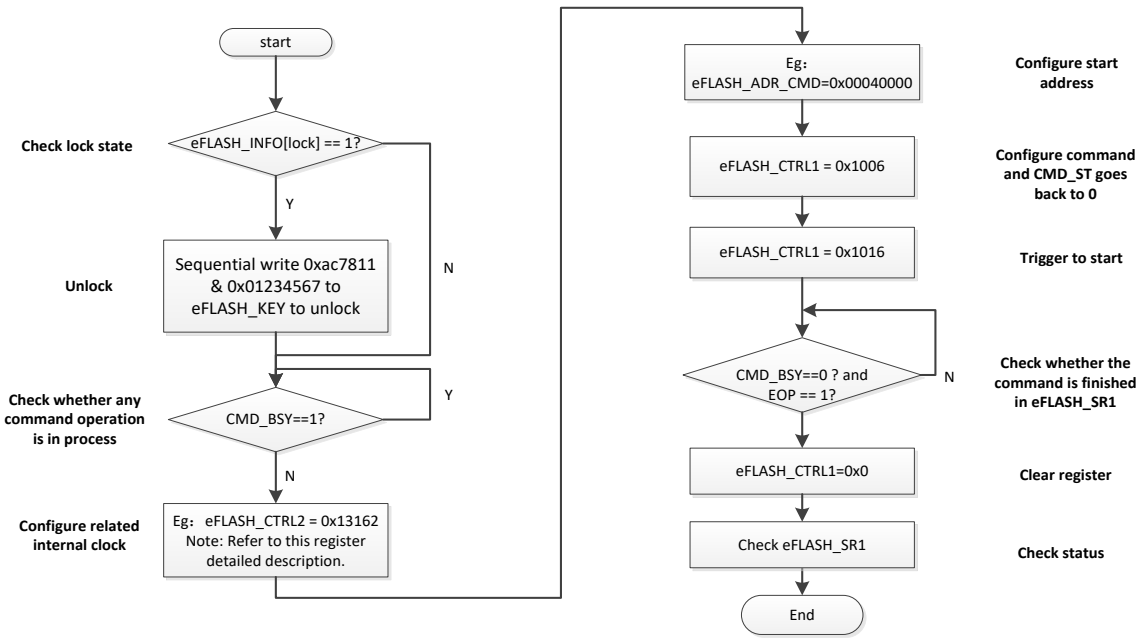

图 23-8 选项字节擦除命令操作流程

23.4.2.7 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 23-9 所示。与页编程命令流程相比，选项字节编程流程只有一个区别：CMD_CTL。 寄存器映像中很容易描述不同之处。

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，以擦除整个用户页的内容。

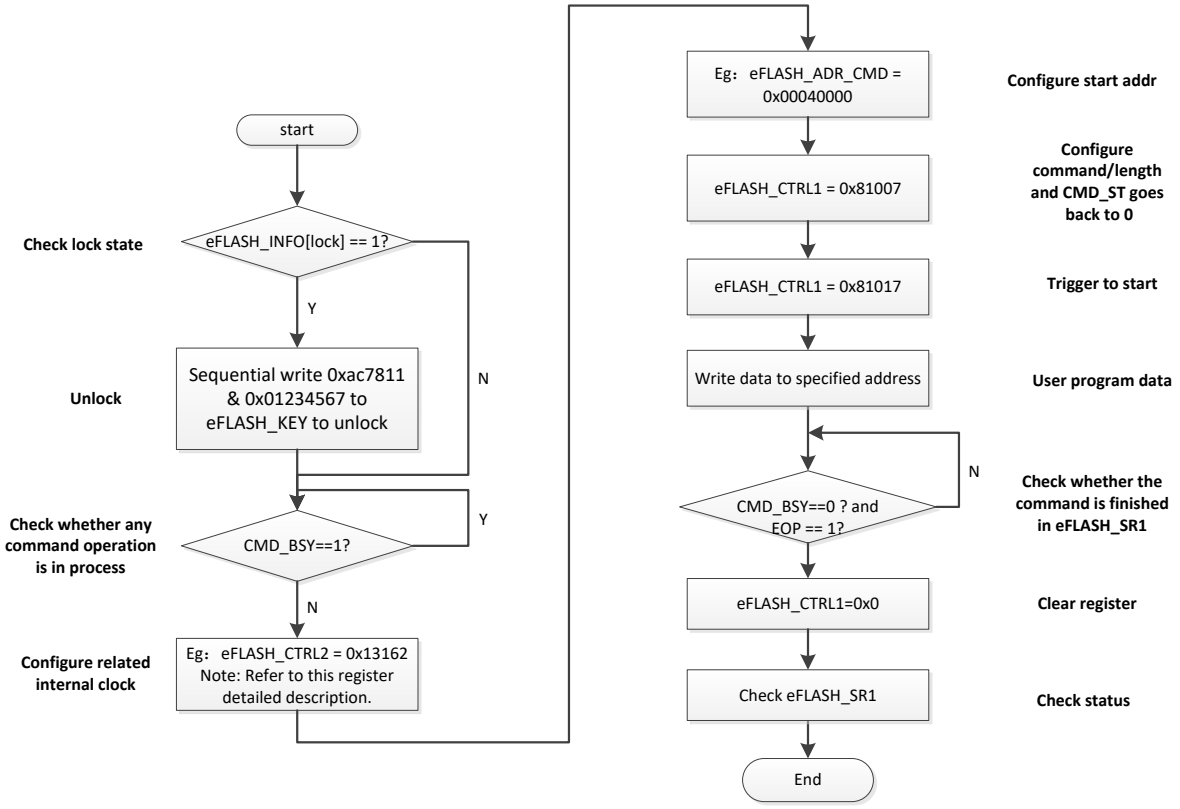

图 23-9 选项字节编程命令操作流程

23.5 寄存器定义

表 23-6 片内 Flash 寄存器映像

地址	名称	宽度	寄存器功能
0x40002000	**eFLASH_KEY**	32	解锁序列寄存器
0x40002004	**eFLASH _INFO**	32	保护信息寄存器
0x40002008	**eFLASH_ADR_CMD**	32	擦除起始地址
0x4000200c	**eFLASH_ CTRL1**	32	控制寄存器 1
0x40002010	**eFLASH_ SR1**	32	状态寄存器
0x40002014	**eFLASH_ CTRL2**	32	控制寄存器 2： 时钟设置
0x40002018 ~	**eFLASH_ WPRT_ENx**	32	写保护使能位[127：0]

0x40002024			
0x4000202C	**eFLASH_ CHIP_ID1**	32	芯片 ID 信息
0x40002030	**eFLASH_ CHIP_ID2**	32	芯片 ID 信息
0x40002034	**eFLASH_ CHIP_ID3**	32	芯片 ID 信息
0x40002038	**eFLASH_ CHIP_ID4**	32	芯片 ID 信息

0x00 eFLASH_KEY Key 序列寄存器 RESET： 0x0

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

位	助记符	名称	说明
31：0	**KEY**	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。

0x04 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**	**保留**														
类型	RW	R														
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	**保留**														**WDT_EN**	**RD_PRT**
类型	R															

位	助记符	名称	说明
31	**LOCK**	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状态，此时直接写 1 会锁定片内 Flash 控制器 1： 表示片内 Flash 控制器寄存器锁定状态，不能直接写 0 来解锁，只能通过操作寄存器 eFLASH_KEY 来解锁

位	助记符	名称	说明
1	**WDT_EN**	WDT_EN	**选项字节中 WDOG 使能 状态** 0： WDOG 关闭 1： WDOG 打开
0	**RD_PRT**	RD_PRT	**片内 Flash 读保护状态** 0： 读保护关闭 1： 读保护打开

0x08 eFLASH_ADR_CMD 擦除/编程起始地址 RESET： 0x0

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

位	助记符	名称	说明
31：0	**ADR_CMD**	ADR_CMD	**起始地址命令** 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. (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 0x0803 F000 到 0x0803 F7FF 的 127 页，那么，用户可以将 ADR_CMD [31：0]配置为从 0x0003 F000 到 0x0003 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

0x0C eFLASH_CTRL1 控制寄存器 1 RESET： 0x0

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	**HDFEN**	**RSV**					**PROG_LENGTH[9：0]**									
类型	**RW**	**R**					**RW**									

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	RSV			OPT_CMD_EN	RSV	WRPTIE	RDRTIE	EOPIE	RSV			CMD_ST	CMD_CTL[3：0]			
类型	R			RW	R	RW			R			RW	RW			

位	助记符	名称	说明
31	**HDFEN**	HDFEN	**硬故障中断使能** 0： 禁用 1： 使能
25：16	**PROG_LENGTH**	PROG_LENGTH	**编程操作长度** **注意：单位是字,** 编程长度.
12	**OPT_CMD_EN**	OPT_CMD_EN	**使能选项字节区域相关的 命令操作** **注意：参考 CMD_CTL** 0： 禁用 4’h6, 4’h7，使能 4’h1, 4’h2, 4’h3 1： 使能 4’h6, 4’h7，禁用 4’h1, 4’h2, 4’h3
10	**WRPTIE**	WRPTIE	**使能写保护错误中断** 0： 禁用 1： 使能
9	**RDRTIE**	RDRTIE	**使能读保护错误中断** 0： 禁用 1： 使能
8	**EOPIE**	EOPIE	**使能操作结束的状态和中断** 0： 禁用 1： 使能
4	**CMD_ST**	CMD_ST	**控制起始命令操作** 写 1 以触发起始命令
3：0	**CMD_CTL**	CMD_CTL	**命令** 4’h0： 空闲; 4’h1： 页擦除; 4’h2： 整片擦除; 4’h3： 页编程; 4’h4： 页擦除验证; 4’h5： 整片擦除验证; 4’h6： 选项字节擦除 ; 4’h7： 选项字节编程

0x10 eFLASH_SR1 状态寄存器 RESET： 0x702

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	RSV								RSV	RSV	RSV	EFLASH_IRQ				
类型	R															
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
类型	R			RW	R											

位	助记符	名称	说明
20	**EFLASH_IRQ**	EFLASH_IRQ	**表示 eflash 存储器中断** 注意： 共有 3 个中断源，分别为命令操作完成，写保护错误和读保护错误。 0： 没有中断 1： 发生中断
12	**FLUSH**	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，剩余内从和 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	**OPT_ER**	OPT_ERR	**表示选项字节的错误状态** OPT_ERR [3]： 0：没有错误，即： DATAx=~nDATAx 1：存在错误，即 DATAx != ~nDATAx, 除了 DATAx = nDATAx=0xff OPT_ERR [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 OPT_ERR [1]： 0：没有错误, 即 WDTEN = ~nWDTEN 1：存在错误，即 WDTEN != ~nWDTEN, 除了 WDTEN = nWDTEN=0xff OPT_ERR [0]： 0：没有错误, 即 RDP = ~nRDP 1：存在错误，即 RDP != ~nRDP, 除了 RDP = nRDP=0xff
7	**VRER**	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**	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**	PPADRER	**表示页编程命令操作中是否存在错误** 0： 没有错误

位	助记符	名称	说明
			1： 存在错误，因为程序地址非法或者所编程区域在编程前验证失败。
4	**PPERER**	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**	RDER	**表示操作违反了读保护规则** **注意：用户具体可参照 23.3.1 节** 写 1 将 RDER 清除为 0. 0：不违反读保护规则 1：违反读保护规则
2	**WRER**	WRER	**表示操作违反了写保护规则** **注意：用户具体可参照 23.3.1 节** 写 1 将 WRER 清除为 0. 0：不违反写保护规则 1：违反写保护规则
1	**EOP**	EOP	**表示命令操作是否完成** **注意： 该位对用户不重要。** 写 1 将 EOP 清除为 0. 0： 没有完成 1： 已完成
0	**CMD_BSY**	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

0x14 eFLASH_CTRL2 控制寄存器 2 RESET： 0x9

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称															**CLK_CHG_BSY**	**CYC_MANUAL**
类型	R														R	RW
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称			**SPEED_LATENCY[1：0]**					**CKDIV_LOCK**		**CKDIV[6：0]**						
类型	R		RW		R			RW	R	RW						

位	助记符	名称	说明
17	**CLK_CHG_BSY**	CLK_CHG_BSY	**表示是否准备将 eflash 控制器时钟频率更改为 72MHz 以上** **注意：CYC_MANUAL 和 CLK_CHG_BSY 总是一起使用。在将 1 写入 CYC_MANUAL 后， CLK_CHG_BSY 变为 1。并且用户可以将 eflash 操作时钟更改为 72MHz 以上，直到 CLK_CHG_BSY 变回 0。**
16	**CYC_MANUAL**	CYC_MANUAL	**控制 eflash 最大操作时钟频率**

位	助记符	名称	说明
			0： eflash 控制器时钟频率 ≤72MHz; 1： 72MHz ＜eflash 控制器时钟频率 < 120MHz.
13：12	**SPEED_LATENCY**	SPEED_LATENCY	**eFlash operation speed latency** 注： 上电系统时钟初始化完后，必须写为 0x3.
8	**CKDIV_LOCK**	CKDIV_LOCK	**锁定对于 eFLASH_CTRL2 的配置** **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_CTRL2 寄存器。** 0： 可以配置该寄存器 1： 不能配置该寄存器
6：0	**CKDIV**	CKDIV	**时钟分频用以产生 1us 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 1us 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 96MHz, CKDIV= 96/1 = 96= 7'h60，但是建议配置的这个值最好是比 7'h60 大一点，比如 7'h62。

0x18~0x24 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[31：16]**															
类型	R															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	**WPRT_ENx[15：0]**															
类型	R															

注意： x=1~4，四个 eFLASH_WPRT_ENx 寄存器组成 128 个使能位，分别决定每个主存储器页的写保护属性。

位	助记符	名称	说明
31：0	**WPRT_ENx**	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

0x2c eFLASH_CHIP_ID1 芯片 ID 1 RESET

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	**UUID3**								**UUID2**							
类型	R															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	**UUID1**								**UUID0**							
类型	R															

位	助记符	名称	说明
31：0	**CHIP_ID1**	CHIP_ID1	**芯片 ID 信息 1** UUID15~UUID0：128 位芯片 ID

0x30 eFLASH_CHIP_ID2 芯片 ID 2 RESET

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	**UUID7**								**UUID6**							
类型	R															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	**UUID5**								**UUID4**							
类型	R															

位	助记符	名称	说明
31：0	**CHIP_ID2**	CHIP_ID2	**芯片 ID 信息 2** UUID15~UUID0：128 位芯片 ID

0x34 eFLASH_CHIP_ID3 芯片 ID 3 RESET

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	**UUID11**								**UUID10**							
类型	**R**															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	**UUID9**								**UUID8**							
类型	**R**															

位	助记符	名称	说明
31：0	**CHIP_ID3**	CHIP_ID3	**芯片 ID 信息 3** UUID15~UUID0：128 位芯片 ID

0x38 eFLASH_CHIP_ID4 芯片 ID 4 RESET

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	**UUID15**								**UUID14**							
类型	R															
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称	**UUID13**								**UUID12**							
类型	R															

位	助记符	名称	说明
31：0	**CHIP_ID4**	CHIP_ID4	**芯片 ID 信息 4** UUID15~UUID0：128 位芯片 ID

24 片外串行 NOR Flash 控制器（Serial Flash Controller）

24.1 串行 NOR Flash 控制器功能概述

24.1.1 简介

串行 NOR Flash 控制器模块将 AHB/APB/R232 操作转换为 Serial flash 协议。模块内部有 32*32 比特 SRAM 用作编程数据缓冲区或读取数据缓存，且可通过编程来调整引脚的延迟满足访问串行 NOR Flash 时序需求。

