HK32F39AxCxDxE 用户手册

版本：2.1

发布日期：2023-12-28

深圳市航顺芯片技术研发有限公司

http://www.hsxp-hk.com

前言

编写目的

本文档介绍了 HK32F39A 系列芯片的功能框图、存储器映射、Flash、中断和事件等功能以及各功能模块的寄存器描述，旨在帮助用户快速开发 HK32F39A 的应用及产品。

读者对象

本文适用于以下读者：

- 开发工程师
- 芯片测试工程师

版本说明

本文档对应的产品系列为 HK32F39A 系列芯片。

修订记录

版本	日期	修订内容
0.2.2	2019/09/02	预发布
1.0	2022/01/17	首次发布
1.1	2022/01/28	更新了“10.3 复用功能 I/O 和调试配置”和“19.2 RTC 功能描述”、新增了“15.5.6 TFT-LCD 驱动”
1.2	2022/03/01	1. 更新了时钟树图中的 MCO 时钟源细节部分。 2. 将原来的章节“温度传感器和内部参考电压”，分成两个章节：“温度传感器”和“内部参考电压”；并各自增加了“温度传感器校准值”以及“内部电压基准测量值”的表格，还对计算公式及变量做了说明。
1.3	2022/03/30	更正了全文的细小错误。
1.4	2022/05/10	1. 更正了“22 USB 全速设备接口（USB）”章节中的错误。 2. 更正了全文的其他小错误，如 WWDG 章节。
1.5	2022/09/22	1. 更新了“7.3.7 BKP_RAM 寄存器”章节。 2. 更新了定时器“捕获/比较通道”章节的图，以及“定时器同步”章节。 3. 更正了其他的细小错误。
1.6	2023/06/21	更正了全文描述中的小错误。
1.7	2023/09/11	更正了“7.3.8 BKP_RAM 寄存器…”章节中的偏移地址的值。
1.8	2023/10/30	1. 优化了“9.2 外部中断/事件控制器（EXTI）”的内容，将内部和外部事件分别更改为固定和可配事件；并优化了“9.2.6 外部中断/事件线映射”中的中断事件映射关系表。

版本	日期	修订内容
		2. 调整了“3.2 Flash 功能描述”的内容组织结构。 3. 更新了“8.3.15 PLL 控制寄存器（RCC_PLLCTL）”。 4. 在“10 通用和复用功能 I/O（GPIO 和 AFIO）”章节的多图添加了 V_{DD_FT} 的上标。 5. 更新了“10.3.8 USART 复用功能重映射”和“10.3.10 SPI/I2S 复用功能重映射”中的重映射表，修改了 AFIO_MAPR.SPI1_REMAP 位的描述和 AFIO_MAPR2.USART2_REMAP 位的定义。 6. 更新了“图 26-1 USART 框图”。 7. 更新了“24.2.1 SPI 特性”中可配置的数据包长度。 8. 更新了“图 31-8 根据数据类型构建 128 位块”的文字描述。 9. 更新了多处关于寄存器的基地址以及多个位域的描述。
1.9	2023/11/15	1. 删除了“10.4.2 复用重映射和调试 I/O 配置寄存器（AFIO_MAPR）”中的 CAN2_REMAP 位，在“10.4.7 复用重映射和调试 I/O 配置寄存器 2（AFIO_MAPR2）”中添加了 CAN2_REMAP 位。 2. 在“27.2.6 QSPI 内存映射模式”中添加了 QSPI 接口管理的内存地址范围。
2.0	2023/11/20	1. 删除了 TRNG 寄存器中的 IE 位和 SEIS 位，以及 TRNG 中断功能。 2. 添加了“10.3.11 AI 复用功能重映射”章节和 SAI_AB_REMAP 寄存器位。 3. 更新了 LATENCY 位最大取值对应的等待周期数。
2.1	2023/12/28	1. 更新了章节“2.2.1 存储器映射”中“USB/CAN1 共享的 512 字节 SRAM”的结束地址。 2. 更新了章节“22.5.3 缓冲区描述表”中空间大小为：“0x200”（原为 0x400）。

目 录

1 简介

本文档为 HK32F39A 系列芯片的用户手册。HK32F39A 系列芯片是由深圳市航顺芯片技术研发有限公司研发的高性能 MCU 芯片，包括以下型号：

HK32F39ARxT6（LQFP64 封装）：

- HK32F39ARCT6
- HK32F39ARDT6
- HK32F39ARET6

HK32F39AVxT6（LQFP100 封装）：

- HK32F39AVCT6
- HK32F39AVDT6
- HK32F39AVET6

在阅读本手册前，你可能需要参考以下文档：

表 1-1 参考文档

名称	说明
HK32F39A 数据手册	航顺 HK32F39A MCU 的数据手册，描述了该 MCU 的外设接口、电气特性、引脚封装等。 下载地址：http://www.hsxp-hk.com
Cortex®-M3 技术参考手册	ARM 公司的 Cortex®-M3 产品的技术手册。 下载地址：www.arm.com

说明：本手册中缩略语的释义请参考章节：“37 缩略语与术语”。

2 存储器和总线构架

2.1 系统构架

主系统由以下部分构成：

- 驱动单元：
  - Cortex®-M3 内核 DCode 总线（D-bus）和系统总线（S-bus）
  - 通用 DMA1 和 DMA2
- 被动单元：
  - 内部 Flash 存储器
  - 内部 SRAM
  - AHB Lite 外设设备
  - AHB 到 APB 的桥（APB1 和 APB2），用于连接所有的 APB 外设
  - FSMC
  - QSPI

这些都是通过一个多级的 AHB 总线构架相互连接的，如下图所示。现以 HK32F39AVET6 为例，示出了 HK32F39A MCU 的系统框图：

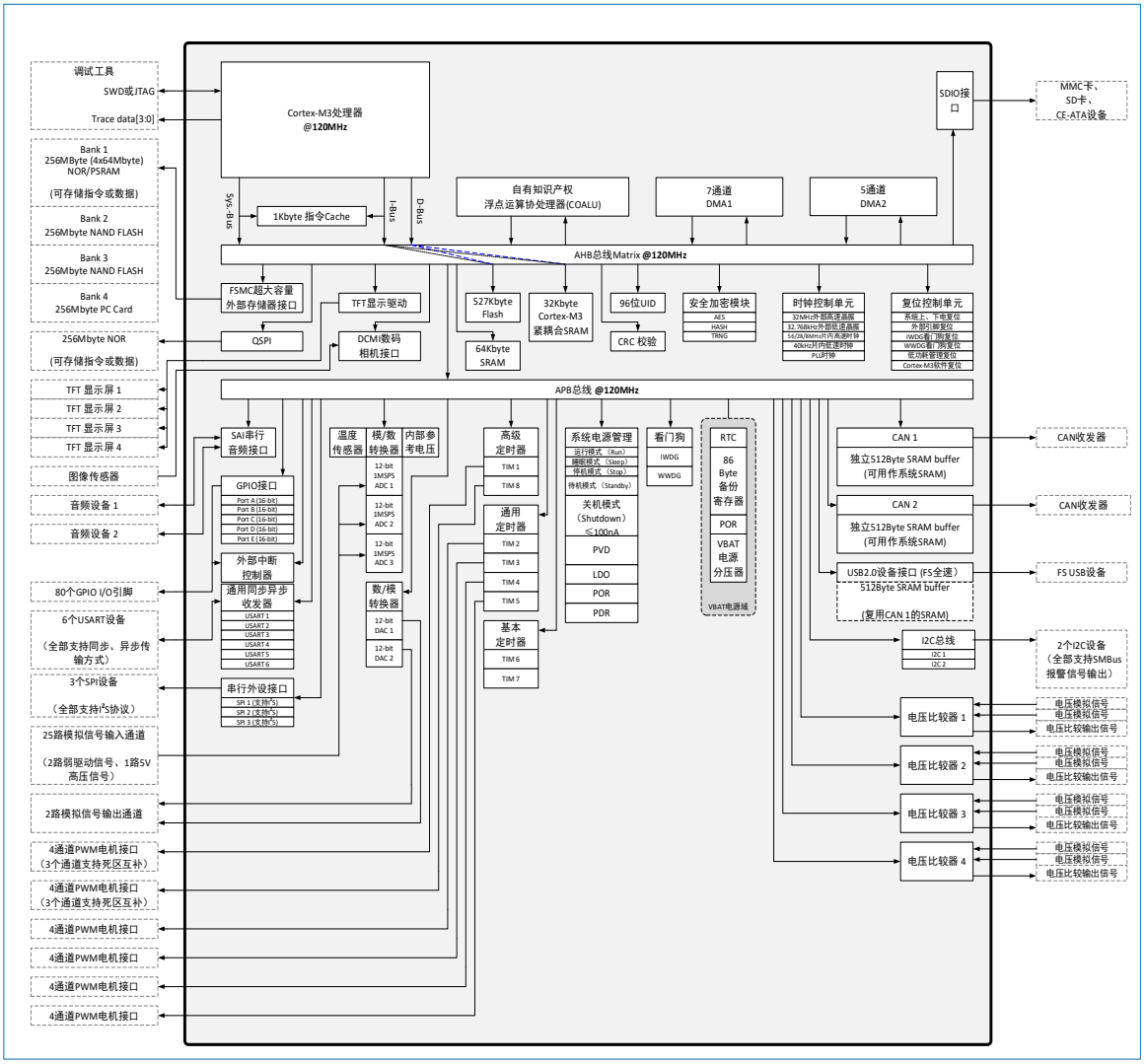


图 2-1 HK32F39AVET6 系统框图

2.1.1.1 指令总线（I-BUS）

I-BUS 总线宽度 32 位，它将 Cortex®-M3 内核的指令总线连接到 AHB 总线矩阵。CPU 通过 I-BUS 从代码段取指。I-BUS 的地址映射范围是 0 ~ 0x2000 0000。此总线的操作对象是内部 Flash 和 CCM SRAM。

2.1.1.2 数据总线（D-BUS）

D-BUS 总线宽度 32 位，它将 Cortex®-M3 内核的数据总线连接到 AHB 总线矩阵。CPU 通过 D-BUS 访问代码段数据，CPU 的立即数加载操作和调试访问操作在此总线上完成，D-BUS 的地址映射范围是 0 ~ 0x2000 0000。这个总线的操作对象是内部 Flash 和 CCM SRAM。

2.1.1.3 系统总线（SYS -BUS）

SYS-BUS 总线宽度 32 位，它将 Cortex®-M3 内核的系统总线连接到 AHB 总线矩阵。CPU 内存访问操作和对所有外设的操作在此总线上完成，SYS-BUS 的地址映射范围是 0x2000 0000 ~ 0xDFFF FFFF 和 0xE010 0000 ~ 0xFFFF FFFF，也可通过此总线获取指令（效率低于 I-BUS）。这个总线的操作对象是内部 64KB SRAM、AHB 总线外设、APB1/2 总线外设、QSPI 和 FSMC 的外部储存器。

2.1.1.4 Cortex®-M3 PPB 总线

PPB 总线拥有 32 位总线宽度，CPU 调试特性在此总线上完成，PPB 的地址映射范围是 0xE000 0000 ~ 0xE010 0000，这个总线的操作对象是 CM3 内核调试部分。

2.1.1.5 DMA 总线

DMA 总线拥有最大 32 位总线宽度，它将 DMA 总线连接到 AHB 总线矩阵，DMA 通过此总线来执行存储器数据的传入和传出。这个总线的操作对象是内部 Flash、内部 64KB SRAM、AHB 总线矩阵的所有外设，QSPI 和通过 FSMC 的外部储存器。

2.1.1.6 总线矩阵

总线矩阵管理内核系统总线和 DMA 主控总线之间的访问仲裁，仲裁利用轮换算法。总线矩阵包含 4 个驱动单元（CPU 的 DCode 总线、系统总线、DMA1 总线和 DMA2 总线）和 4 个被动部件（内部 Flash 存储器、SRAM、AHB-APB 桥、FSMC）。

AHB 外设通过总线矩阵与系统总线相连，以允许 DMA 访问。

总线矩阵如下图所示：

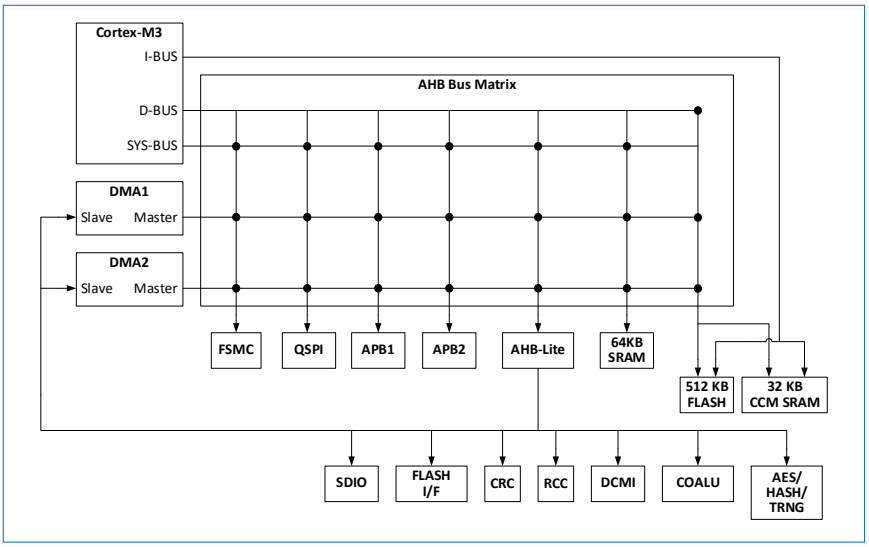


图 2-2 总线矩阵图

2.1.1.7 AHB/APB 桥

两个 AHB/APB 桥在 AHB 和 2 个 APB 总线间提供同步连接。APB1 工作频率限速最高 60 MHz，APB2 工作于全速（最高为 120 MHz）。

若需了解连接到各个桥的外设地址映射，请参见表 2-1。

在每次器件复位以后，除了 SRAM 和 FLITF（Flash 存储器接口）以外的所有外设都被关闭。在使用某个外设之前，必须设置 RCC_AHBENRx、RCC_APB1ENRx 和 RCC_APB2ENRx 寄存器以使能该外设的时钟。

2.2 存储器组织

程序存储器、数据存储器、寄存器和输入输出端口被组织在同一个 4 Gbyte 的线性地址空间内。数据字节以小端格式存放在存储器中。一个字里的最低地址字节被认为是该字的最低有效字节，而最高地址字节是最高有效字节。

外设寄存器的映射请参考相关章节。可访问的存储器空间被分成 8 个主要块，每个块为 512 Mbyte。

其他所有没有分配给片上存储器和外设的存储器空间都是保留的地址空间，请参考器件的数据手册中的存储器映射图。

2.2.1 存储器映射

请参考器件的数据手册中的存储器映射图。

下表列出了器件内置外设的地址分配。

表 2-1 寄存器组地址分配

起始地址- 结束地址	外设	总线
0xA000 1400 - 0xDFFF FFFF	保留	AHB
0xA000 1000 - 0xAFFF 13FF	QSPI	
0xA000 0000 - 0xA000 0FFF	FSMC	
0x5006 0C00 - 0x5FFF FFFF	保留	
0x5006 0800 - 0x5006 0BFF	TRNG	
0x5006 0400 - 0x5006 07FF	HASH	
0x5006 0000 - 0x5006 03FF	AES	
0x5005 0400 - 0x5005 FFFF	保留	
0x5005 0000 - 0x5005 03FF	DCMI	
0x4003 0400 - 0x5004 FFFF	保留	
0x4003 0000 - 0x4003 03FF	算术运算协处理器（COALU）	
0x4002 3400-0x4002 FFFF	保留	
0x4002 3000-0x4002 33FF	CRC	
0x4002 2400-0x4002 2FFF	保留	

起始地址- 结束地址	外设	总线
0x4002 2000-0x4002 23FF	Cache/Flash 控制寄存器	
0x4002 1400-0x4002 1FFF	保留	
0x4002 1000-0x4002 13FF	复位和时钟控制（RCC）	
0x4002 0800-0x4002 0FFF	保留	
0x4002 0400-0x4002 07FF	DMA2	
0x4002 0000-0x4002 03FF	DMA1	
0x4001 8400-0x4001 FFFF	保留	
0x4001 8000-0x4001 83FF	SDIO	
0x4001 5C00-0x4001 7FFF	保留	APB2
0x4001 5800-0x4001 5BFF	SAI	
0x4001 4800-0x4001 57FF	保留	
0x4001 4400-0x4001 47FF	USART6	
0x4001 4000-0x4001 43FF	COMP1/COMP2/COMP3/COMP4	
0x4001 3C00-0x4001 3FFF	ADC3	
0x4001 3800-0x4001 3BFF	USART1	
0x4001 3400-0x4001 37FF	TIM8	
0x4001 3000-0x4001 33FF	SPI1/I2S1	
0x4001 2C00-0x4001 2FFF	TIM1	
0x4001 2800-0x4001 2BFF	ADC2	
0x4001 2400-0x4001 27FF	ADC1	
0x4001 1C00-0x4001 23FF	保留	
0x4001 1800-0x4001 1BFF	GPIOE	
0x4001 1400-0x4001 17FF	GPIOD	
0x4001 1000-0x4001 13FF	GPIOC	
0X4001 0C00-0x4001 0FFF	GPIOB	
0x4001 0800-0x4001 0BFF	GPIOA	
0x4001 0400-0x4001 07FF	EXTI	
0x4001 0000-0x4001 03FF	AFIO	
0x4000 7A00-0x4000 FFFF	保留	APB1
0x4000 7800-0x4000 79FF	CAN2 访问的 512 字节 SRAM	

起始地址- 结束地址	外设	总线
0x4000 7400-0x4000 77FF	DAC	
0x4000 7000-0x4000 73FF	电源控制（PWR）	
0x4000 6C00-0x4000 6FFF	备份寄存器（BKP）	
0x4000 6800-0x4000 6BFF	CAN2	
0x4000 6400-0x4000 67FF	CAN1	
0x4000 6200-0x4000 63FF	保留	
0x4000 6000-0x4000 61FF	USB/CAN1 共享的 512 字节 SRAM	
0x4000 5C00-0x4000 5FFF	USB 全速设备	
0x4000 5800-0x4000 5BFF	I2C2	
0x4000 5400-0x4000 57FF	I2C1	
0x4000 5000-0x4000 53FF	USART5	
0x4000 4C00-0x4000 4FFF	USART4	
0x4000 4800-0x4000 4BFF	USART3	
0x4000 4400-0x4000 47FF	USART2	
0x4000 4000-0x4000 43FF	保留	
0x4000 3C00-0x4000 3FFF	SPI3/I2S3	
0x4000 3800-0x4000 3BFF	SPI2/I2S2	
0x4000 3400-0x4000 37FF	保留	
0x4000 3000-0x4000 33FF	独立看门狗（IWDG）	
0x4000 2C00-0x4000 2FFF	窗口看门狗（WWDG）	
0x4000 2800-0x4000 2BFF	RTC	
0x4000 1800-0x4000 27FF	保留	
0x4000 1400-0x4000 17FF	TIM7	
0x4000 1000-0x4000 13FF	TIM6	
0x4000 0C00-0x4000 0FFF	TIM5	
0x4000 0800-0x4000 0BFF	TIM4	
0x4000 0400-0x4000 07FF	TIM3	
0x4000 0000-0x4000 03FF	TIM2	

2.2.2 嵌入式 SRAM

器件内置 64 Kbyte 的静态 SRAM。它可以以字节、半字（16 位）或全字（32 位）访问。SRAM 的起

始地址是 0x2000 0000。

2.2.3 内核耦合存储器（CCM SRAM）

器件内置 32 Kbyte 的 CCM SRAM。它可以以字节（8 位）、半字（16 位）或全字（32 位）访问。CCM SRAM 的逻辑地址空间：0x1000 0000~0x1000 7FFF。

2.2.4 位段

Cortex®-M3 存储器映射包括两个位段（Bit-band）区。这两个位段区中的每个比特与各自别名区（Alias）中的特定地址某字的比特 0 一一对应。通过这种方式，提供了一种独立位操作（读/写）模式。

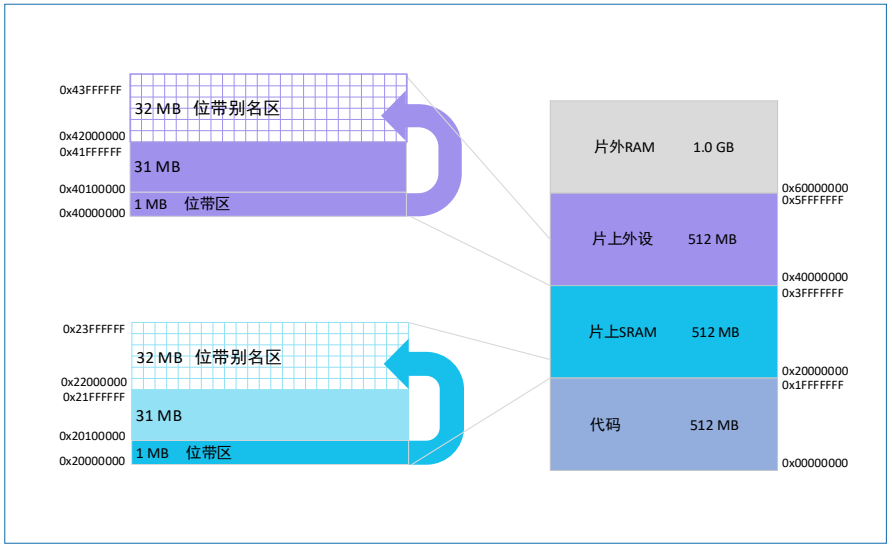


图 2-3 位段结构示意图

器件的外设寄存器和 SRAM 被映射到一个位段区里，这允许执行单一的位段的写和读操作。

下面的映射公式给出了别名区中的每个字是如何对应位段区的相应位的：

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

其中：

- bit_word_addr 是存储器别名区中字的地址，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里的序号。
- bit_number 是目标位所在位置（0~31）。

例子

下面以如何将 SRAM 地址为 0x2000 0300 的字节中的位 2 映射到别名区为例进行说明：

0x2200 6008=0x2200 0000+（0x300*0x20）+（0x2*0x4）

对 0x2200 6008 地址的写操作与对 SRAM 中地址 0x2000 0300 字节的位 2 执行读-改-写操作有着相同的效果。

读 0x2200 6008 地址返回 SRAM 中地址 0x2000 0300 字节的位 2 的值（0x01 或 0x00）。

SRAM 位段别名区与位段的映射关系如下图所示：

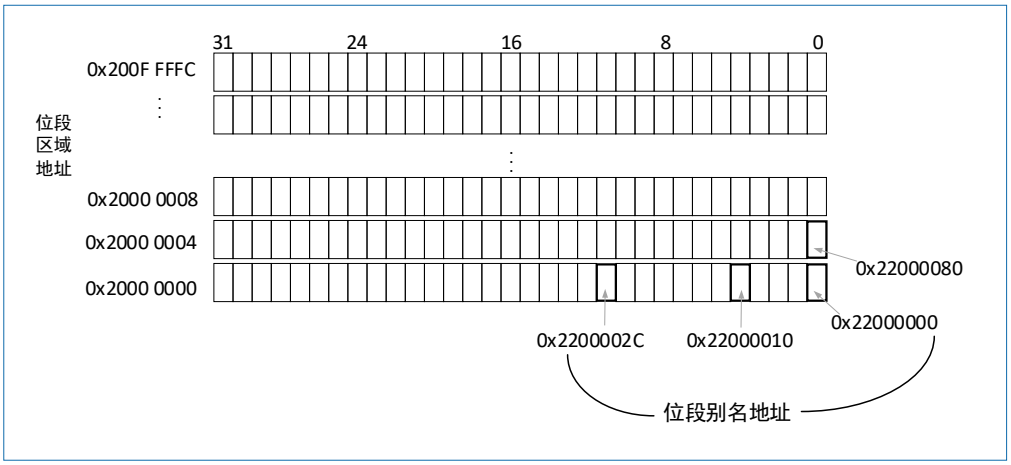


图 2-4 别名区与位段的区间映射关系

表 2-2 SRAM 区域位段地址重映射关系

位段区域	别名等价
0x20000000 bit[0]	0x22000000 bit[0]
0x20000000 bit[1]	0x22000004 bit[0]
0x20000000 bit[2]	0x22000008 bit[0]
…	…
0x20000000 bit[31]	0x2200007C bit[0]
0x20000004 bit[0]	0x22000080 bit[0]
…	…
0x20000004 bit[31]	0x220000FC bit[0]
…	…
0x200FFFFC bit[31]	0x23FFFFFC bit[0]

请参考《Cortex®-M3 技术参考手册》以了解更多有关位段的信息。

2.3 启动配置

器件可以通过 BOOT[1:0]引脚选择三种不同的启动模式。

表 2-3 启动模式

启动模式选择引脚		启动模式	说明
BOOT1	BOOT0		
X	0	主 Flash 存储器	主 Flash 存储器被选为启动区域
0	1	系统存储器	系统存储器被选为启动区域
1	1	内置 SRAM	内置 SRAM 被选为启动区域

在系统复位后，BOOT 引脚的值将被锁存。用户可以通过设置 BOOT1 和 BOOT0 引脚的状态，以选择复位后的启动模式。

在从待机模式退出时，BOOT 引脚的值将被重新锁存。因此，在待机模式下 BOOT 引脚应保持为需要的启动配置。在启动延迟之后，CPU 从地址 0x0000 0000 获取堆栈顶的地址，并从启动存储器的 0x0000 0004 指示的地址开始执行代码。

因为存储器映射是固定的，代码区始终从地址 0x0000 0000 开始（通过 ICode 和 DCode 总线访问），而数据区（SRAM）始终从地址 0x2000 0000 开始（通过系统总线访问）。Cortex®-M3 的 CPU 始终从 ICode 总线获取复位向量，即启动仅适合于从代码区开始（典型地从 Flash 启动）。器件实现了一个特殊的机制，系统可以不仅从 Flash 存储器或系统存储器启动，还可以从内置 SRAM 启动。

根据选定的启动模式，主 Flash 存储器、系统存储器或 SRAM 可以按照以下方式访问：

- 从主 Flash 存储器启动：主 Flash 存储器被映射到启动空间（0x0000 0000），但仍然能够在它原有的地址（0x0800 0000）访问它。即 Flash 存储器的内容可以在两个地址区域访问，0x0000 0000 或 0x0800 0000。
- 从系统存储器启动：系统存储器被映射到启动空间（0x0000 0000），但仍然能够在它原有的地址（0x1FFF F000）访问它。
- 从内置 SRAM 启动：RAM 被映射到启动空间（0x0000 0000），也能在 0x2000 0000 开始的地址区访问 SRAM。

2.3.1 内嵌的自举程序

内嵌的自举程序存放在系统存储区，在生产线上写入，用于通过串行接口对 Flash 存储器进行重新编程：

可以通过 USART1 接口（PA9, PA10）启用自举程序。

3 嵌入式 Flash

3.1 Flash 主要特性

存储器结构：

- 主 Flash 模块：512 Kbyte，该存储块划分为 256 × 2Kbyte(Page)
- 信息块：15 Kbyte
- 选项字节有 60 byte

Flash 的接口特征：

- 带预取缓冲器的读接口
- 选项字节加载器
- Flash 编程/擦除操作
- 访问/写保护
- 低功耗模式
- 指令加密

3.2 Flash 功能描述

3.2.1 Flash 结构框图

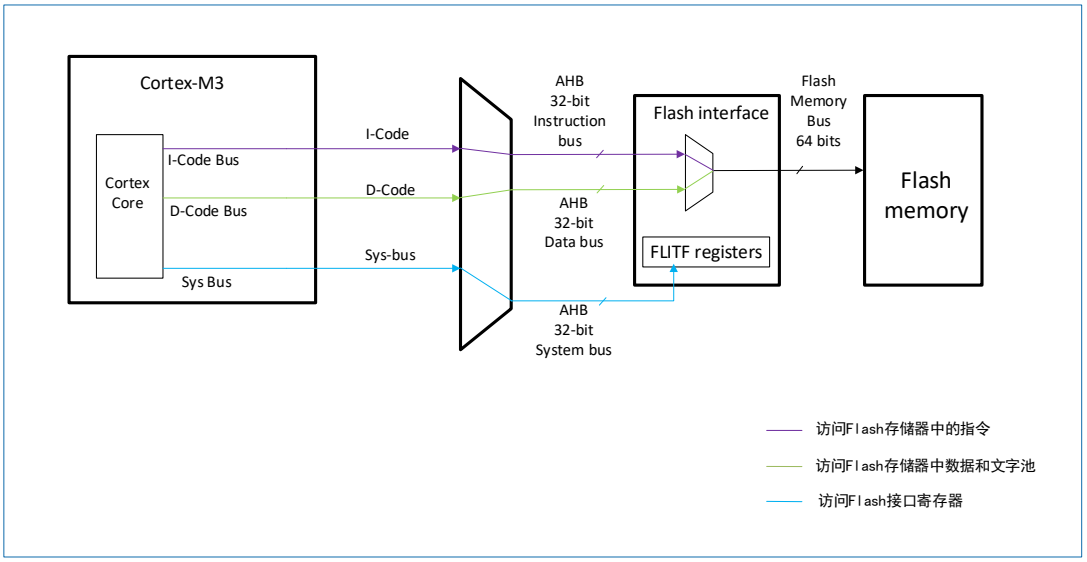


图 3-1 Flash 结构框图

3.2.2 Flash 结构

Flash 空间由 32 位宽的存储单元组成，既可以存代码又可以存数据。主 Flash 块有 256 页（每页 2 Kbyte）。写保护以扇区为单位，每扇区 2 页（4Kbyte）。模块如下表所示：

表 3-1 Flash 模块的组织

模块	名称	地址	大小（字节）
主存储块	页 0	0x0800 0000-0x0800 07FF	2K
	页 1	0x0800 0800-0x0800 0FFF	2K

模块	名称	地址	大小（字节）
	页 2	0x0800 1000-0x0800 17FF	2K
	…	…	…
	页 255	0x0807 F800-0x0807 FFFF	2K
信息块	信息区 1	0x1FFF 0000-0x1FFF 03FF	1K
	保留	0x1FFF 0400-0x1FFF_AFFF	
	信息区 2	0x1FFF B000-0x1FFF E7FF	14K
	保留	0x1FFF E800-0x1FFF EFFF	
	系统存储器	0x1FFF F000-0x1FFF F7FF	2K
	选项字节	0x1FFF F800-0x1FFF F83B	60

3.2.3 选项字节说明

选项字节由用户根据应用的需要进行配置。例如：可以选择使用硬件模式的看门狗或软件模式的看门狗。对于 0x1FFF F800~0x1FFF F810 地址的选项字，在选项字节中每个 32 位的字被划分为下述格式：

表 3-2 选项字的格式

位[31:24]	位[23:16]	位[15:8]	位[7:0]
选项字节 1 的反码	选项字节 1	选项字节 0 的反码	选项字节 0

注意：

编程时，反码由硬件自动实现，软件写无效。

新写入的选项字节（用户的或读/写保护的），在系统复位后才生效。

选项字节块中选项字节的组织结构如表 3-3 所示。选项字节可以从下表列出的存储器地址读出，或从选项字节寄存器（FLASH_OBR）读出。

表 3-3 选项字节结构

地址	[31:24]	[23:16]	[15:8]	[7:0]
0x1FFF F800	nUSER	USER	nRDP	RDP
0x1FFF F804	nDATA1	DATA1	nDATA0	DATA0
0x1FFF F808	nWRP1	WRP1	nWRP0	WRP0
0x1FFF F80C	nWRP3	WRP3	nWRP2	WRP2
0x1FFF F810~0x1FFF F81F	保留			
0x1FFF F820	ENCRY_CFG[31:0]			
0x1FFF F824	DECRY_CFG[31:0]			
0x1FFF F828	UKEY[31:0]			
0x1FFF F82C	UKEY[63:32]			

地址	[31:24]	[23:16]	[15:8]	[7:0]
0x1FFF F830	保留			IWDG_RL_IV[11:0]
0x1FFF F834	IWDG_INI_KEY[31:0]			
0x1FFF F838	LSI_LP_CTL[31:0]			

表 3-4 0x1FFF F800~0x1FFF F81F 地址的选项字节说明

地址	位域	选项字节描述
0x1FFF F800	位 31:24	nUSER：USER 按位取反
	位 23:16	USER：用户选项字节 USER 内容存储于 FLASH_OBR[9:2]，用于配置如下特性： ● 选择硬件或软件看门狗事件。 ● 当进入停机（Stop）模式时，复位事件。 ○ 位[18]：nRST_STDBY - 0：当进入待机模式时产生复位 - 1：进入待机模式时不产生复位 ○ 位[17]：nRST_STOP - 0：当进入停机模式时产生复位 - 1：当进入停机模式不产生复位 ○ 位[16]：WDG_SW - 0：IWDG 硬件看门狗 - 1：IWDG 软件看门狗
	位 15:8	nRDP：RDP 按位取反
	位 7:0	RDP：Flash 读保护选项字节 该字节的值定义了 Flash 读保护级别，可帮助用户保护存储在 Flash 内部的程序。通过设置 RDP 选项字节来使能读保护。当 RDP 选择字配置值非 0xA5 时，使能读保护。读保护状态存储在 FLASH_OBR[1]。
0x1FFF F804	位 31:0	DATAx：用户数据 可以使用选项字节的编程方式编程。 ● 位[31:24]：nDATA1 ● 位[23:16]：DATA1（存于FLASH_OBR[25:18]） ● 位[15:8]：nDATA0 ● 位[7:0]：DATA0（存于FLASH_OBR[17:10]）
0x1FFF F808	位 31:0	WRPx：Flash 写保护选项字节 ● 位[31:24]：nWRP1 ● 位[23:16]：WRP1（存于 FLASH_WRPR[15:8]） ● 位[15:8]：nWRP0 ● 位[7:0]：WRP0（存于 FLASH_WRPR[7:0]） ○ 0：写保护使能 ○ 1：写保护禁能 Flash 的写保护范围是按照每一位（Bit）控制对应的 Flash 页（Page）。除 WRP3[7]作用于 62 ~ 255 页外，WRPx 的每一位均按 2 页为单位作用于 0~61 页。
0x1FFF F80C	位 31:0	WRPx：Flash 写保护选项字节 ● 位[31:24]：nWRP3

地址	位域	选项字节描述
		● 位[23:16]：WRP3（存于 FLASH_WRPR[31:24]） ● 位[15:8]：nWRP2 ● 位[7:0]：WRP2（存于 FLASH_WRPR[23:16]） ○ 0：写保护使能 ○ 1：写保护禁能
0x1FFF F810- 0x1FFF F81F	位 31:0	存储用户自定义数据

表 3-5 0x1FFF F820~0x1FFF F838 地址的选项字说明

地址	位域	选项字节描述
0x1FFF F820	位 31:0	ENCRY_CFG[31:0]：如果存储的值为 0x1357 ECA8，则使能片内 Flash 数据加密。 在使能加密后，再对 Flash 编程时，Flash 上存储的是加密后的密文。
0x1FFF F824	位 31:0	DECRY_CFG[31:0]：如果存储的值为 0x2468 DB97，则使能片内 Flash 数据解密。 在使能解密后，从 Flash 读出的数据自动解密后再返给 CPU。
0x1FFF F828- 0x1FFF F82C	位 31:0	UKEY[63:0]：存储 Flash 加解密的密钥，由用户自己配置。 当地址 0x1FFF F828 和 0x1FFF F82C 的值不为全 FF 时，如果软件读这两个地址，返回的值为 0xAAAA AAAA。
0x1FFF F830	位 31:0	位[31:12]：保留 IWDG_RL_IV[11:0]：存储 IWDG_RLR 寄存器的初始值。 当 IWDG 配置为硬件看门狗时，可以配置 IWDG_RL_IV[11:0]来设计 IWDG 的复位时间间隔。
0x1FFF F834	位 31:0	IWDG_INI_KEY[31:0]：决定 IWDG_RL_IV 是否生效。 当 IWDG_INI_KEY[31:0]为 0xA5A5 5B1E 时，IWDG_RL_IV 配置有效，否则无效。
0x1FFF F838	位 31:0	LSI_LP_CTL[31:0]：决定系统在使能IWDG后再进入停机或待机模式时，是否需要被IWDG周期唤醒。 ● 该位域配置为 0x369C F0F0 时： MCU 进入停机（Stop）或待机（Standby）模式后，可根据 LSION的设置关闭 LSI。 MCU 被唤醒后，LSI 恢复为进入低功耗模式之前的状态。 ● 若未配置该位域： 在使能IWDG 后再进入停机或待机模式，系统会被 IWDG 周期唤醒。

每次系统复位后，选项字节装载器（OBL）读取信息块 0x1FFF F800~0x1FFF F804 的数据，并保存在选项字节寄存器（FLASH_OBR）中。所有的选项字节（不包括它们的反码）用于配置该 MCU，CPU 可以读 FLASH_OBR 寄存器。

每个选项字节都有其值的反码数据存放在信息块中，其目的用于校验选项字节的正确性。当选项字节装载后，CPU 会检查选项字节的正确性。若选项字节与其反码比较不一致时，会产生选项字节校验错（OPTERR）信息。当产生 OPTERR 后，CPU 会强制将相应的选项字节值变为 0xFF。当选项字节与其反码都为 0xFF（擦除后的状态）时，CPU 不会比较其与反码的差异。

3.2.4 读操作

嵌入式 Flash 模块可以像普遍存储空间一样直接寻址访问。

取指令和取数据都是通过 AHB 总线读取访问。AHB 总线矩阵主要的工作就是产生控制信号，然后从 Flash 里读取信息和预取 CPU 所需要的信息。预取模块仅用于在 I-Code 总线上实现指令预取。D-Code 总

线具有更高的预取优先级。

取指

Cortex®-M3 通过 I-Code 总线取指和通过 D-Code 总线取数（常量/数据）。预取模块旨在增加总线的效率。

预取缓冲区

预取缓冲区的内容与 Flash 相同，能够完全替代一次同样大小的读取访问。预取缓冲区使得更快速的 CPU 执行成为可能，因为 CPU 取一个字的同时下一个字的内容已经在预取缓冲区中。

预取控制器

预取控制器会根据预取缓冲区的可用空间来控制访问 Flash 的时机。当预取缓冲区中存在至少一块可用空间时，预取控制器会发起一次读取请求。复位后，预取指缓冲区的默认状态是打开的。

访问等待周期

为了保持读取 Flash 的控制信号，必须在 Flash 访问控制寄存器（LATENCY[4:0]）中设置预取控制器时钟周期与 Flash 访问时间的比值。这个数值等于每次访问 Flash 后到下次访问之间所需插入的等待周期的个数。复位后，这个值默认为零，也就是没有插入等待周期的状态。

3.2.4.1 D-Code 接口

D-Code 接口在 CPU 端包含一个简单的 AHB 接口和产生 Flash 访问控制的仲裁申请发生器。D-Code 具有高于预取指的优先级。这个接口运用在预取缓存的访问时钟调谐器。

3.2.4.2 Flash 访问控制器

该模块主要作为一个简单的仲裁器，用于仲裁预取（I-Code）和 D-Code 之间的读取请求。D-Code 接口请求的优先级高于 I-Code。

3.2.5 Flash 写和擦除操作

在产品的整个工作电压范围内支持执行 Flash 编程和擦除操作。该操作由以下寄存器完成：

- Flash 关键字寄存器（FLASH_KEYR）
- Flash 选项关键字寄存器（FLASH_OPTKEYR）
- Flash 控制寄存器（FLASH_CR）
- Flash 状态寄存器（FLASH_SR）
- Flash 地址寄存器（FLASH_AR）
- Flash 选项字节寄存器（FLASH_OBR）
- Flash 写保护寄存器（FLASH_WRPR）
- Flash 控制寄存器 2（FLASH_ECR）
- 4 个编程数据寄存器（PW0~PW3）

注意：在执行编程/擦除 Flash 的同时，不能对 Flash 取指和访问数据。否则，总线访问将暂停。

3.2.5.1 键值

键值用于对 FLASH_KEYR 寄存器进行解锁：

- KEY1=0x4567 0123
- KEY2=0xCDEF 89AB

3.2.5.2 对 Flash 空间的解锁

复位后，Flash 存储器默认处于受保护状态，以避免意外擦除。FLASH_CR 寄存器的值通常不允许改写。只有对 FLASH_KEYR 寄存器进行解锁操作后，才具有对 FLASH_CR 寄存器的访问权限。解锁操作包括以下步骤：

1. 向 FLASH_KEYR 寄存器写入关键字 KEY1=0x4567 0123。
2. 向 FLASH_KEYR 寄存器写入关键字 KEY2=0xCDEF 89AB。

任何错误的解锁顺序将会锁死 FLASH_CR 直至下次复位。当发生关键字错误时，会由总线错误引发一次硬件错误中断。

- 如果 KEY1 出错，就会立即触发中断。
- 如果 KEY1 正确且 KEY2 错误时，就会在 KEY2 错的时刻触发中断。

3.2.5.3 主 Flash 编程

主 Flash 一次可以编程 16 位、32 位、64 位或者 128 位。编程长度由 FLASH_CR 和 FLASH_ECR 决定。

- 当 FLASH_CR 中的 PG 位为 1 时，直接对相应的地址写一个半字（16 位），是一次编程操作。
- 当 FLASH_ECR 中的 WPG 位为 1 时，直接对相应的地址写一个字（32 位），是一次编程操作。
- 当 FLASH_ECR 中的 2WPG 为 1 时，直接对相应的地址写两个字（64 位），是一次编程操作。
- 当 FLASH_ECR 中的 4WPG 为 1 时，直接对相应的地址写四个字（128 位），是一次编程操作。

注意：如果 FLASH_CR 中的 PG 为 1 并试图写别的长度而不是半字，将引起总线错误中断。

如果在编程 Flash 的过程中，CPU 对 Flash 进行读/写，CPU 会被挂起，直到 Flash 编程完成。

半字编写流程如下图：

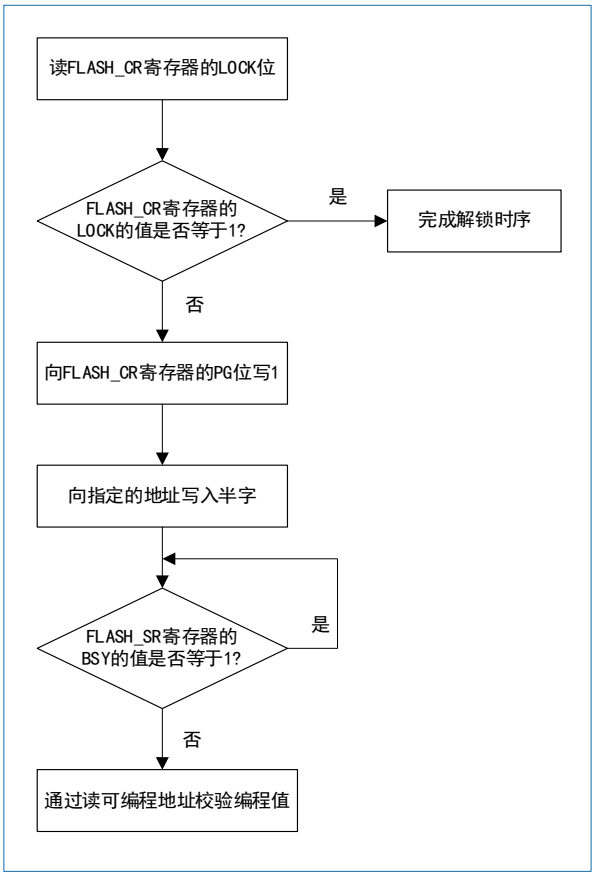


图 3-2 对 Flash 的半字编程

字编程的流程与半字编程流程相似，区别在于将 FLASH_ECR 中的 WPG 位置’1’。

双字编程流程图如下：

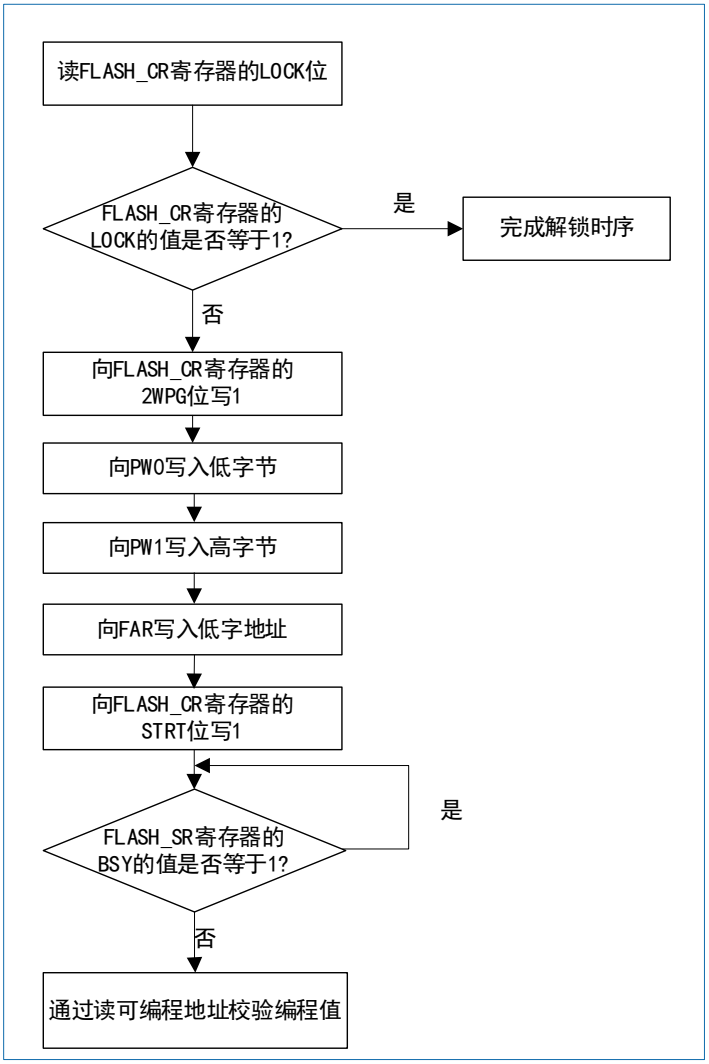


图 3-3 对 Flash 的双字编程

四字编程的流程和双字编程流程类似，区别在于将 FLASH_ECR.4WPG 位置’1’，并且写入 4 个字的数据到 PW0、PW1、PW2、PW3 寄存器中。

标准编程

Flash 存储器接口会预读待编程地址的内容，然后判断其是否已经被擦除。如果不是，那么编程操作会自动取消，并且在 FLASH_SR 寄存器的 PGERR 位上提示编程错误告警。如果被编程的内容为全零，则会例外，这时会正确编程并且不告警。

如果待编程地址所对应的 FLASH_WRPR 中的写保护位有效，同样也不会有编程动作，也会产生编程错误告警。编程动作结束后，FLASH_SR 寄存器中得 EOP 位会给出提示。

主 Flash 存储器标准模式下的编程过程如下：

- 半字和字编程：

1. 检查 FLASH_SR 中的 BSY 位，以确认上次操作已经结束。
2. 置位 FLASH_CR 寄存器中的 PG 位或者 FLASH_ECR 的 WPG 位。
3. 根据配置，以半字/字为单位向目标地址写入数据。

4. 等待 FLASH_SR 寄存器中的 BSY 归零。
5. 读取编程的值然后进行验证。

- 双字和四字编程：

1. 检查 FLASH_SR 中的 BSY 位，以确认上次操作已经结束。
2. 置位 FLASH_ECR 寄存器中的 2WPG 位或者 4WPG 位。
3. 根据配置，向 PW0~PW1 或者 PW0~PW4 写入数据。
4. 向 FLASH_AR 写入待编程位置最低位置地址。
5. 置位 FLASH_CR 寄存器中的 STRT 位为 1。
6. 等待 FLASH_SR 寄存器中的 BSY 归零。
7. 读取编程的值然后验证。

注意：当 FLASH_SR 中的 BSY 被置'1'时，写模式下的寄存器不能被读。

3.2.5.4 Flash 存储器擦除

Flash 存储器可以按页或半页为单位擦除，也可以整片擦除。

页擦除

擦除页的步骤如下：

1. 检查 FLASH_SR 寄存器中的 BSY 位，以确认上次操作已经结束。
2. 将 FLASH_CR 寄存器中的 PER 位置为 1，以选择按页擦除。
3. 写 FLASH_AR 寄存器的 FAR 位，写入待擦除页的地址。
4. 将 FLASH_CR 寄存器中的 STRT 位置为 1，以启动擦除操作。
5. 等待 FLASH_SR 中的 BSY 变为 0，表明擦除操作完成。
6. 检查 FLASH_SR 寄存器的 EOP 标志（若 Flash 擦除成功会置位 EOP），然后软件清除该标志位。

擦除页流程如下图：

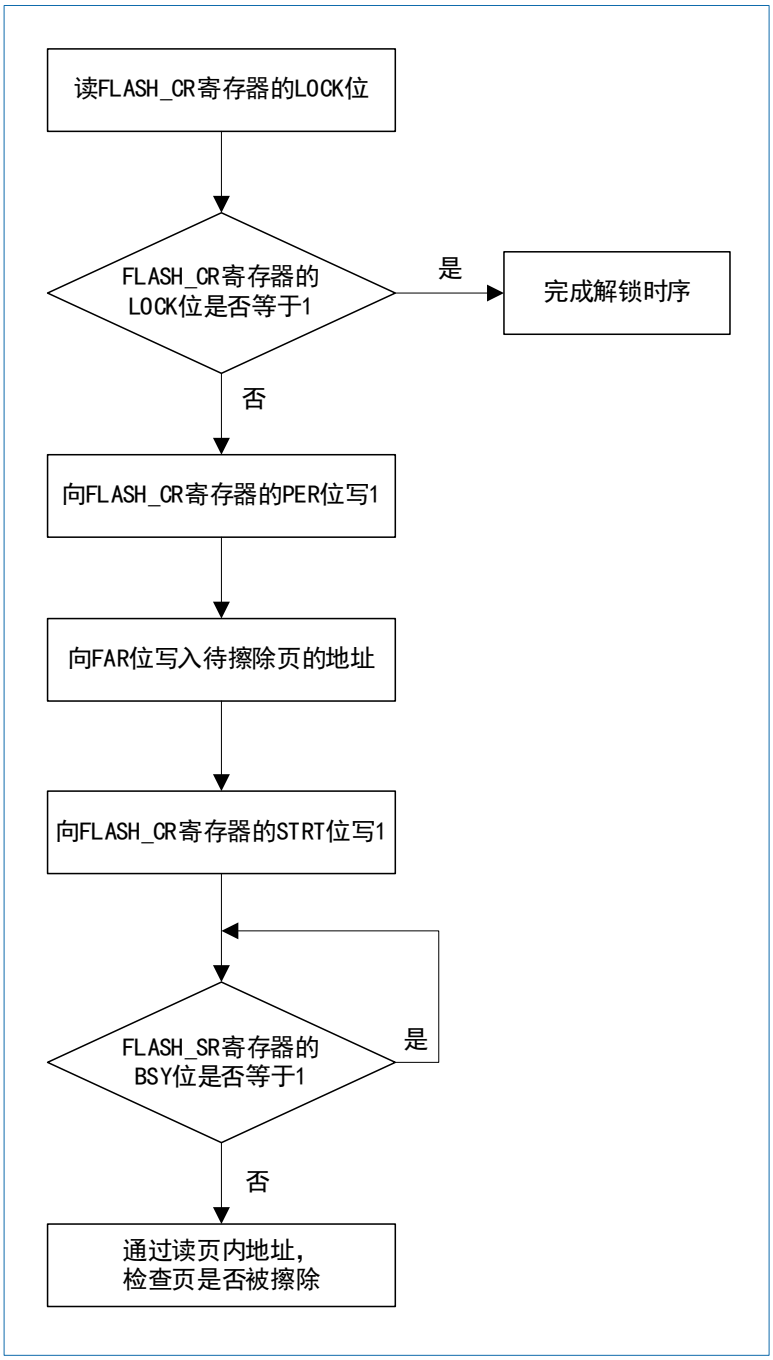


图 3-4 对 Flash 擦除页的流程

半页擦除

Flash 的半页为 1 Kbyte，半页擦除流程和页擦除流程类似，区别在于把 FLASH_ECR 中的 HPER 位置'1'。擦除半页的步骤如下：

1. 检查 FLASH_SR 寄存器中的 BSY 位，以确认上次操作已经结束。
2. 将 FLASH_CR 寄存器中的 HPER 位置为 1，以选择按半页擦除。
3. 写 FLASH_AR 寄存器的 FAR 位，写入待擦除半页的地址。
4. 将 FLASH_CR 寄存器中的 STRT 位置为 1，以启动擦除操作。
5. 等待 FLASH_SR 中的 BSY 变为 0，表明擦除操作完成。
6. 检查 FLASH_SR 寄存器的 EOP 标志（若 Flash 擦除成功会置位 EOP），然后软件清除该标志位。

整片擦除

可以用整片擦除命令一次擦除整个 Flash 区，但该命令不会影响信息块，具体步骤如下：

1. 检查 FLASH_SR 寄存器的 BSY 位，以确认上次操作已经结束。
2. 将 FLASH_CR 寄存器中的 MER 位置为 1，以选择整片擦除。
3. 将 FLASH_CR 寄存器中的 STRT 位置为 1，以启动擦除操作。
4. 等待 FLASH_SR 中的 BSY 位置 0，表明整片擦除操作结束。
5. 检查 FLASH_SR 寄存器的 EOP 标志位（如果 Flash 擦除成功会置位 EOP），然后软件清除该标志位。

3.2.5.5 选项字节编程

选项字节总共 60 字节，包括以下选项字：

- 写保护
- 读保护
- 用户数据
- 硬件配置
- Flash 加密配置
- IWDG 配置
- 低功耗模式下 LSI 硬件配置

在编程地址 0x1FFF F800~0x1FFF F810 操作开始前，LSB 值会自动转化为 MSB，以适应选项字节的位定义。系统会自动检查选项字节是否已被擦除。如果未被擦除，则编程操作会被取消并在 FLASH_SR 中的 PGERR 位提示错误。

注意：选项字节的编程操作开始前，需要对 FLASH_KEYR（解除 Flash 访问限制）和 FLASH_OPTKEYR 写入关键字 KEY 以解锁 Flash，然后进行选项字节编程操作。

选项字节编程的步骤如下：

1. 检查 FLASH_SR 寄存器中的 BSY 位，以确保上次操作结束。
2. 将 FLASH_CR 寄存器中的 OPTPG 位置为 1，以选择半字方式写入。
3. 写数据（半字）到目标地址。
4. 等待 FLASH_SR 中的 BSY 位为 0 和 EOP 为 1，表示编程操作结束。
5. 将 FLASH_CR 寄存器中的 OPTPG 位置为 0，以恢复默认值。

当读保护选项字节由保护状态变成非保护状态时，会执行一次整片擦除，然后才允许改写读保护位数据。若用户仅想改写选项字节区以外的数据，则不会引起整片擦除，该机制用于保护 Flash 的内容。

擦除过程

注意：选项字节的擦除操作开始前，需要对 FLASH_KEYR（解除 Flash 访问限制）和 FLASH_OPTKEYR 写入关键字 KEY 以解锁 Flash，然后进行选项字节擦除操作。

选项字节是按页擦除的，步骤如下：

1. 检查 FLASH_SR 寄存器中的 BSY 位，以确保上次操作结束。
2. 置 FLASH_CR 寄存器中的 OPTER 位为 1，以擦除整个选项字节区域。

3. 将待擦除的地址写入 FLASH_AR 寄存器（待擦除地址必须属于 OPT 选项字节表里已有的地址，否则会出现不可预计的错误）。
4. 将 FLASH_CR 寄存器中的 STRT 位置为 1，以启动擦除操作。
5. 等待 FLASH_SR 中的 BSY 位变成 0 和 EOP 为 1，表明擦除操作结束。
6. 将 FLASH_CR 寄存器中的 OPTER 位置为 0，以恢复默认值。

3.2.6 读写保护

支持保护 Flash 存储器的用户区，以防止被不可信的程序读取，也可防止在程序跑飞时意外擦除 Flash。

3.2.6.1 主 Flash 读保护

通过置位选项字节中的 RDP 位，然后重新复位系统，读保护功能就被激活了。

注意：如果调试器处于连接状态时（通过 JTAG/SWD 连接）设置了读保护，应使用 POR 复位（上电复位），而不是系统复位（无调试器连接）。

一旦保护字节被编程：

- 主 Flash 区的内容只能被用户程序读取（如果连接了调试器，用户程序也不能读取主 Flash 区的内容）。
- 页 0~3 是自动写保护的，剩下的 Flash 空间可以被用户程序编程（写入 IAP，常量储存等）。这个保护只针对在调试模式和从 SRAM 启动时的擦除和编程，不保护整片擦除。
- 可以下载代码到 SRAM 和从 SRAM 中执行程序，它也可以被用来关闭读保护。当读保护选项字节被改变，会引发整片擦除。
- 当从 SRAM 启动的时候，使用 DMA 的 Flash 访问也会被禁止。
- 使用 JTAG、串行线查看器 SWV（Serial wire viewer）、串行线调试 SWD（Serial wire debug）和边界扫描时，Flash 的访问被禁止。

Flash 存储器的读保护级别和RDP 选项字节及其反码内容的对应关系，如下表：

表 3-6 读保护级别和 RDP 字节的对应关系

RDP 字节值	RDP 反码值（按位取反）	读保护级别
0x00FF	0x00FF	保护
0x00A5	0x005A	没有保护
其他值	非 RDP 反码值	保护

注意：擦除选项字节不会触发整页擦除，因为擦除值 0xFF 和保护值对应。

读保护解除

从 SRAM 中解除读保护的方法：

1. 擦除整个选项区间，擦除后读保护代码（RDP）会被置为 0x00FF，此时读保护仍然是使能。
2. 写入正确的 RDP 值 0x00A5 以解除读保护，这个操作会引发一次主 Flash 的整片擦除。
3. POR 复位会重加载选项字节 RDP，从而关闭读保护。

注意：bootloader 可以关闭读保护（在这种情况下只有系统复位能重加载选项字节），同样，bootloader 关闭读保护时，也会触发主 Flash 的整片擦除。

3.2.6.2 主 Flash 写保护

对于页 0 ~ 61，写保护是以 2 页为单位；主 Flash 剩下的页 62 ~ 255 页作为一个整体进行写保护控制。通过配置选项字节中的 WRP 位，然后复位时自动重新加载新的 WRP 选项字节即可使能写保护。如果试图写入或擦除一个受保护的扇区，会引起 FLASH_SR 中的 WRPRTERR 标志位被置位。

说明：只有当 WRP 和 nWRP 按位互反的时候，对应的写保护才会生效。

写保护的解除

以下提供 2 个解除写保护的示例：

- 例 1：在解除写保护后禁止读保护
  - 使用 FLASH_CR 寄存器中的 OPTER 位，擦除整个选项字节区域。
  - 向 RDP 写入 0xA5 从而解除保护，这会引起整片擦除。
  - 复位引起选项字节重新加载，写保护解除。
- 例 2：在解除写保护后，读保护仍然有效，这种方式对使用用户 bootloader 进行在应用编程时很有用。
  - 使用 FLASH_CR 寄存器中的 OPTER 位，擦除整个选项字节区域。
  - 复位引起选项字节重新加载，写保护解除。

3.2.6.3 选项字节的写保护

选项字节默认被写保护且随时可读。必须先向 FLASH_OPTKEYR 寄存器顺序写入关键字，才能对选项字节进行写/擦除操作。填入正确的关键字会引起 FLASH_CR 中的 OPTWRE 置位，表明解锁成功；通过对 OPTWRE 位清零，能够禁止对选项字节的写操作。

3.2.7 Flash 数据加密

Flash 控制器内部有一个加解密模块，用于加解密用户存储在 Flash 中的程序和数据。用户可以选择以密文或者明文的方式存储。当使能数据加解密后，Flash 控制器会自动进行加解密运算，加解密过程对应用软件透明。如果用户使能 Flash 数据加密后，编程写入 Flash 的默认是密文。

加密使用的用户密钥为 64 位。

芯片默认不使能 Flash 加解密。

在 Flash 的选项字节中存储有加密/解密使能位和用户密钥。在系统复位时，Flash 控制器会自动从 Flash 载入加密/解密使能标志和用户密钥。另外还可以配置一组影子寄存器来改写加解密标志和用户密钥，这是为了方便在通过调试口发送程序的时候可以直接使能写入加密，写入密文到 Flash。如果已经把密钥保存到 Flash 选项字节，然后再读选项字地址，则读到的值始终为 0xAAAA AAAA，以保证用户密钥不会泄露。

当使能加密后，只加密主 Flash 空间，不会加密 Flash 系统存储器和选项字节部分。如果主 Flash 空间的内容被加密后，但是没有使能解密，则 CPU 读到的是密文，将执行出错。如果主 Flash 空间的内容未加密，但是使能了解密，则 CPU 读到的是乱码，也将执行出错。因此在程序执行时，必须同时使能或同时不使能加密和解密。

在使能 Flash 数据加解密后，如果应用程序读 Flash 刚擦除完成还没有写数据的地址，则读到的数据不是 0xFFFF FFFF，而是一个乱码，但是这时应用程序可以成功编写数据到这些地址，并不会触发 PGERR。同理，在使能 Flash 数据加密后，CPU 读到的数据为 0xFFFF FFFF 也不代表改地址没有被编写过数据。

当烧录程序到芯片时，使能 Flash 数据加解密的步骤如下：

1. 向寄存器地址 0x4002 2078 写入 0x1357 ECA8（配置 ENCRY_EN）。

2. 向寄存器地址 0x4002 2084 和 0x4002 2080 写入 UKEY，UKEY 的高 32 位存到 0x4002 2084。

3. 向 Flash 烧录程序。

4. 向 Flash 地址 0x1FFF F820 写入 0x1357 ECA8。

通过设置选项字节，使能 Flash 数据加解密的步骤如下：

1. 向 Flash 地址 0x1FFFF820 写入 0x1357 ECA8（配置 ENCRY_EN 以使能 Flash 数据加密）。

```
FLASH->KEYR=0x45670123;
FLASH->KEYR=0xCDEF89AB;
FLASH->OPTKEYR=0x45670123;
FLASH->OPTKEYR=0xCDEF89AB;
FLASH->CR|=0x00000010；
*（（volatileu16*）（0x1FFFF820））=0xeca8;
while（（FLASH->SR&0x00000001）==0x00000001）;
*（（volatileu16*）（0x1FFFF822））=0x1357;
    while（（FLASH->SR&0x00000001）==0x00000001）;
    FLASH->SR=0x20;
FLASH->CR&=0xfffffffe;
```

2. 向 Flash 地址 0x1FFFF824 写入 0x2468 DB97（配置 DECRY_EN 以使能 Flash 数据解密）。

```
FLASH->KEYR=0x45670123;
FLASH->KEYR=0xCDEF89AB;
FLASH->OPTKEYR=0x45670123;
FLASH->OPTKEYR=0xCDEF89AB;
FLASH->CR|=0x00000010；
*（（volatileu16*）（0x1FFFF824））=0xdb97;
while（（FLASH->SR&0x00000001）==0x00000001）;
*（（volatileu16*）（0x1FFFF826））=0x2468;
    while（（FLASH->SR&0x00000001）==0x00000001）;
    FLASH->SR=0x20;
FLASH->CR&=0xfffffffe;
```

3. 向 Flash 地址 0x1FFF F828~0x1FFF F82E 写入 UKEY，UKEY 的高字节写到高地址。

```
FLASH->KEYR=0x45670123;
FLASH->KEYR=0xCDEF89AB;
FLASH->OPTKEYR=0x45670123;
FLASH->OPTKEYR=0xCDEF89AB;
FLASH->CR|=0x00000010；
*（（volatileu16*）（0x1FFFF828））=UKEY[15:0];
while（（FLASH->SR&0x00000001）==0x00000001）;
*（（volatileu16*）（0x1FFFF82a））=UKEY[31:16];
while（（FLASH->SR&0x00000001）==0x00000001）;
*（（volatileu16*）（0x1FFFF82c））=UKEY[47:32];
while（（FLASH->SR&0x00000001）==0x00000001）;
```

```
*（（volatileu16*）（0x1FFFF82e））=UKEY[63:48];
while（（FLASH->SR&0x00000001）==0x00000001）;
FLASH->SR=0x20;
FLASH->CR&=0xfffffffe;
```

4. 烧录程序至 Flash。

注意事项：

向 Flash 地址 0x1FFF F828~0x1FFF F82C 写入 UKEY 后，不能读出来校验。因为向该 Flash 地址写入 UKEY 后，再读该地址得到的值始终为 0xAAAA AAAA。

向密钥寄存器 0x4002 2084 和 0x4002 2080 写入 UKEY 后也不能读出来校验。

3.3 Flash 中断

表 3-7 Flash 中断事件和事件标志

中断事件	事件标志	使能控制位
操作结束	EOP	EOPIE
写保护错误	WRPRTERR	ERRIE
编程错误	PGERR	ERRIE

3.4 FLASH 寄存器

基地址：0x4002 2000

空间大小：0x400

Flash 寄存器可以字节（8 位）、半字（16 位）或字（32 位）为单位进行访问。

3.4.1 FLASH 访问控制寄存器（FLASH_ACR）

偏移地址：0x00

复位值：0x00000030

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res										PRFTBS	PRFTBE	HLFCYA	LATENCY[2:0]		
										r	rw	rw	rw		

位 31:6	Res：保留 必须保持复位值。
位 5	PRFTBS：预取指缓冲区的状态（Prefetch buffer status） • 0：预取缓冲区被禁止 • 1：预取缓冲区被使能
位 4	PRFTBE：预取指缓冲区使能（Prefetch buffer enable） • 0：禁止预取指 • 1：使能预取指
位 3	HLFCYA：Flash 半周期访问使能（Flash half cycle access enable）

	• 0：禁止半周期访问 • 1：使能半周期访问
位 2:0	LATENCY[2:0]：预设 HCLK 周期和 Flash 访问时间的比率关系（Latency） 和 FLASH_LATENCY_EX.LATENCY[4:3]一起组成一个 5 位的 LATENCY 寄存器等待周期。 LATENCY[4:0]： • 00000：0 等待周期 • 00001：1 等待周期 • 00010：2 等待周期 • … • 11111：31 等待周期 当 HCLK 的时钟频率不同，等待周期需满足： • 当 0 MHz<HCLK<=24 MHz，等待周期必须大于等于 0。 • 当 24 MHz<HCLK<=48 MHz，等待周期必须大于等于 1。 • 当 48 MHz<HCLK<=72 MHz，等待周期必须大于等于 2。 • 当 72 MHz<HCLK<=96 MHz，等待周期必须大于等于 3。 • 当 96 MHz<HCLK<=120 MHz，等待周期必须大于等于 4。 LATENCY 配置为 00101~11111 适用于 CPU 以极低频率运行的应用程序。通过配置大的等待周期可以降低整个芯片的功耗。 通过设置更大的等待周期值，在无需高速处理能力的应用场景下 MCU 的功耗可以得到进一步的降低。

3.4.2 FLASH 关键字寄存器（FLASH_KEYR）

偏移地址：0x04

复位值：0xXXXX XXXX

所有寄存器位全部只能写，如果读会返回 0。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FKEYR[31:16]															
w															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FKEYR[15:0]															
w															

位 31:0	FKEYR[31:0]：解锁 Flash 的关键字（Flash key） 该位域用于输入关键字以解锁 Flash。 • KEY1：0x4567 0123 • KEY2：0xCDEF 89AB

3.4.3 FLASH 选项关键字寄存器（FLASH_OPTKEYR）

偏移地址：0x08

复位值：0xXXXX XXXX

所有寄存器位全部只能写，如果读会返回 0。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OPTKEY[31:16]															
w															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPTKEY[15:0]															
w															

位 31:0	OPTKEY[31:0]：选项字节关键字（Option byte key） 该位域用于输入关键字以解锁 OPTWRE。 • KEY1：0x4567 0123 • KEY2：0xCDEF 89AB

3.4.4 FLASH 状态寄存器（FLASH_SR）

偏移地址：0x0C

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res										EOP	WRPRTERR	Res	PGERR	Res	BSY
										rw	rw		rw		r

位 31:6	Res：保留 必须保持复位值。
位 5	EOP：操作结束（End of operation） 当 Flash 操作（写/擦除）完成时，该位由硬件置位。 该位由软件写 1 后可清零。 *注意：只有成功的写或擦除操作才会由硬件置位 EOP。*
位 4	WRPRTERR：写保护错误标志（Write protection error） 当出现对写保护区域的写操作时，该位由硬件置位。 该位由软件写 1 后可清零。
位 3	Res：保留 必须保持复位值。
位 2	PGERR：写入错误标志（Programming error） 半字编程时，如果被编程区域的值不为'0xFFFF'或者字编程时，如果被编程区域的值不为‘0xFFFFFFFF’，则执行写入操作时被硬件置位。 该位由软件写 1 后可清零。
位 1	Res：保留 必须保持复位值。
位 0	BSY：忙标志（Busy） 该位标明 Flash 操作正在进行。 当开始 Flash 操作的时候被硬件置位，当操作结束时或发生错误时被硬件清零。

3.4.5 FLASH 控制寄存器（FLASH_CR）

偏移地址：0x10

复位值：0x0000 0080

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res			EOPIE	Res	ERRIE	OPTWRE	Res	LOCK	STRT	OPTER	OPTPG	Res	MER	PER	PG
			rw		rw	rw		rw	rw	rw	rw		rw	rw	rw

位 31:13	Res：保留 必须保持复位值。
位 12	EOPIE：操作结束中断使能（End of operation interrupt enable） 该位使能操作结束中断，在使能 FLASH_SR 中的 EOP 位变成 1 时产生中断请求。 • 0：中断禁止 • 1：中断使能
位 11	Res：保留 必须保持复位值。
位 10	ERRIE：错误中断使能（Error interrupt enable） 该位使能操作错误中断，在使能 FLASH_SR 中的 PGERR/WRPRTERR 位变成 1 时产生中断请求。 • 0：中断禁止 • 1：中断使能
位 9	OPTWRE：选项字节写使能（Option byte write enable） 该位为 1 时，选项字节即允许改写。对 FLASH_OPTKEYR 寄存器写入正确的关键字序列就可以将它置’1’。 该位由软件清零。
位 8	Res：保留 必须保持复位值。
位 7	LOCK：锁定 Flash 标志（Lock） 该位只能写 1。当该位为 1 时，表明 Flash 为锁定状态。 该位可以由解锁时序来清零。当解锁不成功时，该位就一直为 1 了，除非下次复位重新操作。
位 6	STRT：启动（Start） 该位会触发一个擦除操作，仅由软件置’1’，仅会在 BSY 被清零时清零。
位 5	OPTER：选项字节擦除（Option byte page erase） 选项字节按页擦除。
位 4	OPTPG：选项字节写入（Option byte programming） 选项字节写入时选择。
位 3	Res：保留 必须保持复位值。
位 2	MER：整片擦除（Mass erase） 整片擦除时选择。
位 1	PER：页擦除（Page erase） 页擦除时选择。

位 0	PG：写入（Half-word programming） Flash 写入（以半字为单位）时选择。

3.4.6 FLASH 地址寄存器（FLASH_AR）

偏移地址：0x14

复位值：0x0000 0000

本寄存器由硬件根据当前和上次操作的地址更新。对于页擦除操作，该寄存器该由软件配置以指明要擦除的页或半页。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FAR[31:16]															
w															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FAR[15:0]															
w															

位 31:0	FAR[31:0]：Flash 地址（Flash Address） 当 HPER/PER 位被选中时，选择待擦除的半页/页的地址。当 PG/WPG/2WPG/4WPG 位使能时，该位域为写入 Flash 的地址，FLASH 的地址必须按照操作的位数边界对齐。 *注意：当 FLASH_SR 寄存器中的 BSY 为 1 时，写操作被锁定。*

3.4.7 FLASH 选项字节寄存器（FLASH_OBR）

偏移地址：0x1C

复位值：0x03FF FFFC

本寄存器的复位值取决于选项字节的写入值，OPTERR 位的复位值取决于复位时选项字节加载环节中比较选项字节及其补码的结果。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res						Data1[7:0]								Data0[7:6]	
						r								r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data0[5:0]						Res					nRST_STDBY	nRST_STOP	WDG_SW	RDPRT	OPTERR
r											r	r	r	r	r

位 31:26	Res：保留 必须保持复位值。
位 25:18	Data1[7:0]：用户数据（User Data）
位 17:10	Data0[7:0]：用户数据（User Data）
位 9:5	Res：保留 必须保持复位值。
位 4	nRST_STDBY：进入待机模式时的复位事件（Generate a reset when entering Standby mode） • 0：当进入待机模式时产生复位 • 1：进入待机模式时不产生复位
位 3	nRST_STOP：进入停机模式时的复位事件（Generate a reset when entering Stop mode） • 0：当进入停机（Stop）模式时产生复位 • 1：进入停机（Stop）模式时不产生复位

位 2	WDG_SW：选择软件或硬件看门狗（Hardware or software watchdog selection） • 0：IWDG 硬件看门狗 • 1：IWDG 软件看门狗
位 1	RDPRT：读保护（Read protection） 该位为 1 表示已设置为读保护（只读）。
位 0	OPTERR：选项字节错误（Option byte error） 当置位时，表明加载选项字节时反码校验错误（只读）。

3.4.8 FLASH 写保护寄存器（FLASH_WRPR）

偏移地址：0x20

复位值：0xFFFF FFFF

本寄存器的复位值取决于选项字节的写入值。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WRP[31:16]															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WRP[15:0]															
r															

位 31:0	WRP[31:0]：写保护（Write protection） 该位保持由 OBL 载入的写保护选项字（只读）。 • 0：写保护激活 • 1：写保护禁止

3.4.9 FLASH 等待周期寄存器（FLASH_LATENCY_EX）

偏移地址：0x60

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res														LATENCY[4:3]	
														rw	

位 31:2	Res：保留 必须保持复位值。
位 1:0	LATENCY[4:3]：等待周期控制位（Latency control ） LATENCY[4:3]和 FLASH_ACR.LATENCY[2:0]一起组成一个 5 位的 LATENCY 寄存器。 LATENCY[4:0]： • 00000：0 等待周期 • 00001：1 等待周期 • 00010：2 等待周期 • … • 11111：32 等待周期

	该位域具体用法可参见 FLASH 访问控制寄存器（FLASH_ACR）。

3.4.10 FLASH 控制寄存器 2（FLASH_ECR）

偏移地址：0x70

复位值：0x0000 0000

说明：FLASH_ECR 的寄存器的位均是配置'1'有效，同一时刻只能配置其中 1 位为有效。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res							INF_4WPG	INF_2WPG	INF_WPG	INF_HWPG	INF_HPER	4WPG	2WPG	WPG	HPER
							rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:9	Res：保留 必须保持复位值。
位 8	INF_4WPG：信息块区按四字编程（4-word programming in Information bank）
位 7	INF_2WPG：信息块区按两字编程（2-word programming in Information bank）
位 6	INF_WPG：信息块区按字编程（Word programming in Information bank）
位 5	INF_HWPG：信息块区按半字编程（Half-word programming in Information bank）
位 4	INF_HPER：信息块区半页擦除（Half-page erasing in Information bank）
位 3	4WPG：主 Flash 区按四字编程（4-Word programming in main Flash memory）
位 2	2WPG：主 Flash 区按两字编程（2-Word programming in main Flash memory）
位 1	WPG：主 Flash 区按字编程（Word programming in main Flash memory）
位 0	HPER：主 Flash 区半页擦除（Half-page erasing in main Flash memory）

3.4.11 加密控制寄存器（FLASH_ENCRY_CFG）

偏移地址：0x78

复位值：0x0000 0000

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

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

位 31:1	Res：保留 必须保持复位值。
位 0	ENCRY_EN：加密主 Flash 区使能（Encryption enable in main flash memory） ENCRY_EN 的初始值由选项字节地址 0x1FFF F820 的值决定。当该地址的值为 0x1357 ECA8 时，ENCRY_EN 的初始值等于 1，否则其初始值为 0。

	该位可以通过软件操作来置位或清零。当软件往该寄存器写入 0x1357 ECA8 时置'1'；当软件写入其他值时清零。 • 0：不加密主 Flash 区 • 1：使能加密主 Flash 区 软件读该寄存器的返回值始终为 0。

3.4.12 解密控制寄存器（FLASH_DECRY_CFG）

偏移地址：0x7C

复位值：0x0000 0000

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

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

位 31:1	Res：保留 必须保持复位值。
位 0	DECRY_EN：解密主 Flash 区使能（Decryption enable in main flash memory） DECRY_EN 的初始值由选项字节地址 0x1FFF F824 的值决定。当 0x1FFF F824 地址的值为 0x2468 DB97 时，DECRY_EN 的初始值等于 1，否则其初始值为 0。 该位可以通过软件操作来置位或清零。当软件往该寄存器写入 0x2468 DB97 时置'1'；当软件写入其他值时清零。 • 0：不解密主 Flash 区 • 1：使能解密主 Flash 区 软件读该寄存器的返回值始终为 0。

3.4.13 密钥寄存器 1（FLASH_UKEYL）

偏移地址：0x80

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UKEY[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UKEY[15:0]															
rw															

位 31:0	UKEY[31:0]：用户密钥（Low 32-bit user key for Flash data encryption） 该位域可由软件配置。UKEY[63:0]的值由 UKEYH 和 UKEYL 寄存器共同决定。 读该寄存器的值始终为 0x0000 0000。

3.4.14 密钥寄存器 2（FLASH_UKEYH）

偏移地址：0x84

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UKEY[63:48]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UKEY[47:32]															
rw															

位 31:0	UKEY[63:32]：用户密钥（High 32-bit user key for Flash data encryption） 该位域可由软件配置。UKEY[63:0]的值由 UKEYH 和 UKEYL 寄存器共同决定。 读该寄存器的值始终为 0x0000 0000。

3.4.15 编程数据寄存器 0（FLASH_PW0）

偏移地址：0x90

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PW0[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PW0[15:0]															
rw															

位 31:0	PW0[31:0]：在双字或四字编程时，用于存储编程数据（Storing the programmed data when programming in 2-word or 4-word）

3.4.16 编程数据寄存器 1（FLASH_PW1）

偏移地址：0x94

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PW1[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PW1[15:0]															
rw															

位 31:0	PW1[31:0]：在双字或四字编程时，用于存储编程数据（Storing the programmed data when programming in 2-word or 4-word）

3.4.17 编程数据寄存器 2（FLASH_PW2）

偏移地址：0x98

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PW2[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PW2[15:0]															
rw															

位 31:0	PW2[31:0]：在双字或四字编程时，用于存储编程数据（Storing the programmed data when programming in 2-word or 4-word）

3.4.18 编程数据寄存器 3（FLASH_PW3）

偏移地址：0x9C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PW3[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PW3[15:0]															
rw															

位 31:0	PW3[31:0]：在双字或四字编程时，用于存储编程数据（Storing the programmed data when programming in 2-word or 4-word）

4 缓存器（Cache）

Cache 仅用于缓存指令，并不缓存数据。

4.1 Cache 特性

Cache 的主要特性如下：

- Cache 容量：1 Kbyte
- 使用 8 路组相联方式
- Cache 替换算法：采用“最近最少使用”替换方法
- 内置 Cache 指令访问命中次数计数器，方便用户统计 Cache 命中率

通过配置控制寄存器（CACHE_CTL）选择缓存哪种数据（Cache 可以缓存以下三种取指操作的数据）。同一时刻只能使能缓存一种取指访问数据。

- I-Bus 从片内 Flash 取指
- SYS-Bus 通过 QSPI 从外部 Flash 取指
- SYS-Bus 通过 FSMC 从外部 Flash 取指

4.2 CACHE 寄存器

基地址：0x4002 2000

空间大小：0x400

Cache 控制寄存器与 Flash 控制寄存器共享同一段地址空间：0x4002 2000 - 0x4002 23FF。

4.2.1 CACHE 控制寄存器（CACHE_CTL）

偏移地址：0xD0

复位值：0x0000 0401

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res		PR_CNT_CLR	PR_CNT_EN	FLUSH	PE_FLUSH_EN	Res	IV_CACHE_EN	Res					QSPI_CACHE_EN	FSMC_CACHE_EN	ICACHE_EN
		rw	rw	rw	rw		rw						rw	rw	rw

位 31:14	Res：保留 必须保持复位值。
位 13	PR_CNT_CLR：清空 Cache 命中次数计数器（Clear the Cache hit counter） • 1：清空 Cache 命中次数计数器，会把 Hit 和 Miss 计数器都清零。 • 0：无操作 读 PR_CNT_CLR 位始终返回 0。
位 12	PR_CNT_EN：使能 Cache 命中次数计数器（Enable the Cache hit counter） • 1：打开 Cache 命中次数计数器，Hit 和 Miss 计数器都开始累加计数。 • 0：关闭 Cache 命中次数计数器。
位 11	FLUSH：清空 Cache 内容（Clear the content of Cache）

	• 1：清空 Cache 内容 • 0：无操作 读 FLUSH 位始终返回 0。
位 10	PE_FLUSH_EN：Flash 执行 Program/Erase 操作时，清空 Cache 内容（Clear the content of Cache when programming or erasing Flash） • 1：在片内 Flash 执行 Program/Erase 操作时，清空 Cache 内容。 • 0：在片内 Flash 执行 Program/Erase 操作时，不清空 Cache 内容。
位 9	Res：保留 必须保持复位值。
位 8	IV_CACHE_EN：使能缓存中断向量表（NVIC table cache enable） • 1：使能缓存中断向量表 • 0：不缓存中断向量表 Cortex-M3 中，中断向量表是通过 I-Bus 读取的。Cache 默认不缓存中断向量表，因为一般来说取中断向量表后程序会立即跳转到中断服务程序，然后在中断服务程序的执行过程中很可能会把中断向量表从 Cache 中替换出来。
位 7:3	Res：保留 必须保持复位值。
位 2	QSPI_CACHE_EN：使能缓存通过 QSPI 接口访问外部 Flash 指令（Flash instructions accessed by Cache through the QSPI interface enable） • 1：使能缓存通过 QSPI 接口访问外部 Flash 指令 • 0：不缓存通过 QSPI 接口访问外部 Flash 指令 *注意：同一时刻，仅能配置 QSPI_CACHE_EN、FSMC_CACHE_EN 或 ICACHE_EN 的其中 1 位。*
位 1	FSMC_CACHE_EN：使能缓存通过 FSMC 接口访问外部 Flash 指令（Flash instructions accessed by Cache through the FSMC interface enable） • 1：使能缓存通过 FSMC 接口访问外部 Flash 指令 • 0：不缓存通过 FSMC 接口访问外部 Flash 指令
位 0	ICACHE_EN：使能缓存从 I-Bus 访问内部 Flash（Flash instructions accessed by Cache through the I-Bus interface enable） • 1：使能缓存从 I-Bus 访问内部 Flash 指令 • 0：关闭缓存从 I-Bus 访问内部 Flash 指令

4.2.2 CACHE 命中寄存器 1（CACHE_HIT_CNT_H）

偏移地址：0xD4

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res								HIT_CNT[55:48]							
								rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HIT_CNT[47:32]															
rw															

位 31:24	Res：保留 必须保持复位值。
位 23:0	HIT_CNT[55:32]：命中计数器的 32 至 55 位（Hit counter value 55 to 32 bit） *说明：HIT_CNT_H 和 HIT_CNT_L 联合使用。具体的使用请参考下一节 HIT_CNT_L。*

4.2.3 CACHE 命中寄存器 2（CACHE_HIT_CNT_L）

偏移地址：0xD8

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HIT_CNT[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HIT_CNT[15:0]															
rw															

位 31:0	HIT_CNT[31:0]：命中计数器的 0 至 31 位（Hit counter value 0 to 31 bit） 配置 PR_CNT_EN 为 1 使能计数器后，CPU 取指每命中 Cache 一次，HIT_CNT 计数器加 1。 通过对 PR_CNT_CLR 写 1，将 HIT_CNT 计数器清零。

4.2.4 CACHE 未命中寄存器 1（CACHE_MISS_CNT_H）

偏移地址：0xDC

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res								MISS_CNT[55:48]							
								rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MISS_CNT[47:32]															
rw															

位 31:24	Res：保留 必须保持复位值。
位 23:0	MISS_CNT[55:32]：未命中计数器的 32 至 55 位（Miss counter value 55 to 32 bit） *说明：MISS_CNT_H 和 MISS_CNT_L 联合使用。具体的使用请参考下一节 MISS_CNT_L。*

4.2.5 CACHE 未命中寄存器 2（CACHE_MISS_CNT_L）

偏移地址：0xE0

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MISS_CNT[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MISS_CNT[15:0]															
rw															

位 31:0	MISS_CNT[31:0]：未命中计数器的 0 至 31 位（MISS counter value 0 to 31 bit） 配置 PR_CNT_EN 为 1 使能计数器后，CPU 取指每未命中 Cache 一次，MISS_CNT 计数器加 1。 通过对 PR_CNT_CLR 写 1，将 MISS_CNT 计数器清零。 MISS_CNT 加上 HIT_CNT 的值等于这一段时间内 CPU 通过 Cache 取指的总次数。

5 CRC 计算单元（CRC）

循环冗余校验（Cyclic redundancy check，CRC）计算单元是根据一个固定的生成多项式和一个 32 位的数据得到任一 32 位字的 CRC 计算结果。

在其他的应用中，CRC 技术主要应用于验证数据传输或数据存储的正确性和完整性。在 EN/IEC60335-1 标准中提供了一种验证 Flash 存储器完整性的方法，即 CRC 计算单元在程序运行期间辅助计算软件的签名，并在软件连接期间将该签名与存储在指定存储位置的参考签名进行比较。

5.1 CRC 主要特性

- 使用 CRC-32（以太网）多项式：0x4C11DB7

  $X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^{8}+X^{7}+X^{5}+X^{4}+X^{2}+X+1$
- 单个 32 位输入/输出数据寄存器
- CRC 计算在 4 个 AHB 时钟周期（HCLK）内完成
- 通用 8 位寄存器（可用于存放临时数据）

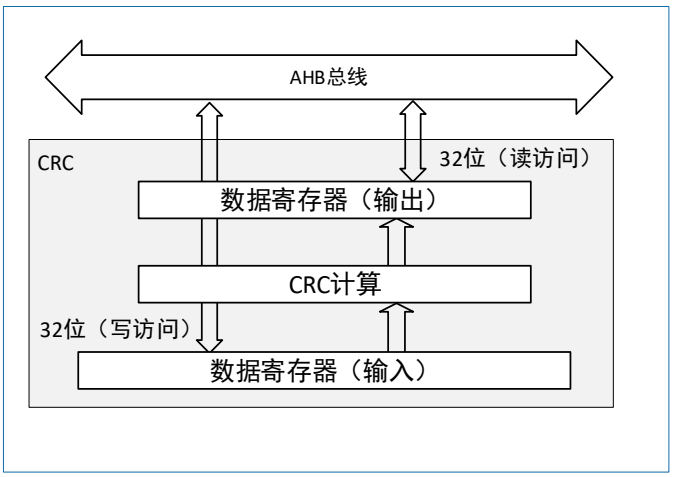


图 5-1 CRC 计算单元框图

5.2 CRC 功能描述

CRC 计算单元有 1 个 32 位数据寄存器：

- 向该寄存器写入数据时，它作为输入寄存器为 CRC 计算提供新数据。
- 当从该寄存器读数据时，它存储的是上一次 CRC 计算的结果。

每一次写入数据寄存器，其计算结果是前一次 CRC 计算结果和新计算结果的组合（对整个 32 位字进行 CRC 计算，而不是逐字节地计算）。

在 CRC 计算期间会暂停 CPU 的写操作，因此可以对寄存器 CRC_DR 进行背靠背写入或者连续地写和读操作。

可以通过设置寄存器 CRC_CR 的 RESET 位来重置寄存器 CRC_DR 为 0xFFFF FFFF。该操作不影响寄存器 CRC_IDR 内的数据。

5.3 CRC 寄存器

基地址：0x4002 3000

空间大小：0x400

CRC 计算单元包括 2 个数据寄存器和 1 个控制寄存器。

CRC 寄存器必须以字为单位进行访问。

5.3.1 数据寄存器（CRC_DR）

偏移地址：0x00

复位值：0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DR[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[15:0]															
rw															

位 31:0	DR[31:0]：数据寄存器（Data register） 该寄存器用于存放待计算的新数据，直接将其写入即可。读取该寄存器得到的是上次 CRC 计算的结果。

5.3.2 独立数据寄存器（CRC_IDR）

偏移地址：0x04

复位值：0x0000 0000

此寄存器不参与 CRC 计算，可以存放任何数据。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								IDR[7:0]							
								rw							

位 31:8	Res：保留 必须保持复位值。
位 7:0	IDR[7:0]：通用 8 位数据寄存器（General purpose 8-bit data register） 该寄存器可用作 1 个字节的临时存储。CRC_CR 寄存器中的 RESET 位引起的复位操作不会影响该寄存器。

5.3.3 控制寄存器（CRC_CR）

偏移地址：0x08

复位值：0x0000 0000

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

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

位 31:1	Res：保留 必须保持复位值。
位 0	RESET：复位控制（Reset control） 该位用于复位 CRC 计算单元，设置数据寄存器为 0xFFFF FFFF。只能对该位写'1'，它由硬件自动清零。

6 电源控制（PWR）

6.1 电源

器件的工作电压（VDD）为 2.0~3.6V。通过内置的电压调节器提供所需的内部 1.2 V 数字电源。当主电源 VDD 掉电后，通过备用电源 V_{BAT} 为实时时钟（RTC）和备份寄存器供电。

器件片内数字逻辑的电源由片内 LDO 提供。

片内 LDO 的输出电压可通过寄存器设置，以方便软件根据应用场景最大程度地优化芯片的电流功耗。芯片运行（Run）模式和停机（Stop）模式的 LDO 输出电压可以分别独立设置。

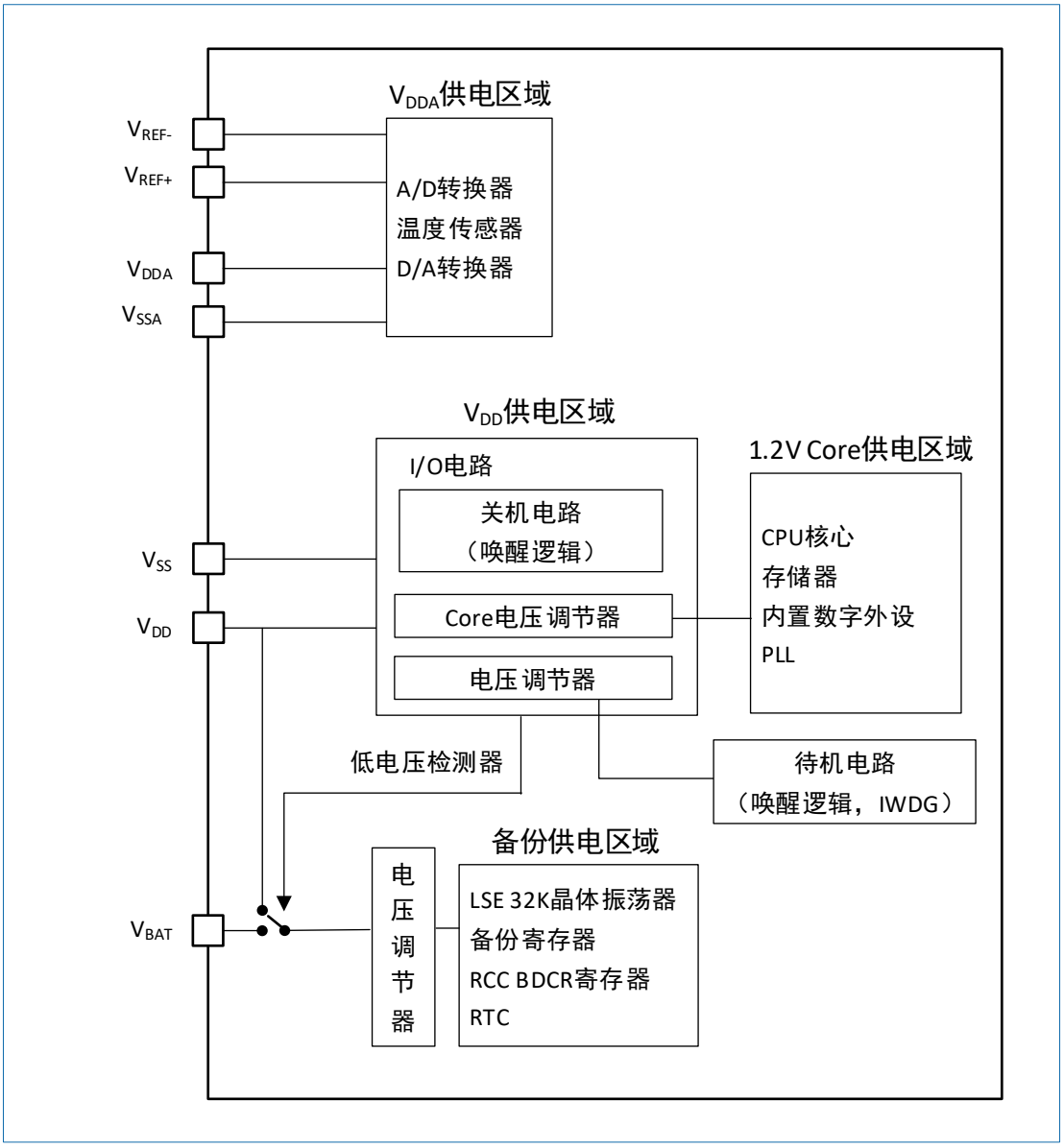


图 6-1 电源框图

说明：V_{DDA} 和 V_{SSA} 必须分别连接到 V_{DD} 和 V_{SS}。

6.1.1 独立的 ADC 和 DAC 供电和参考电压

为了提高转换精度，ADC 和 DAC 带一个独立的电源。该电源能过滤和屏蔽来自印刷电路板的噪声。

- ADC 的电源引脚为 V_{DDA}
- 独立的电源地连接至 V_{SSA} 引脚

若有 V_{REF-}引脚（根据封装而定），它必须连接到 V_{SSA}。

若有 V_{REF+}和 V_{REF-}引脚，它们在芯片内部与 ADC 的电源（V_{DDA}）和地（V_{SSA}）相联。

6.1.2 电池备份域

当 V_{DD} 断电时，为了保持备份寄存器的内容和 RTC 的供电，V_{BAT} 脚可连接至可选的待机电压。该待机电压由电池或其它电源提供。

V_{BAT} 脚为 RTC、LSE 振荡器和 PC13 至 PC15 端口供电，可以确保当主电源被切断时 RTC 能继续工作。切换到 V_{BAT} 供电的开关，由复位模块中的掉电复位（Power Down Reset，PDR）功能控制。

警告：

在复位期间（$t_{RSTTEMPO}$，即 V_{DD} 的上升时间）或检测到复位掉电信号（PDR）之后，V_{BAT} 和 V_{DD} 之间的电源开关仍会保持连接在 V_{BAT}。

在 V_{DD} 上升阶段，如果 V_{DD} 在小于 $t_{RSTTEMPO}$ 的时间内达到稳定状态（关于 $t_{RSTTEMPO}$ 数值可参考数据手册中的相关部分），且 V_{DD}>V_{BAT}+0.6V 时，电流可能通过 V_{DD} 和 V_{BAT} 之间的内部二极管注入到 V_{BAT}。如果与 V_{BAT} 连接的电源或者电池不能承受这样的注入电流，强烈建议在外部 V_{BAT} 和电源之间连接一个低压降二极管。

若在实际应用中没有外部电池，推荐 V_{BAT} 在外部连接至 V_{DD} 并连接一个 100 nF 的陶瓷滤波电容。当该备份域由 V_{DD}（内部模拟开关连到 V_{DD}）供电时，下述功能可用：

- PC14 和 PC15 可以用作 GPIO 或 LSE 引脚
- PC13 可以作为通用 I/O 口、TAMPER 引脚、RTC 校准时钟、RTC 闹钟或秒输出（可参考“7 备份寄存器（BKP）”）

说明：因为模拟开关只能通过少量的电流（3 mA），在输出模式下使用 PC13 至 PC15 口的功能是有限制的：速度必须限制在 2 MHz 以下，最大负载为 30pF，而且这些 I/O 口绝对不能用作电流源（如驱动 LED）。

当备份域由 V_{BAT} 供电时（V_{DD} 不存在，则模拟开关连接到 V_{BAT}），下述功能可用：

- PC14 和 PC15 只能用于 LSE 引脚
- PC13 可以作为 TAMPER 引脚、RTC 闹钟或秒输出（可参考“7.3.3 RTC 时钟校准寄存器（BKP_RTCCR）”）

6.1.3 Core 电压调节器

复位后 Core 电压调节器总是使能的。根据应用方式，它具有以下几种不同的模式工作。

- 运行（Run）模式：调节器以正常功耗模式提供 1.2V 电源（内核，内存和内置数字外设）。
- 停机（Stop）模式：调节器以低功耗模式提供 1.2V 电源，以保存寄存器和 SRAM 的内容。
- 待机（Standby）模式：调节器停止供电。除了备份电路和备份域外，寄存器和 SRAM 的内容全部丢失。
- 关机（Shutdown）模式：调节器停止供电，备份域也停止供电。

6.2 电源监控器

6.2.1 上电复位（POR）和掉电复位（PDR）

器件内部集成了上电复位（POR）和掉电复位（PDR）电路。当供电电压达到 POR/PDR 阈值，系统即能正常工作。

当 V_{DD}/V_{DDA} 低于指定的电压阈值 V_{POR}/V_{PDR} 时，系统保持复位状态，而无需外部复位电路。关于上电

复位和掉电复位阈值，请参考数据手册的电气特性部分。

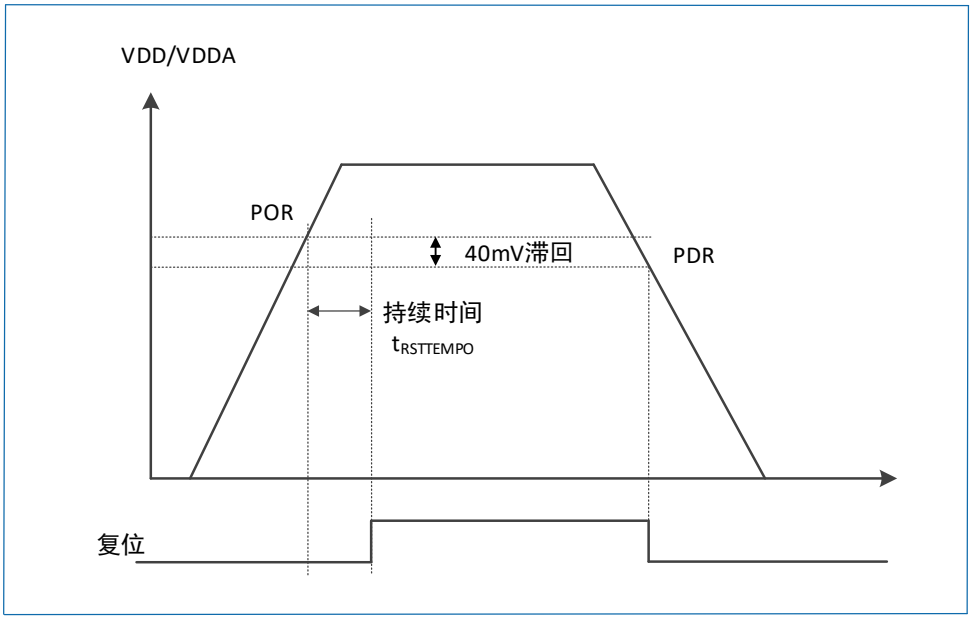


图 6-2 上电复位和掉电复位的波形图

6.2.2 可编程电压监测器（PVD）

可以使用可编程电压监测器（Programmable Voltage Detector，PVD）监测 V_{DD}/V_{DDA} 的供电。通过比较监测值与电源控制寄存器（PWR_CR）的 PLS[2:0]位所设置的阈值，即可监测。

通过设置（PWR_CR）的 PVDE 位以使能 PVD。

电源控制/状态寄存器（PWR_CSR）中的 PVDO 标志用于表示 V_{DD}/V_{DDA} 高于或低于 PVD 阈值。该事件在内部连接到外部中断的第 16 线（EXTI 线 16），如果在外部中断寄存器使能了该中断，则会产生中断。当 V_{DD}/V_{DDA} 下降到 PVD 阈值以下或当 V_{DD} 上升到 PVD 阈值以上时，根据 EXTI 线 16 上升沿/下降沿触发配置，则会产生 PVD 输出中断。例如，这一特性可用于服务程序执行紧急关闭任务。

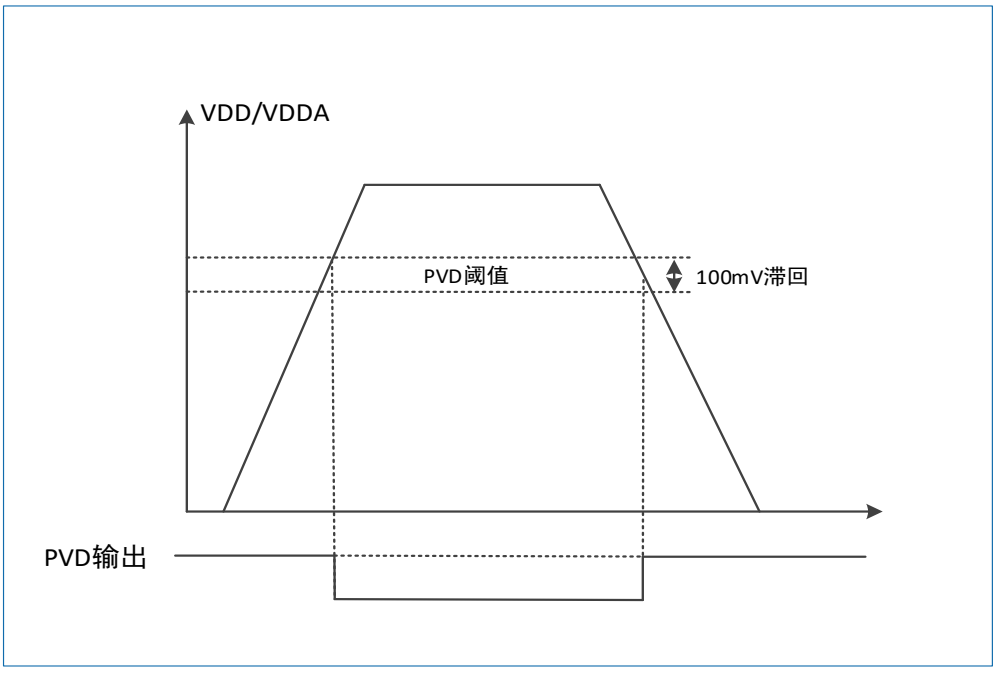


图 6-3 PVD 阈值

6.3 低功耗模式

在系统或电源复位以后，MCU 默认处于运行状态。当 CPU 不需继续运行时（例如在等待某个外部事件时），可使用几种低功耗模式以节省功耗。用户需要根据最低电源消耗、最短启动时间和可用的唤醒源等条件，选定一个最佳的低功耗模式。

该系列芯片支持以下几种低功耗模式：

- 睡眠（Sleep）模式：Cortex®-M3 内核停止，所有外设包括 Cortex-M3 核心的外设，如 NVIC、系统时钟（SysTick）等仍在运行。
- 停机（Stop）模式：CPU 核心停止，内核域所有的时钟都已停止。
- 待机（Standby）模式：1.2 V 电源域关闭。
- 关机（Shutdown）模式：内部所有 LDO 关闭，包括备份域供电默认也关闭。

此外，在运行（Run）模式下，可以通过以下任一方式降低功耗：

- 降低系统时钟频率
- 关闭 APB 和 AHB 总线上未被使用的外设时钟。

表 6-1 低功耗模式一览

工作模式	进入条件	唤醒条件	内部核电源时钟状态	V_{DD} 主区域时钟状态	电压调节器状态
睡眠模式（Sleep）	1. 设置 PWR_CR:LPDS=0 和 PWR_CR:PDDS=0。 2. 软件执行 WFI/WFE 指令进入。	由任何一个普通 IRQ 中断事件唤醒，包括 System Tick。	CPU 时钟关闭，对其他时钟和 ADC 时钟无影响	开启	开启
停机模式（Stop）	1. 设置 PWR_CR:PDDS=0。 2. 设置 Cortex-M3 系统控制寄存器的 SLEEPDEEP 位。 3. 软件执行 WFI/WFE 指令进入。	由任何一个 EXTI 外部中断线唤醒。支持定时 ADC 采样预唤醒。当满足条件后真正唤醒。	所有时钟停止	HSI 和 HSE 关闭	开启或者处于低功耗模式（在 PWR_CR 中设置）
待机模式（Standby）	1. 设置 PWR_CR:LPDS=0 和 PWR_CR:PDDS=1。 2. 设置 Cortex-M3 系统控制寄存器的 SLEEPDEEP 位。 3. 软件执行 WFI/WFE 指令进入。 4. 清除电源控制/状态寄存器（PWR_CSR）中的 WUF 位。	支持 3 个可配置极性的外部唤醒引脚（WKUPx）、RTC 闹钟唤醒，以及 IWDG 复位唤醒。	所有时钟停止	HSI 和 HSE 关闭	关闭
关机模式（Shutdown）	1. 设置 PWR_CR:LPDS =0、PWR_CR:PDDS =1 和 PWR_CSR2:SHDS = 1。 2. 设置 Cortex-M3 系统控制寄存器的 SLEEPDEEP 位。 3. 软件执行 WFI/WFE 指令进入。	支持 3 个可配置极性的外部唤醒引脚（WKUPx）、RTC 闹钟唤醒。	所有时钟停止	所有时钟停止	关闭

6.3.1 降低系统时钟频率

在运行模式下，通过编程预分频寄存器可以降低任意一个系统时钟（SYSCLK、HCLK、PCLK1、PCLK2）的频率。在进入睡眠模式之前，也可以利用预分频器来降低外设的时钟频率。更多信息，可参考章节：“8.3.2 时钟配置寄存器（RCC_CFGR）”。

6.3.2 外部时钟的控制

在运行模式下，任何时候都可以通过停止为外设和内存提供时钟（HCLK 和 PCLKx）来减少功耗。为了在睡眠模式下进一步减少功耗，可在执行 WFI 或 WFE 指令前关闭所有外设的时钟。

通过设置 AHB 外设时钟使能寄存器（RCC_AHBENR）、APB2 外设时钟使能寄存器（RCC_APB2ENR）和 APB1 外设时钟使能寄存器（RCC_APB1ENR）来开关各个外设的时钟。

6.3.3 睡眠（Sleep）模式

6.3.3.1 进入睡眠模式

通过执行 WFI 或 WFE 指令进入睡眠状态。根据 Cortex®-M3 系统控制寄存器中的 SLEEPONEXIT 位的值，提供两种选项选择睡眠模式的进入机制：

- SLEEP-NOW：如果 SLEEPONEXIT 位被清除，当执行 WFI 或 WFE 指令时，MCU 立即进入睡眠模式。
- SLEEP-ON-EXIT：如果 SLEEPONEXIT 位被置位，系统从最低优先级的中断服务程序（ISR）中退出时，MCU 立即进入睡眠模式。

在睡眠模式下，所有的 I/O 引脚都保持在它们运行模式时的状态。

若想了解如何进入睡眠模式，可参考表 6-2 和表 6-3。

6.3.3.2 退出睡眠模式

如果执行 WFI 指令进入睡眠模式，任意一个被嵌套向量中断控制器（Nested Vectored Interrupt Controller，NVIC）响应的外设中断都能将系统从睡眠模式唤醒。

如果执行 WFE 指令进入睡眠模式，则一旦发生唤醒事件时，MCU 将从睡眠模式退出。唤醒事件可以通过下述方式产生：

- 在外设控制寄存器中使能一个中断，而不是在 NVIC 中使能，并且在 Cortex®-M3 系统控制寄存器中使能 SEVONPEND 位。当 MCU 从 WFE 中唤醒后，外设的中断挂起位和外设的 NVIC 中断通道挂起位（在 NVIC 中断清除挂起寄存器中）必须被清除。
- 配置一个外部或内部的 EXTI 线为事件模式。当 MCU 从 WFE 中唤醒后，因为与事件线对应的挂起位未被设置，不必清除外设的中断挂起位或 NVIC 中断请求（IRQ）通道挂起位。

该模式唤醒所需的时间最短，因为没有时间损失在中断的进入或退出上。若需了解如何退出睡眠模式，可参考表 6-2 和表 6-3。

表 6-2 SLEEP-NOW 模式

SLEEP-NOW 模式	说明
进入条件	在以下条件下执行 WFI（等待中断）或 WFE（等待事件）指令： SLEEPDEEP=0 和 SLEEPONEXIT=0 参考 Cortex®-M3 系统控制寄存器。
退出条件	● 如果执行 WFI 进入睡眠模式： 中断：参考“9.1.2 中断和异常向量”。 ● 如果执行 WFE 进入睡眠模式： 唤醒事件：参考“9.2.4 唤醒事件管理”。
唤醒延时	-

表 6-3 SLEEP-ON-EXIT 模式

SLEEP-ON-EXIT 模式	说明
进入条件	在以下条件下执行 WFI 指令： SLEEPDEEP=0 和 SLEEPONEXIT=1 参考 Cortex®-M3 系统控制寄存器。
退出条件	中断：参考“9.1.2 中断和异常向量”。
唤醒延时	-

6.3.4 停机（Stop）模式

停机模式是基于结合了外设时钟门控的 Cortex®-M3 的深睡眠模式。在停机模式下，电压调节器可配置为正常或低功耗模式。在该模式下，Core 供电域的所有时钟都被停止，PLL、HSI 和 HSE RC 振荡器被禁用，SRAM 和寄存器内容被保留下来。

在停机模式下，所有的 I/O 引脚都保持运行模式下的状态。

6.3.4.1 进入停机模式

若需了解如何进入停机模式，可参考表 6-1。

为了进一步降低停机模式下的功耗，可将内部电压调节器设置为低功耗模式。通过配置电源控制寄存器（PWR_CR）中的 LPDS 位来实现。

如果正在进行 Flash 编程，直到对 Flash 访问完成，系统才进入停机模式。如果正在进行对 APB 访问，直到对 APB 访问完成，系统才进入停机模式。

在停机模式下，通过对独立的控制位进行编程，可选择以下功能：

- 独立看门狗（IWDG）：可通过写入看门狗的键寄存器或硬件选择来启动 IWDG。一旦启动了 IWDG，除了系统复位，它不能被停止。详见章节：“20.2 IWDG 功能描述”。
- 实时时钟（RTC）：通过备份域控制寄存器（RCC_BDCR）的 RTCEN 位来设置。
- 内部 RC 振荡器（LSI RC）：通过控制/状态寄存器（RCC_CSR）的 LSION 位来设置。
- 外部 32.768 kHz 振荡器（LSE OSC）：通过备份域控制寄存器（RCC_BDCR）的 LSEON 位设置。

在停机模式下，如果在进入该模式前未关闭 ADC 和 DAC，则这些外设仍然耗电。通过设置寄存器 ADC_CR2 的 ADON 位和寄存器 DAC_CR 的 ENx 位为 0 可关闭这 2 个外设。

6.3.4.2 退出停机模式

关于如何退出停机模式，可参考表 6-1。当一个中断或唤醒事件导致退出停机模式时，HSI RC 振荡器被选为系统时钟。

当电压调节器处于低功耗模式下，当系统从停机模式退出时，将会有一段额外的启动延时。如果在停机模式期间保持内部调节器开启，则退出启动时间会缩短，但相应的功耗会增加。

6.3.5 待机（Standby）模式

待机模式可实现系统的最低功耗。该模式在 Cortex®-M3 深睡眠模式时关闭电压调节器，整个 Core 供电域被断电，PLL、HSI 和 HSE 振荡器也被断电，SRAM 和寄存器内容丢失，仅备份域的寄存器和待机电路维持供电（见图 6-1）。

6.3.5.1 进入待机模式

关于如何进入待机模式，可参考表 6-1。可以通过设置独立的控制位，选择以下待机模式的功能：

- 独立看门狗（IWDG）：可通过写入看门狗的键寄存器或硬件选择来启动 IWDG。一旦启动了独立看门狗，除了系统复位外，它不能被停止。详见章节：“20.2 IWDG 功能描述”。
- 实时时钟（RTC）：通过备份域的控制寄存器（RCC_BDCR）的 RTCEN 位来设置。
- 内部 RC 振荡器（LSI RC）：通过控制/状态寄存器（RCC_CSR）的 LSION 位来设置。
- 外部 32.768 kHz 振荡器（LSE）：通过备份域的控制寄存器（RCC_BDCR）的 LSEON 位设置。

6.3.5.2 退出待机模式

当一个外部复位（NRST 引脚）、IWDG 复位、WKUP 引脚上的上升沿或 RTC 闹钟事件的上升沿发生时 MCU 从待机模式退出。从待机模式唤醒后，除了电源控制/状态寄存器（PWR_CSR），所有寄存器被复位。

从待机模式唤醒后，代码执行与重启后以相同的方式重新启动（采样启动模式引脚、读取复位向量等）。电源控制/状态寄存器（PWR_CSR）将会指示内核由待机状态退出。

关于如何退出待机模式，详见表 6-1。

6.3.5.3 待机模式下的输入/输出端口状态

在待机模式下，所有的 I/O 引脚处于高阻态，除了以下的引脚：

- 复位引脚（始终有效）
- 被设置为防侵入或校准输出时的 TAMPER 引脚
- 被使能的唤醒引脚

6.3.6 关机（Shutdown）模式

关机模式可实现系统的最低功耗。该模式是在 Cortex-M3 深睡眠模式时关闭电压调节器。整个 1.2 V 供电区域被断电。PLL、HSI 和 HSE 的 RC 振荡器也被断电。SRAM 和寄存器内容丢失。备份域是否供电可配置，仅关机唤醒电路可以工作（参见图 6-1）。

6.3.6.1 进入关机模式

关于如何进入关机模式，参见表 6-1。可以通过设置独立的控制位，选择以下关机模式的功能：

- 独立看门狗（IWDG）：可通过写入看门狗的关键字寄存器或硬件选择来启动 IWDG。独立看门狗一旦启动，除了系统复位，它不能被停止。参见“20.2 IWDG 功能描述”。
- 实时时钟（RTC）：通过备份域控制寄存器（RCC_BDCR）的 RTCEN 位来设置。
- 内部 RC 振荡器（LSI RC）：通过控制/状态寄存器（RCC_CSR）的 LSION 位来设置。
- 外部 32.768kHz 振荡器（LSE）：通过备用区域控制寄存器（RCC_BDCR）的 LSEON 位设置。

6.3.6.2 退出关机模式

当一个外部复位（NRST 引脚）、IWDG 复位、WKUP 引脚上的上升沿或 RTC 闹钟事件的上升沿发生时，MCU 从关机模式退出。从关机唤醒后，除了电源控制/状态寄存器（PWR_CSR），所有寄存器被复位。

从关机模式唤醒后，程序以复位后重启一样的方式开始执行。（采样启动模式引脚、读取复位向量等）。电源控制/状态寄存器（PWR_CSR2）将会指示内核由关机状态退出。

关于如何退出关机模式，参见表 6-1。

6.3.7 丰富的唤醒源

- 睡眠（Sleep）模式下，可以由任何一个普通 IRQ 中断事件唤醒。
- 停机（Stop）模式下，可以由任何一个 EXTI 外部中断线唤醒。同时，支持定时 ADC 采样预唤

醒，当满足条件后真正唤醒。

- 待机（Standby）模式下，支持 3 个可配置极性的外部唤醒引脚以及 RTC 闹钟唤醒。
- 关机（Shutdown）模式：支持 3 个可配置极性的外部唤醒引脚以及 RTC 闹钟唤醒。

6.3.8 调试模式

默认情况下，如果调试 MCU 时，使 MCU 进入停机或待机模式将失去调试连接。这是因为 Cortex®-M3 的内核失去了时钟。

然而，通过设置 DBGMCU_CR 寄存器中的某些配置位，可以在低功耗模式下使用调试软件。更多的细节请参考章节：“36.15.1 低功耗模式的调试支持”。

6.3.9 低功耗模式下的自动唤醒（AWU）

RTC 可以在不需要依赖外部中断的情况下唤醒低功耗模式下的 MCU（自动唤醒模式）。RTC 提供一个可编程的时间基数，用于周期性从停机或待机模式下唤醒。通过编程备份域控制寄存器（RCC_BDCR）的 RTCSEL[1:0]位，以下两个时钟源可用作低功耗模式下的 RTC 时钟。

- 低功耗 32.768 kHz 外部晶振（LSE）：该时钟源提供了一个低功耗且精确的时间基准。（在典型情形下消耗小于 1 μA）
- 低功耗内部 RC 振荡器（LSI RC）

  使用该时钟源，节省了一个 32.768 kHz 晶振的成本。但是 RC 振荡器将少许增加电源消耗。为了用 RTC 闹钟事件将系统从停机模式下唤醒，必须进行如下操作：

  - 配置外部中断线 17 为上升沿触发。
  - 配置 RTC 使其可产生 RTC 闹钟事件。

如果要从待机模式中唤醒，则不必配置外部中断线 17。

6.4 PWR 寄存器

基地址：0x4000 7000

空间大小：0x400

可以按半字（16 位）或字（32 位）的方式操作 PWR 寄存器。

6.4.1 电源控制寄存器（PWR_CR）

偏移地址：0x00

复位值：0x0000 0000

说明：该寄存器从待机模式唤醒时清除

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res							DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
							rw	rw			rw	rw	rw	rw	rw

位 31:9	Res：保留 必须保持复位值。
位 8	DBP：取消备份域的写保护（Disable RTC domain write protection） 在复位后，RTC 和备份寄存器处于被保护状态以防意外写入。设置该位以允许写入这些寄存器。 • 0：禁止写入 RTC 和备份寄存器

	• 1：允许写入 RTC 和备份寄存器 *说明：如果 RTC 的时钟是 HSE/128，该位必须保持为'1'。*
位 7:5	PLS[2:0]：PVD 电平选择（PVD level selection） 该位域由软件写入，以选择电源电压监测器所监测的电压阈值。 • 000：2.2 V • 001：2.3 V • 010：2.4 V • 011：2.5 V • 100：2.6 V • 101：2.7 V • 110：2.8 V • 111：2.9 V *说明：详细说明参见数据手册中的电气特性部分。*
位 4	PVDE：PVD 使能（Power voltage detector enable） • 0：禁止 PVD • 1：开启 PVD
位 3	CSBF：清除待机位（Clear standby flag） 读该位始终为 0。 • 0：无功效 • 1：清除 SBF 待机位（写）
位 2	CWUF：清除唤醒位（Clear wakeup flag） 读该位始终为 0。 • 0：不生效 • 1：2 个系统时钟周期后清除 WUF 唤醒位（写）
位 1	PDDS：掉电深睡眠 （Power down deep sleep） 该位与 LPDS 位协同操作。该位由软件设置和清零。 • 0：当 CPU 进入深睡眠时，进入停机模式。停机模式下，电压调压器的状态由 LPDS 位控制。 • 1：当 CPU 进入深睡眠时，进入待机模式。
位 0	LPDS：低功耗深睡眠 （Low-power deep sleep） 该位与 PDDS 位协同操作，当 PDDS=0 时： • 0：在停机模式下，电压调压器开启。 • 1：在停机模式下，电压调压器处于低功耗模式。

6.4.2 电源控制/状态寄存器（PWR_CSR）

偏移地址：0x04

复位值：0x0000 0000

与标准的 APB 读相比，读此寄存器需要额外的 APB 周期。该寄存器从待机模式唤醒时不复位。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res					EWUP3	EWUP2	EWUP1	Res				LDORDY	PVDO	SBF	WUF
					rw	rw	rw					r	r	r	r

位 31: 11	Res：保留 必须保持复位值。
位 10	EWUP3：使能 WKUP3 引脚（Enable WKUP3 pin） 外部唤醒引脚 3 使能。可从待机（Standby）或关机（Shutdown）模式唤醒。
位 9	EWUP2：使能 WKUP2 引脚（Enable WKUP2 pin） 外部唤醒引脚 2 使能。可从待机（Standby）或关机（Shutdown）模式唤醒。
位 8	EWUP1：使能 WKUP1 引脚（Enable WKUP1 pin） • 0：WKUP1 引脚为通用 I/O。WKUP1 引脚上的事件不能将 CPU 从待机模式唤醒。 • 1：WKUP1 引脚用于将 CPU 从待机（Standby）或关机（Shutdown）模式唤醒，WKUP1 引脚被强置为输入下拉的配置（WKUP1 引脚上的上升沿将系统从待机模式唤醒）。
位 7:4	Res：保留 必须保持复位值。
位 3	LDORDY：LDO 准备就绪状态标志（LDO ready flag） • 0：表示内部 LDO 不能带动重负载。不允许主频切换至高速时钟。 • 1：表示内部 LDO 稳定，可带动重负载。此时允许主频切换至高速时钟。
位 2	PVDO：PVD 输出（PVD output） PVDE 置位（PVD 使能）后，该位有效。 • 0：V_{DD}/V_{DDA} 高于由 PLS[2:0]选定的 PVD 阈值。 • 1：V_{DD}/V_{DDA} 低于由 PLS[2:0]选定的 PVD 阈值。 *说明：在待机模式下，PVD 被停止。因此，待机模式后或复位直到 PVDE 置位之前，该位为 0。*
位 1	SBF：待机标志（Standby flag） 该位由硬件设置，并只能由 POR/PDR（上电/掉电复位）或设置电源控制寄存器（PWR_CR）的 CSBF 位清除。 • 0：系统未处于待机模式 • 1：系统处于待机模式
位 0	WUF：唤醒标志（Wakeup flag） 该位由硬件设置，并只能由 POR/PDR（上电/掉电复位）或设置电源控制寄存器（PWR_CR）的 CWUF 位清除。 • 0：未发生唤醒事件 • 1：在 WKUP 引脚上发生唤醒事件或出现 RTC 闹钟事件 *说明：当 WKUP 引脚已经是高电平时，在（通过设置 EWUP 位）使能 WKUP 引脚时，会检测到一个额外的事件。*

6.4.3 电源控制/状态寄存器 2（PWR_CSR2）

偏移地址：0x30

复位值：0x0000 0000

说明：该寄存器从待机模式唤醒时清除

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EWUP_RTC	Res							SHDS	BKPPDS	Res		CSHUTF	CSWUF	SHUTF	SWUF
rw								rw	rw			w1	w1	r	r

位 31:16	Res：保留 必须保持复位值。
位 15	EWUP_RTC：关机模式下，RTC 唤醒使能（Enable RTC wakeup in shutdown mode） • 0：关机模式下，不使能 RTC 唤醒（仅用于关机模式）。 • 1：关机模式下，RTC 唤醒使能（仅用于关机模式）。
位 14:8	Res：保留 必须保持复位值。
位 7	SHDS：关机模式使能位（Shutdown mode enable ） • 0：如果 PWR_CR:PDDS = 1，CPU 执行 WFE/WFI 后不会进入关机模式。 • 1：如果 PWR_CR:PDDS = 1，CPU 执行 WFE/WFI 后进入关机模式。
位 6	BKPPDS：备份（BKP）域掉电控制（Power down control in the BKP area） 在关机模式下，允许关闭 BKP 以及 RTC 域的电源以进一步降低功耗。 • 0：当进入关机时，不会关闭 BKP 和 RTC 电源。 • 1：当进入关机时，自动关闭 BKP 和 RTC 电源。
位 5:4	Res：保留 必须保持复位值。
位 3	CSHUTF：清除 SHUTF 标志（Clear Shutdown Flag） 写 1 清除 PWR_CSR2:SHUTF 标志。
位 2	CSWUF：清除 SWUF 标志（Clear Shutdown wakeup flag） 写 1 清除 PWR_CSR2:SWUF 标志。
位 1	SHUTF：关机标志（Shutdown flag） 该标志是在进入关机模式时，被硬件置位。通过设置 CSHUTF 位可以清除该位（从关机唤醒不会复位此位）。 • 0：曾经未进入过关机模式。 • 1：曾经进入过关机模式。
位 0	SWUF：关机唤醒标志（Shutdown wakeup flag） 当芯片在关机模式下，收到一个来自 WKUP 引脚或 RTC 唤醒事件后，该标志被置位。通过设置 CSWUF 位可以清除该位（从关机唤醒不会复位此位）。 如果是被 NRST 复位引脚产生的唤醒，则此寄存器位不会置 1。 • 0：没有唤醒发生过。 • 1：有 RTC 或者 WKUP 引脚的唤醒事件发生过。

6.4.4 唤醒引脚极性寄存器（PWR_WUP_POL）

偏移地址：0x34

复位值：0x0000 0000

说明：该寄存器从待机模式唤醒时清除

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res													WUPOL3	WUPOL2	WUPOL1
													rw	rw	rw

位 31:3	Res：保留 必须保持复位值。
位 2	WUPOL3：WKUP3 引脚极性选择（Wakeup pin 3 polarity selection） • 0：WKUP3 为内部下拉，上升沿唤醒。 • 1：WKUP3 为内部上拉，下降沿唤醒。
位 1	WUPOL2：WKUP2 引脚极性选择（Wakeup pin 2 polarity selection） • 0：WKUP2 为内部下拉，上升沿唤醒。 • 1：WKUP2 为内部上拉，下降沿唤醒。
位 0	WUPOL1：WKUP1 引脚极性选择 （Wakeup pin 1 polarity selection） • 0：WKUP1 为内部下拉，上升沿唤醒。 • 1：WKUP1 为内部上拉，下降沿唤醒。

7 备份寄存器（BKP）

备份寄存器由 43 个 16 位的寄存器组成，用于存储用户应用数据（共 86 个字节）。这些寄存器工作在备份域。当 V_{DD} 电源被切断，备份域通过 V_{BAT} 保持供电。当器件从待机模式或者通过系统复位/电源复位唤醒时，备份寄存器不会被复位。

此外，BKP 控制寄存器用来管理侵入检测和 RTC 校准功能。

复位后，禁止访问备份寄存器和 RTC，并且备份域被保护以防止发生可能的意外写操作。使能备份寄存器和 RTC 的访问，其步骤如下：

1. 设置寄存器 RCC_APB1ENR 的 PWREN 和 BKPEN 位以打开电源和备份接口时钟。
2. 设置电源控制寄存器（PWR_CR）的 DBP 位，以使能对备份寄存器和 RTC 的访问。

7.1 BKP 特性

- 用于管理防侵入检测并具有中断功能的控制/状态寄存器
- 用于存储 RTC 校验值的校准寄存器
- 在 PC13 引脚（当该引脚不用于侵入检测时）上输出 RTC 校准时钟、RTC 闹钟脉冲或者秒脉冲
- 提供 43 个备份域数据寄存器（BKP_DR0~BKP_DR42），共计 86 个字节

7.2 BKP 功能描述

7.2.1 侵入检测

当 TAMPER 引脚上的信号从’0’变成’1’，或者从’1’变成’0’（取决于备份控制寄存器 BKP_CR 的 TPAL 位），会产生一个侵入检测事件。侵入检测事件会复位所有的备份寄存器。

然而为了避免丢失侵入事件，通过将用于边沿检测的信号和侵入使能信号进行逻辑与运算来检测一个侵入事件。这使得在 TAMPER 引脚被使能之前，产生的侵入事件也能被检测到。

- **当 TPAL=0 时：**在 TAMPER 引脚的侵入检测功能被使能（通过设置 TPE 位）前，若该引脚已为高电平，一旦使能 TAMPER 引脚则可检测额外的侵入事件（尽管在 TPE 位置’1’后并没有出现上升沿）。
- **当 TPAL=1 时：**在 TAMPER 引脚的侵入检测功能被使能（通过设置 TPE 位）前，若该引脚已为低电平，一旦使能 TAMPER 引脚则可检测额外的侵入事件（尽管在 TPE 位置’1’后并没有出现下降沿）。

通过设置 BKP_CSR 寄存器的 TPIE 位为’1’，当检测到侵入事件时就会产生一个中断。

在检测到并清除一个侵入事件后，TAMPER 引脚应该被禁用。然后，在再次写备份域数据寄存器（BKP_DRx）之前，通过配置 TPE 位重新使能 TAMPER 引脚。这样，可以阻止软件在侵入检测引脚上仍然有侵入事件时对备份数据寄存器进行写操作。这相当于对侵入引脚 TAMPER 进行电平检测。

说明：当 V_{DD} 电源断开时，侵入检测功能仍然有效。为了避免不必要地复位数据备份寄存器，TAMPER 引脚应该在片外连接到正确的电平。

7.2.2 RTC 校准

为方便测量，RTC 时钟可以经 64 分频输出到 TAMPER 引脚上。通过设置 RTC 时钟校准寄存器（BKP_RTCCR）的 CCO 位使能该功能。

通过配置 CAL[6:0]位，此时钟可以最多减慢 121 ppm。

7.3 BKP 寄存器

基地址：0x4000 6C00

空间大小：0x400

BKP 寄存器可以半字（16 位）或字（32 位）为单位进行访问。

7.3.1 备份域数据寄存器 0（BKP_DR0）

偏移地址：0x00

复位值：0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D[15:0]															
rw															

位 15:0	D[15:0]：备份数据（Backup data） 可写入用户数据至该位域。 *注意：BKP_DR0 寄存器不会被系统复位、电源复位或从待机模式唤醒所复位。它们可以由备份域复位或 TAMPER 引脚事件（如果 TAMPER 引脚的侵入检测功能被开启）复位。*

7.3.2 备份域数据寄存器 x（BKP_DRx）（x=1..10）

偏移地址：0x04*x

复位值：0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D[15:0]															
rw															

位 15:0	D[15:0]：备份数据（Backup data） 可写入用户数据至该位域。 *注意：BKP_DRx 寄存器不会被系统复位、电源复位或从待机模式唤醒所复位。它们可以由备份域复位或 TAMPER 引脚事件（如果 TAMPER 引脚的侵入检测功能被开启）复位。*

7.3.3 RTC 时钟校准寄存器（BKP_RTCCR）

偏移地址：0x2C

复位值：0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res						ASOS	ASOE	CCO	CAL[6:0]						
						rw	rw	rw	rw						

位 15:10	Res：保留 必须保持复位值。
位 9	ASOS：闹钟或秒输出选择（Alarm or second output selection） 当 ASOE 被置位，ASOS 位可用于选择在 TAMPER 引脚上输出的是 RTC 秒脉冲还是闹钟脉冲信号。 • 0：输出 RTC 闹钟脉冲 • 1：输出秒脉冲 *说明：该位只能通过备份域复位来复位。*
位 8	ASOE：闹钟或秒输出使能（Alarm or second output enable） 根据 ASOS 位的设置，配置该位可输出 RTC 闹钟或秒脉冲到 TAMPER 引脚上。输出脉冲的持续时间为一个

	RTC 时钟的周期。设置了 ASOE 位时不能开启 TAMPER 的功能。 *说明：该位只能被备份域的复位所清除。*
位 7	CCO：校准时钟输出（Calibrated clock output） • 0：无影响 • 1：此位置'1'可以在侵入检测引脚输出经 64 分频后的 RTC 时钟。当 CCO 位置'1'时，必须关闭侵入检测功能以避免检测到无用的侵入信号。 *说明：当 V_{DD} 供电断开时，该位被清除。*
位 6:0	CAL[6:0]：校准值（Calibration value） 校准值表示在每 2^{20} 个时钟脉冲内被忽略的脉冲个数。这可以用来对 RTC 进行校准，2^{20} 个时钟脉冲内被忽略的脉冲个数。 RTC 时钟可以被减慢 0 ~ 121ppm。

7.3.4 备份控制寄存器（BKP_CR）

偏移地址：0x30

复位值：0x0000 0000

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

位 15:2	Res：保留 必须保持复位值。
位 1	TPAL：TAMPER 引脚有效电平（Effective level on the TAMPER pin） • 0：TAMPER 引脚上的高电平会复位所有数据备份寄存器（如果 TPE 位置为 1） • 1：TAMPER 引脚上的低电平会复位所有数据备份寄存器（如果 TPE 位置为 1）
位 0	TPE：启动 TAMPER 引脚（Enable the Tamper function on the TAMPER pin） • 0：TAMPER 引脚用作通用 IO 口 • 1：开启 TAMPER 引脚的侵入检测复用功能

说明：同时设置 TPAL 和 TPE 位总是安全的。然而，同时清除两者会产生一个假的侵入事件。因此，推荐仅在 TPE 为 0 时改变 TPAL 位的值。

7.3.5 备份控制/状态寄存器（BKP_CSR）

偏移地址：0x34

复位值：0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res						TIF	TEF	Res					TPIE	CTI	CTE
						r	r						rw	w	w

位 15:10	Res：保留 必须保持复位值。
位 9	TIF：侵入中断标志（Tamper interrupt flag） 当检测到侵入事件且 TPIE 位为 1 时，此位由硬件置'1'。通过向 CTI 位写 1 来清除此位（同时也清除了中断）。如果 TPIE 位被复位，则此位也会被清除。 • 0：无侵入中断

	• 1：产生侵入中断 *注意：该位仅由系统复位或待机模式唤醒复位。*
位 8	TEF：侵入事件标志（Tamper event flag） 当检测到侵入事件时，此位由硬件置'1'。通过向 CTE 位写 1 可清除此位。 • 0：无侵入事件 • 1：检测到侵入事件 *说明：侵入事件会复位所有的 BKP_DRx 寄存器。只要 TEF 为 1，所有的 BKP_DRx 寄存器就一直保持复位状态。当此位被置'1'时，若对 BKP_DRx 进行写操作，写入的值不会被保存。*
位 7:3	Res：保留 必须保持复位值。
位 2	TPIE：使能侵入 TAMPER 引脚中断（TAMPER interrupt detection enable） • 0：禁止侵入检测中断 • 1：使能侵入检测中断（BKP_CR 寄存器的 TPE 位也必须被置'1'） *说明：* • *侵入中断无法将系统内核从低功耗模式唤醒* • *仅当系统复位或由待机模式唤醒后才复位该位*
位 1	CTI：清除侵入检测中断（TAMPER interrupt detection flag clear） 此位只能写入，读该位值为 0。 • 0：无效 • 1：清除侵入检测中断和 TIF 侵入检测中断标志。
位 0	CTE：清除侵入检测事件（TAMPER event detection flag clear） 此位只能写入，读该位值为 0。 • 0：无效 • 1：清除 TEF 侵入检测事件标志（并且复位侵入检测器）

7.3.6 唤醒定时器控制寄存器（BKP_WUTCR）

偏移地址：0x38

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WUTE	Res			WUTO[3:0]				Res					WUCKSEL[2:0]		
rw				rw									rw		

位 31:16	Res：保留 必须保持复位值。
位 15	WUTE：使能唤醒定时器（Wakeup timer enable） • 0：禁用唤醒定时器 • 1：使能唤醒定时器
位 14:12	Res：保留

	必须保持复位值。
位 11:8	WUTO[3:0]：唤醒定时器输出选择（The output of wakeup timer selection） WUTO 的 4 位可独立工作，也可组合选择。 WUTO[0]： • 0：如果 ASOS、ASOE 和 CCO 都为 0 且 WUTO[0]=0，则唤醒定时器从 PC13 输出。 • 1：如果 ASOS、ASOE 和 CCO 都为 0 且 WUTO[0]=1，则唤醒定时器不从 PC13 输出。 WUTO[1]： • 0：唤醒定时器不作为待机（Standby）模式的唤醒源。 • 1：唤醒定时器作为待机（Standby）模式的唤醒源。 WUTO[2]： • 0：唤醒定时器不输出到 EXTI，也不用于停机（Stop）模式唤醒。 • 1：唤醒定时器输出到 EXTI，作为 EXTI 中断源将系统从停机（Stop）模式唤醒。 WUTO[3]： • 0：唤醒定时器不输出到芯片内部电路作为周期性触发 ADC 信号（用作外设周期性预唤醒）。 • 1：唤醒定时器将输出到芯片内部电路作为周期性触发 ADC 使用（用作外设周期性预唤醒）。 *注意：RTC_Alarm 和 Wakeup timer 都挂在 EXTI 线 17 上，是或的关系。*
位 7:3	Res：保留 必须保持复位值。
位 2:0	WUCKSEL[2:0]：选择唤醒定时器的时钟（The clock of wakeup timer selection） • 000：选择 RTCCLK/16 时钟 • 001：选择 RTCCLK/8 时钟 • 010：选择 RTCCLK/4 时钟 • 011：选择 RTCCLK/2 时钟 • 10x：选择 TR_CLK（通常为 1 Hz）时钟，选择 RTC 输出的秒脉冲作为唤醒定时器的时钟。 • 11x：选择 TR_CLK（通常为 1 Hz）时钟，唤醒定时器的计数器作为 17 位计数器。 选择 RTC 输出的秒脉冲作为唤醒定时器的时钟，同时为 16 位唤醒定时器扩展 1 位高位，并且把 17 位唤醒定时器最高位置为 1。

7.3.7 控制寄存器（BKP_LSE_CTL）

偏移地址：0x3C

复位值：0x0000 82C2

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AGC_EN	Res					SEL_AGC		AUTO_IOP	NFBYP	Res	IOP_MON		Res	IOP	
rw						rw		rw	rw		r			rw	

位 31:16	Res：保留 必须保持复位值。
位 15	AGC_EN：开关 LSE AGC 自动增益电路（LSE AGC automatic gain circuit enable） • 0：关闭 LSE AGC 自动增益电路 • 1：开启并使用 LSE AGC 自动增益电路

位 14:10	Res：保留 必须保持复位值。
位 9:8	SEL_AGC：选择 AGC 电路的增益能力（The gain capacity of AGC automatic gain circuit） • 00：最低档位增益 • … • 11：最高档位增益
位 7	AUTO_IOP：LSE 驱动档位设置（LSE driven capacity setting ） • 0：IOP[1:0]的 LSE 驱动档位设置值立即生效 • 1：LSE 驱动档位将由 2’b11 逐渐降为 IOP[1:0]的设置值
位 6	NFBYP：选择 LSE 时钟信号是否经过片内滤噪器（LSE signal handled by the on-chip noise filter） • 0：LSE 时钟信号经过片内滤噪器后被使用 • 1：LSE 时钟信号不经过片内滤噪器
位 5	Res：保留 必须保持复位值。
位 4:3	IOP_MON：IOP 监测器（IOP detector） 当 AUTO_IOP 位设置为’1’时，IOP[1:0]的设置值不会立即生效；LSE 的驱动档位会由 2’b11 逐渐降为 IOP[1:0]的设置值。在此期间，可通过读取 IOP_MON[1:0]获得当前 LSE 的驱动档位值。
位 2	Res：保留 必须保持复位值。
位 1:0	IOP：用于设置 LSE 晶振电路驱动能力（LSE driven capacity selection） • 00：最低档位驱动能力 • … • 11：最高档位驱动能力

7.3.8 BKP_RAM 寄存器 x（BKP_RAMx）（x=11..42）

偏移地址：0x40+0x04*(x-11)

复位值：0x0000 0000

BKP_RAM 寄存器支持以字节、半字或字为单位进行读写访问。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D[15:0]															
rw															

位 15:0	D[15:0]：备份数据（Backup data） 可写入用户数据至该位域。 BKP_RAMx 可用作备份数据寄存器 BKP_DR11 ~ BKP_DR42。 *注意：BKP_DRx 寄存器不会被系统复位、电源复位或从待机模式唤醒所复位。它们可以由备份域复位或 TAMPER 引脚事件（如果 TAMPER 引脚的侵入检测功能被开启）复位。*

8 复位和时钟控制（RCC）

8.1 复位

器件支持三种复位方式：系统复位、电源复位和备份域复位。

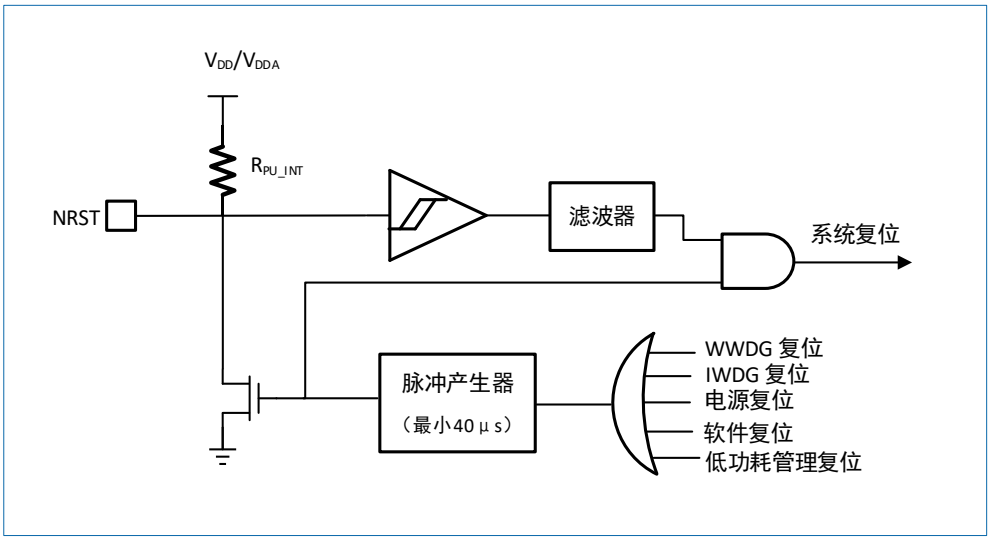


图 8-1 复位电路

8.1.1 系统复位

除了时钟控制/状态寄存器 RCC_CSR 中的复位标志位和备份域中的寄存器以外，系统复位将复位所有寄存器至它们的复位值（参见图 6-1）。可通过查看 RCC_CSR 控制/状态寄存器中的复位状态标志位识别复位事件来源。

当发生以下任一事件时，产生一个系统复位：

- NRST 引脚上的低电平（外部复位）
- 窗口看门狗计数终止（WWDG 复位）
- 独立看门狗计数终止（IWDG 复位）
- 软件复位（SW 复位）
- 低功耗管理复位

系统复位信号会输出至 NRST 引脚。脉冲发生器保证每个复位源（外部和内部）的复位脉冲持续时间至少为 40 µs。若发生外部复位，在 NRST 引脚被拉低时会产生复位脉冲。

8.1.1.1 软件复位

通过将 Cortex®-M3 中断应用和复位控制寄存器中的 SYSRESETREQ 位置’1’，可产生软件复位将系统强制复位。请参考《Cortex®-M3 技术参考手册》获得进一步信息。

8.1.1.2 低功耗管理复位

在以下两种情况下可产生低功耗管理复位：

- 在进入待机模式时产生低功耗管理复位

  通过将用户选项字节中的 nRST_STDBY 位置为’0’，将使能该复位。此时，即使执行了进入待机模式的操作，系统将被复位而不是进入待机模式。
- 在进入停机模式时产生低功耗管理复位

通过将用户选项字节中的 nRST_STOP 位置为'0'，将使能该复位。此时，即使执行了进入停机模式的操作，系统将被复位而不是进入停机模式。

8.1.2 电源复位

当以下事件中之一发生时，产生电源复位：

- 上电/掉电复位（POR/PDR）
- 从待机模式中返回

电源复位将复位除了备份域外的所有寄存器（参见图 6-1）。

电源复位源将作用于 NRST 引脚并在复位过程中保持低电平。复位服务程序入口矢量在存储器映射的地址被固定在 0x0000 0004，详情可参考表 9-1。

8.1.3 备份域复位

备份域拥有两个专用的复位，它们仅作用于备份域（见图 6-1）。

以下任一事件发生，会产生备份域复位。

- 软件复位：通过设置备份域控制寄存器（RCC_BDCR）中的 BDRST 位产生。
- 在 VDD 和 VBAT 两者掉电后，V_{DD} 和（或）V_{BAT} 再上电时将触发备份域复位。

8.2 时钟

以下时钟源均可用于驱动系统时钟（SYSCLK）：

- HSI 振荡器时钟：56MHz/28MHz/8MHz
- HSE 振荡器时钟：4~32MHz
- PLL 输出时钟：120MHz（最大值）
- 低速内部时钟（LSI）：40kHz

  可用于驱动独立看门狗和 RTC。可选择是否由该时钟驱动 RTC。RTC 用在停机/待机模式下自动唤醒系统。
- 低速外部时钟（LSE）：32.768kHz

  可用于驱动 RTC（RTCCLK）。可选择是否由它驱动 RTC。
- GPIO 输入时钟

在未被使用时，以上任一时钟源可被独立地打开或关闭，以优化系统功耗。

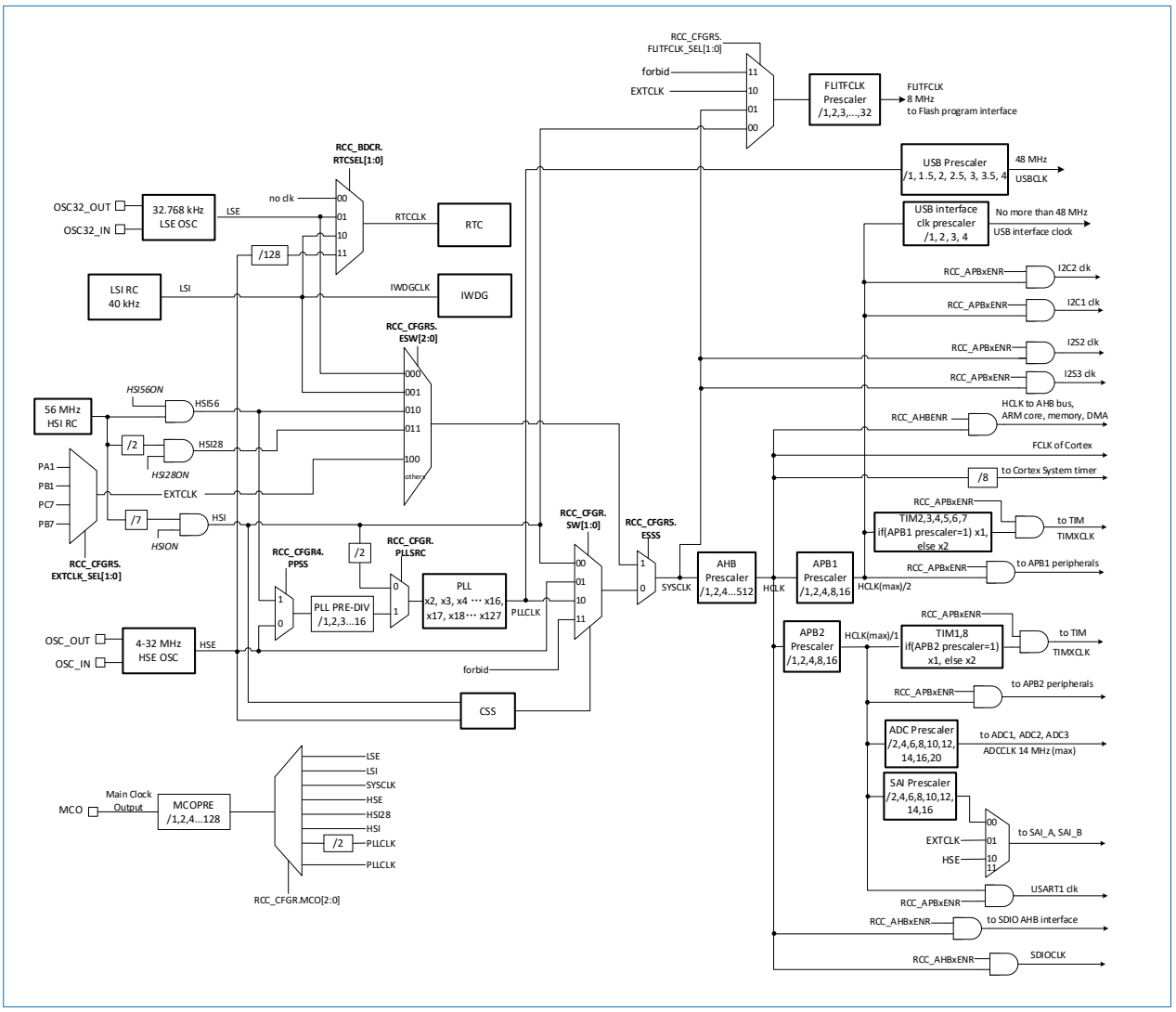


图 8-2 时钟树

上图说明：

- 该系列 MCU 支持 32.768 kHz (LSE)或 40k Hz (LSI)作为系统时钟。
- 若需了解内部和外部时钟源特性，请参见《HK32F39AxCxDxE 数据手册》的“电气特性”章节。

通过多个预分频器配置 AHB、高速 APB（APB2）和低速 APB（APB1）域的频率。AHB 和 APB2 域的最大频率是 120 MHz。APB1 域的最大允许频率是 60 MHz。

RCC 将 AHB 时钟（HCLK）8 分频后作为 Cortex 系统定时器（SysTick）的外部时钟。通过配置 SysTick 控制与状态寄存器，可选择 HCLK/8 时钟或 Cortex（HCLK）时钟作为 SysTick 时钟。

ADC 时钟由高速 APB2 时钟经 2、4、6、8、10、12、14、16 或 20 分频后获得。

USB 时钟可以是 PLL 输出的 1、1.5、2、2.5、3、3.5 或 4 分频，必须保证分频后的 USB 时钟频率为 48 MHz。

FLITFCLK 时钟源可从 HSI、SYSCLK 或 GPIO 输入时钟（EXTCLK）中选择，并支持 1~32 整数分频。在擦除或编程 Flash 时，必须确保分频后的 FLITFCLK 时钟频率为 8 MHz。

定时器时钟频率由硬件自动确定，包括 2 种情况：

- 若 APB 预分频系数为 1，则定时器频率设置为定时器所连接的 APB 域的频率。
- 否则，定时器频率被设置为定时器所连接的 APB 域频率的 2 倍。

FCLK 是 Cortex®-M3 的自由运行时钟。详情见 ARM 的《Cortex®-M3 技术参考手册》。

8.2.1 HSI 时钟

HSI 时钟信号由内部 56 MHz 的 RC 振荡器（称为 56MHz HSI RC）产生。56MHz HSI RC 振荡器可产生 3 个不同频率的时钟（见图 8-2）：

- 56 MHz 的 HSI56 时钟：由 56MHz HSI RC 直接输出得到。
- 28 MHz 的 HSI28 时钟：由 56MHz HSI RC 经过 2 分频后得到。
- 8 MHz 的 HSI 时钟：由 56MHz HSI RC 经过 7 分频后得到。

HSI 时钟默认作为 SYSCLK 的时钟源。

56MHz HSI RC 振荡器的优点在于无需使用外部元器件，也能提供时钟源。与 HSE 晶体振荡器相比，它的启动时间更短。然而，即使在校准之后它的时钟频率精度仍较晶体振荡器/陶瓷谐振器更差。

校准

制造工艺决定了不同芯片的 RC 振荡器频率会不同，这就是为什么每个芯片的 HSI 时钟频率在出厂前已经被校准到 1%（25°C）的原因。系统复位时，工厂校准值被加载到时钟控制寄存器 RCC_CR 中的 HSICAL[7:0]位。

如果用户的应用基于不同的电压或环境温度，这将会影响 RC 振荡器的精度。可以通过寄存器 RCC_CR 中的 HSITRIM[4:0]位来调整用户应用的 HSI 频率。寄存器 RCC_CR 中的 HSIRDY 位用来指示 56MHz HSI RC 振荡器是否稳定。在启动过程中，56MHz HSI RC 输出时钟直到 HSIRDY 位被硬件置'1'才被释放。通过配置寄存器 RCC_CR 中的 HSION 位可打开或关闭 56MHz HSI RC。

HSE 晶体振荡器失效，HSI 时钟会被作为备用时钟源。参考章节："8.2.7 时钟安全系统（CSS）"。

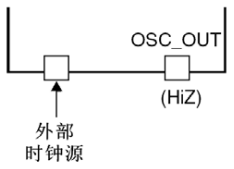
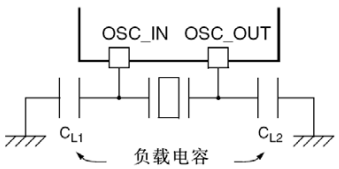
8.2.2 HSE 时钟

高速外部时钟信号（HSE）由以下两种时钟源产生：

- HSE 外部晶体/陶瓷谐振器
- HSE 用户外部时钟

晶体/陶瓷谐振器和负载电容器必须尽可能地靠近振荡器引脚，以使时钟输出的失真和启动稳定时间减到最小。负载电容值必须根据所选择的振荡器来调整。

表 8-1 HSE/LSE 时钟源

时钟源	硬件配置
外部时钟	
晶体/陶瓷谐振器	

器件的 HSE 具有以下特性：

- HSE 时钟信号可经过片内滤噪器后供片内电路使用，以减少环境噪声对 HSE 时钟的影响。
- HSE 晶振振荡稳定的等待时间可配置。

- HSE 晶振引脚驱动能力可配置。

8.2.2.1 外部晶体/陶瓷谐振器（HSE 晶体）

支持 4~32MHz HSE 外部振荡器，可为系统提供更为精确的主时钟。相关的硬件配置见表 8-1。详情可参考《HK32F39AxCxDxE 数据手册》的“电气参数”章节。

时钟控制寄存器（RCC_CR）中的 HSERDY 位可指示高速外部振荡器是否稳定。在启动时，直到这一位被硬件置’1’，该时钟才被释放。如果在时钟中断寄存器 RCC_CIR 中使能中断，将会产生相应中断。

通过设置时钟控制寄存器（RCC_CR）中的 HSEON 位可打开和关闭 HSE 晶振。

在进入停机（Stop）模式后，默认关闭 HSE 以节省功耗。为了从停机（Stop）模式唤醒后，系统能快速切换 SYSCLK 到 HSE 时钟，可通过配置 RCC_HSECTL 寄存器的 HSE_STOP_KEEP 位，使 HSE 在停机（Stop）模式下仍保持进入前的开关状态。

8.2.2.2 外部时钟源（HSE 旁路）

在这个模式里，必须提供外部时钟。它支持频率最高可达 64 MHz。用户可通过设置时钟控制寄存器（RCC_CR）中的 HSEBYP 和 HSEON 位来选择这一模式。外部时钟信号（50%占空比的方波、正弦波或三角波）必须连到 OSC_IN 引脚，见表 8-1。

8.2.3 PLL

内部 PLL 可以用来倍频 56MHz HSI RC 的输出时钟或 HSE 晶体输出时钟。参考图 8-2 和章节：“时钟控制寄存器（RCC_CR）”。

在使能 PLL 之前必须先完成 PLL 配置（选择 HSI/2、HSI56 或 HSE 作为 PLL 输入时钟，PLL 前置分频系数以及 PLL 倍频系数），见图 8-2。PLL 前置分频系数可以是 1~16 的任意整数。PLL 倍频系数可以是 2~127 的任意整数。PLL 的设置（选择输入时钟源、配置前置分频数和倍频因子）必须在其被激活前完成。

注意：一旦 PLL 被激活，这些参数就不能被改动。

如果 PLL 中断在时钟中断寄存器（RCC_CIR）中被使能，当 PLL 准备就绪时，可产生中断。如果需要在应用中使用 USB 接口，PLL 必须被设置为输出 48、72、96 或 120 MHZ 时钟，用于分频后提供 48 MHz 的 USBCLK 时钟。PLL 支持的输入/输出频率范围参考芯片数据手册。

8.2.4 LSE 时钟

LSE 晶体是一个 32.768 kHz 的低速外部晶体或陶瓷谐振器。它为 RTC 或者其他定时功能提供一个低功耗、高精度的时钟源。

器件的 LSE 具有以下的特性：

- 可以选择经过片内滤噪器后再提供给片内电路使用，以减少环境噪声对 LSE 时钟的干扰。
- LSE 晶振引脚驱动能力可配置。
- LSE 晶振电路带有 AGC 自动增益功能；通过使能 AGC 电路可显著降低 LSE 晶振电流功耗。
- 通过配置备份域控制寄存器（RCC_BDCR）中的 LSEON 位，可启动或关闭 LSE 晶体。
- 通过配置寄存器 BKP_LSE_CTL 可设置 LSE 的驱动能力和 AGC 功能。
- 备份域控制寄存器（RCC_BDCR）中的 LSERDY 位指示 LSE 晶体振荡是否已稳定。在启动阶段，直到 LSERDY 位被硬件置’1’，LSE 晶体才被释放。如果在时钟中断寄存器（RCC_CIR）中使能，则可产生中断。

外部时钟源（LSE 旁路）

在该模式下，必须提供外部时钟源，其最高频率为 2 MHz。通过配置备份域控制寄存器（RCC_BDCR）中的 LSEON 位和 LSEBYP 位选择该模式。具有 50%占空比的外部时钟信号（方波、正弦波或三角波）必

须连到 OSC32_IN 引脚，见表 8-1 HSE/LSE 时钟源。

8.2.5 LSI 时钟

LSI RC 40kHz 可作为低功耗时钟源在停机和待机模式下保持运行，为 IWDG 和自动唤醒单元（AWU）提供时钟。LSI 时钟频率大约 40 kHz（在 30 kHz 和 60 kHz 之间）。详情请参考数据手册中有关电气特性部分。

通过配置 RCC_LSICTL 寄存器的 LSI_FREQ 位设置 LSI 的时钟频率。

通过配置控制/状态寄存器（RCC_CSR）里的 LSION 位来启动或关闭 LSI RC。

控制/状态寄存器（RCC_CSR）里的 LSIRDY 位指示 LSI RC 是否已稳定。在启动阶段，直到 LSIRDY 位被硬件设置为'1'，LSI 时钟才被释放。如果在时钟中断寄存器（RCC_CIR）里使能中断，将产生 LSI 中断请求。

8.2.6 系统时钟（SYSCLK）选择

系统复位后，HSI 时钟（8 MHz）被选为系统时钟。当某个时钟源被直接或通过 PLL 间接作为系统时钟时，它将不能被停止。

只有当目标时钟源准备就绪了（经过启动稳定阶段的延迟或 PLL 稳定），系统时钟才从原来的时钟源切换到该目标时钟。在目标时钟源未准备就绪时，不会切换系统时钟。

在时钟控制寄存器（RCC_CR）里的状态位指示哪个时钟已经准备就绪，当前哪个时钟用作系统时钟。

8.2.7 时钟安全系统（CSS）

通过软件激活时钟安全系统。当 CSS 启用后，在 HSE 振荡器启动稳定后会使能时钟检测器；当 HSE 时钟关闭后会禁用该时钟检测器。

如果 HSE 时钟发生故障，HSE 振荡器被自动关闭，时钟故障事件将被送到高级定时器（TIM1）的刹车输入端，并产生时钟安全中断 CSSF，允许软件完成营救操作。此 CSSF 中断连接到 Cortex®-M3 的非屏蔽中断（NMI）。

CSS 判断 HSE 故障的频率阈值可调，可以通过寄存器 RCC_HSECTL.CSS_THRESHOLD 配置。

注意：一旦使能 CSS 且 HSE 时钟出现故障，则发生 CSS 中断，并且自动产生 NMI。NMI 将被不断执行，直到 CSS 中断挂起位被清除。因此，在 NMI 的处理程序中必须通过设置时钟中断寄存器（RCC_CIR）里的 CSSC 位来清除 CSS 中断。

如果 HSE 振荡器被直接或间接地作为系统时钟（间接指的是：它作为 PLL 的输入时钟，PLL 的输出时钟作为系统时钟）。若检测到 HSE 时钟故障将导致系统时钟自动切换到 HSI 振荡器，同时外部 HSE 振荡器被禁用。在 HSE 时钟故障时，如果 HSE 时钟（分频或未分频）间接地作为系统时钟，PLL 也将被禁用。

8.2.8 RTC 时钟

通过设置备份域控制寄存器（RCC_BDCR）里的 RTCSEL[1:0]位，从 HSE/128、LSE 或 LSI 时钟中选择 RTCCLK 时钟源。除非备份域复位，否则不能被改变此选择。LSE 时钟在备份域里，但 HSE 和 LSI 时钟不在此域。因此：

- 如果选择 LSE 为 RTC 时钟：

  只要 V_{BAT} 维持供电，尽管 V_{DD} 供电被切断，RTC 仍继续工作。

- 如果选择 LSI 为自动唤醒单元（AWU）时钟：

如果 V_{DD} 供电被切断，不能保证 AWU 状态。

- 如果 HSE 时钟 128 分频后作为 RTC 时钟：
  - 如果 V_{DD} 供电被切断或内部电压调节器被关闭（core 电压域供电被切断），则无法保证 RTC 状态。
  - 必须设置电源控制寄存器（PWR_CR）的 DPB 位（取消备份域的写保护）为'1'。

8.2.9 看门狗时钟

如果独立看门狗已经由硬件选项或软件启动，LSI 振荡器将被强制在打开状态，并且不能被禁用。在 LSI 振荡器稳定后，该时钟供给 IWDG。

在使能 IWDG 后，可通过 Flash 选项字 LSI_LP_CTL 控制 LSI 在停机或待机模式下的状态。如果将 LSI_LP_CTL 的值配置为 0x369C F0F0 时，在 MCU 进入停机或待机模式后，LSI 可以根据 LSION 位的设置开关 LSI；在 MCU 唤醒后，LSI 恢复成进入模式之前的状态。

8.2.10 时钟输出

MCU 允许输出时钟信号到外部 MCO 引脚。相应的 GPIO 端口寄存器必须被配置为相应功能。以下八个时钟信号可被选作 MCO 时钟：

- LSE
- LSI
- SYSCLK
- HSE
- HSI28
- HSI
- PLLCLK
- PLLCLK/2

MCO 时钟的选择由时钟配置寄存器（RCC_CFGR）中的 MCO[2:0]位控制。

8.3 RCC 寄存器

基地址：0x4002 1000

空间大小：0x400

8.3.1 时钟控制寄存器（RCC_CR）

偏移地址：0x00

复位值：0x0000 XX83

说明：X 代表不定值；该寄存器支持无等待状态访问，支持字、半字和字节访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res						PLLRDY	PLLON	Res				CSSON	HSEBYP	HSERDY	HSEON
						r	rw					rw	rw	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSICAL[7:0]								HSITRIM[4:0]					Res	HSIRDY	HSION
r								rw						r	rw

位 30:26	Res：保留 必须保持复位值。
位 25	PLLRDY：PLL 时钟就绪标志（PLL clock ready flag）

	PLL 锁定后由硬件置'1'。 • 0：PLL 未锁定 • 1：PLL 锁定
位 24	PLLON：PLL 使能位（PLL enable） 此位由软件置位和清零，以使能 PLL。 当进入停机或待机模式时，该位由硬件清零。如果 PLL 时钟用作系统时钟或被选择将要作为系统时钟，该位不能复位。 • 0：PLL 关闭 • 1：PLL 开启
位 23:20	Res：保留 必须保持复位值。
位 19	CSSON：时钟安全系统使能（Clock security system enable） 由软件置'1'或清零。 设置该位来使能时钟监测器。 • 0：时钟监测器关闭 • 1：如果外部 HSE（4-32MHz）振荡器就绪，时钟监测器开启。
位 18	HSEBYP：外部高速时钟旁路（HSE crystal oscillator bypass） 在调试模式下由软件置'1'或置'0'。 设置该位来旁路外部晶体振荡器。只有在 HSE 振荡器关闭的情况下，才能写入该位。 • 0：HSE 振荡器未被旁路 • 1：HSE 晶体振荡器被旁路
位 17	HSERDY：外部高速时钟就绪标志（HSE clock ready flag） 由硬件置'1'来指示 HSE 振荡器已经稳定。在 HSEON 位清零后，该位需要 6 个外部 HSE 振荡器周期清零。 • 0：HSE 振荡器未就绪 • 1：HSE 振荡器就绪
位 16	HSEON：外部高速时钟使能（HSE clock enable） 由软件置'1'或清零。 当进入待机和停止模式时，该位由硬件清零，关闭 HSE 振荡器。当外部 HSE 振荡器被用作或被选择将要作为系统时钟时，该位不能被清零。 • 0：HSE 振荡器关闭 • 1：HSE 振荡器开启
位 15:8	HSICAL[7:0]：内部高速时钟校准（HSI clock calibration） 在系统启动时，这些位被自动初始化。
位 7:3	HSITRIM[4:0]：内部高速时钟调整（HSI clock trimming） 软件可以写入不同值来调整内部高速时钟频率，它们被叠加在 HSICAL[7:0]数值上。这些位在 HSICAL[7:0]的基础上，让用户可以输入一个调整数值，根据电压和温度的变化调整内部 HSI RC 振荡器的频率。 HSITRIM 的调整频率的步进约 1.3‰。
位 2	Res：保留 必须保持复位值。
位 1	HSIRDY：内部高速时钟就绪标志（HSI clock ready flag）

	由硬件置'1'来指示内部 8MHz 振荡器已经稳定。在 HSION 位清零后，该位需要 6 个内部 8MHz 振荡器周期清零。 • 0：内部 8MHz 振荡器未就绪 • 1：内部 8MHz 振荡器就绪
位 0	HSION：内部高速时钟使能（HSI clock enable） 由软件置'1'或清零。 当从待机和停止模式返回或用作系统时钟的 HSE 振荡器发生故障时，该位由硬件置'1'来启动内部 RC 振荡器并分频产生 8MHz 时钟。当内部 8MHz 振荡器被直接或间接地用作或被选择将要用作系统时钟时，该位不能被清零。 • 0：内部 8MHz 时钟关闭 • 1：内部 8MHz 时钟开启

8.3.2 时钟配置寄存器（RCC_CFGR）

偏移地址：0x04

复位值：0x0000 0000

访问：0 到 2 个等待周期，该寄存器支持字、半字和字节访问；只有当访问发生在时钟切换时，才会插入 1 或 2 个等待周期。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res					MCO[2:0]			Res	USBPRE	PLLMULL[3:0]				PLLXTPRE	PLLSRC
					rw				rw	rw				rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCPRE[1:0]		PPRE2[2:0]			PPRE1[2:0]			HPRE[3:0]				SWS[1:0]		SW[1:0]	
rw		rw			rw			rw				r		rw	

位 31:27	Res：保留 必须保持复位值。
位 26:24	MCO[2:0]：MCU 时钟输出（Microcontroller clock output） 由软件置'1'或清零。 该位域和 RCC_CFGR5.MCO[3]一起组成 4 位的 MCO[3:0]。 • 0100：系统时钟（SYSCLK）输出 • 0101：HSI 时钟（8 MHz）输出 • 0110：HSE 时钟输出 • 0111：PLL 或者 PLL/2 时钟输出，PLL 是否 2 分频由 RCC_CFGR5.PLLNODIV 决定 • 1100：LSI 时钟输出 • 1101：LSE 时钟输出 • 1110：HSI28 时钟输出 • 其它：保留 *注意：* • *在启动和切换 MCO 时钟源时，该时钟输出可能会被截断。* • *在系统时钟输出至 MCO 引脚时，请保证输出时钟频率不超过 50 MHz（I/O 口最高频率）。*
位 23	Res：保留 必须保持复位值。
位 22	USBPRE：USB 预分频（USB prescaler） 该位域和 RCC_CFGR3.USBSW[2:0]一起配置 USB 时钟相对于 PLL 时钟的分频数。

	本寄存器位必须在 RCC_APB1ENR 寄存器中使能 USB 时钟之前配置。一旦 USB 时钟被使能，该位不能被清零。
	• RCC_CFGR3.USBSW[2:0]=000 时，由软件置'1'或清零该寄存器位来产生 48 MHz 的 USB 时钟。
	  ○ 0：PLL 时钟 1.5 倍分频作为 USB 时钟（PLL 输出为 72 MHz）
	  ○ 1：PLL 时钟直接作为 USB 时钟（PLL 输出为 48 MHz）
	• RCC_CFGR3.USBSW[2:0]≠0 时，USB 的分频数由 RCC_CFGR3.USBSW[2:0]决定。

位 21:18	PLLMULL[3:0]：PLL 倍频系数低位（PLL multiplication factor）
	PLL 倍频系数由 RCC_CFGR4.PLLMULH[2:0]和 PLLMULL[3:0]共同组成。
	• 当 RCC_CFGR4.PLLMULH=000 时：
	  ○ 0000：PLL 2 倍频输出
	  ○ 0001：PLL 3 倍频输出
	  ○ 0010：PLL 4 倍频输出
	  ○ 0011：PLL 5 倍频输出
	  ○ 0100：PLL 6 倍频输出
	  ○ 0101：PLL 7 倍频输出
	  ○ 0110：PLL 8 倍频输出
	  ○ 0111：PLL 9 倍频输出
	  ○ 1000：PLL 10 倍频输出
	  ○ 1001：PLL 11 倍频输出
	  ○ 1010：PLL 12 倍频输出
	  ○ 1011：PLL 13 倍频输出
	  ○ 1100：PLL 14 倍频输出
	  ○ 1101：PLL 15 倍频输出
	  ○ 1110 和 1111：PLL 16 倍频输出
	• 当 RCC_CFGR4.PLLMULH[2:0]≠0 时，PLL 倍频数由 RCC_CFGR4.PLLMULH 决定。

位 17	PLLXTPRE：HSE 分频器作为 PLL 输入（HSE divider for PLL input clock） 此位和 RCC_CFGR2.PREDIV[0]的作用相同，参见"8.3.11 时钟配置寄存器 2（RCC_CFGR2）"中 PREDIV 位。
位 16	PLLSRC：PLL 输入时钟源（PLL input clock source） 由软件置'1'或清零来选择 PLL 输入时钟源。仅能在 PLL 关闭时，才能写入此位。 • 0：HSI/2 时钟作为 PLL 输入时钟 • 1：PLL 前置分频后的时钟作为 PLL 输入时钟
位 15:14	ADCPRE[1:0]：ADC 预分频（ADC prescaler） 该位域和 RCC_CFGR3.ADC3SW、RCC_CFGR3.ADC2SW、RCC_CFGR3.ADC1SW 配合使用来配置 ADC 时钟频率。该位由软件置'1'或清零。 • 00：PCLK2 2 分频后作为 ADC 时钟 • 01：PCLK2 4 分频后作为 ADC 时钟 • 10：PCLK2 6 分频后作为 ADC 时钟 • 11：PCLK2 8 分频后作为 ADC 时钟
位 13:11	PPRE2[2:0]：高速 APB2 预分频（APB2 prescaler） 该位由软件置'1'或清零，以控制高速 APB2 时钟（PCLK2）的预分频系数。 • 0xx：HCLK 不分频 • 100：HCLK 2 分频 • 101：HCLK 4 分频 • 110：HCLK 8 分频 • 111：HCLK 16 分频

位 10:8	PPRE1[2:0]：低速 APB1 预分频（APB1 prescaler） 该位由软件置'1'或清零，以控制低速 APB1 时钟（PCLK1）的预分频系数。 *警告：软件必须保证 APB1 时钟频率不超过 HCLK(max)/2，即 60MHz。HCLK(max)表示 HCLK 的最高频率，即 120 MHz。* • 0xx：HCLK 不分频 • 100：HCLK 2 分频 • 101：HCLK 4 分频 • 110：HCLK 8 分频 • 111：HCLK 16 分频
位 7:4	HPRE[3:0]：AHB 预分频（HCLK prescaler factor） 该位由软件置'1'或清零，以控制 AHB 时钟的预分频系数。 • 0xxx：SYSCLK 不分频 • 1000：SYSCLK 2 分频 • 1001：SYSCLK 4 分频 • 1010：SYSCLK 8 分频 • 1011：SYSCLK 16 分频 • 1100：SYSCLK 64 分频 • 1101：SYSCLK 128 分频 • 1110：SYSCLK 256 分频 • 1111：SYSCLK 512 分频
位 3:2	SWS[1:0]：系统时钟源状态（System clock source status） 该位由硬件置'1'或清零，用于指示哪个时钟源被作为系统时钟。 • 00：HSI 为系统时钟 • 01：HSE 为系统时钟 • 10：PLL 输出为系统时钟 • 11：保留
位 1:0	SW[1:0]：系统时钟切换（System clock switch） 该位由软件置'1'或清零，以选择系统时钟源。 当在从停机或待机模式中返回或直接/间接作为系统时钟的 HSE 出现故障时，由硬件强制选择 HSI 作为系统时钟（如果 CSS 已经启动）。 • 00：HSI 作为系统时钟 • 01：HSE 作为系统时钟 • 10：PLL 输出作为系统时钟 • 11：保留

8.3.3 时钟中断寄存器（RCC_CIR）

偏移地址：0x08

复位值：0x0000 0000

访问：无等待周期，支持字、半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res								CSSC	Res		PLLRDYC	HSERDYC	HSIRDYC	LSERDYC	LSIRDYC
								w			w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res			PLLRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE	CSSF	Res		PLLRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
			rw	rw	rw	rw	rw	r			r	r	r	r	r

位 31:24	Res：保留 必须保持复位值。
位 23	CSSC：清除时钟安全系统中断（Clock security system interrupt flag clear） 由软件置’1’来清除 CSSF 安全系统中断标志位 CSSF。 • 0：无作用 • 1：清除 CSSF 安全系统中断标志位
位 22:21	Res：保留 必须保持复位值。
位 20	PLLRDYC：清除 PLL 就绪中断（PLL ready interrupt clear） 由软件置’1’来清除 PLL 就绪中断标志位 PLLRDYF。 • 0：无作用 • 1：清除 PLL 就绪中断标志位 PLLRDYF
位 19	HSERDYC：清除 HSE 就绪中断（HSE ready interrupt clear） 由软件置’1’来清除 HSE 就绪中断标志位 HSERDYF。 • 0：无作用 • 1：清除 HSE 就绪中断标志位 HSERDYF
位 18	HSIRDYC：清除 HSI（8 MHz）就绪中断（HSI ready interrupt clear） 由软件置’1’来清除 HSI 就绪中断标志位 HSIRDYF。 • 0：无作用 • 1：清除 HSI 就绪中断标志位 HSIRDYF
位 17	LSERDYC：清除 LSE 就绪中断（LSE ready interrupt clear） 由软件置’1’来清除 LSE 就绪中断标志位 LSERDYF。 • 0：无作用 • 1：清除 LSE 就绪中断标志位 LSERDYF
位 16	LSIRDYC：清除 LSI 就绪中断（LSI ready interrupt clear） 由软件置’1’来清除 LSI 就绪中断标志位 LSIRDYF。 • 0：无作用 • 1：清除 LSI 就绪中断标志位 LSIRDYF
位 15:13	Res：保留 必须保持复位值。
位 12	PLLRDYIE：PLL 就绪中断使能（PLL ready interrupt enable） 由软件置’1’或清零来使能或关闭 PLL 就绪中断。 • 0：PLL 就绪中断关闭 • 1：PLL 就绪中断使能
位 11	HSERDYIE：HSE 就绪中断使能（HSE ready interrupt enable）

	由软件置’1’或清零来使能或关闭外部 HSE 振荡器就绪中断。 • 0：HSE 就绪中断关闭 • 1：HSE 就绪中断使能
位 10	HSIRDYIE：HSI（8 MHz）就绪中断使能（HSI ready interrupt enable） 由软件置’1’或清零来使能或关闭内部 8 MHz RC 振荡器就绪中断。 • 0：HSI 就绪中断关闭 • 1：HSI 就绪中断使能
位 9	LSERDYIE：LSE 就绪中断使能（LSE ready interrupt enable） 由软件置’1’或清零来使能或关闭外部 32 kHz RC 振荡器就绪中断。 • 0：LSE 就绪中断关闭 • 1：LSE 就绪中断使能
位 8	LSIRDYIE：LSI 就绪中断使能（LSI ready interrupt enable） 由软件置’1’或清零来使能或关闭内部 40 kHz RC 振荡器就绪中断。 • 0：LSI 就绪中断关闭 • 1：LSI 就绪中断使能
位 7	CSSF：时钟安全系统中断标志（Clock security system interrupt flag） 在外部 4-32M 振荡器时钟出现故障时，由硬件置’1’。 由软件通过将 CSSC 位置’1’来清除。 • 0：无 HSE 时钟失效产生的安全系统中断 • 1：HSE 时钟失效触发时钟安全系统中断
位 6:5	Res：保留 必须保持复位值。
位 4	PLLRDYF：PLL 就绪中断标志（PLL ready interrupt flag） 在 PLL 就绪且 PLLRDYIE 位被置’1’时，由硬件置’1’。 由软件通过将 PLLRDYC 位置’1’来清除。 • 0：没有由 PLL 锁定引起的时钟就绪中断 • 1：由 PLL 锁定引起的时钟就绪中断
位 3	HSERDYF：HSE 就绪中断标志（HSE ready interrupt flag） 在 HSE 高速时钟就绪且 HSERDYIE 位被置’1’时，由硬件置’1’。 由软件通过将 HSERDYC 位置’1’来清除。 • 0：无外部 HSE 振荡器产生的时钟就绪中断 • 1：外部 HSE 振荡器时钟就绪中断
位 2	HSIRDYF：HSI（8 MHz）就绪中断标志（HSI ready interrupt flag） 在 HSI 时钟就绪且 HSIRDYIE 位被置’1’时，由硬件置’1’。 由软件通过将 HSIRDYC 位置’1’来清除。 • 0：无内部 8 MHz RC 振荡器产生的时钟就绪中断 • 1：内部 8 MHz RC 振荡器时钟就绪中断。
位 1	LSERDYF：LSE 就绪中断标志（LSE ready interrupt flag） 在 LSE 时钟就绪且 LSERDYIE 位被置’1’时，由硬件置’1’。 由软件通过将 LSERDYC 位置’1’来清除。

	• 0：无外部 32 kHz 振荡器产生的时钟就绪中断 • 1：外部 32 kHz 振荡器时钟就绪中断
位 0	LSIRDYF：LSI 就绪中断标志（LSI ready interrupt flag） 在 LSI 时钟就绪且 LSIRDYIE 位被置’1’时，由硬件置’1’。 由软件通过将 LSIRDYC 位置’1’来清除。 • 0：无内部 40 kHz RC 振荡器产生的时钟就绪中断 • 1：内部 40 kHz RC 振荡器时钟就绪中断

8.3.4 APB2 外设复位寄存器（RCC_APB2RSTR）

偏移地址：0x0C

复位值：0x0000 0000

访问：无等待周期，支持字、半字和字节访问

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC3 RST	USART 1RST	TIM8 RST	SPI1 RST	TIM1 RST	ADC2 RST	ADC1 RST	Res		IOPE RST	IOPD RST	IOPC RST	IOPB RST	IOPA RST	R es	AFIO RST
rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw		rw

位 31:16	Res：保留 必须保持复位值。
位 15	ADC3RST：ADC3 接口复位（Reset ADC3 interface） 由软件置’1’或清零。 • 0：无作用 • 1：复位 ADC3 接口
位 14	USART1RST：USART1 复位（Reset USART1） 由软件置’1’或清零。 • 0：无作用 • 1：复位 USART1
位 13	TIM8RST：TIM8 复位（Reset TIM8） 由软件置’1’或清零。 • 0：无作用 • 1：复位 TIM8
位 12	SPI1RST：SPI1 复位（Reset SPI1） 由软件置’1’或清零。 • 0：无作用 • 1：复位 SPI1
位 11	TIM1RST：TIM1 复位（Reset TIM1） 由软件置’1’或清零。 • 0：无作用

	• 1：复位 TIM1 定时器
位 10	ADC2RST：ADC2 接口复位（Reset ADC2 interface） 由软件置’1’或清零。 • 0：无作用 • 1：复位 ADC2 接口
位 9	ADC1RST：ADC1 接口复位（Reset ADC1 interface） 由软件置’1’或清零。 • 0：无作用 • 1：复位 ADC1 接口
位 8:7	Res：保留 必须保持复位值。
位 6	IOPERST：IO 端口 E 复位（Reset IO E） 由软件置’1’或清零。 • 0：无作用 • 1：复位 IO 端口 E
位 5	IOPDRST：IO 端口 D 复位（Reset IO D） 由软件置’1’或清零。 • 0：无作用 • 1：复位 IO 端口 D
位 4	IOPCRST：IO 端口 C 复位（Reset IO C） 由软件置’1’或清零。 • 0：无作用 • 1：复位 IO 端口 C
位 3	IOPBRST：IO 端口 B 复位（Reset IO B） 由软件置’1’或清零。 • 0：无作用 • 1：复位 IO 端口 B
位 2	IOPARST：IO 端口 A 复位（Reset IO A） 由软件置’1’或清零。 • 0：无作用 • 1：复位 IO 端口 A
位 1	Res：保留 必须保持复位值。
位 0	AFIORST：复用功能 I/O 复位（Reset AFIO） 由软件置’1’或清零。 • 0：无作用 • 1：复位复用功能

8.3.5 APB1 外设复位寄存器（RCC_APB1RSTR）

偏移地址：0x10

复位值：0x0000 0000

访问：无等待周期，支持字、半字和字节访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res		DACRST	PWRRST	BKPRST	CAN2RST	CAN1RST	Res	USBRST	I2C2RST	I2C1RST	USART5RST	USART4RST	USART3RST	USART2RST	Res
		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3RST	SPI2RST	Res		WWDGRST	Res					TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST
rw	rw			rw						rw	rw	rw	rw	rw	rw

位 31:30	Res：保留 必须保持复位值。
位 29	DACRST：DAC 接口复位（DAC interface reset） 由软件置’1’或清零。 • 0：无作用 • 1：复位 DAC 接口
位 28	PWRRST：电源接口复位（Power interface reset） 由软件置’1’或清零。 • 0：无作用 • 1：复位电源接口
位 27	BKPRST：备份接口复位（Backup interface reset） 由软件置’1’或清零。 • 0：无作用 • 1：复位备份接口
位 26	CAN2RST：CAN2 复位（CAN2 reset） 由软件置’1’或清零。 • 0：无作用 • 1：复位 CAN2
位 25	CAN1RST：CAN1 复位（CAN1 reset） 由软件置’1’或清零。 • 0：无作用 • 1：复位 CAN1
位 24	Res：保留 必须保持复位值。
位 23	USBRST：USB 复位（USB reset） 由软件置’1’或清零。 • 0：无作用 • 1：复位 USB

位 22	I2C2RST：I2C2 复位（I2C2 reset） 由软件置'1'或清零。 • 0：无作用 • 1：复位 I2C2
位 21	I2C1RST：I2C1 复位（I2C1 reset） 由软件置'1'或清零。 • 0：无作用 • 1：复位 I2C1
位 20	USART5RST：USART5 复位（USART5 reset） 由软件置'1'或清零。 • 0：无作用 • 1：复位 USART5
位 19	USART4RST：USART4 复位（USART4 reset） 由软件置'1'或清零。 • 0：无作用 • 1：复位 USART4
位 18	USART3RST：USART3 复位（USART3 reset） 由软件置'1'或清零。 • 0：无作用 • 1：复位 USART3
位 17	USART2RST：USART2 复位（USART2 reset） 由软件置'1'或清零。 • 0：无作用 • 1：复位 USART2
位 16	Res：保留 必须保持复位值。
位 15	SPI3RST：SPI3 复位（SPI3 reset） 由软件置'1'或清零。 • 0：无作用 • 1：复位 SPI3
位 14	SPI2RST：SPI2 复位（SPI2 reset） 由软件置'1'或清零。 • 0：无作用 • 1：复位 SPI2
位 13:12	Res：保留 必须保持复位值。
位 11	WWDGRST：WWDG 复位（Window watchdog reset） 由软件置'1'或清零。 • 0：无作用

	• 1：复位 WWDG
位 10:6	Res：保留 必须保持复位值。
位 5	TIM7RST：定时器 7 复位（Timer7 reset） 由软件置`1`或清零。 • 0：无作用 • 1：复位 TIM7 定时器
位 4	TIM6RST：定时器 6 复位（Timer6 reset） 由软件置`1`或清零。 • 0：无作用 • 1：复位 TIM6 定时器
位 3	TIM5RST：定时器 5 复位（Timer5 reset） 由软件置`1`或清零。 • 0：无作用 • 1：复位 TIM5 定时器
位 2	TIM4RST：定时器 4 复位（Timer4 reset） 由软件置’1’或清零。 • 0：无作用 • 1：复位 TIM4 定时器
位 1	TIM3RST：定时器 3 复位（Timer3 reset） 由软件置’1’或清零。 • 0：无作用 • 1：复位 TIM3 定时器
位 0	TIM2RST：定时器 2 复位（Timer2 reset） 由软件置’1’或清零。 • 0：无作用 • 1：复位 TIM2 定时器

8.3.6 AHB 外设时钟使能寄存器（RCC_AHBENR）

偏移地址：0x14

复位值：0x00000014

访问：无等待周期，支持字、半字和字节访问

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res					SDIOEN	Res	FSMCEN	Res	CRCEN	Res	FLITFEN	Res	SRAMEN	DMA2EN	DMA1EN
					rw		rw		rw		rw		rw	rw	rw

位 31:11	Res：保留

	必须保持复位值。
位 10	SDIOEN：SDIO 时钟使能（SDIO clock enable） 由软件置'1'或清零。 • 0：SDIO 时钟关闭 • 1：SDIO 时钟开启
位 9	Res：保留 必须保持复位值。
位 8	FSMCEN：FSMC 时钟使能（FSMC clock enable） 由软件置'1'或清零。 • 0：FSMC 时钟关闭 • 1：FSMC 时钟开启
位 7	Res：保留 必须保持复位值。
位 6	CRCEN：CRC 时钟使能（CRC clock enable） 由软件置'1'或清零。 • 0：CRC 时钟关闭 • 1：CRC 时钟开启
位 5	Res：保留 必须保持复位值。
位 4	FLITFEN：Flash 接口电路时钟使能（FLITF clock enable） 由软件置'1'或清零来开启或关闭睡眠模式下 Flash 接口电路时钟。 • 0：睡眠模式下，Flash 接口电路时钟关闭 • 1：睡眠模式下，Flash 接口电路时钟开启
位 3	Res：保留 必须保持复位值。
位 2	SRAMEN：SRAM 时钟使能（SRAM interface clock enable） 由软件置'1'或清零来开启或关闭睡眠模式下 SRAM 时钟。 • 0：睡眠模式时，SRAM 时钟关闭 • 1：睡眠模式时，SRAM 时钟开启
位 1	DMA2EN：DMA2 时钟使能（DMA2 clock enable） 由软件置'1'或清零。 • 0：DMA2 时钟关闭 • 1：DMA2 时钟开启
位 0	DMA1EN：DMA1 时钟使能（DMA1 clock enable） 由软件置'1'或清零。 • 0：DMA1 时钟关闭 • 1：DMA1 时钟开启

8.3.7 APB2 外设时钟使能寄存器（RCC_APB2ENR）

偏移地址：0x18

复位值：0x0000 0000

访问：支持字，半字和字节访问

通常访问无等待周期。但当 APB2 总线上的外设被访问时，将插入等待状态直到 APB2 的外设访问结束。

说明：当外设时钟没有启用时，软件不能读出外设寄存器的数值，返回的数值始终是 0x0。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC3 EN	USART1 EN	TIM8 EN	SPI1 EN	TIM1 EN	ADC2 EN	ADC1 EN	Res		IOPE EN	IOPD EN	IOPC EN	IOPB EN	IOPA EN	Res	AFIO EN
rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw		rw

位 31:16	Res：保留 必须保持复位值。
位 15	ADC3EN：ADC3 接口时钟使能（ADC3 interface clock enable） 由软件置'1'或清零。 • 0：ADC3 接口时钟关闭 • 1：ADC3 接口时钟开启
位 14	USART1EN：USART1 时钟使能（USART1 clock enable） 由软件置'1'或清零。 • 0：USART1 时钟关闭 • 1：USART1 时钟开启
位 13	TIM8EN：TIM8 时钟使能（TIM8 clock enable） 由软件置'1'或清零。 • 0：TIM8 时钟关闭 • 1：TIM8 时钟开启
位 12	SPI1EN：SPI1 时钟使能（SPI1 clock enable） 由软件置'1'或清零。 • 0：SPI1 时钟关闭 • 1：SPI1 时钟开启
位 11	TIM1EN：TIM1 定时器时钟使能（TIM1 Timer clock enable） 由软件置'1'或清零。 • 0：TIM1 定时器时钟关闭 • 1：TIM1 定时器时钟开启
位 10	ADC2EN：ADC2 接口时钟使能（ADC2 interface clock enable） 由软件置'1'或清零。 • 0：ADC2 接口时钟关闭 • 1：ADC2 接口时钟开启
位 9	ADC1EN：ADC1 接口时钟使能（ADC1 interface clock enable）

	由软件置’1’或清零。 • 0：ADC1 接口时钟关闭 • 1：ADC1 接口时钟开启
位 8:7	Res：保留 必须保持复位值。
位 6	IOPEEN：GPIOE 组端口的时钟使能（I/O port E clock enable） 由软件置’1’或清零。 • 0：GPIOE 组端口的时钟关闭 • 1：GPIOE 组端口的时钟开启
位 5	IOPDEN：GPIOD 组端口的时钟使能（I/O port D clock enable） 由软件置’1’或清零。 • 0：GPIOD 组端口的时钟关闭 • 1：GPIOD 组端口的时钟开启
位 4	IOPCEN：GPIOC 组端口的时钟使能（I/O port C clock enable） 由软件置’1’或清零。 • 0：GPIOC 组端口的时钟关闭 • 1：GPIOC 组端口的时钟开启
位 3	IOPBEN：GPIOB 组端口的时钟使能（I/O port B clock enable） 由软件置’1’或清零。 • 0：GPIOB 组端口的时钟关闭 • 1：GPIOB 组端口的时钟开启
位 2	IOPAEN：GPIOA 组端口的时钟使能（I/O port A clock enable） 由软件置’1’或清零。 • 0：GPIOA 组端口的时钟关闭 • 1：GPIOA 组端口的时钟开启
位 1	Res：保留 必须保持复位值。
位 0	AFIOEN：复用功能 I/O 时钟使能（Alternate function I/O clock enable） 由软件置’1’或清零。 • 0：复用功能 I/O 时钟关闭 • 1：复用功能 I/O 时钟开启

8.3.8 APB1 外设时钟使能寄存器（RCC_APB1ENR）

偏移地址：0x1C

复位值：0x0000 0000

访问：支持字、半字和字节访问

通常无访问等待周期。但在 APB1 总线上的外设被访问时，将插入等待状态直到 APB1 外设访问结束。

说明：当外设时钟没有启用时，软件不能读出外设寄存器的数值，返回的数值始终是 0x0。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res		DACEN	PWREN	BKPEN	CAN2EN	CAN1EN	Res	USBEN	I2C2EN	I2C1EN	USART5EN	USART4EN	USART3EN	USART2EN	Res
		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3EN	SPI2EN	Res		WWDGEN	Res					TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
rw	rw			rw						rw	rw	rw	rw	rw	rw

位 31:30	Res：保留 必须保持复位值。
位 29	DACEN：DAC 接口时钟使能（DAC interface clock enable） 由软件置’1’或清零。 • 0：DAC 接口时钟关闭 • 1：DAC 接口时钟开启
位 28	PWREN：电源接口时钟使能（Power interface clock enable） 由软件置’1’或清零。 • 0：电源接口时钟关闭 • 1：电源接口时钟开启
位 27	BKPEN：备份接口时钟使能（Backup interface clock enable） 由软件置’1’或清零。 • 0：备份接口时钟关闭 • 1：备份接口时钟开启
位 26	CAN2EN：CAN2 时钟使能（CAN2 clock enable） 由软件置’1’或清零。 • 0：CAN2 时钟关闭 • 1：CAN2 时钟开启
位 25	CAN1EN：CAN1 时钟使能（CAN1 clock enable） 由软件置’1’或清零。 • 0：CAN1 时钟关闭 • 1：CAN1 时钟开启
位 24	Res：保留 必须保持复位值。
位 23	USBEN：USB 时钟使能（USB clock enable） 由软件置’1’或清零。 • 0：USB 时钟关闭 • 1：USB 时钟开启
位 22	I2C2EN：I2C2 时钟使能（I2C2 clock enable） 由软件置’1’或清零。 • 0：I2C2 时钟关闭

	• 1：I2C2 时钟开启
位 21	I2C1EN：I2C1 时钟使能（I2C1 clock enable） 由软件置’1’或清零。 • 0：I2C1 时钟关闭 • 1：I2C1 时钟开启
位 20	USART5EN：USART5 时钟使能（USART5 clock enable） 由软件置’1’或清零。 • 0：USART5 时钟关闭 • 1：USART5 时钟开启
位 19	USART4EN：USART4 时钟使能（USART4 clock enable） 由软件置’1’或清零。 • 0：USART4 时钟关闭 • 1：USART4 时钟开启
位 18	USART3EN：USART3 时钟使能（USART3 clock enable） 由软件置’1’或清零。 • 0：USART3 时钟关闭 • 1：USART3 时钟开启
位 17	USART2EN：USART2 时钟使能（USART2 clock enable） 由软件置’1’或清零。 • 0：USART2 时钟关闭 • 1：USART2 时钟开启
位 16	Res：保留 必须保持复位值。
位 15	SPI3EN：SPI3 时钟使能（SPI3 clock enable） 由软件置’1’或清零。 • 0：SPI3 时钟关闭 • 1：SPI3 时钟开启
位 14	SPI2EN：SPI2 时钟使能（SPI2 clock enable） 由软件置’1’或清零。 • 0：SPI2 时钟关闭 • 1：SPI2 时钟开启
位 13:12	Res：保留 必须保持复位值。
位 11	WWDGEN：窗口看门狗时钟使能（Window watchdog clock enable） 由软件置’1’或清零。 • 0：窗口看门狗时钟关闭 • 1：窗口看门狗时钟开启
位 10:6	Res：保留

	必须保持复位值。
位 5	TIM7EN：定时器 7 时钟使能（Timer7 clock enable） 由软件置'1'或清零。 • 0：定时器 7 时钟关闭 • 1：定时器 7 时钟开启
位 4	TIM6EN：定时器 6 时钟使能（Timer6 clock enable） 由软件置'1'或清零。 • 0：定时器 6 时钟关闭 • 1：定时器 6 时钟开启
位 3	TIM5EN：定时器 5 时钟使能（Timer5 clock enable） 由软件置'1'或清零。 • 0：定时器 5 时钟关闭 • 1：定时器 5 时钟开启
位 2	TIM4EN：定时器 4 时钟使能（Timer4 clock enable） 由软件置'1'或清零。 • 0：定时器 4 时钟关闭 • 1：定时器 4 时钟开启
位 1	TIM3EN：定时器 3 时钟使能（Timer3 clock enable） 由软件置'1'或清零。 • 0：定时器 3 时钟关闭 • 1：定时器 3 时钟开启
位 0	TIM2EN：定时器 2 时钟使能（Timer2 clock enable） 由软件置'1'或清零。 • 0：定时器 2 时钟关闭 • 1：定时器 2 时钟开启

8.3.9 备份域控制寄存器（RCC_BDCR）

偏移地址：0x20

复位值：0x0000 0000

访问：0 到 3 等待周期，支持字、半字和字节访问；当连续对该寄存器进行访问时，将插入等待状态。

注意：寄存器 RCC_BDCR 中的 LSEON、LSEBYP、RTCSEL 和 RTCEN 位处于备份域。因此，这些位在复位后处于写保护状态，只有在电源控制寄存器（PWR_CR）中的 DBP 位置'1'后，才能对这些位进行修改。详情请参考："7 备份寄存器（BKP）"。这些位只能由备份域复位清除（见章节："8.1.3 备份域复位"）。任何内部或外部复位都不会影响这些位。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCEN	Res					RTCSEL[1:0]		Res					LSEBYP	LSERDY	LSEON
rw						rw							rw	r	rw

位 31:17	Res：保留 必须保持复位值。

位 16	BDRST：备份域软件复位（Backup domain software reset） 由软件置'1'或清零。 • 0：复位未激活 • 1：复位整个备份域
位 15	RTCEN：RTC 时钟使能（RTC clock enable） 由软件置'1'或清零。 • 0：RTC 时钟关闭 • 1：RTC 时钟开启
位 14:10	Res：保留 必须保持复位值。
位 9:8	RTCSEL[1:0]：RTC 时钟源选择（RTC clock source selection） 由软件设置来选择 RTC 时钟源。一旦 RTC 时钟源被选定，直到下次备份域被复位之前，它不能被改变。可通过设置 BDRST 位来清除。 • 00：无时钟 • 01：LSE 振荡器作为 RTC 时钟 • 10：LSI 振荡器作为 RTC 时钟 • 11：HSE 振荡器 128 分频后作为 RTC 时钟
位 7:3	Res：保留 必须保持复位值。
位 2	LSEBYP：外部低速时钟振荡器旁路（External low-speed oscillator bypass） 在调试模式下，由软件置'1'或清零来旁路 LSE。 只有在外部 32 kHz 振荡器关闭时，才能写入该位。 • 0：LSE 时钟未被旁路 • 1：LSE 时钟被旁路
位 1	LSERDY：外部低速振荡器就绪（External low-speed oscillator ready） 由硬件置'1'或清零来指示外部 32 kHz 振荡器是否就绪。 在 LSEON 被清零后，该位需要 6 个外部低速振荡器的周期才被清零。 • 0：外部 32 kHz 振荡器未就绪 • 1：外部 32 kHz 振荡器就绪
位 0	LSEON：外部低速振荡器使能（External low-speed oscillator enable） 由软件置'1'或清零。 • 0：外部 32 kHz 振荡器关闭 • 1：外部 32 kHz 振荡器开启

8.3.10 时钟控制/状态寄存器（RCC_CSR）

偏移地址：0x24

复位值：0x0C000000

说明：除复位标志外由系统复位清除，复位标志只能由电源复位清除。支持 0 到 3 等待周期访问，字、半字和字节访问。

当连续对该寄存器进行访问时，将插入等待状态。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPWRRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	PORRSTF	PINRSTF	Res	RMVF	Res							
rw	rw	rw	rw	rw	rw		rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res														LSIRDY	LSION
														r	rw

位 31	LPWRRSTF：低功耗复位标志（Low-power reset flag） 在低功耗管理复位发生时，由硬件置'1'。由软件通过写 RMVF 位清除。 • 0：无低功耗管理复位发生 • 1：发生低功耗管理复位 关于低功耗管理复位的详细信息，请参考章节："8.1.1.2 低功耗管理复位"。
位 30	WWDGRSTF：窗口看门狗复位标志（Window watchdog reset flag） 在窗口看门狗复位发生时，由硬件置'1'。由软件通过写 RMVF 位清除。 • 0：无窗口看门狗复位发生 • 1：发生窗口看门狗复位
位 29	IWDGRSTF：独立看门狗复位标志（Independent watchdog reset flag） 在独立看门狗复位发生在 V_{DD} 区域时，由硬件置'1'。由软件通过写 RMVF 位清除。 • 0：无独立看门狗复位发生 • 1：发生独立看门狗复位
位 28	SFTRSTF：软件复位标志（Software reset flag） 在软件复位发生时，由硬件置'1'。由软件通过写 RMVF 位清除。 • 0：无软件复位发生 • 1：发生软件复位
位 27	PORRSTF：上电/掉电复位标志（POR/PDR reset flag） 在上电/掉电复位发生时，由硬件置'1'。由软件通过写 RMVF 位清除。 • 0：无上电/掉电复位发生 • 1：发生上电/掉电复位
位 26	PINRSTF：NRST 引脚复位标志（PIN reset flag） 在 NRST 引脚复位发生时，由硬件置'1'。由软件通过写 RMVF 位清除。 • 0：无 NRST 引脚复位发生 • 1：发生 NRST 引脚复位
位 25	Res：保留 必须保持复位值。
位 24	RMVF：清除复位标志（Remove reset flag） 由软件置'1'来清除复位标志。 • 0：无作用 • 1：清除复位标志
位 23:2	Res：保留 必须保持复位值。

位 1	LSIRDY：内部低速振荡器就绪（Internal low-speed oscillator ready） 由硬件置'1'或清零来指示内部 LSI 振荡器是否就绪。 在 LSION 清零后，在 3 个内部 LSI 振荡器的周期后 LSIRDY 被清零。 • 0：内部 LSI 振荡器时钟未就绪 • 1：内部 LSI 振荡器时钟就绪
位 0	LSION：内部低速振荡器使能（Internal low-speed oscillator enable） 由软件置'1'或清零。 • 0：内部 LSI 振荡器关闭 • 1：内部 LSI 振荡器开启

8.3.11 时钟配置寄存器 2（RCC_CFGR2）

偏移地址：0x2C

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res												PREDIV[3:0]			
												rw			

位 31:4	Res：保留 必须保持复位值。
位 3:0	PREDIV[3:0]：用于配置 PLL 的时钟输入前置分频（PLL division factor） 只能在 PLL 关闭的时候修改 PREDIV 的值。 PREDIV[0]和 RCC_CFGR.PLLXTPRE 相同。修改 PREDIV[0]的值时，RCC_CFGR.PLLXTPRE 的值也会同时被修改。 分频数等于 PREDIV 的值加 1，具体为： • 0000：不分频 • 0001：分频系数为 1/2 • 0010：分频系数为 1/3 • … • 1111：分频系数为 1/16

8.3.12 时钟配置寄存器 3（RCC_CFGR3）

偏移地址：0x30

复位值：0x00000800

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res			ADC3SW[2:0]			ADC2SW[2:0]			ADC1SW[2:0]			Res			
			rw			rw			rw						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res		USBIFSW[1:0]		USBCLKEN	USBSW[2:0]			Res							
		rw		rw	rw										

位 31:29	Res：保留

	必须保持复位值。
位 28:26	ADC3SW[2:0]：用于控制 ADC3 的时钟频率（ADC3 clock frequency selection） 由软件置位和清零，与 RCC_CFGR.ADCPRE 配合控制 ADC3 的时钟频率。 • 000：ADC3 时钟由 RCC_CFGR.ADCPRE 控制 • 001：ADC3 时钟为 PCLK2/10 • 010：ADC3 时钟为 PCLK2/12 • 011：ADC3 时钟为 PCLK2/14 • 100：ADC3 时钟为 PCLK2/16 • 101：ADC3 时钟为 PCLK2/20 • 其他：保留
位 25:23	ADC2SW[2:0]：用于控制 ADC2 的时钟频率（ADC2 clock frequency selection） 由软件置位和清零，与 RCC_CFGR.ADCPRE 配合控制 ADC2 的时钟频率。 • 000：ADC2 时钟由 RCC_CFGR.ADCPRE 控制 • 001：ADC2 时钟为 PCLK2/10 • 010：ADC2 时钟为 PCLK2/12 • 011：ADC2 时钟为 PCLK2/14 • 100：ADC2 时钟为 PCLK2/16 • 101：ADC2 时钟为 PCLK2/20 • 其他：保留
位 22:20	ADC1SW[2:0]：用于控制 ADC1 的时钟频率（ADC1 clock frequency selection） 由软件置位和清零，与 RCC_CFGR.ADCPRE 配合控制 ADC1 的时钟频率。 • 000：ADC1 时钟由 RCC_CFGR.ADCPRE 控制 • 001：ADC1 时钟为 PCLK2/10 • 010：ADC1 时钟为 PCLK2/12 • 011：ADC1 时钟为 PCLK2/14 • 100：ADC1 时钟为 PCLK2/16 • 101：ADC1 时钟为 PCLK2/20 • 其他：保留
位 19:14	Res：保留 必须保持复位值。
位 13:12	USBIFSW[1:0]： USB 接口的时钟频率选择（USB clock frequency selection） 由软件置位和清零，用于配置 USB 接口时钟频率。 • 00：PCLK1 作为 USB 接口时钟 • 01：PCLK1/2 作为 USB 接口时钟 • 10：PCLK1/3 作为 USB 接口时钟 • 11：PCLK1/4 作为 USB 接口时钟
位 11	USBCLKEN：USB 时钟使能（USB clock enable） 用于使能 48MHz USB 时钟。 • 0：关闭 48MHz USB 时钟 • 1：打开 48MHz USB 时钟