串行 NOR Flash 控制器以下简称 sflash。

24.1.2 特性列表

- RISC 访问串行 NOR Flash；
- RS232 访问串行 NOR Flash；
- 串行 NOR Flash 读数据校验和；
- 系统从串行串行 NOR Flash 启动。

24.1.3 结构框图

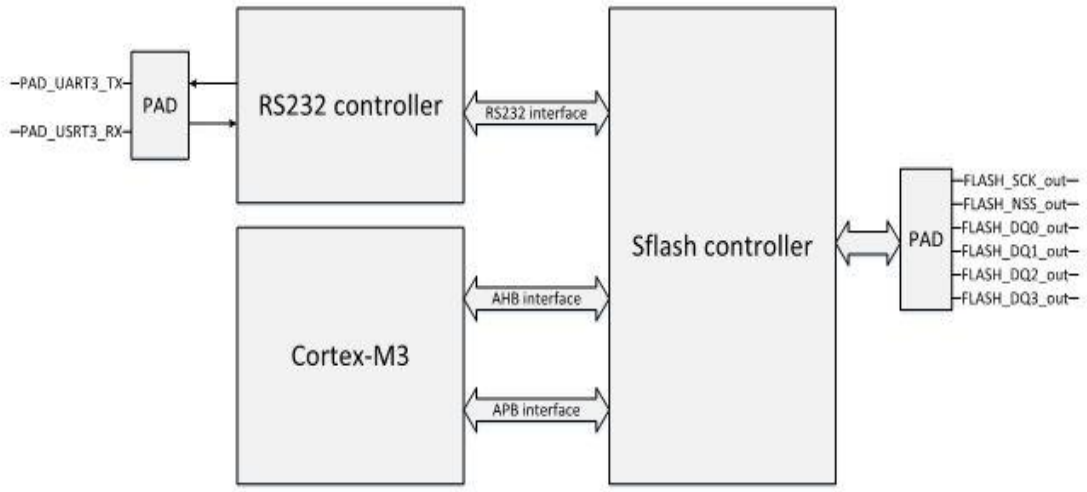

图 24-1 串行 NOR Flash 控制器模块结构框图

24.2 应用说明

24.2.1 时钟设置

串行 NOR Flash 模块有 3 个时钟：bclk，clk_flash 和 clk_rs232。APB 总线时钟（bclk）的默认频率是 AHB 总线时钟的一半，等于 4MHz。另外 2 个时钟的默认频率是相同的，即 8MHz。

相关时钟配置寄存器的绝对地址为 0x40000000，本节仅讨论 sflash_top 模块相关时钟，详细信息如下所示。

表 24-1 时钟配置寄存器

0x40000000 rs_reg_00 时钟配置寄存器

位	名称	类型	复位	说明
24	serial_clk_sel	RW	**0**	**选择不同的 clk_flash 源** 0：选择 pllmul_in 时钟，频率为 8MHz 1：选择 serial_clk_div 时钟，该时钟由 AHB 时钟产生
3：2	rg_serial_clk_div	RW	**0**	**配置 serial_clk_div 的不同分频因子** 00：分频因子为 1/2 01：分频因子为 1/4 10：分频因子为 1/6 11：分频因子为 1/8

24.2.2 读取串行 NOR Flash ID

读取标识（RDID）指令用于读取 1 字节的制造商 ID 和 2 字节的设备 ID。

发出 RDID 指令的顺序是：CS＃变为低电平→发送 RDID 指令代码→SO 上 24 位 ID 数据输出→ CS＃变为高电平。

当编程/擦除操作正在进行时，它不会解码 RDID 指令，因此对当前正在进行的编程/擦除操作的周期没有影响。当 CS＃变为高电平时，设备处于 standby 阶段。

具体步骤如下：

1) 将 REG_SF_WEPROT 写入 0x30；
2) 将 REG_SF_INTREN 写入 0x7f；
3) 将 REG_SF_PRGDATA5 写入 0x9f；
4) 将 REG_SF_ PRGDATA4 写入 0x0；
5) 将 REG_SF_ PRGDATA3 写入 0x0；
6) 将 REG_SF_ PRGDATA2 写入 0x0；
7) 将 REG_SF_CNT 写入 0x20；
8) 将 REG_SF_CMD 写入 0x4；

9) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x4, 然后将 REG_SF_INTRSTUS 写入 0x4;

10) 读取 REG_SF_SHREG2/1/0，可以获得 1 字节的制造商 ID 和 2 字节的设备 ID。

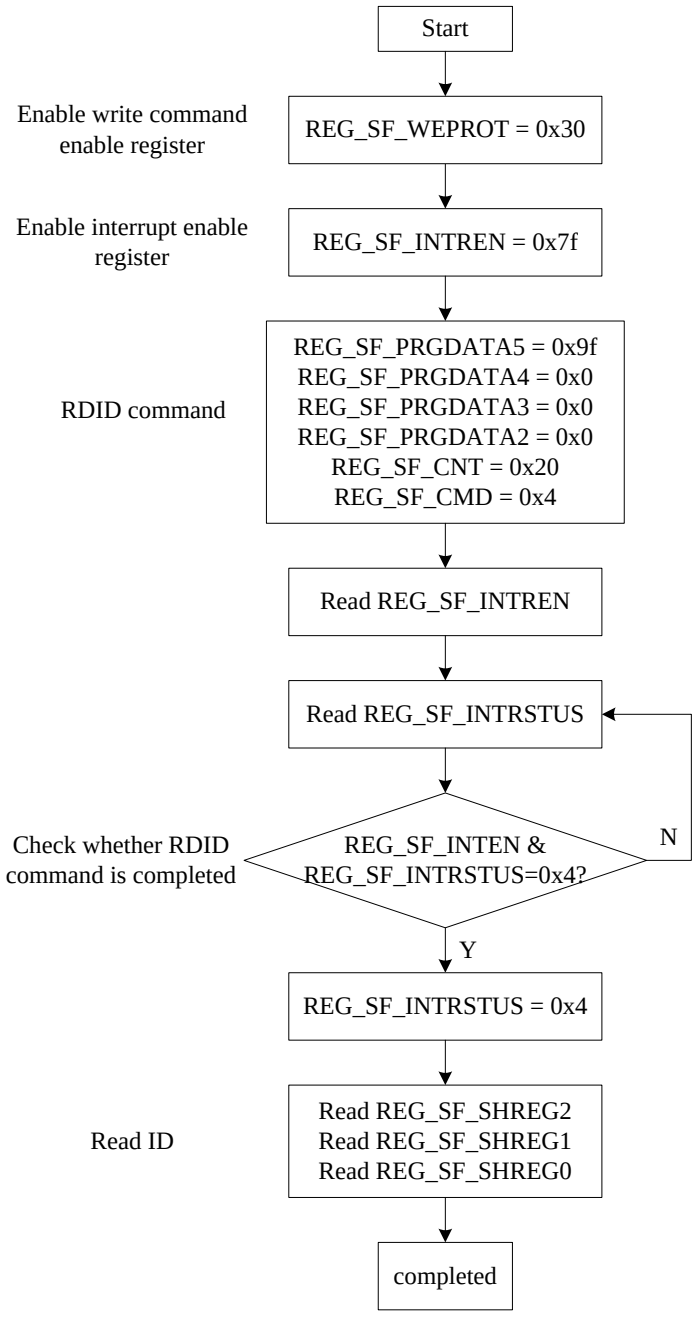

图 24-2 读串行 NOR Flash id 流程

24.2.3 写状态寄存器（WRSR）

WRSR 指令用于更改状态寄存器位和配置寄存器位的值。在发送 WRSR 指令之前，必须先执行写使能（WREN）指令。WRSR 指令可以更改块保护位的值以定义受保护的存储区域。

发出 WRSR 指令的顺序为：CS＃变为低电平→发送 WRSR 指令代码→SI 上状态寄存器数据→CS＃变为高电平。

CS＃必须精确地在 16 位数据边界处变高; 否则，该指令将被拒绝而不执行。一旦片选（CS＃）变为高电平，就会启动自定时写状态寄存器周期时间（tW）。在写入状态寄存器周期进行期间，仍可以检测出写入进行中（WIP）位。WIP 在 tW 时序期间设置为 1，在写状态寄存器周期完成时设置 0，并且写入使能锁存（WEL）位复位。

对于某些制造商，软件状态寄存器中的块写保护位的默认值为 1'b1，因此我们应该通过 WRSR 命令将状态寄存器中的块写保护位的值更改为 1'b0。软件状态寄存器中每个位的功能取决于 nor flash 的类型。

具体步骤如下：

1) 将 REG_SF_WEPROT 写入 0x30；
2) 将 REG_SF_INTREN 写入 0x7f；
3) 将 REG_SF_PRGDATA5 写入 0x6；
4) 将 REG_SF_CNT 写入 0x8；
5) 将 REG_SF_CMD 写入 x4；
6) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x4，然后将 REG_SF_INTRSTUS 写入 0x4；
7) 将 REG_SF_CMD 写入 0x2，触发 RDSR 命令；
8) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x2，然后将 REG_SF_INTRSTUS 写入 0x2；
9) **读取 REG_SF_RDSR，如果 WEL = 1，转至 步骤 3），否则继续;**
10) **将 REG_SF_PRGDATA5** 写入 **0x01;**
11) 将 REG_SF_PRGDATA4 写入 0x02;
12) 将 REG_SF_CNT 写入 0x10;
13) 将 REG_SF_CMD 写入 0x4;
14) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x4，则将 REG_SF_INTRSTUS 写入 0x4；
15) 将 REG_SF_CMD 写入 0x2，触发 RDSR 命令；
16) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x2，则将 REG_SF_INTRSTUS 写入 0x2；
17) 读取 REG_SF_RDSR（默认有效），检查块保护位的值，如果值为 1’b1，转至步骤 3）。

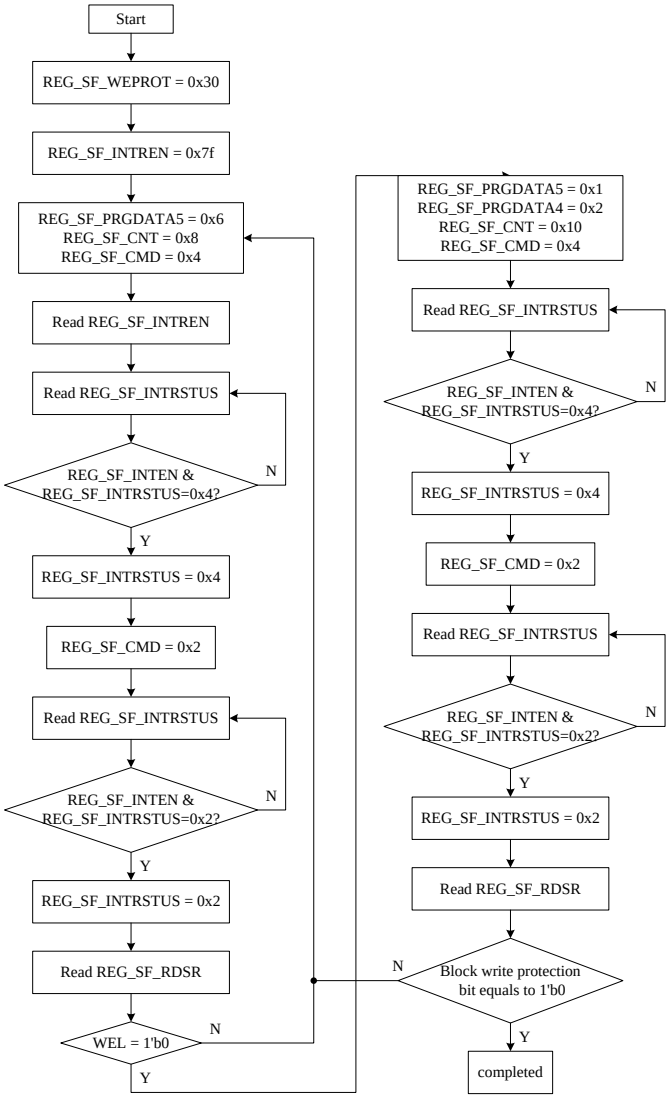

图 24-3 写状态寄存器流程

此外，WRSR 命令可用于使能四线使能（QE）位。请注意，不同型号的 nor flash 设置四线使能（QE）位的方法可能有区别，应根据相应的型号修改以下流程中的步骤 9）和步骤 10）。当 WRSR 命令完成时，RDSR 命令自动触发，这将持续到 WIP 等于 1'b0。

具体步骤如下：

1) 将 REG_SF_WEPROT 写入 0x30；
2) 将 REG_SF_INTREN 写入 0x7f；
3) 将 REG_SF_PRGDATA5 写入 0x6；
4) 将 REG_SF_CNT 写入 0x8；
5) 将 REG_SF_CMD 写入 0x4；

6) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x4，则将 REG_SF_INTRSTUS 写入 0x4；

7) 将 REG_SF_CMD 写入 0x2，触发 RDSR 命令；

8) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x2，则将 REG_SF_INTRSTUS 写入 0x2；

9) 读取 REG_SF_RDSR，如果 WEL = 1，转至 步骤 3），否则继续；

10) 将 REG_SF_PRGDATA5 写入 0x01；

11) 将 REG_SF_PRGDATA4 写入 0x40；

12) 将 REG_SF_CNT 写入 0x10;

13) 将 REG_SF_CMD 写入 0x4;

14) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x4，则将 REG_SF_INTRSTUS 写入 0x4；

15) 将 REG_SF_CMD 写入 0x2，触发 RDSR 命令；

16) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x2，则将 REG_SF_INTRSTUS 写入 0x2；

17) 读取 REG_SF_RDSR，读取块保护位的值，如果值为 1'b1，转至步骤 3）。

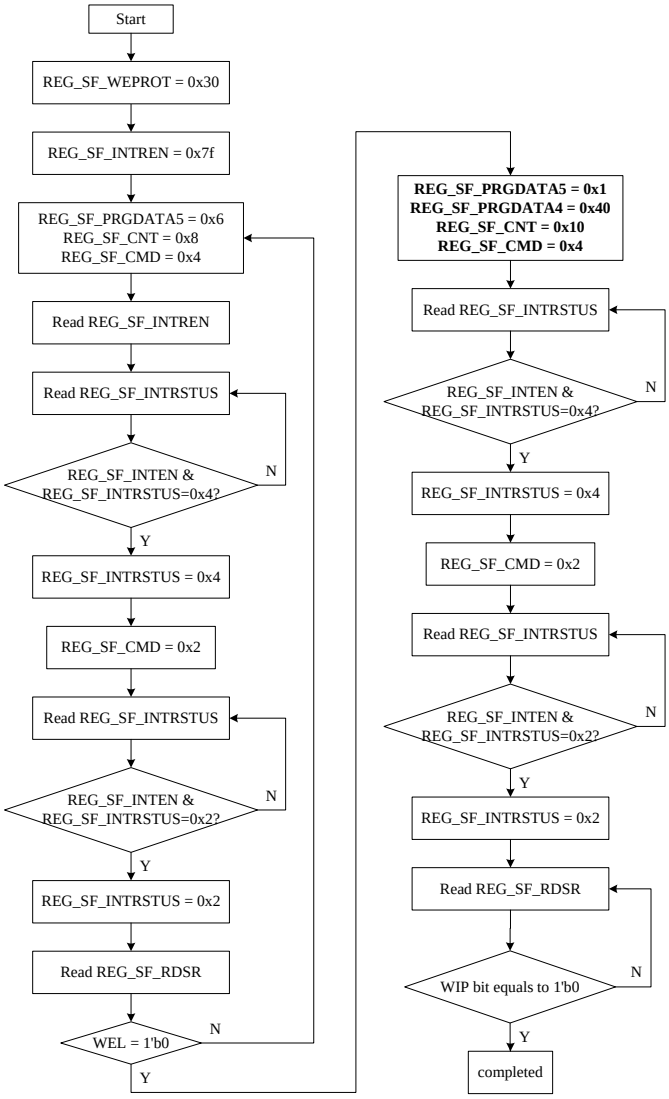

图 24-4 设置四线 QSPI 模式流程

24.2.4 擦除串行 NOR Flash（扇区/块擦除）

扇区/块擦除（SE/BE32K/BE）指令用于将所选扇区的数据擦除为“1”。该指令用于任何 4K 字节扇区和 32K/64K 字节块。在发送 SE/BE32K/BE 之前，必须执行写使能（WREN）指令以设置写使能锁存（WEL）位。扇区的任何地址都是 SE/BE32K/BE 指令的有效地址。CS＃必须正好在字节边界处高电平（地址字节的最低有效位被锁存）。否则，该指令将被拒绝并且不被执行。