位 10:8	USBSW[2:0]：用于控制 USB 的时钟频率（USB clock frequency selection） 由软件置位和清零，和 RCC_CFGR.USBPRE 一起配置产生 48 MHz USB 时钟。 • 000：48M USB 时钟由 RCC_CFGR.USBPRE 配置决定 • 001：PLLCLK/2 • 010：PLLCLK/2.5 • 011：PLLCLK/3 • 100：PLLCLK/3.5 • 101：PLLCLK/4
位 7:0	Res：保留 必须保持复位值。

8.3.13 时钟控制寄存器 2（RCC_CR2）

偏移地址：0x34

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res												HSI28RDY	HSI28ON	HSI56RDY	HSI56ON
												r	rw	r	rw

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

位 31:20	Res：保留 必须保持复位值。
位 19	HSI28RDY：HSI28 时钟是否稳定（HSI28 clock ready flag） • 0：HSI28 还未稳定 • 1：HSI28 已稳定
位 18	HSI28ON：使能 HSI28 时钟（HSI28 clock enable） • 0：HSI28 时钟关闭 • 1：HSI28 时钟使能
位 17	HSI56RDY：HSI56 时钟是否稳定（HSI56 clock ready flag） • 0：HSI56 还未稳定 • 1：HSI56 已稳定
位 16	HSI56ON：使能 HSI56 时钟（HSI56 clock enable） • 0：HSI56 时钟关闭 • 1：HSI56 时钟使能
位 15:0	Res：保留 必须保持复位值。

8.3.14 HSE 控制寄存器（RCC_HSECTL）

偏移地址：0xE0

复位值：0x1F2C0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CSS_THRESHOLD[6:0]							HSENF_BYP	XTAL32M[7:0]							
rw							rw	rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res			HSE_STOP_KEEP	HSEWT[11:0]											
			rw	rw											

位 31:25	CSS_THRESHOLD[6:0]：控制 CSS 检查的 HSE 最低频率阈值（CSS counter threshold control） 在 CSS 使能后，如果 HSE 输入的频率小于 4M/CSS_THRESHOLD，则会产生 CSSF 并触发 NMI。
位 24	HSENF_BYP：开或关 HSE 去噪声功能（HSE noise filter bypass） • 1：关闭 HSE 去噪声功能 • 0：打开 HSE 去噪声功能
位 23:16	XTAL32M[7:0]：调整 AGC 的驱动能力（AGC driven capacity setting）
位 15:13	Res：保留 必须保持复位值。
位 12	HSE_STOP_KEEP：在停机（Stop）模式下，控制 HSE 是否关闭（Quickly switch to HSE on wakeup from Stop mode） • 0：在进入停机（Stop）模式时，强制关闭 HSE。 • 1：在进入停机（Stop）模式时，RCC_CR.HSEON 不会被硬件清除。如果在进入停机（Stop）模式时 HSE 为使能状态，则在该模式下 HSE 也一直保持打开状态。这样从停机（Stop）模式唤醒后，SYSCLK 能快速的从 HSI 切换到 HSE 时钟。
位 11:0	HSEWT[11:0]：HSE 稳定时间设置（HSE wait time for stabilization setting） 当 HSE 从关闭到打开时，收到 HSEWT x 8 个 HSE 时钟周期后 HSERDY 置位 1。

8.3.15 PLL 控制寄存器（RCC_PLLCTL）

偏移地址：0xE4

复位值：0x4XXX XXXX

说明：X 表示不定值。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PLLWT[4:0]					Res										
rw															

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

位 31:27	PLLWT[4:0]：PLL 稳定时间设置（PLL wait time for stabilization setting） 当 PLL 从关闭到打开时，直到 PLLWT x 128 个 PLL 时钟周期后 PLLRDY 置为 1。
位 26:0	Res：保留 必须保持复位值。

8.3.16 时钟配置寄存器 4（RCC_CFGR4）

偏移地址：0xE8

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res											PPSS	Res	PLLMULH[2:0]		
											rw		rw		

位 31:5	Res：保留 必须保持复位值。
位 4	PPSS：PLL 前置分频器输入时钟选择（PLL pre-scaler clock source selection） 由软件置位和复位。 • 0：HSE 时钟输入给 PLL 前置分频器 • 1：HSI56 时钟输入给 PLL 前置分频器
位 3	Res：保留 必须保持复位值。
位 2:0	PLLMULH[2:0]：控制 PLL 倍频数（PLL multiplication factor control） 由软件置位和复位，与 RCC_CFGR.PLLMULL 一起控制 PLL 的倍频数。 • 000：PLL 倍频数由 RCC_CFGR.PLLMULL 决定 • 001~111：PLL 倍频数等于{PLLMULH[2:0]|RCC_CFGR.PLLMUL[3:0]}

8.3.17 时钟配置寄存器 5（RCC_CFGR5）

偏移地址：0xEC

复位值：0x0000 0070

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PLLNODIV	MCOPRE[2:0]			MCO[3]	Res								FLITFCLK_PRE[4:2]		
rw	rw			rw									rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLITFCLK_PRE[1:0]		FLITFCLK_SEL[1:0]		ESSS	Res	EXTCLK_SEL[1:0]		Res	ESWS[2:0]			Res	ESW[2:0]		
rw		rw		rw		rw			rw				rw		

位 31	PLLNODIV：控制 PLL 是否分频后输出至 MCO（PLL divider for MCO） • 0：PLL/2 输出至 MCO • 1：PLL 输出至 MCO
位 30:28	MCOPRE[2:0]：MCO 输出分频系数（Microcontroller clock output prescaler） • 000：MCO/1 • 001：MCO/2 • 010：MCO/4 • … • 111：MCO/128
位 27	MCO[3]：时钟输出选择（Clock output） 该位和 RCC_CFGR.MCO[2:0]一起组成 4 位的 MCO 选择控制寄存器 MCO[3:0]。 具体见时钟配置寄存器（RCC_CFGR）描述部分。
位 26:19	Res：保留

	必须保持复位值。
位 18:14	FLITFCLK_PRE[4:0]：FLITFCLK 分频系数（FLITFCLK prescaler factor） 当执行 Flash 擦除或编程时，分频后的 FLITFCLK 必须等于 8 MHz。分频系数等于 FLITFCLK_PRE+1（最多 32）。 在执行 Flash 编程和擦除时，不能更改 FLITFCLK_SEL 寄存器的值。
位 13:12	FLITFCLK_SEL[1:0]：FLITFCLK 分频时钟源选择（The prescaler clock source of FLITFCLK selection） • 00：选择 HSI（8 MHz）分频时钟作为 FLITFCLK 分频时钟源 • 01：选择 SYSCLK 作为 FLITFCLK 分频时钟源 • 10：选择 GPIO 输入时钟 EXTCLK 作为 FLITFCLK 分频时钟源 • 11：保留 在执行 Flash 编程和擦除时，不能更改 FLITFCLK_SEL 寄存器的值。
位 11	ESSS：系统时钟切换补充寄存器使能标志（SYSCLK setting bit selection） 该位由软件置位和复位。 • 0：使用 RCC_CFGR.SW 和 RCC_CFGR.SWS 选择和标识 SYSCLK • 1：使用 RCC_CFGR5.ESW 和 RCC_CFGR5.ESWS 选择和标识 SYSCLK
位 10	Res：保留 必须保持复位值。
位 9:8	EXTCLK_SEL[1:0]：GPIO 输入时钟选择（External clock pin selection） • 00：选择 PA[1]作为 GPIO 时钟输入 • 01：选择 PB[1]作为 GPIO 时钟输入 • 10：选择 PC[7]作为 GPIO 时钟输入 • 11：选择 PB[7]作为 GPIO 时钟输入
位 7	Res：保留 必须保持复位值。
位 6:4	ESWS[2:0]：系统时钟切换状态（SYSCLK clock source status） • 000：LSE 作为 SYSCLK • 001：LSI 作为 SYSCLK • 010：HSI56 输出作为 SYSCLK • 011：HSI28 分频时钟作为 SYSCLK • 100：EXTCLK 输入时钟作为 SYSCLK • 其他：保留
位 3	Res：保留 必须保持复位值。
位 2:0	ESW[2:0]：系统时钟切换控制（SYSCLK clock source selection） • 000：选择 LSE 作为 SYSCLK • 001：选择 LSI 作为 SYSCLK • 010：选择 HSI56 输出作为 SYSCLK • 011：选择 HSI28 分频时钟作为 SYSCLK • 100：选择 GPIO 输入时钟作为 SYSCLK

	• 其他：保留

8.3.18 时钟配置寄存器 6（RCC_CFGR6）

偏移地址：0xF0

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res						SAI_B_SW[1:0]		SAI_B_PRE[2:0]			SAI_A_SW[1:0]		SAI_A_PRE[2:0]		
						rw		rw			rw		rw		

位 31:10	Res：保留 必须保持复位值。
位 9:8	SAI_B_SW[1:0]：选择 SAI_B 时钟输入（SAI_B clock source selection） • 00：PCLK2/SAI_B_PRE 作为 SAI_B 时钟输入 • 01：GPIO 输入时钟 EXTCLK 作为 SAI_B 时钟输入 • 10：HSE 作为 SAI_B 时钟输入 • 11：保留
位 7:5	SAI_B_PRE[2:0]：选择 PCLK2 分频时钟作为 SAI_B 时钟输入的频率（SAI_B clock selection based on PCLK2 prescaler） 由软件置位和清零。 • 000：PCLK2/2 作为 SAI_B 接口时钟 • 001：PCLK2/4 作为 SAI_B 接口时钟 • 010：PCLK2/6 作为 SAI_B 接口时钟 • 011：PCLK2/8 作为 SAI_B 接口时钟 • 100：PCLK2/10 作为 SAI_B 接口时钟 • 101：PCLK2/12 作为 SAI_B 接口时钟 • 110：PCLK2/14 作为 SAI_B 接口时钟 • 111：PCLK2/16 作为 SAI_B 接口时钟
位 4:3	SAI_A_SW[1:0]：选择 SAI_A 时钟输入（SAI_A clock source selection） • 00：PCLK2/SAI_A_PRE 作为 SAI_A 时钟输入 • 01：GPIO 输入时钟 EXTCLK 作为 SAI_A 时钟输入 • 10：HSE 作为 SAI_A 时钟输入 • 11：保留
位 2:0	SAI_A_PRE[2:0]：选择 PCLK2 分频时钟作为 SAI_A 时钟输入的频率（SAI_A clock selection based on PCLK2 prescaler） 由软件置位和清零。 • 000：PCLK2/2 作为 SAI_A 接口时钟 • 001：PCLK2/4 作为 SAI_A 接口时钟 • 010：PCLK2/6 作为 SAI_A 接口时钟 • 011：PCLK2/8 作为 SAI_A 接口时钟 • 100：PCLK2/10 作为 SAI_A 接口时钟

	• 101：PCLK2/12 作为 SAI_A 接口时钟 • 110：PCLK2/14 作为 SAI_A 接口时钟 • 111：PCLK2/16 作为 SAI_A 接口时钟

8.3.19 AHB 外设时钟使能寄存器 2（RCC_AHBENR2）

偏移地址：0x100

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res										QSPIEN	DCMIEN	TRNGEN	HASHEN	AESEN	COALUEN
										rw	rw	rw	rw	rw	rw

位 31:6	Res：保留 必须保持复位值。
位 5	QSPIEN：QSPI 时钟使能（QSPI clock enable） • 0：关闭 QSPI 时钟 • 1：使能 QSPI 时钟
位 4	DCMIEN：DCMI 时钟使能（DCMI clock enable） • 0：关闭 DCMI 时钟 • 1：使能 DCMI 时钟
位 3	TRNGEN：TRNG 时钟使能（TRNG clock enable） • 0：关闭 TRNG 时钟 • 1：使能 TRNG 时钟
位 2	HASHEN：HASH 时钟使能（HASH clock enable） • 0：关闭 HASH 时钟 • 1：使能 HASH 时钟
位 1	AESEN：AES 时钟使能（AES clock enable） • 0：关闭 AES 时钟 • 1：使能 AES 时钟
位 0	COALUEN：COALU 时钟使能（COALU clock enable） • 0：关闭 COALU 时钟 • 1：使能 COALU 时钟

8.3.20 APB2 外设时钟使能寄存器 2（RCC_APB2ENR2）

偏移地址：0x104

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res												SAIBEN	SAIAEN	COMPEN	USART6EN
												rw	rw	rw	rw

位 31:4	Res：保留 必须保持复位值。
位 3	SAIBEN：SAI_B 时钟使能（SAI_B clock enable） • 0：关闭 SAI_B 时钟 • 1：使能 SAI_B 时钟
位 2	SAIAEN：SAI_A 时钟使能（SAI_A clock enable） • 0：关闭 SAI_A 时钟 • 1：使能 SAI_A 时钟
位 1	COMPEN：比较器时钟使能（Voltage compare clock enable） • 0：关闭比较器时钟 • 1：使能比较器时钟
位 0	USART6EN：USART6 时钟使能（USART6 clock enable） • 0：关闭 USART6 时钟 • 1：使能 USART6 时钟

8.3.21 AHB 外设复位寄存器 2（RCC_AHBRST2）

偏移地址：0x10C

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res					SDIORST	Res	FSMCRST	Res	CRCRST	Res				DMA2RST	DMA1RST
					rw		rw		rw					rw	rw

位 31:11	Res：保留 必须保持复位值。
位 10	SDIORST：控制 SDIO 复位（SDIO reset flag） • 0：无作用 • 1：复位 SDIO
位 9	Res：保留 必须保持复位值。
位 8	FSMCRST：控制 FSMC 复位（FSMC reset flag） • 0：无作用 • 1：复位 FSMC
位 7	Res：保留 必须保持复位值。

位 6	CRCRST：控制 CRC 复位（CRC reset flag） • 0：无作用 • 1：复位 CRC
位 5:2	Res：保留 必须保持复位值。
位 1	DMA2RST：控制 DMA2 复位（DMA2 reset flag） • 0：无作用 • 1：复位 DMA2
位 0	DMA1RST：控制 DMA1 复位（DMA1 reset flag） • 0：无作用 • 1：复位 DMA1

8.3.22 AHB 外设复位寄存器 3（RCC_AHBRST3）

偏移地址：0x110

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res							CACHERST	Res		QSPIRST	DCMIRST	TRNGRST	HASHRST	AESRST	COALURST
							rw			rw	rw	rw	rw	rw	rw

位 31:9	Res：保留 必须保持复位值。
位 8	CACHERST：控制 Cache 复位（Cache reset flag） • 0：无作用 • 1：复位 Cache
位 7:6	Res：保留 必须保持复位值。
位 5	QSPIRST：控制 QSPI 复位（QSPI reset flag） • 0：无作用 • 1：复位 QSPI
位 4	DCMIRST：控制 DCMI 复位（DCMI reset flag） • 0：无作用 • 1：复位 DCMI
位 3	TRNGRST：控制 TRNG 复位（TRNG reset flag） • 0：无作用 • 1：复位 TRNG
位 2	HASHRST：控制 HASH 复位（HASH reset flag）

	• 0：无作用 • 1：复位 HASH
位 1	AESRST：控制 AES 复位（AES reset flag） • 0：无作用 • 1：复位 AES
位 0	COALURST：控制 COALU 复位（COALU reset flag） • 0：无作用 • 1：复位 COALU

8.3.23 APB2 外设复位寄存器 2（RCC_APB2RSTR2）

偏移地址：0x114

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res												SAIBRST	SAIARST	Res	USART6RST
												rw	rw		rw

位 31:4	Res：保留 必须保持复位值。
位 3	SAIBRST：控制 SAI_B 复位（SAI_B reset flag） • 0：无作用 • 1：复位 SAI_B
位 2	SAIARST：控制 SAI_A 复位（SAI_A reset flag） • 0：无作用 • 1：复位 SAI_A
位 1	Res：保留 必须保持复位值。
位 0	USART6RST：控制 USART6 复位（USART6 reset flag） • 0：无作用 • 1：复位 USART6

8.3.24 LSI 时钟控制寄存器（RCC_LSICTL）

偏移地址：0x11C

复位值：0x0000 800D

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LSITRIM[7:0]								Res				LSI_FREQ[1:0]		Res	
rw												rw			

位 31:16	Res：保留 必须保持复位值。
位 15:8	LSITRIM[7:0]：调整 LSI 输出频率（LSI clock trimming） LSITRIM 的值越大，LSI 输出的频率越高。
位 7:4	Res：保留 必须保持复位值。
位 3:2	LSI_FREQ[1:0]：设置 LSI 默认中心频率（The default center frequency of LSI） • 11：LSI 中心频率 40 kHz • 10：LSI 中心频率 10 kHz • 01：LSI 中心频率 20 kHz • 00：LSI 中心频率 5 kHz
位 1:0	Res：保留 必须保持复位值。

9 中断和事件（EXTI 和 NVIC）

9.1 嵌套向量中断控制器（NVIC）

- 71 个可屏蔽中断通道（不包含 16 个 Cortex®-M3 的中断线）
- 16 个可编程的优先等级（使用了 4 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

嵌套向量中断控制器（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晚到的中断。

NVIC 管理着内核异常等中断。更多关于异常和 NVIC 编程的说明，请参考 Cortex®-M3 相关手册。

9.1.1 系统嘀嗒（SysTick）校准值寄存器

系统嘀嗒校准值固定为 9000，当系统嘀嗒时钟设定为 9 MHz（HCLK/8 且 HCLK=72 MHz），产生 1 ms 时间基准。

9.1.2 中断和异常向量

表 9-1 NVIC 表

位置	优先级		名称	描述	地址
-	-	-	-	保留	0x0000 0000
-	-3	固定	Reset	复位（Reset）	0x0000 0004
-	-2	固定	NMI	非屏蔽中断 RCC 时钟安全系统（CSS）联接到 NMI 向量 （Non-maskable interrupt）	0x0000 0008
-	-1	固定	HardFault	所有类型错误（All class of fault）	0x0000 000C
-	0	可配置	MemManage	存储器管理 （Memory management）	0x0000 0010
-	1	可配置	BusFault	预取指错误，存储器访问错误 （Pre-fetch instruction fault/Memory access fault）	0x0000 0014
	2	可配置	UsageFault	未定义的指令或非法状态 （Undefined instruction and illegal state）	0x0000 0018
-	-	-	-	保留	0x0000 001C- 0x0000 002B
-	3	可配置	SVCall	通过 SWI 指令的系统服务调用 （System service call via SWI instruction）	0x0000 002C
-	4	可配置	DebugMonitor	调试监控器	0x0000 0030
-	-	-	-	保留	0x0000 0034

位置	优先级		名称	描述	地址
-	5	可配置	PendSV	可挂起的系统服务请求 （Pendable request for system service）	0x0000 0038
-	6	可配置	SysTick	系统嘀嗒定时器 （System tick timer）	0x0000 003C
0	7	可配置	WWDG	窗口看门狗中断 （Window Watchdog interrupt）	0x0000 0040
1	8	可配置	PVD	连到 EXTI16 的电源电压检测（PVD）中断 （PVD through EXTI Line16 detection interrupt）	0x0000 0044
2	9	可配置	TAMPER	侵入检测中断（Tamper interrupt）	0x0000 0048
3	10	可配置	RTC	RTC 全局中断 （RTC global interrupt）	0x0000 004C
4	11	可配置	FLASH	Flash 全局中断 （Flash global interrupt）	0x0000 0050
5	12	可配置	RCC	复位和时钟控制（RCC）全局中断 （RCC global interrupt）	0x0000 0054
6	13	可配置	EXTI0	EXTI 线 0 的中断（EXTI Line0 interrupt）	0x0000 0058
7	14	可配置	EXTI1	EXTI 线 1 的中断（EXTI Line1 interrupt）	0x0000 005C
8	15	可配置	EXTI2	EXTI 线 2 的中断（EXTI Line2 interrupt）	0x0000 0060
9	16	可配置	EXTI3	EXTI 线 3 的中断（EXTI Line3 interrupt）	0x0000 0064
10	17	可配置	EXTI4	EXTI 线 4 的中断（EXTI Line4 interrupt）	0x0000 0068
11	18	可配置	DMA1_Channel1	DMA1 通道 1 全局中断 （DMA Channel1 global interrupt）	0x0000 006C
12	19	可配置	DMA1_Channel2	DMA1 通道 2 全局中断 （DMA Channel2 global interrupt）	0x0000 0070
13	20	可配置	DMA1_Channel3	DMA1 通道 3 全局中断 （DMA Channel3 global interrupt）	0x0000 0074
14	21	可配置	DMA1_Channel4	DMA1 通道 4 全局中断 （DMA Channel4 global interrupt）	0x0000 0078
15	22	可配置	DMA1_Channel5	DMA1 通道 5 全局中断 （DMA Channel5 global interrupt）	0x0000 007C
16	23	可配置	DMA1_Channel6	DMA1 通道 6 全局中断 （DMA Channel6 global interrupt）	0x0000 0080
17	24	可配置	DMA1_Channel7	DMA1 通道 7 全局中断 （DMA Channel7 global interrupt）	0x0000 0084
18	25	可配置	ADC1_2	ADC1 和 ADC2 全局中断（结合 EXTI24, EXTI25）	0x0000 0088

位置	优先级		名称	描述	地址
				（ADC1/ADC2 interrupt, combined with EXTI24/EXTI25）	
19	26	可配置	USB_HP_CAN_TX	USB 高优先级或 CAN 发送中断 （USB high priority/CAN Tx interrupt）	0x0000 008C
20	27	可配置	USB_LP_CAN_RX0	USB 低优先级或 CAN 接收 0 中断 （USB low priority/CAN Rx0 interrupt）	0x0000 0090
21	28	可配置	CAN_RX1	CAN 接收 1 中断 （CAN Rx1 interrupt）	0x0000 0094
22	29	可配置	CAN_SCE	CAN SCE 中断 （CAN SCE interrupt）	0x0000 0098
23	30	可配置	EXTI9_5	EXTI 线[9:5]中断 （EXTI Line[9:5] interrupt）	0x0000 009C
24	31	可配置	TIM1_BRK	TIM1 刹车中断 （TIM1 break interrupt）	0x0000 00A0
25	32	可配置	TIM1_UP	TIM1 更新中断 （TIM1 update interrupt）	0x0000 00A4
26	33	可配置	TIM1_TRG_COM	TIM1 触发和 COM 中断 （TIM1 trigger and COM interrupt）	0x0000 00A8
27	34	可配置	TIM1_CC	TIM1 捕获/比较中断 （TIM1 Capture/Compare interrupt）	0x0000 00AC
28	35	可配置	TIM2	TIM2 全局中断 （TIM2 global interrupt）	0x0000 00B0
29	36	可配置	TIM3	TIM3 全局中断 （TIM3 global interrupt）	0x0000 00B4
30	37	可配置	TIM4	TIM4 全局中断 （TIM4 global interrupt）	0x0000 00B8
31	38	可配置	I2C1_EV	I2C1 事件中断 （I2C1 event interrupt）	0x0000 00BC
32	39	可配置	I2C1_ER	I2C1 错误中断 （I2C1 fault interrupt）	0x0000 00C0
33	40	可配置	I2C2_EV	I2C2 事件中断 （I2C2 event interrupt）	0x0000 00C4
34	41	可配置	I2C2_ER	I2C2 错误中断 （I2C2 fault interrupt）	0x0000 00C8
35	42	可配置	SPI1	SPI1 全局中断 （SPI1 global interrupt）	0x0000 00CC
36	43	可配置	SPI2	SPI2 全局中断	0x0000 00D0

位置	优先级		名称	描述	地址
				（SPI2 global interrupt）	
37	44	可配置	USART1	USART1 全局中断 （USART1 global interrupt）	0x0000 00D4
38	45	可配置	USART2	USART2 全局中断 （USART2 global interrupt）	0x0000 00D8
39	46	可配置	USART3	USART3 全局中断 （USART3 global interrupt）	0x0000 00DC
40	47	可配置	EXTI15_10	EXTI 线[15:10]中断 （EXTI Line[15:10] interrupt）	0x0000 00E0
41	48	可配置	RTCAlarm	连接到 EXTI17 的 RTC 闹钟中断或唤醒定时器 （RTC alarm or wakeup timer through EXTI Line 17 interrupt）	0x0000 00E4
42	49	可配置	USBWakeUp	连接到 EXTI18 的 USB 待机唤醒中断 （USB standby wake-up through EXTI Line 18 interrupt）	0x0000 00E8
43	50	可配置	TIM8_BRK	TIM8 刹车中断 （TIM8 break interrupt）	0x0000 00EC
44	51	可配置	TIM8_UP	TIM8 更新中断 （TIM8 update interrupt）	0x0000 00F0
45	52	可配置	TIM8_TRG_COM	TIM8 触发和 COM 中断 （TIM8 trigger and COM interrupt）	0x0000 00F4
46	53	可配置	TIM8_CC	TIM8 捕获比较中断 （TIM8 Capture/Compare interrupt）	0x0000 00F8
47	54	可配置	ADC3	ADC3 全局中断（包括 EXTI26） （ADC3 global interrupt）	0x0000 00FC
48	55	可配置	FSMC	FSMC 全局中断 （FSMC global interrupt）	0x0000 0100
49	56	可配置	SDIO	SDIO 全局中断 （SDIO global interrupt）	0x0000 0104
50	57	可配置	TIM5	TIM5 全局中断 （TIM5 global interrupt）	0x0000 0108
51	58	可配置	SPI3	SPI3 全局中断 （SPI3 global interrupt）	0x0000 010C
52	59	可配置	USART4	USART4 全局中断 （USART4 global interrupt）	0x0000 0110
53	60	可配置	USART5	USART5 全局中断 （USART5 global interrupt）	0x0000 0114

位置	优先级		名称	描述	地址
54	61	可配置	TIM6	TIM6 全局中断 （TIM6 global interrupt）	0x0000 0118
55	62	可配置	TIM7	TIM7 全局中断 （TIM7 global interrupt）	0x0000 011C
56	63	可配置	DMA2_Channel1	DMA2 通道 1 全局中断 （DMA2 Channel1 global interrupt）	0x0000 0120
57	64	可配置	DMA2_Channel2	DMA2 通道 2 全局中断 （DMA2 Channel2 global interrupt）	0x0000 0124
58	65	可配置	DMA2_Channel3	DMA2 通道 3 全局中断 （DMA2 Channel3 global interrupt）	0x0000 0128
59	66	可配置	DMA2_Channel4_5	DMA2 通道 4 和通道 5 全局中断 （DMA2 Channel4/5 global interrupt）	0x0000 012C
60	67	可配置	QSPI	QSPI 中断 （QSPI interrupt）	0x0000 0130
61	68	可配置	DCMI	DCMI 中断 （DCMI interrupt）	0x0000 0134
62	69	可配置	COALU_AES	COALU/AES 中断 （COALU/AES interrupt）	0x0000 0138
63	70	可配置	CAN2_TX	CAN2 发送中断 （CAN2 Tx interrupt）	0x0000 013C
64	71	可配置	CAN2_RX0	CAN2 接收 0 中断 （CAN2 Rx0 interrupt）	0x0000 0140
65	72	可配置	CAN2_RX1	CAN2 接收 1 中断 （CAN2 Rx1 interrupt）	0x0000 0144
66	73	可配置	CAN2_SCE	CAN2 的 SCE 中断 （CAN2 SCE interrupt）	0x0000 0148
67	74	可配置	HASH	HASH 通用中断 （HASH interrupt）	0x0000 014C
68	75	可配置	EXTI23_20	EXTI 线[23:20]连接至电压比较器 COMP 的中断 （Voltage comparator through EXTI Line[23:20] interrupt）	0x0000 0150
69	76	可配置	USART6	USART6 全局中断 （USART6 global interrupt）	0x0000 0154
70	77	可配置	SAI	SAI_IRQ 中断 （SAI_IRQ Handler）	0x0000 0158

9.2 外部中断/事件控制器（EXTI）

扩展中断及事件控制器（EXTI）负责管理内、外异步中断和事件：向 CPU 输出事件请求，向中断控制器输出中断请求，向电源管理模块输出唤醒请求。

根据中断/事件触发沿是否可配置，可将 EXTI 分为两类：触发沿可配 EXTI（简称可配 EXTI）和触发沿固定 EXTI（简称固定 EXTI）。可配 EXTI 可选择上升沿/下降沿触发，挂起状态寄存器中记录中断状态。除硬件触发中断外，还可通过写软件中断事件寄存器 EXTI_SWIER 对应位来模拟生成中断/事件。而固定 EXTI 采用上升沿触发，仅工作在停机模式，用于从停机模式唤醒内核。挂起状态寄存器中无法查询固定 EXTI 中断状态，需由对应 IP 提供。

9.2.1 主要特性

- 支持多达 26 个事件/中断请求 EXTI 线
  - 23 根可配置 EXTI 线
    - 触发沿上升沿或下降沿可选
    - 有专用的中断状态位标记
    - 可通过软件方式触发中断、事件
  - 3 根固定 EXTI 线
- 每根中断/事件线都可单独被触发和屏蔽。
- 检测脉冲宽度低于 APB2 时钟宽度的外部信号。

9.2.2 框图

EXTI 结构框图如下所示：

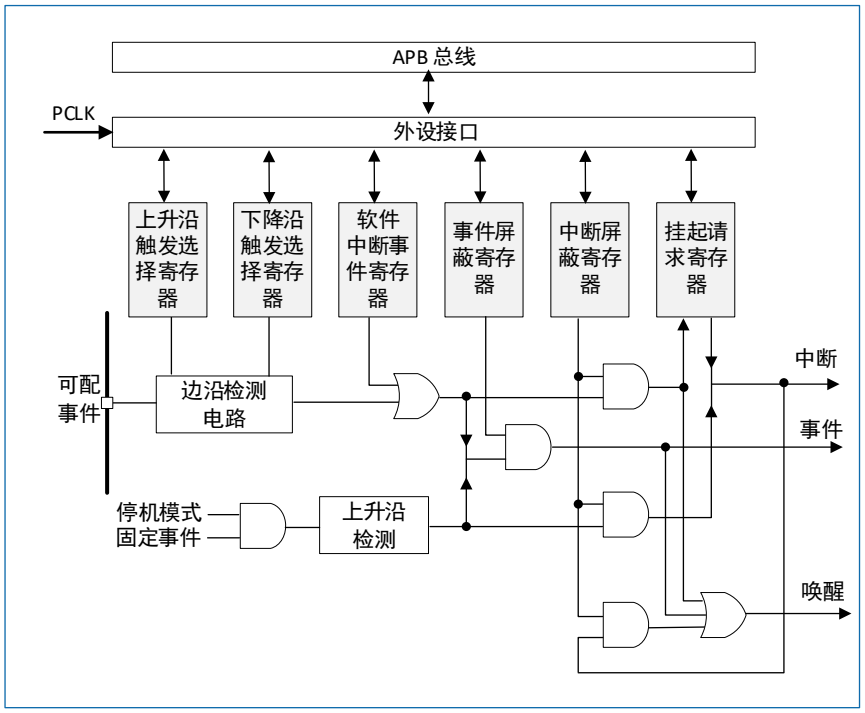


图 9-1 EXTI 框图

说明：以上中断为 26 个。

9.2.3 EXTI 与周边模块关系

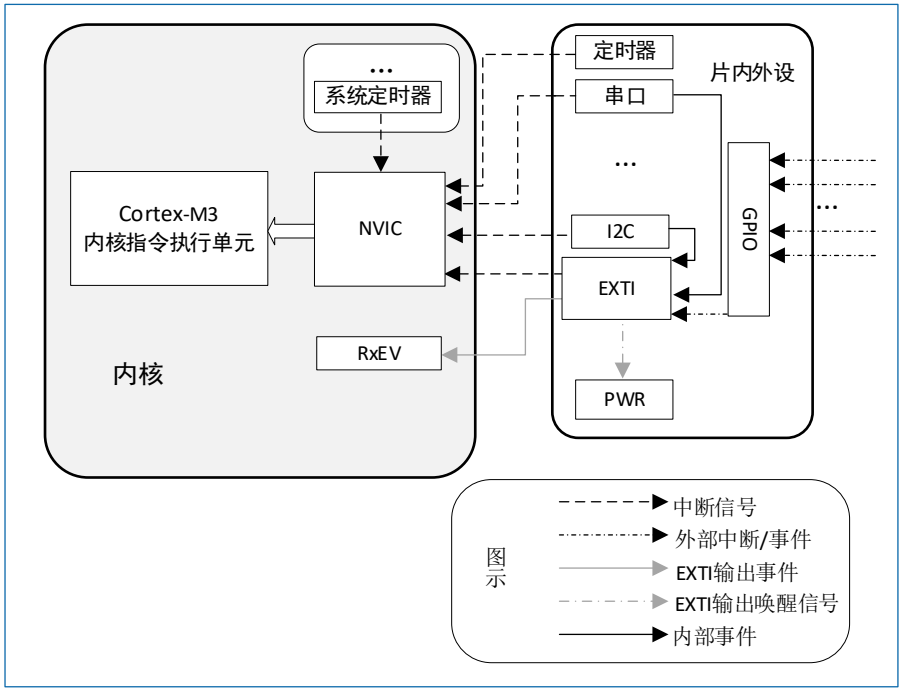


图 9-2 EXTI 与周边模块关系框图

如图 9-2 所示，EXTI 的中断输出与芯片内其他中断源一起汇总于 NVIC；EXTI 输出的事件，输入到 RxEV 模块，用于管理内核唤醒相关操作；EXTI 输出的唤醒信号，输出给 PWR 单元。

9.2.4 唤醒事件管理

器件可以处理外部或内部事件来唤醒内核（WFE）。唤醒事件可以通过下述配置产生：

- 在外设的控制寄存器而非 NVIC 中使能一个中断，同时在 Cortex®-M3 的系统控制寄存器中使能 SEVONPEND 位。当 CPU 从 WFE 恢复后，需要清除相应外设的中断挂起位和外设 NVIC 中断通道挂起位（在 NVIC 中断清除挂起寄存器中）。
- 配置一个外部或内部 EXTI 线为事件模式，当 CPU 从 WFE 恢复后，因为对应事件线的挂起位没有被置位，不必清除相应外设的中断挂起位或 NVIC 中断通道挂起位。

使用外部 I/O 端口作为唤醒事件，请参考“9.2.5 功能说明”。

9.2.5 功能说明

要产生中断，必须先配置并使能中断线。根据所需的边沿检测条件来配置 2 个触发寄存器，并且通过向中断屏蔽寄存器的相应位写’1’来使能中断请求。当外部中断线上发生了所设置的边沿时，将产生一个中断请求。该中断线对应的挂起位也随之被置’1’，向挂起寄存器的对应位写’1’，将清除该中断请求。

如果需要产生事件，必须先配置并使能事件线。根据所需的边沿检测条件来配置 2 个触发寄存器，并且通过对事件屏蔽寄存器的相应位写’1’来使能事件请求。当事件线上发生了需要的边沿时，将产生一个事件请求脉冲，对应的挂起位不被置’1’。

通过软件向软件中断/事件寄存器写’1’，可以产生中断/事件请求。

9.2.5.1 选择硬件中断

配置 EXTI 线作为中断源的步骤如下：

1. 配置其中断屏蔽位（EXTI_IMR）。

2. 配置其触发选择位（EXTI_RTSR 和 EXTI_FTSR）。

3. 配置对应到外部中断控制器（EXTI）的 NVIC 中断通道的使能和屏蔽位，以使得 EXTI 线上来的中断能正确地被响应。

9.2.5.2 选择硬件事件

配置 EXTI 线作为事件源的步骤如下：

1. 配置其事件屏蔽位（EXTI_EMR）。

2. 配置其触发选择位（EXTI_RTSR 和 EXTI_FTSR）。

9.2.5.3 选择软件中断/事件

EXTI 线可以被配置成软件中断/事件线。下面是产生软件中断的步骤：

1. 配置其中断/事件线的屏蔽位（EXTI_IMR / EXTI_EMR）。

2. 设置其软件中断寄存器的请求位（EXTI_SWIER）。

9.2.6 外部中断/事件线映射

80 个 GPIO 口以下图的方式连接到 16 个外部中断/事件线上：

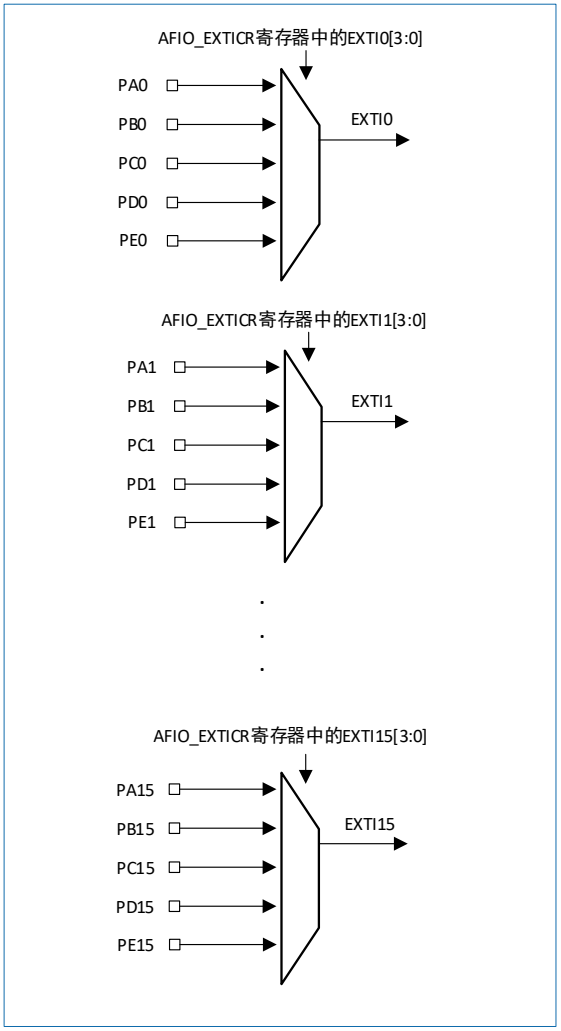


图 9-3 外部中断 GPIO 映射

上图说明：图 9-3 为 GPIO 与 EXTI 映射示意图。不同型号的芯片可用管脚数目可能存在区别。具体芯片各端口可用 IO，请查询对应芯片数据手册。

如图 9-3 所示，通过 AFIO_EXTICRx 配置 GPIO 线上的外部中断/事件，必须先使能 AFIO 时钟。参见："8.3.7 APB2 外设时钟使能寄存器（RCC_APB2ENR）"。

全部 EXTI 线的中断事件映射关系如表 9-2 所示：

表 9-2 EXTI 中断事件映射关系表

EXTI 序号	信号	EXTI 类型
0-15	GPIO	可配
16	PVD	可配
17	RTC/Wakeup timer	可配
18	USB	可配
20	COMP1	可配
21	COMP2	可配
22	COMP3	可配
23	COMP4	可配
24	ADC1 Wakeup	固定
25	ADC2 Wakeup	固定
26	ADC3 Wakeup	固定

其中 EXTI24~26 作为固定事件，不带 RTSR、FTSR、SWIER 和 PR 寄存器。这 3 个 EXTI 口仅能在停机（Stop）模式下采集事件的上升沿以产生 ERQ 和 IRQ 唤醒系统。

9.3 EXTI 寄存器

基地址：0x4001 0400

空间大小：0x400

EXTI 寄存器必须以字（32 位）的方式进行访问。

9.3.1 中断屏蔽寄存器（EXTI_IMR）

偏移地址：0x00

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res					MR26	MR25	MR24	MR23	MR22	MR21	MR20	Res	MR18	MR17	MR16
					rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:27	Res：保留 必须保持复位值。
位 x (x=26..20)	MRx：线 x 上的中断屏蔽（Interrupt Mask on line x） • 0：屏蔽来自线 x 上的中断请求

	• 1：允许来自线 x 上的中断请求
位 19	Res：保留 必须保持复位值。
位 x (x=18..0)	MRx：线 x 上的中断屏蔽（Interrupt Mask on line x） • 0：屏蔽来自线 x 上的中断请求 • 1：允许来自线 x 上的中断请求

9.3.2 事件屏蔽寄存器（EXTI_EMR）

偏移地址：0x04

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res					MR26	MR25	MR24	MR23	MR22	MR21	MR20	Res	MR18	MR17	MR16
					rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:27	Res：保留 必须保持复位值。
位 x (x=26..20)	MRx：线 x 上的事件屏蔽（Event mask on line x） • 0：屏蔽来自线 x 上的事件请求 • 1：允许来自线 x 上的事件请求
位 19	Res：保留 必须保持复位值。
位 x (x=18..0)	MRx：线 x 上的事件屏蔽（Event mask on line x） • 0：屏蔽来自线 x 上的事件请求 • 1：允许来自线 x 上的事件请求

9.3.3 上升沿触发选择寄存器（EXTI_RTSR）

偏移地址：0x08

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res								TR23	TR22	TR21	TR20	Res	TR18	TR17	TR16
								rw	rw	rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:24	Res：保留 必须保持复位值。
位 x (x=23..20)	TRx：线 x 上的上升沿触发事件配置位（Rising trigger event configuration bit of line x） • 0：禁止输入线 x 上的上升沿触发（中断和事件）

	• 1：允许输入线 x 上的上升沿触发（中断和事件） *注意：* • *外部唤醒线是边沿触发的，这些线上不能出现毛刺信号。* • *在写 EXTI_RTSR 寄存器时，不能检测外部中断线上的上升沿信号，挂起位也不会被置位。* • *可以在同一中断线上同时设置上升沿和下降沿触发，即任一边沿均可触发中断。*
位 19	Res：保留 必须保持复位值。
位 x (x=18..0)	TRx：线 x 上的上升沿触发事件配置位（Rising trigger event configuration bit of line x） • 0：禁止输入线 x 上的上升沿触发（中断和事件） • 1：允许输入线 x 上的上升沿触发（中断和事件） *注意：* • *外部唤醒线是边沿触发的，这些线上不能出现毛刺信号。* • *在写 EXTI_RTSR 寄存器时，不能检测外部中断线上的上升沿信号，挂起位也不会被置位。* 可以在同一中断线上同时设置上升沿和下降沿触发，即任一边沿均可触发中断。

9.3.4 下降沿触发选择寄存器（EXTI_FTSR）

偏移地址：0x0C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res								TR23	TR22	TR21	TR20	Res	TR18	TR17	TR16
								rw	rw	rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:24	Res：保留 必须保持复位值。
位 x (x=23..20)	TRx：线 x 上的下降沿触发事件配置位（Falling trigger event configuration bit of line x） • 0：禁止输入线 x 上的下降沿触发（中断和事件） • 1：允许输入线 x 上的下降沿触发（中断和事件） *注意：* • *外部唤醒线是边沿触发的，这些线上不能出现毛刺信号。* • *在写 EXTI_FTSR 寄存器时，不能检测外部中断线上的下降沿信号，挂起位也不会被置位。* • *可以在同一中断线上同时设置上升沿和下降沿触发，即任一边沿均可触发中断。*
位 19	Res：保留 必须保持复位值。
位 x (x=18..0)	TRx：线 x 上的下降沿触发事件配置位（Falling trigger event configuration bit of line x） • 0：禁止输入线 x 上的下降沿触发（中断和事件） • 1：允许输入线 x 上的下降沿触发（中断和事件） *注意：* • *外部唤醒线是边沿触发的，这些线上不能出现毛刺信号。* • *在写 EXTI_FTSR 寄存器时，不能检测外部中断线上的下降沿信号，挂起位也不会被置位。*

	可以在同一中断线上同时设置上升沿和下降沿触发，即任一边沿均可触发中断。

9.3.5 软件中断事件寄存器（EXTI_SWIER）

偏移地址：0x10

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res								SWIER23	SWIER22	SWIER21	SWIER20	Res	SWIER18	SWIER17	SWIER16
								rw	rw	rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWIER15	SWIER14	SWIER13	SWIER12	SWIER11	SWIER10	SWIER9	SWIER8	SWIER7	SWIER6	SWIER5	SWIER4	SWIER3	SWIER2	SWIER1	SWIER0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:24	Res：保留 必须保持复位值。
位 x (x=23..20)	SWIERx：线 x 上的软件中断（Software interrupt on line x） 当该位为'0'时，对该位写'1'将设置 EXTI_PR 中相应的挂起位。如果在 EXTI_IMR 中允许产生该中断，则此时将产生一个中断。 *说明：通过清除 EXTI_PR 的对应位（写入'1'），可清除该位为'0'。*
位 19	Res：保留 必须保持复位值。
位 x (x=18..0)	SWIERx：线 x 上的软件中断（Software interrupt on line x） 当该位为'0'时，对该位写'1'将设置 EXTI_PR 中相应的挂起位。如果在 EXTI_IMR 中允许产生该中断，则此时将产生一个中断。 *说明：通过清除 EXTI_PR 的对应位（写入'1'），可清除该位为'0'。*

9.3.6 挂起寄存器（EXTI_PR）

偏移地址：0x14

复位值：0xXXXX XXXX

说明：X 表示不定值。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res								PR23	PR22	PR21	PR20	Res	PR18	PR17	PR16
								rw	rw	rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:24	Res：保留 必须保持复位值。
位 x (x=23..20)	PRx：线 x 的挂起位（Pending interrupt flag on line x） • 0：没有发生触发请求 • 1：发生了所选择的触发请求 当在外部中断线上发生了所选择的边沿事件，则该位被置'1'。在该位中写入'1'可以清除它，也可以通过改变边沿检测的极性清除。

位 19	Res：保留 必须保持复位值。
位 x (x=18..0)	PRx：线 x 的挂起位（Pending interrupt flag on line x） • 0：没有发生触发请求 • 1：发生了所选择的触发请求 当在外部中断线上发生了所选择的边沿事件，则该位被置’1’。在该位中写入’1’可以清除它，也可以通过改变边沿检测的极性清除。

10 通用和复用功能 I/O（GPIO 和 AFIO）

10.1 GPIO 功能描述

每个 GPIO 端口有两个 32 位配置寄存器（GPIOx_CRL，GPIOx_CRH）、两个 32 位数据寄存器（GPIOx_IDR 和 GPIOx_ODR）、一个 32 位置位/复位寄存器（GPIOx_BSRR）、一个 16 位复位寄存器（GPIOx_BRR）、一个 32 位锁定寄存器（GPIOx_LCKR）、一个 32 位施密特触发配置寄存器（GPIOx_IOSEN）和一个 32 位超高驱动配置寄存器（GPIOx_UHD）。

根据数据手册中列出的每个 I/O 端口的特定硬件特征，GPIO 端口的每个位可以由软件分别配置成多种模式，包括：

- 输入浮空
- 输入上拉
- 输入下拉
- 模拟输入
- 开漏输出
- 推挽式输出
- 推挽式复用功能
- 开漏复用功能

I/O 端口的每个位均可自由编程，但必须按照 32 位，即以字为单位访问（不允许半字或字节访问）I/O 端口寄存器。GPIOx_BSRR 和 GPIOx_BRR 寄存器用于对任一个 GPIO 寄存器进行原子操作（按位读/写）。这样，在读和修改访问时产生的 IRQ 就不会有冲突。

每个 GPIO 引脚都可以由软件配置成输出（推挽或开漏）、输入（浮空输入、上拉输入或下拉输入）或其它的外设功能端口。多数 GPIO 引脚都与数字或模拟的外设共用。I/O 引脚的外设功能可以按需锁定，以避免意外的写入 I/O 寄存器。所有的 GPIO 引脚都有大电流通过能力。

图 10-1 给出了一个 I/O 端口位的基本结构。

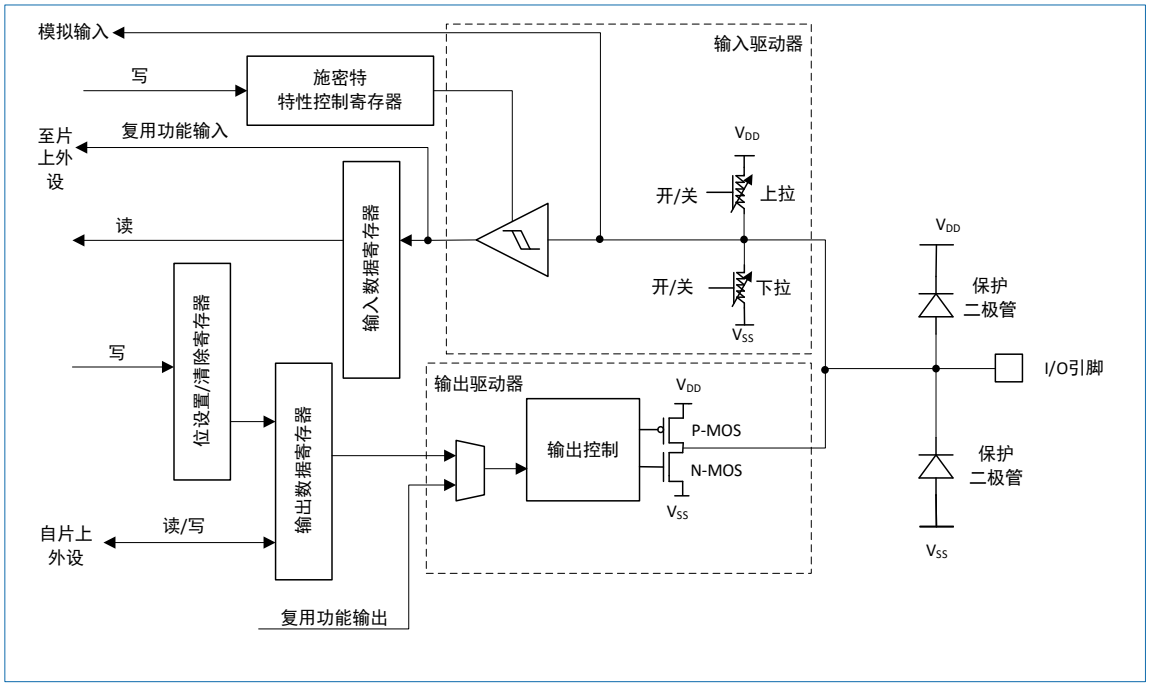


图 10-1 I/O 端口位的基本结构

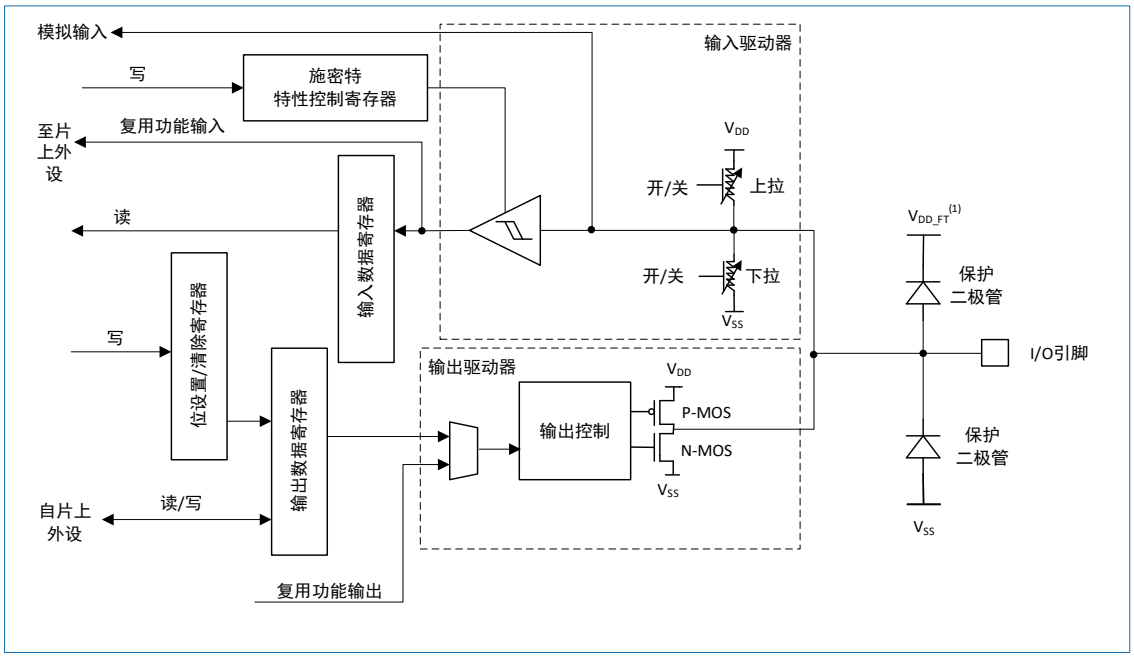


图 10-2 5V 兼容 I/O 端口位的基本结构

上图说明：

(1). V_{DD_FT}：5V 耐受的 I/O 端口，它与 V_{DD} 不同。

表 10-1 端口位配置表

配置模式		CNFx[1]	CNFx[0]	MODEx[1]	MODEx[0]	GPIOx_ODR 寄存器
通用输出	推挽（Push-Pull）	0	0	01 10 11 见表 10-2		0 或 1
	开漏（Open-Drain）		1			0 或 1
复用功能输出	推挽（Push-Pull）	1	0			不使用
	开漏（Open-Drain）		1			不使用
输入	模拟输入	0	0	00		不使用
	浮空输入		1			不使用
	下拉输入	1	0			0
	上拉输入					1

表 10-2 输出模式位

MODEx[1:0]	含义
00	保留
01	最大输出速度为 10 MHz
10	最大输出速度为 2 MHz
11	最大输出速度为 50 MHz

10.1.1 通用 I/O（GPIO）

复位期间和刚复位后，复用功能尚未开启，I/O 端口被配置成浮空输入模式（CNFx[1:0]=01b，

MODEx[1:0]=00b）。

复位后，JTAG 引脚被置于输入上拉或下拉模式：

- PA15：JTDI 置于上拉模式
- PA14：JTCK 置于下拉模式
- PA13：JTMS 置于上拉模式
- PB4：NJTRST 置于上拉模式

当配置为输出时，写到输出数据寄存器（GPIOx_ODR）上的值输出至相应的 I/O 引脚。可以在推挽模式或开漏模式下（当输出 0 时，只有 N-MOS 被打开）使用输出驱动器。

输入数据寄存器（GPIOx_IDR）在每个 APB2 时钟周期捕捉 I/O 引脚上的数据。

所有 GPIO 引脚有一个内部弱上拉和弱下拉功能，当配置为输入时，该功能可被打开或关闭。

10.1.2 单独的位设置或位清除

当对寄存器 GPIOx_ODR 以位进行编程时，软件不需要禁止中断：在单次 APB2 的原子写操作里，可以只更改一个或多个位。这是通过对“位设置/复位寄存器 GPIOx_BSRR”中对应位写‘1’来实现的。未被选择的位将不被更改。

10.1.3 外部中断/唤醒线

所有端口都有外部中断能力。为了使用外部中断线，端口必须配置成输入模式。更多的关于外部中断的信息，可参考“9.2 外部中断/事件控制器（EXTI）”和“9.2.4 唤醒事件管理”。

10.1.4 复用功能（AF）

使用默认复用功能前，必须对端口位配置寄存器编程。

- 对于复用的输入功能，端口必须配置成输入模式（浮空、上拉或下拉）且输入引脚必须由外部驱动。

注意：也可通过软件编程 GPIO 控制器来模拟复用功能输入引脚。此时，端口应当被设置为复用功能输出模式。显然，这时相应的引脚不再由外部驱动，而是通过 GPIO 控制器由软件来驱动。

- 对于复用输出功能，端口必须配置成复用功能输出模式（推挽或开漏）。
- 对于双向复用功能，端口位必须配置成复用功能输出模式（推挽或开漏）。这时，输入驱动器被配置成浮空输入模式。

如果把一个端口配置成复用输出功能，则该引脚和输出寄存器断开，并连接至片上外设的输出信号。

如果软件把一个 GPIO 脚配置成复用输出功能，但是外设没有启用，则它的输出将不确定。

10.1.5 软件重新映射 I/O 复用功能

为了使不同器件封装的外设 I/O 功能的数量达到最优，可以把一些复用功能重新映射到其他一些引脚上。这可通过软件配置相应的寄存器来完成。这时，复用功能就不再映射到它们的原始引脚上了。

10.1.6 GPIO 锁定机制

锁定机制允许冻结 IO 配置。当在一个端口位上执行了锁定（LOCK）程序，在下一次复位之前，将不能更改端口位的配置。

10.1.7 输入配置

当 I/O 端口配置为输入时：

- 输出缓冲器被禁止
- 施密特触发输入默认关闭
- 根据输入配置（上拉、下拉或浮空），启用或不启用弱上拉和弱下拉电阻
- 出现在 I/O 脚上的数据在每个 APB2 时钟被采样到输入数据寄存器
- 对输入数据寄存器的读访问可得到 I/O 状态

图 10-3 给出了 I/O 端口位的输入配置。

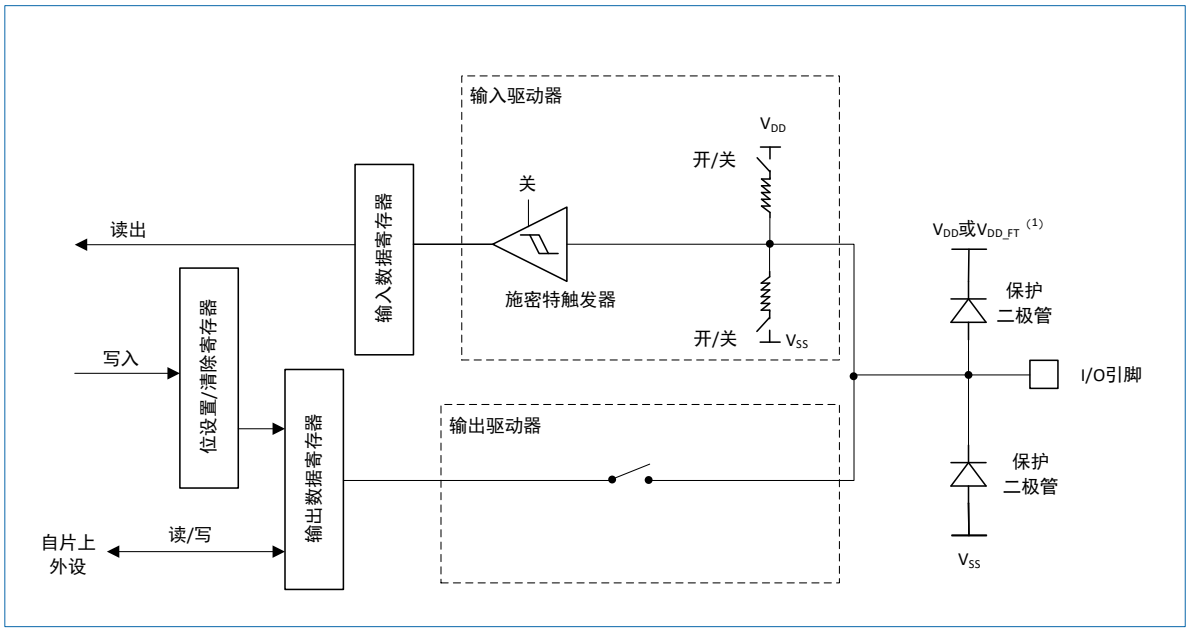


图 10-3 输入浮空/上拉/下拉配置

(1). V_{DD_FT}：5V 耐受的 I/O 端口，它与 V_{DD} 不同。

10.1.8 输出配置

当 I/O 端口被配置为输出时：

- 输出缓冲器被使能
  - 开漏模式：输出寄存器上的’0’激活 N-MOS，而输出寄存器上的’1’将端口置于高阻状态（P-MOS 一直未被激活）。
  - 推挽模式：输出寄存器上的’0’激活 N-MOS，而输出寄存器上的’1’将激活 P-MOS。
- 弱上拉和下拉电阻被禁用。
- 出现在 I/O 脚上的数据在每个 APB2 时钟被采样到输入数据寄存器。
- 在开漏模式时，对输入数据寄存器的读访问可得到 I/O 状态。
- 在推挽式模式时，对输出数据寄存器的读访问得到最后一次写的值。

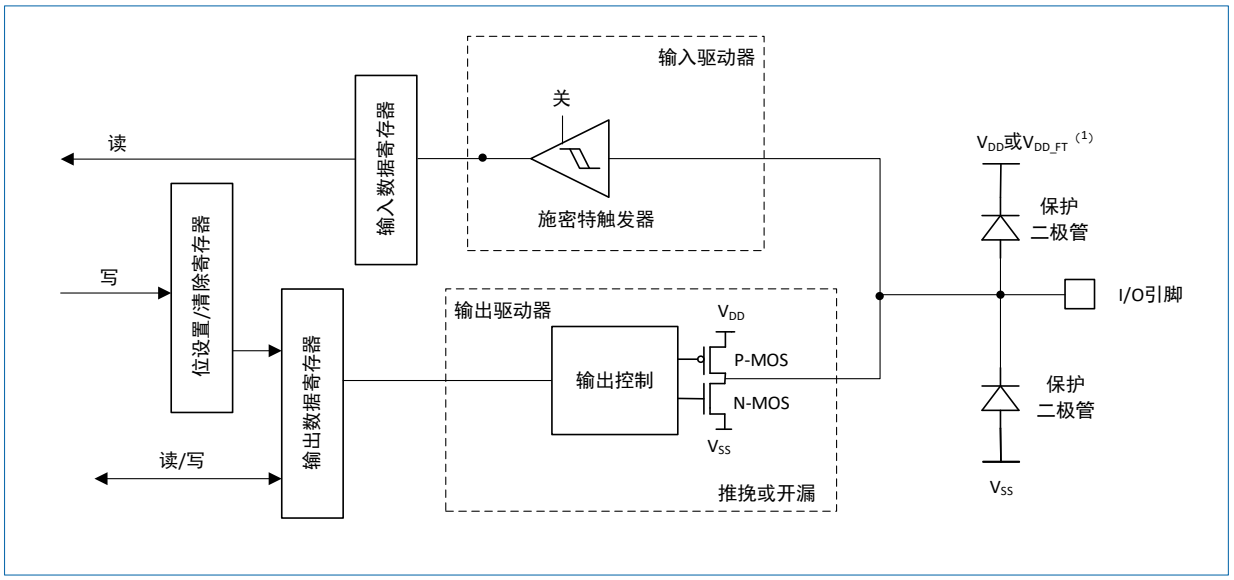


图 10-4 输出配置

（1）V_{DD_FT}：5V 耐受的 I/O 端口，它与 V_{DD} 不同。

10.1.9 复用功能配置

当 I/O 端口被配置为复用功能时：

- 在开漏或推挽式配置中，输出缓冲器被打开。
- 由来自片内外设的信号驱动输出缓冲器（复用功能输出）。
- 施密特触发输入默认关闭。
- 弱上拉和弱下拉电阻被禁用。
- 在每个 APB2 时钟周期，出现在 I/O 脚上的数据被采样到输入数据寄存器。
- 开漏模式时，读输入数据寄存器时可得到 I/O 口状态。
- 在推挽模式时，读输出数据寄存器时可得到最后一次写的值。

图 10-5 示出了 I/O 端口位的复用功能配置。详见“10.3 复用功能 I/O 和调试配置（AFIO）”。

一组复用功能 I/O 寄存器允许用户把一些复用功能重新映射到不同的引脚。

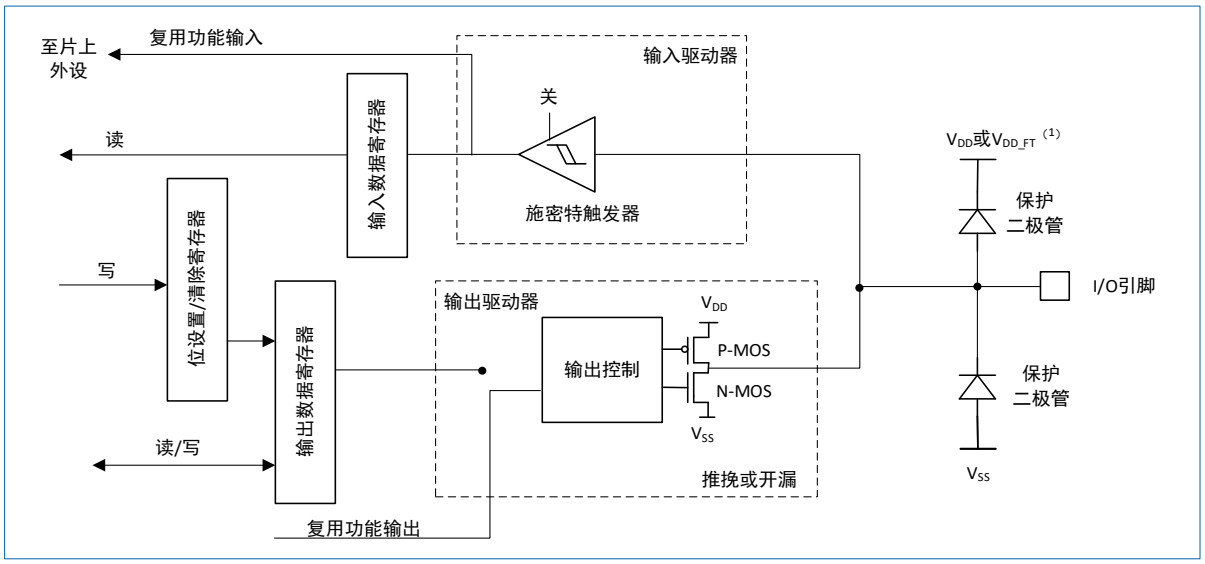


图 10-5 复用功能配置

（1）V_{DD_FT}：5V 耐受的 I/O 端口，它与 V_{DD} 不同。

10.1.10 模拟输入配置

当 I/O 端口被配置为模拟输入配置时：

- 输出缓冲器被禁用。
- 禁止施密特触发输入，实现了每个模拟 I/O 引脚上的零消耗。施密特触发输出值被强置为’0’。
- 弱上拉和弱下拉电阻被禁用。
- 读取输入数据寄存器的数值为’0’。

图 10-6 示出了 I/O 端口位的高阻抗模拟输入配置：

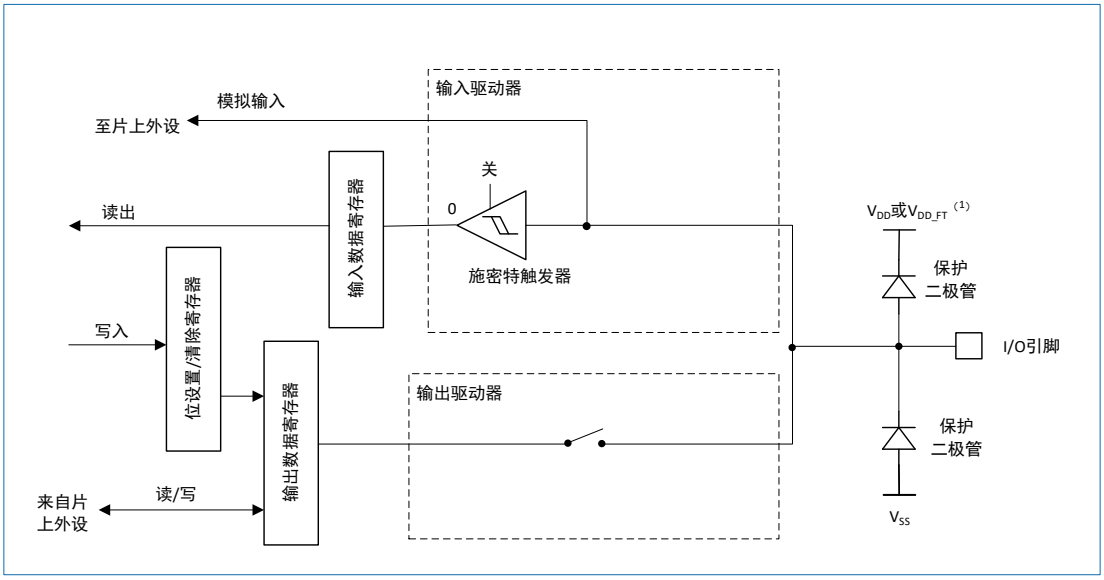


图 10-6 高阻抗的模拟输入配置

（1）V_{DD_FT}：5V 耐受的 I/O 端口，它与 V_{DD} 不同。

10.1.11 外设的 GPIO 配置

表 10-3 至表 10-13 列出了各个外设的 GPIO 配置。

表 10-3 高级定时器 TIM1/8

TIM1 引脚	配置	GPIO 配置
TIM1/8_CHx	输入捕获通道 x	浮空输入
	输出比较通道 x	推挽复用输出
TIM1/8_CHxN	互补输出通道 x	推挽复用输出
TIM1/8_BKIN	刹车输入	浮空输入
TIM1/8_ETR	外部触发时钟输入	浮空输入

表 10-4 通用定时器 TIM2/3/4/5

TIM2/3/4/5 引脚	配置	GPIO 配置
TIM2/3/4/5_CHx	输入捕获通道 x	浮空输入
	输出比较通道 x	推挽复用输出

TIM2/3/4/5 引脚	配置	GPIO 配置
TIM2/3/4/5_ETR	外部触发时钟输入	浮空输入

表 10-5 USART

USART 引脚	配置	GPIO 配置
USARTx_TX (1)	全双工模式	推挽复用输出
	半双工同步模式	推挽复用输出
USARTx_RX	全双工模式	浮空输入或带上拉输入
	半双工同步模式	未用，可作为通用 I/O
USARTx_CK	同步模式	推挽复用输出
USARTx_RTS	硬件流量控制	推挽复用输出
USARTx_CTS	硬件流量控制	浮空输入或带上拉输入

(1). USART_TX 也可以配置为复用开漏模式。

表 10-6 SPI

SPI 引脚	配置	GPIO 配置
SPIx_SCK	主模式	推挽复用输出
	从模式	浮空输入
SPIx_MOSI	全双工模式/主模式	推挽复用输出
	全双工模式/从模式	浮空输入或带上拉输入
	单线半双工数据线/主模式	推挽复用输出
	单线半双工数据线/从模式	未用，可作为通用 I/O
SPIx_MISO	全双工模式/主模式	浮空输入或带上拉输入
	全双工模式/从模式	推挽复用输出
	单线半双工数据线/主模式	未用，可作为通用 I/O
	单线半双工数据线/从模式	推挽复用输出
SPIx_NSS	硬件主/从模式	浮空输入、带上拉输入或带下拉输入
	硬件主模式/NSS 输出使能	推挽复用输出
	软件模式	未用，可作为通用 I/O

表 10-7 I2S 接口

I2S 引脚	配置	GPIO 配置
I2Sx_WS	主	推挽复用

I2S 引脚	配置	GPIO 配置
	从	浮空输入
I2Sx_CK	主	推挽复用
	从	浮空输入
I2Sx_SD	发送	推挽复用
	接收	浮空输入、带上拉输入或带下拉输入
I2Sx_MCK	主	推挽复用
	从	未用，可作为通用 I/O

表 10-8 I2C 接口

I2C 引脚	配置	GPIO 配置
I2Cx_SCL	I2C 时钟	开漏复用
I2Cx_SDA	I2C 数据	开漏复用

表 10-9 CAN

CAN 引脚	GPIO 配置
CAN_TX	推挽复用输出
CAN_RX	浮空输入或上拉输入

表 10-10 USB

USB 引脚	GPIO 配置
USB_DM/USB_DP	一旦使能了 USB 模块，这些引脚会自动连接到内部 USB 收发器

表 10-11 SDIO

SDIO 引脚	GPIO 配置
SDIO_CK	推挽复用输出
SDIO_CMD	推挽复用输出
SDIO[D7:D0]	推挽复用输出

表 10-12 ADC/DAC

ADC/DAC 引脚	GPIO 配置
ADC/DAC (1)	模拟输入

(1). ADC 输入引脚必须配置为模拟输入。

表 10-13 其它 I/O 功能

引脚	复用功能	GPIO 配置
TAMPER-RTC	RTC 输出	当配置 BKP_CR 和 BKP_RTCCR 寄存器时，由硬件强制设置
	侵入事件输入	
MCO	时钟输出	推挽复用输出
EXTI 输入线	外部中断输入	浮空输入、上拉输入或下拉输入

10.2 GPIO 寄存器

基地址：（GPIOA; GPIOB, GPIOC, GPIOD, GPIOE）=（0x4001 0800; 0x4001 0C00, 0x4001 1000, 0x4001 1400, 0x4001 1800）

空间大小：（GPIOA; GPIOB, GPIOC, GPIOD, GPIOE）=（0x400; 0x400, 0x400, 0x400, 0x400）

GPIO 寄存器必须以字（32 位）的方式进行访问。

10.2.1 端口 x 配置低寄存器（GPIOx_CRL）（x=A..E）

偏移地址：0x00

复位值：0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF7[1:0]		MODE7[1:0]		CNF6[1:0]		MODE6[1:0]		CNF5[1:0]		MODE5[1:0]		CNF4[1:0]		MODE4[1:0]	
rw		rw		rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF3[1:0]		MODE3[1:0]		CNF2[1:0]		MODE2[1:0]		CNF1[1:0]		MODE1[1:0]		CNF0[1:0]		MODE0[1:0]	
rw		rw		rw		rw		rw		rw		rw		rw	

位 4y+3:4y+2
(y=7..0)
	CNFy[1:0]：端口 x 配置位 y（Port x configuration bits）
	软件通过这些位配置相应的 I/O 端口，请参考表 10-1。
	• 在输入模式（MODEy[1:0]=00）：
	  ○ 00：模拟输入模式
	  ○ 01：浮空输入模式（复位后的状态）
	  ○ 10：上拉/下拉输入模式
	  ○ 11：保留
	• 在输出模式（MODEy[1:0]>00）：
	  ○ 00：通用推挽输出模式
	  ○ 01：通用开漏输出模式
	  ○ 10：复用功能推挽输出模式
	  ○ 11：复用功能开漏输出模式

位 4y+1:4y+0 (y=7..0)	MODEy[1:0]：端口 x 的 y 引脚工作模式配置位（Port x pin y mode configuration bits） 软件通过这些位配置相应的 I/O 端口，请参考表 10-1。 • 00：输入模式（复位后的状态） • 01：输出模式，最大速度 10 MHz • 10：输出模式，最大速度 2 MHz • 11：输出模式，最大速度 50 MHz

10.2.2 端口 x 配置高寄存器（GPIOx_CRH）（x=A..E）

偏移地址：0x04

复位值：0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF15[1:0]		MODE15[1:0]		CNF14[1:0]		MODE14[1:0]		CNF13[1:0]		MODE13[1:0]		CNF12[1:0]		MODE12[1:0]	
rw		rw		rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF11[1:0]		MODE11[1:0]		CNF10[1:0]		MODE10[1:0]		CNF9[1:0]		MODE9[1:0]		CNF8[1:0]		MODE8[1:0]	
rw		rw		rw		rw		rw		rw		rw		rw	

位 4(y-8)+3:4(y-8)+2 (y=15..8)	CNFy[1:0]：端口 x 配置位 y（Port x configuration bits） 软件通过这些位配置相应的 I/O 端口，请参考表 10-1。 • 在输入模式（MODEy[1:0]=00）： ○ 00：模拟输入模式 ○ 01：浮空输入模式（复位后的状态） ○ 10：上拉/下拉输入模式 ○ 11：保留 • 在输出模式（MODEy[1:0]>00）： ○ 00：通用推挽输出模式 ○ 01：通用开漏输出模式 ○ 10：复用功能推挽输出模式 ○ 11：复用功能开漏输出模式
位 4(y-8)+1:4(y-8) (y=15..8)	MODEy[1:0]：端口 x 的 y 引脚工作模式配置位（Port x pin y mode configuration bits） 软件通过这些位配置相应的 I/O 端口，请参考表 10-1。 • 00：输入模式（复位后的状态） • 01：输出模式，最大速度 10 MHz • 10：输出模式，最大速度 2 MHz • 11：输出模式，最大速度 50 MHz

10.2.3 端口 x 输入数据寄存器（GPIOx_IDR）（x=A..E）

偏移地址：0x08

复位值：0x0000 XXXX

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31:16	Res：保留 必须保持复位值。
位 y (y=15..0)	IDRy：端口输入数据（Port input data） 这些位为只读并只能以字的方式读出。读出的值为对应 I/O 口的状态。

10.2.4 端口 x 输出数据寄存器（GPIOx_ODR）（x=A..E）

偏移地址：0x0C

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:16	Res：保留 必须保持复位值。
位 y (y=15..0)	ODRy：端口输出数据（Port output data） 这些位可读可写并只能以字的方式操作。 *说明：通过 GPIOx_BSRR（x=A..E），可以分别地对各个 ODR 位进行独立的设置/清除。*

10.2.5 端口 x 位设置/清除寄存器（GPIOx_BSRR）（x=A..E）

偏移地址：0x10

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

位 y+16 (y=15..0)	BRy：清除端口 x 的位 y（Port x Reset bit y） 这些位只能写入并只能以字的方式操作。 • 0：对对应的 ODRy 位不产生影响 • 1：清除对应的 ODRy 位为 0 *说明：如果同时设置了 BSy 和 BRy 的对应位，BSy 位起作用。*
位 y (y=15..0)	BSy：设置端口 x 的位 y（Port x Reset bit y） 这些位只能写入并只能以字的方式操作。 • 0：对对应的 ODRy 位不产生影响 • 1：设置对应的 ODRy 位为 1

10.2.6 端口 x 位清除寄存器（GPIOx_BRR）（x=A..E）

偏移地址：0x14

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

位 31:16	Res：保留 必须保持复位值。

位 y (y=15..0)	BRy：清除端口 x 的位 y（Port x Reset bit y） 这些位只能写入并只能以字（16 位）的方式操作。 • 0：不影响对应的 ODRy 位 • 1：清除对应的 ODRy 位为 0

10.2.7 端口 x 配置锁定寄存器（GPIOx_LCKR）（x=A..E）

偏移地址：0x18

复位值：0x0000 0000

当执行正确的写序列设置了位 16（LCKK）时，该寄存器用来锁定端口位的配置。位[15:0]用于锁定 GPIO 端口的配置。在规定的写入操作期间，不能改变 LCKR[15:0]。当对相应的端口位执行了 LOCK 序列后，在下次系统复位之前不能再更改端口位的配置。

每个锁定位锁定控制寄存器（CRL，CRH）中相应的 4 个位。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:17	Res：保留 必须保持复位值。
位 16	LCKK：锁键（Lock key） 该位可随时读出，它只可通过锁键写入序列修改。 • 0：端口配置锁键位激活 • 1：端口配置锁键位被激活，下次系统复位前 GPIOx_LCKR 寄存器被锁住。 锁键的写序列： 1. 写 1 2. 写 0 3. 写 1 4. 读 0 5. 读 1（该步可省略，但该步可用于确认锁键已被激活） *说明：在操作锁键的写入序列时，不能改变 LCK[15:0]的值。* 操作锁键写入序列中的任何错误将不能激活锁键。
位 y (y=15..0)	LCKy：端口 x 的锁位 y（Port x Lock bit y） 这些位可读可写，但只能在 LCKK 位为 0 时写入。 • 0：不锁定端口的配置 • 1：锁定端口的配置

10.2.8 端口 x 施密特触发配置寄存器（GPIOx_IOSEN）（x=A..E）

偏移地址：0x30

复位值：0x0000 FFFF

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IOSEN15	IOSEN14	IOSEN13	IOSEN12	IOSEN11	IOSEN10	IOSEN9	IOSEN8	IOSEN7	IOSEN6	IOSEN5	IOSEN4	IOSEN3	IOSEN2	IOSEN1	IOSEN0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:16	Res：保留 必须保持复位值。
位 y （y=15..0）	IOSENy：端口 x 的 y 引脚的（y = 15..0）施密特特性开关（Port x pin y Schmitt switch） • 0：使能 IO 施密特功能 • 1：禁能 IO 施密特功能

10.2.9 端口 x 超高驱动配置寄存器（GPIOx_UHD）（x=A..E）

偏移地址：0x34

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UHD15	UHD14	UHD13	UHD12	UHD11	UHD10	UHD9	UHD8	UHD7	UHD6	UHD5	UHD4	UHD3	UHD2	UHD1	UHD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:16	Res：保留 必须保持复位值。
位 y （y=15..0）	UHDy：端口 x 的 y 引脚的（y = 15..0）超高驱动能力开关（Port x pin y Ultra high drive switch） • 0：禁能超高驱动能力 • 1：使能超高驱动能力

10.3 复用功能 I/O 和调试配置（AFIO）

由于引脚数量受封装限制，可通过把某些复用功能重新映射到其他引脚，以增加可用的外设数量。设置复用重映射和调试 I/O 配置寄存器（AFIO_MAPR）实现引脚的重新映射。重映射后，复用功能不再映射到它们原来所指定的引脚。

10.3.1 把 OSC32_IN/OSC32_OUT 作为 GPIO 端口 PC14/PC15

当 LSE 振荡器关闭时，LSE 振荡器引脚 OSC32_IN/OSC32_OUT 可以分别用做 GPIO 的 PC14/PC15，LSE 功能始终优先于通用 I/O 的功能。

说明：当关闭 Core 电压域（进入待机模式）或备份域使用 V_{BAT} 供电（不再有 V_{DD} 供电）时，不能使用 PC14/PC15 的 GPIO 口功能。参考“6.1.2 电池备份域”。

10.3.2 把 OSC_IN/OSC_OUT 引脚作为 GPIO 端口 PD0/PD1

外部振荡器引脚 OSC_IN/OSC_OUT 可以用做 GPIO 的 PD0/PD1，通过设置复用重映射和调试 I/O 配置寄存器（AFIO_MAPR）实现。

说明：外部中断/事件功能没有被重映射。

10.3.3 CAN1 复用功能重映射

CAN1 的外部信号可以按下表重新映射：

表 10-14 CAN1 复用功能重映射

复用功能	CAN1_REMAP[1:0]=”00”	CAN1_REMAP[1:0]=”10”	CAN1_REMAP[1:0]=”11”
CAN1_RX	PA11	PB8	PD0
CAN1_TX	PA12	PB9	PD1

10.3.4 CAN2 复用功能重映射

CAN2 的外部信号可以按下表重新映射：

表 10-15 CAN2 复用功能重映射

复用功能	CAN2_REMAP=”00/11”	CAN2_REMAP=”01”	CAN2_REMAP=”10”
CAN2_RX	PB12	PB5	PB0
CAN2_TX	PB13	PB6	PB1

10.3.5 JTAG/SWD 复用功能重映射

调试接口信号被映射到 GPIO 端口上，如表 10-16 所示。

表 10-16 调试接口信号

复用功能	GPIO 端口
JTMS/SWDIO	PA13
JTCK/SWCLK	PA14
JTDI	PA15
JTDO/TRACESWO	PB3
NJTRST	PB4
TRACECKO	PE2
TRACEDO0	PE3
TRACEDO1	PE4
TRACEDO2	PE5
TRACEDO3	PE6

为了在调试期间可以使用更多 GPIOs，通过设置复用重映射和调试 I/O 配置寄存器（AFIO_MAPR）的 SWJ_CFG[2:0]位，可以改变上述重映射配置。参见表 10-17。

表 10-17 调试端口映射

SWJ_CFG[2:0]	可用的调试端口	SWJ I/O 引脚分配				
		PA13/JTMS /SWDIO	PA14/JTCK/ SWCLK	PA15/JTDI	PB3/JTDO/ TRACESWO	PB4/ NJTRST
000	完全 SWJ（JTAG-DP+SW-DP）（复位状态）	I/O 不可用	I/O 不可用	I/O 不可用	I/O 不可用	I/O 不可用
001	完全 SWJ（JTAG-DP+SW-DP）但没有 NJTRST	I/O 不可用	I/O 不可用	I/O 不可用	I/O 不可用	I/O 可用
010	关闭 JTAG-DP，启用 SW-DP	I/O 不可用	I/O 不可用	I/O 可用	I/O 可用(1)	I/O 可用
100	关闭 JTAG-DP，关闭 SW-DP	I/O 可用	I/O 可用	I/O 可用	I/O 可用	I/O 可用
其它	禁用	-	-	-	-	-

(1). I/O 口仅可在不使用异步跟踪时使用。

10.3.6 ADC 复用功能重映射

表 10-18 ADC1 注入转换外部触发重映射

复用功能	ADC1_ETRGINJ_REMAP = 0	ADC1_ETRGINJ_REMAP = 1
ADC1 注入转换外部触发	ADC1 注入转换外部触发连到 EXTI15	ADC1 注入转换外部触发连到 TIM8_CC4

表 10-19 DC1 规则转换外部触发重映射

复用功能	ADC1_ETRGREG_REMAP = 0	ADC1_ETRGREG_REMAP = 1
ADC1 规则转换外部触发	ADC1 规则转换外部触发连到 EXTI11	ADC1 规则转换外部触发连到 TIM8_TRGO

表 10-20 DC2 注入转换外部触发重映射

复用功能	ADC2_ETRGINJ_REMAP = 0	ADC2_ETRGINJ_REMAP = 1
ADC2 注入转换外部触发	ADC2 注入转换外部触发连到 EXTI15	ADC2 注入转换外部触发连到 TIM8_CC4

表 10-21 DC2 规则转换外部触发重映射

复用功能	ADC2_ETRGREG_REMAP = 0	ADC2_ETRGREG_REMAP = 1
ADC2 规则转换外部触发	ADC2 规则转换外部触发连到 EXTI11	ADC2 规则转换外部触发连到 TIM8_TRGO

10.3.7 定时器复用功能重映射

定时器 TIM4 的通道 1 至通道 4 可以从端口 B 重映射到至端口 D。

表 10-22 IM4 复用功能重映射

复用功能	TIM4_REMAP=0	TIM4_REMAP=1
TIM4_CH1	PB6	PD12
TIM4_CH2	PB7	PD13
TIM4_CH3	PB8	PD14

复用功能	TIM4_REMAP=0	TIM4_REMAP=1
TIM4_CH4	PB9	PD15

表 10-23 TIM5 复用功能重映射

复用功能	TIM5CH4_IREMAP3=0 时，由｛TIM5CH4_IREMAP2,TIM5CH4_IREMAP}决定复用功能				TIM5CH4_IREMAP3=1
	｛TIM5CH4_IREMAP2,TIM5CH4_IREMAP}=				
	2'b00	2'b01	2'b10	2'b11	
TIM5_CH4	GPIO 作为 TIM5 的 CH4 输入源	LSI 作为 TIM5 的 CH4 输入源	分频后的 HSI 作为 TIM5 的 CH4 输入源	LSE 作为 TIM5 的 CH4 输入源	USB2.0 接收到的 SOF 信号作为 TIM5 的 CH4 输入源。

表 10-24 TIM3 复用功能重映射

复用功能	TIM3_REMAP[1:0]=00（没有重映射）	TIM3_REMAP[1:0]=10（部分重映射）	TIM3_REMAP[1:0]=11（完全重映射）(1)
TIM3_CH1	PA6	PB4	PC6
TIM3_CH2	PA7	PB5	PC7
TIM3_CH3	PB0	PB0	PC8
TIM3_CH4	PB1	PB1	PC9

表 10-25 TIM2 复用功能重映射

复用功能	TIM2_REMAP[1:0]=00（没有重映射）	TIM2_REMAP[1:0]=01（部分重映射）	TIM2_REMAP[1:0]=10（部分重映射）(1)	TIM2_REMAP[1:0]=11（完全重映射）
TIM2_CH1_ETR (1)	PA0	PA15	PA0	PA15
TIM2_CH2	PA1	PB3	PA1	PB3
TIM2_CH3	PA2		PB10	
TIM2_CH4	PA3		PB11	

(1). TIM2_CH1 和 TIM2_ETR 共用一个引脚，但不能同时使用（因此使用约定的标记：TIM2_CH1_ETR）。

表 10-26 TIM1 复用功能重映射

复用功能映射	TIM1_REMAP[1:0]=00（没有重映射）	TIM1_REMAP[1:0]=01（部分重映射）	TIM1_REMAP[1:0]=11（完全重映射）
TIM1_ETR	PA12	PA12	PE7
TIM1_CH1	PA8	PA8	PE9
TIM1_CH2	PA9	PA9	PE11
TIM1_CH3	PA10	PA10	PE13
TIM1_CH4	PA11	PA11	PE14

复用功能映射	TIM1_REMAP[1:0]=00 （没有重映射）	TIM1_REMAP[1:0]=01 （部分重映射）	TIM1_REMAP[1:0]=11 （完全重映射）
TIM1_BKIN	PB12	PA6	PE15
TIM1_CH1N	PB13	PA7	PE8
TIM1_CH2N	PB14	PB0	PE10
TIM1_CH3N	PB15	PB1	PE12

10.3.8 USART 复用功能重映射

参见复用重映射和调试 I/O 配置寄存器（AFIO_MAPR）。

表 10-27 USART5 重映射

复用功能	USART5_REMAP=0	USART5_REMAP=1
USART5_TX	PC12	PC8
USART5_RX	PD2	PC9

表 10-28 USART4 重映射

复用功能	USART4_REMAP=0	USART4_REMAP=1
USART4_TX	PC10	PB8
USART4_RX	PC11	PB9

表 10-29 USART3 重映射

复用功能	USART3_REMAP[1:0]=00 （没有重映射）	USART3_REMAP[1:0]=01 （部分重映射）(1)	USART3_REMAP[1:0]=11 （完全重映射）(2)
USART3_TX	PB10	PC10	PD8
USART3_RX	PB11	PC11	PD9
USART3_CK	PB12	PC12	PD10
USART3_CTS	PB13	PB13	PD11
USART3_RTS	PB14	PB14	PD12

表 10-30 USART2 重映射

复用功能	USART2_REMAP=0 或 USART2_REMAP2=0	USART2_REMAP=1	USART2_REMAP2=1(1)
USART2_CTS	PA0	PD3	PA14
USART2_RTS	PA1	PD4	PA15
USART2_TX	PA2	PD5	PC10
USART2_RX	PA3	PD6	PC11
USART2_CK	PA4	PD7	PC12

(1). 说明：AFIO_MAPR2.USART2_REMAP2 功能优先级高于 AFIO_MAPR.USART2_REMAP。

表 10-31 USART1 重映射

复用功能	USART1_REMAP=0	USART1_REMAP=1	USART1_CTSRTS_REMAP=0	USART1_CTSRTS_REMAP=1
USART1_TX	PA9	PB6	-	-
USART1_RX	PA10	PB7	-	-
USART1_CTS	-	-	PA11	PC8
USART1_RTS	-	-	PA12	PC9

10.3.9 I2C1 复用功能重映射

参见复用重映射和调试 I/O 配置寄存器（AFIO_MAPR）。

表 10-32 I2C1 重映射

复用功能	I2C1_REMAP=0	I2C1_REMAP=1	I2C1_SMBA_REMAP=0	I2C1_SMBA_REMAP=1
I2C1_SCL	PB6	PB8	-	-
I2C1_SDA	PB7	PB9	-	-
I2C1_SMBA	-	-	PB5	PA14

10.3.10 SPI/I2S 复用功能重映射

参见复用重映射和调试 I/O 配置寄存器（AFIO_MAPR）。

表 10-33 SPI3/I2S3 重映射

复用功能	SPI3_REMAP=0	SPI3_REMAP=1
SPI3_NSS/I2S3_WS	PA15	PD3
SPI3_SCK/I2S3_CK	PB3	PD4
SPI3_MISO	PB4	PD5
SPI3_MOSI/I2S3_SD	PB5	PD6

表 10-34 SPI1/I2S1 重映射

复用功能	SPI1_REMAP=0	SPI1_REMAP=1
SPI1_NSS/I2S1_WS	PA4	PA15
SPI1_SCK/I2S1_CK	PA5	PB3
SPI1_MISO	PA6	PB4
SPI1_MOSI/I2S1_SD	PA7	PB5
I2S1_MCK	PC4	PB6

10.3.11 SAI 复用功能重映射

表 10-35 SAI 重映射

复用功能	SAI_AB_REMAP =00 或 11	SAI_AB_REMAP =01	SAI_AB_REMAP =01
SAI_MCLK_B	PA5	PB4	PE10
SAI_FS_B	PA4	PB6	PE9
SAI_SCK_B	PA6	PB3	PE8
SAI_SD_B	PA7	PB5	PE7
SAI_MCLK_A	PA3	PB8	PE2
SAI_FS_A	PA9	PB9	PE4
SAI_SCK_A	PA8	PB10	PE5
SAI_SD_A	PA10	PB2	PE6

10.3.12 DCMI 复用功能重映射

表 10-36 DCMI 重映射

复用功能	DCMI_REMAP=0	DCMI_REMAP=1
DCMI_VYSNC	PA11	PA13
DCMI_HYSNC	PA12	PA14
DCMI_PIXDI10	PA0	PB6
DCMI_PIXDI11	PC3	PB5
DCMI_PIXDI12	PC2	PB4
DCMI_PIXDI13	PC1	PB3

10.4 AFIO 寄存器

基地址：0x4001 0000

空间大小：0x400

AFIO 寄存器必须以字（32 位）的方式进行访问。

注意：对寄存器 AFIO_EVCR、AFIO_MAPR 和 AFIO_EXTICRX 进行读写操作前，应当首先打开 AFIO 的时钟。参考 APB2 外设时钟使能寄存器（RCC_APB2ENR）。

10.4.1 事件控制寄存器（AFIO_EVCR）

偏移地址：0x00

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								EVOE	PORT[2:0]			PIN[3:0]			
								rw	rw			rw			

位 31:8	Res：保留 必须保持复位值。
位 7	EVOE：允许事件输出（Event output enable） 该位可由软件设置和清零。 当设置该位后，Cortex 的 EVENTOUT 将连接到由 PORT[2:0]和 PIN[3:0]所选择的 I/O 口。
位 6:4	PORT[2:0]：端口选择（Port selection） 选择用于输出 Cortex 的 EVENTOUT 信号的端口： • 000：选择 PA • 001：选择 PB • 010：选择 PC • 011：选择 PD • 100：选择 PE 该位可由软件设置和清零。
位 3:0	PIN[3:0]：引脚选择（Pin selection, x=A..E） 选择用于输出 Cortex 的 EVENTOUT 信号的引脚： • 0000：选择 Px0 • 0001：选择 Px1 • 0010：选择 Px2 • 0011：选择 Px3 • 0100：选择 Px4 • 0101：选择 Px5 • 0110：选择 Px6 • 0111：选择 Px7 • 1000：选择 Px8 • 1001：选择 Px9 • 1010：选择 Px10 • 1011：选择 Px11 • 1100：选择 Px12 • 1101：选择 Px13 • 1110：选择 Px14 • 1111：选择 Px15

10.4.2 复用重映射和调试 I/O 配置寄存器（AFIO_MAPR）

偏移地址：0x04

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res					SWJ_CFG[2:0]			Res			ADC2_ETRGREG_REMAP	ADC2_ETRGINJ_REMAP	ADC1_ETRGREG_REMAP	ADC1_ETRGINJ_REMAP	TIM5CH4_IREMAP
					w						rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD01_REMAP	CAN1_REMAP[1:0]		TIM4_REMAP	TIM3_REMAP[1:0]		TIM2_REMAP[1:0]		TIM1_REMAP[1:0]		USART3_REMAP[1:0]		USART2_REMAP	USART1_REMAP	I2C1_REMAP	SPI1_REMAP

rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:27	Res：保留 必须保持复位值。
位 26:24	SWJ_CFG[2:0]：串行线 JTAG 配置（Serial wire JTAG configuration） 这些位只可由软件写（读这些位，将返回未定义的数值），用于配置 SWJ 和跟踪复用功能的 I/O 口。SWJ（串行线 JTAG）支持 JTAG 或 SWD 访问 Cortex 的调试端口。系统复位后的默认状态是启用 SWJ 但不带跟踪功能。这种状态下可以通过 JTMS/JTCK 脚发送的特定序列选择 JTAG 或 SW（串行线）模式。 • 000：完全 SWJ（JTAG-DP+SW-DP），复位状态 • 001：完全 SWJ（JTAG-DP+SW-DP），但没有 NJTRST • 010：关闭 JTAG-DP，启用 SW-DP • 100：关闭 JTAG-DP，关闭 SW-DP • 其它值：无作用
位 23:21	Res：保留 必须保持复位值。
位 20	ADC2_ETRGREG_REMAP：ADC2 规则转换外部触发重映射（ADC2 external trigger regular conversion remapping） 该位可由软件置’1’或清零。 该位控制与 ADC2 规则转换外部触发相连的触发输入。 • 当该位置’0’时，ADC2 规则转换外部触发与 EXTI11 相连。 • 当该位置’1’时，ADC2 规则转换外部触发与 TIM8_TRGO 相连。
位 19	ADC2_ETRGINJ_REMAP：ADC2 注入转换外部触发重映射（ADC2 external trigger injected conversion remapping） 该位可由软件置’1’或置’0’。 该位控制与 ADC2 注入转换外部触发相连的触发输入。 • 当该位置’0’时，ADC2 注入转换外部触发与 EXTI15 相连。 • 当该位置’1’时，ADC2 注入转换外部触发与 TIM8 通道 4 相连。
位 18	ADC1_ETRGREG_REMAP：ADC1 规则转换外部触发重映射（ADC1 external trigger regular conversion remapping） 该位可由软件置’1’或置’0’。 该位控制与 ADC1 规则转换外部触发相连的触发输入。 • 当该位置’0’时，ADC1 规则转换外部触发与 EXTI11 相连。 • 当该位置’1’时，ADC1 规则转换外部触发与 TIM8_TRGO 相连。
位 17	ADC1_ETRGINJ_REMAP：ADC1 注入转换外部触发重映射（ADC1 External trigger injected conversion remapping） 该位可由软件置’1’或置’0’。 该位控制与 ADC1 注入转换外部触发相连的触发输入。 • 当该位置’0’时，ADC1 注入转换外部触发与 EXTI15 相连。 • 当该位置’1’时，ADC1 注入转换外部触发与 TIM8 通道 4 相连。
位 16	TIM5CH4_IREMAP：TIM5 通道 4 内部重映射（TIM5 channel4 internal remap） 该位可由软件置’1’或置’0’。 该位与"AFIO_MAPR2.TIM5CH4_IREMAP2"组合，选择 TIM5 的 CH4 输入源。 参考 AFIO_MAPR2.TIM5CH4_IREMAP2 位的描述。

位 15	PD01_REMAP：端口 D0/端口 D1 映射到 OSC_IN/OSC_OUT（Port D0/Port D1 mapping on OSC_IN/OSC_OUT） 该位可由软件置'1'或置'0'。 它控制 PD0 和 PD1 的 GPIO 功能映射。当不使用主振荡器 HSE 时（应用程序工作在内部的 8 MHz RC 振荡器下），PD0 和 PD1 可以映射到 OSC_IN 和 OSC_OUT 引脚。 • 0：不进行 PD0 和 PD1 的重映射。 • 1：PD0 映射到 OSC_IN，PD1 映射到 OSC_OUT。
位 14:13	CAN1_REMAP[1:0]：CAN 复用功能重映射（CAN alternate function remapping） 这些位可由软件置'1'或置'0'。 该位域控制复用功能 CAN1_RX 和 CAN1_TX 的重映射。 • 00：CAN1_RX 映射到 PA11，CAN1_TX 映射到 PA12。 • 01：未用组合 • 10：CAN1_RX 映射到 PB8，CAN1_TX 映射到 PB9。 • 11：CAN1_RX 映射到 PD0，CAN1_TX 映射到 PD1。
位 12	TIM4_REMAP：定时器 4 的重映射（TIM4 remapping） 该位可由软件置'1'或置'0'，控制将 TIM4 的通道 1 至 4 映射到 GPIO 端口上。 • 0：没有重映射（TIM4_CH1/PB6，TIM4_CH2/PB7，TIM4_CH3/PB8，TIM4_CH4/PB9） • 1：完全映射（TIM4_CH1/PD12，TIM4_CH2/PD13，TIM4_CH3/PD14，TIM4_CH4/PD15） *说明：重映射不影响在 PE0 上的 TIM4_ETR。*
位 11:10	TIM3_REMAP[1:0]：定时器 3 的重映射（TIM3 remapping） 这些位可由软件置'1'或置'0'，控制定时器 3 的通道 1 至 4 在 GPIO 端口的映射。 • 00：没有重映射（CH1/PA6，CH2/PA7，CH3/PB0，CH4/PB1） • 01：未用组合 • 10：部分映射（CH1/PB4，CH2/PB5，CH3/PB0，CH4/PB1） • 11：完全映射（CH1/PC6，CH2/PC7，CH3/PC8，CH4/PC9） *说明：重映射不影响在 PD2 上的 TIM3_ETR。*
位 9:8	TIM2_REMAP[1:0]：定时器 2 的重映射（TIM2 remapping） 这些位可由软件置'1'或置'0'，控制定时器 2 的通道 1 至 4 和外部触发（ETR）在 GPIO 端口的映射。 • 00：没有重映射（CH1/ETR/PA0，CH2/PA1，CH3/PA2，CH4/PA3） • 01：部分映射（CH1/ETR/PA15，CH2/PB3，CH3/PA2，CH4/PA3） • 10：部分映射（CH1/ETR/PA0，CH2/PA1，CH3/PB10，CH4/PB11） • 11：完全映射（CH1/ETR/PA15，CH2/PB3，CH3/PB10，CH4/PB11）
位 7:6	TIM1_REMAP[1:0]：定时器 1 的重映射（TIM1 remapping） 这些位可由软件置'1'或置'0'，控制定时器 1 的通道 1 至 4、1N 至 3N、外部触发（ETR）和刹车输入（BKIN）在 GPIO 端口的映射。 • 00：没有重映射（ETR/PA12，CH1/PA8，CH2/PA9，CH3/PA10，CH4/PA11，BKIN/PB12，CH1N/PB13，CH2N/PB14，CH3N/PB15） • 01：部分映射（ETR/PA12，CH1/PA8，CH2/PA9，CH3/PA10，CH4/PA11，BKIN/PA6，CH1N/PA7，CH2N/PB0，CH3N/PB1） • 10：未使用 • 11：完全映射（ETR/PE7，CH1/PE9，CH2/PE11，CH3/PE13，CH4/PE14，BKIN/PE15，CH1N/PE8，CH2N/PE10，CH3N/PE12）
位 5:4	USART3_REMAP[1:0]：USART3 的重映射（USART3 remapping）

	这些位可由软件置'1'或置'0'，控制 USART3 的 CTS、RTS、CK、TX 和 RX 复用功能在 GPIO 端口的映射。 • 00：没有重映射（TX/PB10，RX/PB11，CK/PB12，CTS/PB13，RTS/PB14） • 01：部分映射（TX/PC10，RX/PC11，CK/PC12，CTS/PB13，RTS/PB14） • 10：未使用 • 11：完全映射（TX/PD8，RX/PD9，CK/PD10，CTS/PD11，RTS/PD12）
位 3	USART2_REMAP：USART2 的重映射（USART2 remapping） 这些位可由软件置'1'或置'0'，控制 USART2 的 CTS、RTS、CK、TX 和 RX 复用功能在 GPIO 端口的映射。 • 0：没有重映射（CTS/PA0，RTS/PA1，TX/PA2，RX/PA3，CK/PA4） • 1：重映射（CTS/PD3，RTS/PD4，TX/PD5，RX/PD6，CK/PD7）
位 2	USART1_REMAP：USART1 的重映射（USART1 remapping） 该位可由软件置'1'或置'0'，控制 USART1 的 TX 和 RX 复用功能在 GPIO 端口的映射。 • 0：没有重映射（TX/PA9，RX/PA10） • 1：重映射（TX/PB6，RX/PB7）
位 1	I2C1_REMAP：I2C1 的重映射（I2C1 remapping） 该位可由软件置'1'或置'0'，控制 I2C1 的 SCL 和 SDA 复用功能在 GPIO 端口的映射。 • 0：没有重映射（SCL/PB6，SDA/PB7） • 1：重映射（SCL/PB8，SDA/PB9）
位 0	SPI1_REMAP：SPI1 的重映射（SPI1 remapping） 该位可由软件置'1'或置'0'，控制 SPI1 的 NSS、SCK、MISO 和 MOSI，以及 I2S1 的 WS、CK、SD、MCK 复用功能在 GPIO 端口的映射。 • 0：没有重映射（SPI1_NSS(I2S1_WS)/PA4，SPI1_SCK(I2S1_CK)/PA5，SPI1_MISO/PA6，SPI1_MOSI(I2S1_SD)/PA7，I2S_MCK/PC4） • 1：重映射（SPI1_NSS(I2S1_WS)/PA15，SPI1_SCK(I2S1_CK)/PB3，SPI1_MISO/PB4，SPI1_MOSI(I2S1_SD)/PB5，I2S_MCK/PB6）

10.4.3 外部中断配置寄存器 1（AFIO_EXTICR1）

偏移地址：0x08

复位值：0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
rw				rw				rw				rw			

位 31:16	Res：保留 必须保持复位值。
位 4x+3:4x (x=3..0)	EXTIx[3:0]：EXTIx 配置（EXTI x configuration） 这些位可由软件读写，用于选择 EXTIx 外部中断的输入源。参看"9.2.6 外部中断/事件线映射"。 • 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

10.4.4 外部中断配置寄存器 2（AFIO_EXTICR2）

偏移地址：0x0C

复位值：0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
rw				rw				rw				rw			

位 31:16	Res：保留 必须保持复位值。
位 4(x-4)+3:4(x-4) (x=7..4)	EXTIx[3:0]：EXTIx 配置（EXTI x configuration） 这些位可由软件读写，用于选择 EXTIx 外部中断的输入源。 • 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 • 0101：保留 • 0110：保留

10.4.5 外部中断配置寄存器 3（AFIO_EXTICR3）

偏移地址：0x10

复位值：0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
rw				rw				rw				rw			

位 31:16	Res：保留 必须保持复位值。
位 4(x-8)+3:4(x-8) (x=11..8)	EXTIx[3:0]：EXTIx 配置（EXTI x configuration） 这些位可由软件读写，用于选择 EXTIx 外部中断的输入源。 • 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 • 0101：保留

	• 0110：保留

10.4.6 外部中断配置寄存器 4（AFIO_EXTICR4）

偏移地址：0x14

复位值：0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
rw				rw				rw				rw			

位 31:16	Res：保留 必须保持复位值。
位 4(x-12)+3:4(x-12) (x=15..12)	EXTIx[3:0]：EXTIx 配置（EXTI x configuration） 这些位可由软件读写，用于选择 EXTIx 外部中断的输入源。 • 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 • 0101：保留 • 0110：保留

10.4.7 复用重映射和调试 I/O 配置寄存器 2（AFIO_MAPR2）

偏移地址：0x1C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FSMC_100PIN_REMAP	CAN2_REMAP[1:0]		USART5_REMAP	USART4_REMAP	USART2_REMAP2	USART1_CTSRTS_REMAP	I2C1_SMBA_REMAP	SPI3_REMAP	DCMI_REMAP	TIM5CH4_IREMAP2	TIM5CH4_IREMAP3	SAI_AB_REMAP[1:0]			
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			

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

位 31	FSMC_100PIN_REMAP：控制可使用的 FSMC 接口信号（Available FSMC interface signals control） • 0：100 引脚产品不支持下列 FSMC 接口信号。 • 1：100 引脚产品支持下列 FSMC 接口信号。 FSMC 接口信号：NIORD，NE4，NIOWR，NREG，A[25:24]，A[15:0]，INTR，INT3，INT2，NCE4_2，NE3_NCE4_1，NE2_NCE3，CD。
位 30:29	CAN2_REMAP[1:0]：CAN2 复用功能重映射（CAN2 alternate function remapping） 这些位可由软件置’1’或置’0’。 该位域控制复用功能 CAN2_RX 和 CAN2_TX 的重映射。

	• 00：CAN2_RX 映射到 PB12，CAN2_TX 映射到 PB13。 • 01：CAN2_RX 映射到 PB5，CAN2_TX 映射到 PB6。 • 10：CAN2_RX 映射到 PB0，CAN2_TX 映射到 PB1。 • 11：CAN2_RX 映射到 PB12，CAN2_TX 映射到 PB13（等同于设置为 00）。
位 28	USART5_REMAP：控制 USART5_RX 和 USART5_TX 所映射至的引脚（USART5 remapping） • 0：USART5_RX 分配在 PD2，USART5_TX 分配在 PC12 • 1：USART5_RX 分配在 PC9，USART5_TX 分配在 PC8
位 27	USART4_REMAP：控制 USART4_RX 和 USART4_TX 所映射至的引脚（USART4 remapping） • 0：USART4_RX 分配在 PC11，USART4_TX 分配在 PC10。 • 1：USART4_RX 分配在 PB9，USART4_TX 分配在 PB8。
位 26	USART2_REMAP2：控制 USART2 的 CTS、RTS、TX、RX 和 CK 所映射至的引脚（USART2 remapping） • 0：USART2_CTS 分配在 PA0，USART2_RTS 分配在 PA1，USART2_TX 分配在 PA2，USART2_RX 分配在 PA3，USART2_CK 分配在 PA4 • 1：USART2_CTS 分配在 PA14，USART2_RTS 分配在 PA15，USART2_TX 分配在 PC10，USART2_RX 分配在 PC11，USART2_CK 分配在 PC12 *说明：USART2_REMAP2 功能优先级高于 USART2_REMAP（位于 AFIO_MAPR 寄存器）。*
位 25	USART1_CTSRTS_REMAP：控制 USART1_CTS 和 USART1_RTS 所映射至的引脚（USART1_CTS/RTS remapping） • 0：USART1_CTS 分配在 PA11，USART1_RTS 分配在 PA12。 • 1：USART1_CTS 分配在 PC8，USART1_RTS 分配在 PC9。
位 24	I2C1_SMBA_REMAP：控制 I2C1_SMBA 所映射至的引脚（I2C1_SMBA remapping） • 0：I2C1_SMBA 分配在 PB5 • 1：I2C1_SMBA 分配在 PA14
位 23	SPI3_REMAP：控制 SPI3 的 NSS、SCK、MISO 和 MOSI，以及 I2S3 的 WS、CK、SD 所映射至的引脚（SPI3 remapping） • 0：SPI3_NSS(I2S3_WS)/PA15，SPI3_SCK(I2S3_CK)/PB3，SPI3_MISO/PB4，SPI3_MOSI(I2S3_SD)/PB5 • 1：SPI3_NSS(I2S3_WS)/PD3，SPI3_SCK(I2S3_CK)/PD4，SPI3_MISO/PD5，SPI3_MOSI(I2S3_SD)/PD6
位 22	DCMI_REMAP：控制 DCMI 的 VSYNC、HSYNC、PIXDI10/11/12/13 所映射至的引脚（DCMI remapping） • 0：VSYNC/PA11，HSYNC/PA12，PIXDI10/PA0，PIXDI11/PC3，PIXDI12/PC2，PIXDI13/PC1 • 1：VSYNC/PA13，HSYNC/PA14，PIXDI10/PB6，PIXDI11/PB5，PIXDI12/PB4，PIXDI13/PB3
位 21	TIM5CH4_IREMAP2：选择 TIM5 的 CH4 输入源（TIM5CH4 remapping control bit2） 该位与"AFIO_MAPR.TIM5CH4_IREMAP"组合，选择 TIM5 的 CH4 输入源。 ｛TIM5CH4_IREMAP2，TIM5CH4_IREMAP｝=2'b00：GPIO 作为 TIM5 的 CH4 输入源。 ｛TIM5CH4_IREMAP2，TIM5CH4_IREMAP｝=2'b01：LSI 作为 TIM5 的 CH4 输入源。 ｛TIM5CH4_IREMAP2，TIM5CH4_IREMAP｝=2'b10：分频后的 HSI 作为 TIM5 的 CH4 输入源。 ｛TIM5CH4_IREMAP2，TIM5CH4_IREMAP｝=2'b11：LSE 作为 TIM5 的 CH4 输入源。
位 20	TIM5CH4_IREMAP3：选择 TIM5 的 CH4 输入源（TIM5CH4 remapping control bit3） • 0：TIM5 的 CH4 输入源由 TIM5CH4_IREMAP2 和 TIM5CH4_IREMAP 寄存器值决定。 • 1：USB2.0 接收到的 SOF 信号作为 TIM5 的 CH4 输入源。

位 19:18	SAI_AB_REMAP[1:0]：选择 SAI_A 和 SAI_B 所映射至的引脚（SAI_A/B remapping） • 00：SAI_MCLK_B=PA5, SAI_FS_B=PA4, SAI_SCK_B=PA6, SAI_SD_B=PA7； SAI_MCLK_A=PA3, SAI_FS_A=PA9, SAI_SCK_A=PA8, SAI_SD_A=PA10； • 01：SAI_MCLK_B=PB4, SAI_FS_B=PB6, SAI_SCK_B=PB3, SAI_SD_B=PB5； SAI_MCLK_A=PB8, SAI_FS_A=PB9, SAI_SCK_A=PB10, SAI_SD_A=PB2； • 10：SAI_MCLK_B=PE10, SAI_FS_B=PE9, SAI_SCK_B=PE8, SAI_SD_B=PE7; SAI_MCLK_A=PE2, SAI_FS_A=PE4, SAI_SCK_A=PE5, SAI_SD_A=PE6; • 11：等同于设置为 00。
位 17:11	Res：保留 必须保持复位值。
位 10	FSMC_NADV：使能 FSMC_NADV 信号输出（FSMC_NADV remapping） • 0：FSMC_NADV 信号输出使能。 • 1：FSMC_NADV 信号输出关闭，其引脚可用作 GPIO 或其它外设功能。
位 9:0	Res：保留 必须保持复位值。

11 DMA 控制器

直接存储器访问（DMA）用来提供外设和存储器之间或者存储器和存储器之间的高速数据传输。无须 CPU 干预，数据可以通过 DMA 快速地移动，这就节省了 CPU 的资源来做其他操作。

两个 DMA 控制器共有 12 个通道（DMA1 有 7 个通道，DMA2 有 5 个通道），每个通道专门用来管理来自于一个或多个外设对存储器的访问请求。还有一个仲裁器来协调各个 DMA 请求的优先权。

11.1 DMA 主要特性

- 12 个独立的可配置的通道（请求）：DMA1 有 7 个通道，DMA2 有 5 个通道。
- 每个通道都直接连接专用的硬件 DMA 请求，每个通道都同样支持软件触发。通过软件来配置该功能。
- 在同一个 DMA 模块上，多个请求间的优先权可以通过软件编程设置（共有四级：最高、高、中等和低），优先权设置相等时由硬件决定（请求 0 优先于请求 1，依此类推）。
- 独立数据源和目标数据区的传输宽度（字节、半字、字），模拟打包和拆包的过程。源和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 传输出错），这 3 个事件标志逻辑或成为一个单独的中断请求。
- Flash、SRAM、外设的 SRAM、APB1、APB2 和 AHB 外设均可作为访问的源和目标。
- 存储器到存储器的传输、外设和存储器、存储器和外设之间的传输
- 可编程的数据传输数目：最大为 65536
- 支持 DMA1 和 DMA2 之间的优先级设置
- 支持外设的 DMA
- DMA2 的通道 1 的最大传输个数为 0xFFFF FFFF，对应的 CNDTR 宽度和计数器计数值上限为 32 位。

下面为功能框图：

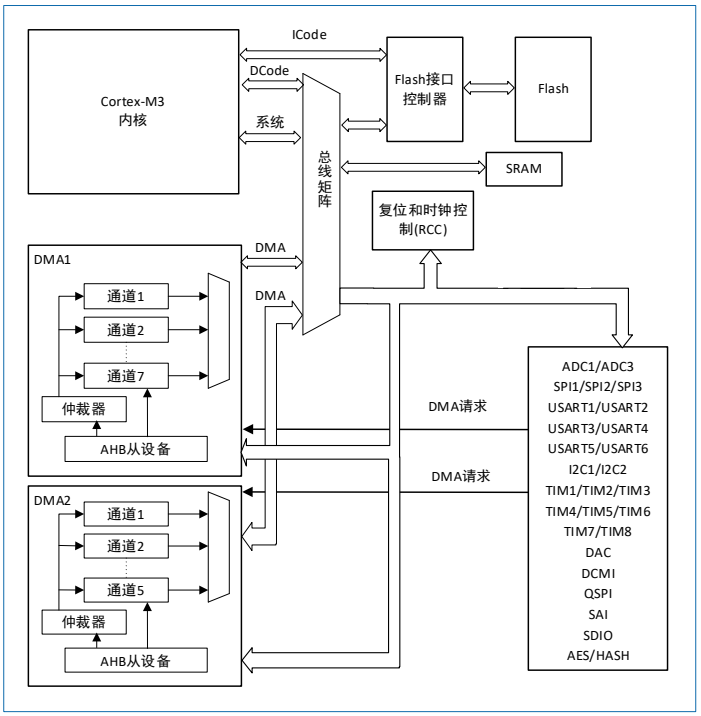


图 11-1 DMA 框图

11.2 功能描述

DMA 控制器通过与 Cortex®-M3 内核共享的系统总线来执行直接存储器数据传输。当 CPU 和 DMA 同时访问相同的目标（存储器或外设）时，DMA 请求会暂停 CPU 在若干个周期内访问系统总线。总线仲裁器执行循环调度，这可以保证 CPU 至少可以得到一半的系统总线（存储器或外设）带宽。

11.2.1 DMA 处理

在发生一个事件后，外设向 DMA 控制器发送一个请求信号。DMA 控制器根据通道的优先权处理请求。当 DMA 控制器开始访问发出请求的外设时，DMA 控制器立即向它发送一个应答信号。当从 DMA 控制器得到应答信号时，外设立即释放它的请求。一旦外设释放了这个请求，DMA 控制器撤销应答信号。如果有更多的请求时，外设可以启动下一个周期。

总之，每次 DMA 传输由 3 个操作组成：

- 从外设数据寄存器或者从当前外设/存储器地址寄存器指示的存储器地址取数据，第一次传输时的开始地址是 DMA_CHANNEL_CPAR 或 DMA_CHANNEL_CMAR 寄存器指定的外设基地址或存储器地址。
- 向外设数据寄存器或者当前外设/存储器地址寄存器指示的存储器地址存数据，第一次传输时的开始地址是 DMA_CHANNEL_CPAR 或 DMA_CHANNEL_CMAR 寄存器指定的外设基地址或存储器地址。
- 执行一次 DMA_CHANNEL_CNDTR 寄存器的递减操作，该寄存器包含未完成的操作次数。

11.2.2 仲裁器

仲裁器根据通道请求的优先级来启动外设/存储器的访问。

优先权管理分 2 个阶段：

- 软件：每个通道的优先权可以在 DMA_CHANNEL_CCR 寄存器中设置，有 4 个等级：
  - 最高优先级
  - 高优先级
  - 中等优先级
  - 低优先级
- 硬件：如果 2 个请求有相同的软件优先级，则较低编号的通道比较高编号的通道有更高的优先级。举个例子，通道 2 优先于通道 4。

注意：DMA1 控制器的优先级高于 DMA2。

11.2.3 DMA 通道

每个通道都可以在有固定地址的外设寄存器和存储器地址之间执行 DMA 传输。在每次传输后，包含了待传输数据量的寄存器会递减。

11.2.3.1 可编程的数据量

外设和存储器的传输数据宽度可以通过 DMA_CHANNEL_CCR 寄存器中的 PSIZE 和 MSIZE 位编程。

11.2.3.2 指针增量

通过设置 DMA_CHANNEL_CCR 寄存器中的 PINC 和 MINC 标志位，外设和存储器的指针在每次传输后可以选成自动增加模式。当设置为增量模式时，下一个要传输的地址将是前一个地址加上增量值，增量值取决由所选的数据宽度决定，可以为 1、2 或 4。第一个传输的地址是存放在 DMA_CHANNEL_CPAR/DMA_CHANNEL_CMAR 寄存器中。在传输过程中，这些寄存器保持它们初始的数值，软件不能改变和读出当前正在传输的地址（它在当前的内部外设/存储器地址寄存器中）。

当通道配置为非循环模式时，传输结束后（即待传输次数变为 0）将不再处理 DMA 请求。为了载入新的传输数目到 DMA_CHANNEL_CNDTR 寄存器中，必须禁用 DMA 通道。

注意：若禁用了 DMA 通道，则 DMA 寄存器不会被复位。DMA 寄存器（DMA_CHANNEL_CCR, DMA_CHANNEL_CPAR 和 DMA_CHANNEL_CMAR）在通道配置期间仍保持初始值。

在循环模式下，最后一次传输结束时，DMA_CHANNEL_CNDTR 寄存器会自动地再次加载为编程的初始值。当前的内部地址寄存器将再次载入 DMA_CHANNEL_CPAR/DMA_CHANNEL_CMAR 寄存器中的基地址。

11.2.3.3 通道配置过程

下面是配置 DMA 通道的步骤：

1. 在 DMA_CHANNEL_CPAR 寄存器中设置外设寄存器的地址。发生外设数据传输请求时，这个地址将是数据传输的源或目标。
2. 在 DMA_CHANNEL_CMAR 寄存器中设置数据存储器的地址。发生外设数据传输请求时，传输的数据将从这个地址读出或写入这个地址。
3. 在 DMA_CHANNEL_CNDTR 寄存器中设置要传输的数据量。在每个数据传输后，这个数值递减。
4. 在 DMA_CHANNEL_CCR 寄存器的 PL[1:0]位中设置通道的优先级。
5. 在 DMA_CHANNEL_CCR 寄存器中设置数据传输的方向、循环模式、外设和存储器的增量模式、外设和存储器的数据宽度、传输一半产生中断或传输完成产生中断。
6. 设置 DMA_CHANNEL_CCR 寄存器的 ENABLE 位，启动该通道。

一旦启动了 DMA 通道，它就能处理连到该通道上的外设的 DMA 请求。

当传输一半的数据后，半传输标志（HTIF）被置'1'，当设置了半传输中断使能位（HTIE）时，将产生一个中断请求。在数据传输结束后，传输完成标志（TCIF）被置'1'，当设置了传输完成中断使能位（TCIE）时，将产生一个中断请求。

11.2.3.4 循环模式

循环模式用于处理循环缓冲区和连续的数据传输（如 ADC 的扫描模式）。在 DMA_CHANNEL_CCR 寄存器中的 CIRC 位用于开启这一功能。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配置通道时设置的初值，DMA 操作将会继续进行。

11.2.3.5 存储器到存储器模式

DMA 通道的操作可以在没有外设请求的情况下进行，这种操作就是存储器到存储器模式。

当设置了 DMA_CHANNEL_CCR 寄存器中的 MEM2MEM 位之后，在软件设置了 DMA_CHANNEL_CCR 寄存器中的 EN 位启动 DMA 通道时，DMA 传输将马上开始。当 DMA_CHANNEL_CNDTR 寄存器变为 0 时，DMA 传输结束。存储器到存储器模式不能与循环模式同时使用。

11.2.4 可编程的数据传输宽度、对齐方式和数据大小端

当 PSIZE 和 MSIZE 不相同时，DMA 模块按照下表进行数据对齐。

表 11-1 可编程的数据传输宽度和大小端操作（当 PINC=MINC=1）

源端口宽度	目标端口宽度	传输数目(NDT)	源内容：地址/数据	传输操作	目标内容：地址/数据
8	8	4	0x0/B0 0x1/B1 0x2/B2 0x3/B3	1：在 0x0 读 B0[7:0]，在 0x0 写 B0[7:0] 2：在 0x1 读 B1[7:0]，在 0x1 写 B1[7:0] 3：在 0x2 读 B2[7:0]，在 0x2 写 B2[7:0] 4：在 0x3 读 B3[7:0]，在 0x3 写 B3[7:0]	0x0/B0 0x1/B1 0x2/B2 0x3/B3
8	16	4	0x0/B0 0x1/B1 0x2/B2 0x3/B3	1：在 0x0 读 B0[7:0]，在 0x0 写 00B0[15:0] 2：在 0x1 读 B1[7:0]，在 0x2 写 00B1[15:0] 3：在 0x2 读 B2[7:0]，在 0x4 写 00B2[15:0] 4：在 0x3 读 B3[7:0]，在 0x6 写 00B3[15:0]	0x0/00B0 0x2/00B1 0x4/00B2 0x6/00B3
8	32	4	0x0/B0 0x1/B1 0x2/B2 0x3/B3	1：在 0x0 读 B0[7:0]，在 0x0 写 000000B0[31:0] 2：在 0x1 读 B1[7:0]，在 0x4 写 000000B1[31:0] 3：在 0x2 读 B2[7:0]，在 0x8 写 000000B2[31:0] 4：在 0x3 读 B3[7:0]，在 0xC 写 000000B3[31:0]	0x0/000000B0 0x4/000000B1 0x8/000000B2 0xC/000000B3
16	8	4	0x0/B1B0 0x2/B3B2 0x4/B5B4 0x6/B7B6	1：在 0x0 读 B1B0[15:0]，在 0x0 写 B0[7:0] 2：在 0x2 读 B3B2[15:0]，在 0x1 写 B2[7:0] 3：在 0x4 读 B5B4[15:0]，在 0x2 写 B4[7:0] 4：在 0x6 读 B7B6[15:0]，在 0x3 写 B6[7:0]	0x0/B0 0x1/B2 0x2/B4 0x3/B6
16	16	4	0x0/B1B0 0x2/B3B2 0x4/B5B4 0x6/B7B6	1：在 0x0 读 B1B0[15:0]，在 0x0 写 B1B0[15:0] 2：在 0x2 读 B3B2[15:0]，在 0x2 写 B3B2[15:0] 3：在 0x4 读 B5B4[15:0]，在 0x4 写 B5B4[15:0] 4：在 0x6 读 B7B6[15:0]，在 0x6 写 B7B6[15:0]	0x0/B1B0 0x2/B3B2 0x4/B5B4 0x6/B7B6
16	32	4	0x0/B1B0 0x2/B3B2 0x4/B5B4 0x6/B7B6	1：在 0x0 读 B1B0[15:0]，在 0x0 写 0000B1B0[31:0] 2：在 0x2 读 B3B2[15:0]，在 0x4 写 0000B3B2[31:0] 3：在 0x4 读 B5B4[15:0]，在 0x8 写 0000B5B4[31:0] 4：在 0x6 读 B7B6[15:0]，在 0xC 写 0000B7B6[31:0]	0x0/0000B1B0 0x4/0000B3B2 0x8/0000B5B4 0xC/0000B7B6
32	8	4	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC	1：在 0x0 读 B3B2B1B0[31:0]，在 0x0 写 B0[7:0] 2：在 0x4 读 B7B6B5B4[31:0]，在 0x1 写 B4[7:0] 3：在 0x8 读 BBBAB9B8[31:0]，在 0x2 写 B8[7:0] 4：在 0xC 读 BFBEBDBC[31:0]，在 0x3 写 BC[7:0]	0x0/B0 0x1/B4 0x2/B8 0x3/BC
32	16	4	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC	1：在 0x0 读 B3B2B1B0[31:0]，在 0x0 写 B1B0[15:0] 2：在 0x4 读 B7B6B5B4[31:0]，在 0x2 写 B5B4[15:0] 3：在 0x8 读 BBBAB9B8[31:0]，在 0x4 写 B9B8[15:0] 4：在 0xC 读 BFBEBDBC[31:0]，在 0x6 写 BDBC[15:0]	0x0/B1B0 0x2/B5B4 0x4/B9B8 0x6/BDBC
32	32	4	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC	1：在 0x0 读 B3B2B1B0[31:0]，在 0x0 写 B3B2B1B0[31:0] 2：在 0x4 读 B7B6B5B4[31:0]，在 0x4 写 B7B6B5B4[31:0] 3：在 0x8 读 BBBAB9B8[31:0]，在 0x8 写 BBBAB9B8[31:0] 4：在 0xC 读 BFBEBDBC[31:0]，在 0xC 写 BFBEBDBC[31:0]	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC

11.2.4.1 寻址一个不支持字节或半字写的 AHB 设备

当 DMA 模块开始一个 AHB 的字节或半字写操作时，数据将在 HWDATA[31:0]总线中未使用的部分

重复。因此，如果 DMA 以字节或半字写入不支持字节或半字写操作的 AHB 设备时（即 HSIZE 不适于该模块），不会发生错误，DMA 将按照下面两个例子写入 32 位 HWDATA 数据：

当 HSIZE=半字时，写入半字'0xABCD'，DMA 将设置 HWDATA 总线为'0xABCDABCD'。

当 HSIZE=字节时，写入字节'0xAB'，DMA 将设置 HWDATA 总线为'0xABABABAB'。

假定 AHB/APB 桥是一个 AHB 的 32 位从设备，它不处理 HSIZE 参数，它将按照下述方式把任何 AHB 上的字节或半字按 32 位传送到 APB 上：

- 一次向地址 0x0（或 0x1、0x2 或 0x3）写一个字节数据'0xB0'（即 AHB 字节写操作），将被转换成对地址 0x0 写入一个字数据'0xB0B0B0B0'。
- 一次向地址 0x0（或 0x2）写半个字数据' 0xB1B0'（即 AHB 半字写操作），将被转换成对地址 0x0 写入一个字数据'0xB1B0B1B0'。

例如，如果要写入 APB 备份寄存器（与 32 位地址对齐的 16 位寄存器），需要配置存储器数据源宽度（MSIZE）为'16 位'，外设目标数据宽度（PSIZE）为'32 位'。

11.2.5 错误管理

读/写一个保留的地址区域，将会产生 DMA 传输错误。当在 DMA 读写操作时发生了 DMA 传输错误，硬件会清除发生错误的通道所对应的通道配置寄存器（DMA_CHANNEL_CCR）的 EN 位，该通道操作自动被停止。此时，在 DMA_ISR 寄存器中对应该通道的传输错误中断标志位（TEIF）将被置位，如果在 DMA_CHANNEL_CCR 寄存器中设置了传输错误中断使能位，则将产生中断。

11.2.6 中断

每个 DMA 通道都可以在 DMA 传输过半、传输完成和传输错误时产生中断。为应用的灵活性考虑，通过设置寄存器的不同位来打开这些中断。

表 11-2 DMA 中断请求

中断事件	事件标志位	使能控制位
传输过半	HTIF	HTIE
传输完成	TCIF	TCIE
传输错误	TEIF	TEIE

注意：DMA 通道都有自己的中断向量，详细的请参考中断向量表。

11.2.7 DMA 请求映射

11.2.7.1 DMA1 控制器

从外设（TIMx[x=1、2、3、4]、ADC1、SPI1、SPI2、I2Cx[x=1、2]和 USARTx[x=1、2、3]）产生的 7 个请求，通过逻辑或输入到 DMA1 控制器，这意味着同时只能有一个请求有效。参见下图的 DMA1 请求映射。

外设的 DMA 请求，可以通过设置相应外设寄存器中的控制位，被独立地开启或关闭。

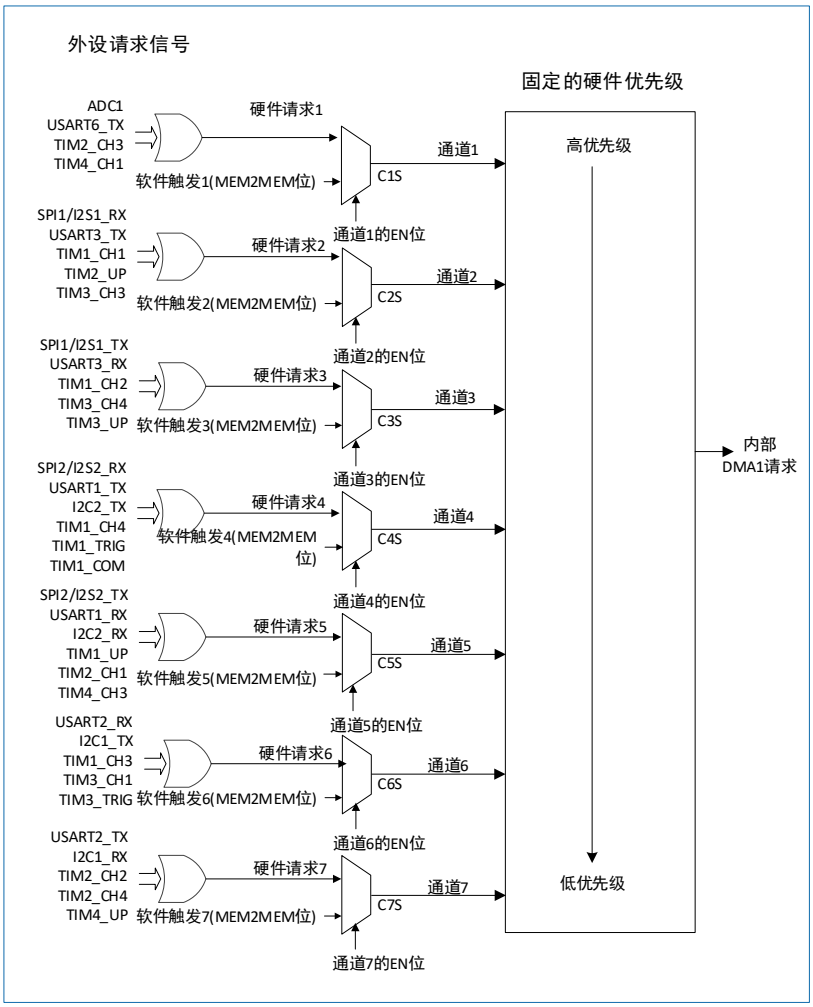


图 11-2 DMA1 请求映射

表 11-3 各个通道的 DMA1 请求一览

外设	通道 1	通道 2	通道 3	通道 4	通道 5	通道 6	通道 7
ADC1	ADC1	-	-	-	-	-	-
SPI	-	SPI1_RX/ I2S1_RX	SPI1_TX/ I2S1_TX	SPI2_RX/ I2S2_RX	SPI2_TX/ I2S2_TX	-	-
USART	USART6_TX	USART3_TX	USART3_RX	USART1_TX	USART1_RX	USART2_RX	USART2_TX
I2C	-	-	-	I2C2_TX	I2C2_RX	I2C1_TX	I2C1_RX
TIM1	-	TIM1_CH1	TIM1_CH2	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3	-
TIM2	TIM2_CH3	TIM2_UP	-	-	TIM2_CH1	-	TIM2_CH2 TIM2_CH4
TIM3	-	TIM3_CH3	TIM3_CH4 TIM3_UP	-	-	TIM3_CH1 TIM3_TRIG	-
TIM4	TIM4_CH1	-	-	TIM4_CH2	TIM4_CH3		TIM4_UP

11.2.7.2 DMA2 控制器

从外设产生的 5 个请求，经逻辑或输入到 DMA2 控制器，这意味着同时只能有一个请求有效。参见下图。外设的 DMA 请求，可以通过设置相应外设寄存器中的 DMA 控制位，被独立地开启或关闭。

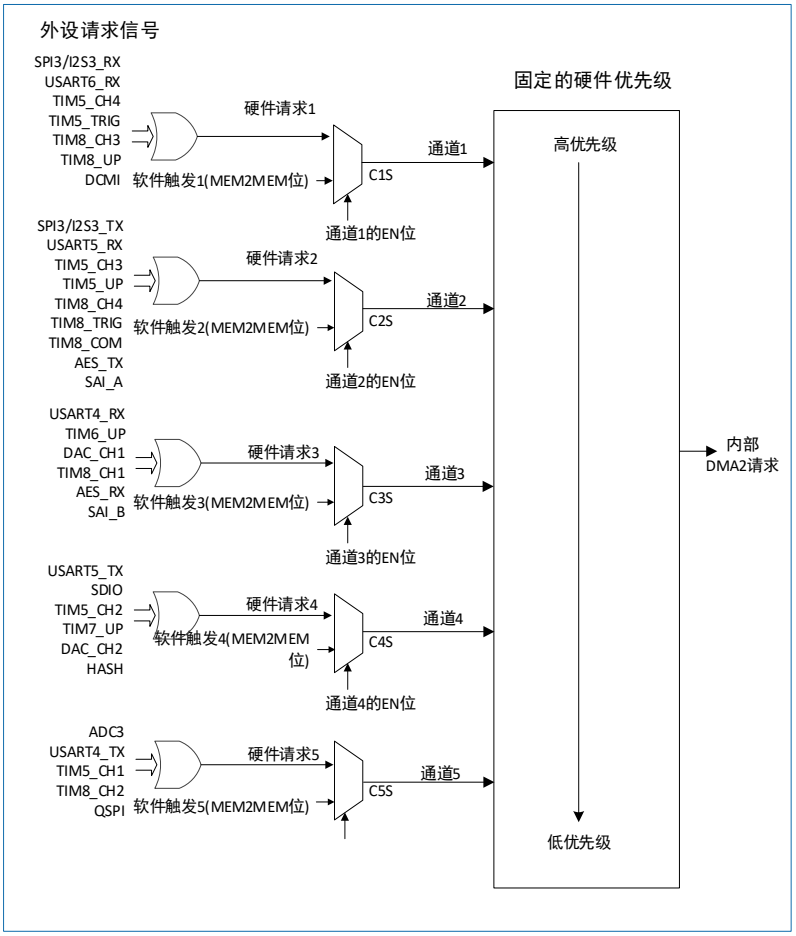


图 11-3 DMA2 请求映射

表 11-4 各个通道的 DMA2 请求一览

外设	通道 1	通道 2	通道 3	通道 4	通道 5
ADC3	-	-	-	-	ADC3
SPI3/I2S3	SPI3_RX/I2S3_RX	SPI3_TX/I2S3_TX	-	-	-
USART	USART6_RX	USART5_RX	USART4_RX	USART5_TX	USART4_TX
SDIO	-	-	-	SDIO	-
TIM5	TIM5_CH4 TIM5_TRIG	TIM5_CH3 TIM5_UP	-	TIM5_CH2	TIM5_CH1
TIM6/DAC	-	-	TIM6_UP/ DAC_CH1	-	-
TIM7/DAC	-	-	-	TIM7_UP/DAC_CH2	-
TIM8	TIM8_CH3 TIM8_UP	TIM8_CH4 TIM8_TRIG TIM8_COM	TIM8_CH1	-	TIM8_CH2

外设	通道 1	通道 2	通道 3	通道 4	通道 5
安全模块	-	AES_TX	AES_RX	HASH	-
其他外设	DCMI	SAI_A	SAI_B	-	QSPI

11.3 DMA 寄存器

基地址：（DMA1, DMA2）=（0x4002 0000, 0x4002 0400）

空间大小：（DMA1, DMA2）=（0x400, 0x400）

11.3.1 DMA 中断状态寄存器（DMA_ISR）

偏移地址：0x00

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res				TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5
				r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31:28	Res：保留 必须保持复位值。
位 4x-1 (x=7..1)	TEIFx：通道 x 的传输错误标志（Channel x transfer error flag） 硬件设置这些位。 在 DMA_IFCR 寄存器的相应位写入’1’可以清除这里对应的标志位。 • 0：在通道 x 没有传输错误（TE） • 1：在通道 x 发生了传输错误（TE）
位 4x-2 (x=7..1)	HTIFx：通道 x 的半传输标志（Channel x half transfer flag） 硬件设置这些位。 在 DMA_IFCR 寄存器的相应位写入’1’可以清除这里对应的标志位。 • 0：在通道 x 没有半传输事件（HT） • 1：在通道 x 产生了半传输事件（HT）
位 4x-3 (x=7..1)	TCIFx：通道 x 的传输完成标志（Channel x transfer complete flag） 硬件设置这些位。 在 DMA_IFCR 寄存器的相应位写入’1’可以清除这里对应的标志位。 • 0：在通道 x 没有传输完成事件（TC） • 1：在通道 x 产生了传输完成事件（TC）
位 4x-4 (x=7..1)	GIFx：通道 x 的全局中断标志（Channel x global interrupt flag） 硬件设置这些位。 在 DMA_IFCR 寄存器的相应位写入’1’可以清除这里对应的标志位。 • 0：在通道 x 没有 TE、HT 或 TC 事件 • 1：在通道 x 产生了 TE、HT 或 TC 事件

11.3.2 DMA 中断标志清除寄存器（DMA_IFCR）

偏移地址：0x04

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res				CTEIF7	CHTIF7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5
				w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTEIF4	CHTIF4	CTCIF4	CGIF4	CTEIF3	CHTIF3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

位 31:28	Res：保留 必须保持复位值。
位 4x-1 (x=7..1)	CTEIFx：清除通道 x 的传输错误标志（Channel x transfer error clear） 这些位由软件设置和清除。 • 0：不起作用 • 1：清除 DMA_ISR 寄存器中的对应 TEIF 标志
位 4x-2 (x=7..1)	CHTIFx：清除通道 x 的半传输标志（Channel x half transfer clear） 这些位由软件设置和清除。 • 0：不起作用 • 1：清除 DMA_ISR 寄存器中的对应 HTIF 标志
位 4x-3 (x=7..1)	CTCIFx：清除通道 x 的传输完成标志（Channel x transfer complete clear） 这些位由软件设置和清除。 • 0：不起作用 • 1：清除 DMA_ISR 寄存器中的对应 TCIF 标志
位 4x-4 (x=7..1)	CGIFx：清除通道 x 的全局中断标志（Channel x global interrupt clear） 这些位由软件设置和清除。 • 0：不起作用 • 1：清除 DMA_ISR 寄存器中的对应的 GIF、TEIF、HTIF 和 TCIF 标志

11.3.3 DMA1 和 DMA2 的优先级设置寄存器（HKDEF）

偏移地址：0x3F

复位值：0x0000 0000

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

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

位 31:1	Res：保留 必须保持复位值。
位 0	HOLD_DMAy：用于让另一个 DMAy 等待当前 DMAx 的当前数据搬移（y=1, x=2; y=2, x=1）（The priority of

	DMAx） 不可同时设置 DMA1 和 DMA2 的该寄存器。 • 0：DMAy 的数据传输不受 DMAx 影响。 • 1：DMAy 的数据传输需要等 DMAx 传输完成才能继续，即当前模块 DMAx 优先级更高。

11.4 DMA_CHANNEL 寄存器

基地址：（DMA1_Channel1, DMA1_Channel2, DMA1_Channel3, DMA1_Channel4, DMA1_Channel5, DMA1_Channel6, DMA1_Channel7; DMA2_Channel1, DMA2_Channel2, DMA2_Channel3, DMA2_Channel4, DMA2_Channel5） =（0x4002 0008, 0x4002 001C, 0x4002 0030, 0x4002 0044, 0x4002 0058,0x4002 006C, 0x4002 0080; 0x4002 0408, 0x4002 041C, 0x4002 0430, 0x4002 0444, 0x4002 0458）

空间大小：（DMA1_Channel1, DMA1_Channel2, DMA1_Channel3, DMA1_Channel4, DMA1_Channel5, DMA1_Channel6, DMA1_Channel7; DMA2_Channel1, DMA2_Channel2, DMA2_Channel3, DMA2_Channel4, DMA2_Channel5） =（0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14; 0x14, 0x14, 0x14, 0x14, 0x14）

11.4.1 DMA 通道配置寄存器（DMA_CHANNEL_CCR）

偏移地址：0x00

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	rw	rw		rw		rw		rw	rw	rw	rw	rw	rw	rw	rw

位 31:15	Res：保留 必须保持复位值。
位 14	MEM2MEM：存储器到存储器模式（Memory to memory mode） 该位由软件设置和清除。 • 0：非存储器到存储器模式 • 1：启动存储器到存储器模式
位 13:12	PL[1:0]：通道优先级（Channel priority level） 这些位由软件设置和清除。 • 00：低 • 01：中 • 10：高 • 11：最高
位 11:10	MSIZE[1:0]：存储器数据宽度（Memory size） 这些位由软件设置和清除。 • 00：8 位 • 01：16 位 • 10：32 位 • 11：保留
位 9:8	PSIZE[1:0]：外设数据宽度（Peripheral size） 这些位由软件设置和清除。

	• 00：8 位 • 01：16 位 • 10：32 位 • 11：保留
位 7	MINC：存储器地址增量模式（Memory increment mode） 该位由软件设置和清除。 • 0：不执行存储器地址增量操作 • 1：执行存储器地址增量操作
位 6	PINC：外设地址增量模式（Peripheral increment mode） 该位由软件设置和清除。 • 0：不执行外设地址增量操作 • 1：执行外设地址增量操作
位 5	CIRC：循环模式（Circular mode） 该位由软件设置和清除。 • 0：不执行循环操作 • 1：执行循环操作
位 4	DIR：数据传输方向（Data transfer direction） 该位由软件设置和清除。 • 0：从外设读 • 1：从存储器读
位 3	TEIE：允许传输错误中断（Transfer error interrupt enable） 该位由软件设置和清除。 • 0：禁止 TE 中断 • 1：允许 TE 中断
位 2	HTIE：允许半传输中断（Half transfer interrupt enable） 该位由软件设置和清除。 • 0：禁止 HT 中断 • 1：允许 HT 中断
位 1	TCIE：允许传输完成中断（Transfer complete interrupt enable） 该位由软件设置和清除。 • 0：禁止 TC 中断 • 1：允许 TC 中断
位 0	EN：通道开启（Channel enable） 该位由软件设置和清除。 • 0：通道不工作 • 1：通道开启

11.4.2 DMA 通道传输数量寄存器（DMA_CHANNEL_CNDTR）

偏移地址： 0x04

复位值：0x0000 0000

适用于除 DMA2 通道 1 以外所有通道的传输数量设置。

注意：为了满足 DCMI 应用的需要，DMA2 通道 1 的 CNDTR 寄存器为 32 位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NDT[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDT[15:0]															
rw															

位 31:0	NDT[31:0]：数据传输数量（Number of data to transfer） 数据传输数量为 0 至 2^{16}-1。这个寄存器只能在通道不工作（DMA_CHANNEL_CCR 的 EN=0）时写入。当寄存器的内容为 0 时，无论通道是否开启，都不会发生任何数据传输。 通道开启后，该寄存器变为只读，用于指示剩余的待传输字节数目。寄存器内容在每次 DMA 传输后递减。 数据传输结束后，寄存器的内容变为 0 或者当该通道配置为自动重加载模式时，寄存器的内容将自动重新加载为之前配置的数值。 *说明：其中数据传输数量高位（NDT[31:16]）仅 DMA2 的通道 1 使用该位域，其他 DMA 通道不使用该位域。*

11.4.3 DMA 通道外设地址寄存器（DMA_CHANNEL_CPAR）

偏移地址： 0x08

复位值：0x0000 0000

当开启通道（DMA_CHANNEL_CCR 的 EN=1）时不能写该寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PA[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PA[15:0]															
rw															

位 31:3	PA[31:0]：外设地址（Peripheral address） 外设数据寄存器的基地址，作为数据传输的源或目标。 • 当 PSIZE=’01’（16 位），不使用 PA[0]位。操作自动地与半字地址对齐。 • 当 PSIZE=’10’（32 位），不使用 PA[1:0]位。操作自动地与字地址对齐。

11.4.4 DMA 通道存储器地址寄存器（DMA_CHANNEL_CMAR）

偏移地址：0x0C

复位值：0x0000 0000

当开启通道（DMA_CHANNEL_CCR 的 EN=1）时不能写该寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MA[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MA[15:0]															
rw															

位 31:0	MA[31:0]：存储器地址（Memory address） 存储器地址作为数据传输的源或目标

	• 当 MSIZE=’01’（16 位），不使用 MA[0]位。操作自动地与半字地址对齐。 • 当 MSIZE=’10’（32 位），不使用 MA[1:0]位。操作自动地与字地址对齐。

12 模拟数字转换（ADC）

12 位 ADC 是一种逐次逼近型模拟数字转换器。它有多达 19 个通道，可测量 16 个外部和 3 个内部信号源。各通道的 A/D 转换可以单次、连续、扫描或间断模式执行。ADC 的结果可以左对齐或右对齐方式存储在 16 位数据寄存器中。

模拟看门狗特性允许应用程序检测输入电压是否超出用户定义的高/低阈值。

ADC 的输入时钟不得超过 14 MHz，它是由 PCLK2 经分频产生，见图 8-2。

12.1 ADC 主要特征

- 12 位分辨率
- 转换结束、注入转换结束和发生模拟看门狗事件时产生中断
- 单次和连续转换模式
- 从通道 0 到通道 n 的自动扫描模式
- 自校准
- 数据对齐功能
- 采样时间可以按通道独立编程
- 规则转换和注入转换均有外部触发选项
- 间断模式
- 双重模式
- ADC 供电要求：2V ~ 3.6V
- ADC 输入范围：$V_{REF-} \leq V_{IN} \leq V_{REF+}$
- 若使能了 DMA 功能，规则通道转换期间有 DMA 请求产生。

注意：如果有 V_{REF-} 引脚（取决于封装），必须和 V_{SSA} 相连接。

12.2 ADC 功能描述

下图为一个 ADC 模块的框图，下表为 ADC 引脚的说明。

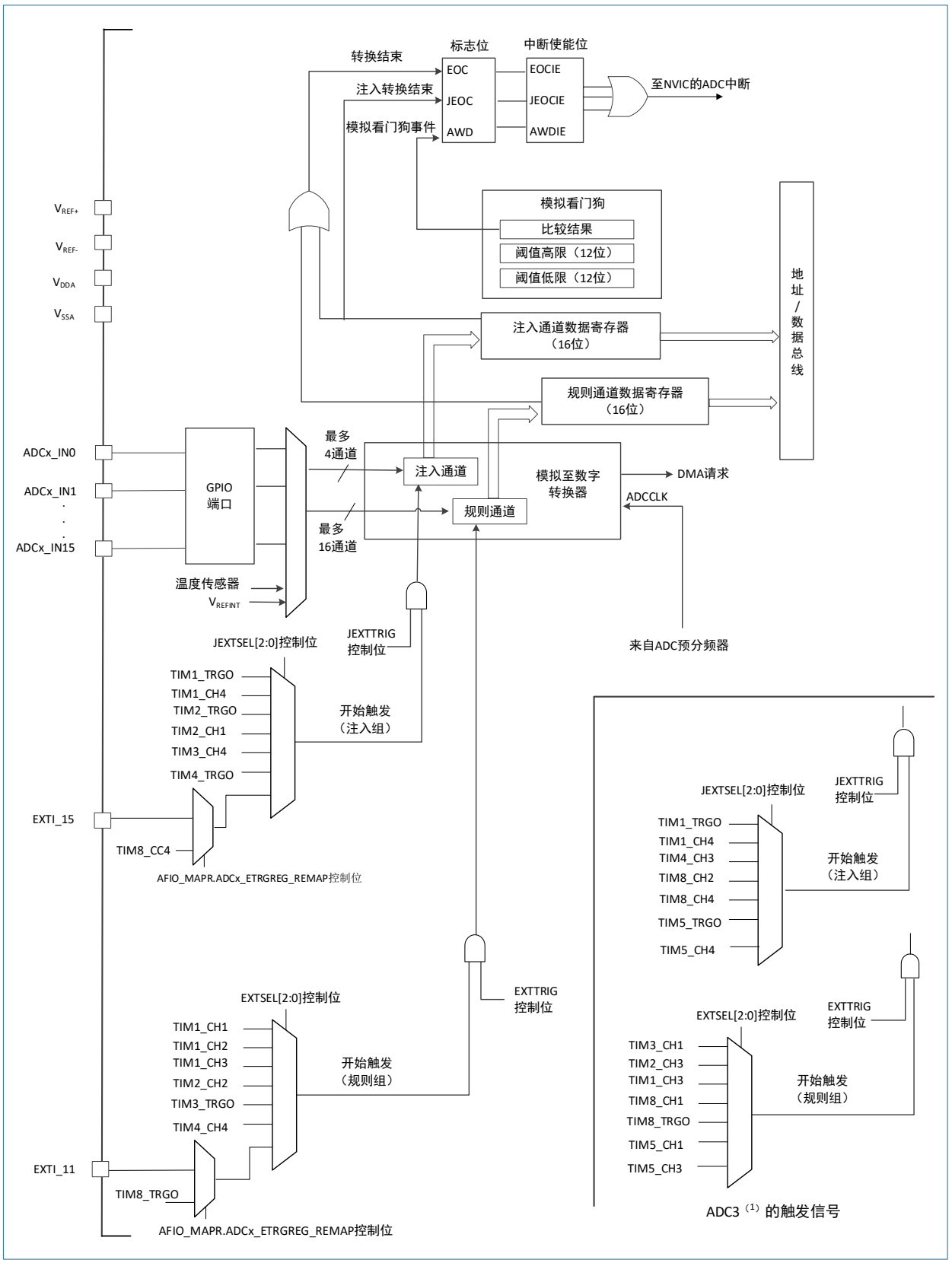


图 12-1 ADC 框图

(1). ADC3 的规则转换和注入转换触发与 ADC1 和 ADC2 的不同。

表 12-1 ADC 引脚

名称	信号类型	说明
V_{REF+}	输入，模拟参考正极	ADC 使用的高端/正极参考电压，$2V \le V_{REF+} \le V_{DDA}$

名称	信号类型	说明
V_{DDA}（1）	输入，模拟电源	等效于 V_{DD} 的模拟电源且：2V≤V_{DDA}≤V_{DD}（3.6 V）
V_{REF-}	输入，模拟参考负极	ADC 使用的低端/负极参考电压，V_{REF-}=V_{SSA}
V_{SSA}（1）	输入，模拟电源地	等效于 V_{SS} 的模拟电源地
ADCx_IN[15:0]	模拟输入信号	16 个模拟输入通道

(1). V_{DDA} 和 V_{SSA} 应该分别连接到 V_{DD} 和 V_{SS}。

ADC 通道与 PIN 脚对应表如下。

表 12-2 ADC 通道与对应的引脚

外部通道	外部引脚	ADC1 通道	ADC2 通道	ADC3 通道
通道 0	PA0	ADC1_IN0	ADC2_IN0	ADC3_IN0
通道 1	PA1	ADC1_IN1	ADC2_IN1	ADC3_IN1
通道 2	PA2	ADC1_IN2	ADC2_IN2	ADC3_IN2
通道 3	PA3	ADC1_IN3	ADC2_IN3	ADC3_IN3
通道 4	PA4	ADC1_IN4	ADC2_IN4	-
通道 5	PA5	ADC1_IN5	ADC2_IN5	-
通道 6	PA6	ADC1_IN6	ADC2_IN6	-
通道 7	PA7	ADC1_IN7	ADC2_IN7	-
通道 8	PB0	ADC1_IN8	ADC2_IN8	-
通道 9	PB1	ADC1_IN9	ADC2_IN9	-
通道 10	PC0	ADC1_IN10	ADC2_IN10	ADC3_IN10
通道 11	PC1	ADC1_IN11	ADC2_IN11	ADC3_IN11
通道 12	PC2	ADC1_IN12	ADC2_IN12	ADC3_IN12
通道 13	PC3	ADC1_IN13	ADC2_IN13	ADC3_IN13
通道 14	PC4	ADC1_IN14	ADC2_IN14	-
通道 15	PC5	ADC1_IN15	ADC2_IN15	-
通道 5’	PB3	-	-	ADC3_IN5（与 COMP3 负端共享引脚）
通道 6’	PB4	-	-	ADC3_IN6（与 COMP3 正端共享引脚）
通道 7’	PB5	-	-	ADC3_IN7（与 COMP4 负端共享引

外部通道	外部引脚	ADC1 通道	ADC2 通道	ADC3 通道
				脚）
通道 8’	PB6	-	-	ADC3_IN8 （与 COMP4 正端共享引脚）
通道 16	PB12 (仅 ADC3_IN16)	ADC1_IN16 （内部通道：连接到温度传感器）	-	ADC3_IN16 （PB12 由 V_{BATEN} 控制，经过 ADC 内部 1/2 分压到 CH16。可通过此 IO 测试外部高电压，例如 USB V_{BUS}、锂电池等。）
通道 17	PE2 (仅 ADC2_BGRBUF)	ADC1_IN17 （内部通道：连接内部电源模块 VREFINT）	ADC2_BGRBUF （经 ADC 内 BGR buffer 到通道 17，用于测量外部弱驱动的信号）	-
通道 18	PE3 (仅 ADC2_AIN18)	ADC1_IN18 （内部通道：由 V_{BATEN} 控制，经过 ADC 内部 1/2 分压到 CH18。可通过此 IO 测试外部高电压，例如 USB V_{BUS}、锂电池等。）	ADC2_AIN18	-
通道 17’	PE4	-	-	ADC3_BGRBUF （经 ADC 内 BGR buffer 到通道 17，用于测量外部弱驱动的信号）
通道 18’	PE5	-	-	ADC3_AIN18

12.2.1 ADC 开关控制

通过设置 ADC_CR2 寄存器的 ADON 位可给 ADC 上电。当第一次设置 ADON 位时，它将 ADC 从断电状态下唤醒。

ADC 上电延迟一段时间后（t_{STAB}），再次设置 ADON 位时开始进行转换。通过清除 ADON 位可以停止转换，并将 ADC 置于断电模式。在这个模式中，ADC 几乎不耗电（仅几个 μA）。

12.2.2 ADC 时钟

由时钟控制器提供的 ADCCLK 时钟和 PCLK2（APB2 时钟）同步。RCC 控制器为 ADC 时钟提供一个专用的可编程预分频器，详见章节：“8 复位和时钟控制（RCC）”。

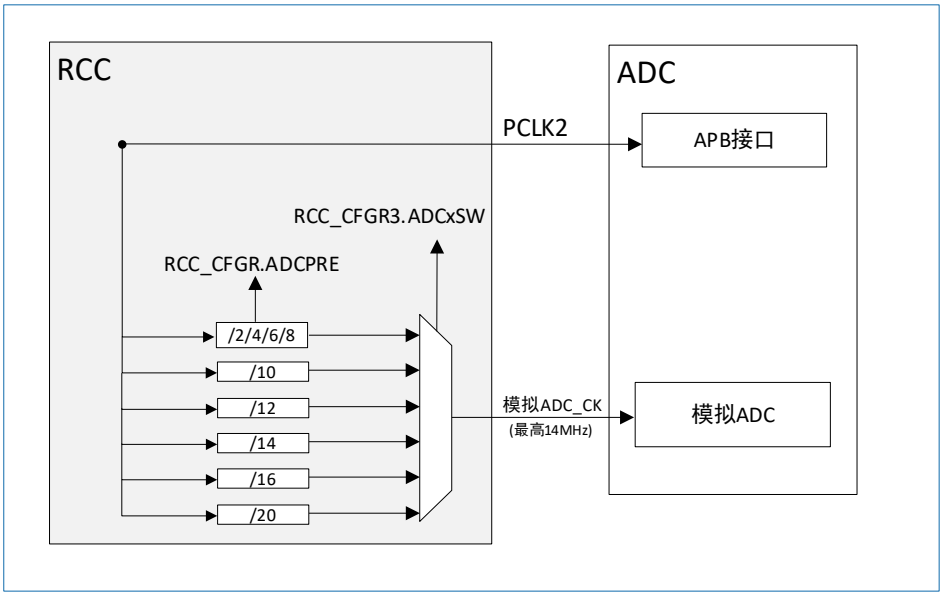


图 12-2 ADC 时钟图

12.2.3 通道选择

有 16 路复用通道。可以把转换分类成两组：规则组和注入组。在任意多个通道上以任意顺序进行的一系列转换构成组转换。例如，可以如下顺序完成转换：通道 3、通道 8、通道 2、通道 2、通道 0、通道 2、通道 2、通道 15。

- **规则组**

  由多达 16 个转换组成。规则通道和它们的转换顺序在 ADC_SQRx 寄存器中选择。规则组中转换的总数应写入 ADC_SQR1 寄存器的 L[3:0]位中。

- **注入组**

  由多达 4 个转换组成。注入通道和它们的转换顺序在 ADC_JSQR 寄存器中选择。注入组中转换的总数应写入 ADC_JSQR 寄存器的 L[1:0]位中。

如果在转换期间修改 ADC_SQRx 或 ADC_JSQR 寄存器，当前的转换则被清除，一个新的启动脉冲将发送到 ADC 以转换新选择的组。

温度传感器/V_{REFINT} 内部通道

温度传感器和通道 ADC1_IN16 相连接，内部参考电压 V_{REFINT} 和 ADC1_IN17 相连接。这两个内部通道可被选择作为注入或规则通道进行转换。

注意：温度传感器和 V_{REFINT} 只能出现在主 ADC1 中。

12.2.4 单次转换模式

单次转换模式下，ADC 只执行一次转换。通过配置 ADC_CR2.CONT=0（默认）选择单次转换模式后，可通过设置 ADC_CR2 寄存器的 ADON 位（只适用于规则通道）或外部触发（适用于规则通道或注入通道）启动单次转换。

一旦所选通道的转换完成：

- 如果转换一个规则通道：
  - 转换数据被储存在 16 位 ADC_DR 寄存器中。
  - EOC（转换结束）标志被置位。
  - 如果设置了 EOCIE，则产生中断。

- 如果转换一个注入通道：
  - 转换数据被储存在 16 位的 ADC_JDR1 寄存器中。
  - JEOC（注入转换结束）标志被置位。
  - 如果设置了 JEOCIE 位，则产生中断。

随后 ADC 停止。

12.2.5 连续转换模式

在连续转换模式中，ADC 完成一次转换后立即开始下一次转换。通过配置 ADC_CR2.CONT 位为 1 选择连续转换模式后，可通过外部触发或设置 ADC_CR2 寄存器上的 ADON 位启动转换。如需退出该模式，通过配置 ADC_CR2.ADON 为零，关闭 ADC。

每次转换后：

- 如果转换一个规则通道：
  - 转换数据被储存在 16 位的 ADC_DR 寄存器中。
  - EOC（转换结束）标志被置位。
  - 如果设置了 EOCIE，则产生中断。
- 如果转换一个注入通道：
  - 转换数据被储存在 16 位的 ADC_JDR1 寄存器中。
  - JEOC（注入转换结束）标志被置位。
  - 如果设置了 JEOCIE 位，则产生中断。

12.2.6 时序图

如下图所示，ADC 在开始精确转换前需要一个稳定时间 t_{STAB}。在开始 ADC 转换和 14 个时钟周期后，EOC 标志被置位，16 位 ADC 数据寄存器包含转换的结果。

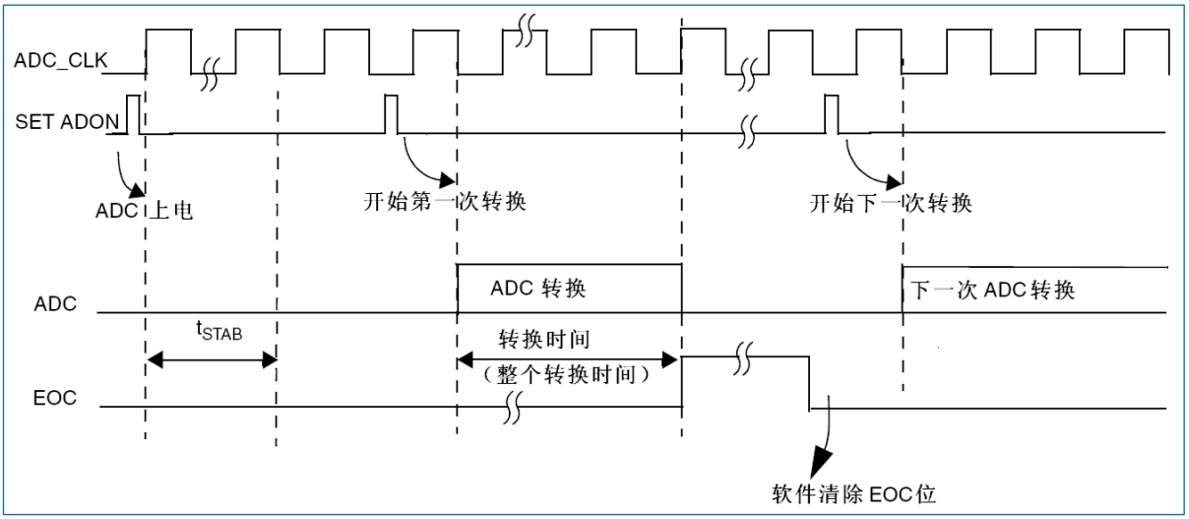


图 12-3 时序图

12.2.7 模拟看门狗

如果被 ADC 转换的模拟电压小于低阈值或大于高阈值，AWD 模拟看门狗的状态位则被置位。阈值位于 16 位的 ADC_HTR 和 ADC_LTR 寄存器的最低 12 个有效位中。通过设置 ADC_CR1 寄存器的 AWDIE 位以允许产生相应中断或唤醒系统（详细的操作请结合 BKP、RTC 和 PWR 相关的寄存器）。

阈值独立于由 ADC_CR2 寄存器上的 ALIGN 位选择的数据对齐模式。比较是在对齐之前完成的（见

“12.4 数据对齐”）。通过配置 ADC_CR1 寄存器，模拟看门狗可以作用于 1 个或多个通道，如表 12-3 所示。

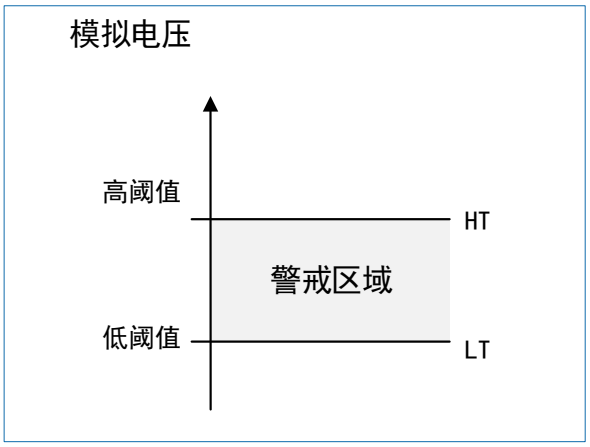


图 12-4 模拟看门狗警戒区

表 12-3 模拟看门狗通道选择

模拟看门狗监视的通道	ADC_CR1 寄存器控制位		
	AWDSGL 位	AWDEN 位	JAWDEN 位
无	任意值	0	0
所有注入通道	0	0	1
所有规则通道	0	1	0
所有注入和规则通道	0	1	1
单个 (1) 注入通道	1	0	1
单个 (1) 规则通道	1	1	0
单个的 (1) 注入或规则通道	1	1	1

(1). 由 AWDCH[4:0]位选择。

12.2.8 扫描模式

此模式用于扫描一组模拟通道。

通过设置 ADC_CR1 寄存器的 SCAN 位来选择扫描模式。一旦设置该位，ADC 扫描被 ADC_SQRx 寄存器（对规则通道）或 ADC_JSQR 寄存器（对注入通道）选中的所有通道。在每个组的每个通道上执行单次转换。在每个转换结束时，同一组的下一个通道被自动转换。如果设置了 CONT 位，转换不会在选择组的最后一个通道上停止，而是再次从所选择组的第一个通道继续转换。

在扫描模式下，必须设置 ADC_CR2 的 DMA 位。每次规则组的通道转换结束，即 ADC_DR 寄存器被更新时，DMA 控制器会把转换结果从 ADC_DR 寄存器搬到 SRAM 中。

而注入通道转换的数据总是存储在 ADC_JDRx 寄存器中。

12.2.9 注入通道管理

触发注入

清除 ADC_CR1 寄存器的 JAUTO 位且设置 SCAN 位，即可使用触发注入功能。

1. 利用外部触发或通过设置 ADC_CR2 寄存器的 ADON 位，启动一组规则通道的转换。

2. 如果在规则组通道转换期间产生一次外部注入触发，当前转换被复位，注入通道序列以单次扫描方式进行转换。

3. 上次发生中断的规则组通道转换被恢复。如果在注入转换期间产生一次规则事件，注入转换不会被中断，但是规则序列将在注入序列结束后被执行。图 12-5 示出了时序图。

说明：当使用触发的注入转换时，必须保证触发事件的间隔长于注入序列。例如：序列长度为 28 个 ADC 时钟周期（两次转换需间隔 1.5 个时钟周期采样时间），触发之间最小的间隔必须是 29 个 ADC 时钟周期。

自动注入

如果设置了 JAUTO 位，注入组通道在规则组通道之后被自动转换。这可以用来转换在 ADC_SQRx 和 ADC_JSQR 寄存器中设置的最多 20 个转换序列。在此模式里，必须禁止注入通道的外部触发。如果除 JAUTO 位外还设置了 CONT 位，规则通道至注入通道的转换序列被连续执行。

对于 ADC 时钟预分频系数为 4 至 8 时，当从规则转换切换到注入序列（或从注入转换切换到规则序列）时，会自动插入 1 个 ADC 时钟间隔；当 ADC 时钟预分频系数为 2 时，则自动插入 2 个 ADC 时钟间隔。

注意：不能同时使用自动注入和间断模式。

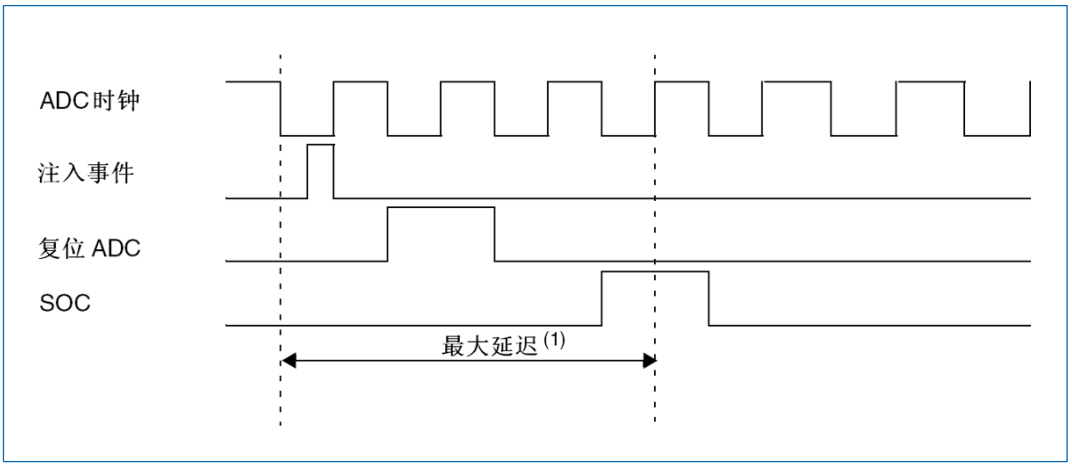


图 12-5 注入转换延时

图 12-5 中的最大延迟数值，可参考器件数据手册中有关电气特性部分。

12.2.10 间断模式

规则组

通过设置 ADC_CR1 寄存器上的 DISCEN 位使能此模式。它可以用来执行一个短序列的 n 次转换（n≤8），此转换是 ADC_SQRx 寄存器所选择的转换序列的一部分。数值 n 由 ADC_CR1 寄存器的 DISCNUM[2:0]位给出。

一个外部触发信号可以启动 ADC_SQRx 寄存器中描述的下一轮 n 次转换，直到此序列所有的转换完成为止。总的序列长度由 ADC_SQR1 寄存器的 L[3:0]定义。

举例：若 n=3，被转换的通道=0、1、2、3、6、7、9、10，则：

1. 第一次触发：转换的序列为 0、1、2

2. 第二次触发：转换的序列为 3、6、7

3. 第三次触发：转换的序列为 9、10，并产生 EOC 事件

4. 第四次触发：转换的序列 0、1、2

注意：当以间断模式转换一个规则组时，转换序列结束后不自动从头开始。当所有子组被转换完成，下一次触发启动第一个子组的转换。在上面的例子中，第四次触发重新转换第一子组的通道 0、1 和 2。

注入组

通过设置 ADC_CR1 寄存器的 JDISCEN 位使能此模式。在一个外部触发事件后，该模式按通道顺序逐个转换 ADC_JSQR 寄存器中选择的序列。

一个外部触发信号可以启动 ADC_JSQR 寄存器选择的下一个通道序列的转换，直到序列中所有的转换完成为止。总的序列长度由 ADC_JSQR 寄存器的 JL[1:0]位定义。

例子：若 n=1，被转换的通道=1、2、3，则：

1. 第一次触发：通道 1 被转换
2. 第二次触发：通道 2 被转换
3. 第三次触发：通道 3 被转换，并且产生 EOC 和 JEOC 事件
4. 第四次触发：通道 1 被转换

注意：

当完成所有注入通道转换，下个触发启动第 1 个注入通道的转换。在上述例子中，第四个触发重新转换第 1 个注入通道 1。

不能同时使用自动注入和间断模式。

必须避免同时为规则和注入组设置间断模式。间断模式只能作用于一组转换。

12.3 校准

ADC 有一个内置自校准模式。校准可显著减小因内部电容器组的变化而造成的精度误差。在校准期间，在每个电容器上都会计算出一个误差修正码（数字值），这个码用于消除在随后的转换中每个电容器上产生的误差。

通过设置 ADC_CR2 寄存器的 CAL 位启动校准。一旦校准结束，CAL 位被硬件复位，可以开始正常转换。建议在上电时执行一次 ADC 校准。校准阶段结束后，校准码储存在 ADC_DR 中。

注意：

建议在每次上电后执行一次校准。

启动校准前，ADC 必须处于开电状态（ADON='1'）超过至少两个 ADC 时钟周期。

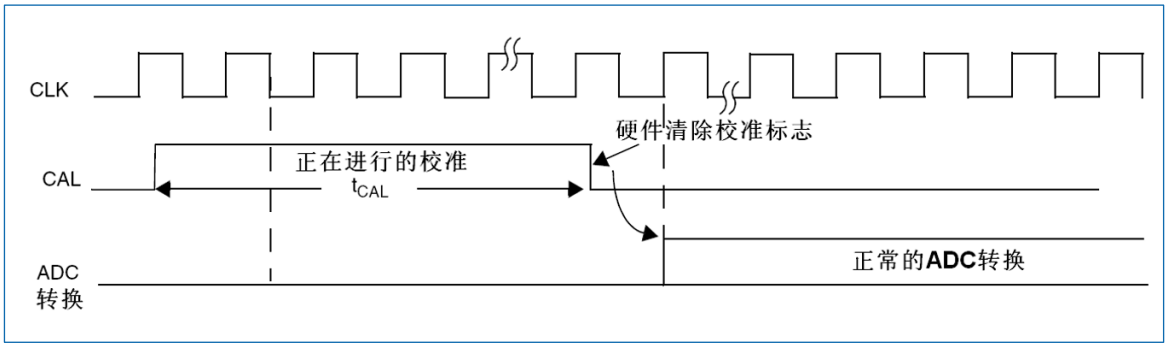


图 12-6 校准时序图

12.4 数据对齐

ADC_CR2 寄存器中的 ALIGN 位选择转换后数据储存的对齐方式。数据可以左对齐或右对齐，如图 12-7 和图 12-8 所示。

注入组通道转换的数据值已经减去了在 ADC_JOFRx 寄存器中定义的偏移量，因此结果可以是一个负值。SEXT 位是扩展的符号值。

对于规则组通道，不需减去偏移值，因此只有 12 个位有效。

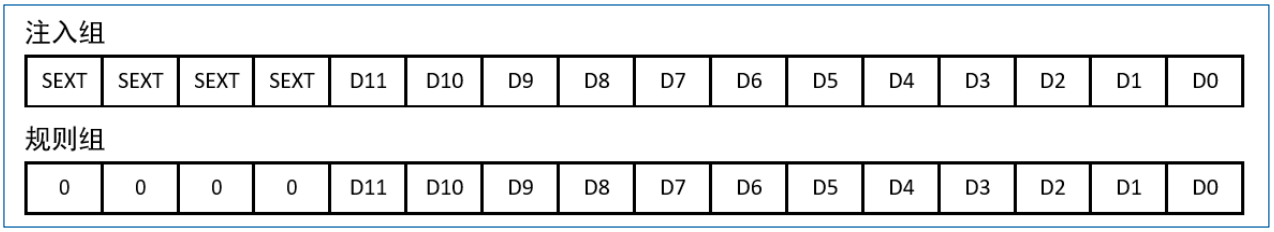


图 12-7 数据右对齐

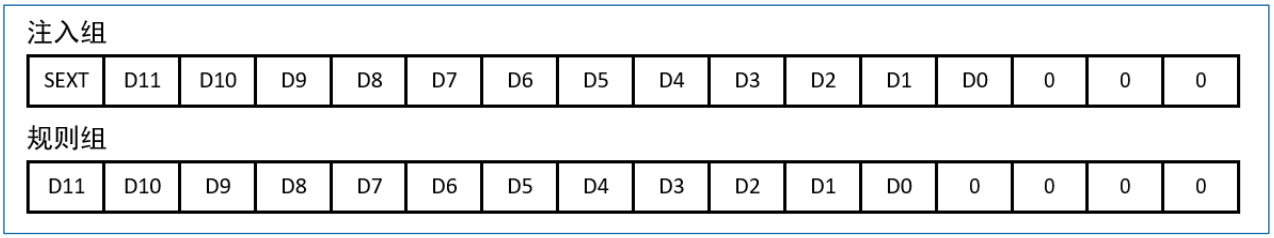


图 12-8 数据左对齐

12.5 可编程的通道采样时间

ADC 使用若干个 ADC_CLK 周期对输入电压采样，采样周期数目可以通过 ADC_SMPR1 和 ADC_SMPR2 寄存器中的 SMPx[2:0]位更改。每个通道可以分别用不同的时间采样。

总转换时间如下计算：

T_{CONV}=采样时间+12.5 个周期

例如：

当 ADCCLK=14 MHz，采样时间为 1.5 周期时：T_{CONV}=1.5+12.5=14 个周期=1 μs

12.6 外部触发转换

转换可以由外部事件触发（例如定时器捕获，EXTI 线）。如果设置了 EXTTRIG 控制位，则外部事件就能够触发转换。EXTSEL[2:0]和 JEXTSEL[2:0]控制位允许应用程序选择 8 个可能的事件中的某一个，以触发规则和注入组的采样。

注意：当外部触发信号被选为 ADC 规则或注入转换时，只有它的上升沿可以启动转换。

表 12-4 ADC1 和 ADC2 用于规则通道的外部触发

触发源	类型	EXTSEL[2:0]
TIM1_CC1 事件	来自片上定时器的内部信号	000
TIM1_CC2 事件		001
TIM1_CC3 事件		010
TIM2_CC2 事件		011

触发源	类型	EXTSEL[2:0]
TIM3_TRGO 事件		100
TIM4_CC4 事件		101
EXTI 线 11/TIM8_TRGO 事件	外部引脚/来自片上定时器的内部信号	110
SWSTART	软件控制位	111

表 12-5 ADC1 和 ADC2 用于注入通道的外部触发

触发源	连接类型	JEXTSEL[2:0]
TIM1_TRGO 事件	来自片上定时器的内部信号	000
TIM1_CC4 事件		001
TIM2_TRGO 事件		010
TIM2_CC1 事件		011
TIM3_CC4 事件		100
TIM4_TRGO 事件		101
EXTI 线 15/TIM8_CC4 事件	外部引脚/来自片上定时器的内部信号	110
JSWSTART	软件控制位	111

表 12-6 ADC3 用于规则通道的外部触发

触发源	连接类型	EXTSEL[2:0]
TIM3_CC1 事件	来自片上定时器的内部信号	000
TIM2_CC3 事件		001
TIM1_CC3 事件		010
TIM8_CC1 事件		011
TIM8_TRGO 事件		100
TIM5_CC1 事件		101
TIM5_CC3 事件		110
SWSTART	软件控制位	111

表 12-7 ADC3 用于注入通道的外部触发

触发源	连接类型	EXTSEL[2:0]
TIM1_TRGO 事件	来自片上定时器的内部信号	000
TIM1_CC4 事件		001
TIM4_CC3 事件		010

触发源	连接类型	EXTSEL[2:0]
TIM8_CC2 事件		011
TIM8_CC4 事件		100
TIM5_TRGO 事件		101
TIM5_CC4 事件		110
JSWSTART	软件控制位	111

软件触发事件可以通过对寄存器 ADC_CR2 的 SWSTART 或 JSWSTART 位置'1'产生。注入触发可中断规则组的转换。

12.7 DMA 请求

因为规则通道已转换的值储存在一个仅有的数据寄存器中，所以当转换多个规则通道时需要使用 DMA。这可以避免已存储在 ADC_DR 寄存器中的数据丢失。

只有在规则通道的转换结束时才产生 DMA 请求，并将转换的数据从 ADC_DR 寄存器传输到用户指定的目的地址。

说明：只有ADC1 和ADC3 拥有DMA 功能。由ADC2 转化的数据可以通过双ADC 模式，利用ADC1 的DMA 功能传输。

12.8 双 ADC 模式

器件具有超过 2 个的 ADC 模块，支持双 ADC 模式（见图 12-9），ADC1（主），ADC2（从）；ADC3 不支持双 ADC 模式。在双 ADC 模式里，根据 ADC1_CR1 寄存器中 DUALMOD[3:0]位所选的模式，转换的启动可由 ADC1 主到 ADC2 从交替触发或同步触发。

注意：在双 ADC 模式里，当转换配置成由外部事件触发时，用户必须将其设置成仅触发主 ADC，从 ADC 设置成软件触发，这样可以防止意外触发从转换。但是，主和从 ADC 的外部触发必须同时使能。

共有 6 种可能的模式：

- 同步注入模式
- 同步规则模式
- 快速交替模式
- 慢速交替模式
- 交替触发模式
- 独立模式

还有可以用下列方式组合使用上面的模式：

- 同步注入模式+同步规则模式
- 同步规则模式+交替触发模式
- 同步注入模式+交替模式

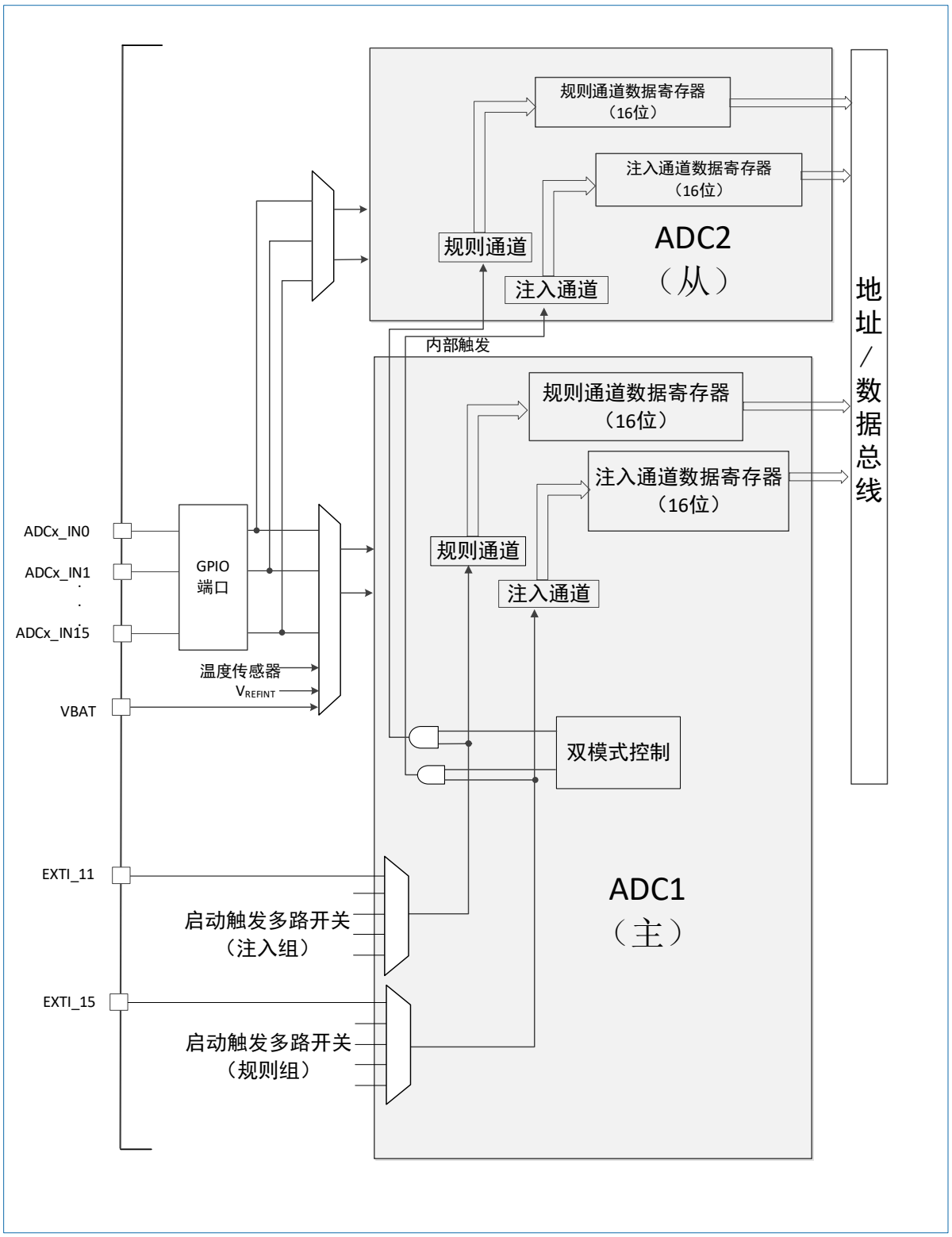


图 12-9 双 ADC 框图

上图说明：

(1). 外部触发信号作用于 ADC2，但在上图中没有显示。

(2). 在某些双 ADC 模式中，在 ADC1（主）的数据寄存器（ADC1_DR）中包含了 ADC1 和 ADC2 的规则转换数据。

12.8.1 注入同步模式

此模式转换一个注入通道组。外部触发来自 ADC1 的注入组多路开关（由 ADC1_CR2 寄存器的 JEXTSEL[2:0]选择），它同时给 ADC2 提供同步触发。

注意：不要在 2 个 ADC 上转换相同的通道（2 个 ADC 在同一个通道上的采样时间不能重叠）。

在 ADC1 或 ADC2 的转换结束时：

- 转换的数据存储在每个 ADC 接口的 ADC_JDRx 寄存器中。
- 当所有 ADC1/ADC2 注入通道都被转换完时，产生 JEOC 中断（若任一 ADC 接口使能了中断）。

说明：在同步注入模式中，在 ADC1 和 ADC2 上同时采样的两个通道必须设置同样的采样时间，来保证两个 ADC 的同步。

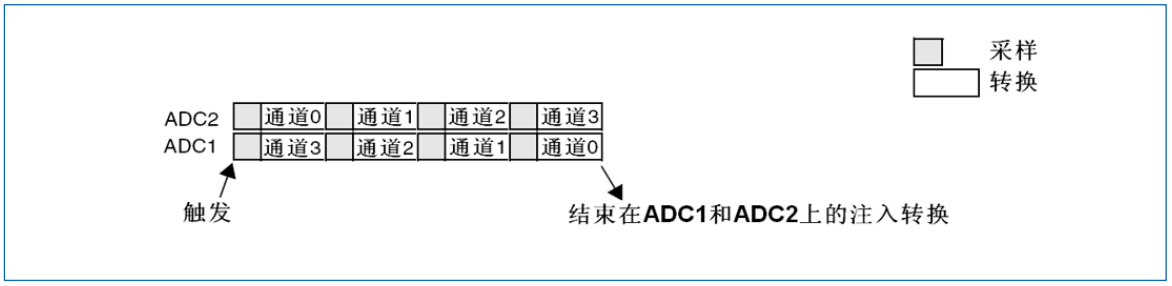


图 12-10 在 4 个通道上的同步注入模式

12.8.2 规则同步模式

此模式在规则通道组上执行。外部触发来自 ADC1 的规则组多路开关（由 ADC1_CR2 寄存器的 EXTSEL[2:0]选择）。它同时给 ADC2 提供同步触发。

注意：不要在 2 个 ADC 上转换相同的通道（两个 ADC 在同一个通道上的采样时间不能重叠）。

在 ADC1 或 ADC2 的转换结束时：

- 如果设置了 DMA 位，则产生一个 32 位 DMA 传输请求，32 位的 ADC_DR 寄存器内容传输到 SRAM 中，它高半个字包含 ADC2 的转换数据，低半个字包含 ADC1 的转换数据。
- 当所有 ADC1/ADC2 规则通道都被转换完时，产生 EOC 中断（若任一 ADC 接口使能了中断）。

说明：在同步规则模式中，在 ADC1 和 ADC2 上同时采样的两个通道必须设置同样的采样时间，来保证两个 ADC 的同步。

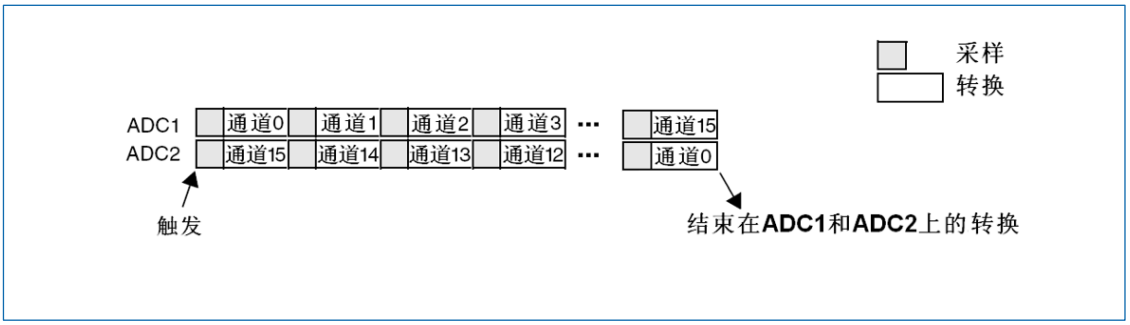


图 12-11 16 个通道上的同步规则模式

12.8.3 快速交替模式

此模式只适用于规则通道组（通常为一个通道）。外部触发来自 ADC1 的规则通道多路开关。外部触发产生后：

- ADC2 立即启动；
- ADC1 在延迟 7 个 ADC 时钟周期后启动。

如果同时设置了 ADC1 和 ADC2 的 CONT 位，ADC1 和 ADC2 的所选择的规则通道将被连续地转换。ADC1 产生一个 EOC 中断后（若 EOCIE 位使能），产生一个 32 位的 DMA 传输请求（如果设置了 DMA 位），

ADC_DR 寄存器的 32 位数据被传输到 SRAM。ADC_DR 的高半个字包含 ADC2 的转换数据，低半个字包含 ADC1 的转换数据。

注意：最大允许采样时间<7 个 ADCCLK 周期，避免 ADC1 和 ADC2 转换相同通道时发生两个采样周期的重叠。

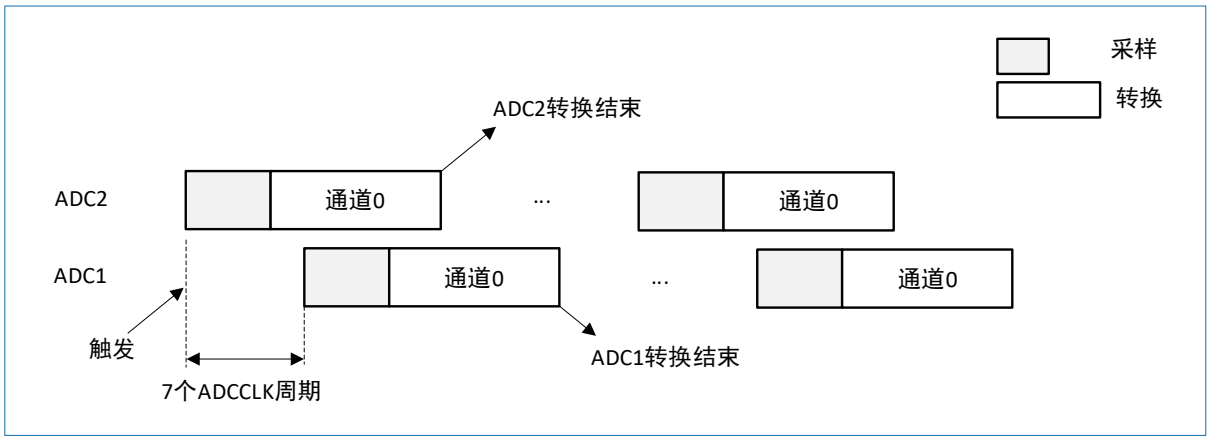


图 12-12 在 1 个通道上连续转换模式下的快速交替模式

12.8.4 慢速交替模式

此模式只适用于规则通道组（只能为一个通道）。外部触发来自 ADC1 的规则通道多路开关。外部触发产生后：

- ADC2 立即启动；
- ADC1 在延迟 14 个 ADC 时钟周期后启动；
- 在延迟第二次 14 个 ADC 周期后 ADC2 再次启动，按此循环。

注意：最大允许采样时间<14 个 ADCCLK 周期，以避免和下个转换重叠。

ADC1 产生一个 EOC 中断后（若 EOCIE 位使能），产生一个 32 位的 DMA 传输请求（如果设置了 DMA 位），ADC_DR 寄存器的 32 位数据被传输到 SRAM，ADC_DR 的高半个字包含 ADC2 的转换数据，低半个字包含 ADC1 的转换数据。

在 28 个 ADC 时钟周期后自动启动第二次 ADC2 转换。在这个模式下不能设置 CONT 位，因为它将连续转换所选择的规则通道。

注意：应用程序必须确保当使用交替模式时，不能有注入通道的外部触发产生。

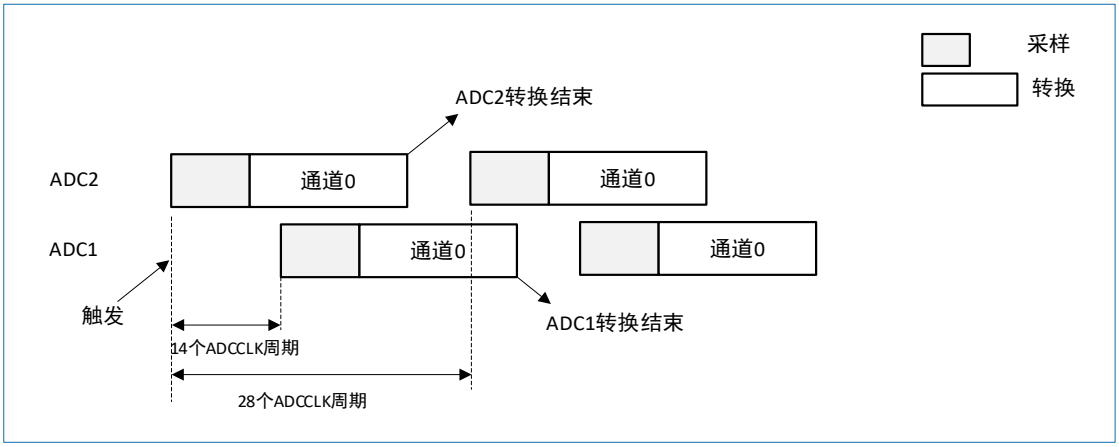


图 12-13 在 1 个通道上的慢速交替模式

12.8.5 交替触发模式

此模式只适用于注入通道组。外部触发源来自 ADC1 的注入通道多路开关。

- 当第一个触发产生时，ADC1 上的所有注入组通道被转换；
- 当第二个触发到达时，ADC2 上的所有注入组通道被转换；
- 依此循环…

如果使能 JEOC 中断，在所有 ADC1 注入组通道转换后产生一个 JEOC 中断。如果使能 JEOC 中断，在所有 ADC2 注入组通道转换后产生一个 JEOC 中断。当所有注入组通道都转换完后，如果又有另一个外部触发，交替触发从转换 ADC1 注入组通道重新开始。

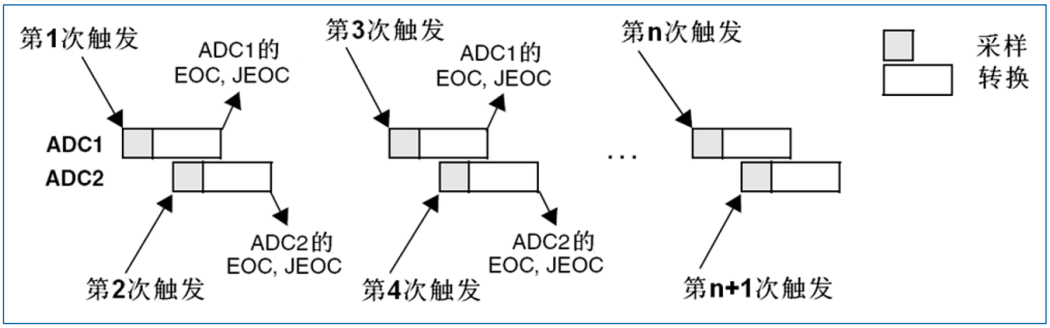


图 12-14 交替触发：每个 ADC1 的注入通道组

如果 ADC1 和 ADC2 上同时使用了注入间断模式：

- 当第一个触发产生时，ADC1 上的第一个注入通道被转换；
- 当第二个触发到达时，ADC2 上的第一个注入通道被转换；
- 依此循环…

如果允许产生 JEOC 中断，在所有 ADC1 注入组通道转换后产生一个 JEOC 中断。如果允许产生 JEOC 中断，在所有 ADC2 注入组通道转换后产生一个 JEOC 中断。当所有注入组通道都转换完后，如果又有另一个外部触发，则重新开始交替触发过程。

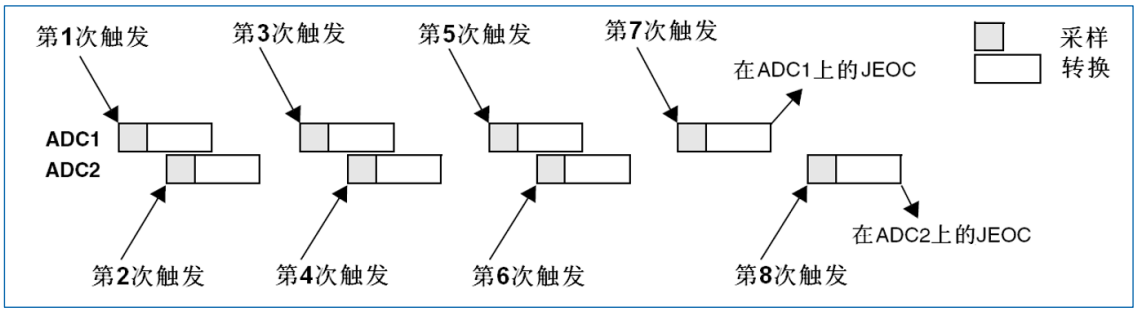


图 12-15 交替触发：在间断模式下每个 ADC 上的 4 个注入通道

12.8.6 独立模式

此模式里，双 ADC 的同步被旁路，每个 ADC 接口独立工作。（此模式为默认工作模式）

12.8.7 混合的规则/注入同步模式

规则组同步转换可以被中断，以启动注入组的同步转换。

说明：在混合的规则/注入同步模式中，在 ADC1 和 ADC2 上同时采样的两个通道必须设置同样的采样时间，来保证两个 ADC 的同步。

12.8.8 混合的同步规则+交替触发模式

规则组同步转换可以被中断，以启动注入组交替触发转换。图 12-16 显示了一个规则同步转换被交替触发所中断。

注入交替转换在注入事件到达后立即启动。如果规则转换正在进行，为了在注入转换后确保同步，所有的 ADC（主和从）的规则转换被停止，并在注入转换结束时同步恢复。

说明：在混合的同步规则+交替触发模式中，在 ADC1 和 ADC2 上同时采样的两个通道必须设置同样的采样时间，来保证两个 ADC 的同步。

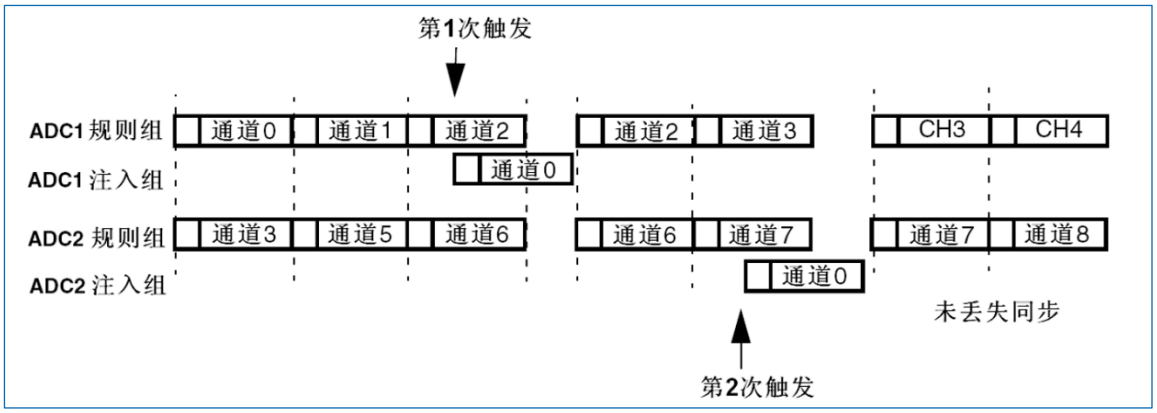


图 12-16 交替+规则同步

如果触发事件发生在一个中断了规则转换的注入转换期间，这个触发事件将被忽略。图 12-17 示出了这种情况的操作（第 2 个触发被忽略）。

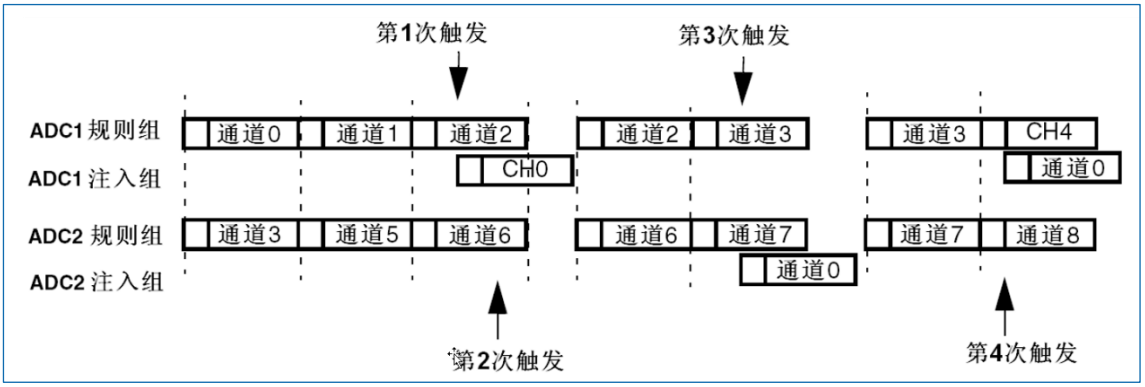


图 12-17 触发事件发生在注入转换期间

12.8.9 混合同步注入+交替模式

一个注入事件可以中断一个交替转换。这种情况下，交替转换被中断，注入转换被启动，在注入序列转换结束时，交替转换被恢复。图 12-18 是这种情况的一个例子。

说明：当 ADC 时钟预分频系数设置为 4 时，交替模式恢复后不会均匀地分配采样时间，比如：采样间隔是 8 个 ADC 时钟周期与 6 个 ADC 时钟周期轮替，而不是均匀的 7 个 ADC 时钟周期。

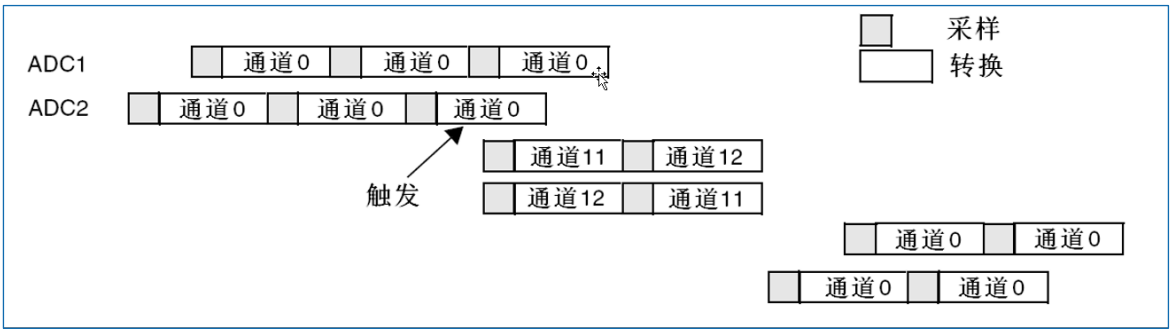


图 12-18 交替的单通道转换被注入序列 CH11 和 CH12 中断

12.9 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）。温度传感器在内部和 ADC1_IN16 输入通道相连接，此通道把传感器输出的电压转换成数字值。温度传感器的推荐采样时间是 17.1μs。