地址位[Am-A12] / [Am-A15] / [Am-A16]（Am 是最重要的地址位）选择扇区地址。

发出 SE 指令的顺序为：CS＃变为低电平→发送 SE 指令代码→SI 上的 3 字节地址→CS＃变为高电平。

一旦片选（CS＃）变为高电平，就会启动自定时 SE/BE32K/ BE 周期时间（tSE/tBE32K/tBE）。当 SE / BE32K/BE 循环正在进行时，仍可以检查正在写入（WIP）位。WIP 在 tSE/tBE32K/tBE 定时期间设置 1，在 SE/BE32K/BE 周期完成时清零，并且写入使能锁存（WEL）位清零。如果块受 BP 位（块保护模式）或 SPB/DPB（高级扇区保护模式）保护，则不会在块上执行 SE/BE32K/BE 指令。

请注意，对于不同的制造商，擦除命令可能不同。对于某些制造商，仅支持 4K 字节扇区和 64K 字节块擦除，不支持 EQPI 模式。

具体步骤如下：

1) 将 REG_SF_WEPROT 写入 0x30；
2) 将 REG_SF_INTREN 写入 0x7f；
3) 释放块写保护；
4) 将 REG_SF_PRGDATA5 写入 0x6；
5) 将 REG_SF_CNT 写入 0x8；
6) 将 REG_SF_CMD 写入 0x4；
7) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x4，则将 REG_SF_INTRSTUS 写入 0x4；
8) 将 REG_SF_CMD 写入 0x2，触发 RDSR 命令；
9) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x2，则将 REG_SF_INTRSTUS 写入 0x2；
10) 读取 REG_SF_RDSR，如果 WEL = 1，转至步骤 4），否则继续；
11) 将 REG_SF_PRGDATA5 写入 0x20/0x52/0xd8（扇区擦除/32 KB 块擦除/64KB 块擦除）；
12) 将 REG_SF_ PRGDATA4 写入 addr[23：16]；
13) 将 REG_SF_ PRGDATA3 写入 addr[15：8]；
14) 将 REG_SF_ PRGDATA2 写入 addr[7：0]；
15) 将 REG_SF_CNT 写入 0x20；
16) 将 REG_SF_CMD 写入 0x4；
17) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x4，则将 REG_SF_INTRSTUS 写入 0x4；
18) 将 REG_SF_CMD 写入 0x2；
19) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x2，则将 REG_SF_INTRSTUS 写入 0x2；
20) 读取 REG_SF_RDSR，如果 WIP = 0，扇区/块循环完成，否则重复 步骤 20）直到 WIP = 0。

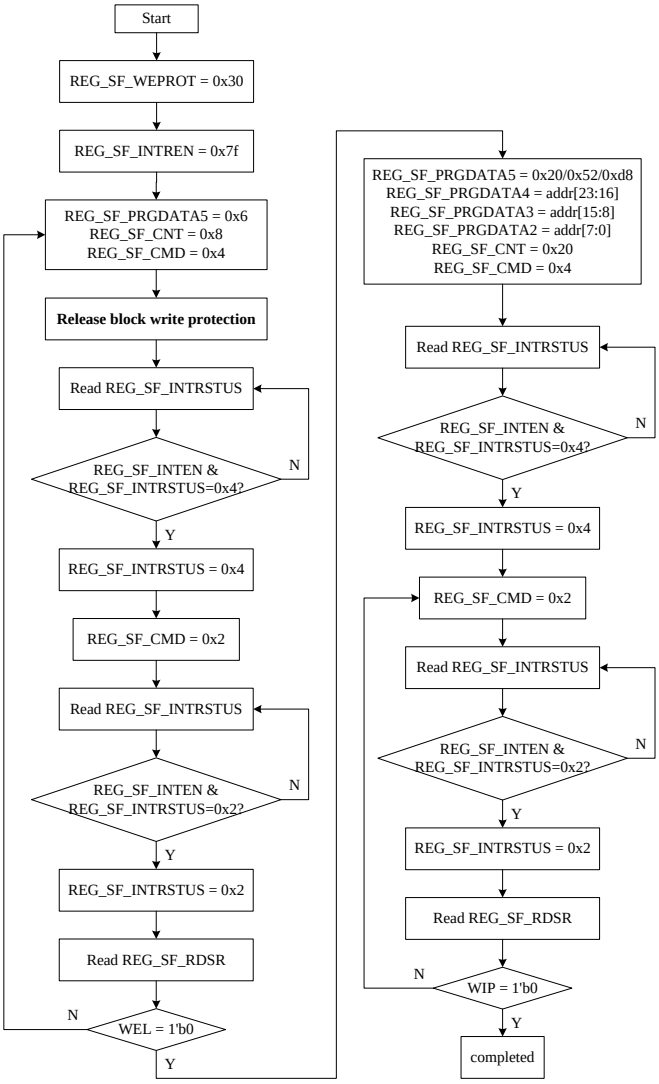

图 24-5 擦除 串行 NOR Flash（扇区/块擦除） 流程

24.2.5 擦除串行 NOR Flash（整个 NOR Flash 擦除）

整个 NOR Flash 擦除（CE）指令用于将整个 NOR Flash 的数据擦除为“1”。在发送芯片擦除（CE）之前，必须执行写使能（WREN）指令以设置写使能锁存（WEL）位。CS＃必须精确地在字节边界处变为高电平，否则指令将被拒绝而不被执行。

发出 CE 指令的顺序为：CS＃变为低电平→发送 CE 指令代码→CS＃变为高电平。

一旦片选（CS＃）变为高电平，就会启动自定时芯片擦除周期时间。芯片擦除周期正在进行时，仍可以检查正在写入（WIP）位。WIP 在 tCE 定时期间设置，并在芯片擦除周期完成时清除，并且写入使能锁存（WEL）位清零。

当芯片处于“块保护（BP）模式”下时，如果一个（或多个）扇区受 BP3-BP0 位保护，则不会执行芯片擦除（CE）指令。 只有当 BP3-BP0 都设置为 “0”时才会执行。

当芯片处于“高级扇区保护模式”时，芯片擦除（CE）指令将在未受保护的块上执行。将跳过受保护的块。 如果一个（或多个）4K 字节扇区在顶部或底部 64K 字节块中受到保护，则受保护块也将跳过芯片擦除命令。

具体步骤如下：

1) 将 REG_SF_WEPROT 写入 0x30；
2) 将 REG_SF_INTREN 写入 0x7f；
3) 释放块写入保护；

方案 1：

4) 将 REG_SF_CMD 写入 0x8；
5) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x8，则将 REG_SF_INTRSTUS 写入 0x8；
6) 将 REG_SF_CMD 写入 0x2；
7) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x2，则将 REG_SF_INTRSTUS 写入 0x2；
8) 读取 REG_SF_RDSR，如果 REG_SF_RDSR[0] = 0，芯片擦除周期完成，否则 重复步骤 6）直到 REG_SF_RDSR[0] = 0。

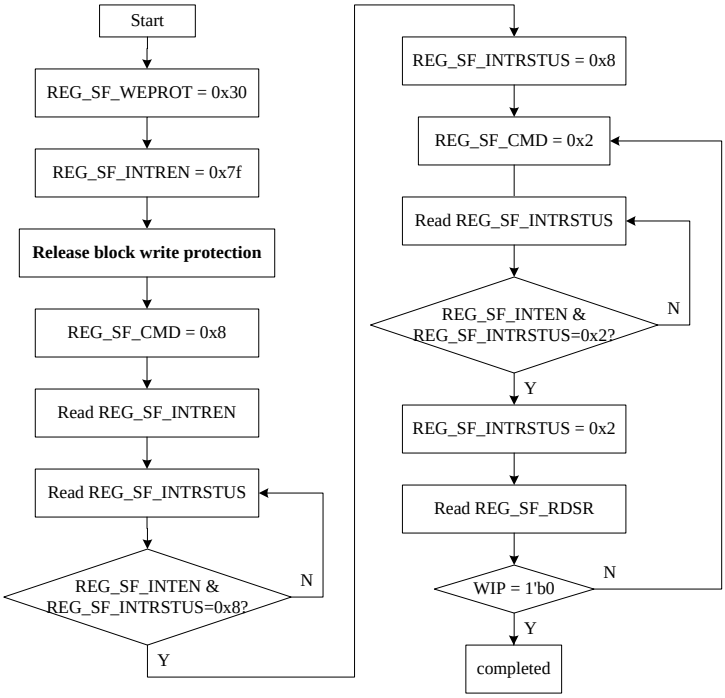

图 24-6 擦除 串行 NOR Flash（芯片擦除） 流程方案 1

方案 2：

4) 将 REG_SF_PRGDATA5 写入 0x6；
5) 将 REG_SF_CNT 写入 0x8；
6) 将 REG_SF_CMD 写入 0x4；
7) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x4，则将 REG_SF_INTRSTUS 写入 0x4；
8) 将 REG_SF_CMD 写入 0x2，触发 RDSR 命令；
9) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x2，则将 REG_SF_INTRSTUS 写入 0x2；
10) 读取 REG_SF_RDSR，如果 WEL = 1，转至步骤 4），否则继续；
11) 将 REG_SF_PRGDATA5 写入 0Xc7；
12) 将 REG_SF_CNT 写入 0x08；
13) 将 REG_SF_CMD 写入 0x4；
14) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x4，则将 REG_SF_INTRSTUS 写入 0x4；
15) 将 REG_SF_CMD 写入 0x2；
16) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x2，则将 REG_SF_INTRSTUS 写入 0x2；
17) 读取 REG_SF_RDSR，如果 WIP = 0，芯片擦除周期完成，否则 重复步骤 17） 直到 WIP = 0。

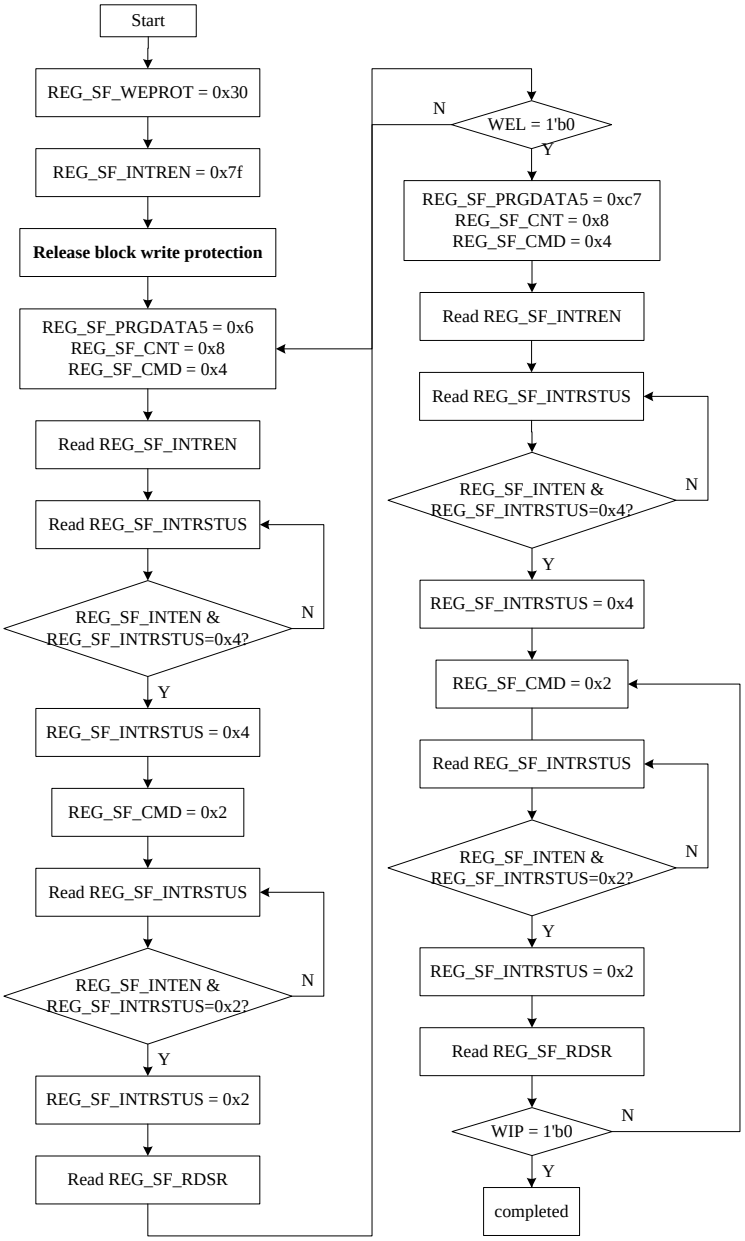

图 24-7 擦除 串行 NOR Flash（芯片） 流程方案 2

24.2.6 页编程串行 NOR Flash

页编程（PP）指令用于将存储器编程为“0”。在发送页编程（PP）之前，必须执行写使能（WREN）指令以设置写使能锁存（WEL）位。

24.2.6.1 缓冲区使能

可以在 flashif/sf_prefetch 块中使用 128 字节的缓冲区，程序数据应通过寄存器 REG_SF_PP_DW_DATA，至少 2 个 32 位，最多 32 个 32 位数据写入 32 乘 32 位缓冲区。128 个数据字节可以一次发送到 NOR flash。如果要对整个 128 个数据字节进行编程，则地址[6：0]（七个最低有效地址位）

应设置为 0。如果地址[6：0]不全为零，则超出页长度的传输数据将从当前所选页的起始地址（24 位地址，最后 7 位全部为 0）编程。

发出 PP 指令的顺序为：CS＃变位低电平→发送 PP 指令代码→SI 上的 3 字节地址→SI 上至少 8 字节，最多 128 字节数据→CS＃上的数据变为高电平。

在整个页编程周期中，CS＃必须保持低电平。CS＃必须正好在字节边界（最后的第 8 位数据被锁存）变高，否则指令将被拒绝并且不会被执行。

一旦片选（CS＃）变为高电平，就会启动自定时页编程周期时间（tPP）。在页编程循环正在进行时，仍可以检查正在写入（WIP）位。WIP 在 tPP 定时期间设置，并在页编程周期完成时清除，并且写入使能锁存（WEL）位清零。如果页受 BP 位（块保护模式）或 SPB/DPB（高级扇区保护模式）保护，则不会执行页编程（PP）指令。

具体步骤如下：

1) 将 REG_SF_WEPROT 写入 0x30；
2) 将 REG_SF_INTREN 写入 0x7f；
3) 释放块写入保护；
4) 将 REG_SF_ CFG2 写入 0x01；
5) 将 REG_SF_ PP_DW_DATA 写入预期值（32 bits），至少 2 个 32 位、最多 32 个 32 位数据（缓冲区满）；
6) 将 REG_SF_ RADR2 写入预期地址 addr[23：16]；
7) 将 REG_SF_ RADR1 写入预期地址 addr[15：8]；
8) 将 REG_SF_ RADR0 写入预期地址 addr[7：0]；
9) 将 REG_SF_CMD 写入 0x10；
10) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x10，则将 REG_SF_INTRSTUS 写入 0x10；
11) 将 REG_SF_CMD 写入 0x2；
12) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x2，则将 REG_SF_INTRSTUS 写入 0x2，否则转至步骤 12）；
13) 读取 REG_SF_RDSR，如果 WIP = 0，页编程周期完成，否则转至步骤 11）。

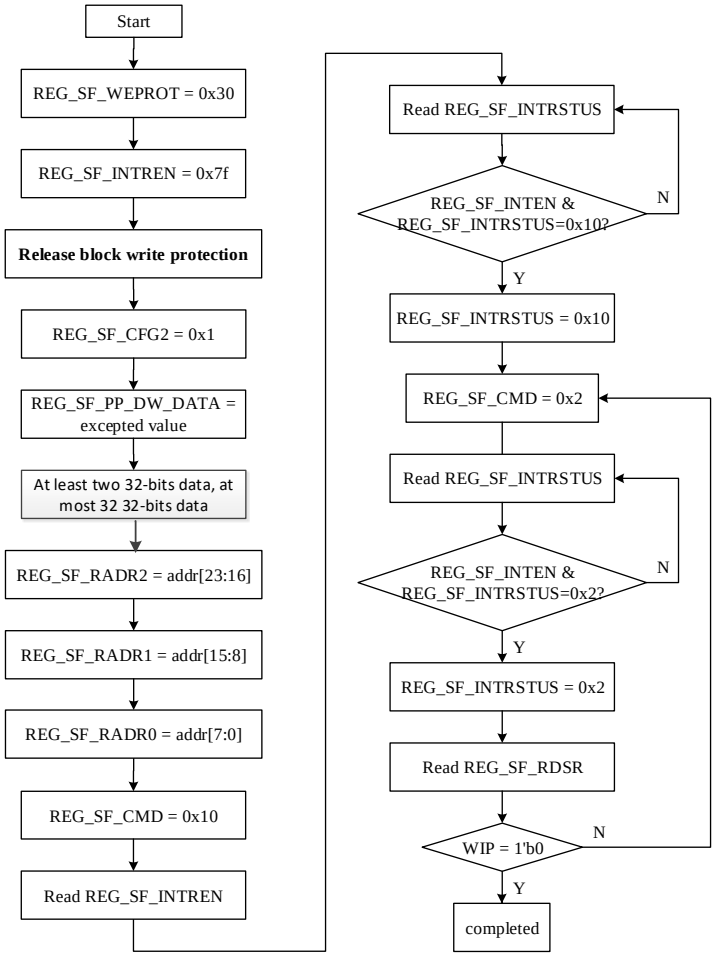

图 24-8 缓冲区使能流程

24.2.6.2 缓冲区禁用

页编程不使用 128 字节缓冲区，每次只发 1 个数据字节到 NOR flash。

发出 PP 指令的顺序为：CS＃变为低电平→发送 PP 指令代码→SI 上的 3 字节地址→1→SI 上 1 字节的数据→CS＃变为高电平。

在整个页编程周期中，CS＃必须保持低电平。CS＃必须正好在字节边界（最后的第 8 位数据被锁存）变为高电平，否则指令将被拒绝并且不会被执行。

一旦片选（CS＃）变为高电平，就会启动自定时页编程周期时间（tPP）。在页编程循环正在进行时，仍可以检查正在写入（WIP）位。WIP 在 tPP 定时期间设置，并在页编程周期完成时清除，并且写入使能锁存（WEL）位清零。 如果页受 BP 位（块保护模式）或 SPB/DPB（高级扇区保护模式）保护，则不会执行页编程（PP）指令。

具体步骤如下：

1) 将 REG_SF_WEPROT 写入 0x30；

2) 将 REG_SF_INTREN 写入 0x7f；

3) 释放块写入保护；
4) 将 REG_SF_ CFG2 写入 0x0；
5) 将 REG_SF_ WDATA 写入预期值（8 bits）；
6) 将 REG_SF_ RADR2 写入预期地址 addr[23：16]；
7) 将 REG_SF_ RADR1 写入预期地址 addr[15：8]；
8) 将 REG_SF_ RADR0 写入预期地址 addr[7：0]；
9) 将 REG_SF_CMD 写入 0x10；
10) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x10，则将 REG_SF_INTRSTUS 写入 0x10；
11) 将 REG_SF_CMD 写入 0x2；
12) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x2，则将 REG_SF_INTRSTUS 写入 0x2，否则转至步骤 12）；
13) 读取 REG_SF_RDSR，如果 WIP = 0，页编程周期完成，否则转至步骤 11）。

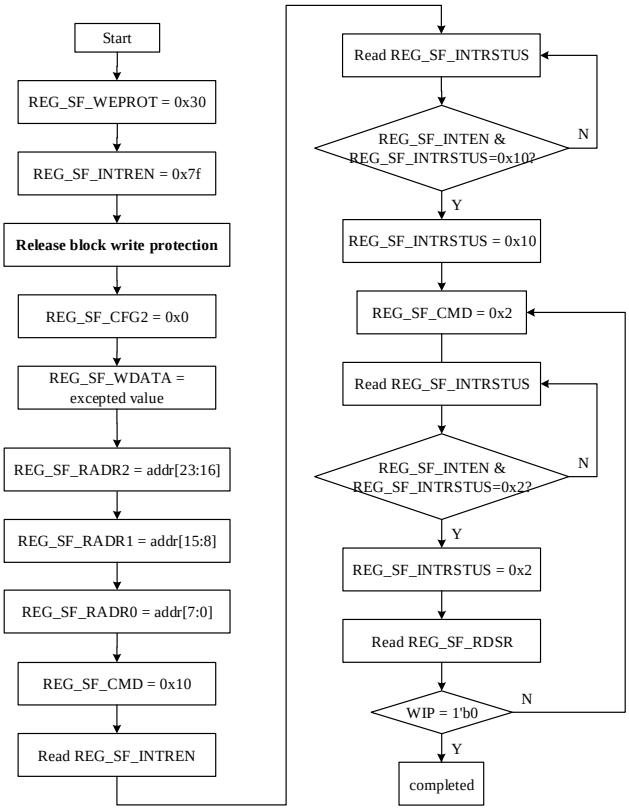

图 24-9 缓冲区禁用流程

24.2.7 4 x I/O 页编程（4PP）

4 x I/O 页编程（4PP）指令用于将存储器编程为“0”。必须执行写使能（WREN）指令以设置写使能锁存（WEL）位，在发送四页编程（4PP）之前，四路使能（QE）位必须设置为 “1”。四页编程需要四个引脚：SIO0，SIO1，SIO2 和 SIO3，它们提高了程序员的效率和应用的有效性。

24.2.7.1 缓冲区使能

发出 4PP 指令的顺序为：CS＃变为低电平→发送 4PP 指令代码→SIO [3：0]上的 3 字节地址→SIO [3：0]上的数据至少为 8 字节且最多为 128 字节 →CS＃变为高电平。

请注意，设置四线使能（QE）位的方法对于不同的制造商而言是不一致的。 不支持 EQPI 模式。

具体步骤如下：

1) 将 REG_SF_WEPROT 写入 0x30；
2) 将 REG_SF_INTREN 写入 0x7f；
3) 释放块写入保护，使能四线使能（QE） 位；
4) 将 REG_QSPI_CFG 写入 0x4/0x6；
5) 将 REG_SF_ CFG2 写入 0x11；
6) 将 REG_SF_ PRGDATA0 写入 0x32/0x38；
7) 将 REG_SF_ PP_DW_DATA 写入预期值（32 位），至少 2 个 32 位，最多 32 个 32 位 数据（缓冲区满）；
8) 将 REG_SF_ RADR2 写入预期地址 addr[23：16]；
9) 将 REG_SF_ RADR1 写入预期地址 addr[15：8]；
10) 将 REG_SF_ RADR0 写入预期地址 addr[7：0]；
11) 将 REG_SF_CMD 写入 0x10；
12) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x10，则将 REG_SF_INTRSTUS 写入 0x10；
13) 将 REG_SF_CMD 写入 0x2；
14) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x2，否则将 REG_SF_INTRSTUS 写入 0x2，否则转至 步骤 12）；
15) 读取 REG_SF_RDSR，如果 WIP = 0，页编程周期完成，否则转至步骤 11）。

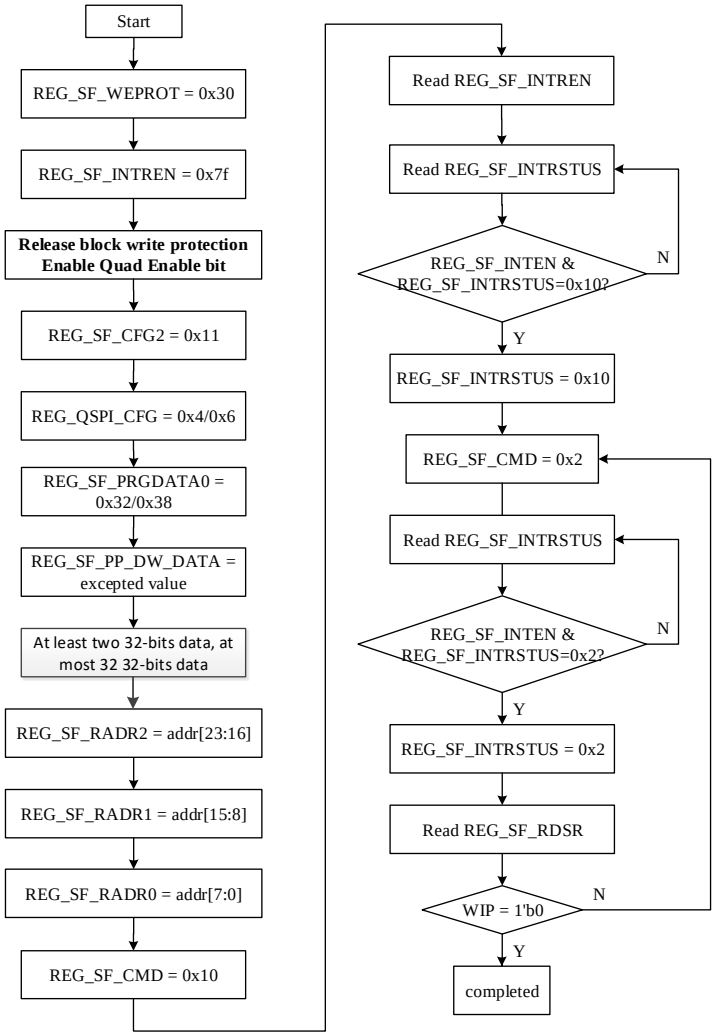

图 24-10 缓冲区使能流程

24.2.7.2 缓冲区禁用

页编程不使用 128 字节缓冲区，每次只发 1 个数据字节到 NOR flash。

发出 4PP 指令的顺序为：CS＃变为低电平→发送 4PP 指令代码→SIO [3：0]上的 3 字节地址→SIO [3：0]上的 1 字节数据→CS＃变为高电平。

请注意，设置四线使能（QE）位的方法对于不同的制造商而言是不一致的。 不支持 EQPI 模式。

具体步骤如下：

1) 将 REG_SF_WEPROT 写入 0x30；
2) 将 REG_SF_INTREN 写入 0x7f；
3) 释放块写入保护，使能四线使能（QE） 位；
4) 将 REG_QSPI_CFG 写入 0x4/0x6；
5) 将 REG_SF_ CFG2 写入 0x10；

6) 将 REG_SF_ PRGDATA0 写入 0x32/0x38；

7) 将 REG_SF_ WDATA 写入预期值（8 位）；

8) 将 REG_SF_ RADR2 写入预期地址 addr[23：16];

9) 将 REG_SF_ RADR1 写入预期地址 addr[15：8]；

10) 将 REG_SF_ RADR0 写入预期地址 addr[7：0]；

11) 将 REG_SF_CMD 写入 0x10；

12) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x10，则将 REG_SF_INTRSTUS 写入 0x10；

13) 将 REG_SF_CMD 写入 0x2；

14) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x2，则将 REG_SF_INTRSTUS 写入 0x2， 否则转至步骤 12）；

15) 读取 REG_SF_RDSR，如果 WIP = 0， 页编程周期完成，否则转至步骤 11）。

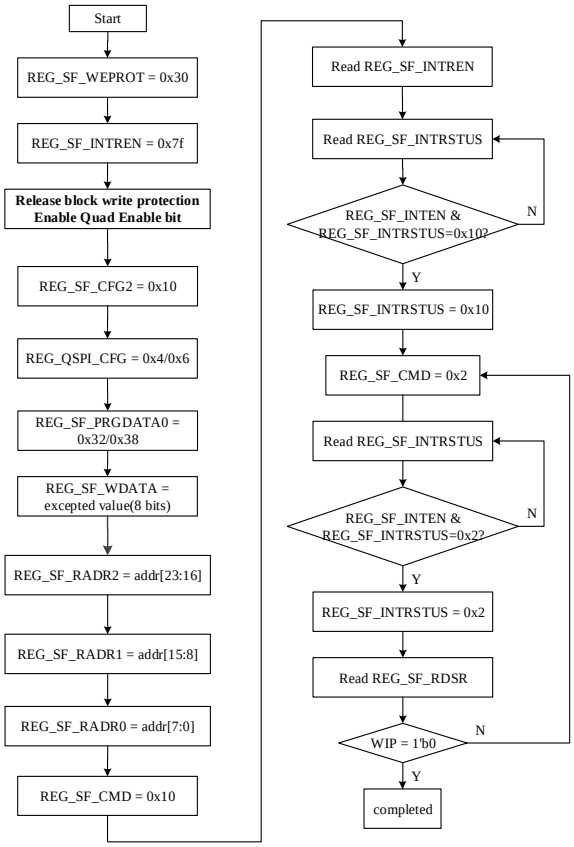

图 24-11 缓冲区禁用流程

24.2.8 AAI 编程串行 NOR Flash

AAI 程序指令允许编程多个字节的数据，而无需重新发出下一个顺序地址的位置。当要编程多个字节或整个存储器阵列时，此功能可减少总编程时间。指向受保护存储区的 AAI 程序指令将被忽略。 当启动 AAI 程序指令时，所选地址范围必须处于擦除状态（FFH）。

在 AAI WORD 编程序列中，唯一有效的指令是 AAI WORD 编程操作，RDSR，WRDI。 通过轮询软件状态寄存器中的 BUSY 进行软件检测可用于确定每个 AAI WORD 编程周期的完成。

在任何写操作之前，必须执行写使能（WREN）指令。通过执行 8 位命令 ADH（ESMT F25L16PA），然后执行地址位[A23 -A0]来启动 AAI WORD 编程指令。在此地址之后，顺序输入两个字节的数据。如果 8 位命令不是 ADH，则可以使用 2 种方法配置操作代码，详细信息请参考以下步骤 9）。

在执行 AAI WORD 编程指令前，必须将片选（CS#）驱动为高电平。在输入下一个有效命令之前，用户必须检查忙碌状态。一旦设备指示其不再处于忙状态，可以编程接下来两个连续地址的数据，依此类推。输入最后一个所需字节后，使用 RDSR 指令检查忙状态并执行 WRDI 指令，以终止 AAI。用户必须在 WRDI 之后检查忙状态，以确定设备是否准备好执行任何命令。AAI 编程期间没有 wrap 模式。一旦达到最高未受保护存储器地址，器件将退出 AAI 操作并复位 Write-Enable-Latch 位（WEL = 0）和 AAI 位（AAI = 0）。