温度传感器的输出电压随温度线性变化。由于工艺不同，该线的偏移量取决于各个芯片（芯片之间的温度变化可达 45°C）。

未校准的内部温度传感器更适用于对温度变量而非绝对温度进行测量的应用。为提高温度传感器测量的准确性，航顺在生产过程中将校准值存储在每个器件的系统存储器中，见下表。访问模式为只读。用户应用可读取这些数据，并使用这些数据提高温度传感器的准确性。

表 12-8 温度传感器校准值

校准值名称	描述	存储器地址
TS_CAL1	在 25°C 温度下获得的 ADC 原始数据，V_{DDA} = 3.3V	0x1FFF F7B8-0x1FFF F7B9

图 12-19 是温度传感器的方框图。当没有被使用时，传感器可以置于关电模式。

注意：必须设置TSVREFE 位激活内部通道：ADC1_IN16（温度传感器）和ADC1_IN17（VREFINT）的转换。

主要特性

- 支持的温度范围：-40°C 到 105°C
- 线性度：最高±2°C，精度取决于校准情况。

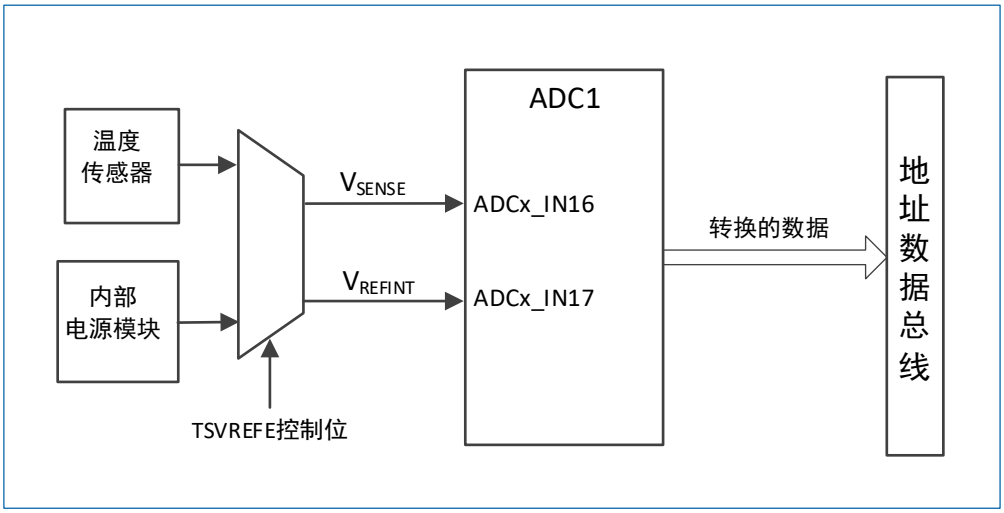


图 12-19 温度传感器和 V_{REFINT} 通道框图

读温度

使用温度传感器的步骤：

1. 选择 ADC1_IN16 输入通道。
2. 选择采样时间为 17.1 μs。
3. 设置 ADC 控制寄存器 2（ADC_CR2）的 TSVREFE 位，以唤醒关电模式下的温度传感器。
4. 通过设置 ADON 位启动 ADC 转换（或用外部触发）。
5. 读 ADC 数据寄存器上的数据结果 VSENSE。
6. 利用下列公式得出温度：

温度(℃) = {[(TS_CAL1-VSENSE)/4095 * 3300]/Avg_Slope} + 25

其中：

- TS_CAL1 是在 25°C 温度下温度传感器的校准值，详见表 12-8。
- VSENSE 是上述步骤 5 中获取的值。
- Avg_Slope 为 VSENSE 和温度曲线的平均斜率值（单位为mV/℃），参见数据手册电气特性章节。

注意：传感器从关电模式唤醒后到可以输出正确电平的 VSENSE 前，有一个建立时间。ADC 在上电后也有一个建立时间，因此为了缩短延时，应该同时设置 ADON 和 TSVREFE 位。

12.10 内部参考电压

内部参考电压（V_{REFINT}）为 ADC 提供了一个稳定的（带隙基准）电压输出。V_{REFINT} 内部连接到 ADC_IN9 输入通道。图 12-19 给出了内部参考电压通道的框图。

它可以精确地监视 V_{DDA} 值（当 ADC 没有外部电压 V_{REF+}时）。V_{REFINT} 的精确电压在生产测试期间由航顺测量，并存储在系统内存区域，见下表。访问模式为只读。用户应用可读取并使用这些数据提高内部参考电压的准确性。

表 12-9 内部电压基准测量值

校准值名称	描述	存储器地址
V_{REFINT_CAL}	在 25° C 温度下获得的原始数据，V_{DDA} = 3.3 V	0x1FFF F7BA-0x1FFF F7BB

使用内部参考电压计算实际的 V_{DDA} 电压

施加给 MCU 的 V_{DDA} 电源电压可能会有变化，或无法获得准确值。在制造过程中由 ADC 在 V_{DDA}=3.3V 的条件下获得的内置内部参考电压（V_{REFINT}）及其校准数据可用于评估实际的 V_{DDA} 电压水平。

以下公式可求得为器件供电的实际的 V_{DDA} 电压：

$$\frac{V_{REFINT\_CAL}}{4096} * 3.3V = \frac{V_{REFINT\_S}}{4096} * V_{DDA}$$

由上述公式可得：

$$V_{DDA} = \frac{3.3 \times V_{REFINT\_CAL}}{V_{REFINT\_S}}$$

其中：

- V_{REFINT_CAL} 的值参见表 12-9。
- V_{REFINT_S} 表示内部参考电压的实际采样值。

12.11 ADC 中断

当模拟看门狗状态位被置位，或者规则和注入组转换结束时，能产生中断。它们都有独立的中断使能位。

说明：ADC1 和 ADC2 的中断映射在同一个中断向量上，而 ADC3 的中断有自己的中断向量。

ADC_SR 寄存器中有 2 个其他标志，但是它们没有相关联的中断：

- JSTRT（注入组通道转换的启动）
- STRT（规则组通道转换的启动）

表 12-10 ADC 中断

中断事件	事件标志	使能控制位
规则组转换结束	EOC	EOCIE
注入组转换结束	JEOC	JEOCIE
设置了模拟看门狗状态位	AWD	AWDIE

12.12 ADC 寄存器

基地址：（ADC1, ADC2, ADC3）=（0x4001 2400, 0x4001 2800, 0x4001 3C00）

空间大小：（ADC1, ADC2, ADC3）=（0x400, 0x400, 0x400）

ADC 寄存器必须以字（32 位）的方式进行访问。

12.12.1 ADC 状态寄存器（ADC_SR）

偏移地址：0x00

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res											STRT	JSTRT	JEOC	EOC	AWD
											rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

位 31: 5	Res：保留 必须保持复位值。
位 4	STRT：规则通道开始位（Regular channel Start flag） 该位由硬件在规则通道转换开始时设置，由软件清除。 • 0：规则通道转换未开始 • 1：规则通道转换已开始
位 3	JSTRT：注入通道开始位（Injected channel Start flag） 该位由硬件在注入通道组转换开始时设置，由软件清除。 • 0：注入通道组转换未开始 • 1：注入通道组转换已开始
位 2	JEOC：注入通道转换结束位（Injected channel end of conversion） 该位由硬件在所有注入通道组转换结束时设置，由软件清除。 • 0：转换未完成 • 1：转换完成
位 1	EOC：转换结束位（End of conversion） 该位由硬件在（规则或注入）通道组转换结束时设置。

	该位由软件或由读取 ADC_DR 时清除。 • 0：转换未完成 • 1：转换完成
位 0	AWD：模拟看门狗标志位（Analog watchdog flag） 该位由硬件在转换的电压值超出了 ADC_LTR 和 ADC_HTR 寄存器定义的范围时设置，由软件清除。 • 0：没有发生模拟看门狗事件 • 1：发生模拟看门狗事件

12.12.2 ADC 控制寄存器 1（ADC_CR1）

偏移地址：0x04

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res								AWDEN	JAWDEN	Res		DUALMOD[3:0]			
								rw	rw			rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DISCNUM[2:0]			JDISCEN	DISCEN	JAUTO	AWDSGL	SCAN	JEOCIE	AWDIE	EOCIE	AWDCH[4:0]				
rw			rw	rw	rw	rw	rw	rw	rw	rw	rw				

位 31:24	Res：保留 必须保持复位值。
位 23	AWDEN：在规则通道上开启模拟看门狗（Analog watchdog enable on regular channels） 该位由软件设置和清除。 • 0：在规则通道上禁用模拟看门狗 • 1：在规则通道上使能模拟看门狗
位 22	JAWDEN：在注入通道上开启模拟看门狗（Analog watchdog enable on injected channels） 该位由软件设置和清除。 • 0：在注入通道上禁用模拟看门狗 • 1：在注入通道上使能模拟看门狗
位 21:20	Res：保留 必须保持复位值。
位 19:16	DUALMOD[3:0]：双模式选择（Dual mode selection） 软件使用这些位选择操作模式。 • 0000：独立模式 • 0001：混合的同步规则+注入同步模式 • 0010：混合的同步规则+交替触发模式 • 0011：混合同步注入+快速交替模式 • 0100：混合同步注入+慢速交替模式 • 0101：注入同步模式 • 0110：规则同步模式 • 0111：快速交替模式 • 1000：慢速交替模式 • 1001：交替触发模式

	说明：在 ADC2 和 ADC3 中这些位为保留位。 *在双模式中，改变通道的配置会产生一个重新开始的条件，这将导致同步丢失。建议在进行任何配置改变前关闭双模式。*
位 15:13	DISCNUM[2:0]：间断模式通道计数（Discontinuous mode channel count） 软件通过这些位定义在间断模式下，收到外部触发后转换规则通道的数目。 • 000：1 个通道 • 001：2 个通道 • … • 111：8 个通道
位 12	JDISCEN：在注入通道上的间断模式（Discontinuous mode on injected channels） 该位由软件设置和清除，用于开启或关闭注入通道组上的间断模式。 • 0：注入通道组上禁用间断模式 • 1：注入通道组上使能间断模式
位 11	DISCEN：在规则通道上的间断模式（Discontinuous mode on regular channels） 该位由软件设置和清除，用于开启或关闭规则通道组上的间断模式。 • 0：规则通道组上禁用间断模式 • 1：规则通道组上使能间断模式
位 10	JAUTO：自动的注入通道组转换（Automatic Injected Group conversion） 该位由软件设置和清除，用于开启或关闭规则通道组转换结束后注入通道组的自动转换。 • 0：关闭自动的注入通道组转换 • 1：开启自动的注入通道组转换
位 9	AWDSGL：扫描模式中在一个单一的通道上使用看门狗（Enable the watchdog on a single channel in scan mode） 该位由软件设置和清除，用于开启或关闭由 AWDCH[4:0]位指定的通道上的模拟看门狗功能。 • 0：在所有的通道上使用模拟看门狗 • 1：在单一通道上使能模拟看门狗
位 8	SCAN：扫描模式（Scan mode） 该位由软件设置和清除，用于开启或关闭扫描模式。在扫描模式中，转换由 ADC_SQRx 或 ADC_JSQRx 寄存器选中的通道。 • 0：关闭扫描模式 • 1：使用扫描模式 *说明：如果分别设置了 EOCIE 或 JEOCIE 位，只在最后一个通道转换完毕后才会产生 EOC 或 JEOC 中断。*
位 7	JEOCIE：允许产生注入通道转换结束中断（Interrupt enable for injected channels） 该位由软件设置和清除，用于禁止或允许所有注入通道转换结束后产生中断。 • 0：禁止 JEOC 中断 • 1：允许 JEOC 中断。当硬件设置 JEOC 位时产生中断。
位 6	AWDIE：允许产生模拟看门狗中断（Analog watchdog interrupt enable） 该位由软件设置和清除，用于禁止或允许模拟看门狗产生中断。 • 0：禁止模拟看门狗中断 • 1：允许模拟看门狗中断

位 5	EOCIE：允许产生 EOC 中断（Interrupt enable for EOC） 该位由软件设置和清除，用于禁止或允许转换结束后产生中断。 • 0：禁止 EOC 中断 • 1：允许 EOC 中断。当硬件设置 EOC 位时产生中断
位 4:0	AWDCH[4:0]：模拟看门狗通道选择位（Analog watchdog channel select bits） 这些位由软件设置和清除，用于选择模拟看门狗保护的输入通道。 • 00000：ADC 模拟输入通道 0 • 00001：ADC 模拟输入通道 1 • … • 01111：ADC 模拟输入通道 15 • 10000：ADC 模拟输入通道 16 • 10001：ADC 模拟输入通道 17 • 其他值：保留 *说明：ADC1 的模拟输入通道 16 和通道 17 在芯片内部分别连到了温度传感器和 V_{REFINT}。* *ADC2 的模拟输入通道 16 和通道 17 在芯片内部连到了 V_{SS}。* *ADC3 模拟输入通道 9、14、15、16、17 与 Vss 相连。*

注意：

RTC 输出的触发信号只能触发规则组转换，不需要设置规则组的触发使能控制 EXTTRIG。

ADC 的配置都在这种模式下生效，所以必须使能 AWDEN，不能使能循环和间断模式，在转换的过程中除 ADC_SR.AWD 以外的状态寄存器不会被置位。

和其他触发信号一样的效果触发 ADC 单个通道或整个规则组的转换，如果在整个规则组转换完成后 ADC_SR.AWD 状态位没有置起则重新回到停机（Stop）模式。

AWD 的唤醒信号根据 AWDSGL 和 AWDCH 设置可以单个通道的结果比较也可以所有通道的结果比较，直接把 ADC_SR.AWD 发到 EXTI，不需要使能 AWDIE。

12.12.3 ADC 控制寄存器 2（ADC_CR2）

偏移地址：0x08

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res								TSVREFE	SWSTART	JSWSTART	EXTTRIG	EXTSEL[2:0]			Res
								rw	rw	rw	rw	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JEXTTRIG	JEXTSEL[2:0]			ALIGN	Res		DMA	Res				RSTCAL	CAL	CONT	ADON
rw	rw			rw			rw					rw	rw	rw	rw

位 31:24	Res：保留 必须保持复位值。
位 23	TSVREFE：温度传感器和 V_{REFINT} 使能（Temperature sensor and V_{REFINT} enable） 该位由软件设置和清除，用于开启或禁止温度传感器和 V_{REFINT} 通道。在多于 1 个 ADC 的器件中，该位仅出现在 ADC1 中。 • 0：禁止温度传感器和 V_{REFINT} • 1：启用温度传感器和 V_{REFINT}

位 22	SWSTART：开始转换规则通道（Start conversion of regular channels） 由软件设置该位以启动转换，转换开始后硬件马上清除此位。如果在 EXTSEL[2:0]位中选择了 SWSTART 为触发事件，该位用于启动一组规则通道的转换。 • 0：复位状态 • 1：开始转换规则通道
位 21	JSWSTART：开始转换注入通道（Start conversion of injected channels） 由软件设置该位以启动转换，软件可清除此位或在转换开始后硬件马上清除此位。如果通过 JEXTSEL[2:0]位选择了 JSWSTART 作为触发事件，该位用于启动一组注入通道的转换。 • 0：复位状态 • 1：开始转换注入通道
位 20	EXTTRIG：规则通道的外部触发转换模式（External trigger conversion mode for regular channels） 该位由软件设置和清除，可开启或禁止用于启动规则通道组转换的外部触发事件。 • 0：禁用外部事件启动转换 • 1：使用外部事件启动转换
位 19:17	EXTSEL[2:0]：选择启动规则通道组转换的外部事件（External event select for regular group） 这些位选择用于启动规则通道组转换的外部事件。 ADC1 和 ADC2 的触发配置如下： • 000：定时器 1 的 CC1 事件 • 001：定时器 1 的 CC2 事件 • 010：定时器 1 的 CC3 事件 • 011：定时器 2 的 CC2 事件 • 100：定时器 3 的 TRGO 事件 • 101：定时器 4 的 CC4 事件 • 110：EXTI 线 11/TIM8_TRGO 事件（参见 10.3.6 ADC 复用功能重映射） • 111：SWSTART ADC3 的触发配置如下： • 000：定时器 3 的 CC1 事件 • 001：定时器 2 的 CC3 事件 • 010：定时器 1 的 CC3 事件 • 011：定时器 8 的 CC1 事件 • 100：定时器 8 的 TRGO 事件 • 101：定时器 5 的 CC1 事件 • 110：定时器 5 的 CC3 事件 • 111：SWSTART
位 16	Res：保留 必须保持复位值。
位 15	JEXTTRIG：注入通道的外部触发转换模式（External trigger conversion mode for injected channels） 该位由软件设置和清除，可开启或禁止用于启动注入通道组转换的外部触发事件。 • 0：禁用外部事件启动转换 • 1：使用外部事件启动转换
位 14:12	JEXTSEL[2:0]：选择启动注入通道组转换的外部事件（External event select for injected group）

	这些位选择用于启动注入通道组转换的外部事件。 ADC1 和 ADC2 的触发配置如下： • 000：定时器 1 的 TRGO 事件 • 001：定时器 1 的 CC4 事件 • 010：定时器 2 的 TRGO 事件 • 011：定时器 2 的 CC1 事件 • 100：定时器 3 的 CC4 事件 • 101：定时器 4 的 TRGO 事件 • 110：EXTI 线 15/TIM8_CC4 事件（参见 10.3.6 ADC 复用功能重映射） • 111：JSWSTART ADC3 的触发配置如下： • 000：定时器 1 的 TRGO 事件 • 001：定时器 1 的 CC4 事件 • 010：定时器 4 的 CC3 事件 • 011：定时器 8 的 CC2 事件 • 100：定时器 8 的 CC4 事件 • 101：定时器 5 的 TRGO 事件 • 110：定时器 5 的 CC4 事件 • 111：JSWSTART
位 11	ALIGN：数据对齐（Data alignment） 该位由软件设置和清除。见图 12-7 和图 12-8。 • 0：右对齐 • 1：左对齐
位 10:9	Res：保留 必须保持复位值。
位 8	DMA：直接存储器访问模式（Direct memory access enable） 该位由软件设置和清除。详见“11 DMA 控制器”。 • 0：禁用 DMA 模式 • 1：使用 DMA 模式 *说明：只有 ADC1 和 ADC3 能产生 DMA 请求。*
位 7:4	Res：保留 必须保持复位值。
位 3	RSTCAL：复位校准（Reset calibration） 该位由软件设置并由硬件清除。在校准寄存器被初始化后，该位将被清除。 • 0：校准寄存器已初始化 • 1：初始化校准寄存器 *说明：如果正在进行转换时设置 RSTCAL，清除校准寄存器需要额外的周期。*
位 2	CAL：A/D 校准（A/D Calibration） 该位由软件设置以开始校准，并在校准结束时由硬件清除。 • 0：校准完成 • 1：开始校准

位 1	CONT：连续转换（Continuous conversion） 该位由软件设置和清除。如果设置了此位，则转换将连续进行直到该位被清除。 • 0：单次转换模式 • 1：连续转换模式
位 0	ADON：开或关 A/D 转换器（A/D converter ON / OFF） 该位由软件设置和清除。 当该位为'0'时，写入'1'将把 ADC 从断电模式下唤醒。当该位为'1'时，写入'1'将启动转换。应用程序需注意，在转换器上电至转换开始有一个延迟 t_{STAB}，参见图 12-3。 • 0：禁用 ADC 转换/校准，并进入断电模式。 • 1：使能 ADC 并启动转换。 *说明：如果在这个寄存器中与 ADON 一起还有其他位被改变，则转换不被触发。这是为了防止触发错误的转换。*

12.12.4 ADC 采样时间寄存器 1（ADC_SMPR1）

偏移地址：0x0C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res								SMP17[2:0]			SMP16[2:0]			SMP15[2:1]	
								rw			rw			rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP15[0]	SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]		
rw	rw			rw			rw			rw			rw		

位 31:24	Res：保留 必须保持复位值。
位 3(x-10)+2:3(x-10) (x=17..10)	SMPx[2:0]：选择通道 x 的采样时间（Channel x Sample time selection） 这些位用于独立地选择每个通道的采样时间。在采样周期中，通道选择位必须保持不变。 • 000：1.5 周期 • 001：7.5 周期 • 010：13.5 周期 • 011：28.5 周期 • 100：41.5 周期 • 101：55.5 周期 • 110：71.5 周期 • 111：239.5 周期 *说明：ADC1 的模拟输入通道 16 和通道 17 在芯片内部分别连到了温度传感器和 V_{REFINT}。* *ADC2 的模拟输入通道 16 在芯片内部连到了 Vss。* *ADC3 模拟输入通道 14、15、16 与 Vss 相连。* *ADC1/2/3 的通道 18 只能用 1.5cycle 的快速采样模式。*

12.12.5 ADC 采样时间寄存器 2（ADC_SMPR2）

偏移地址：0x10

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res		SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:1]	
		rw			rw			rw			rw			rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP5[0]	SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]		
rw	rw			rw			rw			rw			rw		

位 31:30	Res：保留 必须保持复位值。
位 3x+2:3x (x=9..0)	SMPx[2:0]：选择通道 x 的采样时间（Channel x Sample time selection） 这些位用于独立地选择每个通道的采样时间。在采样周期中通道选择位必须保持不变。 • 000：1.5 周期 • 001：7.5 周期 • 010：13.5 周期 • 011：28.5 周期 • 100：41.5 周期 • 101：55.5 周期 • 110：71.5 周期 • 111：239.5 周期 *说明：ADC3 模拟输入通道 9 与 Vss 相连。*

12.12.6 ADC 注入通道 x 数据偏移寄存器（ADC_JOFRx）（x=1..4）

偏移地址：0x14+0x04*(x-1)

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res				JOFFSETx[11:0]											
				rw											

位 31: 12	Res：保留 必须保持复位值。
位 11:0	JOFFSETx[11:0]：注入通道 x 的数据偏移（Data offset for injected channel x） 当转换注入通道时，这些位定义了从原始转换数据中减去的数值。转换的结果可以在 ADC_JDRx 寄存器中读出。

12.12.7 ADC 看门狗高阈值寄存器（ADC_HTR）

偏移地址：0x24

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res				HT[11:0]											
				rw											

位 31:3	Res：保留 必须保持复位值。
位 11:0	HT[11:0]：模拟看门狗阈值上限（Analog watchdog high threshold） 这些位定义了模拟看门狗的阈值上限。

说明：软件可以在 ADC 转换进行的时候写该寄存器。写入的值在下一次转换结束的时候生效。写入该寄存器存在一定的延迟，导致其生效的准确时间存在一定的不确定性。

12.12.8 ADC 看门狗低阈值寄存器（ADC_LTR）

偏移地址：0x28

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res				LT[11:0]											
				rw											

位 31:3	Res：保留 必须保持复位值。
位 11:0	LT[11:0]：模拟看门狗的阈值下限（Analog watchdog low threshold） 这些位定义了模拟看门狗的阈值下限。

说明：软件可以在 ADC 转换进行的时候写该寄存器。写入的值在下一次转换结束的时候生效。写入该寄存器存在一定的延迟，导致其生效的准确时间存在一定的不确定性。

12.12.9 ADC 规则序列寄存器（ADC_SQR1）

偏移地址：0x2C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res								L[3:0]				SQ16[4:1]			
								rw				rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ16[0]	SQ15[4:0]					SQ14[4:0]					SQ13[4:0]				
rw	rw					rw					rw				

位 31:24	Res：保留 必须保持复位值。
位 23:20	L[3:0]：规则通道序列长度（Regular channel sequence length） 这些位由软件定义在规则通道转换序列中的通道数目。 • 0000：1 个转换 • 0001：2 个转换 • … • 1111：16 个转换

位 19:15	SQ16[4:0]：规则序列中的第 16 个转换（16th conversion in regular sequence） 这些位由软件定义转换序列中的第 16 个转换通道的编号（0~18）。
位 14:10	SQ15[4:0]：规则序列中的第 15 个转换（15th conversion in regular sequence）
位 9:5	SQ14[4:0]：规则序列中的第 14 个转换（14th conversion in regular sequence）
位 4:0	SQ13[4:0]：规则序列中的第 13 个转换（13th conversion in regular sequence）

12.12.10 ADC 规则序列寄存器 2（ADC_SQR2）

偏移地址：0x30

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res		SQ12[4:0]					SQ11[4:0]					SQ10[4:1]			
		rw					rw					rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ10[0]	SQ9[4:0]					SQ8[4:0]					SQ7[4:0]				
rw	rw					rw					rw				

位 31:30	Res：保留 必须保持复位值。
位 29:25	SQ12[4:0]：规则序列中的第 12 个转换（12th conversion in regular sequence） 这些位由软件定义转换序列中的第 12 个转换通道的编号（0~18）。
位 24:20	SQ11[4:0]：规则序列中的第 11 个转换（11th conversion in regular sequence）
位 19:15	SQ10[4:0]：规则序列中的第 10 个转换（10th conversion in regular sequence）
位 14:10	SQ9[4:0]：规则序列中的第 9 个转换（9th conversion in regular sequence）
位 9:5	SQ8[4:0]：规则序列中的第 8 个转换（7th conversion in regular sequence）
位 4:0	SQ7[4:0]：规则序列中的第 7 个转换（7th conversion in regular sequence）

12.12.11 ADC 规则序列寄存器 3（ADC_SQR3）

偏移地址：0x34

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res		SQ6[4:0]					SQ5[4:0]					SQ4[4:1]			
		rw					rw					rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ4[0]	SQ3[4:0]					SQ2[4:0]					SQ1[4:0]				
rw	rw					rw					rw				

位 31:30	Res：保留 必须保持复位值。
位 29:25	SQ6[4:0]：规则序列中的第 6 个转换（6th conversion in regular sequence） 这些位由软件定义转换序列中的第 6 个转换通道的编号（0~18）。

位 24:20	SQ5[4:0]：规则序列中的第 5 个转换（5th conversion in regular sequence）
位 19:15	SQ4[4:0]：规则序列中的第 4 个转换（4th conversion in regular sequence）
位 14:10	SQ3[4:0]：规则序列中的第 3 个转换（3rd conversion in regular sequence）
位 9:5	SQ2[4:0]：规则序列中的第 2 个转换（2nd conversion in regular sequence）
位 4:0	SQ1[4:0]：规则序列中的第 1 个转换（1st conversion in regular sequence）

12.12.12 ADC 注入序列寄存器（ADC_JSQR）

偏移地址：0x38

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res										JL[1:0]		JSQ4[4:1]			
										rw		rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JSQ4[0]	JSQ3[4:0]					JSQ2[4:0]					JSQ1[4:0]				
rw	rw					rw					rw				

位 31:22	Res：保留 必须保持复位值。
位 21:20	JL[1:0]：注入通道序列长度（Injected sequence length） 这些位由软件定义在注入通道转换序列中的通道数目。 • 00：1 个转换 • 01：2 个转换 • 10：3 个转换 • 11：4 个转换
位 19:15	JSQ4[4:0]：注入序列中的第 4 个转换（4th conversion in injected sequence when JL[1:0] = 3） 这些位由软件定义转换序列中的第 4 个转换的通道号（0~17）。 *说明：不同于规则转换序列，如果 JL[1:0]的长度小于 4，则转换的序列顺序是从（4-JL）开始。例如：ADC_JSQR[21:0]=10 00011 00011 00111 00010，意味着扫描转换将按下列通道顺序转换：7、3、3，而不是 2、7、3。*
位 14:10	JSQ3[4:0]：注入序列中的第 3 个转换（3rd conversion in injected sequence ,when JL[1:0] = 3）
位 9:5	JSQ2[4:0]：注入序列中的第 2 个转换（2nd conversion in injected sequence , when JL[1:0] = 3）
位 4:0	JSQ1[4:0]：注入序列中的第 1 个转换（1st conversion in injected sequence , when JL[1:0] = 3）

12.12.13 ADC 注入数据寄存器 x（ADC_JDRx）（x=1..4）

偏移地址：0x3C+0x04*(x-1)

复位值：0x0000 0000

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

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

r

位 31:16	Res：保留 必须保持复位值。
位 15:0	JDATA[15:0]：注入转换的数据（Injected conversion data） 这些位为只读，包含了注入通道 x 的转换结果。右对齐或左对齐的数据，见图 12-7 和图 12-8。

12.12.14 ADC 规则数据寄存器（ADC_DR）

偏移地址：0x4C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADC2DATA[15:0]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw															

位 31:16	ADC2DATA[15:0]：ADC2 转换的数据（ADC2 data used with ADC1_DR only for ADC dual mode） 仅在双模式下，ADC1_DR 的该位域包含了 ADC2 转换的规则通道数据。 其他情况下，不使用该位域。
位 15:0	DATA[15:0]：规则转换的数据（Regular data） 这些位为只读，包含了注入通道的转换结果。右对齐或左对齐的数据，见图 12-7 和图 12-8。

13 数字模拟转换（DAC）

数字/模拟转换模块（DAC）是 12 位数字输入、电压输出的数字/模拟转换器。DAC 可以配置为 8 位或 12 位模式，也可以与 DMA 控制器配合使用。DAC 工作在 12 位模式时，数据可以设置成左对齐或右对齐。DAC 模块有 2 个输出通道，每个通道都有单独的转换器。在双 DAC 模式下，2 个通道可以独立地进行转换，也可以同时进行转换并同步地更新 2 个通道的输出。DAC 可以通过引脚输入参考电压 V_{REF+}获得更精确的转换结果。

13.1 DAC 主要特性

- 2 个 DAC 转换器：每个转换器对应 1 个输出通道
- 8 位或者 12 位单调输出
- 12 位模式下支持数据左对齐或者右对齐
- 同步更新功能
- 噪声波形生成
- 三角波形生成
- 双 DAC 通道同时或者分别转换
- 每个通道都有 DMA 功能
- 外部触发转换
- 输入参考电压 V_{REF+}

单个 DAC 通道的框图如图 13-1，表 13-1 给出了引脚的说明。

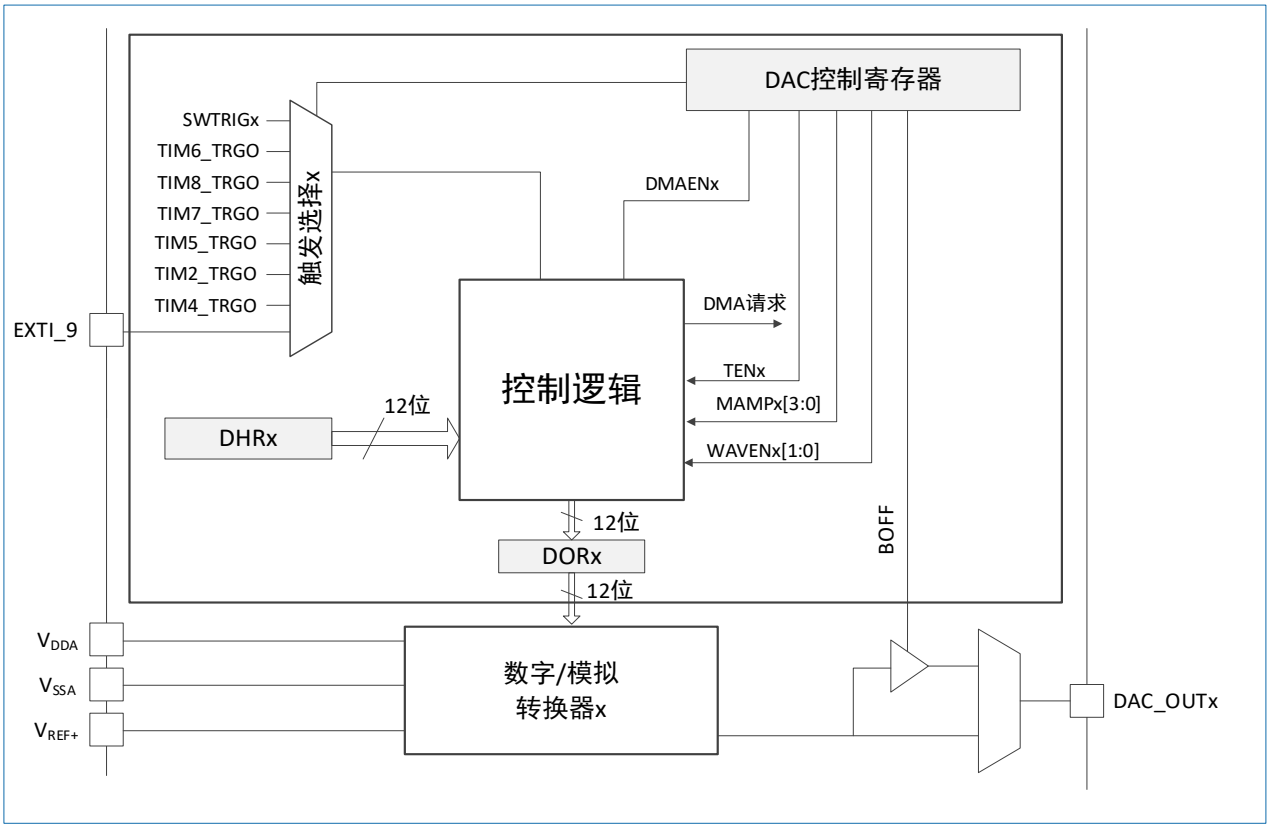


图 13-1 DAC 通道模块框图

表 13-1 DAC 引脚

名称	信号类型	说明
V_{REF+}	输入，正模拟参考电压	DAC 使用的高端/正极参考电压，$2V \le V_{REF+} \le V_{DDA}$
V_{DDA}	输入，模拟电源	模拟电源
V_{SSA}	输入，模拟电源地	模拟电源的地线
DAC_OUTx	模拟输出信号	DAC 通道 x 的模拟输出

注意：一旦使能 DACx 通道，相应的 GPIO 引脚（PA4 或者 PA5）就会自动与 DAC 的模拟输出相连（DAC_OUTx）。为了避免寄生的干扰和额外的功耗，引脚 PA4 或者 PA5 在之前应当设置成模拟输入（AIN）。

13.2 DAC 功能描述

13.2.1 使能 DAC 通道

将 DAC_CR 寄存器的 ENx 位置'1'即可打开对 DAC 通道 x 的供电。经过一段启动时间 t_{WAKEUP}，DAC 通道 x 即被使能。

注意：ENx 位只会使能 DAC 通道 x 的模拟部分，即便该位被置'0'，DAC 通道 x 的数字部分仍然工作。

13.2.2 使能 DAC 输出缓存

DAC 集成了 2 个输出缓冲器，可以用来减少输出阻抗，无需外部运放即可直接驱动外部负载。每个 DAC 通道输出缓冲器可以通过设置 DAC_CR 寄存器的 BOFFx 位来使能或关闭。

13.2.3 DAC 数据格式

根据选择的配置模式，数据按照下文所述写入指定的寄存器：

- 单 DAC 通道 x，有 3 种情况：
  - 8 位数据右对齐：用户须将数据写入寄存器 DAC_DHR8Rx[7:0]位（实际是存入寄存器 DHRx[11:4]位）
  - 12 位数据左对齐：用户须将数据写入寄存器 DAC_DHR12Lx[15:4]位（实际是存入寄存器 DHRx[11:0]位）
  - 12 位数据右对齐：用户须将数据写入寄存器 DAC_DHR12Rx[11:0]位（实际是存入寄存器 DHRx[11:0]位）

根据对 DAC_DHRyyyx 寄存器的操作，经过相应的移位后，写入的数据被转存到 DHRx 寄存器中（DHRx 是内部的数据保存寄存器 x）。随后，DHRx 寄存器的内容或被加载到 DORx 寄存器，或通过软件触发或外部事件触发被加载到 DORx 寄存器。

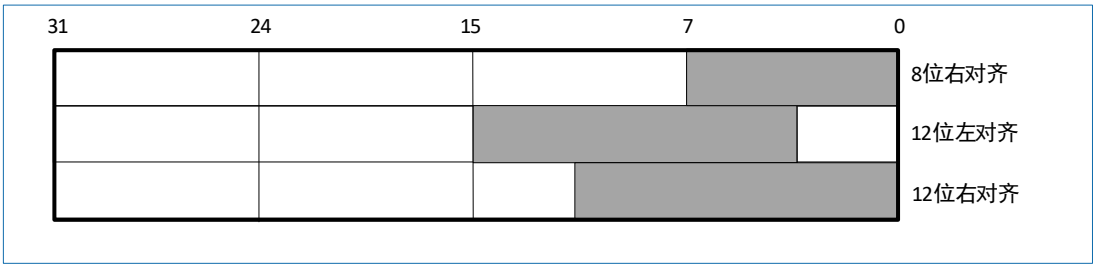


图 13-2 单 DAC 通道模式的数据寄存器

- 双 DAC 通道，有 3 种情况：

- 8 位数据右对齐：用户须将 DAC 通道 1 数据写入寄存器 DAC_DHR8RD[7:0]位（实际是存入寄存器 DHR1[11:4]位），将 DAC 通道 2 数据写入寄存器 DAC_DHR8RD[15:8]位（实际是存入寄存器 DHR2[11:4]位）。
- 12 位数据左对齐：用户须将 DAC 通道 1 数据写入寄存器 DAC_DHR12LD[15:4]位（实际是存入寄存器 DHR1[11:0]位），将 DAC 通道 2 数据写入寄存器 DAC_DHR12LD[31:20]位（实际是存入寄存器 DHR2[11:0]位）。
- 12 位数据右对齐：用户须将 DAC 通道 1 数据写入寄存器 DAC_DHR12RD[11:0]位（实际是存入寄存器 DHR1[11:0]位），将 DAC 通道 2 数据写入寄存器 DAC_DHR12RD[27:16]位（实际是存入寄存器 DHR2[11:0]位）。

根据对 DAC_DHRyyyD 寄存器的操作，经过相应的移位后，写入的数据被转存到 DHR1 和 DHR2 寄存器中（DHR1 和 DHR2 是内部的数据保存寄存器 x）。随后，DHR1 和 DHR2 的内容或被加载到 DORx 寄存器，或通过软件触发或外部事件触发被加载到 DORx 寄存器。

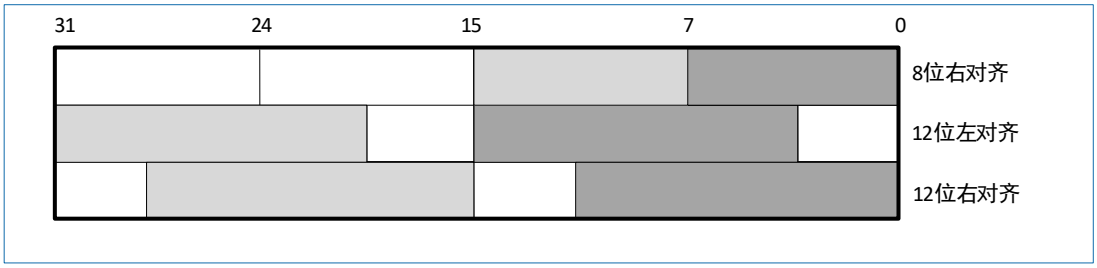


图 13-3 双 DAC 通道模式的数据寄存器

13.2.4 DAC 转换

不能直接对寄存器 DAC_DORx 写入数据，任何输出到 DAC 通道 x 的数据都必须写入 DAC_DHRx 寄存器（数据实际写入 DAC_DHR8Rx、DAC_DHR12Lx、DAC_DHR12Rx、DAC_DHR8RD、DAC_DHR12LD、或者 DAC_DHR12RD 寄存器）。

如果没有选中硬件触发（寄存器 DAC_CR 的 TENx 位置'0'），存入寄存器 DAC_DHRx 的数据会在一个 APB1 时钟周期后自动传至寄存器 DAC_DORx。如果选中硬件触发（寄存器 DAC_CR 的 TENx 位置'1'），数据传输在触发发生以后 3 个 APB1 时钟周期后完成。

一旦数据从 DAC_DHRx 寄存器装入 DAC_DORx 寄存器，在经过时间 $t_{SETTLING}$ 之后，模拟输出电压即有效，这段时间的长短依电源电压和模拟输出负载的不同会有所变化。

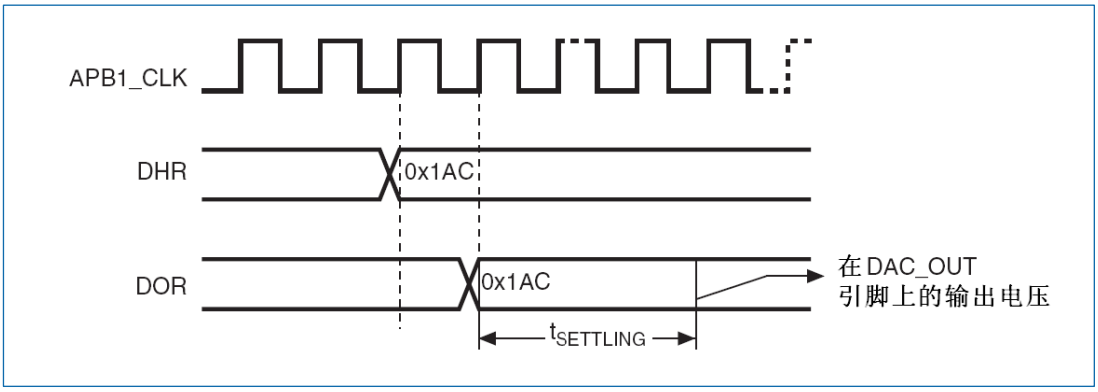


图 13-4 TEN=0 触发失能时转换的时间框图

13.2.5 DAC 输出电压

数字输入经过 DAC 被线性地转换为模拟电压输出，其范围为 0 到 V_{REF+}。任一 DAC 通道引脚上的输出电压满足下面的关系：

$$DAC\ 输出=V_{REF}*（DOR/4095）$$

13.2.6 选择 DAC 触发

如果 TENx 位被置’1’，DAC 转换可以由某外部事件触发（定时器计数器、外部中断线）。配置控制位 TSELx[2:0]可以选择 8 个触发事件之一触发 DAC 转换。

表 13-2 外部触发

触发源	类型	TSELx[2:0]
TIM6 TRGO 事件	来自片上定时器的内部信号	000
TIM8 TRGO 事件		001
TIM7 TRGO 事件		010
TIM5 TRGO 事件		011
TIM2 TRGO 事件		100
TIM4 TRGO 事件		101
EXTI 9	外部引脚	110
SWTRIG（软件触发）	软件控制位	111

每次 DAC 接口侦测到来自选中的定时器 TRGO 输出，或者外部中断线 9 的上升沿，最后一次存放在寄存器 DAC_DHRx 中的数据会被传送到寄存器 DAC_DORx 中。在触发发生 3 个 APB1 时钟周期之后，寄存器 DAC_DORx 更新为新值。如果选择软件触发，一旦 SWTRIG 位置’1’，转换即开始。在数据从 DAC_DHRx 寄存器传送到 DAC_DORx 寄存器后，SWTRIG 位由硬件自动清零。

注意：

不能在 ENx 为’1’时改变 TSELx[2:0]位。

如果选择软件触发，数据从寄存器 DAC_DHRx 传送到寄存器 DAC_DORx 只需要一个 APB1 时钟周期。

13.2.7 DMA 请求

任一 DAC 通道都具有 DMA 功能。2 个 DMA 通道可分别用于 2 个 DAC 通道的 DMA 请求。

如果 DMAENx 位置’1’，一旦有外部触发（而不是软件触发）发生，则产生一个 DMA 请求，然后 DAC_DHRx 寄存器的数据被传送到 DAC_DORx 寄存器。

在双 DAC 模式下，如果 2 个通道的 DMAENx 位都为’1’，则会产生 2 个 DMA 请求。如果实际只需要一个 DMA 传输，则应仅将相应的 DMAENx 位置’1’。这样，程序可以在只使用一个 DMA 请求，一个 DMA 通道的情况下，处理工作在双 DAC 模式的 2 个 DAC 通道。

DAC 的 DMA 请求不会入队列，因此如果第 2 个外部触发在第 1 个外部触发的响应之前到达，则不能处理第 2 个 DMA 请求，也不会报告错误。

13.2.8 噪声生成

可以利用线性反馈移位寄存器（LFSR）产生幅度变化的伪噪声。设置 WAVE[1:0]位为’01’以选择 DAC 噪声生成功能。寄存器 LFSR 的预装入值为 0xAAA。按照特定算法，在每次触发事件后 3 个 APB1 时钟周期之后更新该寄存器的值。

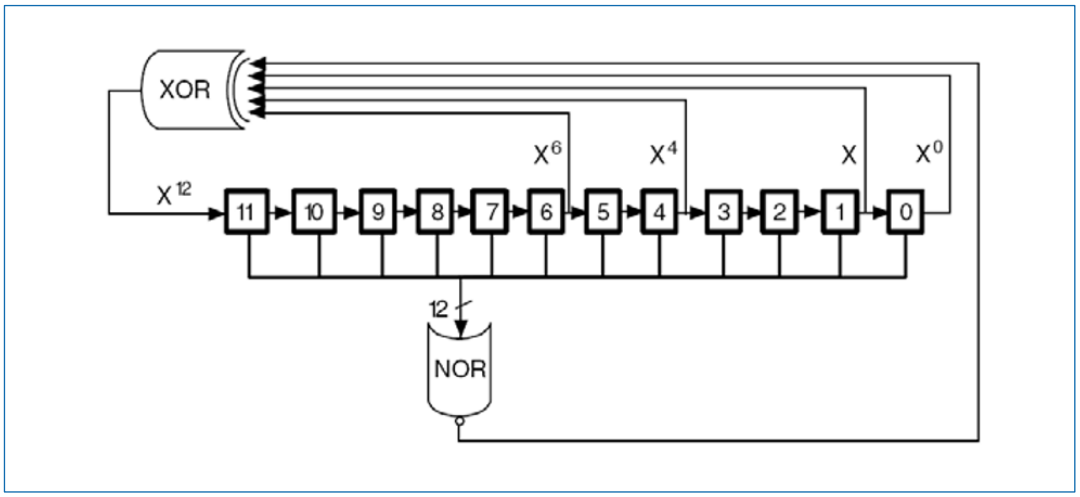


图 13-5 DAC LFSR 寄存器算法

设置 DAC_CR 寄存器的 MAMPx[3:0]位可以屏蔽部分或者全部 LFSR 的数据，这样得到的 LFSR 值与 DAC_DHRx 的数值相加，去掉溢出位之后即被写入 DAC_DORx 寄存器。

如果寄存器 LFSR 值为 0x000，则会注入'1'（防锁定机制）。将 WAVEx[1:0]位置'0'，可以复位 LFSR 波形的生成算法。

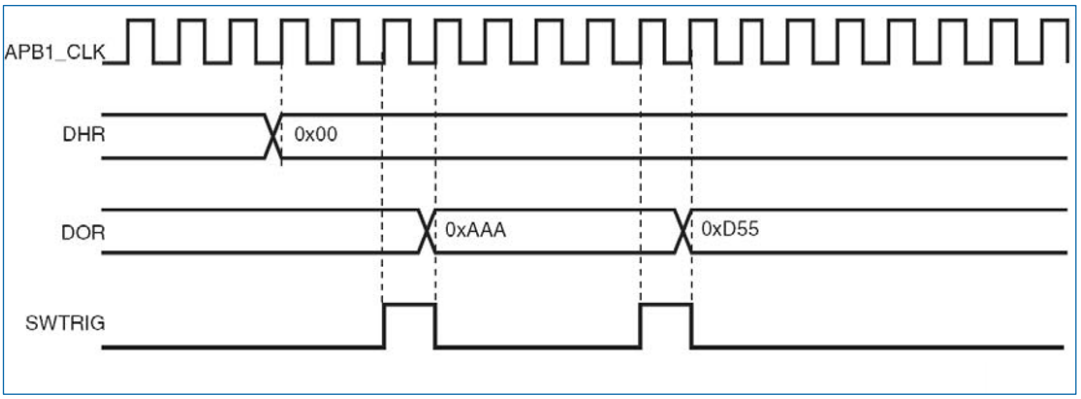


图 13-6 带 LFSR 波形生成的 DAC 转换（使能软件触发）

注意：为了产生噪声，必须使能 DAC 触发，即设 DAC_CR 寄存器的 TENx 位为'1'。

13.2.9 三角波生成

可以在直流电流或慢变信号上加上一个小幅度的三角波。设置 WAVEx[1:0]位为'10'以选择 DAC 的三角波生成功能。设置 DAC_CR 寄存器的 MAMPx[3:0]位来选择三角波的幅度。内部的三角波计数器在每次触发事件之后的 3 个 APB1 时钟周期后累加 1。计数器的值与 DAC_DHRx 寄存器的数值相加并丢弃溢出位后写入 DAC_DORx 寄存器。在传入 DAC_DORx 寄存器的数值小于 MAMP[3:0]位定义的最大幅度时，三角波计数器逐步累加。一旦达到设置的最大幅度，则计数器开始递减，达到 0 后再开始累加，周而复始。

将 WAVEx[1:0]位置'0'可以复位三角波的生成。

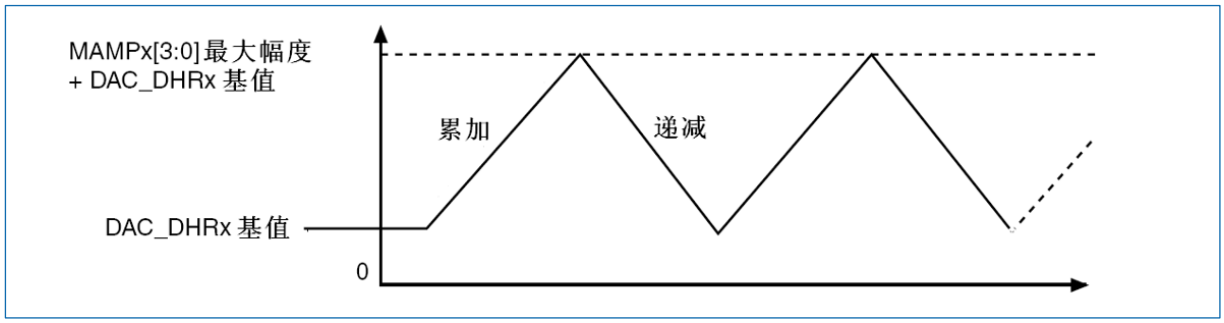


图 13-7 DAC 三角波生成

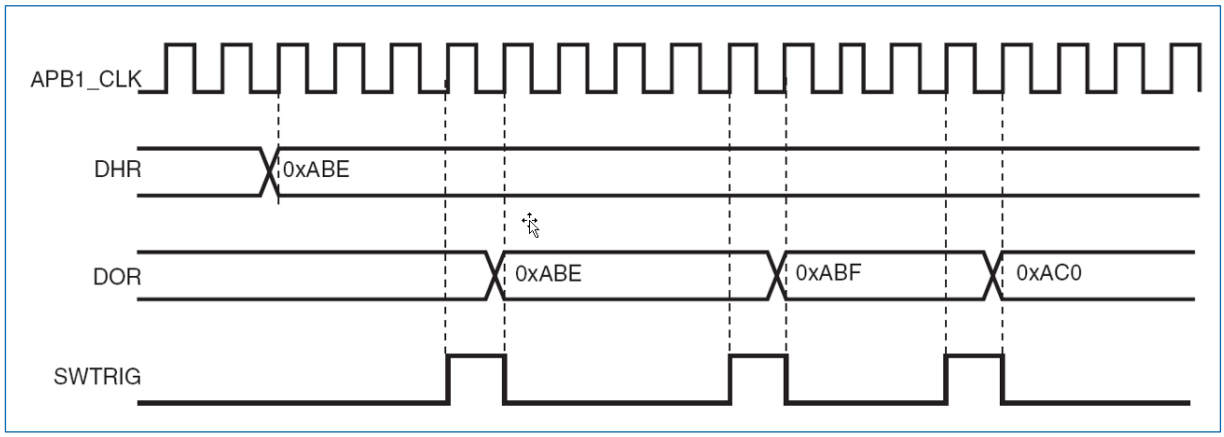


图 13-8 带三角波生成的 DAC 转换（使能软件触发）

注意：

为了产生三角波，必须使能 DAC 触发，即设 DAC_CR 寄存器的 TENx 位为'1'。

MAMPx[3:0]位必须在使能 DAC 之前设置，否则其值不能修改。

13.2.10 双 DAC 通道转换

在需要双 DAC 通道同时工作的情况下，为了更有效地利用总线带宽，DAC 集成了 3 个供双 DAC 模式使用的寄存器：DHR8RD、DHR12RD 和 DHR12LD。只需要访问一个寄存器即可完成同时驱动 2 个 DAC 通道的操作。

对于双 DAC 通道转换和这些专用寄存器，共有 11 种转换模式可用。这些转换模式在只使用一个 DAC 通道的情况下，仍然可通过独立的 DHRx 寄存器操作。

所有模式详述于以下章节。

13.2.10.1 独立触发（不产生波形）

按照下列顺序设置 DAC 工作在此转换模式：

1. 分别设置 2 个 DAC 通道的触发使能位 TEN1 和 TEN2 为'1'。
2. 通过设置 TSEL1[2:0]和 TSEL2[2:0]位为不同值，分别配置 2 个 DAC 通道的不同触发源。
3. 将双 DAC 通道转换数据装入所需的 DHR 寄存器（DHR12RD、DHR12LD 或 DHR8RD）：

- 当发生 DAC 通道 1 触发事件时，在延迟 3 个 APB1 时钟周期后，寄存器 DHR1 的值传入寄存器 DAC_DOR1。
- 当发生 DAC 通道 2 触发事件时，在延迟 3 个 APB1 时钟周期后，寄存器 DHR2 的值传入寄存器 DAC_DOR2。

13.2.10.2 独立触发（生成单个 LFSR）

按照下列步骤将 DAC 配置为该转换模式：

1. 分别设置 2 个 DAC 通道的触发使能位 TEN1 和 TEN2 为’1’。
2. 通过设置 TSEL1[2:0]和 TSEL2[2:0]位为不同值，分别配置 2 个 DAC 通道的不同触发源。
3. 设置 2 个 DAC 通道的 WAVEx[1:0]位为”01”，并设置 MAMPx[3:0]为相同的 LFSR 屏蔽值。
4. 将双 DAC 通道转换数据装入所需的 DHR 寄存器（DHR12RD、DHR12LD 或 DHR8RD）。

- 当发生 DAC 通道 1 触发事件时，具有相同屏蔽的 LFSR1 计数器值与 DHR1 寄存器数值相加，（延迟 3 个 APB1 时钟周期后）结果传入寄存器 DAC_DOR1，然后更新 LFSR1 计数器。
- 当发生 DAC 通道 2 触发事件时，具有相同屏蔽的 LFSR2 计数器值与 DHR2 寄存器数值相加，（延迟 3 个 APB1 时钟周期后）结果传入寄存器 DAC_DOR2，然后更新 LFSR2 计数器。

13.2.10.3 独立触发（生成不同 LFSR）

按照下列步骤将 DAC 配置为该转换模式：

1. 分别设置 2 个 DAC 通道的触发使能位 TEN1 和 TEN2 为’1’。
2. 通过设置 TSEL1[2:0]和 TSEL2[2:0]位为不同值，分别配置 2 个 DAC 通道的不同触发源。
3. 设置 2 个 DAC 通道的 WAVEx[1:0]位为”01”，并设 MAMPx[3:0]为不同的 LFSR 屏蔽值。
4. 将双 DAC 通道转换数据装入所需的 DHR 寄存器（DHR12RD、DHR12LD 或者 DHR8RD）。

- 当发生 DAC 通道 1 触发事件时，按照 MAMP1[3:0]所设屏蔽的 LFSR1 计数器值与 DHR1 寄存器数值相加，（延迟 3 个 APB1 时钟周期后）结果传入寄存器 DAC_DOR1，然后更新 LFSR1 计数器。
- 当发生 DAC 通道 2 触发事件时，按照 MAMP2[3:0]所设屏蔽的 LFSR2 计数器值与 DHR2 寄存器数值相加，（延迟 3 个 APB1 时钟周期后）结果传入寄存器 DAC_DOR2，然后更新 LFSR2 计数器。

13.2.10.4 独立触发（生成单个三角波）

按照下列步骤将 DAC 配置为该转换模式：

1. 分别设置 2 个 DAC 通道的触发使能位 TEN1 和 TEN2 为’1’。
2. 通过设置 TSEL1[2:0]和 TSEL2[2:0]位为不同值，分别配置 2 个 DAC 通道的不同触发源。
3. 设置 2 个 DAC 通道的 WAVEx[1:0]位为”1x”，并设 MAMPx[3:0]为相同的三角波幅值。
4. 将双 DAC 通道转换数据装入所需的 DHR 寄存器（DHR12RD、DHR12LD 或 DHR8RD）。

- 当发生 DAC 通道 1 触发事件时，相同的三角波幅值加上 DHR1 寄存器的值，（延迟 3 个 APB1 时钟周期后）结果传入寄存器 DAC_DOR1，然后更新 DAC 通道 1 三角波计数器。
- 当发生 DAC 通道 2 触发事件时，相同的三角波幅值加上 DHR2 寄存器的值，（延迟 3 个 APB1 时钟周期后）结果传入寄存器 DAC_DOR2，然后更新 DAC 通道 2 三角波计数器。

13.2.10.5 独立触发（生成不同三角波）

按照下列步骤将 DAC 配置为该转换模式：

1. 分别设置 2 个 DAC 通道的触发使能位 TEN1 和 TEN2 为’1’。

2. 通过设置 TSEL1[2:0]和 TSEL2[2:0]位为不同值，分别配置 2 个 DAC 通道的不同触发源。

3. 设置 2 个 DAC 通道的 WAVEx[1:0]位为’1x’，并设 MAMP1[3:0]和 MAMP2[3:0]为不同的三角波幅值。

4. 将双 DAC 通道转换数据装入所需的 DHR 寄存器（DHR12RD、DHR12LD 或 DHR8RD）。

- 当发生 DAC 通道 1 触发事件时，MAMP1[3:0]所设的三角波幅值加上 DHR1 寄存器数值，（延迟 3 个 APB1 时钟周期后）结果传入寄存器 DAC_DOR1，然后更新 DAC 通道 1 三角波计数器。
- 当发生 DAC 通道 2 触发事件时，MAMP2[3:0]所设的三角波幅值加上 DHR2 寄存器数值，（延迟 3 个 APB1 时钟周期后）结果传入寄存器 DAC_DOR2，然后更新 DAC 通道 2 三角波计数器。

13.2.10.6 同步软件启动

按照下列步骤将 DAC 配置为该转换模式：将双 DAC 通道转换数据装入所需的 DHR 寄存器（DHR12RD、DHR12LD 或 DHR8RD）。

在此配置下，一个 APB1 时钟周期后，DHR1 和 DHR2 寄存器的数值即被分别传入 DAC_DOR1 和 DAC_DOR2 寄存器。

13.2.10.7 同步触发（不产生波形）

按照下列步骤将 DAC 配置为该转换模式：

1. 分别设置 2 个 DAC 通道的触发使能位 TEN1 和 TEN2 为’1’。

2. 通过设置 TSEL1[2:0]和 TSEL2[2:0]位为相同值，分别配置 2 个 DAC 通道使用相同触发源。

3. 将双 DAC 通道转换数据装入所需的 DHR 寄存器（DHR12RD、DHR12LD 或 DHR8RD）。

当发生触发事件时，（延迟 3 个 APB1 时钟周期后）DHR1 和 DHR2 寄存器的数值分别传入 DAC_DOR1 和 DAC_DOR2 寄存器。

13.2.10.8 同步触发（生成单个 LFSR）

按照下列步骤将 DAC 配置为该转换模式：

1. 分别设置 2 个 DAC 通道的触发使能位 TEN1 和 TEN2 为’1’。

2. 通过设置 TSEL1[2:0]和 TSEL2[2:0]位为相同值，分别配置 2 个 DAC 通道使用相同触发源。

3. 设置 2 个 DAC 通道的 WAVEx[1:0]位为”01”，并设 MAMP1[3:0]和 MAMP2[3:0]为相同的 LFSR 屏蔽值。

4. 将双 DAC 通道转换数据装入所需的 DHR 寄存器（DHR12RD、DHR12LD 或 DHR8RD）。

- 当发生触发事件时，MAMP1[3:0]所设屏蔽的 LFSR1 计数器值与 DHR1 寄存器的数值相加，（延迟 3 个 APB1 时钟周期后）结果传入 DAC_DOR1 寄存器，然后更新 LFSR1 计数器。
- 同时，MAMP2[3:0]所设屏蔽的 LFSR2 计数器值与 DHR2 寄存器的数值相加，（延迟 3 个 APB1 时钟周期后）结果传入寄存器 DAC_DOR2，然后更新 LFSR2 计数器。

13.2.10.9 同步触发（生成不同 LFSR）

按照下列步骤将 DAC 配置为该转换模式：

1. 分别设置 2 个 DAC 通道的触发使能位 TEN1 和 TEN2 为’1’。

2. 通过设置 TSEL1[2:0]和 TSEL2[2:0]位为相同值，分别配置 2 个 DAC 通道使用相同触发源。

3. 设置 2 个 DAC 通道的 WAVEx[1:0]位为’01’，并设 MAMP1[3:0]和 MAMP2[3:0]为不同的 LFSR 屏蔽

值。

4. 将双 DAC 通道转换数据装入所需的 DHR 寄存器（DHR12RD、DHR12LD 或 DHR8RD）。

- 当发生触发事件时，具有相同屏蔽的 LFSR1 计数器值与 DHR1 寄存器数值相加，（延迟 3 个 APB1 时钟周期后）结果传入寄存器 DAC_DOR1，然后更新 LFSR1 计数器。
- 同时，具有相同屏蔽的 LFSR2 计数器值与 DHR2 寄存器数值相加，（延迟 3 个 APB1 时钟周期后）结果传入寄存器 DAC_DOR2，然后更新 LFSR2 计数器。

13.2.10.10 同步触发（生成单个三角波）

按照下列步骤将 DAC 配置为该转换模式：

1. 分别设置 2 个 DAC 通道的触发使能位 TEN1 和 TEN2 为’1’。
2. 通过设置 TSEL1[2:0]和 TSEL2[2:0]位为相同值，分别配置 2 个 DAC 通道使用相同触发源。
3. 设置 2 个 DAC 通道的 WAVEx[1:0]位为’1x’，并设 MAMPx[3:0]为相同的三角波幅值。
4. 将双 DAC 通道转换数据装入所需的 DHR 寄存器（DHR12RD、DHR12LD 或 DHR8RD）。

- 当发生触发事件时，相同的三角波幅值与 DHR1 寄存器数值相加，（延迟 3 个 APB1 时钟周期后）结果传入寄存器 DAC_DOR1，然后更新三角波计数器 1。
- 同时，相同的三角波幅值与 DHR2 寄存器数值相加，（延迟 3 个 APB1 时钟周期后）结果传入寄存器 DAC_DOR2，然后更新三角波计数器 2。

13.2.10.11 同步触发（生成不同三角波）

按照下列步骤将 DAC 配置为该转换模式：

1. 分别设置 2 个 DAC 通道的触发使能位 TEN1 和 TEN2 为’1’。
2. 通过设置 TSEL1[2:0]和 TSEL2[2:0]位为相同值，分别配置 2 个 DAC 通道使用相同触发源。
3. 设置 2 个 DAC 通道的 WAVEx[1:0]位为’1x’，并设 MAMP1[3:0]和 MAMP2[3:0]为不同的三角波幅值。
4. 将双 DAC 通道转换数据装入所需的 DHR 寄存器（DHR12RD、DHR12LD 或 DHR8RD）。

- 当发生触发事件时，MAMP1[3:0]所设的三角波幅值与 DHR1 寄存器数值相加，（延迟 3 个 APB1 时钟周期后）结果传入寄存器 DAC_DOR1，然后更新三角波计数器 1。
- 同时，MAMP2[3:0]所设的三角波幅值与 DHR2 寄存器数值相加，（延迟 3 个 APB1 时钟周期后）结果传入寄存器 DAC_DOR2，然后更新三角波计数器 2。

13.3 DAC 寄存器

基地址：0x4000 7400

空间大小：0x400

DAC 寄存器必须以字（32 位）的方式进行访问。

13.3.1 DAC 控制寄存器（DAC_CR）

偏移地址：0x00

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res			DMAEN2	MAMP2[3:0]				WAVE2[1:0]		TSEL2[2:0]			TEN2	BOFF2	EN2
			rw	rw				rw		rw			rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res			DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[2:0]			TEN1	BOFF1	EN1
			rw	rw				rw		rw			rw	rw	rw

位 31:29	Res：保留 必须保持复位值。
位 28	DMAEN2：DAC 通道 2 的 DMA 使能（DAC channel2 DMA enable） 该位由软件设置和清除。 • 0：关闭 DAC 通道 2 的 DMA 模式 • 1：使能 DAC 通道 2 的 DMA 模式
位 27:24	MAMP2[3:0]：DAC 通道 2 屏蔽/幅值选择器（DAC channel2 mask/amplitude selector） 由软件设置这些位，用来在噪声生成模式下选择屏蔽位，在三角波生成模式下选择波形的幅值。 • 0000：不屏蔽 LFSR 位 0/三角波幅值等于 1 • 0001：不屏蔽 LFSR 位[1:0]/三角波幅值等于 3 • 0010：不屏蔽 LFSR 位[2:0]/三角波幅值等于 7 • 0011：不屏蔽 LFSR 位[3:0]/三角波幅值等于 15 • 0100：不屏蔽 LFSR 位[4:0]/三角波幅值等于 31 • 0101：不屏蔽 LFSR 位[5:0]/三角波幅值等于 63 • 0110：不屏蔽 LFSR 位[6:0]/三角波幅值等于 127 • 0111：不屏蔽 LFSR 位[7:0]/三角波幅值等于 255 • 1000：不屏蔽 LFSR 位[8:0]/三角波幅值等于 511 • 1001：不屏蔽 LFSR 位[9:0]/三角波幅值等于 1023 • 1010：不屏蔽 LFSR 位[10:0]/三角波幅值等于 2047 • ≥1011：不屏蔽 LFSR 位[11:0]/三角波幅值等于 4095
位 23:22	WAVE2[1:0]：DAC 通道 2 噪声/三角波生成使能（DAC channel2 noise/triangle wave generation enable） 这些位由软件设置和清除。 • 00：关闭波形发生器 • 01：使能噪声波形发生器 • 1x：使能三角波发生器
位 21:19	TSEL2[2:0]：DAC 通道 2 触发选择（DAC channel2 trigger selection） 这些位用于选择 DAC 通道 2 的外部触发事件。 • 000：TIM6 TRGO 事件 • 001：TIM8 TRGO 事件 • 010：TIM7 TRGO 事件 • 011：TIM5 TRGO 事件 • 100：TIM2 TRGO 事件 • 101：TIM4 TRGO 事件 • 110：外部中断线 9 • 111：软件触发 *注意：这些位只能在 TEN2=1（DAC 通道 2 触发使能）时设置。*
位 18	TEN2：DAC 通道 2 触发使能（DAC channel2 trigger enable） 该位由软件设置和清除，用来使能/关闭 DAC 通道 2 的触发。

	• 0：关闭 DAC 通道 2 触发，写入 DAC_DHRx 寄存器的数据在 1 个 APB1 时钟周期后传入 DAC_DOR2 寄存器。 • 1：使能 DAC 通道 2 触发，写入 DAC_DHRx 寄存器的数据在 3 个 APB1 时钟周期后传入 DAC_DOR2 寄存器。 *注意：如果选择软件触发，写入寄存器 DAC_DHRx 的数据只需要 1 个 APB1 时钟周期就可以传入寄存器 DAC_DOR2。*
位 17	BOFF2：DAC 通道 2 输出缓冲器禁能（DAC channel2 output buffer disable） 缓冲器该位由软件设置和清除，用来使能/关闭 DAC 通道 2 的输出缓冲器。 • 0：使能 DAC 通道 2 输出缓冲器 • 1：关闭 DAC 通道 2 输出缓冲器
位 16	EN2：DAC 通道 2 使能（DAC channel2 enable） 该位由软件设置和清除，用来使能/关闭 DAC 通道 2。 • 0：关闭 DAC 通道 2 • 1：使能 DAC 通道 2
位 15:13	Res：保留 必须保持复位值。
位 12	DMAEN1：DAC 通道 1 的 DMA 使能（DAC channel1 DMA enable） 该位由软件设置和清除。 • 0：关闭 DAC 通道 1 的 DMA 模式 • 1：使能 DAC 通道 1 的 DMA 模式
位 11:8	MAMP1[3:0]：DAC 通道 1 屏蔽/幅值选择器（DAC channel1 mask/amplitude selector） 由软件设置这些位，用来在噪声生成模式下选择屏蔽位，在三角波生成模式下选择波形的幅值。 • 0000：不屏蔽 LFSR 位 0/三角波幅值等于 1 • 0001：不屏蔽 LFSR 位[1:0]/三角波幅值等于 3 • 0010：不屏蔽 LFSR 位[2:0]/三角波幅值等于 7 • 0011：不屏蔽 LFSR 位[3:0]/三角波幅值等于 15 • 0100：不屏蔽 LFSR 位[4:0]/三角波幅值等于 31 • 0101：不屏蔽 LFSR 位[5:0]/三角波幅值等于 63 • 0110：不屏蔽 LFSR 位[6:0]/三角波幅值等于 127 • 0111：不屏蔽 LFSR 位[7:0]/三角波幅值等于 255 • 1000：不屏蔽 LFSR 位[8:0]/三角波幅值等于 511 • 1001：不屏蔽 LFSR 位[9:0]/三角波幅值等于 1023 • 1010：不屏蔽 LFSR 位[10:0]/三角波幅值等于 2047 • ≥1011：不屏蔽 LFSR 位[11:0]/三角波幅值等于 4095
位 7:6	WAVE1[1:0]：DAC 通道 1 噪声/三角波生成使能（DAC channel1 noise/triangle wave generation enable） 这些位由软件设置和清除。 • 00：关闭波形生成 • 01：使能噪声波形发生器 • 1x：使能三角波发生器
位 5:3	TSEL1[2:0]：DAC 通道 1 触发选择（DAC channel1 trigger selection） 该位用于选择 DAC 通道 1 的外部触发事件。

	• 000：TIM6 TRGO 事件 • 001：TIM8 TRGO 事件 • 010：TIM7 TRGO 事件 • 011：TIM5 TRGO 事件 • 100：TIM2 TRGO 事件 • 101：TIM4 TRGO 事件 • 110：外部中断线 9 • 111：软件触发 *注意：该位只能在 TEN1=1（DAC 通道 1 触发使能）时设置。*
位 2	TEN1：DAC 通道 1 触发使能（DAC channel1 trigger enable） 该位由软件设置和清除，用来使能/关闭 DAC 通道 1 的触发。 • 0：关闭 DAC 通道 1 触发，写入寄存器 DAC_DHRx 的数据在 1 个 APB1 时钟周期后传入寄存器 DAC_DOR1。 • 1：使能 DAC 通道 1 触发，写入寄存器 DAC_DHRx 的数据在 3 个 APB1 时钟周期后传入寄存器 DAC_DOR1。 *注意：如果选择软件触发，写入寄存器 DAC_DHRx 的数据只需要 1 个 APB1 时钟周期就可以传入寄存器 DAC_DOR1。*
位 1	BOFF1：关闭 DAC 通道 1 输出缓冲器（DAC channel1 output buffer disable） 该位由软件设置和清除，用来使能/关闭 DAC 通道 1 的输出缓冲器。 • 0：使能 DAC 通道 1 输出缓冲器 • 1：关闭 DAC 通道 1 输出缓冲器
位 0	EN1：DAC 通道 1 使能（DAC channel1 enable） 该位由软件设置和清除，用来使能/失能 DAC 通道 1。 • 0：关闭 DAC 通道 1 • 1：使能 DAC 通道 1

13.3.2 DAC 软件触发寄存器（DAC_SWTRIGR）

偏移地址：0x04

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res														SWTRIG2	SWTRIG1
														w	w

位 31:2	Res：保留 必须保持复位值。
位 1	SWTRIG2：DAC 通道 2 软件触发（DAC channel2 software trigger） 该位由软件设置和清除，用来使能/关闭软件触发。 • 0：关闭 DAC 通道 2 软件触发 • 1：使能 DAC 通道 2 软件触发 *注意：一旦寄存器 DAC_DHR2 的数据传入寄存器 DAC_DOR2，（1 个 APB1 时钟周期后）该位由硬件*

	置'0'。
位 0	SWTRIG1：DAC 通道 1 软件触发（DAC channel1 software trigger） 该位由软件设置和清除，用来使能/关闭软件触发。 • 0：关闭 DAC 通道 1 软件触发 • 1：使能 DAC 通道 1 软件触发 *注意：一旦寄存器 DAC_DHR1 的数据传入寄存器 DAC_DOR1，（1 个 APB1 时钟周期后）该位由硬件置'0'。*

13.3.3 DAC 通道 1 的 12 位右对齐数据保持寄存器（DAC_DHR12R1）

偏移地址：0x08

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res				DACC1DHR[11:0]											
				rw											

位 31:12	Res：保留 必须保持复位值。
位 11:0	DACC1DHR[11:0]：DAC 通道 1 的 12 位右对齐数据（DAC channel1 12-bit right-aligned data） 该位由软件写入，表示 DAC 通道 1 的 12 位数据。

13.3.4 DAC 通道 1 的 12 位左对齐数据保持寄存器（DAC_DHR12L1）

偏移地址：0x0C

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Res			
rw															

位 31:16	Res：保留 必须保持复位值。
位 15:4	DACC1DHR[11:0]：DAC 通道 1 的 12 位左对齐数据（DAC channel1 12-bit left-aligned data） 该位由软件写入，表示 DAC 通道 1 的 12 位数据。
位 3:0	Res：保留 必须保持复位值。

13.3.5 DAC 通道 1 的 8 位右对齐数据保持寄存器（DAC_DHR8R1）

偏移地址：0x10

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								DACC1DHR[7:0]							
								rw							

位 31:8	Res：保留 必须保持复位值。
位 7:0	DACC1DHR[7:0]：DAC 通道 1 的 8 位右对齐数据（DAC channel1 8-bit right-aligned data） 该位由软件写入，表示 DAC 通道 1 的 8 位数据。

13.3.6 DAC 通道 2 的 12 位右对齐数据保持寄存器（DAC_DHR12R2）

偏移地址：0x14

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res				DACC2DHR[11:0]											
				rw											

位 31:12	Res：保留 必须保持复位值。
位 11:0	DACC2DHR[11:0]：DAC 通道 2 的 12 位右对齐数据（DAC channel2 12-bit right-aligned data） 该位由软件写入，表示 DAC 通道 2 的 12 位数据。

13.3.7 DAC 通道 2 的 12 位左对齐数据保持寄存器（DAC_DHR12L2）

偏移地址：0x18

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[11:0]												Res			
rw															

位 31:16	Res：保留 必须保持复位值。
位 15:4	DACC2DHR[11:0]：DAC 通道 2 的 12 位左对齐数据（DAC channel2 12-bit left-aligned data） 该位由软件写入，表示 DAC 通道 2 的 12 位数据。
位 3:0	Res：保留 必须保持复位值。

13.3.8 DAC 通道 2 的 8 位右对齐数据保持寄存器（DAC_DHR8R2）

偏移地址：0x1C

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								DACC2DHR[7:0]							
								rw							

位 31:8	Res：保留 必须保持复位值。
位 7:0	DACC2DHR[7:0]：DAC 通道 2 的 8 位右对齐数据（DAC channel2 8-bit right-aligned data） 该位由软件写入，表示 DAC 通道 2 的 8 位数据。

13.3.9 双 DAC 的 12 位右对齐数据保持寄存器（DAC_DHR12RD）

偏移地址：0x20

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res				DACC2DHR[11:0]											
				rw											

15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
Res				DACC1DHR[11:0]												
				rw												

位 31:28	Res：保留 必须保持复位值。
位 27:16	DACC2DHR[11:0]：DAC 通道 2 的 12 位右对齐数据（DAC channel2 12-bit right-aligned data） 该位由软件写入，表示 DAC 通道 2 的 12 位数据。
位 15:12	Res：保留 必须保持复位值。
位 11:0	DACC1DHR[11:0]：DAC 通道 1 的 12 位右对齐数据（DAC channel1 12-bit right-aligned data） 该位由软件写入，表示 DAC 通道 1 的 12 位数据。

13.3.10 双 DAC 的 12 位左对齐数据保持寄存器（DAC_DHR12LD）

偏移地址：0x24

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC2DHR[11:0]												Res			
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Res			
rw															

位 31:20	DACC2DHR[11:0]：DAC 通道 2 的 12 位左对齐数据（DAC channel2 12-bit left-aligned data） 该位由软件写入，表示 DAC 通道 2 的 12 位数据。
位 19:16	Res：保留

	必须保持复位值。
位 15:4	DACC1DHR[11:0]：DAC 通道 1 的 12 位左对齐数据（DAC channel1 12-bit left-aligned data） 该位由软件写入，表示 DAC 通道 1 的 12 位数据。
位 3:0	Res：保留 必须保持复位值。

13.3.11 双 DAC 的 8 位右对齐数据保持寄存器（DAC_DHR8RD）

偏移地址：0x28

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[7:0]								DACC1DHR[7:0]							
rw								rw							

位 31:16	Res：保留 必须保持复位值。
位 15:8	DACC2DHR[7:0]：DAC 通道 2 的 8 位右对齐数据（DAC channel2 8-bit right-aligned data） 该位由软件写入，表示 DAC 通道 2 的 8 位数据。
位 7:0	DACC1DHR[7:0]：DAC 通道 1 的 8 位右对齐数据（DAC channel1 8-bit right-aligned data） 该位由软件写入，表示 DAC 通道 1 的 8 位数据。

13.3.12 DAC 通道 1 数据输出寄存器（DAC_DOR1）

偏移地址：0x2C

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res				DACC1DOR[11:0]											
				r											

位 31:12	Res：保留 必须保持复位值。
位 11:0	DACC1DOR[11:0]：DAC 通道 1 输出数据（DAC channel1 data output） 该位由软件写入，表示 DAC 通道 1 的输出数据。

13.3.13 DAC 通道 2 数据输出寄存器（DAC_DOR2）

偏移地址：0x30

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res				DACC2DOR[11:0]											
				r											

位 31:12	Res：保留 必须保持复位值。
位 11:0	DACC2DOR[11:0]：DAC 通道 2 输出数据（DAC channel2 data output） 该位由软件写入，表示 DAC 通道 2 的输出数据。

14 电压比较器（COMP）

器件具有 4 个电压比较器（Voltage comparator）COMP1、COMP2、COMP3 和 COMP4，这几个比较器可分别独立使用，也可以与定时器结合使用。

14.1 COMP 主要特性

电压比较器的功能如下：

- 分别从 8 个 IO 上输入 4 对电压到四个独立的电压比较器。
- 4 个比较器的比较结果可输出到 IO 或者作为定时器的事件输入。
  - COMP4 输出可以作为定时器 1 和 8 的 OCREF_CLR 事件。
  - COMP1、COMP2、COMP3 输出可以作为定时器 1 和 8 的 Break 事件。
- 比较器输出可以作为 EXTI 的输入事件唤醒系统和产生中断。
- 每个比较器具有可编程的迟滞电压：可选 0 或 30mV。
- 每个比较器可编程的速度和功耗模式：
  - 低速超低功耗模式，功耗低至 3 μA
  - 低速低功耗模式，典型功耗 5 μA
  - 中速中等功耗模式，典型功耗 40 μA
  - 高速高功耗模式，典型功耗 100 μA
- 比较器输出高低极性可配

电压比较模块使用时钟的是系统的 PCLK。

IO 口选择为模拟功能后就使能了电压比较模块的引脚。详细的引脚（COMPx_N）信息请参见数据手册的管脚定义章节。

14.2 COMP 功能描述

14.2.1 COMP 功能框图

比较器的功能框图如下图所示：

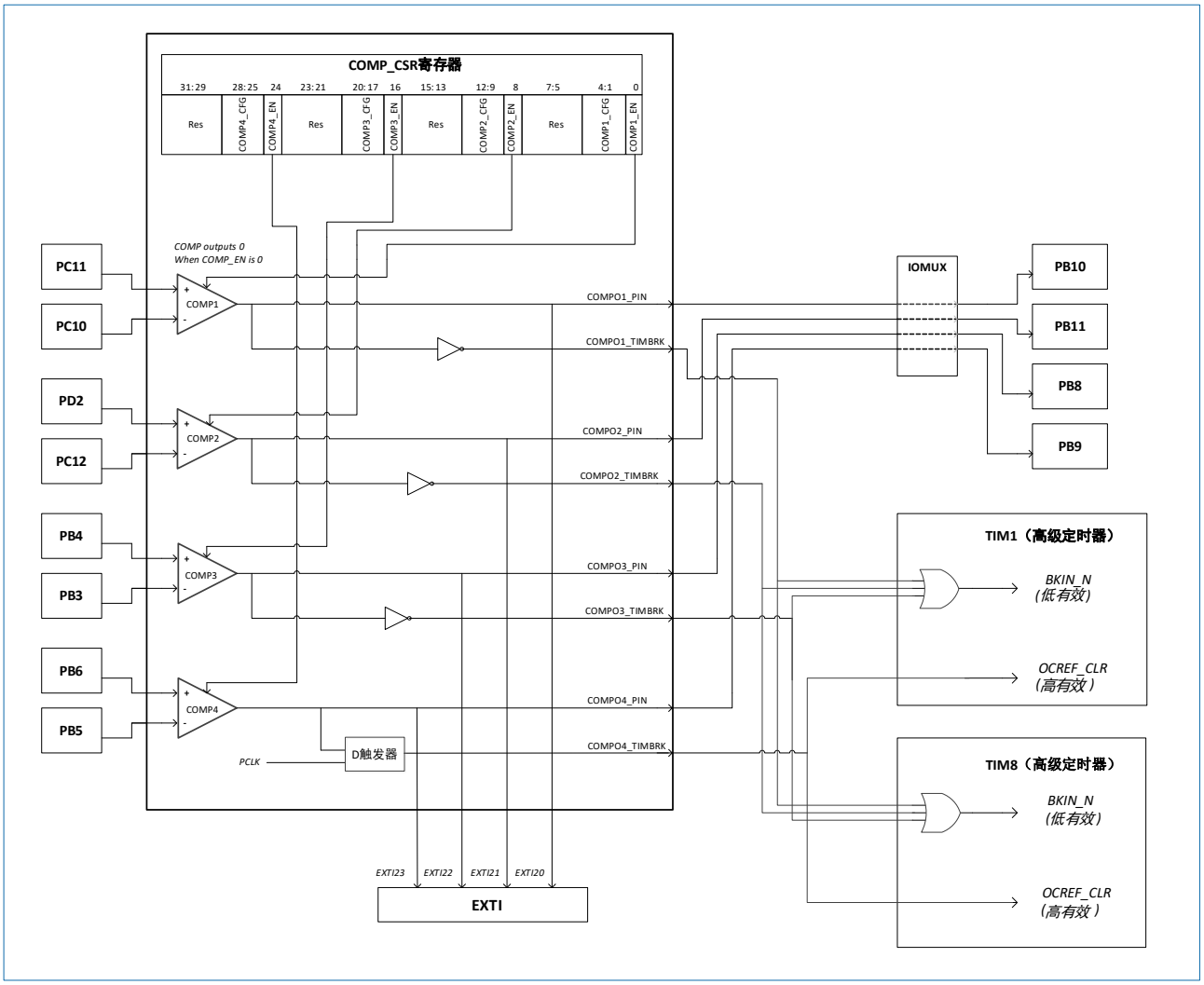


图 14-1 比较器功能框图

14.2.2 比较器的引脚配置和内部信号

- 用作比较器输入的 I/O 必须在 GPIO 寄存器中配置为模拟模式。
- 比较器输出连接的 IO 配置成复用输出功能。
- 输出可以配置为 IO 或者高级定时器的输入。

14.2.3 COMP 中断

比较器输出从内部连接到扩展中断和事件控制器。每个比较器都有其各自独立的 EXTI 线，能够产生中断或事件。

14.2.4 COMP 引脚信号

表 14-1 比较器引脚信号

比较器	COMPx_N	COMPx_P	COMPx_O
COMP1	PC10	PC11	PB10
COMP2	PC12	PD2	PB11
COMP3	PB3	PB4	PB8
COMP4	PB5	PB6	PB9

14.3 COMP 寄存器

基地址：0x4001 4000

空间大小：0x400

14.3.1 电压比较控制寄存器（COMP_CSR）

偏移地址：0x00

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res			COMP4_POL	COMP4_MODE[1:0]		COMP4_HYST	COMP4_EN	Res			COMP3_POL	COMP3_MODE[1:0]		COMP3_HYST	COMP3_EN
			rw	rw		rw	rw				rw	rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res			COMP2_POL	COMP2_MODE[1:0]		COMP2_HYST	COMP2_EN	Res			COMP1_POL	COMP1_MODE[1:0]		COMP1_HYST	COMP1_EN
			rw	rw		rw	rw				rw	rw		rw	rw

位 31:29	Res：保留 必须保持复位值。
位 28	COMP4_POL：比较器 4 的输出极性选择（Comparator 4 polarity selection bit） • 0：VINP>VINN 时输出高，反之输出低（默认）。 • 1：VINP>VINN 时输出低，反之输出高。
位 27:26	COMP4_MODE[1:0]：比较器 4 的模式选择（Comparator 4 mode selection） • 00：低速超低功耗模式，功耗低至 3 μA。 • 01：低速低功耗模式，典型功耗为 5 μA。 • 10：中速中等功耗模式，典型功耗为 40 μA。 • 11：高速高功耗模式，典型功耗为 100 μA 。
位 25	COMP4_HYST：比较器 4 的迟滞电压选择（Comparator 4 hysteresis selection bit） • 0：无迟滞电压（默认） • 1：30mV 迟滞电压
位 24	COMP4_EN：比较器 4 开关控制（Comparator 4 enable bit） • 0：比较器关闭（默认） • 1：比较器打开
位 23:21	Res：保留 必须保持复位值。
位 20	COMP3_POL：比较器 3 的输出极性选择（Comparator 3 polarity selection bit） • 0：VINP>VINN 时输出高，反之输出低（默认） • 1：VINP>VINN 时输出低，反之输出高
位 19:18	COMP3_MODE[1:0]：比较器 3 的模式选择（Comparator 3 mode selection） • 00：低速超低功耗模式，功耗低至 3 μA。 • 01：低速低功耗模式，典型功耗为 5 μA。 • 10：中速中等功耗模式，典型功耗为 40 μA。

	• 11：高速高功耗模式，典型功耗为 100 μA。
位 17	COMP3_HYST：比较器 3 的迟滞电压选择（Comparator 3 hysteresis selection bit） • 0：无迟滞电压（默认） • 1：30mV 迟滞电压
位 16	COMP3_EN：比较器 3 开关控制（Comparator 3 enable bit） • 0：比较器关闭（默认） • 1：比较器打开
位 15:13	Res：保留 必须保持复位值。
位 12	COMP2_POL：比较器 2 的输出极性选择（Comparator 2 polarity selection bit） • 0：VINP>VINN 时输出高，反之输出低（默认）。 • 1：VINP>VINN 时输出低，反之输出高。
位 11:10	COMP2_MODE[1:0]：比较器 2 的模式选择（Comparator 2 mode selection） • 00：低速超低功耗模式，功耗低至 3 μA。 • 01：低速低功耗模式，典型功耗为 5 μA。 • 10：中速中等功耗模式，典型功耗为 40 μA。 • 11：高速高功耗模式，典型功耗为 100 μA
位 9	COMP2_HYST：比较器 2 的迟滞电压选择（Comparator 2 hysteresis selection bit） • 0：无迟滞电压（默认） • 1：30mV 迟滞电压
位 8	COMP2_EN：比较器 2 开关控制（Comparator 2 enable bit） • 0：比较器关闭（默认） • 1：比较器打开
位 7:5	Res：保留 必须保持复位值。
位 4	COMP1_POL：比较器 1 的输出极性选择（Comparator 1 polarity selection bit） • 0：VINP>VINN 时输出高，反之输出低（默认）。 • 1：VINP>VINN 时输出低，反之输出高。
位 3:2	COMP1_MODE[1:0]：比较器 1 的模式选择（Comparator 1 mode selection） • 00：低速超低功耗模式，功耗低至 3 μA。 • 01：低速低功耗模式，典型功耗为 5 μA。 • 10：中速中等功耗模式，典型功耗为 40 μA。 • 11：高速高功耗模式，典型功耗为 100 μA。
位 1	COMP1_HYST：比较器 1 的迟滞电压选择（Comparator 1 hysteresis selection bit） • 0：无迟滞电压（默认） • 1：30mV 迟滞电压
位 0	COMP1_EN：比较器 1 开关控制（Comparator 1 enable bit）

	• 0：比较器关闭（默认） • 1：比较器打开

15 灵活的静态存储器控制器（FSMC）

15.1 FSMC 功能

FSMC 模块有同步或异步存储器和 16 位 PC 存储器卡的接口，它的主要作用是：

- 将 AHB 事务转换为合适的外部设备协议
- 满足访问外部设备的时序要求

所有的外部存储器共享控制器输出的地址、数据和控制信号。可通过唯一的片选信号选择待访问的外部设备。FSMC 在任一时刻只访问一个外部设备。

FSMC 具有下列主要功能：

- 具有静态存储器接口的器件包括：
  - 静态随机访问存储器（SRAM）
  - 只读存储器（ROM）
  - NOR Flash
  - PSRAM（4 个存储器块）
- 两个 NAND Flash 块，支持硬件 ECC 并可检测多达 8 Kbyte 数据。
- 支持 16 位的 PC 卡兼容设备
- 支持对同步器件的成组（Burst）访问模式，例如 NOR Flash 和 PSRAM。
- 8 或 16 位数据总线
- 每一个存储器块都有独立的片选控制。
- 每一个存储器块都可以独立配置。
- 时序可编程以支持各种不同的器件：
  - 等待周期可编程（多达 15 个周期）
  - 总线恢复周期可编程（多达 15 个周期）
  - 输出使能和写使能延迟可编程（多达 15 周期）
  - 独立的读写时序和协议，可支持宽范围的存储器和时序；
- 使用 PSRAM 和 SRAM 器件时，支持写使能和字节片选输出
- 将 32 位宽 AHB 事务转换成连续的 16 位/8 位，访问 16 位/8 位的外部设备。
- 2 个字的写 FIFO，每个字为 32 位宽，允许在写入较慢存储器时释放 AHB 进行其它操作。在开始一次新的 FSMC 操作前，FIFO 要先被清空。FSMC 将插入等待周期直到当前的存储器访问完成。
- 支持外部存储器的异步等待控制逻辑
- 支持对 16 位外部存储器进行写数据加密和读数据解密
- 支持 Intel 8080 模式和 Motorola 6800 模式，可以灵活与各种 LCD 控制器连接

FSMC 寄存器定义了外部设备的类型和在系统启动时需设置相关的特性，这些特性直到下一次系统复位或上电之前都保持不变。然而，可以在任何时候修改这些设置。

15.2 FSMC 框图

FSMC 包含四个主要模块：

- AHB 接口（包含 FSMC 配置寄存器）
- NOR Flash 和 PSRAM 控制器

- NAND Flash 和 PC 卡控制器
- 外部设备接口

FSMC 框图如下：

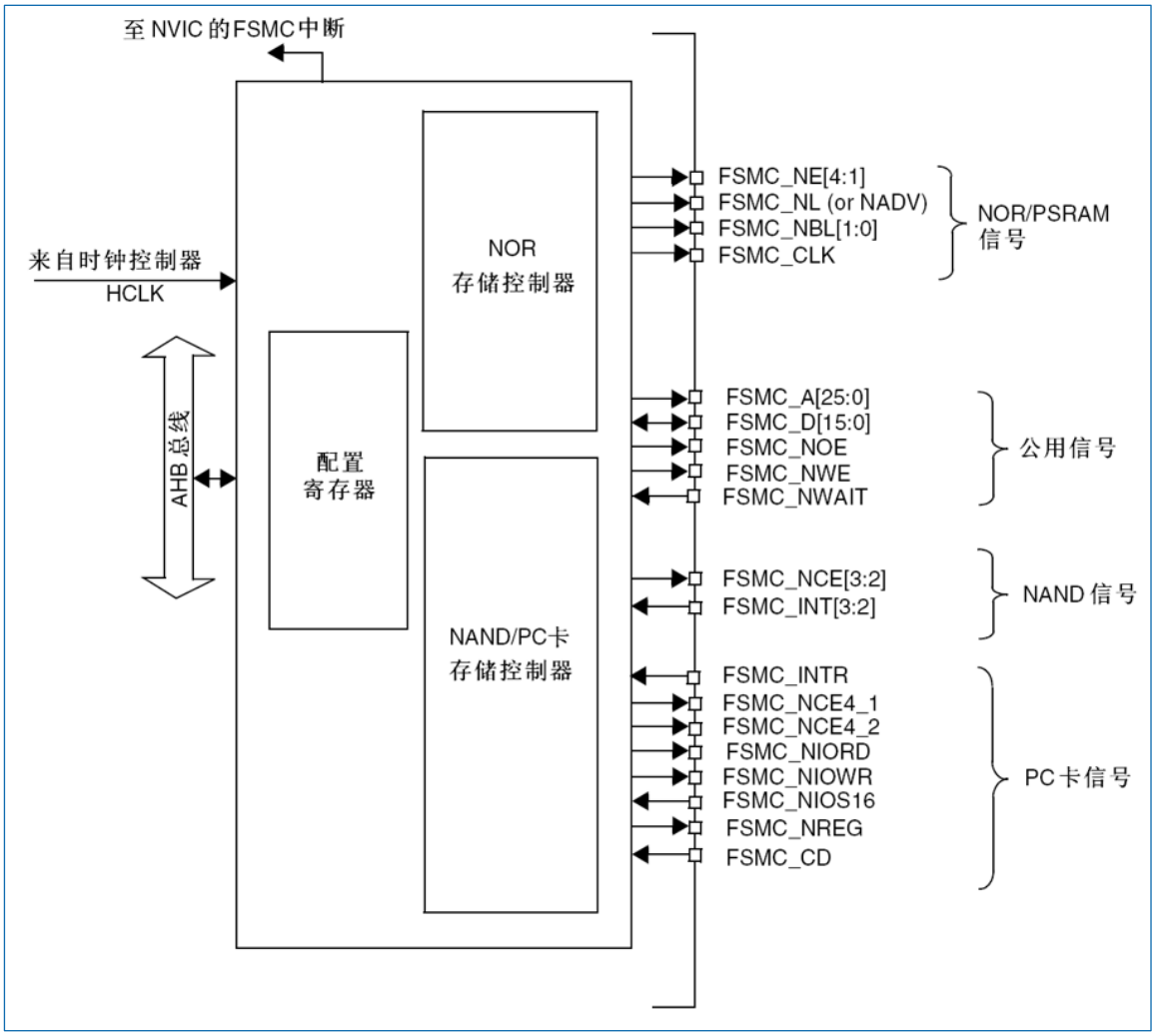


图 15-1 FSMC 框图

15.3 AHB 接口

AHB 作为从接口使得内部 CPU 和其它总线主外设可以访问外部静态存储器。

AHB 事务被转换为外部设备协议。当选择的外部存储器的数据位宽是 16 或 8 位时，在 AHB 上的 32 位数据会被分割成连续的 16 或 8 位的操作。在连续访问期间分别执行 32 位对齐或 32 位非对齐访问时，片选信号保持为低或进行翻转。

AHB 时钟（HCLK）是 FSMC 的参考时钟。

在下述情况中，FSMC 会产生一个 AHB 错误：

- 当读或写某个未被使能的 FSMC 块
- 当读或写某个 NOR Flash 块，而在 FSMC_BCRx 寄存器中的 FACCEN 位被复位了。
- 当读或写某个 PC 卡的块，而 FSMC_CD（卡是否存在的检测）输入引脚为低。

AHB 故障由试图进行读/写访问的 AHB 主器件决定。

- 如果是 Cortex® -M3 CPU，将产生硬件错误中断。
- 如果是 DMA，将产生 DMA 传输错误，并且 DMA 通道自动被禁用。

15.3.1 支持的存储器和操作

15.3.1.1 一般的操作规则

请求 AHB 操作的数据宽度可以是 8 位、16 位或 32 位，而外部设备则是固定的数据宽度。这样可保障实现数据传输的一致性。

因此，FSMC 操作需遵循下述操作规则：

- AHB 操作的数据宽度与存储器数据宽度相同：此种情况无数据传输一致性的问题。
- AHB 操作的数据宽度大于存储器的数据宽度：此时 FSMC 将 AHB 操作分割成几个连续的较小数据宽度的存储器操作，以适应外部设备的数据宽度。
- AHB 操作的数据宽度小于存储器的数据宽度：依据外部设备的类型，异步的数据传输有可能不一致。
    - 与具有字节选择功能的存储器（SRAM、ROM、PSRAM）进行异步传输时，FSMC 执行读写操作并通过它的字节通道 NBL[1:0]访问正确的数据。
    - 与不具有字节选择功能的存储器（NOR 和 16 位 NAND 等）进行异步传输时，即需要对 16 位宽的 Flash 存储器进行字节访问；显然不能对存储器进行字节模式访问（只允许 16 位的数据传输），因此：
        - 不允许进行写操作。
        - 可以进行读操作（控制器读出完整的 16 位存储器数据，无用的字节将被丢弃。在读操作期间，NBL[1:0]将被设置为 0）。

15.3.1.2 配置寄存器

FSMC 由一组寄存器进行配置。15.6 章节详细描述了 NAND Flash 和 PC 卡寄存器。

15.4 外部设备地址映射

从 FSMC 的角度来说，可以把外部存储器划分成固定大小为 256 Mbyte 的四个存储块，见图 15-2。

- 存储块 1 用于访问最多 4 个 NOR Flash 或 PSRAM 存储设备。这个存储区被划分为 4 个 NOR/PSRAM 区并有 4 个专用的片选。
    - 块 1：NOR/PSRAM1
    - 块 1：NOR/PSRAM2
    - 块 1：NOR/PSRAM3
    - 块 1：NOR/PSRAM4
- 存储块 2 和 3 用于访问 NAND Flash 设备，每个存储块连接一个 NAND Flash。
- 存储块 4 用于访问 PC 卡设备。

每一个存储块上的存储器类型是由用户在配置寄存器中定义的。

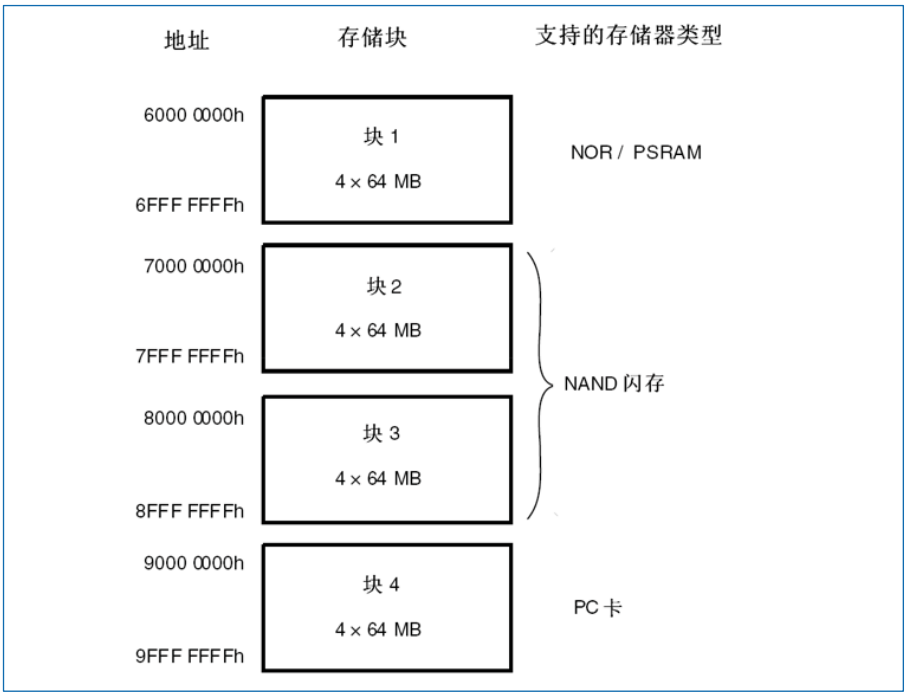


图 15-2 FSMC 存储块

15.4.1 NOR 和 PSRAM 地址映射

HADDR[27:26]位可用于选择四个存储块之一：

表 15-1 NOR/PSRAM 存储块选择

HADDR[27:26] (1)	选择的存储块
00	存储块 1-NOR/PSRAM1
01	存储块 1-NOR/PSRAM2
10	存储块 1-NOR/PSRAM3
11	存储块 1-NOR/PSRAM4

(1). HADDR 是需要转换到外部存储器的内部 AHB 地址线。

HADDR[25:0]包含外部存储器地址。HADDR 是字节地址，而存储器是按字寻址访问，实际发给内存的地址根据内存数据宽度而变化，如下表：

表 15-2 外部存储器地址

数据宽度 (1)	连到存储器的地址线	最大访问存储器空间（位）
8 位	HADDR[25:0]与 FSMC_A[25:0]对应相连	64 Mbyte x 8=512 Mbit
16 位	HADDR[25:1]与 FSMC_A[24:0]对应相连，HADDR[0]未接	64 Mbyte /2 x 16=512 Mbit

(1). 对于 16 位宽度的外部存储器，FSMC 将在内部使用 HADDR[25:1]产生外部存储器的地址 FSMC_A[24:0]。不论外部存储器的宽度是多少（16 位或 8 位），FSMC_A[0]始终应该连到外部存储器的地址线 A[0]。

15.4.1.1 NOR Flash 和 PSRAM 的非对齐访问支持

每个 NOR Flash 或 PSRAM 存储器块都可以配置成支持非对齐的数据访问。在存储器一侧，依据访问的方式是异步或同步，需要考虑两种情况：

- 异步模式

  该模式下，只要每次访问都有准确的地址，完全支持非对齐的数据访问。

- 同步模式

  该模式下，FSMC 只发出一次地址信号，然后成组的数据传输通过时钟 CLK 顺序进行。某些 NOR 存储器支持线性的非对齐成组访问，固定数目的数据字可以从连续的以 N 为模的地址读取（典型的 N 为 8 或 16，可以通过 NOR Flash 的配置寄存器设置）。此种情况下，可以把存储器的非对齐访问模式设置为与 AHB 相同的模式。

如果存储器的非对齐访问模式不能设置为与 AHB 相同的模式，应该通过 FSMC 配置寄存器的相应位禁止非对齐访问，并把非对齐的访问请求分开成两个连续的访问操作。

15.4.2 NAND 和 PC 卡地址映射

三个存储块可以用于 NAND 或 PC 卡的操作，每个存储块被划分为下述访问空间：

表 15-3 存储器映射和时序寄存器

AHB 起始地址	AHB 结束地址	FSMC 存储块	存储空间	时序寄存器
0x9C00 0000	0x9FFF FFFF	块 4–PC 卡	I/O	FSMC_PIO4（0xB0）
0x9800 0000	0x9BFF FFFF		属性	FSMC_PATT4（0xAC）
0x9000 0000	0x93FF FFFF		通用	FSMC_PMEM4（0xA8）
0x8800 0000	0x8BFF FFFF	块 3–NAND Flash	属性	FSMC_PATT3（0x8C）
0x8000 0000	0x83FF FFFF		通用	FSMC_PMEM3（0x88）
0x7800 0000	0x7BFF FFFF	块 2–NAND Flash	属性	FSMC_PATT2（0x6C）
0x7000 0000	0x73FF FFFF		通用	FSMC_PMEM2（0x68）

对于 NAND Flash 存储器，通用和属性存储空间在低 256 Kbyte 部分可划分为 3 个区（见表 15-4）：

- 数据区（通用/属性空间的前 64 Kbyte 区）
- 命令区（通用/属性空间的第 2 个 64 Kbyte 区）
- 地址区（通用/属性空间的下一个 128 Kbyte 区）

表 15-4 NAND 存储块选择

区域名称	HADDR[17:16]	地址范围
地址区	1X	0x020000~0x03FFFF
命令区	01	0x010000~0x01FFFF
数据区	00	0x000000~0x00FFFF

应用软件使用这 3 个区访问 NAND Flash 存储器：

- 发送命令到 NAND Flash 存储器

  软件只需对命令区的任意一个地址写入命令即可。

- 指定操作 NAND Flash 存储器的读写地址

  软件只需对地址区的任意一个地址写入地址的值即可。因为一个 NAND 地址可以是 4 或 5 个字节的长度（依实际的存储器容量而定），指定一个完整的地址需要多次连续地写入地址区。

- 读写数据

  软件只需对数据区的任意一个地址写入或读出数据即可。

  因为 NAND Flash 存储器自动地累加其内部的操作地址，访问连续的存储器位置无需累加数据区的地址。

15.5 NOR Flash 和 PSRAM 控制器

FSMC 可以产生适当的信号时序，以驱动下述类型的存储器：

- 异步 SRAM 和 ROM
  - 8 位
  - 16 位
  - 32 位
- PSRAM（Cellular RAM）
  - 异步模式
  - 突发同步模式
- NOR Flash
  - 异步模式
  - 突发异步模式
  - 复用模式或非复用模式

FSMC 对每个存储块输出一个唯一的片选信号 NE[4:1]，所有其它的（地址、数据和控制）信号则是共享的。

对于同步访问，在读/写操作时，FSMC 为选中的外部设备提供时钟（CLK）。该时钟的频率是 HCLK 时钟的整除因子。每个存储块的大小固定为 64 Mbyte。

每个存储块都由专门的寄存器控制（见“15.7.2 NAND Flash 和 PC 卡控制器寄存器”）。

可编程的存储器参数包括访问时序（见表 15-5）、是否支持非对齐数据存取和等待周期管理（只针对突发模式下访问 PSRAM 和 NOR Flash）。

表 15-5 可编程的 NOR/PSRAM 访问参数

参数	功能	访问方式	单位	最小值	最大值
地址建立时间	地址建立阶段的时间	异步	AHB 时钟周期（HCLK）	1	16
地址保持时间	地址保持阶段的时间	异步， 复用 I/O	AHB 时钟周期（HCLK）	2	16
数据建立时间	数据建立阶段的时间	异步	AHB 时钟周期（HCLK）	2	256
总线恢复时间	总线恢复阶段的时间	异步或同步读	AHB 时钟周期（HCLK）	1	16
时钟分频因子	存储器访问的时钟周期（CLK）与 AHB 时钟周期的比例	同步	AHB 时钟周期（HCLK）	2	16
数据产生时间	突发模式下产生第一个数据所需的时钟数目	同步	存储器时钟周期（CLK）	2	17

15.5.1 外部存储器接口信号

表 15-6 至表 15-8 分别列出了与 NOR Flash、SRAM 和 PSRAM 接口的典型信号。

说明：带前缀"N"的信号表示低有效信号。

NOR Flash，非复用接口：

表 15-6 非复用信号的 NOR Flash 接口

FSMC 信号名称	信号方向	功能
CLK	输出	时钟（同步突发模式使用）
A[25:0]	输出	地址总线
D[15:0]	输入/输出	双向数据总线
NE[x]	输出	片选，x=1..4
NOE	输出	输出使能
NWE	输出	写使能
NWAIT	输入	NOR Flash 要求 FSMC 等待的信号

NOR Flash 存储器是按 16 位的半字寻址，最大容量达 64 Mbyte（26 条地址线）。

NOR Flash，复用接口：

表 15-7 复用 NOR Flash 接口

FSMC 信号名称	信号方向	功能
CLK	输出	时钟（同步突发模式使用）
A[25:16]	输出	地址总线
AD[15:0]	输入/输出	16 位复用的，双向地址/数据总线
NE[x]	输出	片选，x=1~4
NOE	输出	输出使能
NWE	输出	写使能
NL（=NADV）	输出	锁存使能（某些 NOR Flash 器件将该信号命名为地址有效，NADV）
NWAIT	输入	NOR Flash 输出到 FSMC 的等待信号

NOR Flash 存储器是按 16 位的半字寻址，最大容量达 64 Mbyte（26 条地址线）。

15.5.1.1 PSRAM

表 15-8 非复用信号的 PSRAM 接口

FSMC 信号名称	信号方向	功能
CLK	输出	时钟（同步突发模式使用）
A[25:0]	输出	地址总线
D[15:0]	输入/输出	双向数据总线
NE[x]	输出	片选，x=1~4（PSRAM 称其为 NCE（Cellular RAM，如 CRAM））

FSMC 信号名称	信号方向	功能
NOE	输出	输出使能
NWE	输出	写使能
NL（=NADV）	输出	地址有效（存储器信号名称为：NADV）
NWAIT	输入	PSRAM 要求 FSMC 等待的信号
NBL[1]	输出	高字节使能（存储器信号名为：NUB）
NBL[0]	输出	低字节使能（存储器信号名为：NLB）

PSRAM 存储器是按 16 位的半字寻址，最大容量达 64 Mbyte（26 条地址线）。

15.5.2 支持的存储器及其操作

表 15-9 列出了当存储器数据总线宽度为 16 位时，支持的存储器、访问模式和操作方式，FSMC 不支持阴影部分的操作方式。

表 15-9 FSMC 支持的 NOR Flash/PSRAM 存储器和操作方式

存储器	模式	读/写	AHB 数据宽度	存储器数据宽度	是否支持	注释
NOR Flash（数据总线复用和非总线复用）	异步	读	8	16	支持	
	异步	写	8	16	不支持	
	异步	读	16	16	支持	
	异步	写	16	16	支持	
	异步	读	32	16	支持	分成 2 次 FSMC 访问
	异步	写	32	16	支持	分成 2 次 FSMC 访问
	异步页	读	-	16	不支持	不支持这种模式
	同步	读	8	16	不支持	
	同步	读	16	16	支持	
	同步	读	32	16	支持	
PSRAM（数据总线复用和非总线复用）	异步	读	8	16	支持	
	异步	写	8	16	支持	使用字节信号 NBL[1:0]
	异步	读	16	16	支持	
	异步	写	16	16	支持	
	异步	读	32	16	支持	分成 2 次 FSMC 访问
	异步	写	32	16	支持	分成 2 次 FSMC 访问
	异步页	读	-	16	不支持	不支持这种模式
	同步	读	8	16	不支持	

存储器	模式	读/写	AHB 数据宽度	存储器数据宽度	是否支持	注释
	同步	读	16	16	支持	
	同步	读	32	16	支持	
	同步	写	8	16	支持	使用字节信号 NBL[1:0]
	同步	写	16/32	16	支持	
SRAM 和 ROM	异步	读	8/16	16	支持	使用字节信号 NBL[1:0]
	异步	写	8/16	16	支持	使用字节信号 NBL[1:0]
	异步	读	32	16	支持	分成 2 次 FSMC 访问
	异步	写	32	16	支持	分成 2 次 FSMC 访问

15.5.3 时序规则

信号同步

- 所有的控制器输出信号在内部时钟（HCLK）的上升沿变化。
- 在同步读/写模式（PSRAM）下，输出的数据在存储器时钟（CLK）的下降沿变化。

15.5.4 NOR Flash 和 PSRAM 控制器异步传输

异步静态存储器（NOR Flash 和 PSRAM）

- 所有信号由内部时钟 HCLK 来同步，但该时钟不会输出到存储器。
- FSMC 始终在片选信号 NE 失效前对数据线采样，这样能够保证符合存储器的数据保持时序（片选失效至数据失效的间隔，通常最小为 0 ns）。
- 当设置了扩展模式（通过配置 FSMC_BCRx 寄存器的 EXTMOD 位），至多可用 4 种扩展模式（A、B、C 和 D）。可以在读和写时混合使用模式 A、B、C 和 D（例如，允许以模式 A 进行读，而以模式 B 进行写）。

15.5.4.1 模式 1-SRAM/CRAM

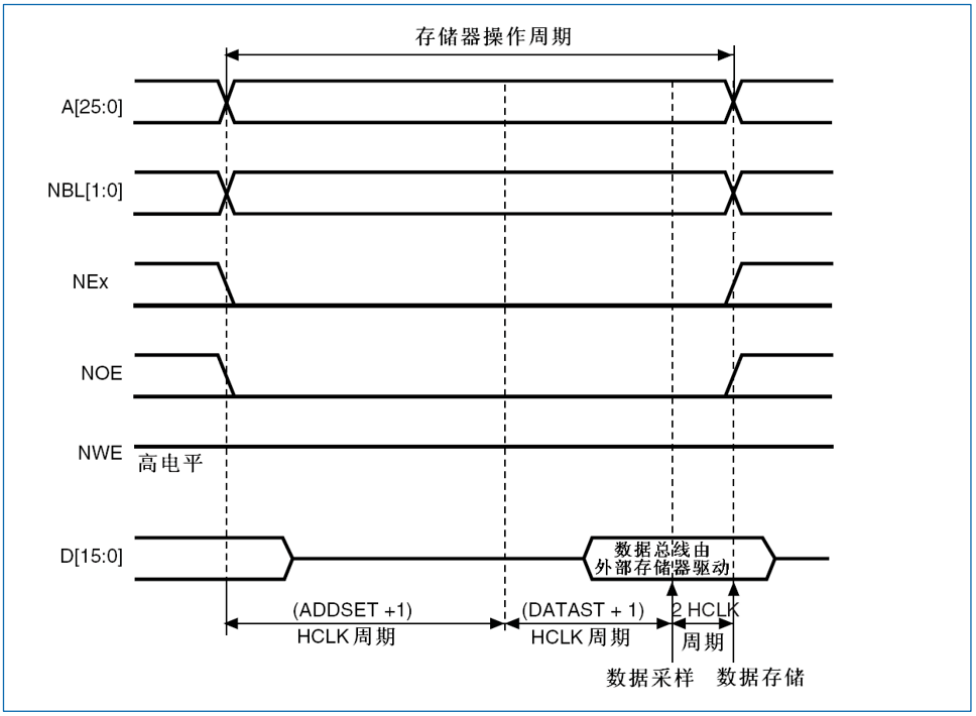


图 15-3 模式 1 读操作

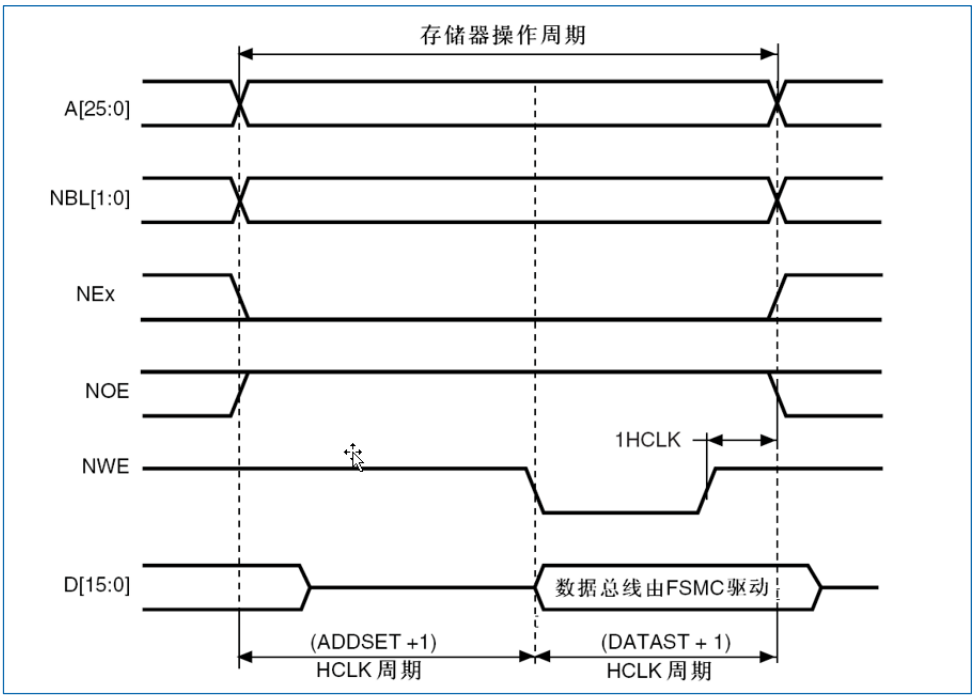


图 15-4 模式 1 写操作

在写操作的最后一个 HCLK 周期可以保证 NWE 上升沿后地址和数据的保持时间，因为存在这个 HCLK 周期，DATAST 的数值必须大于 0（DATAST>0）。

表 15-10 FSMC_BCRx 位域（模式 1）

位编号	位名称	设置的数值
31:20	Res	0x0000
19	CBURSTRW	0x0 （对异步模式无影响）

位编号	位名称	设置的数值
18:16	Res	0x0
15	ASYNCWAIT	如果存储器支持该模式则置‘1’，否则保持为‘0’。
14	EXTMOD	0x0
13	WAITEN	0x0（对异步模式无影响）
12	WREN	需要时设置
11	WAITCFG	忽略
10	WRAPMOD	0x0
9	WAITPOL	仅当位 15 为’1’时有意义
8	BURSTEN	0x0
7	Res	0x1
6	FACCEN	忽略
5:4	MWID	需要时设置
3:2	MTYP	需要时设置，不包含 10（NOR Flash）
1	MUXEN	0x0
0	MBKEN	0x1

表 15-11 FSMC_BTRx 位域（模式 1）

位编号	位名称	设置的数值
31:30	Res	0x0
29:28	ACCMOD	忽略
27:24	DATLAT	忽略
23:20	CLKDIV	忽略
19:16	BUSTURN	NEx 从高到低的时间 （BUSTURN HCLK）
15:8	DATAST	操作的第 2 个阶段的长度，写操作为（DATAST+1 个 HCLK 周期），读操作为（DATAST+3 个 HCLK 周期）。这个域不能为 0，至少为 1
7:4	ADDHLD	忽略
3:0	ADDSET	操作的第 1 个阶段的长度（ADDSET+1 个 HCLK 周期）

15.5.4.2 模式 A-SRAM/PSRAM（CRAM）OE 翻转

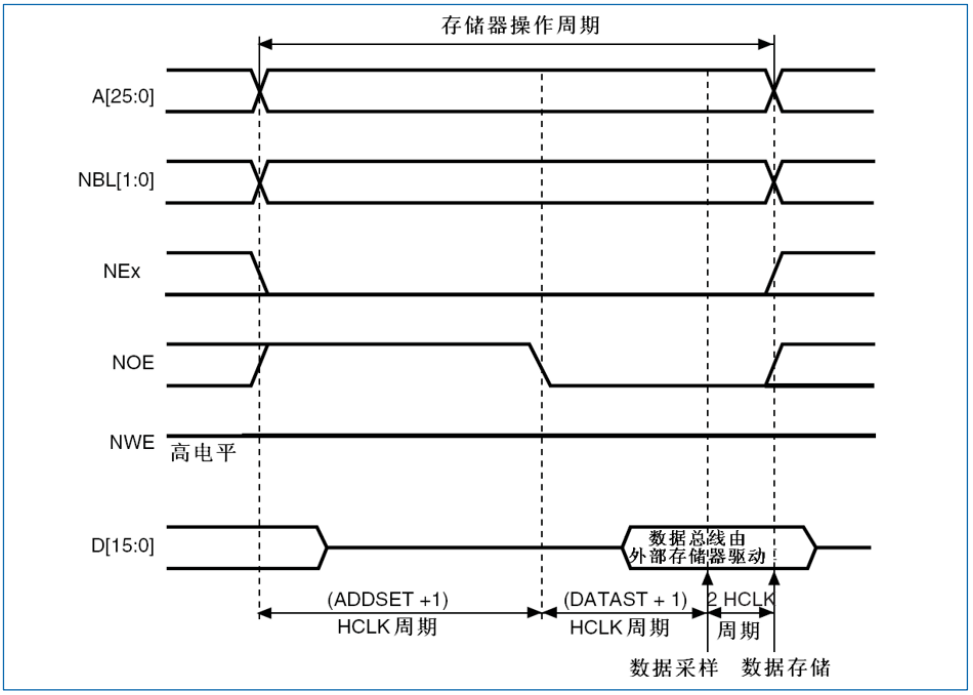


图 15-5 模式 A 读操作

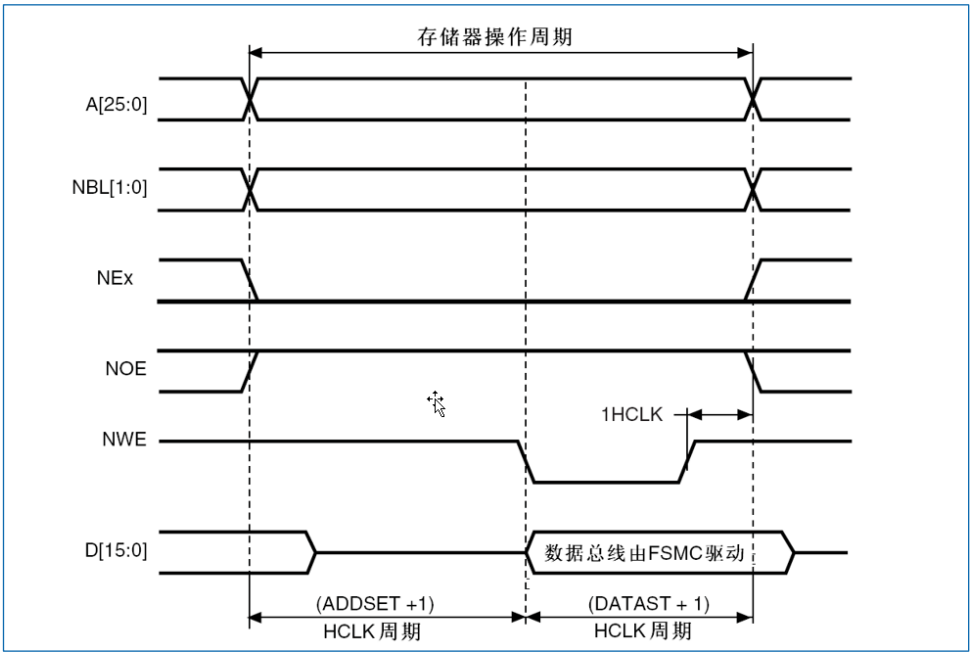


图 15-6 模式 A 写操作

模式 A 与模式 1 的区别是 NOE 的变化和相互独立的读写时序。

表 15-12 FSMC_BCRx 位域（模式 A）

位编号	位名称	设置的数值
31:20	Res	0x000
19	CBURSTRW	0x0（对异步模式无影响）
18:16	Res	0x0

位编号	位名称	设置的数值
15	ASYNCWAIT	如果存储器支持该模式则置为'1'，否则保持为'0'
14	EXTMOD	0x1
13	WAITEN	0x0 （对异步模式无影响）
12	WREN	需要时设置
11	WAITCFG	忽略
10	WRAPMOD	0x0
9	WAITPOL	仅当位 15 为'1'时有意义
8	BURSTEN	0x0
7	Res	0x1
6	FACCEN	忽略
5:4	MWID	需要时设置
3:2	MTYP	需要时设置，不包含 10（NOR Flash）
1	MUXEN	0x0
0	MBKEN	0x1

表 15-13 FSMC_BTRx 位域（模式 A）

位编号	位名称	设置的数值
31:30	Res	0x0
29:28	ACCMOD	0x0
27:24	DATLAT	忽略
23:20	CLKDIV	忽略
19:16	BUSTURN	NEx 从高到低的时间 （BUSTURN HCLK）
15:8	DATAST	读操作的第 2 个阶段的长度（DATAST+3 个 HCLK 周期）。这个域不能为 0（至少为 1）
7:4	ADDHLD	忽略
3:0	ADDSET	读操作的第 1 个阶段的长度（ADDSET+1 个 HCLK 周期）

表 15-14 FSMC_BWTRx 位域（模式 A）

位编号	位名称	设置的数值
31:30	Res	0x0
29:28	ACCMOD	0x0
27:20	Res	0xFF
19:16	BUSTURN	NEx 从高到低的时间 （BUSTURN HCLK）

位编号	位名称	设置的数值
15:8	DATAST	写操作的第 2 个阶段的长度（DATAST+1 个 HCLK 周期）。这个域不能为 0（至少为 1）
7:4	ADDHLD	忽略
3:0	ADDSET	写操作的第 1 个阶段的长度（ADDSET+1 个 HCLK 周期）

15.5.4.3 模式 2/B-NOR Flash

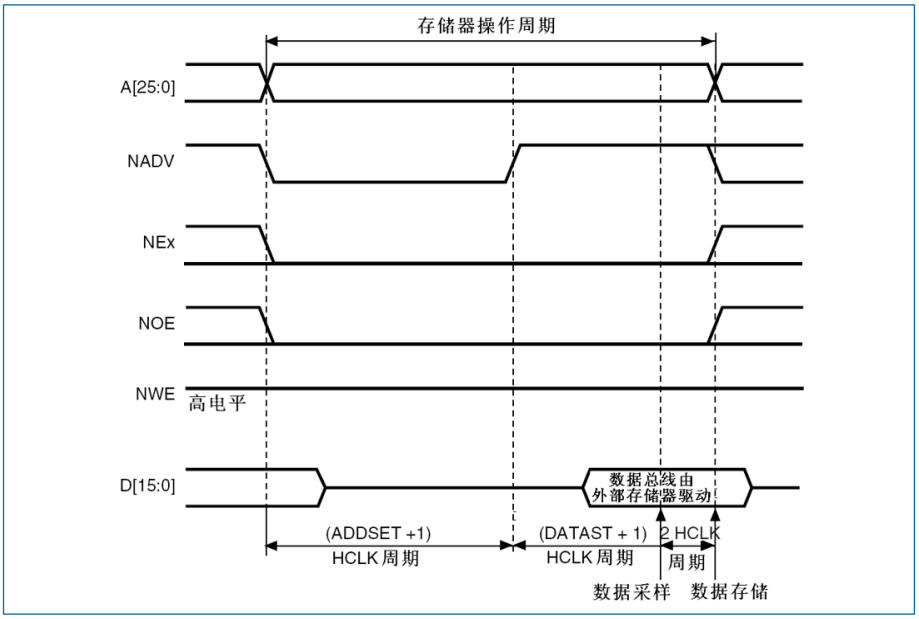


图 15-7 模式 2/B 读操作

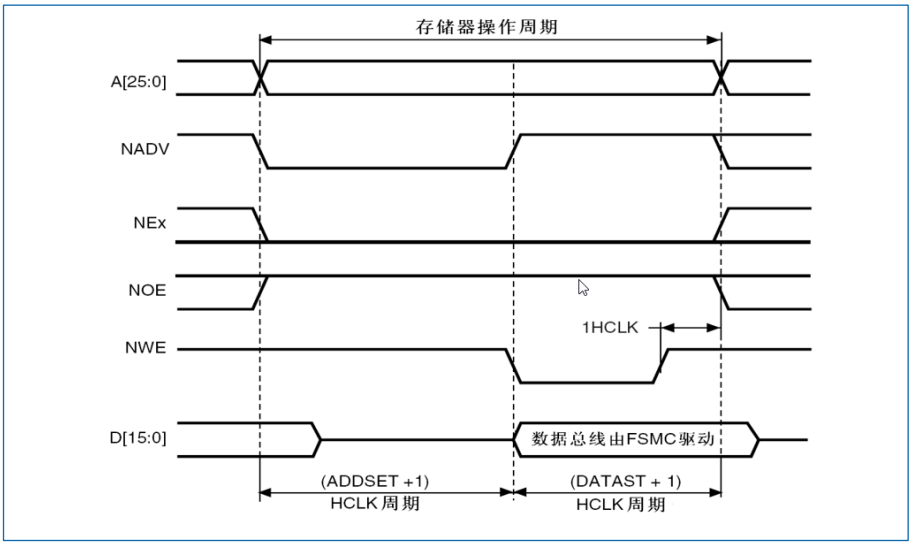


图 15-8 模式 2 写操作

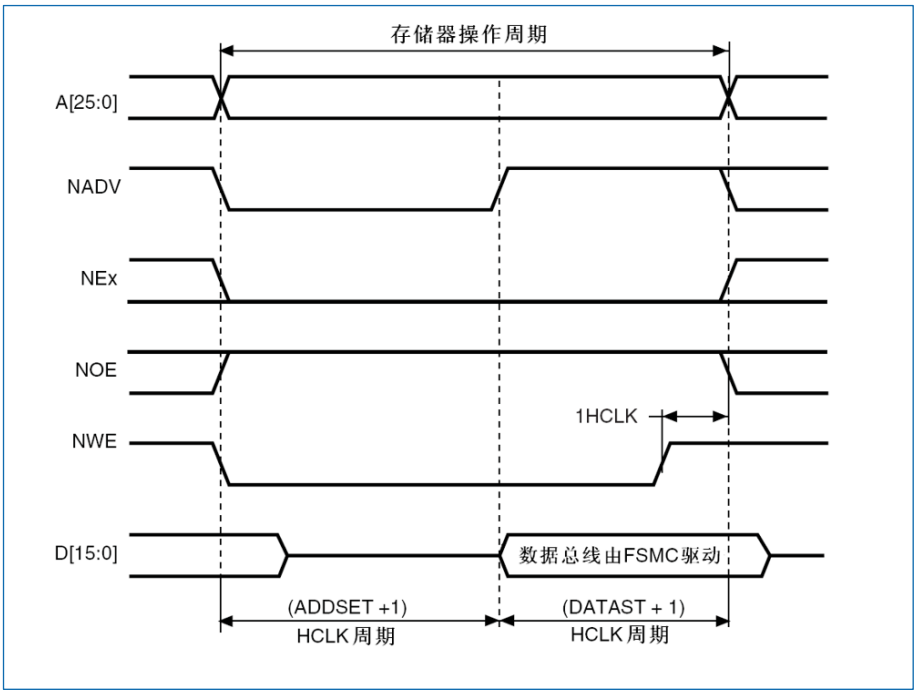


图 15-9 模式 B 写操作

模式 2/B 与模式 1 相比较，不同的是 NADV 的变化，且在扩展模式下（模式 B）读写时序相互独立。

表 15-15 FSMC_BCRx 位域（模式 2/B）

位编号	位名称	设置的数值
31:20	Res	0x000
19	CBURSTRW	0x0（对异步模式无影响）
18:16	Res	0x0
15	ASYNCWAIT	如果存储器支持该模式则置为’1’，否则保持为’0’
14	EXTMOD	模式 B：0x1；模式 2：0x0
13	WAITEN	0x0（对异步模式无影响）
12	WREN	需要时设置
11	WAITCFG	忽略
10	WRAPMOD	0x0
9	WAITPOL	仅当位 15 为’1’时有意义。
8	BURSTEN	0x0
7	Res	0x1
6	FACCEN	0x1
5:4	MWID	需要时设置
3:2	MTYP	0x2（NOR Flash）
1	MUXEN	0x0

位编号	位名称	设置的数值
0	MBKEN	0x1

表 15-16 FSMC_BTRx 位域（模式 2/B）

位编号	位名称	设置的数值
31:30	Res	0x0
29:28	ACCMOD	0x1
27:24	DATLAT	忽略
23:20	CLKDIV	忽略
19:16	BUSTURN	NEx 从高到低的时间 （BUSTURN HCLK）
15:8	DATAST	读操作的第 2 个阶段的长度（DATAST+3 个 HCLK 周期）。这个域不能为 0（至少为 1）。
7:4	ADDHLD	忽略
3:0	ADDSET	读操作的第 1 个阶段的长度（ADDSET+1 个 HCLK 周期）。

表 15-17 FSMC_BWTRx 位域（模式 2/B）

位编号	位名称	设置的数值
31:30	Res	0x0
29:28	ACCMOD	0x1
27:20	Res	0xFF
19:16	BUSTURN	NEx 从高到低的时间 （BUSTURN HCLK）
15:8	DATAST	写操作的第 2 个阶段的长度（DATAST+1 个 HCLK 周期）。这个域不能为 0（至少为 1）
7:4	ADDHLD	忽略
3:0	ADDSET	写操作的第 1 个阶段的长度（ADDSET+1 个 HCLK 周期）

说明：只有当设置了扩展模式时（模式 B），FSMC_BWTRx 才有效，否则该寄存器的内容不起作用。

15.5.4.4 模式 C-NOR Flash-OE 翻转

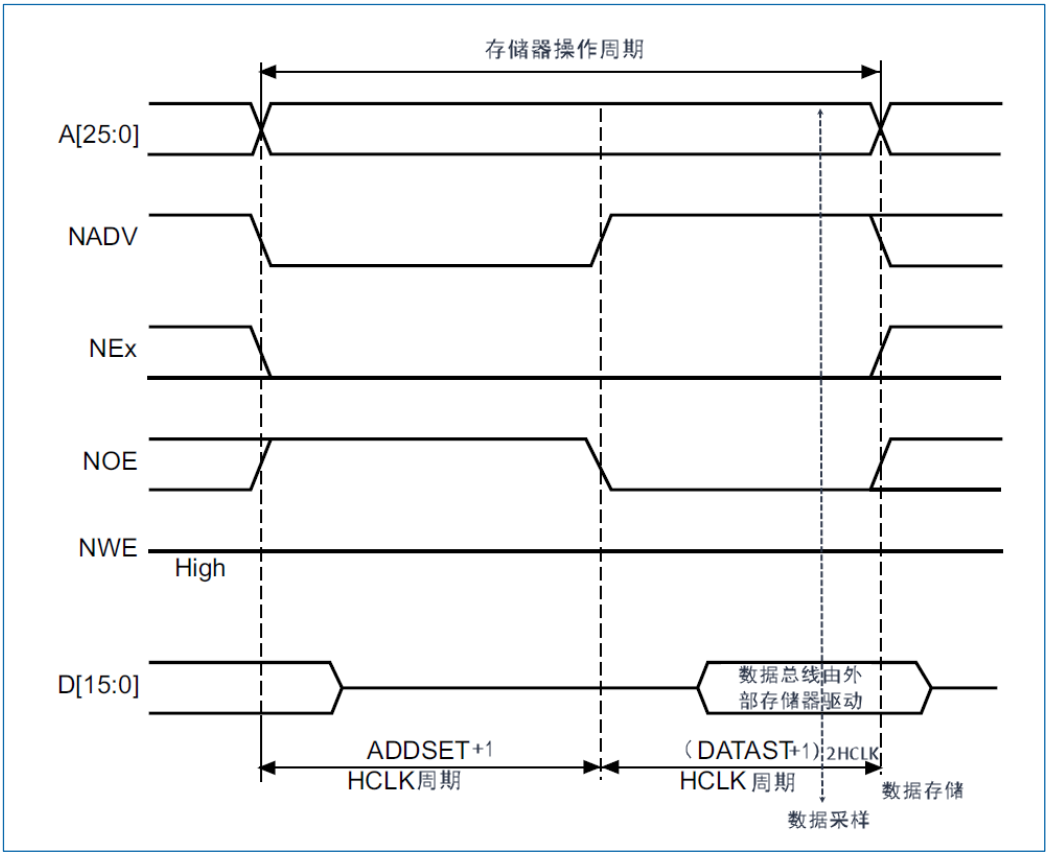


图 15-10 模式 C 读操作

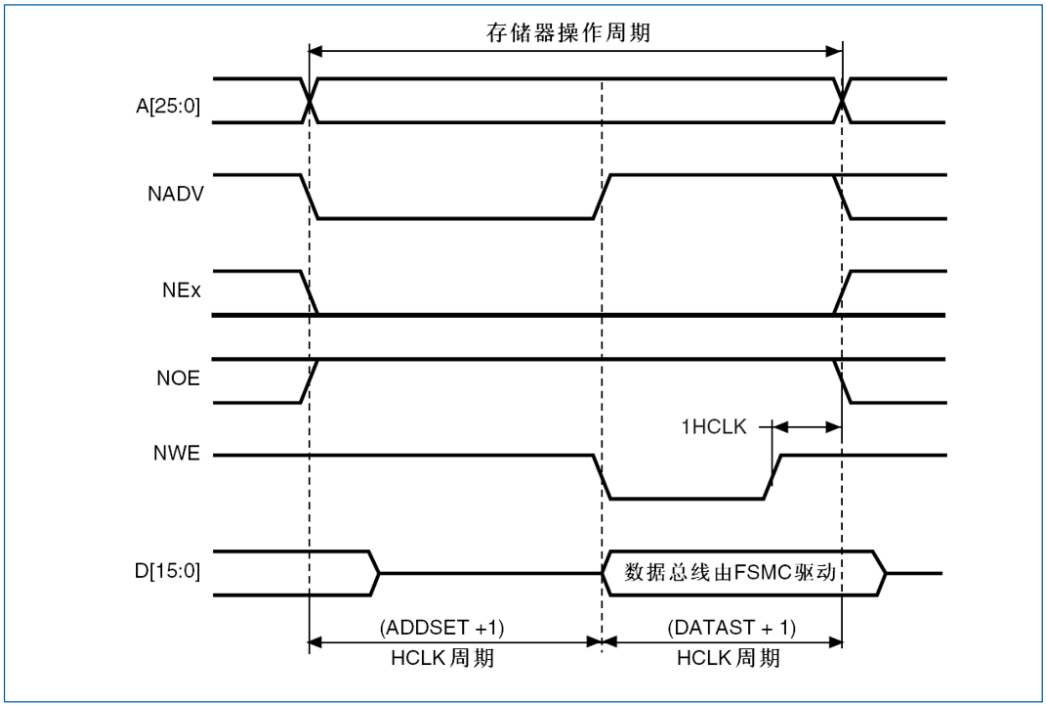


图 15-11 模式 C 写操作

模式 C 与模式 1 不同的是，NOE 和 NADV 的翻转变化，以及独立的读写时序。

表 15-18 FSMC_BCRx 位域（模式 C）

位编号	位名称	设置的数值
31:20	Res	0x000
19	CBURSTRW	0x0 （对异步模式无影响）
18:16	Res	0x0
15	ASYNCWAIT	如果存储器支持该模式则置为'1'，否则保持为'0'
14	EXTMOD	0x1
13	WAITEN	0x0 （对异步模式无影响）
12	WREN	需要时设置
11	WAITCFG	忽略
10	WRAPMOD	0x0
9	WAITPOL	仅当位 15 为'1'时有意义
8	BURSTEN	0x0
7	Res	0x1
6	FACCEN	0x1
5:4	MWID	需要时设置
3:2	MTYP	0x2（NOR Flash）
1	MUXEN	0x0
0	MBKEN	0x1

表 15-19 FSMC_BTRx 位域（模式 C）

位编号	位名称	设置的数值
31:30	Res	0x0
29:28	ACCMOD	0x2
27:24	DATLAT	0x0
23:20	CLKDIV	0x0
19:16	BUSTURN	NEx 从高到低的时间 （BUSTURN HCLK）
15:8	DATAST	读操作的第 2 个阶段的长度（DATAST+3 个 HCLK 周期）。这个域不能为 0（至少为 1）
7:4	ADDHLD	忽略
3:0	ADDSET	读操作的第 1 个阶段的长度（ADDSET+1 个 HCLK 周期）

表 15-20 FSMC_BWTRx 位域（模式 C）

位编号	位名称	设置的数值
31:30	Res	0x0
29:28	ACCMOD	0x2
27:20	Res	0xFF
19:16	BUSTURN	NEx 从高到低的时间 （BUSTURN HCLK）
15:8	DATAST	写操作的第 2 个阶段的长度（DATAST+1 个 HCLK 周期）。这个域不能为 0（至少为 1）
7:4	ADDHLD	忽略
3:0	ADDSET	写操作的第 1 个阶段的长度（ADDSET+1 个 HCLK 周期）

15.5.4.5 模式 D-带地址扩展的异步操作

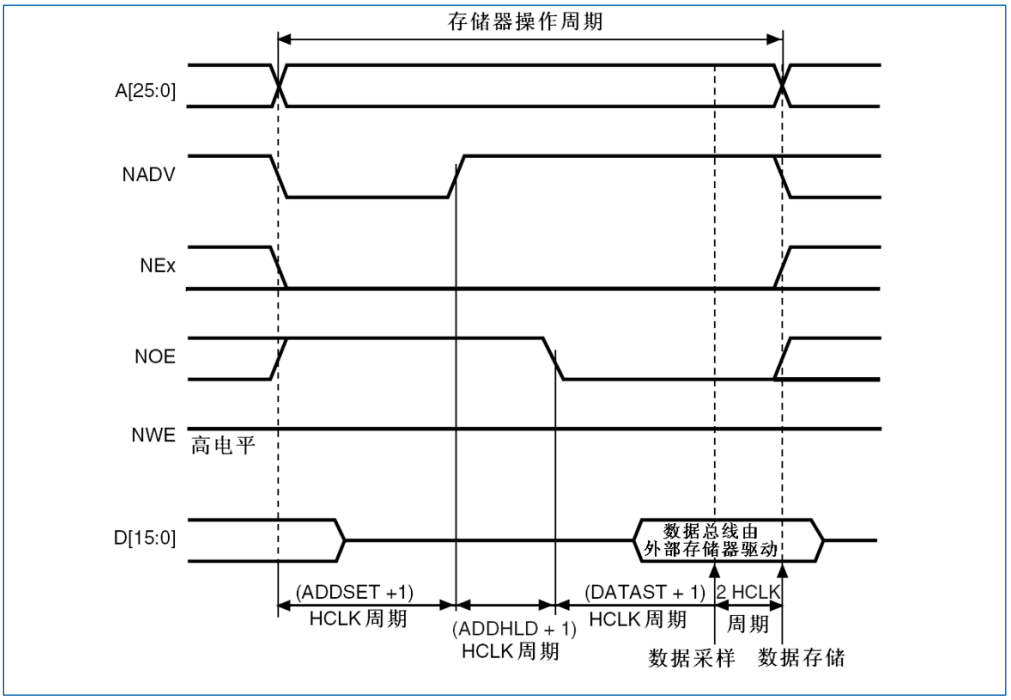


图 15-12 模式 D 读操作

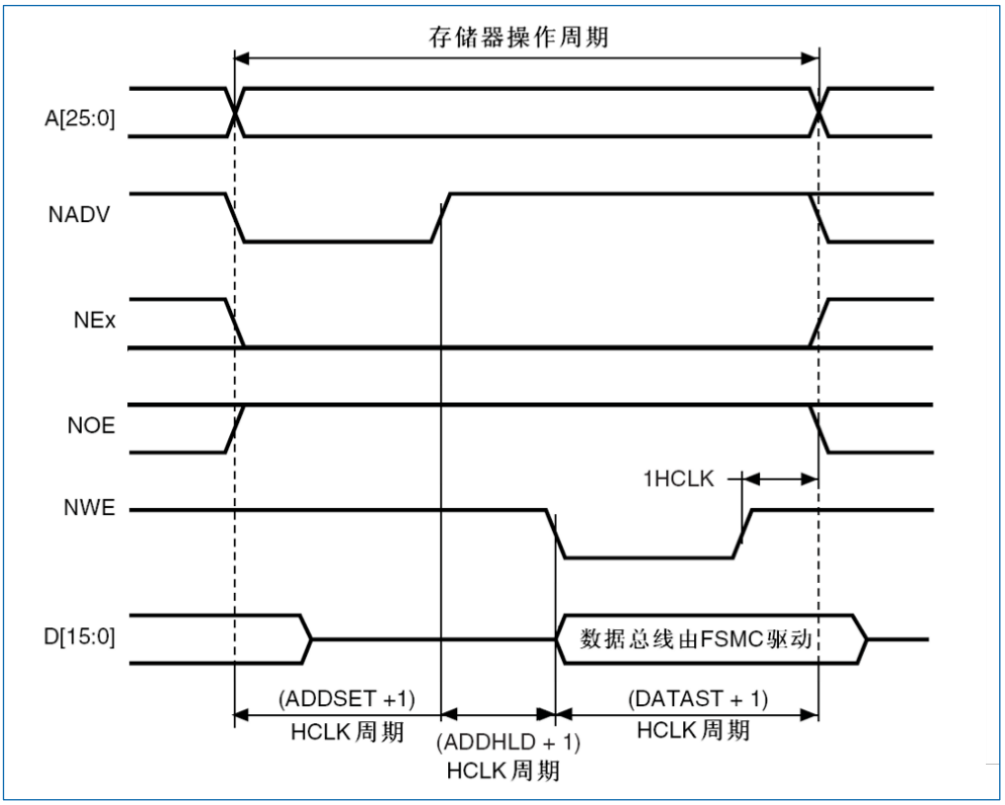


图 15-13 模式 D 写操作

模式 D 与模式 1 不同的是 NADV 的翻转变化，NOE 的翻转出现在 NADV 翻转之后，并且具有独立的读写时序。

表 15-21 FSMC_BCRx 位域（模式 D）

位编号	位名称	设置的数值
31:20	Res	0x000
19	CBURSTRW	0x0 （对异步模式无影响）
18:16	Res	0x0
15	ASYNCWAIT	如果存储器支持该模式则置为’1’，否则保持为’0’
14	EXTMOD	0x1
13	WAITEN	0x0 （对异步模式无影响）
12	WREN	需要时设置
11	WAITCFG	忽略
10	WRAPMOD	0x0
9	WAITPOL	仅当位 15 为’1’时有意义
8	BURSTEN	0x0
7	Res	0x1
6	FACCEN	根据存储器设置
5:4	MWID	需要时设置

位编号	位名称	设置的数值
3:2	MTYP	需要时设置
1	MUXEN	0x0
0	MBKEN	0x1

表 15-22 FSMC_BTRx 位域（模式 D）

位编号	位名称	设置的数值
31:30	Res	0x0
29:28	ACCMOD	0x3
27:24	DATLAT	忽略
23:20	CLKDIV	忽略
19:16	BUSTURN	NEx 从高到低的时间 （BUSTURN HCLK）
15:8	DATAST	读操作的第 2 个阶段的长度（DATAST+3 个 HCLK 周期）。这个域不能为 0（至少为 1）
7:4	ADDHLD	读操作的中间阶段的长度（ADDHLD+1 个 HCLK 周期）
3:0	ADDSET	读操作的第 1 个阶段的长度（ADDSET+1 个 HCLK 周期）

表 15-23 FSMC_BWTRx 位域（模式 D）

位编号	位名称	设置的数值
31:30	Res	0x0
29:28	ACCMOD	0x3
27:20	Res	0xFF
19:16	BUSTURN	NEx 从高到低的时间 （BUSTURN HCLK）
15:8	DATAST	写操作的第 2 个阶段的长度（DATAST+1 个 HCLK 周期）。这个域不能为 0（至少为 1）
7:4	ADDHLD	写操作的中间阶段的长度（ADDHLD+1 个 HCLK 周期）
3:0	ADDSET	写操作的第 1 个阶段的长度（ADDSET+1 个 HCLK 周期）

15.5.4.6 复用模式-地址/数据复用的 NOR Flash 异步操作

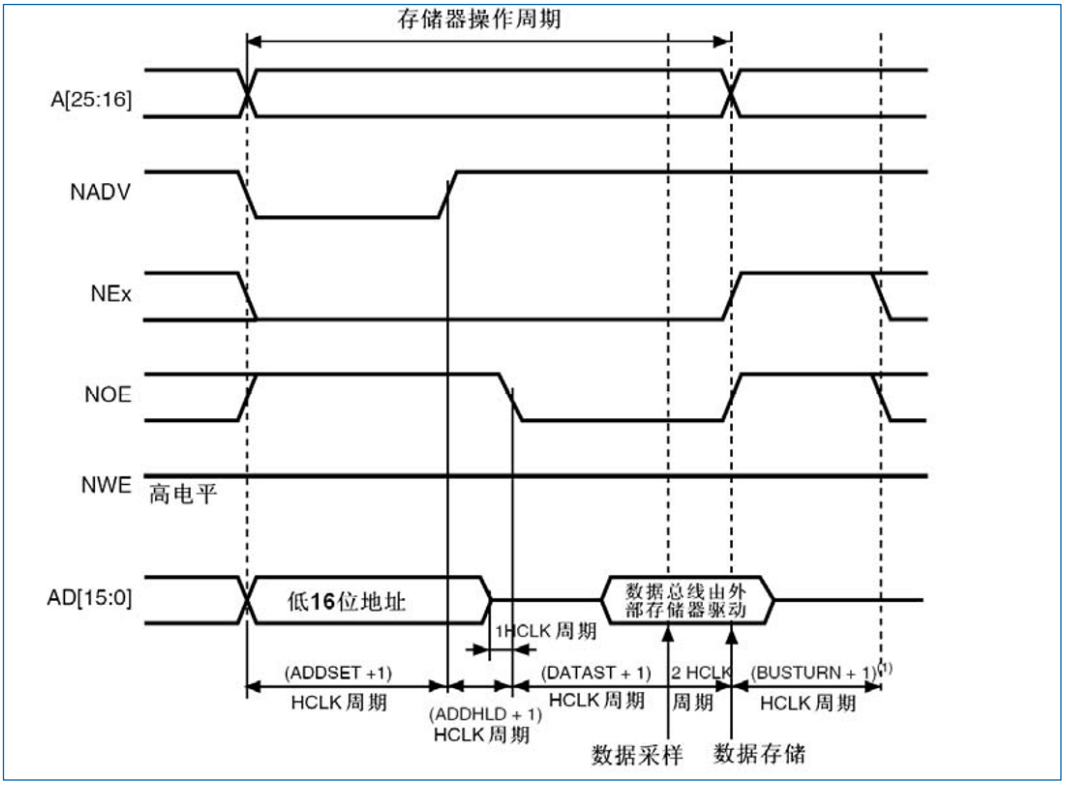


图 15-14 复用读操作

在图 15-14 中，总线恢复延迟（BUSTURN+1）与连续 2 次读操作之间在内部产生的延迟有部分重叠，因此 BUSTURN≤5 时将不影响输出时序。

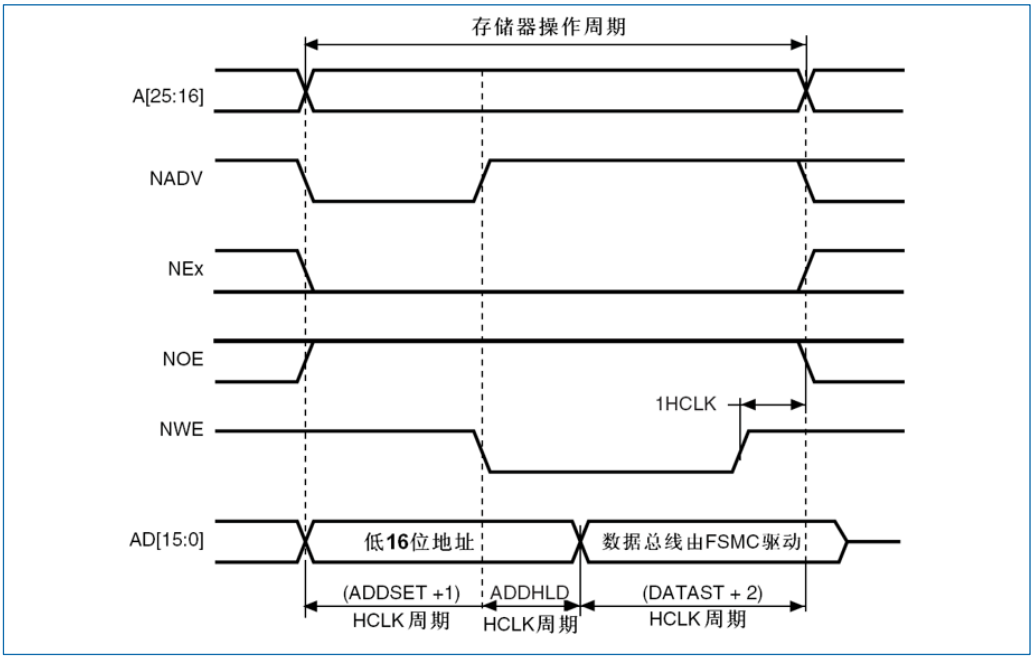


图 15-15 复用写操作

复用模式与模式 D 不同的是地址的低 16 位出现在数据总线上。

表 15-24 FSMC_BCRx 位域（复用模式）

位编号	位名称	设置的数值
31:20	Res	0x0
19	CBURSTRW	0x0 （对异步模式无影响）
18:16	Res	0x0
15	ASYNCWAIT	如果存储器支持该模式则置为’1’，否则保持为’0’
14	EXTMOD	0x0
13	WAITEN	0x0（对异步模式无影响）
12	WREN	需要时设置
11	WAITCFG	忽略
10	WRAPMOD	0x0
9	WAITPOL	仅当位 15 为’1’时有意义
8	BURSTEN	0x0
7	Res	0x1
6	FACCEN	0x1
5:4	MWID	需要时设置
3:2	MTYP	0x2（NOR Flash）
1	MUXEN	0x1
0	MBKEN	0x1

表 15-25 FSMC_BTRx 位域（复用模式）

位编号	位名称	设置的数值
31:30	Res	0x0
29:28	ACCMOD	0x0
27:20	Res	0xFF
19:16	BUSTURN	操作的最后阶段的长度（BUSTURN+1 个 HCLK 周期）
15:8	DATAST	操作的第 2 个阶段的长度，读操作为（DATAST+3 个 HCLK 周期），写操作为（DATAST+1 个 HCLK 周期）。这个域不能为 0（至少为 1）
7:4	ADDHLD	操作的中间阶段的长度（ADDHLD+1 个 HCLK 周期）。这个域不能为 0（至少为 1）
3:0	ADDSET	操作的第 1 个阶段的长度（ADDSET+1 个 HCLK 周期）

15.5.5 同步的成组读

根据参数 CLKDIV 的不同数值，存储器时钟 CLK 的周期是 HCLK 的整数倍。

NOR Flash 存储器有一个从 NADV 有效至 CLK 变高的最小时间限制，为了满足这个限制，在同步访

问（NADV 有效之前）的第一个内部时钟周期中，FSMC 不会输出时钟到存储器。这样可以保证存储器时钟的上升沿产生于 NADV 低脉冲的中间。

15.5.5.1 数据延时与 NOR Flash 的延时

数据延时是指在采样数据之前需等待的周期数目，DATLAT 数值必须与 NOR Flash 配置寄存器中定义的延时值相符合。FSMC 不把 NADV 为低时的时钟周期包含在数据延时这个参数中。

注意：

有些 NOR Flash 把 NADV 为低时的时钟周期包含在数据延时这个参数中，因此 NOR Flash 延时与 FSMC 的 DATLAT 参数的关系可以是：

NOR Flash 延时=DATLAT+2；或

NOR Flash 延时=DATLAT+3

有些新出的存储器会在数据保持阶段产生一个 NWAIT 信号，这种情况下可以设置 DATLAT 为其最小值。FSMC 会对 NWAIT 信号采样并等待足够长的时间直到数据有效，在 FSMC 检测到存储器结束了保持阶段后，读取正确的数据。

另外一些存储器不在数据保持阶段输出 NWAIT 信号，这时 FSMC 和存储器端的数据保持时间必须设置为相同的数值，否则将得不到正确的数据，或在存储器访问的初始阶段会有数据丢失。

15.5.5.2 单次突发传输

当选中的存储器块配置为同步突发模式，如果仅需要进行一次 AHB 单次突发传输，如果 AHB 需要传输 16 位数据，则 FSMC 会执行一次长度为 1 的突发传输；如果 AHB 需要传输 32 位数据，则 FSMC 会分成 2 次 16 位传输，执行一次长度为 2 的突发传输；最后一个数据传输完毕时撤消片选信号。

显然，从效率上讲这种传输方式（与异步读相比）不是最有效的；但是一次随机的异步访问需要重新配置存储器访问模式，这同样需要较长时间。

15.5.5.3 等待管理

对于同步的 NOR Flash 成组访问，在预置的保持时间（DATLAT+2 个 CLK 时钟周期）之后，需检测 NWAIT 信号。

如果检测到 NWAIT 为有效电平时（当 WAITPOL=0 时低电平有效，WAITPOL=1 时高电平有效），FSMC 将插入等待周期直到 NWAIT 变为无效电平（当 WAITPOL=0 时无效电平为高，WAITPOL=1 时无效电平为低）。

当 NWAIT 变为无效时，FSMC 认为数据已经有效（WAITCFG=1），或数据将在下一个时钟边沿有效（WAITCFG=0）。

在 NWAIT 信号控制的等待状态插入期间，控制器会连续地向存储器发送时钟脉冲、保持片选信号和输出有效信号，同时忽略无效的数据信号。

在成组传输模式下，NOR Flash 的 NWAIT 信号有 2 种时序配置：

- Flash 存储器在等待状态之前的一个数据周期插入 NWAIT 信号（复位后的默认设置）。
- Flash 存储器在等待状态期间插入 NWAIT 信号。

通过配置 FSMC_BCRx 寄存器中的 WAITCFG 位，FSMC 在每个片选上都支持这 2 种 NOR Flash 的等待状态配置。

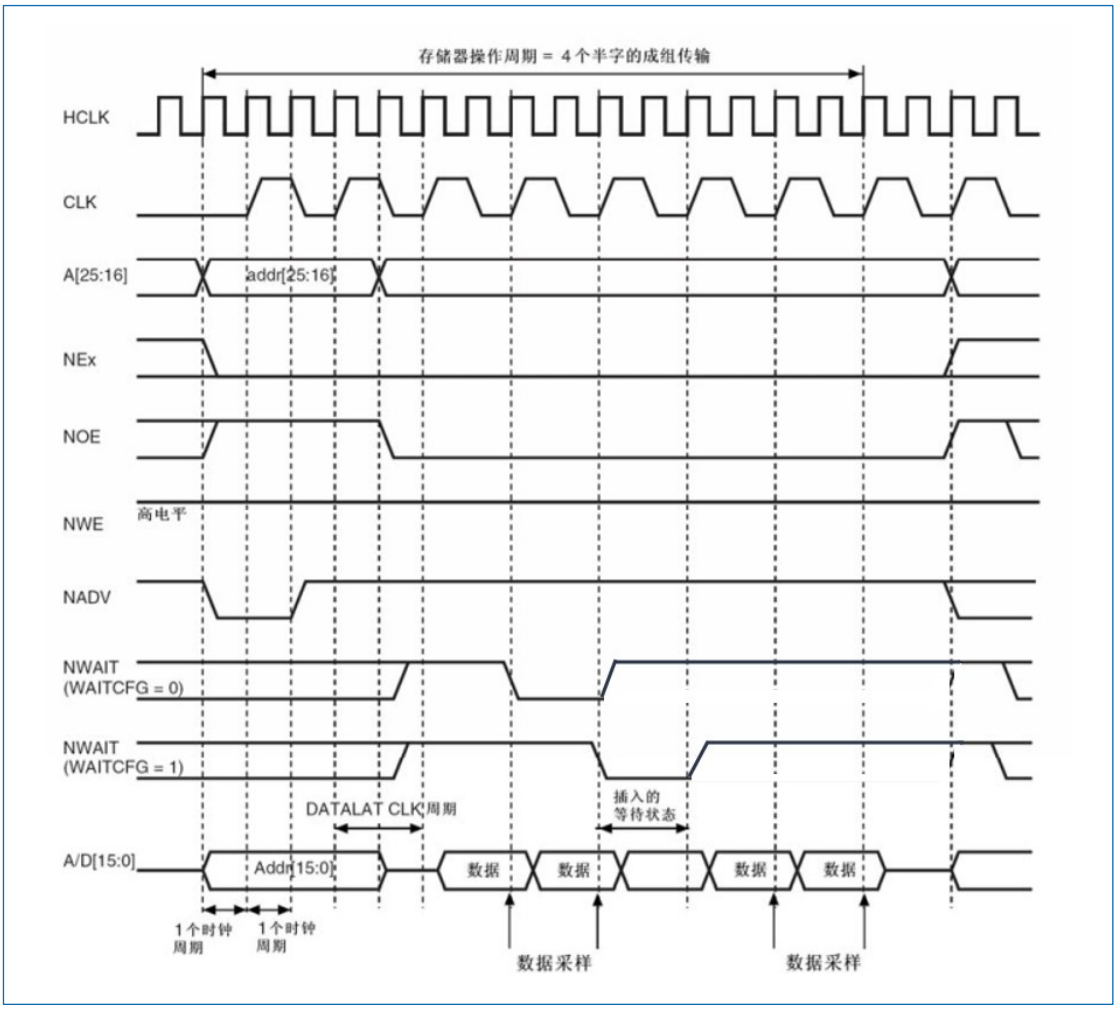


图 15-16 同步的总线复用读模式–NOR，PSRAM（CRAM）

上图说明：BL 信号没有显示在图中。对于 NOR Flash，BL 应该为高；对于 PSRAM（CRAM），BL 应该为低。

表 15-26 FSMC_BCRx 位域（同步模式）

位编号	位名称	设置的数值
31:20	Res	0x0000
19	CBURSTRW	对同步读模式不起作用
18:15	Res	0x0
14	EXTMOD	0x0
13	WAITEN	当此位为高时，不管存储器的等待数值是多少，FSMC 视数据保持阶段结束后的第一个数据有效。
12	WREN	对同步读模式不起作用
11	WAITCFG	根据存储器特性设置
10	WRAPMOD	根据存储器特性设置
9	WAITPOL	根据存储器特性设置
8	BURSTEN	0x1
7	Res	0x1

位编号	位名称	设置的数值
6	FACCEN	根据存储器设置
5:4	MWID	根据需要设置
3:2	MTYP	0x1 或 0x2
1	MUXEN	根据需要设置
0	MBKEN	0x1

表 15-27 FSMC_BTRx 位域（同步模式）

位编号	位名称	设置的数值
31:30	Res	0x0
29:28	ACCMOD	0x0
27:24	DATLAT	数据保持时间
23:20	CLKDIV	● 0x0：得到 CLK=HCLK（不支持） ● 0x1：得到 CLK=2xHCLK
19:16	BUSTURN	不起作用
15:8	DATAST	不起作用
7:4	ADDHLD	不起作用
3:0	ADDSET	不起作用

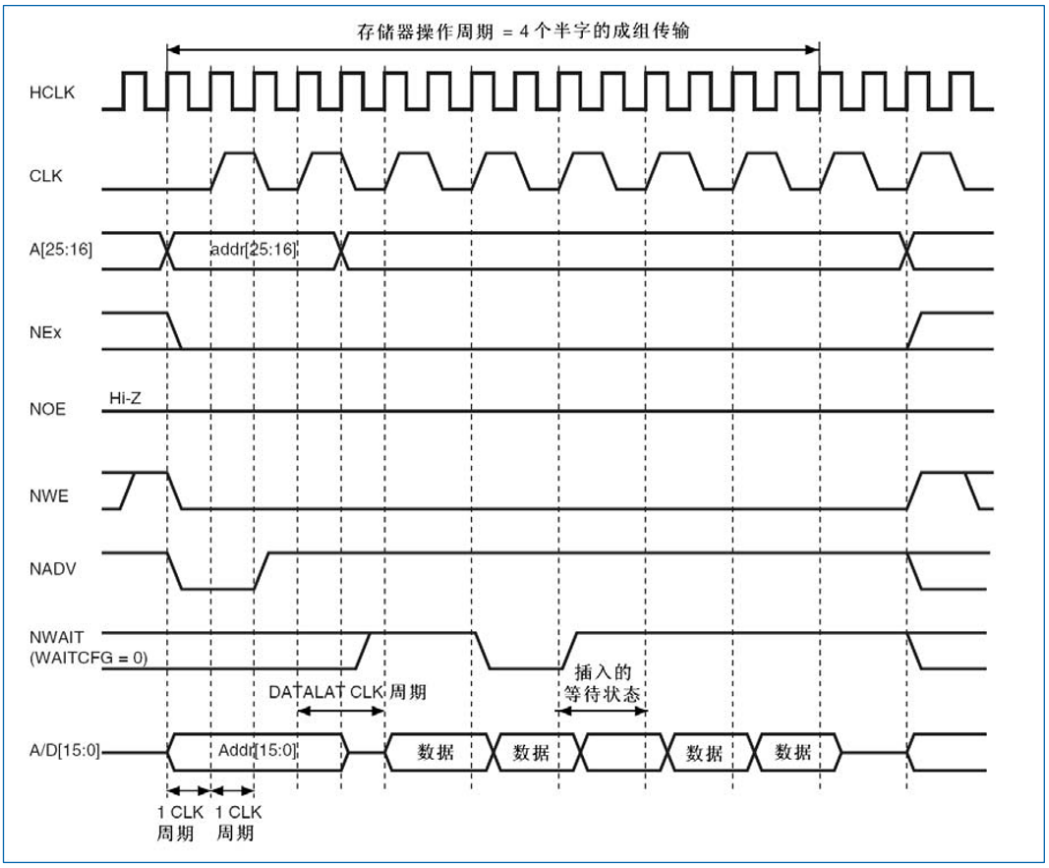


图 15-17 同步写模式–PSRAM（CRAM）

说明：

存储器必须提前一个周期产生 NWAIT 信号，同时 WAITCFG 应配置为 0。

字节选择 BL 输出没有显示在图中，当 NEx 为有效时它们为低。

表 15-28 FSMC_BCRx 位域（同步模式）

位编号	位名称	设置的数值
31:20	Res	0x0000
19	CBURSTRW	0x1
18:16	Res	0x0
15	ASCYCWAIT	0x0
14	EXTMOD	0x0
13	WAITEN	当此位为高时，不管存储器的等待数值是多少，FSMC 视数据保持阶段结束后的第一个数据有效
12	WREN	对同步读模式不起作用
11	WAITCFG	0x0
10	WRAPMOD	根据存储器特性设置
9	WAITPOL	根据存储器特性设置
8	BURSTEN	对同步写模式不起作用

位编号	位名称	设置的数值
7	FWPRLVL	设置此位防止存储器被意外写
6	FACCEN	根据存储器设置
5:4	MWID	根据需要设置
3:2	MTYP	0x1
1	MUXEN	根据需要设置
0	MBKEN	0x1

表 15-29 FSMC_BTRx 位域（同步模式）

位编号	位名称	设置的数值
31:30	Res	0x0
29:28	ACCMOD	0x0
27:24	DATLAT	数据保持时间
23:20	CLKDIV	• 0x0：得到 CLK=HCLK（不支持） • 0x1：得到 CLK=2xHCLK
19:16	BUSTURN	不起作用
15:8	DATAST	不起作用
7:4	ADDHLD	不起作用
3:0	ADDSET	不起作用

15.5.6 TFT-LCD 驱动

15.5.6.1 TFT-LCD 扫描

TFT-LCD 面板显示器以行和列结构进行管理。垂直扫描控制行数据输出，水平扫描控制列数据输出。水平扫描构建一行显示，垂直扫描构建完整框架。以连续方式对图像进行垂直和水平扫描，每秒进行多帧扫描。

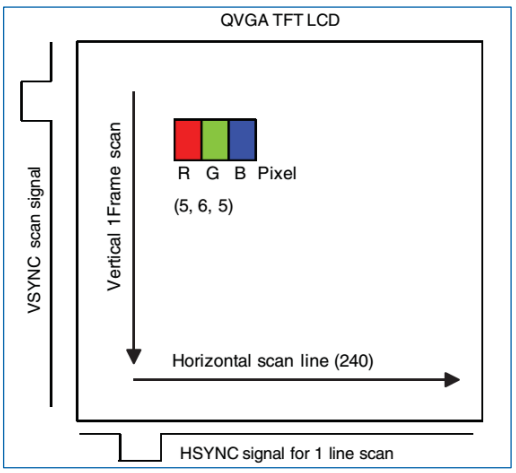


图 15-18 LCD 扫描

下图是 TFT-LCD 单帧信号显示流程。

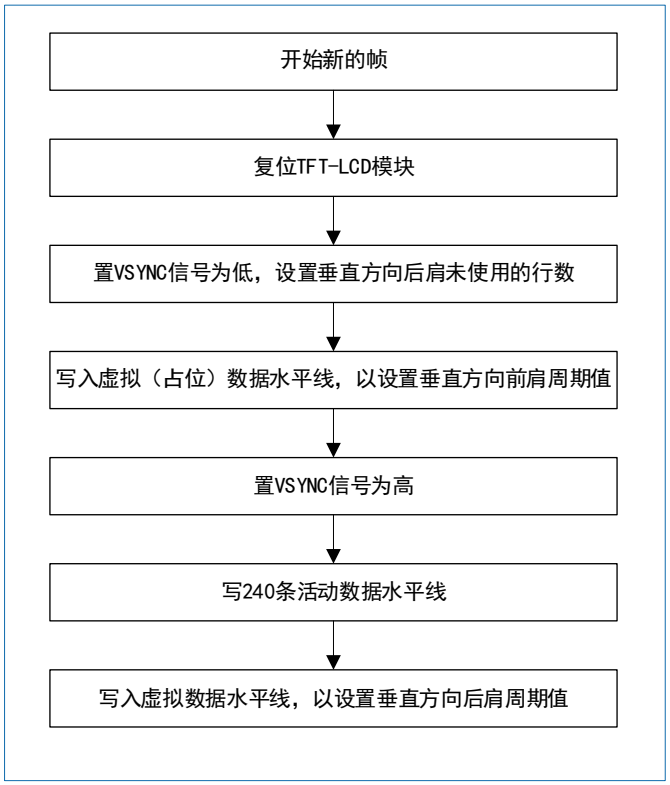


图 15-19 TFT-LCD 单帧信号显示流程

下图展示了 TFT-LCD 单水平线显示流程。

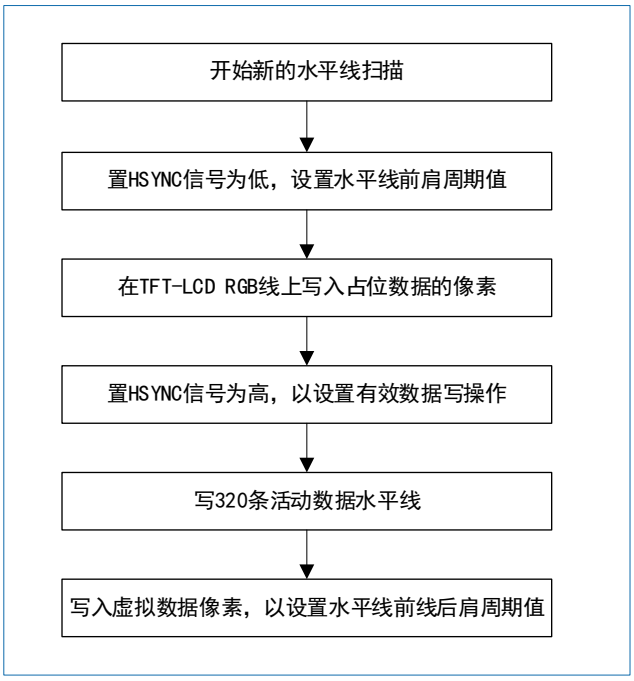


图 15-20 TFT-LCD 单水平线显示流程

15.5.6.2 TFT-LCD 信号

除了不同的数据线配置外，其他数据显示管理信号在所有 TFT-LCD 面板中都是通用的：

- 帧同步信号（VSYNC）管理垂直扫描，并充当图像（帧）更新频闪。
- 线路同步信号（HSYNC）管理水平线扫描，并充当线路显示频闪。

- 同步信号以及像素数据时钟（DCLK）执行数据输出到 TFTRGB 数据线。
- DCLK 只是充当 TFT 的数据有效信号。TFT 仅将数据视为 DCLK 边沿的输入。TFT 数据表中提到了 DCLK 有效边沿（上升/下降）。

TFT 还需要一个 TFT 支持信号，该信号仅充当芯片启用信号和 TFT 复位信号。TFT 信号必须根据显示时序约束进行同步，以确保显示具有连续的视觉效果。

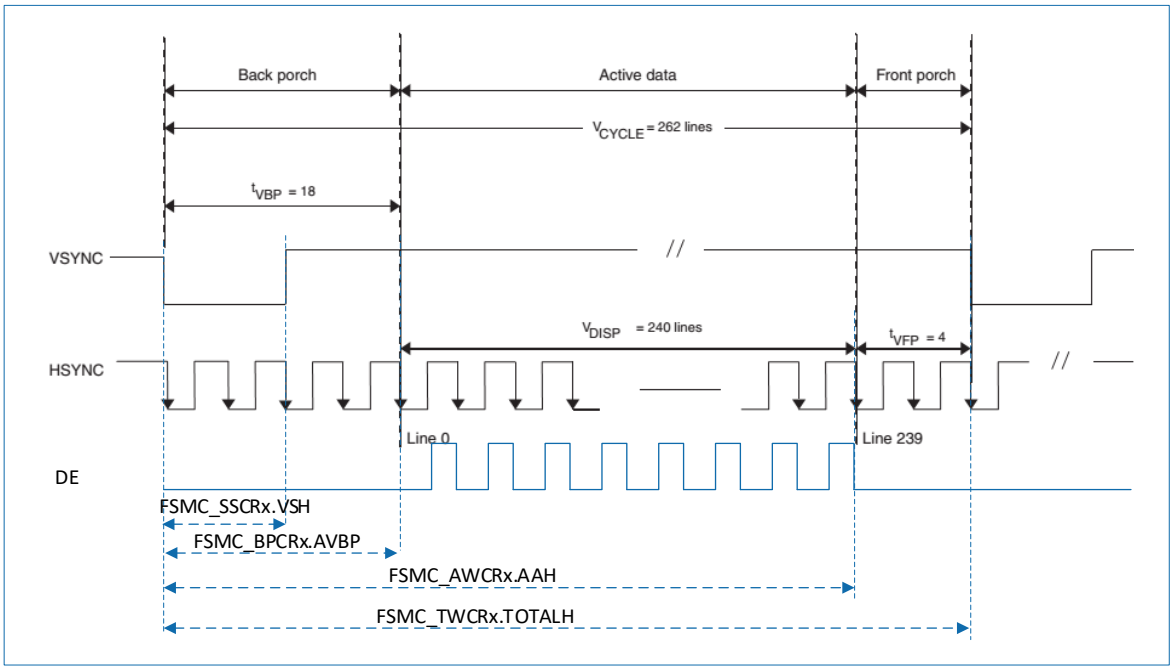


图 15-21 LCD 一帧的扫描时序

注意：后肩和前肩包含在 LINENUM 中。后肩和前肩的虚拟数据由软件或 DMA 编写，与 LCD 驱动程序的活动数据没有区别。

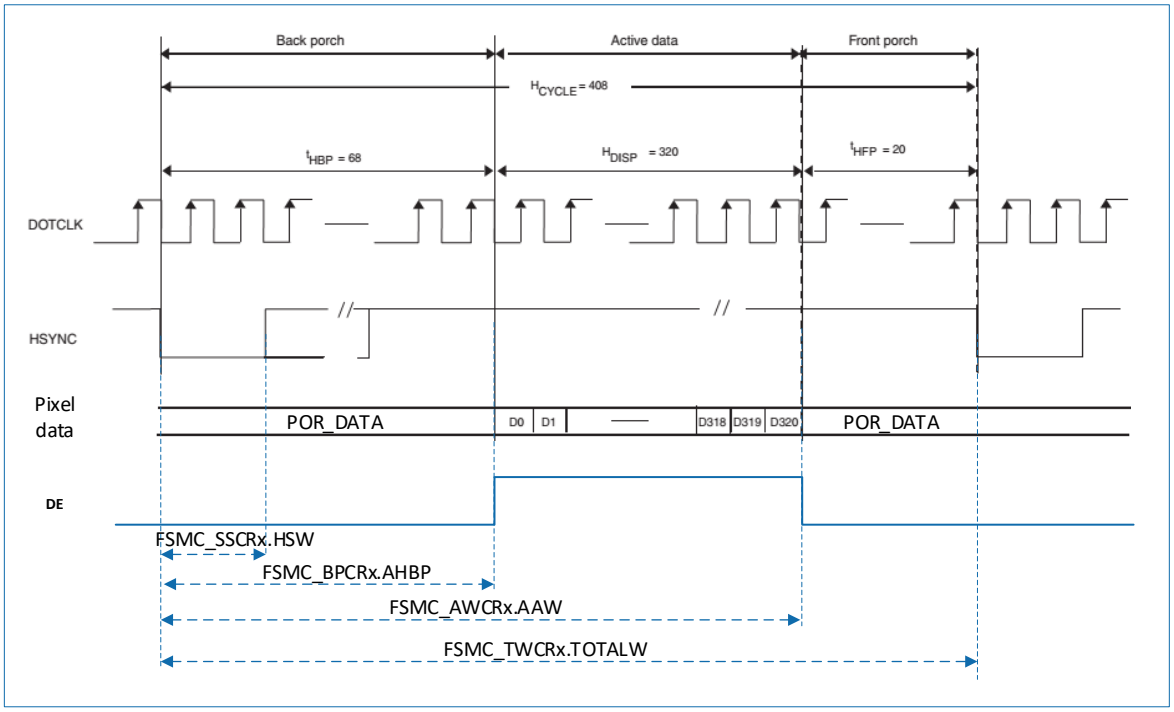


图 15-22 LCD 扫描一次的时序

注意：后肩和前肩包含在 PXLNUM 中。后肩和前肩的虚拟数据由软件或 DMA 编写，与 LCD 驱动程序的活动数据没有区别。

FSMC 总线数据宽度为 16 位。因此，如果 TFT-LCD 面板具有 24 位 RGB 线路，则 LCD RGB 数据线的 MSB 能以 565 格式交互。

15.5.6.3 TFT-LCD 驱动配置

要在 NOR Flash/PSRAM 模式下使用 LCD 驱动器，应同时设置 NOR/PSRAM 控制寄存器和 LCD 寄存器。

图 15-23 显示了应在 LCD 驱动程序中设置的像素的计时参数。

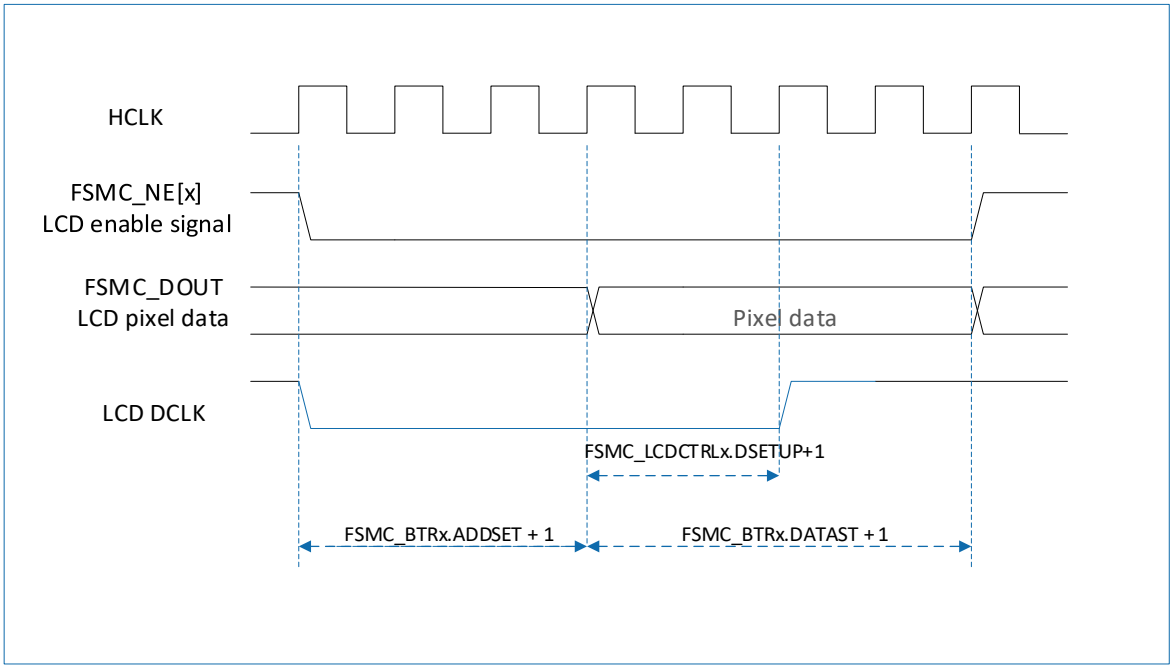


图 15-23 LCD 像素驱动时序

FSMC 可在异步模式下与 NOR Flash/SRAM 接口并行驱动 1/4 TFT-LCD 面板。每个 LCD 面板都通过 FSMC_NE[4：1]输出信号启用。

LCD 面板 x 的配置顺序如下：

1. 设置 FSMC_BCRx：WREN=1，MWID=1，MBKEN=1，所有其他位设置为 0。
2. 将 FSMC_BTRx：ADDSET 和 DATAST 设置为所需的值，所有其他位都不关心。
3. 重置 LCD 面板 x：在 LCD 通道 x 驱动之前重置 LCD 通道 x 驱动器，LCDEN=0。
4. 设置 FSMC_LCDCTRLx：LCDEN=1，DCLKPOL=0/1 根据需要，DSETUP 如图 15-23 所示，VSYNCTIM 如图 15-21 所示，HSYNCTIM 如图 15-22 所示。设置 FSMC_LCDCFGx：线数如图 15-21 所示，PXLNUM 如图 15-22 所示。

15.6 NAND Flash 和 PC 卡控制器

FSMC 可以为下列类型的设备产生合适的信号时序：

- NAND Flash
  - 8 位
  - 16 位
- 与 16 位 PC 卡兼容的设备

NAND/PC 卡控制器能够控制 3 个外部存储块。存储块 2 和存储块 3 支持 NAND Flash，存储块 4 支持 PC 卡设备。

每个存储块由专门的寄存器配置（见“15.7.1 NOR Flash 和 PSRAM 控制器寄存器”），可编程的存储

器参数包括操作时序和 ECC 配置。

表 15-30 编程 NAND/PC 卡操作参数

参数	功能	操作模式	单位	最小	最大
存储器建立时间	发出命令之前建立地址的（HCLK）时钟周期数目	读/写	AHB 时钟 周期（HCLK）	1	255
存储器等待时间	发出命令的最短持续时间（HCLK 周期数目）	读/写		1	256
存储器保持时间	在发送命令结束后保持地址的（HCLK）时钟周期数目，写操作时也是数据的保持时间	读/写		1	254
存储器数据总线高阻时间	启动写操作之后保持数据总线为高阻态的时间	写		0	255

15.6.1 外部存储器接口信号

下表列出了用于接口 NAND Flash 和 PC 卡的典型信号线。

说明：前缀N’代表对应的信号线为低电平有效。

15.6.1.1 8 位 NAND Flash

表 15-31 8 位 NAND Flash

FSMC 信号名称	输入/输出	功能
A[17]	输出	NAND Flash 地址锁存允许信号（ALE）
A[16]	输出	NAND Flash 命令锁存允许信号（CLE）
D[7:0]	输入/输出	8 位复用的、双向地址/数据总线
NCE[x]	输出	片选，x=2/3
NOE（=NRE）	输出	输出使能（存储器端信号名：读使能，NRE）
NWE	输出	写使能
NWAIT/INT[3:2]	输入	NAND Flash 就绪/繁忙，输入至 FSMC 的信号

FSMC 可以根据需要产生多个地址周期，理论上 FSMC 不限制可以访问的 NAND 容量。

15.6.1.2 16 位 NAND Flash

表 15-32 16 位 NAND Flash

FSMC 信号名称	输入/输出	功能
A[17]	输出	NAND Flash 地址锁存允许信号（ALE）
A[16]	输出	NAND Flash 命令锁存允许信号（CLE）
D[15:0]	输入/输出	16 位复用的、双向地址/数据总线
NCE[x]	输出	片选，x=2/3
NOE（=NRE）	输出	输出使能（存储器端信号名：读使能，NRE）

FSMC 信号名称	输入/输出	功能
NWE	输出	写使能
NWAIT/INT[3:2]	输入	NAND Flash 就绪/繁忙，输入至 FSMC 的信号

FSMC 可以根据需要产生多个地址周期，理论上 FSMC 不限制可以访问的 NAND 容量。

表 15-33 16 位 PC 卡

FSMC 信号名称	输入/输出	功能
A[10:0]	输出	地址总线
NIORD	输出	I/O 空间的输出使能
NIOWR	输出	I/O 空间的写使能
NREG	输出	指示操作是在通用或属性空间的寄存器信号
D[15:0]	输入/输出	双向数据总线
NCE4_1	输出	片选 1
NCE4_2	输出	片选 2（指示操作是 16 位还是 8 位）
NOE	输出	输出使能
NWE	输出	写使能
NWAIT	输入	进入 FSMC 的 PC 卡等待信号（存储器信号为 IORDY）
INTR	输入	进入 FSMC 的 PC 卡中断信号（仅适合可以产生中断的 PC 卡）
CD	输入	PC 卡存在的检测信号

15.6.2 NAND Flash/PC 卡支持的存储器及其操作

下表列出了支持的设备、操作模式和操作方式。在表格中有阴影的部分表示 NAND Flash/PC 卡控制器不支持对应的操作方式。

表 15-34 支持的存储器及其操作

存储器	模式	读/写	AHB 数据宽度	存储器数据宽度	是否支持	注释
8 位 NAND	异步	读	8	8	支持	
	异步	写	8	8	支持	
	异步	读	16	8	支持	分成 2 次 FSMC 访问
	异步	写	16	8	支持	分成 2 次 FSMC 访问
	异步	读	32	8	支持	分成 4 次 FSMC 访问
	异步	写	32	8	支持	分成 4 次 FSMC 访问
16 位 NAND	异步	读	8	16	支持	
	异步	写	8	16	不支持	

存储器	模式	读/写	AHB 数据宽度	存储器数据宽度	是否支持	注释
	异步	读	16	16	支持	
	异步	写	16	16	支持	
	异步	读	32	16	支持	分成 2 次 FSMC 访问
	异步	写	32	16	支持	分成 2 次 FSMC 访问

15.6.3 NAND Flash、ATA 和 PC 卡时序图

每个 PC 卡/CF 卡和 NAND Flash 存储器块都是通过一组寄存器管理：

- 控制寄存器：FSMC_PCRx
- 中断状态寄存器：FSMC_SRx
- ECC 寄存器：FSMC_ECCRx
- 通用存储器空间的时序寄存器：FSMC_PMEMx
- 属性存储器空间的时序寄存器：FSMC_PATTx
- I/O 空间的时序寄存器：FSMC_PIOx

每一个时序控制寄存器都包含 3 个参数，用于定义 PC 卡/CF 或 NAND Flash 操作中三个阶段的 HCLK 周期数目，还有一个定义了写操作中 FSMC 开始驱动数据总线时机的参数。下图给出了在通用存储空间中操作的时序参数定义，属性存储空间和 I/O 空间（只适用于 PC 卡）中操作与此相似。

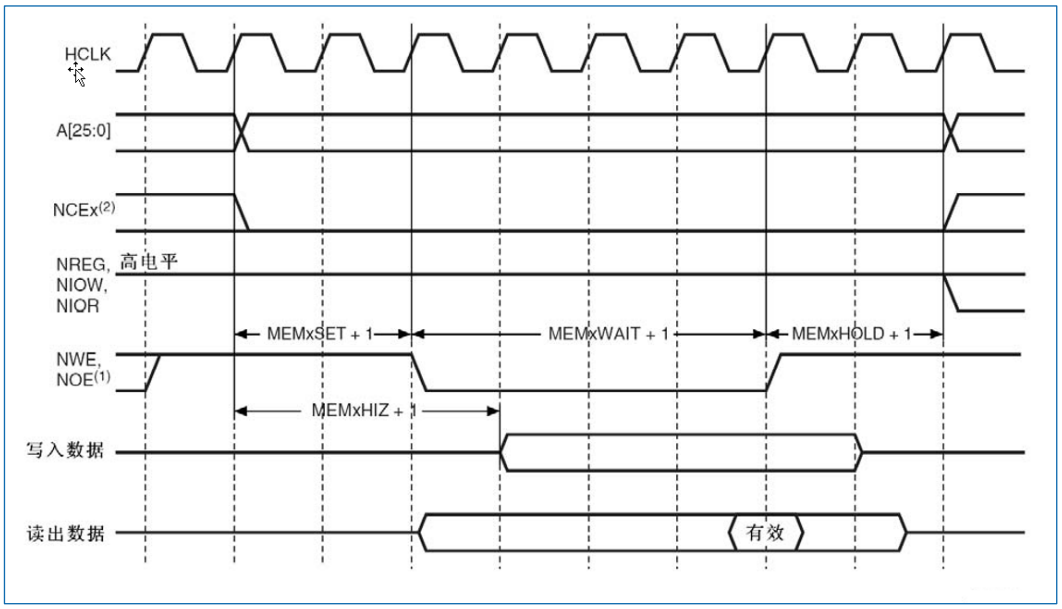


图 15-24 NAND/PC 卡控制器通用存储空间的访问时序

(1). 在写操作时 NOE 始终保持高（无效状态），在读操作时 NWE 始终保持高（无效状态）。

(2). 只要请求 NAND 访问，NCEx 信号就变低并在访问其它存储器块之前保持为低。

15.6.4 NAND Flash 操作

正如前面所述，NAND Flash 的命令锁存使能（CLE）和地址锁存使能（ALE）信号由 FSMC 的地址信号线驱动。这意味着在向 NAND Flash 发送命令或地址时，CPU 需要对存储空间中的特定地址执行写操作。

一个典型的对 NAND Flash 的读操作有如下步骤：

1. 根据 NAND Flash 的特性，通过 FSMC_PCRx 和 FSMC_PMEMx 寄存器配置和使能相应的存储器块，对于某些 NAND Flash 可能还要操作 FSMC_PATTx 寄存器（见章节：“NAND Flash 预等待功能”）。需要配置的位包括：PWID 指示 NAND Flash 的数据总线宽度，PTYP=1，PWAITEN=1，PBKEN=1，参见通用存储空间时序寄存器 x（FSMC_PMEMx）（x=2..4）。

2. CPU 在通用存储空间写入 Flash 命令字节（例如对于 Samsung 的 NAND Flash，该字节为 0x00），在写信号有效期间（NWE 的低脉冲）NAND Flash 的 CLE 输入变为有效（高电平），这时 CPU 写的字节被 NAND Flash 识别为一个命令。一旦 NAND Flash 锁存了这个命令，随后的页读操作不必再发送相同的命令。

3. CPU 在通用存储器空间或属性空间写入四个字节（较小容量的 NAND Flash 可能只需要三个字节）作为读操作的开始地址（STARTAD），以 64MB 的 NAND Flash 为例，按照 STARTAD[7:0]、STARTAD[16:9]、STARTAD[24:17]和 STARTAD[25]的顺序写入。在写信号有效期间（NWE 的低脉冲），NAND Flash 的 ALE 输入变为有效（高电平），这时 CPU 写的字节被 NAND Flash 识别为读操作的开始地址。使用属性存储空间，可以使 FSMC 产生不同的时序，实现某些 NAND Flash 所需的预等待功能（见章节 NAND Flash 预等待功能）。

4. 控制器在开始（对相同的或另一个存储器块）新的操作之前等待 NAND Flash 准备就绪（R/NB 信号变为高），在等待期间控制器保持 NCE 信号有效（低电平）。

5. CPU 可以在通用存储空间执行字节读操作，逐字节地读出 NAND Flash 的存储页（数据域和备份域）。

6. 在 CPU 不写入命令或地址的情况下，NAND Flash 的下一个页可以下述任一种方式读出：
   - 按照步骤 5 进行操作
   - 返回步骤 3 开始输入一个新的地址
   - 返回步骤 2 开始输入一个新的命令

15.6.5 NAND Flash 预等待功能

某些 NAND Flash 要求在输入最后一个地址字节后，控制器等待 R/NB 信号变低，如下图：

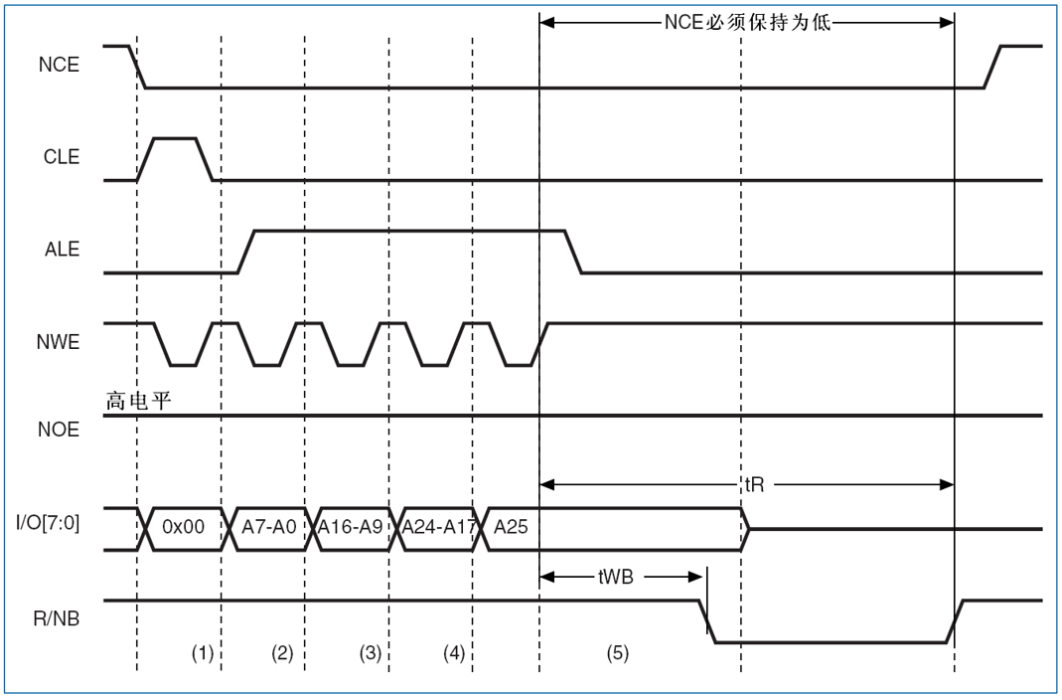


图 15-25 操作 CE 敏感型 NAND Flash

1. CPU 写字节命令 0x00 至 0x70010000

2. CPU 写 NAND 的地址 A7~A0 至 0x70020000

3. CPU 写 NAND 的地址 A16~A9 至 0x70020000

4. CPU 写 NAND 的地址 A24~A17 至 0x70020000

5. CPU 写 NAND 的地址 A25 至 0x78020000：这时 FSMC 使用 FSMC_PATT2 的时序定义执行写操作，此时 ATTHOLD≥7（这里：（7+1）xHCLK=112ns>t_{WB}的最大值）。这样可以保证 R/NB 变低再变高的过程中 NCE 保持为低，只有那些对 NCE 敏感型的 NAND Flash 有此要求。

当需要这样的功能时，可以通过配置 MEMHOLD 的数值来保证 t_{WB} 的时序，但是任何随后的 CPU 对 NAND Flash 的读或写操作中，控制器都会在 NWE 信号的上升沿至下一次操作之间插入一个保持延迟，延迟长度为（MEMHOLD+1）个 HCLK 周期。

为了克服这个时序的限制，这里使用了属性存储空间配置 ATTHOLD 的数值使之符合 t_{WB} 的时序，同时保持 MEMHOLD 为其最小值。此时，CPU 必须在所有 NAND Flash 的读写操作时使用通用存储空间，只有在写入 NAND Flash 地址的最后一个字节时，CPU 需要写入属性存储空间。

15.6.6 NAND Flash 的纠错码 ECC 计算

FSMC 的 PC 卡控制器包含了 2 个纠错码计算的硬件模块，存储块 2 和 3 各有一个。该模块可以用于减小 CPU 在处理纠错码时的软件工作量。

有两个相同的寄存器分别对应存储块 2 和存储块 3，存储块 4 没有包含硬件的 ECC 计算模块。

FSMC 中实现的纠错码（ECC）算法可以在读或写 NAND Flash 时，在每 256、512、1024、2048、4096 或 8192 个字节中，矫正 1 个比特位的错误并且检测出 2 个比特位的错误。

每当 NAND Flash 存储器卡被激活时，ECC 模块监测 NAND Flash 的数据总线和读/写信号（NCE 和 NWE）。

该功能的操作如下：

- 当在存储器块 2 或存储器块 3 访问 NAND Flash 时，出现在 D[15:0]总线上的数据被锁存并用于 ECC 计算。
- 当对 NAND Flash 的操作发生在其它地址时，ECC 电路不进行任何操作。因此输出 NAND Flash 命令和地址的写操作不会参与 ECC 计算。

当规定数目的字节已经写入 NAND Flash 或从 NAND Flash 读出，软件必须读出 FSMC_ECCR2/3 寄存器以获得计算的 ECC 数值。读出 ECC 数值后，再次计算 ECC 时需要通过先置 ECCEN 为’0’清除这个寄存器，再在 FSMC_PCR2/3 寄存器的 ECCEN 位写’1’重新使能 ECC 计算。

15.7 FSMC 寄存器

基地址：0xA000 0000

空间大小：0x400

15.7.1 NOR Flash 和 PSRAM 控制器寄存器

基地址：0xA000 0000

空间大小：0x400

NOR Flash 和 PSRAM 控制器寄存器必须以字（32 位）的方式进行访问。

15.7.1.1 SRAM/NOR Flash 片选控制寄存器 x（FSMC_BCRx）（x=1..4）

偏移地址：8*（x-1）

复位值：0x0000 30DX

这个寄存器包含了每个存储器块的控制信息，可以用于 SRAM、ROM、异步或成组传输的 NOR Flash 存储器。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Res	FACCEN	MWID[1:0]		MTYP[1:0]		MUXEN	MBKEN
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw		rw		rw	rw

位 31:20	Res：保留 必须保持复位值。
位 19	CBURSTRW：成组写使能位（Write burst enable） 对于 Cellular RAM，该位使能写操作的同步成组传输协议。 对于处于成组传输模式的 Flash 存储器，这一位允许/禁止通过 NWAIT 信号插入等待状态。读操作的同步成组传输协议使能位是 FSMC_BCRx 寄存器的 BURSTEN 位。 • 0：写操作始终处于异步模式 • 1：写操作为同步模式。
位 18:16	Res：保留 必须保持复位值。
位 15	ASYNCWAIT：异步传输中的等待信号（Wait signal during asynchronous transfers） 该位允许 FSMC 在异步协议中使用等待信号。 • 0：异步协议中不考虑 NWAIT 信号，这是复位后的默认状态。 • 1：异步协议中考虑 NWAIT 信号。
位 14	EXTMOD：扩展模式使能（Extended mode enable） 该位允许 FSMC 使用 FSMC_BWTR 寄存器，即允许读和写使用不同的时序。 • 0：不使用 FSMC_BWTR 寄存器，这是复位后的默认状态。 • 1：FSMC 使用 FSMC_BWTR 寄存器。
位 13	WAITEN：等待使能位（Wait enable bit） 当 Flash 存储器处于成组传输模式时，这一位允许/禁止通过 NWAIT 信号插入等待状态。 • 0：禁用 NWAIT 信号，在设置的 Flash 保持周期之后不会检测 NWAIT 信号插入等待状态。 • 1：使用 NWAIT 信号，在设置的 Flash 保持周期之后根据 NWAIT 信号插入等待状态；这是复位后的默认状态。
位 12	WREN：写使能位（Write enable bit） 该位指示 FSMC 是否允许/禁止对存储器的写操作。 • 0：禁止 FSMC 对存储器的写操作，否则产生一个 AHB 错误。 • 1：允许 FSMC 对存储器的写操作；这是复位后的默认状态。
位 11	WAITCFG：配置等待时序（Wait timing configuration） 当 Flash 存储器处于成组传输模式时，NWAIT 信号指示从 Flash 存储器出来的数据是否有效或是否需要插入等待周期。 该位决定存储器是在等待状态之前的一个时钟周期产生 NWAIT 信号，还是在等待状态期间产生 NWAIT 信号。 • 0：NWAIT 信号在等待状态前的一个数据周期有效；这是复位后的默认状态。 • 1：NWAIT 信号在等待状态期间有效。
位 10	WRAPMOD：支持非对齐的成组模式（Wrapped burst mode support）

	该位决定控制器是否支持把非对齐的 AHB 成组操作分割成 2 次线性操作；该位仅在存储器的成组模式下有效。 • 0：不允许直接的非对齐成组操作；这是复位后的默认状态。 • 1：允许直接的非对齐成组操作。
位 9	WAITPOL：等待信号极性（Wait signal polarity bit） 设置存储器产生的等待信号的极性；该位仅在存储器的成组模式下有效。 • 0：NWAIT 等待信号为低时有效；这是复位后的默认状态。 • 1：NWAIT 等待信号为高时有效。
位 8	BURSTEN：成组模式使能（Burst enable bit） 允许对 Flash 存储器进行成组模式访问；该位仅在 Flash 存储器的同步成组模式下有效。 • 0：禁用成组访问模式；这是复位后的默认状态。 • 1：使用成组访问模式。
位 7	Res：保留 必须保持复位值。
位 6	FACCEN：Flash 访问使能（Flash access enable） 允许对 NOR Flash 存储器的访问操作。 • 0：禁止对 NOR Flash 存储器的访问操作。 • 1：允许对 NOR Flash 存储器的访问操作。
位 5:4	MWID[1:0]：存储器数据总线宽度（Memory databus width） 定义外部存储器总线的宽度，适用于所有类型的存储器。 • 00：8 位 • 01：16 位（复位后的默认状态） • 10：保留，不能用 • 11：保留，不能用
位 3:2	MTYP[1:0]：存储器类型（Memory type） 定义外部存储器的类型。 • 00：SRAM、ROM（存储器块 2~4 在复位后的默认值） • 01：PSRAM（Cellular RAM：CRAM） • 10：NOR Flash（存储器块 1 在复位后的默认值） • 11：保留
位 1	MUXEN：地址/数据复用使能位（Address/data multiplexing enable bit） 当设置了该位后，地址的低 16 位和数据将共用数据总线，该位仅对 NOR 和 PSRAM 存储器有效。 • 0：地址/数据不复用 • 1：地址/数据复用数据总线；这是复位后的默认状态。
位 0	MBKEN：存储器块使能位（Memory bank enable bit） 开启对应的存储器块。复位后存储器块 1 是开启的，其它所有存储器块为禁用。访问一个禁用的存储器块将在 AHB 总线上产生一个错误。 • 0：禁用对应的存储器块 • 1：启用对应的存储器块

15.7.1.2 SRAM/NOR Flash 片选时序寄存器 x（FSMC_BTRx）（x=1..4）

偏移地址：0x04+8*（x-1）

复位值：0x0FFFFFFF

这个寄存器包含了每个存储器块的控制信息，可以用于 SRAM、ROM 和 NOR Flash 存储器。如果 FSMC_BCRx 寄存器中设置了 EXTMOD 位，则有两个时序寄存器分别对应读（本寄存器）和写操作（FSMC_BWTRx 寄存器）。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res		ACCMOD		DATLAT				CLKDIV				BUSTURN			
		rw		rw				rw				rw			