具体步骤如下：

1) 将 REG_SF_WEPROT 写入 0x30；
2) 将 REG_SF_INTREN 写入 0x7f；
3) 释放块写入保护；
4) 将 REG_SF_ CFG2 写入 0x1；
5) 将 REG_SF_ PP_DW_DATA 写入预期值（32 位），重复 32 次直到缓冲区满为止；
6) 将 REG_SF_ RADR2 写入预期地址 addr[23：16]；
7) 将 REG_SF_ RADR1 写入预期地址 addr[15：8]；
8) 将 REG_SF_ RADR0 写入预期地址 addr[7：0]；
9) 选择合适的操作代码；

方案 1：

将 REG_SF_ CFG2 写入 0x01， 8 位命令为 AFH；

方案 2：

将 REG_SF_ CFG2 写入 0x41， 8 位命令为 ADH；

方案 3：

将 REG_SF_PRGDATA0 写入 0x**；

将 REG_SF_ CFG2 写入 0x11， 8 位命令为 0x**；

10) 将 REG_SF_ AAICMD 写入 0x1；

11) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x40，则将 REG_SF_INTRSTUS 写入 0x40；

12) 将 REG_SF_CMD 读取 0x2；

13) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x2，则将 REG_SF_INTRSTUS 写入 0x2，否则转至步骤 13）；

14) 读取 REG_SF_RDSR，如果 WIP = 0， AAI 编程周期完成，否则重复步骤 14）。

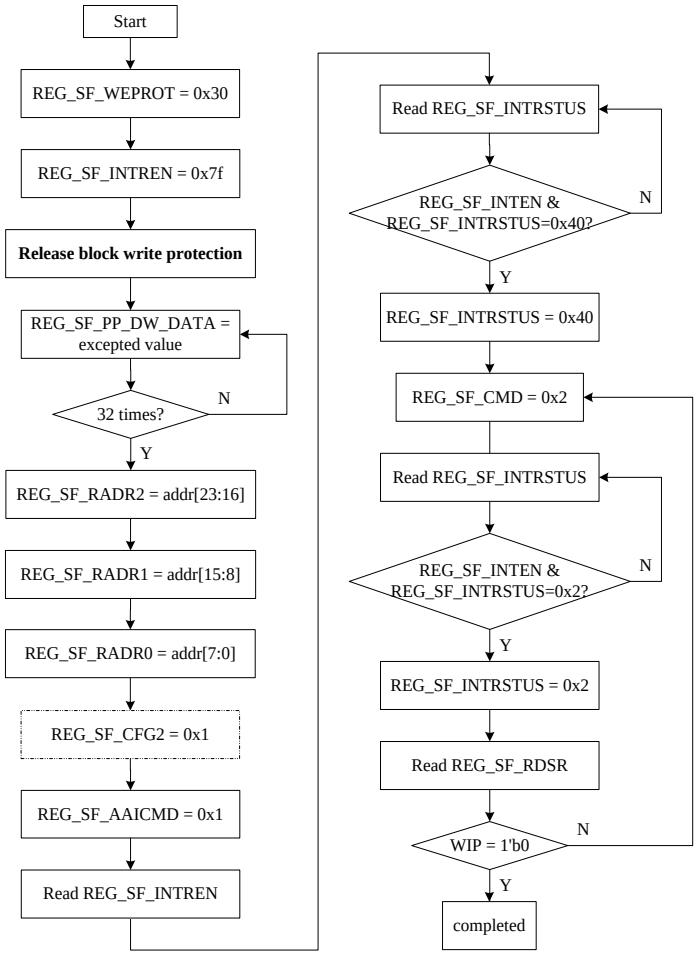

图 24-12 AAI 编程 串行 NOR Flash 流程

24.2.9 读取串行 NOR Flash

24.2.9.1 Normal read 串行 NOR Flash

读指令用于读取数据。地址在 SCLK 的上升沿锁存，数据在 SCLK 的下降沿以最大频率 fR 移出。第一个地址字节可以位于任何位置。在每个字节数据移出后，地址自动增加到下一个更高的地址，因此可以

在单个 READ 指令中读出整个存储器。达到最高地址时，地址计数器将翻转为 0。当 128 字节缓冲区已满时，读取周期完成。

发出 READ 指令的顺序为：CS＃变为低电平→发送 READ 指令代码→SI 上的 3 字节地址→SO 上数据输出→结束 READ 操作可以在数据输出期间随时使 CS＃变为高电平。

具体步骤如下：

1) 将 REG_SF_WEPROT 写入 0x30；
2) 将 REG_SF_INTREN 写入 0x7f；
3) 将 REG_SF_PRGDATA5 写入 0x6；
4) 将 REG_SF_CNT 写入 0x8；
5) 将 REG_SF_CMD 写入 0x4；
6) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x4，则将 REG_SF_INTRSTUS 写入 0x4；
7) 将 REG_SF_CMD 写入 0x2，触发 RDSR 命令；
8) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x2，则将 REG_SF_INTRSTUS 写入 0x2；
9) 读取 REG_SF_RDSR，如果 WEL = 1，转至步骤 3），否则继续；
10) 将 REG_SF_ RADR2 写入预期地址 addr[23：16]；
11) 将 REG_SF_ RADR1 写入预期地址 addr[15：8]；
12) 将 REG_SF_ RADR0 写入预期地址 addr[7：0]；
13) 将 REG_SF_ CMD 写入 0x1；
14) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x1，则将 REG_SF_INTRSTUS 写入 0x1；
15) 所有 128 字节数据都保存在缓冲区中，如果要读出数据，应该将 REG_SF_ CMD 写入 0x81，读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x1，则将 REG_SF_INTRSTUS 写入 0x1，然后读取 REG_SF_RDATA，上面的相关操作应重复 128 次。

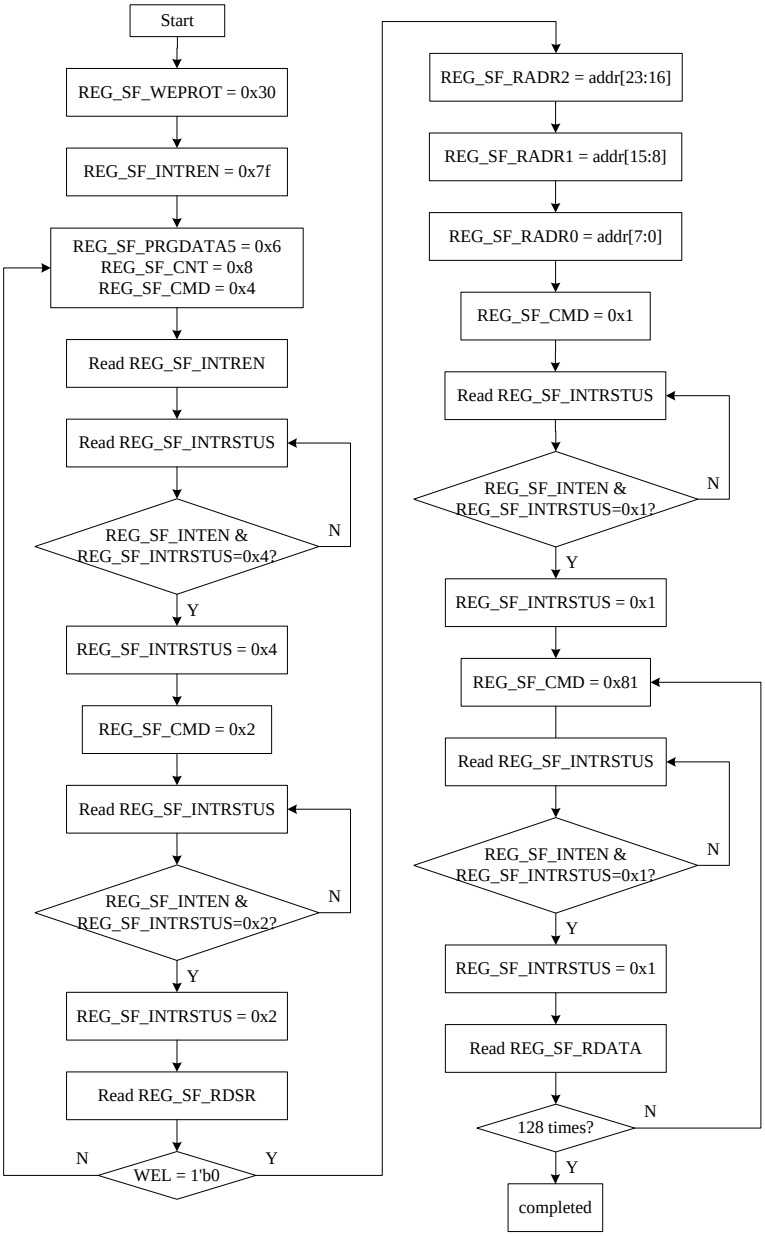

图 24-13 Normal read NOR flash 流程

24.2.9.2 Fast read 串行 NOR Flash

FAST READ 指令用于快速读取数据。地址在 SCLK 的上升沿被锁存，并且每个位的数据在 SCLK 的下降沿以最大频率 fC 移出。第一个地址字节可以位于任何位置。在每个字节数据移出后，地址自动增加到下一个更高的地址，因此可以在单个 FAST_READ 指令中读出整个存储器。当达到最高地址时，地址计数器将翻转为 0。

发出 FAST READ 指令的顺序为：CS＃变为低电平→发送 FAST_READ 指令代码→SI 上的 3 字节地址→8 个虚拟周期（默认）→SO 数据输出→结束 FAST_ READ 操作可以在数据输出的任何时间使 CS＃变为高电平。

当编程/擦除/写入状态寄存器周期正在进行时，FAST READ 指令被拒绝，而不会对编程/擦除/写入状态寄存器当前周期产生任何影响。

具体步骤如下：

1) 将 REG_SF_WEPROT 写入 0x30；
2) 将 REG_SF_INTREN 写入 0x7f；
3) 将 REG_SF_PRGDATA5 写入 0x6；
4) 将 REG_SF_CNT 写入 0x8；
5) 将 REG_SF_CMD 写入 0x4；
6) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x4，则将 REG_SF_INTRSTUS 写入 0x4；
7) 将 REG_SF_CMD 写入 0x2，触发 RDSR 命令；
8) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x2，则将 REG_SF_INTRSTUS 写入 0x2；
9) 读取 REG_SF_RDSR，如果 WEL = 1，转至步骤 3），否则继续；
10) 将 REG_SF_ RADR2 写入预期地址 addr[23：16]；
11) 将 REG_SF_ RADR1 写入预期地址 addr[15：8]；
12) 将 REG_SF_ RADR0 写入预期地址 addr[7：0]；
13) 将 REG_SF_CFG1 写入 0x1；
14) 讲 REG_SF_ CMD 写入 0x1；
15) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x1，则将 REG_SF_INTRSTUS 写入 0x1；
16) 所有 128 字节数据都保存在缓冲区中，如果要读出数据，应该将 REG_SF_ CMD 写入 0x81，然后读取 REG_SF_ RDATA，上面相关操作应重复 128 次。

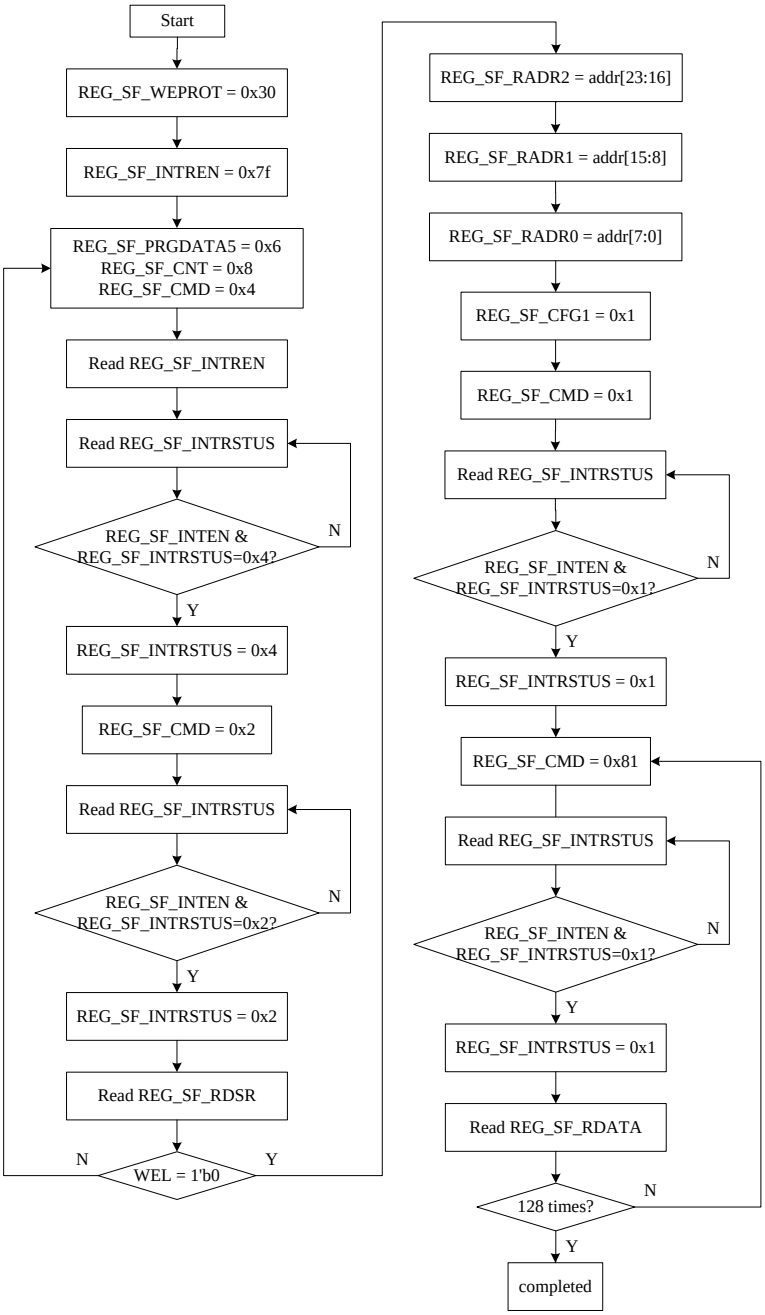

图 24-14 Fast read NOR flash 流程

24.2.9.3 Dual read 串行 NOR flash（DREAD）

DREAD 指令在读取模式下使能串行 Flash（Serial Flash）的双倍吞吐量。 地址在 SCLK 的上升沿被锁存，并且每两位数据（在 2 个 I/O 引脚上交错）在 SCLK 的下降沿以最大频率 fT 移出。第一个地址字节可以位于任何位置。在每个字节数据移出后，地址自动增加到下一个更高的地址，因此可以在单个 DREAD 指令中读出整个存储器。达到最高地址时，地址计数器将翻转为 0。当 128 字节缓冲区已满时，读取周期完成。

发出 DREAD 指令的顺序是：CS＃变为低电平→发送 DREAD 指令代码→SI 上的 3 字节地址→8 位虚拟周期→SO1 和 SO0 上的数据输出→结束 DREAD 操作可以在数据输出期间的任意时间使 CS＃变为高电平。