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

位 31:30	Res：保留 必须保持复位值。
位 29:28	ACCMOD：访问模式（Access mode） 定义异步访问模式。这 2 位只在 FSMC_BCRx 寄存器的 EXTMOD 位为 1 时起作用。 • 00：访问模式 A • 01：访问模式 B • 10：访问模式 C • 11：访问模式 D
位 27:24	DATLAT：同步成组式 NOR Flash 的数据保持时间（Data latency for synchronous NOR Flash memory） 处于同步成组模式的 NOR Flash，需要定义在读取第一个数据之前等待的存储器周期数目。这个时间参数不是以 HCLK 表示，而是以 Flash 时钟（CLK）表示。在访问异步 NOR Flash、SRAM 或 ROM 时，这个参数不起作用。操作 CRAM 时，这个参数必须为 0。 • 0000：第一个数据的保持时间为 2 个 CLK 时钟周期。 • … • 1111：第一个数据的保持时间为 17 个 CLK 时钟周期（这是复位后的默认数值）。
位 23:20	CLKDIV：时钟分频比（CLK 信号）（Clock divide ratio） 定义 CLK 时钟输出信号的周期，以 HCLK 周期数表示： • 0000：保留 • 0001：1 个 CLK 周期=2 个 HCLK 周期 • 0010：1 个 CLK 周期=3 个 HCLK 周期 • … • 1111：1 个 CLK 周期=16 个 HCLK 周期（这是复位后的默认数值）。在访问异步 NOR Flash、SRAM 或 ROM 时，这个参数不起作用。
位 19:16	BUSTURN：总线恢复时间（Bus turnaround phase duration） 这些位用于定义一次读操作之后在总线上的延迟（仅适用于总线复用模式的 NOR Flash 操作），一次读操作之后控制器需要在数据总线上为下次操作送出地址，这个延迟就是为了防止总线冲突。如果扩展的存储器系统不包含总线复用模式的存储器，或最慢的存储器可以在 6 个 HCLK 时钟周期内将数据总线恢复到高阻状态，可以设置这个参数为其最小值。 • 0000：总线恢复时间=1 个 HCLK 时钟周期。 • … • 1111：总线恢复时间=16 个 HCLK 时钟周期（这是复位后的默认数值）。

位 15:8	DATAST：数据保持时间（Data-phase duration） 这些位定义数据的保持时间（见图 15-3 模式 1 读操作至图 15-15 复用写操作），适用于 SRAM、ROM 和异步总线复用模式的 NOR Flash 操作。 • 00000000：保留 • 00000001：DATAST 保持时间=2 个 HCLK 时钟周期 • 00000010：DATAST 保持时间=3 个 HCLK 时钟周期 • … • 11111111：DATAST 保持时间=256 个 HCLK 时钟周期（这是复位后的默认数值）。 对于每一种存储器类型和访问方式的数据保持时间，请参考对应的图表（图 15-3 模式 1 读操作至图 15-15 复用写操作）。 例如：模式 1、读操作、DATAST=1：数据保持时间=DATAST+3=4 个 HCLK 时钟周期。
位 7:4	ADDHLD：地址保持时间（Address-hold phase duration） 这些位定义地址的保持时间（见图 15-12 模式 D 读操作至图 15-15 复用写操作），适用于 SRAM、ROM 和异步总线复用模式的 NOR Flash 操作。 • 0000：ADDHLD 保持时间=1 个 HCLK 时钟周期 • … • 1111：ADDHLD 保持时间=16 个 HCLK 时钟周期（这是复位后的默认数值） *说明：在同步操作中，这个参数不起作用，地址保持时间始终是 1 个存储器时钟周期。*
位 3:0	ADDSET：地址建立时间（Address setup phase duration） 这些位定义地址的建立时间（见图 15-3 模式 1 读操作至图 15-15 复用写操作），适用于 SRAM、ROM 和异步总线复用模式的 NOR Flash 操作。 • 0000：ADDSET 建立时间=1 个 HCLK 时钟周期 • … • 1111：ADDSET 建立时间=16 个 HCLK 时钟周期（这是复位后的默认数值）。 对于每一种存储器类型和访问方式的地址建立时间，请参考对应的图表（图 15-3 模式 1 读操作至图 15-15 复用写操作）。 例如：模式 2、读操作、ADDSET=1：地址建立时间=ADDSET+1=2 个 HCLK 时钟周期 *说明：在同步操作中，这个参数不起作用，地址建立时间始终是 1 个存储器时钟周期。*

说明：因为内部的刷新，PSRAM（CRAM）具有可变的保持延迟，因此这样的存储器会在数据保持期间输出 NWAIT 信号以延长数据的保持时间。

使用 PSRAM（CRAM）时 DATLAT 域应置为 0，这样 FSMC 可以及时地退出自己的保持阶段并开始对存储器发出的 NWAIT 信号进行采样，然后在存储器准备好时开始读或写操作。

这个操作方式还可以用于操作最新的能够输出 NWAIT 信号的同步 Flash 存储器，详细信息请参考相应的 Flash 存储器手册。

15.7.1.3 SRAM/NOR Flash 写时序寄存器（FSMC_BWTRx）（x=1..4）

偏移地址：0x104+8*（x-1）

复位值：0x0FFFFFFF

这个寄存器包含了每个存储器块的控制信息，可以用于 SRAM、ROM 和 NOR Flash 存储器。如果 FSMC_BCRx 寄存器中设置了 EXTMOD 位，则这个寄存器对应写操作。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res		ACCMOD[1:0]		Res								BUSTURN[3:0]			
		rw										rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
rw								rw				rw			

位 31:30	Res：保留 必须保持复位值。
位 29:28	ACCMOD[1:0]：访问模式（Access mode） 定义异步访问模式。这 2 位只在 FSMC_BCRx 寄存器的 EXTMOD 位为 1 时起作用。 • 00：访问模式 A • 01：访问模式 B • 10：访问模式 C • 11：访问模式 D
位 27:20	Res：保留 必须保持复位值。
位 19:16	BUSTURN[3:0]：总线恢复时间（Bus turnaround phase duration） 这些位用于定义一次读操作之后在总线上的延迟（仅适用于总线复用模式的 NOR Flash 操作），一次读操作之后控制器需要在数据总线上为下次操作送出地址，这个延迟就是为了防止总线冲突。如果扩展的存储器系统不包含总线复用模式的存储器，或最慢的存储器可以在 6 个 HCLK 时钟周期内将数据总线恢复到高阻状态，可以设置这个参数为其最小值。 • 0000：总线恢复时间=1 个 HCLK 时钟周期 • … • 1111：总线恢复时间=16 个 HCLK 时钟周期（这是复位后的默认数值）。
位 15:8	DATAST[7:0]：数据保持时间（Data-phase duration） 这些位定义数据的保持时间（见图 15-3 模式 1 读操作至图 15-15 复用写操作），适用于 SRAM、ROM 和异步总线复用模式的 NOR Flash 操作。 • 00000000：保留 • 00000001：DATAST 保持时间=2 个 HCLK 时钟周期 • 00000010：DATAST 保持时间=3 个 HCLK 时钟周期 • … • 11111111：DATAST 保持时间=256 个 HCLK 时钟周期（这是复位后的默认数值）。
位 7:4	ADDHLD[3:0]：地址保持时间（Address-hold phase duration） 这些位定义地址的保持时间（见图 15-12 模式 D 读操作至图 15-15 复用写操作），适用于 SRAM、ROM 和异步总线复用模式的 NOR Flash 操作。 • 0000：保留 • 0001：ADDHLD 保持时间=2 个 HCLK 时钟周期 • 0010：ADDHLD 保持时间=3 个 HCLK 时钟周期 • … • 1111：ADDHLD 保持时间=16 个 HCLK 时钟周期（这是复位后的默认数值）。 *说明：在同步 NOR Flash 操作中，这个参数不起作用，地址保持时间始终是 1 个 Flash 时钟周期。*
位 3:0	ADDSET[3:0]：地址建立时间（Address setup phase duration） 这些位以 HCLK 周期数定义地址的建立时间（见图 15-3 模式 1 读操作至图 15-15 复用写操作），适用于 SRAM、ROM 和异步总线复用模式的 NOR Flash 操作。 • 0000：ADDSET 建立时间=1 个 HCLK 时钟周期

	• … • 1111：ADDSET 建立时间=16 个 HCLK 时钟周期（这是复位后的默认数值）。 *说明：在同步 NOR Flash 操作中，这个参数不起作用，地址建立时间始终是 1 个 Flash 时钟周期。*

15.7.1.4 SRAM/NOR NADV 极性（FSMC_NADV）

偏移地址：0x01F0

复位值：0x00000000

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

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

位 30:2	Res：保留 必须保持复位值。
位 1	ADV_POL：NADV 极性（NADV polarity） • 1：水平同步极性为活动高 • 0：水平同步极性为活动低（默认）
位 0	Res：保留 必须保持复位值。

15.7.1.5 ENCRY 设置寄存器（FSMC_ENCRY）

偏移地址：0x01F4

复位值：0x0000 0000

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

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

位 30:1	Res：保留 必须保持复位值。
位 0	ENCRYEN：将输出数据加密到存储器（The output data to the memory encryption） 写 0x1357ECA8 至该寄存器以使能此功能，任何其他数据都将复位此功能。 • 0：禁用输出数据加密（默认） • 1：启用输出数据加密

15.7.1.6 DECRY 设置寄存器（FSMC_DECRY）

偏移地址：0x01F8

复位值：0x0000 0000

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

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

位 31:1	Res：保留 必须保持复位值。
位 0	DECRYEN：解密来自存储器的输入数据（The output data from the memory decryption） 写 0x2468DB97 入该寄存器以使能此功能，任何其他数据都将复位此功能。 • 0：输入数据解密被禁用（默认） • 1：启用输入数据解密

15.7.1.7 LCD 控制寄存器 x（FSMC_LCDCTRLx）（x=1..4）

偏移地址：0x200+4*（x-1）

复位值：0x00010000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HSPOL	VSPOL	DEPOL	DCLKPOL	DSETUP[7:0]								Res			
rw	rw	rw	rw	rw											

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

位 31	HSPOL：水平同步极性（Horizontal synchronous polarity） 这个位的设置和清除由软件完成。 • 0：水平不同极性激活低 • 1：水平不同极性激活高
位 30	VSPOL：垂直同步极性（Vertical synchronous polarity） 这个位的设置和清除由软件完成。 • 0：垂直不同极性激活低 • 1：垂直不同极性激活高
位 29	DEPOL：数据启用极性（保留，无功能）（Data enable polarity） 此位由软件设置和清除。 • 0：数据启用极性处于活动状态低 • 1：数据启用极性处于活动状态高
位 28	DCLKPOL：DCLK 主动边沿选择（DCLK edge selection） • 0：DCLK 为负边沿活动 • 1：DCLK 是主动的
位 27:20	DSETUP[7:0]：设置有关 DCLK 的像素数据设置时间（以 SoC HCLK 周期为单位）（DCLK pixel data setting time） 1~256 HCLK 循环

位 19:1	Res：保留 必须保持复位值。
位 0	LCDEN：LCD 面板启用信号（LCD panel enable）

15.7.1.8 LCD 同步大小配置寄存器 x（FSMC_LCDSSCRx）（x=1..4）

偏移地址：0x210+4*（x-1）

复位值：0x0000 0000

此寄存器定义水平同步像素数减去 1 和垂直同步线数减去 1。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res				HSW[11:0]											
				rw											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res					VSH[10:0]										
					rw										

位 31:28	Res：保留 必须保持复位值。
位 27:16	HSW[11：0]：水平同步宽度（以像素时钟周期为单位）（Horizontal synchronization width） 该位域定义水平同步像素数减去 1。
位 15:11	Res：保留 必须保持复位值。
位 10:0	VSH[10:0]：垂直同步高度（以水平扫描线为单位）（Vertical synchronization width） 该位域定义垂直同步高度减去 1。它表示水平同步线的数量。

15.7.1.9 LCD 后肩配置寄存器 x（FSMC_LCDBPCRx）（x=1..4）

偏移地址：0x220+4*（x-1）

复位值：0x0000 0000

此寄存器定义水平同步和后肩像素的累积数减去 1（HSYNC 宽度+HBP-1）和垂直同步和后肩线减去 1 的累积数（VSYNC 高度=VBP-1）。参见“15.4 外部设备地址映射”中有关配置示例的 LCD 可编程参数。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res				AHBP[11:0]											
				rw											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res					AVBP[10:0]										
					rw										

位 31:28	Res：保留 必须保持复位值。
位 27:16	AHBP[11:0]：累积水平后肩（以像素时钟周期为单位）（Accumulated Horizonal back porch） 该位域定义累积水平后肩宽度，其中包括水平同步和水平后肩像素减去 1。水平后肩是水平同步变为非活动与下一个扫描线的活动显示部分的开始之间的时间段。
位 15:11	Res：保留

	必须保持复位值。
位 10:0	AVBP[10:0]：累积的垂直后门廊（以水平扫描线的单位）（Accumulated Vertical back porch） 该位域定义累积的垂直后肩宽度，其中包括垂直同步和垂直后肩线减去 1。垂直后肩是帧开头到下一帧的第一个活动扫描线开头的水平扫描线数。

15.7.1.10 LCD 激活宽度配置寄存器 x（FSMC_LCDAWCRx）（x=1..4）

偏移地址：0x230+4*（x-1）

复位值：0x0000 0000

此寄存器定义水平同步、后肩。此寄存器定义了以水平扫描行为单位的累加有效高度 AAH（AAH=AHBP+水面扫描有效高度-1），和以像素时钟为单位的累加有效宽度 AAW（AAW=AHBP+时钟有效宽度-1）。参见“15.4 外部设备地址映射”中有关配置示例的 LCD 可编程参数。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res				AAW[11:0]											
				rw											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res					AAH[10:0]										
					rw										

位 31:28	Res：保留 必须保持复位值。
位 27:16	AAW[11:0]：累积有效宽度（以像素时钟周期为单位）（Accumulated Active width） 该位域定义累积活动宽度，包括水平同步、水平后门廊和有源像素减去 1。面板扫描线的主动显示区域中的像素。支持的最大活动宽度为 0x400。
位 15:11	Res：保留 必须保持复位值。
位 10:0	AAH[10:0]：累积有源高度（以水平扫描线为单位）（Accumulated Active Height） 该位域定义累积高度，包括垂直同步、垂直后门廊和主动高度线减去 1。活动高度是面板中活动行的数量。支持的最大活动高度为 0x300。

15.7.1.11 LCD 总的宽度配置寄存器 x（FSMC_LCDTWCRx）（x=1..4）

偏移地址：0x240+4*（x-1）

复位值：0x0000 0000

此寄存器定义水平同步、后肩、主动和前肩像素减去 1（HSYNC 宽度+HBP+主动宽度+HFP-1）的累积数量以及垂直同步、后肩线、主动线和前线累积数量减去 1（VSYNC 高度+BVBP=有源高度+VFP-1）。参见“15.4 外部设备地址映射”中有关配置示例的 LCD 可编程参数。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res				TOTALW[11:0]											
				rw											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res					TOTALH[10:0]										
					rw										

位 31:28	Res：保留 必须保持复位值。

位 27:16	TOTALW[11:0]：总宽度（以像素时钟周期为单位）（Total width） 该位域定义累积的总宽度，包括水平同步、水平后肩、主动宽度和水平前肩像素减去 1。
位 15:11	Res：保留 必须保持复位值。
位 10:0	TOTALH[10:0]：总高度（以水平扫描线为单位）（Total height） 该位域定义累积高度，包括垂直同步、垂直后肩、主动高度和垂直前肩高度线减去 1。

15.7.1.12 LCDLTDC 电流位置状态寄存器（FSMC_LCDCPSR）

偏移地址：0x250

复位值：0x0000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res				CXPOS[11:0]											
				rw											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res					CYPOS[10:0]										
					rw										

位 31:28	Res：保留 必须保持复位值。
位 27:16	CXPOS[11:0]：当前 X 定位（Current X location） 该位域返回帧中的当前 X 位置。
位 15:11	Res：保留 必须保持复位值。
位 10:0	CYPOS[10:0]：当前 Y 位置（Current Y location） 该位域返回帧中的当前 Y 位置。

15.7.1.13 LCD 肩配置寄存器（FSMC_LCDCFG）

偏移地址：0x254

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POR_DATA[14:0]															
rw															

位 31	AUTOPOR：自动写入门廊功能使能位（Auto writing into the porch enable） 启用此功能时，使用 HREADY_0 保留对 LCD 门廊范围的 AHB 写入访问，直到当前位置为活动像素） • 0：自动写入门廊功能已禁用 • 1：启用自动写入门廊功能
位 30:16	Res：保留

	必须保持复位值。
位 15:0	POR_DATA[14:0]：数据自动写入 LCD 门廊范围（Auto-writing data into the range of the porch ）

15.7.2 NAND Flash 和 PC 卡控制器寄存器

基地址：0xA0000000

空间大小：0x1000

NAND Flash/PC 卡控制寄存器必须以字（32 位）的方式进行访问。

15.7.2.1 PC 卡/NAND Flash 控制寄存器 x（FSMC_PCRx）（x=2..4）

偏移地址：0x40+0x20*（x-1）

复位值：0x0000 0018

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res												ECCPS[2:0]			TAR[3]
												rw			rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAR[2:0]			TCLR[3:0]				TSEL	Res	ECCEN	PWID[1:0]		PTYP	PBKEN	PWAITEN	Res
rw			rw				rw		rw	rw		rw	rw	rw	

位 31:20	Res：保留 必须保持复位值。
位 19:17	ECCPS[2:0]：ECC 页面大小（ECC page size） 该位定义扩展的 ECC 页面大小。 • 000：256 字节 • 001：512 字节 • 010：1024 字节 • 011：2048 字节 • 100：4096 字节 • 101：8192 字节
位 16:13	TAR[3:0]：ALE 至 RE 的延迟（The latency from ALE to RE） 以 AHB 时钟周期（HCLK）为单位设置从 ALE 变低至 RE 变低的时间。 时间计算：t_ar=（TAR+SET+4）x T_{HCLK}，这里 T_{HCLK} 表示 HCLK 周期长度。 • 0000：1 个 HCLK 周期（默认值） • … • 1111：16 个 HCLK 周期 *说明：根据不同的地址空间，SET 是 MEMSET 或是 ATTSET。*
位 12:9	TCLR[3:0]：CLE 至 RE 的延迟（The latency from CLE to RE） 以 AHB 时钟周期（HCLK）为单位设置从 CLE 变低至 RE 变低的时间。 时间计算：t_clr=（TCLR+SET+4）x T_{HCLK}，这里 T_{HCLK} 表示 HCLK 周期长度。 • 0000：1 个 HCLK 周期（默认值） • … • 1111：16 个 HCLK 周期 *说明：根据不同的地址空间，SET 是 MEMSET 或是 ATTSET。*

位 8	TSEL：TAR 和 TCLR 功能选择（TAR and TCLR function selection） • 0：对于所有数据部分访问，TAR 和 TCLR 激活的延迟更大（默认）。 • 1：如果最后一次访问位于 Address 部分，则 TAR 激活用于数据段访问；如果最后一次访问位于命令区，则 TCLR 激活为数据段访问。
位 7	Res：保留 必须保持复位值。
位 6	ECCEN：ECC 计算电路使能位（ECC circuit enable bit） • 0：关闭并复位 ECC 电路（复位后的默认值） • 1：使能 ECC 电路
位 5:4	PWID[1:0]：数据总线宽度（Data bus width） 定义外部 NAND Flash 数据总线的宽度。 • 00：8 位 • 01：16 位（PC 卡必须使用此设置，复位后的默认值） • 10：保留，不使用 • 11：保留，不使用
位 3	PTYP：存储器类型（Memory type） 定义对应的存储器块上连接的存储器类型。 • 0：PC 卡、CF 卡、CF+卡或 PCMCIA • 1：NAND Flash（复位后的默认值）
位 2	PBKEN：PC 卡/NAND 存储器块使能位（PC card/NAND memory bank enable） 使能存储器块。访问一个未使能的存储器块会产生一个 AHB 总线错误。 • 0：关闭对应的存储器块（复位后的默认值） • 1：使能对应的存储器块
位 1	PWAITEN：等待功能使能位（Wait function enable） 使能 PC 卡/NAND Flash 存储器块的等待功能。 • 0：关闭（复位后的默认值） • 1：使能 *说明：对于 PC 卡，如果使能了等待功能，MEMWAITx/ATTWAITx/IOWAITx 位的值必须高于 xxWAITx≥4 + max_wait_assertion_time/HCLK* *其中 max_wait_assertion_time 是一旦 NOE 为低时 NWAIT 变低所需的最长时间。*
位 0	Res：保留 必须保持复位值。

15.7.2.2 FIFO 状态和中断寄存器 x（FSMC_SRx）（x=2..4）

偏移地址：0x44+0x20*（x-1）

复位值：0x00000040

该寄存器包含 FIFO 状态和中断的信息。FSMC 有一个 FIFO，在写存储器时用于保存从 AHB 送来的多达 16 个字的数据。

这个功能可以在操作 FSMC 时不过多地占用 AHB，在 FSMC 从 FIFO 传送数据到存储器时释放 AHB 的带宽并操作其他外设。为了计算 ECC 的需要，该寄存器有一个指示位反映了 FIFO 的状态。数据写到存储器时同时进行 ECC 计算，因此软件必须等待 FIFO 变空后才能读到正确的 ECC 数值。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res									FEMPT	IFEN	ILEN	IREN	IFS	ILS	IRS
									r	rw	rw	rw	rw	rw	rw

位 31:7	Res：保留 必须保持复位值。
位 6	FEMPT：FIFO 空标志（FIFO empty flag） 该位只支持读，指示 FIFO 状态。 • 0：FIFO 不空 • 1：FIFO 空
位 5	IFEN：中断下降沿检测使能（Interrupt falling edge detection enable） • 0：关闭中断下降沿检测请求 • 1：使能中断下降沿检测请求
位 4	ILEN：中断高电平检测使能（Interrupt detection with high level enable） • 0：关闭中断高电平检测请求 • 1：使能中断高电平检测请求
位 3	IREN：上升沿中断检测使能（Interrupt rising edge detection enable） • 0：关闭中断上升沿检测请求 • 1：使能中断上升沿检测请求
位 2	IFS：中断下降沿状态（State of a falling edge of interrupt） 该位由硬件设置，软件清除。 • 0：没有产生中断下降沿 • 1：产生了中断下降沿
位 1	ILS：中断高电平状态（State of a high level of interrupt） 该位由硬件设置，软件清除。 • 0：没有产生中断高电平 • 1：产生了中断高电平
位 0	IRS：中断上升沿状态（State of a rising edge of interrupt） 该位由硬件设置，软件清除。 • 0：没有产生中断上升沿 • 1：产生了中断上升沿

15.7.2.3 通用存储空间时序寄存器 x（FSMC_PMEMx）（x=2..4）

偏移地址： 0x48+0x20*（x-1）

复位值：0xFCFC FCFC

每个 FSMC_PMEMx 寄存器都包含操作 PC 卡或 NAND Flash 存储块 x 的时序参数，这些参数适用于在通用存储空间操作 16 位 PC 卡/CF 卡，或发送 NAND Flash 的命令、地址和进行数据的读写操作。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MEMHIZx[7:0]								MEMHOLDx[7:0]							
rw								rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MEMWAITx[7:0]								MEMSETx[7:0]							
rw								rw							

位 31:24	MEMHIZx[7:0]：在通用空间 x 数据总线的高阻时间（Hi-Z time of the x data bus in general memory） 当在通用存储空间 x 开始执行对 PC 卡/NAND Flash 的写操作后，数据总线需要保持一段时间的高阻状态，该参数以 HCLK 时钟周期数目（NAND 类型时+1）定义数据总线高阻态的时间。这个参数仅对写操作有效。 • 00000000：（0x00）PC 卡为 0 个 HCLK 周期/NAND Flash 为 1 个 HCLK 周期 • … • 11111110：（0xFC）PC 卡为 255 个 HCLK 周期/NAND Flash 为 256 个 HCLK 周期，这是复位后的默认值。 • 11111111：保留
位 23:16	MEMHOLDx[7:0]：在通用空间 x 的保持时间（Hold time of the x data bus in general memory） 当在通用存储空间 x 对 PC 卡/NAND Flash 进行读或写操作时，该参数以 HCLK 时钟周期数目定义了发送命令（NWE、NOE 变高）后，地址信号（对于写操作则是数据信号）保持的时间。 • 00000000：保留 • 00000001：1 个 HCLK 周期 • … • 11111110：254 个 HCLK 周期（复位后的默认值） • 11111111：保留
位 15:8	MEMWAITx[7:0]：在通用空间 x 的等待时间（Wait time of the x data bus in general memory） 当在通用存储空间 x 对 PC 卡/NAND Flash 进行读或写操作时，该参数以 HCLK（+1）时钟周期数目定义了保持命令（NWE、NOE 为低）的最小时间。当该参数定义的时间结束时，如果等待信号（NWAIT）有效（低），则命令的保持时间会被拉长。 • 00000000：保留 • 00000001：2 个 HCLK 周期（加上由 NWAIT 信号变低引入的等待周期） • … • 11111110：255 个 HCLK 周期（加上由卡的 NWAIT 信号变低引入的等待周期），这是复位后的默认值。 • 11111111：保留
位 7:0	MEMSETx[7:0]：在通用空间 x 的建立时间（Setup time of the x data bus in general memory） 当在通用存储空间 x 对 PC 卡/NAND Flash 进行读或写操作时，该参数以 HCLK（操作 PC 卡时+1，操作 NAND Flash 时+2）时钟周期数目定义了发送命令（NWE、NOE 变低）之前建立地址信号的时间。 • 00000000：PC 卡为 1 个 HCLK 周期/NAND Flash 为 2 个 HCLK 周期 • … • 11111110：PC 卡为 256 个 HCLK 周期/NAND Flash 为 257 个 HCLK 周期，这是复位后的默认值。 • 11111111：保留

15.7.2.4 属性存储空间时序寄存器 x（FSMC_PATTx）（x=2..4）

偏移地址：0x4C+0x20*（x-1）

复位值：0xFCFC FCFC

每个 FSMC_PATTx 读/写寄存器都包含操作 PC 卡/CF 卡或 NAND Flash 存储块 x 的时序参数，这些参数适用于在属性存储空间操作 8 位 PC 卡/CF 卡（每个 AHB 操作被分解为一系列的 8 位操作），或在 NAND Flash 的最后一个地址写操作的时序与其它操作不同的时候（关于就绪/繁忙的管理，参见 NAND Flash 预等待功能）。

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

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

位 31:24	ATTHIZx：在属性空间 x 数据总线的高阻时间（Hi-Z time of the x data bus in attribute memory） 当在属性存储空间 x 开始执行对 PC 卡/NAND Flash 的写操作后，数据总线需要保持一段时间的高阻状态，该参数以 HCLK 时钟周期数目定义数据总线高阻态的时间。这个参数仅对写操作有效。 • 00000000：0 个 HCLK 周期 • … • 11111110：255 个 HCLK 周期 • 11111111：保留
位 23:16	ATTHOLDx：在属性空间 x 的保持时间（Hold time of the x data bus in attribute memory） 当在属性存储空间 x 对 PC 卡/NAND Flash 进行读或写操作时，该参数以 HCLK 时钟周期数目定义了发送命令（NWE、NOE 变高）后，地址信号（对于写操作则是数据信号）保持的时间。 • 00000000：保留 • 00000001：1 个 HCLK 周期 • … • 11111110：254 个 HCLK 周期 • 11111111：保留
位 15:8	ATTWAITx：在属性空间 x 的等待时间（Wait time of the x data bus in attribute memory） 当在属性存储空间 x 对 PC 卡/NAND Flash 进行读或写操作时，该参数以 HCLK（+1）时钟周期数目定义了保持命令（NWE、NOE 为低）的最小时间。当该参数定义的时间结束时，如果等待信号（NWAIT）有效（低），则命令的保持时间会被拉长。 • 00000000：1 个 HCLK 周期（加上由 NWAIT 信号变低引入的等待周期） • … • 111111110：255 个 HCLK 周期（加上由卡的 NWAIT 信号变低引入的等待周期）。 • 11111111：保留
位 7:0	ATTSETx：在属性空间 x 的建立时间（Setup time of the x data bus in attribute memory） 当在属性存储空间 x 对 PC 卡/NAND Flash 进行读或写操作时，该参数以 HCLK（+1）时钟周期数目定义了发送命令（NWE、NOE 变低）之前建立地址信号的时间。 • 00000000：1 个 HCLK 周期 • … • 11111110：255 个 HCLK 周期 • 11111111：保留

15.7.2.5 I/O 空间时序寄存器 4（FSMC_PIO4）

偏移地址：0xB0

复位值：0xFCFCFCFC

FSMC_PIO4 寄存器包含了在 I/O 空间操作 16 位 PC 卡/CF 卡的时序参数。

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

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

位 31:24	IOHIZx：在 I/O 空间 x 数据总线的高阻时间（Hi-Z time of the x data bus in I/O space） 当在 I/O 空间 x 开始执行对 PC 卡的写操作后，数据总线需要保持一段时间的高阻状态，该参数以 HCLK 时钟周期数目定义数据总线高阻态的时间。这个参数仅对写操作有效。 • 00000000：0 个 HCLK 周期 • … • 11111111：255 个 HCLK 周期
位 23:16	IOHOLDx：在 I/O 空间 x 的保持时间（Hold time of the x data bus in I/O space） 当在 I/O 空间 x 对 PC 卡进行读或写操作时，该参数以 HCLK 时钟周期数目定义了发送命令（NWE、NOE 变高）后，地址信号（对于写操作则是数据信号）保持的时间。 • 00000000：保留 • 00000001：1 个 HCLK 周期 • … • 11111111：255 个 HCLK 周期
位 15:8	IOWAITx：在 I/O 空间 x 的等待时间（Wait time of the x data bus in I/O space） 当在 I/O 空间 x 对 PC 卡进行读或写操作时，该参数以 HCLK（+1）时钟周期数目定义了保持命令（SMNWE、SMNOE 为低）的最小时间。当该参数定义的时间结束时，如果等待信号（NWAIT）有效（低），则命令的保持时间会被拉长。 • 00000000：保留，不要使用这个数值 • 00000001：2 个 HCLK 周期（加上由 NWAIT 信号变低引入的等待周期） • … • 11111111：256 个 HCLK 周期（加上由卡的 NWAIT 信号变低引入的等待周期）
位 7:0	IOSETx：在 I/O 空间 x 的建立时间（Setup time of the x data bus in I/O space） 当在 I/O 空间 x 对 PC 卡进行读或写操作时，该参数以 HCLK（+1）时钟周期数目定义了发送命令（NWE、NOE 变低）之前建立地址信号的时间。 • 00000000：1 个 HCLK 周期 • … • 11111111：256 个 HCLK 周期

15.7.2.6 ECC 结果寄存器 x（FSMC_ECCRx）（x=2..3）

偏移地址：0x54+0x20*（x-1）

复位值：0x0000 0000

这 2 个寄存器包含了由 FSMC 控制器的 ECC 计算模块得到的纠错码的当前数值，每个 NAND Flash 存储器块有一个 ECC 计算模块。当 CPU 在正确的地址（参见 NAND Flash 的纠错码 ECC 计算）读/写 NAND Flash 的数据时，ECC 模块会自动地处理写入或读出的数据。根据 FSMC_PCRx 中 ECCPS 域的设置，在读出了每页的最后一个字节后，CPU 必须读出 FSMC_ECCx 寄存器中的 ECC 数值，并与记录在 NAND Flash 备份域的数据进行比较，据此判断该页的数据是否正确并在可能的情况下，实行矫正。在读出 FSMC_ECCRx 寄存器的数值后应设置 ECCEN 位为'0'清除它的内容。需要计算一个新的数据页时，再次设置 ECCEN 为'1'。

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

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

rw

位 31:0	ECCx：ECC 结果（ECC result） ECC 计算电路产生的计算结果。下表显示了该位域的内容。

表 15-35 ECC 结果相关位

ECCPS[2:0]	页大小（字节）	ECC 有效位
000	256	ECC[21:0]
001	512	ECC[23:0]
010	1024	ECC[25:0]
011	2048	ECC[27:0]
100	4096	ECC[29:0]
101	8192	ECC[31:0]

16 高级控制定时器（TIM1 和 TIM8）

高级控制定时器（TIM1 和 TIM8）由一个 16 位的自动装载计数器组成，它由一个可编程的预分频器驱动。

高级控制定时器适合多种用途，包含测量输入信号的脉冲宽度（输入捕获），或者产生输出波形（输出比较、PWM、带死区插入的互补 PWM 等）。

使用定时器预分频器和 RCC 时钟控制预分频器，可以实现脉冲宽度和波形周期从几个微秒到几个毫秒的调节。

高级控制定时器（TIM1 和 TIM8）和通用定时器（TIMx）是完全独立的，它们不共享任何资源。这两种定时器可以同步操作，详情可参考“16.2.20 定时器同步”。

TIM8 功能和 TIM1 完全一样，只是寄存器基地址与 TIM1 不同；寄存器基地址请查阅“2.2.1 存储器映射”。

16.1 TIM1 和 TIM8 主要特性

TIM1 和 TIM8 定时器的功能包括：

- DAC 触发功能
- 四路输入通道均支持上升沿、下降沿和双边沿触发功能
- 16 位向上、向下、向上/下自动装载计数器
- 16 位可编程（支持实时修改）预分频器，计数器时钟频率的分频系数为 1～65536 之间的任意数值
- 4 个独立通道：
  - 输入捕获
  - 输出比较
  - PWM 生成（边沿或中央对齐模式）
  - 单脉冲模式输出
- 带死区时间可编程的互补输出
- 使用外部信号控制定时器和定时器互联的同步电路
- 允许在指定数目的计数器周期之后更新定时器寄存器的重复计数器
- 刹车输入信号可以将定时器输出信号置于复位状态或者一个已知状态
- 如下事件发生时产生中断/DMA：
  - 更新：计数器向上溢出/向下溢出，计数器初始化（通过软件或者内部/外部触发）
  - 触发事件（计数器启动、停止、初始化或者由内部/外部触发计数）
  - 输入捕获
  - 输出比较
  - 刹车信号输入
- 支持针对定位的增量（正交）编码器和霍尔传感器电路
- 触发输入作为外部时钟或者按周期的电流管理

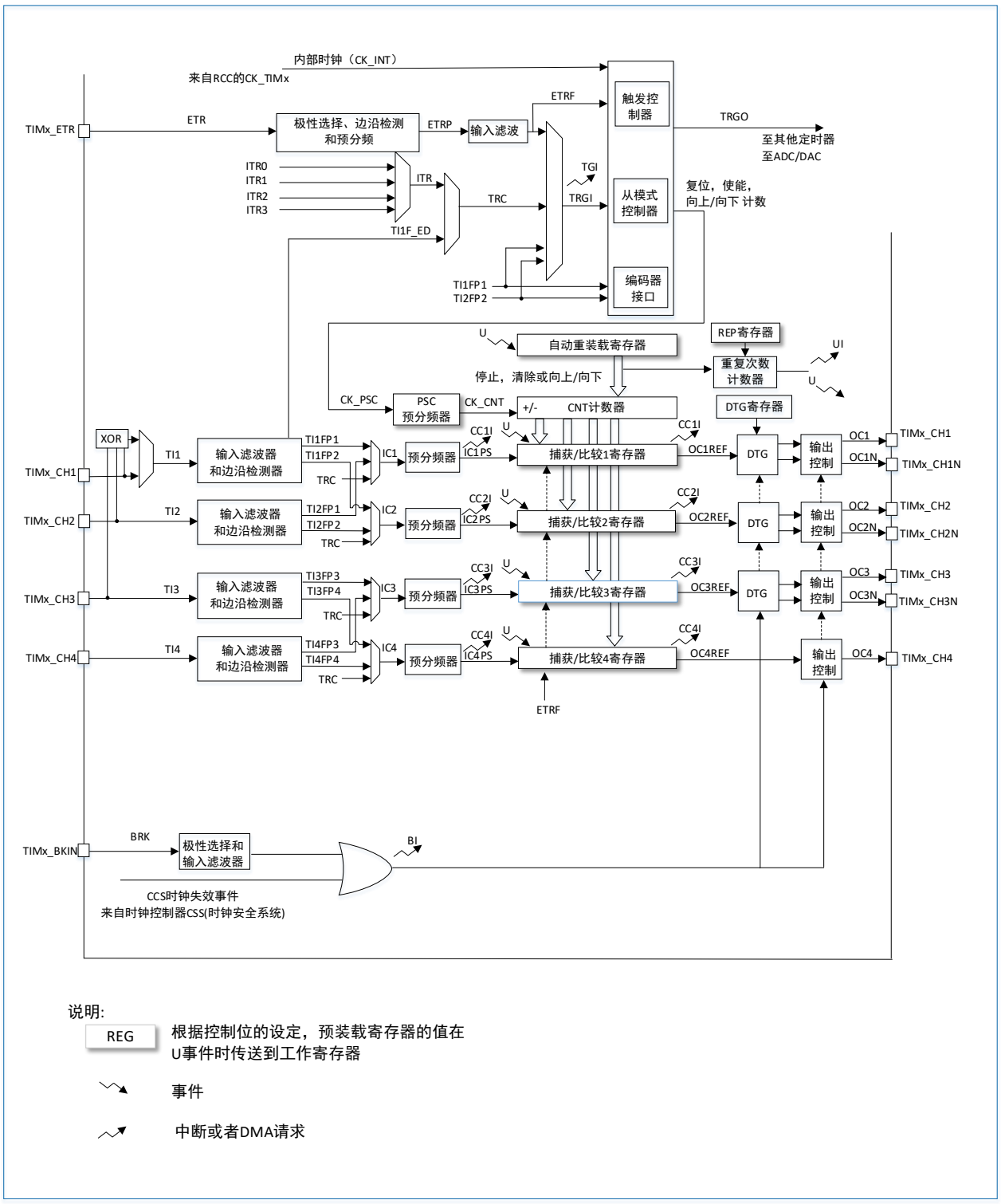


图 16-1 高级控制定时器框图

上图的说明：TIM1 不具有 DAC 触发功能。

16.2 TIM1 和 TIM8 功能描述

16.2.1 时基单元

可编程高级控制定时器的主要部分是一个 16 位计数器和与其相关的自动装载寄存器。这个计数器可以向上计数、向下计数或者向上向下双向计数。此计数器时钟由预分频器分频得到。

计数器、自动装载寄存器和预分频器寄存器可以由软件读写。即使计数器还在运行，读写仍然有效。

时基单元包含：

- 计数器寄存器（TIMx_CNT）
- 预分频器寄存器（TIMx_PSC）
- 自动重装载寄存器（TIMx_ARR）
- 重复次数寄存器（TIMx_RCR）

自动重装载寄存器是预先装载的，写或读自动重装载寄存器将访问预装载寄存器。根据在 TIMx_CR1 寄存器中的自动装载预装载使能位（ARPE）的设置，预装载寄存器的内容被立即或在每次的更新事件 UEV 时传送到影子寄存器。当计数器达到溢出条件（如向下计数时的下溢条件）并当 TIMx_CR1 寄存器中的 UDIS 位等于 0 时，产生更新事件。更新事件也可以由软件产生。随后会详细描述每一种配置下更新事件的产生。

计数器由预分频器的时钟输出 CK_CNT 驱动，仅当设置了计数器 TIMx_CR1 寄存器中的计数器使能位（CEN）时，CK_CNT 才有效。（更多有关使能计数器的细节，请参见控制器的从模式描述）。

注意：在设置了 TIMx_CR 寄存器的 CEN 位的一个时钟周期后，计数器开始计数。

TIMx_PCS、TIMx_ARR、TIMx_RCR 等寄存器具有预装载功能，分别由各自的预装载寄存器和影子寄存器组成。影子寄存器是实际起作用的寄存器。

16.2.1.1 预分频器描述

预分频器可以将计数器的时钟频率按 1 到 65536 之间的任意值分频。它是基于一个（在 TIMx_PSC 寄存器中的）16 位寄存器控制的 16 位计数器。因为这个控制寄存器带有缓冲功能，它能够在运行时被改变。新的预分频器的参数在下一次更新事件到来时被采用。

下面两个图给出了在预分频器运行时，更改计数器参数的例子。

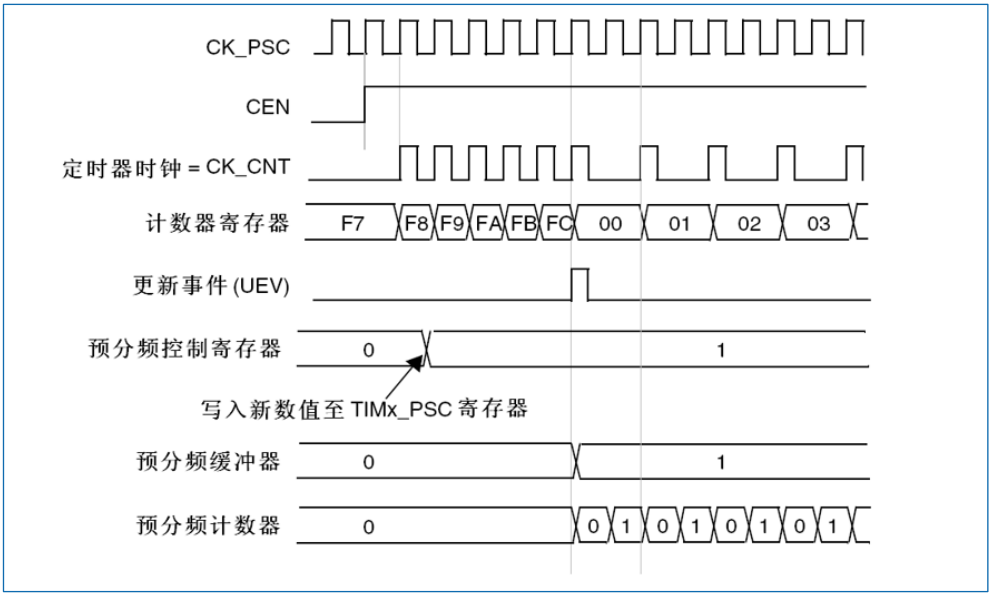


图 16-2 当预分频器的参数从 1 变到 2 时，计数器的时序图

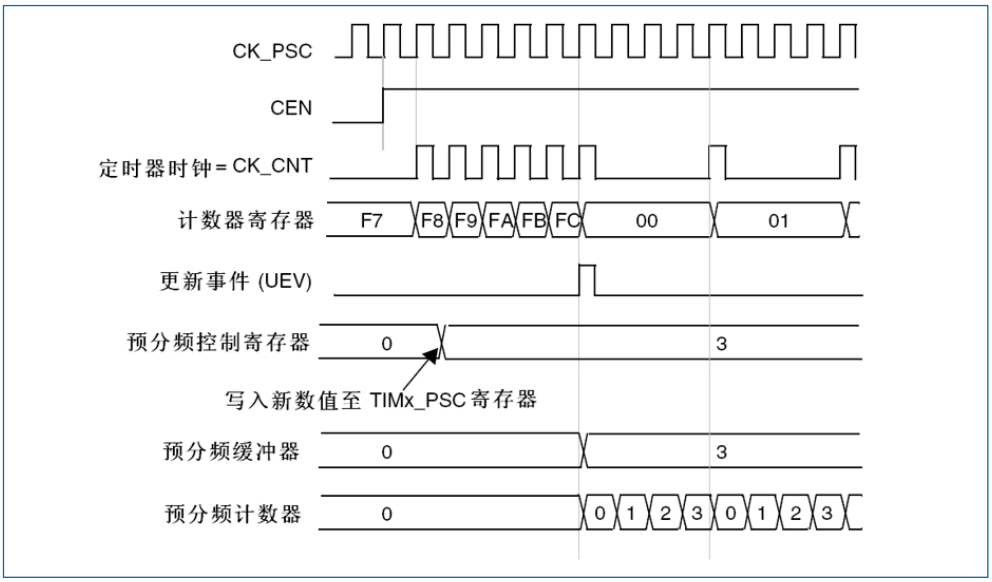


图 16-3 当预分频器的参数从 1 变到 4 时，计数器的时序图

16.2.2 计数器模式

16.2.2.1 向上计数模式

在向上计数模式中，计数器从 0 计数到自动加载值（TIMx_ARR 寄存器的值），然后重新从 0 开始计数并且产生一个计数器溢出事件。

如果使用了重复计数器功能，在向上计数达到设置的重复计数次数加 1（TIMx_RCR+1）时，产生更新事件（UEV）；否则每次计数器溢出时才产生更新事件。

在 TIMx_EGR 寄存器中（通过软件方式或者使用从模式控制器）设置 UG 位也可产生一个更新事件。

设置 TIMx_CR1 寄存器中的 UDIS 位，可以禁止更新事件；这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。在 UDIS 位被清零之前，将不产生更新事件。但是计数器仍会被清零，同时预分频器的计数也会被清 0（但预分频器的数值不变）。此外，如果设置了 TIMx_CR1 寄存器中的 URS 位（选择更新请求），设置 UG 位将产生一个更新事件 UEV，但硬件不设置 UIF 标志（即不产生中断或 DMA 请求）。这是为了避免在捕获模式下清除计数器时，同时产生更新和捕获中断。

当发生一个更新事件时，所有的寄存器都被更新，且硬件同时（依据 URS 位）设置更新标志位（TIMx_SR 寄存器中的 UIF 位）：

- 重复计数器被重新加载为 TIMx_RCR 寄存器的内容。
- 自动装载影子寄存器被重新置入预装载寄存器的值（TIMx_ARR）。
- 预分频器的缓冲区被置入预装载寄存器的值（TIMx_PSC 寄存器的内容）。

下面给出一些例子，当 TIMx_ARR=0x36 时计数器在不同时钟频率下的动作。

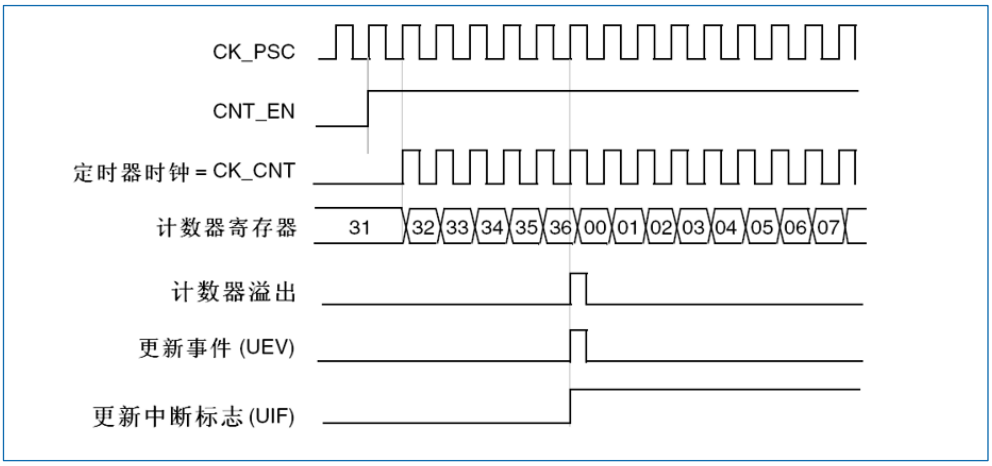


图 16-4 计数器时序图，内部时钟分频因子为 1

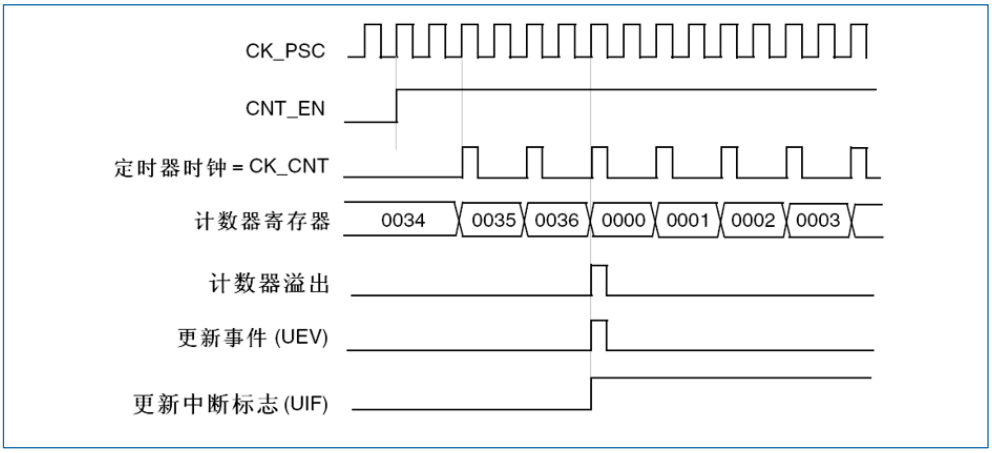


图 16-5 计数器时序图，内部时钟分频因子为 2

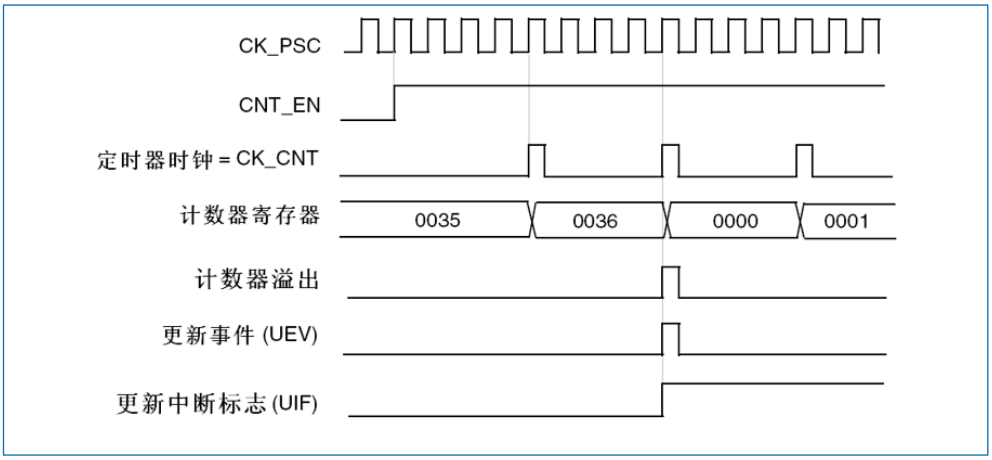


图 16-6 计数器时序图，内部时钟分频因子为 4

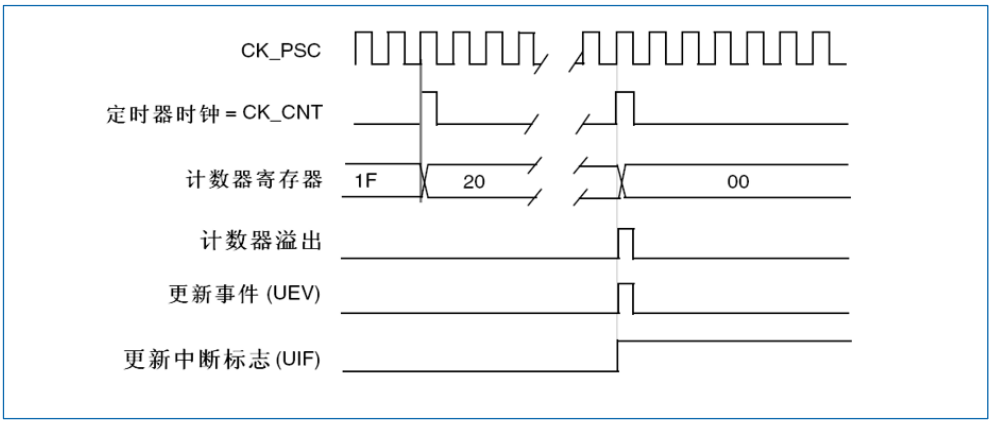


图 16-7 计数器时序图，内部时钟分频因子为 N

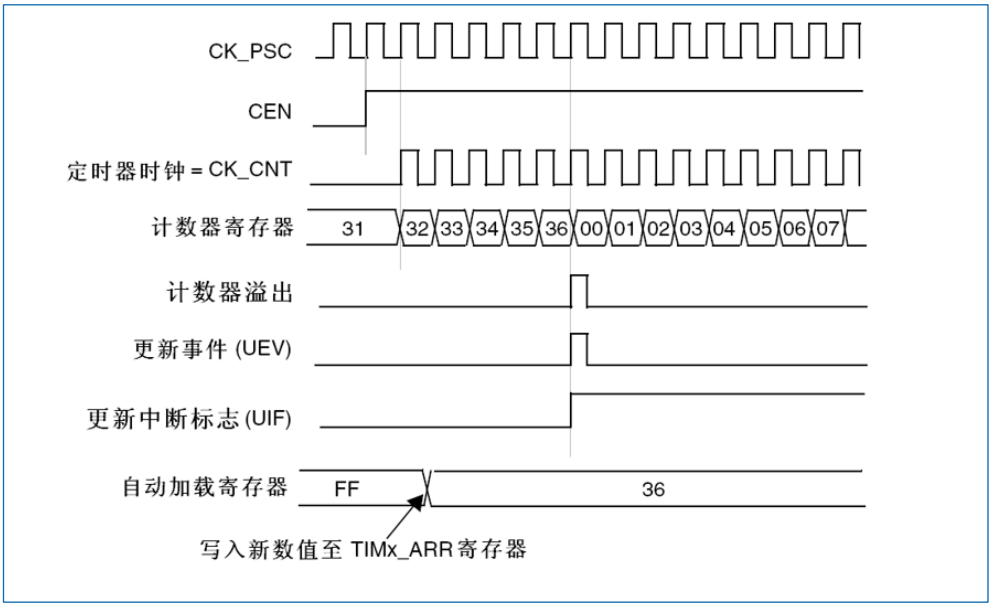


图 16-8 计数器时序图，当 ARPE=0 时的更新事件（TIMx_ARR 没有预装入）

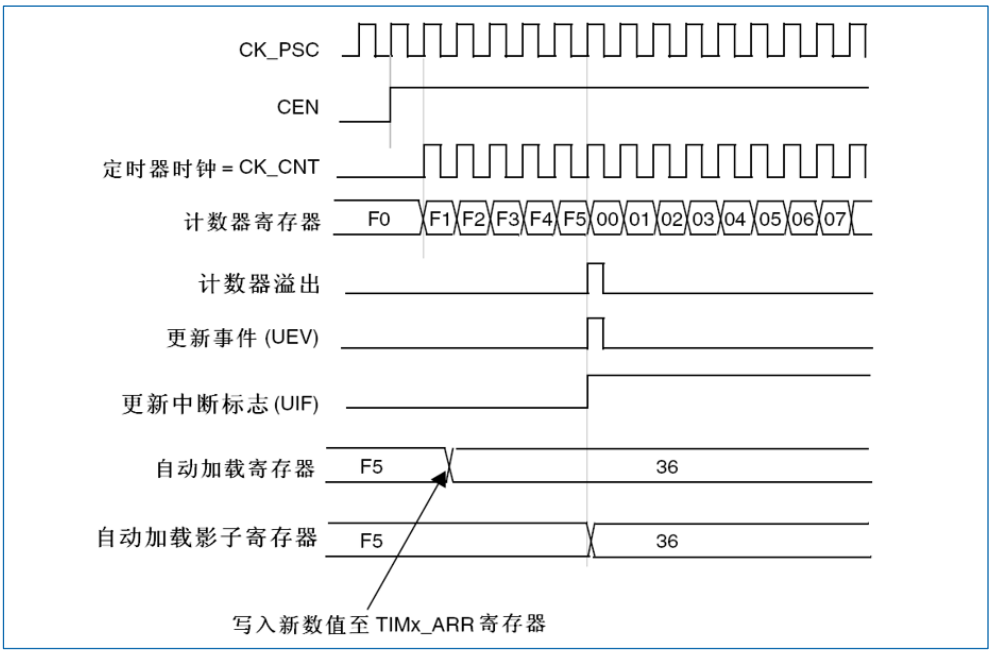


图 16-9 计数器时序图，当 ARPE=1 时的更新事件（预装入了 TIMx_ARR）

16.2.2.2 向下计数模式

在向下模式中，计数器从自动装载的值（TIMx_ARR 计数器的值）开始向下计数到 0，然后从自动装载的值重新开始并且产生一个计数器向下溢出事件。

如果使用了重复计数器，当向下计数达到了重复计数寄存器（TIMx_RCR）中设定的次数后，将产生更新事件（UEV），否则每次计数器下溢时才产生更新事件。

在 TIMx_EGR 寄存器中（通过软件方式或者使用从模式控制器）设置 UG 位，也同样可以产生一个更新事件。

设置 TIMx_CR1 寄存器的 UDIS 位可以禁止 UEV 事件。这样可以避免向预装载寄存器中写入新值时更新影子寄存器。因此 UDIS 位被清为 0 之前不会产生更新事件。然而，计数器仍会从当前自动加载值重新开始计数，并且预分频器的计数器重新从 0 开始（但预分频系数不变）。

此外，如果设置了 TIMx_CR1 寄存器中的 URS 位（选择更新请求），设置 UG 位将产生一个更新事件（UEV）但不设置 UIF 标志（因此不产生中断和 DMA 请求）。这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

当发生更新事件时，所有的寄存器都被更新，并且（根据 URS 位的设置）更新标志位（TIMx_SR 寄存器中的 UIF 位）也被设置：

- 重复计数器被重置为 TIMx_RCR 寄存器中的值。
- 预分频器的缓冲器被加载为预装载的值（TIMx_PSC 寄存器的值）。
- 当前的自动重装载寄存器被更新为预装载值（TIMx_ARR 寄存器中的内容）。

注意：自动装载在计数器重装载之前被更新，因此下一个周期将是预期的值。

以下是一些当 TIMx_ARR=0x36 时，计数器在不同时钟频率下的操作例子。

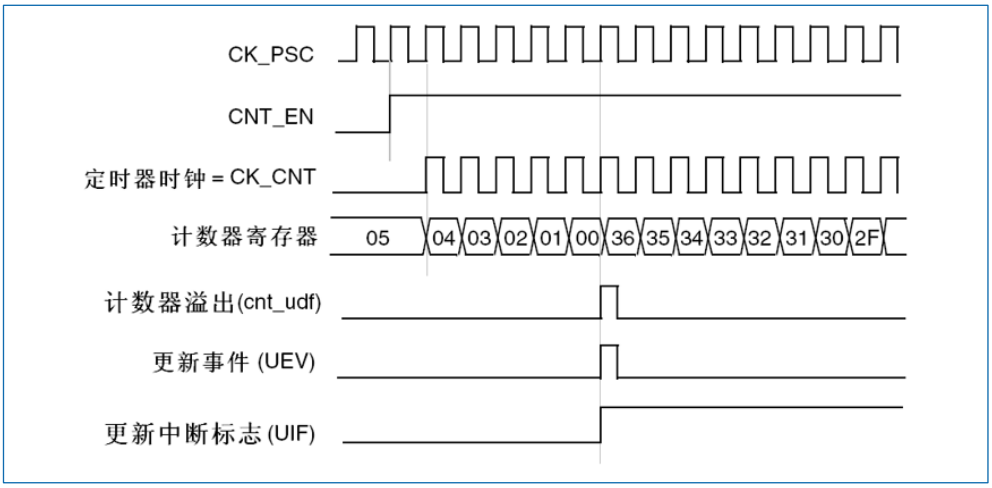


图 16-10 计数器时序图，内部时钟分频因子为 1

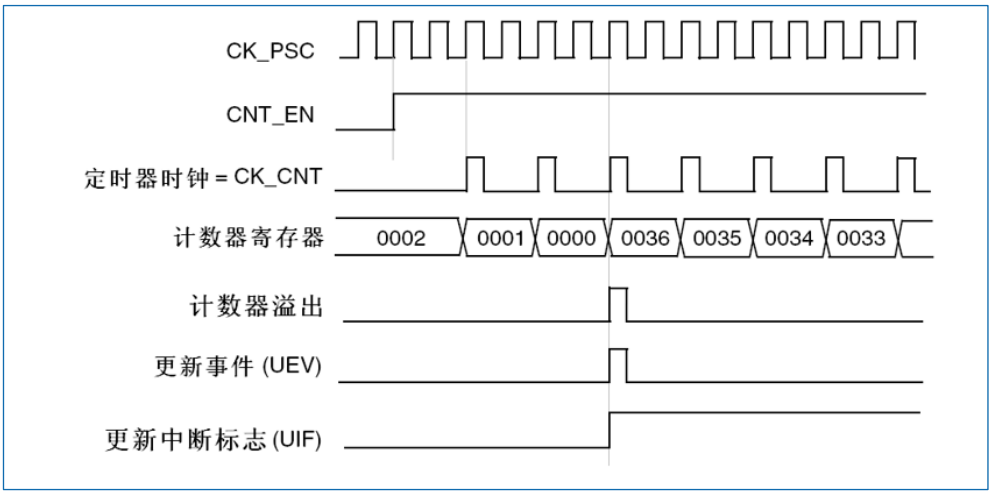


图 16-11 计数器时序图，内部时钟分频因子为 2

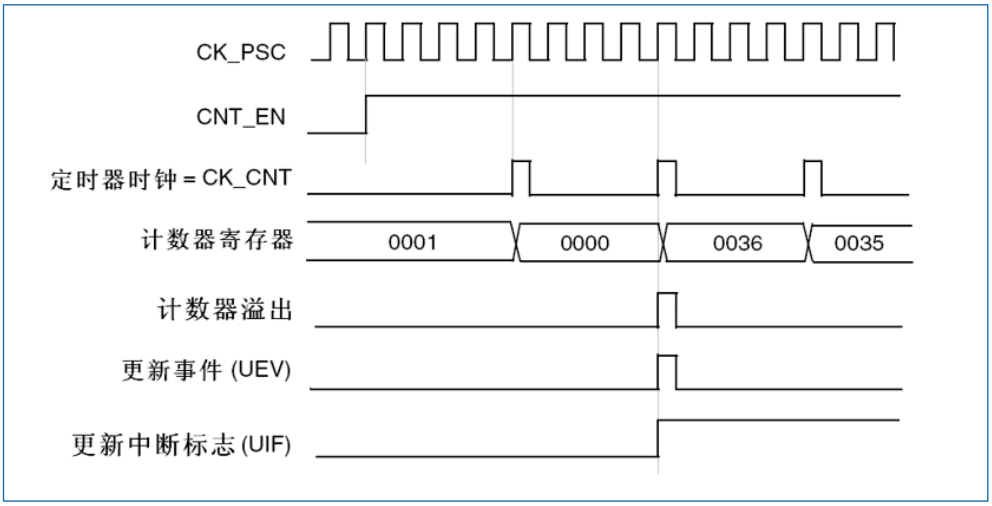


图 16-12 计数器时序图，内部时钟分频因子为 4

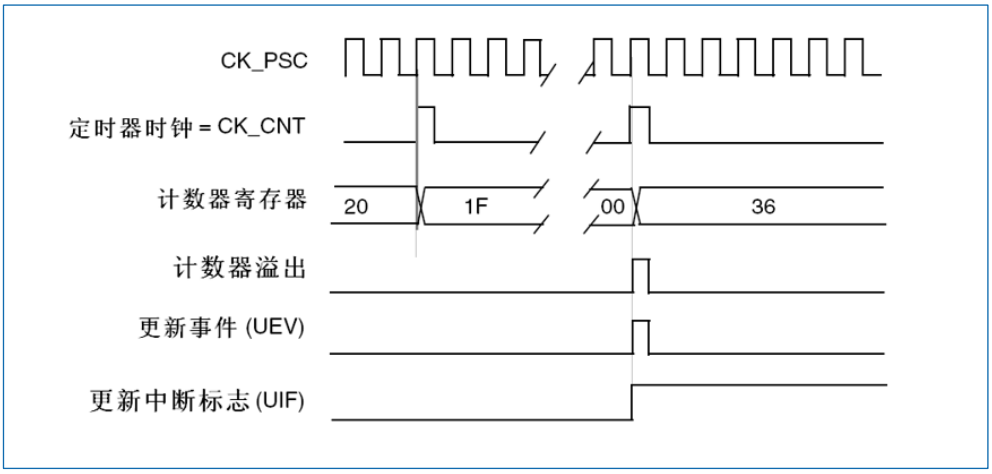


图 16-13 计数器时序图，内部时钟分频因子为 N

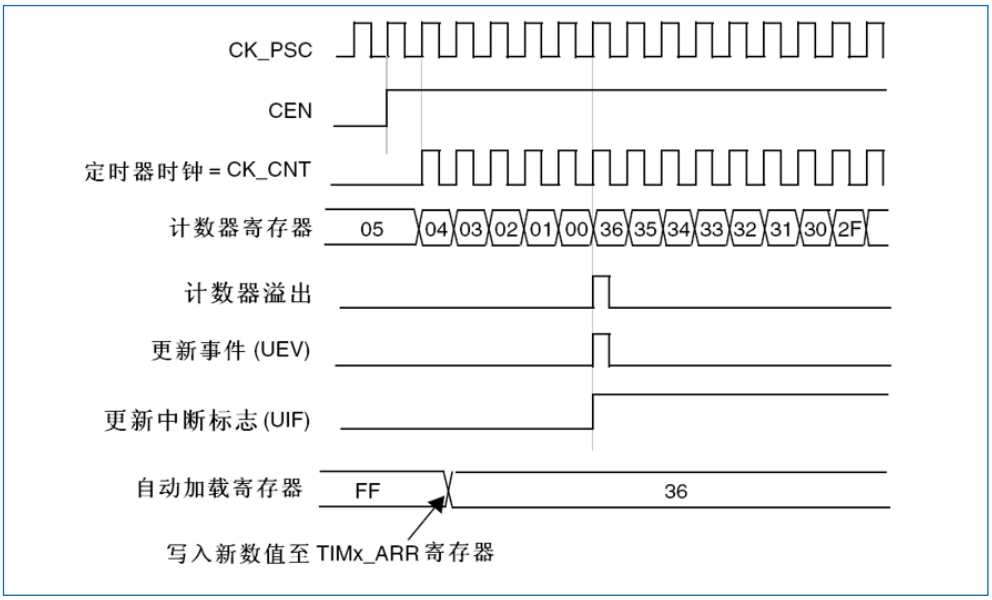


图 16-14 计数器时序图，当没有使用重复计数器时的更新事件

16.2.2.3 中央对齐模式（向上/向下计数）

在中央对齐模式，计数器从 0 开始计数到自动装载的值减 1（TIMx_ARR 寄存器的值−1），产生一个计数器溢出事件，然后向下计数到 1 并且产生一个计数器下溢事件；然后再从 0 开始重新计数。

在此模式下，不能写入 TIMx_CR1 中的 DIR 方向位。它由硬件更新并指示当前的计数方向。

可以在每次计数上溢和每次计数下溢时产生更新事件；也可以通过（软件或者使用从模式控制器）设置 TIMx_EGR 寄存器中的 UG 位产生更新事件。然后，计数器重新从 0 开始计数，预分频器也重新从 0 开始计数。

设置 TIMx_CR1 寄存器中的 UDIS 位可以禁止 UEV 事件。这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。因此 UDIS 位被清为 0 之前不会产生更新事件。然而，计数器仍会根据当前自动重加载的值，继续向上或向下计数。

此外，如果设置了 TIMx_CR1 寄存器中的 URS 位（选择更新请求），设置 UG 位将产生一个更新事件（UEV）但不设置 UIF 标志（因此不产生中断和 DMA 请求），这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

当发生更新事件时，所有的寄存器都被更新，并且（根据 URS 位的设置）更新标志位（TIMx_SR 寄存器中的 UIF 位）也被设置：

- 重复计数器被重置为 TIMx_RCR 寄存器中的值。
- 预分频器的缓存器被加载为预装载（TIMx_PSC 寄存器）的值。
- 当前的自动加载寄存器被更新为预装载值（TIMx_ARR 寄存器中的内容）。

注意：如果因为计数器溢出而产生更新，自动重装载将在计数器重载入之前被更新，因此下一个周期将是预期的值（计数器被装载为新的值）。

以下是一些计数器在不同时钟频率下的操作的例子：

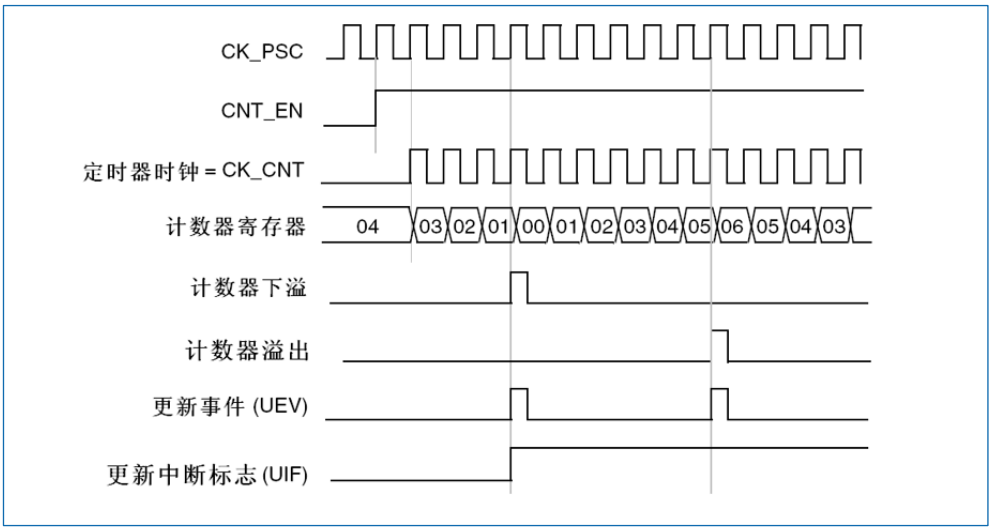


图 16-15 计数器时序图，内部时钟分频因子为 1，TIMx_ARR=0x6

图 16-15 中使用了中央对齐模式 1，详见“TIM1/TIM8 控制寄存器 1（TIMx_CR1）”。

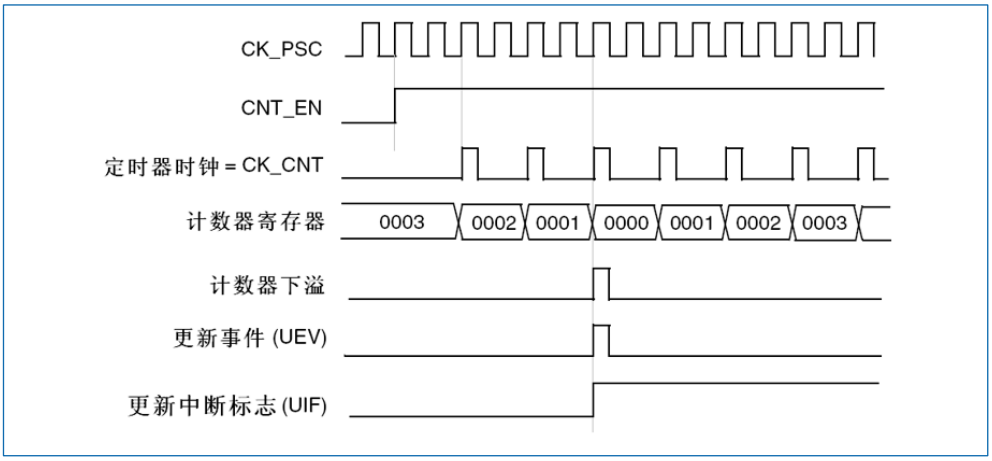


图 16-16 计数器时序图，内部时钟分频因子为 2

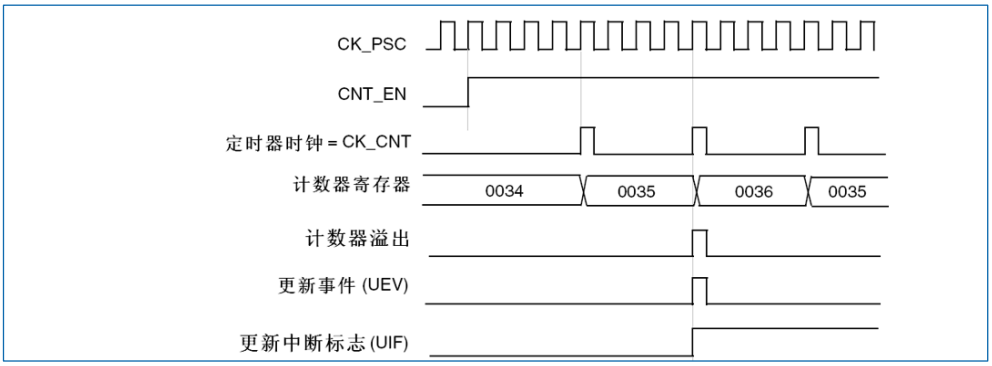


图 16-17 计数器时序图，内部时钟分频因子为 4，TIMx_ARR=0x36

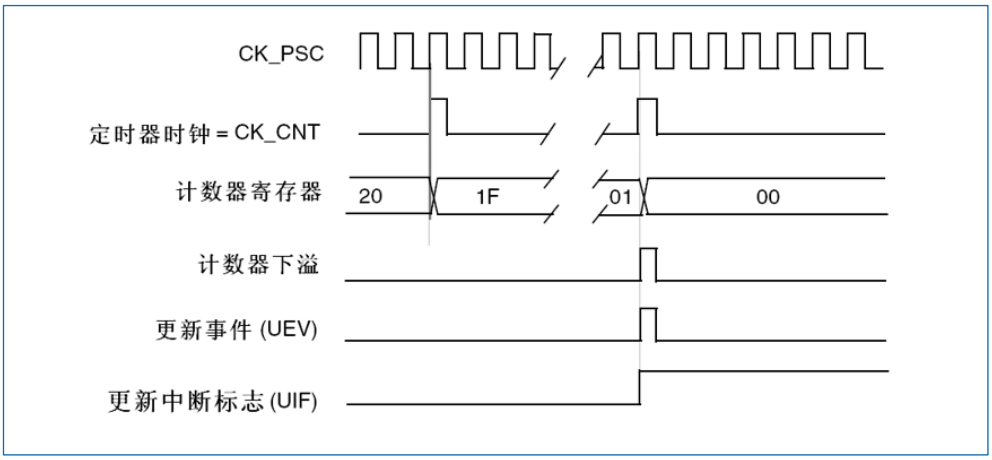


图 16-18 计数器时序图，内部时钟分频因子为 N

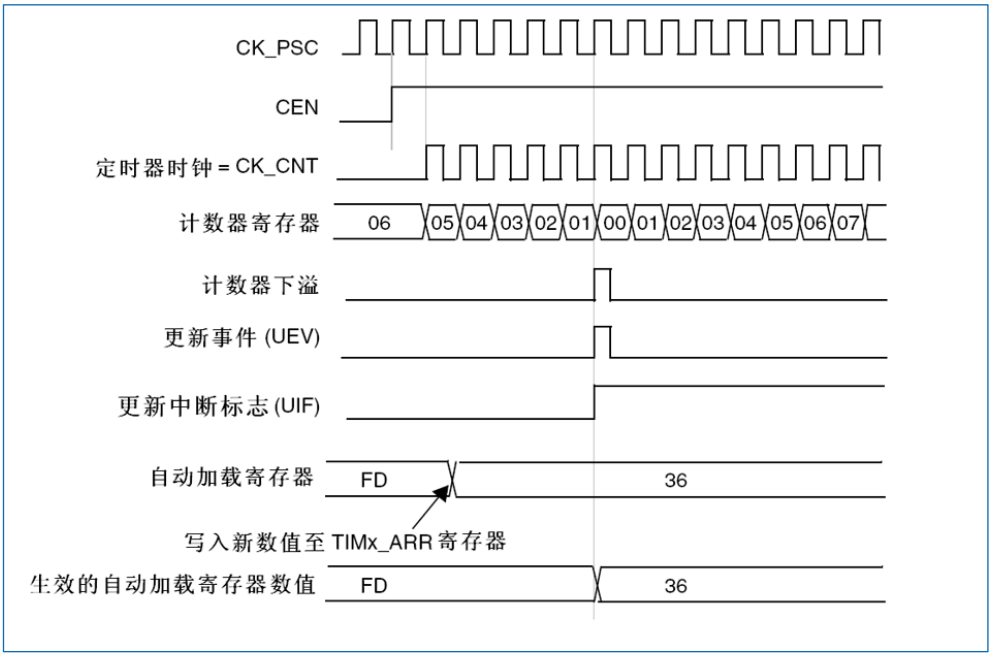


图 16-19 计数器时序图，ARPE=1 时的更新事件（计数器下溢）

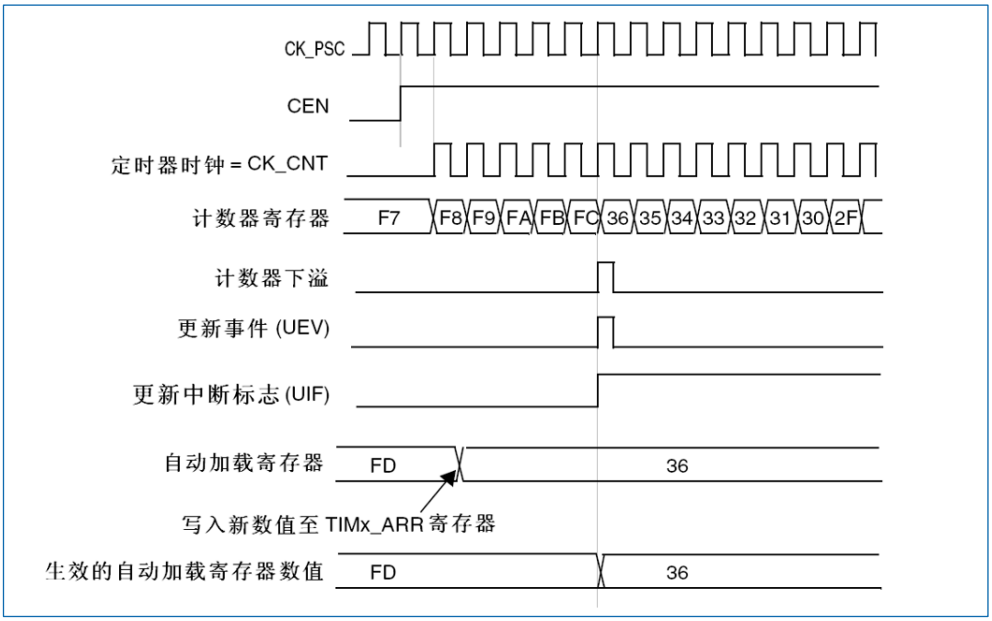


图 16-20 计数器时序图，ARPE=1 时的更新事件（计数器上溢）

16.2.3 重复计数器

“16.2.1 时基单元”解释了计数器上溢/下溢时更新事件（UEV）是如何产生的，然而事实上它只能在重复计数达到 0 的时候产生。这个特性对产生 PWM 信号非常有用。

这意味着在每 N+1 次计数上溢或下溢时，数据从预装载寄存器传输到影子寄存器（TIMx_ARR 自动重装载寄存器，TIMx_PSC 预装载寄存器，还有在比较模式下的捕获/比较寄存器 TIMx_CCRx），N 是 TIMx_RCR 重复计数寄存器中的值。

重复计数器在下述任一条件成立时递减：

- 向上计数模式下每次计数器上溢时
- 向下计数模式下每次计数器下溢时
- 中央对齐模式下每次上溢和每次下溢时。虽然这样限制了 PWM 的最大循环周期为 128，但它能够在每个 PWM 周期 2 次更新占空比。在中央对齐模式下，因为波形是对称的，如果每个 PWM 周期中仅刷新一次比较寄存器，则最大的分辨率为 $2xT_{ck}$。

重复计数器是自动加载的，重复速率是由 TIMx_RCR 寄存器的值定义。当更新事件由软件产生（通过设置 TIMx_EGR 中的 UG 位）或者通过硬件的从模式控制器产生，则无论重复计数器的值是多少，立即发生更新事件，并且 TIMx_RCR 寄存器中的内容被重载入到重复计数器。

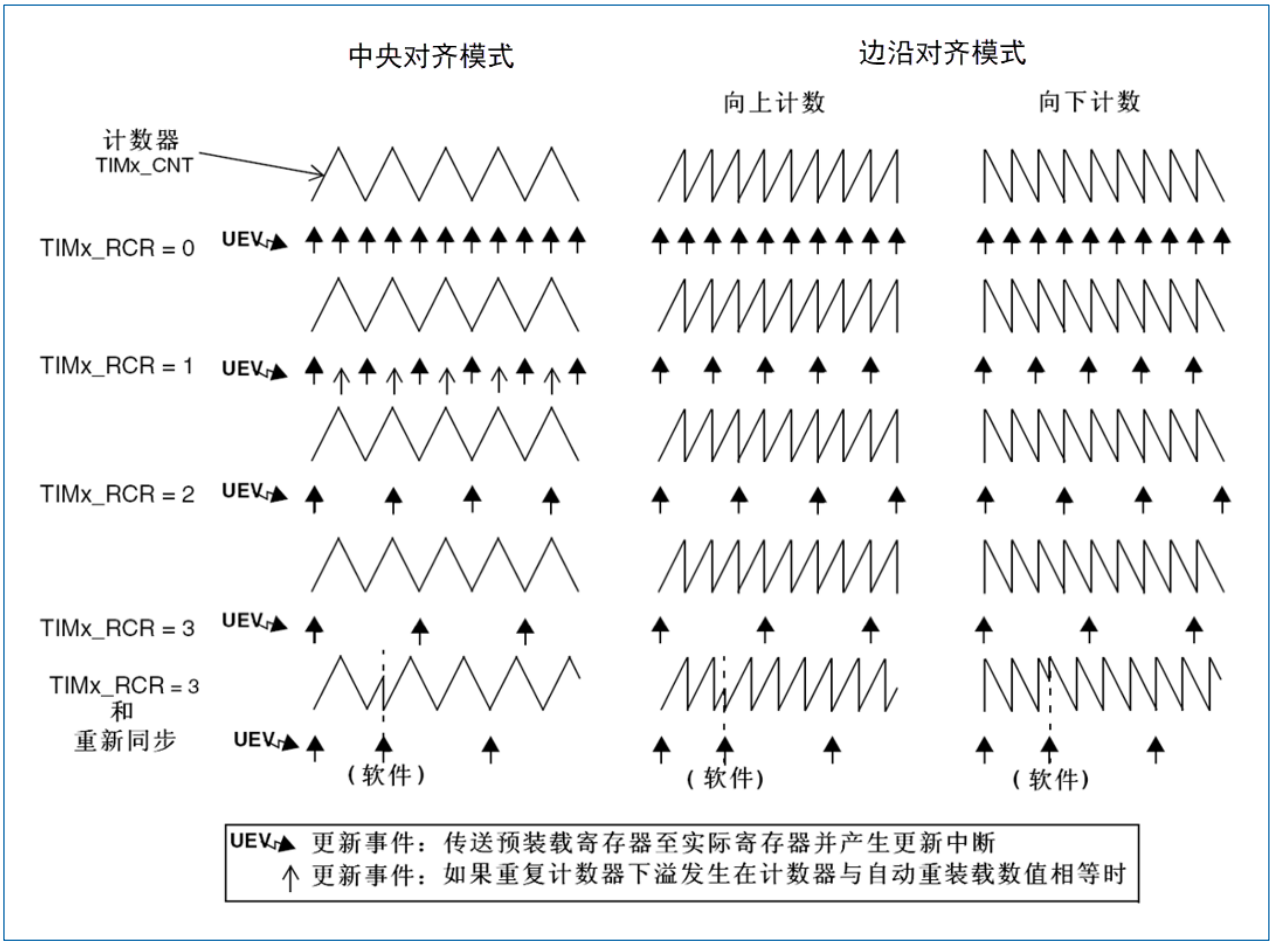


图 16-21 不同模式下更新速率的例子，及 TIMx_RCR 的寄存器设置

16.2.4 时钟选择

计数器时钟可由下列时钟源提供：

- 内部时钟（CK_INT）
- 外部时钟模式 1：外部输入引脚

- 外部时钟模式 2：外部触发输入 ETR
- 内部触发输入（ITRx）：使用一个定时器作为另一个定时器的预分频器。如可以配置一个定时器 Timer1 而作为另一个定时器 Timer2 的预分频器。详见“16.2.20 定时器同步”。

16.2.4.1 内部时钟源（CK_INT）

如果禁止了从模式控制器（SMS=000），则 CEN、DIR（TIMx_CR1 寄存器）和 UG 位（TIMx_EGR 寄存器）是事实上的控制位，并且只能被软件修改（UG 位仍被自动清除）。只要 CEN 位被写成’1’，预分频器的时钟就由内部时钟 CK_INT 提供。

下图显示控制电路和向上计数器在一般模式下，不带预分频器时的操作。

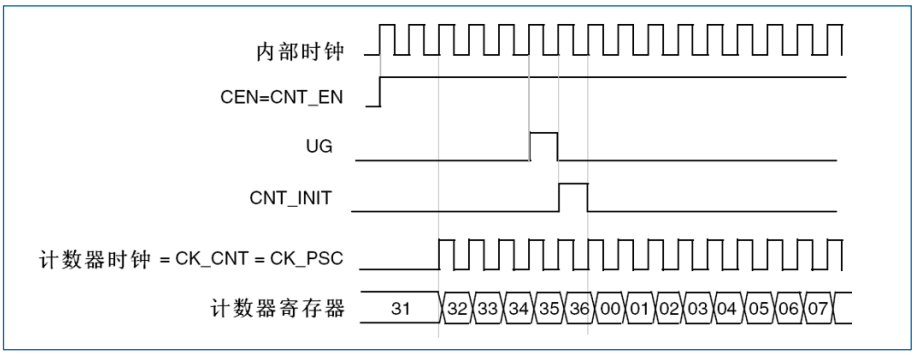


图 16-22 一般模式下的控制电路，内部时钟分频因子为 1

16.2.4.2 外部时钟源模式 1

当 TIMx_SMCR 寄存器的 SMS=111 时，此模式被选中。计数器可以在选定输入端的每个上升沿或下降沿计数。

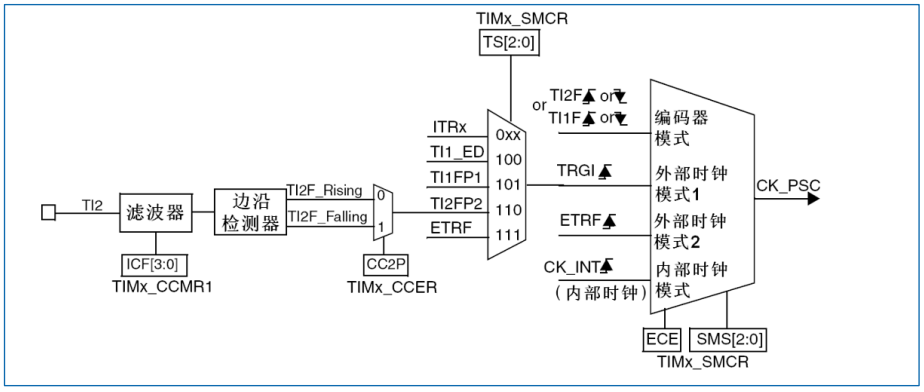


图 16-23 TI2 外部时钟连接例子

例如，配置向上计数器在 T12 输入端的上升沿计数，步骤如下：

1. 配置 TIMx_CCMR1 寄存器 CC2S=01，以设置通道 2 检测 TI2 输入的上升沿。
2. 配置 TIMx_CCMR1 寄存器的 IC2F[3:0]，以选择输入滤波器带宽（如果不需要滤波器，保持 IC2F=0000）。
3. 配置 TIMx_CCER 寄存器的 CC2P=0，以选择上升沿极性。
4. 配置 TIMx_SMCR 寄存器的 SMS=111，以选择定时器工作在外部时钟模式 1。
5. 配置 TIMx_SMCR 寄存器中的 TS=110，以选择 TI2 作为触发输入源。
6. 配置 TIMx_CR1 寄存器的 CEN=1，以使能计数器。

注意：捕获预分频器不用作触发，所以不需要对它进行配置。

当上升沿出现在 TI2，计数器计数一次，且 TIF 标志被设置。

在 TI2 的上升沿和计数器实际时钟之间的延时，取决于在 TI2 输入端的重新同步电路。

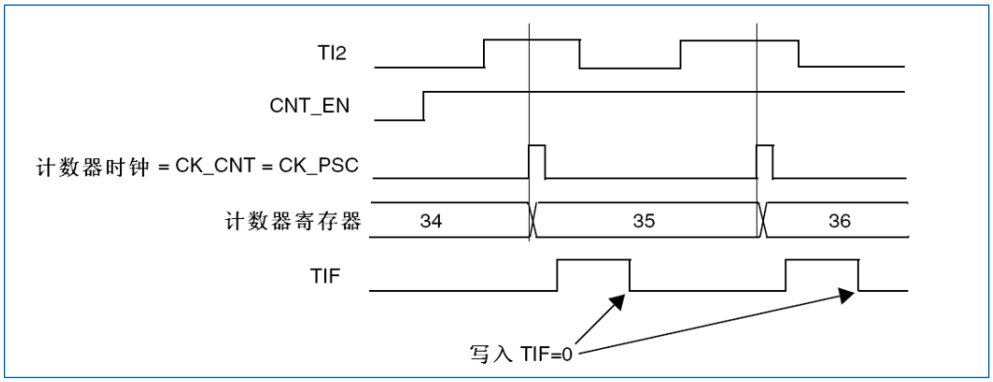


图 16-24 外部时钟模式 1 下的控制电路

16.2.4.3 外部时钟源模式 2

当 TIMx_SMCR 寄存器的 ECE=1 时，此模式被选中。计数器能够在外部触发 ETR 的每一个上升沿或下降沿计数。

下图是外部触发输入的框图。

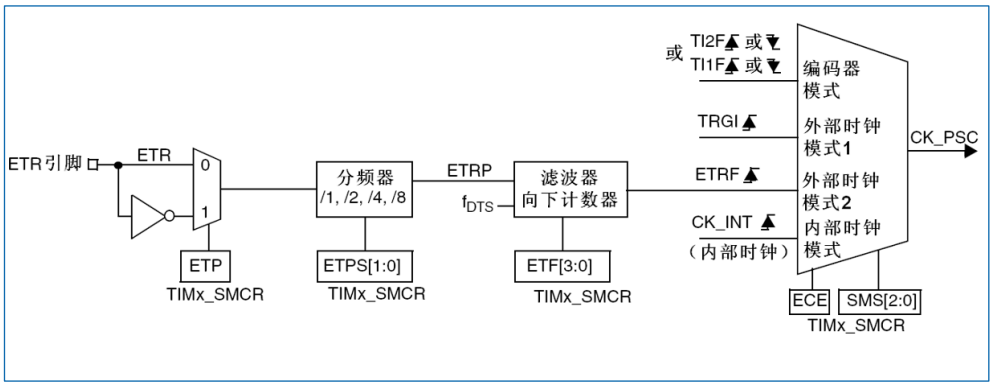


图 16-25 外部触发输入框图

例如，配置在 ETR 下每 2 个上升沿计数一次的向上计数器，步骤如下：

1. 本例中不需要滤波器，置 TIMx_SMCR 寄存器中的 ETF[3:0]=0000。
2. 设置预分频器，置 TIMx_SMCR 寄存器中的 ETPS[1:0]=01。
3. 选择 ETR 的上升沿检测，置 TIMx_SMCR 寄存器中的 ETP=0。
4. 开启外部时钟模式 2，写 TIMx_SMCR 寄存器中的 ECE=1。
5. 启动计数器，写 TIMx_CR1 寄存器中的 CEN=1。
6. 计数器在每 2 个 ETR 上升沿计数一次。

在 ETR 的上升沿和计数器实际时钟之间的延时取决于在 ETRP 信号端的重新同步电路。

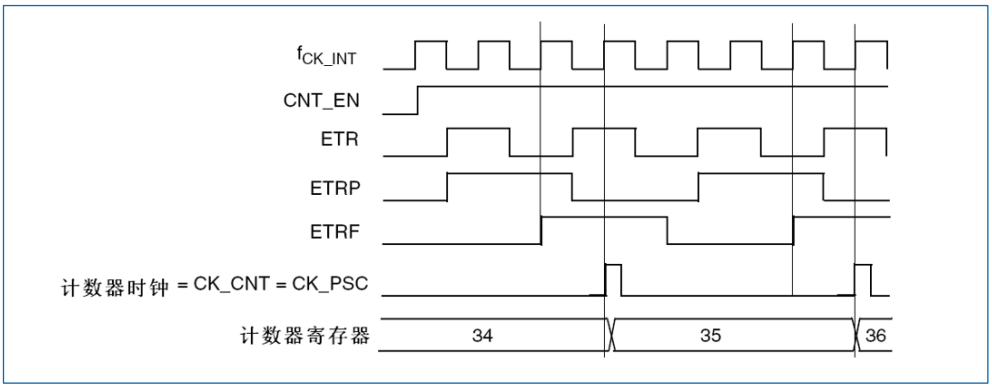


图 16-26 外部时钟模式 2 下的控制电路

16.2.5 捕获/比较通道

每一个捕获/比较通道都是围绕着一个捕获/比较寄存器（包含影子寄存器），包括捕获的输入部分（数字滤波、多路复用和预分频器），和输出部分（比较器和输出控制）。

图 16-27 和图 16-28 是一个捕获/比较通道概览。

输入部分对相应的 TIx 输入信号采样，并产生一个滤波后的信号 TIxF。然后，一个带极性选择的边沿检测器产生一个信号（TIxFPx），它可以作为从模式控制器的输入触发或者作为捕获控制。该信号通过预分频进入捕获寄存器（ICxPS）。

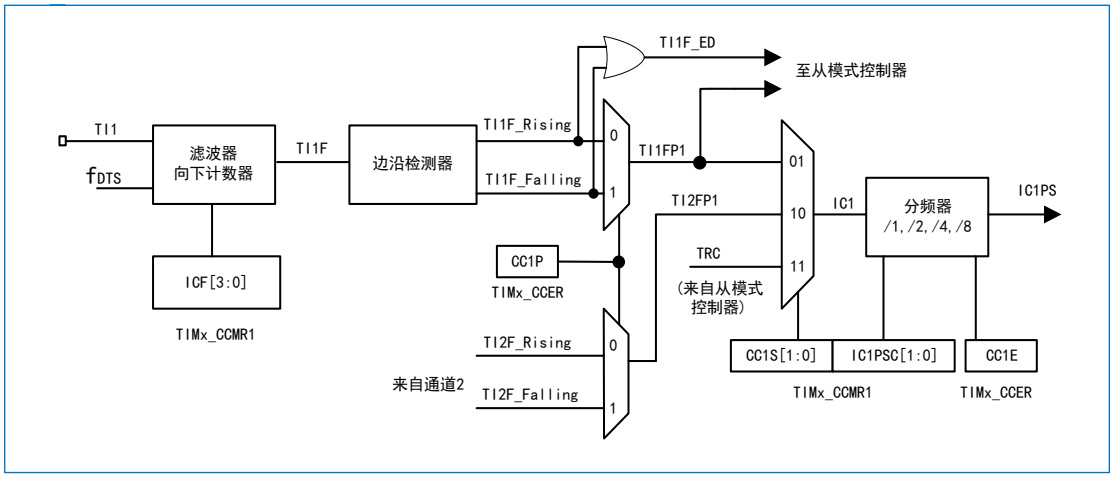


图 16-27 捕获/比较通道（如：通道 1 输入部分）

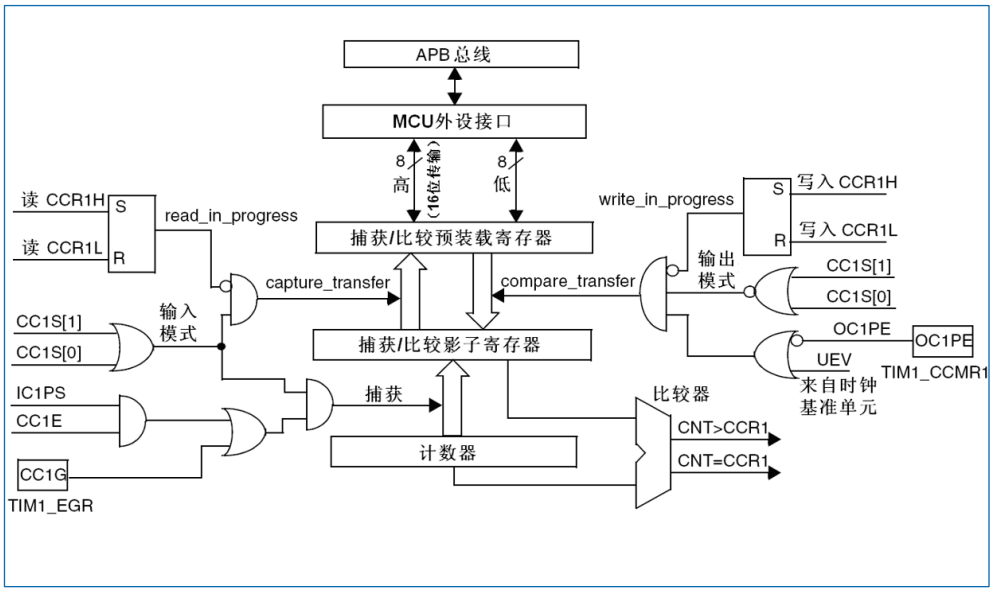


图 16-28 捕获/比较通道 1 的主电路

输出部分产生一个中间波形 OCxRef（高有效）作为基准，链的末端决定最终输出信号的极性。

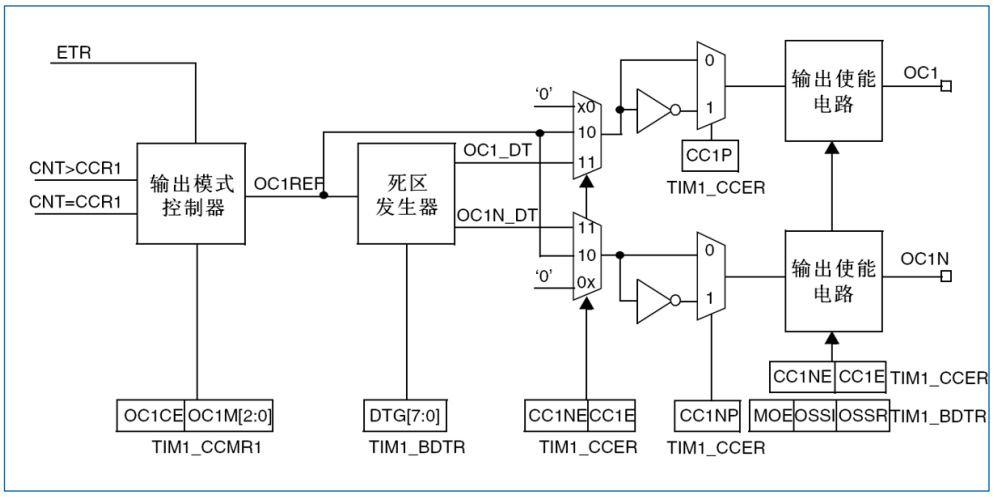


图 16-29 捕获/比较通道的输出部分（通道 1 至 3）

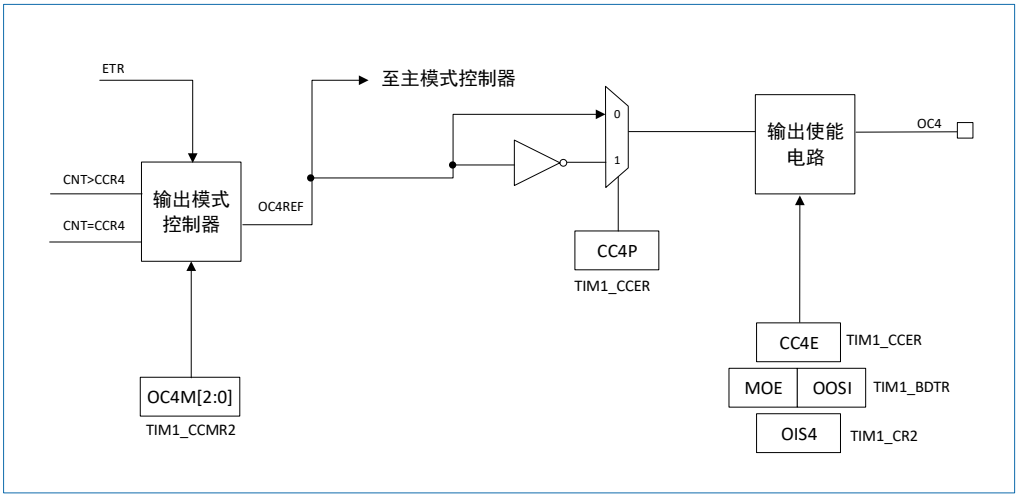


图 16-30 捕获/比较通道的输出部分（通道 4）

捕获/比较模块由一个预装载寄存器和一个影子寄存器组成。读写过程仅操作预装载寄存器。

在捕获模式下，捕获发生在影子寄存器上，然后再复制到预装载寄存器中。

在比较模式下，预装载寄存器的内容被复制到影子寄存器中，然后影子寄存器的内容和计数器进行比较。

16.2.6 输入捕获模式

在输入捕获模式下，当检测到 ICx 信号上相应的边沿后，计数器的当前值被锁存到捕获/比较寄存器（TIMx_CCRx）中。当发生捕获事件时，相应的 CCxIF 标志（TIMx_SR 寄存器）被置'1'，如果开放了中断或者 DMA 操作，则将产生中断或者 DMA 请求。如果发生捕获事件时 CCxIF 标志已经为高，那么重复捕获标志 CCxOF（TIMx_SR 寄存器）被置'1'。写 CCxIF=0 可清除 CCxIF，或读取存储在 TIMx_CCRx 寄存器中的捕获数据也可清除 CCxIF。写 CCxOF=0 可清除 CCxOF。

以下例子说明如何在 TI1 输入的上升沿时捕获计数器的值到 TIMx_CCR1 寄存器中，步骤如下：

1. 选择有效输入端：TIMx_CCR1 必须连接到 TI1 输入，所以写入 TIMx_CCMR1 寄存器中的 CC1S=01。只要 CC1S 不为'00'，通道被配置为输入，并且 TIMx_CCR1 寄存器变为只读。
2. 根据输入信号的特点，配置输入滤波器为所需的带宽（即输入为 TIx 时，输入滤波器控制位是 TIMx_CCMRx 寄存器中的 ICxF 位）。假设输入信号在最多 5 个内部时钟周期的时间内抖动，必须配置滤波器的带宽大于 5 个时钟周期；因此可以（以 f_{DTS} 频率）连续采样 8 次，以确认在 TI1 上一次真实的边沿变换，即在 TIMx_CCMR1 寄存器中写入 IC1F=0011。
3. 选择 TI1 通道的有效转换边沿，在 TIMx_CCER 寄存器中写入 CC1P=0（上升沿）。
4. 配置输入预分频器。在本例中，希望捕获发生在每一个有效的电平转换时刻，因此预分频器被禁止（写 TIMx_CCMR1 寄存器的 IC1PS=00）。
5. 设置 TIMx_CCER 寄存器的 CC1E=1，允许捕获计数器的值到捕获寄存器中。
6. 如果需要，通过设置 TIMx_DIER 寄存器中的 CC1IE 位允许相关中断请求，通过设置 TIMx_DIER 寄存器中的 CC1DE 位允许 DMA 请求。

当发生一个输入捕获时：

- 产生有效的电平转换时，计数器的值被传送到 TIMx_CCR1 寄存器。
- CC1IF 标志被设置（中断标志）。当发生至少 2 个连续的捕获时，而 CC1IF 未曾被清除，则 CC1OF 也被置'1'。
- 如设置了 CC1IE 位，则会产生一个中断。
- 如设置了 CC1DE 位，则还会产生一个 DMA 请求。

为了处理捕获溢出，建议在读出捕获溢出标志之前读取数据，这是为了避免丢失在读出捕获溢出标志之后和读取数据之前可能产生的捕获溢出信息。

注意：设置 TIMx_EGR 寄存器中相应的 CCxG 位，可以通过软件产生输入捕获中断和/或 DMA 请求。

16.2.7 PWM 输入模式

该模式是输入捕获模式的一个特例，除下列区别外，操作与输入捕获模式相同：

- 两个 ICx 信号被映射至同一个 TIx 输入。
- 这 2 个 ICx 信号为边沿有效，但是极性相反。
- 其中一个 TIxFP 信号被作为触发输入信号，而从模式控制器被配置成复位模式。

例如，你需要测量输入到 TI1 上的 PWM 信号的周期（TIMx_CCR1 寄存器）和占空比（TIMx_CCR2 寄存器），具体步骤如下（取决于 CK_INT 的频率和预分频器的值）：

1. 选择 TIMx_CCR1 的有效输入：置 TIMx_CCMR1 寄存器的 CC1S=01（选中 TI1）。

2. 选择 TI1FP1 的有效极性（用来捕获数据到 TIMx_CCR1 和清除计数器）：置 CC1P=0（上升沿有效）。

3. 选择 TIMx_CCR2 的有效输入：置 TIMx_CCMR1 寄存器的 CC2S=10（选中 TI1）。

4. 选择 TI1FP2 的有效极性（捕获数据到 TIMx_CCR2）：置 CC2P=1（下降沿有效）。

5. 选择有效的触发输入信号：置 TIMx_SMCR 寄存器中的 TS=101（选择 TI1FP1）。

6. 配置从模式控制器为复位模式：置 TIMx_SMCR 中的 SMS=100。

7. 使能捕获：置 TIMx_CCER 寄存器中 CC1E=1 且 CC2E=1。

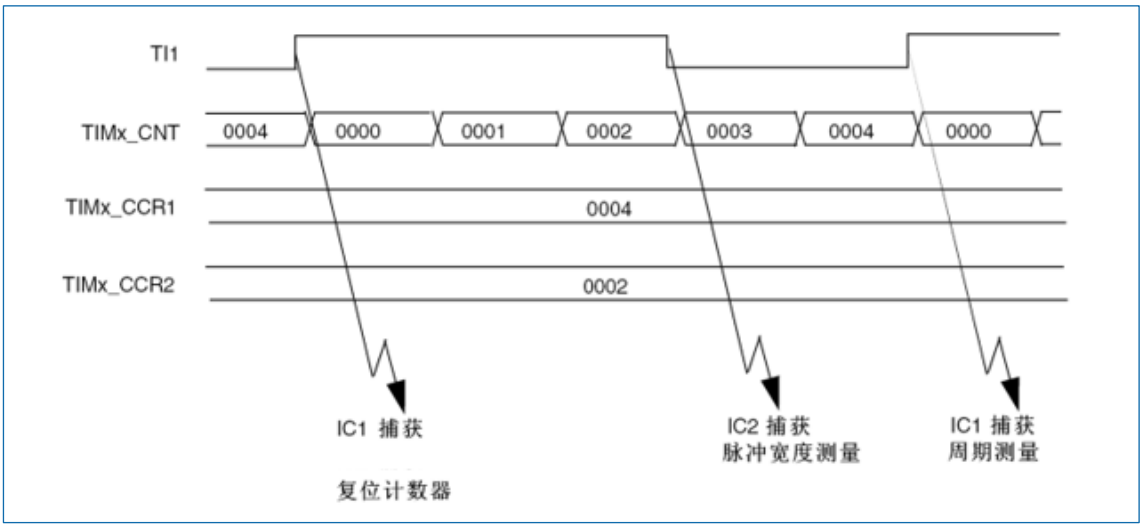


图 16-31 PWM 输入模式时序

因为只有 TI1FP1 和 TI2FP2 连到了从模式控制器，所以 PWM 输入模式只能使用 TIMx_CH1/TIMx_CH2 信号。

16.2.8 强置输出模式

在输出模式（TIMx_CCMRx 寄存器中 CCxS=00）下，输出比较信号（OCxREF 和相应的 OCx/OCxN）能够直接由软件强置为有效或无效状态，而不依赖于输出比较寄存器和计数器间的比较结果。

置 TIMx_CCMRx 寄存器中相应的 OCxM=101，即可强置输出比较信号（OCxREF/OCx）为有效状态。这样 OCxREF 被强置为高电平（OCxREF 始终为高电平有效），同时 OCx 得到 CCxP 极性相反的信号。

例如：CCxP=0（OCx 高电平有效），则 OCx 被强置为高电平。置 TIMx_CCMRx 寄存器中的 OCxM=100，可强置 OCxREF 信号为低。

该模式下，在 TIMx_CCRx 影子寄存器和计数器之间的比较仍然在进行，相应的标志也会被修改。因此仍然会产生相应的中断和 DMA 请求。这将会在下面的输出比较模式一节中介绍。

16.2.9 输出比较模式

该功能用于控制输出波形，或者指示一段给定的时间已经到时。当计数器与捕获/比较寄存器的内容相同时，输出比较功能做如下操作：

- 将输出比较模式（TIMx_CCMRx 寄存器中的 OCxM 位）和输出极性（TIMx_CCER 寄存器中的 CCxP 位）定义的值输出到对应的引脚上。在比较匹配时，输出引脚可以保持它的电平（OCxM=000）、被设置成有效电平（OCxM=001）、被设置成无效电平（OCxM=010）或进行翻转（OCxM=011）。
- 设置中断状态寄存器中的标志位（TIMx_SR 寄存器中的 CCxIF 位）。

- 若设置了相应的中断使能（TIMx_DIER 寄存器中的 CCxIE 位），则产生一个中断。
- 若设置了相应的使能位（TIMx_DIER 寄存器中的 CCxDE 位，TIMx_CR2 寄存器中的 CCDS 位选择 DMA 请求功能），则产生一个 DMA 请求。

TIMx_CCMRx 中的 OCxPE 位选择 TIMx_CCRx 寄存器是否需要使用预装载寄存器。在输出比较模式下，更新事件 UEV 对 OCxREF 和 OCx 输出没有影响。

同步的精度可以达到计数器的一个计数周期。输出比较模式（在单脉冲模式下）也能用来输出一个单脉冲。

配置输出比较模式的步骤：

1. 选择计数器时钟（内部、外部或预分频器）。
2. 将相应的数据写入 TIMx_ARR 和 TIMx_CCRx 寄存器中。
3. 如果要产生一个中断请求，设置 CCxIE 位。
4. 选择输出模式，例如：
   - 要求计数器与 CCRx 匹配时翻转 OCx 的输出引脚，设置 OCxM=011。
   - 设置 OCxPE=0，以禁用预装载寄存器。
   - 设置 CCxP=0，以选择极性为高电平有效。
   - 设置 CCxE=1，以使能输出。
5. 设置 TIMx_CR1 寄存器的 CEN 位启动计数器。

TIMx_CCRx 寄存器能够在任何时候通过软件进行更新以控制输出波形，条件是未使用预装载寄存器（OCxPE=’0’，否则 TIMx_CCRx 的影子寄存器只能在发生下一次更新事件时被更新）。下图给出了一个例子。

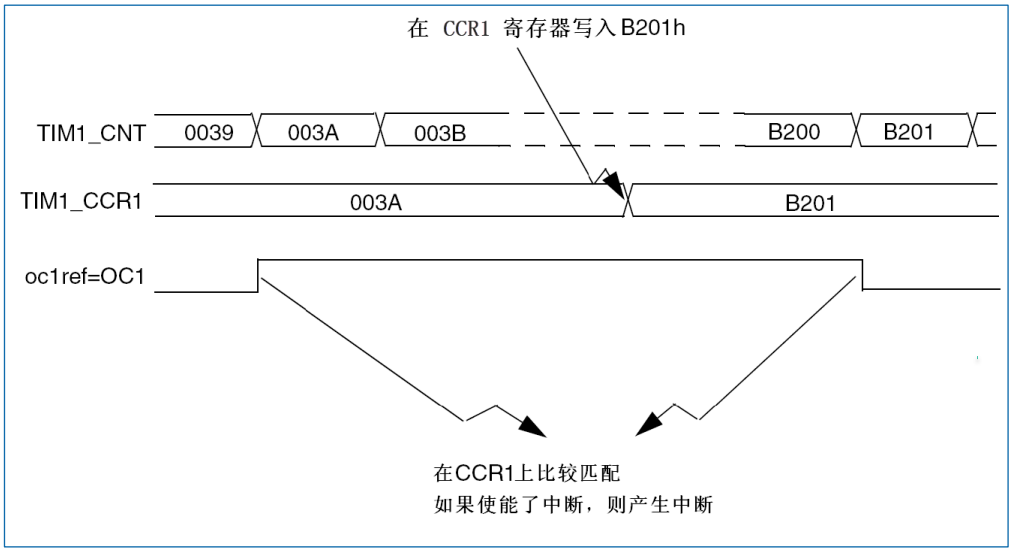


图 16-32 输出比较模式，翻转 OC1

16.2.10 PWM 模式

脉冲宽度调制模式可以产生一个由 TIMx_ARR 寄存器确定频率、由 TIMx_CCRx 寄存器确定占空比的信号。

在 TIMx_CCMRx 寄存器中的 OCxM 位写入’110’（PWM 模式 1）或’111’（PWM 模式 2），能够独立地设置每个 OCx 输出通道产生一路 PWM。必须通过设置 TIMx_CCMRx 寄存器的 OCxPE 位使能相应的预装载寄存器，最后还要设置 TIMx_CR1 寄存器的 ARPE 位，（在向上计数或中心对称模式中）使能自动重装载的预装载寄存器。

仅当发生一个更新事件的时候，预装载寄存器才能被传送到影子寄存器，因此在计数器开始计数之前，必须通过设置 TIMx_EGR 寄存器中的 UG 位来初始化所有的寄存器。OCx 的极性可以通过软件在 TIMx_CCER 寄存器中的 CCxP 位设置，它可以设置为高电平有效或低电平有效。OCx 的输出使能通过（TIMx_CCER 和 TIMx_BDTR 寄存器中）CCxE、CCxNE、MOE、OSSI 和 OSSR 位的组合控制。详见 TIMx_CCER 寄存器的描述。

在 PWM 模式（模式 1 或模式 2）下，TIMx_CNT 和 TIMx_CCRx 始终在进行比较，（依据计数器的计数方向）以确定是否符合 TIMx_CCRx≤TIMx_CNT 或者 TIMx_CNT≤TIMx_CCRx。

根据 TIMx_CR1 寄存器中 CMS 位的状态，定时器能够产生边沿对齐或中央对齐的 PWM 信号。

16.2.10.1 PWM 边沿对齐模式

- 向上计数配置

当 TIMx_CR1 寄存器中的 DIR 位为低的时候执行向上计数。可参考“16.2.2.1 向上计数模式”。

下面是一个 PWM 模式 1 的例子。当 TIMx_CNT<TIMx_CCRx 时，PWM 参考信号 OCxREF 为高，否则为低。如果 TIMx_CCRx 中的比较值大于自动重装载值（TIMx_ARR），则 OCxREF 保持为’1’。如果该比较值为 0，则 OCxREF 保持为’0’。下图为 TIMx_ARR=8 时边沿对齐的 PWM 波形实例。

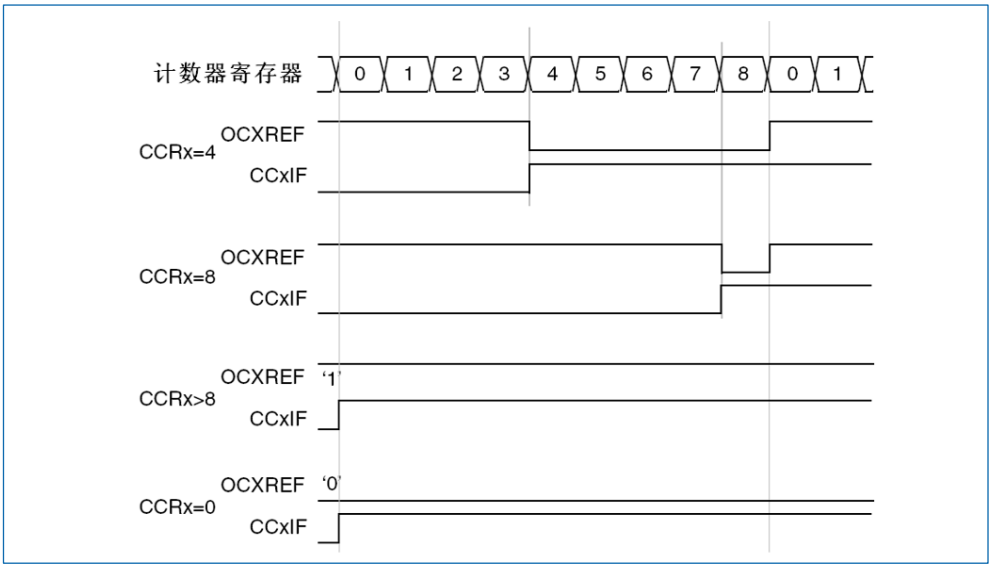


图 16-33 边沿对齐的 PWM 波形（ARR=8）

- 向下计数的配置

当 TIMx_CR1 寄存器的 DIR 位为高时执行向下计数。可参考“16.2.2.2 向下计数模式”。

在 PWM 模式 1，当 TIMx_CNT>TIMx_CCRx 时参考信号 OCxREF 为低，否则为高。如果 TIMx_CCRx 中的比较值大于 TIMx_ARR 中的自动重装载值，则 OCxREF 保持为’1’。该模式下不能产生 0%的 PWM 波形。

16.2.10.2 PWM 中央对齐模式

当 TIMx_CR1 寄存器中的 CMS 位不为’00’时为中央对齐模式（所有其他的配置对 OCxREF/OCx 信号都有相同的作用）。根据不同的 CMS 位设置，比较标志可以在计数器向上计数时被置’1’、在计数器向下计数时被置’1’、或在计数器向上和向下计数时被置’1’。TIMx_CR1 寄存器中的计数方向位（DIR）由硬件更新，不能用软件修改它。可参考“16.2.2.3 中央对齐模式（向上/向下计数）”。

图 16-34 示出了中央对齐的 PWM 波形的一个例子：

- TIMx_ARR=8
- PWM 模式 1

- TIMx_CR1 寄存器的 CMS=01，在中央对齐模式 1 下，当计数器向下计数时设置比较标志。

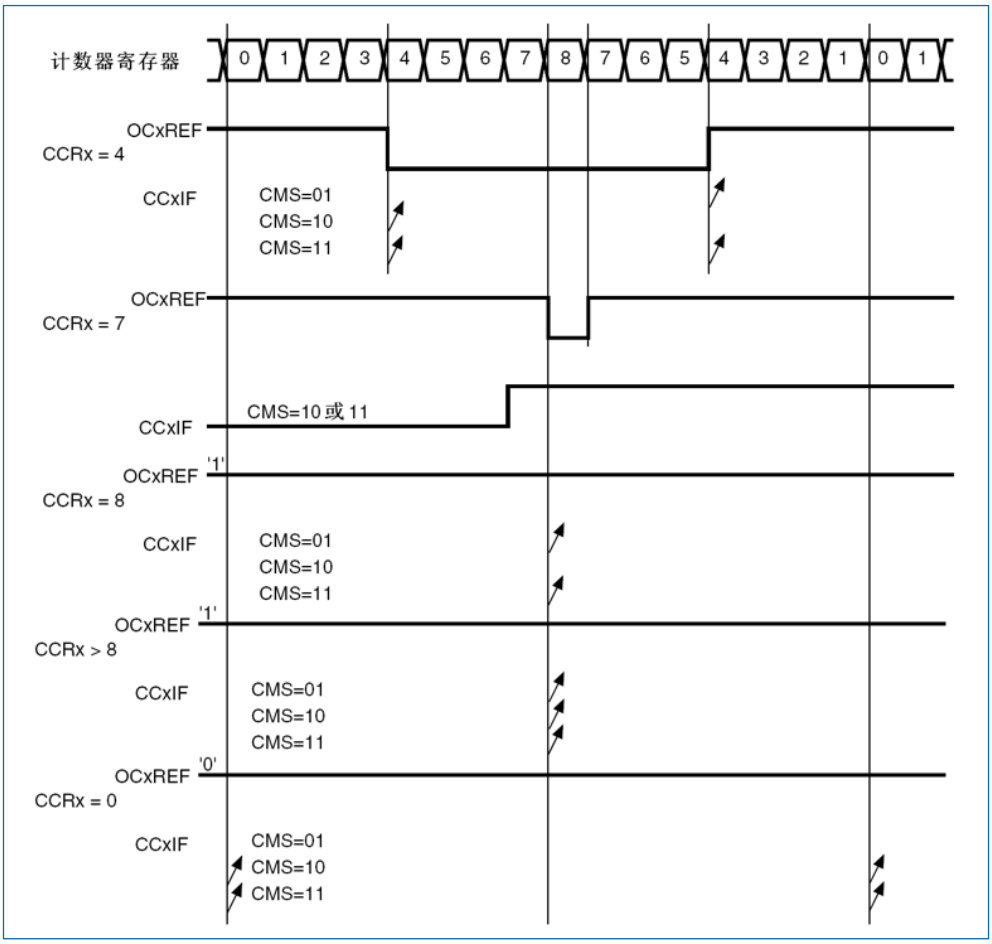


图 16-34 中央对齐的 PWM 波形（APR=8）

16.2.10.3 使用中央对齐模式的提示

- 进入中央对齐模式时，使用当前的向上/向下计数配置；这意味着计数器向上还是向下计数取决于 TIMx_CR1 寄存器中 DIR 位的当前值。此外，软件不能同时修改 DIR 和 CMS 位。
- 不推荐当运行在中央对齐模式时改写计数器，因为这会产生不可预知的结果。特别地：
  - 如果写入计数器的值大于自动重加载的值（TIMx_CNT>TIMx_ARR），则方向不会被更新。例如，如果计数器正在向上计数，它就会继续向上计数。
  - 如果将 0 或者 TIMx_ARR 的值写入计数器，方向被更新，但不产生更新事件 UEV。
- 使用中央对齐模式最保险的方法，就是在启动计数器之前产生一个软件更新（设置 TIMx_EGR 寄存器中的 UG 位），并且不要在计数进行过程中修改计数器的值。

16.2.11 互补输出和死区插入

高级控制定时器（TIM1 和 TIM8）能够输出两路互补信号，并且能够管理输出的瞬时关断和接通。

这段时间通常被称为死区，用户应该根据连接的输出器件和它们的特性（电平转换的延时、电源开关的延时等）来调整死区时间。

配置 TIMx_CCER 寄存器中的 CCxP 和 CCxNP 位，可以为每一个输出独立地选择极性（主输出 OCx 或互补输出 OCxN）。

互补信号 OCx 和 OCxN 通过下列控制位的组合进行控制：TIMx_CCER 寄存器的 CCxE 和 CCxNE 位，TIMx_BDTR 和 TIMx_CR2 寄存器中的 MOE、OISx、OISxN、OSSI 和 OSSR 位，详见表 16-3 带刹车功能的互补输出通道 OCx 和 OCxN 的控制位。特别的是，在转换到 IDLE 状态时（MOE 下降到 0）死区被激活。

同时设置 CCxE 和 CCxNE 位将插入死区，如果存在刹车电路，则还要设置 MOE 位。每一个通道都有一个 10 位的死区发生器。参考信号 OCxREF 可以产生 2 路输出 OCx 和 OCxN。

如果 OCx 和 OCxN 为高有效：

- OCx 输出信号与参考信号相同，只是它的上升沿相对于参考信号的上升沿有一个延迟。
- OCxN 输出信号与参考信号相反，只是它的上升沿相对于参考信号的下降沿有一个延迟。如果延迟大于当前有效的输出宽度（OCx 或者 OCxN），则不会产生相应的脉冲。

下列几张图显示了死区发生器的输出信号和当前参考信号 OCxREF 之间的关系。（假设 CCxP=0、CCxNP=0、MOE=1、CCxE=1 并且 CCxNE=1）

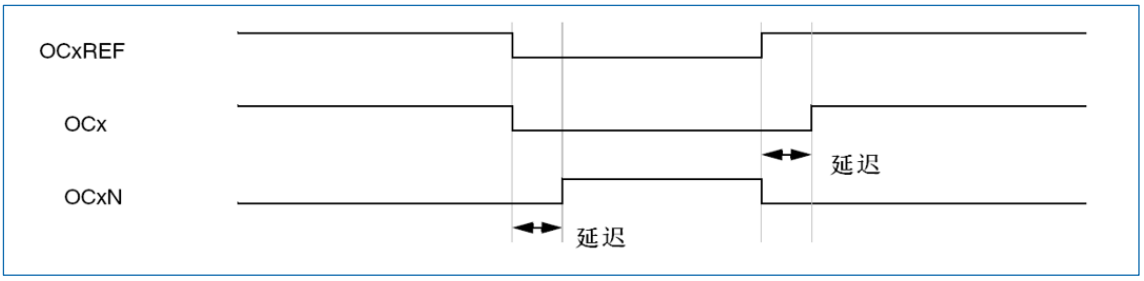


图 16-35 带死区插入的互补输出

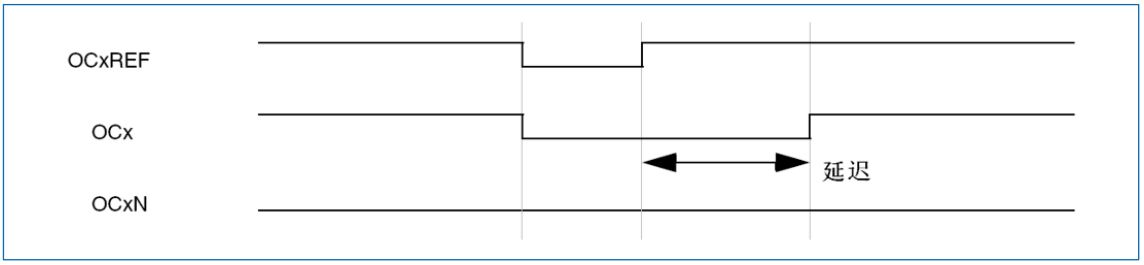


图 16-36 死区波形延迟大于负脉冲

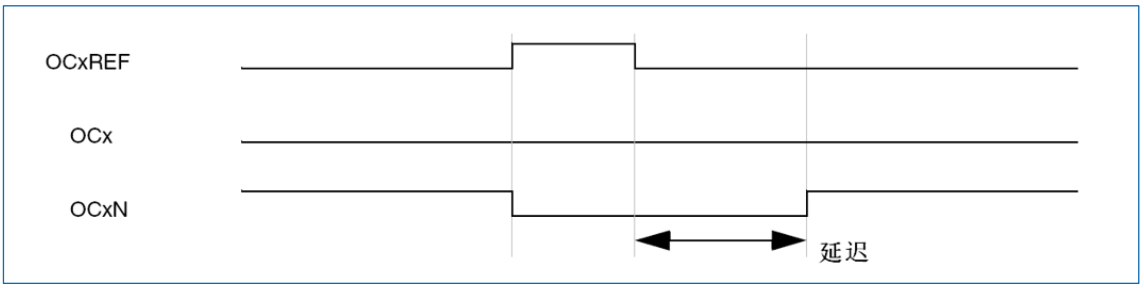


图 16-37 死区波形延迟大于正脉冲

每一个通道的死区延时都是相同的，是由 TIMx_BDTR 寄存器中的 DTG 位编程配置。详见 TIM1/TIM8 刹车和死区寄存器（TIMx_BDTR）中的延时计算。

16.2.11.1 重定向 OCxREF 到 OCx 或 OCxN

在输出模式下（强置、输出比较或 PWM），通过配置 TIMx_CCER 寄存器的 CCxE 和 CCxNE 位，OCxREF 可以被重定向到 OCx 或者 OCxN 的输出。

这个功能可以在互补输出处于无效电平时，在某个输出上送出一个特殊的波形（例如 PWM 或者静态有效电平）。另一个作用是，让两个输出同时处于无效电平，或处于有效电平和带死区的互补输出。

说明：当只使能 OCxN（CCxE=0，CCxNE=1）时，它不会反相，当 OCxREF 有效时立即变高。例如，如果 CCxNP=0，则 OCxN=OCxREF。另一方面，当 OCx 和 OCxN 都被使能时（CCxE=CCxNE=1），当 OCxREF 为高时 OCx 有效；而 OCxN 相反，当 OCxREF 低时 OCxN 变为有效。

16.2.12 使用刹车功能

当使用刹车功能时，依据相应的控制位（TIMx_BDTR 寄存器中的 MOE、OSSI 和 OSSR 位，TIMx_CR2 寄存器中的 OISx 和 OISxN 位），输出使能信号和无效电平都会被修改。但无论何时，OCx 和 OCxN 输出不能在同一时间同时处于有效电平上。详见表 16-3。

刹车源既可以是刹车输入引脚又可以是一个时钟故障事件。时钟故障事件由复位时钟控制器中的时钟安全系统产生，详见“8.2.7 时钟安全系统（CSS）”。系统复位后，刹车电路被禁止，MOE 位为低。设置 TIMx_BDTR 寄存器中的 BKE 位可以使能刹车功能，刹车输入信号的极性可以通过配置同一个寄存器中的 BKP 位选择。BKE 和 BKP 可以同时被修改。当写入 BKE 和 BKP 位时，在真正写入之前会有 1 个 APB 时钟周期的延迟，因此需要等待一个 APB 时钟周期之后，才能正确地读回写入的位。

因为 MOE 下降沿可以是异步的，在实际信号（作用在输出端）和同步控制位（在 TIMx_BDTR 寄存器中）之间设置了一个再同步电路。这个再同步电路会在异步信号和同步信号之间产生延迟。特别的，如果当它为低时写 MOE=1，则读出它之前必须先插入一个延时（空指令）才能读到正确的值。这是因为写入的是异步信号而读的是同步信号。

当发生刹车时（在刹车输入端出现选定的电平），有下述动作：

- MOE 位被异步地清除，将输出置于无效状态、空闲状态或者复位状态（由 OSSI 位选择）。这个特性在 MCU 的振荡器关闭时依然有效。
- 一旦 MOE=0，每一个输出通道输出由 TIMx_CR2 寄存器中的 OISx 位设定的电平。如果 OSSI=0，则定时器释放使能输出，否则使能输出始终为高。
- 当使用互补输出时：
  - 输出首先被置于复位状态即无效的状态（取决于极性）。这是异步操作，即使定时器没有时钟时，此功能也有效。
  - 如果定时器的时钟依然存在，死区生成器将会重新生效，在死区之后根据 OISx 和 OISxN 位指示的电平驱动输出端口。即使在这种情况下，OCx 和 OCxN 也不能被同时驱动到有效的电平。注，因为重新同步 MOE，死区时间比通常情况下长一些（大约 2 个 ck_tim 的时钟周期）。
  - 如果 OSSI=0，定时器释放使能输出，否则保持使能输出；或一旦 CCxE 与 CCxNE 之一变高时，使能输出变为高。
- 设置了刹车状态标志（TIMx_SR 寄存器中的 BIF 位）。当 TIMx_DIER 寄存器中的 BIE 位置为’1’时，则产生一个中断。
- 如果设置了 TIMx_BDTR 寄存器中的 AOE 位，在下一个更新事件 UEV 时 MOE 位被自动置位；例如，这可以用来进行整形。否则，MOE 始终保持低直到被再次置’1’；这种情况下，该特性可以被用在安全方面，你可以把刹车输入连到电源驱动的报警输出、热敏传感器或者其他安全器件上。

说明：

刹车输入为电平有效。所以，当刹车输入有效时，不能同时（自动地或者通过软件）设置 MOE。同时，状态标志 BIF 不能被清除。

刹车由 BKIN 输入产生，它的有效极性是可编程的，且由 TIMx_BDTR 寄存器中的 BKE 位开启。除了刹车输入和输出管理，刹车电路中还实现了写保护以保证应用程序的安全。它允许用户冻结几个配置参数（死区长度、OCx/OCxN 极性和被禁止的状态、OCxM 配置、刹车使能和极性）。

用户可以通过 TIMx_BDTR 寄存器中的 LOCK 位，从三级保护中选择一种，可参考 TIM1/TIM8 刹车和死区寄存器（TIMx_BDTR）。在 MCU 复位后 LOCK 位只能被修改一次。下图显示响应刹车的输出实例。

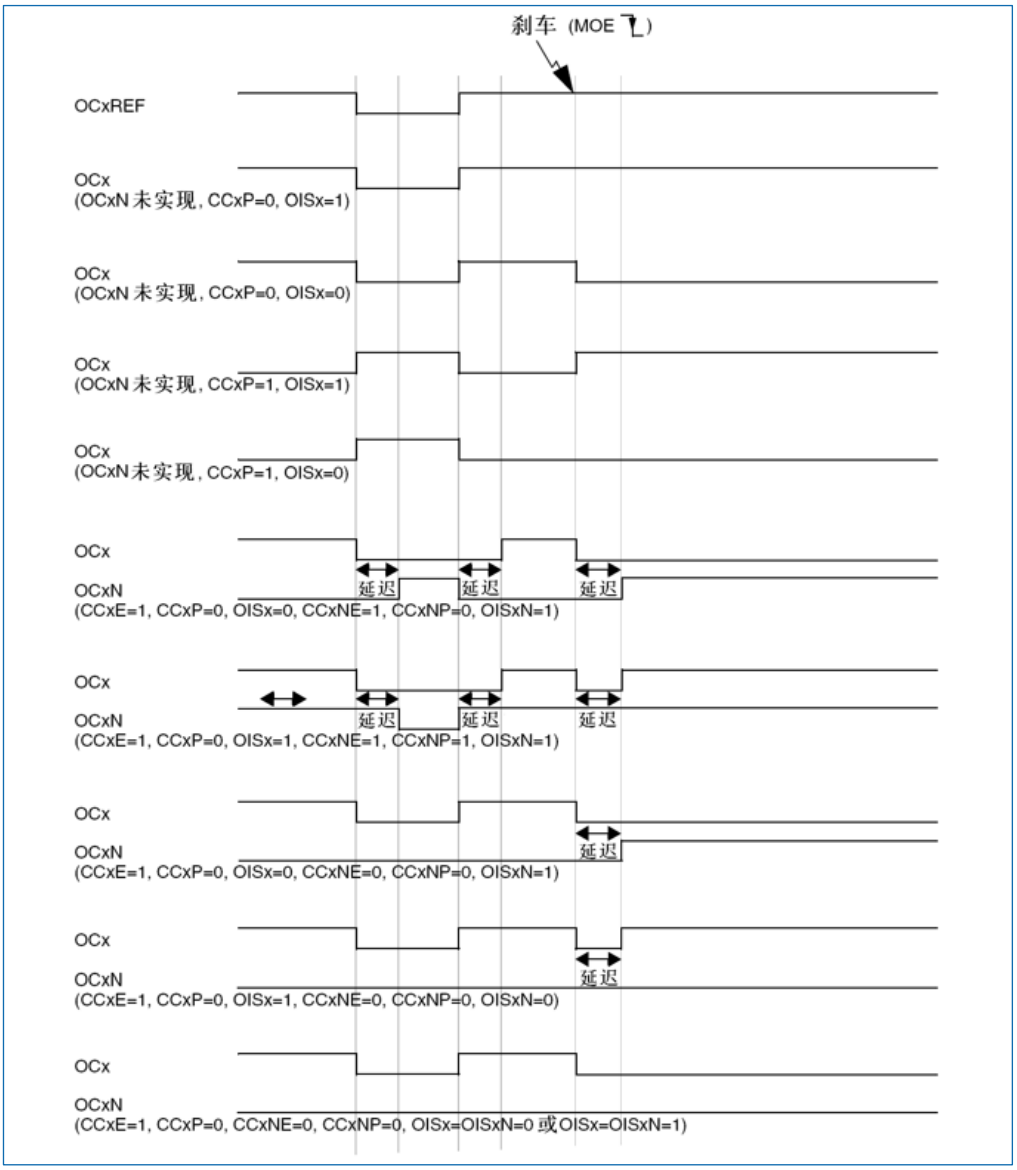


图 16-38 响应刹车的输出

16.2.13 在外部事件时清除 OCxREF 信号

对于一个给定的通道，设置 TIMx_CCMRx 寄存器中对应的 OCxCE 位为'1'，能够用 ETRF 输入端的高电平把 OCxREF 信号拉低，OCxREF 信号将保持为低直到发生下一次的更新事件 UEV。

该功能只能用于输出比较和 PWM 模式，而不能用于强置模式。例如，OCxREF 信号可以连到一个比较器的输出，用于控制电流。这时，ETR 必须配置如下：

- 外部触发预分频器必须处于关闭：TIMx_SMCR 寄存器中的 ETPS[1:0]=00。
- 必须禁止外部时钟模式 2：TIMx_SMCR 寄存器中的 ECE=0。
- 外部触发极性（ETP）和外部触发滤波器（ETF）可以根据需要配置。

下图显示了当 ETRF 输入变为高时，对应不同 OCxCE 的值，OCxREF 信号的动作。在这个例子中，定时器 TIMx 被置于 PWM 模式。

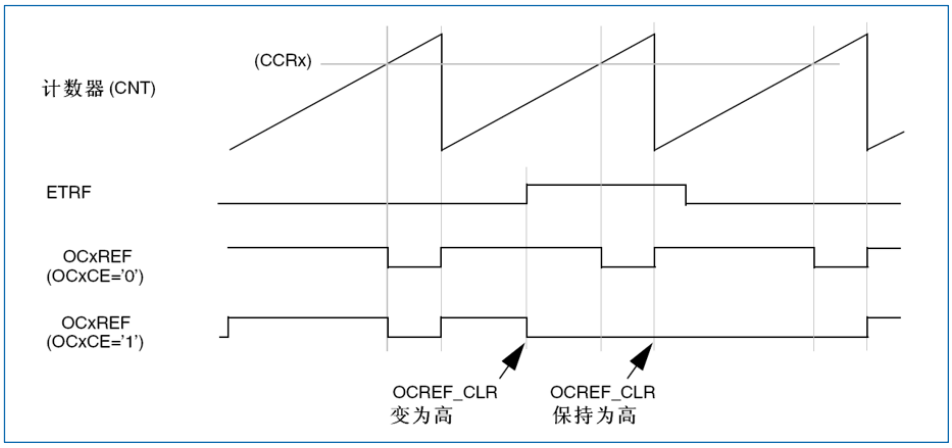


图 16-39 清除 TIMx 的 OCxREF

16.2.14 产生六步 PWM 输出

当在一个通道上需要互补输出时，预装载位有 OCxM、CCxE 和 CCxNE。在发生 COM 换相事件时，这些预装载位被传送到影子寄存器位。这样你就可以预先设置好下一步配置，并在同一个时刻同时更改所有通道的配置。COM 可以通过设置 TIMx_EGR 寄存器的 COMG 位由软件产生，或在 TRGI 上升沿由硬件产生。

当发生 COM 事件时会设置一个标志位（TIMx_SR 寄存器中的 COMIF 位），这时如果已设置了 TIMx_DIER 寄存器的 COMIE 位，则产生一个中断；如果已设置了 TIMx_DIER.COMDE 位，则产生一个 DMA 请求。

下图显示当发生 COM 事件时，三种不同配置下 OCx 和 OCxN 输出。

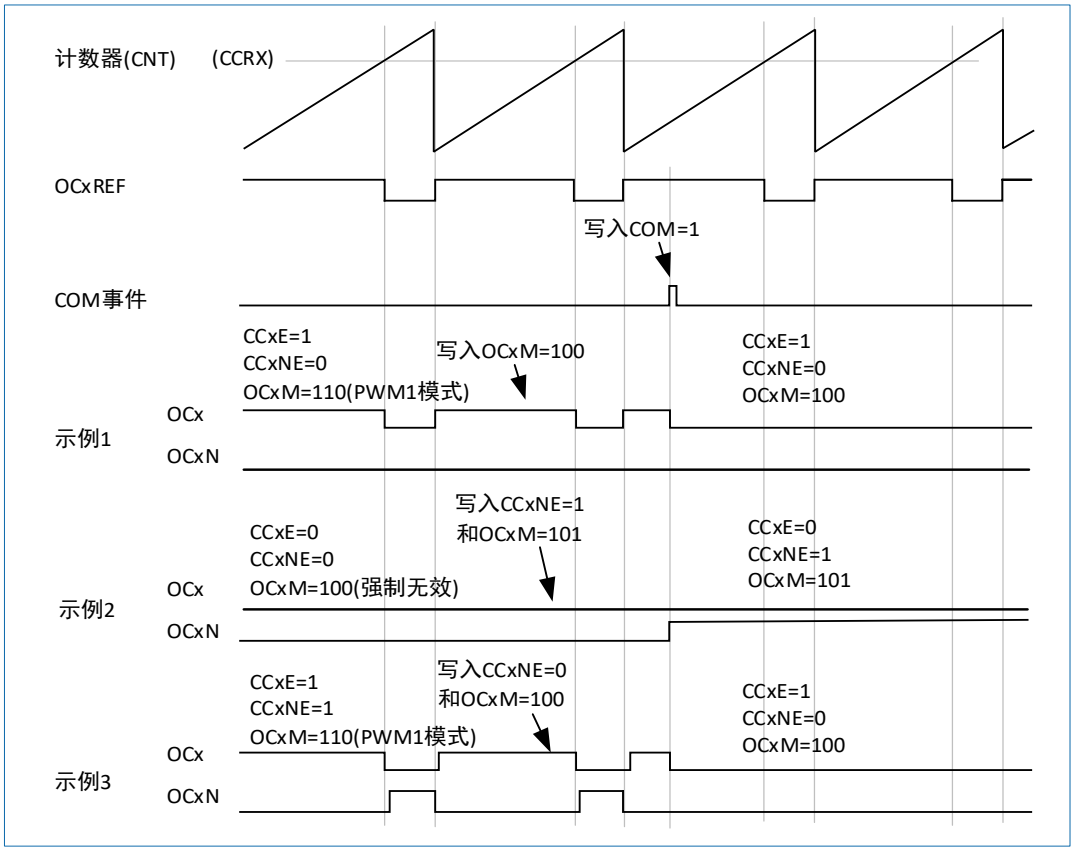


图 16-40 产生六步 PWM，使用 COM 的例子（OSSR=1）

16.2.15 单脉冲模式

单脉冲模式（OPM）是前述众多模式的一个特例。这种模式允许计数器响应一个激励，并在一个程序可控的延时之后产生一个脉宽可程序控制的脉冲。

可以通过从模式控制器启动计数器，在输出比较模式或者 PWM 模式下产生波形。设置 TIMx_CR1 寄存器中的 OPM 位将选择单脉冲模式，这样可以让计数器自动地在产生下一个更新事件 UEV 时停止。

仅当比较值与计数器的初始值不同时，才能产生一个脉冲。启动之前（当定时器正在等待触发），必须按如下配置：

- 向上计数方式：计数器 CNT<CCRx≤ARR（特别地，0<CCRx）
- 向下计数方式：计数器 CNT>CCRx

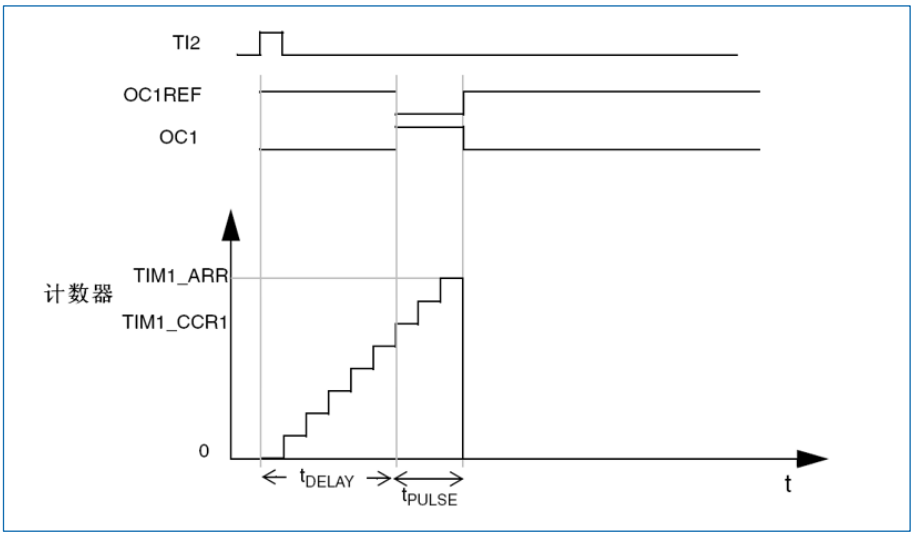


图 16-41 单脉冲模式的例子

例如，你需要在从 TI2 输入脚上检测到一个上升沿开始，延迟 t_{DELAY} 之后，在 OC1 上产生一个长度为 t_{PULSE} 的正脉冲。

假定 TI2FP2 作为触发 1：

- 置 TIMx_CCMR1 寄存器中的 CC2S=01，把 TI2FP2 映射到 TI2。
- 置 TIMx_CCER 寄存器中的 CC2P=0，使 TI2FP2 能够检测上升沿。
- 置 TIMx_SMCR 寄存器中的 TS=110，TI2FP2 作为从模式控制器的触发（TRGI）。
- 置 TIMx_SMCR 寄存器中的 SMS=110（触发模式），TI2FP2 被用来启动计数器。

OPM 的波形由写入比较寄存器的数值决定（要考虑时钟频率和计数器预分频器）。

- t_{DELAY} 由 TIMx_CCR1 寄存器中的值定义。
- t_{PULSE} 由自动装载值和比较值之间的差值定义（TIMx_ARR-TIMx_CCR1）。
- 假定当发生比较匹配时要产生从 0 到 1 的波形，当计数器达到预装载值时要产生一个从 1 到 0 的波形；首先要置 TIMx_CCMR1 寄存器的 OC1M=111，进入 PWM 模式 2；根据需要有选择地使能预装载寄存器：置 TIMx_CCMR1 中的 OC1PE=1 和 TIMx_CR1 寄存器中的 ARPE；然后在 TIMx_CCR1 寄存器中填写比较值，在 TIMx_ARR 寄存器中填写自动装载值，设置 UG 位来产生一个更新事件，然后等待在 TI2 上的一个外部触发事件。本例中，CC1P=0。

在这个例子中，TIMx_CR1 寄存器中的 DIR 和 CMS 位应该置低。

因为只需要一个脉冲，所以必须设置 TIMx_CR1 寄存器中的 OPM=1，在下一个更新事件（当计数器从自动装载值翻转到 0）时停止计数。

16.2.15.1 特殊情况：OCx 快速使能：

在单脉冲模式下，在 TIx 输入脚的边沿检测逻辑设置 CEN 位以启动计数器。然后计数器和比较值间的比较操作产生了输出的转换。但是这些操作需要一定的时钟周期，因此它限制了可得到的最小延时 t_{DELAY}。如果要以最小延时输出波形，可以设置 TIMx_CCMRx 寄存器中的 OCxFE 位；此时 OCxREF（和 OCx）直接响应激励而不再依赖比较的结果，输出的波形与比较匹配时的波形一样。OCxFE 只在通道配置为 PWM1 和 PWM2 模式时起作用。

16.2.16 编码器接口模式

选择编码器接口模式的方法是：如果计数器只在 TI2 的边沿计数，则置 TIMx_SMCR 寄存器中的 SMS=001；如果只在 TI1 边沿计数，则置 SMS=010；如果计数器同时在 TI1 和 TI2 边沿计数，则置 SMS=011。

通过设置 TIMx_CCER 寄存器中的 CC1P 和 CC2P 位，可以选择 TI1 和 TI2 极性；如果需要，还可以对输入滤波器编程。

两个输入 TI1 和 TI2 被用来作为增量编码器的接口。参看表 16-1，假定计数器已经启动（TIMx_CR1 寄存器中的 CEN=1），则计数器由每次在 TI1FP1 或 TI2FP2 上的有效跳变驱动。TI1FP1 和 TI2FP2 是 TI1 和 TI2 在通过输入滤波器和极性控制后的信号；如果没有滤波和变相，则 TI1FP1=TI1，TI2FP2=TI2。根据两个输入信号的跳变顺序，产生了计数脉冲和方向信号。依据两个输入信号的跳变顺序，计数器向上或向下计数，同时硬件对 TIMx_CR1 寄存器的 DIR 位进行相应的设置。不管计数器是依靠 TI1 计数、依靠 TI2 计数或者同时依靠 TI1 和 TI2 计数，在任一输入端（TI1 或者 TI2）的跳变都会重新计算 DIR 位。

编码器接口模式基本上相当于使用了一个带有方向选择的外部时钟。这意味着计数器只在 0 到 TIMx_ARR 寄存器的自动装载值之间连续计数（根据方向，或是 0 到 ARR 计数，或是 ARR 到 0 计数）。所以在开始计数之前必须配置 TIMx_ARR；同样，捕获器、比较器、预分频器、重复计数器、触发输出特性等仍工作如常。编码器模式和外部时钟模式 2 不兼容，因此不能同时操作。

在这个模式下，计数器依照增量编码器的速度和方向被自动的修改，因此计数器的内容始终指示着编码器的位置。计数方向与相连的传感器旋转的方向对应。下表列出了所有可能的组合，假设 TI1 和 TI2 不同时变换。

表 16-1 计数方向与编码器信号的关系

有效边沿	相对信号的电平（TI1FP1 对应 TI2，TI2FP2 对应 TI1）	有效边沿在 TI1 上时，使用 TI1FP1 信号计数		有效边沿在 TI2 上时，使用 TI2FP2 信号计数	
		TI1FP1 信号		TI2FP2 信号	
		上升	下降	上升	下降
仅在 TI1 计数	TI2FP2 为高	向下计数	向上计数	不计数	不计数
	TI2FP2 为低	向上计数	向下计数	不计数	不计数
仅在 TI2 计数	TI1FP1 为高	不计数	不计数	向上计数	向下计数
	TI1FP1 为低	不计数	不计数	向下计数	向上计数
在 TI1（和 TI2）上计数	TI2FP2（和 TI1FP1）为高	向下计数	向上计数	向上计数	向下计数
	TI2FP2（和 TI1FP1）为低	向上计数	向下计数	向下计数	向上计数

一个外部的增量编码器可以直接与 MCU 连接而不需要外部接口逻辑。但是，一般会使用比较器将编码器的差动输出转换到数字信号，这大大增加了抗噪声干扰能力。编码器输出的第三个信号表示机

械零点，可以把它连接到一个外部中断输入并触发一个计数器复位。

下图是一个计数器操作的实例，显示了计数信号的产生和方向控制。它还显示了当选择了双边沿时，输入抖动是如何被抑制的；抖动可能会在传感器的位置靠近一个转换点时产生。在这个例子中，我们假定配置如下：

- CC1S=’01’（TIMx_CCMR1 寄存器，IC1FP1 映射到 TI1）
- CC2S=’01’（TIMx_CCMR2 寄存器，IC2FP2 映射到 TI2）
- CC1P=’0’（TIMx_CCER 寄存器，IC1FP1 不反相，IC1FP1=TI1）
- CC2P=’0’（TIMx_CCER 寄存器，IC2FP2 不反相，IC2FP2=TI2）
- SMS=’011’（TIMx_SMCR 寄存器，所有的输入均在上升沿和下降沿有效）
- CEN=’1’（TIMx_CR1 寄存器，计数器使能）

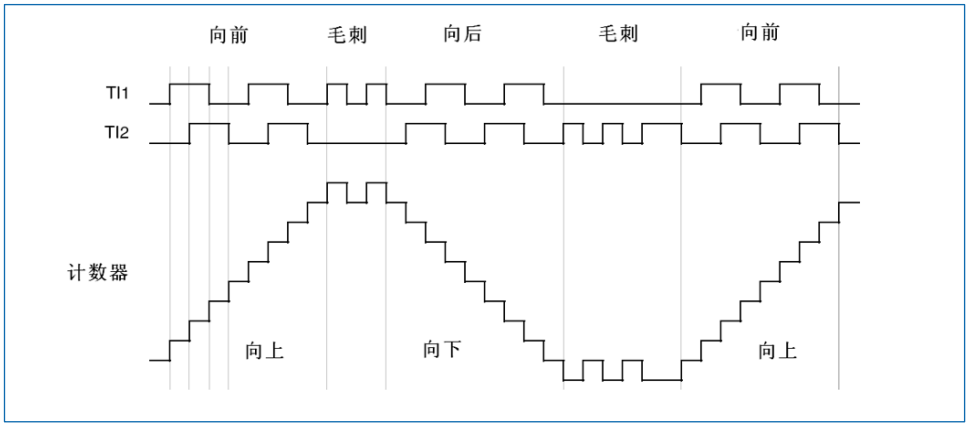


图 16-42 编码器模式下的计数器操作实例

下图为当 IC1FP1 极性反相时计数器的操作实例（CC1P=’1’，其他配置与上例相同）。

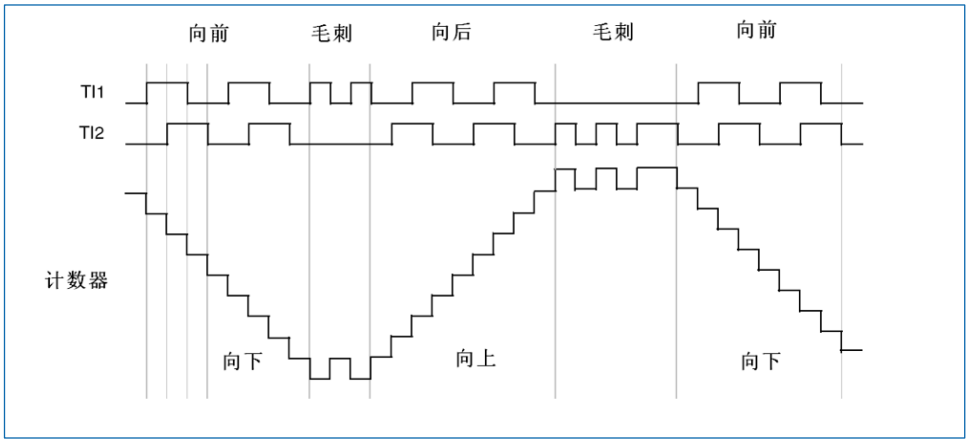


图 16-43 IC1FP1 反相的编码器接口模式实例

当定时器配置成编码器接口模式时，提供传感器当前位置的信息。使用第二个配置在捕获模式的定时器，可以测量两个编码器事件的间隔，获得动态的信息（速度，加速度，减速度）。指示机械零点的编码器输出可被用做此目的。根据两个事件间的间隔，可以按照固定的时间读出计数器。如果可能的话，你可以把计数器的值锁存到第三个输入捕获寄存器（捕获信号必须是周期的并且可以由另一个定时器产生）；也可以通过一个由实时时钟产生的 DMA 请求来读取它的值。

16.2.17 定时器输入异或功能

TIMx_CR2 寄存器中的 TI1S 位，允许通道 1 的输入滤波器连接到一个异或门的输出端，异或门的 3 个输入端为 TIMx_CH1、TIMx_CH2 和 TIMx_CH3。

异或输出能够被用于所有定时器的输入功能，如触发或输入捕获。与霍尔传感器的接口给出了此特性用于连接霍尔传感器的例子。

16.2.18 与霍尔传感器的接口

使用高级控制定时器（TIM1 或 TIM8）产生 PWM 信号驱动马达时，可以用另一个通用 TIMx（TIM2、TIM3、TIM4 或 TIM5）定时器作为“接口定时器”来连接霍尔传感器，见下图，3 个定时器输入脚（CC1、CC2、CC3）通过一个异或门连接到 TI1 输入通道（通过设置 TIMx_CR2 寄存器中的 TI1S 位来选择），“接口定时器”捕获这个信号。

从模式控制器被配置于复位模式，输入是 TI1F_ED。每当 3 个输入之一变化时，计数器重新从’0’开始计数。这样产生一个由霍尔输入端的任何变化而触发的时间基准。

“接口定时器”上的捕获/比较通道 1 配置为捕获模式，捕获信号为 TRC（见图 16-27）。捕获值反映了两个输入变化间的时间延迟，给出了马达速度的信息。

“接口定时器”可以在输出模式产生一个脉冲，这个脉冲可以（通过触发一个 COM 事件）用于改变高级定时器 TIM1 或 TIM8 各个通道的属性，而高级控制定时器产生 PWM 信号驱动马达。因此“接口定时器”通道必须编程为在一个指定的延时（输出比较或 PWM 模式）之后产生一个正脉冲，这个脉冲通过 TRGO 输出被送到高级控制定时器 TIM1 或 TIM8。

举例：霍尔输入连接到 TIMx 定时器，要求每次任一霍尔输入上发生变化之后的一个指定的时刻，改变高级控制定时器 TIM1/8 的 PWM 配置。

- 置 TIMx_CR2 寄存器的 TI1S 位为’1’，配置三个定时器输入逻辑或到 TI1 输入，
- 时基编程：置 TIMx_ARR 为其最大值（计数器必须通过 TI1 的变化清零）。设置预分频器得到一个最大的计数器周期，它长于传感器上的两次变化的时间间隔。
- 设置通道 1 为捕获模式（选中 TRC）：置 TIMx_CCMR1 寄存器中 CC1S=01，如果需要，还可以设置数字滤波器。
- 设置通道 2 为 PWM2 模式，并具有要求的延时：置 TIMx_CCMR1 寄存器中的 OC2M=111 和 CC2S=00。
- 选择 OC2REF 作为 TRGO 上的触发输出：置 TIMx_CR2 寄存器中的 MMS=101。

在高级控制寄存器 TIM1/8 中，正确的 ITR 输入必须是触发器输入，定时器被编程为产生 PWM 信号，捕获/比较控制信号为预装载的（TIMx_CR2 寄存器中 CCPC=1），同时触发输入控制 COM 事件（TIMx_CR2 寄存器中 CCUS=1）。在一次 COM 事件后，写入下一步的 PWM 控制位（CCxE、OCxM），这可以在处理 OC2REF 上升沿的中断子程序里实现。

下图显示了这个实例。

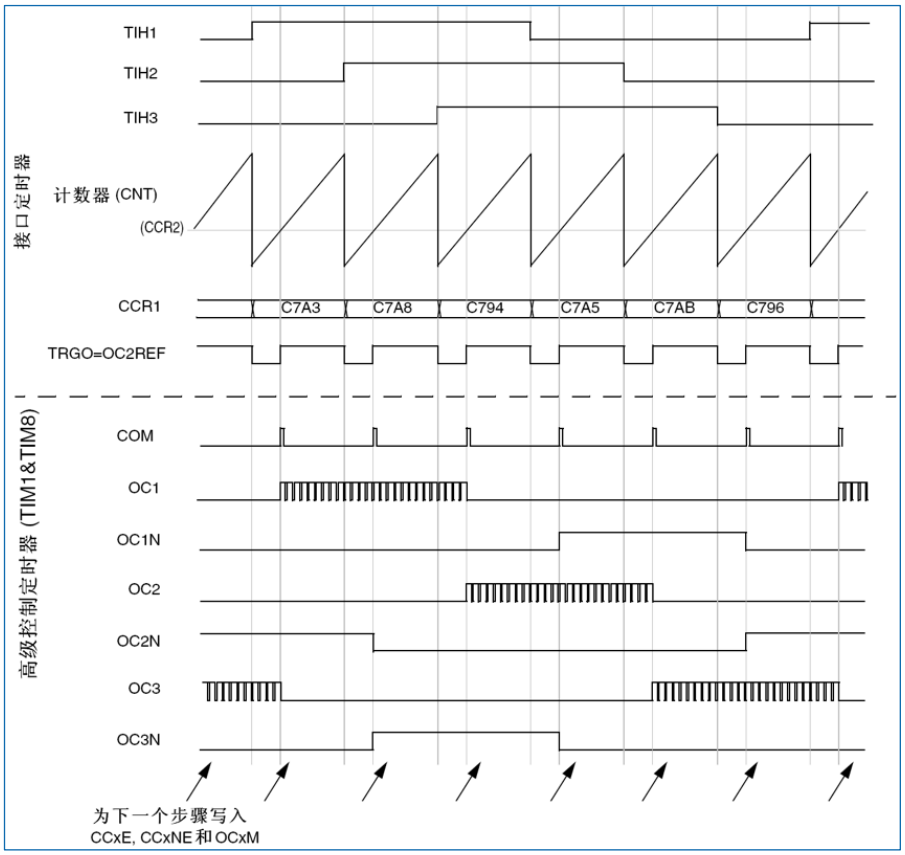


图 16-44 霍尔传感器接口的实例

16.2.19 TIM1/8 定时器和外部触发的同步

TIM1/8 定时器能够在多种模式下和一个外部的触发同步：复位模式、门控模式和触发模式。

16.2.19.1 从模式：复位模式

在发生一个触发输入事件时，计数器和它的预分频器能够重新被初始化；同时，如果 TIMx_CR1 寄存器的 URS 位为低，还产生一个更新事件 UEV；然后所有的预装载寄存器（TIMx_ARR，TIMx_CCRx）都被更新了。

在以下的例子中，TI1 输入端的上升沿导致向上计数器被清零：

- 配置通道 1 以检测 TI1 的上升沿。配置输入滤波器的带宽（在本例中，不需要任何滤波器，因此保持 IC1F=0000）。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位只选择输入捕获源，即 TIMx_CCMR1 寄存器中 CC1S=01。置 TIMx_CCER 寄存器中 CC1P=0 以确定极性（只检测上升沿）。
- 置 TIMx_SMCR 寄存器中 SMS=100，配置定时器为复位模式；置 TIMx_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。
- 置 TIMx_CR1 寄存器中 CEN=1，启动计数器。

计数器开始依据内部时钟计数，然后正常运转直到 TI1 出现一个上升沿；此时，计数器被清零然后从 0 重新开始计数。同时，触发标志（TIMx_SR 寄存器中的 TIF 位）被设置，根据 TIMx_DIER 寄存器中 TIE（中断使能）位和 TDE（DMA 使能）位的设置，产生一个中断请求或一个 DMA 请求。

下图显示当自动重装载寄存器 TIMx_ARR=0x36 时的动作。在 TI1 上升沿和计数器的实际复位之间的延时取决于 TI1 输入端的重同步电路。

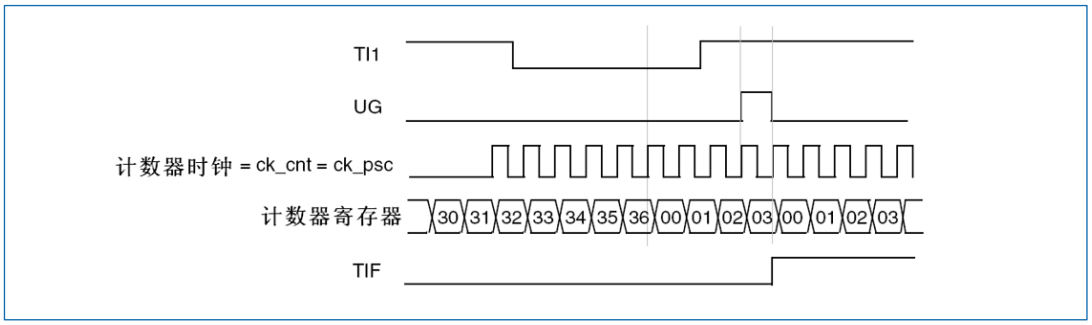


图 16-45 复位模式下的控制电路

16.2.19.2 从模式：门控模式

按照选中的输入端电平使能计数器。

在如下的例子中，计数器只在 TI1 为低时向上计数：

- 配置通道 1 以检测 TI1 上的低电平。配置输入滤波器带宽（本例中，不需要滤波，所以保持 IC1F=0000）。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位用于选择输入捕获源，置 TIMx_CCMR1 寄存器中 CC1S=01。置 TIMx_CCER 寄存器中 CC1P=1 以确定极性（只检测低电平）。
- 置 TIMx_SMCR 寄存器中 SMS=101，配置定时器为门控模式；置 TIMx_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。
- 置 TIMx_CR1 寄存器中 CEN=1，启动计数器。在门控模式下，如果 CEN=0，则计数器不能启动，不论触发输入电平如何。

只要 TI1 为低，计数器开始依据内部时钟计数，一旦 TI1 变高则停止计数。当计数器开始或停止时都设置 TIMx_SR 中的 TIF 标志。

TI1 上升沿和计数器实际停止之间的延时取决于 TI1 输入端的重同步电路。

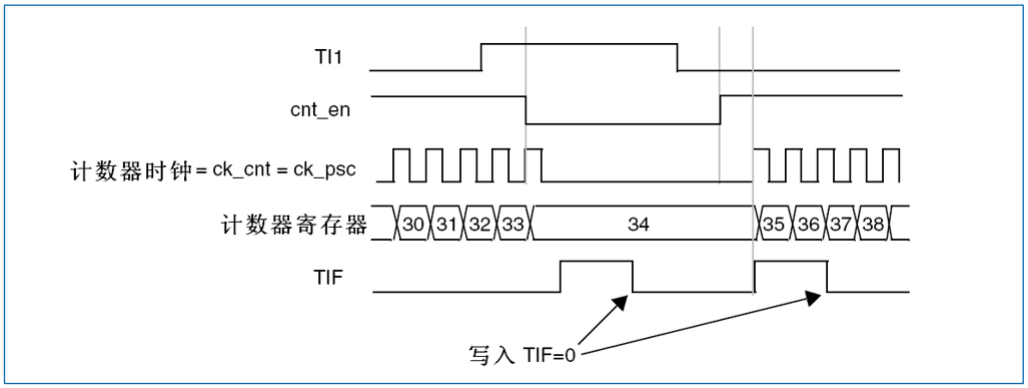


图 16-46 门控模式下的控制电路

16.2.19.3 从模式：触发模式

输入端上选中的事件使能计数器。

在下面的例子中，计数器在 TI2 输入的上升沿开始向上计数：

- 配置通道 2 检测 TI2 的上升沿。配置输入滤波器带宽（本例中，不需要任何滤波器，保持 IC2F=0000）。触发操作中不使用捕获预分频器，不需要配置。CC2S 位只用于选择输入捕获源，置 TIMx_CCMR1 寄存器中 CC2S=01。置 TIMx_CCER 寄存器中 CC2P=1 以确定极性（只检测低电平）。
- 置 TIMx_SMCR 寄存器中 SMS=110，配置定时器为触发模式；置 TIMx_SMCR 寄存器中 TS=110，选择 TI2 作为输入源。

当 TI2 出现一个上升沿时，计数器开始在内部时钟驱动下计数，同时设置 TIF 标志。

TI2 上升沿和计数器启动计数之间的延时，取决于 TI2 输入端的重同步电路。

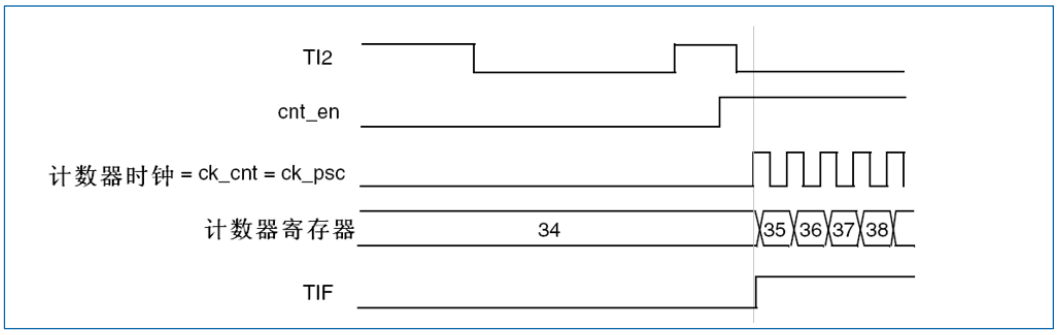


图 16-47 触发器模式下的控制电路

16.2.19.4 从模式：外部时钟模式 2＋触发模式

外部时钟模式 2 可以与另一种从模式（外部时钟模式 1 和编码器模式除外）一起使用。这时，ETR 信号被用作外部时钟的输入，在复位模式、门控模式或触发模式可以选择另一个输入作为触发输入。不建议使用 TIMx_SMCR 寄存器的 TS 位选择 ETR 作为 TRGI。

在下面的例子中，一旦在 TI1 上出现一个上升沿，计数器即在 ETR 的每一个上升沿向上计数一次：

1. 通过 TIMx_SMCR 寄存器配置外部触发输入电路：
   - ETF=0000：没有滤波
   - ETPS=00：不用预分频器
   - ETP=0：检测 ETR 的上升沿，置 ECE=1 使能外部时钟模式 2。
2. 按如下配置通道 1，检测 TI 的上升沿：
   - IC1F=0000：没有滤波。
   - 触发操作中不使用捕获预分频器，不需要配置。
   - 置 TIMx_CCMR1 寄存器中 CC1S=01，选择输入捕获源。
   - 置 TIMx_CCER 寄存器中 CC1P=0 以确定极性（只检测上升沿）。
3. 置 TIMx_SMCR 寄存器中 SMS=110，配置定时器为触发模式。置 TIMx_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。

当 TI1 上出现一个上升沿时，TIF 标志被设置，计数器开始在 ETR 的上升沿计数。

ETR 信号的上升沿和计数器实际复位之间的延时，取决于 ETRP 输入端的重同步电路。

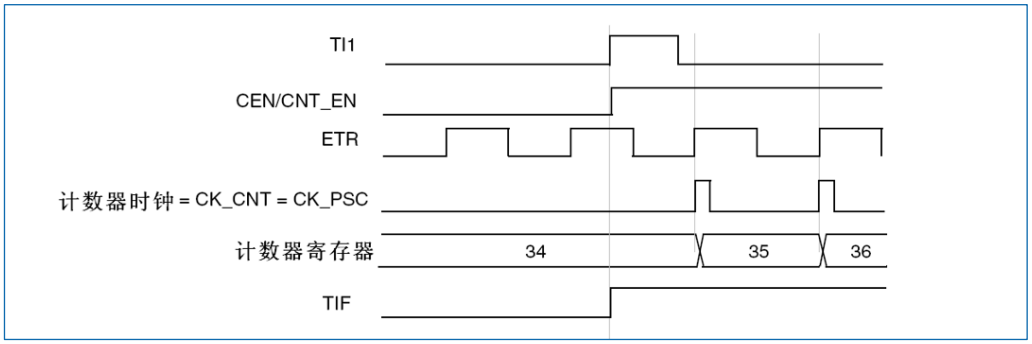


图 16-48 外部时钟模式 2+触发模式下的控制电路

16.2.20 定时器同步

所有 TIM 定时器在内部相连，用于定时器同步或链接，参见章节“17.2.15 定时器同步”。

16.2.21 调试模式

当 MCU 进入调试模式时（Cortex®-M3 内核停止），根据 DBG 模块中 DBG_TIMx_STOP 的设置，TIMx 计数器可以或者继续正常操作，或者停止。详细信息，请参考章节：“36.15.2 支持定时器、看门狗、CAN 和 I2C 的调试”。

16.3 TIM1/8 寄存器

基地址：（TIM1, TIM8）=（0x4001 2C00, 0x4001 3400）

空间大小：（TIM1, TIM8）=（0x400, 0x400）

TIM1 和 TIM8 寄存器必须半字（16 位）或字（32 位）的方式进行访问。

16.3.1 TIM1/TIM8 控制寄存器 1（TIMx_CR1）（x=1,8）

偏移地址：0x00

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETR_CLR_SEL	BRK_SEL	Res				CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
rw	rw					rw		rw	rw		rw	rw	rw	rw	rw

位 15	ETR_CLR_SEL：PWM 输出清除控制位（PWM output clear bit） • 0：选择用外部引脚清除 PWM 输出 • 1：选择用比较器 COMP4 清除 PWM 输出
位 14	BRK_SEL：PWM 刹车控制位（PWM break control bit） • 0：选择用外部引脚来实现 PWM 刹车 • 1：选择用比较器 COMP1/2/3 来实现 PWM 刹车
位 13:10	Res：保留 必须保持复位值。
位 9:8	CKD[1:0]：时钟分频因子（Clock division factor） 该位域定义了定时器时钟频率（CK_INT）与死区发生器和数字滤波器（ETR, TIx）所使用的采样时钟（t_{DTS}）之间的分频比例。 • 00：$t_{DTS}=t_{CK\_INT}$ • 01：$t_{DTS}=2 \times t_{CK\_INT}$ • 10：$t_{DTS}=4 \times t_{CK\_INT}$ • 11：保留，不使用该配置
位 7	ARPE：自动重装载预装载使能位（Auto-reload preload enable） • 0：TIMx_ARR 寄存器没有缓冲 • 1：TIMx_ARR 寄存器被装入缓冲器
位 6:5	CMS[1:0]：选择中央对齐模式（Center-aligned mode selection） • 00：边沿对齐模式 计数器依据方向位（DIR）向上或向下计数。 • 01：中央对齐模式 1 计数器交替地向上和向下计数。配置为输出的通道（TIMx_CCMRx 寄存器中 CCxS=00）的输出比较中断标志位，只在计数器向下计数时被设置。 • 10：中央对齐模式 2

	计数器交替地向上和向下计数。配置为输出的通道（TIMx_CCMRx 寄存器中 CCxS=00）的输出比较中断标志位，只在计数器向上计数时被设置。 • 11：中央对齐模式 3 计数器交替地向上和向下计数。配置为输出的通道（TIMx_CCMRx 寄存器中 CCxS=00）的输出比较中断标志位，在计数器向上和向下计数时均被设置。 *注意：在计数器开启时（CEN=1），不允许从边沿对齐模式转换到中央对齐模式。*
位 4	DIR：计数方向（Direction） • 0：计数器向上计数 • 1：计数器向下计数 *说明：当计数器配置为中央对齐模式或编码器模式时，该位为只读。*
位 3	OPM：单脉冲模式（One pulse mode） • 0：在发生更新事件时，计数器不停止。 • 1：在发生下一次更新事件（清除 CEN 位）时，计数器停止。
位 2	URS：更新请求源（Update request source） 软件通过该位选择 UEV 事件的源。 • 0：如果使能了更新中断或 DMA 请求，则下述任一事件产生更新中断或 DMA 请求： o 计数器溢出/下溢 o 设置 UG 位 o 从模式控制器产生的更新 • 1：如果使能了更新中断或 DMA 请求，则只有计数器溢出/下溢才产生更新中断或 DMA 请求。
位 1	UDIS：禁止更新（Update disable） 软件通过该位允许/禁止 UEV 事件的产生。 • 0：允许 UEV 更新（UEV）事件由下述任一事件产生： o 计数器溢出/下溢 o 设置 UG 位 o 从模式控制器产生的更新 • 1：禁止 UEV 不产生更新事件，影子寄存器（ARR、PSC、CCRx）保持它们的值。如果设置了 UG 位或从模式控制器发出了一个硬件复位，则计数器和预分频器被重新初始化。 *说明：具有缓存的寄存器被装入它们的预装载值，即更新影子寄存器。*
位 0	CEN：使能计数器（Counter enable） • 0：禁止计数器 • 1：使能计数器 *说明：在软件设置了 CEN 位后，外部时钟、门控模式和编码器模式才能工作。触发模式可以自动地通过硬件设置 CEN 位。*

16.3.2 TIM1/TIM8 控制寄存器 2（TIMx_CR2）（x=1,8）

偏移地址：0x04

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res	CCPC
	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw		rw

位 15	Res：保留 必须保持复位值。
位 14	OIS4：输出空闲状态 4（OC4 输出）（OC4 output idle state） 参见 OIS1 位。
位 13	OIS3N：输出空闲状态 3（OC3N 输出）（OC3N output idle state） 参见 OIS1N 位。
位 12	OIS3：输出空闲状态 3（OC3 输出）（OC3 Output Idle state） 参见 OIS1 位。
位 11	OIS2N：输出空闲状态 2（OC2N 输出）（OC2N output idle state） 参见 OIS1N 位。
位 10	OIS2：输出空闲状态 2（OC2 输出）（OC2 output idle state） 参见 OIS1 位。
位 9	OIS1N：输出空闲状态 1（OC1N 输出）（OC1N output idle state） • 0：当 MOE=0 时，死区后 OC1N=0。 • 1：当 MOE=0 时，死区后 OC1N=1。 说明：已经设置了 LOCK（TIMx_BKR 寄存器）级别 1、2 或 3 后，该位不能被修改。
位 8	OIS1：输出空闲状态 1（OC1 输出）（OC1 output idle state ） • 0：当 MOE=0 时，如果实现了 OC1N，则死区后 OC1=0。 • 1：当 MOE=0 时，如果实现了 OC1N，则死区后 OC1=1。 说明：已经设置了 LOCK（TIMx_BKR 寄存器）级别 1、2 或 3 后，该位不能被修改。
位 7	TI1S：TI1 选择（TI1 selection） • 0：TIMx_CH1 引脚连到 TI1 输入。 • 1：TIMx_CH1、TIMx_CH2 和 TIMx_CH3 引脚经异或后连到 TI1 输入。
位 6:4	MMS[2:0]：主模式选择（Master mode selection） 这 3 位用于选择在主模式下送到从定时器的同步信息（TRGO）。可能的组合如下： • 000：复位 TIMx_EGR 寄存器的 UG 位被用于作为触发输出（TRGO）。如果是触发输入产生的复位（从模式控制器处于复位模式），则 TRGO 上的信号相对实际的复位会有一个延迟。 • 001：使能 计数器使能信号 CNT_EN 被用于作为触发输出（TRGO）。有时需要在同一时间启动多个定时器或控制在一段时间内使能从定时器。计数器使能信号是通过 CEN 控制位和门控模式下的触发输入信号的逻辑或产生。当计数器使能信号受控于触发输入时，TRGO 上会有一个延迟，除非选择了主/从模式（见 TIMx_SMCR 寄存器中 MSM 位的描述）。 • 010：更新 更新事件被选为触发输入（TRGO）。例如，一个主定时器的时钟可以被用作一个从定时器的预分频器。 • 011：比较脉冲 在发生一次捕获或一次比较成功时，当要设置 CC1IF 标志时（即使它已经为高），触发输出送出一个正脉冲（TRGO）。 • 100：比较

	OC1REF 信号被用于作为触发输出（TRGO）。 • 101：比较 OC2REF 信号被用于作为触发输出（TRGO）。 • 110：比较 OC3REF 信号被用于作为触发输出（TRGO）。 • 111：比较 OC4REF 信号被用于作为触发输出（TRGO）。
位 3	CCDS：捕获/比较的 DMA 选择（Capture/Compare DMA selection） • 0：当发生 CCx 事件时，送出 CCx 的 DMA 请求。 • 1：当发生更新事件时，送出 CCx 的 DMA 请求。
位 2	CCUS：捕获/比较控制更新选择（Capture/Compare control update selection） • 0：如果捕获/比较控制位是预装载的（CCPC=1），只能通过设置 COM 位更新它们。 • 1：如果捕获/比较控制位是预装载的（CCPC=1），可以通过设置 COM 位或 TRGI 上的一个上升沿更新它们。 *说明：该位只对具有互补输出的通道起作用。*
位 1	Res：保留 必须保持复位值。
位 0	CCPC：捕获/比较预装载控制位（Capture/Compare preloaded control） • 0：CCxE，CCxNE 和 OCxM 位不是预装载的。 • 1：CCxE，CCxNE 和 OCxM 位是预装载的；设置该位后，它们只在设置了 COM 位后被更新。 *说明：该位只对具有互补输出的通道起作用。*

16.3.3 TIM1/TIM8 从模式控制寄存器（TIMx_SMCR）（x=1,8）

偏移地址：0x08

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			Res	SMS[2:0]		
rw	rw	rw		rw				rw	rw				rw		

位 15	ETP：外部触发极性（External trigger polarity） 该位选择是用 ETR 还是 ETR 的反相作为触发操作。 • 0：ETR 不反相，高电平或上升沿有效 • 1：ETR 被反相，低电平或下降沿有效
位 14	ECE：外部时钟使能位（External clock enable） 该位启用外部时钟模式 2。 • 0：禁止外部时钟模式 2。 • 1：使能外部时钟模式 2。计数器由 ETRF 信号上的任意有效边沿驱动。 *说明：* • 设置 ECE 位与选择外部时钟模式 1 并将 TRGI 连到 ETRF（SMS=111 和 TS=111）具有相同功效。 • 下述从模式可以与外部时钟模式 2 同时使用：复位模式，门控模式和触发模式；但是，这时 TRGI 不能连到 ETRF（TS 位不能是’111’）。 • 外部时钟模式 1 和外部时钟模式 2 同时被使能时，外部时钟的输入是 ETRF。

位 13:12	ETPS[1:0]：外部触发预分频（External trigger prescaler） 外部触发信号 ETRP 的频率必须满足最多为 TIMxCLK 频率的 1/4。当输入较快的外部时钟时，可以使用预分频降低 ETRP 的频率。 • 00：关闭预分频 • 01：ETRP 频率除以 2 • 10：ETRP 频率除以 4 • 11：ETRP 频率除以 8
位 11:8	ETF[3:0]：外部触发滤波（External trigger filter） 该位域定义了对 ETRP 信号采样的频率和对 ETRP 数字滤波的带宽。实际上，数字滤波器是一个事件计数器，它记录到 N 个事件后会产生一个输出的跳变。 • 0000：无滤波器，以 f_{DTS} 采样 • 0001：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N=2 • 0010：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N=4 • 0011：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N=8 • 0100：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N=6 • 0101：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N=8 • 0110：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N=6 • 0111：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N=8 • 1000：采样频率 $f_{SAMPLING}=f_{DTS}/8$，N=6 • 1001：采样频率 $f_{SAMPLING}=f_{DTS}/8$，N=8 • 1010：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N=5 • 1011：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N=6 • 1100：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N=8 • 1101：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N=5 • 1110：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N=6 • 1111：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N=8
位 7	MSM：主/从模式（Master/slave mode） • 0：无作用 • 1：触发输入（TRGI）上的事件被延迟了，以允许在当前定时器（通过 TRGO）与它的从定时器间的完美同步。这对要求把几个定时器同步到一个单一的外部事件时是非常有用的。
位 6:4	TS[2:0]：触发选择（Trigger selection） 这 3 位选择用于同步计数器的触发输入。 • 000：内部触发 0（ITR0） • 001：内部触发 1（ITR1） • 010：内部触发 2（ITR2） • 011：内部触发 3（ITR3） • 100：TI1 的边沿检测器（TI1F_ED） • 101：滤波后的定时器输入 1（TI1FP1） • 110：滤波后的定时器输入 2（TI2FP2） • 111：外部触发输入（ETRF） 更多有关 ITRx 的细节，参见表 16-2。 *说明：该位域只能在未用到（如 SMS=000）时被改变，以避免在改变时产生错误的边沿检测。*

位 3	Res：保留 必须保持复位值。
位 2:0	SMS[2:0]：从模式选择（Slave mode selection） 当选择了外部信号，触发信号（TRGI）的有效边沿与选中的外部输入极性相关（见输入控制寄存器和控制寄存器的说明） • 000：关闭从模式 如果 CEN=1，则预分频器直接由内部时钟驱动。 • 001：编码器模式 1 根据 TI1FP1 的电平，计数器在 TI2FP2 的边沿向上/下计数。 • 010：编码器模式 2 根据 TI2FP2 的电平，计数器在 TI1FP1 的边沿向上/下计数。 • 011：编码器模式 3 根据 TI1FP1（和 TI2FP2）的电平，计数器在 TI2FP2（和 TI1FP1）的边沿向上/下计数。 • 100：复位模式 选中的触发输入（TRGI）的上升沿重新初始化计数器，并且产生一个更新寄存器的信号。 • 101：门控模式 当触发输入（TRGI）为高时，计数器的时钟开启。一旦触发输入变为低，则计数器停止（但不复位）。计数器的启动和停止都是受控的。 • 110：触发模式 计数器在触发输入 TRGI 的上升沿启动（但不复位），只有计数器的启动是受控的。 • 111：外部时钟模式 1 选中的触发输入（TRGI）的上升沿驱动计数器。 *说明：如果 TI1F_EN 被选为触发输入（TS=100）时，不要使用门控模式。这是因为，TI1F_ED 在每次 TI1F 变化时输出一个脉冲，然而门控模式是要检查触发输入的电平。*

表 16-2 主从定时器内部触发连接

从定时器	ITR0（TS=000）	ITR1（TS=001）	ITR2（TS=010）	ITR3（TS=011）
TIM1	TIM5	TIM2	TIM3	TIM4
TIM8	TIM1	TIM2	TIM4	TIM5

16.3.4 TIM1/TIM8DMA/中断使能寄存器（TIMx_DIER）（x=1,8）

偏移地址：0x0C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15	Res：保留 必须保持复位值。
位 14	TDE：允许触发 DMA 请求（Trigger DMA request enable） • 0：禁止触发 DMA 请求 • 1：允许触发 DMA 请求

位 13	COMDE：允许 COM 的 DMA 请求（COM DMA request enable） • 0：禁止 COM 的 DMA 请求 • 1：允许 COM 的 DMA 请求
位 12	CC4DE：允许捕获/比较 4 的 DMA 请求（Capture/Compare 4 DMA request enable） • 0：禁止捕获/比较 4 的 DMA 请求 • 1：允许捕获/比较 4 的 DMA 请求
位 11	CC3DE：允许捕获/比较 3 的 DMA 请求（Capture/Compare 3 DMA request enable） • 0：禁止捕获/比较 3 的 DMA 请求 • 1：允许捕获/比较 3 的 DMA 请求
位 10	CC2DE：允许捕获/比较 2 的 DMA 请求（Capture/Compare 2 DMA request enable） • 0：禁止捕获/比较 2 的 DMA 请求 • 1：允许捕获/比较 2 的 DMA 请求
位 9	CC1DE：允许捕获/比较 1 的 DMA 请求（Capture/Compare 1 DMA request enable） • 0：禁止捕获/比较 1 的 DMA 请求 • 1：允许捕获/比较 1 的 DMA 请求
位 8	UDE：允许更新的 DMA 请求（Update DMA request enable） • 0：禁止更新的 DMA 请求 • 1：允许更新的 DMA 请求
位 7	BIE：允许刹车中断（Break interrupt enable） • 0：禁止刹车中断 • 1：允许刹车中断
位 6	TIE：触发中断使能（Trigger interrupt enable） • 0：禁止触发中断 • 1：使能触发中断
位 5	COMIE：允许 COM 中断（COM interrupt enable） • 0：禁止 COM 中断 • 1：允许 COM 中断
位 4	CC4IE：允许捕获/比较 4 中断（Capture/Compare 4 interrupt enable） • 0：禁止捕获/比较 4 中断 • 1：允许捕获/比较 4 中断
位 3	CC3IE：允许捕获/比较 3 中断（Capture/Compare 3 interrupt enable） • 0：禁止捕获/比较 3 中断 • 1：允许捕获/比较 3 中断
位 2	CC2IE：允许捕获/比较 2 中断（Capture/Compare 2 interrupt enable） • 0：禁止捕获/比较 2 中断 • 1：允许捕获/比较 2 中断
位 1	CC1IE：允许捕获/比较 1 中断（Capture/Compare 1 interrupt enable）

	• 0：禁止捕获/比较 1 中断 • 1：允许捕获/比较 1 中断
位 0	UIE：允许更新中断（Update interrupt enable） • 0：禁止更新中断 • 1：允许更新中断

16.3.5 TIM1/TIM8 状态寄存器（TIMx_SR）（x=1,8）

偏移地址：0x10

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res			CC4OF	CC3OF	CC2OF	CC1OF	Res	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

位 15:13	Res：保留 必须保持复位值。
位 12	CC4OF：捕获/比较 4 重复捕获标志（Capture/Compare 4 overcapture flag） 参见 CC1OF 描述。
位 11	CC3OF：捕获/比较 3 重复捕获标志（Capture/Compare 3 overcapture flag） 参见 CC1OF 描述。
位 10	CC2OF：捕获/比较 2 重复捕获标志（Capture/Compare 2 overcapture flag） 参见 CC1OF 描述。
位 9	CC1OF：捕获/比较 1 重复捕获标志（Capture/Compare 1 overcapture flag） 仅当相应的通道被配置为输入捕获时，该标志可由硬件置’1’。写 0 可清除该位。 • 0：无重复捕获产生 • 1：计数器的值被捕获到 TIMx_CCR1 寄存器时，CC1IF 的状态已经为’1’。
位 8	Res：保留 必须保持复位值。
位 7	BIF：刹车中断标志（Break interrupt flag） 一旦刹车输入有效，由硬件对该位置’1’。如果刹车输入无效，则该位可由软件清零。 • 0：无刹车事件产生 • 1：刹车输入上检测到有效电平
位 6	TIF：触发器中断标志（Trigger interrupt flag） 当发生触发事件（当从模式控制器处于除门控模式外的其它模式时，在 TRGI 输入端检测到有效边沿，或门控模式下的任一边沿）时，由硬件对该位置’1’。它由软件清零。 • 0：无触发器事件产生 • 1：触发中断等待响应
位 5	COMIF：COM 中断标志（COM interrupt flag） 一旦产生 COM 事件（当捕获/比较控制位：CCxE、CCxNE、OCxM 已被更新），该位由硬件置’1’。它由软

	件清零。 • 0：无 COM 事件产生 • 1：COM 中断等待响应
位 4	CC4IF：捕获/比较 4 中断标志（Capture/Compare 4 interrupt flag） 参考 CC1IF 描述。
位 3	CC3IF：捕获/比较 3 中断标志（Capture/Compare 3 interrupt flag） 参考 CC1IF 描述。
位 2	CC2IF：捕获/比较 2 中断标志（Capture/Compare 2 interrupt flag） 参考 CC1IF 描述。
位 1	CC1IF：捕获/比较 1 中断标志（Capture/Compare 1 interrupt flag） • 如果通道 CC1 配置为输出模式： 当计数器值与比较值匹配时该位由硬件置'1'，但在中心对称模式下除外（参考 TIMx_CR1 寄存器的 CMS 位）。它由软件清零。 ○ 0：无匹配发生。 ○ 1：TIMx_CNT 的值与 TIMx_CCR1 的值匹配。 当 TIMx_CCR1 的内容大于 TIMx_APR 的内容时，在向上或向上/下计数模式时计数器溢出，或向下计数模式时的计数器下溢条件下，CC1IF 位变高。 • 如果通道 CC1 配置为输入模式： 当捕获事件发生时该位由硬件置'1'，它由软件清零或通过读 TIMx_CCR1 清零。 ○ 0：无输入捕获产生。 ○ 1：计数器值已被捕获（拷贝）至 TIMx_CCR1（在 IC1 上检测到与所选极性相同的边沿）。
位 0	UIF：更新中断标志（Update interrupt flag） 当产生更新事件时该位由硬件置'1'。它由软件清零。 • 0：无更新事件产生 • 1：更新中断等待响应。当寄存器被更新时该位由硬件置'1'： ○ 若 TIMx_CR1 寄存器的 UDIS=0，当重复计数器数值上溢或下溢时（重复计数器=0 时产生更新事件）。 ○ 若 TIMx_CR1 寄存器的 URS=0、UDIS=0，当设置 TIMx_EGR 寄存器的 UG=1 时产生更新事件，通过软件对计数器 CNT 重新初始化时。 ○ 若 TIMx_CR1 寄存器的 URS=0、UDIS=0，当计数器 CNT 被触发事件重新初始化时。参考 TIM1/TIM8 从模式控制寄存器（TIMx_SMCR）。

16.3.6 TIM1/TIM8 事件产生寄存器（TIMx_EGR）（x=1,8）

偏移地址：0x14

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
								w	w	w	w	w	w	w	w

位 15:8	Res：保留 必须保持复位值。
位 7	BG：产生刹车事件（Break generation） 该位由软件置'1'，用于产生一个刹车事件，由硬件自动清零。 • 0：无动作。 • 1：产生一个刹车事件。此时 MOE=0、BIF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。

位 6	TG：产生触发事件（Trigger generation） 该位由软件置'1'，用于产生一个触发事件，由硬件自动清零。 • 0：无动作 • 1：TIMx_SR 寄存器的 TIF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。
位 5	COMG：捕获/比较控制更新生成（Capture/Compare control update generation） 该位由软件置'1'，由硬件自动清零。 • 0：无动作。 • 1：当 CCPC=1，允许更新 CCxE、CCxNE、OCxM 位。 *说明：该位只对拥有互补输出的通道有效。*
位 4	CC4G：产生捕获/比较 4 事件（Capture/Compare 4 generation） 参考 CC1G 描述。
位 3	CC3G：产生捕获/比较 3 事件（Capture/Compare 3 generation） 参考 CC1G 描述。
位 2	CC2G：产生捕获/比较 2 事件（Capture/Compare 2 generation） 参考 CC1G 描述。
位 1	CC1G：产生捕获/比较 1 事件（Capture/Compare 1 generation） 该位由软件置'1'，用于产生一个捕获/比较事件，由硬件自动清零。 • 0：无动作。 • 1：在通道 CC1 上产生一个捕获/比较事件： o 若通道 CC1 配置为输出： 设置 CC1IF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。 o 若通道 CC1 配置为输入： 当前的计数器值被捕获至 TIMx_CCR1 寄存器；设置 CC1IF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。若 CC1IF 已经为 1，则设置 CC1OF=1。
位 0	UG：产生更新事件（Update generation） 该位由软件置'1'，由硬件自动清零。 • 0：无动作。 • 1：重新初始化计数器，并产生一个更新事件。注意预分频器的计数器也被清零（但是预分频系数不变）。若在中心对称模式下或 DIR=0（向上计数）则计数器被清零；若 DIR=1（向下计数）则计数器取 TIMx_ARR 的值。

16.3.7 TIM1/TIM8 捕获/比较模式寄存器 1（TIMx_CCMR1）（x=1,8）

偏移地址：0x18

复位值：0x0000

通道可用于输入（捕获模式）或输出（比较模式），通道的方向由相应的 CCxS 位定义。该寄存器其它位的作用在输入和输出模式下不同。OCxx 描述了通道在输出模式下的功能，ICxx 描述了通道在输入模式下的功能。因此必须注意，同一个位在输出模式和输入模式下的功能是不同的。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	

输出比较模式：

位 15	OC2CE：输出比较 2 清零使能（Output Compare 2 clear enable） 参见 OC1CE 的描述。
位 14:12	OC2M[2:0]：输出比较 2 模式（Output Compare 2 mode） 参见 OC1M 的描述。
位 11	OC2PE：输出比较 2 预装载使能（Output Compare 2 preload enable） 参见 OC1PE 的描述。
位 10	OC2FE：输出比较 2 快速使能（Output Compare 2 fast enable） 参见 OC1FE 的描述。
位 9:8	CC2S[1:0]：捕获/比较 2 选择（Capture/Compare 2 selection） 该位定义通道的方向（输入/输出），及输入脚的选择： • 00：CC2 通道被配置为输出。 • 01：CC2 通道被配置为输入，IC2 映射在 TI2 上。 • 10：CC2 通道被配置为输入，IC2 映射在 TI1 上。 • 11：CC2 通道被配置为输入，IC2 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时（由 TIMx_SMCR 寄存器的 TS 位选择）。 *说明：CC2S 仅在通道关闭时（TIMx_CCER 寄存器的 CC2E=0）才是可写的。*
位 7	OC1CE：输出比较 1 清零使能（Output Compare 1 clear enable） • 0：OC1REF 不受 ETRF 输入的影响。 • 1：一旦检测到 ETRF 输入高电平，清除 OC1REF=0。
位 6:4	OC1M[2:0]：输出比较 1 模式（Output Compare 1 mode） 该 3 位定义了输出参考信号 OC1REF 的动作，而 OC1REF 决定了 OC1、OC1N 的值。OC1REF 是高电平有效，而 OC1、OC1N 的有效电平取决于 CC1P、CC1NP 位。 • 000：冻结 输出比较寄存器 TIMx_CCR1 与计数器 TIMx_CNT 间的比较对 OC1REF 不起作用。 • 001：匹配时设置通道 1 为有效电平 当计数器 TIMx_CNT 的值与捕获/比较寄存器 1（TIMx_CCR1）相同时，强制 OC1REF 为高。 • 010：匹配时设置通道 1 为无效电平 当计数器 TIMx_CNT 的值与捕获/比较寄存器 1（TIMx_CCR1）相同时，强制 OC1REF 为低。 • 011：翻转 当 TIMx_CCR1=TIMx_CNT 时，翻转 OC1REF 的电平。 • 100：强制为无效电平 强制 OC1REF 为低。 • 101：强制为有效电平 强制 OC1REF 为高。 • 110：PWM 模式 1 o 在向上计数时，一旦 TIMx_CNT<TIMx_CCR1 时通道 1 为有效电平，否则为无效电平。 o 在向下计数时，一旦 TIMx_CNT>TIMx_CCR1 时通道 1 为无效电平（OC1REF=0），否则为有效电平（OC1REF=1）。 • 111：PWM 模式 2 o 在向上计数时，一旦 TIMx_CNT<TIMx_CCR1 时通道 1 为无效电平，否则为有效电平。 o 在向下计数时，一旦 TIMx_CNT>TIMx_CCR1 时通道 1 为有效电平，否则为无效电平。 *说明：*

	• 一旦 LOCK 级别设为 3（TIMx_BDTR 寄存器中的 LOCK 位）并且 CC1S=00（该通道配置成输出）则该位不能被修改。 • 在 PWM 模式 1 或 PWM 模式 2 中，只有当比较结果改变了或在输出比较模式中从冻结模式切换到 PWM 模式时，OC1REF 电平才改变。
位 3	OC1PE：输出比较 1 预装载使能（Output Compare 1 preload enable） • 0：禁止 TIMx_CCR1 寄存器的预装载功能，可随时写入 TIMx_CCR1 寄存器，并且新写入的数值立即起作用。 • 1：开启 TIMx_CCR1 寄存器的预装载功能，读写操作仅对预装载寄存器操作，TIMx_CCR1 的预装载值在更新事件到来时被加载至当前寄存器中。 说明： • 一旦 LOCK 级别设为 3（TIMx_BDTR 寄存器中的 LOCK 位）并且 CC1S=00（该通道配置成输出）则该位不能被修改。 • 仅在单脉冲模式下（TIMx_CR1 寄存器的 OPM=1），可以在未确认预装载寄存器情况下使用 PWM 模式，否则其动作不确定。
位 2	OC1FE：输出比较 1 快速使能（Output Compare 1 fast enable） 该位用于加快 CC 输出对触发输入事件的响应。 • 0：根据计数器与 CCR1 的值，CC1 正常操作，即使触发器是打开的。当触发器的输入有一个有效沿时，激活 CC1 输出的最小延时为 5 个时钟周期。 • 1：输入到触发器的有效沿的作用就像发生了一次比较匹配。因此，OC 被设置为比较电平而与比较结果无关。采样触发器的有效沿和 CC1 输出间的延时被缩短为 3 个时钟周期。 OCxFE 只在通道被配置成 PWM1 或 PWM2 模式时起作用。
位 1:0	CC1S[1:0]：捕获/比较 1 选择。（Capture/Compare 1 selection） 这 2 位定义通道的方向（输入/输出），及输入脚的选择： • 00：CC1 通道被配置为输出。 • 01：CC1 通道被配置为输入，IC1 映射在 TI1 上。 • 10：CC1 通道被配置为输入，IC1 映射在 TI2 上。 • 11：CC1 通道被配置为输入，IC1 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时（由 TIMx_SMCR 寄存器的 TS 位选择）。 *说明：CC1S 仅在通道关闭时（TIMx_CCER 寄存器的 CC1E=0）才是可写的。*

输入捕获模式：

位 15:12	IC2F[3:0]：输入/捕获 2 滤波器（Input capture 2 filter） 可参考 IC1F 的描述。
位 11:10	IC2PSC[1:0]：输入/捕获 2 预分频器（Input capture 2 prescaler） 可参考 IC1PSC 的描述。
位 9:8	CC2S[1:0]：捕获/比较 2 选择（Capture/Compare 2 selection） 这 2 位定义通道的方向（输入/输出），及输入脚的选择： • 00：CC2 通道被配置为输出。 • 01：CC2 通道被配置为输入，IC2 映射在 TI2 上。 • 10：CC2 通道被配置为输入，IC2 映射在 TI1 上。 • 11：CC2 通道被配置为输入，IC2 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时（由 TIMx_SMCR 寄存器的 TS 位选择）。 *说明：CC2S 仅在通道关闭时（TIMx_CCER 寄存器的 CC2E=0）才是可写的。*

位 7:4	IC1F[3:0]：输入捕获 1 滤波器（Input capture 1 filter） 这几位定义了 TI1 输入的采样频率及数字滤波器长度。数字滤波器由一个事件计数器组成，它记录到 N 个事件后会产生一个输出的跳变： • 0000：无滤波器，以 f_{DTS} 采样 • 0001：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N=2 • 0010：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N=4 • 0011：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N=8 • 0100：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N=6 • 0101：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N=8 • 0110：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N=6 • 0111：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N=8 • 1000：采样频率 $f_{SAMPLING}=f_{DTS}/8$，N=6 • 1001：采样频率 $f_{SAMPLING}=f_{DTS}/8$，N=8 • 1010：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N=5 • 1011：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N=6 • 1100：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N=8 • 1101：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N=5 • 1110：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N=6 • 1111：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N=8
位 3:2	IC1PSC[1:0]：输入/捕获 1 预分频器（Input capture 1 prescaler） 这 2 位定义了 CC1 输入（IC1）的预分频系数。 一旦 CC1E=0（TIMx_CCER 寄存器中），则预分频器复位。 • 00：无预分频器，捕获输入口上检测到的每一个边沿都触发一次捕获。 • 01：每 2 个事件触发一次捕获。 • 10：每 4 个事件触发一次捕获。 • 11：每 8 个事件触发一次捕获。
位 1:0	CC1S[1:0]：捕获/比较 1 选择（Capture/Compare 1 Selection） 这 2 位定义通道的方向（输入/输出），及输入脚的选择： • 00：CC1 通道被配置为输出。 • 01：CC1 通道被配置为输入，IC1 映射在 TI1 上。 • 10：CC1 通道被配置为输入，IC1 映射在 TI2 上。 • 11：CC1 通道被配置为输入，IC1 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时（由 TIMx_SMCR 寄存器的 TS 位选择）。 *说明：CC1S 仅在通道关闭时（TIMx_CCER 寄存器的 CC1E=0）才是可写的。*

16.3.8 TIM1/TIM8 捕获/比较模式寄存器 2（TIMx_CCMR2）（x=1,8）

偏移地址：0x1C

复位值：0x0000

参考以上 CCMR1 寄存器的描述。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	

输出比较模式：

位 15	OC4CE：输出比较 4 清零使能（Output compare 4 clear enable） 参见 OC1CE 的描述。
位 14:12	OC4M[2:0]：输出比较 4 模式（Output compare 4 mode） 参见 OC1M 的描述。
位 11	OC4PE：输出比较 4 预装载使能（Output compare 4 preload enable） 参见 OC1PE 的描述。
位 10	OC4FE：输出比较 4 快速使能（Output compare 4 fast enable） 参见 OC1FE 的描述。
位 9:8	CC4S[1:0]：捕获/比较 4 选择（Capture/Compare 4 selection） 该 2 位定义通道的方向（输入/输出），及输入脚的选择： • 00：CC4 通道被配置为输出。 • 01：CC4 通道被配置为输入，IC4 映射在 TI4 上。 • 10：CC4 通道被配置为输入，IC4 映射在 TI3 上。 • 11：CC4 通道被配置为输入，IC4 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时（由 TIMx_SMCR 寄存器的 TS 位选择）。 *说明：CC4S 仅在通道关闭时（TIMx_CCER 寄存器的 CC4E=0）才是可写的。*
位 7	OC3CE：输出比较 3 清零使能（Output compare 3 clear enable） 参见 OC1CE 的描述。
位 6:4	OC3M[2:0]：输出比较 3 模式（Output compare 3 mode） 参见 OC1M 的描述。
位 3	OC3PE：输出比较 3 预装载使能（Output compare 3 preload enable） 参见 OC1PE 的描述。
位 2	OC3FE：输出比较 3 快速使能（Output compare 3 fast enable） 参见 OC1FE 的描述。
位 1:0	CC3S[1:0]：捕获/比较 3 选择（Capture/Compare 3 selection） 这 2 位定义通道的方向（输入/输出），及输入脚的选择： • 00：CC3 通道被配置为输出。 • 01：CC3 通道被配置为输入，IC3 映射在 TI3 上。 • 10：CC3 通道被配置为输入，IC3 映射在 TI4 上。 • 11：CC3 通道被配置为输入，IC3 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时（由 TIMx_SMCR 寄存器的 TS 位选择）。 *说明：CC3S 仅在通道关闭时（TIMx_CCER 寄存器的 CC3E=0）才是可写的。*

输入捕获模式：

位 15:12	IC4F[3:0]：输入捕获 4 滤波器（Input capture 4 filter） 参考 CC1P 的描述。
位 11:10	IC4PSC[1:0]：输入/捕获 4 预分频器（Input capture 4 prescaler）
位 9:8	CC4S[1:0]：捕获/比较 4 选择（Capture/Compare 4 selection）

	这 2 位定义通道的方向（输入/输出），及输入脚的选择： • 00：CC4 通道被配置为输出。 • 01：CC4 通道被配置为输入，IC4 映射在 TI4 上。 • 10：CC4 通道被配置为输入，IC4 映射在 TI3 上。 • 11：CC4 通道被配置为输入，IC4 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时（由 TIMx_SMCR 寄存器的 TS 位选择）。 *说明：CC4S 仅在通道关闭时（TIMx_CCER 寄存器的 CC4E=0）才是可写的。*
位 7:4	IC3F[3:0]：输入捕获 3 滤波器（Input capture 3 filter）
位 3:2	IC3PSC[1:0]：输入/捕获 3 预分频器（Input capture 3 prescaler）
位 1:0	CC3S[1:0]：捕获/比较 3 选择（Capture/Compare 3 selection） 这 2 位定义通道的方向（输入/输出），及输入脚的选择： • 00：CC3 通道被配置为输出。 • 01：CC3 通道被配置为输入，IC3 映射在 TI3 上。 • 10：CC3 通道被配置为输入，IC3 映射在 TI4 上。 • 11：CC3 通道被配置为输入，IC3 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时（由 TIMx_SMCR 寄存器的 TS 位选择）。 *说明：CC3S 仅在通道关闭时（TIMx_CCER 寄存器的 CC3E=0）才是可写的。*

16.3.9 TIM1/TIM8 捕获/比较使能寄存器（TIMx_CCER）（x=1,8）

偏移地址：0x20

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 15	CC4NP：捕获/比较 4 极性（Capture/Compare 4 complementary output polarity） 这一位和 CC1P 联合使用来定义 TI3FP4 和 TI4FP4。详细的参考 CC1P 的描述。 *说明：该位仅用于通道配置为输入时使用。*
位 14	Res：保留 必须保持复位值。
位 13	CC4P：捕获/比较 4 输出极性（Capture/Compare 4 output polarity） 参考 CC1P 的描述。
位 12	CC4E：捕获/比较 4 输出使能（Capture/Compare 4 output enable） 参考 CC1E 的描述。
位 11	CC3NP：捕获/比较 3 互补输出极性（Capture/Compare 3 complementary output polarity） 参考 CC1NP 的描述。
位 10	CC3NE：捕获/比较 3 互补输出使能（Capture/Compare 3 complementary output enable） 参考 CC1NE 的描述。
位 9	CC3P：捕获/比较 3 输出极性（Capture/Compare 3 output polarity） 参考 CC1P 的描述。

位 8	CC3E：捕获/比较 3 输出使能（Capture/Compare 3 output enable） 参考 CC1E 的描述。
位 7	CC2NP：捕获/比较 2 互补输出极性（Capture/Compare 2 complementary output polarity） 参考 CC1NP 的描述。
位 6	CC2NE：捕获/比较 2 互补输出使能（Capture/Compare 2 complementary output enable） 参考 CC1NE 的描述。
位 5	CC2P：捕获/比较 2 输出极性（Capture/Compare 2 output polarity） 参考 CC1P 的描述。
位 4	CC2E：捕获/比较 2 输出使能（Capture/Compare 2 output enable） 参考 CC1E 的描述。
位 3	CC1NP：捕获/比较 1 互补输出极性（Capture/Compare 1 complementary output polarity） • CC1 通道配置为输出： o 0：OC1N 高电平有效 o 1：OC1N 低电平有效 • CC1 通道配置为输入： 这一位和 CC1P 联合使用来定义 TI1FP1 和 TI2FP1。详细的参考 CC1P 的描述。 *说明：* • *在具有互补输出的通道上，此位是预加载的。如果 CCPC 位是在 TIMx_CR2 寄存器中设置，CC1NP 仅在通讯事件产生时加载新的值。* • *一旦 LOCK 级别（TIMx_BDTR 寄存器中的 LOCK 位）设为 3 或 2 且 CC1S=00（通道配置为输出）则该位不能被修改。*
位 2	CC1NE：捕获/比较 1 互补输出使能（Capture/Compare 1 complementary output enable） • 0：关闭 OC1N 禁止输出，因此 OC1N 的电平依赖于 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1E 位的值。 • 1：开启 OC1N 信号输出到对应的输出引脚，其输出电平依赖于 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1E 位的值。 *说明：在具有互补输出的通道上，此位是预加载的。如果 CCPC 位是在 TIMx_CR2 寄存器中设置，CC1NE 仅在通讯事件产生时加载新的值。*
位 1	CC1P：捕获/比较 1 输出极性（Capture/Compare 1 output polarity） • CC1 通道配置为输出： o 0：OC1 高电平有效 o 1：OC1 低电平有效 • CC1 通道配置为输入： CC1NP/CC1P 联合组成两位控制来选择 TI1FP1 和 TI2FP1 的激活极性： o 00：没有翻转/上升沿 电路对 TIxFP1 的上升沿敏感（捕获或触发操作复位，外部时钟或触发模式），TIxFP1 没有翻转（触发操作在门控或编码模式）。 o 01：翻转/下降沿 电路对 TIxFP1 的下降沿敏感（捕获或触发操作复位，外部时钟或触发模式），TIxFP1 翻转（触发操作在门控或编码模式）。 o 10：保留

	○ 11：没有翻转/双沿 电路对 TIxFP1 的上升沿和下降沿敏感（捕获或触发操作复位，外部时钟或触发模式），TIxFP1 没有翻转（触发操作在门控模式）。这个配置不能用在编码模式。 *说明：* • *在具有互补输出的通道上，此位是预加载的。如果 CCPC 位是在 TIMx_CR2 寄存器中设置，CC1P 仅在通讯事件产生时加载新的值。* • *一旦 LOCK 级别（TIMx_BDTR 寄存器中的 LOCK 位）设为 3 或 2，则该位不能被修改。*
位 0	CC1E：捕获/比较 1 输出使能（Capture/Compare 1 output enable） • CC1 通道配置为输出： ○ 0：关闭 OC1 禁止输出，因此 OC1 的输出电平依赖于 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1NE 位的值。 ○ 1：开启 OC1 信号输出到对应的输出引脚，其输出电平依赖于 MOE、OSSI、OSSR、OIS1、OIS1N 和 CC1NE 位的值。 • CC1 通道配置为输入： 该位决定了计数器的值是否能捕获入 TIMx_CCR1 寄存器。 ○ 0：捕获禁止 ○ 1：捕获使能 *说明：在具有互补输出的通道上，此位是预加载的。如果 CCPC 位是在 TIMx_CR2 寄存器中设置，CC1E 仅在通讯事件产生时加载新的值。*

表 16-3 带刹车功能的互补输出通道 OCx 和 OCxN 的控制位

控制位					输出状态 (1)	
MOE 位	OSSI 位	OSSR 位	CCxE 位	CCxNE 位	OCx 输出状态	OCxN 输出状态
1	X	0	0	0	输出禁止（与定时器断开） OCx=0，OCx_EN=0	输出禁止（与定时器断开） OCxN=0，OCxN_EN=0
		0	0	1	输出禁止（与定时器断开） OCx=0，OCx_EN=0	OCxREF+极性， OCxN=OCxREF xor CCxNP， OCxN_EN=1
		0	1	0	OCxREF+极性， OCx=OCxREFxorCCxP， OCx_EN=1	输出禁止（与定时器断开） OCxN=0，OCxN_EN=0
		0	1	1	OCxREF+极性+死区， OCx_EN=1	OCxREF 反相+极性+死区， OCxN_EN=1
		1	0	0	输出禁止（与定时器断开） OCx=CCxP，OCx_EN=0	输出禁止（与定时器断开） OCxN=CCxNP，OCxN_EN=0
		1	0	1	关闭状态（输出使能且为无效电平）OCx=CCxP，OCx_EN=1	OCxREF+极性， OCxN=OCxREF xor CCxNP， OCxN_EN=1
		1	1	0	OCxREF+极性， OCx=OCxREF xor CCxP， OCx_EN=1	关闭状态（输出使能且为无效电平）OCxN=CCxNP，OCxN_EN=1

控制位					输出状态 (1)	
		1	1	1	OCxREF+极性+死区， OCx_EN=1	OCxREF 反相+极性+死区， OCxN_EN=1
0	0	X	0	0	输出禁止（与定时器断开） 异步地：OCx=CCxP，OCx_EN=0，OCxN=CCxNP，OCxN_EN=0； 若时钟存在：经过一个死区时间后 OCx=OISx，OCxN=OISxN，假设 OISx 与 OISxN 并不都对应 OCx 和 OCxN 的有效电平。	
	0		0	1		
	0		1	0		
	0		1	1		
	1		0	0	关闭状态（输出使能且为无效电平） 异步地：OCx=CCxP，OCx_EN=1，OCxN=CCxNP，OCxN_EN=1； 若时钟存在：经过一个死区时间后 OCx=OISx，OCxN=OISxN，假设 OISx 与 OISxN 并不都对应 OCx 和 OCxN 的有效电平。	
	1		0	1		
	1		1	0		
	1		1	1		

(1). 如果一个通道的 2 个输出都没有使用（CCxE=CCxNE=0），那么 OISx，OISxN，CCxP 和 CCxNP 都必须清零。

说明：引脚连接到互补的 OCx 和 OCxN 通道的外部 I/O 引脚的状态，取决于 OCx 和 OCxN 通道状态和 GPIO 以及 AFIO 寄存器。

16.3.10 TIM1/TIM8 计数器寄存器（TIMx_CNT）（x=1,8）

偏移地址：0x24

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw															

位 15:0	CNT[15:0]：计数器的值（Counter value）

16.3.11 TIM1/TIM8 预分频器寄存器（TIMx_PSC）（x=1,8）

偏移地址：0x28

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw															

位 15:0	PSC[15:0]：预分频器的值（Prescaler value） 计数器的时钟频率（CK_CNT）等于 f_{CK_PSC}/（PSC[15:0]+1）。 每次更新事件产生时，PSC 的值被加载到当前分频器的寄存器中；更新事件包括计数器被 TIM_EGR 的 UG 位清零或计数器被配置为复位模式时通过触发控制器清零。

16.3.12 TIM1/TIM8 自动重装载寄存器（TIMx_ARR）（x=1,8）

偏移地址：0x2C

复位值：0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw															

位 15:0	ARR[15:0]：自动重装载的值（Auto-reload value） ARR 包含了将要装载入实际的自动重装载寄存器的值。有关 ARR 的更新和动作，可参考“16.2.1 时基单元”。 当自动重装载的值为空时，计数器不工作。

16.3.13 TIM1/TIM8 重复计数寄存器（TIMx_RCR）（x=1,8）

偏移地址：0x30

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								REP[7:0]							
								rw							

位 15:8	Res：保留 必须保持复位值。
位 7:0	REP[7:0]：重复计数器的值（Repetition counter value） 开启了预装载功能后，该位域允许用户设置比较寄存器的更新速率（即周期性地从预装载寄存器传输到当前寄存器）；如果允许产生更新中断，则会同时影响产生更新中断的速率。 每次向下计数器（REP_CNT）达到 0，会产生一个更新事件并且 REP_CNT 重新从 REP 值开始计数。由于 REP_CNT 只有在周期更新事件 U_RC 发生时才重载 REP 值，因此对 TIMx_RCR 寄存器写入的新值只在下次周期更新事件发生时才起作用。 这意味着在 PWM 模式中，（REP+1）对应着： • 在边沿对齐模式下，PWM 周期的数目。 • 在中心对称模式下，PWM 半周期的数目。

16.3.14 TIM1/TIM8 捕获/比较寄存器 1（TIMx_CCR1）（x=1,8）

偏移地址：0x34

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw															

位 15:0	CCR1[15:0]：捕获/比较通道 1 的值（Capture/Compare 1 value） • 若 CC1 通道配置为输出： CCR1 包含了装入当前捕获/比较 1 寄存器的值（预装载值）。 如果在 TIMx_CCMR1 寄存器（OC1PE 位）中未选择预装载功能，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 1 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC1 端口上产生输出信号。 • 若 CC1 通道配置为输入： CCR1 包含了由上一次输入捕获 1 事件（IC1）传输的计数器值。TIMx_CCR1 只能读不能被编程。

16.3.15 TIM1/TIM8 捕获/比较寄存器 2（TIMx_CCR2）（x=1,8）

偏移地址：0x38

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw															

位 15:0	CCR2[15:0]：捕获/比较通道 2 的值（Capture/Compare 2 value） • 若 CC2 通道配置为输出： CCR2 包含了装入当前捕获/比较 2 寄存器的值（预装载值）。 如果在 TIMx_CCMR2 寄存器（OC2PE 位）中未选择预装载特性，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 2 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC2 端口上产生输出信号。 • 若 CC2 通道配置为输入： CCR2 包含了由上一次输入捕获 2 事件（IC2）传输的计数器值。TIMx_CCR2 只能读不能被编程。

16.3.16 TIM1/TIM8 捕获/比较寄存器 3（TIMx_CCR3）（x=1,8）

偏移地址：0x3C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw															

位 15:0	CCR3[15:0]：捕获/比较通道 3 的值（Capture/Compare 3 value） • 若 CC3 通道配置为输出： CCR3 包含了装入当前捕获/比较 3 寄存器的值（预装载值）。 如果在 TIMx_CCMR3 寄存器（OC3PE 位）中未选择预装载特性，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 3 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC3 端口上产生输出信号。 • 若 CC3 通道配置为输入： CCR3 包含了由上一次输入捕获 3 事件（IC3）传输的计数器值。TIMx_CCR3 只能读不能被编程。

16.3.17 TIM1/TIM8 捕获/比较寄存器 4（TIMx_CCR4）（x=1,8）

偏移地址：0x40

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw															

位 15:0	CCR4[15:0]：捕获/比较通道 4 的值（Capture/Compare 4 value） • 若 CC4 通道配置为输出： CCR4 包含了装入当前捕获/比较 4 寄存器的值（预装载值）。 如果在 TIMx_CCMR4 寄存器（OC4PE 位）中未选择预装载特性，写入的数值会立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 4 寄存器中。 当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC4 端口上产生输出信号。 • 若 CC4 通道配置为输入： CCR4 包含了由上一次输入捕获 4 事件（IC4）传输的计数器值。TIMx_CCR4 只能读不能被编程。

16.3.18 TIM1/TIM8 刹车和死区寄存器（TIMx_BDTR）（x=1,8）

偏移地址：0x44

复位值：0x0000 0000

说明：根据锁定设置，该寄存器的 AOE、BKP、BKE、OSSI、OSSR 和 DTG[7:0]位均可被写保护，有必要在第一次写入 TIMx_BDTR 寄存器时对它们进行配置。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw		rw							

位 15	MOE：主输出使能（Main output enable） 一旦刹车输入有效，该位被硬件异步清零。根据 AOE 位的设置值，该位可以由软件清零或被自动置'1'。它仅对配置为输出的通道有效。 • 0：禁止 OC 和 OCN 输出或强制为空闲状态。 • 1：如果设置了相应的使能位（TIMx_CCER 寄存器的 CCxE、CCxNE 位），则开启 OC 和 OCN 输出。 有关 OC/OCN 使能的细节，参见 TIM1/TIM8 捕获/比较使能寄存器（TIMx_CCER）。
位 14	AOE：自动输出使能（Automatic output enable） • 0：MOE 只能被软件置'1'。 • 1：MOE 能被软件置'1'或在下一个更新事件被自动置'1'（如果刹车输入无效）。 *说明：一旦 LOCK 级别（TIMx_BDTR 寄存器中的 LOCK 位）设为'1'，则该位不能被修改。*
位 13	BKP：刹车输入极性（Break polarity） • 0：刹车输入低电平有效。 • 1：刹车输入高电平有效。 *说明：* • 一旦 LOCK 级别（TIMx_BDTR 寄存器中的 LOCK 位）设为'1'，则该位不能被修改。 • 任何对该位的写操作都需要一个 APB 时钟的延迟以后才能起作用。
位 12	BKE：刹车功能使能（Break enable） • 0：禁止刹车输入（BRK、CCS 时钟失效事件及比较器输出结果）。 • 1：开启刹车输入（BRK、CCS 时钟失效事件及比较器输出结果）。 *说明：* • 当设置了 LOCK 级别 1 时（TIMx_BDTR 寄存器中的 LOCK 位），该位不能被修改。 • 任何对该位的写操作都需要一个 APB 时钟的延迟以后才能起作用。
位 11	OSSR：运行模式下"关闭状态"选择（Off-state selection for Run mode） 该位用于当 MOE=1 且通道为互补输出时。没有互补输出的定时器中不存在 OSSR 位。 OC/OCN 使能的详细说明，可参考 TIM1/TIM8 捕获/比较使能寄存器（TIMx_CCER）。 • 0：当定时器不工作时，禁止 OC/OCN 输出（OC/OCN 使能输出信号=0）。 • 1：当定时器不工作时，一旦 CCxE=1 或 CCxNE=1，首先开启 OC/OCN 并输出无效电平，然后置 OC/OCN 使能输出信号=1。 *说明：一旦 LOCK 级别（TIMx_BDTR 寄存器中的 LOCK 位）设为 2，则该位不能被修改。*
位 10	OSSI：空闲模式下"关闭状态"选择（Off-state selection for Idle mode） 该位用于当 MOE=0 且通道设为输出时。 OC/OCN 使能的详细说明，可参考 TIM1/TIM8 捕获/比较使能寄存器（TIMx_CCER）。

	• 0：当定时器不工作时，禁止 OC/OCN 输出（OC/OCN 使能输出信号=0）。 • 1：当定时器不工作时，一旦 CCxE=1 或 CCxNE=1，OC/OCN 首先输出其空闲电平，然后 OC/OCN 使能输出信号=1。 *说明：一旦 LOCK 级别（TIMx_BDTR 寄存器中的 LOCK 位）设为 2，则该位不能被修改。*
位 9:8	LOCK[1:0]：锁定设置（Lock configuration） 该位为防止软件错误而提供写保护。 • 00：锁定关闭，寄存器无写保护。 • 01：锁定级别 1，不能写入 TIMx_BDTR 寄存器的 DTG、BKE、BKP、AOE 位和 TIMx_CR2 寄存器的 OISx/OISxN 位。 • 10：锁定级别 2，不能写入锁定级别 1 中的各位，也不能写入 CC 极性位（一旦相关通道通过 CCxS 位设为输出，CC 极性位是 TIMx_CCER 寄存器的 CCxP/CCNxP 位）以及 OSSR/OSSI 位。 • 11：锁定级别 3，不能写入锁定级别 2 中的各位，也不能写入 CC 控制位（一旦相关通道通过 CCxS 位设为输出，CC 控制位是 TIMx_CCMRx 寄存器的 OCxM/OCxPE 位）。 *说明：在系统复位后，只能写一次 LOCK 位，一旦写入 TIMx_BDTR 寄存器，则其内容冻结直至复位。*
位 7:0	DTG[7:0]：死区发生器设置（Dead-time generator setup） 该位域定义了插入互补输出之间的死区持续时间。假设 DT 表示其持续时间： • DTG[7:5]=0xx=>DT=DTG[7:0]×Tdtg，其中 Tdtg=TDTS； • DTG[7:5]=10x=>DT=（64+DTG[5:0]）×Tdtg，其中 Tdtg=2×T_{DTS}； • DTG[7:5]=110=>DT=（32+DTG[4:0]）×Tdtg，其中 Tdtg=8×T_{DTS}； • DTG[7:5]=111=>DT=（32+DTG[4:0]）×Tdtg，其中 Tdtg=16×T_{DTS}； 例：若 T_{DTS}=125 ns（8 MHz），可能的死区时间为： • 0 到 15875 ns，步长时间为 125 ns； • 16 µs 到 31750 ns，步长时间为 250 ns； • 32 µs 到 63 µs，步长时间为 1 µs； • 64 µs 到 126 µs，步长时间为 2 µs； *说明：一旦 LOCK 级别（TIMx_BDTR 寄存器中的 LOCK 位）设为 1、2 或 3，则不能修改这些位。*

16.3.19 TIM1/TIM8 DMA 控制寄存器（TIMx_DCR）（x=1,8）

偏移地址：0x48

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res			DBL[4:0]					Res			DBA[4:0]				
			rw								rw				

位 15:13	Res：保留 必须保持复位值。
位 12:8	DBL[4:0]：DMA 连续传送长度（DMA burst length） 该位域定义了 DMA 在连续模式下的传送长度（当对 TIMx_DMAR 寄存器进行读或写时，定时器则进行一次连续传送），即：定义传输的次数，传输可以是半字（双字节）或字节： • 00000：1 次传输 • 00001：2 次传输 • 00010：3 次传输 • ...