具体步骤如下：

1) 将 REG_SF_WEPROT 写入 0x30；
2) 将 REG_SF_INTREN 写入 0x7f；
3) 将 REG_SF_PRGDATA5 写入 0x6；
4) 将 REG_SF_CNT 写入 0x8；
5) 将 REG_SF_CMD 写入 0x4；
6) 读取 REG_SF_INTRSTUS 和 REG_SF_INTRE，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x4，则将 REG_SF_INTRSTUS 写入 0x4；
7) 将 REG_SF_CMD 写入 0x2，触发 RDSR 命令；
8) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x2，则将 REG_SF_INTRSTUS 写入 0x2；
9) 读取 REG_SF_RDSR，如果 WEL = 1，转至步骤 3），否则继续；
10) 将 REG_SF_DUAL 写入 0x1；
11) 将 REG_SF_PRGDATA3 写入 0x3B；
12) 将 REG_SF_ RADR2 写入预期地址 addr[23：16]；
13) 将 REG_SF_ RADR1 写入预期地址 addr[15：8]；
14) 将 REG_SF_ RADR0 写入预期地址 addr[7：0]；
15) 将 REG_SF_ CMD 写入 0x1；
16) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x1，则将 REG_SF_INTRSTUS 写入 0x1；
17) 所有 128 字节数据都保存在缓冲区中，如果要读出数据，应该将 REG_SF_CMD 写入 0x81，读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x1，则将 REG_SF_INTRSTUS 写入 0x1，然后 读取 REG_SF_RDATA，上面的相关操作应重复 128 次。

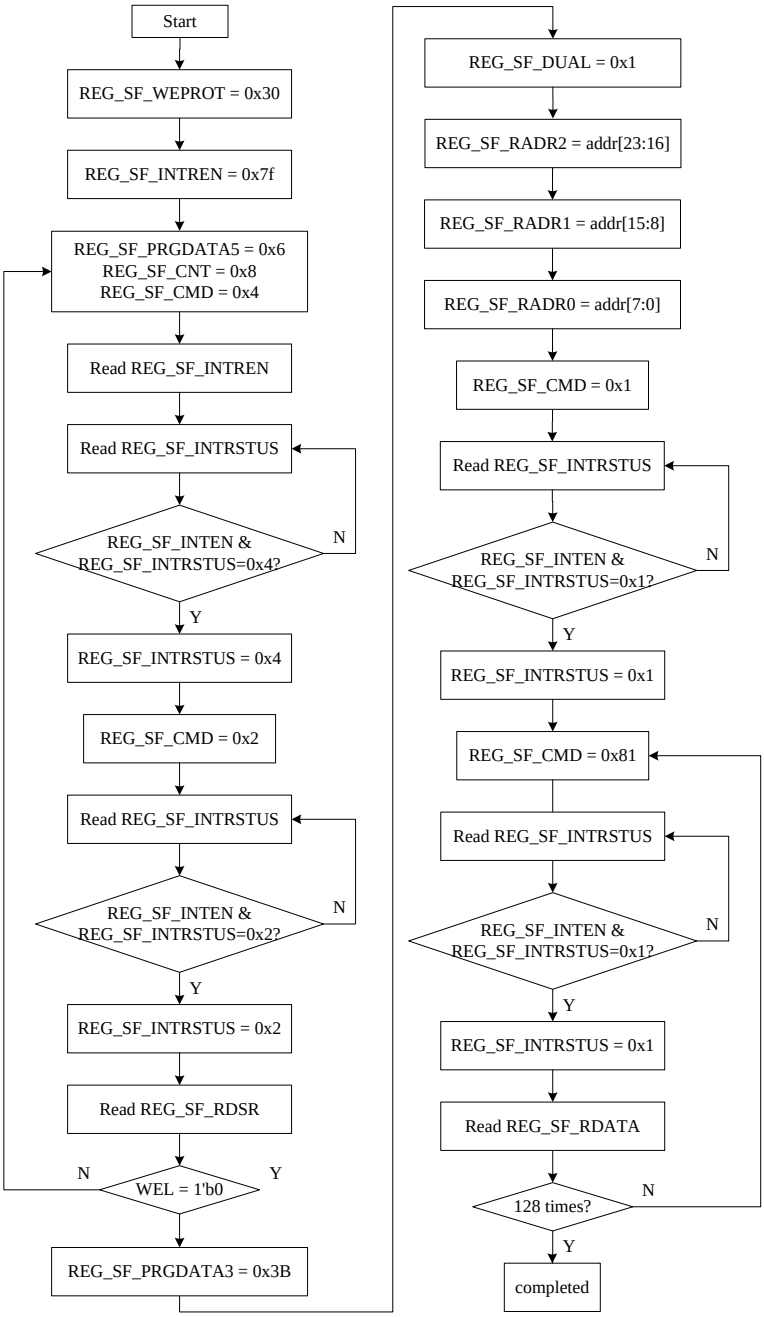

图 24-15 Dual read NOR flash 流程

24.2.9.4 2xI/O read 串行 NOR Flash（2READ）

2READ 指令在读取模式下使能串行 Flash（Serial Flash）的双倍传输速率。 地址在 SCLK 的上升沿被锁存，并且每两位数据（在 2 个 I/O 引脚上交错）在 SCLK 的下降沿以最大频率 fT 移出。第一个地址字节可以位于任何位置。在每个字节数据移出后，地址自动增加到下一个更高的地址，因此可以通过单个 2READ 指令读出整个存储器。达到最高地址时，地址计数器将翻转为 0。 当 128 字节缓冲区已满时，读取周期完成。

发出 2READ 指令的顺序为：CS＃变为低电平→发送 2READ 指令代码→SIO1 和 SIO0 上的 3 字节地址→SIO1 和 SIO0 上的 4 个虚拟周期（默认）→SIO1 和 SIO0 上的数据输出交错→结束 2READ 操作在数据输出期间可以随时使 CS＃变为高电平。

具体步骤如下：

1) 将 REG_SF_WEPROT 写入 0x30；
2) 将 REG_SF_INTREN 写入 0x7f；
3) 将 REG_SF_PRGDATA5 写入 0x6；
4) 将 REG_SF_CNT 写入 0x8；
5) 将 REG_SF_CMD 写入 0x4；
6) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x4，则将 REG_SF_INTRSTUS 写入 0x4；
7) 将 REG_SF_CMD 写入 0x2，触发 RDSR 命令；
8) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x2，则将 REG_SF_INTRSTUS 写入 0x2；
9) 读取 REG_SF_RDSR，如果 WEL = 1，转至步骤 3），否则继续；
10) 将 REG_SF_DUAL 写入 0x3；
11) 将 REG_SF_PRGDATA3 写入 0xBB；
12) 将 REG_SF_ RADR2 写入预期地址 addr[23：16]；
13) 将 REG_SF_RADR1 写入预期地址 addr[15：8]；
14) 将 REG_SF_RADR0 写入预期地址 addr[7：0]；
15) 将 REG_SF_CMD 写入 0x1；
16) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x1，则将 REG_SF_INTRSTUS 写入 0x1；
17) 所有 128 字节数据都保存在缓冲区中，如果要读出数据，应该将 REG_SF_ CMD 写入 0x81，读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x1，则将 REG_SF_INTRSTUS 写入 0x1，然后读取 REG_SF_ RDATA，上面的相关操作应重复 128 次。

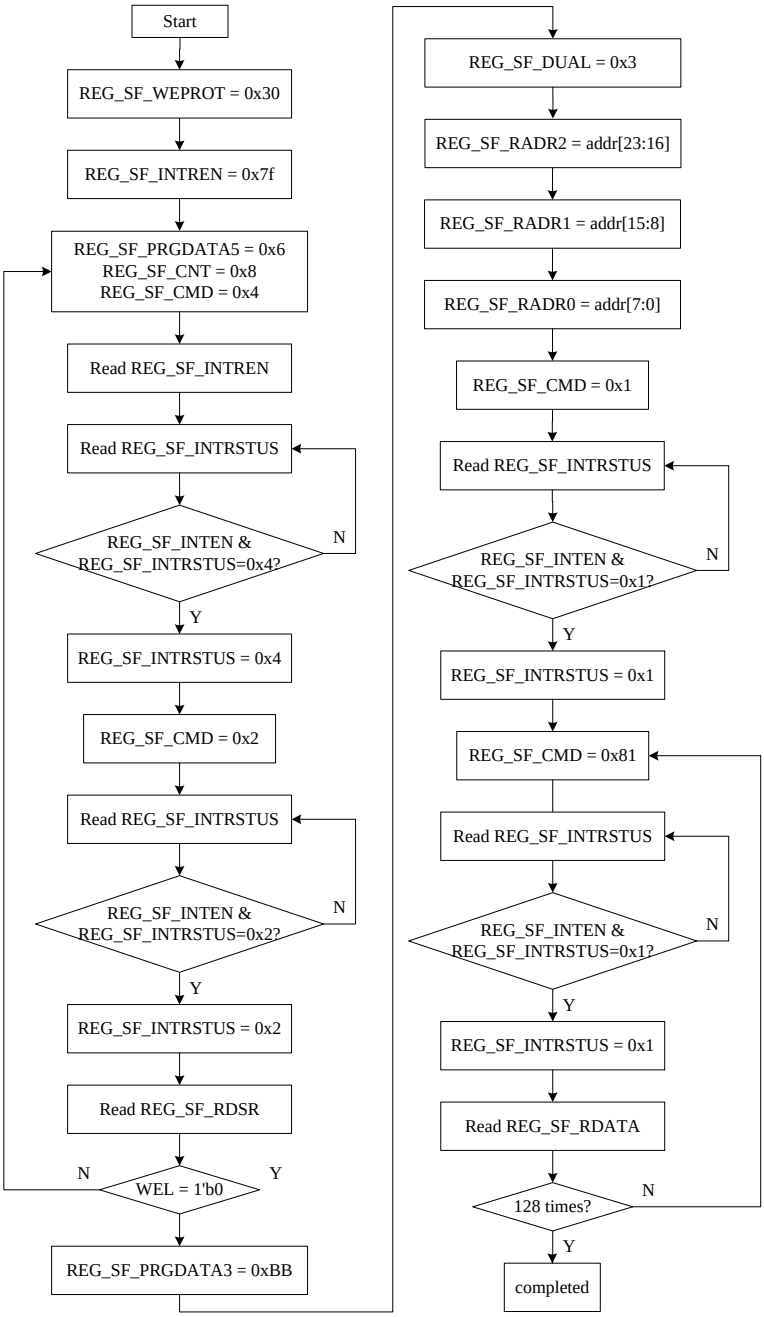

图 24-16 2 x I/O Read 串行 NOR Flash 流程

24.2.9.5 Quad read 串行 NOR Flash（QREAD）

QREAD 指令在读取模式下启用串行 Flash 的四倍吞吐量。 地址在 SCLK 的上升沿锁存，每 4 位数据（4 个 I/O 引脚上的交错）在 SCLK 的下降沿以最大频率 fQ 移出。第一个地址字节可以位于任何位置。在每个字节数据移出后，地址自动增加到下一个更高的地址，因此可以在单个 QREAD 指令中读出整个存储器。达到最高地址时，地址计数器将翻转为 0。 当 128 字节缓冲区已满时，读取周期完成。

发出 QREAD 指令的顺序为：CS＃变为低电平→发送 QREAD 指令代码→SI 上的 3 字节地址→8 位虚拟周期→SIO3，SIO2，SIO1 和 SIO0 上的数据输出交错→结束 READ 操作可以使 CS ＃在数据输出期间随时变为高电平。

具体步骤如下：

1) 将 REG_SF_WEPROT 写入 0x30；
2) 将 REG_SF_INTREN 写入 0x7f；
3) 将 REG_SF_PRGDATA5 写入 0x6；
4) 将 REG_SF_CNT 写入 0x8；
5) 将 REG_SF_CMD 写入 0x4；
6) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x4，则将 REG_SF_INTRSTUS 写入 0x4；
7) 将 REG_SF_CMD 写入 0x2，触发 RDSR 命令；
8) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x2，则将 REG_SF_INTRSTUS 写入 0x2；
9) 读取 REG_SF_RDSR，如果 WEL = 1，转至步骤 3），否则继续；
10) 释放块读取保护，使能四线使能（QE）位；
11) 将 REG_SF_DUAL 写入 0x4；
12) 将 REG_SF_PRGDATA4 写入 0x6B；
13) 将 REG_SF_ RADR2 写入预期地址 addr[23：16]；
14) 将 REG_SF_ RADR1 写入预期地址 addr[15：8]；
15) 将 REG_SF_ RADR0 写入预期地址 addr[7：0]；
16) 将 REG_SF_ CMD 写入 0x1；
17) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x1，则将 REG_SF_INTRSTUS 写入 0x1；
18) 所有 128 字节数据都保存在缓冲区中，如果要读出数据，应该将 REG_SF_ CMD 写入 0x81，读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x1，则将 REG_SF_INTRSTUS 写入 0x1，然后读取 REG_SF_ RDATA，上面的相关操作应重复 128 次。

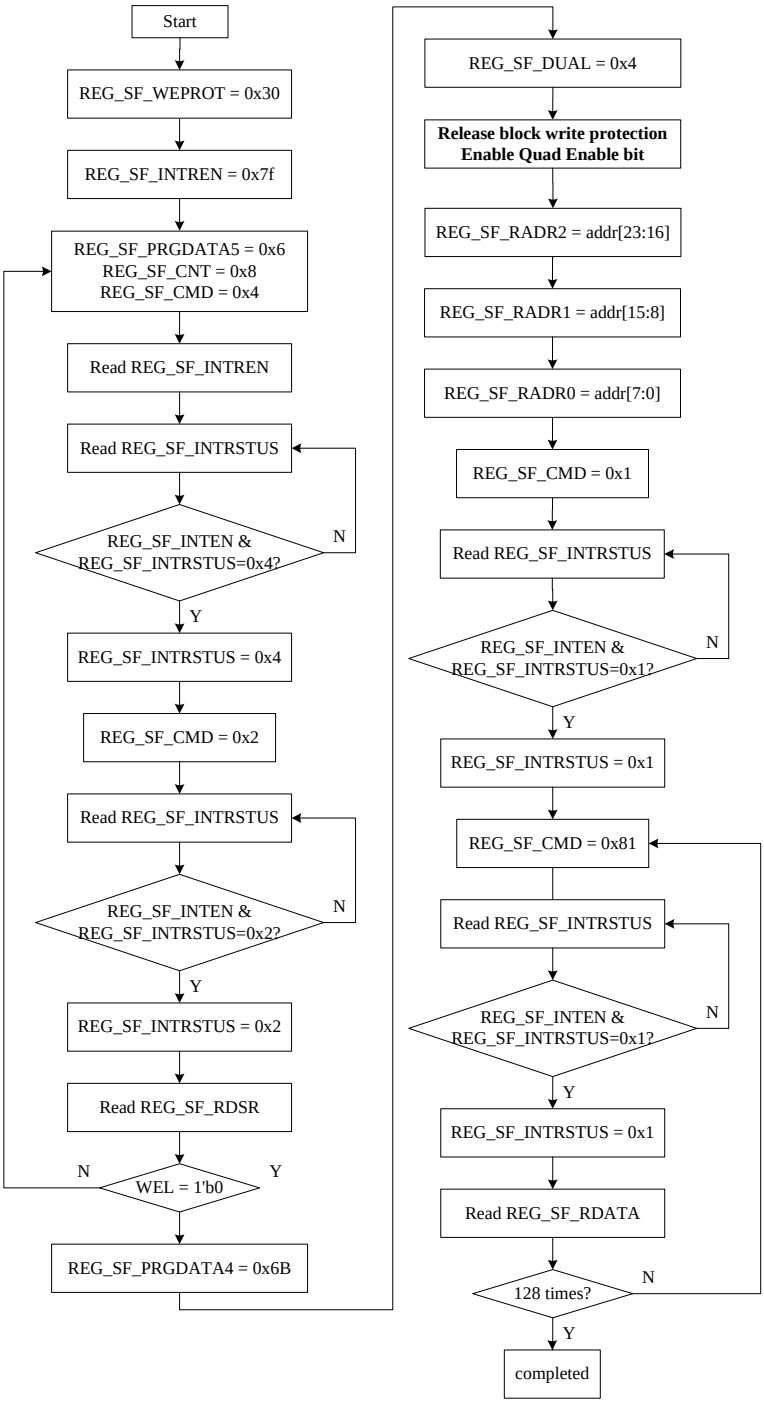

图 24-17 Quad read 串行 NOR Flash 流程

24.2.9.6 4xI/O read 串行 NOR Flash

4READ 指令在读取模式下使能串行 Flash（Serial Flash）的四倍吞吐量。在发送 4READ 指令之前，状态寄存器的四路使能（QE）位必须设置为“1”。 地址在 SCLK 的上升沿锁存，每 4 位数据（4 个 I/O 引脚上的交错）在 SCLK 的下降沿以最大频率 fQ 移出。第一个地址字节可以位于任何位置。在每个字节数据移出后，地址自动增加到下一个更高的地址，因此可以在单个 4READ 指令中读出整个存储器。 达到最高地址时，地址计数器将翻转为 0。当 128 字节缓冲区已满时，读取周期完成。

发出 4READ 指令的顺序为：CS＃变为低电平→发送 4READ 指令代码→SIO3，SIO2，SIO1 和 SIO0 上字节地址交织→2+ 4 个虚拟周期（默认）→SIO3，SIO2，及 SIO1 & SIO0 上的数据输出交织→结束 4READ 操作可以在数据输出任何时候使 CS＃变为高电平。

具体步骤如下：

1) 将 REG_SF_WEPROT 写入 0x30；
2) 将 REG_SF_INTREN 写入 0x7f；
3) 将 REG_SF_PRGDATA5 写入 0x6；
4) 在 REG_SF_CNT 写入 0x8；
5) 在 REG_SF_CMD 写入 0x4；
6) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x4，则将 REG_SF_INTRSTUS 写入 0x4；
7) 将 REG_SF_CMD 写入 0x2，触发 RDSR 命令；
8) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x2，则将 REG_SF_INTRSTUS 写入 0x2；
9) 读取 REG_SF_RDSR，如果 WEL = 1，转至步骤 3），否则继续；
10) 释放块写入保护，使能 QE 位；
11) 将 REG_SF_DUAL 写入 0xc；
12) 将 REG_SF_PRGDATA4 写入 0xEB；
13) 将 REG_DUMMY_CFG 写入合适的值；
14) 将 REG_SF_ RADR2 写入预期地址 addr[23：16]；
15) 将 REG_SF_ RADR1 写入预期地址 addr[15：8]；
16) 将 REG_SF_ RADR0 写入预期地址 addr[7：0]；
17) 将 REG_SF_ CMD 写入 0x1；
18) 读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x1，则将 REG_SF_INTRSTUS 写入 0x1；
19) 所有 128 字节数据都保存在缓冲区中，如果要读出数据，应该将 REG_SF_ CMD 写入 0x81，读取 REG_SF_INTRSTUS 和 REG_SF_INTREN，如果 REG_SF_INTRSTUS & REG_SF_INTREN 的值为 0x1，则将 REG_SF_INTRSTUS 写入 0x1，然后读取 REG_SF_RDATA，上面的相关操作应重复 128 次。

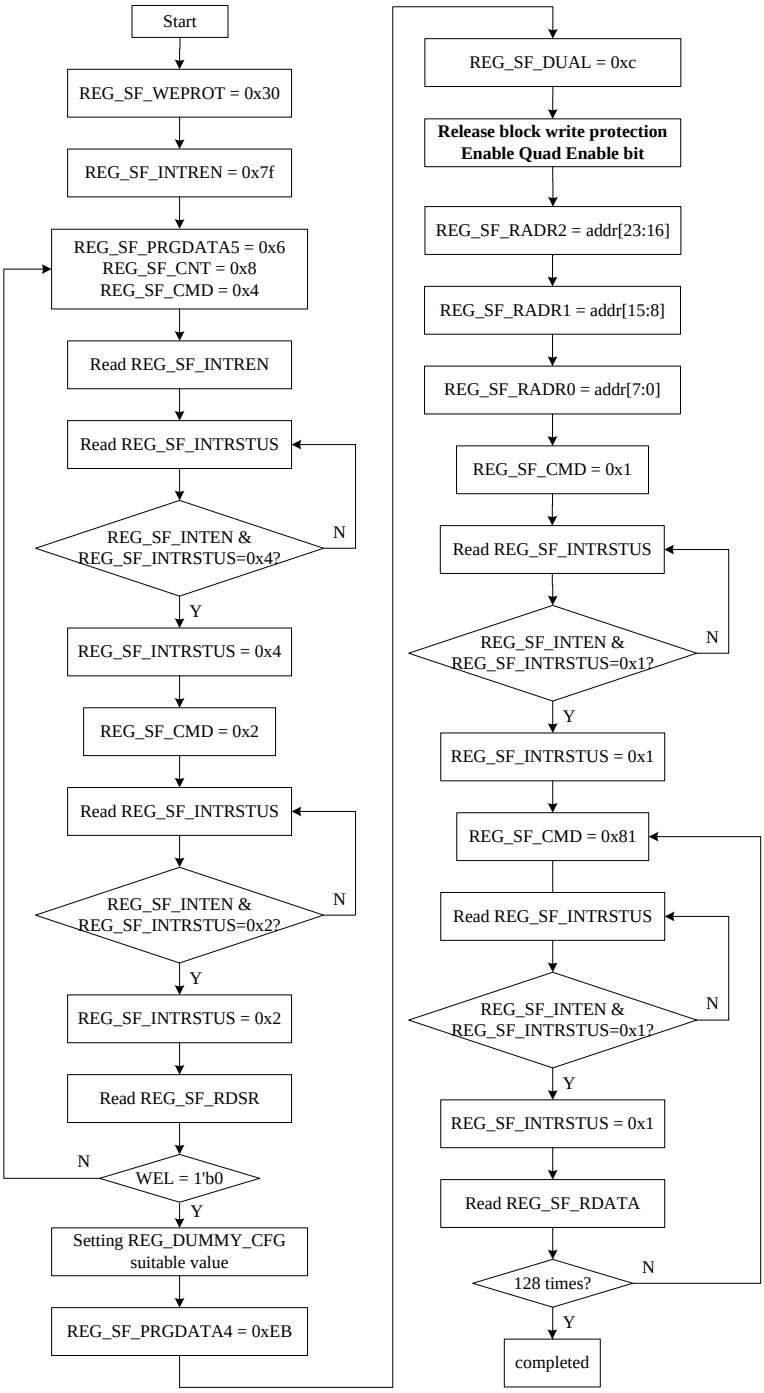

图 24-18 4 x I/O read 串行 NOR Flash 流程

24.3 系统从串行 NOR Flash 启动

当 trapping 设置为串行 NOR Flash 启动模式时，将选择从串行 NOR Flash 启动系统。相关请求信号和地址信号（基低址为 0x6000_0000）有效，则可以从串行 NOR Flash 中读出相应的数据。

24.4 寄存器定义

表 24-2 串行 NOR Flash 相关的寄存器映像

串行 Nor Flash 控制器基地址：（+40010000h）

地址	名称	宽度	寄存器功能
000	REG_SF_CMD	8	串行 NOR Flash 命令寄存器
004	REG_SF_CNT	8	通过 PRG 命令传输的位数
008	REG_SF_RDSR	8	通过 RDSR 命令回读状态寄存器
00C	REG_SF_RDATA	8	通过 RD 命令回读 Flash 数据
010	REG_SF_RADR0	8	读命令或写命令的读/写地址
014	REG_SF_RADR1	8	读命令或写命令的读/写地址
018	REG_SF_RADR2	8	读命令或写命令的读/写地址
01C	REG_SF_WDATA	8	写命令使用的串行 NORFlash 写数据
020	REG_SF_PRGDATA0	8	PRG 命令使用的串行 NOR Flash 编程移位数据
024	REG_SF_PRGDATA1	8	PRG 命令使用的串行 NORFlash 编程移位数据
028	REG_SF_PRGDATA2	8	PRG 命令使用的串行 NORFlash 编程移位数据
02C	REG_SF_PRGDATA3	8	PRG 命令使用的串行 NORFlash 编程移位数据
030	REG_SF_PRGDATA4	8	PRG 命令使用的串行 NORFlash 编程移位数据
034	REG_SF_PRGDATA5	8	PRG 命令使用的串行 NORFlash 编程移位数据
038	REG_SF_SHREG0	8	串行 NOR Flash 接口移位寄存器，仅用于 debug
03C	REG_SF_SHREG1	8	串行 NOR Flash 接口移位寄存器，仅用于 debug
040	REG_SF_SHREG2	8	串行 NOR Flash 接口移位寄存器，仅用于 debug
060	REG_SF_CFG1	8	模块配置寄存器 1
064	REG_SF_CFG2	8	模块配置寄存器 2
088	REG_QSPI_CFG	8	QSPI 模式配置
08c	REG_SF_PRGDATA6	8	PRG 命令使用的串行 NOR Flash 编程移位数据
098	REG_SF_PP_DW_DATA	32	串行 NOR Flash 页编程数据寄存器
0A8	REG_SF_INTRSTUS	8	中断寄存器
0AC	REG_SF_INTREN	8	中断使能寄存器
0B4	REG_SF_CFG3	8	模块配置寄存器 3
0B8	REG_FL_CHKSUM_CTL	8	串行 NOR Flash 校验和控制寄存器
0BC	REG_FL_CHKSUM	32	校验和输出寄存器
0C0	REG_SF_AAICMD	8	AAI 编程命令使能寄存器
0C4	REG_SF_WRPROT	8	写命令使能寄存器
0C8	REG_SF_RADR3	8	读命令或写命令的读/写地址
0CC	REG_SF_DUAL	8	串行 NOR Flash 双模式配置寄存
0E0	REG_DUMMY_CFG	8	为 QSP 读命令配置虚拟周期
0E4	REG_DUMMY_CFG2	8	为 QSPI 读命令配置虚拟周期 2

表 24-3 串行 NOR Flash 相关的寄存器定义

000　　REG_SF_CMD　串行 NOR Flash 命令寄存器　　00

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									AINC		WRSR	WR	ERASE	PRG	RDSR	RD
类型									RW		RW	RW	A0	A0	A0	A0
复位									0		0	0	0	0	0	0

位	助记符	名称	说明
7	**AINC**	AUTO_INCR	**在触发 RD 或 WR 命令后，RADR 将自动加 1** 0： 地址不会自动增加 1：地址自动增加
5	**WRSR**	WRSR_CMD	**写状态寄存器，此命令适用于 ST，SST 和兼容的命令集 Flash，但不适用于 ATMEL** 0： 没有写串行 NOR Flash 状态寄存器命令 1：写串行 NOR Flash 状态寄存器命令
4	**WR**	WR_CMD	**写 1 触发单字节写入。必须在 WDATA 上准备写数据，且必须在触发前在 RADR 准备写地址。完成后，该位将自动清零。此命令适用于 ST，SST 和兼容的命令集 Flash，但不适用于 ATMEL。** 设置 buf2wr_en（CFG2[0]）将进入页编程模式。在该模式下，该位为触发位。 0：没有触发写数据命令 1：触发写数据命令
3	**ERASE**	ERASE_CMD	**完成后，该位将自动清零。此命令适用于 ST 和兼容的命令集 Flash，但不适用于 SST 和 ATMEL。** 0： 没有触发整片擦除 1：触发整片擦除命令
2	**PRG**	PRG_CMD	**必须在 REG_SF_CNT 准备程序位计数，并且必须在触发前在 PRGDATA0~5 准备程序数据。完成后，该位将自动清零。此命令适用于所有串行 NOR Flash。** 0： 没有触发编程数据 1： 触发用户编程命令
1	**RDSR**	RDSR_CMD	**该状态寄存器将在 RDSR 寄存器中出现。该命令适用于 ST, SST, ATMEL 和兼容的命令集串行 NOR Flash。** 0：没有触发读状态寄存器 1：触发读状态寄存器命令
0	**RD**	RD	**必须在触发前在 RADR 准备读地址。 读地址会在 RDATA 寄存器中出现。该命令适用于 ST, SST, ATME 和兼容的命令集 Flash。** 0：没有触发读命令 1： 触发读数据命令

004 **REG_SF_CNT** **通过 PRG 命令传输的位数** **00**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称										**CNT**						
类型										RW						
复位										0	0	0	0	0	0	0

位	助记符	名称	说明
6：0	**CNT**	SF_CNT	**通过 PRG 命令传输的位数，最大 48 位**

008 **REG_SF_RDSR** **通过 RDSR 命令回读状态寄存器** **00**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									**RDSRDAT**							
类型									RU							
复位									0	0	0	0	0	0	0	0

位	助记符	名称	说明
7：0	**RDSRDAT**	RDSR_DATA	**通过 RDSR 命令回读状态寄存器**

00C **REG_SF_RDATA** 通过 RD 命令回读串行 NOR Flash 数据 00

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									SFRDAT							
类型									RU							
复位									0	0	0	0	0	0	0	0

位	助记符	名称	说明
7：0	**SFRDAT**	SF_RDATA	**通过 RD 命令回读串行 NOR Flash 数据**

010 **REG_SF_RADR0** 读命令或写命令的读/写地址 00

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									SFPADR0							
类型									RW							
复位									0	0	0	0	0	0	0	0

位	助记符	名称	说明
7：0	**SFPADR0**	SFP_ADR0	**读命令或写命令的读/写地址**

014 **REG_SF_RADR1** 读命令或写命令的读/写地址 00

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									SFPADR1							
类型									RW							
复位									0	0	0	0	0	0	0	0

位	助记符	名称	说明
7：0	**SFPADR1**	SFP_ADR1	**读命令或写命令的读/写地址**

018 **REG_SF_RADR2** 读命令或写命令的读/写地址 00

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									SFPADR2							
类型									RW							
复位									0	0	0	0	0	0	0	0

位	助记符	名称	说明
7：0	**SFPADR2**	SFP_ADR2	**读命令或写命令的读/写地址**

01C **REG_SF_WDATA** 写命令使用的串行 NOR Flash 写数据 00

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									SFWRDAT							
类型									RW							
复位									0	0	0	0	0	0	0	0

位	助记符	名称	说明
7：0	**SFWRDAT**	SFP_WR_DATA	**写命令使用的串行 NOR Flash 写数据**

020 **REG_SF_PRGDATA0** **PRG 命令使用的串行 NOR Flash 编程移位数据** **00**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									PRGDATA0							
类型									RW							
复位									0	0	0	0	0	0	0	0

位	助记符	名称	说明
7：0	**PRGDATA0**	SFP_PRGDATA0	**PRG 命令使用的串行 NOR Flash 编程移位数据。首先移位 PRGDATA6，最后移位 PRGDATA0。在每个字节移位中，首先移位 MSB。**

024 **REG_SF_PRGDATA1** **PRG 命令使用的串行 NOR Flash 编程移位数据** **00**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									PRGDATA1							
类型									RW							
复位									0	0	0	0	0	0	0	0

位	助记符	名称	说明
7：0	**PRGDATA1**	SFP_PRGDATA1	**PRG 命令使用的串行 NOR Flash 编程移位数据。首先移位 PRGDATA6，最后移位 PRGDATA0。在每个字节移位中，首先移位 MSB。**

028 **REG_SF_PRGDATA2** **PRG 命令使用的串行 Flash 编程移位数据** **00**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									PRGDATA2							
类型									RW							
复位									0	0	0	0	0	0	0	0

位	助记符	名称	说明
7：0	**PRGDATA2**	SFP_PRGDATA2	**PRG 命令使用的串行 NOR Flash 编程移位数据。首先移位 PRGDATA6，最后移位 PRGDATA0。在每个字节移位中，首先移位 MSB。**

02C **REG_SF_PRGDATA3** **PRG 命令使用的串行 Flash 编程移位数据** **00**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									PRGDATA3							
类型									RW							
复位									0	0	0	0	0	0	0	0