	• 10001：18 次传输 例：考虑这样的传输：DBL=7，DBA=TIM2_CR1 • 如果 DBL=7，DBA=TIM2_CR1 表示待传输数据的地址，那么传输的地址由下式给出： （TIMx_CR1 的地址）+DBA+（DMA 索引），其中：DMA 索引=DBL=7；DBA：参考 DBA 描述。 （TIMx_CR1 的地址）+DBA+7，给出了将要写入或者读出数据的地址，这样数据的传输将发生在从地址（TIMx_CR1 的地址）+DBA 开始的 7 个寄存器。 根据 DMA 数据长度的设置，可能发生以下情况： • 如果设置数据为半字（16 位），那么数据就会传输给全部 7 个寄存器。 • 如果设置数据为字节，数据仍然会传输给全部 7 个寄存器：第一个寄存器包含第一个 MSB 字节，第二个寄存器包含第一个 LSB 字节，以此类推。因此对于定时器，用户必须指定由 DMA 传输的数据宽度。
位 7:5	Res：保留 必须保持复位值。
位 4:0	DBA[4:0]：DMA 基地址（DMA base address） 该位域定义了 DMA 在连续模式下的基地址（当对 TIMx_DMAR 寄存器进行读或写时），DBA 定义为从 TIMx_CR1 寄存器所在地址开始的偏移量： • 00000：TIMx_CR1 • 00001：TIMx_CR2 • 00010：TIMx_SMCR ... 示例：考虑以下传输：DBL=7 次传输，DBA=TIMx_CR1。这种情况下将向/从自 TIMx_CR1 地址开始的 7 个寄存器传输数据。

16.3.20 TIM1/TIM8 连续模式的 DMA 地址寄存器（TIMx_DMAR）（x=1,8）

偏移地址：0x4C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw															

位 15:0	DMAB[15:0]：DMA 连续传送寄存器（DMA register for burst accesses） 对 TIMx_DMAR 寄存器的读或写会导致对以下地址所在寄存器的存取操作： （TIMx_CR1 地址）+（DBA + DMA 索引）*4，其中： • “TIMx_CR1 地址”是控制寄存器 1（TIMx_CR1）所在的地址； • “DBA”是 TIMx_DCR 寄存器中定义的基地址； • “DMA 索引”是由 DMA 自动控制的偏移量，它取决于 TIMx_DCR 寄存器中定义的 DBL 位。 **DMA 连续传送功能使用方法示例** 例如：使用定时器 DMA 连续传送功能，将 TIM1_CCRx 寄存器（x = 2、3、4）更新为 DMA 传输到 TIM1_CCRx 寄存器的新内容。 具体操作步骤如下： 1. 将相应的 DMA 通道配置如下： o DMA 通道外设地址为 TIM1_DMAR 寄存器地址。 o DMA 通道存储器地址为包含要通过 DMA 传输到 TIM1_CCRx 寄存器的数据的 RAM 缓冲区地址。 o 要传输的数据量为 3（参见下文“注意”）。 o 禁止循环模式。 2. 配置 TIM1_DCR 寄存器的 DBA 和 DBL 位如下：

	○ DBL = 3 次传输，DBA = 0xE。 3. 使能 TIM1 更新 DMA 请求（将 TIM1_DIER.UDE 置 1）。 4. 使能 TIM1（将 TIM1_CR1.CEN 置 1）。 5. 使能 DMA 通道。 *注意：本例适用于每个 TIM1_CCRx 寄存器只更新一次的情况。如果每个 CCRx 寄存器要更新两次，则要传输的数据量应为 6。* *下面以包含 data1、data2、data3、data4、data5 和 data6 的 RAM 缓冲区为例说明。* 数据将按照如下方式传输到 TIM1_CCRx 寄存器： 1. 在第一个更新 DMA 请求期间，data1 传输到 CCR2，data2 传输到 CCR3，data3 传输到 CCR4； 2. 在第二个更新 DMA 请求期间，data4 传输到 CCR2，data5 传输到 CCR3，data6 传输到 CCR4。

17 通用定时器（TIMx）

通用定时器由一个通过可编程预分频器驱动的 16 位自动装载计数器构成。它适用于多种场合，包括测量输入信号的脉冲长度（输入捕获）或者产生输出波形（输出比较和 PWM）。

使用定时器预分频器和 RCC 时钟控制器预分频器，脉冲长度和波形周期可以在几个微秒到几个毫秒间调整。

每个定时器都是完全独立的，没有互相共享任何资源。它们可以一起同步操作，参见：“定时器同步”。

TIM2/TIM3/TIM4/TIM5 的功能完全一样，只是寄存器基地址不同；寄存器基地址请参见“17.3 TIM2/3/4/5 寄存器”。

17.1 TIMx 主要功能

通用 TIMx（TIM2、TIM3、TIM4、TIM5）定时器功能包括：

- 16 位向上、向下、向上/向下自动装载计数器
- 16 位可编程（可以实时修改）预分频器，计数器时钟频率的分频系数为 1～65536 之间的任意数值
- 4 个独立通道：
  - 输入捕获
  - 输出比较
  - PWM 生成（边沿或中央对齐模式）
  - 单脉冲模式输出
- 使用外部信号控制定时器和定时器互联的同步电路
- 如下事件发生时产生中断/DMA：
  - 更新：计数器向上溢出/向下溢出，计数器初始化（通过软件或者内部/外部触发）
  - 触发事件（计数器启动、停止、初始化或者由内部/外部触发计数）
  - 输入捕获
  - 输出比较
- 支持针对定位的增量（正交）编码器和霍尔传感器电路
- 触发输入作为外部时钟或者按周期的电流管理

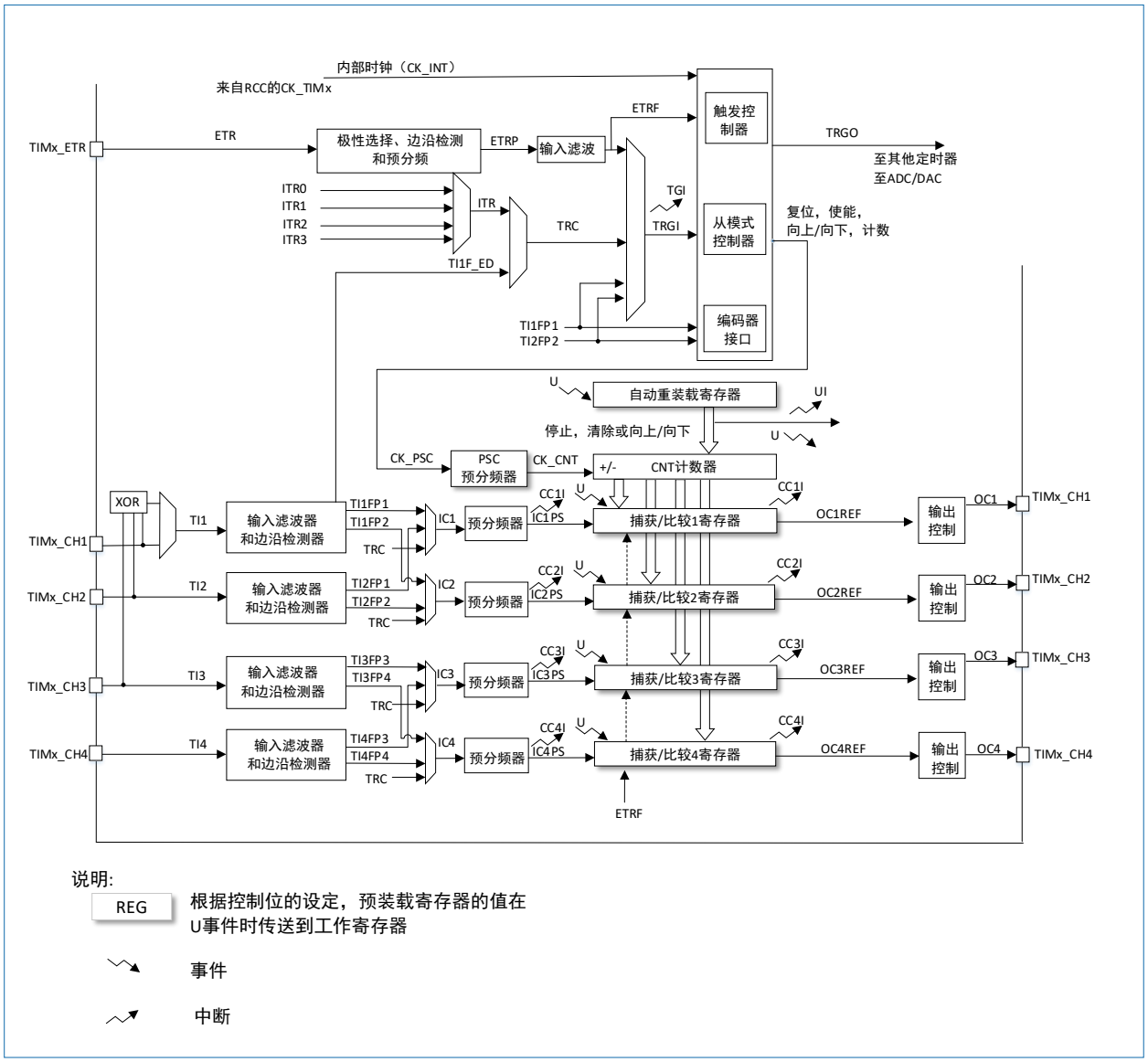


图 17-1 通用定时器框图

上图的说明：TIM3 不具有 DAC 触发功能。

17.2 TIMx 功能

17.2.1 时基单元

可编程通用定时器的主要部分是一个 16 位计数器和与其相关的自动装载寄存器。这个计数器可以向上计数、向下计数或者向上向下双向计数。此计数器时钟由预分频器分频得到。

计数器寄存器、自动装载寄存器和预分频器寄存器可以由软件读写，在计数器运行时仍可以读写。时基单元包含：

- 计数器寄存器（TIMx_CNT）
- 预分频器寄存器（TIMx_PSC）
- 自动重装载寄存器（TIMx_ARR）

自动重装载寄存器是预先装载的，写或读自动重装载寄存器将访问预装载寄存器。根据在 TIMx_CR1 寄存器中的自动装载预装载使能位（ARPE）的设置，预装载寄存器的内容被立即或在每次的更新事件 UEV 时传送到影子寄存器。当计数器达到溢出条件（向下计数时的下溢条件）并当TIMx_CR1

寄存器中的 UDIS 位等于’0’时，产生更新事件。更新事件也可以由软件产生。随后会详细描述每一种配置下更新事件的产生。

计数器由预分频器的时钟输出 CK_CNT 驱动，只有当设置了计数器 TIMx_CR1 寄存器中的计数器使能位（CEN）时，CK_CNT 才有效。（有关计数器使能的细节，请参见控制器的从模式描述）。

注意：真正的计数器使能信号 CNT_EN 是在 CEN 的一个时钟周期后被设置。

17.2.1.1 预分频器描述

预分频器可以将计数器的时钟频率按 1 到 65536 之间的任意值分频。它是基于一个（在 TIMx_PSC 寄存器中的）16 位寄存器控制的 16 位计数器。这个控制寄存器带有缓冲器，它能够在工作时被改变。新的预分频器参数在下一次更新事件到来时被采用。

下面两个图给出了在预分频器运行时，更改计数器参数的例子。

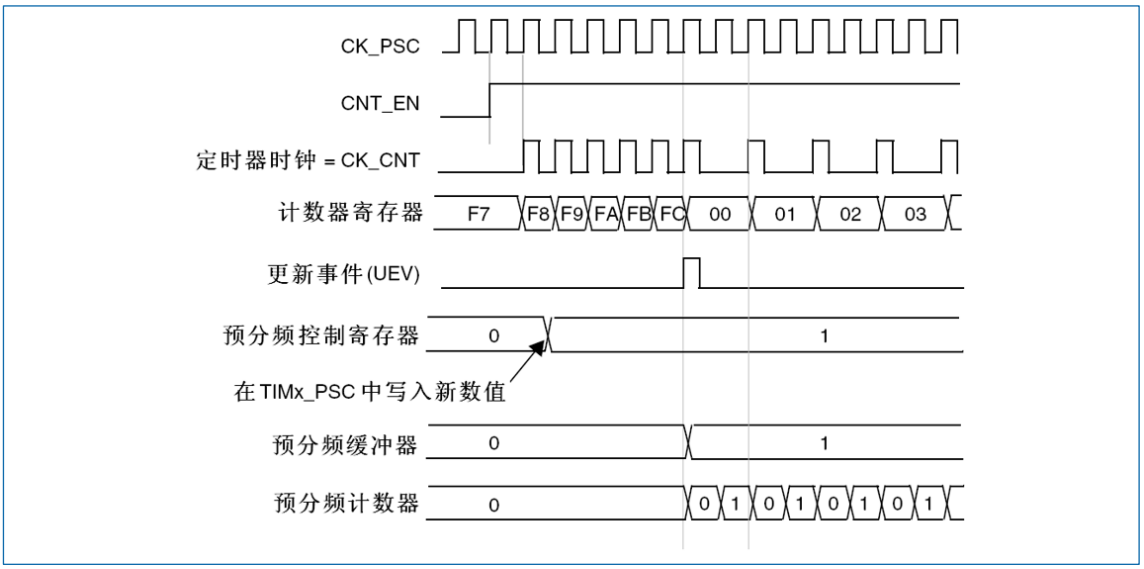


图 17-2 当预分频器的参数从 1 变到 2 时，计数器的时序图

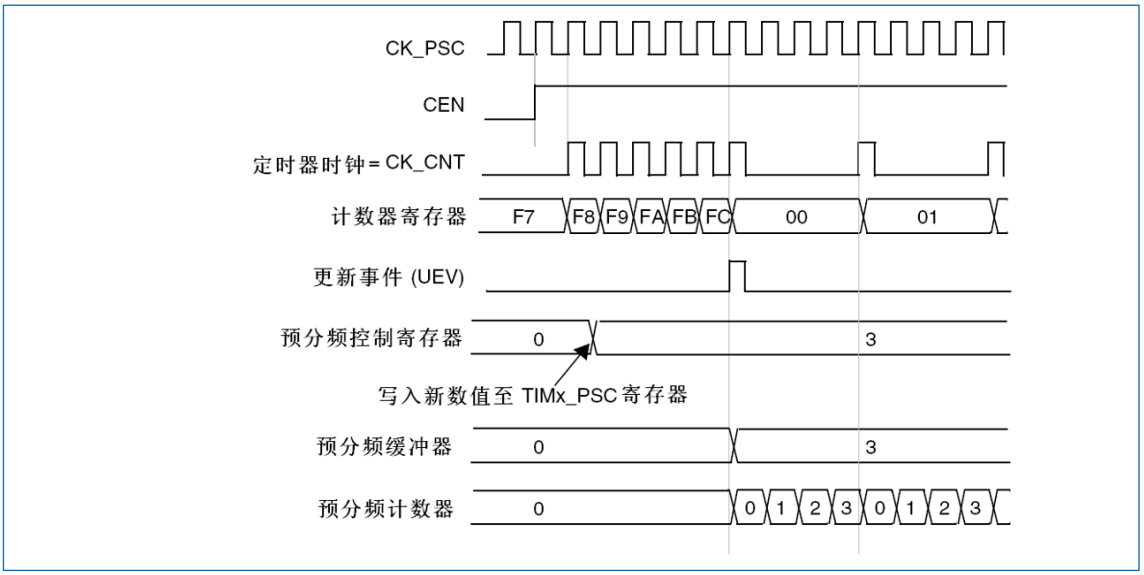


图 17-3 当预分频器的参数从 1 变到 4 时，计数器的时序图

17.2.2 计数器模式

17.2.2.1 向上计数模式

在向上计数模式中，计数器从 0 计数到自动加载值（TIMx_ARR 计数器的值），然后重新从 0 开始计数并且产生一个计数器溢出事件。

每次计数器溢出时可以产生更新事件，在 TIMx_EGR 寄存器中（通过软件方式或者使用从模式控制器）设置 UG 位也同样可以产生一个更新事件。设置 TIMx_CR1 寄存器中的 UDIS 位，可以禁止更新事件；这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。在 UDIS 位被清零之前，将不产生更新事件。但是在应该产生更新事件时，计数器仍会被清零，同时预分频器的计数也被清零（但预分频系数不变）。此外，如果设置了 TIMx_CR1 寄存器中的 URS 位（选择更新请求），设置 UG 位将产生一个更新事件 UEV，但硬件不设置 UIF 标志（即不产生中断或 DMA 请求）；这是为了避免在捕获模式下清除计数器时，同时产生更新和捕获中断。

当发生一个更新事件时，所有的寄存器都被更新，且硬件同时（依据 URS 位）设置更新标志位（TIMx_SR 寄存器中的 UIF 位）。

- 预分频器的缓冲区被置入预装载寄存器的值（TIMx_PSC 寄存器的内容）。
- 自动装载影子寄存器被重新置入预装载寄存器的值（TIMx_ARR）。

下图给出一些例子，当 TIMx_ARR=0x36 时计数器在不同时钟频率下的动作。

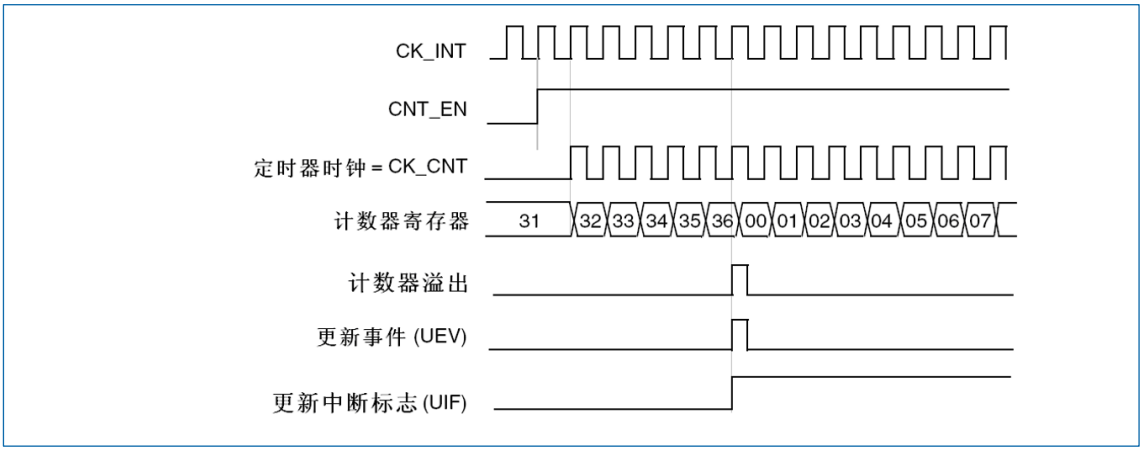


图 17-4 计数器时序图，内部时钟分频因子为 1

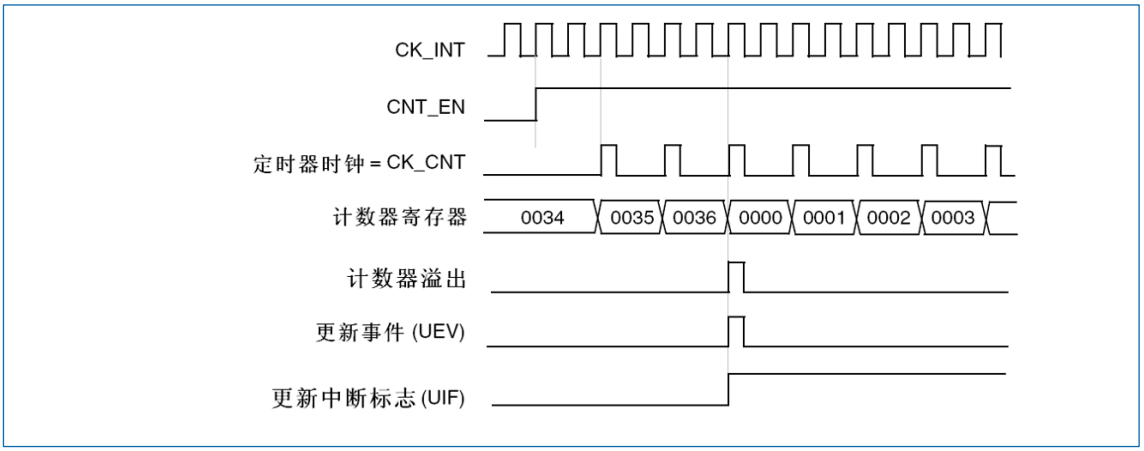


图 17-5 计数器时序图，内部时钟分频因子为 2

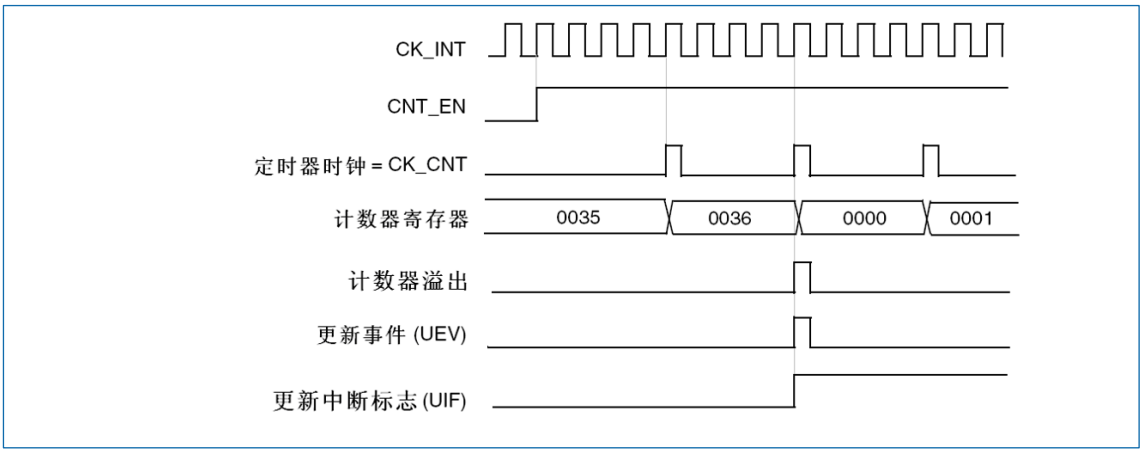


图 17-6 计数器时序图，内部时钟分频因子为 4

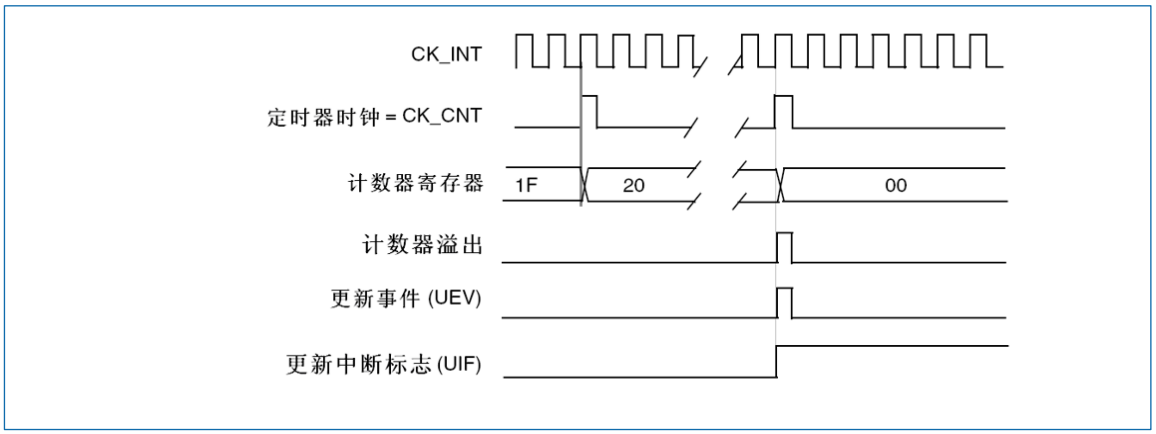


图 17-7 计数器时序图，内部时钟分频因子为 N

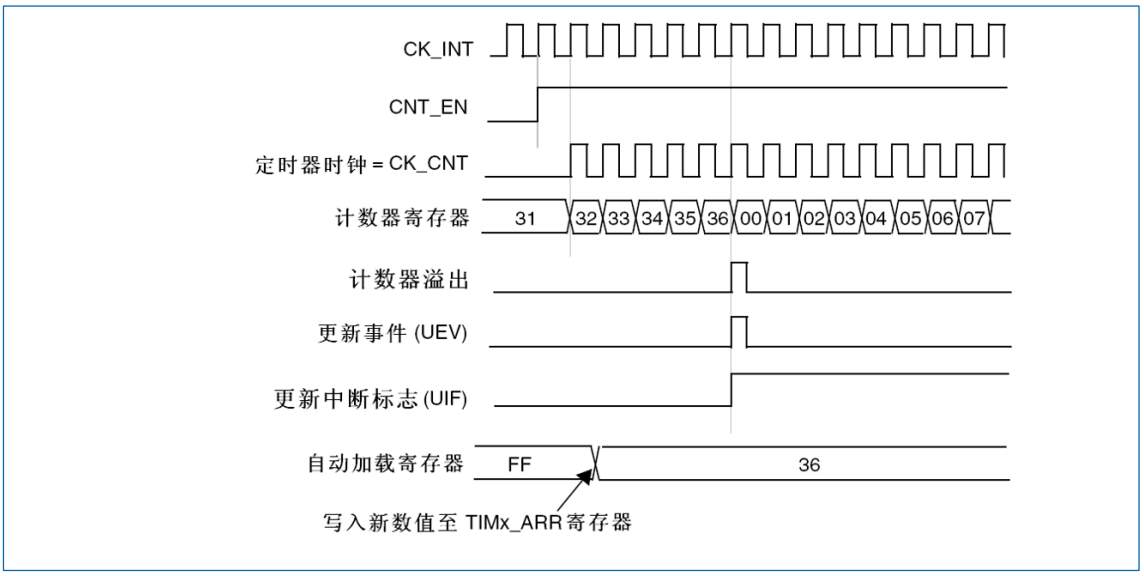


图 17-8 计数器时序图，当 ARPE=0 时的更新事件（TIMx_ARR 没有预装入）

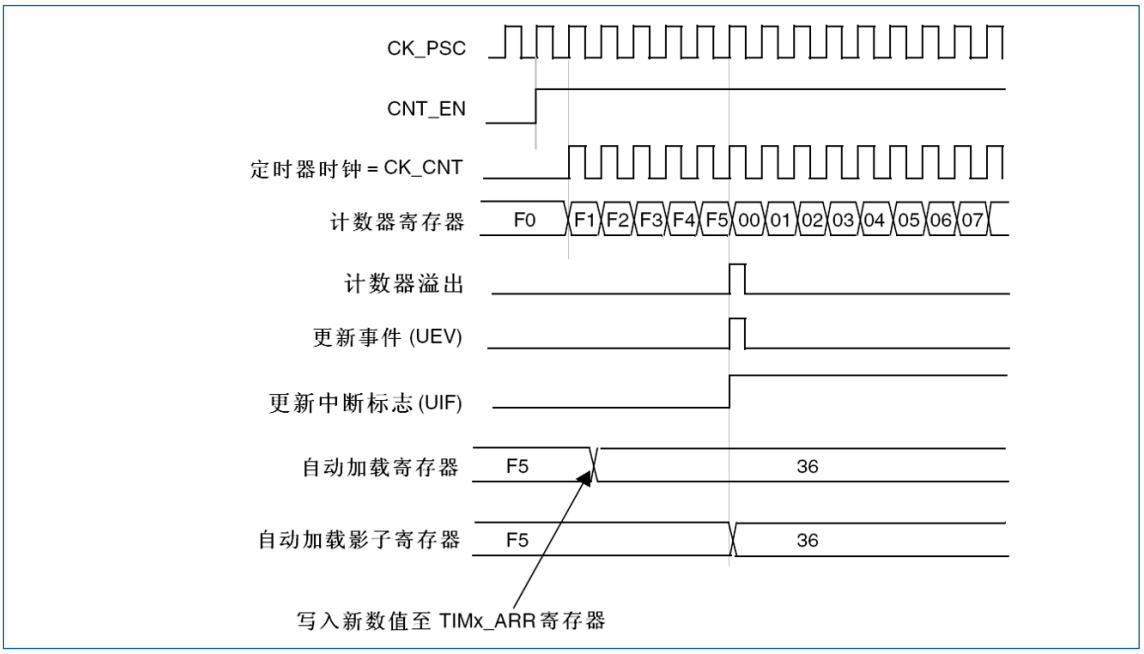


图 17-9 计数器时序图，当 ARPE=1 时的更新事件（预装入了 TIMx_ARR）

17.2.2.2 向下计数模式

在向下模式中，计数器从自动装入的值（TIMx_ARR 计数器的值）开始向下计数到 0，然后从自动装入的值重新开始并且产生一个计数器向下溢出事件。

每次计数器溢出时可以产生更新事件，在 TIMx_EGR 寄存器中（通过软件方式或者使用从模式控制器）设置 UG 位，也同样可以产生一个更新事件。

设置 TIMx_CR1 寄存器的 UDIS 位可以禁止 UEV 事件。这样可以避免向预装载寄存器中写入新值时更新影子寄存器。因此 UDIS 位被清零之前不会产生更新事件。然而，计数器仍会从当前自动加载值重新开始计数，同时预分频器的计数器重新从 0 开始（但预分频系数不变）。

此外，如果设置了 TIMx_CR1 寄存器中的 URS 位（选择更新请求），设置 UG 位将产生一个更新事件 UEV 但不设置 UIF 标志（因此不产生中断和 DMA 请求）。这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

当发生更新事件时，所有的寄存器都被更新，并且（根据 URS 位的设置）更新标志位（TIMx_SR 寄存器中的 UIF 位）也被设置。

- 预分频器的缓冲器被置入预装载寄存器的值（TIMx_PSC 寄存器的值）。
- 当前的自动加载寄存器被更新为预装载值（TIMx_ARR 寄存器中的内容）。

注意：自动装载在计数器重载入之前被更新，因此下一个周期将是预期的值。

以下是一些当 TIMx_ARR=0x36 时，计数器在不同时钟频率下的操作例子。

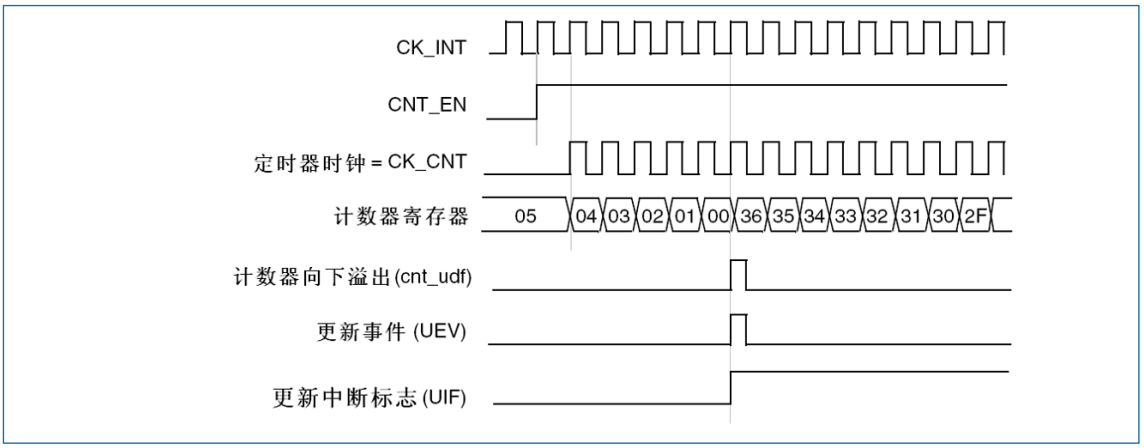


图 17-10 计数器时序图，内部时钟分频因子为 1

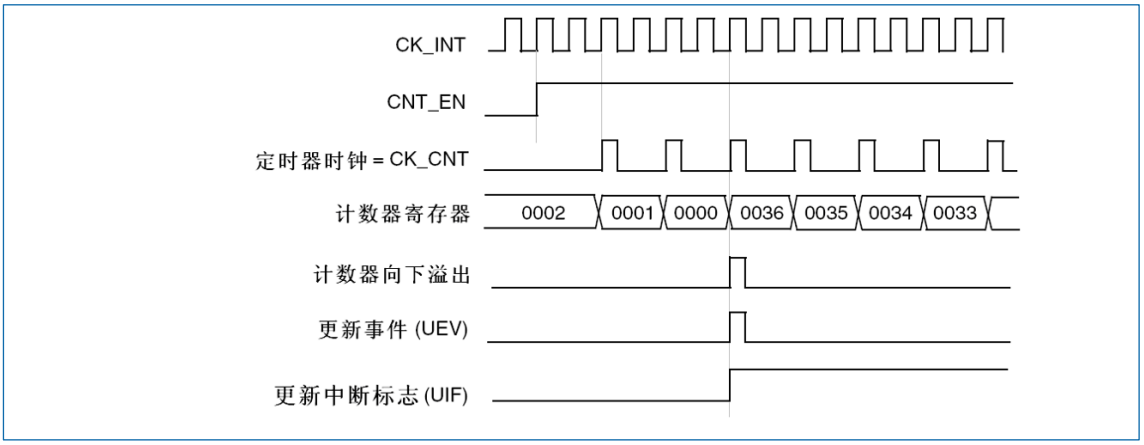


图 17-11 计数器时序图，内部时钟分频因子为 2

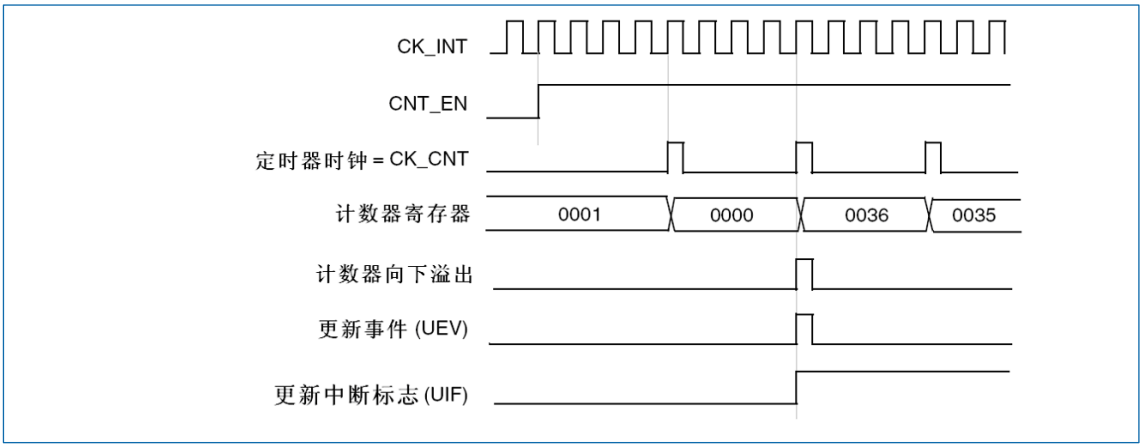


图 17-12 计数器时序图，内部时钟分频因子为 4

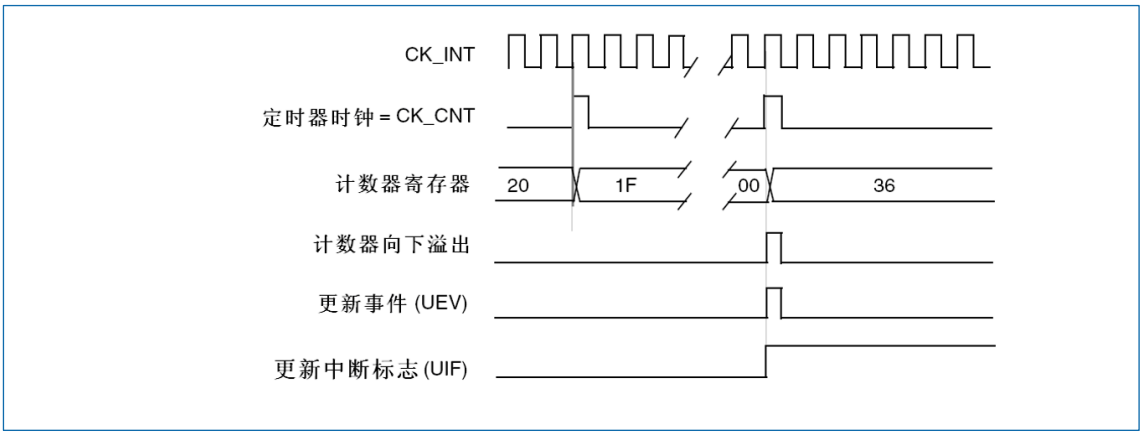


图 17-13 计数器时序图，内部时钟分频因子为 N

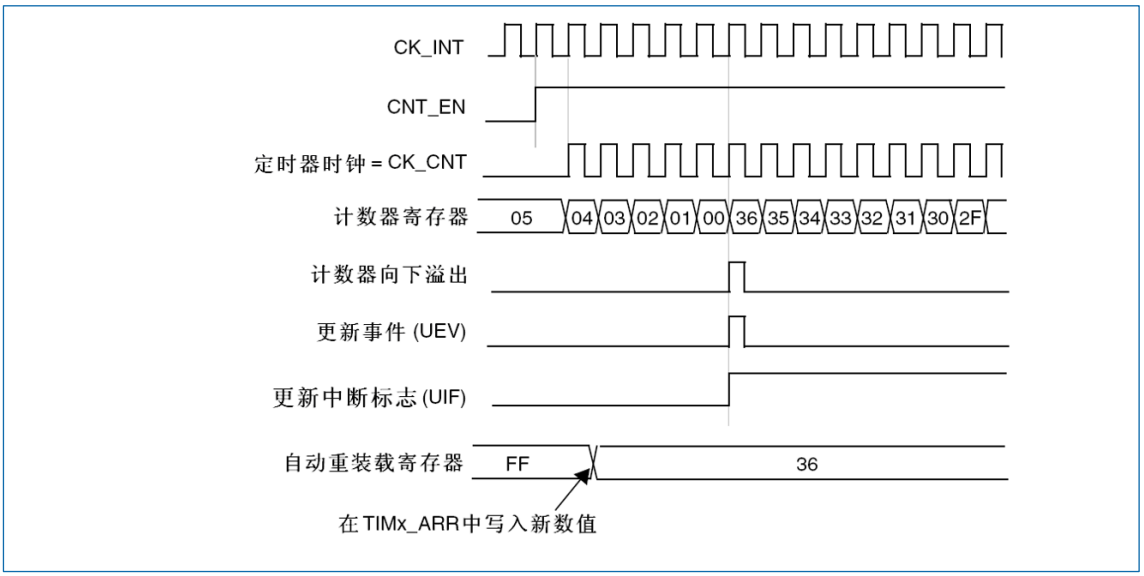


图 17-14 计数器时序图，当没有使用重复计数器时的更新事件

17.2.2.3 中央对齐模式（向上/向下计数）

在中央对齐模式，计数器从 0 开始计数到自动加载的值减 1（TIMx_ARR 寄存器-1），产生一个计数器溢出事件，然后向下计数到 1 并且产生一个计数器下溢事件；然后再从 0 开始重新计数。

在这个模式，不能写入 TIMx_CR1 中的 DIR 方向位。它由硬件更新并指示当前的计数方向。

可以在每次计数上溢和每次计数下溢时产生更新事件；也可以通过（软件或者使用从模式控制器）设置 TIMx_EGR 寄存器中的 UG 位产生更新事件。然后，计数器重新从 0 开始计数，预分频器也重新从 0 开始计数。

设置 TIMx_CR1 寄存器中的 UDIS 位可以禁止 UEV 事件。这样可以避免在向预装载寄存器中写入新值时更新影子寄存器。因此 UDIS 位被清为'0'之前不会产生更新事件。然而，计数器仍会根据当前自动重加载的值，继续向上或向下计数。

此外，如果设置了 TIMx_CR1 寄存器中的 URS 位（选择更新请求），设置 UG 位将产生一个更新事件 UEV 但不设置 UIF 标志（因此不产生中断和 DMA 请求），这是为了避免在发生捕获事件并清除计数器时，同时产生更新和捕获中断。

当发生更新事件时，所有的寄存器都被更新，并且（根据 URS 位的设置）更新标志位（TIMx_SR 寄存器中的 UIF 位）也被设置。

- 预分频器的缓存器被加载为预装载（TIMx_PSC 寄存器）的值。
- 当前的自动加载寄存器被更新为预装载值（TIMx_ARR 寄存器中的内容）。

注意：如果因为计数器溢出而产生更新，自动重装载将在计数器重载入之前被更新，因此下一个周期将是预期的值（计数器被装载为新的值）。

以下是一些计数器在不同时钟频率下的操作的例子：

图 17-15 使用了中央对齐模式 1（详见 TIMx_CR1 章节）。

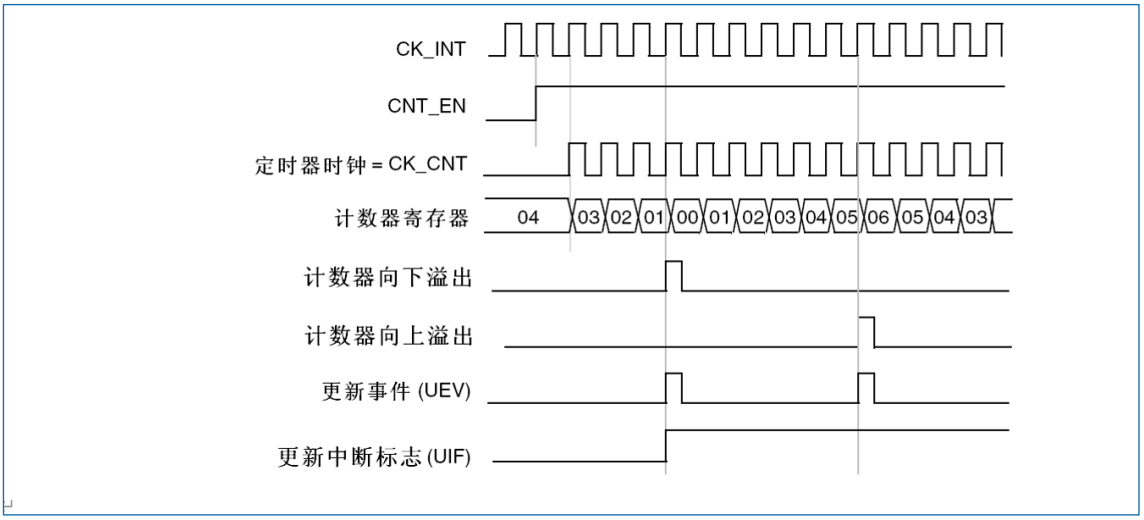


图 17-15 计数器时序图，内部时钟分频因子为 1，TIMx_ARR=0x6

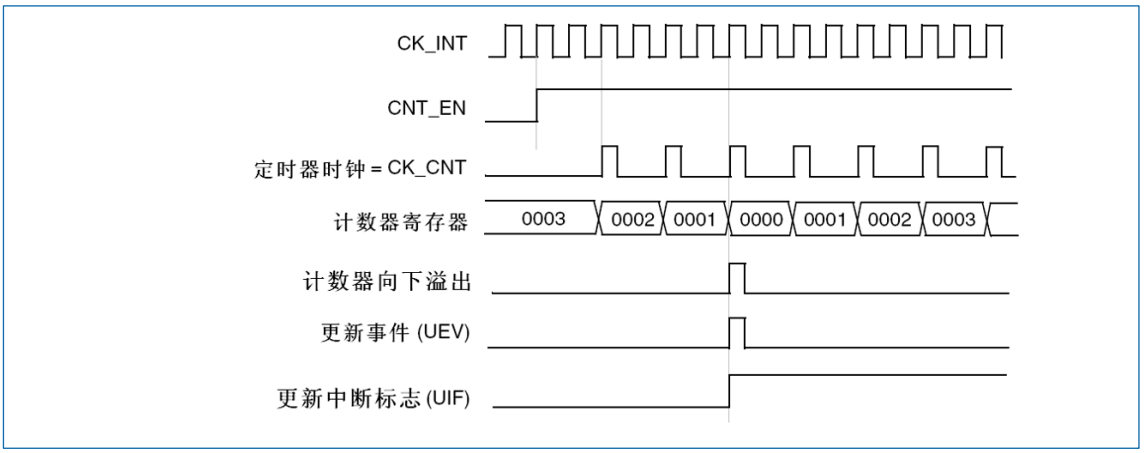


图 17-16 计数器时序图，内部时钟分频因子为 2

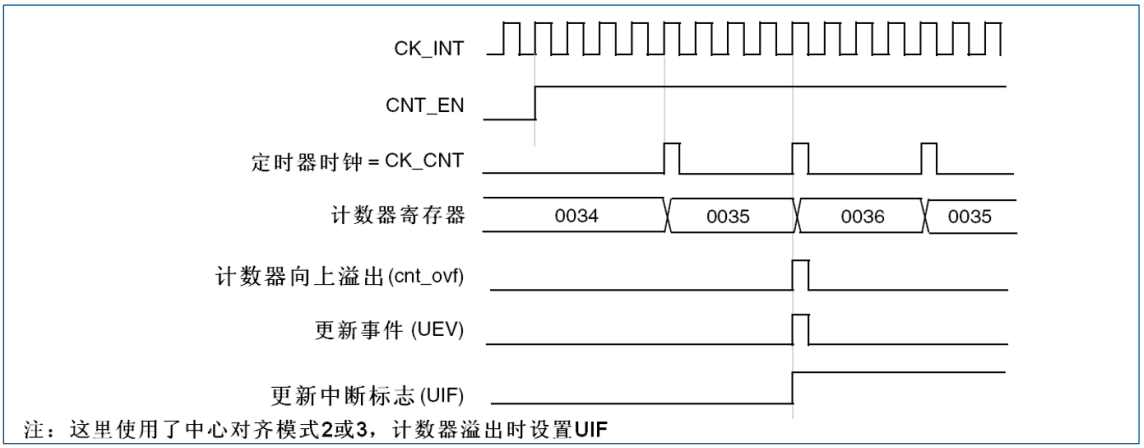


图 17-17 计数器时序图，内部时钟分频因子为 4，TIMx_ARR=0x36

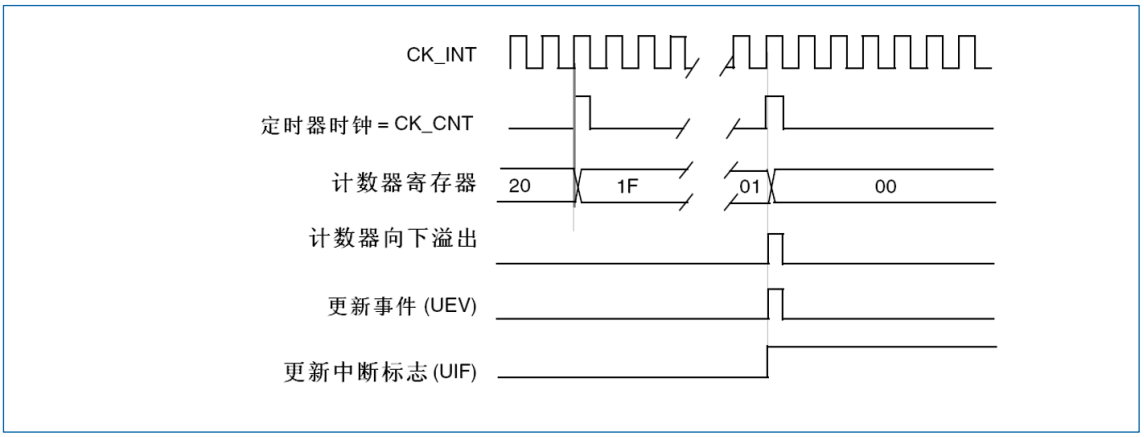


图 17-18 计数器时序图，内部时钟分频因子为 N

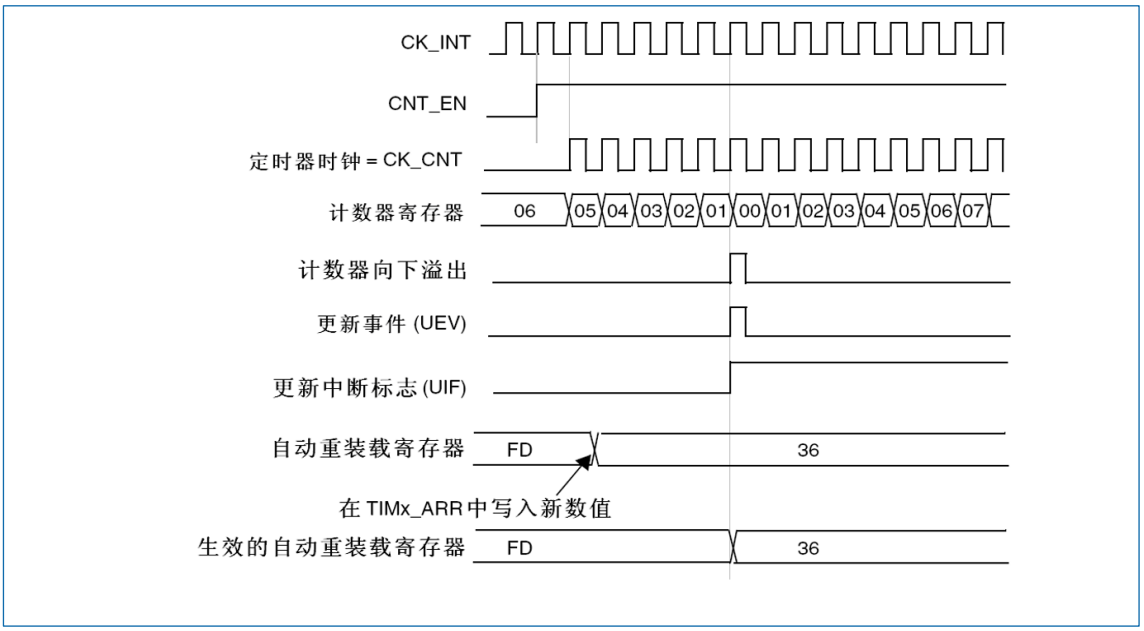


图 17-19 计数器时序图，ARPE=1 时的更新事件（计数器下溢）

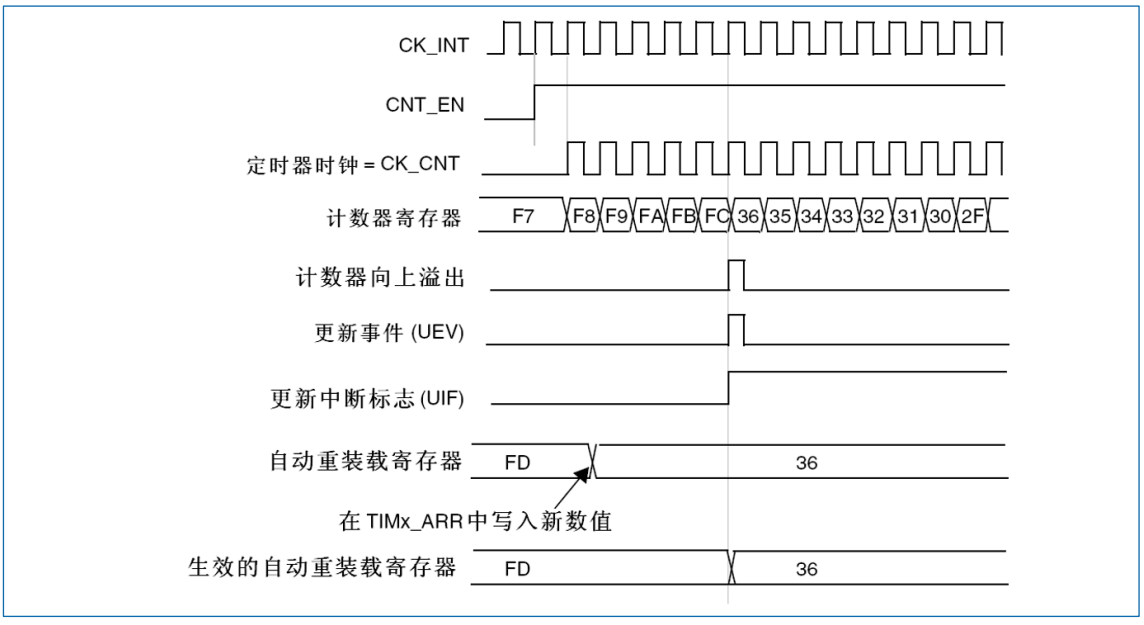


图 17-20 计数器时序图，ARPE=1 时的更新事件（计数器溢出）

17.2.3 时钟选择

计数器时钟可由下列时钟源提供：

- 内部时钟（CK_INT）
- 外部时钟模式 1：外部输入脚（TIx）
- 外部时钟模式 2：外部触发输入（ETR）
- 内部触发输入（ITRx）：使用一个定时器作为另一个定时器的预分频器，如可以配置一个定时器 Timer1 而作为另一个定时器 Timer2 的预分频器。参见：“定时器同步”。

17.2.3.1 内部时钟源（CK_INT）

如果禁止了从模式控制器（TIMx_SMCR 寄存器的 SMS=000），则 CEN、DIR（TIMx_CR1 寄存器）和 UG 位（TIMx_EGR 寄存器）是实际控制位，并且只能被软件修改（UG 位仍被自动清除）。只要 CEN 位被写成’1’，预分频器的时钟就由内部时钟 CK_INT 提供。

下图显示了控制电路和向上计数器在一般模式下，不带预分频器时的操作。

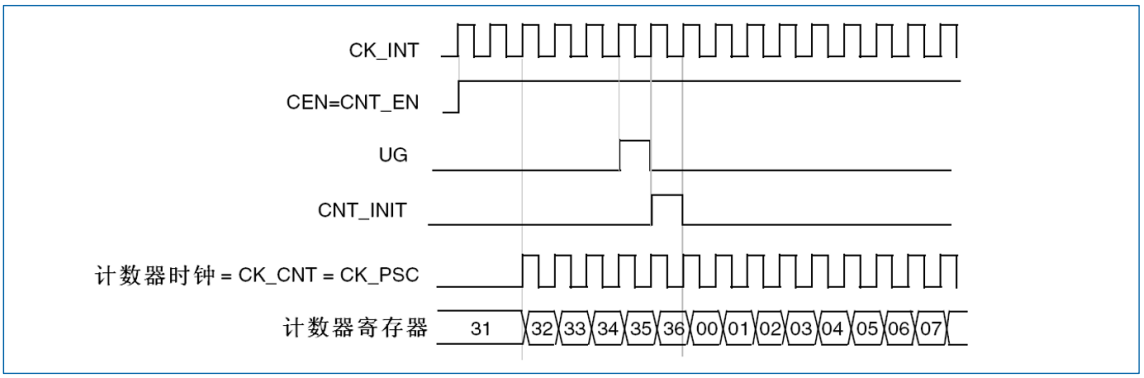


图 17-21 一般模式下的控制电路，内部时钟分频因子为 1

17.2.3.2 外部时钟源模式 1

当 TIMx_SMCR 寄存器的 SMS=111 时，此模式被选中。计数器可以在选定输入端的每个上升沿或下降沿计数。

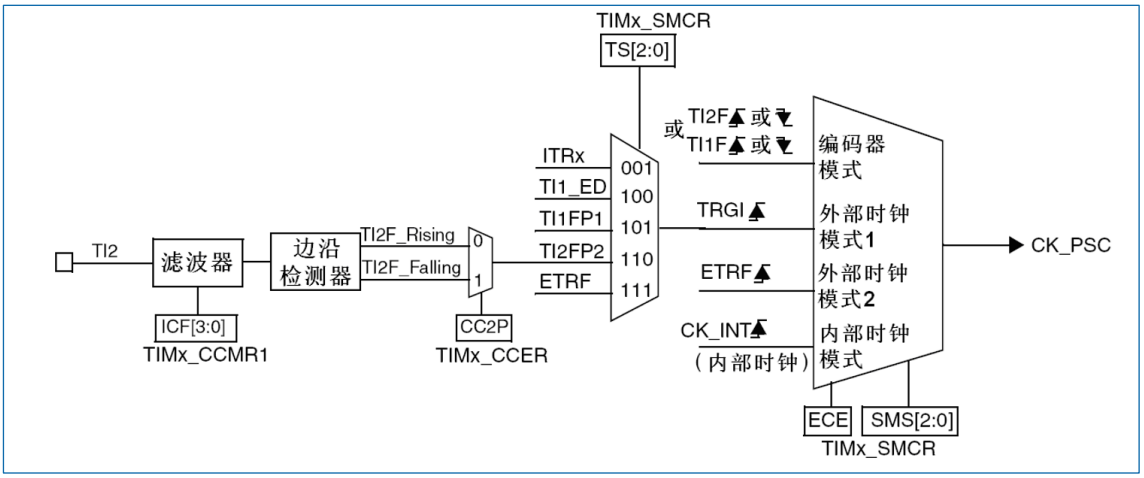


图 17-22 TI2 外部时钟连接例子

例如，要配置向上计数器在 T12 输入端的上升沿计数，使用下列步骤：

1. 配置 TIMx_CCMR1 寄存器 CC2S=’01’，配置通道 2 检测 TI2 输入的上升沿。
2. 配置 TIMx_CCMR1 寄存器的 IC2F[3:0]，选择输入滤波器带宽（如果不需要滤波器，保持

IC2F=0000）。

注意：捕获预分频器不用作触发，所以不需要对它进行配置。

3. 配置 TIMx_CCER 寄存器的 CC2P=’0’，选定上升沿极性。
4. 配置 TIMx_SMCR 寄存器的 SMS=’111’，选择定时器外部时钟模式 1。
5. 配置 TIMx_SMCR 寄存器中的 TS=’110’，选定 TI2 作为触发输入源。
6. 设置 TIMx_CR1 寄存器的 CEN=’1’，启动计数器。

当上升沿出现在 TI2，计数器计数一次，且 TIF 标志被设置。在 TI2 的上升沿和计数器实际时钟之间的延时，取决于在 TI2 输入端的重新同步电路。

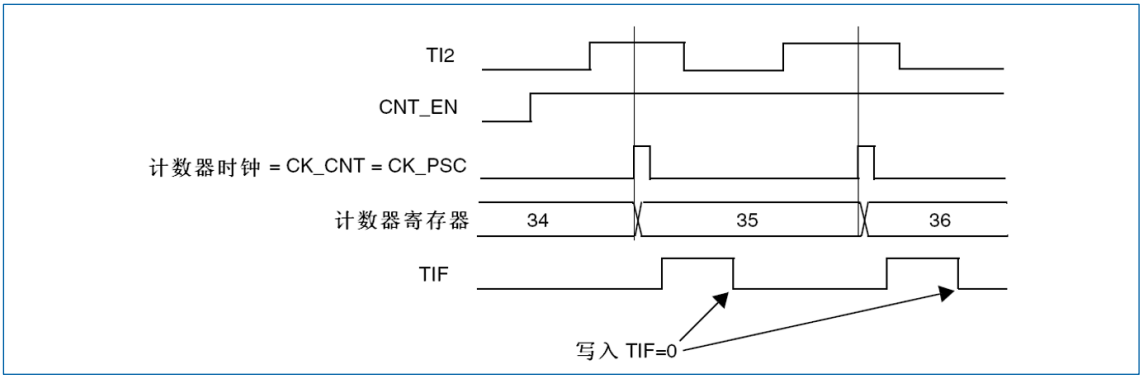


图 17-23 外部时钟模式 1 下的控制电路

17.2.3.3 外部时钟源模式 2

选定此模式的方法为：设置 TIMx_SMCR 寄存器中的 ECE=1。

计数器能够在外部触发 ETR 的每一个上升沿或下降沿计数。

下图是外部触发输入的框图。

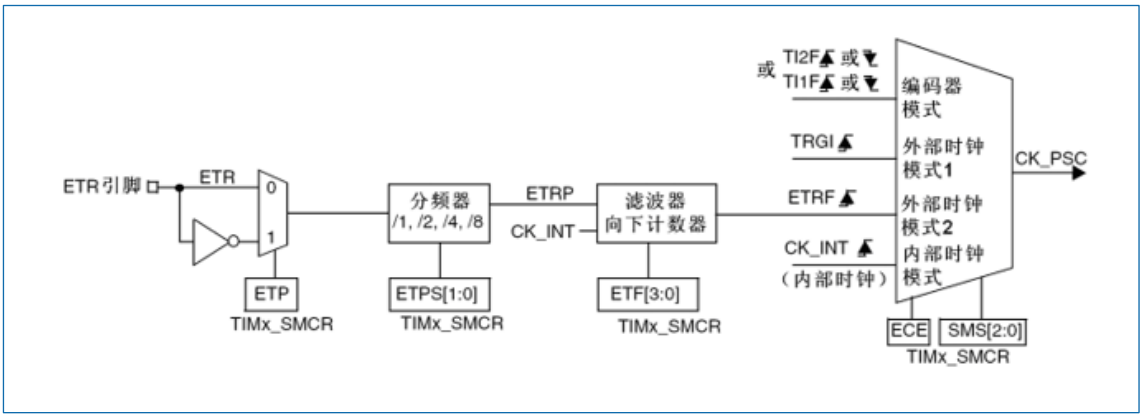


图 17-24 外部触发输入框图

例如，要配置在 ETR 下每 2 个上升沿计数一次的向上计数器，使用下列步骤：

1. 本例中不需要滤波器，置 TIMx_SMCR 寄存器中的 ETF[3:0]=0000。
2. 设置预分频器，置 TIMx_SMCR 寄存器中的 ETPS[1:0]=01。
3. 设置在 ETR 的上升沿检测，置 TIMx_SMCR 寄存器中的 ETP=0。
4. 开启外部时钟模式 2，置 TIMx_SMCR 寄存器中的 ECE=1。

5. 启动计数器，置 TIMx_CR1 寄存器中的 CEN=1。计数器在每 2 个 ETR 上升沿计数一次。

在 ETR 的上升沿和计数器实际时钟之间的延时取决于在 ETRP 信号端的重新同步电路。

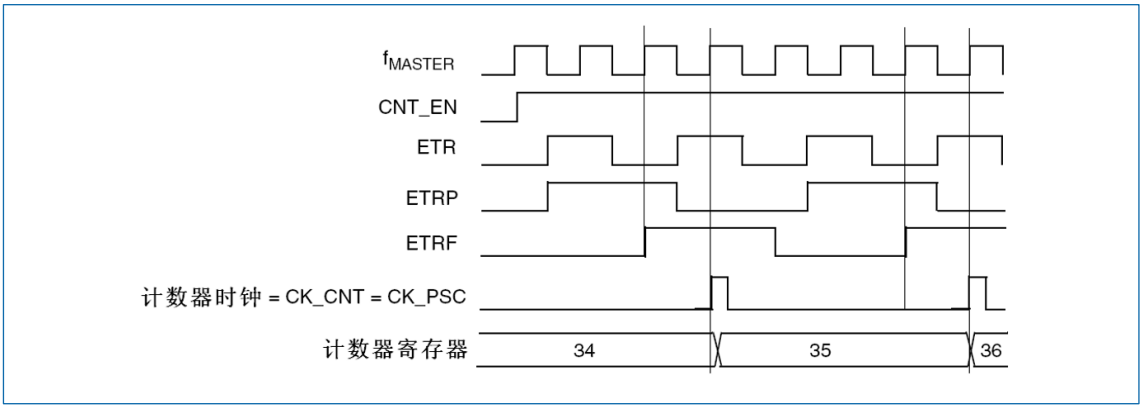


图 17-25 外部时钟模式 2 下的控制电路

17.2.4 捕获/比较通道

每一个捕获/比较通道都是围绕着一个捕获/比较寄存器（包含影子寄存器），包括捕获的输入部分（数字滤波、多路复用和预分频器），和输出部分（比较器和输出控制）。

下面几张图是一个捕获/比较通道概览。

输入部分对相应的 TIx 输入信号采样，并产生一个滤波后的信号 TIxF。然后，一个带极性选择的边沿检测器产生一个信号（TIxFPx），它可以作为从模式控制器的输入触发或者作为捕获控制。该信号通过预分频进入捕获寄存器（ICxPS）。

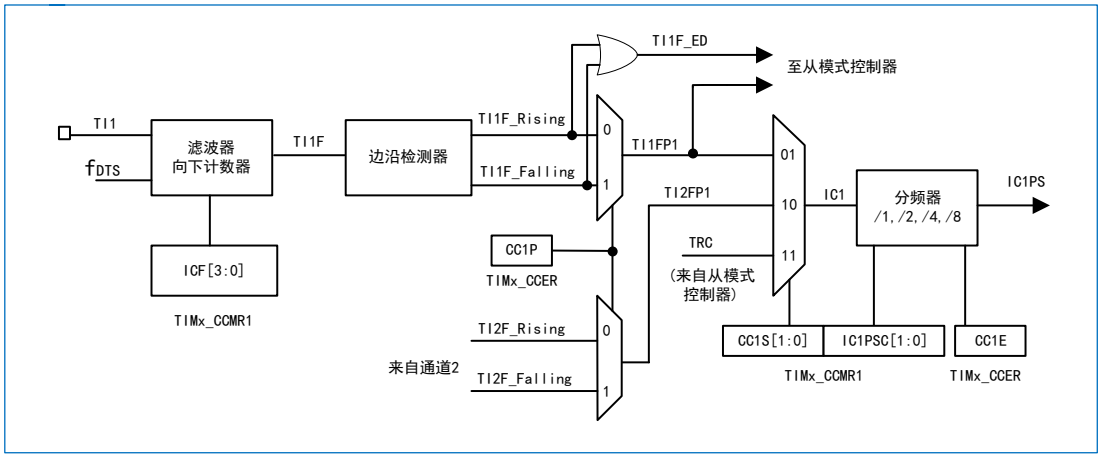


图 17-26 捕获/比较通道（如：通道 1 输入部分）

输出部分产生一个中间波形 OCxRef（高有效）作为基准，链的末端决定最终输出信号的极性。

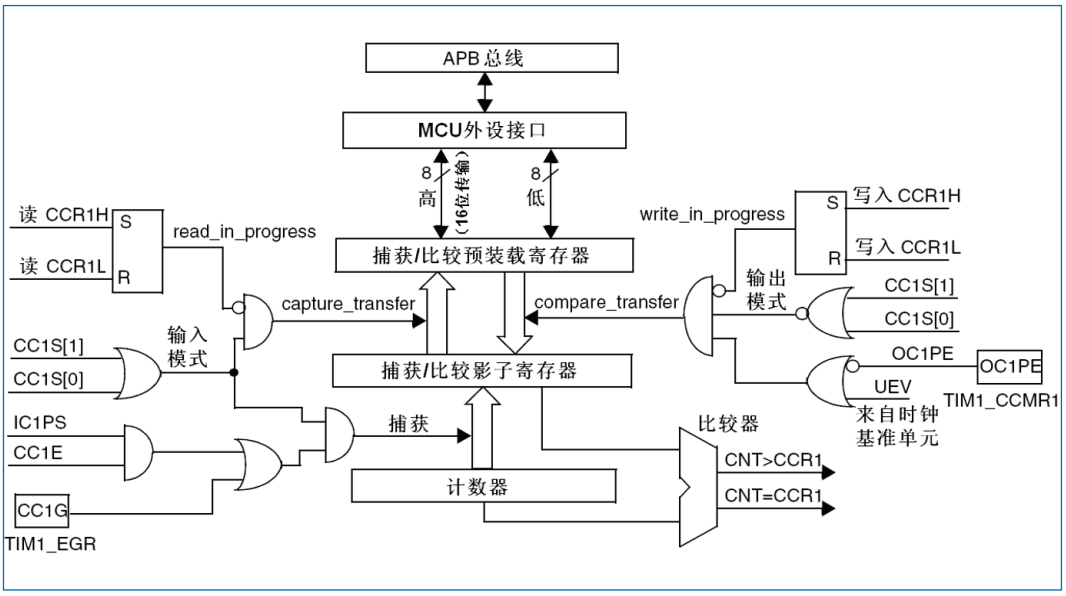


图 17-27 捕获/比较通道 1 的主电路

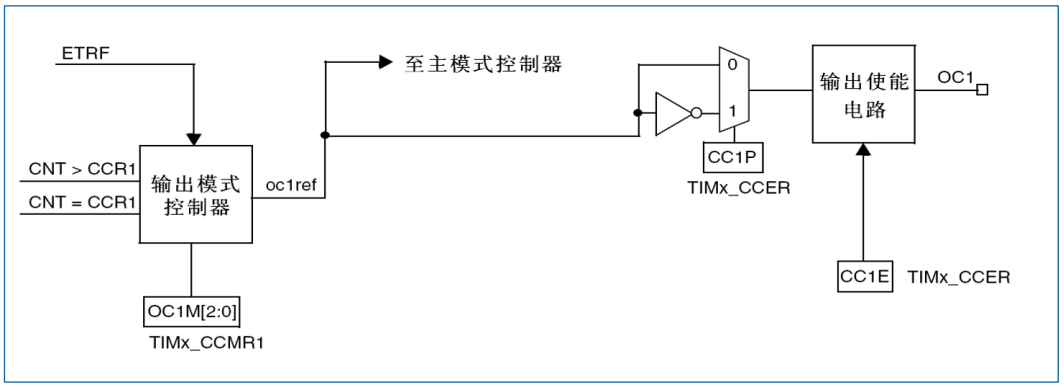


图 17-28 捕获/比较通道的输出部分（通道 1）

捕获/比较模块由一个预装载寄存器和一个影子寄存器组成。读写过程仅操作预装载寄存器。

在捕获模式下，捕获发生在影子寄存器上，然后再复制到预装载寄存器中。

在比较模式下，预装载寄存器的内容被复制到影子寄存器中，然后影子寄存器的内容和计数器进行比较。

17.2.5 输入捕获模式

在输入捕获模式下，当检测到 ICx 信号上相应的边沿后，计数器的当前值被锁存到捕获/比较寄存器（TIMx_CCRx）中。当捕获事件发生时，相应的 CCxIF 标志（TIMx_SR 寄存器）被置'1'，如果使能了中断或者 DMA 操作，则将产生中断或者 DMA 操作。如果捕获事件发生时 CCxIF 标志已经为高，那么重复捕获标志 CCxOF（TIMx_SR 寄存器）被置'1'。写 CCxIF=0 可清除 CCxIF，或读取存储在 TIMx_CCRx 寄存器中的捕获数据也可清除 CCxIF。写 CCxOF=0 可清除 CCxOF。

以下例子说明如何在 TI1 输入的上升沿时捕获计数器的值到 TIMx_CCR1 寄存器中，步骤如下：

1. 选择有效输入端：TIMx_CCR1 必须连接到 TI1 输入，所以写入 TIMx_CCMR1 寄存器中的 CC1S=01，只要 CC1S 不为'00'，通道被配置为输入，并且 TM1_CCR1 寄存器变为只读。

2. 根据输入信号的特点，配置输入滤波器为所需的带宽（即输入为 TIx 时，输入滤波器控制位是 TIMx_CCMRx 寄存器中的 ICxF 位）。假设输入信号在最多 5 个内部时钟周期的时间内抖动，我们须配置滤波器的带宽长于 5 个时钟周期。因此可以（以 f_{DTS} 频率）连续采样 8 次，以确认在 TI1 上一次真实的边沿变换，即在 TIMx_CCMR1 寄存器中写入 IC1F=0011。

3. 选择 TI1 通道的有效转换边沿，在 TIMx_CCER 寄存器中写入 CC1P=0（上升沿）。
4. 配置输入预分频器。在本例中，希望捕获发生在每一个有效的电平转换时刻，因此预分频器被禁止（写 TIMx_CCMR1 寄存器的 IC1PS=00）。
5. 设置 TIMx_CCER 寄存器的 CC1E=1，允许捕获计数器的值到捕获寄存器中。
6. 如果需要，通过设置 TIMx_DIER 寄存器中的 CC1IE 位允许相关中断请求，通过设置 TIMx_DIER 寄存器中的 CC1DE 位允许 DMA 请求。

当发生一个输入捕获时：

- 产生有效的电平转换时，计数器的值被传送到 TIMx_CCR1 寄存器。
- CC1IF 标志被设置（中断标志）。当发生至少 2 个连续的捕获时，而 CC1IF 未曾被清除，CC1OF 也被置'1'。
- 如设置了 CC1IE 位，则会产生一个中断。
- 如设置了 CC1DE 位，则还会产生一个 DMA 请求。

为了处理捕获溢出，建议在读出捕获溢出标志之前读取数据，这是为了避免丢失在读出捕获溢出标志之后和读取数据之前可能产生的捕获溢出信息。

注意：设置 TIMx_EGR 寄存器中相应的 CCxG 位，可以通过软件产生输入捕获中断和/或 DMA 请求。

17.2.6 PWM 输入模式

该模式是输入捕获模式的一个特例，除下列区别外，操作与输入捕获模式相同：

- 两个 ICx 信号被映射至同一个 TIx 输入。
- 这 2 个 ICx 信号为边沿有效，但是极性相反。
- 其中一个 TIxFP 信号被作为触发输入信号，而从模式控制器被配置成复位模式。

例如，你需要测量输入到 TI1 上的 PWM 信号的长度（TIMx_CCR1 寄存器）和占空比（TIMx_CCR2寄存器），具体步骤如下（取决于 CK_INT 的频率和预分频器的值）

1. 选择 TIMx_CCR1 的有效输入：置 TIMx_CCMR1 寄存器的 CC1S=01（选择 TI1）。
2. 选择 TI1FP1 的有效极性（用来捕获数据到 TIMx_CCR1 中和清除计数器）：置 CC1P=0（上升沿有效）。
3. 选择 TIMx_CCR2 的有效输入：置 TIMx_CCMR1 寄存器的 CC2S=10（选择 TI1）。
4. 选择 TI1FP2 的有效极性（捕获数据到 TIMx_CCR2）：置 CC2P=1（下降沿有效）。
5. 选择有效的触发输入信号：置 TIMx_SMCR 寄存器中的 TS=101（选择 TI1FP1）。
6. 配置从模式控制器为复位模式：置 TIMx_SMCR 中的 SMS=100。
7. 使能捕获：置 TIMx_CCER 寄存器中 CC1E=1 且 CC2E=1。

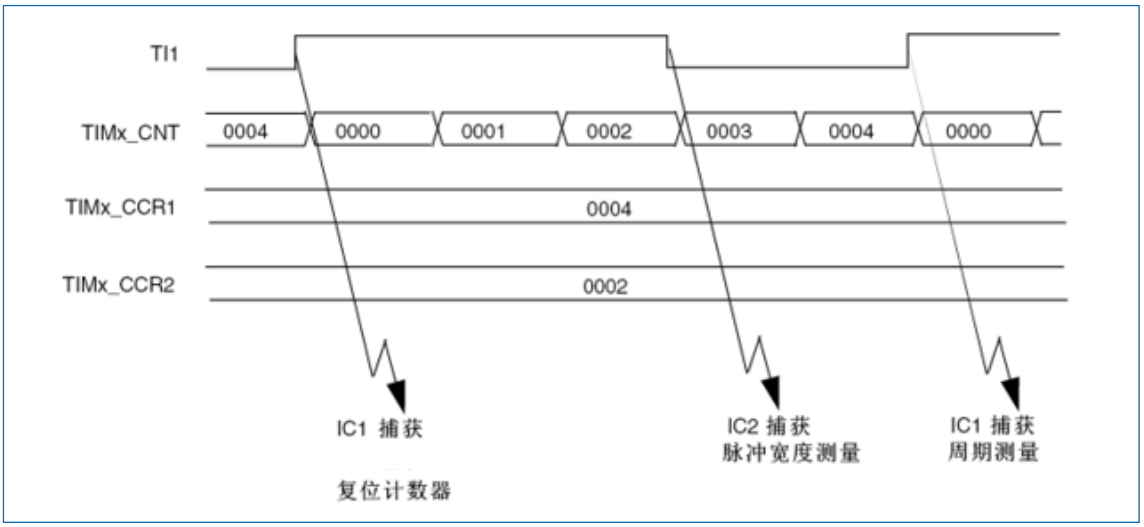


图 17-29 PWM 输入模式时序

由于只有 TI1FP1 和 TI2FP2 连到了从模式控制器，所以 PWM 输入模式只能使用 TIMx_CH1/TIMx_CH2 信号。

17.2.7 强置输出模式

在输出模式（TIMx_CCMRx 寄存器中 CCxS=00）下，输出比较信号（OCxREF 和相应的 OCx）能够直接由软件强置为有效或无效状态，而不依赖于输出比较寄存器和计数器间的比较结果。

置 TIMx_CCMRx 寄存器中相应的 OCxM=101，即可强置输出比较信号（OCxREF/OCx）为有效状态。这样 OCxREF 被强置为高电平（OCxREF 始终为高电平有效），同时 OCx 得到 CCxP 极性相反的值。

例如：CCxP=0（OCx 高电平有效），则 OCx 被强置为高电平。

置 TIMx_CCMRx 寄存器中的 OCxM=100，可强置 OCxREF 信号为低。

该模式下，在 TIMx_CCRx 影子寄存器和计数器之间的比较仍然在进行，相应的标志也会被修改。因此仍然会产生相应的中断和 DMA 请求。这将会在下面的输出比较模式一节中介绍。

17.2.8 输出比较模式

此项功能是用来控制一个输出波形，或者指示一段给定的的时间已经到时。当计数器与捕获/比较寄存器的内容相同时，输出比较功能做如下操作：

- 将输出比较模式（TIMx_CCMRx 寄存器中的 OCxM 位）和输出极性（TIMx_CCER 寄存器中的 CCxP 位）定义的值输出到对应的引脚上。在比较匹配时，输出引脚可以保持它的电平（OCxM=000）、被设置成有效电平（OCxM=001）、被设置成无效电平（OCxM=010）或进行翻转（OCxM=011）。
- 设置中断状态寄存器中的标志位（TIMx_SR 寄存器中的 CCxIF 位）。
- 若设置了相应的中断使能（TIMx_DIER 寄存器中的 CCxIE 位），则产生一个中断。
- 若设置了相应的使能位（TIMx_DIER 寄存器中的 CCxDE 位，TIMx_CR2 寄存器中的 CCDS 位选择 DMA 请求功能），则产生一个 DMA 请求。

TIMx_CCMRx 中的 OCxPE 位选择 TIMx_CCRx 寄存器是否需要使用预装载寄存器。

在输出比较模式下，更新事件 UEV 对 OCxREF 和 OCx 输出没有影响。

同步的精度可以达到计数器的一个计数周期。输出比较模式（在单脉冲模式下）也能用来输出一个单脉冲。

输出比较模式的配置步骤：

1. 选择计数器时钟（内部、外部、预分频器）。
2. 将相应的数据写入 TIMx_ARR 和 TIMx_CCRx 寄存器中。
3. 如果要产生一个中断请求和/或一个 DMA 请求，设置 CCxIE 位和/或 CCxDE 位。
4. 选择输出模式，例如当计数器 CNT 与 CCRx 匹配时翻转 OCx 的输出引脚，CCRx 预装载未用，开启 OCx 输出且高电平有效，则必须设置 OCxM='011'、OCxPE='0'、CCxP='0'和 CCxE='1'。
5. 设置 TIMx_CR1 寄存器的 CEN 位启动计数器。

TIMx_CCRx 寄存器能够在任何时候通过软件进行更新以控制输出波形，条件是未使用预装载寄存器（OCxPE='0'，否则 TIMx_CCRx 的影子寄存器只能在发生下一次更新事件时被更新）。下图给出了一个例子。

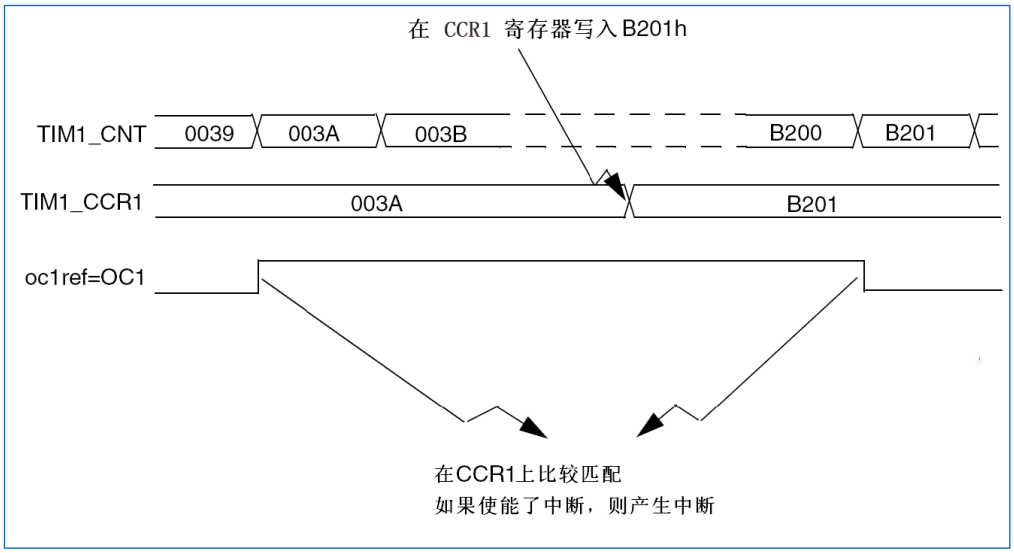


图 17-30 输出比较模式，翻转 OC1

17.2.9 PWM 模式

脉冲宽度调制（PWM）模式可以产生一个由 TIMx_ARR 寄存器确定频率、由 TIMx_CCRx 寄存器确定占空比的信号。

在 TIMx_CCMRx 寄存器中的 OCxM 位写入'110'（PWM 模式 1）或'111'（PWM 模式 2），能够独立地设置每个 OCx 输出通道产生一路 PWM。必须设置 TIMx_CCMRx 寄存器 OCxPE 位以使能相应的预装载寄存器，最后还要设置 TIMx_CR1 寄存器的 ARPE 位，（在向上计数或中心对称模式中）使能自动重装载的预装载寄存器。

仅当发生一个更新事件的时候，预装载寄存器才能被传送到影子寄存器，因此在计数器开始计数之前，必须通过设置 TIMx_EGR 寄存器中的 UG 位来初始化所有的寄存器。OCx 的极性可以通过软件在 TIMx_CCER 寄存器中的 CCxP 位设置，它可以设置为高电平有效或低电平有效。TIMx_CCER 寄存器中的 CCxE 位控制 OCx 输出使能。详见 TIMx_CCERx 寄存器的描述。

在 PWM 模式（模式 1 或模式 2）下，TIMx_CNT 和 TIMx_CCRx 始终在进行比较，（依据计数器的计数方向）以确定是否符合TIMx_CCRx≤TIMx_CNT 或者TIMx_CNT≤TIMx_CCRx。然而为了与 OCREF_CLR 的功能（在下一个 PWM 周期之前，ETR 信号上的一个外部事件能够清除 OCxREF）一致，OCxREF 信号只能在下述条件下产生：

- 当比较的结果改变。
- 当输出比较模式（TIMx_CCMRx 寄存器中的 OCxM 位）从"冻结"（无比较，OCxM='000'）切换到某个 PWM 模式（OCxM='110'或'111'）。

这样在运行中可以通过软件强置 PWM 输出。根据 TIMx_CR1 寄存器中 CMS 位的状态，定时器能够产生边沿对齐的 PWM 信号或中央对齐的 PWM 信号。

17.2.9.1 PWM 边沿对齐模式

17.2.9.2 向上计数配置

当 TIMx_CR1 寄存器中的 DIR 位为低的时候执行向上计数。参看章节：“计数器模式”。

下面是一个 PWM 模式 1 的例子。当 TIMx_CNT<TIMx_CCRx 时 PWM 参考信号 OCxREF 为高，否则为低。如果 TIMx_CCRx 中的比较值大于自动重装载值（TIMx_ARR），则 OCxREF 保持为’1’。如果比较值为 0，则 OCxREF 保持为’0’。下图为 TIMx_ARR=8 时边沿对齐的 PWM 波形实例。

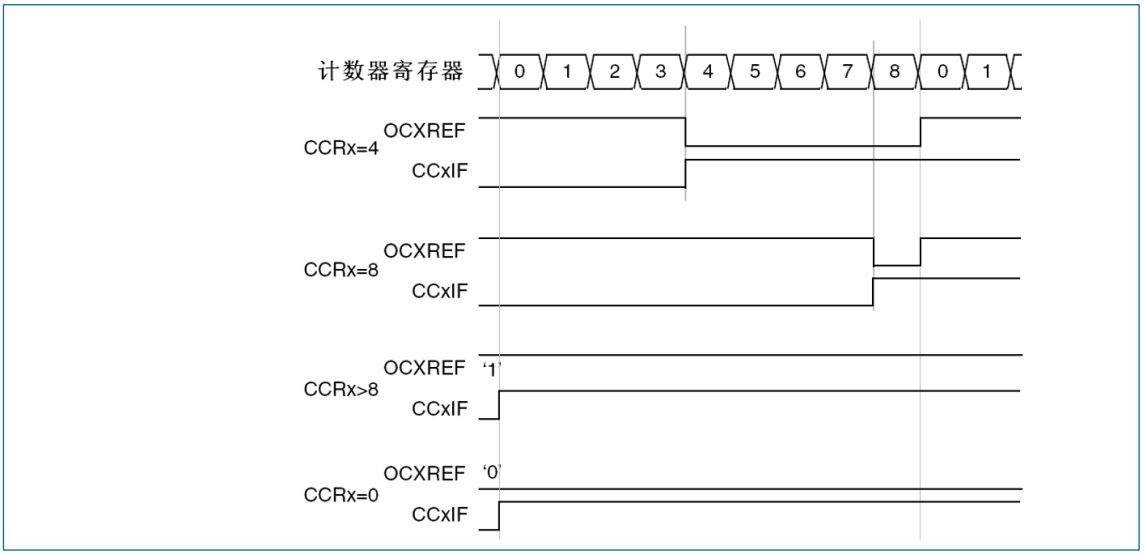


图 17-31 边沿对齐的 PWM 波形（ARR=8）

17.2.9.3 向下计数的配置

当 TIMx_CR1 寄存器的 DIR 位为高时执行向下计数。参看章节：“17.2.2.2 向下计数模式”。

在 PWM 模式 1，当 TIMx_CNT>TIMx_CCRx 时参考信号 OCxREF 为低，否则为高。如果 TIMx_CCRx 中的比较值大于 TIMx_ARR 中的自动重装载值，则 OCxREF 保持为’1’。该模式下不能产生 0%的 PWM 波形。

17.2.9.4 PWM 中央对齐模式

当 TIMx_CR1 寄存器中的 CMS 位不为’00’时，为中央对齐模式（所有其他的配置对 OCxREF/OCx 信号都有相同的作用）。根据不同的 CMS 位设置，比较标志可以在计数器向上计数时被置’1’、在计数器向下计数时被置’1’、或在计数器向上和向下计数时被置’1’。TIMx_CR1 寄存器中的计数方向位（DIR）由硬件更新，不要用软件修改它。参看章节计数器模式的中央对齐模式。

下图给出了一些中央对齐的 PWM 波形的例子，当：

- TIMx_ARR=8
- PWM 模式 1
- TIMx_CR1 寄存器中的 CMS=01，在中央对齐模式 1 时，当计数器向下计数时设置比较标志。

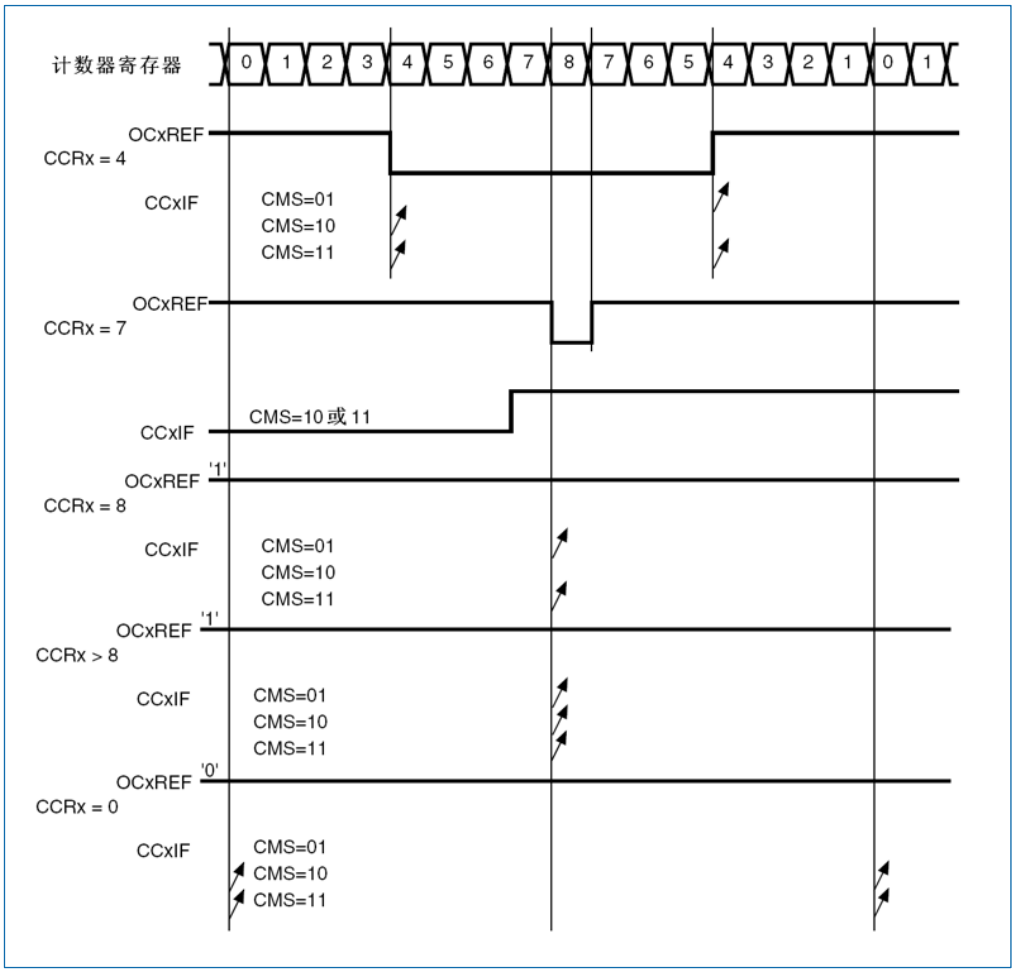


图 17-32 中央对齐的 PWM 波形（APR=8）

17.2.9.5 使用中央对齐模式的提示：

- 进入中央对齐模式时，使用当前的向上/向下计数配置；这就意味着计数器向上还是向下计数取决于 TIMx_CR1 寄存器中 DIR 位的当前值。此外，软件不能同时修改 DIR 和 CMS 位。
- 不推荐当运行在中央对齐模式时改写计数器，因为这会产生不可预知的结果。尤其是：
  - 如果写入计数器的值大于自动重加载的值（TIMx_CNT>TIMx_ARR），则方向不会被更新。例如，如果计数器正在向上计数，它就会继续向上计数。
  - 如果将 0 或者 TIMx_ARR 的值写入计数器，方向被更新，但不产生更新事件 UEV。
- 使用中央对齐模式最保险的方法，就是在启动计数器之前产生一个软件更新（设置 TIMx_EGR 寄存器中的 UG 位），不要在计数进行过程中修改计数器的值。

17.2.10 单脉冲模式

单脉冲模式（OPM）是前述众多模式的一个特例。这种模式允许计数器响应一个激励，并在一个程序可控的延时之后，产生一个脉宽程序可控的脉冲。

可以通过从模式控制器启动计数器，在输出比较模式或者 PWM 模式下产生波形。设置TIMx_CR1 寄存器中的 OPM 位将选择单脉冲模式，这样可以让计数器自动地在产生下一个更新事件 UEV 时停止。

仅当比较值与计数器的初始值不同时，才能产生一个脉冲。启动之前（当定时器正在等待触发），必须如下配置：

- 向上计数方式：计数器 CNT < CCRx ≤ ARR（特别地，0 < CCRx），
- 向下计数方式：计数器 CNT > CCRx。

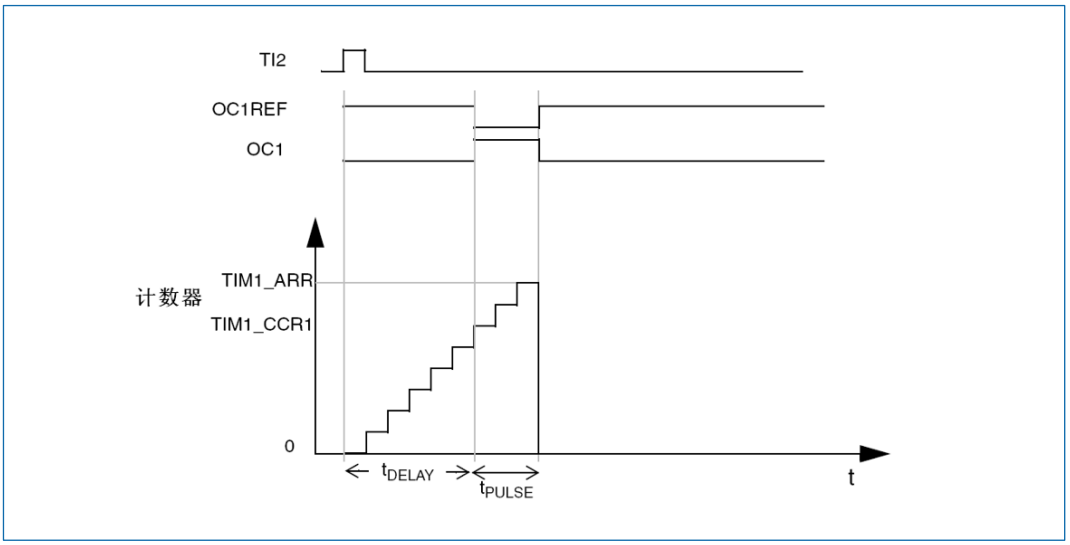


图 17-33 单脉冲模式的例子

例如，你需要在从 TI2 输入脚上检测到一个上升沿开始，延迟 t_{DELAY} 之后，在 OC1 上产生一个长度为 t_{PULSE} 的正脉冲。

假定 TI2FP2 作为触发 1:

- 置 TIMx_CCMR1 寄存器中的 CC2S='01'，把 TI2FP2 映射到 TI2。
- 置 TIMx_CCER 寄存器中的 CC2P='0'，使 TI2FP2 能够检测上升沿。
- 置 TIMx_SMCR 寄存器中的 TS='110'，TI2FP2 作为从模式控制器的触发（TRGI）。
- 置 TIMx_SMCR 寄存器中的 SMS='110'（触发模式），TI2FP2 被用来启动计数器。

OPM 波形由写入比较寄存器的数值决定（要考虑时钟频率和计数器预分频器）。

- t_{DELAY} 由 TIMx_CCR1 寄存器中的值定义。
- t_{PULSE} 由自动装载值和比较值之间的差值定义（TIMx_ARR-TIMx_CCR1）。

假定当发生比较匹配时要产生从'0'到'1'的波形，当计数器到达预装载值时要产生一个从'1'到'0'的波形：

1. 首先要置 TIMx_CCMR1 寄存器的 OC1M='111'，进入 PWM 模式 2；
2. 根据需要有选择地使能预装载寄存器：置 TIMx_CCMR1 中的 OC1PE='1'和 TIMx_CR1 寄存器中的 ARPE；
3. 然后在 TIMx_CCR1 寄存器中填写比较值，在 TIMx_ARR 寄存器中填写自动装载值，修改 UG 位来产生一个更新事件，然后等待在 TI2 上的一个外部触发事件。本例中，CC1P='0'。

在这个例子中，TIMx_CR1 寄存器中的 DIR 和 CMS 位应该置低。

因为只需要一个脉冲，所以必须设置 TIMx_CR1 寄存器中的 OPM='1'，在下一个更新事件（当计数器从自动装载值翻转到 0）时停止计数。

17.2.10.1 特殊情况：OCx 快速使能：

在单脉冲模式下，在 TIx 输入脚的边沿检测逻辑设置 CEN 位以启动计数器。然后计数器和比较值间的比较操作产生了输出的转换。但是这些操作需要一定的时钟周期，因此它限制了可得到的最小延时 t_{DELAY}。

如果要以最小延时输出波形，可以设置 TIMx_CCMRx 寄存器中的 OCxFE 位；此时 OCxREF（和OCx）被强制响应激励而不再依赖比较的结果，输出的波形与比较匹配时的波形一样。OCxFE 只在通道配置为 PWM1 和 PWM2 模式时起作用。

17.2.11 在外部事件时清除 OCxREF 信号

对于一个给定的通道，设置 TIMx_CCMRx 寄存器中对应的 OCxCE 位为'1'，能够用 ETRF 输入端的高电平把 OCxREF 信号拉低，OCxREF 信号将保持为低，直到发生下一次的更新事件 UEV。

该功能只能用于输出比较和 PWM 模式，而不能用于强置模式。

例如，OCxREF 信号可以连到一个比较器的输出，用于控制电流。这时，ETR 必须配置如下：

1. 外部触发预分频器必须处于关闭：TIMx_SMCR 寄存器中的 ETPS[1:0]='00'。
2. 必须禁止外部时钟模式 2：TIMx_SMCR 寄存器中的 ECE='0'。
3. 外部触发极性（ETP）和外部触发滤波器（ETF）可以根据需要配置。下图显示了当 ETRF 输入变为高时，对应不同 OCxCE 的值，OCxREF 信号的变化。在这个例子中，定时器 TIMx 被置于 PWM 模式。

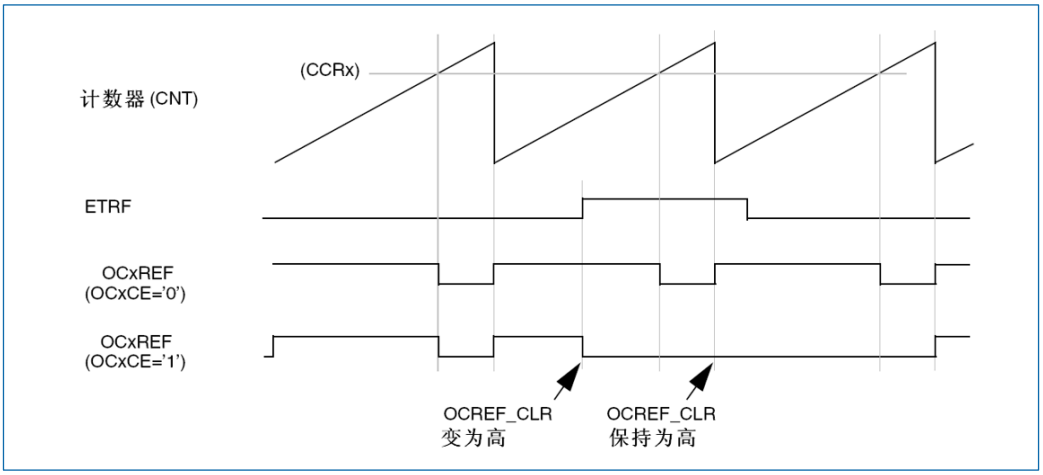


图 17-34 清除 TIMx 的 OCxREF

17.2.12 编码器接口模式

选择编码器接口模式的方法是：如果计数器只在 TI2 的边沿计数，则置 TIMx_SMCR 寄存器中的 SMS=001；如果只在 TI1 边沿计数，则置 SMS=010；如果计数器同时在 TI1 和 TI2 边沿计数，则置 SMS=011。

通过设置 TIMx_CCER 寄存器中的 CC1P 和 CC2P 位，可以选择 TI1 和 TI2 极性；如果需要，还可以对输入滤波器编程。

两个输入 TI1 和 TI2 被用来作为增量编码器的接口。参看表 17-1，假定计数器已经启动（TIMx_CR1 寄存器中的 CEN='1'），则计数器由每次在 TI1FP1 或 TI2FP2 上的有效跳变驱动。TI1FP1 和 TI2FP2 是 TI1 和 TI2 在通过输入滤波器和极性控制后的信号；如果没有滤波和变相，则 TI1FP1=TI1，TI2FP2=TI2。根据两个输入信号的跳变顺序，产生了计数脉冲和方向信号。依据两个输入信号的跳变顺序，计数器向上或向下计数，同时硬件对 TIMx_CR1 寄存器的 DIR 位进行相应的设置。不管计数器是依靠 TI1 计数、依靠 TI2 计数或者同时依靠 TI1 和 TI2 计数。在任一输入端（TI1 或者TI2）的跳变都会重新计算 DIR 位。

编码器接口模式基本上相当于使用了一个带有方向选择的外部时钟。这意味着计数器只在 0 到 TIMx_ARR 寄存器的自动装载值之间连续计数（根据方向，或是 0 到 ARR 计数，或是 ARR 到 0 计数）。所以在开始计数之前必须配置 TIMx_ARR；同样，捕获器、比较器、预分频器、触发输出特性等仍工作如常。

在这个模式下，计数器依照增量编码器的速度和方向被自动的修改，因此计数器的内容始终指示着编码器的位置。计数方向与相连的传感器旋转的方向对应。下表列出了所有可能的组合，假设 TI1 和 TI2 不同时变换。

表 17-1 计数方向与编码器信号的关系

有效边沿	相对信号的电平（TI1FP1 对应 TI2，TI2FP2 对应 TI1）	有效边沿在 TI1 上时，使用 TI1FP1 信号计数		有效边沿在 TI2 上时，使用 TI2FP2 信号计数	
		TI1FP1 信号		TI2FP2 信号	
		上升	下降	上升	下降
仅在 TI1 计数	TI2FP2 为高	向下计数	向上计数	不计数	不计数
	TI2FP2 为低	向上计数	向下计数	不计数	不计数
仅在 TI2 计数	TI1FP1 为高	不计数	不计数	向上计数	向下计数
	TI1FP1 为低	不计数	不计数	向下计数	向上计数
在 TI1（和 TI2）上计数	TI2FP2（和 TI1FP1）为高	向下计数	向上计数	向上计数	向下计数
	TI2FP2（和 TI1FP1）为低	向上计数	向下计数	向下计数	向上计数

一个外部的增量编码器可以直接与 MCU 连接而不需要外部接口逻辑。但是，一般会使用比较器将编码器的差动输出转换到数字信号，这大大增加了抗噪声干扰能力。编码器输出的第三个信号表示机械零点，可以把它连接到一个外部中断输入并触发一个计数器复位。

下图是一个计数器操作的实例，显示了计数信号的产生和方向控制。它还显示了当选择了双边沿时，输入抖动是如何被抑制的；抖动可能会在传感器的位置靠近一个转换点时产生。在这个例子中，我们假定配置如下：

- CC1S=’01’（TIMx_CCMR1 寄存器，IC1FP1 映射到 TI1）
- CC2S=’01’（TIMx_CCMR2 寄存器，IC2FP2 映射到 TI2）
- CC1P=’0’（TIMx_CCER 寄存器，IC1FP1 不反相，IC1FP1=TI1）
- CC2P=’0’（TIMx_CCER 寄存器，IC2FP2 不反相，IC2FP2=TI2）
- SMS=’011’（TIMx_SMCR 寄存器，所有的输入均在上升沿和下降沿有效）
- CEN=’1’（TIMx_CR1 寄存器，计数器使能）

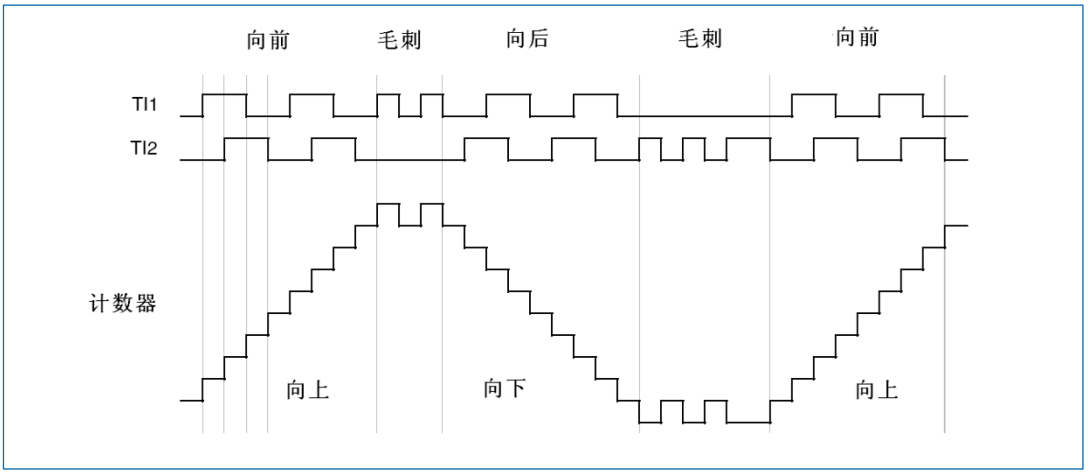


图 17-35 编码器模式下的计数器操作实例

下图为当 IC1FP1 极性反相时计数器的操作实例（CC1P=’1’，其他配置与上例相同）：

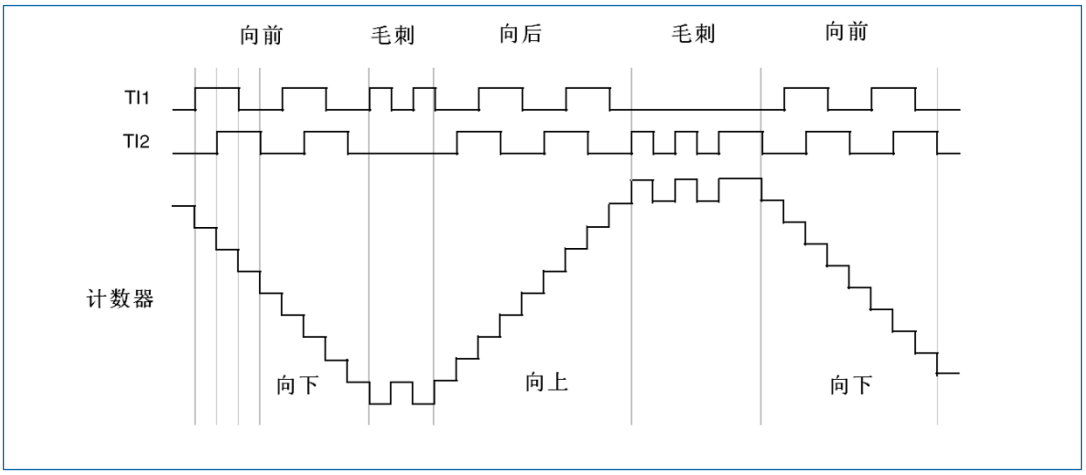


图 17-36 IC1FP1 反相的编码器接口模式实例

当定时器配置成编码器接口模式时，提供传感器当前位置的信息。使用第二个配置在捕获模式的定时器，可以测量两个编码器事件的间隔，获得动态的信息（速度，加速度，减速度）。指示机械零点的编码器输出可被用做此目的。根据两个事件间的间隔，可以按照固定的时间读出计数器。如果可能的话，你可以把计数器的值锁存到第三个输入捕获寄存器（捕获信号必须是周期的并且可以由另一个定时器产生）；也可以通过一个由实时时钟产生的 DMA 请求来读取它的值。

17.2.13 定时器输入异或功能

TIMx_CR2 寄存器中的 TI1S 位，允许通道 1 的输入滤波器连接到一个异或门的输出端，异或门的3 个输入端为 TIMx_CH1、TIMx_CH2 和 TIMx_CH3。

异或输入功能能够被用于所有定时器的输入功能，如触发或输入捕获。章节“16.2.18 与霍尔传感器的接口”给出了此特性用于连接霍尔传感器的例子。

17.2.14 定时器和外部触发的同步

TIMx 定时器能够在多种模式下和一个外部的触发同步：复位模式、门控模式和触发模式。

17.2.14.1 从模式：复位模式

在发生一个触发输入事件时，计数器和它的预分频器能够重新被初始化；同时，如果 TIMx_CR1寄存器的 URS 位为低，还会产生一个更新事件 UEV；然后所有的预装载寄存器（TIMx_ARR，TIMx_CCRx）都会被更新。

在下面的例子中，TI1 输入端的上升沿导致向上计数器被清零：

- 配置通道 1 以检测 TI1 的上升沿。配置输入滤波器的带宽（在本例中，不需要任何滤波器，因此保持 IC1F=0000）。触发操作中不使用捕获预分频器，所以不需要配置它。CC1S 位只选择输入捕获源，即 TIMx_CCMR1 寄存器中 CC1S=01。置 TIMx_CCER 寄存器中 CC1P=0 以确定极性（只检测上升沿）。
- 置 TIMx_SMCR 寄存器中 SMS=100，配置定时器为复位模式；置 TIMx_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。
- 置 TIMx_CR1 寄存器中 CEN=1，启动计数器。

计数器开始依据内部时钟计数，然后正常运转直到 TI1 出现一个上升沿；此时，计数器被清零然后从 0 重新开始计数。同时，触发标志（TIMx_SR 寄存器中的 TIF 位）被设置，根据 TIMx_DIER 寄存器中 TIE（中断使能）位和 TDE（DMA 使能）位的设置，产生一个中断请求或一个 DMA 请求。

下图显示当自动重装载寄存器 TIMx_ARR=0x36 时的动作。在 TI1 上升沿和计数器的实际复位之间的延时，取决于 TI1 输入端的重同步电路。

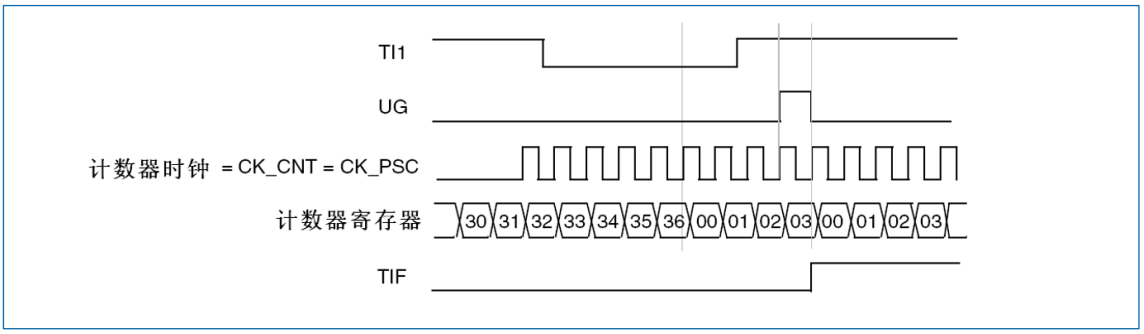


图 17-37 复位模式下的控制电路

17.2.14.2 从模式：门控模式

按照选中的输入端电平使能计数器。

在如下的例子中，计数器只在 TI1 为低时向上计数：

- 配置通道 1 以检测 TI1 上的低电平。配置输入滤波器带宽（本例中，不需要滤波，所以保持 IC1F=0000）。触发操作中不使用捕获预分频器，所以不需要配置。CC1S 位用于选择输入捕获源，置 TIMx_CCMR1 寄存器中 CC1S=01。置 TIMx_CCER 寄存器中 CC1P=1 以确定极性（只检测低电平）。
- 置 TIMx_SMCR 寄存器中 SMS=101，配置定时器为门控模式；置 TIMx_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。
- 置 TIMx_CR1 寄存器中 CEN=1，启动计数器。在门控模式下，如果 CEN=0，则计数器不能启动，不论触发输入电平如何。

只要 TI1 为低，计数器开始依据内部时钟计数，在 TI1 变高时停止计数。当计数器开始或停止时都设置 TIMx_SR 中的标志。

TI1 上升沿和计数器实际停止之间的延时，取决于 TI1 输入端的重同步电路。

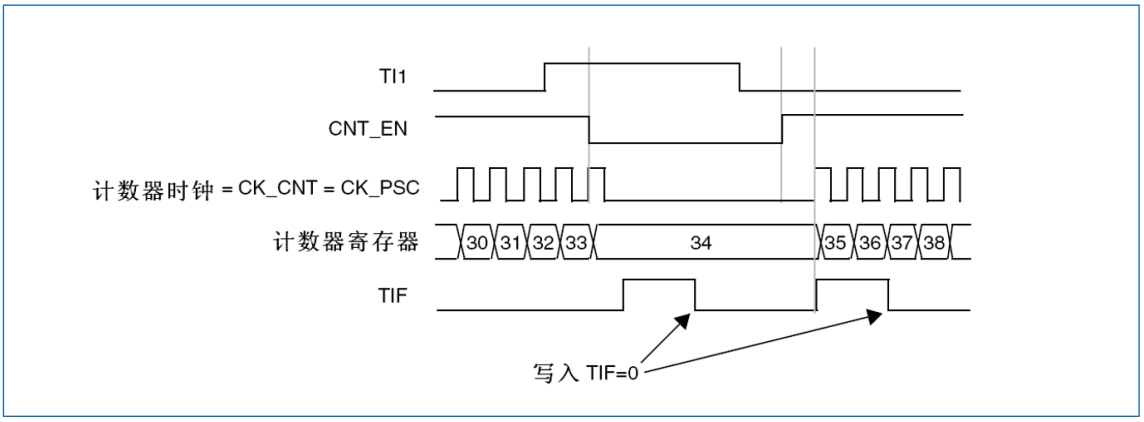


图 17-38 门控模式下的控制电路

17.2.14.3 从模式：触发模式

输入端上选中的事件使能计数器。

在下面的例子中，计数器在 TI2 输入的上升沿开始向上计数：

- 配置通道 2 检测 TI2 的上升沿。配置输入滤波器带宽（本例中，不需要任何滤波器，保持 IC2F=0000）。触发操作中不使用捕获预分频器，不需要配置。CC2S 位只用于选择输入捕获源，置 TIMx_CCMR1 寄存器中 CC2S=01。置 TIMx_CCER 寄存器中 CC2P=1 以确定极性（只检测低电平）。
- 置 TIMx_SMCR 寄存器中 SMS=110，配置定时器为触发模式；置 TIMx_SMCR 寄存器中 TS=110，选择 TI2 作为输入源。当 TI2 出现一个上升沿时，计数器开始在内部时钟驱动下计

数，同时设置 TIF 标志。

TI2 上升沿和计数器启动计数之间的延时，取决于 TI2 输入端的重同步电路。

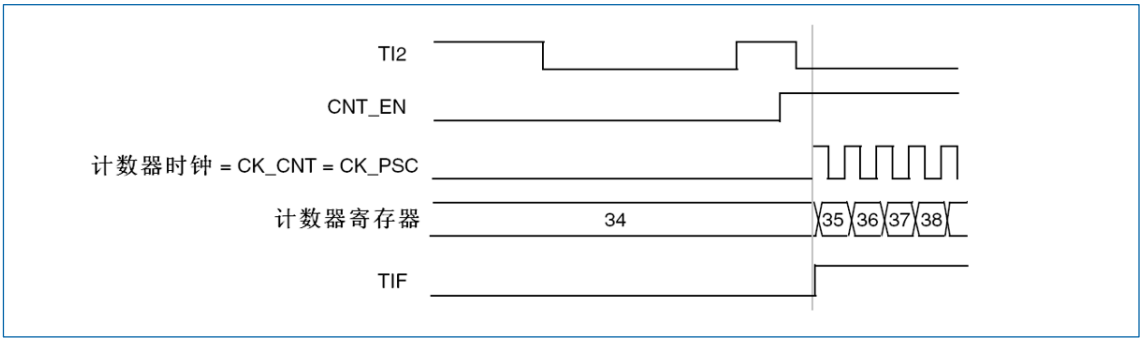


图 17-39 触发器模式下的控制电路

17.2.14.4 从模式：外部时钟模式 2 和触发模式

外部时钟模式 2 可以与另一种从模式（外部时钟模式 1 和编码器模式除外）一起使用。这时，ETR 信号被用作外部时钟的输入，在复位模式、门控模式或触发模式时可以选择另一个输入作为触发输入。不建议使用 TIMx_SMCR 寄存器的 TS 位选择 ETR 作为 TRGI。

下面的例子中，TI1 上出现一个上升沿之后，计数器即在 ETR 的每一个上升沿向上计数一次：

1. 通过 TIMx_SMCR 寄存器配置外部触发输入电路：
   - ETF=0000：没有滤波
   - ETPS=00：不用预分频器
   - ETP=0：检测 ETR 的上升沿，置 ECE=1 使能外部时钟模式
2. 按如下配置通道 1，检测 TI 的上升沿：
   - IC1F=0000：没有滤波
   - 触发操作中不使用捕获预分频器，不需要配置
   - 置 TIMx_CCMR1 寄存器中 CC1S=01，选择输入捕获源
   - 置 TIMx_CCER 寄存器中 CC1P=0 以确定极性（只检测上升沿）
3. 置 TIMx_SMCR 寄存器中 SMS=110，配置定时器为触发模式。置 TIMx_SMCR 寄存器中 TS=101，选择 TI1 作为输入源。

当 TI1 上出现一个上升沿时，TIF 标志被设置，计数器开始在 ETR 的上升沿计数。ETR 信号的上升沿和计数器实际复位间的延时，取决于 ETRP 输入端的重同步电路。

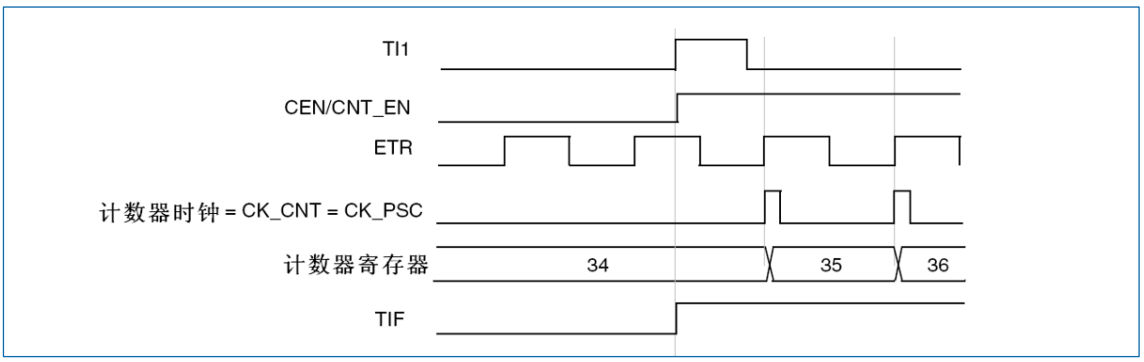


图 17-40 外部时钟模式 2＋触发模式下的控制电路

17.2.15 定时器同步

所有 TIMx 定时器在内部相连，用于定时器同步或链接。当一个定时器处于主模式时，它可以对另一个处于从模式的定时器的计数器进行复位、启动、停止或提供时钟等操作。

下图显示了触发选择和主模式选择模块的概况。

17.2.15.1 使用一个定时器作为另一个定时器的预分频器

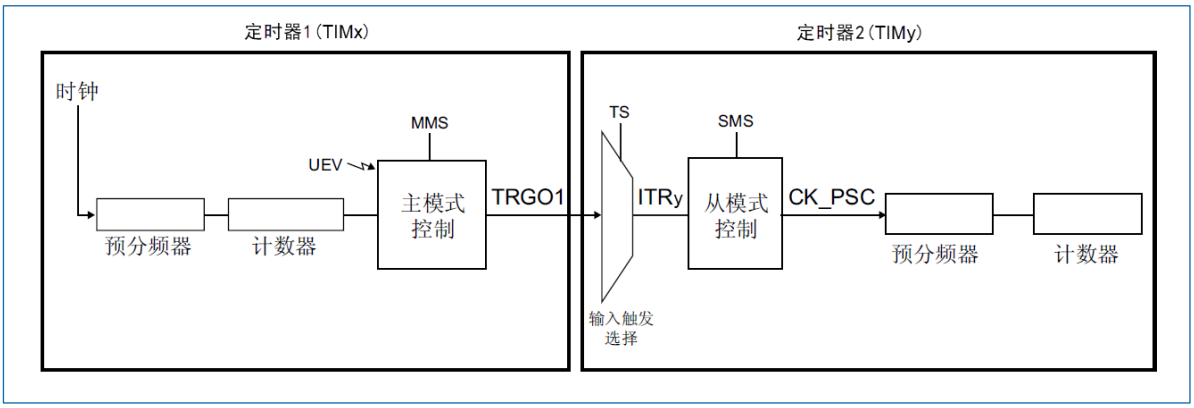


图 17-41 主/从定时器的例子

如：可以配置定时器 1（TIMx）作为定时器 2（TIMy）的预分频器。参考上图，进行下述操作：

1. 配置 TIMx 为主模式，它可以在每一个更新事件 UEV 时输出一个周期性的触发信号。在 TIMx_CR2 寄存器的 MMS=’010’时，每当产生一个更新事件时在 TRGO1 上输出一个上升沿信号。

2. 连接 TIMx 的 TRGO1 输出至 TIMy，设置 TIMy_SMCR 寄存器的 TS 位，配置 TIMy 使用相应的 ITRy 作为内部触发的从模式。

3. 然后把从模式控制器置于外部时钟模式 1（TIMy_SMCR 寄存器的 SMS=111）；这样 TIMy 即可由 TIMx 周期性的上升沿（即 TIMx 的计数器溢出）信号驱动。

4. 最后，必须设置相应（TIMy 的 TIMy_CR1 寄存器）的 CEN 位分别启动两个定时器。

说明：如果 OCx 已被选中为定时器 1 TIMx 的触发输出（MMS=1xx），它的上升沿用于驱动定时器 2 TIMy 的计数器。

17.2.15.2 使用一个定时器使能另一个定时器

在这个例子中，定时器 2（TIMy）的使能由定时器 1（TIMx）的输出比较控制。参考图主/从定时器的例子的连接。仅当 TIMx 的 OC1REF 为高时，TIMy 才对分频后的内部时钟计数。两个定时器的时钟频率都是由预分频器对 CK_INT 除以 3（$f_{CK\_CNT}=f_{CK\_INT}/3$）得到。

1. 配置 TIMx 为主模式，送出它的输出比较参考信号（OC1REF）为触发输出（TIMx 的 TIMx_CR2 寄存器的 MMS=100）。

2. 配置 TIMx 的 OC1REF 波形（TIMx 的 TIMx_CCMR1 寄存器）。

3. 配置 TIMy 从 TIMx 获得输入触发（TIMy 的 TIMy_SMCR 寄存器的 TS=000）。

4. 配置 TIMy 为门控模式（TIMy_SMCR 寄存器的 SMS=101）。

5. 置 TIMy 的 TIMy_CR1 寄存器的 CEN=1 以使能 TIMy。

6. 置 TIMx 的 TIMx_CR1 寄存器的 CEN=1 以启动 TIMx。

说明：定时器 2 (TIMy)的时钟不与定时器 1(TIMx)的时钟同步，这个模式只影响定时器 2 TIMy 计数器的使能信号。

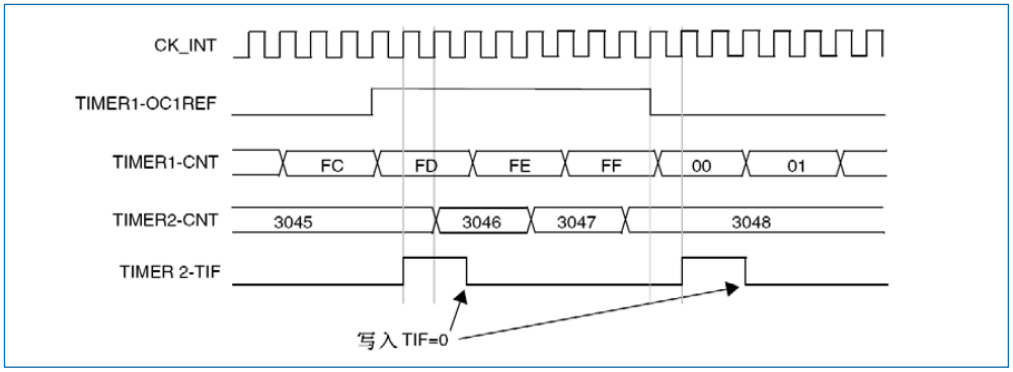


图 17-42 定时器 1 的 OC1REF 控制定时器 2

在上图的例子中，在定时器 2（TIMy）启动之前，它们的计数器和预分频器未被初始化，因此它们从当前的数值开始计数。可以在启动定时器 1（TIMx）之前复位 2 个定时器，使它们从给定的数值开始，即在定时器计数器中写入需要的任意数值。写 TIMy_EGR 寄存器的 UG 位即可复位定时器。

在下一个例子中，需要同步定时器 1（TIMx）和定时器 2（TIMy）。TIMx 是主模式并从 0 开始，TIMy 是从模式并从 0xE7 开始；2 个定时器的预分频器系数相同。写’0’到 TIMx 的 TIMx_CR1 的 CEN 位将禁止 TIMx，TIMy 随即停止。

1. 配置 TIMx 为主模式，送出输出比较 1 参考信号（OC1REF）做为触发输出（TIMx_CR2 寄存器的 MMS=100）。
2. 配置 TIMx 的 OC1REF 波形（TIMx 的 TIMx_CCMR1 寄存器）。
3. 配置 TIMy 从 TIMx 获得输入触发（TIMy 的 TIMy_SMCR 寄存器的 TS=000）。
4. 配置 TIMy 为门控模式（TIMy_SMCR 寄存器的 SMS=101）。
5. 置 TIMx 的 TIMx_EGR 寄存器的 UG=‘1’，复位 TIMx。
6. 置 TIMy 的 TIMy_EGR 寄存器的 UG=‘1’，复位 TIMy。
7. 写’0xE7’至 TIMy 的计数器（TIMy_CNTL），初始化它为 0xE7。
8. 置 TIMy_CR1 寄存器的 CEN=‘1’以使能 TIMy。
9. 置 TIMx_CR1 寄存器的 CEN=‘1’以启动 TIMx。
10. 置 TIMx 的 TIMx _CR1 寄存器的 CEN=‘0’以停止 TIMx。

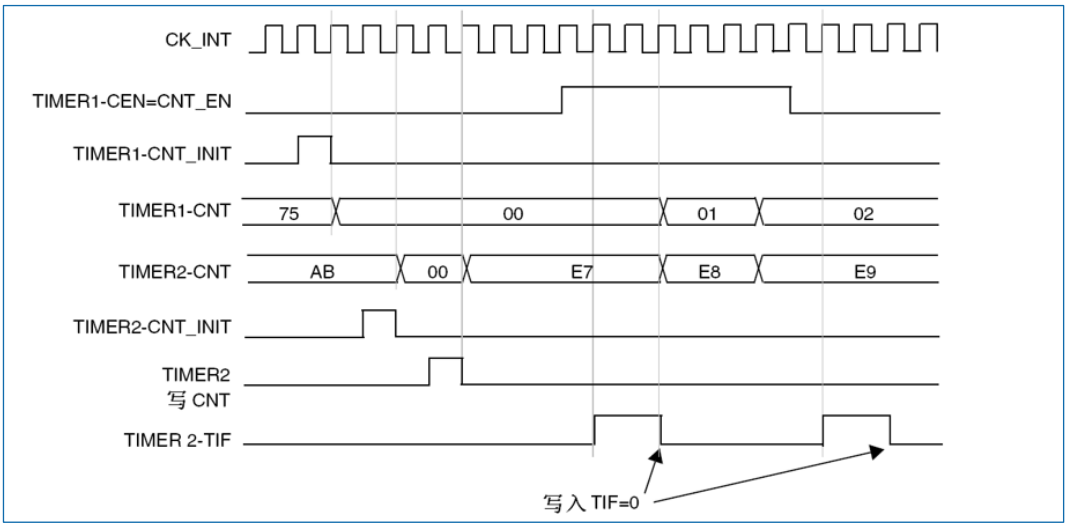


图 17-43 通过使能定时器 1 可以控制定时器 2

17.2.15.3 使用一个定时器去启动另一个定时器

在这个例子中，使用定时器 1（TIMx）的更新事件使能定时器 2（TIMy），参见图 17-41。一旦 TIMx 产生更新事件，TIMy 即从它当前的数值（可以是非 0）按照分频的内部时钟开始计数。在收到触发信号时，TIMy 的 CEN 位被自动地置'1'，同时计数器开始计数直到写'0'到定时器 2 的 TIMy_CR1 寄存器的 CEN位。两个定时器的时钟频率都是由预分频器对 CK_INT 除以 3（$f_{CK\_CNT}=f_{CK\_INT}/3$）。

1. 配置 TIMx 为主模式，送出它的更新事件（UEV）做为触发输出（TIMx_CR2 寄存器的 MMS=010）。
2. 配置 TIMx 的周期（TIMx_ARR 寄存器）。
3. 配置 TIMy 从 TIMx 获得输入触发（TIMy_SMCR 寄存器的 TS 位）。
4. 配置 TIMy 为触发模式（TIMy_SMCR 寄存器的 SMS=110）。
5. 置 TIMx_CR1 寄存器的 CEN=1 以启动 TIMx。

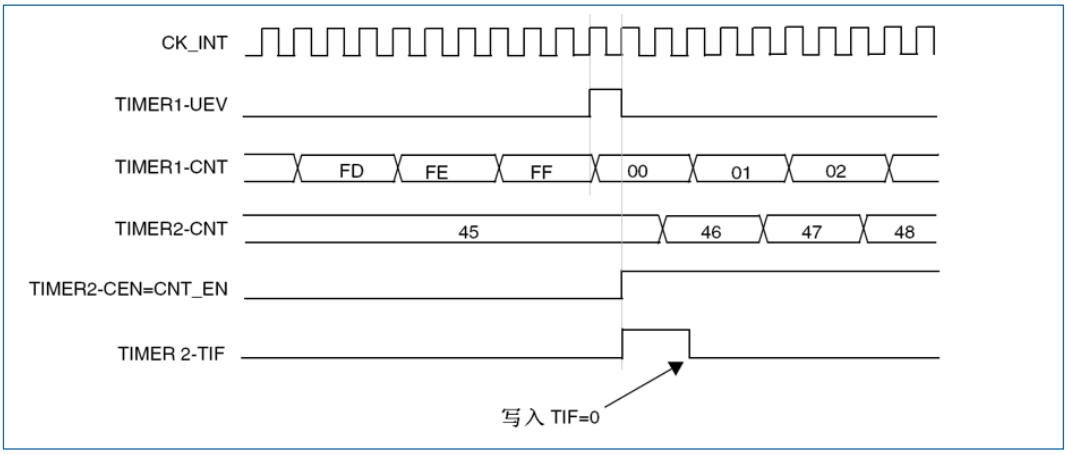


图 17-44 使用定时器 1 的更新触发定时器 2

在上一个例子中，可以在启动计数之前初始化两个计数器。下图显示在与 0 相同配置情况下，使用触发模式而不是门控模式（TIMy_SMCR 寄存器的 SMS=110）的动作。

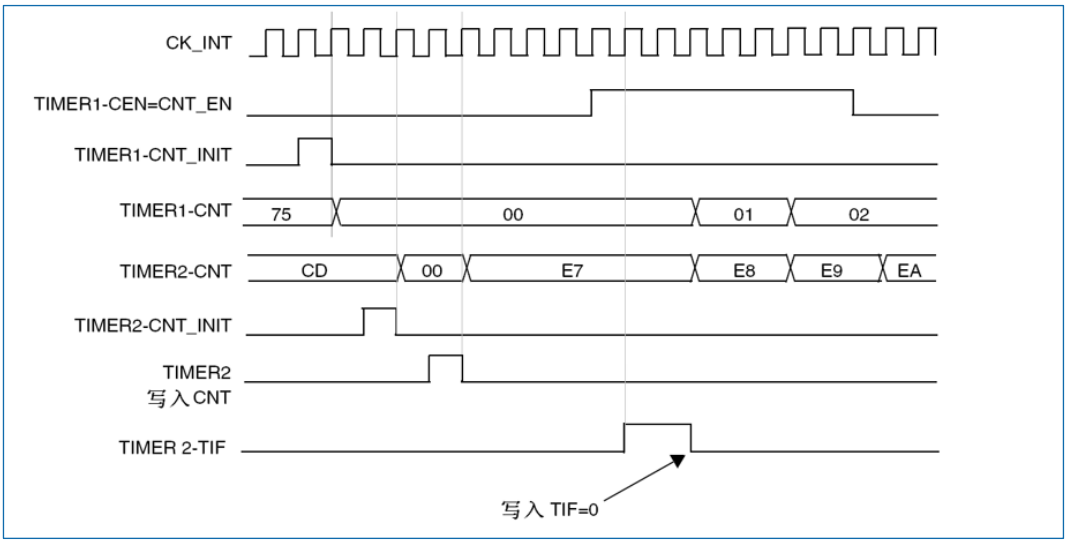


图 17-45 利用定时器 1 的使能触发定时器 2

17.2.15.4 使用一个外部触发同步地启动两个定时器

这个例子中当定时器 1（TIMx）的 TI1 输入上升时使能 TIMx，使能 TIMx 的同时使能定时器 2（TIMy），参见图 17-41。为保证计数器的对齐，TIMx 必须配置为主/从模式（对应 TI1 为从，对应 TIMy 为主）：

1. 配置 TIMx 为主模式，送出它的使能做为触发输出（TIMx_CR2 寄存器的 MMS=’001’）。
2. 配置 TIMx 为从模式，从 TI1 获得输入触发（TIMx_SMCR 寄存器的 TS=’100’）。
3. 配置 TIMx 为触发模式（TIMx_SMCR 寄存器的 SMS=’110’）。
4. 配置 TIMx 为主/从模式， TIMx_SMCR 寄存器的 MSM=’1’。
5. 配置 TIMy 从 TIMx 获得输入触发（TIMx_SMCR 寄存器的 TS 位）。
6. 配置 TIMy 为触发模式（TIMy_SMCR 寄存器的 SMS=’110’）。

当 TIMx 的 TI1 上出现一个上升沿时，两个定时器同步地按照内部时钟开始计数，两个 TIF 标志也同时被设置。

注意：在这个例子中，在启动之前两个定时器都被初始化（设置相应的 UG 位），两个计数器都从 0 开始，但可以通过写入任意一个计数器寄存器（TIMx_CNT）在定时器间插入一个偏移。下图中能看到主/从模式下在 TIMx 的 CNT_EN 和 CK_PSC 之间有个延迟。

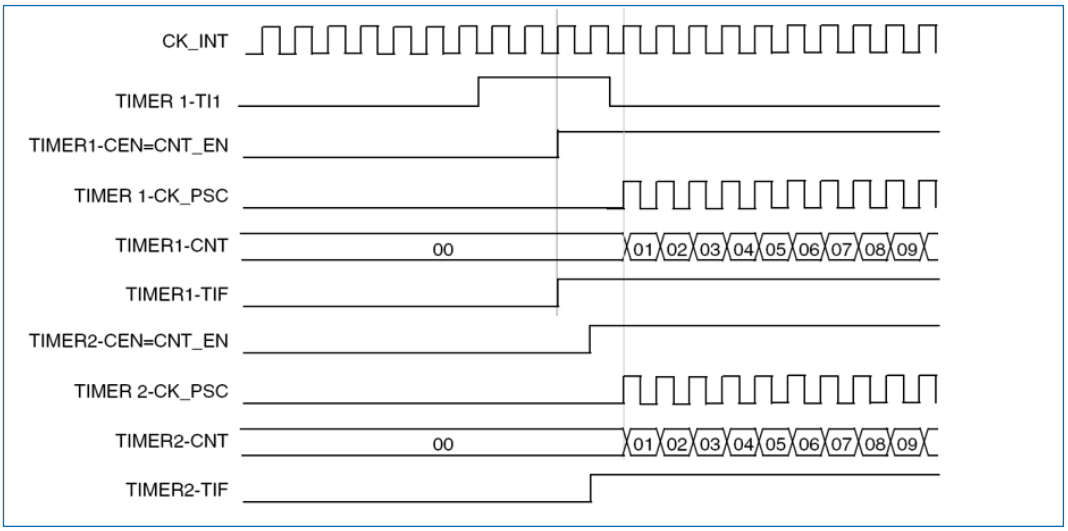


图 17-46 使用定时器 1 的 TI1 输入触发定时器 1 和定时器 2

17.2.16 调试模式

当微控制器进入调试模式（Cortex-M3 核心停止），根据 DBG 模块中 DBG_TIMx_STOP 的设置，TIMx 计数器或者继续正常操作，或者停止。详见随后章节：支持定时器、看门狗、 CAN 和 I2C 的调试。

17.3 TIM2/3/4/5 寄存器

基地址：（TIM2, TIM3, TIM4, TIM5）=（0x4000 0000, 0x4000 0400, 0x4000 0800, 0x4000 0C00）

空间大小：（TIM2, TIM3, TIM4, TIM5）=（0x400, 0x400, 0x400, 0x400）

可以用半字（16 位）或字（32 位）的方式操作这些外设寄存器。

17.3.1 控制寄存器 1（TIMx_CR1）（x=2,3,4,5）

偏移地址：0x00

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res						CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
						rw		rw	rw		rw	rw	rw	rw	rw

位 15:10	Res：保留 必须保持复位值。
位 9:8	CKD[1:0]：时钟分频因子（Clock division） 定义在定时器时钟（CK_INT）频率与数字滤波器（ETR，TIx）使用的采样频率之间的分频比例。 • 00：$t_{DTS}=t_{CK\_INT}$ • 01：$t_{DTS}=2 \times t_{CK\_INT}$ • 10：$t_{DTS}=4 \times t_{CK\_INT}$ • 11：保留
位 7	ARPE：自动重装载预装载允许位（Auto-reload preload enable） • 0：TIMx_ARR 寄存器没有缓冲 • 1：TIMx_ARR 寄存器被装入缓冲器
位 6:5	CMS[1:0]：选择中央对齐模式（Center-aligned mode selection）

	• 00：边沿对齐模式 计数器依据方向位（DIR）向上或向下计数。 • 01：中央对齐模式 1 计数器交替地向上和向下计数。配置为输出的通道（TIMx_CCMRx 寄存器中 CCxS=00）的输出比较中断标志位，只在计数器向下计数时被设置。 • 10：中央对齐模式 2 计数器交替地向上和向下计数。配置为输出的通道（TIMx_CCMRx 寄存器中 CCxS=00）的输出比较中断标志位，只在计数器向上计数时被设置。 • 11：中央对齐模式 3 计数器交替地向上和向下计数。配置为输出的通道（TIMx_CCMRx 寄存器中 CCxS=00）的输出比较中断标志位，在计数器向上和向下计数时均被设置。 *注意：在计数器开启时（CEN=1），不允许从边沿对齐模式转换到中央对齐模式。*
位 4	DIR：方向（Direction） • 0：计数器向上计数 • 1：计数器向下计数 *注意：当计数器配置为中央对齐模式或编码器模式时，该位为只读。*
位 3	OPM：单脉冲模式（One pulse mode） • 0：在发生更新事件时，计数器不停止 • 1：在发生下一次更新事件（清除 CEN 位）时，计数器停止。
位 2	URS：更新请求源（Update request source） 软件通过该位选择 UEV 事件的源。 • 0：如果使能了更新中断或 DMA 请求，则下述任一事件产生更新中断或 DMA 请求： o 计数器溢出/下溢 o 设置 UG 位 o 从模式控制器产生的更新 • 1：如果使能了更新中断或 DMA 请求，则只有计数器溢出/下溢才产生更新中断或 DMA 请求。
位 1	UDIS：禁止更新（Update disable） 软件通过该位允许/禁止 UEV 事件的产生。 • 0：允许 UEV 更新（UEV）事件由下述任一事件产生： o 计数器溢出/下溢 o 设置 UG 位 o 从模式控制器产生的更新 具有缓存的寄存器被装入它们的预装载值。 • 1：禁止 UEV 不产生更新事件，影子寄存器（ARR、PSC、CCRx）保持它们的值。如果设置了 UG 位或从模式控制器发出了一个硬件复位，则计数器和预分频器被重新初始化。
位 0	CEN：使能计数器（Counter enable） • 0：禁止计数器 • 1：使能计数器 *注意：在软件设置了 CEN 位后，外部时钟、门控模式和编码器模式才能工作。触发模式可以自动地通过硬件设置 CEN 位。在单脉冲模式下，当发生更新事件时，CEN 被自动清除。*

17.3.2 控制寄存器 2（TIMx_CR2）（x=2,3,4,5）

偏移地址：0x04

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								TI1S	MMS[2:0]			CCDS	Res		
								rw	rw			rw			

位 15:8	Res：保留 必须保持复位值。
位 7	TI1S：TI1 选择（TI1 selection） • 0：TIMx_CH1 引脚连到 TI1 输入。 • 1：TIMx_CH1、TIMx_CH2 和 TIMx_CH3 引脚经异或后连到 TI1 输入。 详见“16.2.18 与霍尔传感器的接口”。
位 6:4	MMS[2:0]：主模式选择（Master mode selection） • 这 3 位用于选择在主模式下送到从定时器的同步信息（TRGO）。 • 可能的组合如下： • 000：复位 TIMx_EGR 寄存器的 UG 位被用于作为触发输出（TRGO）。如果是触发输入产生的复位（从模式控制器处于复位模式），则 TRGO 上的信号相对实际的复位会有一个延迟。 • 001：使能 计数器使能信号 CNT_EN 被用于作为触发输出（TRGO）。有时需要在同一时间启动多个定时器或控制在一段时间内使能从定时器。计数器使能信号是通过 CEN 控制位和门控模式下的触发输入信号的“或”逻辑产生。当计数器使能信号受控于触发输入时，TRGO 上会有一个延迟，除非选择了主/从模式（见下章“TIMx_SMCR 寄存器”中 MSM 位的描述）。 • 010：更新 更新事件被选为触发输入（TRGO）。例如，一个主定时器的时钟可以被用作一个从定时器的预分频器。 • 011：比较脉冲 在发生一次捕获或一次比较成功时，当要设置 CC1IF 标志时（即使它已经为高），触发输出送出一个正脉冲（TRGO）。 • 100：比较 OC1REF 信号被用于作为触发输出（TRGO）。 • 101：比较 OC2REF 信号被用于作为触发输出（TRGO）。 • 110：比较 OC3REF 信号被用于作为触发输出（TRGO）。 • 111：比较 OC4REF 信号被用于作为触发输出（TRGO）。
位 3	CCDS：捕获/比较的 DMA 选择（Capture/Compare DMA selection） • 0：当发生 CCx 事件时，送出 CCx 的 DMA 请求； • 1：当发生更新事件时，送出 CCx 的 DMA 请求。
位 2:0	Res：保留 必须保持复位值。

17.3.3 从模式控制寄存器（TIMx_SMCR）（x=2,3,4,5）

偏移地址：0x08

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			Res	SMS[2:0]		
rw	rw	rw		rw				rw	rw				rw		

位 15	ETP：外部触发极性（External trigger polarity） 该位选择是用 ETR 还是 ETR 的反相来作为触发操作。 • 0：ETR 不反相，高电平或上升沿有效。 • 1：ETR 被反相，低电平或下降沿有效。
位 14	ECE：外部时钟使能位（External clock enable） 该位启用外部时钟模式 2 • 0：禁止外部时钟模式 2 • 1：使能外部时钟模式 2。计数器由 ETRF 信号上的任意有效边沿驱动。 *注意：* • *设置 ECE 位与选择外部时钟模式 1 并将 TRGI 连到 ETRF（SMS=111 和 TS=111）具有相同功效。* • *下述从模式可以与外部时钟模式 2 同时使用：复位模式、门控模式和触发模式；但是，这时 TRGI 不能连到 ETRF（TS 位不能是'111'）。* • *外部时钟模式 1 和外部时钟模式 2 同时被使能时，外部时钟的输入是 ETRF。*
位 13:12	ETPS[1:0]：外部触发预分频（External trigger prescaler） 外部触发信号 ETRP 的频率必须最多是 CK_INT 频率的 1/4。当输入较快的外部时钟时，可以使用预分频降低 ETRP 的频率。 • 00：关闭预分频 • 01：ETRP 频率除以 2 • 10：ETRP 频率除以 4 • 11：ETRP 频率除以 8
位 11:8	ETF[3:0]：外部触发滤波（External trigger filter） 该位域定义了对 ETRP 信号采样的频率和对 ETRP 数字滤波的带宽。实际上，数字滤波器是一个事件计数器，它记录到 N 个事件后会产生一个输出的跳变。 • 0000：无滤波器，以 f_{DTS} 采样 • 0001：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N=2 • 0010：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N=4 • 0011：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N=8 • 0100：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N=6 • 0101：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N=8 • 0110：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N=6 • 0111：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N=8 • 1000：采样频率 $f_{SAMPLING}=f_{DTS}/8$，N=6 • 1001：采样频率 $f_{SAMPLING}=f_{DTS}/8$，N=8 • 1010：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N=5 • 1011：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N=6 • 1100：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N=8

	• 1110：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N=6 • 1111：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N=8
位 7	MSM：主/从模式（Master/Slave mode） • 0：无作用 • 1：触发输入（TRGI）上的事件被延迟了，以允许在当前定时器（通过 TRGO）与它的从定时器间的美同步。这对要求把几个定时器同步到一个单一的外部事件时是非常有用的。
位 6:4	TS[2:0]：触发选择（Trigger selection） 这 3 位选择用于同步计数器的触发输入。 • 000：内部触发 0（ITR0） • 001：内部触发 1（ITR1） • 010：内部触发 2（ITR2） • 011：内部触发 3（ITR3） • 100：TI1 的边沿检测器（TI1F_ED） • 101：滤波后的定时器输入 1（TI1FP1） • 110：滤波后的定时器输入 2（TI2FP2） • 111：外部触发输入（ETRF）关于每个定时器中 ITRx 的细节，参见表 17-2。 *注意：该位域只能在未用到（如 SMS=000）时被改变，以避免在改变时产生错误的边沿检测。*
位 3	Res：保留 必须保持复位值。
位 2:0	SMS[2:0]：从模式选择（Slave mode selection） 当选择了外部信号，触发信号（TRGI）的有效边沿与选中的外部输入极性相关（见输入控制寄存器和控制寄存器的说明）。 • 000：关闭从模式 如果 CEN=1，则预分频器直接由内部时钟驱动。 • 001：编码器模式 1 根据 TI1FP1 的电平，计数器在 TI2FP2 的边沿向上/下计数。 • 010：编码器模式 2 根据 TI2FP2 的电平，计数器在 TI1FP1 的边沿向上/下计数。 • 011：编码器模式 3 根据 TI1FP1（和 TI2FP2）的电平，计数器在 TI2FP2（和 TI1FP1）的边沿向上/下计数。 • 100：复位模式 选中的触发输入（TRGI）的上升沿重新初始化计数器，并且产生一个更新寄存器的信号。 • 101：门控模式 当触发输入（TRGI）为高时，计数器的时钟开启。一旦触发输入变为低，则计数器停止（但不复位）。计数器的启动和停止都是受控的。 • 110：触发模式 计数器在触发输入 TRGI 的上升沿启动（但不复位），只有计数器的启动是受控的。 • 111：外部时钟模式 1 选中的触发输入（TRGI）的上升沿驱动计数器。 *注意：如果 TI1F_EN 被选为触发输入（TS=100）时，不要使用门控模式。这是因为，TI1F_ED 在每次 TI1F 变化时输出一个脉冲，然而门控模式是要检查触发输入的电平。*

表 17-2 TIMx 内部触发连接

从定时器	ITR0（TS=000）	ITR1（TS=001）	ITR2（TS=010）	ITR3（TS=011）
TIM2	TIM1	TIM8	TIM3	TIM4
TIM3	TIM1	TIM2	TIM5	TIM4
TIM4	TIM1	TIM2	TIM3	TIM8
TIM5	TIM2	TIM3	TIM4	TIM8

17.3.4 DMA/中断使能寄存器（TIMx_DIER）（x=2,3,4,5）

偏移地址：0x0C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	TDE	Res	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res	TIE	Res	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

位 15	Res：保留 必须保持复位值。
位 14	TDE：允许触发 DMA 请求（Trigger DMA request enable） • 0：禁止触发 DMA 请求 • 1：允许触发 DMA 请求
位 13	Res：保留 必须保持复位值。
位 12	CC4DE：允许捕获/比较 4 的 DMA 请求（Capture/Compare 4 DMA request enable） • 0：禁止捕获/比较 4 的 DMA 请求 • 1：允许捕获/比较 4 的 DMA 请求
位 11	CC3DE：允许捕获/比较 3 的 DMA 请求（Capture/Compare 3 DMA request enable） • 0：禁止捕获/比较 3 的 DMA 请求 • 1：允许捕获/比较 3 的 DMA 请求
位 10	CC2DE：允许捕获/比较 2 的 DMA 请求（Capture/Compare 2 DMA request enable） • 0：禁止捕获/比较 2 的 DMA 请求 • 1：允许捕获/比较 2 的 DMA 请求
位 9	CC1DE：允许捕获/比较 1 的 DMA 请求（Capture/Compare 1 DMA request enable） • 0：禁止捕获/比较 1 的 DMA 请求 • 1：允许捕获/比较 1 的 DMA 请求
位 8	UDE：允许更新的 DMA 请求（Update DMA request enable） • 0：禁止更新的 DMA 请求 • 1：允许更新的 DMA 请求
位 7	Res：保留 必须保持复位值。

位 6	TIE：触发中断使能（Trigger interrupt enable） • 0：禁止触发中断 • 1：使能触发中断
位 5	Res：保留 必须保持复位值。
位 4	CC4IE：允许捕获/比较 4 中断（Capture/Compare 4 interrupt enable） • 0：禁止捕获/比较 4 中断 • 1：允许捕获/比较 4 中断
位 3	CC3IE：允许捕获/比较 3 中断（Capture/Compare 3 interrupt enable） • 0：禁止捕获/比较 3 中断 • 1：允许捕获/比较 3 中断
位 2	CC2IE：允许捕获/比较 2 中断（Capture/Compare 2 interrupt enable） • 0：禁止捕获/比较 2 中断 • 1：允许捕获/比较 2 中断
位 1	CC1IE：允许捕获/比较 1 中断（Capture/Compare 1 interrupt enable） • 0：禁止捕获/比较 1 中断 • 1：允许捕获/比较 1 中断
位 0	UIE：允许更新中断（Update interrupt enable） • 0：禁止更新中断 • 1：允许更新中断

17.3.5 状态寄存器（TIMx_SR）（x=2,3,4,5）

偏移地址：0x10

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res			CC4OF	CC3OF	CC2OF	CC1OF	Res		TIF	Res	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

位 15:13	Res：保留 必须保持复位值。
位 12	CC4OF：捕获/比较 4 重复捕获标志（Capture/Compare 4 overcapture flag） 参见 CC1OF 描述。
位 11	CC3OF：捕获/比较 3 重复捕获标志（Capture/Compare 3 overcapture flag） 参见 CC1OF 描述。
位 10	CC2OF：捕获/比较 2 重复捕获标志（Capture/Compare 2 overcapture flag） 参见 CC1OF 描述。
位 9	CC1OF：捕获/比较 1 重复捕获标志（Capture/Compare 1 overcapture flag） 仅当相应的通道被配置为输入捕获时，该标志可由硬件置'1'。写'0'可清除该位。 • 0：无重复捕获产生。

	• 1：当计数器的值被捕获到 TIMx_CCR1 寄存器时，CC1IF 的状态已经为’1’。
位 8:7	Res：保留 必须保持复位值。
位 6	TIF：触发器中断标志（Trigger interrupt flag） 当发生触发事件（当从模式控制器处于除门控模式外的其它模式时，在 TRGI 输入端检测到有效边沿，或门控模式下的任一边沿）时由硬件对该位置’1’。它由软件清零。 • 0：无触发器事件产生 • 1：触发器中断等待响应
位 5	Res：保留 必须保持复位值。
位 4	CC4IF：捕获/比较 4 中断标志（Capture/Compare 4 interrupt flag） 参考 CC1IF 描述。
位 3	CC3IF：捕获/比较 3 中断标志（Capture/Compare 3 interrupt flag） 参考 CC1IF 描述。
位 2	CC2IF：捕获/比较 2 中断标志（Capture/Compare 2 interrupt flag） 参考 CC1IF 描述。
位 1	CC1IF：捕获/比较 1 中断标志（Capture/Compare 1 interrupt flag） • 如果通道 CC1 配置为输出模式： 当计数器值与比较值匹配时该位由硬件置’1’，但在中心对称模式下除外（参考 TIMx_CR1 寄存器的 CMS 位）。它由软件清零。 o 0：无匹配发生 o 1：TIMx_CNT 的值与 TIMx_CCR1 的值匹配 • 如果通道 CC1 配置为输入模式： 当捕获事件发生时该位由硬件置’1’，它由软件清零或通过读 TIMx_CCR1 清零。 o 0：无输入捕获产生 o 1：计数器值已被捕获（拷贝）至 TIMx_CCR1（在 IC1 上检测到与所选极性相同的边沿）
位 0	UIF：更新中断标志（Update interrupt flag） • 当产生更新事件时该位由硬件置’1’。它由软件清零。 o 0：无更新事件产生 o 1：更新中断等待响应 • 当寄存器被更新时该位由硬件置’1’： o 若 TIMx_CR1 寄存器的 UDIS=0、URS=0，当 TIMx_EGR 寄存器的 UG=1 时产生更新事件（软件对计数器 CNT 重新初始化）。 o 若 TIMx_CR1 寄存器的 UDIS=0、URS=0，当计数器 CNT 被触发事件重初始化时产生更新事件（参考同步控制寄存器的说明）。

17.3.6 事件产生寄存器（TIMx_EGR）（x=2,3,4,5）

偏移地址：0x14

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res									TG	Res	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

位 15:7	Res：保留 必须保持复位值。
位 6	TG：产生触发事件（Trigger generation） 该位由软件置’1’，用于产生一个触发事件，由硬件自动清零。 • 0：无动作 • 1：TIMx_SR 寄存器的 TIF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。
位 5	Res：保留 必须保持复位值。
位 4	CC4G：产生捕获/比较 4 事件（Capture/Compare 4 generation） 参考 CC1G 描述。
位 3	CC3G：产生捕获/比较 3 事件（Capture/Compare 3 generation） 参考 CC1G 描述。
位 2	CC2G：产生捕获/比较 2 事件（Capture/Compare 2 generation） 参考 CC1G 描述。
位 1	CC1G：产生捕获/比较 1 事件（Capture/Compare 1 generation） 该位由软件置’1’，用于产生一个捕获/比较事件，由硬件自动清零。 • 0：无动作 • 1：在通道 CC1 上产生一个捕获/比较事件： o 若通道 CC1 配置为输出： 设置 CC1IF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。 o 若通道 CC1 配置为输入： 当前的计数器值捕获至 TIMx_CCR1 寄存器；设置 CC1IF=1，若开启对应的中断和 DMA，则产生相应的中断和 DMA。若 CC1IF 已经为 1，则设置 CC1OF=1。
位 0	UG：产生更新事件（Update generation） 该位由软件置’1’，由硬件自动清零。 • 0：无动作 • 1：重新初始化计数器，并产生一个更新事件 注意预分频器的计数器也被清零（但是预分频系数不变）。 • 若在中心对称模式下或 DIR=0（向上计数）则计数器被清零。 • 若 DIR=1（向下计数）则计数器取 TIMx_ARR 的值。

17.3.7 捕获/比较模式寄存器 1（TIMx_CCMR1）（x=2,3,4,5）

偏移地址：0x18

复位值：0x0000

通道可用于输入（捕获模式）或输出（比较模式），通道的方向由相应的 CCxS 定义。该寄存器其它位的作用在输入和输出模式下不同。OCxx 描述了通道在输出模式下的功能，ICxx 描述了通道在输入模式下的功能。因此必须注意，同一个位在输出模式和输入模式下的功能是不同的。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	

输出比较模式：

位 15	OC2CE：输出比较 2 清零使能 （Output compare 2 clear enable）
位 14:12	OC2M[2:0]：输出比较 2 模式（Output compare 2 mode）
位 11	OC2PE：输出比较 2 预装载使能（Output compare 2 preload enable）
位 10	OC2FE：输出比较 2 快速使能（Output compare 2 fast enable）
位 9:8	CC2S[1:0]：捕获/比较 2 选择（Capture/Compare 2 selection） 该位定义通道的方向（输入/输出），及输入脚的选择： • 00：CC2 通道被配置为输出。 • 01：CC2 通道被配置为输入，IC2 映射在 TI2 上。 • 10：CC2 通道被配置为输入，IC2 映射在 TI1 上。 • 11：CC2 通道被配置为输入，IC2 映射在 TRC 上。 此模式仅工作在内部触发器输入被选中时（由 TIMx_SMCR 寄存器的 TS 位选择）。 *注意：CC2S 仅在通道关闭时（TIMx_CCER 寄存器的 CC2E='0'）才是可写的。*
位 7	OC1CE：输出比较 1 清零使能（Output compare 1 clear enable） • 0：OC1REF 不受 ETRF 输入的影响。 • 1：一旦检测到 ETRF 输入高电平，清除 OC1REF=0。
位 6:4	OC1M[2:0]：输出比较 1 模式（Output compare 1 mode） 该 3 位定义了输出参考信号 OC1REF 的动作，而 OC1REF 决定了 OC1 的值。OC1REF 是高电平有效，而 OC1 的有效电平取决于 CC1P 位。 • 000：冻结 输出比较寄存器 TIMx_CCR1 与计数器 TIMx_CNT 间的比较对 OC1REF 不起作用。 • 001： 匹配时设置通道 1 为有效电平 当计数器TIMx_CNT 的值与捕获/比较寄存器1（TIMx_CCR1）相同时，强制 OC1REF 为高。 • 010：匹配时设置通道 1 为无效电平 当计数器TIMx_CNT 的值与捕获/比较寄存器 1（TIMx_CCR1）相同时，强制 OC1REF 为低。 • 011：翻转 当 TIMx_CCR1=TIMx_CNT 时，翻转 OC1REF 的电平。 • 100：强制为无效电平。强制 OC1REF 为低。 • 101：强制为有效电平。强制 OC1REF 为高。 • 110：PWM 模式 1 在向上计数时，一旦 TIMx_CNT<TIMx_CCR1 时通道 1 为有效电平，否则为无效电平；在向下计数时，一旦 TIMx_CNT>TIMx_CCR1 时通道 1 为无效电平（OC1REF=0），否则为有效电平（OC1REF=1）。 • 111：PWM 模式 2 在向上计数时，一旦 TIMx_CNT<TIMx_CCR1 时通道 1 为无效电平，否则为有效电平；在向下计数时，一旦 TIMx_CNT>TIMx_CCR1 时通道 1 为有效电平，否则为无效电平。 *注意：在 PWM 模式 1、2 时，仅在比较结果发生改变或输出比较模式从 Frozen 切换到 PWM 模式时 OCREF 才改变。*
位 3	OC1PE：输出比较 1 预装载使能（Output compare 1 preload enable） • 0：禁止 TIMx_CCR1 寄存器的预装载功能，可随时写入 TIMx_CCR1 寄存器，并且新写入的数值立即起作用。

	• 1：开启 TIMx_CCR1 寄存器的预装载功能，读写操作仅对预装载寄存器操作，TIMx_CCR1 的预装载值在更新事件到来时被传送至当前寄存器中。 *注:* • *一旦 LOCK 级别设为 3（TIMx_BDTR 寄存器中的 LOCK 位）并且 CC1S='00'（该通道配置成输出）则该位不能被修改。* • *仅在单脉冲模式下（TIMx_CR1 寄存器的 OPM='1'），可以在未确认预装载寄存器情况下使用 PWM 模式，否则其动作不确定。*
位 2	OC1FE：输出比较 1 快速使能（Output compare 1 fast enable） 该位用于加快 CC 输出对触发器输入事件的响应。 • 0：根据计数器与 CCR1 的值，CC1 正常操作，即使触发器是打开的。当触发器的输入出现一个有效沿时，激活 CC1 输出的最小延时为 5 个时钟周期。 • 1：输入到触发器的有效沿的作用就像发生了一次比较匹配。因此，OC 被设置为比较电平而与比较结果无关。采样触发器的有效沿和 CC1 输出间的延时被缩短为 3 个时钟周期。 该位只在通道被配置成 PWM1 或 PWM2 模式时起作用。
位 1:0	CC1S[1:0]：捕获/比较 1 选择（Capture/Compare 1 selection） 这 2 位定义通道的方向（输入/输出），及输入脚的选择： • 00：CC1 通道被配置为输出。 • 01：CC1 通道被配置为输入，IC1 映射在 TI1 上。 • 10：CC1 通道被配置为输入，IC1 映射在 TI2 上。 • 11：CC1 通道被配置为输入，IC1 映射在 TRC 上。 此模式仅工作在内部触发器输入被选中时（由 TIMx_SMCR 寄存器的 TS 位选择）。 *注意：CC1S 仅在通道关闭时（TIMx_CCER 寄存器的 CC1E='0'）才是可写的。*

输入捕获模式：

位 15:12	IC2F[3:0]：输入捕获 2 滤波器（Input capture 2 filter）
位 11:10	IC2PSC[1:0]：输入/捕获 2 预分频器（Input capture 2 prescaler）
位 9:8	CC2S[1:0]：捕获/比较 2 选择（Capture/Compare 2 selection） 这 2 位定义通道的方向（输入/输出），及输入脚的选择： • 00：CC2 通道被配置为输出。 • 01：CC2 通道被配置为输入，IC2 映射在 TI2 上。 • 10：CC2 通道被配置为输入，IC2 映射在 TI1 上。 • 11：CC2 通道被配置为输入，IC2 映射在 TRC 上。 此模式仅工作在内部触发器输入被选中时（由TIMx_SMCR 寄存器的 TS 位选择）。 *注意：CC2S 仅在通道关闭时（TIMx_CCER 寄存器的 CC2E='0'）才是可写的。*
位 7:4	IC1F[3:0]：输入捕获 1 滤波器（Input capture 1 filter） 这几位定义了 TI1 输入的采样频率及数字滤波器长度。数字滤波器由一个事件计数器组成，它记录到 N 个事件后会产生一个输出的跳变： • 0000：无滤波器，以 f_{DTS} 采样 • 0001：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N=2 • 0010：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N=4 • 0011：采样频率 $f_{SAMPLING}=f_{CK\_INT}$，N=8 • 0100：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N=6

	• 0101：采样频率 $f_{SAMPLING}=f_{DTS}/2$，N=8 • 0110：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N=6 • 0111：采样频率 $f_{SAMPLING}=f_{DTS}/4$，N=8 • 1000：采样频率 $f_{SAMPLING}=f_{DTS}/8$，N=6 • 1001：采样频率 $f_{SAMPLING}=f_{DTS}/8$，N=8 • 1010：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N=5 • 1011：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N=6 • 1100：采样频率 $f_{SAMPLING}=f_{DTS}/16$，N=8 • 1101：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N=5 • 1110：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N=6 • 1111：采样频率 $f_{SAMPLING}=f_{DTS}/32$，N=8 *注意：在现在的芯片版本中，当 ICxF[3:0]=1、2 或 3 时，公式中的 f_{DTS} 由 CK_INT 替代。*
位 3:2	IC1PSC[1:0]：输入/捕获 1 预分频器（Input capture 1 prescaler） 这 2 位定义了 CC1 输入（IC1）的预分频系数。一旦 CC1E=’0’（TIMx_CCER 寄存器中），则预分频器复位。 • 00：无预分频器，捕获输入口上检测到的每一个边沿都触发一次捕获 • 01：每 2 个事件触发一次捕获 • 10：每 4 个事件触发一次捕获 • 11：每 8 个事件触发一次捕获
位 1:0	CC1S[1:0]：捕获/比较 1 选择（Capture/Compare 1 selection） 这 2 位定义通道的方向（输入/输出），及输入脚的选择： • 00：CC1 通道被配置为输出。 • 01：CC1 通道被配置为输入，IC1 映射在 TI1 上。 • 10：CC1 通道被配置为输入，IC1 映射在 TI2 上。 • 11：CC1 通道被配置为输入，IC1 映射在 TRC 上。 此模式仅工作在内部触发器输入被选中时（由TIMx_SMCR 寄存器的 TS 位选择）。 *注意：CC1S 仅在通道关闭时（TIMx_CCER 寄存器的 CC1E=’0’）才是可写的。*

17.3.8 捕获/比较模式寄存器 2（TIMx_CCMR2）（x=2,3,4,5）

偏移地址：0x1C

复位值：0x0000

参看以上 CCMR1 寄存器的描述

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	

输出比较模式：

位 15	OC4CE：输出比较 4 清零使能（Output compare 4 clear enable）
位 14:12	OC4M[2:0]：输出比较 4 模式（Output compare 4 mode）
位 11	OC4PE：输出比较 4 预装载使能（Output compare 4 preload enable）
位 10	OC4FE：输出比较 4 快速使能（Output compare 4 fast enable）

位 9:8	CC4S[1:0]：捕获/比较 4 选择（Capture/Compare 4 selection） 这 2 位定义通道的方向（输入/输出），及输入脚的选择： • 00：CC4 通道被配置为输出。 • 01：CC4 通道被配置为输入，IC4 映射在 TI4 上。 • 10：CC4 通道被配置为输入，IC4 映射在 TI3 上。 • 11：CC4 通道被配置为输入，IC4 映射在 TRC 上。 此模式仅工作在内部触发器输入被选中时（由 TIMx_SMCR 寄存器的 TS 位选择）。 *注意：CC4S 仅在通道关闭时（TIMx_CCER 寄存器的 CC4E='0'）才是可写的。*
位 7	OC3CE：输出比较 3 清零使能（Output compare 3 clear enable）
位 6:4	OC3M[2:0]：输出比较 3 模式（Output compare 3 mode）
位 3	OC3PE：输出比较 3 预装载使能（Output compare 3 preload enable）
位 2	OC3FE：输出比较 3 快速使能（Output compare 3 fast enable）
位 1:0	CC3S[1:0]：捕获/比较 3 选择（Capture/Compare 3 selection） 这 2 位定义通道的方向（输入/输出），及输入脚的选择： • 00：CC3 通道被配置为输出。 • 01：CC3 通道被配置为输入，IC3 映射在 TI3 上。 • 10：CC3 通道被配置为输入，IC3 映射在 TI4 上。 • 11：CC3 通道被配置为输入，IC3 映射在 TRGI 上。 此模式仅工作在内部触发器输入被选中时（由 TIMx_SMCR 寄存器的 TS 位选择）。 注意：CC3S 仅在通道关闭时（TIMx_CCER 寄存器的 CC3E='0'）才是可写的。

输入捕获模式：

位 15:12	IC4F[3:0]：输入捕获 4 滤波器（Input capture 4 filter）
位 11:10	IC4PSC[1:0]：输入/捕获 4 预分频器（Input capture 4 prescaler）
位 9:8	CC4S[1:0]：捕获/比较 4 选择（Capture/Compare 4 selection） 这 2 位定义通道的方向（输入/输出），及输入脚的选择： • 00：CC4 通道被配置为输出。 • 01：CC4 通道被配置为输入，IC4 映射在 TI4 上。 • 10：CC4 通道被配置为输入，IC4 映射在 TI3 上。 • 11：CC4 通道被配置为输入，IC4 映射在 TRC 上。此模式仅工作在内部触发器输入被选中时（由 TIMx_SMCR 寄存器的 TS 位选择）。 *注意：CC4S 仅在通道关闭时（TIMx_CCER 寄存器的 CC4E='0'）才是可写的。*
位 7:4	IC3F[3:0]：输入捕获 3 滤波器（Input capture 3 filter）
位 3:2	IC3PSC[1:0]：输入/捕获 3 预分频器（Input capture 3 prescaler）
位 1:0	CC3S[1:0]：捕获/比较 3 选择（Capture/Compare 3 selection） 这 2 位定义通道的方向（输入/输出），及输入脚的选择： • 00：CC3 通道被配置为输出。 • 01：CC3 通道被配置为输入，IC3 映射在 TI3 上。 • 10：CC3 通道被配置为输入，IC3 映射在 TI4 上。

	• 11：CC3 通道被配置为输入，IC3 映射在 TRC 上。 此模式仅工作在内部触发器输入被选中时（由 TIMx_SMCR 寄存器的 TS 位选择）。 *注意：CC3S 仅在通道关闭时（TIMx_CCER 寄存器的 CC3E='0'）才是可写的。*

17.3.9 捕获/比较使能寄存器（TIMx_CCER）（x=2,3,4,5）

偏移地址：0x20

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res	CC4P	CC4E	CC3NP	Res	CC3P	CC3E	CC2NP	Res	CC2P	CC2E	CC1NP	Res	CC1P	CC1E
rw		rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw

位 15	CC4NP：捕获/比较 4 极性（Capture/Compare 4 complementary output polarity） 这一位和 CC1P 联合使用来定义 TI3FP4 和 TI4FP4。详细的参考 CC1P 的描述。 *说明：该位仅用于通道配置为输入时使用。*
位 14	Res：保留 必须保持复位值。
位 13	CC4P：捕获/比较 4 输出极性（Capture/Compare 4 output polarity） 参考 CC1P 的描述。
位 12	CC4E：捕获/比较 4 输出使能（Capture/Compare 4 output enable） 参考 CC1E 的描述。
位 11	CC3NP：捕获/比较 3 极性（Capture 3 polarity） 用于输入双沿触发，详细的参考 CC1P 的描述。
位 10	Res：保留 必须保持复位值。
位 9	CC3P：捕获/比较 3 输出极性（Capture/Compare 3 output polarity） 参考 CC1P 的描述。
位 8	CC3E：捕获/比较 3 输出使能（Capture/Compare 3 output enable） 参考 CC1E 的描述。
位 7	CC2NP：捕获 2 极性（Capture 2 polarity） 用于输入双沿触发，详细的参考 CC1P 的描述。
位 6	Res：保留 必须保持复位值。
位 5	CC2P：捕获/比较 2 输出极性（Capture/Compare 2 output polarity） 参考 CC1P 的描述。
位 4	CC2E：捕获/比较 2 输出使能（Capture/Compare 2 output enable） 参考 CC1E 的描述。
位 3	CC1NP：捕获 1 极性（Capture 1 polarity） 用于输入双沿触发，详细的参考 CC1P 的描述。

位 2	Res：保留 必须保持复位值。
位 1	CC1P：捕捉/比较 1 输出极性（Capture/Compare 1 output polarity） • CC1 通道配置为输出： ○ 0：OC1 高电平有效； ○ 1：OC1 低电平有效。 • CC1 通道配置为输入： CC1P 和 CC1NP 组成两位控制来选择 TI1FP1 和 TI2FP1 的激活极性： ○ 00：没有翻转/上升沿 电路对 TIxFP1 的上升沿敏感（捕获或触发操作复位，外部时钟或触发模式），TIxFP1 没有翻转（触发操作在门控或编码模式）。 ○ 01：翻转/下降沿 电路对 TIxFP1 的下降沿敏感（捕获或触发操作复位，外部时钟或触发模式），TIxFP1 翻转（触发操作在门控或编码模式）。 ○ 10：保留 ○ 11：没有翻转/双沿 电路对 TIxFP1 的上升沿和下降沿敏感（捕获或触发操作复位，外部时钟或触发模式），TIxFP1 没有翻转（触发操作在门控模式）。这个配置不能用在编码模式。 *注意：* • *在具有互补输出的通道上，此位是预加载的。如果 CCPC 位是在 TIMx_CR2 寄存器中设置，CC1P 仅在通讯事件产生时加载新的值。* • *一旦 LOCK 级别（TIMx_BDTR 寄存器中的 LOCK 位）设为 3 或 2，则该位不能被修改。*
位 0	CC1E：捕获/比较 1 输出使能（Capture/Compare 3 output enable） • CC1 通道配置为输出： ○ 0：关闭 OC1 禁止输出。 ○ 1：开启 OC1 信号输出到对应的输出引脚。 • CC1 通道配置为输入： 该位决定了计数器的值是否能捕获入 TIMx_CCR1 寄存器。 ○ 0：捕获禁止； ○ 1：捕获使能。 *注意：在具有互补输出的通道上，此位是预加载的。如果 CCPC 位是在 TIMx_CR2 寄存器中设置，CC1E 仅在通讯事件产生时加载新的值。*

表 17-3 标准 OCx 通道的输出控制位

CCxE 位	OCx 输出状态
0	禁止输出（OCx=0，OCx_EN=0）
1	OCx=OCxREF +极性，OCx_EN=1

注意：连接到标准 OCx 通道的外部 I/O 引脚状态，取决于 OCx 通道状态和 GPIO 以及 AFIO 寄存器。

17.3.10 计数器寄存器（TIMx_CNT）（x=2,3,4,5）

偏移地址：0x24

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw															

位 15:0	CNT[15:0]：计数器的值（Counter value high bits of TIMx）

17.3.11 预分频器寄存器（TIMx_PSC）（x=2,3,4,5）

偏移地址：0x28

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw															

位 15:0	PSC[15:0]：预分频器的值（Prescaler value） 计数器的时钟频率 CK_CNT 等于 f_{CK_PSC}/（PSC[15:0]+1）。PSC 包含了当更新事件产生时装入当前预分频器寄存器的值。

17.3.12 自动重装载寄存器（TIMx_ARR）（x=2,3,4,5）

偏移地址：0x2C

复位值：0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw															

位 15:0	ARR[15:0]：自动重装载的值（Auto-reload value） ARR 包含了将要传送至实际的自动重装载寄存器的数值。详细参考章节有关 ARR 的更新和动作。当自动重装载的值为空时，计数器不工作。

17.3.13 捕获/比较寄存器 1（TIMx_CCR1）（x=2,3,4,5）

偏移地址：0x34

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw															

位 15:0	CCR1[15:0]：捕获/比较 1 的值（Capture/Compare 1 value） • 若 CC1 通道配置为输出： CCR1 包含了装入当前捕获/比较 1 寄存器的值（预装载值）。 如果在 TIMx_CCMR1 寄存器（OC1PE 位）中未选择预装载特性，写入的数值会被立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 1 寄存器中。当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC1 端口上产生输出信号。 • 若 CC1 通道配置为输入： CCR1 包含了由上一次输入捕获 1 事件（IC1）传输的计数器值。CCR1 是只读寄存器。

17.3.14 捕获/比较寄存器 2（TIMx_CCR2）（x=2,3,4,5）

偏移地址：0x38

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw															

位 15:0	CCR2[15:0]：捕获/比较 2 的值（Capture/Compare 2 value） • 若 CC2 通道配置为输出： CCR2 包含了装入当前捕获/比较 2 寄存器的值（预装载值）。 如果在 TIMx_CCMR2 寄存器（OC2PE 位）中未选择预装载特性，写入的数值会被立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 2 寄存器中。当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC2 端口上产生输出信号。 • 若 CC2 通道配置为输入： CCR2 包含了由上一次输入捕获 2 事件（IC2）传输的计数器值。CCR2 是只读寄存器。

17.3.15 捕获/比较寄存器 3（TIMx_CCR3）（x=2,3,4,5）

偏移地址：0x3C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw															

位 15:0	CCR3[15:0]：捕获/比较 3 的值（Capture/Compare 3 value） • 若 CC3 通道配置为输出： CCR3 包含了装入当前捕获/比较 3 寄存器的值（预装载值）。 如果在 TIMx_CCMR3 寄存器（OC3PE 位）中未选择预装载特性，写入的数值会被立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 3 寄存器中。当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC3 端口上产生输出信号。 • 若 CC3 通道配置为输入： CCR3 包含了由上一次输入捕获 3 事件（IC3）传输的计数器值。CCR3 是只读寄存器。

17.3.16 捕获/比较寄存器 4（TIMx_CCR4）（x=2,3,4,5）

偏移地址：0x40

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw															

位 15:0	CCR4[15:0]：捕获/比较 4 的值（Capture/Compare 4 value） • 若 CC4 通道配置为输出： CCR4 包含了装入当前捕获/比较 4 寄存器的值（预装载值）。 如果在 TIMx_CCMR4 寄存器（OC4PE 位）中未选择预装载特性，写入的数值会被立即传输至当前寄存器中。否则只有当更新事件发生时，此预装载值才传输至当前捕获/比较 4 寄存器中。当前捕获/比较寄存器参与同计数器 TIMx_CNT 的比较，并在 OC4 端口上产生输出信号。 • 若 CC4 通道配置为输入： CCR4 包含了由上一次输入捕获 4 事件（IC4）传输的计数器值。CCR4 是只读寄存器。

17.3.17 DMA 控制寄存器（TIMx_DCR）（x=2,3,4,5）

偏移地址：0x48

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res			DBL[4:0]					Res			DBA[4:0]				
			rw								rw				

位 15:13	Res：保留 必须保持复位值。
位 12:8	DBL[4:0]：DMA 连续传送长度（DMA burst length） 该位域定义了 DMA 在连续模式下的传送长度（当对 TIMx_DMAR 寄存器进行读或写时，定时器则进行一次连续传送），即：定义传输的字节数目： • 00000：1 个字节 • 00001：2 个字节 • 00010：3 个字节 • ... • 10001：18 个字节
位 7:5	Res：保留 必须保持复位值。
位 4:0	DBA[4:0]：DMA 基地址（DMA base address） 该位域定义了 DMA 在连续模式下的基地址（当对 TIMx_DMAR 寄存器进行读或写时），DBA 定义为从 TIMx_CR1 寄存器所在地址开始的偏移量： • 00000：TIMx_CR1 • 00001：TIMx_CR2 • 00010：TIMx_SMCR ...

17.3.18 连续模式的 DMA 地址寄存器（TIMx_DMAR）（x=2,3,4,5）

偏移地址：0x4C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw															

位 15:0	DMAB[15:0]：DMA 连续传送寄存器（DMA register for burst accesses） 对 TIMx_DMAR 寄存器的读或写会导致对以下地址所在寄存器的存取操作： （TIMx_CR1 地址）+（DBA +DMA 索引）X4，其中： • TIMx_CR1 地址是控制寄存器 1（TIMx_CR1）所在的地址 • DBA 是 TIMx_DCR 寄存器中定义的基地址 • DMA 索引是由 DMA 自动控制的偏移量，它取决于 TIMx_DCR 寄存器中定义的 DBL。 *说明：关于 DMA 连续传送功能使用方法示例，可参见“16.3.20 TIM1/TIM8 连续模式的 DMA 地址寄存器（TIMx_DMAR）（x=1,8）”。*

18 基本定时器（TIM6 和 TIM7）

基本定时器 TIM6 和 TIM7 各自包含一个 16 位自动装载计数器，由各自的可编程预分频器驱动。

它们可以作为通用定时器提供时间基准，但特别的是它们还可为数模转换器（DAC）提供时钟。实际上，它们在芯片内部直接连接到 DAC 并可以通过触发输出直接驱动 DAC。

这 2 个定时器是互相独立的，不共享任何资源。

基本定时器主要是用于产生 DAC 触发信号，也可当成通用的 16-bit 定时器产生 CPU 中断或发起 DMA 请求。

18.1 TIM6 和 TIM7 的主要特性

TIM6 和 TIM7 定时器的主要功能包括：

- 16 位自动重装载累加计数器
- 16 位可编程（可实时修改）预分频器，用于对输入的时钟按系数为 1～65536 之间的任意数值分频
- 触发 DAC 的同步电路
- 在更新事件（计数器溢出）时产生中断/DMA 请求

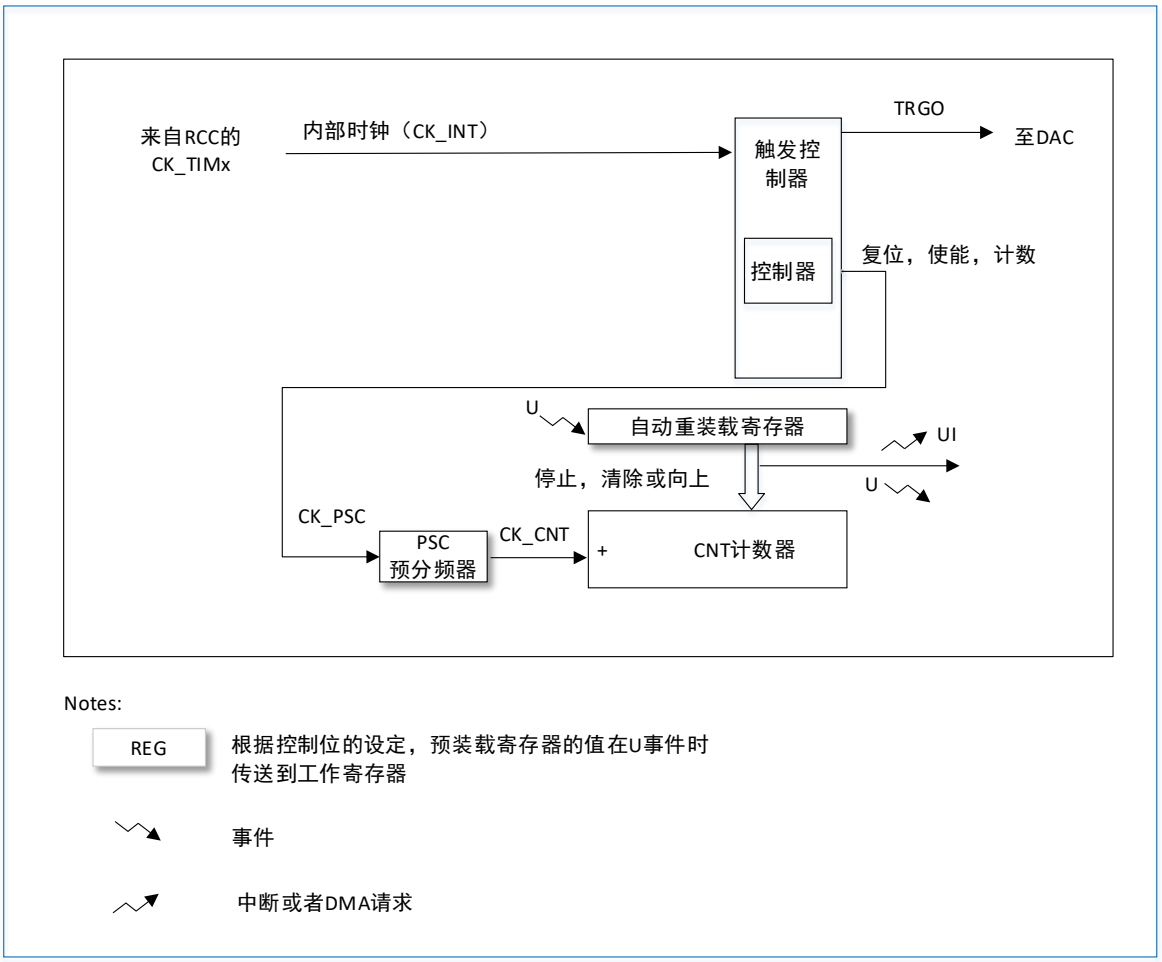


图 18-1 基本定时器框图

18.2 TIM6 和 TIM7 的功能

18.2.1 时基单元

这个可编程定时器的主要部分是一个带有自动重装载的 16 位累加计数器，计数器的时钟通过一个预分频器得到。

软件可以读写计数器、自动重装载寄存器和预分频寄存器，即使计数器运行时也可以操作。时基单元包含：

- 计数器寄存器（TIMx_CNT）
- 预分频寄存器（TIMx_PSC）
- 自动重装载寄存器（TIMx_ARR）自动重装载寄存器是预加载的，每次读写自动重装载寄存器时，实际上是通过读写预加载寄存器实现。根据 TIMx_CR1 寄存器中的自动重装载预加载使能位（ARPE），写入预加载寄存器的内容能够立即或在每次更新事件时，传送到它的影子寄存器。当 TIMx_CR1 寄存器的 UDIS 位为'0'，则每当计数器达到溢出值时，硬件发出更新事件；软件也可以产生更新事件；关于更新事件的产生，随后会有详细的介绍。

计数器由预分频输出 CK_CNT 驱动，设置 TIMx_CR1 寄存器中的计数器使能位（CEN）使能计数器计数。

注意：实际的设置计数器使能信号 CNT_EN 相对于 CEN 滞后一个时钟周期。

18.2.1.1 预分频器

预分频可以以系数介于 1 至 65536 之间的任意数值对计数器时钟分频。它是通过一个 16 位寄存器（TIMx_PSC）的计数实现分频。因为 TIMx_PSC 控制寄存器具有缓冲，可以在运行过程中改变它的数值，新的预分频数值将在下一个更新事件时起作用。

以下两图是在运行过程中改变预分频系数的例子。

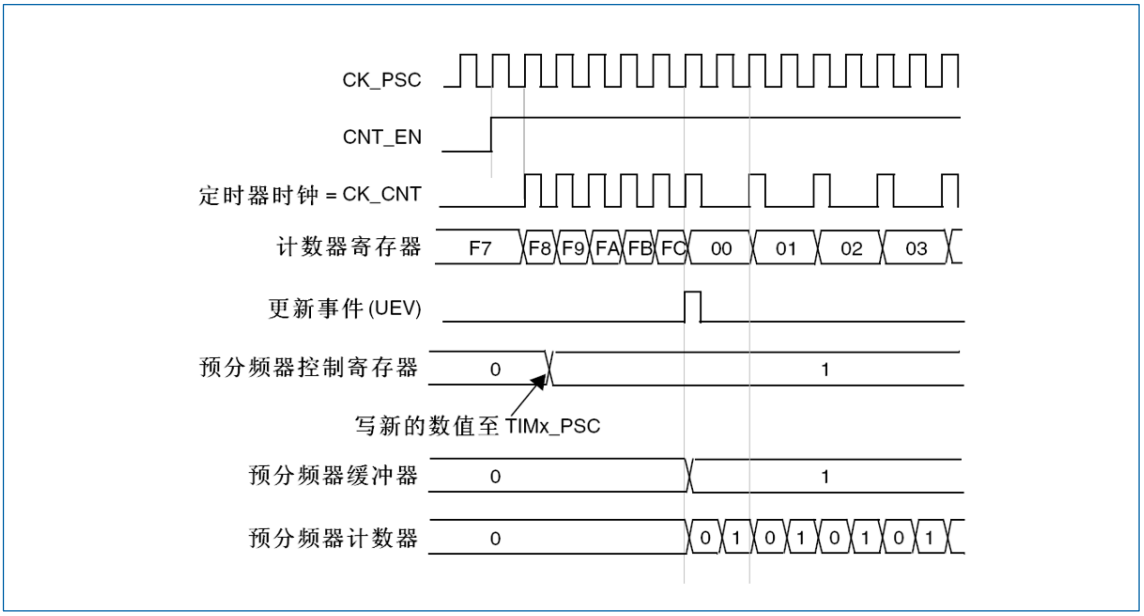


图 18-2 预分频系数从 1 变到 2 的计数器时序图

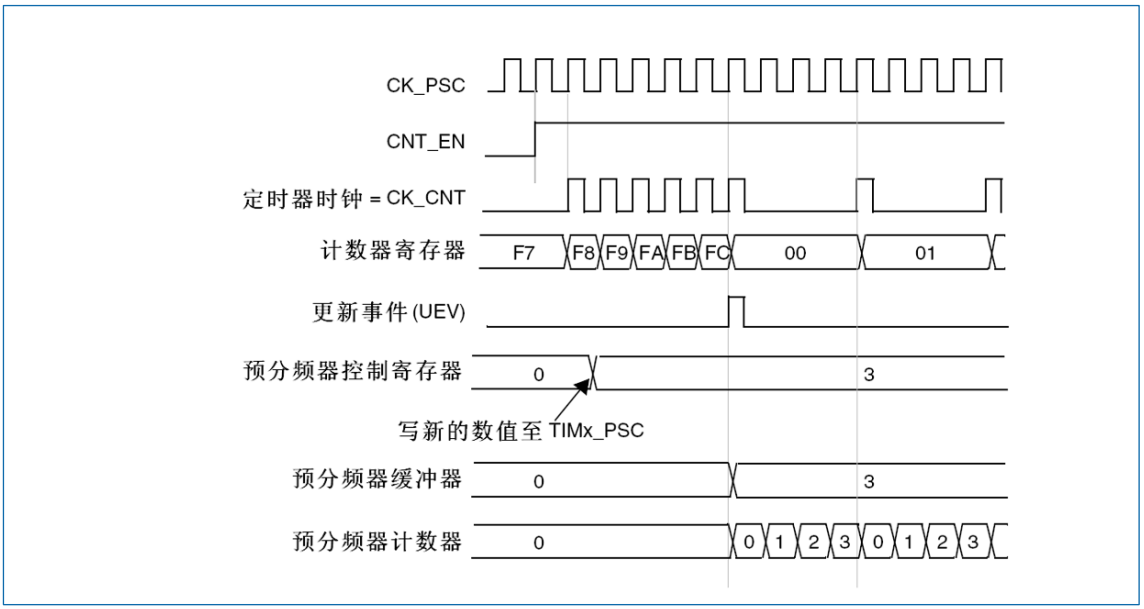


图 18-3 预分频系数从 1 变到 4 的计数器时序图

18.2.2 计数模式

计数器从 0 累加计数到自动重装载数值（TIMx_ARR 寄存器），然后重新从 0 开始计数并产生一个计数器溢出事件。

每次计数器溢出时可以产生更新事件；（通过软件或使用从模式控制器）设置 TIMx_EGR 寄存器的 UG 位也可以产生更新事件。

设置 TIMx_CR1 中的 UDIS 位可以禁止产生 UEV 事件，这可以避免在写入预加载寄存器时更改影子寄存器。在清除 UDIS 位为'0'之前，将不再产生更新事件，但计数器和预分频器依然会在应产生更新事件时重新从 0 开始计数（但预分频系数不变）。另外，如果设置了 TIMx_CR1 寄存器中的URS（选择更新请求），设置 UG 位可以产生一次更新事件 UEV，但不设置 UIF 标志（即没有中断或DMA 请求）。

当发生一次更新事件时，所有寄存器会被更新并（根据 URS 位）设置更新标志（TIMx_SR 寄存器的 UIF 位）：

- 传送预装载值（TIMx_PSC 寄存器的内容）至预分频器的缓冲区。
- 自动重装载影子寄存器被更新为预装载值（TIMx_ARR）。

以下是一些在 TIMx_ARR=0x36 时不同时钟频率下计数器工作的图示例子。

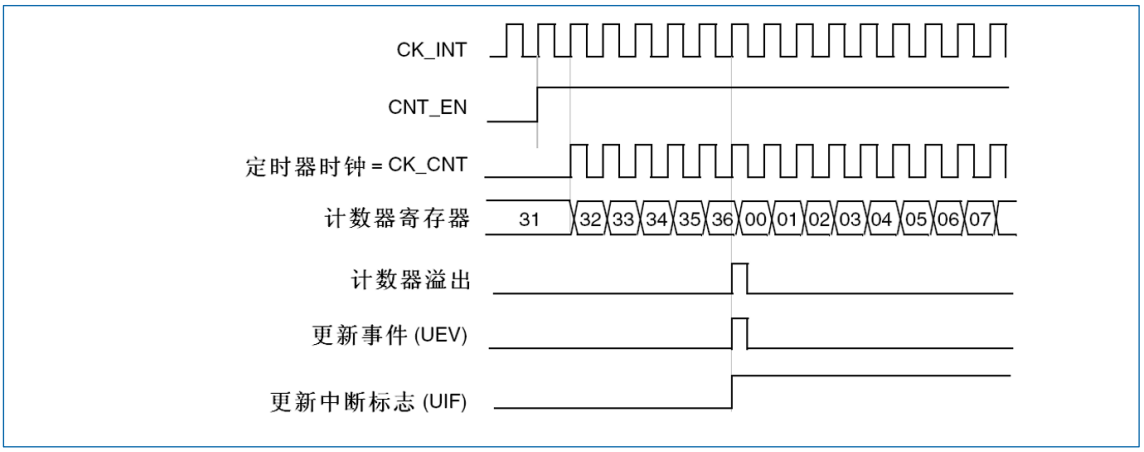


图 18-4 计数器时序图，内部时钟分频系数为 1

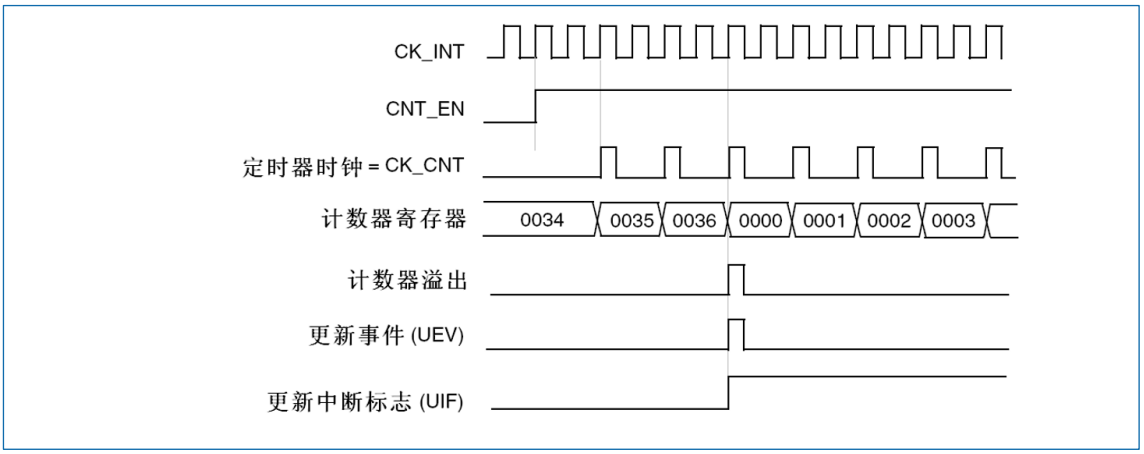


图 18-5 计数器时序图，内部时钟分频系数为 2

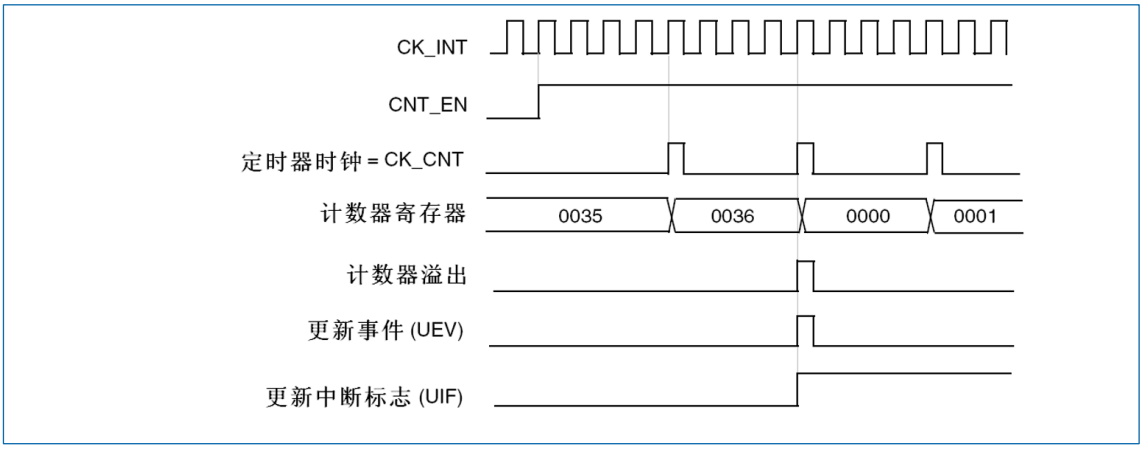


图 18-6 计数器时序图，内部时钟分频系数为 4

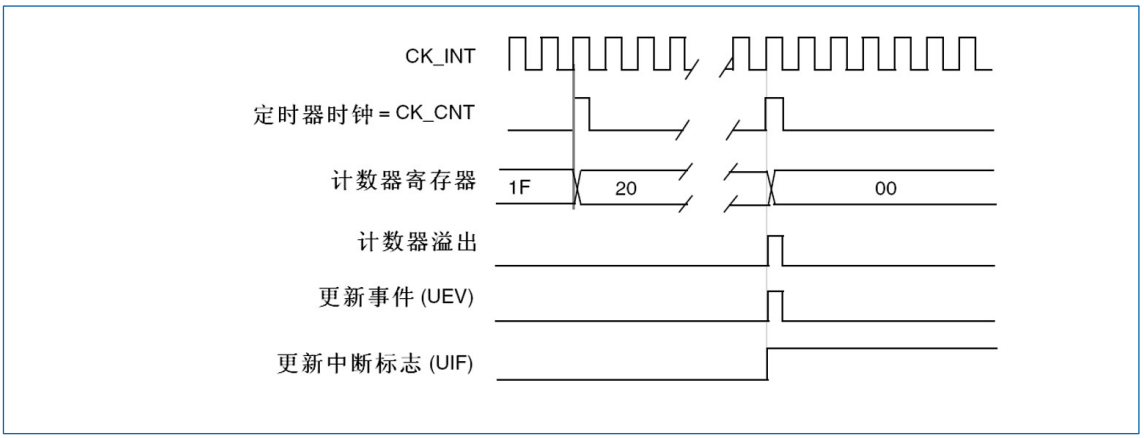


图 18-7 计数器时序图，内部时钟分频系数为 N

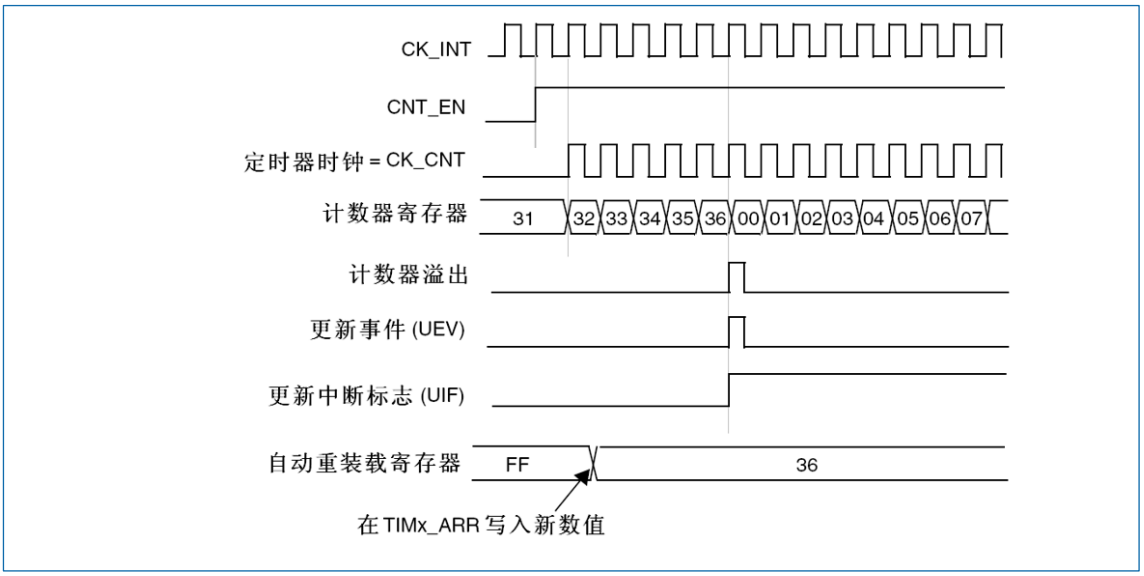


图 18-8 计数器时序图，当 ARPE=0 时的更新事件（TIMx_ARR 没有预装载）

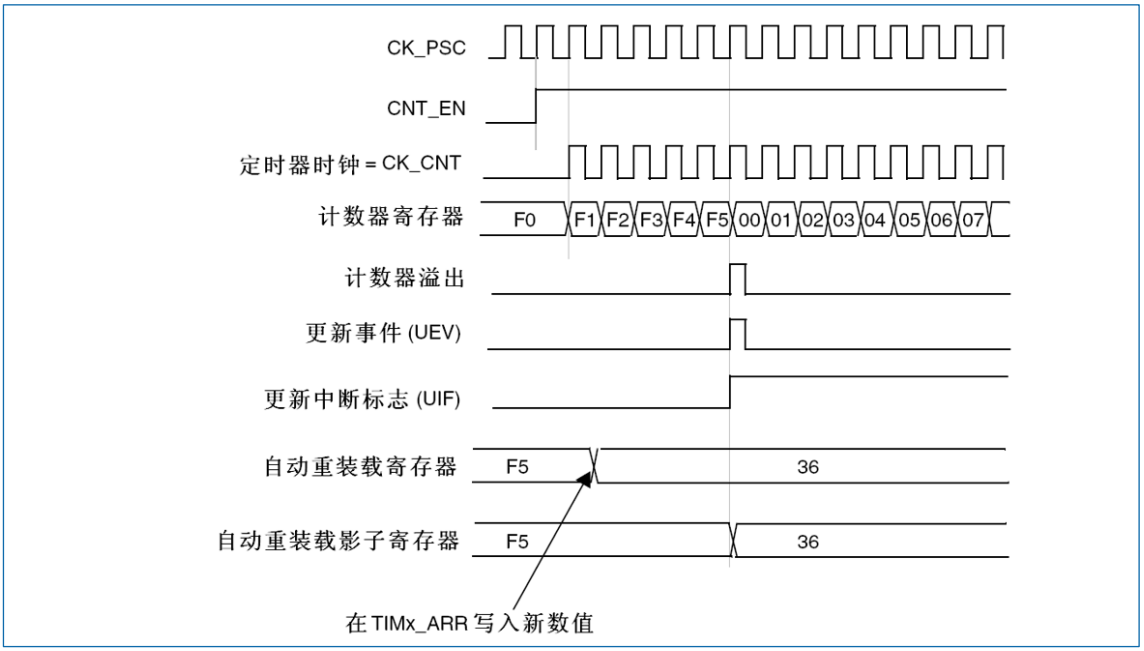


图 18-9 计数器时序图，当 ARPE=1 时的更新事件（预装载 TIMx_ARR）

18.2.3 时钟源

计数器的时钟由内部时钟（CK_INT）提供。

TIMx_CR1 寄存器的 CEN 位和 TIMx_EGR 寄存器的 UG 位是实际的控制位，（除了 UG 位被自动清除外）只能通过软件改变它们。一旦置 CEN 位为'1'，内部时钟即向预分频器提供时钟。

下图为控制电路和向上计数器在普通模式下，没有预分频器时的操作。

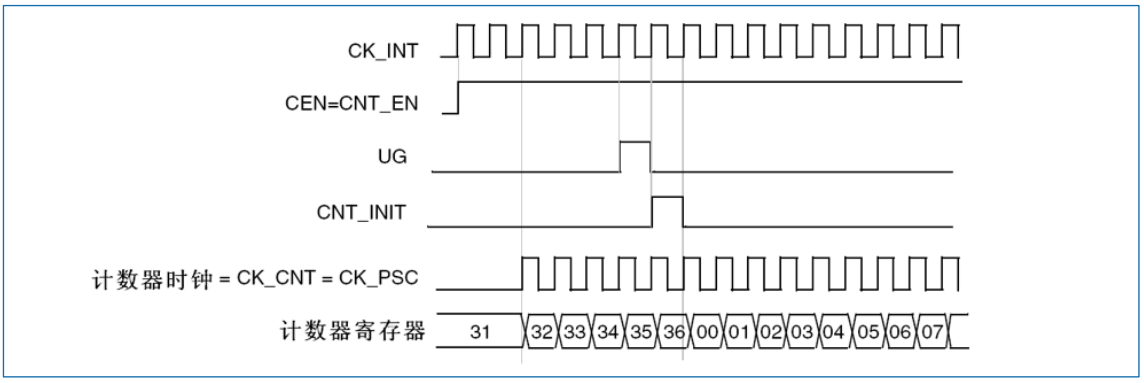


图 18-10 普通模式时序图，内部时钟分频系数为 1

18.2.4 调试模式

当微控制器进入调试模式（Cortex-M3 核心停止）时，根据 DBG 模块中的配置位 DBG_TIMx_STOP 的设置，TIMx 计数器或者继续计数或者停止工作。详见章节：“支持定时器、看门狗、CAN 和 I2C 的调试”。

18.3 TIM6/7 寄存器

基地址：（TIM6, TIM7）=（0x4000 1000, 0x4000 1400）

空间大小：（TIM6, TIM7）=（0x400, 0x400）

可以用半字（16 位）或字（32 位）的方式操作这些外设寄存器。

18.3.1 TIM6/TIM7 控制寄存器 1（TIMx_CR1）（x=6,7）

偏移地址：0x00

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								ARPE	Res			OPM	URS	UDIS	CEN
								rw				rw	rw	rw	rw

位 15:8	Res：保留 必须保持复位值。
位 7	ARPE：自动重装载预装载使能（Auto-reload preload enable） • 0：TIMx_ARR 寄存器没有缓冲 • 1：TIMx_ARR 寄存器具有缓冲
位 6:4	Res：保留 必须保持复位值。
位 3	OPM：单脉冲模式（One pulse mode） • 0：在发生更新事件时，计数器不停止 • 1：在发生下次更新事件时，计数器停止计数（清除 CEN 位）
位 2	URS：更新请求源（Update request source） 该位由软件设置和清除，以选择 UEV 事件的请求源。 • 0：如果使能了中断或 DMA，以下任一事件可以产生一个更新中断或 DMA 请求： o 计数器上溢或下溢 o 设置 UG 位 o 通过从模式控制器产生的更新 • 1：如果使能了中断或 DMA，只有计数器上溢或下溢可以产生更新中断或 DMA 请求

位 1	UDIS：禁止更新（Update disable） 该位由软件设置和清除，以使能或禁止 UEV 事件的产生。 • 0：UEV 使能 更新事件（UEV）可以由下列事件产生： o 计数器上溢或下溢 o 设置 UG 位 o 通过从模式控制器产生的更新产生更新事件后，带缓冲的寄存器被加载为预加载数值 • 1：禁止 UEV 不产生更新事件（UEV），影子寄存器保持它的内容（ARR、PSC）。但是如果设置了 UG 位或从模式控制器产生了一个硬件复位，则计数器和预分频器将被重新初始化
位 0	CEN：计数器使能（Counter enable） • 0：关闭计数器 • 1：使能计数器 *注意：* • *门控模式只能在软件已经设置了 CEN 位时有效，而触发模式可以自动地由硬件设置 CEN 位。* • *在单脉冲模式下，当产生更新事件时 CEN 被自动清除。*

18.3.2 TIM6/TIM7 控制寄存器 2（TIMx_CR2）（x=6,7）

偏移地址：0x04

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res									MMS[2:0]			Res			
									rw						

位 15:7	Res：保留 必须保持复位值。
位 6:4	MMS[2:0]：主模式选择（Master mode selection） 该位域用于选择在主模式下向从定时器发送的同步信息（TRGO），有以下几种组合： • 000：复位 使用 TIMx_EGR 寄存器的 UG 位作为触发输出（TRGO）。如果触发输入产生了复位（从模式控制器配置为复位模式），则相对于实际的复位信号，TRGO 上的信号有一定的延迟。 • 001：使能 计数器使能信号 CNT_EN 被用作为触发输出（TRGO）。它可用于在同一时刻启动多个定时器，或控制使能从定时器的时机。计数器使能信号是通过 CEN 控制位和配置为门控模式时的触发输入的“或”逻辑产生。当计数器使能信号是通过触发输入控制时，在 TRGO 输出上会有一些延迟，除非选择了主/从模式（见 TIMx_SMCR 寄存器的 MSM 位）。 • 010：更新 更新事件被用作为触发输出（TRGO）。例如一个主定时器可以作为从定时器的预分频器使用。
位 3:0	Res：保留 必须保持复位值。

18.3.3 TIM6/TIM7 DMA/中断使能寄存器（TIMx_DIER）（x=6,7）

偏移地址：0x0C

复位值：0x0000

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

位 15:9	Res：保留 必须保持复位值。
位 8	UDE：更新 DMA 请求使能（Update DMA request enable） • 0：禁止更新 DMA 请求 • 1：使能更新 DMA 请求
位 7:1	Res：保留 必须保持复位值。
位 0	UIE：更新中断使能（Update interrupt enable） • 0：禁止更新中断 • 1：使能更新中断

18.3.4 TIM6/TIM7 状态寄存器（TIMx_SR）（x=6,7）

偏移地址：0x10

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res															UIF
															rc_w0

位 15:1	Res：保留 必须保持复位值。
位 0	UIF：更新中断标志（Update interrupt flag） 硬件在更新中断时设置该位，它由软件清除： • 0：没有产生更新 • 1：产生了更新中断 下述情况下由硬件设置该位： • 计数器产生上溢或下溢并且 TIMx_CR1 中的 UDIS=0 • 如果 TIMx_CR1 中的 URS=0 并且 UDIS=0，当使用 TIMx_EGR 寄存器的 UG 位重新初始化计数器 CNT 时

18.3.5 TIM6/TIM7 事件产生寄存器（TIMx_EGR）（x=6,7）

偏移地址：0x14

复位值：0x0000

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

位 15:1	Res：保留 必须保持复位值。
位 0	UG：产生更新事件（Update generation） 该位由软件设置，由硬件自动清除。

	• 0：无作用 • 1：重新初始化定时器的计数器并产生对寄存器的更新 *注意：预分频器也被清除（但预分频系数不变）。如果 TIMx_CR1 中的 URS=0 并且 UDIS=0，当使用 TIMx_EGR 寄存器的 UG 位重新初始化计数器 CNT 时。*

18.3.6 TIM6/TIM7 计数器寄存器（TIMx_CNT）（x=6,7）

偏移地址：0x24

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw															

位 15:0	CNT[15:0]：计数器数值（Counter value）

18.3.7 TIM6/TIM7 预分频器寄存器（TIMx_PSC）（x=6,7）

偏移地址：0x28

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw															

位 15:0	PSC[15:0]：预分频器数值（Prescaler value） 计数器的时钟频率 CK_CNT 等于 fCK_PSC/（PSC[15:0]+1）。 在每一次更新事件时，PSC 的数值被传送到实际的预分频寄存器中。

18.3.8 TIM6/TIM7 自动重装载寄存器（TIMx_ARR）（x=6,7）

偏移地址：0x2C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw															

位 15:0	ARR[15:0]：自动重装载数值（Auto-reload value） ARR 的数值将传送到实际的自动重装载寄存器中。关于 ARR 的更新和作用，详见章节：时基单元。 如果自动重装载数值为 0，则计数器停止。

19 实时时钟（RTC）

实时时钟是一个独立的定时器。RTC 模块拥有一组连续计数的计数器，在相应软件配置下，可提供时钟日历的功能。修改计数器的值可以重新设置系统当前的时间和日期。

RTC 模块和时钟配置系统（RCC_BDCR 寄存器）处于备份区域，即在系统复位或从待机模式唤醒后，RTC 的设置和时间维持不变。

系统复位后，对备份寄存器和 RTC 的访问被禁止，这是为了防止对备份区域（BKP）的意外写操作。执行以下操作将使能对备份寄存器和 RTC 的访问：

1. 设置寄存器 RCC_APB1ENR 的 PWREN 和 BKPEN 位，使能电源和备份接口时钟。
2. 设置寄存器 PWR_CR 的 DBP 位，使能对备份寄存器和 RTC 的访问。

19.1 主要特性

- 可编程的预分频系数：分频系数最高为 2^{20}。
- 32 位的可编程计数器，可用于较长时间段的测量。
- 2 个独立的时钟：用于 APB1 接口的 PCLK1 和 RTC 时钟（必须满足：RTCCLK≤PCLK1/4（或更低））。
- 可以选择以下三种 RTC 的时钟源：
  - HSE 时钟除以 128
  - LSE 振荡器时钟
  - LSI 振荡器时钟（详见章节：“8.2.8 RTC 时钟”。）
- 2 个独立的复位类型：
  - APB1 接口由系统复位
  - RTC 核心（预分频器、闹钟、计数器和分频器）只能由备份域复位（详见章节：“8.1.3 备份域复位”）
- 3 个专门的可屏蔽中断：
  - 闹钟中断，用来产生一个软件可编程的闹钟中断。
  - 秒中断，用来产生一个可编程的周期性中断信号（最长可达 1 秒）。
  - 溢出中断，指示内部可编程计数器溢出并回转为 0 的状态。
- 一个唤醒计时器（Wakeup timer）用于周期性唤醒

19.2 RTC 功能描述

19.2.1 概述

RTC 由两个主要部分组成（参见下图）。

- 第一部分为 APB1 接口，用来和 APB1 总线相连。此单元还包含一组 16 位寄存器，可通过 APB1 总线对其进行读写操作（参见章节：“19.3 RTC 寄存器”）。APB1 接口由 APB1 总线时钟驱动，用来与 APB1 总线接口。
- 另一部分为 RTC 核心，由一组可编程计数器组成，分成两个主要模块：
  - 第一个模块是 RTC 的预分频模块，它可编程产生周期最长为 1 秒的 RTC 时间基准 TR_CLK。RTC 的预分频模块包含了一个 20 位的可编程分频器（RTC 预分频器）。如果在RTC_CR寄存器中设置了相应的允许位，则在每个TR_CLK 周期中，RTC 产生一个中断（秒中断）。
  - 第二个模块是一个 32 位的可编程计数器，可被初始化为当前的系统时间。系统时间按

TR_CLK 周期累加并与存储在 RTC_ALR 寄存器中的可编程时间相比较，如果 RTC_CR 控制寄存器中设置了相应允许位，比较匹配时将产生一个闹钟中断。

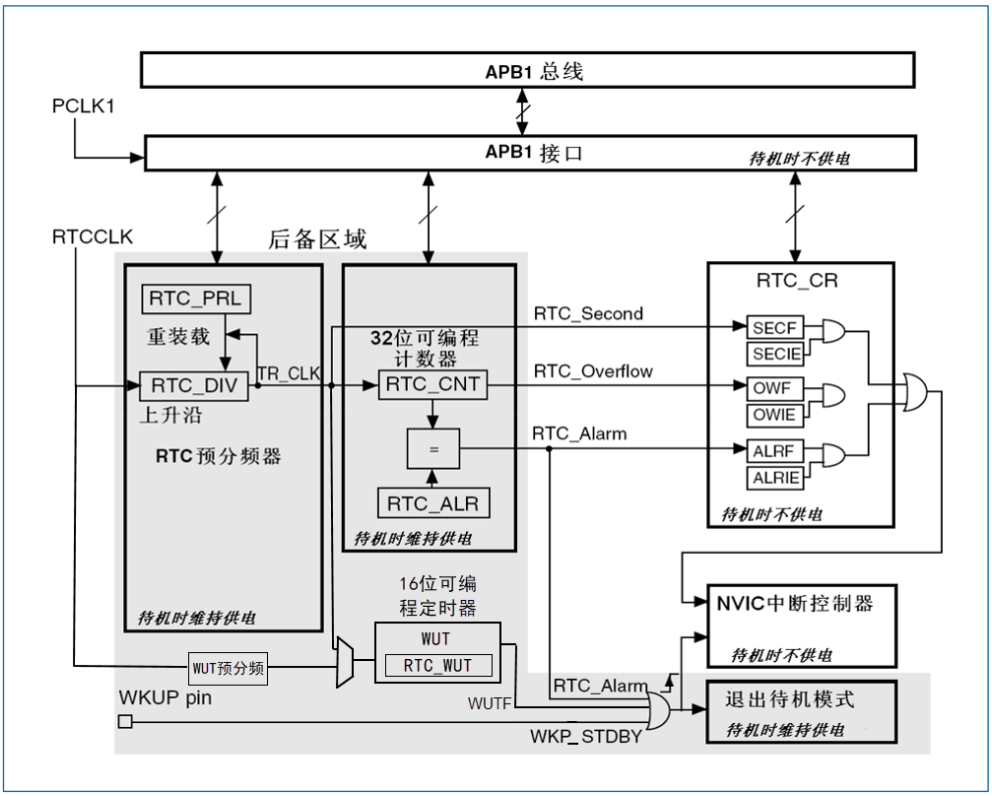


图 19-1 简化的 RTC 框图

19.2.2 复位过程

除了 RTC_PRL、RTC_ALR、RTC_CNT 和 RTC_DIV 寄存器外，所有的系统寄存器都由系统复位或电源复位进行异步复位。

RTC_PRL、RTC_ALR、RTC_CNT 和 RTC_DIV 寄存器仅能通过备份域复位信号复位，详见章节：“8.1.3 备份域复位”。

19.2.3 读 RTC 寄存器

RTC 核完全独立于 RTC APB1 接口。软件通过 APB1 接口访问 RTC 的预分频值、计数器值和闹钟值。但是，相关的可读寄存器只在与 RTC APB1 时钟进行重新同步的 RTC 时钟的上升沿被更新。RTC 标志也是如此的。

这意味着，如果 APB1 接口曾经被关闭，而读操作又是在刚刚重新开启 APB1 接口之后，在第一次的内部寄存器更新之前，那么从 APB1 上读出的 RTC 寄存器数值可能被破坏了（通常读到 0）。下述几种情况下能够发生这种情形：

- 发生系统复位或电源复位
- 系统刚从待机模式唤醒（参见：“低功耗模式”）。
- 系统刚从停机模式唤醒（参见：“低功耗模式”）。

所有以上情况中，当 APB1 接口被禁止时（复位、无时钟或断电），RTC 核仍保持运行状态。

因此，在读取 RTC 寄存器时，如果 RTC 的 APB1 接口曾被禁止，那么软件首先必须等待 RTC_CRL 寄存器中的 RSF 位（寄存器同步标志）被硬件置’1’。

注意：RTC 的 APB1 接口不受 WFI 和 WFE 等低功耗模式的影响。

19.2.4 配置 RTC 寄存器

必须先设置RTC_CRL寄存器中的CNF 位，使RTC 进入配置模式后，才能写入RTC_PRL、RTC_CNT、RTC_ALR 寄存器。

另外，对RTC任何寄存器的写操作，都必须在前一次写操作结束后再进行。可以通过查询RTC_CR 寄存器中的 RTOFF 状态位，判断 RTC 寄存器是否处于更新中。只有当 RTOFF 状态位是'1'时，才可以写入 RTC 寄存器。

配置过程：

1. 查询 RTOFF 位，直到 RTOFF 的值变为'1'。
2. 置 CNF 值为 1，进入配置模式。
3. 对一个或多个 RTC 寄存器进行写操作。
4. 清除 CNF 标志位，退出配置模式。
5. 查询 RTOFF，直至 RTOFF 位变为'1'以确认写操作已经完成。

只有当 CNF 标志位被清除时，写操作才能进行，这个过程至少需要 3 个 RTCCLK 周期。

19.2.5 RTC 标志的设置

在每一个 RTC 核的时钟周期中，在更改 RTC 计数器之前，RTC 秒标志（SECF）被设置。在计数器到达 0x0000 之前的最后一个 RTC 时钟周期中，设置 RTC 溢出标志（OWF）。

在计数器的值到达闹钟寄存器的值加 1（RTC_ALR+1）之前的最后一个 RTC 时钟周期中，设置 RTC_Alarm 和 RTC 闹钟标志（ALRF）。对 RTC 闹钟的写操作必须使用下述过程之一与 RTC 秒标志（SECF）同步：

- 使用 RTC 闹钟中断，并在中断处理程序中修改 RTC 闹钟和/或 RTC 计数器。
- 等待 RTC 控制寄存器中的 SECF 位被设置，再更改 RTC 闹钟和/或 RTC 计数器。

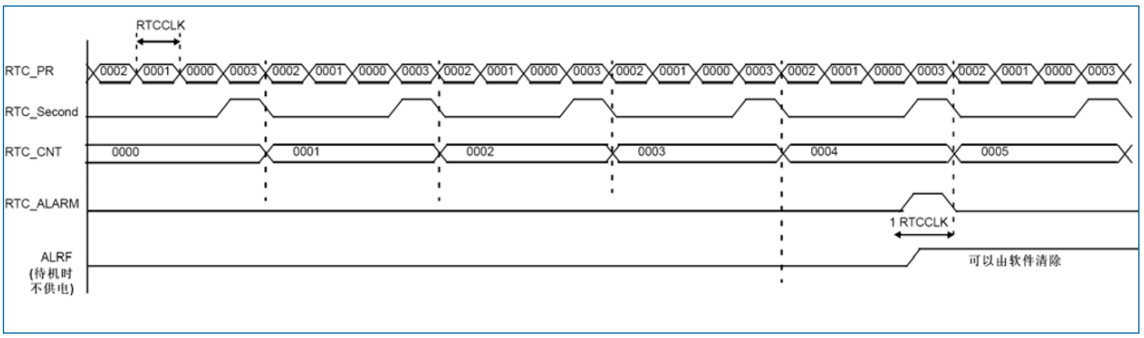


图 19-2 RTC 秒和闹钟波形图示例，PR=0003，ALARM=00004

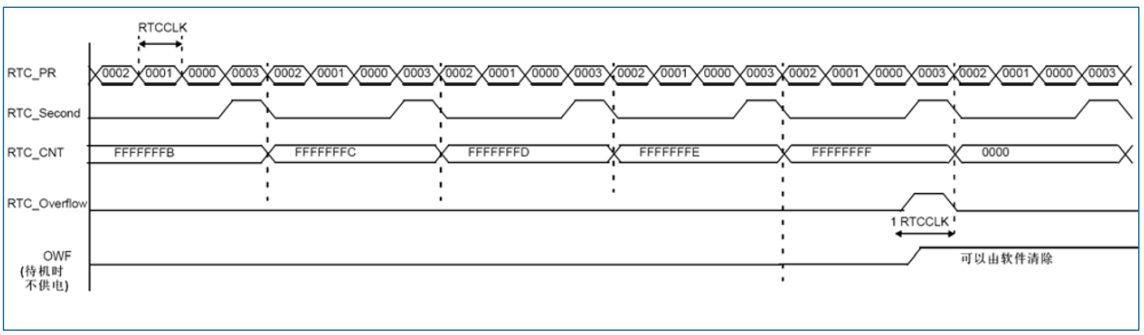


图 19-3 RTC 溢出波形图示例，PR=0003

19.3 RTC 寄存器

基地址：0x4000 2800

空间大小：0x400

可以用半字（16 位）或字（32 位）的方式操作这些外设寄存器。

19.3.1 RTC 控制寄存器高位（RTC_CRH）

偏移地址：0x00

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSM_MD	Res						WUTIE	Res					OWIE	ALRIE	SECIE
rw							rw						rw	rw	rw

位 15	RSM_MD：RSM 模式，用于设置 RTC_DIVx 的重载条件。（RSM mode in which the reload condition of RTCDIVx can be set） 用于设置 RTC_DIVx 的清除条件。 • 0：写 RTC_PRLx 或 RTC_CNTx，均可让 RTC_DIVx 重载。 • 1：写 RTC_PRLx 可让 RTC_DIVx 重载，写 RTC_CNTx 不可让 RTC_DIVx 重载。
位 14:9	Res：保留 必须保持复位值。
位 8	WUTIE：允许唤醒计数器中断（Wake-up counter interrupt enable）
位 7:3	Res：保留 必须保持复位值。
位 2	OWIE：允许溢出中断位（Overflow interrupt enable） • 0：屏蔽（不允许）溢出中断 • 1：允许溢出中断
位 1	ALRIE：允许闹钟中断（Alarm interrupt enable） • 0：屏蔽（不允许）闹钟中断 • 1：允许闹钟中断
位 0	SECIE：允许秒中断（Second interrupt enable） • 0：屏蔽（不允许）秒中断 • 1：允许秒中断

这些位用来屏蔽中断请求。

注意：系统复位后所有的中断被屏蔽，因此可通过写 RTC 寄存器来确保在初始化后没有挂起的中断请求。当外设正在完成前一次写操作时（标志位 RTOFF=0），不能对 RTC_CRH 寄存器进行写操作。RTC 功能由这个控制寄存器控制。一些位的写操作必须经过一个特殊的配置过程来完成（详见"19.2.4 配置 RTC 寄存器"）。

19.3.2 RTC 控制寄存器低位（RTC_CRL）

偏移地址：0x04

复位值：0x0020

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

Res	WUTF	Res	RTOFF	CNF	RSF	OWF	ALRF	SECF
	rc_w0		r	rw	rc_w0	rc_w0	rc_w0	rc_w0

位 15:9	Res：保留 必须保持复位值。
位 8	WUTF：唤醒定时器中断标志（Wake-up counter interrupt flag）
位 7:6	Res：保留 必须保持复位值。
位 5	RTOFF：RTC 操作关闭（RTC operation OFF） RTC 模块利用这位来指示对其寄存器进行的最后一次操作的状态，指示操作是否完成。若此位为'0'，则表示无法对任何的 RTC 寄存器进行写操作。此位为只读位。 • 0：上一次对 RTC 寄存器的写操作仍在进行 • 1：上一次对 RTC 寄存器的写操作已经完成
位 4	CNF：配置标志（Configuration flag） 此位必须由软件置'1'以进入配置模式，从而允许向 RTC_CNT、RTC_ALR 或 RTC_PRL 寄存器写入数据。只有当此位在被置'1'并重新由软件清零后，才会执行写操作。 • 0：退出配置模式（开始更新 RTC 寄存器） • 1：进入配置模式
位 3	RSF：寄存器同步标志（Registers synchronized flag） 每当 RTC_CNT 寄存器和 RTC_DIV 寄存器由软件更新或清零时，此位由硬件置'1'。在 APB1 复位后，或 APB1 时钟停止后，此位必须由软件清零。要进行任何的读操作之前，用户程序必须等待这位被硬件置'1'，以确保 RTC_CNT、RTC_ALR 或 RTC_PRL 已经被同步。 • 0：寄存器尚未被同步 • 1：寄存器已经被同步
位 2	OWF：溢出标志（Overflow flag） 当 32 位可编程计数器溢出时，此位由硬件置'1'。如果 RTC_CRH 寄存器中 OWIE=1，则产生中断。此位只能由软件清零。对此位写'1'是无效的。 • 0：无溢出 • 1：32 位可编程计数器溢出
位 1	ALRF：闹钟标志（Alarm flag） 当 32 位可编程计数器达到 RTC_ALR 寄存器所设置的预定值，此位由硬件置'1'。如果 RTC_CRH 寄存器中 ALRIE=1，则产生中断。此位只能由软件清零。对此位写'1'是无效的。 • 0：无闹钟 • 1：有闹钟
位 0	SECF：秒标志（Second flag） 当 32 位可编程预分频器溢出时，此位由硬件置'1'同时 RTC 计数器加 1。因此，此标志为分辨率可编程的 RTC 计数器提供一个周期性的信号（通常为 1 秒）。如果 RTC_CRH 寄存器中 SECIE=1，则产生中断。此位只能由软件清除。对此位写'1'是无效的。 • 0：秒标志条件不成立 • 1：秒标志条件成立

RTC 的功能由这个控制寄存器控制。当前一个写操作还未完成时（RTOFF=0 时，详见："19.2.4 配置 RTC 寄存器"），不能写 RTC_CR 寄存器。

注意：

任何标志位都将保持挂起状态，直到适当的 RTC_CR 请求位被软件复位，表示所请求的中断已经被接受。

在复位时禁止所有中断，无挂起的中断请求，可以对 RTC 寄存器进行写操作。

当 APB1 时钟不运行时，OWF、ALRF、SECF 和 RSF 位不被更新。

OWF、ALRF、SECF 和 RSF 位只能由硬件置位，由软件来清零。

若 ALRF=1 且 ALRIE=1，则允许产生 RTC 全局中断。如果在 EXTI 控制器中允许产生 EXTI 线 17 中断，则允许产生 RTC 全局中断和 RTC 闹钟中断。

若 ALRF=1，如果在 EXTI 控制器中设置了 EXTI 线 17 的中断模式，则允许产生 RTC 闹钟中断；如果在 EXTI 控制器中设置了 EXTI 线 17 的事件模式，则这条线上会产生一个脉冲（不会产生 RTC 闹钟中断）。

19.3.3 RTC 预分频装载寄存器高位（RTC_PRLH）

偏移地址：0x08

复位值：0x0000

只写（参见“19.2.4 配置 RTC 寄存器”）

预分频装载寄存器用来保存 RTC 预分频器的周期计数值。它们受 RTC_CR 寄存器的 RTOFF 位写保护，仅当 RTOFF 值为’1’时允许进行写操作。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res												PRL[19:16]			
												w			

位 15:4	Res：保留 必须保持复位值。
位 3:0	PRL[19:16]：RTC 预分频装载值高位（RTC prescaler reload value high） 根据以下公式，该位域结合 RTC_PRLL 用来定义计数器的时钟频率： $f_{TR\_CLK}=f_{RTCCLK}/$（PRL[19:0]+1） *注意：不推荐使用 0 值，否则无法正确的产生 RTC 中断和标志位。*

19.3.4 RTC 预分频装载寄存器低位（RTC_PRLL）

偏移地址：0x0C

复位值：0x8000

只写（参见:“19.2.4 配置 RTC 寄存器”）

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRL[15:0]															
w															

位 15:0	PRL[15:0]：RTC 预分频装载值低位（RTC prescaler reload value low） 根据以下公式，该位域结合 RTC_PRLH.PRL[19:16]用来定义计数器的时钟频率：$f_{TR\_CLK}= f_{RTCCLK}/$（PRL[19:0]+1）

注意：如果输入时钟频率是 32.768kHz（f_{RTCCLK}），这个寄存器中写入 7FFFh 可获得周期为 1 秒钟的信号。

19.3.5 RTC 预分频器余数寄存器高位（RTC_DIVH）

偏移地址：0x10

复位值：0x0000

在 TR_CLK 的每个周期里，RTC 预分频器中计数器的值都会被重新设置为 RTC_PRL 寄存器的值。用户可通过读取 RTC_DIV 寄存器，获得预分频计数器的当前值，而不停止分频计数器的工作，从而获得精确的时间测量。此寄存器是只读寄存器，其值在 RTC_PRL 或 RTC_CNT 寄存器中的值发生改变后，由硬件重新装载。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res												RTC_DIV[19:16]			
												r			

位 15:4	Res：保留 必须保持复位值。
位 3:0	RTC_DIV[19:16]：RTC 时钟分频器余数高位（RTC clock divider high）

19.3.6 RTC 预分频器余数寄存器低位（RTC_DIVL）

偏移地址：0x14

复位值：0x8000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_DIV[15:0]															
r															

位 15:0	RTC_DIV[15:0]：RTC 时钟器余数低位（RTC clock divider low）

19.3.7 RTC 计数器寄存器高位（RTC_CNTH）

偏移地址：0x18

复位值：0x0000

RTC 核有一个 32 位可编程的计数器，可通过两个 16 位的寄存器访问。计数器以预分频器产生的 TR_CLK 时间基准为参考进行计数。RTC_CNT 寄存器用来存放计数器的计数值。它们受 RTC_CR 的位 RTOFF 写保护，仅当 RTOFF 值为'1'时，允许写操作。在高或低寄存器（RTC_CNTH 或 RTC_CNTL）上的写操作，能够直接装载到相应的可编程计数器，并且重新装载 RTC 预分频器。当进行读操作时，直接返回计数器内的计数值（系统时间）。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_CNT[31:16]															
rw															

位 15:0	RTC_CNT[31:16]：RTC 计数器高位（RTC counter high） 可通过读 RTC_CNTH 寄存器来获得 RTC 计数器当前值的高位部分。要对此寄存器进行写操作前，必须先进入配置模式（参见"19.2.4 配置 RTC 寄存器"）。

19.3.8 RTC 计数器寄存器低位（RTC_CNTL）

偏移地址：0x1C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_CNT[15:0]															
rw															

位 15:0	RTC_CNT[15:0]：RTC 计数器低位（RTC counter low） 可通过读 RTC_CNTL 寄存器来获得 RTC 计数器当前值的低位部分。要对此寄存器进行写操作，必须先进入配置模式（参见"19.2.4 配置 RTC 寄存器"）。

19.3.9 RTC 闹钟寄存器高位（RTC_ALRH）

偏移地址：0x20

复位值：0xFFFF

当可编程计数器的值与 RTC_ALR 中的 32 位值相等时，即触发一个闹钟事件，并且产生 RTC 闹钟中断。此寄存器受 RTC_CR 寄存器里的 RTOFF 位写保护，仅当 RTOFF 值为'1'时，允许写操作。

该寄存器仅支持写（参见"19.2.4 配置 RTC 寄存器"）。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_ALR[31:16]															
w															

位 15:0	RTC_ALR[31:16]：RTC 闹钟值高位（RTC alarm high） 此寄存器用来保存由软件写入的闹钟时间的高位部分。要对此寄存器进行写操作，必须先进入配置模式（参见"19.2.4 配置 RTC 寄存器"）。

19.3.10 RTC 闹钟寄存器低位（RTC_ALRL）

偏移地址：0x24

复位值：0xFFFF

该寄存器仅支持写（参见"19.2.4 配置 RTC 寄存器"）。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_ALR[15:0]															
w															

位 15:0	RTC_ALR[15:0]：RTC 闹钟值低位（RTC alarm low） 此寄存器用来保存由软件写入的闹钟时间的低位部分。要对此寄存器进行写操作，必须先进入 配置模式（参见"19.2.4 配置 RTC 寄存器"）。

19.3.11 RTC 唤醒定时器寄存器（RTC_WUT）

偏移地址：0x30

复位值：0x0000 FFFF

系统复位：不受影响

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WUT[15:0]															
rw															

位 31:16	Res：保留 必须保持复位值。
位 15:0	WUT[15:0]：唤醒自动重载值（Wakeup auto-reload value bits） 当使能唤醒定时器时（BKP_WUTCR.WUTE 置 1），每（WUT[15:0]+1）个 ck_wut 周期将 WUTF 标志置 1 一次。若 WUTIE 置位，则产生中断。 ck_wut 周期通过 RTC_CR 寄存器的 WUCKSEL[2:0]位进行选择。 当 WUCKSEL[2]=1 时，唤醒定时器变为 17 位，WUCKSEL[1]等效为 WUT[16]，即要重载到定时器的最高有效位。

	WUTF 第一次置 1 发生在 WUTE 置 1 之后（WUT+1）个 ck_wut 周期。禁止在 WUCKSEL[2:0]=011（RTCCLK/2）时将 WUT[15:0]设置为 0x0000。

20 独立看门狗（IWDG）

器件内置一个独立看门狗，可用来检测和解决由软件错误引起的故障。当计数器达到给定的超时值时，产生系统复位。

IWDG 由专用的低速时钟（LSI）驱动，即使主时钟发生故障它也仍然有效。

IWDG 最适合应用于那些需要看门狗作为一个在主程序之外，能够完全独立工作，并且对时间精度要求较低的场合。

20.1 IWDG 主要性能

- 自由运行的递减计数器
- 时钟由独立的 RC 振荡器提供（可在停机和待机模式下工作）
- 看门狗被激活后，则在计数器计数至 0x000 时产生复位
- IWDG 计数器复位初始值可由 FLASH 选项字设置（请参见“3.2.3 选项字节”），通过配置 Flash 选项字，可以保证当芯片复位后如果程序跑飞，IWDG 复位的时间间隔不会太长。
- 可以通过 Flash 选项字 LSI_LP_CTL 控制芯片进入停机（Stop）或者待机（Standby）模式后 LSI 的状态。通过配置 Flash 选项字，可以选择使芯片进入停机（Stop）或者待机（Standby）模式后，如果 LSION 设置为 0，则关掉 LSI；在芯片唤醒后，LSI 恢复成进模式之前的状态。如果不配置 Flash 选项字，则在使能 IWDG 后再进入 Stop 或者待机（Standby）模式，系统一定会被 IWDG 周期唤醒。

20.2 IWDG 功能描述

图 20-1 为独立看门狗模块的功能框图。

在键寄存器（IWDG_KR）中写入 0xCCCC，开始启用 IWDG；此时计数器开始从其复位值 0xFFF 递减计数。当计数器计数到末尾 0x000 时，会产生一个复位信号（IWDG_RESET）。

无论何时，只要在键寄存器 IWDG_KR 中写入 0xAAAA，IWDG_RLR 中的值就会被重新加载到计数器，从而避免产生看门狗复位。

支持看门狗窗口（window）模式，详细的描述请参考章节：“20.2.1 窗口选项”。

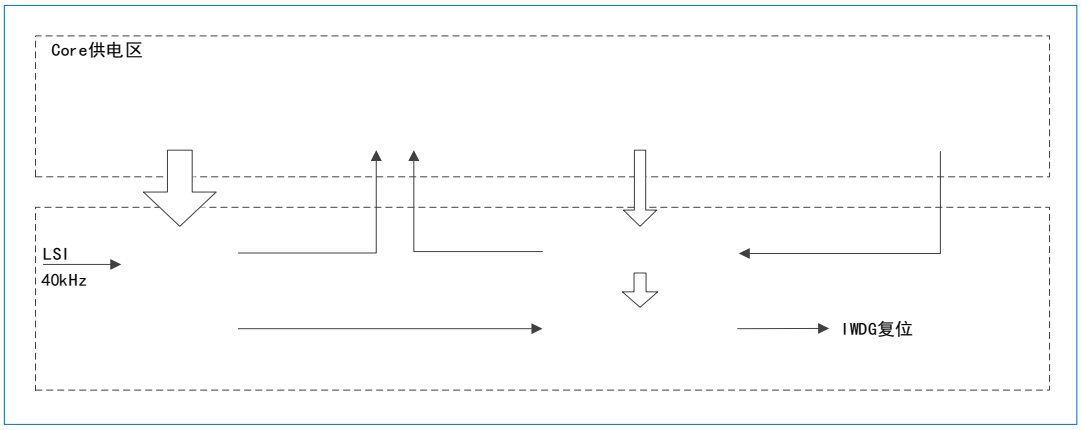


图 20-1 IWDG 框图

注意：看门狗功能处于 V_{DD} 供电区，即在停机和待机模式时仍能正常工作。

计算超时的公式为：

T_{IWDG}=（（1/f_{LSI}）/PR[2:0]）*（RL[11:0]+1）

其中：

- T_{IWDG} 为 IWDG 超时时间；
- f_{LSI}：LSI 的频率。

表 20-1 IWDG 超时时间表（40kHz 的输入时钟（LSI））

预分频系数	PR[2:0]位	最短时间（ms） RL[11:0]=0x000	最长时间（ms） RL[11:0]=0xFFF
/4	0	0.1	409.6
/8	1	0.2	819.2
/16	2	0.4	1638.4
/32	3	0.8	3276.8
/64	4	1.6	6553.6
/128	5	3.2	13107.2
/256	6 或 7	6.4	26214.4

注意:

这些时间是按照 40 kHz 时钟给出。实际上，MCU 内部的 RC 频率会在 30 kHz 到 60 kHz 之间变化。此外，即使 RC 振荡器的频率是精确的，确切的时序仍然依赖于 APB 接口时钟与 RC 振荡器时钟之间的相位差，因此总会有一个完整的 RC 周期是不确定的。

通过对 LSI 进行校准可获得相对精确的看门狗超时时间。

20.2.1 窗口选项

通过在 IWDG_WINR 寄存器中设置合适的窗口，IWDG 也可以用作窗口看门狗。当计数器值大于窗口寄存器（IWDG_WINR）中存储的值时，如果执行重载操作，则会产生复位。IWDG_WINR 的默认值为 0x0000 0FFF，因此如果不更新此默认值，窗口选项将一直处于禁用状态。窗口值一经更改，便会执行重载操作，以便将递减计数器的值复位为 IWDG_RLR 值，方便计算周期数以生成下一次重载。

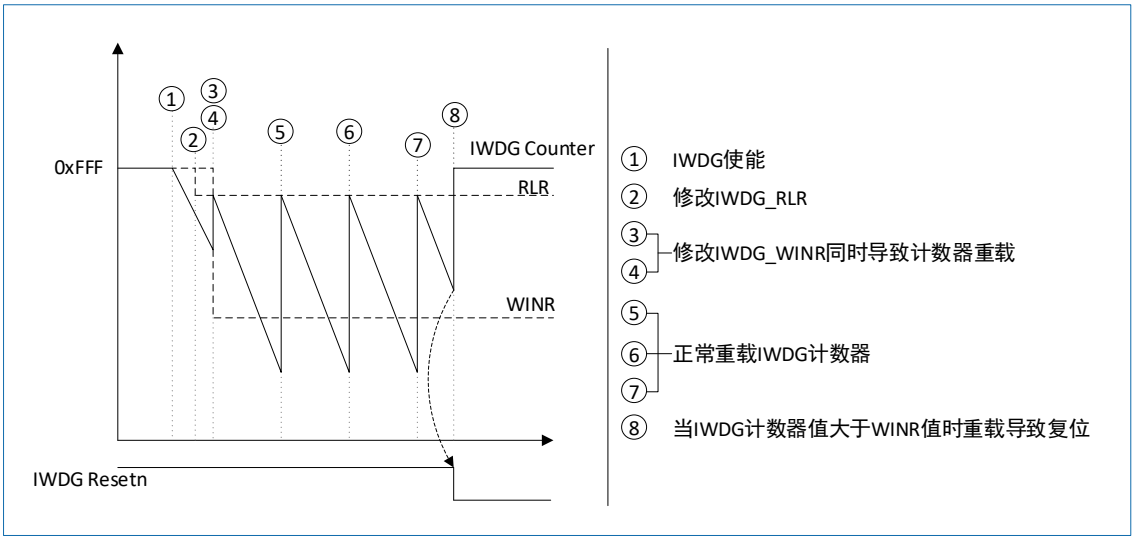


图 20-2 IWDG 使能窗口选项时的工作说明

使能窗口选项时配置 IWDG 的流程：

1. 向 IWDG_KR 写入 0x0000 CCCC 来使能 IWDG。

2. 向 IWDG_KR 写入 0x0000 5555 来使能寄存器访问。
3. 修改 IWDG_PR 的值为 0~7 中的值，配置 IWDG 的预分频器。
4. 写重载寄存器 IWDG_RLR。
5. 等待 IWDG_RLR 寄存器值更新完成（IWDG_SR = 0x0000 0000）。
6. 写窗口寄存器 IWDG_WINR，这个操作会导致 IWDG 计数器自动更新为 IWDG_RLR 中的值。

注意：当 IWDG_SR 为 0x0000 0000 时, 才能写 IWDT_WINR, 否则 IWDG 计数器可能不会正确的重载为 IWDG_RLR 中的值。

禁能窗口选项时配置 IWDG 的流程：

1. 向 IWDG_KR 写入 0x0000 CCCC 来使能 IWDG。
2. 向 IWDG_KR 写入 0x0000 5555 来使能寄存器访问。
3. 修改 IWDG_PR 的值为 0~7 中的值， 配置 IWDG 的预分频器。
4. 写重载寄存器 IWDG_RLR。
5. 等待 IWDG_RLR 寄存器值更新完成（IWDG_SR = 0x0000 0000）。
6. 刷新计数器值为 IWDG_RLR 值（IWDG_KR = 0x0000 AAAA）。

20.2.2 硬件看门狗

如果用户在选择字节中启用了硬件看门狗功能，在系统上电复位后，看门狗会自动开始运行；如果在计数器计数结束前，若软件没有向键寄存器写入相应的值，则系统会产生复位。

20.2.3 寄存器访问保护

IWDG_PR 和 IWDG_RLR 寄存器具有写保护功能。要修改这两个寄存器的值，必须先向IWDG_KR 寄存器写入 0x5555。以不同的值写入这个寄存器将会打乱操作顺序，寄存器将重新被保护。重装载操作（即写入 0xAAAA）也会启动写保护功能。

状态寄存器指示预分频值和递减计数器是否正在被更新。

20.2.4 调试模式

当微控制器进入调试模式时（Cortex-M3 核心停止），根据调试模块中的 DBG_IWDG_STOP 配置位的状态，IWDG 的计数器能够继续工作或停止。详细信息，请参见：“36.15.2 支持定时器、看门狗、CAN 和 I2C 的调试”。

20.3 IWDG 寄存器

基地址：0x4000 3000

空间大小：0x400

可以用半字（16 位）或字（32 位）的方式操作这些外设寄存器。

20.3.1 键寄存器（IWDG_KR）

偏移地址：0x00

复位值：0x0000 0000

该寄存器在待机模式复位。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w															

位 31:16	Res：保留 必须保持复位值。
位 15:0	KEY[15:0]：键值（只写寄存器，读出值为 0x0000）（Key value, write only, read 0x0000)） 软件必须以一定的间隔写入 0xAAAA，否则，当计数器为 0 时，看门狗会产生复位。 • 写入 0x5555 表示允许访问 IWDG_PR 和 IWDG_RLR 寄存器。（见章节：“20.2.3 寄存器访问保护”。） • 写入 0xCCCC，启动看门狗工作（若选择了硬件看门狗则不受此命令字限制）。 本寄存器为只写寄存器，读出值为 0x0000。

20.3.2 预分频寄存器（IWDG_PR）

偏移地址：0x04

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res													PR[2:0]		
													rw		

位 31:3	Res：保留 必须保持复位值。
位 2:0	PR[2:0]：预分频因子（Prescaler divider） 该位域具有写保护设置，参见“20.2.3 寄存器访问保护”。通过设置该位域来选择计数器时钟的预分频因子。要改变预分频因子，IWDG_SR 寄存器的 PVU 位必须为 0。 • 000：预分频因子=4 • 001：预分频因子=8 • 010：预分频因子=16 • 011：预分频因子=32 • 100：预分频因子=64 • 101：预分频因子=128 • 110：预分频因子=256 • 111：预分频因子=256 *注意：对此寄存器进行读操作，将从 V_{DD} 电压域返回预分频值。如果写操作正在进行，则读回的值可能是无效或是过期的。因此，只有当 IWDG_SR 寄存器的 PVU 位为 0 时，读出的值才有效。*

20.3.3 重装载寄存器（IWDG_RLR）

偏移地址：0x08

复位值：0x0000 0FFF

该寄存器在待机模式复位。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res				RL[11:0]											
				rw											

位 31:12	Res：保留 必须保持复位值。
位 11:0	RL[11:0]：看门狗计数器重装载值 （Watchdog counter reload value） 该位域具有写保护功能，参看“20.2.3 寄存器访问保护”。用于定义看门狗计数器的重装载值，每当向 IWDG_KR 寄存器写入 0xAAAA 时，重装载值会被传送到计数器中。随后计数器从这个值开始递减计数。看门狗超时周期可通过此重装载值和时钟预分频值来计算，参见表 20-1。只有当 IWDG_SR 寄存器中的 RVU 位为 0 时，才能对此寄存器进行修改。 *注意：对此寄存器进行读操作，将从 VDD 电压域返回预分频值。如果写操作正在进行，则读回的值可能是无效或是过期的。因此，只有当 IWDG_SR 寄存器的 RVU 位为 0 时，读出的值才有效。*

20.3.4 状态寄存器（IWDG_SR）

偏移地址：0x0C

复位值：0x0000 0000

该寄存器在待机模式时不复位。

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

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

位 31:2	Res：保留 必须保持复位值。
位 1	RVU：看门狗计数器重装载值更新 （Watchdog counter reload value update） 此位由硬件置’1’用来指示重装载值的更新正在进行中。当在 V_{DD} 域中的重装载更新结束后，此位由硬件清零（最多需 5 个 40kHz 的 RC 周期）。 重装载值只有在 RVU 位被清零后才可更新。
位 0	PVU：看门狗预分频值更新（Watchdog prescaler value update） 此位由硬件置’1’用来指示预分频值的更新正在进行中。当在 V_{DD} 域中的预分频值更新结束后，此位由硬件清零（最多需 5 个 40kHz 的 RC 周期）。 预分频值只有在 PVU 位被清零后才可更新。

注意：如果在应用程序中使用了多个重装载值或预分频值，则必须在 RVU 位被清零后才能重新改变预装载值，在 PVU 位被清零后才能重新改变预分频值。然而，在更新预分频和/或重装载值后，不必等待 RVU 或 PVU 位清零，可继续执行下面的代码。（即使是在低功耗模式下，此写操作仍会被继续执行完成。）

20.3.5 窗口寄存器（IWDG_WINR）

偏移地址：0x10

复位值：0x0000 0FFF

该寄存器在待机模式复位。

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

Res

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res				WIN[11:0]											
				rw											

位 31:12	Res：保留 必须保持复位值。
位 11:0	WIN[11:0]：看门狗计数器窗口值（Watchdog counter window value）

21 窗口看门狗（WWDG）

器件内置一个窗口看门狗。窗口看门狗通常被用来监测由外部干扰或不可预见的逻辑条件所造成的应用程序背离正常的运行顺序而产生的软件故障。

WWDG 由从 APB1 时钟分频后得到的时钟驱动，通过可配置的时间窗口来检测应用程序非正常的过迟或过早的操作。当计数器达到给定的超时值时，触发一个中断或产生系统复位。除非递减计数器的值在 T6 位变成 0 前被刷新，否则看门狗电路在达到预置的时间周期时，会产生一个 MCU 复位。在递减计数器达到窗口寄存器数值之前，如果 7 位的递减计数器的数值（在控制寄存器中）被刷新，那么也将产生一个 MCU 复位。这表明递减计数器必须在一个有限的时间窗口中被刷新。

WWDG 最适合那些要求看门狗在精确计时窗口起作用的应用程序。

21.1 WWDG 主要特性

- 可编程的自由运行递减计数器。
- 复位条件：
  - 当递减计数器的值小于 0x40，（若看门狗被启动）则产生复位。
  - 当递减计数器在窗口外被重新装载，（若看门狗被启动）则产生复位。具体信息请参见：图 21-2。
- 提前唤醒中断（EWI）：如果启动了看门狗并且允许中断，当递减计数器等于 0x40 时产生提前唤醒中断，它可以被用于重装载计数器以避免 WWDG 复位。

21.2 WWDG 功能描述

如果看门狗被启动（WWDG_CR 寄存器中的 WDGA 位被置'1'），且当 7 位（T[6:0]）递减计数器从 0x40 翻转到 0x3F（T6 位清零）时，则产生一个复位。如果软件在计数器值大于配置寄存器中的数值时，重新装载计数器，将产生一个复位。

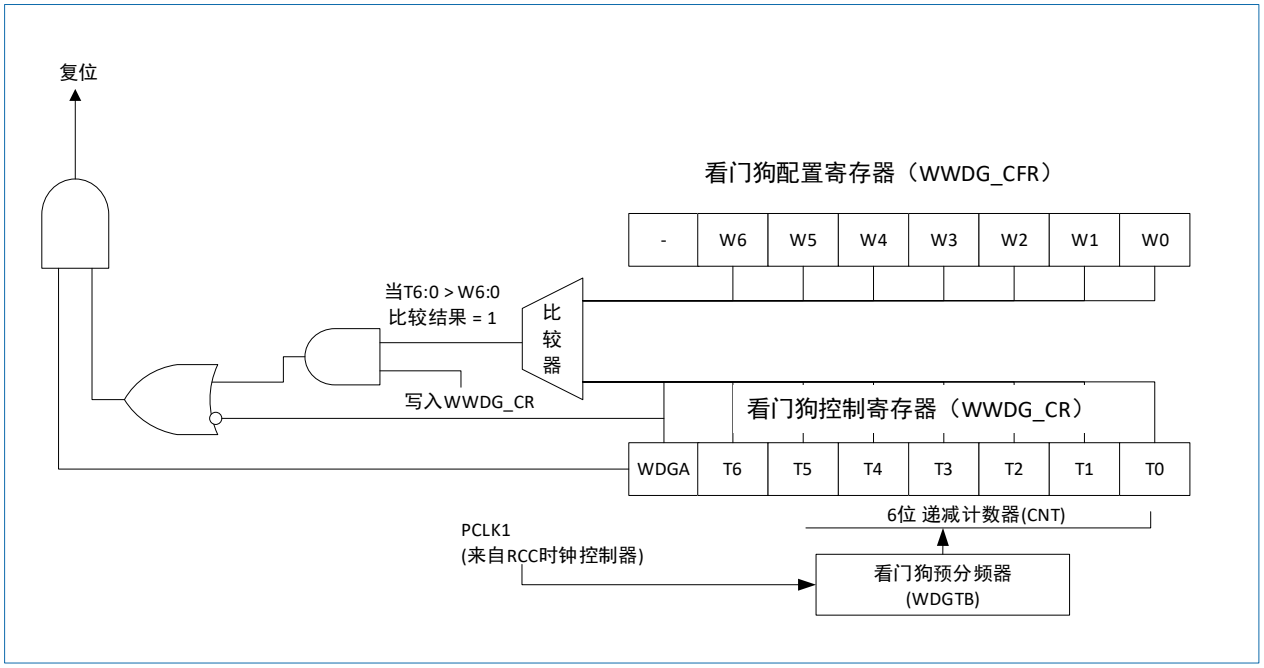


图 21-1WWDG 框图

应用程序在正常运行过程中必须定期地写入 WWDG_CR 寄存器以防止 MCU 发生复位。只有当计数器值小于配置寄存器的值时，才能进行该操作。

储存在 WWDG_CR 寄存器中的数值必须在0xFF 和 0xC0 之间：

- 启动看门狗：

  在系统复位后，看门狗总是处于关闭状态，设置 WWDG_CR 寄存器的 WDGA 位能够开启看门狗，随后它不能再被关闭，除非发生复位。

- 控制递减计数器：

  递减计数器处于自由运行状态，即使看门狗被禁止，递减计数器仍继续递减计数。当看门狗被启用时，T6 位必须被设置，以防止立即产生一个复位。

  T[5:0]位包含了看门狗产生复位之前的计时数目；复位前的延时时间在一个最小值和一个最大值之间变化，这是因为写入 WWDG_CR 寄存器时，预分频值是未知的。配置寄存器（WWDG_CFR）中包含窗口的上限值：要避免产生复位，递减计数器必须在其值小于窗口寄存器的数值并且大于 0x3F 时被重新装载，图 21-2 描述了窗口寄存器的工作过程。另一个重装载计数器的方法是利用提前唤醒中断（EWI）。设置 WWDG_CFR 寄存器中的 EWI 位开启该中断。当递减计数器到达 0x40 时，则产生此中断，相应的中断服务程序（ISR）可以用来加载计数器以防止 WWDG 复位。在 WWDG_SR 寄存器中写'0'可以清除该中断。

注意：可以用 T6 位产生一个软件复位（设置 WDGA 位为'1'，T6 位为'0'）。

21.3 如何编写看门狗超时程序

可以使用图 21-2 提供的公式计算窗口看门狗的超时时间。

警告：当写入 WWDG_CR 寄存器时，确保 T6 位始终为'1'以避免立即产生一个复位。

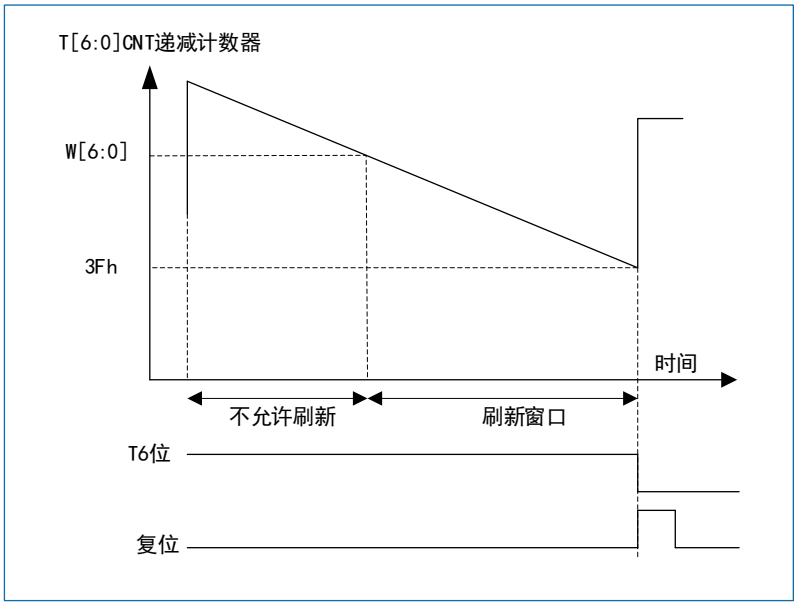


图 21-2 WWDG 时序图

计算超时的公式如下：

$T_{WWDG}=T_{PCLK1} * 4096 * 2^{WDGTB}*$（T[5:0] +1）

- T_{WWDG}：WWDG 超时时间
- T_{PCLK1}：APB1 以 ms 为单位的时钟间隔，在 PCLK1=36MHz 时的最小/最大超时值（具体值如下表）。

表 21-1 WDGTB 超时值

WDGTB	最小超时值 （T[5:0]=00 0000）	最大超时值 （T[5:0]=11 1111）
0	113μs	7.28ms
1	227μs	14.56ms
2	455μs	29.12ms
3	910μs	58.25ms

21.4 调试模式

当微控制器进入调试模式时（Cortex® -M3 核心停止），根据调试模块中的 DBG_WWDG_STOP 配置位的状态，WWDG 的计数器能够继续工作或停止。详见章节：“36.15.2 支持定时器、看门狗、CAN 和 I2C 的调试”。

21.5 WWDG 寄存器

基地址：0x4000 2C00

空间大小：0x400

可以用半字（16 位）或字（32 位）的方式操作这些外设寄存器。

21.5.1 控制寄存器（WWDG_CR）

偏移地址：0x00

复位值：0x0000 007F

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								WDGA	T[6:0]						
								rw	rw						

位 31:8	Res：保留 必须保持复位值。
位 7	WDGA：激活位（Activation bit） 此位由软件置’1’，但仅能由硬件在复位后清零。当 WDGA=1 时，看门狗可以产生复位。 • 0：禁止看门狗 • 1：启用看门狗
位 6:0	T[6:0]：7 位计数器（MSB 至 LSB）（7-bit counter, MSB to LSB） 该位域用来存储看门狗的计数器值。每（$4096x2^{WDGTB}$）个 PCLK1 周期减 1。当计数器值从 0x40 翻转为 0x3F 时（T6 变成 0），产生看门狗复位。

21.5.2 配置寄存器（WWDG_CFR）

偏移地址：0x04

复位值：0x7F

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res						EWI	WDGTB[1:0]		W[6:0]						
						rw	rw		rw						

位 31:10	Res：保留 必须保持复位值。
位 9	EWI：提前唤醒中断（Early wakeup interrupt） 此位若置’1’，则当计数器值达到 0x40，即产生中断。此中断只能由硬件在复位后清除。
位 8:7	WDGTB[1:0]：时基（Timer base） 预分频器的时基可以设置如下： • 00：CK 计时器时钟（PCLK1 除以 4096）除以 1 • 01：CK 计时器时钟（PCLK1 除以 4096）除以 2 • 10：CK 计时器时钟（PCLK1 除以 4096）除以 4 • 11：CK 计时器时钟（PCLK1 除以 4096）除以 8
位 6:0	W[6:0]：7 位窗口值（7-bit window value） 该位域包含了用来与递减计数器进行比较用的窗口值。

21.5.3 状态寄存器（WWDG_SR）

偏移地址：0x08

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res															EWIF
															rc_w0

位 31:1	Res：保留 必须保持复位值。
位 0	EWIF：提前唤醒中断标志（Early wakeup interrupt flag） 当计数器值达到 0x40 时，此位由硬件置’1’。它必须通过软件写’0’来清除。对此位写’1’无效。若中断未被使能，此位也会被置’1’。

22 USB 全速设备接口（USB）

USB 外设实现了 USB2.0 全速总线和 APB1 总线间的接口。USB 外设支持 USB 挂起/唤醒操作，可以停止设备时钟实现低功耗。

22.1 USB 主要特征

- 符合 USB2.0 全速设备的技术规范
- 可配置 1 到 8 个 USB 端点
- 循环冗余校验（CRC）生成/校验，反向不归零（NRZI）编码/解码和位填充
- 支持同步传输
- 支持批量/同步端点的双缓冲区机制
- 支持 USB 挂起/唤醒操作
- 帧锁定时钟脉冲生成

注意：USB 和 CAN1 共用一个专用的 512 字节的 SRAM 存储器用于数据的发送和接收，因此不能同时使用 USB 和 CAN1（共享的 SRAM 被 USB 和 CAN1 模块互斥地访问）。USB 和 CAN1 可以同时用于一个应用中但不能在同一个时间使用。

下图是 USB 外设的方框图。

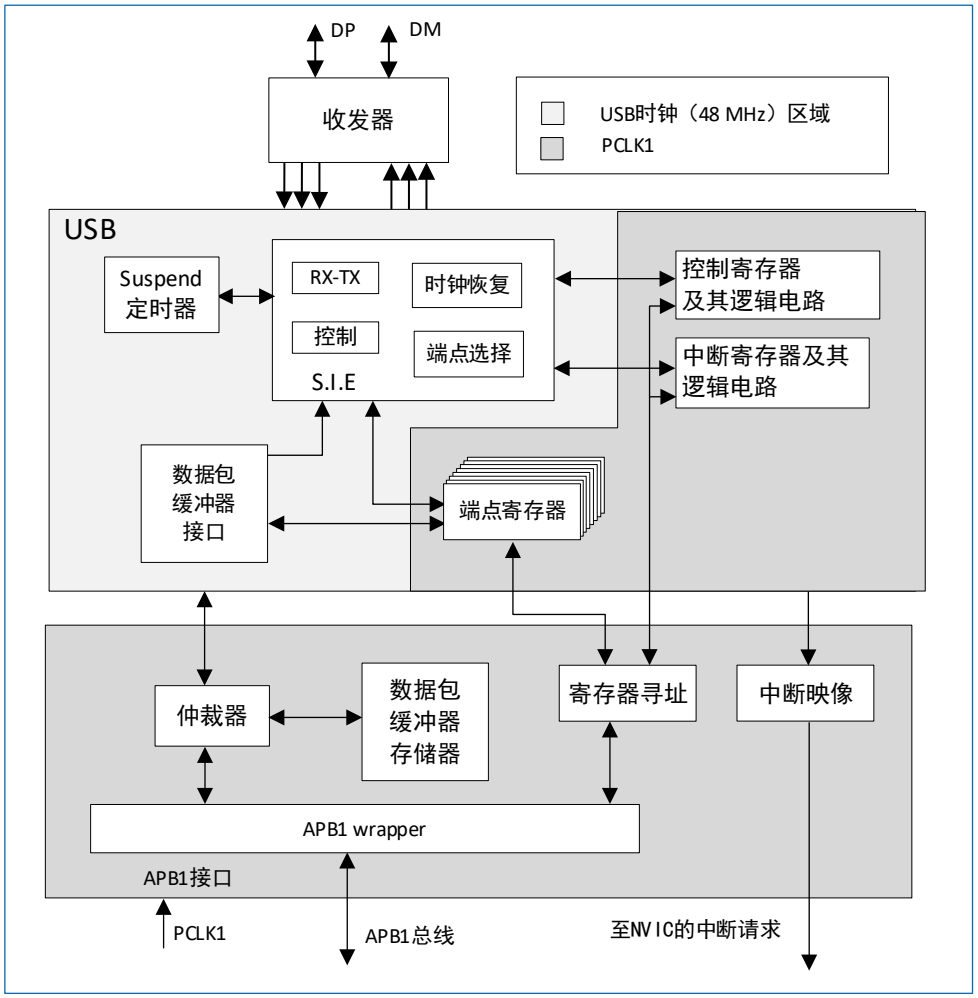


图 22-1 USB 设备框图

22.2 USB 功能描述

USB 模块为 PC 主机和微控制器所实现的功能之间提供了符合 USB 规范的通信连接。PC 主机和微控制器之间的数据传输是通过共享同一专用的数据缓冲区来完成的，该数据缓冲区能被 USB 外设直接访问。这块专用数据缓冲区的大小由所使用的端点数目和每个端点最大的数据分组大小所决定，每个端点最大可使用 512 字节缓冲区，最多可用于 16 个单向或 8 个双向端点。USB 模块同 PC 主机通信，根据 USB 规范实现令牌分组的检测，数据发送/接收的处理，和握手分组的处理。整个传输格式化由硬件完成，其中包括 CRC 的生成和校验。

每个端点都有一个缓冲区描述块，描述该端点使用的缓冲区地址、大小和需要传输的字节数。

当 USB 模块识别出一个有效的功能/端点的令牌分组时，（如果需要传输数据并且端点已配置）随之发生相关的数据传输。USB 模块通过一个内部的 16 位寄存器实现端口与专用缓冲区的数据交换。在所有的数据传输完成后，如果需要，则根据传输的方向，发送或接收适当的握手分组。

在数据传输结束时，USB 模块将触发与端点相关的中断，通过读状态寄存器和/或者利用不同的中断处理程序，微控制器可以确定：

- 哪个端点需要得到服务。
- 当发生位填充、格式、CRC、协议、缺失 ACK、缓冲区溢出/缓冲区未满等错误时，正在进行的是哪种类型的传输。

USB 模块对同步传输和高吞吐量的批量传输提供了特殊的双缓冲区机制，在微控制器使用一个缓冲区的时候，该机制保证了 USB 外设总是可以使用另一个缓冲区。

在任何不需要使用 USB 模块的时候，通过写控制寄存器可以使 USB 模块置于低功耗模式（SUSPEND 模式）。在这种模式下，不产生任何静态电流消耗，同时 USB 时钟也会减慢或停止。通过对 USB 线上数据传输的检测，可以在低功耗模式下唤醒 USB 模块。也可以将一特定的中断输入源直接连接到唤醒引脚上，以使系统能立即恢复正常的时钟系统，并支持直接启动或停止时钟系统。

22.2.1 USB 功能模块描述

USB 模块实现了标准 USB 接口的所有特性，它由以下部分组成：

- 串行接口控制器（SIE）：

  该模块包括的功能有：帧头同步域的识别，位填充，CRC 的产生和校验，PID 的验证/产生，和握手分组处理等。它与 USB 收发器交互，利用分组缓冲接口提供的虚拟缓冲区存储局部数据。它也根据 USB 事件，和类似于传输结束或一个包正确接收等与端点相关事件生成信号，例如帧起始位（SOF）、USB 复位、数据错误等等，这些信号用来产生中断。

- 定时器：

  本模块的功能是产生一个与帧起始位报文同步的时钟脉冲，将 3ms 内没有数据传输的状态，检测为（主机的）全局挂起条件。

- 分组缓冲器接口：

  此模块管理那些用于发送和接收的临时本地内存单元。它根据 SIE 的要求分配合适的缓冲区，并将其定位到端点寄存器所指向的存储区地址。它在每个字节传输后，自动递增地址，直到数据分组传输结束。它记录传输的字节数并防止缓冲区溢出。

- 端点相关寄存器：

  每个端点都有一个与之相关的寄存器，用于描述端点类型和当前状态。对于单向和单缓冲器端点，一个寄存器就可以用于实现两个不同的端点。一共 8 个寄存器，可以用于实现最多 16 个单向/单缓冲的端点或者 7 个双缓冲的端点或者这些端点的组合。例如，可以同时实现 4 个双缓冲端点和 8 个单缓冲/单向端点。

- 控制寄存器：

这些寄存器包含整个 USB 模块的状态信息，用来触发诸如恢复、低功耗等 USB 事件。

- 中断寄存器：

  这些寄存器包含中断屏蔽信息和中断事件的记录信息。配置和访问这些寄存器可以获取中断源，中断状态等信息，并能清除待处理中断的状态标志。

注意：端点 0 总是作为单缓冲模式下的控制端点。

USB 模块通过 APB1 接口部件与 APB1 总线相连，APB1 接口部件包括以下部分：

- 分组缓冲区：

  数据分组缓存在分组缓冲区中，它由分组缓冲接口控制并创建数据结构。应用软件可以直接访问该缓冲区。它的大小为 512 字节，由 256 个 16 位的半字构成。

- 仲裁器：

  该部件负责处理来自 APB1 总线和 USB 接口的存储器请求。它通过向 APB1 提供较高的访问优先权来解决总线的冲突，并且总是保留一半的存储器带宽供 USB 完成传输。它采用时分复用的策略实现了虚拟的双端口 SRAM，即在 USB 传输的同时，允许应用程序访问存储器。此策略也允许任意长度的多字节 APB1 传输。

- 寄存器映射单元：

  此部件将 USB 模块的各种字节宽度和位宽度的寄存器映射成能被 APB1 寻址的 16 位宽度的内存集合。

- APB1 封装：

  此部件为缓冲区和寄存器提供了到 APB1 的接口，并将整个 USB 模块映射到 APB1 地址空间。

- 中断映射单元：

  将可能产生中断的 USB 事件映射到不同的 NVIC 请求线上：

  - USB 低优先级中断：可由所有 USB 事件触发（正确传输，USB 复位等）。固件在处理中断前应当首先确定中断源。
  - USB 高优先级中断：仅能由同步和双缓冲批量传输的正确传输事件触发，目的是保证最大的传输速率。
  - USB 唤醒中断：由 USB 挂起模式的唤醒事件触发。

22.3 编程中需要考虑的问题

在下面的章节中，将介绍 USB 模块和应用程序之间的交互过程，有利于简化应用程序的开发。

22.3.1 通用 USB 设备编程

这一部分描述了实现 USB 设备功能的应用程序需要完成的任务。除了介绍一般的 USB 事件中应该采取的操作外，还着重介绍了双缓冲端点和同步传输的操作。这些相关的操作都是由 USB 模块初始化，并由以下几节所描述的 USB 事件所驱动。

22.3.2 系统复位和上电复位

发生系统复位或者上电复位时，应用程序首先需要做的是提供 USB 模块所需要的时钟信号，然后清除复位信号，使程序可以访问 USB 模块的寄存器。复位之后的初始化流程如下所述：

首先，由应用程序激活寄存器单元的时钟，再配置设备时钟管理逻辑单元的相关控制位，清除复位信号。

然后，应用程序需要通过配置设备时钟管理逻辑的相应控制位来为 USB 模块提供标准所定义的 48 MHz 时钟。

当系统复位时，应用程序应该初始化所有需要的寄存器和分组缓冲区描述表，使 USB 模块能够产生

正常的中断和完成数据传输。所有与端点无关的寄存器需要根据应用的需求进行初始化（比如中断使能的选择，分组缓冲区地址的选择等）。接下来按照 USB 复位处理（参见下段）。

22.3.2.1 USB 复位（RESET 中断）

发生 USB 复位时，USB 模块进入前面章节中描述过的系统复位状态：所有端点的通信都被禁止（USB 模块不会响应任何分组）。在 USB 复位后，USB 模块被使能，同时地址为 0 的默认控制端点（端点 0）也需要被使能。这可以通过配置 USB_DADDR 寄存器的 EF 位，USB_EP0R 寄存器和相关的分组缓冲区来实现。在 USB 设备的枚举阶段，主机将分配给设备一个唯一的地址，这个地址必须写入 USB_DADDR 寄存器的 ADD[6:0]位中，同时配置其他所需的端点。

当复位中断产生时，应用程序必须在中断产生后的 10 ms 之内使能端点 0 的传输。

22.3.2.2 分组缓冲区的结构和用途

每个双向端点都可以接收或发送数据。接收到的数据存储在该端点指定的专用缓冲区内，而另一个缓冲区则用于存放待发送的数据。对这些缓冲区的访问由分组缓冲区接口模块实现，它提出缓冲区访问请求，并等待确认信息后返回。为防止产生微控制器与 USB 模块对缓冲区的访问冲突，缓冲区接口模块使用仲裁机制，使 APB1 总线的一半周期用于微控制器的访问，另一半保证 USB 模块的访问。这样，微控制器和 USB 模块对分组缓冲区的访问如同对一个双端口 SRAM 的访问，即使微控制器连续访问缓冲区，也不会产生访问冲突。

USB 模块使用固定的时钟，按照 USB 标准，此时钟频率被固定为 48MHz。APB1 总线的时钟可以大于或者小于这个频率。

注意：为满足 USB 数据传输率和分组缓冲区接口的系统需求，APB1 总线时钟的频率必须大于 8MHz，以避免数据缓冲区溢出或不满。

每个端点对应于两个分组缓冲区（一般一个用于发送，另一个用于接收）。这些缓冲区可以位于整个分组存储区的任意位置，因为它们的地址和长度都定义在缓冲区描述表中，而缓冲区描述表也同样位于分组缓冲区中，其地址由 USB_BTABLE 寄存器确定。

缓冲区描述表的每个表项都关联到一个端点寄存器，它由 4 个 16 位的半字组成，因此缓冲区描述表的起始地址按 8 字节对齐（寄存器的最低 3 位总是'000'）。章节“22.5.3 缓冲区描述表”详细介绍缓冲区描述表表项。如果是非同步非双缓冲的单向端点，只需要一个分组缓冲区（即发送方向上的分组缓冲区）。其他未用到的端点或某个未使用的方向上的缓冲区描述表项可以用于其他用途。同步和双缓冲批量端点有特殊的分组缓冲区处理方法（请分别参考章节：“22.3.3 双缓冲端点”和“22.3.4 同步传输”。下图描述了缓冲区描述表项和分组缓冲区区域的关系。

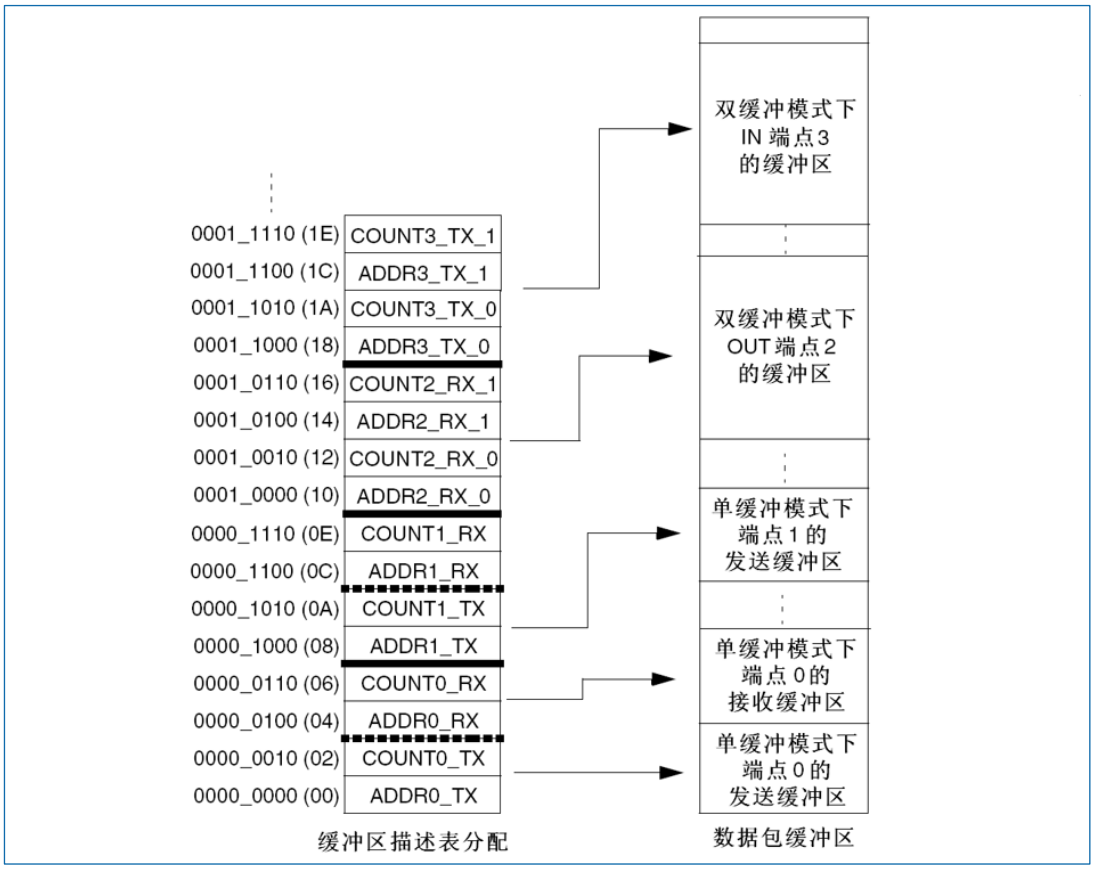


图 22-2 分组缓冲区对应的缓冲区描述表项定位

不管是接收还是发送，分组缓冲区都是从底部开始使用的。USB 模块不会改变超出当前分配到的缓冲区区域以外的其他缓冲区的内容。如果缓冲区收到一个比自己大的数据分组，它只会接收最大为自身大小的数据，其他的丢掉，即发生了所谓的缓冲区溢出异常。

22.3.2.3 端点初始化

初始化端点的第一步是把适当的值写到 USB_ADDRx_TX 或 USB_ADDRx_RX 寄存器中，以便 USB 模块能找到要传输的数据或准备好接收数据的缓冲区。USB_EPxR 寄存器的 EP_TYPE 位确定端点的基本类型，EP_KIND 位确定端点的特殊特性。作为发送方，需要设置 USB_EPxR 寄存器的 STAT_TX 位来使能端点，并配置 USB_COUNTx_TX 位寄存器决定发送长度。作为接收方，需要设置 USB_EPxR 寄存器的 STAT_RX 位来使能端点，并且设置 BL_SIZE 和 NUM_BLOCK 位，确定接收缓冲区的大小，以检测缓冲区溢出的异常。对于非同步非双缓冲批量传输的单向端点，只需要初始化与支持的传输方向相关的寄存器和位。一旦端点被使能，应用程序就不能再修改 USB_EPxR 寄存器的值和以及 USB_ADDRx_TX/USB_ADDRx_RX，COUNTx_TX/USB_COUNTx_RX 所在的位置，因为这些值会被硬件实时修改。当数据传输完成时，CTR 中断会产生，此时上述寄存器可以被访问，并重新使能新的传输。

22.3.2.4 IN 分组（用于数据发送）

当接收到一个 IN 令牌分组时，如果接收到的地址和一个配置好的端点地址相符合的话，USB 模块将会根据缓冲区描述表的表项，访问相应的 USB_ADDRx_TX 和 USB_COUNTx_TX 寄存器，并将这些寄存器中的数值存储到内部的 16 位寄存器 ADDR 和 COUNT（应用程序无法访问）中。此时，USB 模块开始根据 USB_EPxR 寄存器的 DTOG_TX 位发送 DATA0 或 DATA1 分组，并访问缓冲区（参见：“22.3.2.2 分组缓冲区的结构和用途”）。在 IN 分组传输完毕之后，从缓冲区读到的第一个字节将被装载到输出移位寄存器中，并开始发送。最后一个数据字节发送完成之后，计算好的 CRC 将被发送。如果收到的分组所对应的端点是无效的，将根据 USB_EPxR 寄存器上的 STAT_TX 位发送 NAK 或 STALL 握手分组而不发送数据。

内部的 ADDR 寄存器被用作当前缓冲区的指针，COUNT 寄存器用于记录剩下未传输的字节数。USB

总线使用低字节在先的方式传输从缓冲区中读出的每一个字。数据从 ADDRx_TX 指向的数据分组缓冲区开始读取，长度为 COUNTx_TX/2 个字。如果发送的数据分组为奇数个字节，则只使用最后一个字的低 8 位。

在接收到主机响应的 ACK 后，USB_EPxR 寄存器的值有以下更新：DTOG_TX 位被翻转，STAT_TX 位为'10'，使端点无效，CTR_TX 位被置位。应用程序需要通过 USB_ISTR 寄存器的 EP_ID 和 DIR 位识别产生中断的 USB 端点。CTR_TX 事件的中断服务程序需要首先清除中断标志位，然后准备好需要发送的数据缓冲区，更新 COUNTx_TX 为下次需要传输的字节数，最后再设置 STAT_TX 位为'11'（端点有效），再次使能数据传输。当 STAT_TX 位为'10'时（端点为 NAK 状态），任何发送到该端点的 IN 请求都会被 NAK，USB 主机会重发 IN 请求直到该端点确认请求有效。上述操作过程是必须遵守的，以避免丢失紧随上一次 CTR 中断请求的下一个 IN 传输请求。

22.3.2.5 OUT 分组和 SETUP 分组（用于数据接收）

USB 模块对这两种分组的处理方式基本相同；对 SETUP 分组的特殊处理将在下面关于控制传输部分详细说明。当接收到一个 OUT 或 SETUP 分组时，如果地址和某个有效端点的地址相匹配，USB 模块将访问缓冲区描述表，找到与该端点相关的 USB_ADDRx_RX 和 USB_COUNTx_RX 寄存器，并将 ADDRx_RX 寄存器的值保存在内部 ADDR 寄存器中。同时，COUNT 会被复位，从 USB_COUNTx_RX 中读出的 BL_SIZE 和 NUM_BLOCK 的值用于初始化内部 16 位寄存器 BUF_COUNT，该寄存器用于检测缓冲区溢出（所有的内部寄存器都不能被应用程序访问）。USB 模块将随后收到的数据按字方式组织（先收到的为低字节），并存储到 ADDR 指向的分组缓冲区中。同时，随着每个字节的传输，BUF_COUNT 值自动递减，COUNT 值自动递增。当检测到数据分组的结束信号时，USB 模块校验收到 CRC 的正确性。如果传输中没有任何错误发生，则发送 ACK 握手分组到主机。即使发生 CRC 错误或者其他类型的错误（位填充错误，帧错误等），数据还是会被保存到分组缓冲区中，至少会保存到发生错误的数据点，只是不会发送 ACK 分组，并且 USB_ISTR 寄存器的 ERR 位将会置位。在这种情况下，应用程序通常不需要干涉处理，USB 模块将从传输错误中自动恢复，并为下一次传输做好准备。如果收到的分组所对应的端点没有准备好，USB 模块将根据 USB_EPxR 寄存器的 STAT_RX 位发送 NAK 或 STALL 分组，数据将不会被写入接收缓冲区。

ADDRx_RX 的值决定接收缓冲区的起始地址，长度由包含 CRC 的数据分组的长度（即有效数据长度+2）决定，但不能超过 BL_SIZE 和 NUM_BLOCK 所定义的缓冲区的长度。如果接收到的数据分组的长度超出了缓冲区的范围，超过范围的数据不会被写入缓冲区，USB 模块将报告缓冲区发生溢出，并向主机发送 STALL 握手分组，通知此次传输失败，也不产生中断。

如果传输正确完成，USB 模块将发送 ACK 握手分组，内部的 COUNT 寄存器的值会被复制到相应的 COUNTx_RX 寄存器中，BL_SIZE 和 NUM_BLOCK 的值保持不变，也不需要重写。USB_EPxR 寄存器按下列方式更新：DTOG_RX 位翻转，STAT_RX=10（NAK）使端点无效，CTR_RX 位置位（如果 CTR 中断已使能（CTRM 位被置位），将触发中断）。如果传输过程中发生了错误或者缓冲区溢出，前面所列出的动作都不会发生。CRT 中断发生时，应用程序需要首先根据 USB_ISTR 寄存器的 EP_ID 和 DIR 位识别是哪个端点的中断请求。在处理 CTR_RX 中断事件时，应用程序首先要确定传输的类型（根据 USB_EPxR 寄存器的 SETUP 位），同时清除中断标志位，然后读相关的缓冲区描述表表项指向的 COUNTx_RX 寄存器，获得此次传输的总字节数。处理完接收到的数据后，应用程序需要将 USB_EPxR 中的 STAT_RX 位置成'11'，使能下一次的传输。当 STAT_RX 位为'10'时（NAK），任何一个发送到端点上的 OUT 请求都会被 NAK，PC 主机将不断重发被 NAK 的分组，直到收到端点的 ACK 握手分组。以上描述的操作次序是必须遵守的，以避免丢失紧随上一个 CTR 中断的另一个 OUT 分组请求。

22.3.2.6 控制传输

控制传输由 3 个阶段组成，首先是主机发送 SETUP 分组的 SETUP 阶段，然后是主机发送零个或多个数据的数据阶段，最后是状态阶段，由与数据阶段方向相反的数据分组构成。SETUP 传输只发生在控制端点，它非常类似于 OUT 分组的传输过程。使能 SETUP 传输除了需要分别初始化 DTOG_TX 位为'1'，DTOG_RX 位为'0'外，还需要设置 STAT_TX 位和 STAT_RX 位为 10（NAK），由应用程序根据 SETUP 分组的

相应字段决定后面的传输是 IN 还是 OUT。控制端点在每次发生 CTR_RX 中断时，都必须检查 USB_EPxR 寄存器的 SETUP 位，以识别是普通的 OUT 分组还是 SETUP 分组。USB 设备应该能够通过 SETUP 分组中的相应数据决定数据阶段传输的字节数和方向，并且能在发生错误的情况下发送 STALL 分组，拒绝数据的传输。因此在数据阶段，未被使用到的方向都应该被设置成 STALL，并且在开始传输数据阶段的最后一个数据分组时，其反方向的传输仍设成 NAK 状态，这样，即使主机立刻改变了传输方向（进入状态阶段），仍然可以保持为等待控制传输结束的状态。在控制传输成功结束后，应用程序可以把 NAK 变为 VALID，如果控制传输出错，就改为 STALL。此时，如果状态分组是由主机发送给设备的，那么 STATUS_OUT 位（USB_EPxR 寄存器中的 EP_KIND）应该被置位，只有这样，在状态传输过程中收到了非零长度的数据分组，才会产生传输错误。在完成状态传输阶段后，应用程序应该清除 STATUS_OUT 位，并且将 STAT_RX 设为 VALID 表示已准备好接收一个新的命令请求，STAT_TX 则设为 NAK，表示在下一个 SETUP 分组传输完成前，不接受数据传输的请求。

USB 规范定义 SETUP 分组不能以非 ACK 握手分组来响应，如果 SETUP 分组传输失败，则会引发下一个 SETUP 分组。因此，以 NAK 或 STALL 分组响应主机的 SETUP 分组是被禁止的。

当 STAT_RX 位被设置为’01’（STALL）或’10’（NAK）时，如果收到 SETUP 分组，USB 模块会接收分组，开始分组所要求的数据传输，并回送 ACK 握手分组。如果应用程序在处理前一个 CTR_RX 事件时 USB 模块又收到了 SETUP 分组（即 CTR_RX 仍然保持置位），USB 模块会丢掉收到的 SETUP 分组，并且不回答任何握手分组，以此来模拟一个接收错误，迫使主机再次发送 SETUP 分组。这样做是为了避免丢失紧随一次 CTR_RX 中断之后的又一个 SETUP 分组传输。

22.3.3 双缓冲端点

USB 标准不仅为不同的传输模式定义了不同的端点类型，而且对这些数据传输所需要的系统要求做了描述。其中，批量端点适用于在主机 PC 和 USB 设备之间传输大批量的数据，因为主机可以在一帧内利用尽可能多的带宽批量传输数据，使传输效率得到提高。然而，当 USB 设备处理前一次的数据传输时，又收到新的数据分组，它将回应 NAK 分组，使 PC 主机不断重发同样的数据分组，直到设备在可以处理数据时回应 ACK 分组。这样的重传占用了很多带宽，影响了批量传输的速率，因此引入了批量端点的双缓冲机制，提高数据传输率。

使用双缓冲机制时，单向端点的数据传输将使用到该端点的接收和发送两块数据缓冲区。数据翻转位用来选择当前使用到两块缓冲区中的哪一块，使应用程序可以在 USB 模块访问其中一块缓冲区的同时，对另一块缓冲区进行操作。例如，对一个双缓冲批量端点进行 OUT 分组传输时，USB 模块将来自 PC 主机的数据保存到一个缓冲区，同时应用程序可以对另一个缓冲区中的数据进行处理（对于 IN 分组来说，情况是一样的）。

因为切换缓冲区的管理机制需要用到所有 4 个缓冲区描述表的表项，分别用来表示每个方向上的两个缓冲区的地址指针和缓冲区大小，因此用来实现双缓冲批量端点的 USB_EPxR 寄存器必须配置为单向。所以只需要设定 STAT_RX 位（作为双缓冲批量接收端点）或者 STAT_TX 位（作为双缓冲批量发送端点）。如果需要一个双向的双缓冲批量端点，则须使用两个 USB_EPxR 寄存器。

为尽可能利用双缓冲的优势，达到较高的传输速率，双缓冲批量端点的流量控制流程与其他端点的稍有不同。它只在缓冲区发生访问冲突时才会设置端点为 NAK 状态，而不是在每次传输成功后都将端点设为 NAK 状态。

DTOG 位用来标识 USB 模块当前所使用的储存缓冲区。双缓冲批量端点接收方向的缓冲区由 DTOG_RX（USB_EPxR 寄存器的第 14 位）标识，而双缓冲批量端点发送方向的缓冲区由 DTOG_TX（USB_EPxR 寄存器的第 6 位）标识。同时，USB 模块也需要知道当前哪个缓冲区正在被应用程序使用，以避免发生冲突。由于 USB_EPxR 寄存器中有 2 个 DTOG 位，而 USB 模块只使用其中的一位来标识硬件所使用的缓冲区，因此，应用程序可使用另一位来标识当前正在使用哪个缓冲区，这个新的标识被称为 SW_BUF 位。下表列出了双缓冲批量端点在实现发送和接收操作时，USB_EPxR 寄存器的 DTOG 位和 SW_BUF 位之间的关系。

表 22-1 双缓冲批量端点缓冲区标识定义

缓冲区标识位	作为发送端点	作为接收端点
DTOG	DTOG_TX（USB_EPxR 寄存器的第 6 位）	DTOG_RX（USB_EPxR 寄存器的第 14 位）
SW_BUF	USB_EPxR 寄存器的第 14 位	USB_EPxR 寄存器的第 6 位

USB 模块当前使用的缓冲区由 DTOG 位标识，而应用程序所使用的缓冲区由 SW_BUF 位标识，这两个位的标识方式相同，下表描述了这种标识方式。

表 22-2 双缓冲批量端点的缓冲区使用标识

端点类型	DTOG 位	SW_BUF 位	USB 模块使用的缓冲区	应用程序使用的缓冲区
IN 端点	0	1	ADDRx_TX_0/COUNTx_TX_0	ADDRx_TX_1/COUNTx_TX_1
	1	0	ADDRx_TX_1/COUNTx_TX_1	ADDRx_TX_0/COUNTx_TX_0
	0	0	无(1)	ADDRx_TX_0/COUNTx_TX_0
	1	1	无(1)	ADDRx_TX_0/COUNTx_TX_0
OUT 端点	0	1	ADDRx_RX_0/COUNTx_RX_0	ADDRx_RX_1/COUNTx_RX_1
	1	0	ADDRx_RX_1/COUNTx_RX_1	ADDRx_RX_0/COUNTx_RX_0
	0	0	无(1)	ADDRx_RX_0/COUNTx_RX_0
	1	1	无(1)	ADDRx_RX_0/COUNTx_RX_0

(1) 端点处于 NAK 状态。

可以通过以下方式设置一个双缓冲批量端点：

- 将 USB_EPxR 寄存器的 EP_TYPE 位设为’00’，定义端点为批量端点。
- 将 USB_EPxR 寄存器的 EP_KIND 位设为’1’，定义端点为双缓冲端点。

应用程序根据传输开始时用到的缓冲区来初始化 DTOG 和 SW_BUF 位；这需要考虑到这两位的数据翻转特性。设置好 DBL_BUF 位之后，每完成一次传输后，USB 模块将根据双缓冲批量端点的流量控制操作，并且持续到 DBL_BUF 变为无效为止。每次传输结束，根据端点的传输方向，CTR_RX 位或 CTR_TX 位将会置为’1’。与此同时，硬件将设置相应的 DTOG 位，完全独立于软件来实现缓冲区交换机制。DBL_BUF 位设置后，每次传输结束时，双缓冲批量端点的 STAT 位的取值不会像其他类型端点一样受到传输过程的影响，而是一直保持为’11’（有效）。但是，如果在收到新的数据分组的传输请求时，USB 模块和应用程序发生了缓冲区访问冲突（即 DTOG 和 SW_BUF 为相同的值，见表 22-2，状态位将会被置为’10’（NAK）。应用程序响应 CTR 中断时，首先要清除中断标志，然后再处理传输完成的数据。应用程序访问缓冲区之后，需要翻转 SW_BUF 位，以通知 USB 模块该块缓冲区已变为可用状态。由此，双缓冲批量传输的 NAK 分组的数目只由应用程序处理一次数据传输的快慢所决定：如果数据处理的时间小于 USB 总线上完成一次数据传输的时间，则不会发生重传，此时，数据的传输率仅受限于 USB 主机。

应用程序也可以不考虑双缓冲批量端点的特殊控制流程，直接在相应 USB_EPxR 寄存器的 STAT 位写入不同于’11’（有效）的任何状态，在这种情况下，USB 模块将按照写入的状态执行流程而忽略缓冲器实际的使用情况。

22.3.4 同步传输

USB 标准定义了一种全速的需要保持固定和精确的数据传输率的传输方式：同步传输。同步传输一般用于传输音频流、压缩的视频流等对数据传输率有严格要求的数据。一个端点如果在枚举时被定义为

“同步端点”，USB 主机则会为每个帧分配固定的带宽，并且保证每个帧正好传送一个 IN 分组或者 OUT 分组（由端点传输方向确定分组类型）。为了满足带宽要求，同步传输中没有出错重传；这也就意味着，同步传输在发送或接收数据分组之后，无握手协议，即不会发送 ACK 分组。同样，同步传输只传送 PID（分组 ID）为 DATA0 的数据包，而不会用到数据翻转机制。

通过设置 USB_EPxR 寄存器 EP_TYPE 为’10’，可以使其成为同步端点。同步端点没有握手机制，根据 USB 标准中的说明，USB_EPxR 寄存器的 STAT_RX 位和 STAT_TX 位分别只能设成’00’（禁止）和’11’（有效）。同步传输通过实现双缓冲机制来简化软件应用程序开发，它同样使用两个缓冲区，以确保在 USB 模块使用其中一块缓冲区时，应用程序可以访问另外一块缓冲区。

USB 模块使用的缓冲区根据不同的传输方向，由不同的 DTOG 位来标识。（同一寄存器中的 DTOG_RX 位用来标识接收同步端点，DTOG_TX 位用来标识发送同步端点），见下表。

表 22-3 同步端点的缓冲区使用标识

端点类型	DTOG 位值	USB 模块使用的缓冲区	应用程序使用的缓冲区
IN 端点	0	ADDRx_TX_0/COUNTx_TX_0	ADDRx_TX_1/COUNTx_TX_1
	1	ADDRx_TX_1/COUNTx_TX_1	ADDRx_TX_0/COUNTx_TX_0
OUT 端点	0	ADDRx_RX_0/COUNTx_RX_0	ADDRx_RX_1/COUNTx_RX_1
	1	ADDRx_RX_1/COUNTx_RX_1	ADDRx_RX_0/COUNTx_RX_0

与双缓冲批量端点一样，一个 USB_EPxR 寄存器只能处理同步端点单方向的数据传输，如果要求同步端点在两个传输方向上都有效，则需要使用两个 USB_EPxR 寄存器。

应用程序需要根据首次传输的数据分组来初始化 DTOG 位；它的取值还需要考虑到 DTOG_RX 或 DTOG_TX 两位的数据翻转特性。每次传输完成时，USB_EPxR 寄存器的 CTR_RX 位或 CTR_TX 位置位。与此同时，相关的 DTOG 位由硬件翻转，从而使得交换缓冲区的操作完全独立于应用程序。传输结束时，STAT_RX 或 STAT_TX 位不会发生变化，因为同步传输没有握手机制，所以不需要任何流量控制，而一直设为’11’（有效）。同步传输中，即使 OUT 分组发生 CRC 错误或者缓冲区溢出，本次传输仍被看作是正确的，并且可以触发 CTR_RX 中断事件；但是，发生 CRC 错误时硬件会设置 USB_ISTR 寄存器的 ERR 位，提醒应用程序数据可能损坏。

22.3.5 挂起/唤醒事件

USB 标准中定义了一种特殊的设备状态，即挂起状态，在这种状态下 USB 总线上的平均电流消耗不超过 2.5mA。这种电流限制对于由总线供电的 USB 设备至关重要，而自供电的设备则不需要严格遵守这样的电流消耗限制。USB 主机以 3 毫秒内不发送任何信号标志进入挂起状态。通常情况下 USB 主机每毫秒会发送一个 SOF，当 USB 模块检测到 3 个连续的 SOF 分组丢失事件即可判定主机发出了挂起请求，接着它会置位 USB_ISTR 寄存器的 SUSP 位，以触发挂起中断。USB 设备进入挂起状态之后，将由“唤醒”序列唤醒。所谓的“唤醒”序列，可以由 USB 主机发起，也可以由 USB 设备本身触发；但是，只有 USB 主机可以结束“唤醒”序列。被挂起的 USB 模块必须至少还具备检测 RESET 信号的功能，它会将其当作一次正常的复位操作来执行。

实际的挂起操作过程对于不同的 USB 设备来说是不同的，因为需要不同的操作来降低电源消耗。下面描述了一起典型的挂起操作，重点介绍应用程序如何响应 USB 模块的 SUSP 信号：

1. 将 USB_CNTR 寄存器的 FSUSP 置为’1’，这将使 USB 模块进入挂起状态。USB 模块一旦进入挂起状态，对 SOF 的检测立刻停止，以避免在 USB 挂起时又发生新的 SUSP 事件。

2. 消除或减少 USB 模块以外的其他模块的静态电流消耗。

3. 可以选择关闭外部振荡器和设备的 PLL，以停止设备内部的任何活动。

当设备处于挂起状态时发生 USB 事件，该设备会被唤醒，并需要调用“唤醒”程序来恢复正常时钟和 USB 数据传输。如果唤醒设备的是 USB 复位操作，则应该保证唤醒的过程不要超 10 毫秒（参见 USB 协议规范）。即使唤醒事件可以立刻触发一个使能了的 WKUP 中断事件，但由于恢复系统时钟需要比较长的延迟时间，处理 WKUP 中断的中断服务程序必须考虑到这点；为了减短系统唤醒的时间，建议将唤醒代码直接写在挂起代码后面，这样就可以在系统时钟重启后迅速进入唤醒代码中执行。为防止或减少 ESD 等干扰意外地唤醒系统（从挂起模式退出是一个异步事件），在挂起过程中数据线被过滤，滤波宽度大约为 70ns。

下面是唤醒操作的过程：

1. 启动外部振荡器和设备的 PLL（此项可选）。

2. 清零 USB_CNTR 寄存器的 FSUSP 位。

3. USB_FNR 寄存器的 RXDP 和 RXDM 位可以用来判断是什么触发了唤醒事件，如表 22-4 所示，它还同时列出了各种情况软件应该采取的操作。如果需要的话，可以通过检测这两位变成’10’（代表空闲总线状态）的时间来知道唤醒或复位事件的结束。此外，在复位事件结束时，USB_ISTR 寄存器的 RESET 位被置为’1’，如果 RESET 中断被使能，就会产生中断。此中断应该按正常的复位操作处理。

表 22-4 唤醒事件检测

[RXDP，RXDM]的状态	唤醒事件	应用程序应执行的操作
00	复位	无
10	无（总线干扰）	恢复到挂起状态
01	恢复挂起	无
11	未定义的值（总线干扰）	恢复到挂起状态

设备可能不是被与 USB 模块相关的事件唤醒的（例如一个鼠标的移动可唤醒整个系统）。在这种情况下，先将 USB_CNTR 寄存器的 RESUME 位置为’1’，然后在 1ms 到 15ms 之间再把它清为 0 可以启动唤醒序列（这个间隔可以用 ESOF 中断来实现，该中断在内核正常运行时每 1ms 发生一次）。RESUME 位被清零后，唤醒过程将由主机 PC 完成，可以利用 USB_FNR 寄存器的 RXDP 和 RXDM 位来判断唤醒是否完成。

注意：只有在 USB 模块被设置为挂起状态时（设置 USB_CNTR 寄存器的 FSUSP 位为’1’），才可以设置 RESUME 位。

22.4 USB 接口特性

USB（全速）接口已通过 USB-IF 认证。

表 22-5 USB 启动时间

符号	参数	最大值	单位
$t_{STARTUP}$(1)	USB 收发器启动时间	1	μs

(1). 由设计保证，不在生产中测试。

表 22-6 USB 直流特性

符号	参数	条件	最小值(1)	最大值(1)	单位
输入电平					
V_{DD}	USB 操作电压(2)	-	3.0(3)	3.6	V
V_{DI}(4)	差分输入灵敏度	I(USBDP，USBDM)	0.2		V
V_{CM}(4)	差分共模范围	包含 V_{DI} 范围	0.8	2.5	V
V_{SE}(4)	单端接收器阈值	-	1.3	2.0	V
输出电平					
V_{OL}	静态输出低电平	1.5KΩ 的 R_L 接至 3.6V(5)	-	0.3	V
V_{OH}	静态输出高电平	1.5KΩ 的 R_L 接至 VSS(5)	2.8	3.6	V

(1). 所有的电压测量都是以设备端地线为准。

(2). 为了与 USB2.0 全速电气规范兼容，USBDP（D+）引脚必须通过一个 1.5kΩ电阻接至 3.0~3.6V 电压。

(3). 正确 USB 功能可以在 2.7V 得到保证，而不是在 2.7~3.0V 电压范围下降级的电气特性。

(4). 由综合评估保证，不在生产中测试。

(5). RL 是连接到 USB 驱动器上的负载。

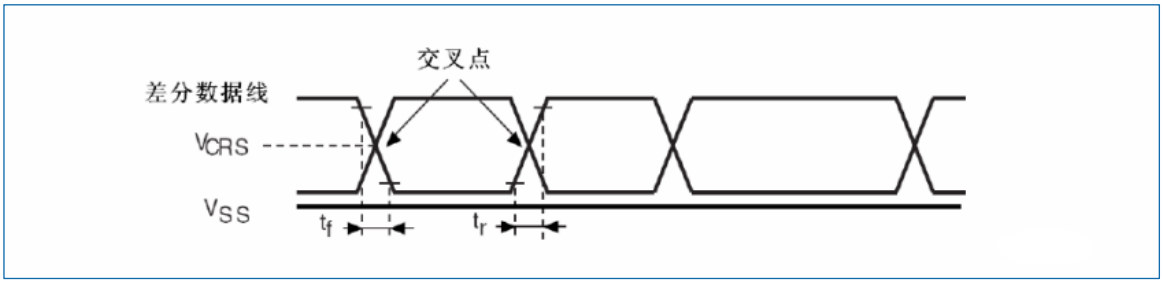


图 22-3 USB 时序:数据信号上升和下降时间定义

表 22-7 USB 全速电气特性

符号	参数	条件	最小值(1)	最大值(1)	单位
t_r	上升时间(2)	CL≤50pF	4	20	ns
t_f	下降时间(2)	CL≤50pF	4	20	ns
t_{rfm}	上升下降时间匹配	t_r/t_f	90	110	%
V_{CRS}	输出信号交叉电压	-	1.3	2.0	V

(1). 由设计保证，不在生产中测试。

(2). 测量数据信号从 10%至 90%。更多详细信息，参见 USB 规范第 7 章(2.0 版)。

22.5 USB 寄存器

基地址：0x4000 5C00

空间大小：0x400

USB 模块的寄存器有以下三类：

- 通用类寄存器：中断寄存器和控制寄存器。
- 端点类寄存器：端点配置寄存器和状态寄存器。
- 缓冲区描述表类寄存器：用来确定数据分组存放地址的寄存器缓冲区描述表类寄存器的基地址由 USB_BTABLE 寄存器指定，所有其他寄存器的基地址则为 USB 模块的基地址 0x4000 5C00。由于 APB1 总线按 32 位寻址，因此所有的 16 位寄存器的地址都是按 32 位字对齐的。同样的地址对齐方式也用于从 0x4000 6000 开始的分组缓冲存储区。

可以用半字（16 位）或字（32 位）的方式操作这些外设寄存器。

22.5.1 通用寄存器

这组寄存器用于定义 USB 模块的工作模式，中断的处理，设备的地址和读取当前帧的编号。

22.5.1.1 USB 控制寄存器（USB_CNTR）

偏移地址：0x40

复位值：0x0003

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTRM	PMAOVRM	ERRM	WKUPM	SUSPM	RESETM	SOFM	ESOFM	Res			RESUME	FSUSP	Res		FRES
rw	rw	rw	rw	rw	rw	rw	rw				rw	rw			rw

位 15	CTRM：正确传输中断屏蔽位（Correct transfer interrupt mask） ● 0：正确传输中断禁止。 ● 1：正确传输中断使能，在中断寄存器的相应位被置 1 时产生中断。
位 14	PMAOVRM：分组缓冲区溢出中断屏蔽位（Packet memory area over/underrun interrupt mask） ● 0：分组缓冲区溢出中断禁止。 ● 1：分组缓冲区溢出中断使能，在中断寄存器的相应位被置 1 时产生中断。
位 13	ERRM：出错中断屏蔽位（Error interrupt mask） ● 0：出错中断禁止。 ● 1：出错中断使能，在中断寄存器的相应位被置 1 时产生中断。
位 12	WKUPM：唤醒中断屏蔽位（Wakeup interrupt mask） ● 0：唤醒中断禁止。 ● 1：唤醒中断使能，在中断寄存器的相应位被置 1 时产生中断。
位 11	SUSPM：挂起中断屏蔽位（Suspend mode interrupt mask） ● 0：挂起中断禁止。 ● 1：挂起中断使能，在中断寄存器的相应位被置 1 时产生中断。
位 10	RESETM：USB 复位中断屏蔽位（USB reset interrupt mask） ● 0：USB 复位中断禁止。 ● 1：USB 复位中断使能，在中断寄存器的相应位被置 1 时产生中断。
位 9	SOFM：帧起始位中断屏蔽位（Start of frame interrupt mask） ● 0：帧起始位中断禁止。 ● 1：帧起始位中断使能，在中断寄存器的相应位被置 1 时产生中断。
位 8	ESOFM：期望帧起始位中断屏蔽位（Expected start of frame interrupt mask） ● 0：期望帧起始位中断禁止。

	● 1：期望帧起始位中断使能，在中断寄存器的相应位被置 1 时产生中断。
位 7:5	Res：保留 必须保持复位值。
位 4	RESUME：唤醒请求（Resume request） 设置此位将向 PC 主机发送唤醒请求。根据 USB 协议，如果此位在 1ms 到 15ms 内保持有效，主机将对 USB 模块实行唤醒操作。
位 3	FSUSP：强制挂起（Force suspend） 当 USB 总线上保持 3ms 没有数据通信时，SUSP 中断会被触发，此时软件必需设置此位。 ● 0：无效 ● 1：进入挂起模式
位 2:1	Res：保留 必须保持复位值。
位 0	FRES：强制 USB 复位（Force USB reset） ● 0：清除 USB 复位信号。 ● 1：对 USB 模块强制复位，类似于 USB 总线上的复位信号。 USB 模块将一直保持在复位状态下直到软件清除此位。如果 USB 复位中断被使能，将产生一个复位中断。

22.5.1.2 USB 中断状态寄存器（USB_ISTR）

偏移地址：0x44

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR	PMAOVR	ERR	WKUP	SUSP	RESET	SOF	ESOF	Res			DIR	EP_ID[3:0]			
rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0				r	r			

位 15	CTR：正确的传输（Correct transfer） 此位在端点正确完成一次数据传输后由硬件置位。应用程序可以通过 DIR 和 EP_ID 位来识别是哪个端点完成了正确的数据传输。 此位应用程序只读。
位 14	PMAOVR：分组缓冲区溢出（Packet memory area over/underrun） 此位在微控制器长时间没有响应一个访问 USB 分组缓冲区请求时由硬件置位。USB 模块通常在以下情况时置位该位：在接收过程中一个 ACK 握手分组没有被发送，或者在发送过程中发生了比特填充错误，在以上两种情况下主机都会要求数据重传。在正常的数据传输中不会产生 PMAOVR 中断。由于失败的传输都将由主机发起重传，应用程序就可以在这个中断的服务程序中加速设备的其他操作，并准备重传。但这个中断不会在同步传输中产生（同步传输不支持重传）因此数据可能会丢失。 此位应用程序可读可写，但只有写 0 有效，写 1 无效。
位 13	ERR：出错（Error） 在下列错误发生时硬件会置位此位： ● NANS：无应答 主机的应答超时。 ● CRC：循环冗余校验码错误 数据或令牌分组中的 CRC 校验出错。 ● BST：位填充错误 PID，数据或 CRC 中检测出位填充错误。

	● FVIO：帧格式错误 收到非标准帧（如 EOP 出现在错误的时刻，错误的令牌等）。 USB 应用程序通常可以忽略这些错误，因为 USB 模块和主机在发生错误时都会启动重传机制此位产生的中断可以用于应用程序的开发阶段，可以用来监测 USB 总线的传输质量，标识用户可能发生的错误（连接线松，环境干扰严重，USB 线损坏等）。 此位应用程序可读可写，但只有写 0 有效，写 1 无效。
位 12	WKUP：唤醒请求（Wakeup request） 当 USB 模块处于挂起状态时，如果检测到唤醒信号，此位将由硬件置位。USB_WAKEUP 被激活，通知设备的其他部分（如唤醒单元）将开始唤醒过程。 此位应用程序可读可写，但只有写 0 有效，写 1 无效。
位 11	SUSP：挂起模块请求（Suspend mode request） 此位在 USB 线上超过 3ms 没有信号传输时由硬件置位，用以指示一个来自 USB 总线的挂起请求。USB 复位后硬件立即使能对挂起信号的检测，但在挂起模式下（FSUSP＝1）硬件不会再检测挂起信号直到唤醒过程结束。 此位应用程序可读可写，但只有写 0 有效，写 1 无效。
位 10	RESET：USB 复位请求（USB reset request） 此位在 USB 模块检测到 USB 复位信号输入时由硬件置位。此时 USB 模块将复位内部协议状态机，并在中断使能的情况下触发复位中断来响应复位信号。USB 模块的发送和接收部分将被禁止，直到此位被清除。所有的配置寄存器不会被复位，除非应用程序对它们清零。这用来保证在复位后 USB 传输还可以立即正确执行。但设备的地址和端点寄存器会被 USB 复位所复位。 此位应用程序可读可写，但只有写 0 有效，写 1 无效。
位 9	SOF：帧起始位标志（Start of frame） 此位在 USB 模块检测到总线上的 SOF 分组时由硬件置位，标志一个新的 USB 帧的开始。中断服务程序可以通过检测 SOF 事件来完成与主机的 1ms 同步，并正确读出寄存器在收到 SOF 分组时的更新内容（此功能在同步传输时非常有意义）。 此位应用程序可读可写，但只有写 0 有效，写 1 无效。
位 8	ESOF：期望帧起始位标识位（Expected start of frame） 此位在 USB 模块未收到期望的 SOF 分组时由硬件置位。主机应该每毫秒都发送 SOF 分组，但如果 USB 模块没有收到，挂起定时器将触发此中断。如果连续发生 3 次 ESOF 中断，也就是连续 3 次未收到 SOF 分组，将产生 SUSP 中断。即使在挂起定时器未被锁定时发生 SOF 分组丢失，此位也会被置位。 此位应用程序可读可写，但只有写 0 有效，写 1 无效。
位 7:5	Res：保留 必须保持复位值。
位 4	DIR：传输方向（Direction of transaction） 此位在完成数据传输产生中断后由硬件根据传输方向写入。 ● 0：相应端点的 CTR_TX 位被置位 标志一个 IN 分组（数据从 USB 模块传输到 PC 主机）的传输完成。 ● 1：相应端点的 CTR_RX 位被置位 标志一个 OUT 分组（数据从 PC 主机传输到 USB 模块）的传输完成。如果 CTR_TX 位同时也被置位，就标志同时存在挂起的 OUT 分组和 IN 分组。应用程序可以利用该信息访问 USB_EPxR 位对应的操作，它表示挂起中断传输方向的信息该位为只读。
位 3:0	EP_ID[3:0]：端点 ID（Endpoint Identifier） 此位在 USB 模块完成数据传输产生中断后由硬件根据请求中断的端点号写入。如果同时有多个端点的请求中断，硬件写入优先级最高的端点号。

	端点的优先级按以下方法定义：同步端点和双缓冲批量端点具有高优先级，其他的端点为低优先级。如果多个同优先级的端点请求中断，则根据端点号来确定优先级，即端点 0 具有最高优先级，端点号越小，优先级越高。应用程序可以通过上述的优先级策略顺序处理端点的中断请求。 该位为只读。

此寄存器包含所有中断源的状态信息，以供应用程序确认产生中断请求的事件。

寄存器的高 8 位各表示一个中断源。当相关事件发生时，这些位被硬件置位，如果 USB_CNTR 寄存器上的相应位也被置位，则会产生相应的中断。中断服务程序需要检查每个位，在执行必要的操作后必需清除相应的状态位，不然中断信号线一直保持为高，同样的中断会再次被触发。如果同时多个中断标志被设置，也只会产生一个中断。

应用程序可以使用不同的方式处理传输完成中断，以减少中断响应的延迟时间。端点在成功完 成一次传输后，CTR 位会被硬件置起，如果 USB_CNTR 上的相应位也被设置的话，就会产生中断。与端点相关的中断标志和 USB_CNTR 寄存器的 CTRM 位无关。这两个中断标志位将一直保持有效，直到应用程序清除了 USB_EPxR 寄存器中的相关中断挂起位（CTR 位是个只读位）。

USB 模块有两路中断请求源：

- 高优先级的 USB IRQ：用于高优先级的端点（同步和双缓冲批量端点）的中断请求，并且该中断不能被屏蔽。
- 低优先级 USB IRQ：用于其他中断事件，可以是低优先级的不可屏蔽中断，也可以是由 USB_ISTR 寄存器的高 8 位标识的可屏蔽中断。

对于端点产生的中断，应用程序可以通过 DIR 寄存器和 EP_ID 只读位来识别中断请求由哪个端点 产生，并调用相应的中断服务程序。

用户在处理同时发生的多个中断事件时，可以在中断服务程序里检查 USB_ISTR 寄存器各个位的顺序来确定这些事件的优先级。在处理完相应位的中断后需要清零该中断标志。完成一次中 断服务后，另一中断请求将会产生，用以请求处理剩下的中断事件。

为了避免意外清零某些位，建议使用加载指令，对所有不需改变的位写'1'，对需要清除的位写'0'。对于该寄存器，不建议使用读出－修改－写入的流程，因为在读写操作之间，硬件可能 需要设置某些位，而这些位会在写入时被清零。

22.5.1.3 USB 帧编号寄存器（USB_FNR）

偏移地址：0x48

复位值：0x0XXX

说明：X 代表未定义数值。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDP	RXDM	LCK	LSOF[1:0]		FN[10:0]										
r	r	r	r		r										

位 15	RXDP：D+状态位（Receive data+line status） 此位用于观察 USBD+数据线的状态，可在挂起状态下检测唤醒条件的出现。
位 14	RXDM：D-状态位（Receive data-line status） 此位用于观察 USBD-数据线的状态，可在挂起状态下检测唤醒条件的出现。
位 13	LCK：锁定位（Locked） USB 模块在复位或唤醒序列结束后会检测 SOF 分组，如果连续检测到至少 2 个 SOF 分组，则硬件会置位此位。此位一旦锁定，帧计数器将停止计数，一直等到 USB 模块复位或总线挂起时再恢复计数。

位 12:11	LSOF[1:0]：帧起始位丢失标志位（Lost SOF） 当 ESOF 事件发生时，硬件会将丢失的 SOF 分组的数目写入此位。如果再次收到 SOF 分组，引脚会清除此位。
位 10:0	FN[10:0]：帧编号（Frame number） 此部分记录了最新收到的 SOF 分组中的 11 位帧编号。主机每发送一个帧，帧编号都会自加，这对于同步传输非常有意义。此部分发生 SOF 中断时更新。

22.5.1.4 USB 设备地址寄存器（USB_DADDR）

偏移地址：0x4C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								EF	ADD[6:0]						
								rw	rw						

位 15:8	Res：保留 必须保持复位值。
位 7	EF：USB 模块使能位（Enable function） 此位在需要使能 USB 模块时由应用程序置位。ADD[6:0]包含了设备地址。如果此位为 0，USB 模块将停止工作，忽略所有寄存器的设置，不响应任何 USB 通信。
位 6:0	ADD[6:0]：设备地址（Device address） 该位域记录了 USB 主机在枚举过程中为 USB 设备分配的地址值。该地址值和端点地址（EA）必需和 USB 令牌分组中的地址信息匹配，才能在指定的端点进行正确的 USB 传输。

22.5.1.5 USB 分组缓冲区描述表地址寄存器（USB_BTABLE）

偏移地址：0x50

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BTABLE[15:3]													Res		
rw															

位 15:3	BTABLE[15:3]：缓冲表（Buffer table） 该位域记录分组缓冲区描述表的起始地址。分组缓冲区描述表用来指示每个端点的分组缓冲区地址和大小，按 8 字节对齐（即最低 3 位为 000）。每次传输开始时，USB 模块读取相应端点所对应的分组缓冲区描述表获得缓冲区地址和大小信息。
位 2:0	Res：保留 必须保持复位值。

22.5.2 端点寄存器

端点寄存器的数量由 USB 模块所支持的端点数目决定。USB 模块最多支持 8 个双向端点。每个 USB 设备必须支持一个控制端点，控制端点的地址（EA 位）必需为 0。不同的端点必需使用不同的端点号，否则端点的状态不定。每个端点都有与之对应的 USB_EPxR 寄存器，用于存储该端点的各种状态信息。

22.5.2.1 USB 端点 x 寄存器（USB_EPxR）（x=0..7）

偏移地址：x*4

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR_RX	DTOG_RX	STAT_RX[1:0]		SETUP	EP_TYPE[1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX[1:0]		EA[3:0]			
rc_w0	t	t		r	rw		rw	rc_w0	t	t		rw			

位 15	CTR_RX：正确接收标志位（Correct Transfer for reception） 此位在正确接收到 OUT 或 SETUP 分组时由硬件置位，应用程序只能对此位清零。如果 CTRM 位已置位，相应的中断会产生。收到的是 OUT 分组还是 SETUP 分组可以通过下面描述的 SETUP 位确定。以 NAK 或 STALL 结束的分组和出错的传输不会导致此位置位，因为没有真正传输数据。 此位应用程序可读可写，但只有写 0 有效，写 1 无效。
位 14	DTOG_RX：用于数据接收的数据翻转位（Data Toggle, for reception transfers） 对于非同步端点，此位由硬件设置，用于标记希望接收的下一个数据分组的 Toggle 位（0=DATA0，1=DATA1）。在接收到 PID（分组 ID）正确的数据分组之后，USB 模块发送 ACK 握手分组，并翻转此位。 对于控制端点，硬件在收到 SETUP 分组后清除此位。 对于双缓冲端点，此位还用于支持双缓冲区的交换（请参考章节：“22.3.3 双缓冲端点”）。 对于同步端点，由于仅发送 DATA0，因此该位仅用于支持双缓冲区的交换（请参考章节：“22.3.4 同步传输”）而不需进行翻转。同步传输不需要握手分组，因此硬件在收到数据分组后立即设置此位。 应用程序可以对此位进行初始化（对于非控制端点，初始化是必需的），或者翻转此位用于特殊用途。 此位应用程序可读可写，但写 0 无效，写 1 可以翻转此位。
位 13:12	STAT_RX[1:0]：用于数据接收的状态位（Status bits, for reception transfers） 该位域用于指示端点当前的状态，表 22-8 列出了端点的所有状态。当一次正确的 OUT 或 SETUP 数据传输完成后（CTR_RX=1），硬件会自动设置此位为 NAK 状态，使应用程序有足够的时间在处理完当前传输的数据后，响应下一个数据分组。 对于双缓冲批量端点，由于使用特殊的传输流量控制策略，因此根据使用的缓冲区状态控制传输状态（请参考章节：“22.3.3 双缓冲端点”）。 对于同步端点，由于端点状态只能是有效或禁用，因此硬件不会在正确的传输之后设置此位。如果应用程序将此位设为 STALL 或者 NAK，USB 模块响应的操作是未定义的。 此位应用程序可读可写，但写 0 无效，写 1 翻转此位。
位 11	SETUP：SETUP 分组传输完成标志位（Setup transaction completed） 此位在 USB 模块收到一个正确的 SETUP 分组后由硬件置位，只有控制端点才使用此位。在接收完成后（CTR_RX=1），应用程序需要检测此位以判断完成的传输是否是 SETUP 分组。为了防止中断服务程序在处理 SETUP 分组时下一个令牌分组修改了此位，只有 CTR_RX 为 0 时，此位才可以被修改，CTR_RX 为 1 时不能修改。 此位应用程序只读。
位 10:9	EP_TYPE[1:0]：端点类型位（Endpoint type） 该位域用于指示端点当前的类型，所有的端点类型都在表 22-9 中列出。 所有的 USB 设备都必需包含一个地址为 0 的控制端点，如果需要可以有其他地址的控制端点。只有控制端点才会有 SETUP 传输，其他类型的端点无视此类传输。 SETUP 传输不能以 NAK 或 STALL 分组响应，如果控制端点在收到 SETUP 分组时处于 NAK 状态，USB 模块将不响应分组，就会出现接收错误。如果控制端点处于 STALL 状态，SETUP 分组会被正确接收，数据会被正确传输，并产生一个正确传输完成的中断。控制端点的 OUT 分组安装普通端点的方式处理。 批量端点和中断端点的处理方式非常类似，仅在对 EP_KIND 位的处理上有差别。同步端点的用法请参考章节：“22.3.4 同步传输”。
位 8	EP_KIND：端点特殊类型位（Endpoint kind） 此位的需要和 EP_TYPE 位配合使用，具体的定义请参考表 22-10。 DBL_BUF：应用程序设置此位能使能批量端点的双缓冲功能。详见章节：“22.3.3 双缓冲端点”。 STATUS_OUT：应用程序设置此位表示 USB 设备期望主机发送一个状态数据分组，此时，设备对于任何长

	度不为 0 的数据分组都响应 STALL 分组。此功能仅用于控制端点，有利于提供应用程序对于协议层错误的检测。如果 STATUS_OUT 位被清除，OUT 分组可以包含任意长度的数据。
位 7	CTR_TX：正确发送标志位（Correct Transfer for transmission） 此位由硬件在一个正确的 IN 分组传输完成后置位。如果 CTRM 位已被置位，会产生相应的中断。应用程序需要在处理完该事件后清除此位。在 IN 分组结束时，如果主机响应 NAK 或 STALL 则此位不会被置位，因为数据传输没有成功。 此位应用程序可读可写，但写 0 有效，写 1 无效。
位 6	DTOG_TX：发送数据翻转位（Data Toggle, for transmission transfers） 对于非同步端点，此位用于指示下一个要传输的数据分组的 Toggle 位（0＝DATA0，1=DATA1）。在一个成功传输的数据分组后，如果 USB 模块接收到主机发送的 ACK 分组，就会翻转此位。 对于控制端点，USB 模块会在收到正确的 SETUP PID 后置位此位。 对于双缓冲端点，此位还可用于支持分组缓冲区交换（请参考章节：“22.3.3 双缓冲端点”）。 对于同步端点，由于只传送 DATA0，因此该位只用于支持分组缓冲区交换（请参考章节：“22.3.4 同步传输”）。由于同步传输不需要握手分组，因此硬件在接收到数据分组后即设置该位。 应用程序可以初始化该位（对于非控制端点，初始化此位时必需的），也可以设置该位用于特殊用途。 此位应用程序可读可写，但写 0 无效，写 1 翻转此位。
位 5:4	STAT_TX[1:0]：用于发送数据的状态位（Status bits, for transmission transfers） 该位域用于标识端点的当前状态，表 22-11 列出了所有的状态。应用程序可以翻转这些位来初始化状态信息。在正确完成一次 IN 或者 SETUP 分组的传输后（CTR_TX=1），硬件会自动设置此位为 NAK 状态，保证应用程序有足够的时间准备好数据响应后续的数据传输。 对于双缓冲批量端点，由于使用特殊的传输流量控制策略，是根据缓冲区的状态控制传输的状态的（请参考章节：“22.3.3 双缓冲端点”）。 对于同步端点，由于端点的状态只能是有效或禁用，因此硬件不会在数据传输结束时改变端点的状态。如果应用程序将此位设为 STALL 或者 NAK，则 USB 模块后续的操作是未定义的。 此位应用程序可读可写，但写 0 无效，写 1 翻转此位。
位 3:0	EA[3:0]：端点地址（Endpoint address） 应用程序必需设置此 4 位，在使能一个端点前为它定义一个地址。

当 USB 模块收到 USB 总线复位信号，或 CTLR 寄存器的 FRES 位置位时，USB 模块将会复位。该寄存器除了 CTR_RX 和 CTR_TX 位保持不变以处理紧随的 USB 传输外，其他位都被复位。每个端点对应一个 USB_EPxR 寄存器，其中 n 为端点地址，即端点 ID 号。

对于此类寄存器应避免执行读出－修改－写入操作，因为在读和写操作之间，硬件可能会设置 某些位，而这些位又会在写入时被修改，导致应用程序错过相应的操作。因此，这些位都有一个写入无效的值，建议用 Load 指令修改这些寄存器，以免应用程序修改了不需要修改的位。

表 22-8 接收状态编码

STAT_RX[1:0]	描述
00	DISABLED：端点忽略所有的接收请求。
01	STALL：端点以 STALL 分组响应所有的接收请求。
10	NAK：端点以 NAK 分组响应所有的接收请求。
11	VALID：端点可用于接收。

表 22-9 端点类型编码

EP_TYPE[1:0]	描述
00	BULK：批量端点
01	CONTROL：控制端点
10	ISO：同步端点
11	INTERRUPT：中断端点

表 22-10 端点特殊类型定义

EP_TYPE[1:0]		EP_KIND 意义
00	BULK	DBL_BUF：双缓冲端点
01	CONTROL	STATUS_OUT
10	ISO	未使用
11	INTERRUPT	未使用

表 22-11 发送状态编码

STAT_TX[1:0]	描述
00	DISABLED：端点忽略所有的发送请求。
01	STALL：端点以 STALL 分组响应所有的发送请求。
10	NAK：端点以 NAK 分组响应所有的发送请求。
11	VALID：端点可用于发送。

22.5.3 缓冲区描述表

基地址：0x4000 6000

空间大小：0x200

说明：此缓冲区和 CAN 共用，USB 和 CAN 可以同时用于一个应用中但不能在同一个时间使用。

虽然缓冲区描述表位于分组缓冲区内，但仍可将它看作是特殊的寄存器，用以配置 USB 模块和微控制器内核共享的分组缓冲区的地址和大小。由于 APB1 总线按 32 位寻址，所以所有的分组缓冲区地址都使用 32 位对齐的地址，而不是 USB_BTABLE 寄存器和缓冲区描述表所使用的地址。

以下介绍两种地址表示方式：一种是应用程序访问分组缓冲区时使用的，另一种是相对于 USB 模块的本地地址。供应用程序使用的分组缓冲区地址需要乘以 2 才能得到缓冲区在微控制器中的真正地址。分组缓冲区的首地址为 0x4000 6000。下面将描述与 USB_EPxR 寄存器相关的缓冲区描述表。完整的分组缓冲区的说明和缓冲区描述表的用法请参考“22.3.2.2 分组缓冲区的结构和用途”。

22.5.3.1 发送缓冲区地址寄存器 x（USB_ADDRx_TX）（x=0..7）

偏移地址：[USB_BTABLE]+x*16

USB 本地地址：[USB_BTABLE]+x*8

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRx_TX[15:1]															0
rw															r

位 15:1	ADDRx_TX[15:1]：发送缓冲区地址（Transmission buffer address） 此位记录了收到下一个 IN 分组时，需要发送的数据所在的缓冲区起始地址。
位 0	因为分组缓冲区的地址必须按字对齐，所以此位必须写为'0'。

22.5.3.2 发送数据字节数寄存器 x（USB_COUNTx_TX）（x=0..7）

偏移地址：[USB_BTABLE]+x*16+4

USB 本地地址：[USB_BTABLE]+x*8+2

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res						COUNTx_TX[9:0]									
						rw									

位 15:10	Res：保留 必须保持复位值（由于 USB 模块支持的最大数据分组为 1023 个字节，所以 USB 模块忽略该位域）。
位 9:0	COUNTx_TX[9:0]：发送数据字节数（Transmission byte count） 该位域记录了收到下一个 IN 分组时要传输的数据字节数。

双缓冲区和同步 IN 端点有两个 USB_COUNTx_TX 寄存器：分别为 USB_COUNTx_TX_1 和 USB_COUNTx_TX_0，内容如下：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res						COUNTx_TX_1[9:0]									
						rw									

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res						COUNTx_TX_0[9:0]									
						rw									

位 31:26	Res：保留 必须保持复位值（由于 USB 模块支持的最大数据分组为 1023 个字节，所以 USB 模块忽略该位域，相当于非双缓冲和非同步端点的发送数据字节寄存器）。
位 25:16	COUNTx_TX_1[9:0]：发送数据字节数 1（Transmission byte count1） 该位域记录了收到下一个 IN 分组时要传输的数据字节数，相当于非双缓冲和非同步端点的发送数据字节寄存器。
位 15:10	Res：保留 必须保持复位值（由于 USB 模块支持的最大数据分组为 1023 个字节，所以 USB 模块忽略该位域，相当于非双缓冲和非同步端点的发送数据字节寄存器）。
位 9:0	COUNTx_TX_0[9:0]：发送数据字节数 0（Transmission byte count0） 该位域记录了收到下一个 IN 分组时要传输的数据字节数，相当于非双缓冲和非同步端点的接收数据字节寄存器。

22.5.3.3 接收缓冲区地址寄存器 x（USB_ADDRx_RX）（x=0..7）

偏移地址：[USB_BTABLE]+x*16+8

USB 本地地址：[USB_BTABLE]+x*8+4

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRx_RX[15:1]															0
rw															r

位 15:1	ADDRx_RX[15:1]：接收缓冲区地址（Reception buffer address）

	此位记录了收到下一个 OUT 或者 SETUP 分组时，用于保存数据的缓冲区起始地址。
位 0	因为分组缓冲区的地址按字对齐，所以此位必须写为’0’。

22.5.3.4 接收数据字节数寄存器 x（USB_COUNTx_RX）（x=0..7）

偏移地址：[USB_BTABLE]+x*16+12

USB 本地地址：[USB_BTABLE]+x*8+6

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BL_SIZE	NUM_BLOCK[4:0]					COUNTx_RX[9:0]									
rw	rw					r									

位 15	BL_SIZE：存储区块的大小（BLock size） 此位用于定义决定缓冲区大小的存储区块的大小。 ● 如果 BL_SIZE=0，存储区块的大小为 2 字节，因此能分配的分组缓冲区的大小范围为 2-62 个字节。 ● 如果 BL_SIZE=1，存储区块的大小为 32 字节，因此能分配的分组缓冲区的大小范围为 32-1024 字节，符合 USB 协议定义的最大分组长度限制。
位 14:10	NUM_BLOCK[4:0]：存储区块的数目（Number of blocks） 此位用以记录分配的存储区块的数目，从而决定最终使用的分组缓冲区的大小。具体请参考表 22-12。
位 9:0	COUNTx_RX[9:0]：接收到的字节数（Reception byte count） 此位由 USB 模块写入，用以记录端点收到的和 USB_EPxR 相关的最新的 OUT 或 SETUP 分组的实际字节数。

该寄存器用于存放接收分组时需要使用到的两个参数。最高的标志位定义了接收分组缓冲区的大小，以便 USB 模块检测缓冲区的溢出。低标志位则用于 USB 模块记录实际接收到的字节数。由于有效位数的限制，缓冲区的大小由分配到的存储区块数表示，而存储区块的大小则由所需的缓冲区大 小决定。缓冲区的大小在设备枚举过程中定义，由端点描述符的参数 maxPacketSize 表述。（具体信息请参考 USB 2.0 协议规范）

双缓冲区和同步 IN 端点有两个 USB_COUNTx_RX 寄存器：分别为 USB_COUNTx_RX_1 和 USB_COUNTx_RX_0，内容如下（与发送数据字节数寄存器相同，高 16 位为接收数据字节数寄存器，低 16 位为发送数据字节数寄存器）：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLSIZE_1	NUM_BLOCK_1[4:0]					COUNTx_RX_1[9:0]									
rw	rw					r									

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BLSIZE_0	NUM_BLOCK_0[4:0]					COUNTx_RX_0[9:0]									
rw	rw					r									

表 22-12 分组缓冲区大小的定义

NUM_BLOCK[4:0]的值	BL_SIZE=0 时的分组缓冲区大小	当 BL_SIZE=1 时的分组缓冲区大小
00000	不允许使用	32 字节
00001	2 字节	64 字节
00010	4 字节	96 字节
00011	6 字节	128 字节
…	…	…

NUM_BLOCK[4:0]的值	BL_SIZE=0 时的分组缓冲区大小	当 BL_SIZE=1 时的分组缓冲区大小
01111	30 字节	512 字节
0000	32 字节	保留
10001	34 字节	保留
10010	36 字节	保留
…	…	…
11110	60 字节	保留
11111	62 字节	保留

23 控制器局域网（CAN）

CAN 是基本扩展 CAN（Basic Extended CAN）的缩写，它支持 CAN 协议 2.0A 和 2.0B。它的设计目标是，以最小的 CPU 负荷来高效处理大量收到的报文。它也支持报文发送的优先级要求（优先级特性可软件配置）。

对于安全紧要的应用，CAN 提供所有支持时间触发通信模式所需的硬件功能。

23.1 CAN 主要特点

- 支持 CAN 协议 2.0A 和 2.0B 主动模式
- 波特率最高可达 1 Mbit/s
- 支持时间触发通信功能
- 发送：
  - 3 个发送邮箱
  - 发送报文的优先级特性可软件配置
  - 记录发送 SOF 时刻的时间戳
- 接收：
  - 2 个 3 级深度的接收 FIFO
  - 14 个可变的过滤器组
  - 标识符列表
  - FIFO 溢出处理方式可配置
  - 记录接收 SOF 时刻的时间戳
- 时间触发通信模式：
  - 禁止自动重传模式
  - 16 位自由运行定时器
  - 可在最后 2 个数据字节发送时间戳
- 管理：
  - 可屏蔽中断
  - 邮箱占用单独 1 块地址空间，便于提高软件效率。
- 双 CAN
  - CAN1：它负责管理在从 CAN1 和 512 字节的 SRAM 存储器之间的通信。
  - CAN2：它负责管理在从 CAN2 和独立的 512 字节的 SRAM 存储器之间的通信。
  - CAN1 和 CAN2 功能完全一样，使用各自独立的 SRAM 空间，互不干涉。

注意：USB 和 CAN1 共用一个专用的 512 字节的 SRAM 存储器用于数据的发送和接收，因此不能同时使用 USB 和 CAN1（共享的 SRAM 被 USB 和 CAN1 模块互斥地访问）。USB 和 CAN1 可以同时用于一个应用中但不能在同一个时间使用。

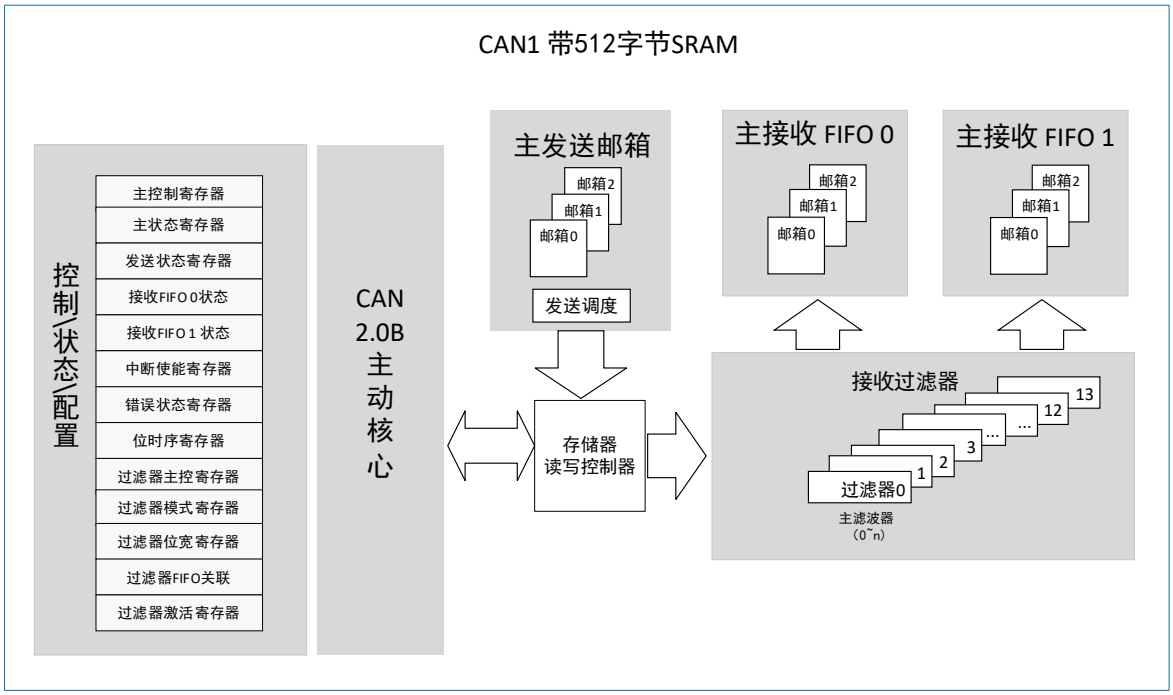


图 23-1 CAN1 框图（CAN2）

23.2 CAN 总体描述

在当今的 CAN 应用中，CAN 网络的节点在不断增加，并且多个 CAN 常常通过网关连接起来，因此整个 CAN 网中的报文数量（每个节点都需要处理）急剧增加。除了应用层报文外，网络管理和诊断报文也被引入。

- 需要一个增强的过滤机制来处理各种类型的报文。
  此外，应用层任务需要更多 CPU 时间，因此报文接收所需的实时响应程度需要减轻。
- 接收 FIFO 的方案允许 CPU 花很长时间处理应用层任务而不会丢失报文。
  构筑在底层 CAN 驱动程序上的高层协议软件，要求跟 CAN 控制器之间有高效的接口。

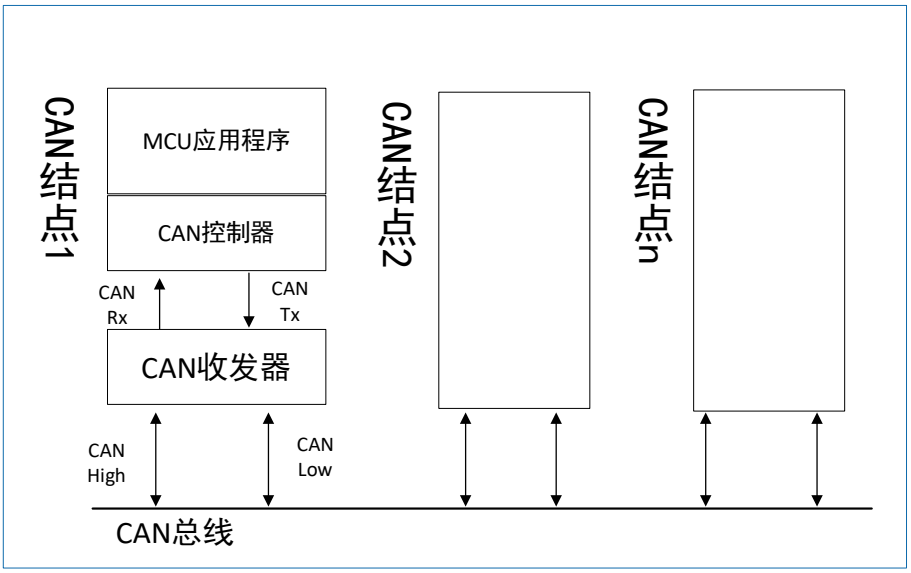


图 23-2 CAN 网拓扑结构

23.2.1 CAN 2.0B 主动内核

CAN 模块可以完全自动地接收和发送 CAN 报文；且完全支持标准标识符（11 位）和扩展标识符（29

位）。

23.2.2 控制、状态和配置寄存器

应用程序通过这些寄存器，可以：

- 配置 CAN 参数，如波特率
- 请求发送报文
- 处理报文接收
- 管理中断
- 获取诊断信息

23.2.3 发送邮箱

共有 3 个发送邮箱供软件来发送报文。发送调度器根据优先级决定哪个邮箱的报文先被发送。

23.2.4 接收过滤器

CAN 提供 14 个位宽可变/可配置的标识符过滤器组，软件通过对它们编程，从收到的报文中选择它需要的报文，而把其它报文丢弃掉。

23.2.4.1 接收 FIFO

共有 2 个接收 FIFO，每个 FIFO 都可以存放 3 个完整的报文。它们完全由硬件来管理。

23.3 CAN 工作模式

CAN 有 3 个主要的工作模式：初始化、正常和睡眠模式。

在硬件复位后，CAN 工作在睡眠模式以节省电能，同时 CANTX 引脚的内部上拉电阻被激活。软件通过对 CAN_MCR 寄存器的INRQ 或 SLEEP 位置’1’，可以请求 CAN 进入初始化或睡眠模式。一旦进入了初始化或睡眠模式，CAN 就对 CAN_MSR 寄存器的 INAK 或 SLAK 位置’1’来进行确认，同时内部上拉电阻被禁用。当 INAK 和 SLAK 位都为’0’时，CAN 就处于正常模式。在进入正常模式前，CAN 必须跟 CAN 总线取得同步；为取得同步，CAN 要等待 CAN 总线达到空闲状态，即在 CANRX 引脚上监测到 11 个连续的隐性位。

23.3.1 初始化模式

软件初始化应该在硬件处于初始化模式时进行。设置 CAN_MCR 寄存器的 INRQ 位为’1’，请求 CAN 进入初始化模式，然后等待硬件对 CAN_MSR 寄存器的 INAK 位置’1’来进行确认。

清除 CAN_MCR 寄存器的 INRQ 位为’0’，请求 CAN 退出初始化模式，当硬件对 CAN_MSR 寄存器的 INAK 位清零就确认了初始化模式的退出。

当 CAN 处于初始化模式时，禁止报文的接收和发送，并且 CANTX 引脚输出隐性位（高电平）。初始化模式的进入，不会改变配置寄存器。

软件对 CAN 的初始化，至少包括位时间特性（CAN_BTR）和控制（CAN_MCR）这 2 个寄存器。在对 CAN 的过滤器组（模式、位宽、FIFO 关联、激活和过滤器值）进行初始化前，软件要对 CAN_FMR 寄存器的 FINIT 位设置’1’。对过滤器的初始化可以在非初始化模式下进行。

注意：当 FINIT=1 时，报文的接收被禁止。可以先对过滤器激活位清零（在 CAN_FA1R 中），然后修改相应过滤器的值。如果过滤器组没有使用，那么就应该让它处于非激活状态（保持其 FACT 位为清零状态）。

23.3.2 正常模式

在初始化完成后，软件应该让硬件进入正常模式，以便正常接收和发送报文。软件可以通过对 CAN_MCR 寄存器的 INRQ 位清零，来请求从初始化模式进入正常模式，然后要等待硬件对 CAN_MSR 寄存器的 INAK 位置'1'的确认。在跟 CAN 总线取得同步，即在 CANRX 引脚上监测到 11 个连续的隐性位（等效于总线空闲）后，CAN 才能正常接收和发送报文。

不需要在初始化模式下进行过滤器初值的设置，但必须在它处在非激活状态下完成（相应的 FACT 位为 0）。而过滤器的位宽和模式的设置，则必须在初始化模式中进入正常模式前完成。

23.3.3 睡眠模式（低功耗）

CAN 可工作在低功耗的睡眠模式。软件通过对 CAN_MCR 寄存器的 SLEEP 位置'1'，来请求进入这一模式。在该模式下，CAN 的时钟停止了，但软件仍然可以访问邮箱寄存器。

当 CAN 处于睡眠模式，软件必须对 CAN_MCR 寄存器的 INRQ 位置'1'并且同时对 SLEEP 位清零，才能进入初始化模式。

有 2 种方式可以唤醒（退出睡眠模式）CAN：通过软件对 SLEEP 位清零，或硬件检测到 CAN 总线的活动。

如果 CAN_MCR 寄存器的 AWUM 位为'1'，一旦检测到 CAN 总线的活动，硬件就自动对 SLEEP 位清零来唤醒 CAN。如果 CAN_MCR 寄存器的 AWUM 位为'0'，软件必须在唤醒中断里对 SLEEP 位清零才能退出睡眠状态。

注意：如果唤醒中断被允许（CAN_IER 寄存器的 WKUIE 位为'1'），那么一旦检测到 CAN 总线活动就会产生唤醒中断，而不管硬件是否会自动唤醒 CAN。

在对 SLEEP 位清零后，睡眠模式的退出必须与 CAN 总线同步，请参考下图。当硬件对 SLAK 位清零时，就确认了睡眠模式的退出。

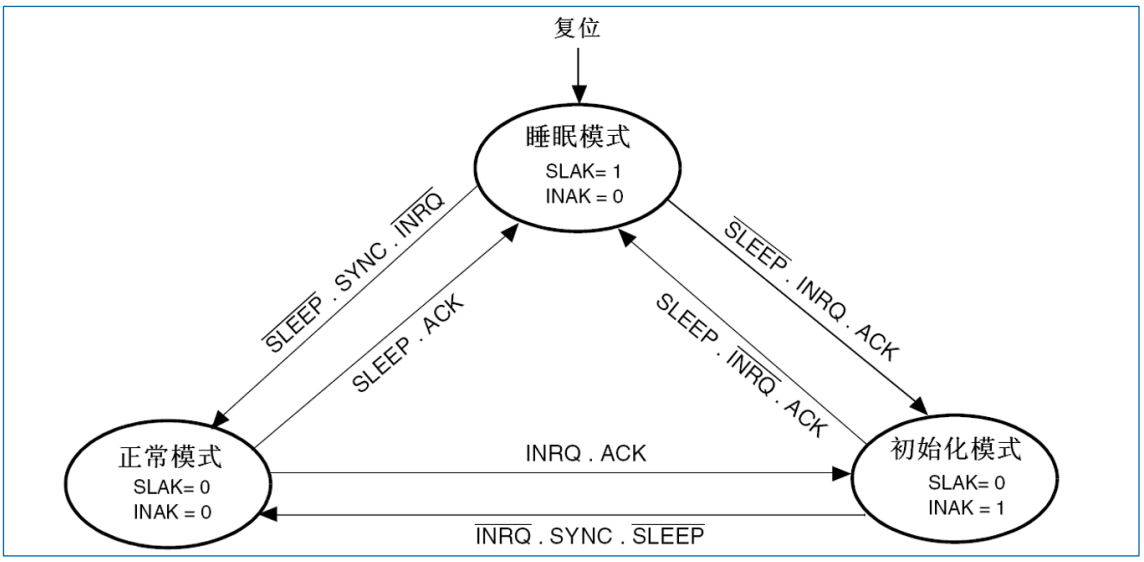


图 23-3 CAN 工作模式

注意：

ACK =硬件响应睡眠或初始化请求,而对 CAN_MSR 寄存器的 INAK 或 SLAK 位置 1 的状态。

SYNC=CAN 等待 CAN 总线变为空闲的状态，即在 CANRX 引脚上检测到连续的 11 个隐性位。

23.4 测试模式

通过对 CAN_BTR 寄存器的 SILM 和/或 LBKM 位置'1'，来选择一种测试模式。只能在初始化模式下，

修改这 2 位。在选择了一种测试模式后，软件需要对 CAN_MCR 寄存器的 INRQ 位清零，来真正进入测试模式。

23.4.1 静默模式

通过对 CAN_BTR 寄存器的 SILM 位置’1’，来选择静默模式。

在静默模式下，CAN 可以正常地接收数据帧和远程帧，但只能发出隐性位，而不能真正发送报文。如果 CAN 需要发出显性位（确认位、过载标志、主动错误标志），那么这样的显性位在内部被接回来从而可以被 CAN 内核检测到，同时 CAN 总线不会受到影响而仍然维持在隐性位状态。因此，静默模式通常用于分析 CAN 总线的活动，而不会对总线造成影响－显性位（确认位、错误帧）不会真正发送到总线上。

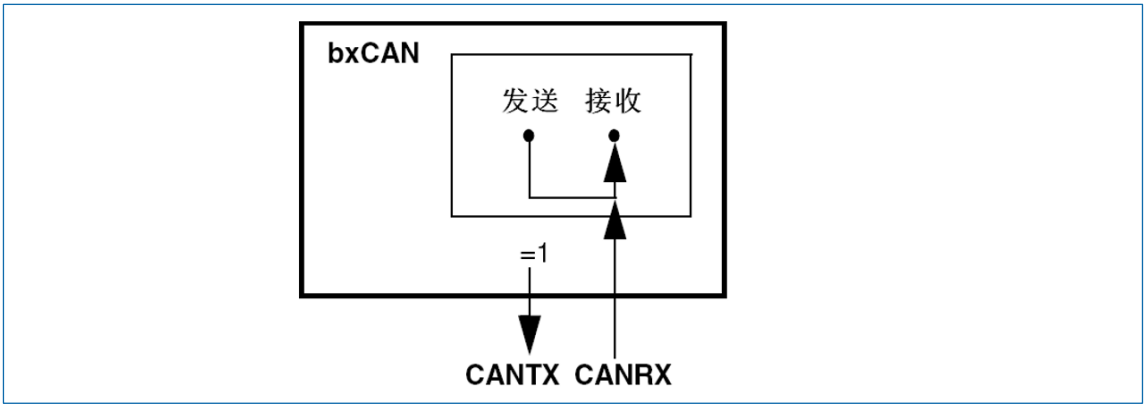


图 23-4 CAN 工作在静默模式

23.4.2 环回模式

通过对 CAN_BTR 寄存器的 LBKM 位置’1’，来选择环回模式。在环回模式下，CAN 把发送的报文当作接收的报文并保存（如果可以通过接收过滤）在接收邮箱里。

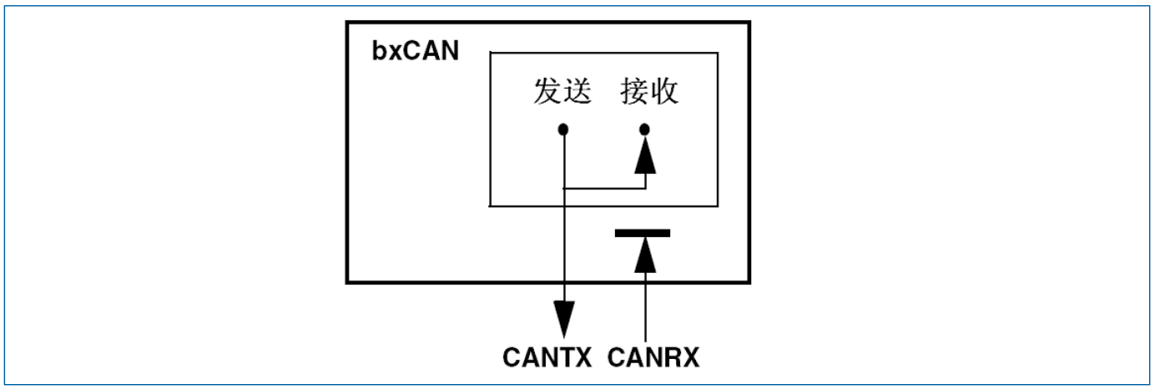


图 23-5 CAN 工作在环回模式

环回模式可用于自测试。为了避免外部的影响，在环回模式下 CAN 内核忽略确认错误（在数据/远程帧的确认位时刻，不检测是否有显性位）。在环回模式下，CAN 在内部把 Tx 输出回馈到 Rx 输入上，而完全忽略 CANRX 引脚的实际状态。发送的报文可以在 CANTX 引脚上检测到。

23.4.3 环回静默模式

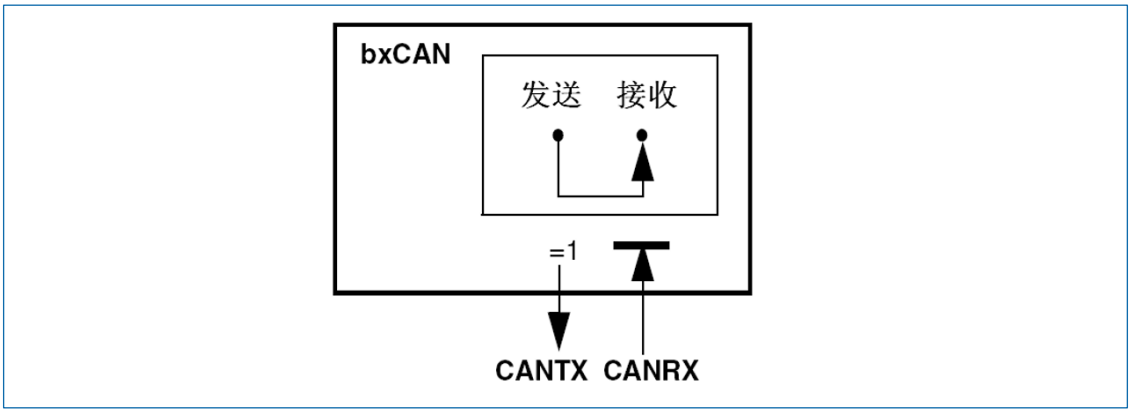


图 23-6 CAN 工作在环回静默模式

通过对 CAN_BTR 寄存器的 LBKM 和 SILM 位同时置'1'，可以选择环回静默模式。该模式可用于"热自测试"，即可以像环回模式那样测试 CAN，但却不会影响 CANTX 和 CANRX 所连接的整个 CAN 系统。在环回静默模式下，CANRX 引脚与 CAN 总线断开，同时 CANTX 引脚被驱动到隐性位状态。

23.5 处于调试模式时

当微控制器处于调试模式时，Cortex-M3 核心处于暂停状态，依据下述配置位的状态，CAN 可以继续正常工作或停止工作：

- 调试（DBG）模块中 CAN1 的 DBG_CAN1_STOP 位或 CAN2 的 DBG_CAN2_STOP 位。详见章节："36.15.2 支持定时器、看门狗、CAN 和 I2C 的调试"。
- CAN_MCR 中的 DBF 位。详见章节："23.8.2 CAN 控制和状态寄存器"。

23.6 CAN 功能描述

23.6.1 发送处理

发送报文的流程为：

1. 应用程序选择 1 个空置的发送邮箱；
2. 设置标识符，数据长度和待发送数据；
3. 对 CAN_TIxR 寄存器的 TXRQ 位置'1'，来请求发送。TXRQ 位置'1'后，邮箱就不再是空邮箱；而一旦邮箱不再为空置，软件对邮箱寄存器就不再有写的权限。TXRQ 位置 1 后，邮箱马上进入挂号状态，并等待成为最高优先级的邮箱，参见："23.6.1.1 发送优先级"。
4. 一旦邮箱成为最高优先级的邮箱，其状态就变为预定发送状态。一旦 CAN 总线进入空闲状态，预定发送邮箱中的报文就马上被发送（进入发送状态）。
5. 一旦邮箱中的报文被成功发送后，它马上变为空置邮箱；硬件相应地对 CAN_TSR 寄存器的 RQCP 和 TXOK 位置 1，来表明一次成功发送。如果发送失败，由于仲裁引起的就对 CAN_TSR 寄存器的 ALST 位置'1'，由于发送错误引起的就对 TERR 位置'1'。

23.6.1.1 发送优先级

- 由标识符决定：

当有超过 1 个发送邮箱在挂号时，发送顺序由邮箱中报文的标识符决定。根据 CAN 协议，标识符数值最低的报文具有最高的优先级。如果标识符的值相等，那么邮箱号小的报文先被发送。

- 由发送请求次序决定：

通过对 CAN_MCR 寄存器的 TXFP 位置’1’，可以把发送邮箱配置为发送 FIFO。在该模式下，发送的优先级由发送请求次序决定。该模式对分段发送很有用。

23.6.1.2 中止

通过对 CAN_TSR 寄存器的 ABRQ 位置’1’，可以中止发送请求。

- 邮箱如果处于挂号或预定状态，发送请求马上就被中止了。
- 如果邮箱处于发送状态，那么中止请求可能导致两种结果：
  - 如果邮箱中的报文被成功发送，那么邮箱变为空置邮箱，并且 CAN_TSR 寄存器的 TXOK 位被硬件置’1’。
  - 如果邮箱中的报文发送失败了，那么邮箱变为预定状态，然后发送请求被中止，邮箱变为空置邮箱且 TXOK 位被硬件清零。

因此如果邮箱处于发送状态，那么在发送操作结束后，邮箱都会变为空置邮箱。

23.6.1.3 禁止自动重传模式

该模式主要用于满足 CAN 标准中，时间触发通信选项的需求。通过对 CAN_MCR 寄存器的 NART 位置’1’，来让硬件工作在该模式。

在该模式下，发送操作只会执行一次。如果发送操作失败了，不管是由于仲裁丢失或出错，硬件都不会再自动发送该报文。

在一次发送操作结束后，硬件认为发送请求已经完成，从而对 CAN_TSR 寄存器的 RQCP 位置’1’，同时发送的结果反映在 TXOK、ALST 和 TERR 位上。

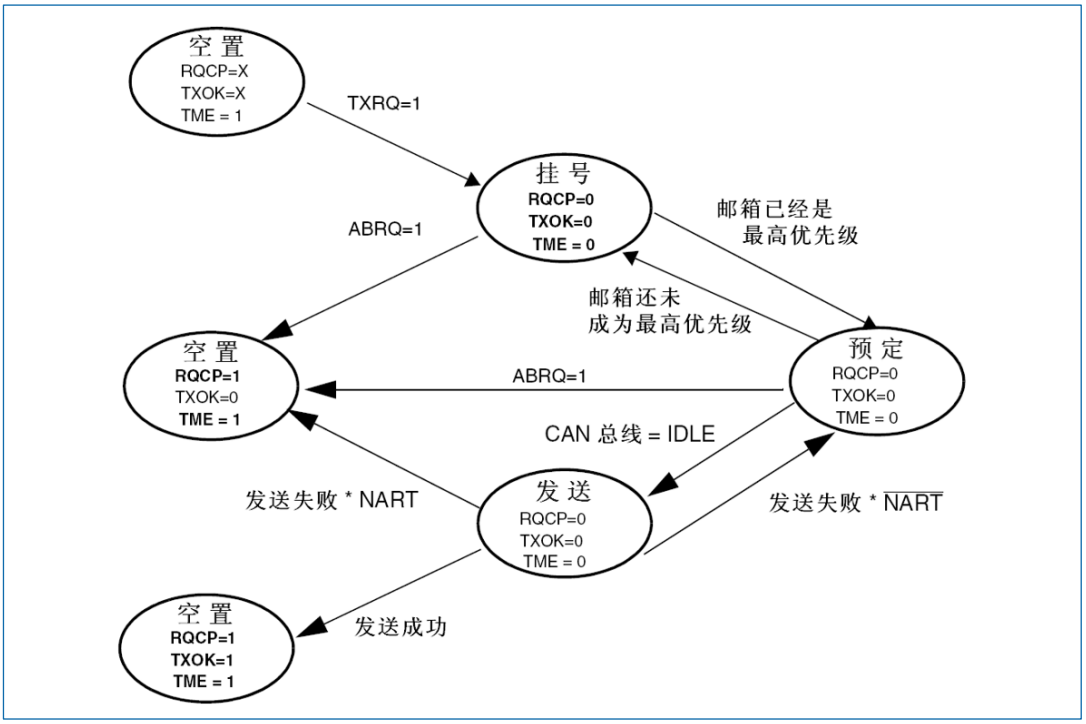


图 23-7 发送邮箱状态

23.6.2 时间触发通信模式

在该模式下，CAN 硬件的内部定时器被激活，并且被用于产生（发送与接收邮箱的）时间戳，分别存储在 CAN_RDTxR/CAN_TDTxR 寄存器中。内部定时器在每个 CAN 位时间累加（见章节：“23.6.7 位时间特性”）。内部定时器在接收和发送的帧起始位的采样点位置被采样，并生成时间戳。

23.6.3 接收管理

接收到的报文，被存储在 3 级邮箱深度的 FIFO 中。FIFO 完全由硬件来管理，从而节省了 CPU 的处理负荷，简化了软件并保证了数据的一致性。应用程序只能通过读取 FIFO 输出邮箱，来读取FIFO 中最先收到的报文。

23.6.3.1 有效报文

根据 CAN 协议，当报文被正确接收（直到 EOF 域的最后一位都没有错误），且通过了标识符过滤，那么该报文被认为是有效报文。请参考章节：“23.6.4 标识符过滤”。

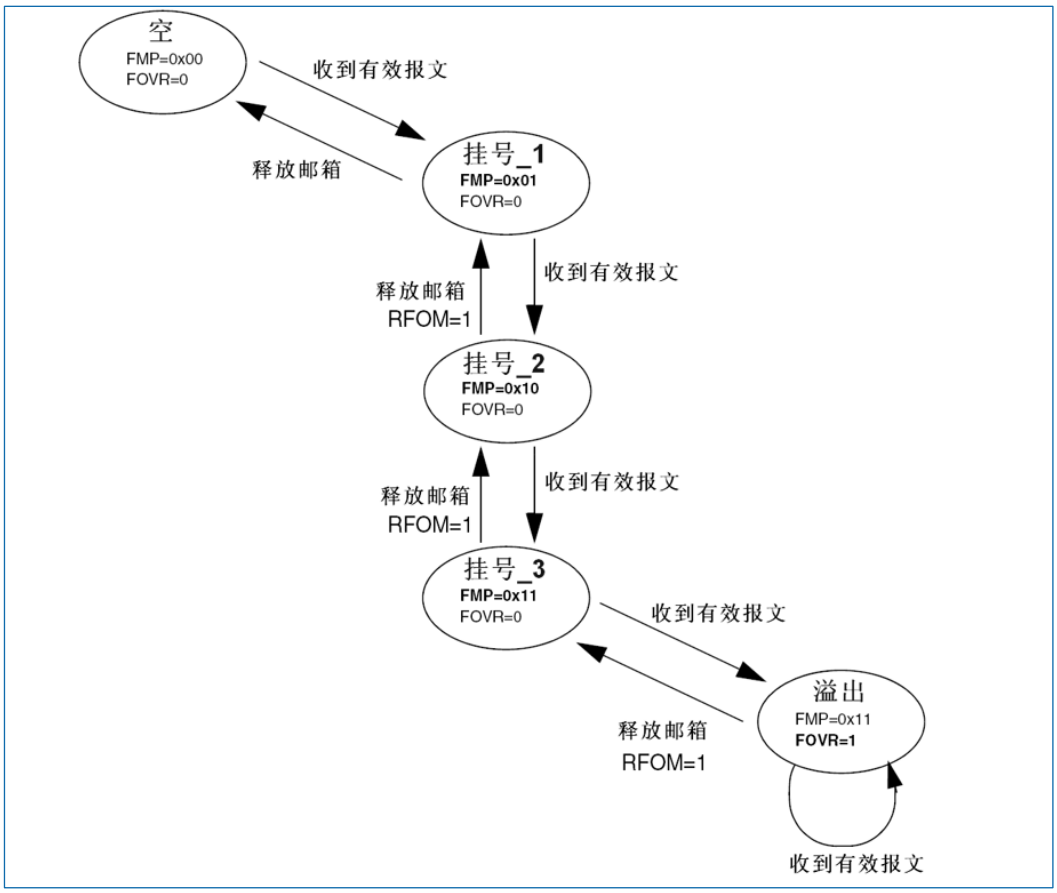


图 23-8 接收 FIFO 状态

23.6.3.2 FIFO 管理

FIFO 从空状态开始，在接收到第一个有效的报文后，FIFO 状态变为“挂号_1”（pending_1），硬件相应地把 CAN_RFR 寄存器的 FMP[1:0]设置为’01’（二进制 01b）。软件可以读取 FIFO 输出邮箱来读出邮箱中的报文，然后通过对 CAN_RFR 寄存器的 RFOM 位设置’1’来释放邮箱，这样 FIFO 又变为空状态了。如果在释放邮箱的同时，又收到了一个有效的报文，那么 FIFO 仍然保留在“挂号_1”状态，软件可以读取 FIFO 输出邮箱来读出新收到的报文。

如果应用程序不释放邮箱，在接收到下一个有效的报文后，FIFO 状态变为“挂号_2”（pending_2），硬件相应地把 FMP[1:0]设置为’10’（二进制 10b）。重复上面的过程，第三个有效的报文把 FIFO 变为“挂号_3”状态（FMP[1:0]＝11b）。此时，软件必须对 RFOM 位设置 1 来释放邮箱，以便 FIFO 可以有空间来存放下一个有效的报文；否则，下一个有效的报文到来时就会导致一个报文的丢失。参见：“23.6.5 报文存储”。

23.6.3.3 溢出

当 FIFO 处于“挂号_3”状态（即 FIFO 的 3 个邮箱都是满的），下一个有效的报文就会导致溢出，并且一个报文会丢失。此时，硬件对 CAN_RFR 寄存器的 FOVR 位进行置’1’来表明溢出情况。至于哪个报文会被丢弃，取决于对 FIFO 的设置：

- 如果禁用了 FIFO 锁定功能（CAN_MCR 寄存器的 RFLM 位被清零），那么 FIFO 中最后收到的报文就被新报文所覆盖。这样，最新收到的报文不会被丢弃掉。
- 如果启用了 FIFO 锁定功能（CAN_MCR 寄存器的 RFLM 位被置’1’），那么新收到的报文就被丢弃，软件可以读到 FIFO 中最早收到的 3 个报文。

23.6.3.4 接收相关的中断

一旦往 FIFO 存入一个报文，硬件就会更新 FMP[1:0]位，并且如果 CAN_IER 寄存器的 FMPIE 位为’1’，那么就会产生一个中断请求。

当 FIFO 变满时（即第 3 个报文被存入），CAN_RFR 寄存器的 FULL 位就被置’1’，并且如果 CAN_IER 寄存器的 FFIE 位为’1’，那么就会产生一个满中断请求。

在溢出的情况下，FOVR 位被置’1’，并且如果 CAN_IER 寄存器的 FOVIE 位为’1’，那么就会产生一个溢出中断请求。

23.6.4 标识符过滤

在 CAN 协议里，报文的标识符不代表节点的地址，而是跟报文的内容相关的。因此，发送者以广播的形式把报文发送给所有的接收者。节点在接收报文时，根据标识符的值决定软件是否需要该报文；如果需要，就拷贝到 SRAM 里；如果不需要，报文就被丢弃且无需软件的干预。

为满足这一需求，CAN 控制器为应用程序提供了 14 个位宽可变的、可配置的过滤器组（13~0），以便只接收那些软件需要的报文。硬件过滤的做法节省了 CPU 开销，否则就必须由软件过滤从而占用一定的 CPU 开销。每个过滤器组 x 由 2 个 32 位寄存器，CAN_FiR1 和 CAN_FiR2 组成。

23.6.4.1 可变的位宽

每个过滤器组的位宽都可以独立配置，以满足应用程序的不同需求。根据位宽的不同，每个过滤器组可提供：

- 1 个 32 位过滤器，包括：STID[10:0]、EXID[17:0]、IDE 和 RTR 位
- 2 个 16 位过滤器，包括：STID[10:0]、IDE、RTR 和 EXID[17:15]位

更多信息，可参见图 23-9。此外过滤器可配置为，屏蔽位模式和标识符列表模式。

23.6.4.2 屏蔽位模式

在屏蔽位模式下，标识符寄存器和屏蔽寄存器一起，指定报文标识符的任何一位，应该按照“必须匹配”或“不用关心”处理。

23.6.4.3 标识符列表模式

在标识符列表模式下，屏蔽寄存器也被当作标识符寄存器用。因此，不是采用一个标识符加一个屏蔽位的方式，而是使用 2 个标识符寄存器。接收报文标识符的每一位都必须跟过滤器标识符相同。

23.6.4.4 过滤器组位宽和模式的设置

过滤器组可以通过相应的 CAN_FM1R 寄存器配置。在配置一个过滤器组前，必须通过清除 CAN_FA1R 寄存器的 FACTx 位，把它设置为禁用状态。通过设置 CAN_FS1R 的相应 FSCx 位，可以配置一

个过滤器组的位宽，请参见下图。通过 CAN_FM1R 的 FBMx 位，可以配置对应的屏蔽/标识符寄存器的标识符列表模式或屏蔽位模式。

为了过滤出一组标识符，应该设置过滤器组工作在屏蔽位模式。

为了过滤出一个标识符，应该设置过滤器组工作在标识符列表模式。

应用程序不用的过滤器组，应该保持在禁用状态。

过滤器组中的每个过滤器，都被编号（过滤器号）。编号从 0 到某个最大数值（最大值取决于过滤器组的模式和位宽的设置）。

关于过滤器配置，参见下图。

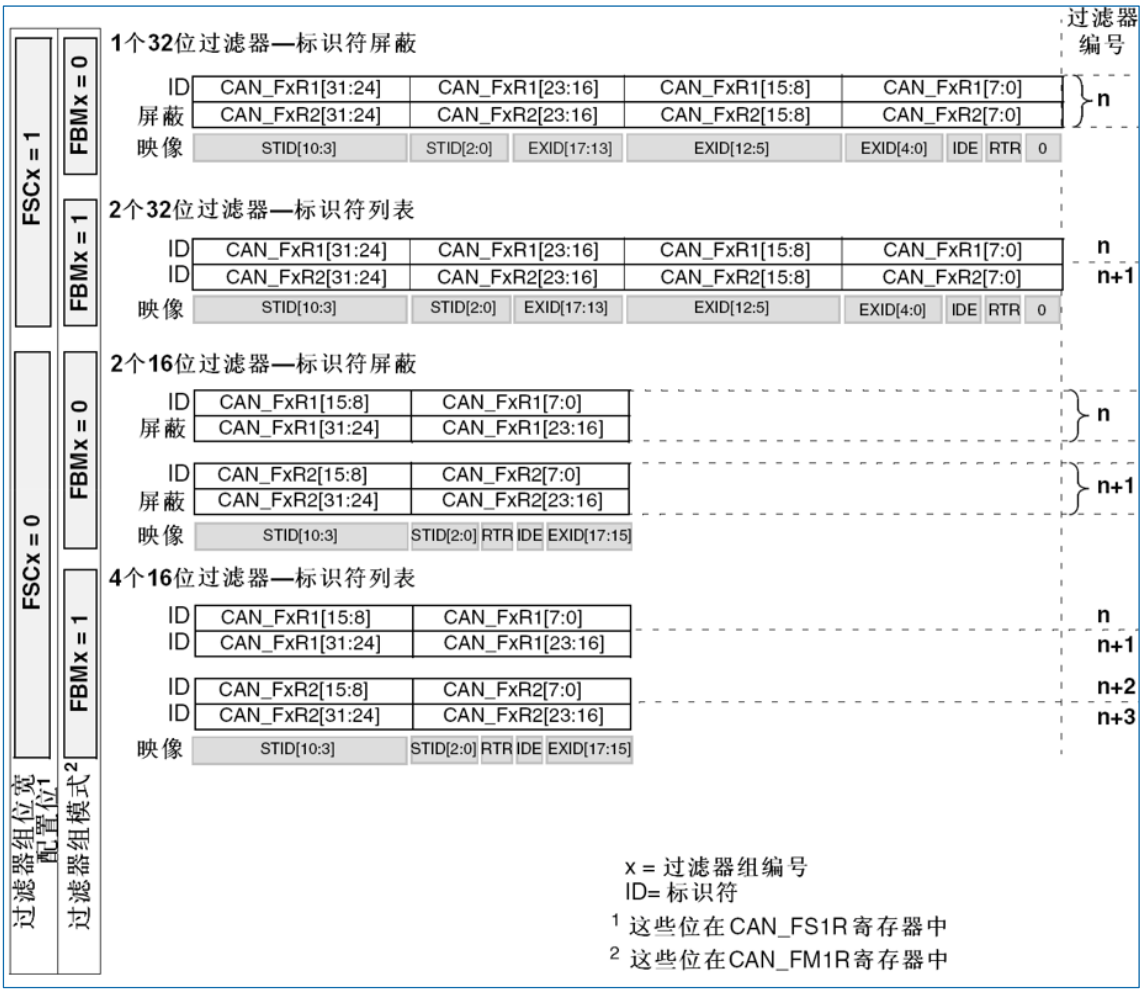


图 23-9 过滤器组位宽设置－寄存器组织

23.6.4.5 过滤器匹配序号

一旦收到的报文被存入 FIFO，就可被应用程序访问。通常情况下，报文中的数据被拷贝到SRAM 中；为了把数据拷贝到合适的位置，应用程序需要根据报文的标识符来辨别不同的数据。CAN 提供了过滤器匹配序号，以简化这一辨别过程。

根据过滤器优先级规则，过滤器匹配序号和报文一起，被存入邮箱中。因此每个收到的报文，都有与它相关联的过滤器匹配序号。

过滤器匹配序号可以通过下面两种方式来使用：

- 把过滤器匹配序号跟一系列所期望的值进行比较。
- 把过滤器匹配序号当作一个索引来访问目标地址。

对于标识符列表模式下的过滤器（非屏蔽方式的过滤器），软件不需要直接跟标识符进行比较。

对于屏蔽位模式下的过滤器，软件只须对需要的那些屏蔽位（必须匹配的位）进行比较即可。在给过滤器编号时，并不考虑过滤器组是否为激活状态。另外，每个 FIFO 各自对其关联的过滤器进行编号。请参考下图的例子。

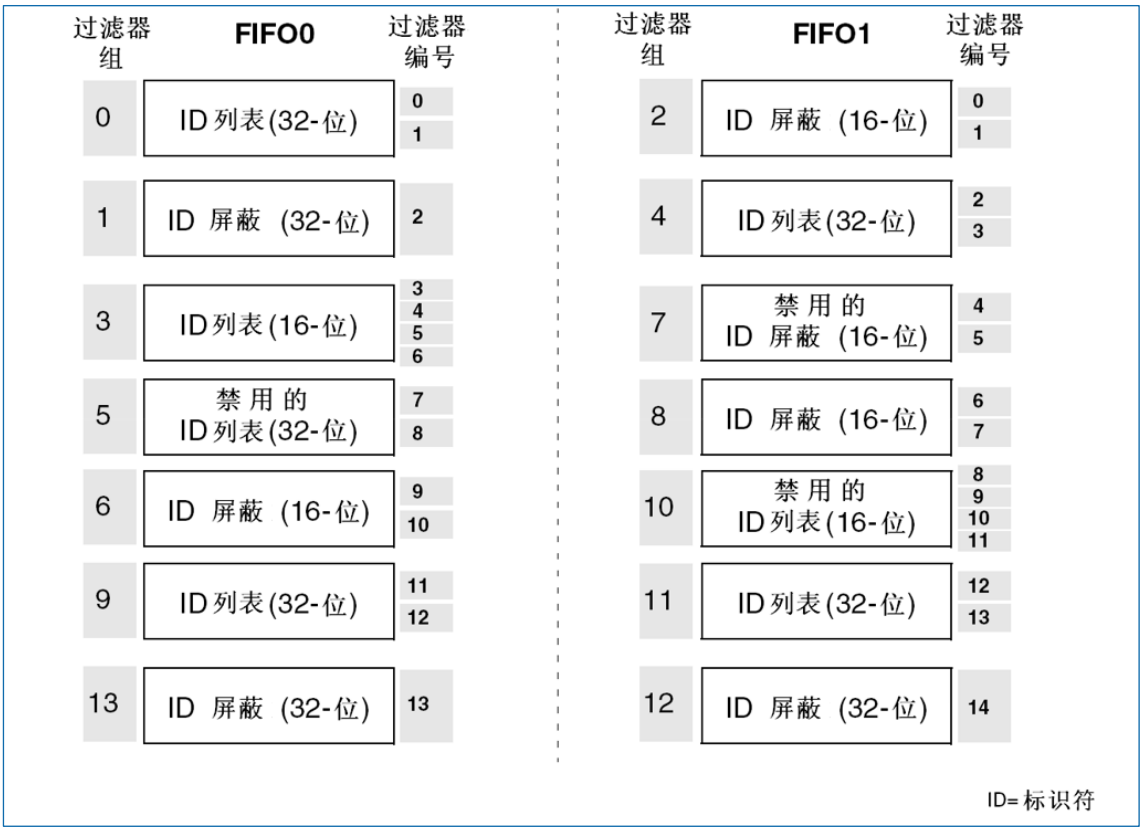


图 23-10 过滤器编号的例子

23.6.4.6 过滤器优先级规则

根据过滤器的不同配置，有可能一个报文标识符能通过多个过滤器的过滤；在这种情况下，存放在接收邮箱中的过滤器匹配序号，根据下列优先级规则来确定：

- 位宽为 32 位的过滤器，优先级高于位宽为 16 位的过滤器。
- 对于位宽相同的过滤器，标识符列表模式的优先级高于屏蔽位模式。
- 位宽和模式都相同的过滤器，优先级由过滤器号决定，过滤器号小的优先级高。

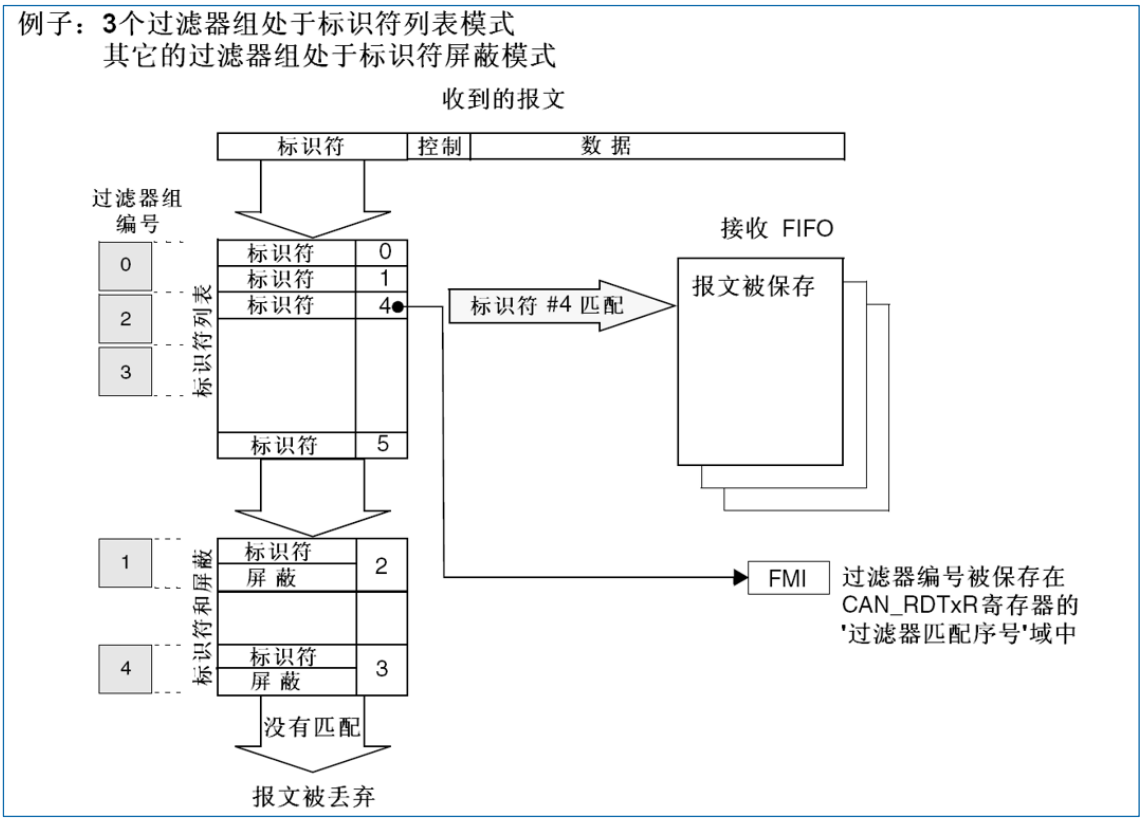


图 23-11 过滤器机制的例子

上面的例子说明了 CAN 的过滤器规则：在接收一个报文时，其标识符首先与配置在标识符列表模式下的过滤器相比较；如果匹配上，报文就被存放到相关联的 FIFO 中，并且所匹配的过滤器的序号被存入过滤器匹配序号中。如同例子中所显示，报文标识符跟#4 标识符匹配，因此报文内容和 FMI4 被存入 FIFO。

如果没有匹配，报文标识符接着与配置在屏蔽位模式下的过滤器进行比较。

如果报文标识符没有跟过滤器中的任何标识符相匹配，那么硬件就丢弃该报文，且不会对软件有任何打扰。

23.6.5 报文存储

邮箱是软件和硬件之间传递报文的接口。邮箱包含了所有跟报文有关的信息：标识符、数据、控制、状态和时间戳信息。

23.6.5.1 发送邮箱

软件需要在一个空的发送邮箱中，把待发送报文的各种信息设置好（然后再发出发送的请求）。发送的状态可通过查询 CAN_TSR 寄存器获知。

表 23-1 发送邮箱寄存器列表

相对发送邮箱基地址的偏移量	寄存器名
0	CAN_TIxR
4	CAN_TDTxR
8	CAN_TDLxR
12	CAN_TDHxR

23.6.5.2 接收邮箱（FIFO）

在接收到一个报文后，软件就可以访问接收 FIFO 的输出邮箱来读取它。一旦软件处理了报文（如把它读出来），软件就应该对 CAN_RFxR 寄存器的 RFOM 位进行置'1'，来释放该报文，以便为后面收到的报文留出存储空间。过滤器匹配序号存放在 CAN_RDTxR 寄存器的 FMI 域中。16 位的时间戳存放在 CAN_RDTxR 寄存器的 TIME[15:0]域中。

表 23-2 接收邮箱寄存器列表

相对接收邮箱基地址的偏移量	寄存器名
0	CAN_RIxR
4	CAN_RDTxR
8	CAN_RDLxR
12	CAN_RDHxR

23.6.6 出错管理

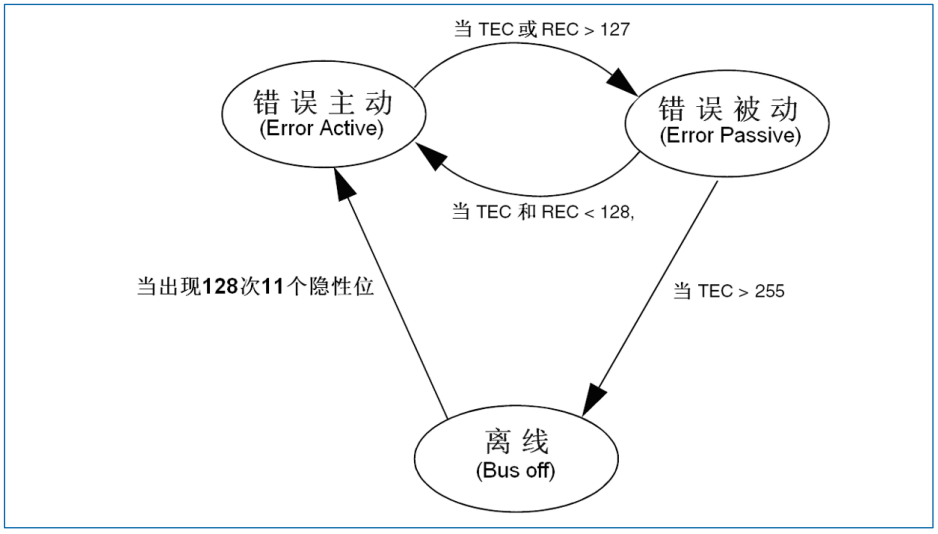


图 23-12 CAN 错误状态图

CAN 协议描述的出错管理，完全由硬件通过发送错误计数器（CAN_ESR 寄存器里的 TEC 域），和接收错误计数器（CAN_ESR 寄存器里的 REC 域）来实现，其值根据错误的情况而增加或减少。关于 TEC 和 REC 管理的详细信息，请参考 CAN 标准。

软件可以读出它们的值来判断 CAN 网络的稳定性。

此外，CAN_ESR 寄存器提供了当前错误状态的详细信息。通过设置 CAN_IER 寄存器（比如 ERRIE 位），当检测到出错时软件可以灵活地控制中断的产生。

23.6.6.1 离线恢复

当 TEC 大于 255 时，CAN 就进入离线状态，同时 CAN_ESR 寄存器的 BOFF 位被置'1'。在离线状态下，CAN 无法接收和发送报文。

根据 CAN_MCR 寄存器中 ABOM 位的设置，CAN 可以自动或在软件的请求下，从离线状态恢复（变为错误主动状态）。在这两种情况下，CAN 都必须等待一个 CAN 标准所描述的恢复过程（CAN RX 引脚上检测到 128 次 11 个连续的隐性位）。

- 如果 ABOM 位为'1'，CAN 进入离线状态后，就自动开启恢复过程。

- 如果 ABOM 位为'0'，软件必须先请求 CAN 进入然后再退出初始化模式，随后恢复过程才被开启。

注意：在初始化模式下，CAN 不会监视 CAN RX 引脚的状态，这样就不能完成恢复过程。为了完成恢复过程，CAN 必须工作在正常模式。

23.6.7 位时间特性

位时间特性逻辑通过采样来监视串行的 CAN 总线，并且通过与帧起始位的边沿进行同步，及通过与后面的边沿进行重新同步，来调整其采样点。

它的操作可以简单解释为，如下所述把名义上的每位时间分为 3 段：

- 同步段（SYNC_SEG）：通常期望位的变化发生在该时间段内。其值固定为 1 个时间单元（1x t_{CAN}）。
- 时间段 1（BS1）：定义采样点的位置。它包含 CAN 标准里的 PROP_SEG 和 PHASE_SEG1。其值可以编程为 1 到 16 个时间单元，但也可以被自动延长，以补偿因为网络中不同节点的频率差异所造成的相位的正向漂移。
- 时间段 2（BS2）：定义发送点的位置。它代表 CAN 标准里的 PHASE_SEG2。其值可以编程为 1 到 8 个时间单元，但也可以被自动缩短以补偿相位的负向漂移。