位	助记符	名称	说明
7：0	**PRGDATA3**	SFP_PRGDATA3	**PRG 命令使用的串行 NOR Flash 编程移位数据。首先移位 PRGDATA6，最后移位 PRGDATA0。在每个字节移位中，首先移位 MSB。**

030 **REG_SF_PRGDATA4** **PRG 命令使用的串行 NOR Flash 编程移位数据** **00**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									PRGDATA4							
类型									RW							
复位									0	0	0	0	0	0	0	0

位	助记符	名称	说明
7：0	**PRGDATA4**	SFP_PRGDATA4	**PRG 命令使用的串行 NOR Flash 编程移位数据。首先移位 PRGDATA6，最后移位 PRGDATA0。在每个字节移位中，首先移位 MSB。**

034 **REG_SF_PRGDATA5** **PRG 命令使用的串行 NOR Flash 编程移位数据** **00**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									PRGDATA5							
类型									RW							
复位									0	0	0	0	0	0	0	0

位	助记符	名称	说明
7：0	**PRGDATA5**	SFP_PRGDATA5	**PRG 命令使用的串行 NOR Flash 编程移位数据。首先移位 PRGDATA6，最后移位 PRGDATA0。在每个字节移位中，首先移位 MSB。**

038 **REG_SF_SHREG0** **串行 NOR Flash 接口移位寄存器，仅用于 debug** **00**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									SHREG0							
类型									RU							
复位									0	0	0	0	0	0	0	0

位	助记符	名称	说明
7：0	**SHREG0**	SHIFT_REG0	**串行 NOR Flash 接口移位寄存器，仅用于 debug**

03C **REG_SF_SHREG1** **串行 NOR Flash 接口移位寄存器，仅用于 debug** **00**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									SHREG1							
类型									RU							
复位									0	0	0	0	0	0	0	0

位	助记符	名称	说明
7：0	**SHREG1**	SHIFT_REG1	**串行 NOR Flash 接口移位寄存器，仅用于 debug**

040 **REG_SF_SHREG2** **串行 NOR Flash 接口移位寄存器，仅用于 debug** **00**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									SHREG2							
类型									RU							
复位									0	0	0	0	0	0	0	0

位	助记符	名称	说明
7：0	**SHREG2**	SHIFT_REG2	**串行 NOR Flash 接口移位寄存器，仅用于 debug**

060 **REG_SF_CFG1** **模块配置寄存器 1** **00**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									**FL_MON_SEL**							**FR**
类型									RW							RW
复位									0	0						0

位	助记符	名称	说明
7：6	**FL_MON_SEL**	FL_MON_SEL	**串行 Nor flash 控制器监空信号选择** 00： nor flash 控制状态 0 01： nor flash 控制状态 1 10： nor flash 控制状态 11： nor flash 控制状态 3
0	**FR**	FAST_READ	**快速读** 0： 正常读命令 1： 支持 ST 快速读命令（读命令 0Bh，及 1 个虚拟字节）

064 **REG_SF_CFG2** **模块配置寄存器 2** **00**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称										**AAICFG**	**PRGWRSR**	**PRGWREN**			**RS232WRBUFEN**	**BUF2WREN**
类型										RW	**RW**	RW			RW	RW
复位										0	**0**	0			0	0

位	助记符	名称	说明
6	**AAICFG**	AAI_CFG	**地址自动增加命令配置** 0： AAI 命令为 0xAD 1： AAI 命令为 0xAF
5	**PRGWRSR**	PRG_WRSR_OPCODE_EN	**编程操作代码使能** 0： 使用 0x01 作为写状态寄存器命令操作码 1： 使用 PRGDATA6 作为写状态寄存器命令操作码
4	**PRGWREN**	PRG_WR_OPCODE_EN	**编程操作代码使能** 0： 使用 0x02 作为写命令操作码 1： 使用 PRGDATA0 作为写命令操作码
1	**RS232WRBUFEN**	RS232WRBUF_EN	**内部缓冲区数据输入选择** 0： 来自 RISC 的内部缓冲区输入数据 1：来自 RS232 的内部缓冲区输入数据
0	**BUF2WREN**	BUF2WR_EN_SET	**设置用于 Flash 读取的预取缓冲区，用于页编程。保证在预取缓冲区空闲之前不会切换数据路径。** 0： 预取缓冲区用于读 1：预取缓冲区用于页编程

088 **REG_QSPI_CFG** **配置寄存器** **00**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称													**prg_rdsr_dis**	**prg_data_quad**	**prg_addr_quad**	**rdsr_bypass**
类型													R/W	R/W	R/W	R/W
复位													0	0	0	0

位	助记符	名称	说明
3	**PROGRAM**	prg_rdsr_dis	**在页编程后禁用 rdsr 命令** 0：无操作 1：在页编程后禁用 rdsr 命令
2	**PROGRAM**	prg_data_quad	**使能 qspi 页编程模式** 0：无操作 1：Qspi 页编程
1	**PROGRAM**	prg_addr_quad	**使能 qspi 页编程模式（地址）.** 0：无操作 1：Qspi 页编程（地址）
0	**RDSR**	rdsr_bypass	**在 wrsr 命令后，旁路 rdsr 命令** 0：无操作 1：在 wrsr 命令后，旁路 rdsr 命令

08C **REG_SF_PRGDATA6** **PRG 命令使用的串行 NOR Flash 编程移位数据** **00**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									**PRGDATA6**							
类型									RW							
复位									0	0	0	0	0	0	0	0

位	助记符	名称	说明
7：0	**PRGDATA5**	SFP_PRGDATA6	**PRG 命令使用的串行 NOR Flash 编程移位数据。首先移位 PRGDATA6，最后移位 PRGDATA0。在每个字节移位中，首先移位 MSB。**

098 **REG_SF_PP_DW_DATA** **串行 NOR Flash 页编程数据寄存器** **0**

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

位	助记符	名称	说明
31：0	**PP_DW_DATA**	REG_SF_PP_DW_DATA	**Flash 页编程数据寄存器**

0A8 **REG_SF_INTRSTUS** **中断寄存器** **0**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称										AAIINT	WSRINT	WRINT	ERSINT	PRGINT	RSRINT	RDINT
类型										W1C	W1C	W1C	W1C	W1C	W1C	W1C
复位										0	0	0	0	0	0	0

位	助记符	名称	说明
6	**AAIINT**	AAI_INT	**AAI 编程完成中断，写 1 时清除该位** 0： no AAI interrupt 1： AAI interrupt occur
5	**WSRINT**	WRSR_INT	**写状态寄存器完成中断，写 1 时清除该位** 0： 非写状态寄存器，没有中断 1： 写状态寄存器完成中断
4	**WRINT**	WR_INT	**写/页编程完成中断，写 1 时清除该位** 0： 没有写/编程中断 1： 写/编程完成中断
3	**ERSINT**	ERASE_INT	**整片擦除中断，写 1 时清除该位** 0： 没有擦除中断 1： 擦除完成中断
2	**PRGINT**	PRG_INT	**可编程操作码完成中断，写 1 时清除该位** 0：无编程操作码完成中断 1： 可编程操作码完成中断
1	**RSRINT**	RDSR_INT	**读状态完成中断，写 1 时清除该位** 0：无读状态完成中断 1：读状态完成中断
0	**RDINT**	RD_INT	**读操作完成中断，写 1 时清除该位** 0：无读操作完成中断 1：读操作完成中断

0AC **REG_SF_INTREN** **中断使能寄存器** **0**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称										AAIINTEN	WSRINTEN	WRINTEN	ERSINTEN	PRGINTEN	RSRINTEN	RDINTEN
类型										RW	RW	RW	RW	RW	RW	RW
复位										0	0	0	0	0	0	0

位	助记符	名称	说明
6	**AAIINTEN**	AAI_INTREN	**AAI 使能中断控制位** 0： AAI 中断禁用 1： AAI 中断使能
5	**WSRINTEN**	WRSR_INTREN	**WRSR 使能中断控制位** 0： WRSR 中断禁用 1： WRSR 中断使能
4	**WRINTEN**	WR_INTREN	**WR 使能中断控制位** 0： WR 中断禁用 1： WR 中断使能
3	**ERSINTEN**	ERASE_INTREN	**ERASE 使能中断控制位** 0： ERASE 中断禁用 1： ERASE 中断使能
2	**PRGINTEN**	PRG_INTREN	**PRG 使能中断控制位** 0： PRG 中断禁用

位	助记符	名称	说明
			1： PRG 中断使能
1	**RSRINTEN**	RDSR_INTREN	**RDSR 使能中断控制位** 0： RDSR 中断禁用 1： RDSR 中断使能
0	**RDINTEN**	RD_INTREN	**RD 使能中断控制位** 0： RD 中断禁用 1： RD 中断使能

0B4 **REG_SF_CFG3** **模块控制寄存器 3** **0**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									**WRDI**	**PRGWEOPEN**	**POLRDYDIS**	**PRGRDOPEN**	**RDYVLUE**	**STUSPOS**		
类型									RW	RW	RW	RW	RW	RW		
复位									0	0	0	0	0	0	0	0

位	助记符	名称	说明
7	**WRDI**	WREN_DIS	**串行 NOR Flash 写使能控制位** 0： 发送写使能命令 1： 跳过写使能状态
6	**PRGWEOPEN**	PRG_WREN_OPCODE_EN	**写使能 命令操作码配置** 0： 使用 0x06 作为写使能命令操作码 1： 使用 PRGDATA2 作为写使能命令操作码
5	**POLRDYDIS**	POL_RDY_BIT_DIS	**禁用轮询就绪位，禁用配置位** 0： 轮询状态寄存器 1： 跳过轮询状态
4	**PRGRDOPEN**	PRG_RD_OPCODE_EN	**配置写状态命令操作码** 0： 使用 0x05 作为读状态命令操作码，如果 flash 标记为 0，否则为 0xd7 1： 使用 PRGDATA1 作为读状态命令操作码
3	**RDYVLUE**	RDY_VLUE	**就绪时，Serial flash 输出位值** 0： 等待轮询位，直到它变为逻辑 0 1：等待轮询位，直到它变为逻辑 1
2：0	**STUSPOS**	STUS_POS	**7~0 状态寄存器哪个位轮询**

0B8 **REG_FL_CHKSUM_CTL** **串行 NOR Flash 校验和控制寄存器** **0**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称											**RS232CHKSUMS**	**RS232CHKSUME**				
类型											RW	RW				
复位											0	0				

位	助记符	名称	说明
5	**RS232CHKSUMS**	RS232_CHKSUM_START	**RS232 校验和配置寄存器** 0： RS232 校验和未开始 1： RS232 校验开始
4	**RS232CHK**	RS232_CHECKSU	**RS232 校验和配置寄存器**

位	助记符	名称	说明
	SUME	M_MODE	0： RS232 校验和模式禁用 1： RS232 校验和模式使能

0BC **REG_FL_CHKSUM** **校验和输出寄存器** **0**

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
名称	**CHKSUM[31：16]**															
类型	RU															
复位	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	**CHKSUM[15：0]**															
类型	RU															
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	助记符	名称	说明
31：0	**CHKSUM**	CHKSUM	**校验和输出数据**

0C0 **REG_SF_AAICMD** **AAI 编程命令使能寄存器** **0**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称																**AAIWR**
类型																A0
复位																0

位	助记符	名称	说明
0	**AAIWR**	AAI_WR_CMD	**向该寄存器写入 1 将使能 AAI（自动地址递增）编程过程。必须在 RADR（RADR3，RADR2，RADR1，RADR0）准备写地址，并在触发前设置合适的 AAI_CFG（CFG2 [6]）。完成后，该位将自动清零。必须为 AAI 编程设置 buf2wr_en（CFG2 [0]）。** 0： AAI 过程未开始 1： 触发 AAI 过程

0C4 **REG_SF_WRPROT** **写命令使能寄存器** **85**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									**WRPROT**							
类型									RW							
复位									1	0	0	0	0	1	0	1

位	助记符	名称	说明
7：0	**WRPROT**	WRITE_PROTECT	**将此字节置为 0x30 会关闭写保护，使能 CMD 和 AAI_CMD 中的 SF 命令**

0C8 **REG_SF_RADR3** **读命令或写命令的读/写地址** **0**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									**SFPADR3**							
类型									RW							
复位									0	0	0	0	0	0	0	0

位	助记符	名称	说明

位	助记符	名称	说明
7：0	**SFPADR3**	SFP_ADR3	**读命令或写命令的读/写地址**

0CC **REG_SF_DUAL** **串行 NOR Flash 双模式配置寄存器** **00**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称													**ADRQUAD**	**DUALRDEN**	**ADRDUAL**	**DUALRDEN**
类型													RW	RW	RW	RW
复位													0	0	0	0

位	助记符	名称	说明
3	**ADRQUAD**	**ADDR_QUAD**	**四地址模式** 0： 四地址模式禁用 1： 四地址模式使能
2	**DUALRDEN**	QUAD_READ_EN	**在上一次读取操作完成后，将使能 Quad 读取模式。Quad 读取命令必须在 PRGDATA4 中写入，读取数据可以在 RDATA 中获得。** 0： quad 读取模式禁用 1： quad 读取模式使能
1	**ADRDUAL**	ADDR_DUAL	**双地址模式** 0：双地址模式禁用 1：双地址模式使能
0	**DUALRDEN**	DUAL_READ_EN	**在上一次读取操作完成后，将使能 Dual 读取模式。dual 读取命令必须在 PRGDATA3 中写入，读取数据可以在 RDATA 中获得。** 0： dual 读模式禁用 1： dual 读模式使能

0E0 **REG_DUMMY_CFG** **串行 NOR Flash dummy cycle 配置寄存器** **00**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称									**dummy_cfg**							
类型									RW							
复位									0							

位	助记符	名称	说明
7：0	**DUMMY_CFG**	dummy_cfg	**QSPI 读操作 dummy cycle 配置**

0E4 **REG_DUMMY_CFG2** **串行 NOR Flash dummy cycle 配置寄存器 2** **00**

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
名称												**dummy_num_en**	**dummy_num**			
类型												RW	RW			
复位												0	0			

位	助记符	名称	说明
4	**DUMMY_NUM_EN**	dummy_num_en	**dummy cycle 时钟周期数配置使能** 0： 禁用 1： 使能
3：0	**DUMMY_NUM**	dummy_num	**dummy cycle 数**

25 缩略语

表 25-1 缩略语

名称	英文全称	中文全称
SYSPLL	System PLL	系统时钟
CKGEN	Clock Generator	时钟产生器
LVD	Low Voltage Detect	低电压检测
PVD	Programmble Voltage Detect	可编程电压检测
XOSC	External Crytal Oscillator	晶体振荡器
OSC	Oscillator	振荡器
SPM	System Power Manager	系统电源管理
POR	Power On Reset	上电复位
HSI	High Speed Clock Internal	高速内部时钟
HSE	High Speed Clock External	高速外部时钟
LSI	Low Speed Clock Internal	低速内部时钟
VCO	Voltage Controlled Oscillator	压控振荡器
PREDIV	Previous Divider	前置除法器
PLL	Phase Locked Loop	锁相环
FBKDIV	Feedback Divider	反馈除法器
POSDIV	Post Divider	预置除法器
AHB	Advanced High Performance Bus	高级高性能总线
APB	Advanced Peripheral Bus	高级外设总线
ICode	Instruction Code	指令代码
DCode	Data Code	数据代码
I-Cache	Instruction Cache	指令缓存
D-Cache	Data Cache	数据缓存
LRU	Least Recently Used	最少最近使用
ISP	In-System Programming	在系统编程
NMI	Non Maskable Interrupt	不可屏蔽中断
SRAM	Static Random-Access Memory	静态随机存取存储器
AAI	Auto Address Increment	地址自增模式