重新同步跳跃宽度（SJW）定义了在每位中可以延长或缩短多少个时间单元的上限。其值可以编程为 1 到 4 个时间单元。

有效跳变被定义为当 CAN 自己没有发送隐性位时，从显性位到隐性位的第 1 次转变。

如果在时间段 1（BS1）而不是在同步段（SYNC_SEG）检测到有效跳变，那么 BS1 的时间就被延长最多 SJW 那么长，从而采样点被延迟了。

相反如果在时间段 2（BS2）而不是在 SYNC_SEG 检测到有效跳变，那么 BS2 的时间就被缩短最多 SJW 那么长，从而采样点被提前了。

为了避免软件的编程错误，对位时间特性寄存器（CAN_BTR）的设置，只能在 CAN 处于初始化状态下进行。

注意：关于 CAN 位时间特性和重同步机制的详细信息，请参考 ISO11898 标准。

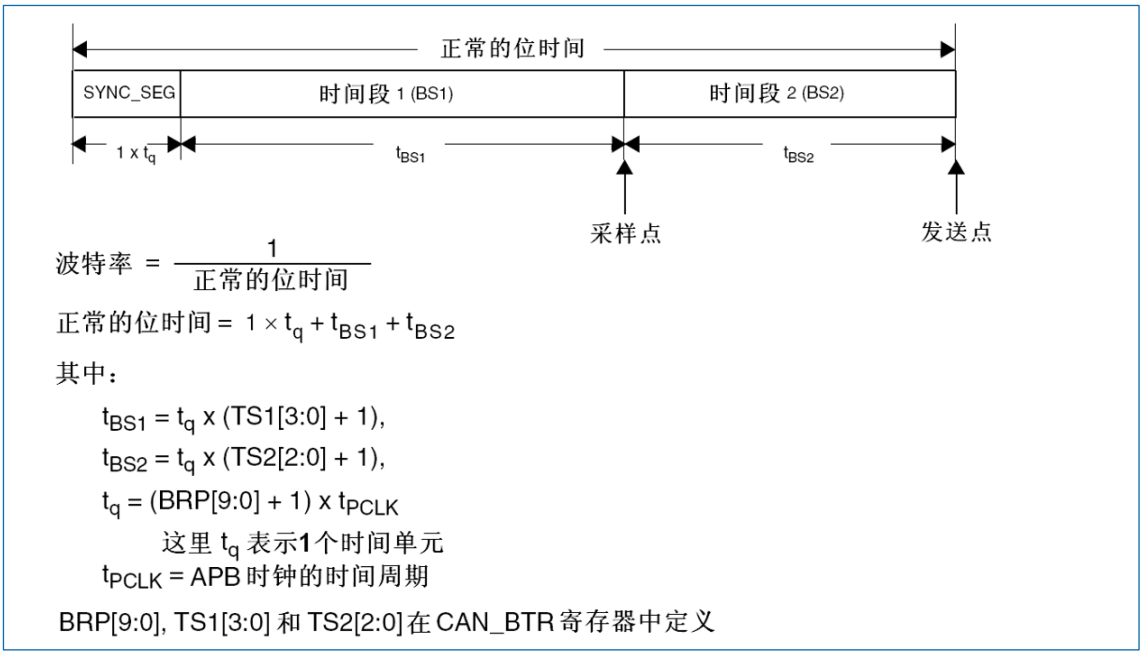


图 23-13 位时序

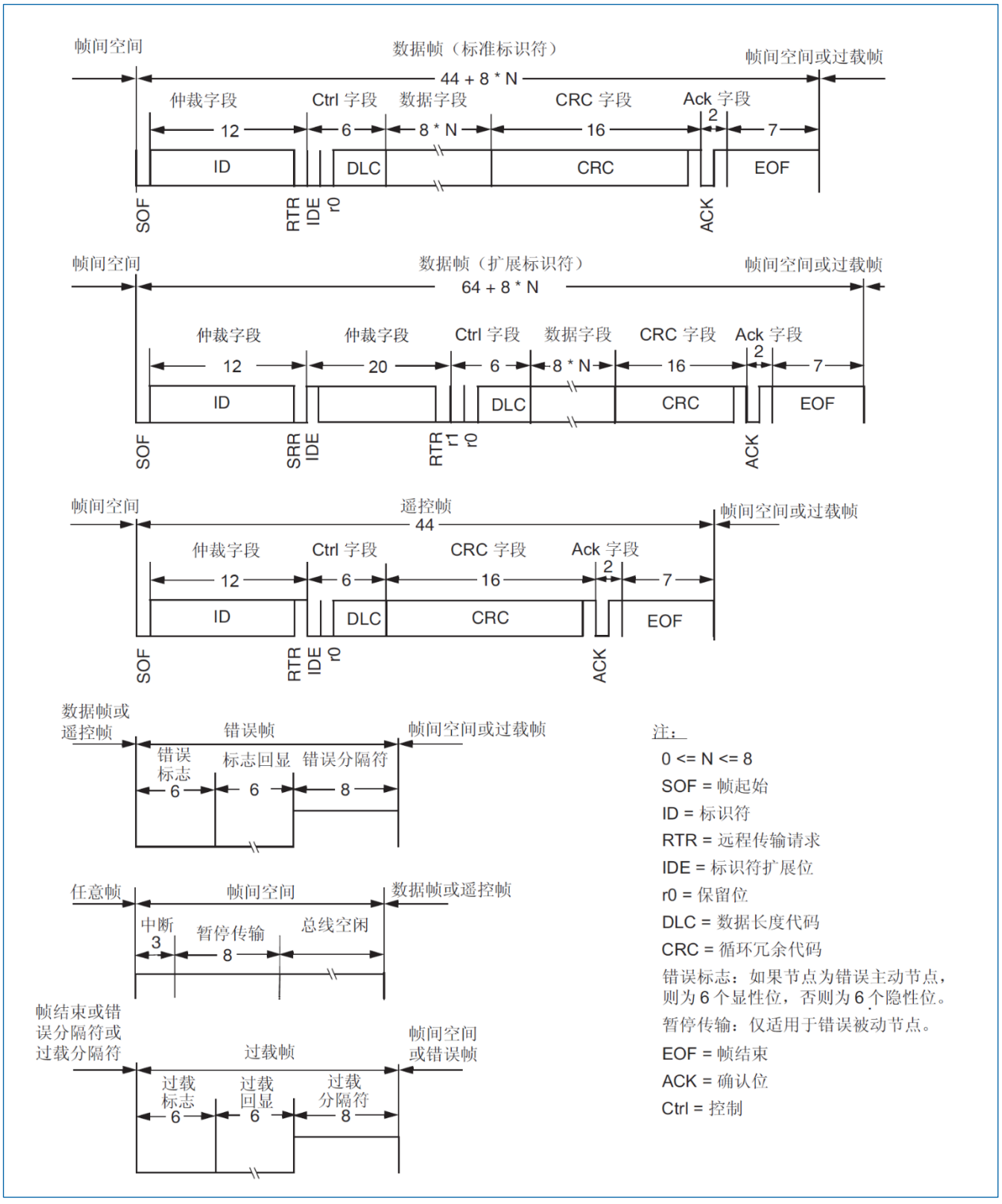


图 23-14 各种 CAN 帧

23.7 CAN 中断

CAN 占用 4 个专用的中断向量。通过设置 CAN 中断允许寄存器（CAN_IER），每个中断源都可以单独允许和禁用。

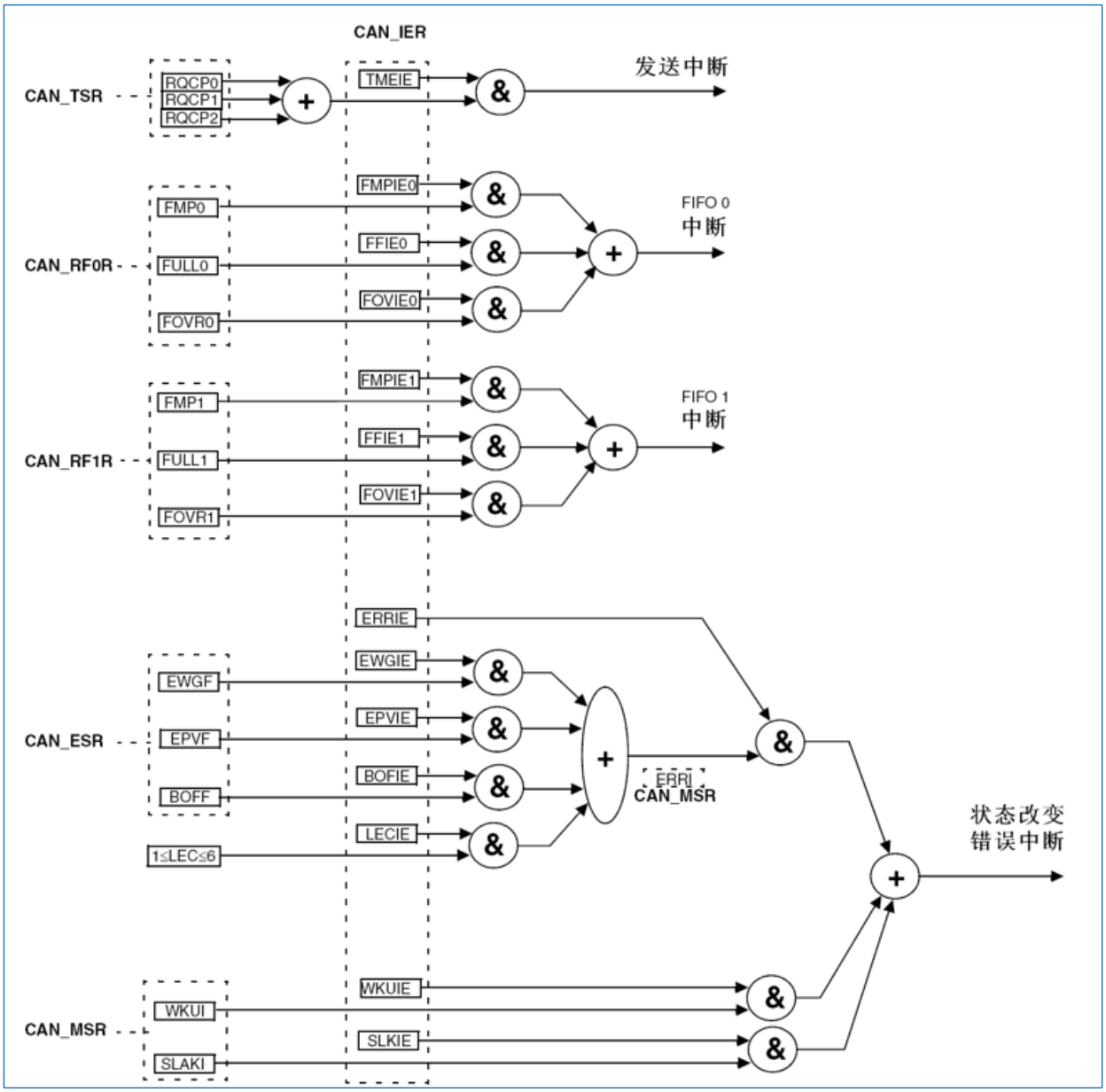


图 23-15 事件标志和中断产生

- 发送中断可由下列事件产生：
  - 发送邮箱 0 变为空，CAN_TSR 寄存器的 RQCP0 位被置’1’。
  - 发送邮箱 1 变为空，CAN_TSR 寄存器的 RQCP1 位被置’1’。
  - 发送邮箱 2 变为空，CAN_TSR 寄存器的 RQCP2 位被置’1’。
- FIFO0 中断可由下列事件产生：
  - FIFO0 接收到一个新报文，CAN_RF0R 寄存器的 FMP0 位不再是’00’。
  - FIFO0 变为满的情况，CAN_RF0R 寄存器的 FULL0 位被置’1’。
  - FIFO0 发生溢出的情况，CAN_RF0R 寄存器的 FOVR0 位被置’1’。
- FIFO1 中断可由下列事件产生：
  - FIFO1 接收到一个新报文，CAN_RF1R 寄存器的 FMP1 位不再是’00’。
  - FIFO1 变为满的情况，CAN_RF1R 寄存器的 FULL1 位被置’1’。
  - FIFO1 发生溢出的情况，CAN_RF1R 寄存器的 FOVR1 位被置’1’。
- 错误和状态变化中断可由下列事件产生：
  - 出错情况，关于出错情况的详细信息请参考 CAN 错误状态寄存器（CAN_ESR）。

- 唤醒情况，在 CAN 接收引脚上监视到帧起始位（SOF）。
- CAN 进入睡眠模式。

23.8 CAN 寄存器

基地址：（CAN1, CAN2）=（0x4000 6400, 0x4000 6800）

空间大小：（CAN1, CAN2）=（0x400, 0x400）

必须以字（32 位）的方式操作这些外设寄存器。

23.8.1 寄存器访问保护

对某些寄存器的错误访问会导致一个 CAN 节点对整个 CAN 网络的暂时性干扰。因此，软件只能在 CAN 处于初始化模式时修改 CAN_BTR 寄存器。

虽然错误数据的发送对 CAN 网的网络层不会带来问题，但却会对应用程序造成严重影响。因此，软件只能在发送邮箱为空的状态改变它，参见图 23-7 。

过滤器的数值只能在关闭对应过滤器组的状态下，或设置 FINIT 位为'1'后才能修改。此外，只有在设置整个过滤器为初始化模式下（即 FINIT=1），才能修改过滤器的设置，即修改 CAN_FM1R，CAN_FS1R 和 CAN_FFA1R 寄存器。

23.8.2 CAN 控制和状态寄存器

23.8.2.1 CAN 主控制寄存器（CAN_MCR）

偏移地址：0x00

复位值：0x0001 0002

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESET	Res							TTCM	ABOM	AWUM	NART	RFLM	TXFP	SLEEP	INRQ
rs								rw	rw	rw	rw	rw	rw	rw	rw

位 31:17	Res：保留 必须保持复位值。
位 16	DBF：调试冻结（Debug freeze） • 0：在调试时，CAN 照常工作。 • 1：在调试时，冻结 CAN 的接收/发送。仍然可以正常地读写和控制接收 FIFO。
位 15	RESET：CAN 软件复位（CAN software master） • 0：本外设正常工作。 • 1：对 CAN 进行强行复位。 复位后 CAN 进入睡眠模式（FMP 位和 CAN_MCR 寄存器被初始化为其复位值）。 此后硬件自动对该位清零。
位 14:8	Res：保留 必须保持复位值。
位 7	TTCM：时间触发通信模式（Time triggered communication mode） • 0：禁止时间触发通信模式。

	• 1：允许时间触发通信模式。 *注意：要想了解关于时间触发通信模式的更多信息，请参考章节：“23.6.2 时间触发通信模式”。*
位 6	ABOM：自动离线管理（Automatic bus-off management） 该位决定 CAN 硬件在什么条件下可以退出离线状态。 • 0：软件对 CAN_MCR 寄存器的 INRQ 位进行置’1’随后清零后，一旦硬件检测到 128 次 11 位连续的隐性位，则退出离线状态。 • 1：硬件检测到 128 次 11 位连续的隐性位，则自动退出离线状态。 *注意：关于离线状态的更多信息，请参考章节：“图 23-7 发送邮箱状态”。*
位 5	AWUM：自动唤醒模式（Automatic wakeup mode） 该位决定 CAN 处在睡眠模式时由硬件还是软件唤醒。 • 0：软件通过清除 CAN_MCR 寄存器的 SLEEP 位，将 CAN 从睡眠模式中唤醒。 • 1：硬件通过检测 CAN 报文，将 CAN 从睡眠模式中自动唤醒。 唤醒的同时，硬件自动对 CAN_MCR 的 SLEEP 位和 CAN_MSR 寄存器的 SLEEP 和 SLAK 位清零。
位 4	NART：禁止报文自动重发（No automatic retransmission） • 0：按照 CAN 标准，CAN 硬件在发送报文失败时会一直自动重传直到发送成功。 • 1：CAN 报文只被发送 1 次，不管发送的结果如何（成功、出错或仲裁丢失）。
位 3	RFLM：接收 FIFO 锁定模式（Receive FIFO locked mode） • 0：在接收溢出时 FIFO 未被锁定，当接收 FIFO 的报文未被读出，下一个收到的报文会覆盖原有的报文。 • 1：在接收溢出时 FIFO 被锁定，当接收 FIFO 的报文未被读出，下一个收到的报文会被丢弃。
位 2	TXFP：发送 FIFO 优先级（Transmit FIFO priority） 当有多个报文同时在等待发送时，该位决定这些报文的发送顺序。 • 0：优先级由报文的标识符来决定。 • 1：优先级由发送请求的顺序来决定。
位 1	SLEEP：睡眠模式请求（Sleep mode request） 软件对该位置’1’可以请求 CAN 进入睡眠模式，一旦当前的 CAN 活动（发送或接收报文）结束，CAN 就进入睡眠。 软件对该位清零使 CAN 退出睡眠模式。 当设置了 AWUM 位且在 CAN Rx 信号中检测出 SOF 位时，硬件对该位清零。 在复位后该位被置’1’，即 CAN 在复位后处于睡眠模式。
位 0	INRQ：初始化请求（Initialization request） 软件对该位清零可使 CAN 从初始化模式进入正常工作模式： 当 CAN 在接收引脚检测到连续的 11 个隐性位后，CAN 就达到同步，并为接收和发送数据作好准备了。为此，硬件相应地对 CAN_MSR 寄存器的 INAK 位清零。 软件对该位置 1 可使 CAN 从正常工作模式进入初始化模式： 一旦当前的 CAN 活动（发送或接收）结束，CAN 就进入初始化模式。相应地，硬件对 CAN_MSR 寄存器的 INAK 位置’1’。

23.8.2.2 CAN 主状态寄存器（CAN_MSR）

偏移地址：0x04

复位值：0x0000 0C02

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res				RX	SAMP	RXM	TXM	Res			SLAKI	WKUI	ERRI	SLAK	INAK
				r	r	r	r				rc_w1	rc_w1	rc_w1	r	r

位 31:12	Res：保留 必须保持复位值。
位 11	RX：CAN 接收电平（CAN Rx signal） 该位反映 CAN 接收引脚（CAN_RX）的实际电平。
位 10	SAMP：上次采样值（Last sample point） CAN 接收引脚的上次采样值（对应于当前接收位的值）。
位 9	RXM：接收模式（Receive mode） 该位为’1’表示 CAN 当前为接收器。
位 8	TXM：发送模式（Transmit mode） 该位为’1’表示 CAN 当前为发送器。
位 7:5	Res：保留 必须保持复位值。
位 4	SLAKI：睡眠确认中断（Sleep acknowledge interrupt） 当 SLKIE=1，一旦 CAN 进入睡眠模式硬件就对该位置’1，紧接着相应的中断被触发。当设置该位为’1’时，如果设置了 CAN_IER 寄存器中的 SLKIE 位，将产生一个状态改变中断。 软件可对该位清零，当 SLAK 位被清零时硬件也对该位清零。 *注意：当 SLKIE=0，不应该查询该位，而应该查询 SLAK 位来获知睡眠状态。*
位 3	WKUI：唤醒中断（Wakeup interrupt） 当 CAN 处于睡眠状态，一旦检测到帧起始位（SOF），硬件就置该位为’1’；并且如果 CAN_IER 寄存器的 WKUIE 位为’1’，则产生一个状态改变中断。 该位由软件清零。
位 2	ERRI：出错中断（Error interrupt） 当检测到错误时，CAN_ESR 寄存器的某位被置’1’，如果 CAN_IER 寄存器的相应中断使能位也被置’1’时，则硬件对该位置’1’；如果 CAN_IER 寄存器的 ERRIE 位为’1’，则产生状态改变中断。 该位由软件清零。
位 1	SLAK：睡眠模式确认（Sleep mode acknowledge） 该位由硬件置’1’，指示软件 CAN 模块正处于睡眠模式。该位是对软件请求进入睡眠模式的确认（对 CAN_MCR 寄存器的 SLEEP 位置’1’）。 当 CAN 退出睡眠模式时硬件对该位清零（需要跟 CAN 总线同步）。这里跟 CAN 总线同步是指，硬件需要在 CAN 的 RX 引脚上检测到连续的 11 位隐性位。 *注意：通过软件或硬件对 CAN_MCR 的 SLEEP 位清零，将启动退出睡眠模式的过程。有关清除 SLEEP 位的详细信息，参见 CAN_MCR 寄存器的 AWUM 位的描述。*
位 0	INAK：初始化确认（Initiation acknowledge） 该位由硬件置’1’，指示软件 CAN 模块正处于初始化模式。该位是对软件请求进入初始化模式的确认（对 CAN_MCR 寄存器的 INRQ 位置’1’）。

	当 CAN 退出初始化模式时硬件对该位清零（需要跟 CAN 总线同步）。这里跟 CAN 总线同步是指，硬件需要在 CAN 的 RX 引脚上检测到连续的 11 位隐性位。

23.8.2.3 CAN 发送状态寄存器（CAN_TSR）

偏移地址：0x08

复位值：0x1C00 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOW2	LOW1	LOW0	TME2	TME1	TME0	CODE[1:0]		ABRQ2	Res			TERR2	ALST2	TXOK2	RQCP2
r	r	r	r	r	r	r		rs				rc_w1	rc_w1	rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRQ1	Res			TERR1	ALST1	TXOK1	RQCP1	ABRQ0	Res			TERR0	ALST0	TXOK0	RQCP0
rs				rc_w1	rc_w1	rc_w1	rc_w1	rs				rc_w1	rc_w1	rc_w1	rc_w1

位 31	LOW2：邮箱 2 最低优先级标志（Lowest priority flag for mailbox 2） 当多个邮箱在等待发送报文，且邮箱 2 的优先级最低时，硬件对该位置’1’。
位 30	LOW1：邮箱 1 最低优先级标志（Lowest priority flag for mailbox 1） 当多个邮箱在等待发送报文，且邮箱 1 的优先级最低时，硬件对该位置’1’。
位 29	LOW0：邮箱 0 最低优先级标志（Lowest priority flag for mailbox 0） 当多个邮箱在等待发送报文，且邮箱 0 的优先级最低时，硬件对该位置’1’。 *注意：如果只有 1 个邮箱在等待，则 LOW[2:0]被清零。*
位 28	TME2：发送邮箱 2 空（Transmit mailbox 2 empty） 当邮箱 2 中没有等待发送的报文时，硬件对该位置’1’。
位 27	TME1：发送邮箱 1 空（Transmit mailbox1 empty） 当邮箱 1 中没有等待发送的报文时，硬件对该位置’1’。
位 26	TME0：发送邮箱 0 空（Transmit mailbox 0 empty） 当邮箱 0 中没有等待发送的报文时，硬件对该位置’1’。
位 25:24	CODE[1:0]：邮箱号（Mailbox code） • 当有至少 1 个发送邮箱为空时，这 2 位表示下一个空的发送邮箱号。 • 当所有的发送邮箱都为空时，这 2 位表示优先级最低的那个发送邮箱号。
位 23	ABRQ2：邮箱 2 中止发送（Abort request for mailbox 2） 软件对该位置’1’，可以中止邮箱 2 的发送请求，当邮箱 2 的发送报文被清除时硬件对该位清零。如果邮箱 2 中没有等待发送的报文，则对该位置’1’没有任何效果。
位 22:20	Res：保留 必须保持复位值。
位 19	TERR2：邮箱 2 发送失败（Transmission error of mailbox 2） 当邮箱 2 因为出错而导致发送失败时，对该位置’1’。
位 18	ALST2：邮箱 2 仲裁丢失（Arbitration lost for mailbox 2） 当邮箱 2 因为仲裁丢失而导致发送失败时，对该位置’1’。

位 17	TXOK2：邮箱 2 发送成功（Transmission OK of mailbox 2） 每次在邮箱 2 进行发送尝试后，硬件对该位进行更新： • 0：上次发送尝试失败。 • 1：上次发送尝试成功。 当邮箱 2 的发送请求被成功完成后，硬件对该位置’1’。请参见：图 23-7 发送邮箱状态
位 16	RQCP2：邮箱 2 请求完成（Request completed mailbox 2） 当上次对邮箱 2 的请求（发送或中止）完成后，硬件对该位置’1’。 软件对该位写’1’可以对其清零；当硬件接收到发送请求时也对该位清零（CAN_TI2R 寄存器的 TXRQ 位被置’1’）。 该位被清零时，邮箱 2 的其它发送状态位（TXOK2，ALST2 和 TERR2）也被清零。
位 15	ABRQ1：邮箱 1 中止发送（Abort request for mailbox 1） 软件对该位置’1’，可以中止邮箱 1 的发送请求，当邮箱 1 的发送报文被清除时硬件对该位清零。如果邮箱 1 中没有等待发送的报文，则对该位置’1’没有任何效果。
位 14:12	Res：保留 必须保持复位值。
位 11	TERR1：邮箱 1 发送失败（Transmission error of mailbox 1） 当邮箱 1 因为出错而导致发送失败时，对该位置’1’。
位 10	ALST1：邮箱 1 仲裁丢失（Arbitration lost for mailbox 1） 当邮箱 1 因为仲裁丢失而导致发送失败时，对该位置’1’。
位 9	TXOK1：邮箱 1 发送成功（Transmission OK of mailbox 1） 每次在邮箱 1 进行发送尝试后，硬件对该位进行更新： • 0：上次发送尝试失败。 • 1：上次发送尝试成功。 当邮箱 1 的发送请求被成功完成后，硬件对该位置’1’。请参见图 23-7 发送邮箱状态。
位 8	RQCP1：邮箱 1 请求完成（Request completed mailbox 1） 当上次对邮箱 1 的请求（发送或中止）完成后，硬件对该位置’1’。 软件对该位写’1’可以对其清零；当硬件接收到发送请求时也对该位清零（CAN_TI1R 寄存器的 TXRQ 位被置’1’）。 该位被清零时，邮箱 1 的其它发送状态位（TXOK1，ALST1 和 TERR1）也被清零。
位 7	ABRQ0：邮箱 0 中止发送（Abort request for mailbox 0） 软件对该位置’1’可以中止邮箱 0 的发送请求，当邮箱 0 的发送报文被清除时硬件对该位清零。如果邮箱 0 中没有等待发送的报文，则对该位置 1 没有任何效果。
位 6:4	Res：保留 必须保持复位值。
位 3	TERR0：邮箱 0 发送失败（Transmission error of mailbox 0） 当邮箱 0 因为出错而导致发送失败时，对该位置’1’。
位 2	ALST0：邮箱 0 仲裁丢失（Arbitration lost for mailbox 0） 当邮箱 0 因为仲裁丢失而导致发送失败时，对该位置’1’。
位 1	TXOK0：邮箱 0 发送成功（Transmission OK of mailbox 0）

	每次在邮箱 0 进行发送尝试后，硬件对该位进行更新： • 0：上次发送尝试失败 • 1：上次发送尝试成功 当邮箱 0 的发送请求被成功完成后，硬件对该位置’1’。请参见图 23-7。
位 0	RQCP0：邮箱 0 请求完成（Request completed mailbox 0） 当上次对邮箱 0 的请求（发送或中止）完成后，硬件对该位置’1’。 软件对该位写’1’可以对其清零；当硬件接收到发送请求时也对该位清零（CAN_TI0R 寄存器的 TXRQ 位被置’1’）。 该位被清零时，邮箱 0 的其它发送状态位（TXOK0，ALST0 和 TERR0）也被清零。

23.8.2.4 CAN 接收 FIFO0 寄存器（CAN_RF0R）

偏移地址：0x0C

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res										RFOM0	FOVR0	FULL0	Res	FMP0[1:0]	
										rs	rc_w1	rc_w1		r	

位 31:6	Res：保留 必须保持复位值。
位 5	RFOM0：释放接收 FIFO 0 输出邮箱（Release FIFO 0 output mailbox） 软件通过对该位置’1’来释放接收 FIFO 的输出邮箱。如果接收 FIFO 为空，那么对该位置’1’没有任何效果，即只有当 FIFO 中有报文时对该位置’1’才有意义。如果 FIFO 中有 2 个以上的报文，由于 FIFO 的特点，软件需要释放输出邮箱才能访问第 2 个报文。 当输出邮箱被释放时，硬件对该位清零。
位 4	FOVR0：FIFO 0 溢出（FIFO 0 overrun） 当 FIFO 0 已满，又收到新的报文且报文符合过滤条件，硬件对该位置’1’。该位由软件清零。
位 3	FULL0：FIFO 0 满（FIFO 0 full） 当 FIFO 0 中有 3 个报文时，硬件对该位置’1’。 该位由软件清零。
位 2	Res：保留 必须保持复位值。
位 1:0	FMP0[1:0]：FIFO 0 报文数目（FIFO 0 message pending） FIFO 0 报文数目这 2 位反映了当前接收 FIFO 0 中存放的报文数目。 每当 1 个新的报文被存入接收 FIFO 0，硬件就对 FMP0 加 1。 每当软件对 RFOM0 位写’1’来释放输出邮箱，FMP0 就被减 1，直到其为 0。

23.8.2.5 CAN 接收 FIFO1 寄存器（CAN_RF1R）

偏移地址：0x10

复位值：0x0000 0000

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

Res

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res										RFOM1	FOVR1	FULL1	Res	FMP1[1:0]	
										rs	rc_w1	rc_w1		r	

位 31:6	Res：保留 必须保持复位值。
位 5	RFOM1：释放接收 FIFO 1 输出邮箱（Release FIFO1 output mailbox） 软件通过对该位置’1’来释放接收 FIFO 的输出邮箱。如果接收 FIFO 为空，那么对该位置’1’没有任何效果，即只有当 FIFO 中有报文时对该位置’1’才有意义。如果 FIFO 中有 2 个以上的报文，由于 FIFO 的特点，软件需要释放输出邮箱才能访问第 2 个报文。 当输出邮箱被释放时，硬件对该位清零。
位 4	FOVR1：FIFO1 溢出（FIFO1 overrun） 当 FIFO 1 已满，又收到新的报文且报文符合过滤条件，硬件对该位置’1’。该位由软件清零。
位 3	FULL1：FIFO1 满（FIFO1 full） 当 FIFO 1 中有 3 个报文时，硬件对该位置’1’。 该位由软件清零。
位 2	Res：保留 必须保持复位值。
位 1:0	FMP1[1:0]：FIFO 1 报文数目（FIFO1message pending） FIFO 1 报文数目这 2 位反映了当前接收 FIFO 1 中存放的报文数目。 每当 1 个新的报文被存入接收 FIFO 1，硬件就对 FMP1 加 1。 每当软件对 RFOM1 位写 1 来释放输出邮箱，FMP1 就被减 1，直到其为 0。

23.8.2.6 CAN 中断使能寄存器（CAN_IER）

偏移地址：0x14

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERRIE	Res			LECIE	BOFIE	EPVIE	EWGIE	Res	FOVIE1	FFIE1	FMPIE1	FOVIE0	FFIE0	FMPIE0	TMEIE
rw				rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

位 31:18	Res：保留 必须保持复位值。
位 17	SLKIE：睡眠中断使能（Sleep interrupt enable） • 0：当 SLAKI 位被置’1’时，不产生中断。 • 1：当 SLAKI 位被置’1’时，产生中断。
位 16	WKUIE：唤醒中断使能（Wakeup interrupt enable） • 0：当 WKUI 位被置’1’时，不产生中断。

	• 1：当 WKUI 位被置’1’时，产生中断。
位 15	ERRIE：错误中断使能（Error interrupt enable） • 0：当 CAN_ESR 寄存器有错误挂号时，不产生中断。 • 1：当 CAN_ESR 寄存器有错误挂号时，产生中断。
位 14:12	Res：保留 必须保持复位值。
位 11	LECIE：上次错误号中断使能（Last error code interrupt enable） • 0：当检测到错误，硬件设置 LEC[2:0]时，不设置 ERRI 位。 • 1：当检测到错误，硬件设置 LEC[2:0]时，设置 ERRI 位为’1’。
位 10	BOFIE：离线中断使能（Bus-off interrupt enable） • 0：当 BOFF 位被置’1’时，不设置 ERRI 位。 • 1：当 BOFF 位被置’1’时，设置 ERRI 位为’1’。
位 9	EPVIE：错误被动中断使能（Error Passive Interrupt Enable） • 0：当 EPVF 位被置’1’时，不设置 ERRI 位。 • 1：当 EPVF 位被置’1’时，设置 ERRI 位为’1’。
位 8	EWGIE：错误警告中断使能（Error warning interrupt enable） • 0：当 EWGF 位被置’1’时，不设置 ERRI 位。 • 1：当 EWGF 位被置’1’时，设置 ERRI 位为’1’。
位 7	Res：保留 必须保持复位值。
位 6	FOVIE1：FIFO 1 溢出中断使能（FIFO overrun interrupt enable） • 0：当 FIFO 1 的 FOVR 位被置’1’时，不产生中断。 • 1：当 FIFO 1 的 FOVR 位被置’1’时，产生中断。
位 5	FFIE1：FIFO 1 满中断使能（FIFO full interrupt enable） • 0：当 FIFO 1 的 FULL 位被置’1’时，不产生中断。 • 1：当 FIFO 1 的 FULL 位被置’1’时，产生中断。
位 4	FMPIE1：FIFO 1 消息保持中断使能（FIFO message pending interrupt enable） • 0：当 FIFO 1 的 FMP[1:0]位为非 0 时，不产生中断。 • 1：当 FIFO 1 的 FMP[1:0]位为非 0 时，产生中断。
位 3	FOVIE0：FIFO 0 溢出中断使能（FIFO overrun interrupt enable） • 0：当 FIFO 0 的 FOVR 位被置’1’时，不产生中断。 • 1：当 FIFO 0 的 FOVR 位被置’1’时，产生中断。
位 2	FFIE0：FIFO 0 满中断使能（FIFO full interrupt enable） • 0：当 FIFO 0 的 FULL 位被置’1’时，不产生中断。 • 1：当 FIFO 0 的 FULL 位被置’1’时，产生中断。
位 1	FMPIE0：FIFO 0 消息挂号中断使能（FIFO message pending interrupt enable） • 0：当 FIFO 0 的 FMP[1:0]位为非 0 时，不产生中断。

	• 1：当 FIFO 0 的 FMP[1:0]位为非 0 时，产生中断。
位 0	TMEIE：发送邮箱空中断使能（Transmit mailbox empty interrupt enable） • 0：当 RQCPx 位被置'1'时，不产生中断。 • 1：当 RQCPx 位被置'1'时，产生中断。 *注意：请参考章节："CAN 中断"。*

23.8.2.7 CAN 错误状态寄存器（CAN_ESR）

偏移地址：0x18

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REC[7:0]								TEC[7:0]							
r								r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res									LEC[2:0]			Res	BOFF	EPVF	EWGF
									rw				r	r	r

位 31:24	REC[7:0]：接收错误计数器（Receive error counter） 这个计数器按照 CAN 协议的故障界定机制的接收部分实现。按照 CAN 的标准，当接收出错时，根据出错的条件，该计数器加 1 或加 8；而在每次接收成功后，该计数器减 1，或当该计数器的值大于 128 时，设置它的值为 120。当该计数器的值超过 127 时，CAN 进入错误被动状态。
位 23:16	TEC[7:0]：9 位发送错误计数器的低 8 位（Least significant byte of the 9-bit transmit error counter） 与上面相似，这个计数器按照 CAN 协议的故障界定机制的发送部分实现。
位 15:7	Res：保留 必须保持复位值。
位 6:4	LEC[2:0]：上次错误代码（Last error code） 在检测到 CAN 总线上发生错误时，硬件根据出错情况设置。当报文被正确发送或接收后，硬件清除其值为'0'。 硬件没有使用错误代码 7，软件可以设置该值，从而可以检测代码的更新。 • 000：没有错误 • 001：位填充错 • 010：格式（Form）错 • 011：确认（ACK）错 • 100：隐性位错 • 101：显性位错 • 110：CRC 错 • 111：由软件设置
位 3	Res：保留 必须保持复位值。
位 2	BOFF：离线标志（Bus-off flag） 当进入离线状态时，硬件对该位置'1'。当发送错误计数器 TEC 溢出，即大于 255 时，CAN 进入离线状态。请参考章节："位时间特性"。
位 1	EPVF：错误被动标志（Error passive flag）

	当出错次数达到错误被动的阈值时，硬件对该位置’1’。（接收错误计数器或发送错误计数器的值>127）。
位 0	EWGF：错误警告标志（Error warning flag） 当出错次数达到警告的阈值时，硬件对该位置’1’。 （接收错误计数器或发送错误计数器的值≥96）。

23.8.2.8 CAN 位时序寄存器（CAN_BTR）

偏移地址：0x1C

复位值：0x0123 0000

注意：当 CAN 处于初始化模式时，该寄存器只能由软件访问。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SILM	LBKM	Res				SJW[1:0]		Res	TS2[2:0]			TS1[3:0]			
rw	rw					rw			rw			rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res						BRP[9:0]									
						rw									

位 31	SILM：静默模式（用于调试）（Silent mode for debug） • 0：正常状态 • 1：静默模式
位 30	LBKM：环回模式（用于调试）（Loop back mode for debug） • 0：禁止环回模式 • 1：允许环回模式
位 29:26	Res：保留 必须保持复位值。
位 25:24	SJW[1:0]：重新同步跳跃宽度（Resynchronization jump width） 为了重新同步，该位域定义了 CAN 硬件在每位中可以延长或缩短多少个时间单元的上限。 $t_{RJW}=t_q$ *（SJW[1:0] +1）
位 23	Res：保留 必须保持复位值。
位 22:20	TS2[2:0]：时间段 2（Time segment 2） 该位域定义了时间段 2 占用了多少个时间单元 $t_{BS2}=t_q$ *（TS2[2:0] +1）
位 19:16	TS1[3:0]：时间段 1（Time segment 1） 该位域定义了时间段 1 占用了多少个时间单元。 $t_{BS1}=t_q$ *（TS1[3:0] +1） 关于位时间特性的详细信息，请参考章节：“位时间特性”。
位 15:10	Res：保留 必须保持复位值。
位 9:0	BRP[9:0]：波特率分频器（Baud rate prescaler） 该位域定义了时间单元（t_q）的时间长度。

	$t_{q=}$（BRP[9:0]+1）* t_{PCLK}

23.8.3 CAN 邮箱寄存器

本节描述发送和接收邮箱寄存器。关于寄存器映像的详细信息，请参考章节：“报文存储”。

除了下述例外，发送和接收邮箱几乎一样：

- CAN_RDTxR 寄存器的 FMI 域；
- 接收邮箱是只读的。

发送邮箱只有在它为空时才是可写的，CAN_TSR 寄存器的相应 TME 位为‘1’，表示发送邮箱为空。

共有 3 个发送邮箱和 2 个接收邮箱。每个接收邮箱为 3 级深度的 FIFO，并且只能访问 FIFO 中最先收到的报文。

每个邮箱包含 4 个寄存器。

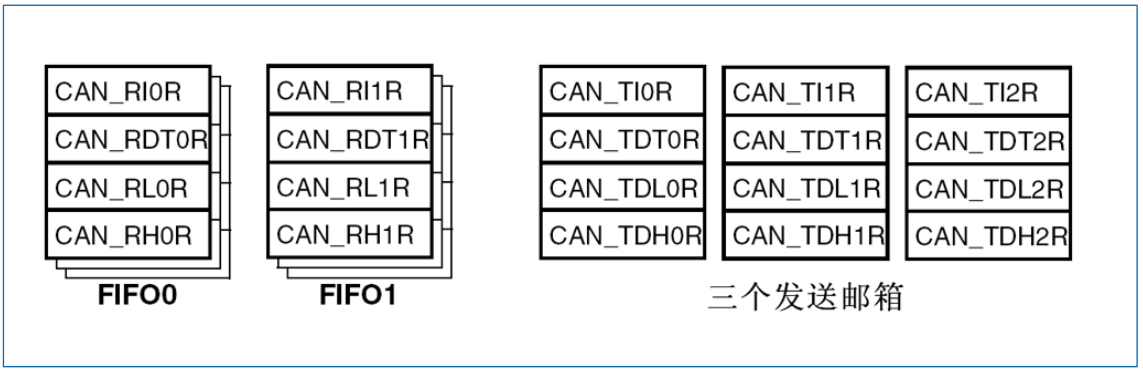


图 23-16 CAN 邮箱寄存器

23.8.3.1 发送邮箱标识符寄存器（CAN_TIxR）（x=0..2）

偏移地址：0x180+x*16

复位值：0xXXXX XXXX

X 代表不定值（除了第 0 位，复位时 TXRQ=0）。

注意：

当其所属的邮箱处在等待发送的状态时，该寄存器是写保护的。

该寄存器实现了发送请求控制功能（第 0 位）－复位值为 0。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STID[10:0]/EXID[28:18]											EXID[17:13]				
rw											rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXID[12:0]													IDE	RTR	TXRQ
rw													rw	rw	rw

位 31:21	STID[10:0]/EXID[28:18]：标准标识符或扩展标识符（Standard identifier or extended identifier） 依据 IDE 位的内容，该位域或是标准标识符，或是扩展身份标识的高字节。
位 20:3	EXID[17:0]：扩展标识符（Extended identifier） 扩展身份标识的低字节。
位 2	IDE：标识符选择（Identifier extension） 该位决定发送邮箱中报文使用的标识符类型

	• 0：使用标准标识符 • 1：使用扩展标识符
位 1	RTR：远程发送请求（Remote transmission request） • 0：数据帧 • 1：远程帧
位 0	TXRQ：发送邮箱请求（Transmit mailbox request） 由软件对其置‘1’，来请求发送邮箱的数据。当数据发送完成，邮箱为空时，硬件对其清零。

23.8.3.2 发送邮箱数据长度和时间戳寄存器（CAN_TDTxR）（x=0..2）

偏移地址：0x184+x*16

复位值：0xXXXX XXXX

X 代表不定值。

当邮箱不在空置状态时，该寄存器的所有位为写保护。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIME[15:0]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res							TGT	Res				DLC[3:0]			
							rw					rw			

位 31:16	TIME[15:0]：报文时间戳（Message time stamp） 该域包含了，在发送该报文 SOF 的时刻，16 位定时器的值。
位 15:9	Res：保留 必须保持复位值。
位 8	TGT：发送时间戳（Transmit global time） 只有在 CAN 处于时间触发通信模式，即 CAN_MCR 寄存器的 TTCM 位为‘1’时，该位才有效。 • 0：不发送时间戳 TIME[15:0] • 1：发送时间戳 TIME[15:0] 在长度为 8 的报文中，时间戳 TIME[15:0]是最后 2 个发送的字节：TIME[7:0]作为第 7 个字节，TIME[15:8]为第 8 个字节，它们替换了写入 CAN_TDHxR[31:16]的数据（DATA6[7:0]和 DATA7[7:0]）。为了把时间戳的 2 个字节发送出去，DLC 必须编程为 8。
位 7:4	Res：保留 必须保持复位值。
位 3:0	DLC[3:0]：发送数据长度（Data length code） 该域指定了数据报文的数据长度或者远程帧请求的数据长度。1 个报文包含 0 到 8 个字节数据，而这由 DLC 决定。

23.8.3.3 发送邮箱低字节数据寄存器（CAN_TDLxR）（x=0..2）

偏移地址：0x188+x*16

复位值：0xXXXX XXXX

X 代表不定值。

当邮箱不在空置状态时，该寄存器的所有位为写保护。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA3[7:0]								DATA2[7:0]							
rw								rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA1[7:0]								DATA0[7:0]							
rw								rw							

位 31:24	DATA3[7:0]：数据字节 3（Data byte 3） 报文的数据字节 3。
位 23:16	DATA2[7:0]：数据字节 2（Data byte 2） 报文的数据字节 2。
位 15:8	DATA1[7:0]：数据字节 1（Data byte 1） 报文的数据字节 1。
位 7:0	DATA0[7:0]：数据字节 0（Data byte 0） 报文的数据字节 0。 报文包含 0 到 8 个字节数据，且从字节 0 开始。

23.8.3.4 发送邮箱高字节数据寄存器（CAN_TDHxR）（x=0..2）

偏移地址：0x18C+x*16

复位值：0xXXXX XXXX

X 代表不定值。

当邮箱不在空置状态时，该寄存器的所有位为写保护。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA7[7:0]								DATA6[7:0]							
rw								rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA5[7:0]								DATA4[7:0]							
rw								rw							

位 31:24	DATA7[7:0]：数据字节 7（Data byte 7） 报文的数据字节 7 注意：如果 CAN_MCR 寄存器的 TTCM 位为’1’，且该邮箱的 TGT 位也为’1’，那么 DATA7 和 DATA6 将被 TIME 时间戳代替。
位 23:16	DATA6[7:0]：数据字节 6（Data byte 6） 报文的数据字节 6。
位 15:8	DATA5[7:0]：数据字节 5（Data byte 5） 报文的数据字节 5。
位 7:0	DATA4[7:0]：数据字节 4（Data byte 4） 报文的数据字节 4。

23.8.3.5 接收 FIFO 邮箱标识符寄存器（CAN_RIxR）（x=0..1）

偏移地址：0x1B0+x*16

复位值：0xXXXX XXXX

X 代表不定值。

所有接收邮箱寄存器都是只读的。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STID[10:0]/EXID[28:18]											EXID[17:13]				
r											r				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXID[12:0]													IDE	RTR	Res
r													r	r	

位 31:21	STID[10:0]/EXID[28:18]：标准标识符或扩展标识符（Standard identifier or extended identifier） 依据 IDE 位的内容，该位域或是标准标识符，或是扩展身份标识的高字节。
位 20:3	EXID[17:0]：扩展标识符（Extended identifier） 扩展标识符的低字节。
位 2	IDE：标识符选择（Identifier extension） 该位决定接收邮箱中报文使用的标识符类型。 • 0：使用标准标识符 • 1：使用扩展标识符
位 1	RTR：远程发送请求（Remote transmission request） • 0：数据帧 • 1：远程帧
位 0	Res：保留 必须保持复位值。

23.8.3.6 接收 FIFO 邮箱数据长度和时间戳寄存器（CAN_RDTxR）（x=0..1）

偏移地址：0x1B4+x*16

复位值：0xXXXX XXXX

X 代表不定值。

所有接收邮箱寄存器都是只读的。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIME[15:0]															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FMI[7:0]								Res				DLC[3:0]			
r												r			

位 31:16	TIME[15:0]：报文时间戳（Message time stamp） 该域包含了，在接收该报文 SOF 的时刻，16 位定时器的值。
位 15:8	FMI[7:0]：过滤器匹配序号（Filter match index） 这里是存在邮箱中的信息传送的过滤器序号。关于标识符过滤的细节，请参考章节：“标识符过滤”中有关过滤器匹配序号。
位 7:4	Res：保留 必须保持复位值。
位 3:0	DLC[3:0]：接收数据长度（Data length code） 该域表明接收数据帧的数据长度（0~8）。对于远程帧请求，数据长度 DLC 恒为 0。

23.8.3.7 接收 FIFO 邮箱低字节数据寄存器（CAN_RDLxR）（x=0..1）

偏移地址：0x1B8+x*16

复位值：0xXXXX XXXX

X 代表不定值。

所有接收邮箱寄存器都是只读的。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA3[7:0]								DATA2[7:0]							
r								r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA1[7:0]								DATA0[7:0]							
r								r							

位 31:24	DATA3[7:0]：数据字节 3（Data byte 3） 报文的数据字节 3。
位 23:16	DATA2[7:0]：数据字节 2（Data byte 2） 报文的数据字节 2。
位 15:8	DATA1[7:0]：数据字节 1（Data byte 1） 报文的数据字节 1。
位 7:0	DATA0[7:0]：数据字节 0（Data byte 0） 报文的数据字节 0。报文包含 0 到 8 个字节数据，且从字节 0 开始。

23.8.3.8 接收 FIFO 邮箱高字节数据寄存器（CAN_RDHxR）（x=0..1）

偏移地址：0x1BC+x*16

复位值：0xXXXX XXXX

X 代表不定值。

所有接收邮箱寄存器都是只读的。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA7[7:0]								DATA6[7:0]							
r								r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA5[7:0]								DATA4[7:0]							
r								r							

位 31:24	DATA7[7:0]：数据字节 7（Data byte 7） 报文的数据字节 7
位 23:16	DATA6[7:0]：数据字节 6（Data byte 6） 报文的数据字节 6。
位 15:8	DATA5[7:0]：数据字节 5（Data byte 5） 报文的数据字节 5。
位 7:0	DATA4[7:0]：数据字节 4（Data byte 4） 报文的数据字节 4。

23.8.4 CAN 过滤器寄存器

23.8.4.1 CAN 过滤器主控寄存器（CAN_FMR）

偏移地址：0x200

复位值：0x2A1C 0E01

该寄存器的非保留位完全由软件控制。

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

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

位 31:1	Res：保留 必须保持复位值。
位 0	FINIT：过滤器初始化模式（Filter initiate mode） 针对所有过滤器组的初始化模式设置： • 0：过滤器组工作在正常模式 • 1：过滤器组工作在初始化模式

23.8.4.2 CAN 过滤器模式寄存器（CAN_FM1R）

偏移地址：0x204

复位值：0x0000 0000

只有在设置 CAN_FMR（FINIT=1），使过滤器处于初始化模式下，才能对该寄存器写入。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res		FBM13	FBM12	FBM11	FBM10	FBM9	FBM8	FBM7	FBM6	FBM5	FBM4	FBM3	FBM2	FBM1	FBM0
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:14	Res：保留 必须保持复位值。
位 x （x=13..0）	FBMx：过滤器模式（Filter mode） 过滤器组 x 的工作模式： • 0：过滤器组 x 的 2 个 32 位寄存器工作在标识符屏蔽位模式 • 1：过滤器组 x 的 2 个 32 位寄存器工作在标识符列表模式

注意：请参考图：过滤器组位宽设置－寄存器组织。

23.8.4.3 CAN 过滤器位宽寄存器（CAN_FS1R）

偏移地址：0x20C

复位值：0x0000 0000

只有在设置 CAN_FMR（FINIT=1），使过滤器处于初始化模式下，才能对该寄存器写入。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res		FSC13	FSC12	FSC11	FSC10	FSC9	FSC8	FSC7	FSC6	FSC5	FSC4	FSC3	FSC2	FSC1	FSC0
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:14	Res：保留 必须保持复位值。
位 x （x=13..0）	FSCx：过滤器位宽设置（Filter scale configuration） 过滤器组 x 的位宽。 ● 0：过滤器位宽为 2 个 16 位 ● 1：过滤器位宽为单个 32 位

注意：请参考图：过滤器组位宽设置－寄存器组织。

23.8.4.4 CAN 过滤器 FIFO 关联寄存器（CAN_FFA1R）

偏移地址：0x214

复位值：0x0000 0000

只有在设置 CAN_FMR（FINIT=1），使过滤器处于初始化模式下，才能对该寄存器写入。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res		FFA13	FFA12	FFA11	FFA10	FFA9	FFA8	FFA7	FFA6	FFA5	FFA4	FFA3	FFA2	FFA1	FFA0
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:14	Res：保留 必须保持复位值。
位 x (x=13..0)	FFAx：过滤器位宽设置（Filter scale configuration） 报文在通过了某过滤器的过滤后，将被存放到其关联的 FIFO 中。 ● 0：过滤器组 x 被关联到 FIFO0。 ● 1：过滤器组 x 被关联到 FIFO1。

23.8.4.5 CAN 过滤器激活寄存器（CAN_FA1R）

偏移地址：0x21C

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res		FACT13	FACT12	FACT11	FACT10	FACT9	FACT8	FACT7	FACT6	FACT5	FACT4	FACT3	FACT2	FACT1	FACT0
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:14	Res：保留 必须保持复位值。
位 x (x=13..0)	FACTx：过滤器激活（Filter activate） 软件对某位设置'1'来激活相应的过滤器。只有对 FACTx 位清零，或对 CAN_FMR 寄存器的 FINIT 位设置'1'后，才能修改相应的过滤器寄存器 x（CAN_FiRx[0:1]） ● 0：过滤器被禁用 ● 1：过滤器被激活

23.8.4.6 CAN 过滤器组 i 的寄存器 x（CAN_FiRx）（i=0..13, x=1..2）

偏移地址：0x240+i*8+(x-1)*4

复位值：0xXXXX XXXX

X 表示未定义值。

注意：共有 14 组过滤器：i=0..13。每组过滤器由 2 个 32 位的寄存器，CAN_FiR[2:1]组成。只有在 CAN_FAxR 寄存器相应的 FACTx 位清零，或 CAN_FMR 寄存器的 FINIT 位为'1'时，才能修改相应的过滤器寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FB31	FB30	FB29	FB28	FB27	FB26	FB25	FB24	FB23	FB22	FB21	FB20	FB19	FB18	FB17	FB16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FB15	FB14	FB13	FB12	FB11	FB10	FB9	FB8	FB7	FB6	FB5	FB4	FB3	FB2	FB1	FB0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 y （y=31..0）	FBy：过滤器位（Filter bits） • 标识符模式 寄存器的每位对应于所期望的标识符的相应位的电平。 ○ 0：期望相应位为显性位 ○ 1：期望相应位为隐性位 • 屏蔽位模式 寄存器的每位指示是否对应的标识符寄存器位一定要与期望的标识符的相应位一致。 ○ 0：不关心，该位不用于比较。 ○ 1：必须匹配，到来的标识符位必须与滤波器对应的标识符寄存器位相一致。

注意：

根据过滤器位宽和模式的不同设置，过滤器组中的两个寄存器的功能也不尽相同。关于过滤器的映射，功能描述和屏蔽寄存器的关联，请参见：“标识符过滤”。

屏蔽位模式下的屏蔽/标识符寄存器，跟标识符列表模式下的寄存器位定义相同。

24 串行外设接口（SPI/I2S）

24.1 SPI/I2S 简介

SPI/I2S 接口支持 SPI 协议和 I2S 音频协议，由软件决定配置为 SPI 接口或 I2S 接口。复位后默认为 SPI 模式。

SPI 接口支持半双工、全双工、单工通信，器件可以作为 SPI 主机，也可作为 SPI 从机。

集成电路内置音频总线（I2S）也是同步串行通信接口。它能够在从模式或主模式下作为接收器或发送器。

I2S 接口也是同步串行通信接口，支持半双工通信，支持四种 I2S 标准：Philips、MSB、LSB、PCM。

24.2 SPI 和 I2S 主要特征

24.2.1 SPI 特征

- 主模式或从模式操作
- 3 线全双工同步传输
- 多主模式功能
- 带或不带第三根双向数据线的双线单工同步传输
- 8 个波特率预分频系数（最大为 $f_{PCLK}/2$）
- 主模式和从模式的快速通信
- 主模式和从模式下均可以由软件或硬件进行 NSS 管理：主/从操作模式的动态改变
- 时钟极性和相位可配置
- 数据顺序可配置，MSB 在前或 LSB 在前
- 可触发中断的专用发送和接收标志
- SPI 总线忙状态标志
- 支持可靠通信的硬件 CRC：
  - 在发送模式下可将 CRC 值作为最后一个字节发送
  - 根据收到的最后一个字节自动进行 CRC 错误校验
- 可触发中断的主模式故障、过载以及 CRC 错误标志
- 支持 DMA 功能的 1 字节发送和接收缓冲器：产生发送和接收请求
- SPI 接口支持 8 或 16 比特数据包长度可配（此功能无法与 SPI 的 CRC 校验功能同时使用）
- 3 个 SPI 都具备 I2S 的功能

24.2.2 I2S 功能

- 单工通信（仅作为发送器或接收器）
- 主/从操作
- 8 位线性可编程预分频器，获得精确的音频采样频率（8 kHz 到 192 kHz）
- 数据格式支持 16 位，24 位或者 32 位
- 数据包的帧由音频通道固定为 16 位（可容纳 16 位数据帧）或 32 位（可容纳 16 位、24 位、32 位数据帧）
- 时钟极性（稳定态）可配置

- 从发送模式下的下溢标志位和主/从接收模式下的溢出标志位
- 发送和接收共用一个 16 位数据寄存器，在通道两端各有一个寄存器
- 支持的 I2S 协议：
  - I2S 飞利浦标准
  - MSB 对齐标准（左对齐）
  - LSB 对齐标准（右对齐）
  - PCM 标准（16 位通道帧上带长或短帧同步或者 16 位数据帧扩展为 32 位通道帧）
- 数据方向总是 MSB 在先
- 发送和接收都具有 DMA 能力
- 主时钟可以输出到外部音频设备，比率固定为 256xFs（Fs 为音频采样频率）

24.3 SPI/I2S 实现

表 24-1 SPI 实现

SPI 特性	SPI1	SPI2	SPI3
硬件 CRC 计算	支持	支持	支持
RX/TX FIFO	支持	支持	支持
I2S 模式	支持	支持	支持
TI 模式	支持	支持	支持

24.4 SPI 功能描述

24.4.1 概述

SPI 的方框图见下图：

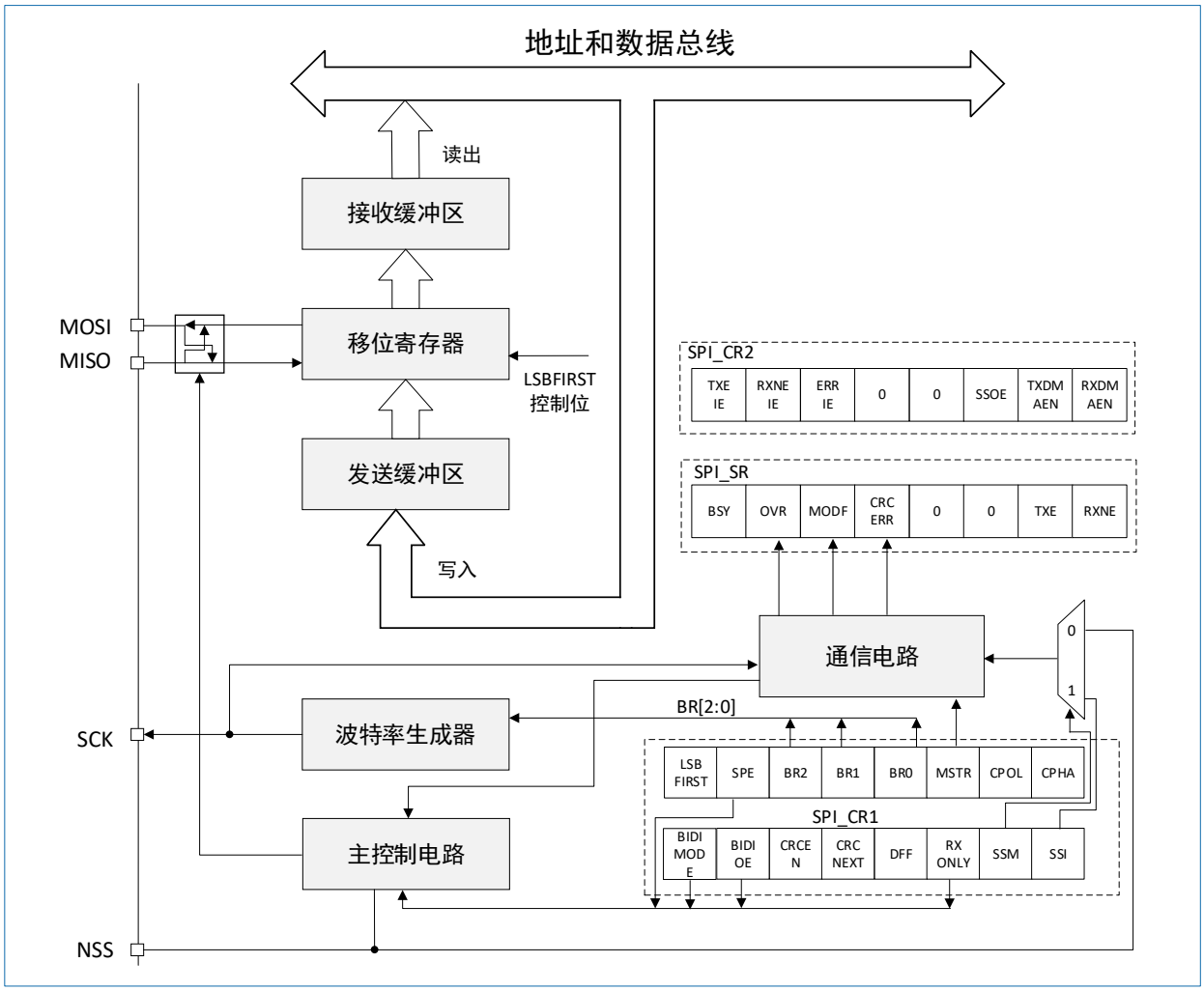


图 24-1 SPI 框图

通常 SPI 通过 4 个引脚与外部器件相连：

- MISO：主设备输入/从设备输出引脚。该引脚在从模式下发送数据，在主模式下接收数据。
- MOSI：主设备输出/从设备输入引脚。该引脚在主模式下发送数据，在从模式下接收数据。
- SCK：串口时钟，作为主设备的输出，从设备的输入。
- NSS：从设备选择。这是一个可选的引脚，用来选择主/从设备。它的功能是用来作为“片选引脚”，让主设备可以单独地与特定的从设备通讯，避免数据线上的冲突。从设备的 NSS 引脚可以由主设备的一个标准 I/O 引脚来驱动。一旦被使能（SSOE 位），NSS 引脚也可以作为输出引脚，并在 SPI 处于主模式时拉低；此时，所有的 SPI 设备，如果它们的 NSS 引脚连接到主设备的 NSS 引脚，则会检测到低电平，如果它们被设置为 NSS 硬件模式，就会自动进入从设备状态。当配置为主设备、NSS 配置为输入引脚（MSTR=1，SSOE=0）时，如果 NSS 被拉低，则这个 SPI 设备进入主模式失败状态：即 MSTR 位被自动清除，此设备进入从模式。

下图是一个单主和单从设备互连的例子：

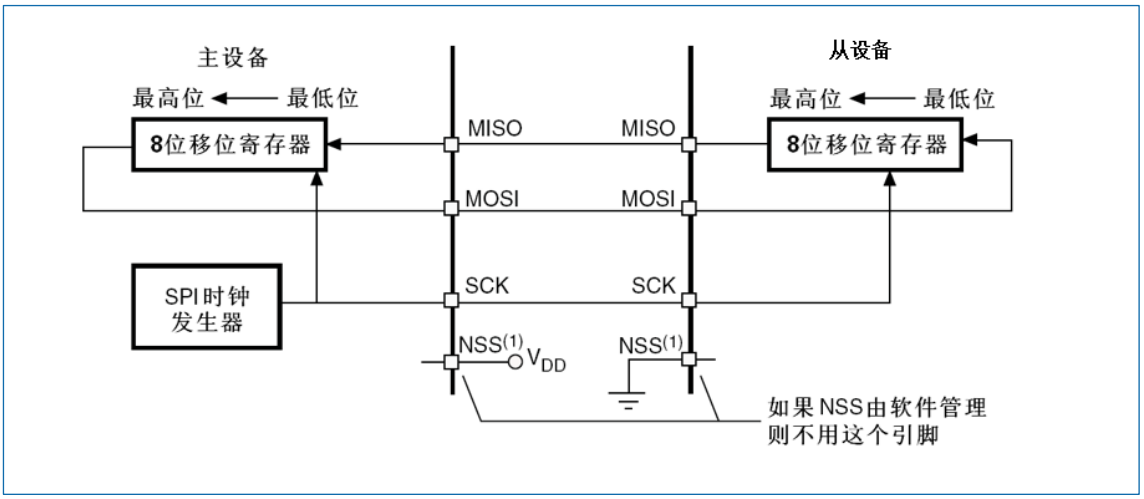


图 24-2 单主和单从应用

上图说明：（1）这里 NSS 引脚设置为输入。

MOSI 脚相互连接，MISO 脚相互连接。这样，数据在主和从之间串行地传输（MSB 位在前）。

通信总是由主设备发起。主设备通过 MOSI 脚把数据发送给从设备，从设备通过 MISO 引脚回传数据。这意味全双工通信的数据输出和数据输入是用同一个时钟信号同步的；时钟信号由主设备通过 SCK 脚提供。

24.4.1.1 从设备选择（NSS）脚管理

可以通过 SPIx_CR1 的 SSM 位设置硬件或者软件管理从设备选择：

- 软件 NSS 模式（SSM=1）：NSS 信号电平通过写内部 SPIx_CR1 的 SSI 位来控制，在这种模式下 SPI 不需要占用外部 NSS 引脚，可以用作其他功能。
- 硬件 NSS 模式，基于 NSS 的输出配置（SPIx_CR2 的 SSOE 位）有两种情况：
  - NSS 输出使能（SSM=0，SSOE=1）：这种配置只能用于 SPI 主设备，并且在主设备开始传输后 NSS 被驱动为低，并且一直保持到 SPI 关闭。
  - NSS 输出关闭（SSM=0，SSOE=0）：这种配置可以用于主机模式下实现多主机系统。在从机模式下，这种配置就是经典的 NSS 引脚作为从机输入信号，当 NSS 为低时从机被选中，当 NSS 为高时从机未被选中。

24.4.1.2 时钟信号的相位和极性

SPIx_CR1 寄存器的 CPOL（时钟极性）和 CPHA 位，能够组合成四种可能的时序关系。CPOL 位控制在没有 16 位或 32 位时，时钟的空闲状态电平，此位对主模式和从模式下的设备都有效。如果 CPOL 被清零，SCK 引脚在空闲状态保持低电平；如果 CPOL 被置'1'，SCK 引脚在空闲状态保持高电平。如果 CPHA（时钟相位）位被置'1'，SCK 时钟的第二个边沿（CPOL 位为 0 时就是下降沿，CPOL 位为'1'时就是上升沿）进行数据位的采样，数据在第二个时钟边沿被锁存。如果 CPHA 位被清零，SCK 时钟的第一边沿（CPOL 位为'0'时就是上升沿，CPOL 位为'1'时就是下降沿）进行数据位采样，数据在第一个时钟边沿被锁存。

CPOL 时钟极性和 CPHA 时钟相位的组合选择数据捕捉的时钟边沿。下图显示了 SPI 传输的 4 种 CPHA 和 CPOL 位组合。此图可以解释为主设备和从设备的 SCK 脚、MISO 脚、MOSI 脚直接连接的主或从时序图。

注意：

在改变 CPOL/CPHA 位之前，必须清除 SPE 位将 SPI 禁止。

主和从必须配置成相同的时序模式。

SCK 的空闲状态必须和 SPIx_CR1 寄存器指定的极性一致（CPOL 为'1'时，空闲时应上拉 SCK 为高电平；CPOL 为'0'时，空闲时应下拉 SCK 为低电平）。

数据帧格式（8 位或 16 位）由 SPIx_CR1 寄存器的 DFF 位选择，并且决定发送/接收的数据长度。

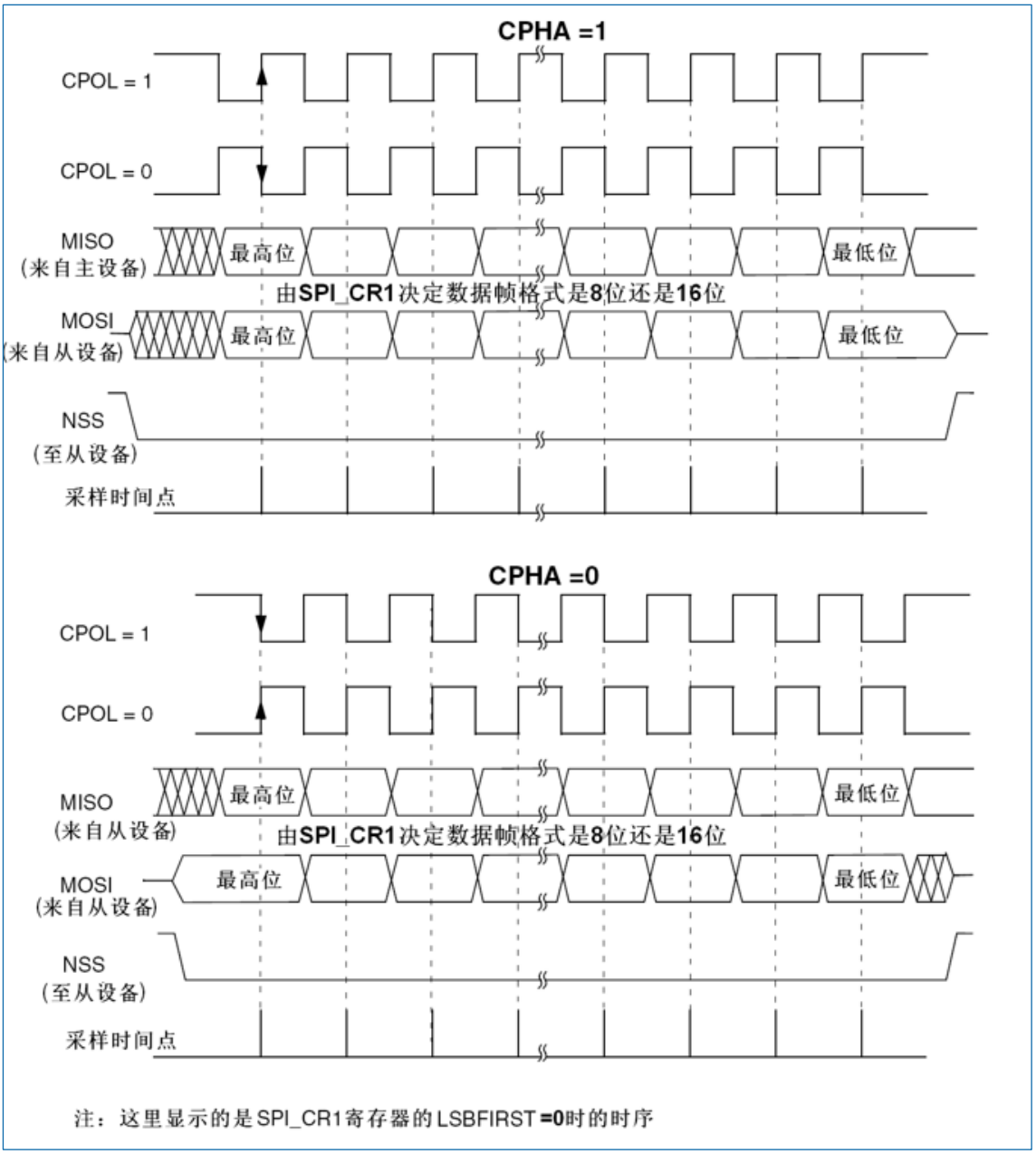


图 24-3 数据时钟时序图

24.4.1.3 数据帧格式

根据 SPIx_CR1 寄存器中的 LSBFIRST 位，输出数据位时可以 MSB 在先也可以 LSB 在先。根据 SPIx_CR1 寄存器的 DFF 位，每个数据帧可以是 8 位或是 16 位。所选择的数据帧格式对发送和/或接收都有效。

24.4.2 配置 SPI 为从模式

在从模式下，SCK 引脚用于接收从主设备来的串行时钟。SPIx_CR1 寄存器中 BR[2:0]的设置不会影响数据传输速率。

注意：建议在主设备发送时钟之前使能 SPI 从设备，否则可能会发生意外的数据传输。在通信时钟的第一个边沿到来之前或正在进行的通信结束之前，从设备的数据寄存器必须就绪。在使能从设备和主设备之前，通信时钟的极性必须处于稳定的数值。

配置步骤：

1. 设置 DFF 位以定义数据帧格式为 8 位或 16 位。
2. 选择 CPOL 和 CPHA 位来定义数据传输和串行时钟之间的相位关系（见图 24-3）。为保证正确的数据传输，从设备和主设备的 CPOL 和 CPHA 位必须配置成相同的方式。
3. 帧格式（SPIx_CR1 寄存器中的 LSBFIRST 位定义的”MSB 在前”还是”LSB 在前”）必须与主设备相同。
4. 硬件模式下（参考从选择（NSS）脚管理部分），在完整的数据帧（8 位或 16 位）传输过程中，NSS 引脚必须为低电平。在 NSS 软件模式下，设置 SPIx_CR1 寄存器中的 SSM 位并清除 SSI 位。
5. 清除 MSTR 位、设置 SPE 位（SPIx_CR1 寄存器），使相应引脚工作于 SPI 模式下。在这个配置中，MOSI 引脚是数据输入，MISO 引脚是数据输出。

数据发送过程：

在写操作中，数据字被并行地写入发送缓冲器。

当从设备收到时钟信号，并且在 MOSI 引脚上出现第一个数据位时，发送过程开始。余下的位（对于 8 位数据帧格式，还有 7 位；对于 16 位数据帧格式，还有 15 位）被装进移位寄存器。当发送缓冲器中的数据传输到移位寄存器时，SPIx_SR 寄存器的 TXE 标志被设置，如果设置了 SPIx_CR2 寄存器的 TXEIE 位，将会产生中断。

数据接收过程：

对于接收器，当数据接收完成时：

- 移位寄存器中的数据传送到接收缓冲器，SPIx_SR 寄存器中的 RXNE 标志被设置。
- 如果设置了 SPIx_CR2 寄存器中的 RXNEIE 位，则产生中断。

在最后一个采样时钟边沿后，RXNE 位被置’1’，移位寄存器中接收到的数据字节被传送到接收缓冲器。当读 SPIx_DR 寄存器时，SPI 设备返回这个接收缓冲器的数值。读 SPIx_DR 寄存器时，RXNE 位被清除。

24.4.3 配置 SPI 为主模式

在主配置时，在 SCK 脚产生串行时钟。

配置步骤：

1. 通过 SPIx_CR1 寄存器的 BR[2:0]位定义串行时钟波特率。
2. 选择 CPOL 和 CPHA 位，定义数据传输和串行时钟间的相位关系（见图 24-3）。
3. 设置 DFF 位来定义 8 位或 16 位数据帧格式。
4. 配置 SPIx_CR1 寄存器的 LSBFIRST 位定义帧格式。
5. 如果需要 NSS 引脚工作在输入模式，硬件模式下，在整个数据帧传输期间应把 NSS 脚连接到高电平；在软件模式下，需设置 SPIx_CR1 寄存器的 SSM 位和 SSI 位。如果 NSS 引脚工作 在输出模式，则只需设置 SSOE 位。
6. 必须设置 MSTR 位和 SPE 位（只当 NSS 脚被连到高电平，这些位才能保持置位）。在这个配置中，MOSI 引脚是数据输出，而 MISO 引脚是数据输入。

数据发送过程：

当写入数据至发送缓冲器时，发送过程开始。在发送第一个数据位时，通过内部总线，数据字被

并行地传入移位寄存器，而后串行地移出到 MOSI 脚上；MSB 在先还是 LSB 在先，取决于 SPIx_CR1 寄存器中的 LSBFIRST 位的设置。数据从发送缓冲器传输到移位寄存器时 TXE 标志将被置位，如果设置了 SPIx_CR2 寄存器中的 TXEIE位，将产生中断。

数据接收过程：

对于接收器来说，当数据传输完成时：

- 传送移位寄存器里的数据到接收缓冲器，并且 RXNE 标志被置位。
- 如果设置了 SPIx_CR2 寄存器中的 RXNEIE 位，则产生中断。

在最后采样时钟沿，RXNE 位被设置，在移位寄存器中接收到的数据字被传送到接收缓冲器。读 SPIx_DR 寄存器时，SPI 设备返回接收缓冲器中的数据。读 SPIx_DR 寄存器将清除 RXNE 位。一旦传输开始，如果下一个将发送的数据被放进了发送缓冲器，就可以维持一个连续的传输流。在试图写发送缓冲器之前，需确认 TXE 标志应该为'1'。

注意：当一个主机与多个从机通信时，从机需要在传输的某些时刻不被选中，那么该从机的 NSS 引脚必须被配置成 GPIO，或者使用其他某个 GPIO 用来被软件翻转。

24.4.4 配置 SPI 为单工通信

SPI 模块能够以两种配置工作于单工方式：

- 1 条时钟线和 1 条双向数据线（BIDIMODE=1）：

  设置 SPIx_CR1 寄存器中的 BIDIMODE 位而启用此模式。在这个模式下，SCK 引脚作为时钟，主设备使用 MOSI 引脚而从设备使用 MISO 引脚作为数据通信。传输的方向由 SPIx_CR1 寄存器里的 BIDIOE 控制，当这个位是'1'的时候，数据线是输出，否则是输入。

- 1 条时钟线和 1 条单向数据线（只接收或只发送）（BIDIMODE=0）：

  在这个模式下，SPI 模块可以或者作为只发送，或者作为只接收：

  - 只发送模式类似于全双工模式（BIDIMODE=0，RXONLY=0）：数据在发送引脚（主模式时是 MOSI、从模式时是 MISO）上传输，而接收引脚（主模式时是 MISO、从模式时是 MOSI）可以作为通用的 I/O 使用。此时，软件不必理会接收缓冲器中的数据（如果读出数据寄存器，它不包含任何接收数据）。
  - 在只接收模式，可以通过设置 SPIx_CR1 寄存器的 RXONLY 位而关闭 SPI 的输出功能；此时，发送引脚（主模式时是 MOSI、从模式时是 MISO）被释放，可以作为其它功能使用。

    配置并使能 SPI 模块为只接收模式的方式是：

    - 在主模式时，一旦使能 SPI，通信立即启动，当清除 SPE 位时立即停止当前的接收。在此模式下，不必读取 BSY 标志，在 SPI 通信期间这个标志始终为'1'。
    - 在从模式时，只要 NSS 被拉低（或在 NSS 软件模式时，SSI 位为'0'）同时 SCK 有时钟脉冲，SPI 就一直在接收。

24.4.5 数据发送与接收过程

24.4.5.1 接收与发送缓冲器

在接收时，接收到的数据被存放在一个内部的接收缓冲器中；在发送时，在被发送之前，数据将首先被存放在一个内部的发送缓冲器中。

对 SPIx_DR 寄存器的读操作，将返回接收缓冲器的内容；写入 SPIx_DR 寄存器的数据将被写入发送缓冲器中。

24.4.5.2 主模式下开始传输

- 全双工模式（BIDIMODE=0 并且 RXONLY=0）：
  - 当写入数据到 SPIx_DR 寄存器（发送缓冲器）后，传输开始；
  - 在传送第一位数据的同时，数据被并行地从发送缓冲器传送到 8 位的移位寄存器中，然后按顺序被串行地移位送到 MOSI 引脚上；
  - 与此同时，在 MISO 引脚上接收到的数据，按顺序被串行地移位进入 8 位的移位寄存器中，然后被并行地传送到 SPIx_DR 寄存器（接收缓冲器）中。
- 单向的只接收模式（BIDIMODE=0 并且 RXONLY=1）：
  - SPE=1 时，传输开始；
  - 只有接收器被激活，在 MISO 引脚上接收到的数据，按顺序被串行地移位进入 8 位的移位寄存器中，然后被并行地传送到 SPIx_DR 寄存器（接收缓冲器）中。
- 双向模式，发送时（BIDIMODE=1 并且 BIDIOE=1）
  - 当写入数据到 SPIx_DR 寄存器（发送缓冲器）后，传输开始；
  - 在传送第一位数据的同时，数据被并行地从发送缓冲器传送到 8 位的移位寄存器中，然后按顺序被串行地移位送到 MOSI 引脚上；
  - 不接收数据。
- 双向模式，接收时（BIDIMODE=1 并且 BIDIOE=0）
  - SPE=1 并且 BIDIOE=0 时，传输开始；
  - 在 MOSI 引脚上接收到的数据，按顺序被串行地移位进入 8 位的移位寄存器中，然后被并行地传送到 SPIx_DR 寄存器（接收缓冲器）中。
  - 不激活发送器，没有数据被串行地送到 MOSI 引脚上。

24.4.5.3 从模式下开始传输

- 全双工模式（BIDIMODE=0 并且 RXONLY=0）
  - 当从设备接收到时钟信号并且第一个数据位出现在它的 MOSI 时，数据传输开始，随后的数据位依次移动进入移位寄存器；
  - 与此同时，在传输第一个数据位时，发送缓冲器中的数据被并行地传送到 8 位的移位寄存器，随后被串行地发送到 MISO 引脚上。软件必须保证在 SPI 主设备开始数据传输之前在发送寄存器中写入要发送的数据。
- 单向的只接收模式（BIDIMODE=0 并且 RXONLY=1）
  - 当从设备接收到时钟信号并且第一个数据位出现在它的 MOSI 时，数据传输开始，随 后数据位依次移动进入移位寄存器；
  - 不启动发送器，没有数据被串行地传送到 MISO 引脚上。
- 双向模式，发送时（BIDIMODE=1 并且 BIDIOE=1）
  - 当从设备接收到时钟信号并且发送缓冲器中的第一个数据位被传送到 MISO 引脚上的时候，数据传输开始；
  - 在第一个数据位被传送到 MISO 引脚上的同时，发送缓冲器中要发送的数据被平行地传送到 8 位的移位寄存器中，随后被串行地发送到 MISO 引脚上。软件必须保证在 SPI 主设备开始数据传输之前在发送寄存器中写入要发送的数据；
  - 不接收数据。
- 双向模式，接收时（BIDIMODE=1 并且 BIDIOE=0）
  - 当从设备接收到时钟信号并且第一个数据位出现在它的 MOSI 时，数据传输开始；
  - 从 MISO 引脚上接收到的数据被串行地传送到 8 位的移位寄存器中，然后被平行地传送到

SPIx_DR 寄存器（接收缓冲器）；

- 不启动发送器，没有数据被串行地传送到 MISO 引脚上。

24.4.5.4 处理数据的发送与接收

当数据从发送缓冲器传送到移位寄存器时，设置 TXE 标志（发送缓冲器空），它表示内部的发送缓冲器可以接收下一个数据；如果在 SPIx_CR2 寄存器中设置了 TXEIE 位，则此时会产生一个中断；写入 SPIx_DR 寄存器即可清除 TXE 位。

注意：在写入发送缓冲器之前，软件必须确认 TXE 标志为1’，否则新的数据会覆盖已经在发送缓冲器中的数据。

在采样时钟的最后一个边沿，当数据被从移位寄存器传送到接收缓冲器时，设置 RXNE 标志（接收缓冲器非空）；它表示数据已经就绪，可以从 SPIx_DR 寄存器读出；如果在 SPIx_CR2 寄存器中设置了 RXNEIE 位，则此时会产生一个中断；读出 SPIx_DR 寄存器即可清除 RXNIE 标志位。

在一些配置中，传输最后一个数据时，可以使用 BSY 标志等待数据传输的结束。

24.4.5.5 主或从模式下（BIDIMODE=0 并且 RXONLY=0）全双工发送和接收过程模式

软件必须遵循下述过程，发送和接收数据（参见 图 24-4 和图 24-5）：

1. 设置 SPE 位为’1’，使能 SPI 模块；
2. 在 SPIx_DR 寄存器中写入第一个要发送的数据，这个操作会清除 TXE 标志；
3. 等待 TXE=1，然后写入第二个要发送的数据。等待 RXNE=1，然后读出 SPIx_DR 寄存器并获得第一个接收到的数据，读 SPIx_DR 的同时清除了 RXNE 位。重复这些操作，发送后续 的数据同时接收 n-1 个数据；
4. 等待 RXNE=1，然后接收最后一个数据；
5. 等待 TXE=1，在 BSY=0 之后关闭 SPI 模块。也可以在响应 RXNE 或 TXE 标志的上升沿产生的中断的处理程序中实现这个过程。

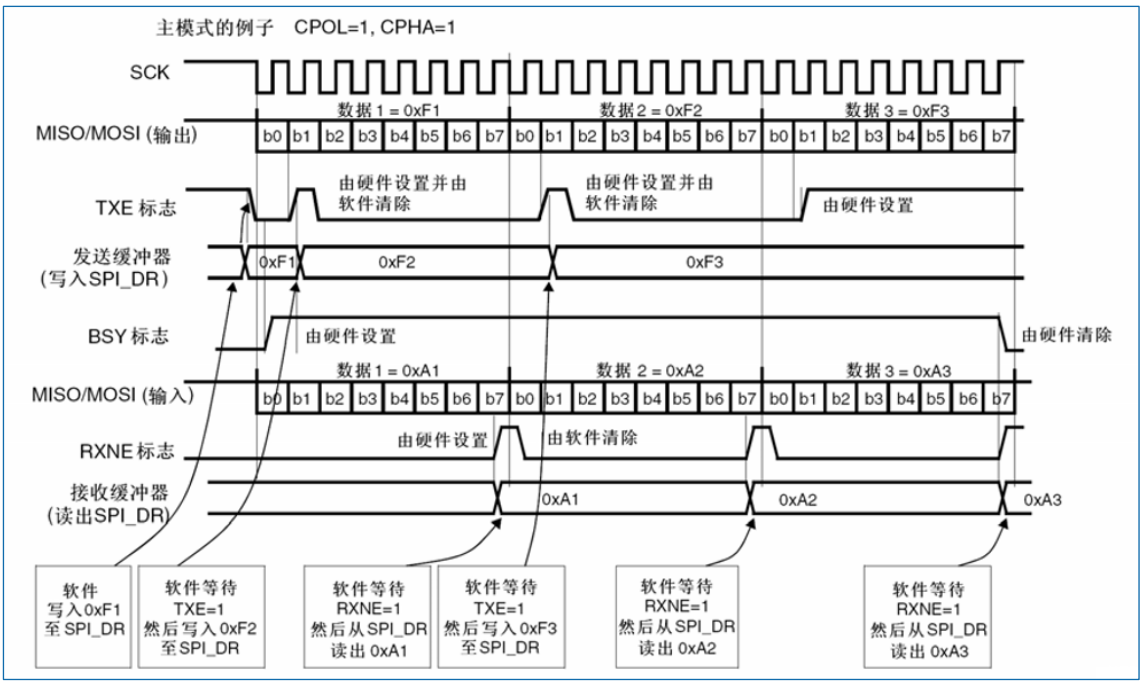


图 24-4 主模式、全双工模式下（BIDIMODE=0 并且 RXONLY=0）连续传输时，TXE/RXNE/BSY 的变化示意图

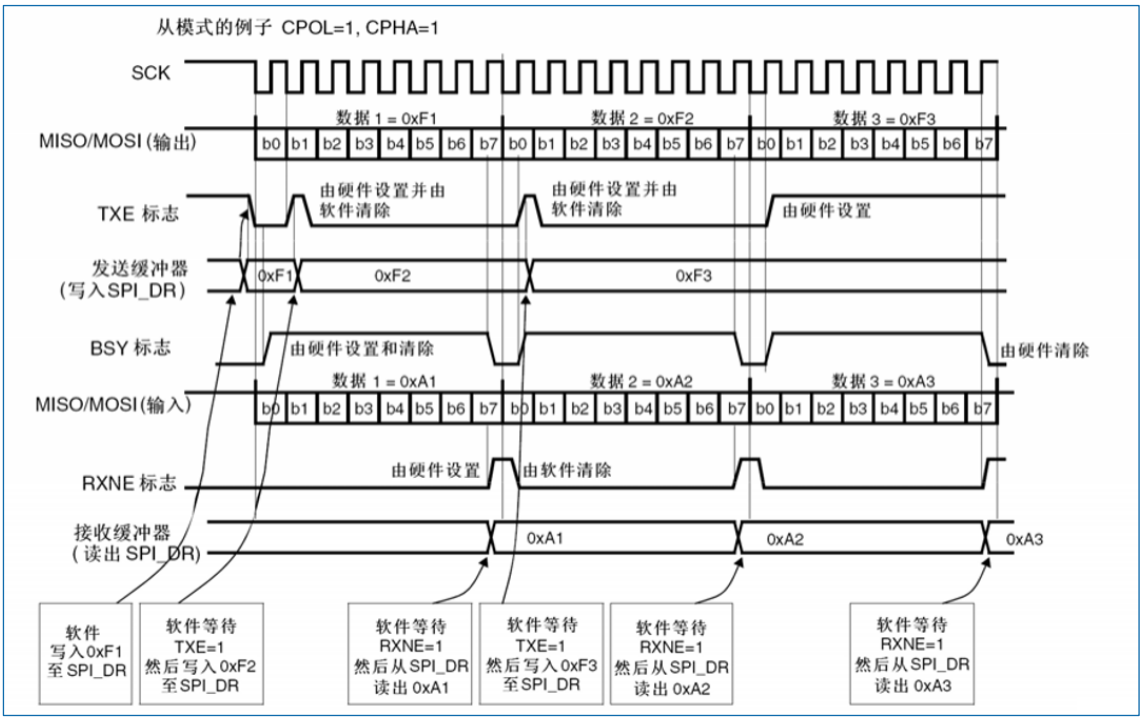


图 24-5 从模式、全双工模式下（BIDIMODE=0 并且 RXONLY=0）连续传输时，TXE/RXNE/BSY 的变化示意图

24.4.5.6 只发送过程（BIDIMODE=0 并且 RXONLY=0）

在此模式下，传输过程可以简要说明如下，并且使用 BSY 位等待传输的结束（参见图 24-6 和图 24-7）：

1. 设置 SPE 位为'1'，使能 SPI 模块；
2. 在 SPIx_DR 寄存器中写入第一个要发送的数据，这个操作会清除 TXE 标志；
3. 等待 TXE=1，然后写入第二个要发送的数据。重复这个操作，发送后续的数据；
4. 写入最后一个数据到 SPIx_DR 寄存器之后，等待 TXE=1；然后等待 BSY=0，这表示最后一个数据的传输已经完成。

此过程也可以在响应 TXE 标志的上升沿产生的中断的处理程序中得以实现。

注意：

对于不连续的传输，在写入 SPIx_DR 寄存器的操作与设置 BSY 位之间有 2 个 APB 时钟周期的延迟，因此在只发送模式下，写入最后一个数据后，最好先等待 TXE=1，然后再等待 BSY=0。

只发送模式下，在传输 2 个数据之后，由于不会读出接收到的数据，SPIx_SR 寄存器中的 OVR 位会变为'1'。

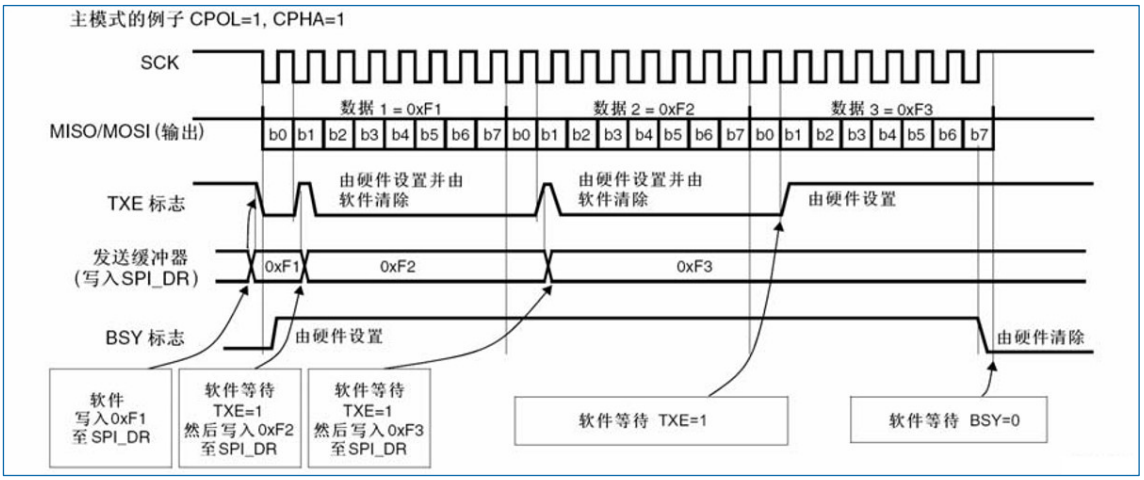


图 24-6 主设备只发送模式（BIDIMODE=0 并且 RXONLY=0）下连续传输时，TXE/BSY 变化示意图

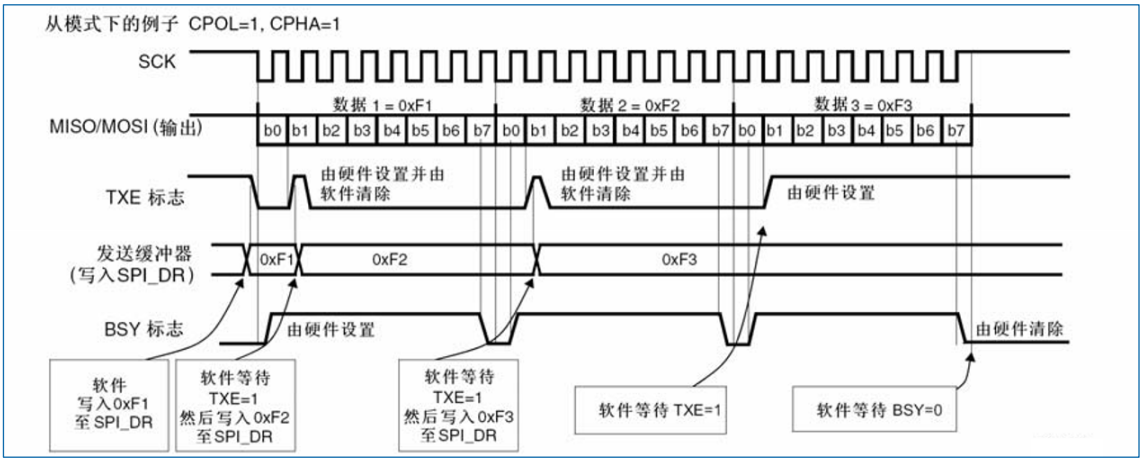


图 24-7 从设备只发送模式（BIDIMODE=0 并且 RXONLY=0）下连续传输时，TXE/BSY 变化示意图

24.4.5.7 双向发送过程（BIDIMODE=1 并且 BIDIOE=1）

在此模式下，操作过程类似于只发送模式，不同的是：在使能 SPI 模块之前，需要在 SPIx_CR1 寄存器中同时设置 BIDIMODE 和 BIDIOE 位为'1'。

24.4.5.8 单向只接收模式（BIDIMODE=0 并且 RXONLY=1）

在此模式下，传输过程可以简要说明如下：

1. 在 SPIx_CR1 寄存器中，设置 RXONLY=1；
2. 设置 SPE=1，使能 SPI 模块：
   - 主模式下，立刻产生 SCK 时钟信号，在关闭 SPI（SPE=0）之前，不断地接收串行数据；
   - 从模式下，当 SPI 主设备拉低 NSS 信号并产生 SCK 时钟时，接收串行数据。
3. 待 RXNE=1，然后读出 SPIx_DR 寄存器以获得收到的数据（同时会清除 RXNE 位）。重复这个操作接收所有数据。

该过程也可以在响应 RXNE 标志的上升沿产生的中断的处理程序中得以实现。

注意：如果在最后一个数据传输结束后关闭 SPI 模块，请按照本章节关闭 SPI 节的建议操作。

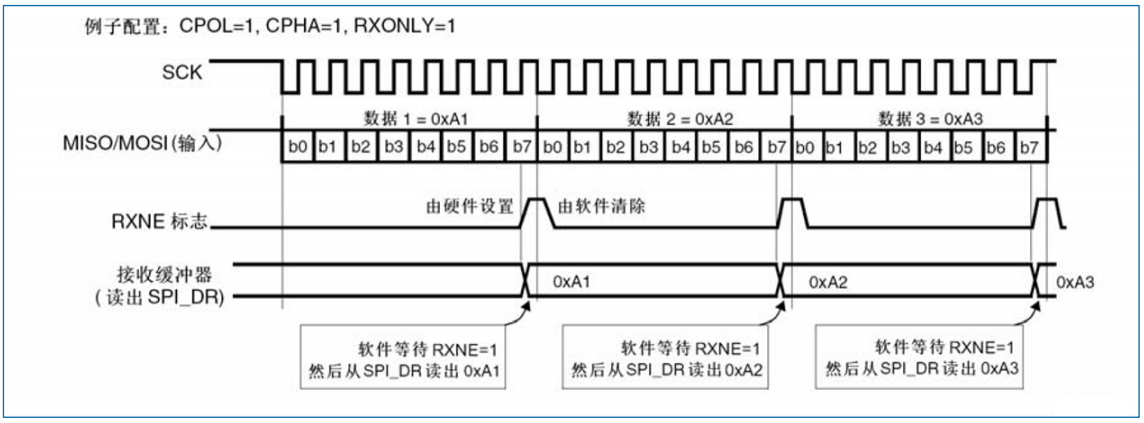


图 24-8 只接收模式（BIDIMODE=0 并且 RXONLY=1）下连续传输时，RXNE 变化示意图

24.4.5.9 单向接收过程（BIDIMODE=1 并且 BIDIOE=0）

在此模式下，操作过程类似于只接收模式，不同的是：在使能 SPI 模块之前，需要在 SPIx_CR1 寄存器中设置 BIDIMODE 为'1'并清除 BIDIOE 位为'0'。

24.4.5.10 连续和非连续传输

当在主模式下发送数据时，如果软件足够快，能够在检测到每次 TXE 的上升沿（或 TXE 中断），并立即在正在进行的传输结束之前写入 SPIx_DR 寄存器，则能够实现连续的通信；此时，在每个数据项的传输之间的 SPI 时钟保持连续，同时 BSY 位不会被清除。

如果软件不够快，则会导致不连续的通信；这时，在每个数据传输之间会被清除（见下图）。在主模式的只接收模式下（RXONLY=1），通信总是连续的，而且 BSY 标志始终为'1'。

在从模式下，通信的连续性由 SPI 主设备决定。不管怎样，即使通信是连续的，BSY 标志会在每个数据项之间至少有一个 SPI 时钟周期为低（参见 图 24-7）。

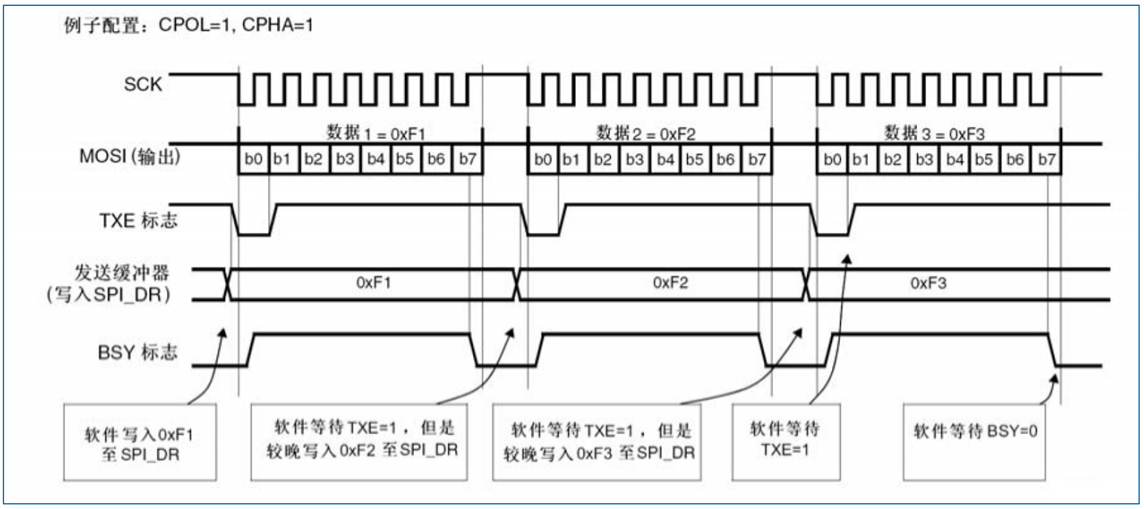


图 24-9 非连续传输发送（BIDIMODE=0 并且 RXONLY=0）时，TXE/BSY 变化示意图

24.4.6 CRC 计算

CRC 校验用于保证全双工通信的可靠性。数据发送和数据接收分别使用单独的 CRC 计算器。通过对每一个接收位进行可编程的多项式运算来计算 CRC。CRC 的计算是在由 SPIx_CR1 寄存器中CPHA 和 CPOL 位定义的采样时钟边沿进行的。

注意：该 SPI 接口提供了两种 CRC 计算方法，取决于所选的发送和/或接收的数据帧格式：8 位数据帧采用 CR8；16 位数据帧采用 CRC16。

CRC 计算是通过设置 SPIx_CR1 寄存器中的 CRCEN 位启用的。设置 CRCEN 位时同时复位 CRC 寄存器（SPIx_RXCRCR 和 SPIx_TXCRCR）。在全双工和单发送模式下，如果传输是由软件控制（CPU 模式），需要在最后一个数据被写入 SPIx_DR 寄存器后立即写 1 置位 CRCNEXT。在最后一个数据传输完成后，SPIx_TXCRCR 的值会被发出。

在单接收模式下，如果传输是由软件控制（CPU 模式），需要在接收完倒数第二个数据的时候置位 CRCNEXT 位。CRC 数据会在最后一个数据之后接收，并且进行 CRC 结果比对。

在数据和 CRC 传输完成后，如果接收到的 CRC 数据与 SPIx_RXCRCR 的内容不匹配，则 SPIx_SR 寄存器的 CRCERR 标志位会被硬件置 1。

如果在 TX 缓冲器中还有数据，CRC 的数值仅在数据字节传输结束后传送。在传输 CRC 期间，

CRC 计算器关闭，寄存器的数值保持不变。

SPI 通信可以通过以下步骤使用 CRC：

1. 设置 CPOL、CPHA、LSBFirst、BR、SSM、SSI 和 MSTR 的值；
2. 在 SPIx_CRCPR 寄存器输入多项式；
3. 通过设置 SPIx_CR1 寄存器 CRCEN 位使能 CRC 计算，该操作也会清除寄存器 SPIx_RXCRCR 和 SPIx_TXCRC；
4. 设置 SPIx_CR1 寄存器的 SPE 位启动 SPI 功能；
5. 启动通信并且维持通信，直到只剩最后一个字节或者半字；
   - 在全双工和单发送模式下，如果是软件控制传输，那么需要在传输最后一个字节或者半字时通过置位 SPIx_CR1 中的 CRCNEXT 位来指示在最后一个数据传输完成后会传输 CRC 值；
   - 在单接收模式下，需要在接收完倒数第二个数据开始接收最后一个数据的时候就置位 CRCNEXT 位，让 SPI 在接收完最后一个数据后进入 CRC 阶段。在 CRC 阶段的时候内部 CRC 计算会暂停。
6. 当最后一个字节或半字被发送后，SPI 进入 CRC 发送和校验阶段。在全双工和单接收模式下，接收到的 CRC 与 SPIx_RXCRCR 值进行比较。如果比较结果不等，则 SPIx_SR 上的 CRCERR 标志位被置起。如果设置了 SPIx_CR2 寄存器的 ERRIE 时，则同时会产生中断。

注意：

当 SPI 模块处于从设备模式时，请注意在时钟稳定之后再使能 CRC 计算，否则可能会得到错误的 CRC 计算结果。事实上，只要设置了 CRCEN 位，只要在 SCK 引脚上有输入时钟，不管 SPE 位的状态，都会进行 CRC 的计算。

当 SPI 时钟频率较高时，用户在发送 CRC 时必须小心。在 CRC 传输期间，使用 CPU 的时间应尽可能少；为了避免在接收最后的数据和 CRC 时出错，在发送 CRC 过程中应禁止函数调用。必须在发送/接收最后一个数据之前完成设置 CRCNEXT 位的操作。

当 SPI 时钟频率较高时，因为 CPU 的操作会影响 SPI 的带宽，建议采用 DMA 模式以避免 SPI 降低的速度。

当 SPI 被配置为从模式并且使用了 NSS 硬件模式，NSS 引脚应该在数据传输和 CRC 传输期间保持为低。

当配置 SPI 为从模式并且使用 CRC 的功能，即使 NSS 信号为高时，如果 SCK 引脚上有时钟脉冲，则 CRC 计算会继续执行。例如在多从机的环境下，当主设备交替地与不同的从设备进行通信时，就会出现这种情况。

在不选中一个从设备（NSS 信号为高）转换到选中一个新的从设备（NSS 信号为低）的时候，为了

保持主从设备端下次 CRC 计算结果的同步，应该清除主从两端的 CRC 数值。

按照下述步骤清除 CRC 数值：

1. 关闭 SPI 模块（SPE=0）；
2. 清除 CRCEN 位为'0'；
3. 设置 CRCEN 位为'1'；
4. 使能 SPI 模块（SPE=1）。

24.4.7 状态标志

应用程序通过 3 个状态标志可以完全监控 SPI 总线的状态。

24.4.7.1 发送缓冲器空闲标志（TXE）

此标志为'1'时表明发送缓冲器为空，可以写下一个待发送的数据进入缓冲器中。当写入 SPIx_DR 时，TXE 标志被清除。

24.4.7.2 接收缓冲器非空（RXNE）

此标志为'1'时表明在接收缓冲器中包含有效的接收数据。读 SPI 数据寄存器可以清除此标志。

24.4.7.3 忙标志（BSY）

BSY 标志由硬件设置与清除（写入此位无效果），此标志表明 SPI 通信层的状态。

当它被设置为'1'时，表明 SPI 正忙于通信，但有一个例外：在主模式的双向接收模式下（MSTR=1、BDM=1 并且 BDOE=0），在接收期间 BSY 标志保持为低。

在软件要关闭 SPI 模块并进入停机模式（或关闭设备时钟）之前，可以使用 BSY 标志检测传输是否结束，这样可以避免破坏最后一次传输，因此需要严格按照下述过程执行。

BSY 标志还可以用于在多主系统中避免写冲突。

除了主模式的双向接收模式（MSTR=1、BDM=1 并且 BDOE=0），当传输开始时，BSY 标志被置'1'。

以下情况时此标志将被清除为'0'：

- 当传输结束（主模式下，如果是连续通信的情况例外）；
- 当关闭 SPI 模块；
- 当产生主模式失效（MODF=1）。

如果通信不是连续的，则在每个数据项的传输之间，BSY 标志为低。

当通信是连续时：

- 主模式下：在整个传输过程中，BSY 标志保持为高；
- 从模式下：在每个数据项的传输之间，BSY 标志在一个 SPI 时钟周期中为低。

注意：不要使用 BSY 标志处理每一个数据项的发送和接收，最好使用 TXE 和 RXNE 标志。

24.4.8 关闭 SPI

当通讯结束，可以通过关闭 SPI 模块来终止通讯。清除 SPE 位即可关闭 SPI。

在某些配置下，如果再传输还未完成时，就关闭 SPI 模块并进入停机模式，则可能导致当前的传输被破坏，而且 BSY 标志也变得不可信。

为了避免发生这种情况，关闭 SPI 模块时，建议按照下述步骤操作：

- 在主或从模式下的全双工模式（BIDIMODE=0，RXONLY=0）

  A. 等待 RXNE=1 并接收最后一个数据；

  B. 等待 TXE=1；

  C. 等待 BSY=0；

  D. 关闭 SPI（SPE=0），最后进入停机模式（或关闭该模块的时钟）。

- 在主或从模式下的单向只发送模式（BIDIMODE=0，RXONLY=0）或双向的发送模式（BIDIMODE=1，BIDIOE=1）

  在 SPIx_DR 寄存器中写入最后一个数据后：

  A. 等待 TXE=1；

  B. 等待 BSY=0；

  C. 关闭 SPI（SPE=0），最后进入停机模式（或关闭该模块的时钟）。

- 在主或从模式下的单向只接收模式（MSTR=1，BIDIMODE=0，RXONLY=1）或双向 的接收模式（MSTR=1，BIDIMODE=1，BIDIOE=0）

  这种情况需要特别处理，以保证 SPI 不会开始一次新的传输：

  A. 等待倒数第二个（第 n-1 个）RXNE=1；

  B. 在关闭 SPI（SPE=0）之前等待一个 SPI 时钟周期（使用软件延迟）；

  C. 在进入停机模式（或关闭该模块的时钟）之前等待最后一个 RXNE=1。

注意：在主模式下的单向只发送模式（MSTR=1，BDM=1，BDOE=0）时，传输过程中 BSY 标志始终为低。

- 在从模式下的只接收模式（MSTR=0，BIDIMODE=0，RXONLY=1）或双向的接收模式（MSTR=0，BIDIMODE=1，BIDIOE=0）：

  A. 可以在任何时候关闭 SPI（SPE=0），SPI 会在当前的传输结束后被关闭；

  B. 如果希望进入停机模式，在进入停机模式（或关闭该模块的时钟）之前必须首先等待 BSY=0。

24.4.9 利用 DMA 的 SPI 通信

为了达到最大通信速度，需要及时往 SPI 发送缓冲器填数据，同样接收缓冲器中的数据也必须及时读走以防止溢出。为了方便高速率的数据传输，SPI 实现了一种采用简单的请求/应答的 DMA 机制。

当 SPIx_CR2 寄存器上的对应使能位被设置时，SPI 模块可以发出 DMA 传输请求。发送缓冲器和接收缓冲器亦有各自的 DMA 请求：

- 发送时，在每次 TXE 被设置为'1'时发出 DMA 请求，DMA 控制器则写数据至 SPIx_DR 寄存器，TXE 标志因此而被清除。
- 接收时，在每次 RXNE 被设置为'1'时发出 DMA 请求，DMA 控制器则从 SPIx_DR 寄存器读出数据，RXNE 标志因此而被清除。

当只使用 SPI 发送数据时，只需使能 SPI 的发送 DMA 通道。此时，因为没有读取收到的数据，OVR 被置为'1'。当只使用 SPI 接收数据时，只需使能 SPI 的接收 DMA 通道。

在发送模式下，当 DMA 已经传输了所有要发送的数据（DMA_ISR 寄存器的 TCIF 标志变为'1'）后，可以通过监视 BSY 标志以确认 SPI 通信结束，这样可以避免在关闭 SPI 或进入停止模式时，破坏最后一个数据的传输。因此软件需要先等待 TXE=1，然后等待 BSY=0。

注意：在不连续的通信中，在写数据到 SPIx_DR 的操作与 BSY 位被置为'1'之间，有 2 个 APB 时钟周期的延迟，因此，在写完最后一个数据后需要先等待 TXE=1 再等待 BSY=0。

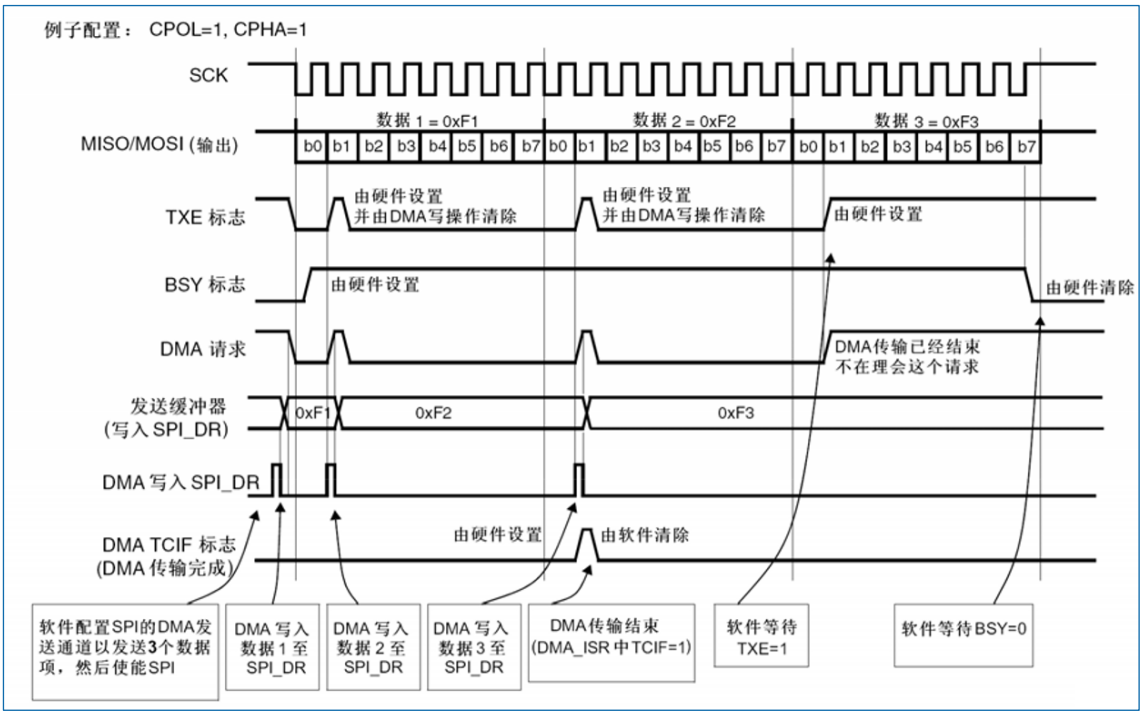


图 24-10 使用 DMA 发送

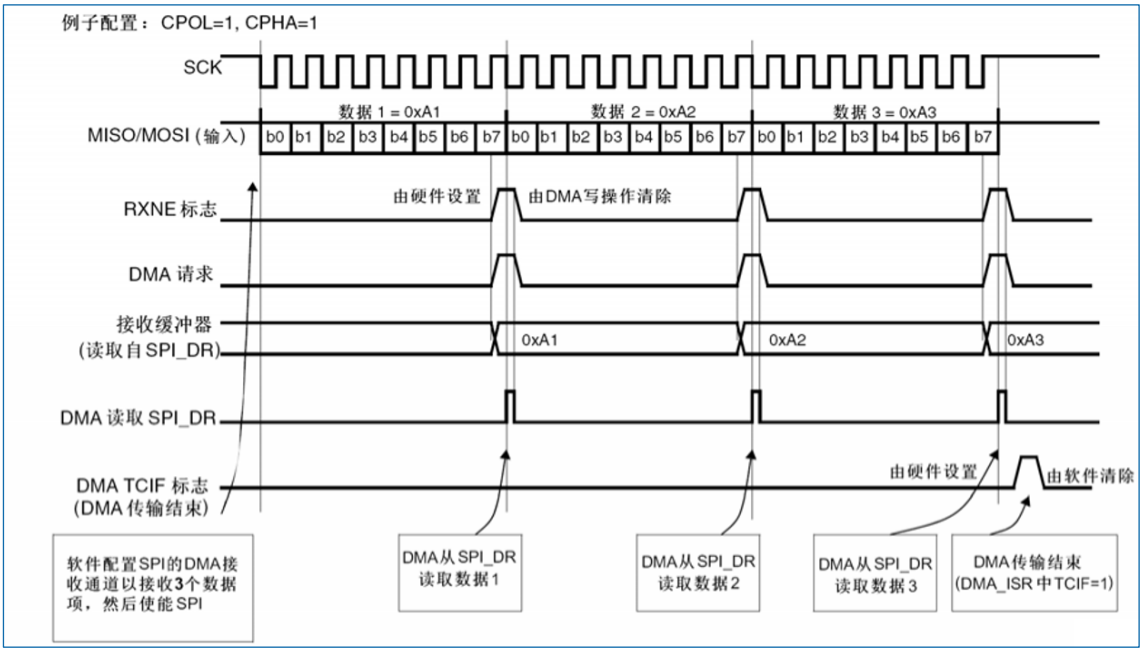


图 24-11 使用 DMA 接收

带 CRC 的 DMA 功能

当使能 SPI 使用 CRC 检验并且启用 DMA 模式时，在通信结束时，CRC 字节的发送和接收是自动完成的。数据和 CRC 传输结束时，SPIx_SR 寄存器的 CRCERR 标志为’1’表示在传输期间发生错误。

24.4.10 错误标志

24.4.10.1 主模式失效错误（MODF）

主模式失效仅发生在：NSS 引脚硬件模式管理下，主设备的 NSS 脚被拉低；或者在 NSS 引脚软件模式管理下，SSI 位被置为’0’时；MODF 位被自动置位。

主模式失效对 SPI 设备有以下影响：

- MODF 位被置为’1’，如果设置了 ERRIE 位，则产生 SPI 中断；
- SPE 位被清为’0’。这将停止一切输出，并且关闭 SPI 接口；
- MSTR 位被清为’0’，因此强迫此设备进入从模式。

下面的步骤用于清除 MODF 位：

1. 当 MODF 位被置为’1’时，执行一次对 SPIx_SR 寄存器的读或写操作；
2. 然后写 SPIx_CR1 寄存器。

在有多个 MCU 的系统中，为了避免出现多个从设备的冲突，必须先拉高该主设备的 NSS 脚，再对 MODF 位进行清零。在完成清零之后，SPE 和 MSTR 位可以恢复到它们的原始状态。

出于安全的考虑，当 MODF 位为’1’时，硬件不允许设置 SPE 和 MSTR 位。通常配置下，从设备的 MODF 位不能被置为’1’。然而，在多主配置里，一个设备可以在设置了 MODF 位的情况下，处于从设备模式；此时，MODF 位表示可能出现了多主冲突。中断程序可以执行一个复位或返回到默认状态来从错误状态中恢复。

24.4.10.2 溢出错误（OVR）

当主设备已经发送了数据字节，而从设备还没有清除前一个数据字节产生的 RXNE 时，即为溢出错误。当产生溢出错误时，OVR 位被置为’1’；当设置了 ERRIE 位时，则产生中断。

此时，接收器缓冲器的数据不是主设备发送的新数据，读 SPIx_DR 寄存器返回的是之前未读的数据，所有随后传送的数据都被丢弃。

依次读出 SPIx_DR 寄存器和 SPIx_SR 寄存器可将 OVR 清除。

24.4.10.3 CRC 错误（CRCERR）

当设置了 SPIx_CR1 寄存器上的 CRCEN 位时，CRC 错误标志用来核对接收数据的有效性。如果移位寄存器中接收到的值（发送方发送的 SPIx_TXCRCR 数值）与接收方 SPIx_RXCRCR 寄存器中的数值不匹配，则 SPIx_SR 寄存器上的 CRCERR 标志被置位为’1’。

24.4.11 SPI 中断

表 24-2 SPI 中断

中断事件	事件标志	使能控制位
发送缓冲器空标志	TXE	TXEIE
接收缓冲器非空标志	RXNE	RXNEIE
主模式失效事件	MODF	ERRIE
溢出错误	OVR	
CRC 错误标志	CRCERR	

24.5 I2S 功能描述

I2S 的框图如下图所示：

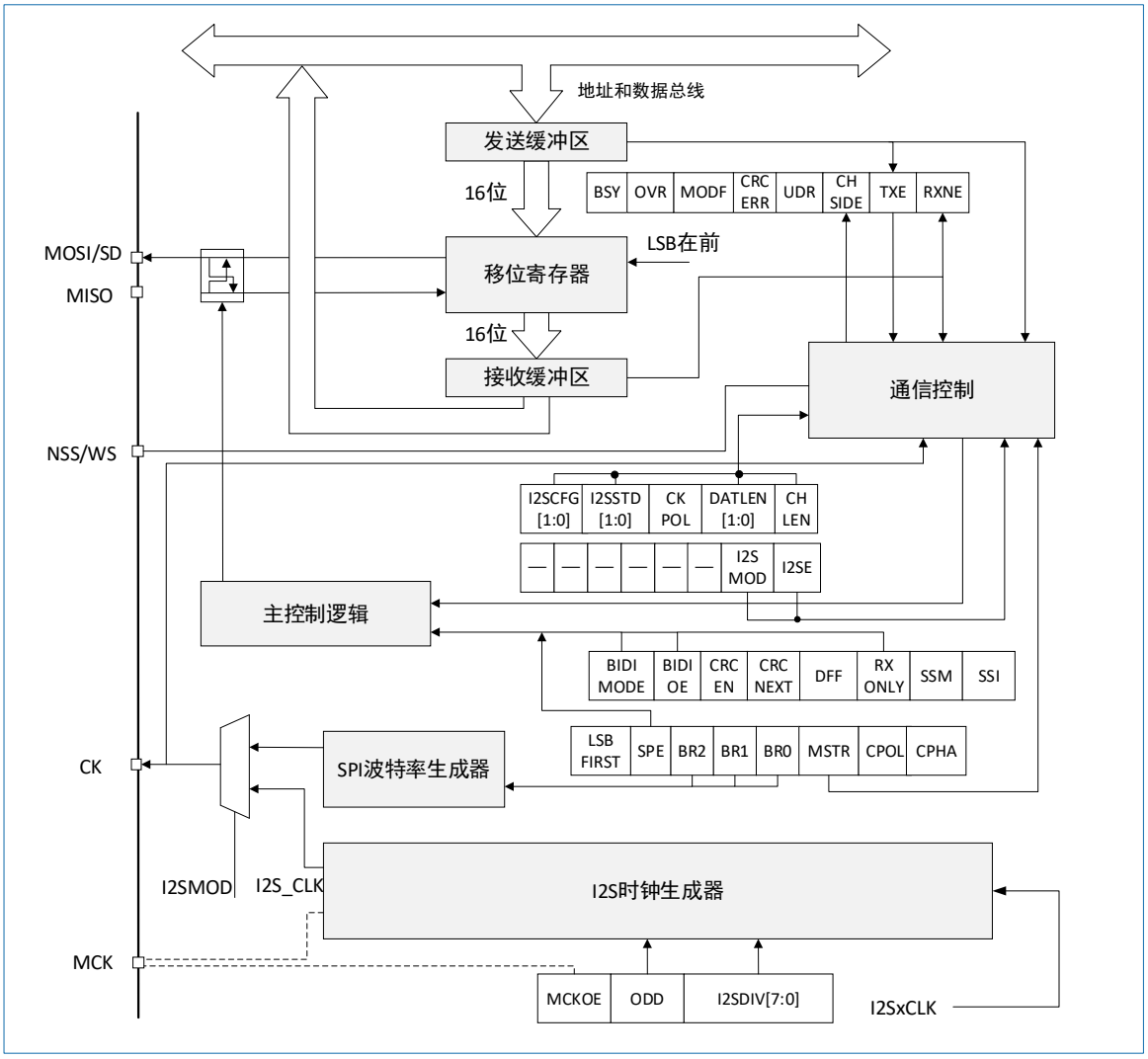


图 24-12 I2S 的框图

通过将寄存器 SPIx_I2SCFGR 的 I2SMOD 位置为'1'，即可使能 I2S 功能。此时，可以把 SPI 模块用作 I2S 音频接口。I2S 接口与 SPI 接口使用大致相同的引脚、标志和中断。

I2S 与 SPI 共用 3 个引脚：

- SD：串行数据（映射至 MOSI 引脚），用来发送和接收 2 路时分复用通道的数据；
- WS：字选（映射至 NSS 引脚），主模式下作为数据控制信号输出，从模式下作为输入；
- CK：串行时钟（映射至 SCK 引脚），主模式下作为时钟信号输出，从模式下作为输入。

在某些外部音频设备需要主时钟时，可以另有一个附加引脚输出时钟：

- MCK：主时钟（独立映射），在 I2S 配置为主模式，寄存器 SPIx_I2SPR 的 MCKOE 位为'1'时，作为输出额外的时钟信号引脚使用。输出时钟信号的频率预先设置为 256 ×Fs，其中 Fs 是音频信号的采样频率。

设置成主模式时，I2S 使用自身的时钟发生器来产生通信用的时钟信号。这个时钟发生器也是主时钟输出的时钟源。I2S 模式下有 2 个额外的寄存器，一个是与时钟发生器配置相关的寄存器 SPIx_I2SPR，另一个是 I2S 通用配置寄存器 SPIx_I2SCFGR（可设置音频标准、从/主模式、数据格式、数据包帧、时钟极性等参数）。

在 I2S 模式下不使用寄存器 SPIx_CR1 和所有的 CRC 寄存器。同样，I2S 模式下也不使用寄存器 SPIx_CR2 的 SSOE 位，和寄存器 SPIx_SR 的 MODF 位和 CRCERR 位。I2S 使用与 SPI 相同的寄存器 SPIx_DR 用作 16 位宽模式数据传输。

24.5.1 支持的音频协议

三线总线支持 2 个声道上音频数据的时分复用：左声道和右声道，但是只有一个 16 位寄存器用作发送或接收。因此，软件必须在对数据寄存器写入数据时，根据当前传输中的声道写入相应的数据；同样，在读取寄存器数据时，通过检查寄存器 SPIx_SR 的 CHSIDE 位来判明接收到的数据属于哪个声道。左声道总是先于右声道发送数据（CHSIDE 位在 PCM 协议下无意义）。

有四种可用的数据和包帧组合。数据可以按照以下四种数据格式发送：

- 16 位数据打包进 16 位帧
- 16 位数据打包进 32 位帧
- 24 位数据打包进 32 位帧
- 32 位数据打包进 32 位帧

在使用 16 位数据扩展到 32 位帧时，前 16 位（MSB）是有意义的数据，后 16 位（LSB）被强制为 0。该操作不需要软件干预，也不需要有 DMA 请求（仅需要一次读/写操作）。

24 位和 32 位数据帧需要 CPU 对寄存器 SPIx_DR 进行两次读或写操作，在使用 DMA 时，需要两次 DMA 传输。对于 24 位数据，扩展到 32 位后，最低 8 位由硬件置 0。

对于所有的数据格式和通讯标准，总是先发送最高位（MSB）。

I2S 接口支持四种音频标准，可以通过设置寄存器 SPIx_I2SCFGR 的 I2SSTD[1:0]位和 PCMSYNC 位来选择。

24.5.1.1 I2S 飞利浦标准

在此标准下，引脚 WS 用来指示正在发送的数据属于哪个声道。在发送第一位数据（MSB）前 1 个时钟周期，该引脚即为有效。

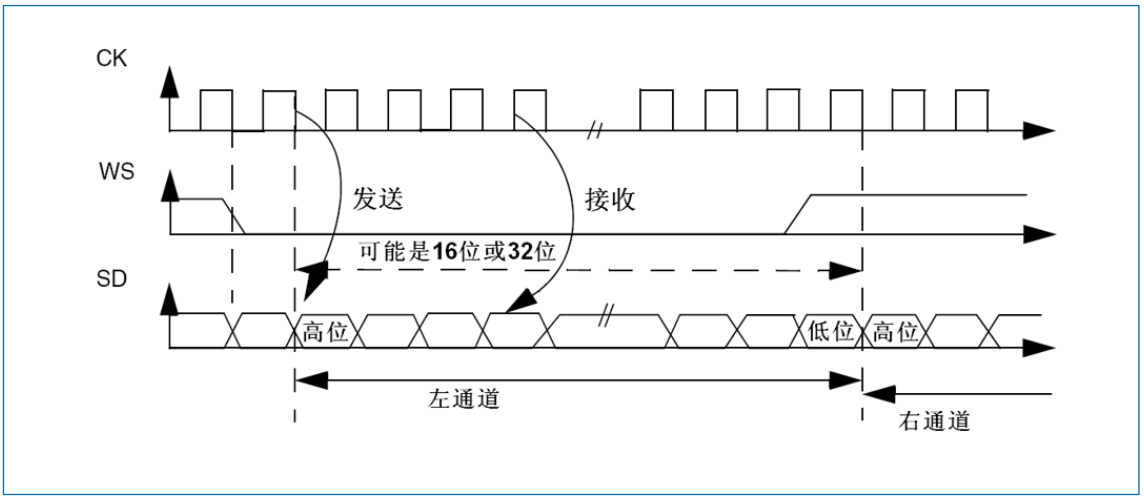


图 24-13 I2S 飞利浦协议波形（16/32 位全精度，CPOL=0）

发送方在时钟信号（CK）的下降沿改变数据，接收方在上升沿读取数据。WS 信号也在时钟信号的下降沿变化。

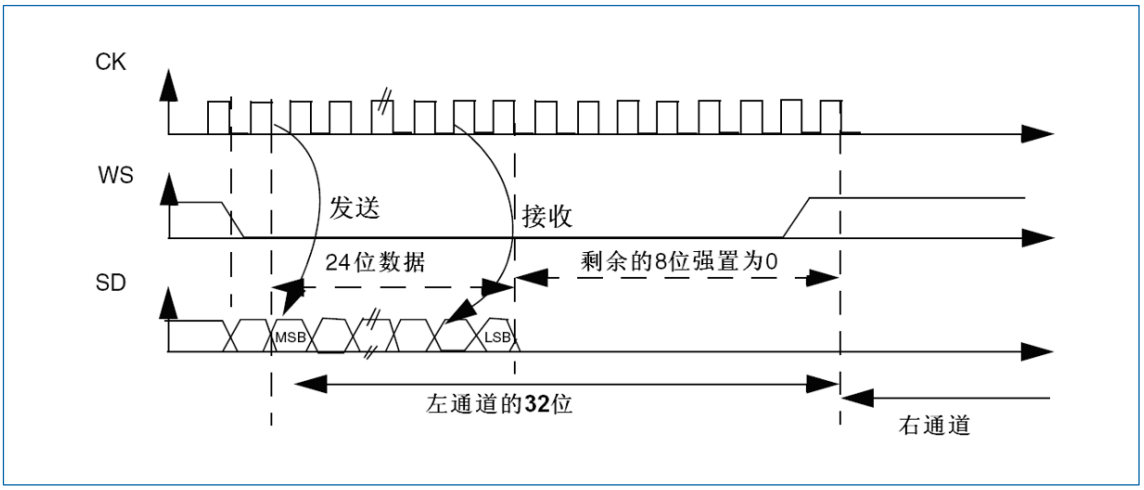


图 24-14 I2S 飞利浦协议标准波形（24 位帧，CPOL=0）

此模式需要对寄存器 SPIx_DR 进行 2 次读或写操作。

- 在发送模式下：

  如果需要发送 0x8EAA33（24 位）：

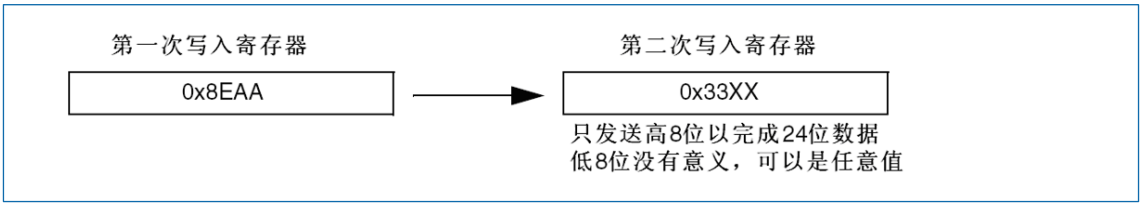


图 24-15 发送 0x8EAA33

- 在接收模式下：

如果接收 0x8EAA33：

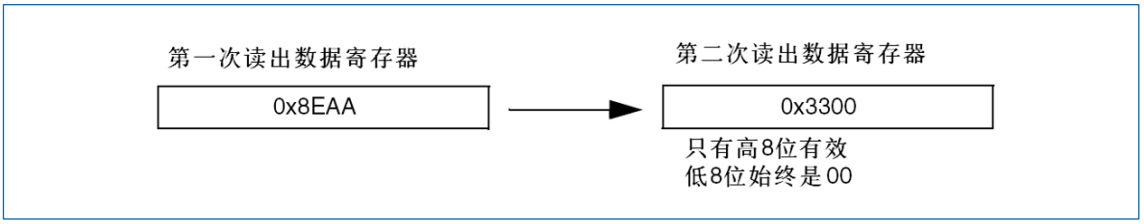


图 24-16 接收 0x8EAA33

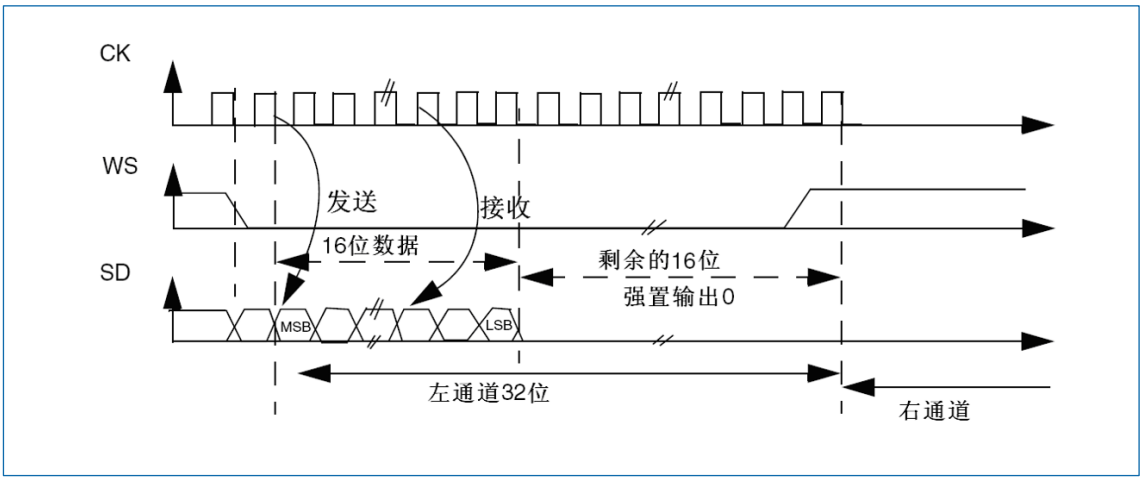


图 24-17 I2S 飞利浦协议标准波形（16 位扩展至 32 位包帧，CPOL=0）

在 I2S 配置阶段，如果选择将 16 位数据扩展到 32 位声道帧，只需要访问一次寄存器 SPIx_DR。用来

扩展到 32 位的低 16 位被硬件置为 0x0000。

如果待传输或者接收的数据是 0x76A3（扩展到 32 位是 0x76A30000），需要的操作如下图所示。

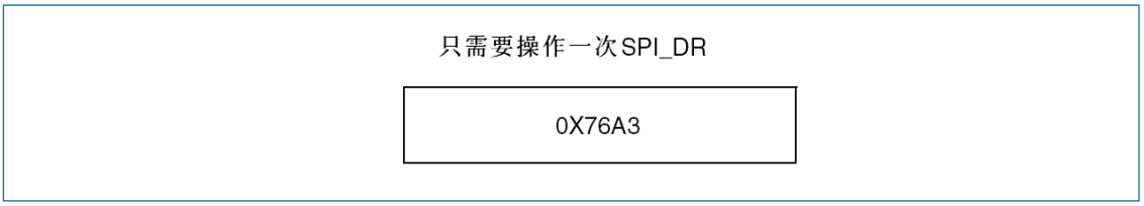


图 24-18 示例

在发送时需要将 MSB 写入寄存器 SPIx_DR；标志位 TXE 为'1'表示可以写入新的数据，如果允许了相应的中断，则可以产生中断。发送是由硬件完成的，即使还未发送出后 16 位的 0x0000，也会设置 TXE 并产生相应的中断。

接收时，每次收到高 16 位半字（MSB）后，标志位 RXNE 置'1'，如果允许了相应的中断，则可以产生中断。

这样，在 2 次读和写之间有更多的时间，可以防止下溢或者上溢的情况发生。

24.5.1.2 MSB 对齐标准

在此标准下，WS 信号和第一个数据位，即最高位（MSB）同时产生。

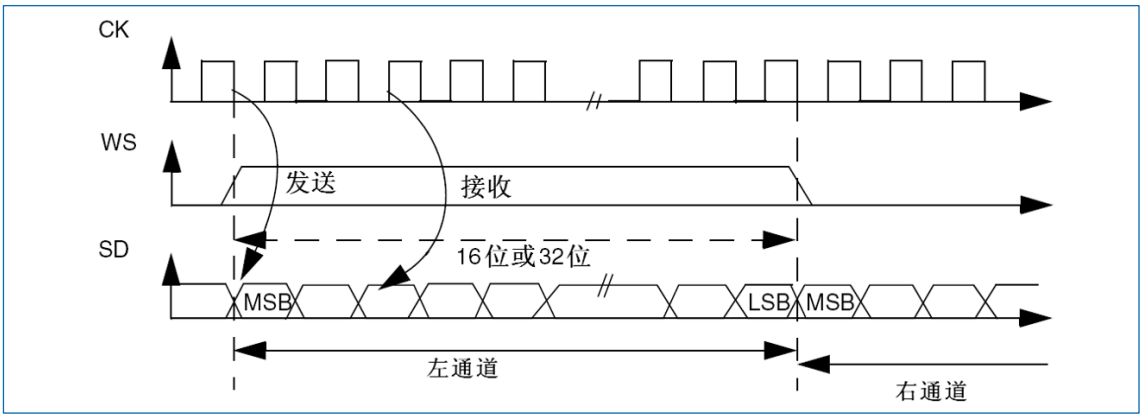


图 24-19 MSB 对齐 16 位或 32 位全精度，CPOL=0

发送方在时钟信号的下降沿改变数据；接收方是在上升沿读取数据。

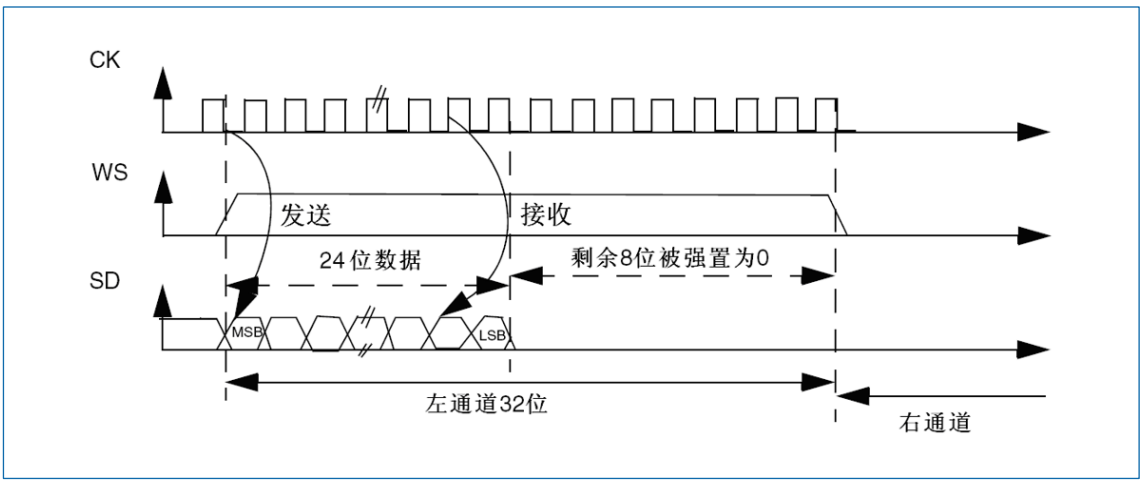


图 24-20 MSB 对齐 24 位数据，CPOL=0

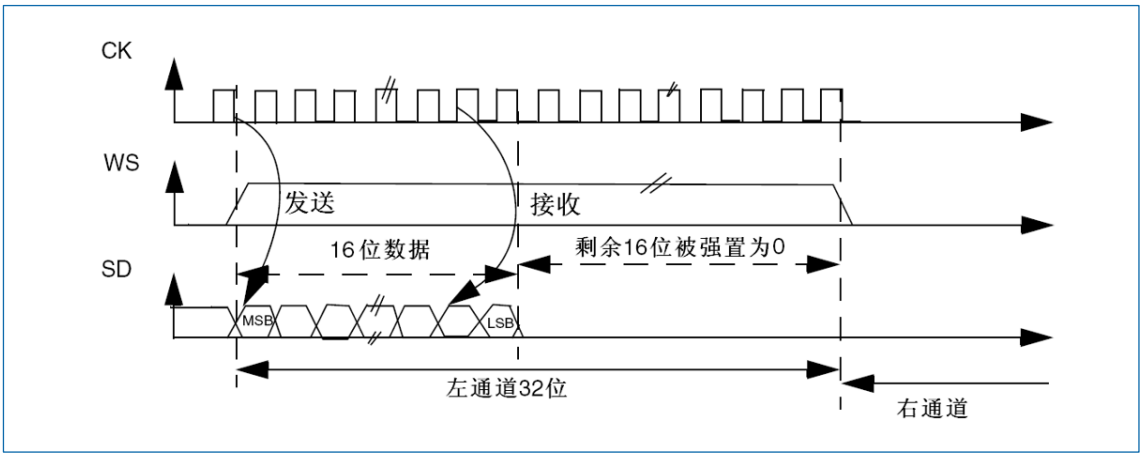


图 24-21 MSB 对齐 16 位数据扩展到 32 位包帧，CPOL=0

24.5.1.3 LSB 对齐标准

此标准与 MSB 对齐标准类似（在 16 位或 32 位全精度帧格式下无区别）。

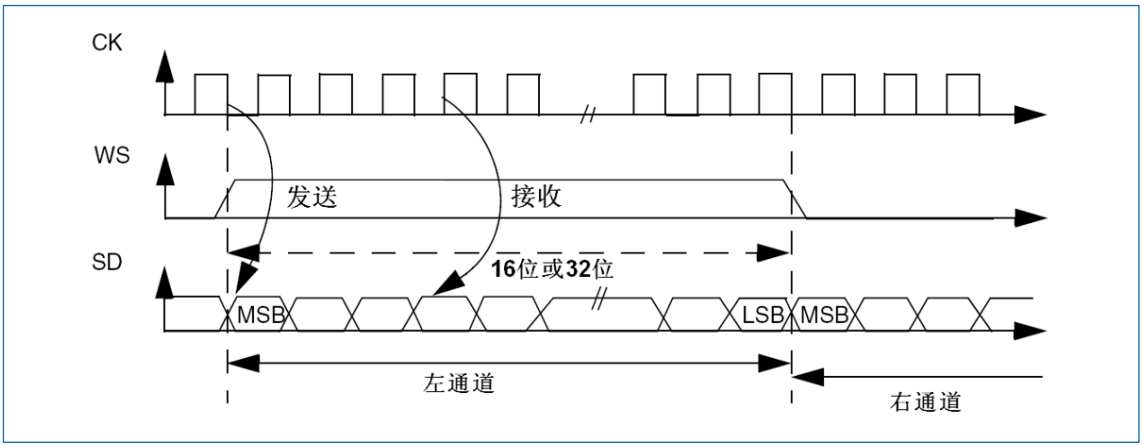


图 24-22 LSB 对齐 16 位或 32 位全精度，CPOL=0

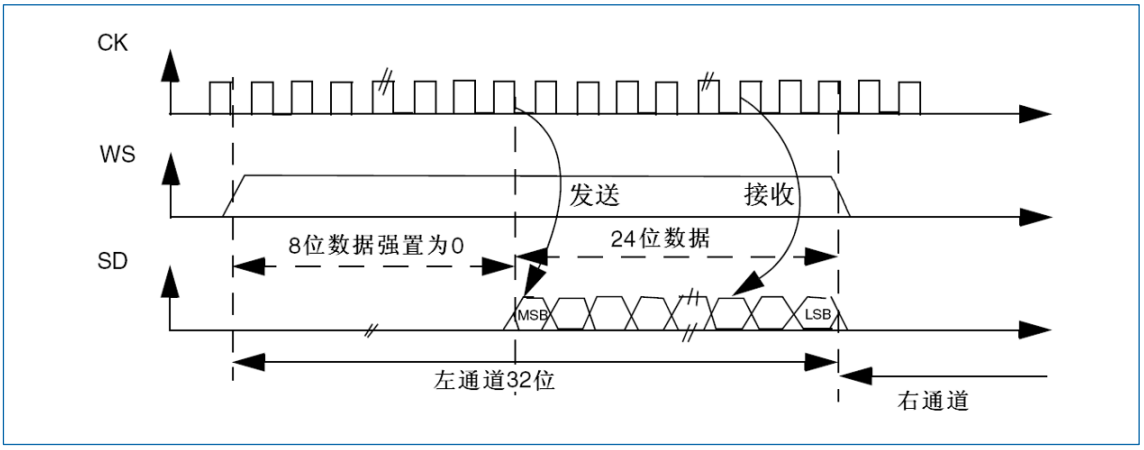


图 24-23 LSB 对齐 24 位数据，CPOL=0

- 在发送模式下：

  如果要发送数据 0x3478AE，需要通过软件或者 DMA 对寄存器 SPIx_DR 进行 2 次写操作。操作流程如下图所示。

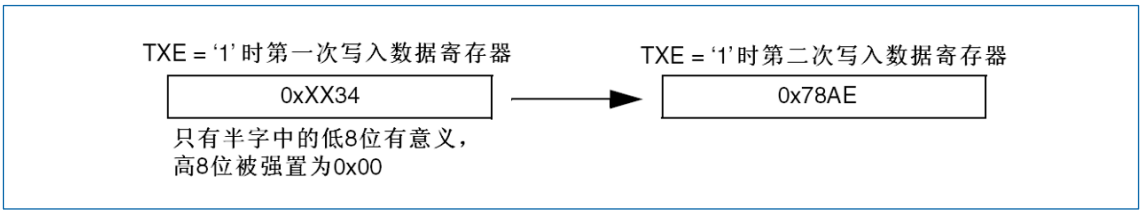


图 24-24 要求发送 0x3478AE 的操作

- 在接收模式下：

  如果要接收数据 0x3478AE，需要在 2 个连续的 RXNE 事件发生时，分别对寄存器 SPIx_DR 进行 1次读操作。

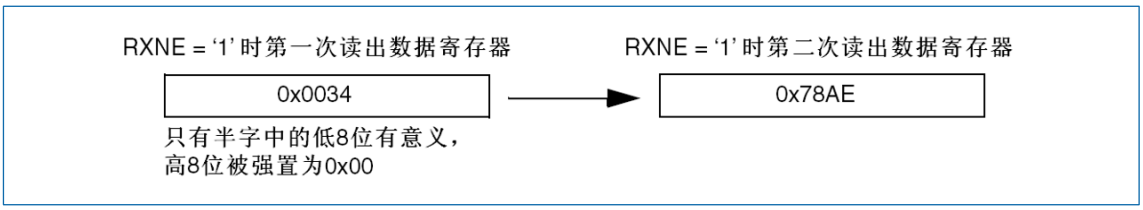


图 24-25 要求接收 0x3478AE 的操作

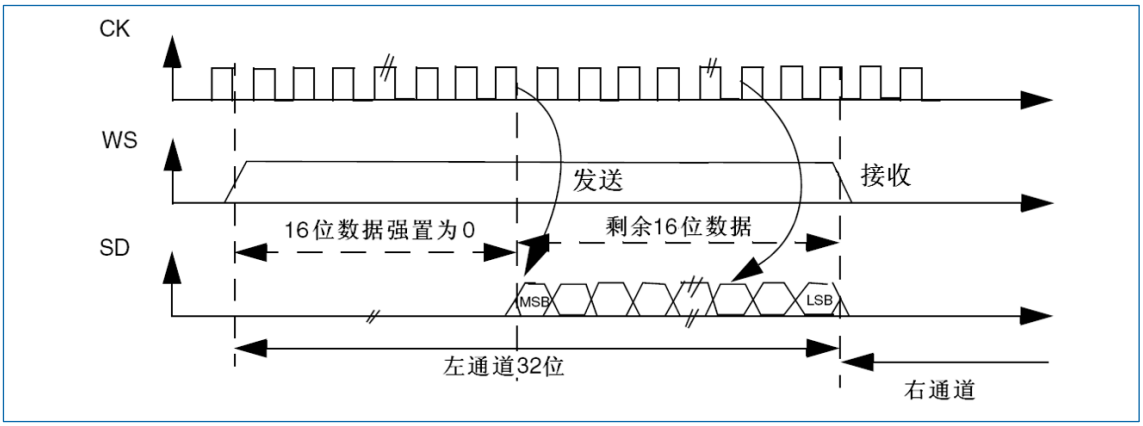


图 24-26 LSB 对齐 16 位数据扩展到 32 位包帧，CPOL=0

在 I2S 配置阶段，如果选择将 16 位数据扩展到 32 声道帧，只需要访问一次寄存器 SPIx_DR。此时，扩展到 32 位后的高半字（16 位 MSB）被硬件置为 0x0000。

如果待传输或者接收的数据是 0x76A3（扩展到 32 位是 0x0000 76A3），需要的操作如下图所示。

图 24-27 示例

在发送时，如果 TXE 为’1’，用户需要写入待发送的数据（即 0x76A3）。用来扩展到 32 位的 0x0000 部分由硬件首先发送出去，一旦有效数据开始从 SD 引脚送出，即发生下一次 TXE 事件。

在接收时，一旦接收到有效数据（而不是 0x0000 部分），即发生 RXNE 事件。

这样，在 2 次读和写之间有更多的时间，可以防止下溢或者上溢的情况发生。

24.5.1.4 PCM 标准

在 PCM 标准下，不存在声道选择的信息。PCM 标准有 2 种可用的帧结构，短帧或者长帧，可以通

过设置寄存器 SPIx_I2SCFGR 的 PCMSYNC 位来选择。

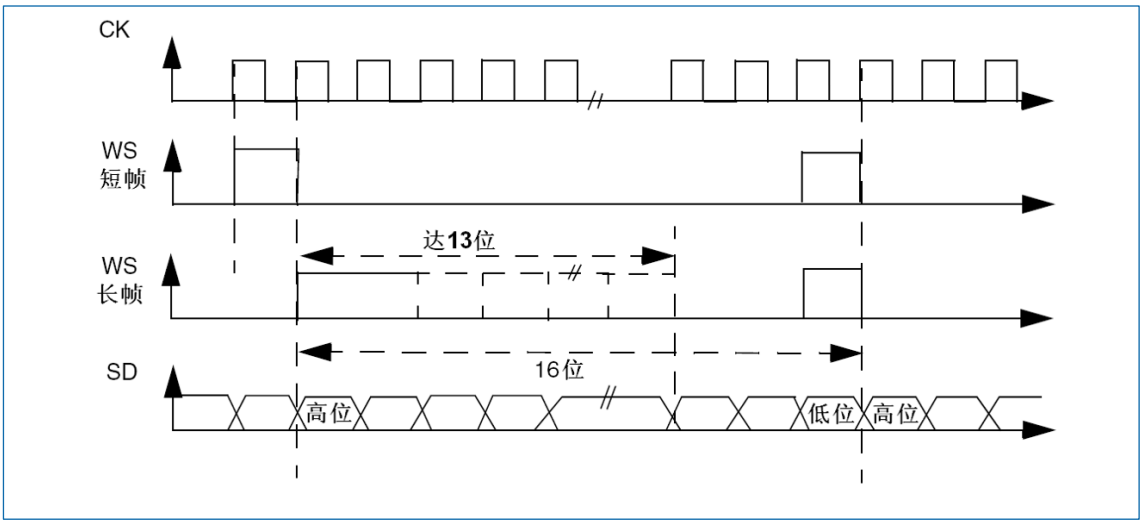


图 24-28 PCM 标准波形（16 位）

对于长帧，主模式下，用来同步的 WS 信号有效的时间固定为 13 位。对于短帧，用来同步的 WS 信号长度只有 1 位。

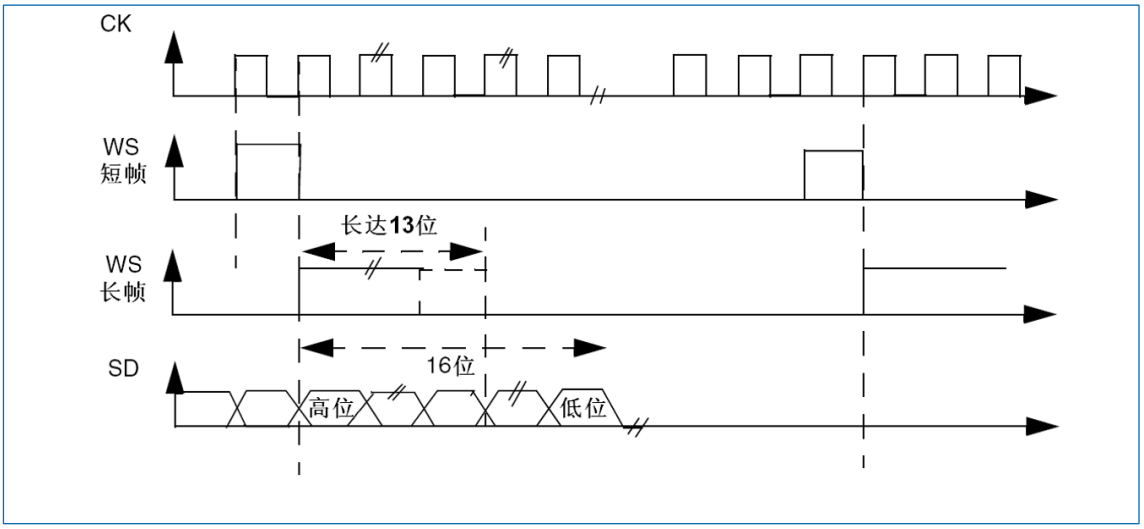


图 24-29 PCM 标准波形（16 位扩展到 32 位包帧）

注意：无论哪种模式（主或从）、哪种同步方式（短帧或长帧），连续的 2 帧数据之间和 2 个同步信号之间的时间差，（即使是从模式）需要通过设置 SPIx_I2SCFGR 寄存器的 DATLEN 位和 CHLEN 位来确定。

24.5.2 时钟发生器

I2S 的比特率即确定了在 I2S 数据线上的数据流和 I2S 的时钟信号频率。

I2S 比特率 =每个声道的比特数×声道数目×音频采样频率

对于一个具有左右声道和 16 位音频信号，I2S 比特率计算如下：

I2S 比特率=16*2*Fs

如果包长为 32 位，则：I2S 比特率=32*2*Fs

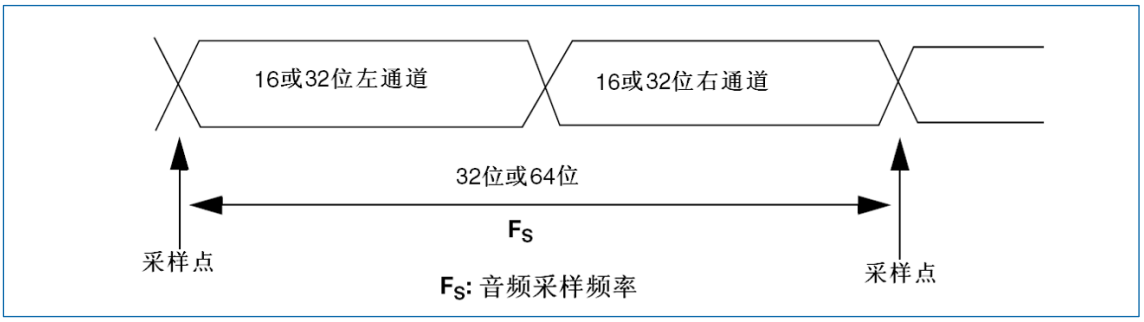


图 24-30 音频采样频率定义

在主模式下，为了获得需要的音频频率，需要正确地对线性分频器进行设置。

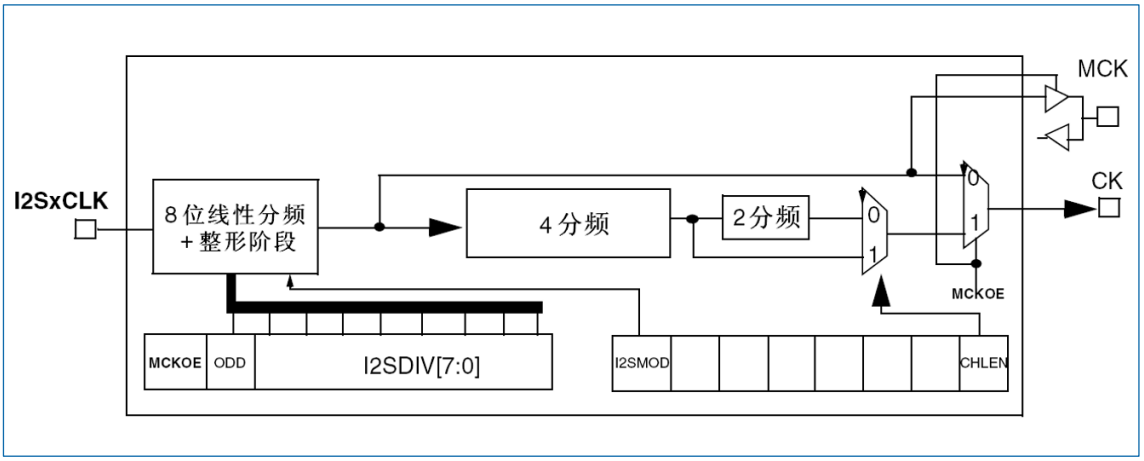


图 24-31 I2S 时钟发生器结构

说明：

- 图中 x 可以是 2 或者 3。
- 上图中 I2SxCLK 的时钟源是系统时钟（即驱动 AHB 时钟的 HSI、HSE 或 PLL）。I2SxCLK 可以来自 SYSCLK，或 PLL3 VCO（2xPLL3CLK）时钟从而得到最精确的时钟，可以通过 RCC_CFGR2 寄存器的 I2S2SRC 和 I2S3SRC 位选择。
- 音频的采样频率可以是 96kHz、48kHz、44.1kHz、32kHz、22.05kHz、16kHz、11.025kHz 或者 8kHz（或任何此范围内的数值）。为了获得需要的频率，需按照以下公式设置线性分频器：
  - 当需要生成主时钟时（寄存器 SPIx_I2SPR 的 MCKOE 位为'1'）：
    - 声道的帧长为 16 位时，Fs=I2SxCLK/[（16*2）*（(2*I2SDIV）+ODD）*8]
    - 声道的帧长为 32 位时，Fs=I2SxCLK/[（32*2）*（(2*I2SDIV）+ODD）*4]
  - 当关闭主时钟时（MCKOE 位为'0'）：
    - 声道的帧长为 16 位时，Fs=I2SxCLK/[（16*2）*（(2*I2SDIV）+ODD）]
    - 声道的帧长为 32 位时，Fs=I2SxCLK/[（32*2）*（(2*I2SDIV）+ODD）]

下面 2 张表给出了不同时钟配置时，精确参数的例子。

注意：可以使用其它配置以达到优化时钟精确度的目的。

表 24-3 使用标准的 8MHz HSE 时钟得到精确的音频频率

SYSCLK（MHz）	I2S_DIV		I2S_ODD		MCLK	期望值 FS（Hz）	实际 FS（Hz）		误差	
	16 位	32 位	16 位	32 位			16 位	32 位	16 位	32 位
72	11	6	1	0	无	96000	97826.09	93750	1.90%	2.34%

SYSCLK（MHz）	I2S_DIV		I2S_ODD		MCLK	期望值 FS（Hz）	实际 FS（Hz）		误差	
	16 位	32 位	16 位	32 位			16 位	32 位	16 位	32 位
72	23	11	1	1	无	48000	47872.34	48913.04	0.27%	1.90%
72	25	13	1	0	无	44100	44117.65	43269.23	0.04%	1.88%
72	35	17	0	1	无	32000	32142.86	32142.86	0.44%	0.44%
72	51	25	0	1	无	22050	22058.82	22058.82	0.04%	0.04%
72	70	35	1	0	无	16000	15675.75	16071.43	0.27%	0.45%
72	102	51	0	0	无	11025	11029.41	11029.41	0.04%	0.04%
72	140	70	1	1	无	8000	8007.11	7978.72	0.09%	0.27%
72	2	2	0	0	有	96000	70312.15	70312.15	26.76%	26.76%
72	3	3	0	0	有	48000	46875	46875	2.34%	2.34%
72	3	3	0	0	有	44100	46875	46875	6.29%	6.29%
72	9	9	0	0	有	32000	31250	31250	2.34%	2.34%
72	6	6	1	1	有	22050	21634.61	21634.61	1.88%	1.88%
72	9	9	0	0	有	16000	15625	15625	2.34%	2.34%
72	13	13	0	0	有	11025	10817.3	10817.3	1.88%	1.88%
72	17	17	1	1	有	8000	8035.71	8035.71	0.45%	0.45%

24.5.3 I2S 主模式

设置 I2S 工作在主模式，串行时钟由引脚 CK 输出，字选信号由引脚 WS 产生。可以通过设置寄存器 SPIx_I2SPR 的 MCKOE 位来选择输出或者不输出主时钟（MCK）。

24.5.3.1 流程

1. 设置寄存器 SPIx_I2SPR 的 I2SDIV[7:0]定义与音频采样频率相符的串行时钟波特率。同时也要定义寄存器 SPIx_I2SPR 的 ODD 位。
2. 设置 CKPOL 位定义通信用时钟在空闲时的电平状态。如果需要向外部的 DAC/ADC 音频器件提供主时钟 MCK，将寄存器 SPIx_I2SPR 的 MCKOE 位置为'1'。（按照不同的 MCK 输出状态，计算 I2SDIV 和 ODD 的值，详见章节："24.5.2 时钟发生器"）。
3. 设置寄存器 SPIx_I2SCFGR 的 I2SMOD 位为'1'激活 I2S 功能，设置 I2SSTD[1:0]和 PCMSYNC 位选择所用的 I2S 标准，设置 CHLEN 选择每个声道的数据位数。还要设置寄存器 SPIx_I2SCFGR 的 I2SCFG[1:0]选择 I2S 主模式和方向（发送端还是接收端）。

4. 如果需要，可以通过设置寄存器 SPIx_CR2 来打开所需的中断功能和 DMA 功能。
5. 必须将寄存器 SPIx_I2SCFGR 的 I2SE 位置为'1'。
6. 引脚 WS 和 CK 需要配置为输出模式。如果寄存器 SPIx_I2SPR 的 MCKOE 位为'1'，引脚 MCK 也要配置成输出模式。

24.5.3.2 发送流程

当写入 1 个半字（16 位）的数据至发送缓存，发送流程开始。假设第一个写入发送缓存的数据对应的是左声道数据。当数据从发送缓存移到移位寄存器时，标志位 TXE 置'1'，这时，要把对应右声道的数据写入发送缓存。标志位 CHSIDE 提示了目前待传输的数据对应哪个声道。标志位 CHSIDE 的值在 TXE 为'1'时更新，因此它在 TXE 为'1'时有意义。

在先左声道后右声道的数据都传输完成后，才能被认为是一个完整的数据帧。不可以只传输部分数据帧，如仅有左声道的数据。

当发出第一位数据的同时，半字数据被并行地传送至 16 位移位寄存器，然后后面的位依次按高位在先的顺序从引脚 MOSI/SD 发出。每次数据从发送缓存移至移位寄存器时，标志位 TXE 置为'1'，如果寄存器 SPIx_CR2 的 TXEIE 位为'1'，则产生中断。

写入数据的操作取决于所选择的 I2S 标准，详见"24.5.1 支持的音频协议"。

为了保证连续的音频数据传输，建议在当前传输完成之前，对寄存器 SPIx_DR 写入下一个要传输的数据。建议在要关闭 I2S 功能时，等待标志位 TXE=1 及 BSY=0，再将 I2SE 位清零。

24.5.3.3 接收流程

接收流程的配置步骤除了第 3 点外，与发送流程的一致（参见前述的"发送流程"），需要通过配置 I2SCFG[1:0]来选择主接收模式。

无论何种数据和声道长度，音频数据总是以 16 位包的形式接收。即每次填满接收缓存后，标志位 RXNE 置'1'，如果寄存器 SPIx_CR2 的 RXNEIE 位为'1'，则产生中断。根据配置的数据和声道长度，收到左声道或右声道的数据会需要 1 次或者 2 次把数据传送到接收缓存的过程。

对寄存器 SPIx_DR 进行读操作即可清除 RXNE 标志位。每次接收以后即更新 CHSIDE。它的值取决于 I2S 单元产生的 WS 信号。读取数据的操作取决于所选择的 I2S 标准，详见"24.5.1 支持的音频协议"。

如果前一个接收到的数据还没有被读取，又接收到新数据，即发生上溢，标志位 OVR 被置为'1'，如果寄存器 SPIx_CR2 的 ERRIE 位为'1'，则产生中断，表示发生了错误。

若要关闭 I2S 功能，需要执行特别的操作，以保证 I2S 模块可以正常地完成传输周期而不会开始新的数据传输。操作过程与数据配置和通道长度、以及音频协议的模式相关：

- 16 位数据扩展到 32 位通道长度（DATLEN=00 并且 CHLEN=1），使用 LSB（低位）对齐模式（I2SSTD=10）
  A. 等待倒数第二个（n-1）RXNE=1；
  B. 等待 17 个 I2S 时钟周期（使用软件延迟）；
  C. 关闭 I2S（I2SE=0）。
- 16 位数据扩展到 32 位通道长度（DATLEN=00 并且 CHLEN=1），使用 MSB（高位）对齐、I2S 或 PCM 模式（分别为 I2SSTD=00，I2SSTD=01 或 I2SSTD=11）
  A. 等待最后一个 RXNE=1；
  B. 等待 1 个 I2S 时钟周期（使用软件延迟）；
  C. 关闭 I2S（I2SE=0）。
- 所有其它 DATLEN 和 CHLEN 的组合，I2SSTD 选择的任意音频模式，使用下述方式关闭 I2S：

A. 等待倒数第二个（n-1）RXNE=1；

B. 等待一个 I2S 时钟周期（使用软件延迟）；

C. 关闭 I2S（I2SE=0）。

注意：在传输期间 BSY 标志始终为低。

24.5.4 I2S 从模式

在从模式下，I2S 可以设置成发送和接收模式。从模式的配置方式基本遵循和配置主模式一样的流程。在从模式下，不需要 I2S 接口提供时钟。时钟信号和 WS 信号都由外部主 I2S 设备提供，连接到相应的引脚上。因此用户无需配置时钟。

24.5.4.1 配置步骤

1. 设置寄存器 SPIx_I2SCFGR 的 I2SMOD 位激活 I2S 功能；设置 I2SSTD[1:0]来选择所用的 I2S 标准；设置 DATLEN[1:0]选择数据的比特数；设置 CHLEN 选择每个声道的数据位数。设置寄存器 SPIx_I2SCFGR 的 I2SCFG[1:0]选择 I2S 从模式的数据方向（发送端还是接收端）。
2. 根据需要，设置寄存器 SPIx_CR2 打开所需的中断功能和 DMA 功能。
3. 必须设置寄存器 SPIx_I2SCFGR 的 I2SE 位为'1'。

24.5.4.2 发送流程

当外部主设备发送时钟信号，并且当 NSS_WS 信号请求传输数据时，发送流程开始。必须先使能从设备，并且写入 I2S 数据寄存器之后，外部主设备才能开始通信。

对于 I2S 的 MSB 对齐和 LSB 对齐模式，第一个写入数据寄存器的数据项对应左声道的数据。当开始通信时，数据从发送缓冲器传送到移位寄存器，然后标志位 TXE 置为'1'；这时，要把对应右声道的数据项写入 I2S 数据寄存器。

标志位 CHSIDE 提示了目前待传输的数据对应哪个声道。与主模式的发送流程相比，在从模式中，CHSIDE 取决于来自外部主 I2S 的 WS 信号。这意味着从 I2S 在接收到主端生成的时钟信号之前，就要准备好第一个要发送的数据。WS 信号为'1'表示先发送左声道。

注意：设置 I2SE 位为'1'的时间，应当比 CK 引脚上的主 I2S 时钟信号早至少 2 个 PCLK 时钟周期。

当发出第一位数据的时候，半字数据并行地通过 I2S 内部总线传输至 16 位移位寄存器，然后其它位依次按高位在先的顺序从引脚 MOSI/SD 发出。每次数据从发送缓冲器传送至移位寄存器时，标志位 TXE 置'1'，如果寄存器 SPIx_CR2 的 TXEIE 位为'1'，则产生中断。

注意，在对发送缓冲器写入数据前，要确认标志位 TXE 为'1'。写入数据的操作取决于所选中的 I2S 标准，详见"24.5.1 支持的音频协议"。

为了保证连续的音频数据传输，建议在当前传输完成之前，对寄存器 SPIx_DR 写入下一个要传输的数据。如果在代表下一个数据传输的第一个时钟边沿到达之前，新的数据仍然没有写入寄存器 SPIx_DR，下溢标志位会置'1'，并可能产生中断；它指示软件发送数据错误。如果寄存器 SPIx_CR2 的 ERRIE 位为'1'，在寄存器 SPIx_SR 的标志位 UDR 为高是，就会产生中断。建议在这时关闭 I2S，然后重新从左声道开始发送数据。

建议在清除 I2SE 位关闭 I2S 之前，先等待 TXE=1 并且 BSY=0。

24.5.4.3 接收流程

配置步骤除了第 1 点外，与发送流程一致。需要通过配置 I2SCFG[1:0]来选择主接收模式。

无论何种数据和声道长度，音频数据总是以 16 位包的形式接收，即每次填满接收缓存，标志位

RXNE 置’1’，如果寄存器 SPIx_CR2 的 RXNEIE 位为’1’，则产生中断。按照不同的数据和声道长度设置，收到左声道或者右声道数据会需要 1 次或者 2 次传输数据至接收缓冲器的过程。

每次接收到数据（将要从 SPIx_DR 读出）以后即更新 CHSIDE，它对应 I2S 单元产生的 WS 信号。读取 SPIx_DR 寄存器，将清除 RXNE 位。读取数据的操作取决于所选中的 I2S 标准，详见章节：“支持的音频协议”。

在还没有读出前一个接收到的数据，又接收到新数据时，即产生上溢，并设置标志位 OVR 为’1’；如果寄存器 SPIx_CR2 的 ERRIE 位为’1’，则产生中断，指示发生了错误。

要关闭 I2S 功能时，需要在接收到最后一次 RXNE=1 时将 I2SE 位清零。

注意：外部主 I2S 器件需要有通过音频声道发送/接收 16 位或 32 位数据包的功能。

24.5.5 状态标志位

有 3 个状态标志位供用户监控 I2S 总线的状态。

24.5.5.1 忙标志位（BSY）

BSY 标志由硬件设置与清除（写入此位无效果），该标志位指示 I2S 通信层的状态。

该位为’1’时表明 I2S 通讯正在进行中，但有一个例外：主接收模式（I2SCFG=11）下，在接收期间 BSY 标志始终为低。

在软件要关闭 SPI 模块之前，可以使用 BSY 标志检测传输是否结束，这样可以避免破坏最后一次传输，因此需要严格按照下述过程执行。

当传输开始时，BSY 标志被置为’1’，除非 I2S 模块处于主接收模式。

下述情况时，该标志位被清除：

- 当传输结束时（除了主发送模式，这种模式下通信是连续的）；
- 当关闭 I2S 模块时。

当通信是连续的时候：

- 在主发送模式时，整个传输期间，BSY 标志始终为高；
- 在从模式时，每个数据项传输之间，BSY 标志在 1 个 I2S 时钟周期内变低。

注意：不要使用 BSY 标志处理每一个数据项的发送和接收，最好使用 TXE 和 RXNE 标志。

24.5.5.2 发送缓存空标志位（TXE）

该标志位为’1’表示发送缓冲器为空，可以对发送缓冲器写入新的待发送数据。在发送缓冲器中已有数据时，标志位清零。在 I2S 被关闭时（I2SE 位为’0’），该标志位也为’0’。

24.5.5.3 接收缓存非空标志位（RXNE）

该标志位置’1’表示在接收缓存里有接收到的有效数据。在读取寄存器 SPIx_DR 时，该位清零。

24.5.5.4 声道标志位（CHSIDE）

在发送模式下，该标志位在 TXE 为高时刷新，指示从 SD 引脚上发送的数据所在的声道。如果在从发送模式下发生了下溢错误，该标志位的值无效，在重新开始通讯前需要把 I2S 关闭再打开。

在接收模式下，该标志位在寄存器 SPIx_DR 接收到数据时刷新，指示接收到的数据所在的声道。注意，如果发生错误（如上溢 OVR），该标志位无意义，需要将 I2S 关闭再打开（同时，如果必要修改 I2S 的配置）。

在 PCM 标准下，无论短帧格式还是长帧格式，这个标志位都没有意义。

如果寄存器 SPIx_SR 的标志位 OVR 或 UDR 为'1'，且寄存器 SPIx_CR2 的 ERRIE 位为'1'，则会产生中断。（中断源已经被清除后）可以通过读寄存器 SPIx_SR 来清除中断标志。

24.5.6 错误标志位

I2S 单元有 2 个错误标志位。

24.5.6.1 下溢标志位（UDR）

在从发送模式下，如果数据传输的第一个时钟边沿到达时，新的数据仍然没有写入 SPIx_DR 寄存器，该标志位会被置'1'。在寄存器 SPIx_I2SCFGR 的 I2SMOD 位置'1'后，该标志位才有效。如果寄存器 SPIx_CR2 的 ERRIE 位为'1'，就会产生中断。通过对寄存器 SPIx_SR 进行读操作来清除该标志位。

24.5.6.2 上溢标志位（OVR）

如果还没有读出前一个接收到的数据时，又接收到新的数据，即产生上溢，该标志位置'1'，如果寄存器 SPIx_CR2 的 ERRIE 位为'1'，则产生中断指示发生了错误。

这时，接收缓存的内容，不会刷新为从发送设备送来的新数据。对寄存器 SPIx_DR 的读操作返回最后一个正确接收到的数据。其他所有在上溢发生后由发送设备发出的 16 位数据都会丢失。

通过先读寄存器 SPIx_SR 再读寄存器 SPIx_DR，来清除该标志位。

24.5.7 I2S 中断

下表列举了全部 I2S 中断：

表 24-4 I2S 中断请求

中断事件	事件标志位	使能标志位
发送缓冲器空标志位	TXE	TXEIE
接收缓冲器非空标志位	RXNE	RXNEIE
下溢标志位	OVR	ERRIE
上溢标志位	UDR	

24.5.8 DMA 功能

DMA 的工作方式在 I2S 模式除了 CRC 功能不可用以外，与在 SPI 模式完全相同。因为在 I2S 模式下没有数据传输保护系统。

24.6 SPI 寄存器

基地址：（SPI1, SPI2, SPI3）=（0x4001 3000, 0x4000 3800, 0x4000 3C00）

空间大小：（SPI1, SPI2, SPI3）=（0x400, 0x400, 0x400）

可以用半字（16 位）或字（32 位）的方式操作这些外设寄存器。

24.6.1 SPI 控制寄存器 1（SPIx_CR1）（x=1..3）

偏移地址：0x00

复位值：0x0000

说明：该寄存器在 I2S 模式下不使用

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIDIMODE	BIDIOE	CRCEN	CRCNEXT	DFF	RXONLY	SSM	SSI	LSBFIRST	SPE	BR[2:0]			MSTR	CPOL	CPHA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw

位 15	BIDIMODE：双向数据模式使能（Bidirectional data mode enable） • 0：选择“双线单向”模式 • 1：选择“单线双向”模式 *注意：I2S 模式下不使用。*
位 14	BIDIOE：双向模式下的输出使能（Output enable in bidirectional mode） 和 BIDIMODE 位一起决定在“单线双向”模式下数据的输出方向： • 0：输出禁止（只收模式） • 1：输出使能（只发模式） 这个“单线”数据线在主设备端为 MOSI 引脚，在从设备端为 MISO 引脚。 *注意：I2S 模式下不使用。*
位 13	CRCEN：使能 CRC（Hardware CRC calculation enable） • 0：禁止 CRC 计算 • 1：使能 CRC 计算 *注意：只有当 SPI 禁止时（SPE=0），此位可以被写。I2S 模式下不使用。*
位 12	CRCNEXT：下一个发送 CRC（Transmit CRC next） • 0：下一个发送的值来自发送缓冲区 • 1：下一个发送的值来自发送 CRC 寄存器 *注意：* • *在 SPIx_DR 寄存器写入最后一个数据后应马上设置该位。* • *I2S 模式下不使用。*
位 11	DFF：数据帧格式（Data frame format） • 0：使用 8 位数据帧格式进行发送/接收 • 1：使用 16 位数据帧格式进行发送/接收 *注意：* • *只有当 SPI 禁止（SPE=0）时，才能写该位，否则出错。* • *I2S 模式下不使用。*
位 10	RXONLY：只接收（Receive only） 该位和 BIDIMODE 位一起决定在“双线单向”模式下的传输方向。在多个从设备的配置中，在未被访问的从设备上该位被置 1，使得只有被访问的从设备有输出，从而不会造成数据线上数据冲突。 • 0：全双工（发送和接收） • 1：禁止输出（只接收模式） *注意：I2S 模式下不使用。*
位 9	SSM：软件从设备管理（Software slave management） 当 SSM 被置位时，NSS 引脚上的电平由 SSI 位的值决定。 • 0：禁止软件从设备管理 • 1：启用软件从设备管理 *注意：I2S 模式下不使用。*

位 8	SSI：内部从设备选择（Internal slave select） 该位只在 SSM 位为'1'时有意义。它决定了 NSS 上的电平，在 NSS 引脚上的 I/O 操作无效。 *注意：I2S 模式下不使用。*
位 7	LSBFIRST：帧格式（Frame format） • 0：先发送 MSB • 1：先发送 LSB *注意：* • *当通信在进行时不能改变该位的值。* • *I2S 模式下不使用。*
位 6	SPE：SPI 使能（SPI enable） • 0：禁止 SPI 设备 • 1：开启 SPI 设备 *注意：* • *当关闭 SPI 设备时，请按照章节："关闭 SPI"节的过程操作。* • *I2S 模式下不使用。*
位 5:3	BR[2:0]：波特率控制（Baud rate control） • 000：$f_{PCLK}/2$ • 001：$f_{PCLK}/4$ • 010：$f_{PCLK}/8$ • 011：$f_{PCLK}/16$ • 100：$f_{PCLK}/32$ • 101：$f_{PCLK}/64$ • 110：$f_{PCLK}/128$ • 111：$f_{PCLK}/256$ 当通信正在进行的时候，不能修改该位域。 *注意：I2S 模式下不使用。*
位 2	MSTR：主设备选择（Master selection） • 0：配置为从设备 • 1：配置为主设备 *注意：* • *当通信正在进行的时候，不能修改该位。* • *I2S 模式下不使用。*
位 1	CPOL：时钟极性（Clock polarity） • 0：空闲状态时，SCK 保持低电平 • 1：空闲状态时，SCK 保持高电平 *注意：* • *当通信正在进行的时候，不能修改该位。* • *I2S 模式下不使用。*
位 0	CPHA：时钟相位（Clock phase） • 0：数据采样从第一个时钟边沿开始

	• 1：数据采样从第二个时钟边沿开始 *注意：* • *当通信正在进行的时候，不能修改该位。* • *I2S 模式下不使用。*

24.6.2 SPI 控制寄存器 2（SPIx_CR2）（x=1..3）

偏移地址：0x04

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								TXEIE	RXNEIE	ERRIE	Res		SSOE	TXDMAEN	RXDMAEN
								rw	rw	rw			rw	rw	rw

位 15:8	Res：保留 必须保持复位值。
位 7	TXEIE：发送缓冲区空中断使能（Tx buffer empty interrupt enable） • 0：禁止 TXE 中断 • 1：允许 TXE 中断，当 TXE 标志置位为’1’时产生中断请求。
位 6	RXNEIE：接收缓冲区非空中断使能（RX buffer not empty interrupt enable） • 0：禁止 RXNE 中断 • 1：允许 RXNE 中断，当 RXNE 标志置位时产生中断请求。
位 5	ERRIE：错误中断使能（Error interrupt enable） 错误（CRCERR、OVR、UDR、MODF）产生时，该位控制是否产生中断。 • 0：禁止错误中断 • 1：允许错误中断
位 4:3	Res：保留 必须保持复位值。
位 2	SSOE：SS 输出使能（SS output enable） • 0：禁止在主模式下 SS 输出，该设备可以工作在多主设备模式。 • 1：设备开启时，开启主模式下 SS 输出，该设备不能工作在多主设备模式。 *注意：I2S 模式下不使用。*
位 1	TXDMAEN：发送缓冲区 DMA 使能（Tx buffer DMA enable） 当该位被设置时，TXE 标志一旦被置位就发出 DMA 请求。 • 0：禁止发送缓冲区 DMA • 1：启动发送缓冲区 DMA
位 0	RXDMAEN：接收缓冲区 DMA 使能（Rx buffer DMA enable） 当该位被设置时，RXNE 标志一旦被置位就发出 DMA 请求。 • 0：禁止接收缓冲区 DMA • 1：启动接收缓冲区 DMA

24.6.3 SPI 状态寄存器（SPIx_SR）（x=1..3）

偏移地址：0x08

复位值：0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								BSY	OVR	MODF	CRCERR	UDR	CHSIDE	TXE	RXNE
								r	r	r	rc_w0	r	r	r	r

位 15:8	Res：保留 必须保持复位值。
位 7	BSY：忙标志（Busy flag） • 0：SPI 不忙 • 1：SPI 正忙于通信，或者发送缓冲非空 该位由硬件置位或者复位。 *注意：使用这个标志时需要特别注意，详见“24.4.7 状态标志”和“24.4.8 关闭 SPI”。*
位 6	OVR：溢出标志（Overrun flag） • 0：没有出现溢出错误 • 1：出现溢出错误 该位由硬件置位，由软件序列复位。关于软件序列的详细信息，参考“24.6.7 SPI Tx CRC 寄存器（SPIx_TXCRCR）（x=1..3）”。
位 5	MODF：模式错误（Mode fault） • 0：没有出现模式错误 • 1：出现模式错误 该位由硬件置位，由软件序列复位。关于软件序列的详细信息，参考“24.4.10 错误标志”。 *注意：I2S 模式下不使用。*
位 4	CRCERR：CRC 错误标志（CRC error flag） • 0：收到的 CRC 值和 SPIx_RXCRCR 寄存器中的值匹配 • 1：收到的 CRC 值和 SPIx_RXCRCR 寄存器中的值不匹配 该位由硬件置位，由软件写’0’而复位。 *注意：I2S 模式下不使用。*
位 3	UDR：下溢标志位（Underrun flag） • 0：未发生下溢 • 1：发生下溢 该标志位由硬件置’1’，由一个软件序列清零，详见章节“24.4.10 错误标志”。 *注意：SPI 模式下不使用。*
位 2	CHSIDE：声道（Channel side） • 0：需要传输或者接收左声道 • 1：需要传输或者接收右声道 *注意：在 SPI 模式下不使用。在 PCM 模式下无意义。*
位 1	TXE：发送缓冲为空（Transmit buffer empty） • 0：发送缓冲非空 • 1：发送缓冲为空
位 0	RXNE：接收缓冲非空（Receive buffer not empty） • 0：接收缓冲为空

	• 1：接收缓冲非空

24.6.4 SPI 数据寄存器（SPIx_DR）（x=1..3）

偏移地址：0x0C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[15:0]															
rw															

位 15:0	DR[15:0]：数据寄存器（Data register） 存放待发送或者已经收到的数据 数据寄存器对应两个缓冲区：一个用于写（发送缓冲）；另外一个用于读（接收缓冲）。写操作将数据写到发送缓冲区；读操作将返回接收缓冲区里的数据。 对 SPI 模式的注释： 根据 SPIx_CR1 的 DFF 位对数据帧格式的选择，数据的发送和接收可以是 8 位或者 16 位的。为保证正确的操作，需要在启用 SPI 之前就确定好数据帧格式。对于 8 位的数据，缓冲器是 8 位的，发送和接收时只会用到 SPIx_DR[7:0]。在接收时，SPIx_DR[15:8]被强制为 0。 对于 16 位的数据，缓冲器是 16 位的，发送和接收时会用到整个数据寄存器，即 SPIx_DR[15:0]。

24.6.5 SPI CRC 多项式寄存器（SPIx_CRCPR）（x=1..3）

偏移地址：0x10

复位值：0x0007

说明：该寄存器在 I2S 模式下不使用。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRCPOLY[15:0]															
rw															

位 15:0	CRCPOLY[15:0]：CRC 多项式寄存器（CRC polynomial register） 该寄存器包含了 CRC 计算时用到的多项式。其复位值为 0x0007，根据应用可以设置其他数值。 *注意：I2S 模式下不使用。*

24.6.6 SPI Rx CRC 寄存器（SPIx_RXCRCR）（x=1..3）

偏移地址：0x14

复位值：0x0000

说明：该寄存器在 I2S 模式下不使用

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RxCRC[15:0]															
r															

位 15:0	RxCRC[15:0]：接收 CRC 寄存器（Rx CRC register） 在启用 CRC 计算时，RXCRC[15:0]中包含了依据收到的字节计算的 CRC 数值。当在 SPIx_CR1 的 CRCEN 位写入’1’时，该寄存器被复位。 CRC 计算使用 SPIx_CRCPR 中的多项式： • 当数据帧格式被设置为 8 位时，仅低 8 位参与计算，并且按照 CRC8 的方法进行 • 当数据帧格式 为 16 位时，寄存器中的所有 16 位都参与计算，并且按照 CRC16 的标准 *注意：* • *当 BSY 标志为’1’时读该寄存器，将可能读到不正确的数值。*

	• *I2S 模式下不使用。*

24.6.7 SPI Tx CRC 寄存器（SPIx_TXCRCR）（x=1..3）

偏移地址：0x18

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TxCRC[15:0]															
r															

位 15:0	TxCRC[15:0]：发送 CRC 寄存器（Tx CRC register） 在启用 CRC 计算时，TXCRC[15:0]中包含了依据将要发送的字节计算的 CRC 数值。当在 SPIx_CR1 中的 CRCEN 位写入'1'时，该寄存器被复位。 CRC 计算使用 SPIx_CRCPR 中的多项式： • 当数据帧格式被设置为 8 位时，仅低 8 位参与计算，并且按照 CRC8 的方法进行 • 当数据帧格式为 16 位时，寄存器中的所有 16 个位都参与计算，并且按照 CRC16 的标准 *注意：* • *当 BSY 标志为'1'时读该寄存器，将可能读到不正确的数值。* • *I2S 模式下不使用。*

24.6.8 SPIx_I2S 配置寄存器（SPIx_I2SCFGR）（x=1..3）

偏移地址：0x1C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res				I2SMOD	I2SE	I2SCFG[1:0]		PCMSYNC	Res	I2SSTD[1:0]		CKPOL	DATLEN		CHLEN
				rw	rw	rw		rw		rw		rw	rw		rw

位 15:12	Res：保留 必须保持复位值。
位 11	I2SMOD：I2S 模式选择（I2S mode selection） • 0：选择 SPI 模式 • 1：选择 I2S 模式 *注意：该位只有在关闭了 SPI 或者 I2S 时才能设置。*
位 10	I2SE：I2S 使能（I2S enable） • 0：关闭 I2S • 1：I2S 使能 *注意：在 SPI 模式下不使用。*
位 9:8	I2SCFG[1:0]：I2S 模式设置（I2S configuration mode） • 00：从设备发送 • 01：从设备接收 • 10：主设备发送 • 11：主设备接收 *注意：* • *该位只有在关闭了 I2S 时才能设置。*

	• *在 SPI 模式下不使用。*
位 7	PCMSYNC：PCM 帧同步（PCM frame synchronization） • 0：短帧同步 • 1：长帧同步 *注意：* • *该位只在 I2SSTD=11（使用 PCM 标准）时有意义。* • *在 SPI 模式下不使用。*
位 6	Res：保留 必须保持复位值。
位 5:4	I2SSTD[1:0]：I2S 标准选择（I2S standard selection） • 00：I2S 飞利浦标准 • 01：高字节对齐标准（左对齐） • 10：低字节对齐标准（右对齐） • 11：PCM 标准 关于 I2S 标准的细节，详见“24.5.1 支持的音频协议”。 *注意：* • *为了正确操作，只有在关闭了 I2S 时才能设置该位。* • *在 SPI 模式下不使用。*
位 3	CKPOL：静止态时钟极性（Steady state clock polarity） • 0：I2S 时钟静止态为低电平 • 1：I2S 时钟静止态为高电平 *注意：* • *为了正确操作，该位只有在关闭了 I2S 时才能设置。* • *在 SPI 模式下不使用。*
位 2:1	DATLEN：待传输数据长度（Data length to be transferred） • 00：16 位数据长度 • 01：24 位数据长度 • 10：32 位数据长度 • 11：不允许 *注意：* • *为了正确操作，该位只有在关闭了 I2S 时才能设置。* • *在 SPI 模式下不使用。*
位 0	CHLEN：声道长度（每个音频声道的数据位数）（Channel length, number of bits per audio channel） • 0：16 位宽 • 1：32 位宽 只有在DATLEN=00 时该位的写操作才有意义，否则声道长度都由硬件固定为 32 位。 *注意：* • *为了正确操作，该位只有在关闭了 I2S 时才能设置。* • *在 SPI 模式下不使用。*

24.6.9 SPIx_I2S 预分频寄存器（SPIx_I2SPR）（x=1..3）

偏移地址：0x20

复位值：0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res						MCKOE	ODD	I2SDIV[7:0]							
						rw	rw	rw							

位 15:10	Res：保留 必须保持复位值。
位 9	MCKOE：主设备时钟输出使能（Master clock output enable） • 0：关闭主设备时钟输出 • 1：主设备时钟输出使能 *注意：* • *为了正确操作，该位只有在关闭了 I2S 时才能设置。仅在 I2S 主设备模式下使用该位。* • *在 SPI 模式下不使用。*
位 8	ODD：奇系数预分频（Odd factor for the prescaler） • 0：实际分频系数=I2SDIV *2 • 1：实际分频系数=（I2SDIV *2）+1 参见“24.5.2 时钟发生器”。 *注意：* • *为了正确操作，该位只有在关闭了 I2S 时才能设置。仅在 I2S 主设备模式下使用该位。* • *在 SPI 模式下不使用。*
位 7:0	I2SDIV[7:0]：I2S 线性预分频（I2S linear prescaler） 禁止设置 I2SDIV[7:0]=0 或者 I2SDIV[7:0]=1 参见“24.5.2 时钟发生器”。 *注意：* • *为了正确操作，该位只有在关闭了 I2S 时才能设置。仅在 I2S 主设备模式下使用该位。* • *在 SPI 模式下不使用。*

25 I2C 接口

内部集成电路（I2C）总线接口处理微控制器与串行 I2C 总线间的通信。它遵循 I2C 规范，支持标准模式（Standard mode，Sm）、快速模式（Fast mode，Fm）同时与 SMBus 2.0 兼容。

I2C 模块有多种用途，包括 CRC 码的生成和校验、SMBus（系统管理总线）和 PMBus（电源管理总线）。

根据特定设备的需要，该接口还支持 DMA 数据传输方式，减轻 CPU 的工作量。

25.1 I2C 主要特点

- 并行总线/I2C 总线协议转换器
- 多主机功能：该模块既可做主设备也可做从设备
- I2C 主设备功能：
  - 产生时钟
  - 产生起始和停止信号
- I2C 从设备功能：
  - 可编程的 I2C 地址检测
  - 可响应 2 个从地址的双地址能力
  - 停止位检测
- 产生和检测 7 位/10 位地址和广播呼叫
- 支持不同的通讯速度：
  - 标准模式（高达 100 kHz）
  - 快速模式（高达 400 kHz）
- 状态标志：
  - 发送器/接收器模式标志
  - 字节发送结束标志
  - I2C 总线忙标志
- 错误标志
  - 主模式时的仲裁丢失
  - 地址/数据传输后的应答（ACK）错误
  - 检测到错位的起始或停止条件
  - 禁止拉长时钟功能时的上溢或下溢
- 2 个中断向量
  - 1 个中断用于地址/数据通讯成功
  - 1 个中断用于错误
- 可选的拉长时钟功能
- 具单字节缓冲器的 DMA
- 可配置的 PEC（信息包错误检测）的产生或校验：
  - 发送模式中 PEC 值可以作为最后一个字节传输
  - 用于最后一个接收字节的 PEC 错误校验
- 兼容 SMBus 2.0

- 25 ms 时钟低超时延时
- 10 ms 主设备累积时钟低扩展时间
- 25 ms 从设备累积时钟低扩展时间
- 带 ACK 控制的硬件 PEC 产生/校验
- 支持地址分辨协议（ARP）

- 兼容 SMBus

25.2 I2C 功能描述

I2C 模块接收和发送数据，并将数据从串行转换成并行，或并行转换成串行。可以开启或禁止中断。接口通过数据引脚（SDA）和时钟引脚（SCL）连接到 I2C 总线。允许连接到标准（高达 100kHz）或快速（高达 400kHz）的 I2C 总线。

25.2.1 模式选择

接口可以下述 4 种模式中的一种运行：

- 从发送器模式
- 从接收器模式
- 主发送器模式
- 主接收器模式

该模块默认地工作于从模式。接口在生成起始条件后自动地自从模式切换到主模式；当仲裁丢失或产生停止信号时，则从主模式切换到从模式。允许多主机功能。

25.2.1.1 通信流

主模式时，I2C 接口启动数据传输并产生时钟信号。串行数据传输总是以起始条件开始并以停止条件结束。起始条件和停止条件都是在主模式下由软件控制产生。

从模式时，I2C 接口能识别它自己的地址（7 位或 10 位）和广播呼叫地址。软件能够控制开启或禁止广播呼叫地址的识别。

数据和地址按 8 位/字节进行传输，高位在前。跟在起始条件后的 1 或 2 个字节是地址（7 位模式为 1 个字节，10 位模式为 2 个字节）。地址只在主模式发送。在一个字节传输的 8 个时钟后的第 9 个时钟期间，接收器必须回送一个应答位（ACK）给发送器。参考下图。

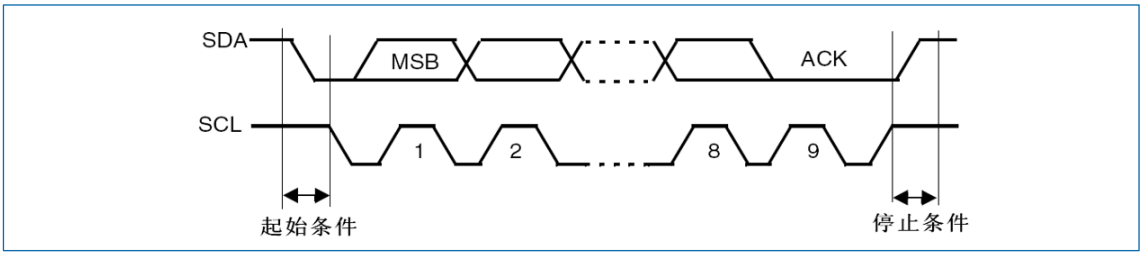


图 25-1 I2C 总线协议

软件可以开启或禁止应答（ACK），并可以设置 I2C 接口的地址（7 位、10 位地址或广播呼叫地址）。I2C 接口的功能框图示于下图：

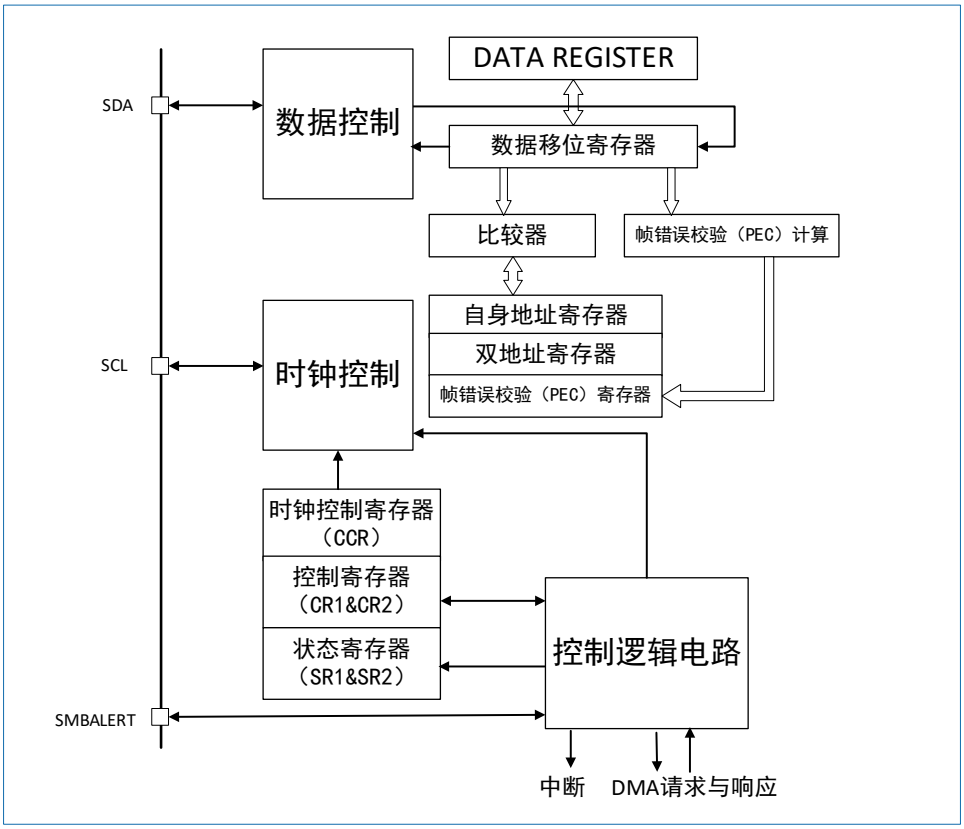


图 25-2 I2C 的功能框图

注意：在 SMBus 模式下，SMBALERT 是可选信号。如果禁止了 SMBus，则不能使用该信号。

25.2.2 I2C 从模式

默认情况下，I2C 接口总是工作在从模式。自从模式切换到主模式，需要产生一个起始条件。

为了产生正确的时序，必须在 I2Cx_CR2 寄存器中设定该模块的输入时钟。输入时钟的频率必须至少是：

- 标准模式下为：2MHz
- 快速模式下为：4MHz

一旦检测到起始条件，在 SDA 线上接收到的地址被送到移位寄存器。然后与芯片自己的地址 OAR1 和 OAR2（当 ENDUAL=1）或者广播呼叫地址（如果 ENGC=1）相比较：

注意：在 10 位地址模式时，比较包括头段序列（11110xx0），其中的 xx 是地址的两个最高有效位。

- 头段或地址不匹配：I2C 接口将其忽略并等待另一个起始条件。
- 头段匹配（仅 10 位模式）：如果 ACK 位被置'1'，I2C 接口产生一个应答脉冲并等待 8 位从地址。
- 地址匹配：I2C 接口产生以下时序：
  - 如果 ACK 被置'1'，则产生一个应答脉冲。
  - 硬件设置 ADDR 位；如果设置了 ITEVTEN 位，则产生一个中断。
  - 如果 ENDUAL=1，软件必须读 DUALF 位，以确认响应了哪个从地址。

在 10 位模式，接收到地址序列后，从设备总是处于接收器模式。在收到与地址匹配的头序列并且最低位为'1'（即 11110xx1）后，当接收到重复的起始条件时，将进入发送器模式。

在从模式下 TRA 位指示当前是处于接收器模式还是发送器模式。

25.2.2.1 从发送器

在接收到地址和清除 ADDR 位后，从发送器将字节从 DR 寄存器经由内部移位寄存器发送到 SDA 线上。

从设备保持 SCL 为低电平，直到 ADDR 位被清除并且待发送数据已写入 DR 寄存器。（见下图中的 EV1 和 EV3）。

当收到应答脉冲时：

- TxE 位被硬件置位，如果设置了 ITEVTEN 和 ITBUFEN 位，则产生一个中断。

  如果 TxE 位被置位，但在下一个数据发送结束之前没有新数据写入到 I2Cx_DR 寄存器，则 BTF 位被置位，在清除 BTF 之前 I2C 接口将保持 SCL 为低电平；读出 I2Cx_SR1 之后再写入 I2Cx_DR 寄存器将清除 BTF 位。

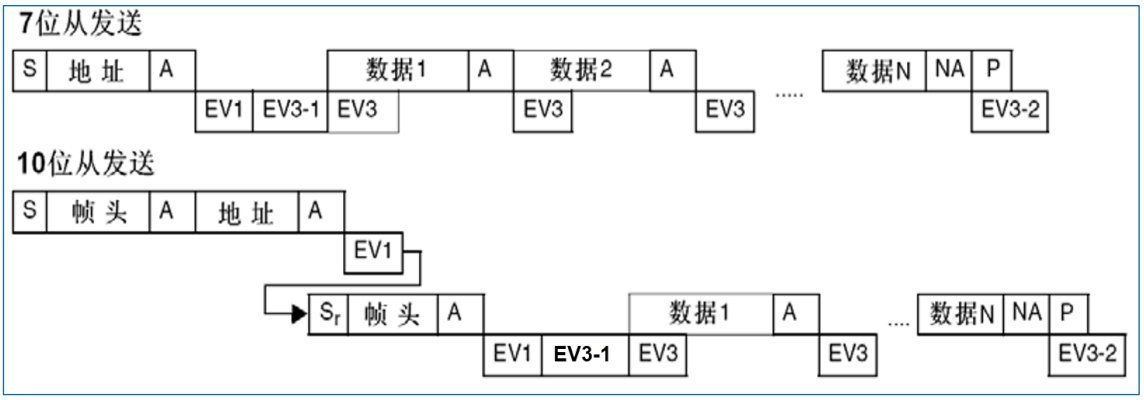


图 25-3 从发送器的传送序列图

说明：

- S=Start（起始条件），S_r=重复的起始条件，P=Stop（停止条件），A=响应，NA=非响应，EVx=事件（ITEVTEN=1 时产生中断）
- EV1：ADDR=1，读 SR1 然后读 SR2 将清除该事件。
- EV3-1：TxE=1，移位寄存器空，数据寄存器空，写 DR。
- EV3：TxE=1，移位寄存器非空，数据寄存器空，写 DR 将清除该事件。
- EV3-2：AF=1，在 SR1 寄存器的 AF 位写'0'可清除 AF 位。

注意：

EV1 和 EV3_1 事件拉长 SCL 低的时间，直到对应的软件序列结束。

EV3 的软件序列必须在当前字节传输结束之前完成。

25.2.2.2 从接收器

在接收到地址并清除 ADDR 后，从接收器将通过内部移位寄存器从 SDA 线接收到的字节存进 DR 寄存器。I2C 接口在接收到每个字节后都执行下列操作：

- 如果设置了 ACK 位，则产生一个应答脉冲
- 硬件设置 RxNE=1。如果设置了 ITEVTEN 和 ITBUFEN 位，则产生一个中断。

如果 RxNE 被置位，并且在接收新的数据结束之前 DR 寄存器未被读出，BTF 位被置位，在清除 BTF 之前 I2C 接口将保持 SCL 为低电平；读出 I2Cx_SR1 之后再写入 I2Cx_DR 寄存器将清除 BTF 位。（见下图）。

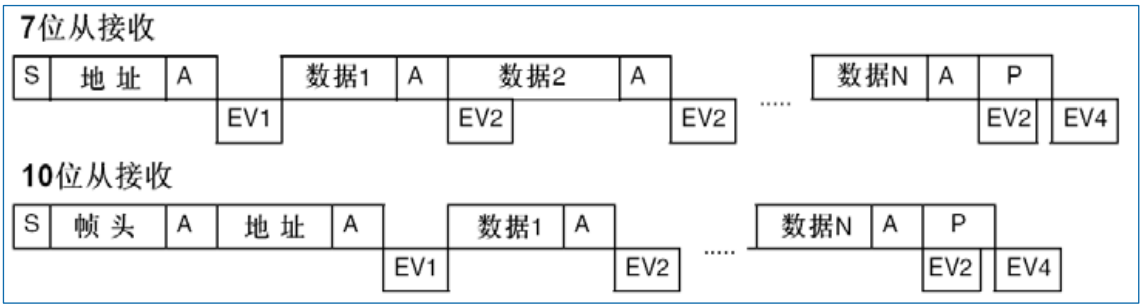


图 25-4 从接收器的传送序列图

说明：

- S=Start（起始条件），Sr=重复的起始条件，P=Stop（停止条件），A=响应，NA=非响应，EVx=事件（ITEVTEN=1 时产生中断）
- EV1：ADDR=1，读 SR1 然后读 SR2 将清除该事件。
- EV2：RxNE=1，读 DR 将清除该事件。
- EV4：STOPF=1，读 SR1 然后写 CR1 寄存器将清除该事件。

  注意：

  EV1 事件拉长 SCL 低的时间，直到对应的软件序列结束。

  EV2 的软件序列必须在当前字节传输结束之前完成。

25.2.2.3 关闭从通信

在传输完最后一个数据字节后，主设备产生一个停止条件，I2C 接口检测到这一条件时，设置 STOPF=1，如果设置了 ITEVTEN 位，则产生一个中断。然后 I2C 接口等待读 SR1 寄存器，再写 CR1 寄存器。（见上图的 EV4）。

25.2.3 I2C 主模式

在主模式时，I2C 接口启动数据传输并产生时钟信号。串行数据传输总是以起始条件开始并以停止条件结束。当通过 START 位在总线上产生了起始条件，设备就进入了主模式。

以下是主模式所要求的操作顺序：

1. 在 I2Cx_CR2 寄存器中设定该模块的输入时钟以产生正确的时序
2. 配置时钟控制寄存器
3. 配置上升时间寄存器
4. 编程 I2Cx_CR1 寄存器启动外设
5. 置 I2Cx_CR1 寄存器中的 START 位为 1，产生起始条件

I2C 模块的输入时钟频率必须至少是：

- 标准模式下为：2MHz
- 快速模式下为：4MHz

25.2.3.1 起始条件

当 BUSY=0 时，设置 START=1，I2C 接口将产生一个开始条件并切换至主模式（M/SL 位置位）。

注意：在主模式下，设置 START 位将在当前字节传输完后由硬件产生一个重开始条件。

一旦发出开始条件，SB 位被硬件置位，如果设置了 ITEVTEN 位，则会产生一个中断。

然后主设备等待读 SR1 寄存器，紧跟着将从地址写入 DR 寄存器（见图 25-5 和图 25-6 的 EV5）。

25.2.3.2 从地址的发送

从地址通过内部移位寄存器被送到 SDA 线上。

- 在 10 位地址模式时，发送一个头段序列产生以下事件：
  - ADD10 位被硬件置位，如果设置了 ITEVTEN 位，则产生一个中断。

  然后主设备等待读 SR1 寄存器，再将第二个地址字节写入 DR 寄存器（见图 25-5 和图 25-6）。
  - ADDR 位被硬件置位，如果设置了 ITEVTEN 位，则产生一个中断。

  随后主设备等待一次读 SR1 寄存器，跟着读 SR2 寄存器（见图 25-5 和图 25-6）。
- 在 7 位地址模式时，只需送出一个地址字节。

  一旦该地址字节被送出：
  - ADDR 位被硬件置位，如果设置了 ITEVTEN 位，则产生一个中断。

  随后主设备等待一次读 SR1 寄存器，跟着读 SR2 寄存器（见图 25-5 和图 25-6）。

根据送出从地址的最低位，主设备决定进入发送器模式还是进入接收器模式。

- 在 7 位地址模式时：
  - 要进入发送器模式，主设备发送从地址时置最低位为’0’。
  - 要进入接收器模式，主设备发送从地址时置最低位为’1’。
- 在 10 位地址模式时
  - 要进入发送器模式，主设备先送头字节（11110xx0），然后送最低位为’0’的从地址。（这里 xx 代表 10 位地址中的最高 2 位。）
  - 要进入接收器模式，主设备先送头字节（11110xx0），然后送最低位为’1’的从地址。然后再重新发送一个开始条件，后面跟着头字节（11110xx1）（这里 xx 代表 10 位地址中的最高 2 位。）

TRA 位指示主设备是在接收器模式还是发送器模式。

25.2.3.3 主发送器

在发送了地址和清除了 ADDR 位后,主设备通过内部移位寄存器将字节从 DR 寄存器发送到 SDA线上。

主设备等待，直到 TxE 被清除，（见图 25-5 的 EV8）。

当收到应答脉冲时:

- TxE 位被硬件置位，如果设置了 INEVFEN 和 ITBUFEN 位，则产生一个中断。

  如果 TxE 被置位并且在上一次数据发送结束之前没有写新的数据字节到 DR 寄存器，则 BTF 被硬件置位，在清除 BTF 之前 I2C 接口将保持 SCL 为低电平；读出 I2Cx_SR1 之后再写入 I2Cx_DR 寄存器将清除 BTF 位。

25.2.3.4 关闭通信

在 DR 寄存器中写入最后一个字节后，通过设置 STOP 位产生一个停止条件（见图 25-5 的 EV8_2），然后 I2C 接口将自动回到从模式（M/S 位清除）。

注意：当 TxE 或 BTF 位置位时，停止条件应安排在出现 EV8_2 事件时。

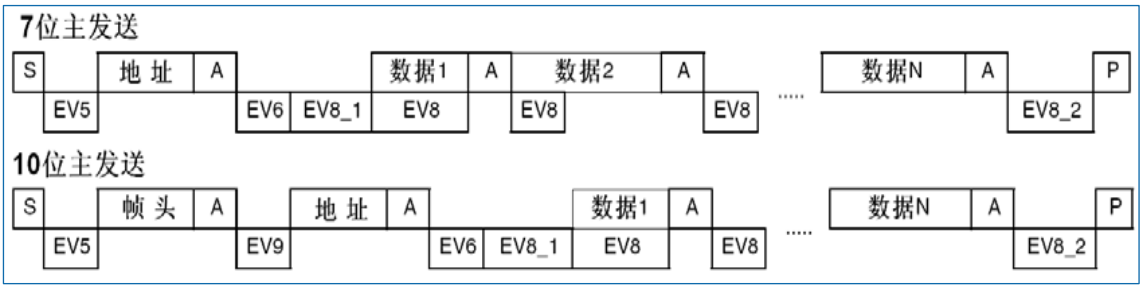


图 25-5 主发送器传送序列图

说明：

- S=Start（起始条件），Sr=重复的起始条件，P=Stop（停止条件），A=响应，NA=非响应，EVx=事件（ITEVTEN=1 时产生中断）。
- EV5：SB=1，读 SR1 然后将地址写入 DR 寄存器将清除该事件。
- EV6：ADDR=1，读 SR1 然后读 SR2 将清除该事件。
- EV8_1：TxE=1，移位寄存器空，数据寄存器空，写 DR 寄存器。
- EV8：TxE=1，移位寄存器非空，数据寄存器空，写入 DR 寄存器将清除该事件。
- EV8_2：TxE=1，BTF=1，请求设置停止位。TxE 和 BTF 位由硬件在产生停止条件时清除。
- EV9：ADDR10=1，读 SR1 然后写入 DR 寄存器将清除该事件。

  注意：

  EV5、EV6、EV9、EV8_1 和 EV8_2 事件拉长 SCL 低的时间，直到对应的软件序列结束。

  EV8 的软件序列必须在当前字节传输结束之前完成。

25.2.3.5 主接收器

在发送地址和清除 ADDR 之后，I2C 接口进入主接收器模式。在此模式下，I2C 接口从 SDA 线接收数据字节，并通过内部移位寄存器送至 DR 寄存器。在每个字节后，I2C 接口依次执行以下操作：

1. 如果 ACK 位被置位，发出一个应答脉冲。
2. 硬件设置 RxNE=1，如果设置了 INEVFEN 和 ITBUFEN 位，则会产生一个中断（见图 25-6 的 EV7）。

   如果 RxNE 位被置位，并且在接收新数据结束前，DR 寄存器中的数据没有被读走，硬件将设置 BTF=1，在清除 BTF 之前 I2C 接口将保持 SCL 为低电平；读出 I2Cx_SR1 之后再读出 I2Cx_DR 寄存器将清除 BTF 位。

25.2.3.6 关闭通信

主设备在从设备接收到最后一个字节后发送一个 NACK。接收到 NACK 后，从设备释放对 SCL 和 SDA 线的控制；主设备就可以发送一个停止/重起始条件。

- 为了在收到最后一个字节后产生一个 NACK 脉冲，在读倒数第二个数据字节之后（在倒数第 二个 RxNE 事件之后）必须清除 ACK 位。
- 为了产生一个停止/重起始条件，软件必须在读倒数第二个数据字节之后（在倒数第二个 RxNE 事件之后）设置 STOP/START 位。
- 在只接收一个字节数据的情况下，应该在 EV6 之后（在 EV6_1 中，即在 ADDR 位被清除之后）设置 NACK 和设置发送停止信号。

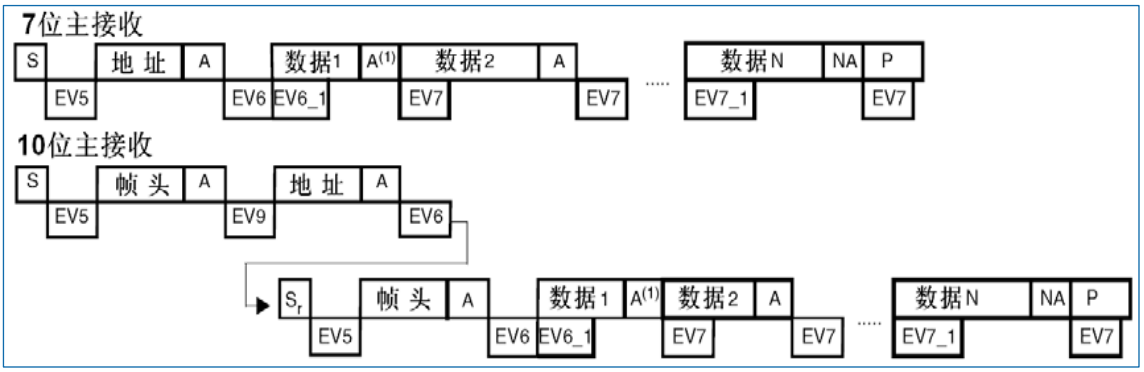


图 25-6 主接收器传送序列图

说明：

- S=Start（起始条件），Sr=重复的起始条件，P=Stop（停止条件），A=响应，NA=非响应，EVx=事件（ITEVTEN=1 时产生中断）
- EV5：SB=1，读 SR1 然后将地址写入 DR 寄存器将清除该事件。
- EV6：ADDR=1，读 SR1 然后读 SR2 将清除该事件。在 10 位主接收模式下，该事件后应设置 CR2 的 START=1。
- EV6_1：没有对应的事件标志，只适于接收 1 个字节的情况。恰好在 EV6 之后（即清除了 ADDR 之后），要清除响应和停止条件的产生位。
- EV7：RxNE=1，读 DR 寄存器清除该事件。
- EV7_1：RxNE=1，读 DR 寄存器清除该事件。设置 ACK=0 和 STOP 请求。
- EV9：ADDR10=1，读 SR1 然后写入 DR 寄存器将清除该事件。

  注意：

  如果收到一个单独的字节，则是 NA。

  EV5、EV6 和 EV9 事件拉长 SCL 低电平，直到对应的软件序列结束。

  EV7 的软件序列必须在当前字节传输结束前完成。

  EV6_1 或 EV7_1 的软件序列必须在当前传输字节的 ACK 脉冲之前完成。

25.2.4 错误条件

以下条件可能造成通讯失败。

25.2.4.1 总线错误（BERR）

在一个地址或数据字节传输期间，当 I2C 接口检测到一个外部的停止或起始条件则产生总线错误。此时：

- BERR 位被置位为'1'；如果设置了 ITERREN 位，则产生一个中断；
- 在从模式情况下，数据被丢弃，硬件释放总线：
  - 如果是错误的开始条件，从设备认为是一个重启动，并等待地址或停止条件。
  - 如果是错误的停止条件，从设备按正常的停止条件操作，同时硬件释放总线。
- 在主模式情况下，硬件不释放总线，同时不影响当前的传输状态。此时由软件决定是否要中止当前的传输。

25.2.4.2 应答错误（AF）

当接口检测到一个无应答位时，产生应答错误。此时：

- AF 位被置位，如果设置了 ITERREN 位，则产生一个中断；
- 当发送器接收到一个 NACK 时，必须复位通讯：
  - 如果是处于从模式，硬件释放总线。
  - 如果是处于主模式，软件必须生成一个停止条件。

25.2.4.3 仲裁丢失（ARLO）

当 I2C 接口检测到仲裁丢失时产生仲裁丢失错误，此时：

- ARLO 位被硬件置位，如果设置了 ITERREN 位，则产生一个中断；
- I2C 接口自动回到从模式（M/SL 位被清除）。当 I2C 接口丢失了仲裁，则它无法在同一个传输中响应它的从地址，但它可以在赢得总线的主设备发送重起始条件之后响应；
- 硬件释放总线。

25.2.4.4 过载/欠载错误（OVR）

在从模式下，如果禁止时钟延长，I2C 接口正在接收数据时，当它已经接收到一个字节（RxNE=1），但在 DR 寄存器中前一个字节数据还没有被读出，则发生过载错误。此时：

- 最后接收的数据被丢弃；
- 在过载错误时，软件应清除 RxNE 位，发送器应该重新发送最后一次发送的字节。

在从模式下，如果禁止时钟延长，I2C 接口正在发送数据时，在下一个字节的时钟到达之前，新的数据还未写入 DR 寄存器（TxE=1），则发生欠载错误。此时：

- 在 DR 寄存器中的前一个字节将被重复发出；
- 用户应该确定在发生欠载错时，接收端应丢弃重复接收到的数据。发送端应按 I2C 总线标准在规定的时间更新 DR 寄存器。

在发送第一个字节时，必须在清除 ADDR 之后并且第一个 SCL 上升沿之前写入 DR 寄存器；如果不能做到这点，则接收方应该丢弃第一个数据。

25.2.5 SDA/SCL 线控制

- 如果允许时钟延长：
  - 发送器模式：如果 TxE=1 且 BTF=1：I2C 接口在传输前保持时钟线为低，以等待软件读取 SR1，然后把数据写进数据寄存器（缓冲器和移位寄存器都是空的）。
  - 接收器模式：如果 RxNE=1 且 BTF=1：I2C 接口在接收到数据字节后保持时钟线为低，以等待软件读 SR1，然后读数据寄存器 DR（缓冲器和移位寄存器都是满的）。
- 如果在从模式中禁止时钟延长：
  - 如果 RxNE=1，在接收到下个字节前 DR 还没有被读出，则发生过载错。接收到的最后一个字节丢失。
  - 如果 TxE=1，在必须发送下个字节之前却没有新数据写进 DR，则发生欠载错。相同的字节将被重复发出。
  - 不控制重复写冲突。

通过配置 I2C_THOLD.THOLD[7:0]位来调节 SDA 高电平保持时间（默认值为一个 PCLK 周期），其最大值不得大于低电平保持时间。

25.2.6 SMBus

25.2.6.1 介绍

系统管理总线（SMBus）是一个双线接口。通过它，各设备之间以及设备与系统的其他部分之间

可以互相通信。它基于 I2C 操作原理。SMBus 为系统和电源管理相关的任务提供一条控制总线。一个系统利用 SMBus 可以和多个设备互传信息，而不需使用独立的控制线路。

SMBus 标准涉及以下三类设备。

- 从设备：接收或响应命令的设备。
- 主设备：用来发送命令、产生时钟和终止发送的设备。
- 主机：一种专用的主设备，它提供与系统 CPU 的主接口。主机必须具有主-从机功能并且必须支持 SMBus 提醒协议。一个系统里只允许有一个主机。

25.2.6.2 SMBus 和 I2C 之间的相似点

- 2 条线的总线协议（1 个时钟，1 个数据）加可选的 SMBus 提醒线；
- 主从通信，主设备提供时钟；
- 多主机功能
- SMBus 数据格式类似于 I2C 的 7 位地址格式（见图 25-1）；

25.2.6.3 SMBus 和 I2C 之间的不同点

下表列出了 SMBus 和 I2C 的不同点。

表 25-1 SMBus 与 I2C 的差异

SMBus	I2C
最大传输速度 100kHz	最大传输速度 400kHz
最小传输速度 10kHz	无最小传输速度
35ms 时钟低超时	无时钟超时
固定的逻辑电平	逻辑电平由 VDD 决定
不同的地址类型（保留的、动态的等）	7 位、10 位和广播呼叫从地址类型
不同的总线协议（快速命令、处理呼叫等）	无总线协议

25.2.6.4 SMBus 应用用途

利用系统管理总线，设备可提供制造商信息，告诉系统它的型号/部件号，保存暂停事件的状态，报告不同类型的错误，接收控制参数，和返回它的状态。SMBus 为系统和电源管理相关的任务提供控制总线。

25.2.6.5 设备标识

在系统管理总线上，任何一个作为从模式的设备都有一个唯一的地址，叫做从地址。保留的从地址表请参考 2.0 版的 SMBus 规范（http://smbus.org/specs/）。

25.2.6.6 总线协议

SMBus 技术规范支持 9 个总线协议。有关这些协议的详细资料和 SMBus 地址类型，请参考 2.0 版的 SMBus 规范（http://smbus.org/specs/）。这些协议由用户的软件来执行。

25.2.6.7 地址解析协议（ARP）

通过给每个从设备动态地分配一个新的唯一地址，可以解决 SMBus 的从地址冲突。地址解析协议（ARP）具有以下的特性：

- 使用标准 SMBus 物理层仲裁机制分配地址；
- 当设备维持供电期间，分配的地址仍保持不变，也允许设备在断电后保留其地址。
- 在地址分配后，没有额外的 SMBus 的打包开销（也就是说访问分配地址的设备与访问固定地址的设备所用时间是一样的）；
- 任何一个 SMBus 主设备可以遍历总线。

25.2.6.8 唯一的设备标识符（UDID）

为了分配地址，需要一种区分每个设备的机制，每个设备必须拥有一个唯一的设备标识符。

关于在 ARP 上 128 位的 UDID 的详细信息，参考 2.0 版的 SMBus 规范（http://smbus.org/specs/）。

25.2.6.9 SMBus 提醒模式

SMBus 提醒是一个带中断线的可选信号，用于那些希望扩展它们的控制能力而牺牲一个引脚的设备。SMBALERT 和 SCL、SDA 信号一样，是一种线与信号。SMBALERT 通常和 SMBus 广播呼叫地址一起使用。与 SMBus 有关的消息为 2 字节。

一个只具有从功能的设备，可以通过设置 I2Cx_CR1 寄存器上的 ALERT 位，使用 SMBALERT 给主机发信号表示它希望进行通信。主机处理该中断并通过提醒响应地址 ARA（*Alert Response Address*，地址值为 0001100x）访问所有 SMBALERT 设备。只有那些将 SMBALERT 拉低的设备能应答 ARA。此状态是由 I2Cx_SR1 寄存器中的 SMBALERT 状态标记来标识的。主机执行一个修改过的接收字节操作。由从发送设备提供的 7 位设备地址被放在字节的 7 个最高位上，第八个位可以是'0'或'1'。

如果多个设备把 SMBALERT 拉低，最高优先级设备（最小的地址）将在地址传输期间通过标准仲 裁赢得通信权。在确认从地址后，此设备不得再拉低它的 SMBALERT，如果当信息传输完成后，主机仍看到 SMBALERT 低，就知道需要再次读 ARA。

没有实现 SMBALERT 信号的主机可以定期访问 ARA。有关 SMBus 提醒模式的更多详细资料，请参考 2.0 版的 SMBus 规范（http://smbus.org/specs/）。

25.2.6.10 超时错误

在定时规范上 I2C 和 SMBus 之间有很多差别。

SMBus 定义了一个时钟低超时，35ms 的超时。SMBus 规定 T_{LOW}：SEXT 为从设备的累积时钟低扩展时间。SMBus 规定 T_{LOW}：MEXT 为主设备的累积时钟低扩展时间。更多超时细节请参考 2.0 版的 SMBus 规范（http://smbus.org/specs/）。

I2Cx_SR1 中的状态标志 Timeout 或 T_{Low} 错误表明了这个特性的状态。

25.2.6.11 如何使用 SMBus 模式的接口

为了从 I2C 模式切换到 SMBus 模式，应该执行下列步骤：

1. 设置 I2Cx_CR1 寄存器中的 SMBus 位；
2. 按应用要求配置 I2Cx_CR1 寄存器中的 SMBTYPE 和 ENARP 位。

如果要把设备配置成主设备，产生起始条件的步骤见章节："25.2.3 I2C 主模式"。否则，参见："25.2.2 I2C 从模式"。

软件程序必须处理多种 SMBus 协议。

- 如果 ENARP=1 且 SMBTYPE=0，使用 SMB 设备默认地址。
- 如果 ENARP=1 且 SMBTYPE=1，使用 SMB 主设备头字段。
- 如果 SMBALERT=1，使用 SMB 提醒响应地址。

25.2.7 DMA 请求

DMA 请求（当被使能时）仅用于数据传输。发送时数据寄存器变空或接收时数据寄存器变满，则产生 DMA 请求。DMA 请求必须在当前字节传输结束之前被响应。当为相应 DMA 通道设置的数据传输量已经完成时，DMA 控制器发送传输结束信号 ETO 到 I2C 接口，并且在中断允许时产生一个传输完成中断：

- 主发送器：在 EOT 中断服务程序中，需禁止 DMA 请求，然后在等到 BTF 事件后设置停止条件。
- 主接收器：当要接收的数据数目大于或等于 2 时，DMA 控制器发送一个硬件信号 EOT_1，它对应 DMA 传输（字节数－1）。如果在 I2Cx_CR2 寄存器中设置了 LAST 位，硬件在发送完 EOT_1 后的下一个字节，将自动发送 NACK。在中断允许的情况下，用户可以在 DMA 传输完成的中断服务程序中产生一个停止条件。

25.2.7.1 利用 DMA 发送

通过设置 I2Cx_CR2 寄存器中的 DMAEN 位可以激活 DMA 模式。只要 TxE 位被置位，数据将由 DMA 从预置的存储区装载进 I2Cx_DR 寄存器。为 I2C 分配一个 DMA 通道，须执行以下步骤（x 是通道号）：

1. 在 DMA_CHANNEL_CPAR 寄存器中设置 I2Cx_DR 寄存器地址。数据将在每个 TxE 事件后从存储器传 送至这个地址。
2. 在 DMA_CHANNEL_CMAR 寄存器中设置存储器地址。数据在每个 TxE 事件后从这个存储区传送至 I2Cx_DR。
3. 在 DMA_CHANNEL_CNDTR 寄存器中设置所需的传输字节数。在每个 TxE 事件后，此值将被递减。
4. 利用 DMA_CHANNEL_CCR 寄存器中的 PL[0:1]位配置通道优先级。
5. 设置 DMA_CHANNEL_CCR 寄存器中的 DIR 位，并根据应用要求可以配置在整个传输完成一半或全 部完成时发出中断请求。
6. 通过设置 DMA_CHANNEL_CCR 寄存器上的 EN 位激活通道。当 DMA 控制器中设置的数据传输数目已经完成时，DMA 控制器给 I2C 接口发送一个传输结束的EOT/EOT_1 信号。在中断允许的情况下，将产生一个 DMA 中断。

注意：如果使用 DMA 进行发送时，不要设置 I2Cx_CR2 寄存器的 ITBUFEN 位。

25.2.7.2 利用 DMA 接收

通过设置 I2Cx_CR2 寄存器中的 DMAEN 位可以激活 DMA 接收模式。每次接收到数据字节时，将由 DMA 把 I2Cx_DR 寄存器的数据传送到设置的存储区（参考 DMA 说明）。设置 DMA 通道进行 I2C 接收，须执行以下步骤（x 是通道号）：

1. 在 DMA_CHANNEL_CPAR 寄存器中设置 I2Cx_DR 寄存器的地址。数据将在每次 RxNE 事件后从此地址传送到存储区。
2. 在 DMA_CHANNEL_CMAR 寄存器中设置存储区地址。数据将在每次 RxNE 事件后从 I2Cx_DR 寄存器 传送到此存储区。
3. 在 DMA_CHANNEL_CNDTR 寄存器中设置所需的传输字节数。在每个 RxNE 事件后，此值将被递减。
4. 用 DMA_CHANNEL_CCR 寄存器中的 PL[0:1]配置通道优先级。

5. 清除 DMA_CHANNEL_CCR 寄存器中的 DIR 位，根据应用要求可以设置在数据传输完成一半或全部 完成时发出中断请求。

6. 设置 DMA_CHANNEL_CCR 寄存器中的 EN 位激活该通道。当 DMA 控制器中设置的数据传输数目已经完成时，DMA 控制器给 I2C 接口发送一个传输结束的EOT/ EOT_1 信号。在中断允许的情况下，将产生一个 DMA 中断。

注意：如果使用 DMA 进行接收时，不要设置 I2Cx_CR2 寄存器的 ITBUFEN 位。

25.2.8 包错误校验（PEC）

包错误校验（PEC）计算器是用于提高通信的可靠性，这个计算器使用下述 CRC-8 多项式对每一位串行数据进行计算：

$C(x) = x^8 + x^2 + x + 1$

- PEC 计算由 I2Cx_CR1 寄存器的 ENPEC 位激活。PEC 使用 CRC-8 算法对所有信息字节进行计算，包括地址和读/写位在内。
  - 在发送时：在最后一个 TxE 事件时设置 I2Cx_CR1 寄存器的 PEC 传输位，PEC 将在最后一个字节后被发送。
  - 在接收时：在最后一个 RxNE 事件之后设置 I2Cx_CR1 寄存器的 PEC 位，如果下个接收到的字节不等于内部计算的 PEC，接收器发送一个 NACK。如果是主接收器，不管校对的结果如何，PEC 后都将发送 NACK。PEC 位必须在接收当前字节的 ACK 脉冲之前设置。
- 在 I2Cx_SR1 寄存器中可获得 PECERR 错误标记/中断。
- 如果 DMA 和 PEC 计算器都被激活：
  - 在发送时：当 I2C 接口从 DMA 控制器处接收到 EOT 信号时，它在最后一个字节后自动发送 PEC。
  - 在接收时：当 I2C 接口从 DMA 处接收到一个 EOT_1 信号时，它将自动把下一个字节作为 PEC，并且将检查它。在接收到 PEC 后产生一个 DMA 请求。
- 为了允许中间 PEC 传输，在 I2Cx_CR2 寄存器中有一个控制位（LAST 位）用于判别是否真是最后一个 DMA 传输。如果确实是最后一个主接收器的 DMA 请求，在接收到最后一个字节后自动发送 NACK。
- 仲裁丢失时 PEC 计算失效。

25.3 I2C 中断请求

下表列出了所有的 I2C 中断请求

表 25-2 I2C 中断请求表

中断事件	事件标志	开启控制位
起始位已发送（主）	SB	ITEVTEN
地址已发送（主）或地址匹配（从）	ADDR	
10 位头段已发送（主）	ADD10	
已收到停止（从）	STOPF	
数据字节传输完成	BTF	
接收缓冲区非空	RxNE	ITEVTEN 和 ITBUFEN
发送缓冲区空	TxE	

中断事件	事件标志	开启控制位
总线错误	BERR	ITERREN
仲裁丢失（主）	ARLO	
响应失败	AF	
过载/欠载	OVR	
PEC 错误	PECERR	
超时/T_{low} 错误	TIMEOUT	
SMBus 提醒	SMBALERT	

注意：

SB、ADDR、ADD10、STOPF、BTF、RxNE 和 TxE 通过逻辑或汇到同一个中断通道中。

BERR、ARLO、AF、OVR、PECERR、TIMEOUT 和 SMBALERT 通过逻辑或汇到同一个中断通道中。

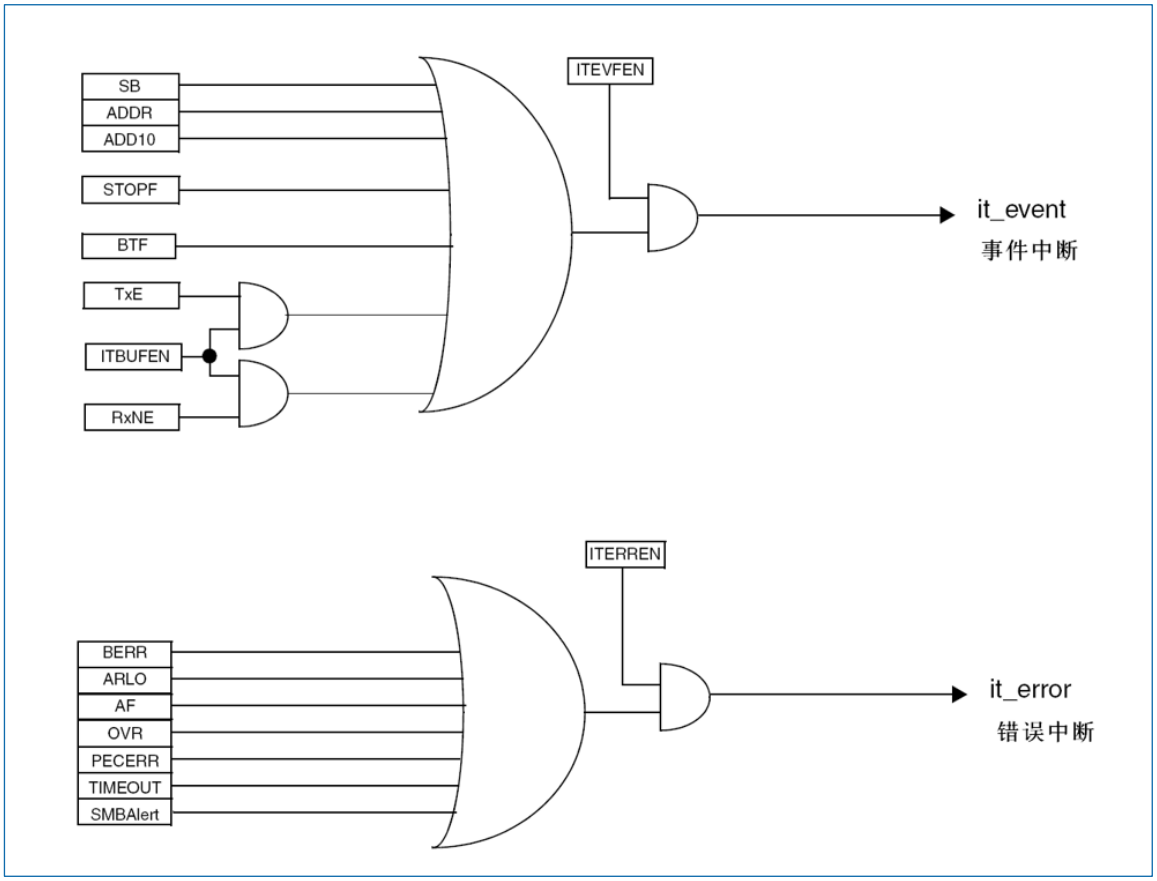


图 25-7 I2C 中断映射图

25.4 I2C 调试模式

当微控制器进入调试模式（Cortex®-M3 核心处于停止状态）时，根据 DBG 模块中的 DBG_I2Cx_SMBUS_TIMEOUT 配置位，SMBUS 超时控制或者继续正常工作或者可以停止。详见章节：“36.15.2 支持定时器、看门狗、CAN 和 I2C 的调试”。

25.5 I2C 寄存器

基地址：（I2C1, I2C2）=（0x4000 5400, 0x4000 5800）

空间大小：（I2C1, I2C2）=（0x400, 0x400）

可以用半字（16 位）或字（32 位）的方式操作这些外设寄存器。

25.5.1 控制寄存器 1（I2Cx_CR1）（x=1..2）

偏移地址：0x00

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWRST	Res	ALERT	PEC	POS	ACK	STOP	START	NOSTRETCH	ENGC	ENPEC	ENARP	SMBTYPE	Res	SMBUS	PE
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

位 15	SWRST：软件复位（Software reset） 当被置位时，I2C 处于复位状态。在复位该位前确信 I2C 的引脚被释放，总线是空的。 • 0：I2C 模块不处于复位状态 • 1：I2C 模块处于复位状态 *注意：该位可以用于 BUSY 位为'1'，在总线上又没有检测到停止条件时。*
位 14	Res：保留 必须保持复位值。
位 13	ALERT：SMBus 提醒（SMBus alert） 软件可以设置或清除该位；当 PE=0 时，由硬件清除。 • 0：释放 SMBAlert 引脚使其变高。提醒响应地址头紧跟在 NACK 信号后面 • 1：驱动 SMBAlert 引脚使其变低。提醒响应地址头紧跟在 ACK 信号后面
位 12	PEC：数据包出错检测（Packet error checking） 软件可以设置或清除该位；当传送 PEC 后，或起始或停止条件时，或当 PE=0 时硬件将其清除。 • 0：无 PEC 传输 • 1：PEC 传输（在发送或接收模式） *注意：仲裁丢失时，PEC 的计算失效。*
位 11	POS：应答/PEC 位置（用于数据接收）（Acknowledge/PEC Position of data reception） 软件可以设置或清除该位，或当 PE=0 时，由硬件清除。 • 0：ACK 位控制当前移位寄存器内正在接收的字节的（N）ACK PEC 位表明当前移位寄存器内的字节是 PEC。 • 1：ACK 位控制在移位寄存器里接收的下一个字节的（N）ACK PEC 位表明在移位寄存器里接收的下一个字节是 PEC。 *注意：POS 位只能用在 2 字节的接收配置中，必须在接收数据之前配置。* *为了 NACK 第 2 个字节，必须在清除 ADDR 为之后清除 ACK 位。为了检测第 2 个字节的 PEC，必须在配置了 POS 位之后，拉伸 ADDR 事件时设置 PEC 位。*
位 10	ACK：应答使能（Acknowledge enable） 软件可以设置或清除该位，或当 PE=0 时，由硬件清除。 • 0：无应答返回 • 1：在接收到一个字节后返回一个应答（匹配的地址或数据）。
位 9	STOP：停止条件产生（Stop generation） 软件可以设置或清除该位；或当检测到停止条件时，由硬件清除；当检测到超时错误时，硬件将其置

	位。
	• 在主模式下：
	  ○ 0：无停止条件产生
	  ○ 1：在当前字节传输或在当前起始条件发出后产生停止条件
	• 在从模式下：
	  ○ 0：无停止条件产生
	  ○ 1：在当前字节传输或释放 SCL 和 SDA 线
	注意：当设置了 STOP、START 或PEC 位，在硬件清除这个位之前，软件不要执行任何对I2Cx_CR1 的写操作；否则有可能会第 2 次设置 STOP、START 或 PEC 位。

位 8	START：起始条件产生（Start generation）
	软件可以设置或清除该位，或当起始条件发出后或 PE=0 时，由硬件清除。
	• 在主模式下：
	  ○ 0：无起始条件产生
	  ○ 1：重复产生起始条件
	• 在从模式下：
	  ○ 0：无起始条件产生
	  ○ 1：当总线空闲时，产生起始条件

位 7	NOSTRETCH：禁止时钟延长（从模式）（Clock stretching disable of slave mode） 该位用于当 ADDR 或 BTF 标志被置位，在从模式下禁止时钟延长，直到它被软件复位。 • 0：允许时钟延长 • 1：禁止时钟延长
位 6	ENGC：广播呼叫使能（General call enable） • 0：禁止广播呼叫，以非应答响应地址 00h • 1：允许广播呼叫，以应答响应地址 00h
位 5	ENPEC：PEC 使能（PEC enable） • 0：禁止 PEC 计算 • 1：开启 PEC 计算

位 4	ENARP：ARP 使能（ARP enable）
	• 0：禁止 ARP
	• 1：使能 ARP
	  ○ 如果 SMBTYPE=0，使用 SMBus 设备的默认地址。
	  ○ 如果 SMBTYPE=1，使用 SMBus 的主地址。

位 3	SMBTYPE：SMBus 类型（SMBus type） • 0：SMBus 设备 • 1：SMBus 主机
位 2	Res：保留 必须保持复位值。
位 1	SMBUS：SMBus 模式（SMBus mode） • 0：I2C 模式 • 1：SMBus 模式

位 0	PE：I2C 模块使能（Peripheral enable） • 0：禁用 I2C 模块； • 1：启用 I2C 模块：根据 SMBus 位的设置，相应的 I/O 口需配置为复用功能。 *注意：如果清除该位时通讯正在进行，在当前通讯结束后，I2C 模块被禁用并返回空闲状态。由于在通讯结束后发生 PE =0，所有的位被清除。在主模式下，通讯结束之前，绝不能清除该位。*

注意：当 STOP，START 或者 PEC 位被设置时，在这些位被清除前，软件不能写 I2Cx_CR1。否则会在第二次 STOP，START 或 PEC 请求时出现风险。

25.5.2 控制寄存器 2（I2Cx_CR2）（x=1..2）

偏移地址：0x04

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res			LAST	DMAEN	ITBUFEN	ITEVTEN	ITERREN	Res		FREQ[5:0]					
			rw	rw	rw	rw	rw			rw					

位 15:13	Res：保留 必须保持复位值。
位 12	LAST：DMA 最后一次传输（DMA last transfer） • 0：下一次 DMA 的 EOT 不是最后的传输 • 1：下一次 DMA 的 EOT 是最后的传输 *注意：该位在主接收模式使用，使得在最后一次接收数据时可以产生一个 NACK。*
位 11	DMAEN：DMA 请求使能（DMA requests enable） • 0：禁止 DMA 请求 • 1：当 TxE=1 或 RxNE =1 时，允许 DMA 请求。
位 10	ITBUFEN：缓冲器中断使能（Buffer interrupt enable） • 0：当 TxE=1 或 RxNE=1 时，不产生任何中断。 • 1：当 TxE=1 或 RxNE=1 时，产生事件中断（不管 DMAEN 是何种状态）。
位 9	ITEVTEN：事件中断使能（Event interrupt enable） • 0：禁止事件中断 • 1：允许事件中断 在下列条件下，将产生该中断： o SB=1（主模式） o ADDR=1（主/从模式） o ADD10= 1（主模式） o STOPF=1（从模式） o BTF=1，但是没有 TxE 或 RxNE 事件。 o 如果 ITBUFEN=1，TxE 事件为 1。 o 如果 ITBUFEN=1，RxNE 事件为 1。
位 8	ITERREN：出错中断使能（Error interrupt enable） • 0：禁止出错中断 • 1：允许出错中断 在下列条件下，将产生该中断：

	○ BERR=1 ○ ARLO=1 ○ AF=1 ○ OVR=1 ○ PECERR=1 ○ TIMEOUT=1 ○ SMBAlert=1
位 7:6	Res：保留 必须保持复位值。
位 5:0	FREQ[5:0]：I2C 模块时钟频率（Peripheral clock frequency） 必须设置正确的输入时钟频率以产生正确的时序，允许的范围在 2~50MHz 之间： • 000000：禁用 • 000001：禁用 • 000010：2MHz • ... • 110010：50MHz

25.5.3 自身地址寄存器 1（I2Cx_OAR1）（x=1..2）

偏移地址：0x08

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDMODE	Res					ADD[9:0]									
rw						rw									

位 15	ADDMODE：寻址模式（从模式）（Addressing mode of Slave mode） • 0：7 位从地址（不响应 10 位地址） • 1：10 位从地址（不响应 7 位地址）
位 14:10	Res：保留 必须保持复位值。
位 9:0	ADD[9:0]：接口地址（Interface address） • 7 位地址模式下：ADD[9:8]和 ADD[0]位域的值不起作用，ADD[7:1]位域写入要发送的七位从机地址位。 • 10 位地址模式下：ADD[9:0]应写入要发送的从机地址位。

25.5.4 自身地址寄存器 2（I2Cx_OAR2）（x=1..2）

偏移地址：0x0C

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								ADD2[7:1]							ENDUAL
								rw							rw

位 15:8	Res：保留 必须保持复位值。

位 7:1	ADD2[7:1]：接口地址（Interface address） 在双地址模式下地址的 7~1 位。
位 0	ENDUAL：双地址模式使能位（Dual addressing mode enable） • 0：在 7 位地址模式下，只有 OAR1 被识别 • 1：在 7 位地址模式下，OAR1 和 OAR2 都被识别

25.5.5 数据寄存器（I2Cx_DR）（x=1..2）

偏移地址：0x10

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								DR[7:0]							
								rw							

位 15:8	Res：保留 必须保持复位值。
位 7:0	DR[7:0]：8 位数据寄存器（8-bit data register） 用于存放接收到的数据或放置用于发送到总线的数据 • 发送器模式：当写一个字节至 DR 寄存器时，自动启动数据传输。一旦传输开始（TxE=1），如果能及时把下一个需传输的数据写入 DR 寄存器，I2C 模块将保持连续的数据流 • 接收器模式：接收到的字节被拷贝到 DR 寄存器（RxNE=1）。在接收到下一个字节（RxNE=1）之前读出数据寄存器，即可实现连续的数据传送 *注意：* • *在从模式下，地址不会被拷贝进数据寄存器 DR；* • *硬件不管理写冲突（如果 TxE=0，仍能写入数据寄存器）；* • *如果在处理 ACK 脉冲时发生 ARLO 事件，接收到的字节不会被拷贝到数据寄存器里，因此不能读到它。*

25.5.6 状态寄存器 1（I2Cx_SR1）（x=1..2）

偏移地址：0x14

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMBALERT	TIMEOUT	Res	PECERR	OVR	AF	ARLO	BERR	TxE	RxNE	Res	STOPF	ADD10	BTF	ADDR	SB
rc_w0	rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	r	r		r	r	r	r	r

位 15	SMBALERT：SMBus 提醒（SMBus alert）
	• 在 SMBus 主机模式下：
	  ○ 0：无 SMBus 提醒
	  ○ 1：在引脚上产生 SMBAlert 提醒事件
	• 在 SMBus 从机模式下：
	  ○ 0：没有 SMBAlert 响应地址头序列
	  ○ 1：收到 SMBAlert 响应地址头序列至 SMBAlert 变低
	该位由软件写’0’清除，或在 PE=0 时由硬件清除。

位 14	TIMEOUT：超时或 Tlow 错误（Timeout or Tlow error） • 0：无超时错误 • 1：SCL 处于低已达到25ms（超时） 或主机低电平累积时钟扩展时间超过 10ms（Tlow:mext）； 或从设备低电平累积时钟扩展时间超过 25ms（Tlow:sext）。 当在从模式下设置该位：从设备复位通讯，硬件释放总线。 当在主模式下设置该位：硬件发出停止条件。 该位由软件写'0'清除，或在 PE=0 时由硬件清除 *注意：这个功能仅在 SMBus 模式下运用。*
位 13	Res：保留 必须保持复位值。
位 12	PECERR：在接收时发生 PEC 错误（PEC Error in reception） • 0：无 PEC 错误：接收到 PEC 后接收器返回 ACK（如果 ACK=1） • 1：有 PEC 错误：接收到 PEC 后接收器返回 NACK（不管 ACK 是什么值） 该位由软件写'0'清除，或在 PE=0 时由硬件清除。

位 11	OVR：过载/欠载（Overrun/Underrun）
	• 0：无过载/欠载
	• 1：出现过载/欠载
	• 当 NOSTRETCH=1 时，在从模式下该位被硬件置位，同时：
	  ○ 在接收模式中当收到一个新的字节时（包括 ACK 应答脉冲），数据寄存器里的内容还未被读出，则新接收的字节将丢失。
	  ○ 在发送模式中当要发送一个新的字节时，却没有新的数据写入数据寄存器，同样的字节将被发送两次。
	该位由软件写'0'清除，或在 PE=0 时由硬件清除。
	注意：如果数据寄存器的写操作发生时间非常接近 SCL 的上升沿，发送的数据是不确定的，并发生保持时间错误。

位 10	AF：应答失败（Acknowledge failure） • 0：没有应答失败 • 1：应答失败 当没有返回应答时，硬件将置该位为'1'。 该位由软件写'0'清除，或在 PE=0 时由硬件清除。
位 9	ARLO：仲裁丢失（主模式）（Arbitration lost of Master mode） • 0：没有检测到仲裁丢失 • 1：检测到仲裁丢失 当接口失去对总线的控制给另一个主机时，硬件将将置该位为'1'。 该位由软件写'0'清除，或在 PE=0 时由硬件清除。在 ARLO 事件之后，I2C 接口自动切换回从模式（M/SL=0） *注意：在 SMBUS 模式下，在从模式下对数据的仲裁仅仅发生在数据阶段，或应答传输区间（不包括地址的应答）。*
位 8	BERR：总线出错（Bus error） • 0：无起始或停止条件出错 • 1：起始或停止条件出错

	当接口检测到错误的起始或停止条件，硬件将该位置’1’ 该位由软件写’0’清除，或在 PE=0 时由硬件清除。
位 7	TxE：数据寄存器为空（发送时）（Data register empty of transmitters） • 0：数据寄存器非空 • 1：数据寄存器。 在发送数据时，数据寄存器为空时该位被置’1’，在发送地址阶段不设置该位。 软件写数据到 DR 寄存器可清除该位；或在发生一个起始或停止条件后，或当 PE=0 时由硬件自动清除。 如果收到一个 NACK，或下一个要发送的字节是 PEC（PEC=1），该位不被置位。 *注意：在写入第 1 个要发送的数据后，或设置了 BTF 时写入数据，都不能清除 TxE 位，这是因为数据寄存器仍然为空。*
位 6	RxNE：数据寄存器非空（接收时）（Data register not empty of receivers） • 0：数据寄存器为空； • 1：数据寄存器非空。 在接收时，当数据寄存器不为空，该位被置’1’。在接收地址阶段，该位不被置位。软件对数据寄存器的读写操作清除该位，或当 PE=0 时由硬件清除。 在发生 ARLO 事件时，RxNE 不被置位。 *注意：当设置了 BTF 时，读取数据不能清除 RxNE 位，因为数据寄存器仍然为满。*
位 5	Res：保留 必须保持复位值。
位 4	STOPF：停止条件检测位（从模式）（Stop detection of Slave mode） • 0：没有检测到停止条件； • 1：检测到停止条件。 在一个应答之后（如果 ACK=1），当从设备在总线上检测到停止条件时，硬件将该位置’1’。 软件读取 SR1 寄存器后，对 CR1 寄存器的写操作将清除该位，或当 PE=0 时，硬件清除该位。 *注意：在收到 NACK 后，STOPF 位不被置位。推荐在 STOPF 被设置后，执行完整清除序列（READ SR1，然后 WRITE CR1）*
位 3	ADD10：10 位头序列已发送（主模式）（10-bit header sent in Master mode） • 0：没有 ADD10 事件发生； • 1：主设备已经将第一个地址字节发送出去。 在 10 位地址模式下，当主设备已经将第一个字节发送出去时，硬件将该位置’1’。 软件读取 SR1 寄存器后，对 CR1 寄存器的写操作将清除该位，或当 PE=0 时，硬件清除该位。 *注意：收到一个 NACK 后，ADD10 位不被置位。*
位 2	BTF：字节发送结束（Byte transfer finished） • 0：字节发送未完成 • 1：字节发送结束 当 NOSTRETCH=0 时，在下列情况下硬件将该位置’1’： • 在接收时，当收到一个新字节（包括 ACK 脉冲）且数据寄存器还未被读取（RxNE=1） • 在发送时，当一个新数据将被发送且数据寄存器还未被写入新的数据（TxE=1） 在软件读取 SR1 寄存器后，对数据寄存器的读或写操作将清除该位；或在传输中发送一个起始或停止条件后，或当 PE=0 时，由硬件清除该位 *注意：在收到一个 NACK 后，BTF 位不会被置位。如果下一个要传输的字节是 PEC（I2Cx_SR2 寄存器*

	中 TRA 为'1'，同时 I2Cx_CR1 寄存器中 PEC 为'1'），BTF 位不会被置位。
位 1	ADDR：地址已被发送（主模式）/地址匹配（从模式）（Address sent of Master mode/matched of Slave mode） 在软件读取 SR1 寄存器后，对 SR2 寄存器的读操作将清除该位，或当 PE=0 时，由硬件清除该位。 • 地址匹配（从模式） ○ 0：地址不匹配或没有收到地址 ○ 1：收到的地址匹配 当收到的从地址与 OAR 寄存器中的内容相匹配、或发生广播呼叫、或 SMBus 设备默认地址。或 SMBus 主机识别出 SMBus 提醒时，硬件就将该位置'1'（当对应的设置被使能时）。 • 地址已被发送（主模式） ○ 0：地址发送没有结束 ○ 1：地址发送结束 10 位地址模式时，当收到地址的第二个字节的 ACK 后该位被置'1'。7 位地址模式时，当收到地址的 ACK 后该位被置'1'。 *注意：在收到 NACK 后，ADDR 位不会被置位。*
位 0	SB：起始位（主模式）（Start bit of Master mode） • 0：未发送起始条件 • 1：起始条件已发送 当发送出起始条件时该位被置'1'。 软件读取 SR1 寄存器后，写数据寄存器的操作将清除该位，或当 PE=0 时，硬件清除该位。

25.5.7 状态寄存器 2（I2Cx_SR2）（x=1..2）

偏移地址：0x18

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PEC[7:0]								DUALF	SMBHOST	SMBDEFAULT	GENCALL	Res	TRA	BUSY	MSL
r								r	r	r	r		r	r	r

位 15:8	PEC[7:0]：数据包出错检测（Packet error checking register） 当 ENPEC=1 时，PEC[7:0]存放内部的 PEC 的值。
位 7	DUALF：双标志（从模式）（Dual flag of Slave mode） • 0：接收到的地址与 OAR1 内的内容相匹配 • 1：接收到的地址与 OAR2 内的内容相匹配 在产生一个停止条件或一个重复的起始条件时，或 PE=0 时，硬件将该位清除。
位 6	SMBHOST：SMBus 主机头系列（从模式）（SMBus host header of slave mode） • 0：未收到 SMBus 主机的地址 • 1：当 SMBTYPE=1 且 ENARP=1 时，收到 SMBus 主机地址 在产生一个停止条件或一个重复的起始条件时，或 PE=0 时，硬件将该位清除。
位 5	SMBDEFAULT：SMBus 设备默认地址（从模式）（SMBus device default address of Slave mode） • 0：未收到 SMBus 设备的默认地址。 • 1：当 ENARP=1 时，收到 SMBus 设备的默认地址。 在产生一个停止条件或一个重复的起始条件时，或 PE=0 时，硬件将该位清除。

位 4	GENCALL：广播呼叫地址（从模式）（General call address of slave mode） • 0：未收到广播呼叫地址 • 1：当 ENGC=1 时，收到广播呼叫的地址 在产生一个停止条件或一个重复的起始条件时，或 PE=0 时，硬件将该位清除。
位 3	Res：保留 必须保持复位值。
位 2	TRA：发送/接收（Transmitter/receiver） • 0：接收到数据 • 1：数据已发送 在整个地址传输阶段的结尾，该位根据地址字节的 R/W 位来设定。 在检测到停止条件（STOPF=1）、重复的起始条件或总线仲裁丢失（ARLO=1）后，或当 PE=0 时，硬件将其清除。
位 1	BUSY：总线忙（Bus busy） • 0：在总线上无数据通讯； • 1：在总线上正在进行数据通讯。 在检测到 SDA 或 SCl 为低电平时，硬件将该位置'1'； 当检测到一个停止条件时，硬件将该位清除。 该位指示当前正在进行的总线通讯，当接口被禁用（PE=0）时该信息仍然被更新。
位 0	MSL：主从模式（Master/slave） • 0：从模式； • 1：主模式。 当接口处于主模式（SB=1）时，硬件将该位置位； 当总线上检测到一个停止条件、仲裁丢失（ARLO=1 时）、或当 PE=0 时，硬件清除该位。

25.5.8 时钟控制寄存器（I2Cx_CCR）（x=1..2）

偏移地址：0x1C

复位值：0x0000

注意：

要求 FPCLK1 至少应当是 2MHz 去实现 SM 模式 I2C 时钟，要求 FPCLK1 至少应当是 4MHz 去实现 FM 模式 I2C 时钟，这样可以正确地产生 400KHz 的快速时钟；要求 FPCLK1 应当是 10MHz 的整数倍，这样可以正确地产生最大的 400KHz 的 I2C FM 时钟。

CCR 寄存器只有在关闭 I2C 时（PE=0）才能设置。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F_S	DUTY	Res		CCR[11:0]											
rw	rw			rw											

位 15	F_S：I2C 主模式选项（I2C master mode selection） • 0：标准模式（SM）的 I2C • 1：快速模式（FM）的 I2C
位 14	DUTY：快速模式时的占空比（Fast mode duty cycle） • 0：快速模式下：$T_{low}/T_{high}=2$ • 1：快速模式下：$T_{low}/T_{high}=16/9$（见 CCR）

位 13:12	Res：保留 必须保持复位值。
位 11:0	CCR[11:0]：快速/标准模式下的时钟控制分频系数（主模式）（Clock control register in fast/standard mode of Master mode） 该分频系数用于设置主模式下的 SCL 时钟。 • 在 I2C 标准模式或 SMBus 模式下 $T_{high}=CCR*T_{PCLK1}$; $T_{low}=CCR * T_{PCLK1}$ • 在 I2C 快速模式下 o 如果 DUTY=0 $T_{high}=CCR * T_{PCLK1}$; $T_{low}=2 * CCR * T_{PCLK1}$ o 如果 DUTY=1（速度达到 400kHz） $T_{high}=9 * CCR \times T_{PCLK1}$; $T_{low}=16 * CCR \times T_{PCLK1}$ 例如：在标准模式下，产生 100kHz 的 SCL 的频率： 如果 FREQR=08，T_{PCLK1}=125ns，则 CCR 必须写入 0x28（40 *125ns=5000 ns）。 *注意：* *1. 允许设定的最小值为 0x04，在快速 DUTY 模式下允许的最小值为 0x01；* *2. $T_{high}=t_{r（SCL）}+t_{w（SCLH）}$，详见数据手册中对这些参数的定义；* *3. $T_{low}=t_{f（SCL）}+t_{w（SCLL）}$，详见数据手册中对这些参数的定义；* *4. I2C 通讯速度，FSCL~1/（$T_{high}+T_{low}$）,因为模拟噪声滤波输入的延时，实际的频率可能会有点偏差；* *5. 只有在关闭 I2C 时（PE=0）才能设置 CCR 寄存器；*

25.5.9 TRISE 寄存器（I2Cx_TRISE）（x=1..2）

偏移地址：0x20

复位值：0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res										TRISE[5:0]					
										rw					

位 15:6	Res：保留 必须保持复位值。
位 5:0	TRISE[5:0]：在快速/标准模式下的最大上升时间（主模式）（Maximum rise time in fast/standard mode of Master mode） 这些位必须设置为 I2C 总线规范里给出的最大的 SCL 上升时间，增长步幅为 1。 例如：标准模式中最大允许 SCL 上升时间为 1000ns。如果在 I2Cx_CR2 寄存器中 FREQ[5:0]中的值等于 0x08 且 T_{PCLK1}=125ns，故 TRISE[5:0]中必须写入 09h（1000ns/125 ns=8+1）。滤波器的值也可以加到 TRISE[5:0]内。如果结果不是一个整数，则将整数部分写入 TRISE[5:0]以确保 t_{HIGH} 参数。 *注意：只有当 I2C 被禁用（PE=0）时，才能设置 TRISE[5:0]。*

25.5.10 THOLD 寄存器（I2Cx_THOLD）（x=1..2）

偏移地址：0x24

复位值：0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								THOLD[7:0]							
								rw							

位 15:8	Res：保留 必须保持复位值。
位 7:0	THOLD[7:0]：SDA 高电平保持时间（SDA high level hold time） *高电平保持时间，范围为0~127 个 PCLK，不得大于低电平保持时间。*

26 通用同步异步收发器（USART）

26.1 USART 介绍

通用同步异步收发器（USART）提供了一种灵活的方法与使用工业标准 NRZ 异步串行数据格式的外部设备之间进行全双工数据交换。USART 利用分数波特率发生器提供宽范围的波特率选择。

它支持同步单向通信和半双工单线通信，也支持 LIN（局部互连网），智能卡协议和 IrDA（红外数据组织）SIR ENDEC 规范，以及调制解调器（CTS/RTS）操作。它还允许多处理器通信。

使用多缓冲器配置的 DMA 方式，可以实现高速数据通信。

26.2 USART 主要特性

- 全双工异步通信
- NRZ 标准格式（标记/空格）
- 16 倍过采样率
- 分数波特率发生器系统
  - 发送和接收共用的可编程波特率，最高波特率为最高时钟频率除以 16
- 可编程数据字长度（8 位或 9 位）
- 可配置的停止位：支持 1 或 2 个停止位
- LIN 主同步断开符发送能力以及 LIN 从断开符检测能力
  - 当 USART 硬件配置成 LIN 时，生成 13 位断开符；检测 10/11 位断开符
- 发送方为同步传输提供时钟
- IrDA SIR 编码器解码器
  - 在正常模式下支持 3/16 位的持续时间
- 智能卡模拟功能
  - 智能卡接口支持 ISO7816-3 标准里定义的异步智能卡协议
  - 智能卡用到的 0.5 和 1.5 个停止位
- 单线半双工通信
- 可配置的使用 DMA 的多缓冲器通信
  - 在 SRAM 里利用集中式 DMA 缓冲接收/发送字节
- 单独的发送器和接收器使能位
- 检测标志
  - 接收缓冲器满
  - 发送缓冲器空
  - 传输结束标志
- 校验控制
  - 发送校验位
  - 对接收数据进行校验
- 四个错误检测标志
  - 溢出错误
  - 噪音错误

  - 帧错误
  - 校验错误
- 10 个带标志的中断源
  - CTS 改变
  - LIN 断开符检测
  - 发送数据寄存器空
  - 发送完成
  - 接收数据寄存器满
  - 检测到总线为空闲
  - 溢出错误
  - 帧错误
  - 噪音错误
  - 校验错误
- 多处理器通信：如果地址不匹配，则进入静默模式
- 从静默模式中唤醒（通过空闲总线检测或地址标志检测）
- 两种唤醒接收器的方式：
  - 地址位（MSB，第 9 位）
  - 总线空闲

26.3 USART 功能概述

接口通过三个引脚与其他设备连接在一起（详见图 26-1）。任何 USART 双向通信至少需要两个管脚：接收数据输入（RX）和发送数据输出（TX）：

- RX：接收数据串行输入。通过过采样技术来区别数据和噪音，从而恢复数据。
- TX：发送数据输出。当发送器被禁止时，输出引脚恢复到它的 I/O 端口配置。当发送器被激活，但无待发送数据时，TX 引脚处于高电平。在单线和智能卡模式里，此 I/O 口被同时用于数据的发送和接收（在 USART 层次，数据在 SW_RX 上收到）。

串行数据经由这些管脚在常规 USART 模式下被发送和接收，其框架组成如下：

- 一个在发送或接收前应处于空闲状态的总线
- 一个起始位
- 一个数据字（8 或 9 位），最低有效位在前
- 0.5，1，1.5，2 个的停止位，由此表明数据帧的结束
- 分数波特率发生器（12 位整数和 4 位小数）
- 一个状态寄存器（USARTx_SR）
- 数据寄存器（USARTx_DR）
- 一个波特率寄存器（USARTx_BRR）（12 位的整数和 4 位小数）
- 一个智能卡模式下的保护时间寄存器（USARTx_GTPR）

关于以上寄存器中每个位的具体定义，请参考寄存器描述章节：“26.6 USART 寄存器描述”。

在同步模式中需要下列引脚：

- CK：发送器时钟输出。此引脚输出用于 SPI 主模式下同步传输的时钟，（在 Start 位和 Stop 位上没有时钟脉冲，软件可选地实现在最后一个数据位送出一个时钟脉冲）。数据可以在 RX 上同步被接收。这可以用来控制带有移位寄存器的外部设备（例如 LCD 驱动器）。时钟相位和极性都

是软件可编程的。在智能卡模式里，CK 可以为智能卡提供时钟。

在硬件流控模式中需要下列引脚：

- CTS：清除发送，若是高电平，在当前数据传输结束时阻断下一次的数据发送。
- RTS：发送请求，若是低电平，表明 USART 准备好接收数据。

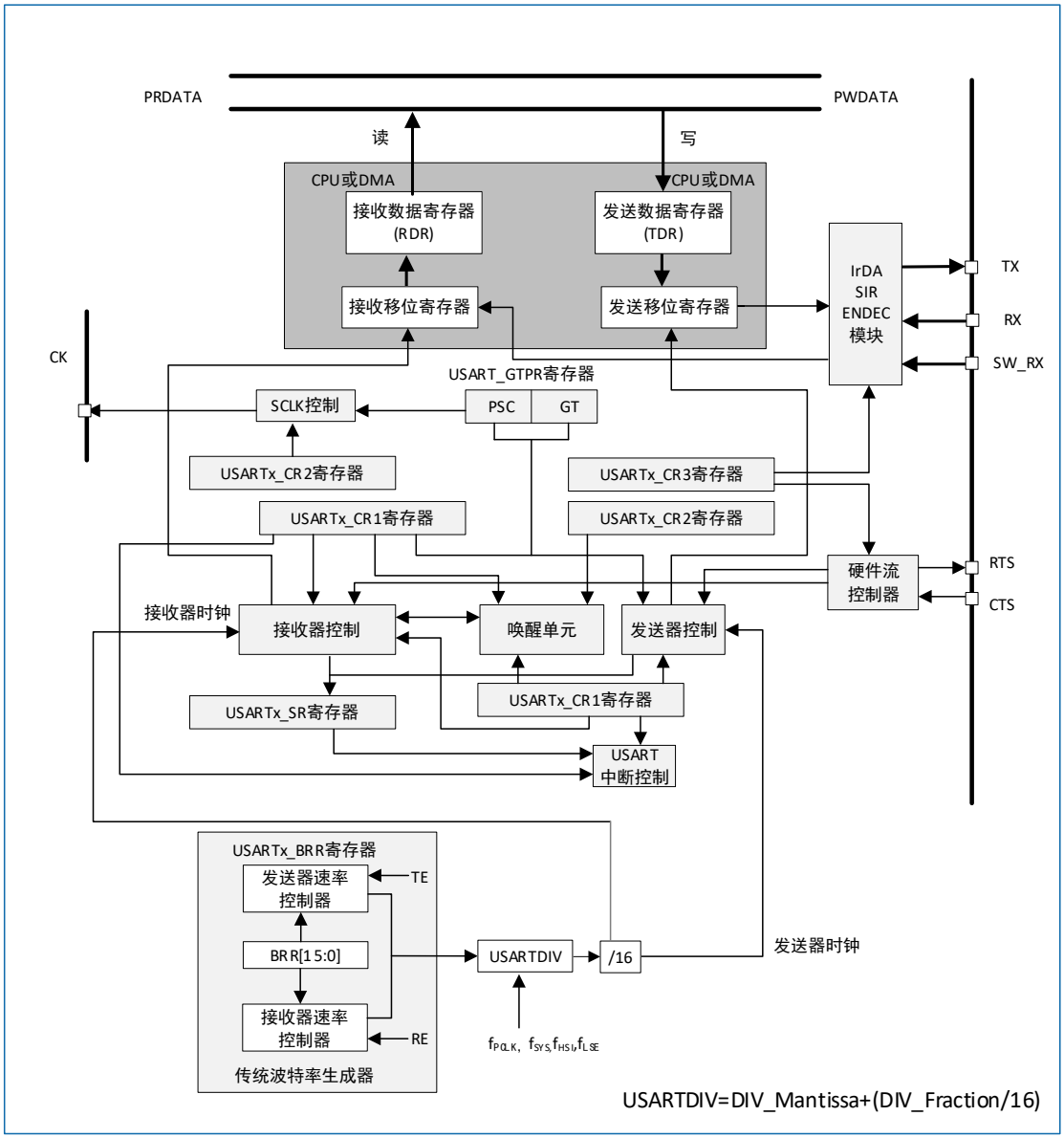


图 26-1 USART 框图

26.3.1 USART 特性描述

字长可以通过编程 USARTx_CR1 寄存器中的 M 位，选择成 8 或 9 位（详见图 26-2）。TX 脚在起始位期间处于低电平，在停止位期间处于高电平。

空闲符号被视为完全由'1'组成的一个完整的数据帧，后面跟着包含了数据的下一帧的开始位（'1'的位数也包括了停止位的位数）。

断开符号被视为在一个帧周期内全部收到'0'（包括停止位期间，也是'0'）。在断开帧结束时，发送器再插入 1 或 2 个停止位（'1'）来应答起始位。

发送和接收由一共用的波特率发生器驱动，当发送器和接收器的使能位分别置位时，分别为其产生时钟。随后将有每个功能块的详细说明。

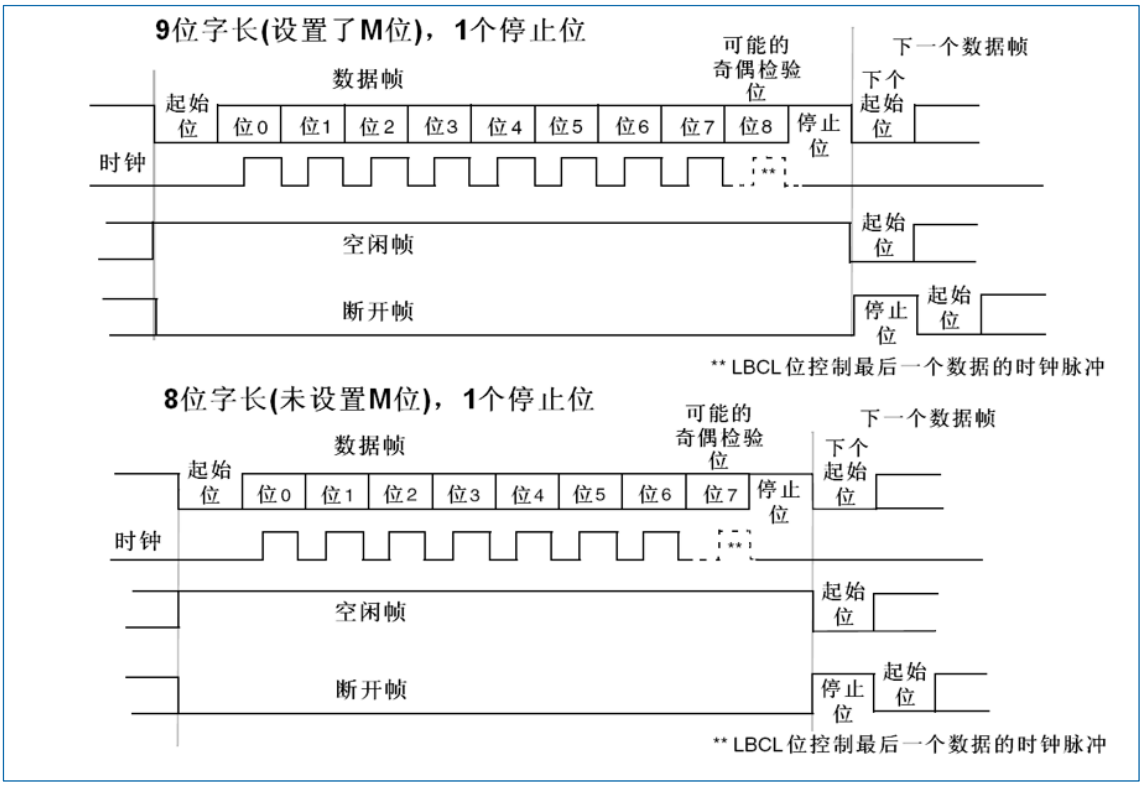


图 26-2 字长设置

26.3.2 发送器

发送器根据 M 位的状态发送 8 位或 9 位的数据字。当发送使能位（TE）被设置时，发送移位寄存器中的数据在 TX 脚上输出，相应的时钟脉冲在 CK 脚上输出。

26.3.2.1 字符发送

在 USART 发送期间，在 TX 引脚上首先移出数据的最低有效位。在此模式里，USARTx_DR 寄存器包含了一个内部总线和发送移位寄存器之间的缓冲器（详见图 26-1）。

每个字符之前都有一个低电平的起始位；之后跟着的停止位，其数目可配置。

USART 支持多种停止位的配置：0.5、1、1.5 和 2 个停止位。

注意：

1．在数据传输期间不能复位 TE 位，否则将破坏 TX 脚上的数据，因为波特率计数器停止计数。正在传输的当前数据将丢失。

2．TE 位被激活后将发送一个空闲帧。

26.3.2.2 可配置的停止位

随每个字符发送的停止位的位数可以通过控制寄存器 2 的位 13、12 进行编程。

1. 1 个停止位：停止位位数的默认值。
2. 2 个停止位：可用于常规 USART 模式、单线模式以及调制解调器模式。
3. 0.5 个停止位：在智能卡模式下接收数据时使用。
4. 1.5 个停止位：在智能卡模式下发送和接收数据时使用。

空闲帧包括了停止位。

断开帧是 10 位低电平，后跟停止位（当 m=0 时）；或者 11 位低电平，后跟停止位（m=1 时）。不可能传输更长的断开帧（长度大于 10 或者 11 位）。

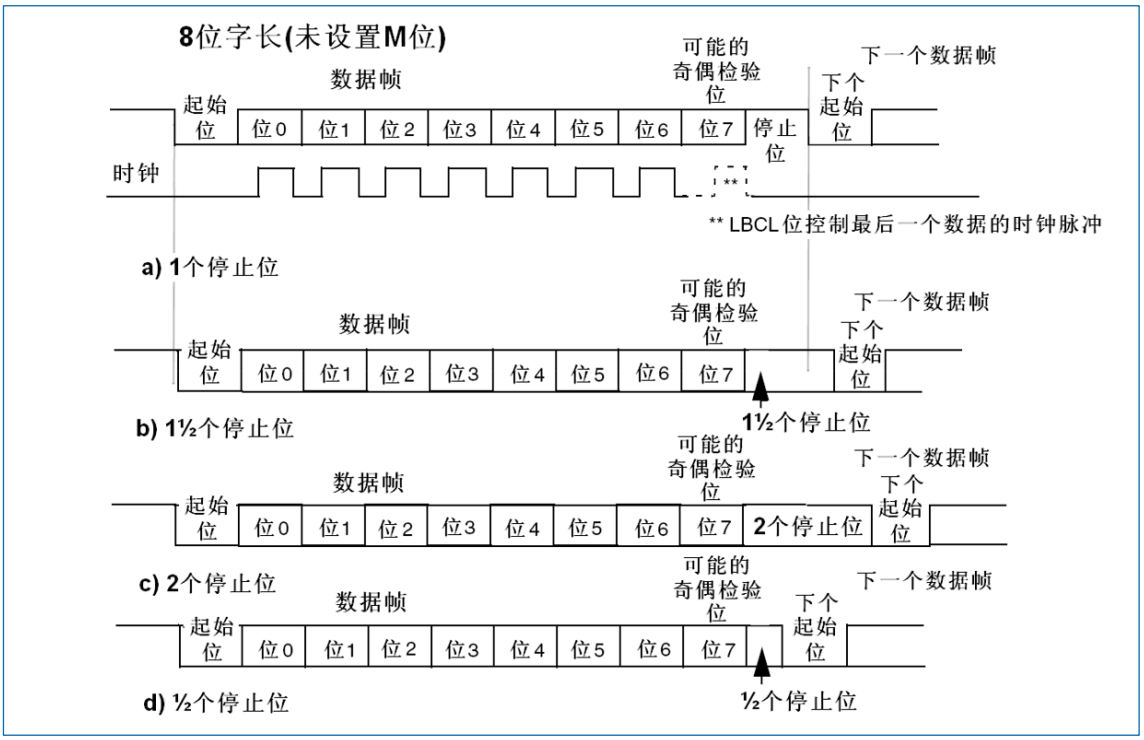


图 26-3 配置停止位

配置步骤：

1. 通过在 USARTx_CR1 寄存器上置 UE 位为’1’来激活 USART。
2. 编程 USARTx_CR1 的 M 位来定义字长。
3. 设置 USARTx_CR2 中 STOP 位，编程停止位的位数。
4. 如果采用多缓冲器通信，配置 USARTx_CR3 中的 DMA 使能位（DMAT）。按多缓冲器通信中的描述配置 DMA 寄存器。
5. 利用 USARTx_BRR 寄存器选择要求的波特率。
6. 设置 USARTx_CR1 中的 TE 位，发送一个空闲帧作为第一次数据发送。
7. 把要发送的数据写进 USARTx_DR 寄存器（此动作清除 TXE 位）。在只有一个缓冲器的情况下，对每个待发送的数据重复该步骤。
8. 在 USARTx_DR 寄存器中写入最后一个数据字后，要等待 TC=1，它表示最后一个数据帧的传输结束。当需要关闭 USART 或需要进入停机模式之前，需要确认传输结束，避免破坏最后一次传输。

26.3.2.3 单字节通信

清零 TXE 位总是通过对数据寄存器的写操作来完成的。TXE 位由硬件来设置，它表明：

- 数据已经从 TDR 移送到移位寄存器，数据发送已经开始。
- TDR 寄存器被清空。
- 下一个数据可以被写进 USARTx_DR 寄存器而不会覆盖先前的数据。

如果 TXEIE 位被设置，此标志将产生一个中断。

如果此时 USART 正在发送数据，对 USARTx_DR 寄存器的写操作把数据存进 TDR 寄存器，并在当前传输结束时把该数据复制进移位寄存器。

如果此时 USART 没有在发送数据，处于空闲状态，对 USARTx_DR 寄存器的写操作直接把数据放进移位寄存器，数据传输开始，TXE 位立即被置起。

当一帧发送完成时（停止位发送后）并且设置了 TXE 位，TC 位被置起，如果 USARTx_CR1 寄存器中的 TCIE 位被置起时，则会产生中断。在 USARTx_DR 寄存器中写入了最后一个数据字后，在关闭 USART 模块之前或设置微控制器进入低功耗模式（详见下图）之前，必须先等待 TC=1。

使用下列软件过程清除 TC 位：

1. 读一次 USARTx_SR 寄存器；
2. 写一次 USARTx_DR 寄存器。

注意：TC 位也可以通过软件对它写'0'来清除。此清零方式只推荐在多缓冲器通信模式下使用。

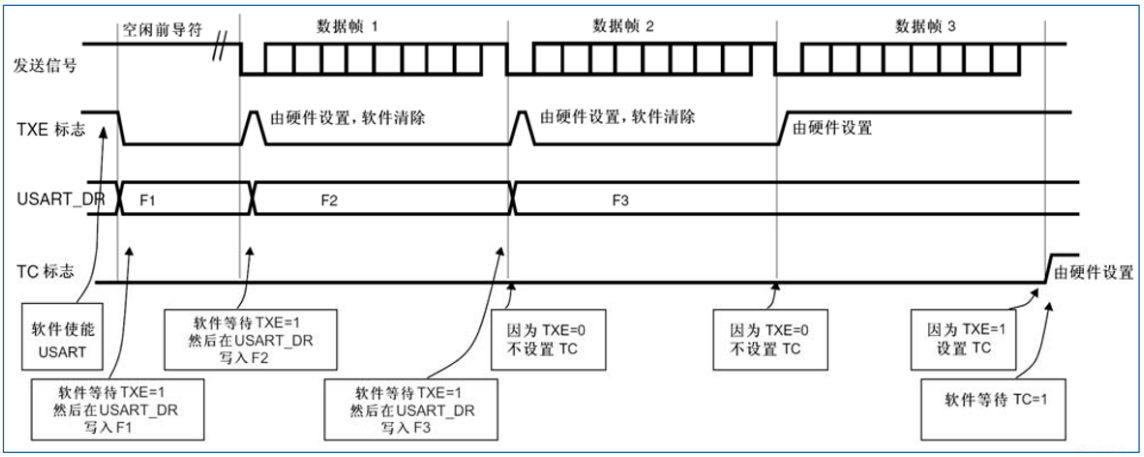


图 26-4 发送时 TC/TXE 的变化情况

26.3.2.4 断开符号

设置 SBK 可发送一个断开符号。断开帧长度取决 M 位（见图 26-2）。如果设置 SBK=1，在完成当前数据发送后，将在 TX 线上发送一个断开符号。断开字符发送完成时（在断开符号的停止位时）SBK 被硬件复位。USART 在最后一个断开帧的结束处插入一逻辑'1'，以保证能识别下一帧的起始位。

注意：如果在开始发送断开帧之前，软件又复位了 SBK 位，断开符号将不被发送。如果要发送两个连续的断开帧，SBK 位应该在前一个断开符号的停止位之后置位。

26.3.2.5 空闲符号

置位 TE 将使得 USART 在第一个数据帧前发送一空闲帧。

26.3.3 接收器

USART 可以根据 USARTx_CR1 的 M 位接收 8 位或 9 位的数据字。

26.3.3.1 起始位侦测

在 USART 中，如果辨认出一个特殊的采样序列，那么就认为侦测到一个起始位。该序列为：1110X0X0X0000

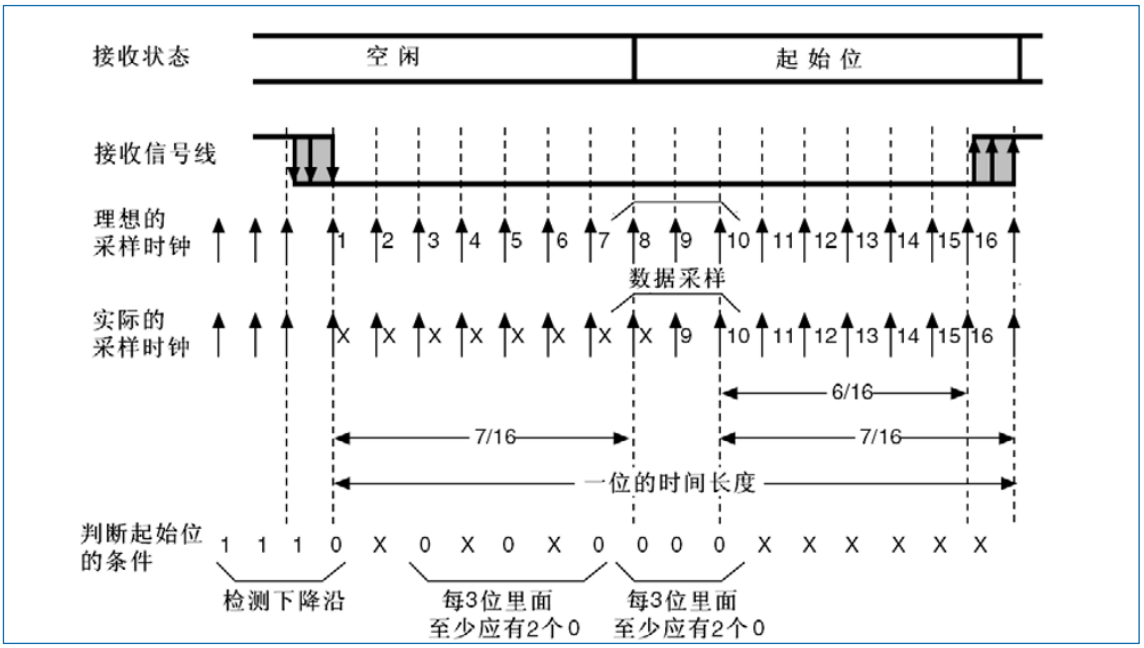


图 26-5 起始位侦测

注意：

如果该序列不完整，那么接收端将退出起始位侦测并回到空闲状态（不设置标志位）等待下降沿。

如果 3 个采样点都为'0'（在第 3、5、7 位的第一次采样，和在第 8、9、10 的第二次采样都为'0'），则确认收到起始位，这时设置 RXNE 标志位，如果 RXNEIE=1，则产生中断。

如果两次 3 个采样点上仅有 2 个是'0'（第 3、5、7 位的采样点和第 8、9、10 位的采样点），那么起始位仍然是有效的，但是会设置 NE 噪声标志位。如果不能满足这个条件，则中止起始位的侦测过程，接收器会回到空闲状态（不设置标志位）。

如果有一次 3 个采样点上仅有 2 个是'0'（第 3、5、7 位的采样点或第 8、9、10 位的采样点），那么起始位仍然是有效的，但是会设置 NE 噪声标志位。

26.3.3.2 字符接收

在 USART 接收期间，数据的最低有效位首先从 RX 脚移进。在此模式里，USARTx_DR 寄存器包含的缓冲器位于内部总线和接收移位寄存器之间。

配置步骤：

1. 将 USARTx_CR1 寄存器的 UE 置 1 来激活 USART。
2. 编程 USARTx_CR1 的 M 位定义字长。
3. 在 USARTx_CR2 中编写停止位的个数。
4. 如果需多缓冲器通信，选择 USARTx_CR3 中的 DMA 使能位（DMAR）。按多缓冲器通信所要求的配置 DMA 寄存器。
5. 利用波特率寄存器 USARTx_BRR 选择希望的波特率。
6. 设置 USARTx_CR1 的 RE 位。激活接收器，使它开始寻找起始位。

当一个字符被接收到时：

- RXNE 位被置位。它表明移位寄存器的内容被转移到 RDR。换句话说，数据已经被接收并且可以被读出（包括与之有关的错误标志）。

- 如果 RXNEIE 位被设置，产生中断。
- 在接收期间如果检测到帧错误，噪音或溢出错误，错误标志将被置起，
- 在多缓冲器通信时，RXNE 在每个字节接收后被置起，并由 DMA 对数据寄存器的读操作而清零。
- 在单缓冲器模式里，由软件读 USARTx_DR 寄存器完成对 RXNE 位清除。RXNE 标志也可以通过对它写 0 来清除。RXNE 位必须在下一字符接收结束前被清零，以避免溢出错误。

注意：在接收数据时，RE 位不应该被复位。如果 RE 位在接收时被清零，当前字节的接收被丢失。

26.3.3.3 断开符号

当接收到一个断开帧时，USART 像处理帧错误一样处理它。

26.3.3.4 空闲符号

当一空闲帧被检测到时，其处理步骤和接收到普通数据帧一样，但如果 IDLEIE 位被设置将产生一个中断。

26.3.3.5 溢出错误

如果 RXNE 还没有被复位，又接收到一个字符，则发生溢出错误。只有当 RXNE 位被清零后，数据才能从移位寄存器转移到 RDR 寄存器。RXNE 标记是接收到每个字节后被置位的。如果下一个数据已被收到或先前 DMA 请求还没被服务时，RXNE 标志仍是置起的，溢出错误产生。

当溢出错误产生时：

- ORE 位被置位。
- RDR 内容将不会丢失。读 USARTx_DR 寄存器仍能得到先前的数据。
- 移位寄存器中以前的内容将被覆盖。随后接收到的数据都将丢失。
- 如果 RXNEIE 位被设置或 EIE 和 DMAR 位都被设置，中断产生。
- 顺序执行对 USARTx_SR 和 USARTx_DR 寄存器的读操作，可复位 ORE 位。

注意：当 ORE 位置位时，表明至少有 1 个数据已经丢失。

有两种可能性：

如果 RXNE=1，上一个有效数据还在接收寄存器 RDR 上，可以被读出。

如果 RXNE=0，这意味着上一个有效数据已经被读走，RDR 已经没有东西可读。当上一个有效数据在 RDR 中被读取的同时又接收到新的（也就是丢失的）数据时，此种情况可能发生。在读序列期间（在 USARTx_SR 寄存器读访问和 USARTx_DR 读访问之间）接收到新的数据，此种情况也可能发生。

26.3.3.6 噪音错误

使用过采样技术（同步模式除外），通过区别有效输入数据和噪音来进行数据恢复。

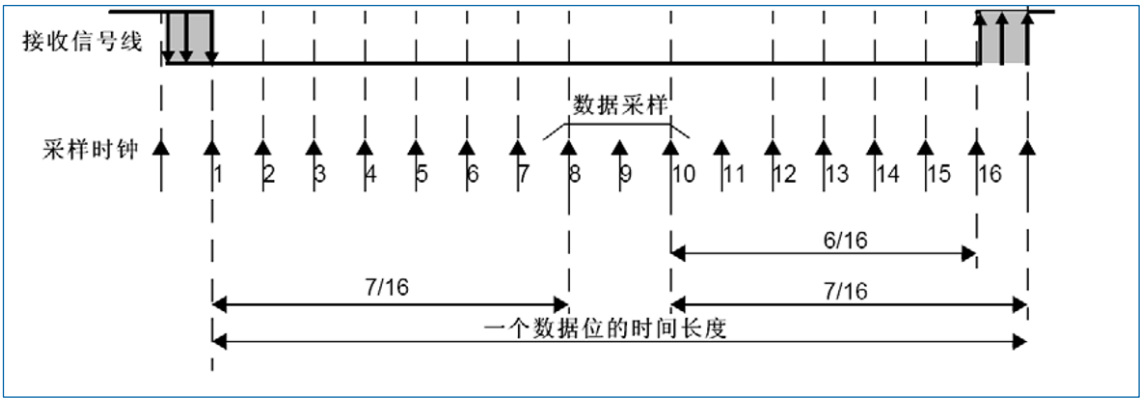


图 26-6 检测噪声的数据采样

表 26-1 检测噪声的数据采样

采样值	NE 状态	接收的位	数据有效性
000	0	0	有效
001	1	0	无效
010	1	0	无效
011	1	1	无效
100	1	0	无效
101	1	1	无效
110	1	1	无效
111	0	1	有效

当在接收帧中检测到噪音时：

- 在 RXNE 位的上升沿设置 NE 标志。
- 无效数据从移位寄存器传送到 USARTx_DR 寄存器。
- 在单个字节通信情况下，没有中断产生。然而，如果 NE 标志位和 RXNE 标志位是同时被设置，RXNE 将产生中断。在多缓冲器通信情况下，如果已经设置了 USARTx_CR3 寄存器中 EIE 位，将产生一个中断。

先读出 USARTx_SR，再读出 USARTx_DR 寄存器，将清除 NE 标志位。

26.3.3.7 帧错误

当以下情况发生时检测到帧错误：伴随同步失败或大量噪音，停止位没有在预期的时间被接收识别。

当帧错误被检测到时：

- FE 位被硬件置起。
- 无效数据从移位寄存器传送到 USARTx_DR 寄存器。
- 在单字节通信时，没有中断产生。然而，这个位和 RXNE 位同时置起时，后者将产生中断。在多缓冲器通信情况下，如果 USARTx_CR3 寄存器中 EIE 位被置位的话，将产生中断。

顺序执行对 USARTx_SR 和 USARTx_DR 寄存器的读操作，可复位 FE 位。

26.3.3.8 接收期间的可配置的停止位

被接收的停止位的个数可以通过控制寄存器 2 的控制位来配置，在正常模式时，可以是 1 或 2 个，

在智能卡模式里可能是 0.5 或 1.5 个。

1. 0.5 个停止位（智能卡模式中的接收）：不对 0.5 个停止位进行采样。因此，如果选择 0.5 个停止位则不能检测帧错误和断开帧。

2. 1 个停止位：对 1 个停止位的采样在第 8，第 9 和第 10 采样点上进行。

3. 1.5 个停止位（智能卡模式）：当以智能卡模式发送时，器件必须检查数据是否被正确的发送出去。所以接收器功能块必须被激活（USARTx_CR1 寄存器中的 RE=1），并且在停止位的发送期间采样数据线上的信号。如果出现校验错误，智能卡会在发送方采样 NACK 信号时，即总线上停止位对应的时间内时，拉低数据线，以此表示出现了帧错误。FE 在 1.5 个停止位结束时和 RXNE 一起被置起。对 1.5 个停止位的采样是在第 16，第 17 和第 18 采样点进行的。1.5 个的停止位可以被分成 2 部分：一个是 0.5 个时钟周期，期间不做任何事情。随后是 1 个时钟周期的停止位，在这段时间的中点处采样。详见章节：“26.3.11 智能卡”。

4. 2 个停止位：对 2 个停止位的采样是在第一停止位的第 8，第 9 和第 10 个采样点完成的。如果第一个停止位期间检测到一个帧错误，帧错误标志将被设置。第二个停止位不再检查帧错误。在第一个停止位结束时 RXNE 标志将被设置。

26.3.4 分数波特率的产生

接收器和发送器的波特率在 USARTDIV 的整数和小数寄存器中的值应设置成相同。

$$\text{Tx（或 Rx）波特率} = \frac{f_{CK}}{\text{（16*USARTDIV）}}$$

这里的 f_{CK} 是给外设的时钟（PCLK1 用于 USART2/3/4/5，PCLK2 用于 USART1/6）。

USARTDIV 是一个无符号的定点数，这 12 位的值设置在 USARTx_BRR 寄存器。

注意：在写入 USARTx_BRR 之后，波特率计数器会被波特率寄存器的新值替换。因此，不要在通信进行中改变波特率寄存器的数值。

如何从 USARTx_BRR 寄存器值得到 USARTDIV

例 1：

如果DIV_Mantissa=27，DIV_Fraction=12（USARTx_BRR=0x1BC），

于是：

Mantissa（USARTDIV）=27

Fraction（USARTDIV）=12/16=0.75

所以USARTDIV=27.75

例 2：

要求USARTDIV=25.62,

就有：

DIV_Fraction=16*0.62=9.92

最接近的整数是：10=0x0A

DIV_Mantissa=mantissa（25.620）=25=0x19

于是，USARTx_BRR=0x19A

例 3：

要求USARTDIV=50.99

就有：

DIV_Fraction=16*0.99=15.84

最接近的整数是：16=0x10 => DIV_frac[3:0]溢出=> 进位必须加到小数部分

DIV_Mantissa=mantissa（50.990 +进位）=51=0x33

于是：USARTx_BRR=0x330，USARTDIV=51

表 26-2 设置波特率时的误差计算

波特率		f_{PCLK}=36MHz			f_{PCLK}=72MHz		
序号	Kbps	实际	置于波特率寄存器中的值	误差%	实际	置于波特率 寄存器中的值	误差%
1	2.4	2.400	937.5	0%	2.4	1875	0%
2	9.6	9.600	234.375	0%	9.6	468.75	0%
3	19.2	19.2	117.1875	0%	19.2	234.375	0%
4	57.6	57.6	39.0625	0%	57.6	78.125	0%
5	115.2	115.384	19.5	0.15%	115.2	39.0625	0%
6	230.4	230.769	9.75	0.16%	230.769	19.5	0.16%
7	460.8	461.538	4.875	0.16%	461.538	9.75	0.16%
8	921.6	923.076	2.4375	0.16%	923.076	4.875	0.16%
9	2250	2250	1	0%	2250	2	0%
10	4500	不可能	不可能	不可能	4500	1	0%

注意：

1.CPU 的时钟频率越低，则某一特定波特率的误差也越低。可以达到的波特率上限可以由这组数据得到。

2.只有 USART1/6 使用 PCLK2。其它 USART 使用 PCLK1。

26.3.5 USART 接收器容忍时钟的变化

只有当整体的时钟系统地变化小于 USART 异步接收器能够容忍的范围，USART 异步接收器才能正常地工作。影响这些变化的因素有：

- DTRA：由于发送器误差而产生的变化（包括发送器端振荡器的变化）。
- DQUANT：接收器端波特率取整所产生的误差。
- DREC：接收器端振荡器的变化。
- DTCL：由于传输线路产生的变化（通常是由于收发器在由低变高的转换时序，与由高变低转换时序之间的不一致性所造成）。

需要满足：DTRA +DQUANT +DREC +DTCL < USART 接收器的容忍度

对于正常接收数据，USART 接收器的容忍度等于最大能容忍的变化，它依赖于下述选择：

- 由 USARTx_CR1 寄存器的 M 位定义的 10 或 11 位字符长度
- 是否使用分数波特率产生

表 26-3 当 DIV_Fraction=0 时，USART 接收器的容忍度

M 位	认为 NF 是错误	不认为 NF 是错误
0	3.75%	4.375%
1	3.41%	3.97%

表 26-4 当 DIV_Fraction!=0 时，USART 接收器的容忍度

M 位	认为 NF 是错误	不认为 NF 是错误
0	3.33%	3.88%
1	3.03%	3.53%

注意：在特殊的情况下，当收到的帧包含一些在 M=0 时，正好是 10 位（M=1 时是 11 位）的空闲帧，上面 2 个表格中的数据可能会有些微不同。

26.3.6 多处理器通信

通过 USART 可以实现多处理器通信（将几个 USART 连在一个网络里）。例如某个 USART 设备可以是主设备，它的 TX 输出和其他 USART 从设备的 RX 输入相连接；其他 USART 从设备各自的 TX 输出逻辑地连接在一起，并且和主设备的 RX 输入相连接。

在多处理器配置中，我们通常希望只有被寻址的接收者才被激活，来接收随后的数据。这样就可以减少未被寻址的接收器的参与，造成多余的 USART 服务开销。未被寻址的设备可启用其静默功能置于静默模式。在静默模式里：

- 任何接收状态位都不会被设置。
- 所有接收中断被禁止。
- USARTx_CR1 寄存器中的 RWU 位被置 1。RWU 可以被硬件自动控制或在某个条件下由软件写入。

根据 USARTx_CR1 寄存器中的 WAKE 位状态，USART 可以用二种方法进入或退出静默模式。

- 如果 WAKE 位被复位：进行空闲总线检测。
- 如果 WAKE 位被设置：进行地址标记检测。

26.3.6.1 空闲总线检测（WAKE=0）

当 RWU 位被写 1 时，USART 进入静默模式。当检测到一空闲帧时，它被唤醒。随后 RWU 位被硬件清零，但是 USARTx_SR 寄存器中的 IDLE 位并不置位。RWU 还可以被软件写 0。下图给出利用空闲总线检测来唤醒和进入静默模式的一个例子：

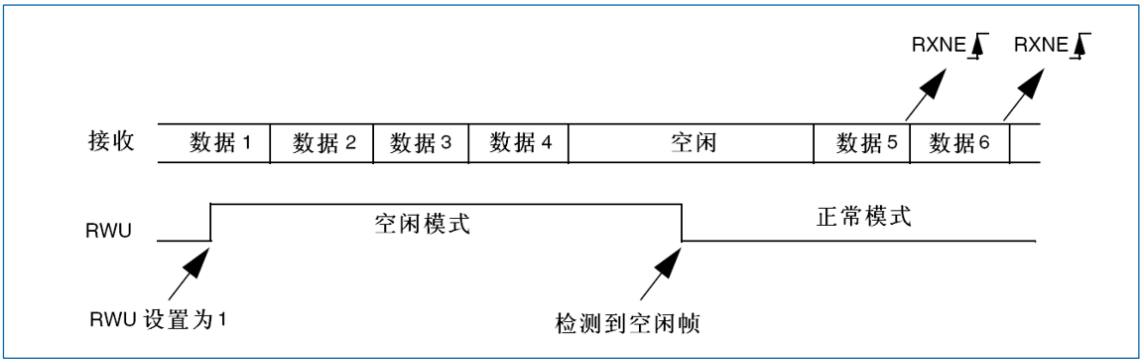


图 26-7 利用空闲总线检测的静默模式

26.3.6.2 地址标记检测（WAKE=1）

在这个模式里，如果 MSB 是 1，该字节被认为是地址，否则被认为是数据。在一个地址字节中，目标接收器的地址被放在 4 个 LSB 中。这个 4 位地址被接收器同它自己地址做比较，接收器的地址被编程在 USARTx_CR2 寄存器的 ADD。

如果接收到的字节与它的编程地址不匹配时，USART 进入静默模式。此时，硬件设置 RWU 位。接收该字节既不会设置 RXNE 标志也不会产生中断或发出 DMA 请求，因为 USART 已经在静默模式。

当接收到的字节与接收器内编程地址匹配时，USART 退出静默模式。然后 RWU 位被清零，随后的字节被正常接收。收到这个匹配的地址字节时将设置 RXNE 位，因为 RWU 位已被清零。

当接收缓冲器不包含数据时（USARTx_SR 的 RXNE=0），RWU 位可以被写 0 或 1。否则，该次写操作被忽略。下图给出利用地址标记检测来唤醒和进入静默模式的例子。

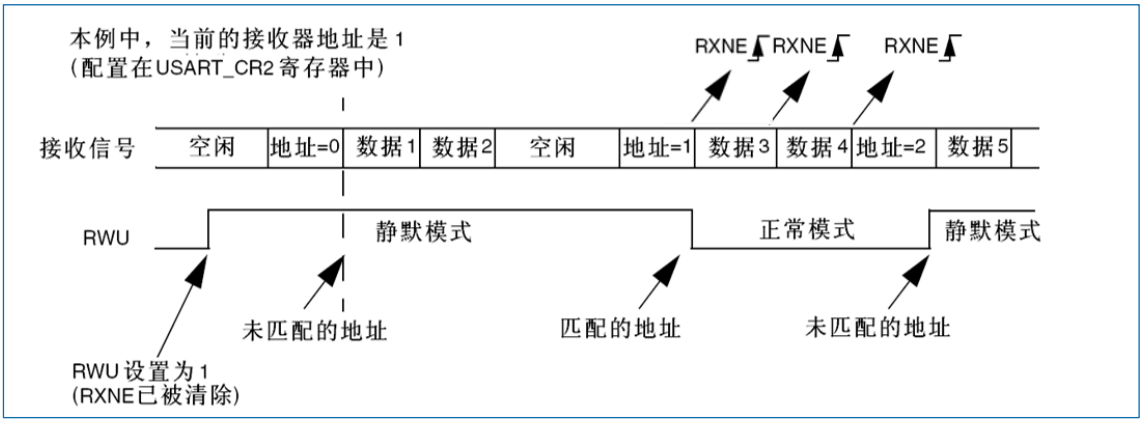


图 26-8 利用地址标记检测的静默模式

26.3.7 校验控制

设置 USARTx_CR1 寄存器上的 PCE 位，可以使能奇偶控制（发送时生成一个奇偶位，接收时进行奇偶校验）。根据 M 位定义的帧长度，可能的 USART 帧格式列在下表中。

表 26-5 帧格式

M 位	PCE 位	USART 帧
0	0	|起始位| 8 位数据| 停止位|
0	1	| 起始位| 7 位数据| 奇偶检验位| 停止位|
1	0	| 起始位| 9 位数据| 停止位|
1	1	| 起始位| 8 位数据| 奇偶检验位| 停止位|

注意：在用地址标记唤醒设备时，地址的匹配只考虑到数据的 MSB 位，而不用关心校验位。（MSB 是数据位中最后发出的，后面紧跟校验位或者停止位）

偶校验：校验位使得一帧中的 7 或 8 个 LSB 数据以及校验位中'1'的个数为偶数。

例如：数据=00110101，有 4 个'1'，如果选择偶校验（在 USARTx_CR1 中的 PS＝0），校验位将是'0'。

奇校验：此校验位使得一帧中的 7 或 8 个 LSB 数据以及校验位中'1'的个数为奇数。

例如：数据=00110101，有 4 个'1'，如果选择奇校验（在 USARTx_CR1 中的 PS＝1），校验位将是'1'。

传输模式：如果 USARTx_CR1 的 PCE 位被置位，写进数据寄存器的数据的 MSB 位被校验位替换后发送出去（如果选择偶校验偶数个'1'，如果选择奇校验奇数个'1'）。如果奇偶校验失败，

USARTx_SR 寄存器中的 PE 标志被置’1’，并且如果 USARTx_CR1 寄存器的 PEIE 在被预先设置的话，中断产生。

26.3.8 LIN（局域互联网）模式

LIN 模式是通过设置 USARTx_CR2 寄存器的 LINEN 位选择。在 LIN 模式下，下列位必须保持为 0：

- USARTx_CR2 寄存器的 CLKEN 位。
- USARTx_CR2 寄存器的 STOP[1:0]，USARTx_CR3 寄存器的 SCEN，HDSEL 和 IREN。

26.3.8.1 LIN 发送

章节 26.3.2 发送器所描述的同样步骤适用于 LIN 主发送，但和正常 USART 发送有以下区别：

- 清零 M 位以配置 8 位字长
- 置位 LINEN 位以进入 LIN 模式。这时，置位 SBK 将发送 13 位’0’作为断开符号。然后发一位’1’，以允许对下一个开始位的检测。

26.3.8.2 LIN 接收

当 LIN 模式被使能时，断开符号检测电路被激活。该检测完全独立于 USART 接收器。断开符号只要一出现就能检测到，不管是在总线空闲时还是在发送某数据帧其间。

当接收器被激活时（USARTx_CR1 的 RE=1），电路监测 RX 上的起始信号。监测起始位的方法同检测断开符号或数据是一样的。当起始位被检测到后，电路对每个接下来的位，在每个位的第 8，9，10 个过采样时钟点上进行采样。如果 10 个（当 USARTx_CR2 的 LBDL=0）或 11 个（当 USARTx_CR2 的 LBDL=1）连续位都是’0’，并且又跟着一个定界符，USARTx_SR 的 LBD 标志被设置。如果 LBDIE 位=1，中断产生。在确认断开符号前，要检查定界符，因为它意味 RX 线已经回到高电平。

如果在第 10 或 11 个采样点之前采样到了’1’，检测电路取消当前检测并重新寻找起始位。如果 LIN 模式被禁止，接收器继续如正常 USART 那样工作，不需要考虑检测断开符号。

如果 LIN 模式没有被激活（LINEN=0），接收器仍然正常工作于 USART 模式，不会进行断开检测。

如果 LIN 模式被激活（LINEN=1），只要一发生帧错误（也就是停止位检测到’0’，这种情况出现在断开帧），接收器就停止，直到断开符号检测电路接收到一个’1’（这种情况发生于断开符号没有完整的发出来），或一个定界符（这种情况发生于已经检测到一个完整的断开符号）。

图 26-9 说明了断开符号检测器状态机的行为和断开符号标志的关系。图 26-10 给出了一个断开帧的例子。

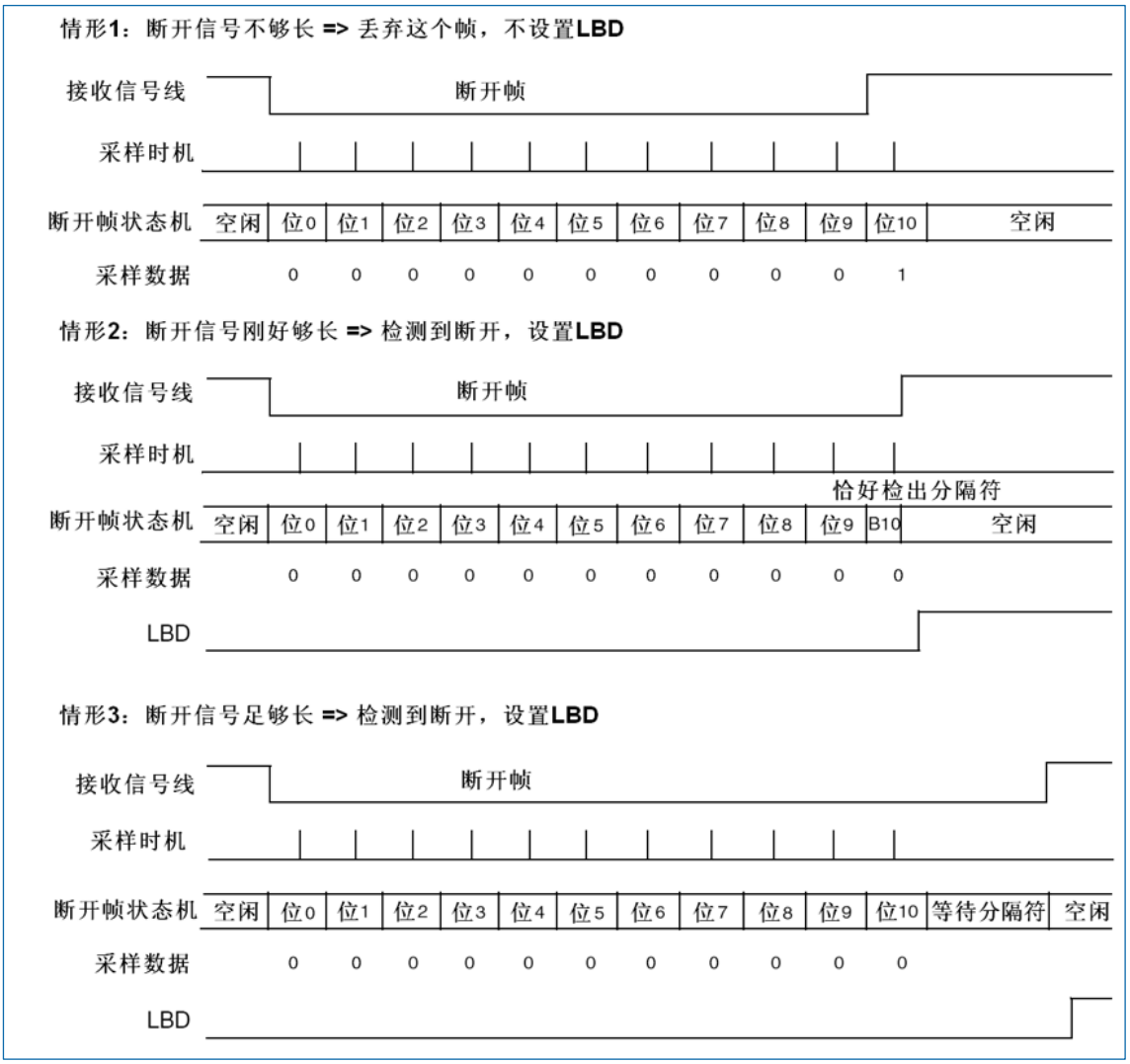


图 26-9 LIN 模式下的断开检测（11 位断开长度 – 设置了 LBDL 位）

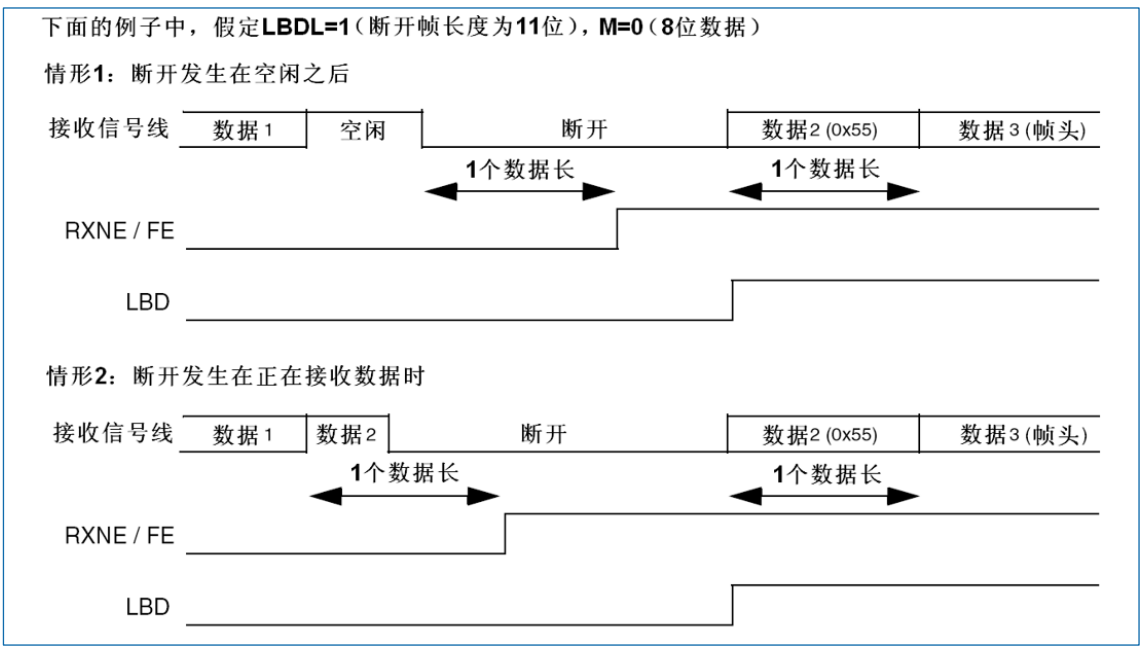


图 26-10 LIN 模式下的断开检测与帧错误的检测

26.3.9 USART 同步模式

通过在 USARTx_CR2 寄存器上写 CLKEN 位为’1’选择同步模式。

在同步模式里，下列位必须保持清零状态：

- USARTx_CR2 寄存器中的 LINEN 位。
- USARTx_CR3 寄存器中的 SCEN,HDSEL 和 IREN 位。

USART 允许用户以主模式方式控制双向同步串行通信。CK 脚是 USART 发送器时钟的输出。在起始位和停止位期间，CK 脚上没有时钟脉冲。根据 USARTx_CR2 寄存器中 LBCL 位的状态，决定 在最后一个有效数据位期间产生或不产生时钟脉冲。USARTx_CR2 寄存器的 CPOL 位允许用户选择时钟极性，USARTx_CR2 寄存器上的 CPHA 位允许用户选择外部时钟的相位（见图 26-11、图 26-12 和图 26-13）。

在总线空闲期间，实际数据到来之前以及发送断开符号的时候，外部 CK 时钟不被激活。

同步模式时，USART 发送器和异步模式里工作一模一样。但是因为 CK 是与 TX 同步的（根据 CPOL 和 CPHA），所以 TX 上的数据是随 CK 同步发出的。

同步模式的 USART 接收器工作方式与异步模式不同。如果 RE=1，数据在 CK 上采样（根据 CPOL 和 CPHA 决定在上升沿还是下降沿），不需要任何的过采样。但必须考虑建立时间和持续时间（取决于波特率，1/16 位时间）。

注意：

1. CK 脚同 TX 脚一起联合工作。因而，只有在使能了发送器（TE＝1），并且发送数据时（写入数据至 USARTx_DR 寄存器）才提供时钟。这意味着在没有发送数据时是不可能接收一个同步数据 的。
2. 当使能了发送器或接收器时，LBCL，CPOL 和 CPHA 位不能被改变。
3. 建议在同一条指令中设置 TE 和 RE，以减少接收器的建立时间和保持时间。
4. USART 只支持主模式：它不能用来自其他设备的输入时钟接收或发送数据（CK 永远是输出）。

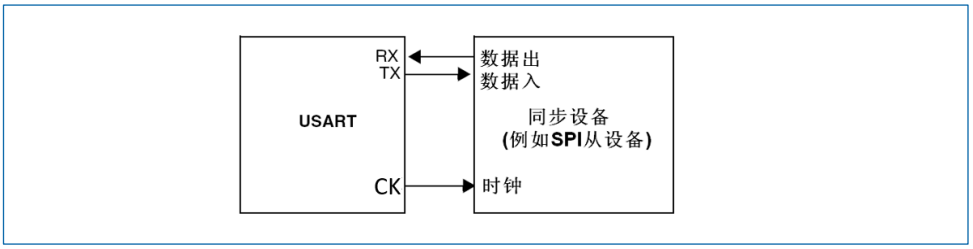


图 26-11 USART 同步传输的例子

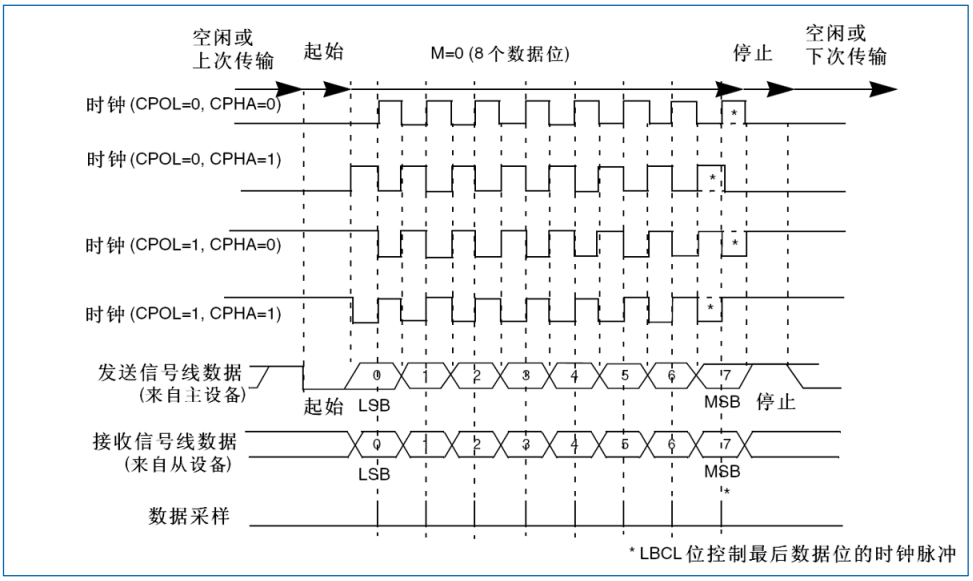


图 26-12 USART 数据时钟时序示例（M=0）

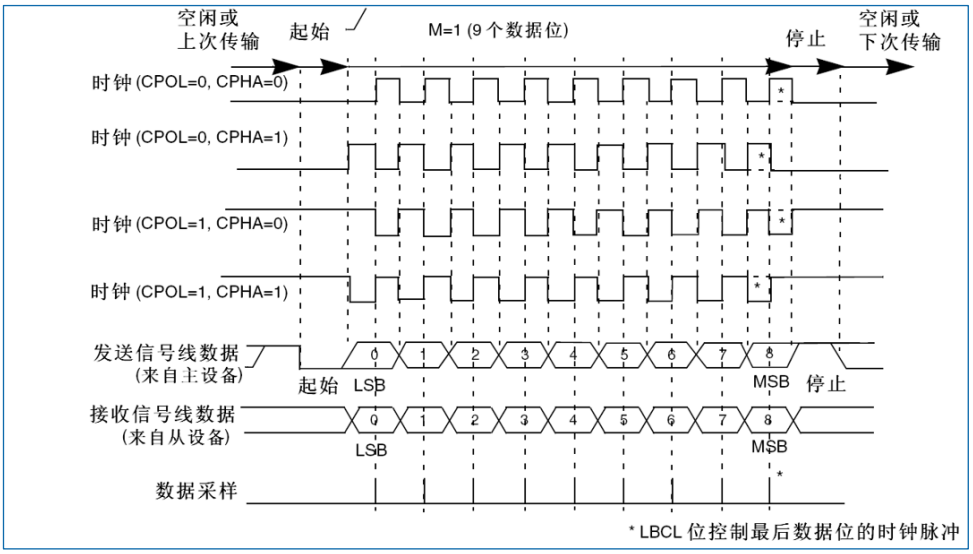


图 26-13 USART 数据时钟时序示例（M=1）

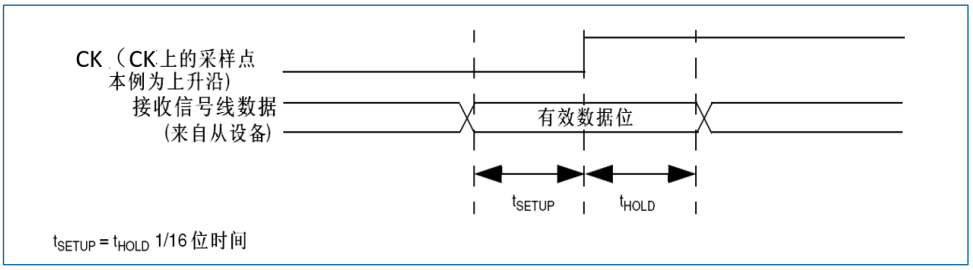


图 26-14 RX 数据采样/保持时间

说明：在智能卡模式下 CK 的功能不同，有关细节请参考智能卡模式部分。

26.3.10 单线半双工通信

单线半双方模式通过设置 USARTx_CR3 寄存器的 HDSEL 位选择。在这个模式里，下面的位必须保持清零状态：

- USARTx_CR2 寄存器的 LINEN 和 CLKEN 位
- USARTx_CR3 寄存器的 SCEN 和 IREN 位

USART 可以配置成遵循单线半双工协议。在单线半双工模式下，TX 和 RX 引脚在芯片内部互连。使用控制位“HALF DUPLEX SEL”（USARTx_CR3 中的 HDSEL 位）选择半双工和全双工通信。

当 HDSEL 为’1’时：

- RX 不再被使用。
- 当没有数据传输时，TX 总是被释放。因此，它在空闲状态的或接收状态时表现为一个标准 I/O 口。这就意味该 I/O 在不被 USART 驱动时，必须配置成悬空输入（或开漏的输出高）。

除此以外，通信与正常 USART 模式类似。由软件来管理线上的冲突（例如通过使用一个中央仲裁器）。特别的是，发送从不会被硬件所阻碍。当 TE 位被设置时，只要数据一写到数据寄存器上，发送就继续。

26.3.11 智能卡

设置 USARTx_CR3 寄存器的 SCEN 位选择智能卡模式。在智能卡模式下，下列位必须保持清零：

- USARTx_CR2 寄存器的 LINEN 位。
- USARTx_CR3 寄存器的 HDSEL 位和 IREN 位。

此外，CLKEN 位可以被设置，以提供时钟给智能卡。

该接口符合 ISO7816-3 标准，支持智能卡异步协议。USART 应该被设置为：

- 8 位数据位加校验位：此时 USARTx_CR1 寄存器中 M=1、PCE=1。
- 发送和接收时为 1.5 个停止位：即 USARTx_CR2 寄存器的 STOP=11。

注意：也可以在接收时选择 0.5 个停止位，但为了避免在 2 种配置间转换，建议在发送和接收时使用 1.5 个停止位。

下图给出的例子说明了数据线上，在有校验错误和没校验错误两种情况下的信号。

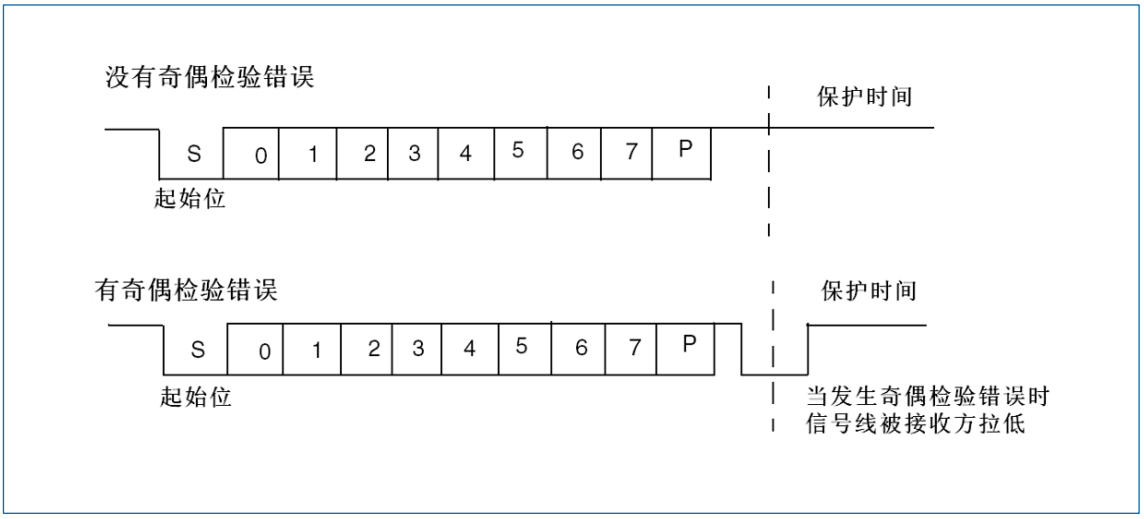


图 26-15 ISO7816-3 异步协议

当与智能卡相连接时，USART 的 TX 输出驱动一根智能卡也驱动的双向线。所以 TX 必须配置成开漏。

智能卡是一个单线半双工通信协议。

- 从发送移位寄存器把数据发送出去，要被延时最小 1/2 波特时钟。在正常操作时，一个满的发送移位寄存器将在下一个波特时钟沿开始向外移出数据。在智能卡模式里，此发送被延迟 1/2 波特时钟。
- 如果在接收一个设置为 0.5 或 1.5 个停止位的数据帧期间，检测到一奇偶校验错误，在完成接收该帧后（即停止位结束时），发送线被拉低一个波特时钟周期。这是告诉智能卡发送到 USART 的数据没有被正确地接收到。此 NACK 信号（拉低发送线一个波特时钟周期）在发送端将产生一个帧错误（发送端被配置成 1.5 个停止位）。应用程序可以根据协议处理重新发送数据。如果设置了 NACK 控制位，发生校验错误时接收器会给出一个 NACK 信号；否则就不会发送 NACK 信号。
- TC 标志的置起可以通过编程保护时间寄存器得以延时。在正常操作时，当发送移位寄存器 变空并且没有新的发送请求出现时，TC 被置起。在智能卡模式里，空的发送移位寄存器将 触发保护时间计数器开始向上计数，直到保护时间寄存器中的值。TC 在这段时间被强制拉 低。当保护时间计数器达到保护时间寄存器中的值时，TC 被置高。
- TC 标志的撤销不受智能卡模式的影响。
- 如果发送器检测到一个帧错误（收到接收器的 NACK 信号），发送器的接收功能模块不会把 NACK 当作起始位检测。根据 ISO 协议，接收到的 NACK 的持续时间可以是 1 或 2 波特时钟周期。
- 在接收器这边，如果一个校验错误被检测到，并且 NACK 被发送，接收器不会把 NACK 检测成起始位。

注意：

1.断开符号在智能卡模式里没有意义。一个带帧错误的 00h 数据将被当成数据而不是断开符号。

2.当来回切换 TE 位时，没有 IDLE 帧被发送。ISO 协议没有定义 IDLE 帧。

下图详述了 USART 是如何采样 NACK 信号的。在这个例子里，USART 正在发送数据，并且被配置成 1.5 个停止位。为了检查数据的完整性和 NACK 信号，USART 的接收功能块被激活。

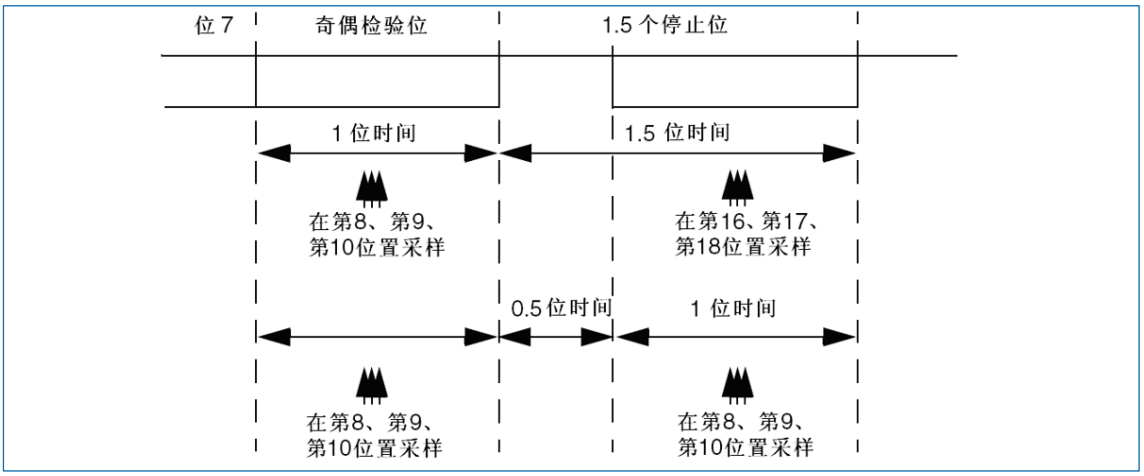


图 26-16 使用 1.5 停止位检测奇偶检验错

USART 可以通过 CK 输出为智能卡提供时钟。在智能卡模式里，CK 不和通信直接关联，而是先通过一个 5 位预分频器简单地用内部的外设输入时钟来驱动智能卡的时钟。分频率在预分频寄存器 USARTx_GTPR 中配置。CK 频率可以从 $f_{CK}/2$ 到 $f_{CK}/62$，这里的 f_{CK} 是外设输入时钟。

26.3.12 IrDA SIR ENDEC 功能模块

通过设置 USARTx_CR3 寄存器的 IREN 位选择 IrDA 模式。在 IRDA 模式里，下列位必须保持清零：

- USARTx_CR2 寄存器的 LINEN，STOP 和 CLKEN 位。
- USARTx_CR3 寄存器的 SCEN 和 HDSEL 位。

IrDA SIR 物理层规定使用反相归零调制方案（RZI），该方案用一个红外光脉冲代表逻辑'0'（图 26-17）。SIR 发送编码器对从 USART 输出的 NRZ（非归零）比特流进行调制。输出脉冲流被传送到一个外部输出驱动器和红外 LED。USART 为 SIR ENDEC 最高只支持到 115.2Kbit/s 速率。在正常模式里，脉冲宽度规定为一个位周期的 3/16。

SIR 接收解码器对来自红外接收器的归零位比特流进行解调，并将接收到的 NRZ 串行比特流输出到 USART。在空闲状态里，解码器输入通常是高（标记状态）。发送编码器输出的极性和解码器的输入相反。当解码器输入低时，检测到一个起始位。

- IrDA 是一个半双工通信协议。如果发送器忙（也就是 USART 正在送数据给 IrDA 编码器），IrDA 接收线上的任何数据将被 IrDA 解码器忽视。如果接收器忙（也就是 USART 正在接收从 IrDA 解码器来的解码数据），从 USART 到 IrDA 的 TX 上的数据将不会被 IrDA 编码。当接收数据时，应该避免发送，因为将被发送的数据可能被破坏。
- SIR 发送逻辑把'0'作为高脉冲发送，把'1'作为低电平发送。脉冲的宽度规定为正常模式时位周期的 3/16（详见图 26-18）。
- SIR 接收逻辑把高电平状态解释为'1'，把低脉冲解释为'0'。
- 发送编码器输出与解码器输入有着相反的极性。当空闲时，SIR 输出处于低状态。
- SIR 解码器把 IrDA 兼容的接收信号转变成给 USART 的比特流。
- IrDA 规范要求脉冲要宽于 1.41μs。脉冲宽度是可编程的。接收器端的尖峰脉冲检测逻辑滤除宽度小于 2 个 PSC 周期的脉冲（PSC 是在 IrDA 低功耗波特率寄存器 USARTx_GTPR 中编程的预分频值）。宽度小于 1 个 PSC 周期的脉冲一定会被滤除掉，但是那些宽度大于 1 个而小于 2 个 PSC 周期的脉冲可能被接收或滤除，那些宽度大于 2 个周期的将被视为一个有效的脉冲。当 PSC=0 时，IrDA 编码器/解码器不工作。
- 接收器可以与低功耗发送器通信。

- 在 IrDA 模式里，USARTx_CR2 寄存器上的 STOP 位必须配置成 1 个停止位。

26.3.12.1 IrDA 低功耗模式

发送器

在低功耗模式，脉冲宽度不再持续 3/16 个位周期。取而代之，脉冲的宽度是低功耗波特率的 3 倍，它最小可以是 1.42 MHz。通常这个值是 1.8432MHz（1.42 MHz < PSC < 2.12 MHz）。一个低功耗模式可编程分频器把系统时钟进行分频以达到这个值。

接收器

低功耗模式的接收类似于正常模式的接收。为了滤除尖峰干扰脉冲，USART 应该滤除宽度短于 1 个 PSC 的脉冲。只有持续时间大于 2 个周期的 IrDA 低功耗波特率时钟（USARTx_GTPR 中的 PSC）的低电平信号才被接受为有效的信号。

注意：

宽度小于 2 个大于 1 个 PSC 周期的脉冲可能会也可能不会被滤除。

接收器的建立时间应该由软件管理。IrDA 物理层技术规范规定了在发送和接收之间最小要有 10ms 的延时（IrDA 是一个半双工协议）。

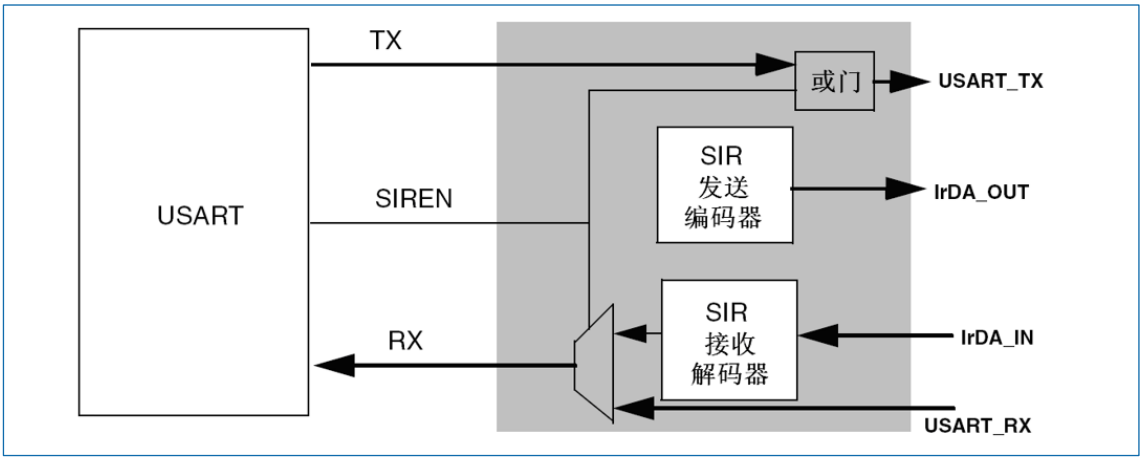


图 26-17 IrDA SIR ENDEC 框图

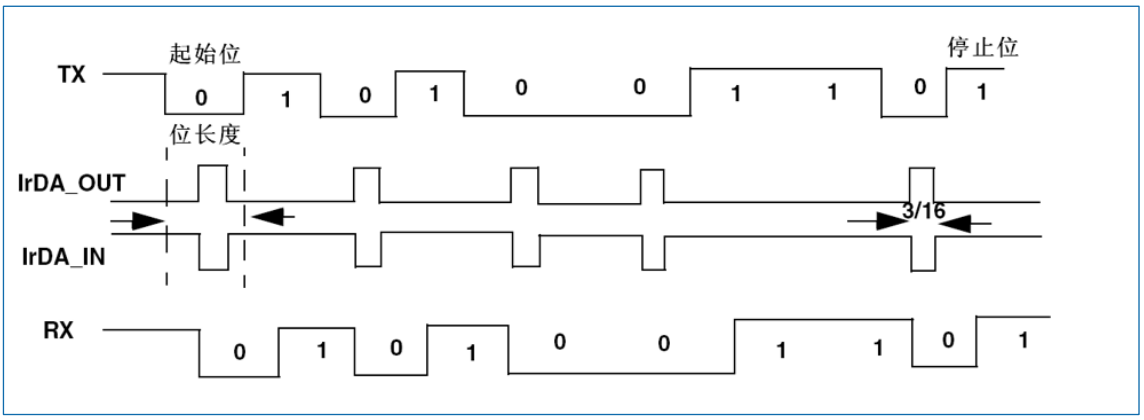


图 26-18 IrDA 数据调制（3/16）普通模式

26.3.13 利用 DMA 连续通信

USART 可以利用 DMA 连续通信。Rx 缓冲器和 Tx 缓冲器的 DMA 请求是分别产生的。

注意：参考“11.2.7 DMA 请求映射”以确定 USARTx 是否可用 DMA 控制器。如果不可用，应按章节 26.3.2 或章节 26.3.3 里所描述的方法使用 USART。在 USART2_SR 寄存器里，可以清零 TXE/RXNE 标志来

实现连续通信。

26.3.13.1 利用 DMA 发送

使用 DMA 进行发送，可以通过设置 USARTx_CR3 寄存器上的 DMAT 位激活。当 TXE 位被置为'1'时，DMA 就从指定的 SRAM 区传送数据到 USARTx_DR 寄存器。为 USART 的发送分配一个 DMA 通道的步骤如下（x 表示通道号）：

1. 在 DMA 控制寄存器上将 USARTx_DR 寄存器地址配置成 DMA 传输的目的地址。在每个 TXE 事件后，数据将被传送到这个地址。
2. 在 DMA 控制寄存器上将存储器地址配置成 DMA 传输的源地址。在每个 TXE 事件后，将从此存储器区读出数据并传送到 USARTx_DR 寄存器。
3. 在 DMA 控制寄存器中配置要传输的总的字节数。
4. 在 DMA 寄存器上配置通道优先级。
5. 根据应用程序的要求，配置在传输完成一半还是全部完成时产生 DMA 中断。
6. 在 DMA 寄存器上激活该通道。当传输完成 DMA 控制器指定的数据量时，DMA 控制器在该 DMA 通道的中断向量上产生一中断。

在发送模式下，当 DMA 传输完所有要发送的数据时，DMA 控制器设置 DMA_ISR 寄存器的 TCIF 标志；监视 USARTx_SR 寄存器的 TC 标志可以确认 USART 通信是否结束，这样可以在关闭 USART 或进入停机模式之前避免破坏最后一次传输的数据；软件需要先等待 TXE=1，再等待 TC=1。

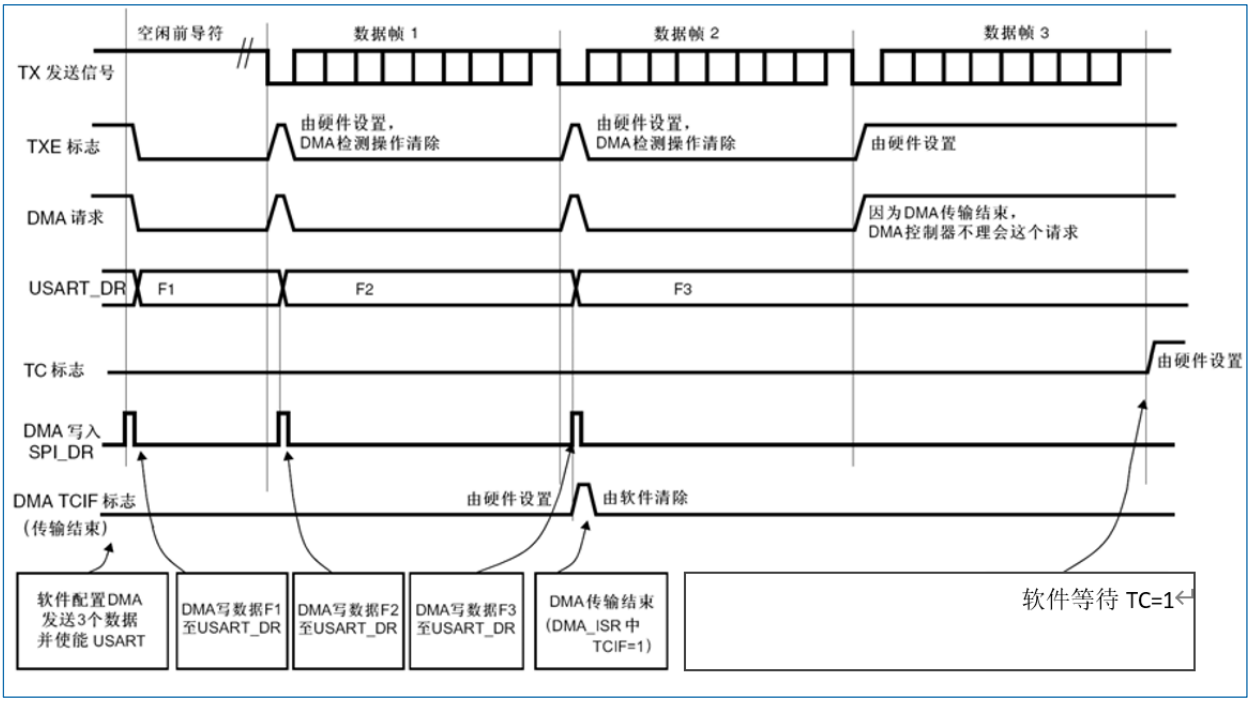


图 26-19 利用 DMA 发送

26.3.13.2 利用 DMA 接收

可以通过设置 USARTx_CR3 寄存器的 DMAR 位激活使用 DMA 进行接收，每次接收到一个字节，DMA 控制器就把数据从 USARTx_DR 寄存器传送到指定的 SRAM 区（参考 DMA 相关说明）。为 USART 的接收分配一个 DMA 通道的步骤如下（x 表示通道号）：

1. 通过 DMA 控制寄存器把 USARTx_DR 寄存器地址配置成传输的源地址。在每个 RXNE 事件 后，将从此地址读出数据并传输到存储器。

2. 通过 DMA 控制寄存器把存储器地址配置成传输的目的地址。在每个 RXNE 事件后，数据将从 USARTx_DR 传输到此存储器区。
3. 在 DMA 控制寄存器中配置要传输的总的字节数。
4. 在 DMA 寄存器上配置通道优先级。
5. 根据应用程序的要求配置在传输完成一半还是全部完成时产生 DMA 中断。
6. 在 DMA 控制寄存器上激活该通道。

当接收完成 DMA 控制器指定的传输量时，DMA 控制器在该 DMA 通道的中断矢量上产生一中断。

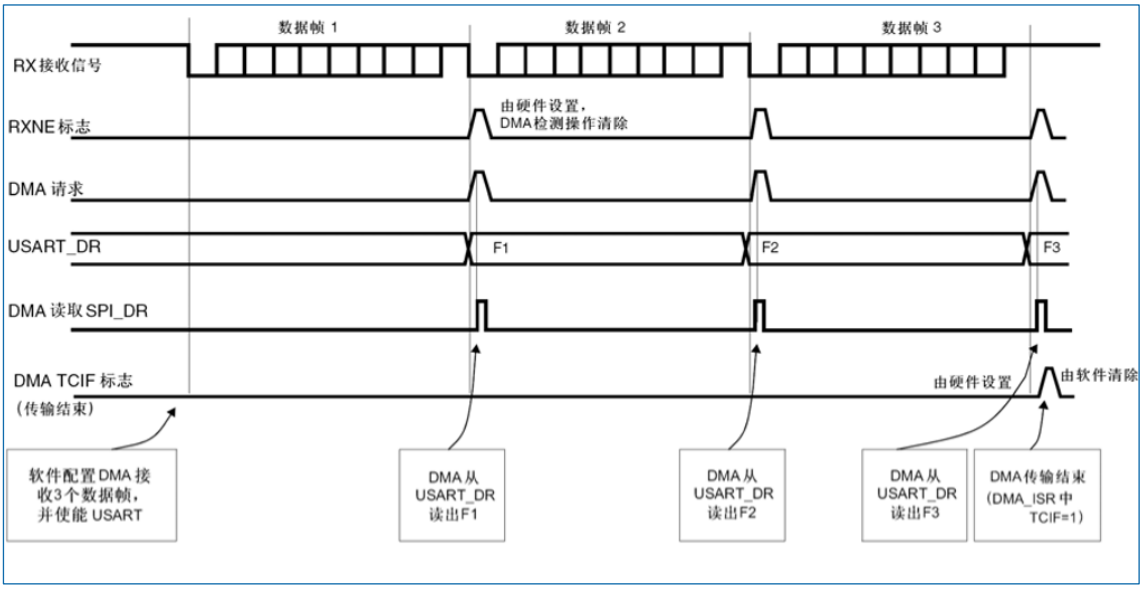


图 26-20 利用 DMA 接收

26.3.13.3 多缓冲器通信中的错误标志和中断产生

在多缓冲器通信的情况下，通信期间如果发生任何错误，在当前字节传输后将置起错误标志。如果中断使能位被设置，将产生中断。在单个字节接收的情况下，和 RXNE 一起被置起的帧错误、溢出错误和噪音标志，有单独的错误标志中断使能位；如果设置了此位，会在当前字节传输结束后，产生中断。

26.3.14 硬件流控制

利用 CTS 输入和 RTS 输出可以控制 2 个设备间的串行数据流。下图表明在这个模式里如何连接 2 个设备。

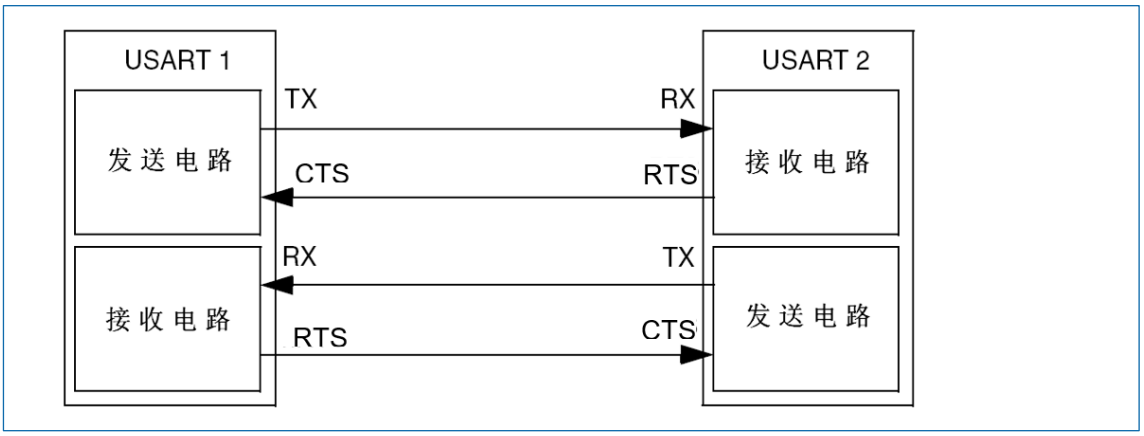


图 26-21 两个 USART 间的硬件流控制

通过将 UASRT_CR3 中的 RTSE 和 CTSE 置位，可以分别独立地使能 RTS 和 CTS 流控制。

26.3.14.1 RTS 流控制

如果 RTS 流控制被使能（RTSE=1），只要 USART 接收器准备好接收新的数据，RTS 就变成有效（接低电平）。当接收寄存器内有数据到达时，RTS 被释放，由此表明希望在当前帧结束时停止数据传输。下图是一个启用 RTS 流控制的通信的例子。

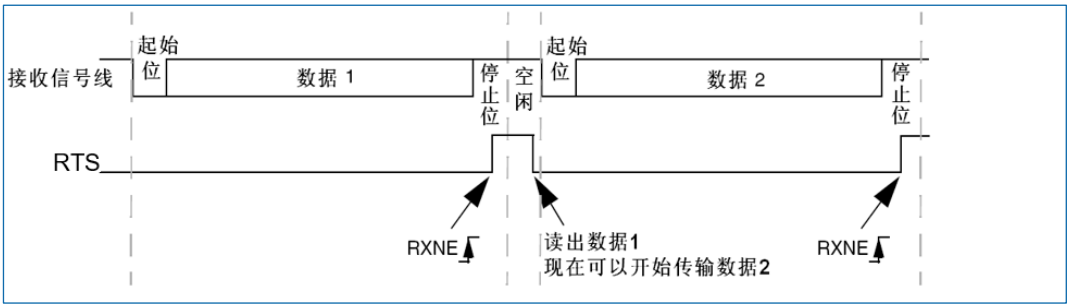


图 26-22 RTS 流控制

26.3.14.2 CTS 流控制

如果 CTS 流控制被使能（CTSE=1），发送器在发送下一帧前检查 CTS 输入。如果 CTS 有效（被拉成低电平），则下一个数据被发送（假设那个数据是准备发送的，也就是 TXE=0），否则下一帧数据不被发出去。若 CTS 在传输期间被变成无效，当前的传输完成后停止发送。

当 CTSE=1 时，只要 CTS 输入变换状态，硬件就自动设置 CTSIF 状态位。它表明接收器是否准备好进行通信。如果设置了 USARTx_CR3 寄存器的 CTSIE 位，则产生中断。下图是一个启用CTS 流控制通信的例子。

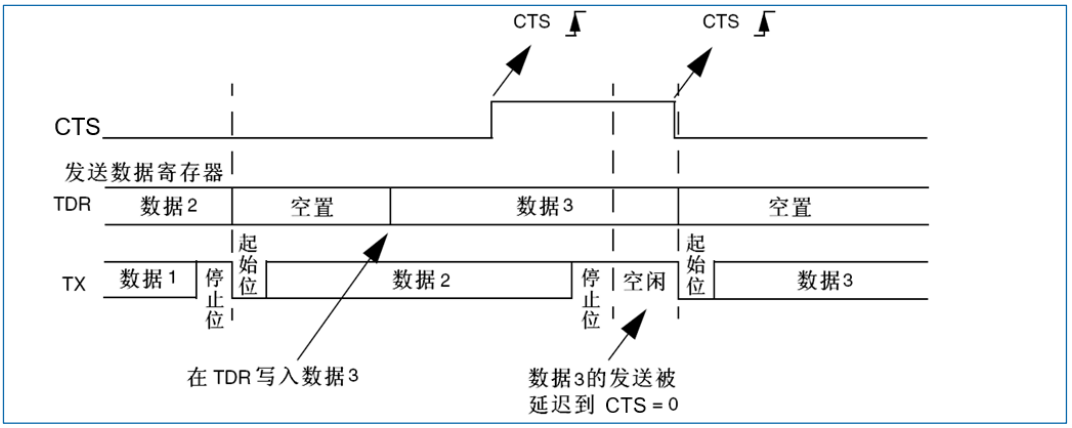


图 26-23 CTS 流控制

26.4 USART 中断请求

表 26-6 USART 中断请求

中断事件	事件标志	使能位
发送数据寄存器空	TXE	TXEIE
CTS 标志	CTS	CTSIE
发送完成	TC	TCIE
接收数据就绪可读	RXNE	RXNEIE

中断事件	事件标志	使能位
检测到数据溢出	ORE	
检测到空闲线路	IDLE	IDLEIE
奇偶检验错	PE	PEIE
断开标志	LBD	LBDIE
噪声标志，多缓冲通信中的溢出错误和帧错误	NE 或 ORE 或 FE	EIE (1)

(1). 仅当使用 DMA 接收数据时，才使用这个标志位。

USART 的各种中断事件被连接到同一个中断向量（见下图），有以下各种中断事件：

- 发送期间：发送完成、清除发送、发送数据寄存器空。
- 接收期间：空闲总线检测、溢出错误、接收数据寄存器非空、校验错误、LIN 断开符号检 测、噪音标志（仅在多缓冲器通信）和帧错误（仅在多缓冲器通信）。

如果设置了对应的使能控制位，这些事件就可以产生各自的中断。

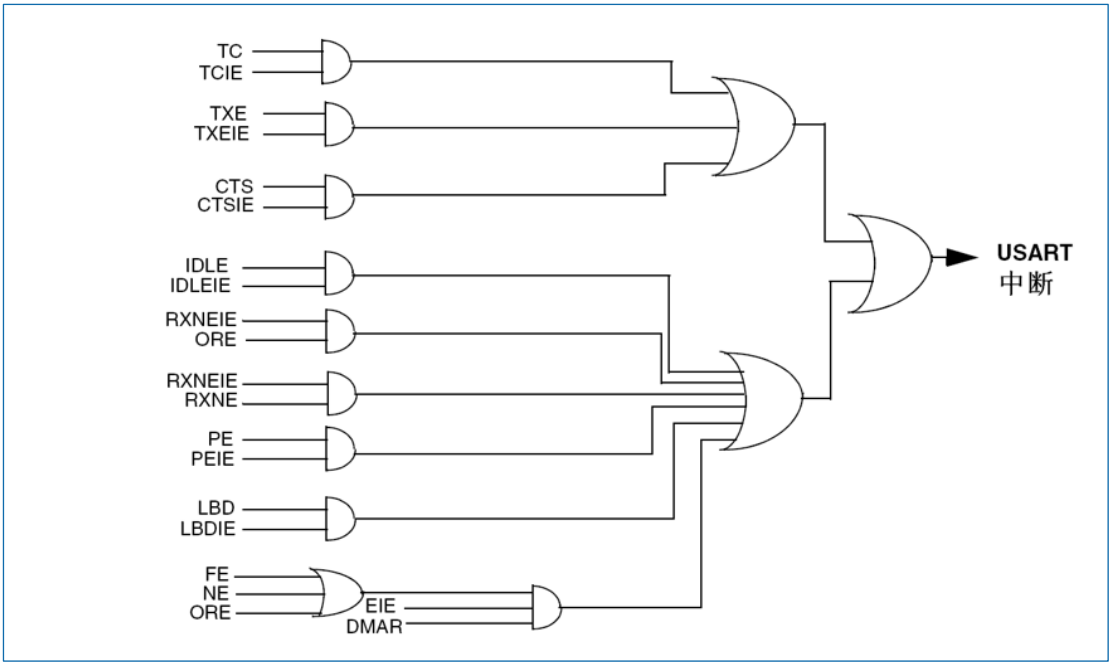


图 26-24 USART 中断

26.5 USART 模式配置

表 26-7 USART 模式设置

USART 模式	USART1/2/3/4/5/6
异步模式	支持
硬件流控制	支持
多缓存通讯（DMA）	支持
多处理器通讯	支持
同步	支持

USART 模式	USART1/2/3/4/5/6
智能卡	支持
半双工（单线模式）	支持
IrDA	支持
LIN	支持

26.6 USART 寄存器

基地址：（USART1, USART2, USART3, USART4, USART5, USART6）=（0x4001 3800, 0x4000 4400, 0x4000 4800, 0x4000 4C00, 0x4000 5000, 0x4001 4400）

空间大小：（USART1, USART2, USART3, USART4, USART5, USART6）=（0x400, 0x400, 0x400, 0x400, 0x400, 0x400）

可以用半字（16 位）或字（32 位）的方式操作这些外设寄存器。

26.6.1 状态寄存器（USARTx_SR）（x=1..6）

偏移地址：0x00

复位值：0x00C0

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res						CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
						rc_w0	rc_w0	r	rc_w0	rc_w0	r	r	r	r	r

位 31:10	Res：保留 必须保持复位值。
位 9	CTS：CTS 标志（CTS flag） 如果设置了 CTSE 位，当 CTS 输入变化状态时，该位被硬件置高。由软件将其清零（向该位写 0）。 如果 USARTx_CR3 中的 CTSIE 为'1'，则产生中断。 • 0：CTS 状态线上没有变化 • 1：CTS 状态线上发生变化
位 8	LBD：LIN 断开检测标志（LIN break detection flag） 当探测到 LIN 断开时，该位由硬件置'1'，由软件清零（向该位写 0）。 • 0：没有检测到 LIN 断开 • 1：检测到 LIN 断开 *注意：若 USARTx_CR3 中的 LBDIE=1，当 LBD 为'1'时要产生中断。*
位 7	TXE：发送数据寄存器空（Transmit data register empty） 当 TDR 寄存器中的数据被硬件转移到移位寄存器的时候，该位被硬件置位。如果 USARTx_CR1 寄存器中的 TXEIE 为 1，则产生中断。对 USARTx_DR 的写操作，将该位清零。 • 0：数据还没有被转移到移位寄存器 • 1：数据已经被转移到移位寄存器 *注意：单缓冲器传输中使用该位。*
位 6	TC：发送完成（Transmission complete） 当包含有数据的一帧发送完成后，并且 TXE=1 时，由硬件将该位置'1'。如果 USARTx_CR1 中的 TCIE 为'1'，

	则产生中断。由软件序列清除该位（先读 USARTx_SR，然后写入 USARTx_DR）。TC 位也可以通过写入'0'来清除，只有在多缓存通讯中才推荐这种清除程序。 • 0：发送还未完成 • 1：发送完成
位 5	RXNE：读数据寄存器非空（Read data register not empty） 当 RDR 移位寄存器中的数据被转移到 USARTx_DR 寄存器中，该位被硬件置位。如果 USARTx_CR1 寄存器中的 RXNEIE 为 1，则产生中断。对 USARTx_DR 的读操作可以将该位清零。RXNE 位也可以通过写入 0 来清除，只有在多缓存通讯中才推荐这种清除程序。 • 0：数据没有收到 • 1：收到数据，可以读出
位 4	IDLE：监测到总线空闲（IDLE line detected） 当检测到总线空闲时，该位被硬件置位。如果 USARTx_CR1 中的 IDLEIE 为'1'，则产生中断。由软件序列清除该位（先读 USARTx_SR，然后读 USARTx_DR）。 • 0：没有检测到空闲总线 • 1：检测到空闲总线 *注意：IDLE 位不会再次被置高直到 RXNE 位被置起（即又检测到一次空闲总线）。*
位 3	ORE：过载错误（Overrun error） 当 RXNE 仍然是'1'的时候，当前被接收在移位寄存器中的数据，需要传送至 RDR 寄存器时，硬件将该位置位。如果 USARTx_CR1 中的 RXNEIE 为'1'的话，则产生中断。由软件序列将其清零（先读 USARTx_SR，然后读 USARTx_CR）。 • 0：没有过载错误 • 1：检测到过载错误 *注意：该位被置位时，RDR 寄存器中的值不会丢失，但是移位寄存器中的数据会被覆盖。如果设置了 EIE 位，在多缓冲器通信模式下，ORE 标志置位会产生中断的。*
位 2	NE：噪声错误标志（Noise error flag） 在接收到的帧检测到噪音时，由硬件对该位置位。由软件序列对其清零（先读 USARTx_SR，再读 USARTx_DR）。 • 0：没有检测到噪声 • 1：检测到噪声 *注意：该位不会产生中断，因为它和 RXNE 一起出现，硬件会在设置 RXNE 标志时产生中断。在多缓冲区通信模式下，如果设置了 EIE 位，则设置 NE 标志时会产生中断。*
位 1	FE：帧错误（Framing error） 当检测到同步错位，过多的噪声或者检测到断开符，该位被硬件置位。由软件序列将其清零（先读 USARTx_SR，再读 USARTx_DR）。 • 0：没有检测到帧错误 • 1：检测到帧错误或者 break 符 *注意：* • *该位不会产生中断，因为它和 RXNE 一起出现，硬件会在设置 RXNE 标志时产生中断。* • *如果当前传输的数据既产生了帧错误，又产生了过载错误，硬件还是会继续该数据的传输，并且只设置 ORE 标志位。* • *在多缓冲区通信模式下，如果设置了 EIE 位，则设置 FE 标志时会产生中断。*
位 0	PE：校验错误（Parity error） 在接收模式下，如果出现奇偶校验错误，硬件对该位置位。由软件序列对其清零（依次读 USARTx_SR 和

	USARTx_DR）。在清除 PE 位前，软件必须等待 RXNE 标志位被置'1'。如果 USARTx_CR1 中的 PEIE 为'1'，则产生中断。 • 0：没有奇偶校验错误 • 1：奇偶校验错误

26.6.2 数据寄存器（USARTx_DR）（x=1..6）

偏移地址：0x04

复位值：0xXXXX XXXX

说明：X 表示不定值

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res							DR[8:0]								
							rw								

位 31:9	Res：保留 必须保持复位值。
位 8:0	DR[8:0]：数据值（Data value） 包含了发送或接收的数据。由于它是由两个寄存器组成的，一个给发送用（TDR），一个给接收用（RDR），该寄存器兼具读和写的功能。 TDR 寄存器提供了内部总线和输出移位寄存器之间的并行接口（参见图 26-1）。RDR 寄存器提供了输入移位寄存器和内部总线之间的并行接口。 当使能校验位（USARTx_CR1 中 PCE 位被置位）进行发送时，写到 MSB 的值（根据数据的长度不同，MSB 是第 7 位或者第 8 位）会被后来的校验位该取代。 当使能校验位进行接收时，读到的 MSB 位是接收到的校验位。

26.6.3 波特比率寄存器（USARTx_BRR）（x=1..6）

偏移地址：0x08

复位值：0x0000

注意：如果 TE 或 RE 被分别禁止，波特计数器停止计数

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV_Mantissa[11:0]												DIV_Fraction[3:0]			
rw												rw			

位 31:16	Res：保留 必须保持复位值。
位 15:4	DIV_Mantissa[11:0]：USARTDIV 的整数部分（Mantissa of USARTDIV） 这 12 位定义了 USART 分频器除法因子（USARTDIV）的整数部分。
位 3:0	DIV_Fraction[3:0]：USARTDIV 的小数部分（Fraction of USARTDIV） 这 4 位定义了 USART 分频器除法因子（USARTDIV）的小数部分。

26.6.4 控制寄存器 1（USARTx_CR1）（x=1..6）

偏移地址：0x0C

复位值：0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res		UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:14	Res：保留 必须保持复位值。
位 13	UE：USART 使能（USART enable） 当该位被清零，在当前字节传输完成后 USART 的分频器和输出停止工作，以减少功耗。该位由软件设置和清零。 • 0：USART 分频器和输出被禁止 • 1：USART 模块使能
位 12	M：字长（Word length） 该位定义了数据字的长度，由软件对其设置和清零 • 0：一个起始位，8 个数据位，n 个停止位 • 1：一个起始位，9 个数据位，n 个停止位 *注意：在数据传输过程中（发送或者接收时），不能修改这个位。*
位 11	WAKE：唤醒的方法（Wakeup method） 这位决定了把 USART 唤醒的方法，由软件对该位设置和清零。 • 0：被空闲总线唤醒 • 1：被地址标记唤醒
位 10	PCE：检验控制使能（Parity control enable） 用该位选择是否进行硬件校验控制（对于发送来说就是校验位的产生；对于接收来说就是校验位的检测）。当使能了该位，在发送数据的最高位（如果 M=1，最高位就是第 9 位；如果 M=0，最高位就是第 8 位）插入校验位；对接收到的数据检查其校验位。软件对它置’1’或清零。一旦设置了该位，当前字节传输完成后，校验控制才生效。 • 0：禁止校验控制 • 1：使能校验控制
位 9	PS：校验选择（Parity selection） 当校验控制使能后，该位用来选择是采用偶校验还是奇校验。软件对它置’1’或清零。当前字节传输完成后，该选择生效。 • 0：偶校验 • 1：奇校验
位 8	PEIE：PE 中断使能（PE interrupt enable） 该位由软件设置或清除。 • 0：禁止产生中断 • 1：当 USARTx_SR 中的 PE 为’1’时，产生 USART 中断

位 7	TXEIE：发送缓冲区空中断使能（TXE interrupt enable） 该位由软件设置或清除。 • 0：禁止产生中断 • 1：当 USARTx_SR 中的 TXE 为’1’时，产生 USART 中断
位 6	TCIE：发送完成中断使能（Transmission complete interrupt enable） 该位由软件设置或清除。 • 0：禁止产生中断 • 1：当 USARTx_SR 中的 TC 为’1’时，产生 USART 中断
位 5	RXNEIE：接收缓冲区非空中断使能（RXNE interrupt enable） 该位由软件设置或清除。 • 0：禁止产生中断 • 1：当 USARTx_SR 中的 ORE 或者 RXNE 为’1’时，产生 USART 中断
位 4	IDLEIE：IDLE 中断使能（IDLE interrupt enable） 该位由软件设置或清除。 • 0：禁止产生中断 • 1：当 USARTx_SR 中的 IDLE 为’1’时，产生 USART 中断
位 3	TE：发送使能（Transmitter enable） 该位使能发送器。该位由软件设置或清除。 • 0：禁止发送 • 1：使能发送 *注意：* • *在数据传输过程中，除了在智能卡模式下，如果 TE 位上有个 0 脉冲（即设置为’0’之后再设置为’1’），会在当前数据字传输完成后，发送一个前导符（空闲总线）。* • *当 TE 被设置后，在真正发送开始之前，有一个比特时间的延迟。*
位 2	RE：接收使能（Receiver enable） 该位由软件设置或清除。 • 0：禁止接收 • 1：使能接收，并开始搜寻 RX 引脚上的起始位
位 1	RWU：接收唤醒（Receiver wakeup） 该位用来决定是否把 USART 置于静默模式。该位由软件设置或清除。当唤醒序列到来时，硬件也会将其清零。 • 0：接收器处于正常工作模式 • 1：接收器处于静默模式 *注意：* • *在把 USART 置于静默模式（设置 RWU 位）之前，USART 必须已经先接收了一个数据字节。否则在静默模式下，不能被空闲总线检测唤醒。* • *当配置成地址标记检测唤醒（WAKE 位=1），在 RXNE 位被置位时，不能用软件修改 RWU 位。*
位 0	SBK：发送断开帧（Send break） 使用该位来发送断开字符。该位可以由软件设置或清除。操作过程应该是软件设置位它，然后在断开帧的停止位时，由硬件将该位复位。 • 0：没有发送断开字符

	• 1：将要发送断开字符

26.6.5 控制寄存器 2（USARTx_CR2）（x=1..6）

偏移地址：0x10

复位值：0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res	LBDIE	LBDL	Res	ADD[3:0]			
	rw	rw		rw	rw	rw	rw		rw	rw		rw			

位 31:15	Res：保留 必须保持复位值。
位 14	LINEN：LIN 模式使能（LIN mode enable） 该位由软件设置或清除。 • 0：禁止 LIN 模式 • 1：使能 LIN 模式 在 LIN 模式下，可以用 USARTx_CR1 寄存器中的 SBK 位发送 LIN 同步断开符（低 13 位），以及检测 LIN 同步断开符。
位 13:12	STOP[1:0]：停止位（STOP bits） 这 2 位用来设置停止位的位数。 • 00：1 个停止位 • 01：0.5 个停止位 • 10：2 个停止位 • 11：1.5 个停止位
位 11	CLKEN：时钟使能（Clock enable） 该位用来使能 CK 引脚。 • 0：禁止 CK 引脚 • 1：使能 CK 引脚
位 10	CPOL：时钟极性（Clock polarity） 在同步模式下，可以用该位选择 CK 引脚上时钟输出的极性。和 CPHA 位一起配合来产生需要的时钟/数据的采样关系。 • 0：总线空闲时 CK 引脚上保持低电平 • 1：总线空闲时 CK 引脚上保持高电平
位 9	CPHA：时钟相位（Clock phase） 在同步模式下，可以用该位选择 CK 引脚上时钟输出的相位。和 CPOL 位一起配合来产生需要的时钟/数据的采样关系（参见图 26-12 和图 26-13）。 • 0：在时钟的第一个边沿进行数据捕获 • 1：在时钟的第二个边沿进行数据捕获
位 8	LBCL：最后一位时钟脉冲（Last bit clock pulse） 在同步模式下，使用该位来控制是否在 CK 引脚上输出最后发送的那个数据字节（MSB）对应的时钟脉冲。

	• 0：最后一位数据的时钟脉冲不从 CK 输出 • 1：最后一位数据的时钟脉冲会从 CK 输出 *注意：* *最后一个数据位就是第 8 或者第 9 个发送的位（根据 USARTx_CR1 寄存器中的 M 位所定义的 8 或者 9 位数据帧格式）。*
位 7	Res：保留 必须保持复位值。
位 6	LBDIE：LIN 断开符检测中断使能（LIN break detection interrupt enable） 断开符中断屏蔽（使用断开分隔符来检测断开符）。 • 0：禁止中断 • 1：只要 USARTx_SR 寄存器中的 LBD 为’1’就产生中断
位 5	LBDL：LIN 断开符检测长度（LIN break detection length） 该位用来选择是 11 位还是 10 位的断开符检测。 • 0：10 位的断开符检测 • 1：11 位的断开符检测
位 4	Res：保留 必须保持复位值。
位 3:0	ADD[3:0]：本设备的 USART 节点地址（Address of the USART node） 该位域给出本设备 USART 节点的地址。 这是在多处理器通信下的静默模式中使用的，使用地址标记来唤醒某个 USART 设备。

注意：在使能发送后不能改写这三个位：CPOL、CPHA、LBCL。

26.6.6 控制寄存器 3（USARTx_CR3）（x=1..6）

偏移地址：0x14

复位值：0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res					CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:11	Res：保留 必须保持复位值。
位 10	CTSIE：CTS 中断使能（CTS interrupt enable） • 0：禁止中断 • 1：USARTx_SR 寄存器中的 CTS 为’1’时产生中断
位 9	CTSE：CTS 使能（CTS enable） • 0：禁止 CTS 硬件流控制 • 1：CTS 模式使能 只有 CTS 输入信号有效（拉成低电平）时才能发送数据。如果在数据传输的过程中，CTS 信号变成无效，那么发完这个数据后，传输就停止下来。如果当 CTS 为无效时，向数据寄存器里写数据，则要等到

	CTS 有效时才会发送这个数据。
位 8	RTSE：RTS 使能（RTS enable） • 0：禁止 RTS 硬件流控制 • 1：RTS 中断使能 只有接收缓冲区内有空余的空间时才请求下一个数据。当前数据发送完成后，发送操作就需要暂停下来。如果可以接收数据了，将 RTS 输出置为有效（拉至低电平）。
位 7	DMAT：DMA 使能发送（DMA enable transmitter） 该位由软件设置或清除。 • 0：禁止发送时的 DMA 模式 • 1：使能发送时的 DMA 模式
位 6	DMAR：DMA 使能接收（DMA enable receiver） 该位由软件设置或清除。 • 0：禁止接收时的 DMA 模式 • 1：使能接收时的 DMA 模式
位 5	SCEN：智能卡模式使能（Smartcard mode enable） 该位用来使能智能卡模式 • 0：禁止智能卡模式 • 1：使能智能卡模式。
位 4	NACK：智能卡 NACK 使能（Smartcard NACK enable） • 0：在奇偶校验没有打开时，发送 NACK • 1：在奇偶校验错误出现时，发送 NACK
位 3	HDSEL：半双工选择（Half-duplex selection） 选择单线半双工模式 • 0：不选择半双工模式 • 1：选择半双工模式
位 2	IRLP：红外低功耗（IrDA low-power） 该位用来选择普通模式还是低功耗红外模式 • 0：通常模式 • 1：低功耗模式
位 1	IREN：红外模式使能（IrDA mode enable） 该位由软件设置或清除。 • 0：不使能红外模式 • 1：使能红外模式
位 0	EIE：错误中断使能（Error interrupt enable） 在多缓冲区通信模式下，当有帧错误、过载或者噪声错误时（USARTx_SR 中的 FE=1，或者 ORE=1，或者 NE=1）产生中断。 • 0：禁止中断 • 1：只要 USARTx_CR3 中的 DMAR=1，并且 USARTx_SR 中的 FE=1，或者 ORE=1，或者 NE=1，则产生中断

26.6.7 保护时间和预分频寄存器（USARTx_GTPR）（x=1..6）

偏移地址：0x18

复位值：0x0000

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

15		14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GT[7:0]									PSC[7:0]							
rw									rw							

位 31:16	Res：保留 必须保持复位值。
位 15:8	GT[7:0]：保护时间值（Guard time value） 该位域规定了以波特时钟为单位的保护时间。在智能卡模式下，需要这个功能。当保护时间过去后，才会设置发送完成标志。
位 7:0	PSC[7:0]：预分频器值（Prescaler value） • 在红外（IrDA）低功耗模式下： PSC[7:0]=红外低功耗波特率 • 对系统时钟分频以获得低功耗模式下的频率：源时钟被寄存器中的值（仅有 8 位有效）分频 o 00000000：保留– 不要写入该值 o 00000001：对源时钟 1 分频 o 00000010：对源时钟 2 分频 o … • -在红外（IrDA）的正常模式下： PSC 只能设置为 00000001 • 在智能卡模式下： PSC[4:0]：预分频值 对系统时钟进行分频，给智能卡提供时钟。寄存器中给出的值（低 5 位有效）乘以 2 后，作为对源时钟的分频因子 o 00000：保留，不要写入该值 o 00001：对源时钟进行 2 分频 o 00010：对源时钟进行 4 分频 o 00011：对源时钟进行 6 分频 o … *注意：位[7:5]在智能卡模式下没有意义。*

27 Quad-SPI 接口（QSPI）

QSPI 是一种专用的通信接口，连接单、双或四（条数据线）SPI Flash 存储介质。该接口可以在以下三种模式下工作：

- 间接模式：使用 QSPI 寄存器执行全部操作。
- 状态轮询模式：周期性读取外部 Flash 状态寄存器，而且标志位置 1 时会产生中断（如擦除或写入完成，会产生中断）。
- 内存映射模式：外部 Flash 映射到微控制器地址空间，从而系统将其视作内部寄存器。

采用双闪存模式时，将同时访问两个 Quad-SPI Flash，吞吐量和容量均可提高二倍。

27.1 QSPI 主要特性

- 三种功能模式：间接模式、状态轮询模式和内存映射模式。
- 双闪存模式，通过并行访问两个 Flash，可同时发送/接收 8 位数据。
- 支持 SDR 和 DDR 模式。
- 针对间接模式和内存映射模式，完全可编程操作码。
- 针对间接模式和内存映射模式，完全可编程帧格式。
- 集成 FIFO，用于发送和接收。
- 允许 8、16 和 32 位数据访问。
- 具有适用于间接模式的 DMA 通道。
- 在达到 FIFO 阈值、超时、操作完成以及发生访问错误时产生中断。

27.2 QSPI 功能说明

27.2.1 QSPI 框图

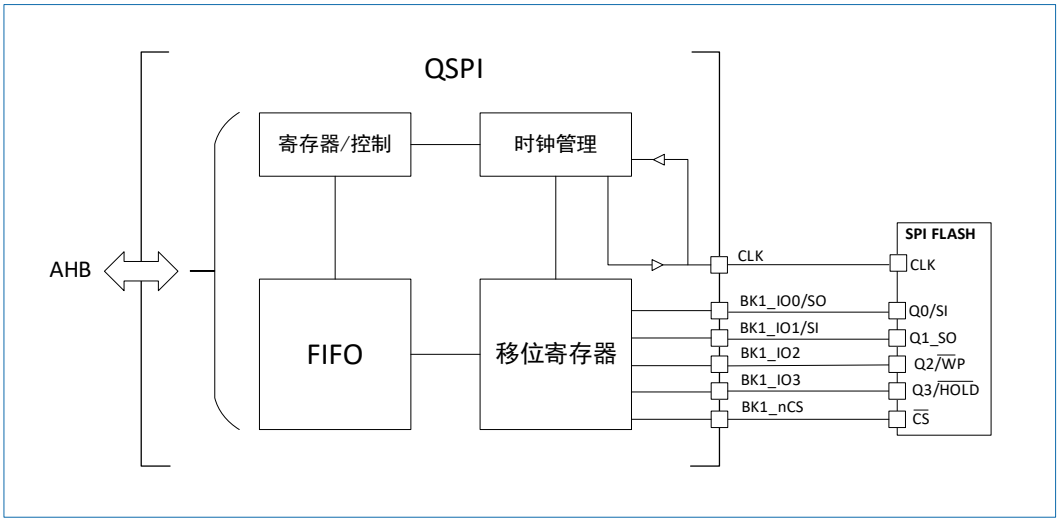


图 27-1 QSPI 功能框图（双闪存模式禁止）

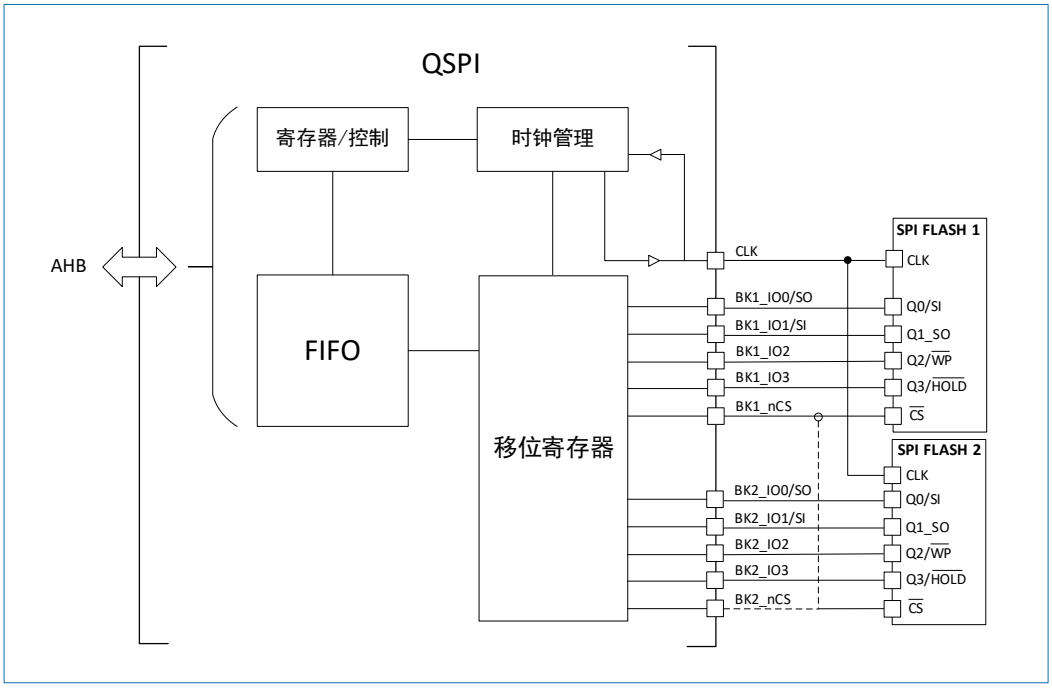


图 27-2 QSPI 功能框图（双闪存模式使能）

在双闪存模式中，QSPI 使用 6 个信号连接单个 Flash，使用 10 到 11 个信号以 dual 的方式连接两个 Flash（FLASH1 和 FLASH2）：

- CLK：时钟输出（适用于 Flash1 和 Flash2）
- BK1_nCS/BK2_nCS：QSPI 工作于单个 Flash 模式时，片选输出（低电平有效）；工作在双闪存模式时，则 BK1_nCS/BK2_nCS 适用于 Flash1 和 Flash2。
- 以下信号适用于 Flash1:
  - BK1_IO0/SO：在双线/四线模式中为双向 IO，单线模式中为串行输出。
  - BK1_IO1/SI：在双线/四线模式中为双向 IO，单线模式中为串行输入。
  - BK1_IO2：在四线模式中为双向 IO。
  - BK1_IO3：在四线模式中为双向 IO。
- 以下信号适用于 Flash2:
  - BK2_IO0/SO：在双线/四线模式中为双向 IO，单线模式中为串行输出。
  - BK2_IO1/SI：在双线/四线模式中为双向 IO，单线模式中为串行输入。
  - BK2_IO2：在四线模式中为双向 IO。
  - BK2_IO3：在四线模式中为双向 IO。

27.2.2 QSPI 命令序列

QSPI 通过命令与 Flash 通信每条命令包括指令、地址、交替字节、空指令周期和数据。这五个阶段任一阶段均可跳过，但至少要包含指令、地址、交替字节或数据阶段四者之一。

nCS 在每条指令开始前下降，在每条指令完成后再次上升。

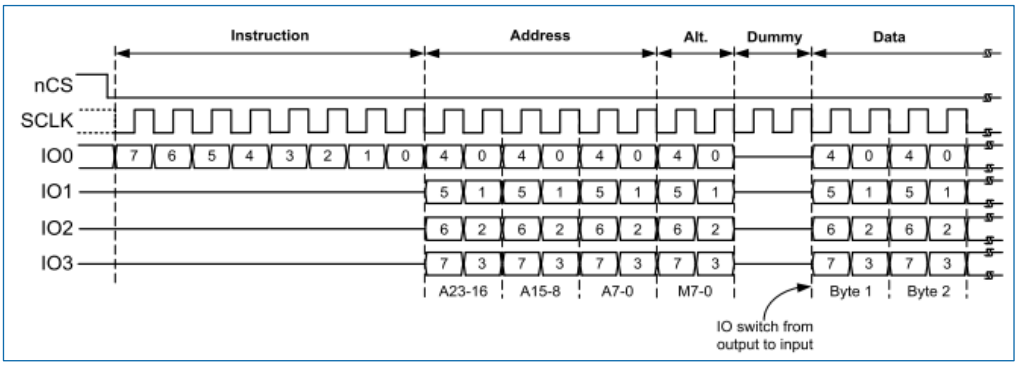


图 27-3 四线模式下的读命令示例

指令阶段

这一阶段，将在 QSPI_CCR 寄存器的 INSTRUCTION[7:0]字段中配置的一条 8 位指令发送到 Flash，指定待执行操作的类型。

尽管大多数 Flash 从 IO0/SO 信号（单线 SPI 模式）只能以一次 1 比特的方式接收指令，但指令阶段可选择一次发送 2 比特（在双线 SPI 模式中通过 IO0/IO1）或一次发送 4 比特（在四线 SPI 模式中通过 IO0/IO1/IO2/IO3）。这可通过 QSPI_CCR 寄存器中的 IMODE[1:0]字段进行配置。

若 IMODE=00，则跳过指令阶段，命令序列从地址阶段（如果存在）开始。

地址阶段

在地址阶段，将 1-4 字节发送到 Flash，指示操作地址。待发送的地址字节数在 QSPI_CCR 寄存器的 ADSIZE[1:0]字段中进行配置。在间接模式和自动轮询模式下，待发送的地址字节在 QSPI_AR 寄存器的 ADDRESS[31:0]中指定。在内存映射模式下，则通过 AHB（来自于 Cortex®或 DMA）直接给出地址。

地址阶段可一次发送 1 比特（在单线 SPI 模式中通过 SO）、2 比特（在双线 SPI 模式中通过 IO0/IO1）或 4 比特（在四线 SPI 模式中通过 IO0/IO1/IO2/IO3）。这可通过 QSPI_CCR 寄存器中的 ADMODE[1:0]字段进行配置。

若 ADMODE=00，则跳过地址阶段，如果存在下一阶段，则命令序列直接进入下一阶段。

交替字节阶段

在交替字节阶段，将 1-4 字节发送到 Flash，一般用于控制操作模式。待发送的交替字节数在 QSPI_CCR 寄存器的 ABSIZE[1:0]字段中进行配置。待发送的字节在 QSPI_ABR 寄存器中指定。

交替字节阶段可一次发送 1 比特（在单线 SPI 模式中通过 SO）、2 比特（在双线 SPI 模式中通过 IO0/IO1）或 4 比特（在四线 SPI 模式中通过 IO0/IO1/IO2/IO3）。这可通过 QSPI_CCR 寄存器中的 ABMODE[1:0]字段进行配置。

若 ABMODE=00，则跳过交替字节阶段，如果存在下一阶段，命令序列直接进入下一阶段。

交替字节阶段存在仅需发送单个半字节而不是一个全字节的情况，比如采用双线模式并且仅使用两个周期发送交替字节时。在这种情况下，固件可采用四线模式（ABMODE=11）并发送一个字节，方法是 ALTERNATE 的位 7 和 3 置“1”（IO3 保持高电平）且位 6 和 2 置“0”（IO2 线保持低电平）。此时，半字节的高 2 位存放在 ALTERNATE 的位 5:4，低 2 位存放在位 1 和 0 中。例如，如果半字节 2（0010）通过 IO0/IO1 发送，则 ALTERNATE 应设置为 0x8A（1000_1010）。

空指令周期阶段

在空指令周期阶段，给定的 1-31 个周期内不发送或接收任何数据，目的是当采用更高的时钟频率时，给 Flash 留出准备数据阶段的时间。这一阶段中给定的周期数在 QSPI_CCR 寄存器的 DCYC[4:0]字段中指定。在 SDR 和 DDR 模式下，持续时间被指定为一定个数的全时钟周期。

若 DCYC 为零，则跳过空指令周期阶段，如果存在数据阶段，命令序列将直接进入数据阶段。空指

令周期阶段的操作模式由 DMODE 确定。

注意：为确保数据信号从输出模式转变为输入模式有足够的“周转”时间，使用双线和四线模式从 Flash 接收数据时，至少需要指定一个空指令周期。

数据阶段

在数据阶段，可从 Flash 接收或向其发送任意数量的字节。在间接模式和自动轮询模式下，待发送/接收的字节数在 QSPI_DLR 寄存器中指定。在间接写入模式下，发送到 Flash 的数据必须写入 QSPI_DR 寄存器。在间接读取模式下，通过读取 QSPI_DR 寄存器获得从 Flash 接收的数据。

在内存映射模式下，读取的数据通过 AHB 直接发送回 Cortex 或 DMA。

数据阶段可一次发送/接收 1 比特（在单线 SPI 模式中通过 SO）、2 比特（在双线 SPI 模式中通过 IO0/IO1）或 4 比特（在四线 SPI 模式中通过 IO0/IO1/IO2/IO3）。这可通过 QSPI_CCR 寄存器中的 DMODE[1:0]字段进行配置。

若 DMODE=00，则跳过数据阶段，命令序列在拉高 nCS 时立即完成。这一配置仅可用于仅间接写入模式。

27.2.3 QSPI 信号接口协议模式

单线 SPI 模式

传统 SPI 模式仅允许串行发送/接收单独的 1 比特。在此模式下，数据通过 SO 信号（其 I/O 与 IO0 共享）发送到 Flash。从 Flash 接收到的数据通过 SI（其 I/O 与 IO1 共享）送达。

通过将（QSPI_CCR 中的）IMODE/ADMODE/ABMODE/DMODE 字段设置为 01，可对不同的命令阶段分别进行配置，以使用此单个位模式。

在每个配置为单线模式的命令阶段中：

- IO0（SO）处于输出模式
- IO1（SI）处于输入模式（高阻抗）
- IO2 处于输出模式并强制置“0”（以禁止“写保护”功能）
- IO3 处于输出模式并强制置“1”（以禁止“保持”功能）

若 DMODE=01，这对于空指令阶段也同样如此。

双线 SPI 模式

在双线模式下，通过 IO0/IO1 信号同时发送/接收两比特。

通过将 QSPI_CCR 寄存器的 IMODE/ADMODE/ABMODE/DMODE 字段设置为 10，可对不同的命令阶段分别进行配置，以使用双线 SPI 模式。

在每个已配置为双线模式的阶段中：

- IO0/IO1 在数据阶段进行读取操作时处于高阻态（输入），在其他情况下为输出。
- IO2 处于输出模式并强制置’’0”。
- IO3 处于输出模式并强制置’’1”。

在空指令阶段，若 DMODE=10，则 IO0/IO1 始终保持高阻态。

四线 SPI 模式

在四线模式下，通过 IO0/IO1/IO2/IO3 信号同时发送/接收四比特。

通过将 QSPI_CCR 寄存器的 IMODE/ADMODE/ABMODE/DMODE 字段设置为 11，可对不同的命令阶段分别进行配置，以使用四线 SPI 模式。

在每个已配置为四线模式的阶段中，IO0/IO1/IO2/IO3 在数据阶段进行读取操作时均处于高阻态（输入），在其他情况下为输出。

在空指令阶段中，若 DMODE=11，则 IO0/IO1/IO2/IO3 均为高阻态。

IO2 和 IO3 仅用于 QSPI 模式，如果未配置任何阶段使用四线 SPI 模式，即使 QSPI 激活，对应 IO2 和 IO3 的引脚也可用于其他功能。

SDR 模式

默认情况下，DDRM 位（QSPI_CCR）为 0，QSPI 在单倍数据速率（SDR）模式下工作。

在 SDR 模式下，当 QSPI 驱动 IO0/SO、IO1、IO2、IO3 信号时，这些信号仅在 CLK 的下降沿发生转变。

在 SDR 模式下接收数据时，QSPI 假定 Flash 也通过 CLK 的下降沿发送数据。默认情况下（SSHIFT=0 时），将使用 CLK 后续的边沿（上升沿）对信号进行采样。

DDR 模式

若 DDRM 位（QSPI_CCR[31]）置 1，则 QSPI 在双倍数据速率（DDR）模式下工作。

在 DDR 模式下，当 QSPI 在地址/交替字节/数据阶段驱动 IO0/SO、IO1、IO2、IO3 信号时，将在 CLK 的每个上升沿和下降沿发送 1 位。

指令阶段不受 DDRM 的影响。始终通过 CLK 的下降沿发送指令。

在 DDR 模式下接收数据时，QSPI 假定 Flash 通过 CLK 的上升沿和下降沿均发送数据。若 DDRM=1，固件必须清零 SSHIFT 位（QSPI_CR）。因此，在半个 CLK 周期后（下一个反向边沿）对信号采样。

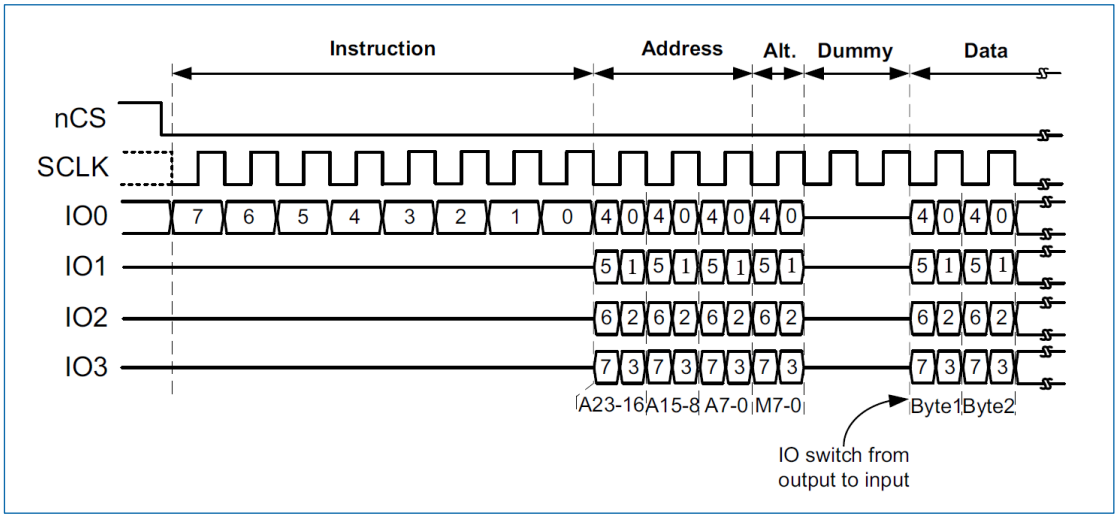


图 27-4 四线模式下 DDR 命令示例

双闪存模式

若 DFM 位（QSPI_CR）为 1，QSPI 处于双闪存模式。QSPI 使用两个外部四线 SPI Flash（FLASH1 和 FLASH2），在每个周期中发送/接收 8 位（在 DDR 模式下为 16 位），能够有效地将吞吐量和容量扩大一倍。

两个 Flash 使用同一个 CLK 并可选择使用同一个 nCS 信号，但其 IO0、IO1、IO2 和 IO3 信号是各自独立的。

双闪存模式可与单比特模式、双比特模式以及四比特模式结合使用，也可与 SDR 或 DDR 模式相结合。

Flash 的大小在 FSIZE[4:0]（QSPI_DCR）中指定，指定的值应能够反映 Flash 的总容量，即单个组件容量的 2 倍。

如果地址 X 为偶数，QSPI 赋给地址 X 的字节是存放于 FLASH1 的地址 X/2 中的字节，QSPI 赋给地址 X+1 的字节是存放于 FLASH2 的地址 X/2 中的字节。也就是说，偶地址中的字节存储于 FLASH1，奇地址中的字节存储于 FLASH2。

在双闪存模式下读取 Flash 状态寄存器时，需要读取的字节数是单闪存模式下的 2 倍。这意味着如果每个 Flash 在获取状态寄存器的指令后给出了 8 位有效位，则 QSPI 必须配置为 2 个字节（16 位）的数据长度，它将从每个 Flash 接收 1 个字节。如果每个 Flash 给出一个 16 位的状态，则 QSPI 必须配置为读取 4 字节，以在双闪存模式下可获取两个 Flash 的所有状态位。结果（在数据寄存器中）的最低有效字节是 FLASH1 状态寄存器的最低有效字节，而下一个字节是 FLASH2 状态寄存器的最低有效字节。数据寄存器的第三个字节是 FLASH1 的第二个字节，第四个字节是 FLASH2 的第二个字节（Flash 具有 16 位状态寄存器时）。

偶数字节必须始终在双闪存模式下访问。因此，若 DDRM=1，则数据长度字段（QSPI_DLR）的位 0 始终保持为 1。

在双闪存模式下，FLASH1 接口信号的行为基本上与正常模式下相同。在指令、地址、交替字节以及空指令周期阶段，FLASH2 接口信号具有与 FLASH1 接口信号完全相同的波形。也就是说，每个 Flash 总是接收相同的指令与地址。然后，在数据阶段，BK1_IOx 和 BK2_IOx 总线并行传输数据，但发送到 FLASH1（或从其接收）的数据与 FLASH2 中的不同。

27.2.4 QSPI 间接模式

在间接模式下，通过写 QSPI 寄存器来触发命令；并通过读写数据寄存器来传输数据，就如同对待其他通信外设那样。

若 FMODE=00（QSPI_CCR），则 QSPI 处于间接写入模式，字节在数据阶段中发送到 Flash。通过写入数据寄存器（QSPI_DR）的方式提供数据。

若 FMODE=01，则 QSPI 处于间接读取模式，在数据阶段中从 Flash 接收字节。通过读取 QSPI_DR 来获取数据。

读取/写入的字节数在数据长度寄存器（QSPI_DLR）中指定。如果 QSPI_DLR=0xFFFF_FFFF（全为“1”），则数据长度视为未定义，QSPI 将持续传输数据，直到到达由 FSIZE 定义的）Flash 的结尾。如果不传输任何字节，DMODE（QSPI_CCR）应设置为 00。

如果 QSPI_DLR=0xFFFF_FFFF 并且 FSIZE=0x1F（最大值指示一个 4GB 的 Flash），在此特殊情况下，传输将无限继续下去，仅在出现终止请求或 QSPI 被禁止后停止。在读取最后一个存储器地址后（地址为 0xFFFF_FFFF），将从地址=0x0000_0000 开始继续读取。

当发送或接收的字节数达到编程设定值时，如果 TCIE=1，则 TCF 置 1 并产生中断。在数据数量不确定的情况下，将根据 QSPI_DCR 中定义的 Flash 大小，在达到外部 SPI 的限制时，TCF 置 1。

触发命令启动

从本质上讲，在固件给出命令所需的最后一点信息时，命令即会启动。根据 QSPI 的配置，在间接模式下有三种触发命令启动的方式。在出现以下情形时，命令立即启动：

1. 在此条件下对 INSTRUCTION[7:0]（QSPI_CCR）执行写入操作：命令序列无地址（当 ADMODE=00）并且不需要固件提供数据（当 FMODE=01 或 DMODE=00）。
2. 在此条件下对 ADDRESS[31:0]（QSPI_AR）执行写入操作：地址是必需的（当 ADMODE!=00）并且不需要固件提供数据（当 FMODE=01 或 DMODE=00）。
3. 在此条件下对 DATA[31:0]（QSPI_DR）执行写入操作：地址是必需的（当 ADMODE!=00）并且需要固件提供数据（当 FMODE=00 并且 DMODE!=00）。

写入交替字节寄存器（QSPI_ABR）始终不会触发命令启动。如果需要交替字节，必须预先进行编程。

如果命令启动，BUSY 位（QSPI_SR 的位 5）将自动置 1。

FIFO 和数据管理

在间接模式中，数据将通过 QSPI 内部的一个 16 字节 FIFO。FLEVEL[4:0]（QSPI_SR）指示 FIFO 目前保存了多少字节。

在间接写入模式下（FMODE=00），固件写入 QSPI_DR 时，将在 FIFO 中加入数据。字写入将在 FIFO 中增加 4 个字节，半字写入增加 2 个字节，而字节写入仅增加 1 个字节。如果固件在 FIFO 中加入的数据过多（超出了 DL[31:0]配置的值），将在写入操作结束（TCF 置 1）时从 FIFO 中清除超出的字节。

对 QSPI_DR 的字节/半字访问必须仅针对该 32 位寄存器的最低有效字节/半字。

FTHRES[3:0]用于定义 FIFO 的阈值,如果达到阈值，FTF（FIFO 阈值标志）置 1。在间接读取模式下，从 FIFO 中读取的有效字节数超过阈值时，FTF 置 1。从 Flash 中读取最后一个字节后，如果 FIFO 中依然有数据，则无论 FTHRES 设置为何值，FTF 也都会置 1。在间接写入模式下，当 FIFO 中的空字节数超过阈值时，FTF 置 1。

如果 FTIE=1，则 FTF 置 1 时产生中断。如果 DMAEN=1，则 FTF 置 1 时启动数据传送。如果阈值条件不再为“真”（CPU 或 DMA 传输了足够的数据后），则 FTF 由 HW 清零。

在间接读模式下，当 FIFO 已满，QSPI 将暂时停止从 Flash 读取字节以避免上溢。请注意，只有在 FIFO 中的 4 个字节为空（FLEVEL≤11）时才会重新开始读取 Flash。因此，若 FTHRES≥13，应用程序必须读取足够的字节以确保 QSPI 再次从 Flash 检索数据。否则，只要 11<FLEVEL<FTHRES，FTF 标志将保持为“0”。

27.2.5 QSPI 状态标志轮询模式

在自动轮询模式下，QSPI 周期性启动命令以读取一定数量的状态字节（最多 4 个）。可屏蔽接收的字节以隔离一些状态位，从而在所选的位具有定义的值时可产生中断。

对 Flash 的访问最初与在间接读取模式下相同：如果不需要地址（AMODE=00），则在写入 QSPI_CCR 时即开始访问。反之，如果需要地址，则在写入 QSPI_AR 时开始第一次访问。BUSY 在此时变为高电平，并且在周期性访问期间保持不变。

在自动轮询模式下，MASK[31:0]（QSPI_PSMAR）用于屏蔽来自 Flash 的数据。如果 MASK[n]=0，则屏蔽结果的位 n，从而不考虑该位。如果 MASK[n]=1 并且位[n]的内容与 MATCH[n]（QSPI_PSMAR）相同，说明存在位 n 匹配。

如果轮询匹配模式位（QSPI_CR.PMM）为 0，将激活“AND”匹配模式。这意味着状态匹配标志（SMF）仅在全部未屏蔽位均存在匹配时置 1。

如果 PMM=1，则激活“OR”匹配模式。这意味着 SMF 在任意未屏蔽位存在匹配时置 1。如果 SMIE=1，则在 SMF 置 1 时调用一个中断。

如果自动轮询模式停止（APMS）位置 1，则操作停止并且 BUSY 位在检测到匹配时清零。否则，BUSY 位保持为“1”，在发生中止或禁止 QSPI（EN=0）前继续进行周期性访问。

数据寄存器（QSPI_DR）包含最新接收的状态字节（FIFO 停用）。数据寄存器的内容不受匹配逻辑所用屏蔽方法的影响。FTF 状态位在新一次状态读取完成后置 1，并且 FTF 在数据读取后清零。

27.2.6 QSPI 内存映射模式

在配置为内存映射模式时，外部 SPI 器件被视为是内部存储器。QSPI 接口能够管理从 0x3000 0000 到 0x3FFF FFFF 的最大 256M 内存。

QSPI 外设若没有正确配置并使能，禁止访问 QSPI Flash 的存储区域。即便 Flash 容量再大，寻址空间也无法超过 256MB。

如果访问的地址超出 FSIZE 定义的范围但仍在 256MB 范围内，则生成 AHB 错误。此错误的具体影响取决于尝试进行访问的 AHB 主设备：

- 如果是 CPU 访问超出范围，总线会返回总线异常，此时进入硬件异常（Hard fault）中断。
- 如果是 DMA 访问超出范围，则生成 DMA 传输错误，并自动禁用相应的 DMA 通道。

支持字节、半字和字访问类型。

支持芯片内执行（XIP）操作，QSPI 接受下一个微控制器访问并提前加载后面地址中的字节。如果之后访问的是连续地址，由于值已经预取，访问将更快完成。

默认情况下，即便在很长时间内不访问 Flash，QSPI 也不会停止预取操作，之前的读取操作将保持激活状态并且 nCS 保持低电平。由于 nCS 保持低电平时，Flash 功耗增加，应用程序可能会激活超时计数器（TCEN=1,QSPI_CR[3]）。从而在 FIFO 中写满预取的数据后，如果 TIMEOUT[15:0]（QSPI_LPTR）个周期的时长内没有访问，则会释放 nCS。

BUSY 在第一个存储器映射访问发生时变为高电平。由于进行预取操作，BUSY 在发生超时、中止或外设禁止前不会下降。

27.2.7 QSPI Flash 配置

器件配置寄存器（QSPI_DCR）可用于指定外部 SPI Flash 的特性。

FSIZE[4:0]字段使用下面的公式定义外部存储器的大小：

Flash 中的字节数=$2^{[FSIZE+1]}$

FSIZE+1 是对 Flash 寻址所需的地址位数。在间接模式下，Flash 容量最高可达 4GB（使用 32 位进行寻址），但在内存映射模式下的可寻址空间限制为 256MB。

如果 DFM=1，FSIZE 表示两个 Flash 容量的总和。

QSPI 连续执行两条命令时，它在两条命令之间将片选信号（nCS）置为高电平默认仅一个 CLK 周期时长。如果 Flash 需要命令之间的时间更长，可使用片选高电平时间（CSHT）字段指定 nCS 必须保持高电平的最少 CLK 周期数（最大为 8）。

时钟模式（CKMODE）位指示命令之间的 CLK 信号逻辑电平（nCS=1 时）。

27.2.8 QSPI 延迟数据采样

默认情况下，QSPI 在 Flash 驱动信号后过半个 CLK 周期才对 Flash 驱动的数据采样。在外部信号延迟时，推迟采样时间是有利的。使用 SSHIFT 位（QSPI_CR[4]），可将数据采样移位半个 CLK 周期。

DDR 模式下不支持时钟移位：若 DDRM 位置 1，SSHIFT 位必须清零。

27.2.9 QSPI 配置

QSPI 配置分两个阶段

- QSPI IP 配置
- QSPI Flash 配置

QSPI 在配置完毕并使能后，即可在间接模式、状态轮询模式和内存映射模式这三种操作模式之一下工作。

QSPI IP 配置

通过 QSPI_CR 配置 QSPI IP。用户应配置传入数据的时钟预分频器的分频系数以及采样移位设置。

DDR 模式可通过 DDRM 位进行设置。使能该模式后，在每个时钟的上升沿和下降沿都会发送地址和交替字节，以及发送/接收数据。无论 DDRM 位如何设置，都始终在 SDR 模式下发送指令。

DMA 请求通过 DMAEN 位置 1 使能。若是用于中断，则相关使能位也可在该阶段置 1。生成 DMA 请求或生成中断的 FIFO 深度在 FTHRES 位中进行编程。如果需要超时计数器，则可将 TCEN 位置 1 并在 QSPI_LPTR 寄存器中编程超时值。双闪存模式可通过将 DFM 置 1 来激活。

QSPI Flash 配置

与外部目标 Flash 相关的参数通过 QSPI_DCR 寄存器进行配置。用户应在 FSIZE 位中编程 Flash 的大小、在 CSHT 位中编程片选保持高电平的最短时间以及在 MODE 位中编程功能模式（模式 0 或模式 3）。

27.2.10 QSPI 的用法

使用 FMODE[1:0]（QSPI_CCR[27:26]）选择操作模式。

间接模式的操作步骤

FMODE 编程为 00 将选择间接写入模式，将数据发送到 Flash。FMODE 编程为 01 可选择间接读取模式，读取 Flash 中的数据。

QSPI 用于间接模式时，采用以下方式构建帧：

1. 在 QSPI_DLR 中指定待读取或写入的字节数。
2. 在 QSPI_CCR 中指定帧格式、模式和指令代码。
3. 在 QSPI_ABR 中指定要在地址阶段后立即发送的可选交替字节。
4. 在 QSPI_CR 中指定工作模式。若 FMODE=00（间接写入模式）并且 DMAEN=1，则应在 QSPI_CR 前指定 QSPI_AR。否则在 QSPI_AR 更新前（如果已经使能 DMA 控制器），DMA 便可能写入 QSPI_DR。
5. 在 QSPI_AR 中指定目标地址。
6. 通过 QSPI_DR 从 FIFO 读取数据/向 FIFO 写入数据。

在写入控制寄存器（QSPI_CR）时，用户可指定以下设置：

- 使能位（EN）设置为”1”
- DMA 使能位（DMAEN），用于向/从 RAM 传输/接收数据。
- 超时计数器使能位（TCEN）
- 采样移位设置（SSHIFT）
- FIFO 阈值（FTRHES），以指示 FTF 标志在何时置 1。
- 中断使能
- 自动轮询模式参数：匹配模式和停止模式（在 FMODE=11 时有效）。
- 时钟预分频器

在写入通信配置寄存器（QSPI_CCR）时，用户指定以下参数：

- 通过 INSTRUCTION 位指定指令字节
- 通过 IMODE 位指定指令发送方式（无/1/2/4 线）
- 通过 ADMODE 位指定地址发送方式（无/1/2/4 线）
- 通过 ADSIZE 位指定地址长度（8/16/24/32 位）
- 通过 ABMODE 位指定交替字节发送方式（无/1/2/4 线）
- 通过 ABSIZE 位指定交替字节数（1/2/3/4）
- 通过 DMODE 位指定是否存在空指令字节

- 通过 DCYC 位指定空指令字节数
- 通过 DMODE 位指定数据发送/接收方式（无/1/2/4 线）

如果无需为某个命令更新地址寄存器（QSPI_AR）与数据寄存器（QSPI_DR），则在写入 QSPI_CCR 时，该命令序列便立即启动。在 ADMODE 和 DMODE 均为 00 时，或在间接读取模式（FMODE=01）下仅 ADMODE=00 时，便属于此情况。

在需要地址（ADMODE 不为 00），但无需写入数据寄存器（FMODE=01 或 DMODE=00）时，通过写入 QSPI_AR 更新地址后，命令序列便立即启动。

在数据传输（FMODE=00 并且 DMODE!=00）中，通过 QSPI_DR 写入 FIFO 触发通信启动。

状态标志轮询模式

将 FMODE 字段（QSPI_CCR）设置为 10，使能状态标志轮询模式在此模式下，将发送编程的帧并周期性检索数据。

每帧中读取的最大数据量为 4 字节。如果 QSPI_DLR 请求更多的数据，则将忽略多余的部分仅读取 4 个字节。

在 QSPI_PIR 寄存器中指定周期性。在检索到状态数据后，可在内部进行处理，以达到以下目的：

- 将状态匹配标志位置 1，如果使能了中断，还将产生中断
- 自动停止周期性检索状态字节

接收到的值可通过存储于 QSPI_PSMKR 中的值进行屏蔽，并与存储在 QSPI_PSMAR 中的值进行或运算或与运算。

若是存在匹配，则状态匹配标志置 1，并且在使能了中断的情况下还将产生中断；如果 APMS 位置 1，则 QSPI 自动停止。

在任何情况下，最新的检索值都在 QSPI_DR 中可用。

内存映射模式

在内存映射模式下，外部 Flash 被视为内部存储器，只是存在访问延迟。在该模式下，仅允许对外部 Flash 执行读取操作。

将 QSPI_CCR 寄存器中的 FMODE 设置为 11 可进入内存映射模式。当 AHB 主器件访问存储器映射空间时，将发送已编程的指令和帧。

FIFO 用作预取缓冲区以接受线性读取。在此模式中，对于 QSPI_DR 的任何访问均返回零。

数据长度寄存器（QSPI_DLR）在内存映射模式中无意义。

27.2.11 指令仅发送一次

一些 Flash（例如 Winbound）能够提供一种模式，指令在该模式中仅通过第一个命令序列进行发送，后续的命令根据地址直接启动。用户通过使用 SIOO 位（QSPI_CCR）可利用此功能的优势。

SIOO 对于所有功能模式（间接模式、状态轮询模式和内存映射模式）均有效。如果 SIOO 位置 1，仅第一条命令发送指令，接着对 QSPI_CCR 执行写入操作。后续命令序列都将跳过指令阶段，直到 QSPI_CCR 被写入为止。

在 IMODE=00（无指令）时，SIOO 不起作用。

27.2.12 QSPI 差错管理

在以下情况下可能产生错误：

- 在间接模式或状态标志轮询模式下，如果在 QSPI_AR 中编程了错误的地址（根据 QSPI_DCR 中

FSIZE[4:0]定义的 Flash 大小）：TEF 将置 1，如果使能了中断，还将产生中断。

- 另外，在间接模式下，如果地址加数据的长度超过 Flash 的大小，TEF 将在访问被触发时置 1。
- 在内存映射模式下，当 AHB 主器件执行的访问超出范围或 QSPI 被禁止时：将产生 AHB 错误，以响应故障 AHB 请求。
- 当 AHB 主机访问存储器映射空间，但内存映射模式被禁止时：将产生 AHB 错误，以响应故障 AHB 请求。

27.2.13 QSPI 的繁忙位和中止功能

在 QSPI 启动对 Flash 的操作时，QSPI_SR 中的 BUSY 位自动置 1。在间接模式下，在 QSPI 完成了请求的命令序列并且 FIFO 为空时，BUSY 位复位。

在自动轮询模式下，仅当最后一次周期性访问完成时（因 APMS=1 时发生匹配，或因中止），BUSY 位才变为低电平。

在内存映射模式下进行第一次访问后，仅在发生超时事件或中止时 BUSY 位变为低电平。任何操作都可通过将 QSPI_CR 中的 ABORT 位置 1 来中止。在完成中止时，BUSY 位和 ABORT 位自动复位，FIFO 清空。

注意：如果中止对状态寄存器的写入操作，有些 Flash 可能发生错误行为。

27.2.14 nCS 行为

默认情况下，nCS 为高电平，取消选择外部 Flash。nCS 在操作开始前下降，在操作完成时立即上升。

当 CKMODE=0（“模式 0”，在未进行任何操作时 CLK 保持低电平）时，nCS 执行在首次 CLK 上升沿之前，保持一个 CLK 周期的低电平；在执行最后一个 CLK 上升沿之后，nCS 保持一个 CLK 周期的低电平，如下图所示。

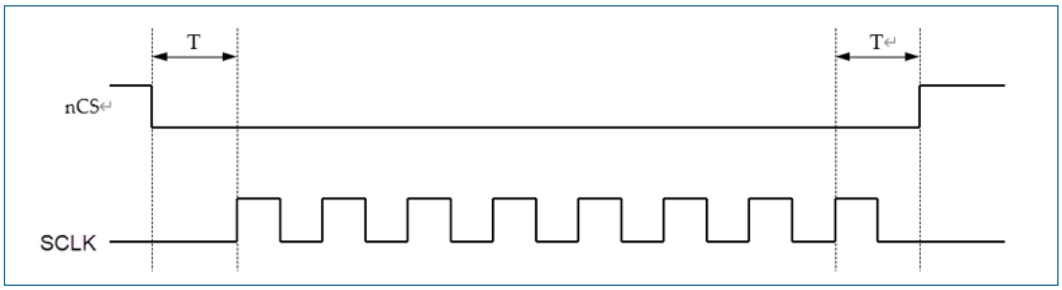


图 27-5 CKMODE=0 时的 nCS（T=CLK 周期）

当 CKMODE=1（“模式 3”，在未进行任何操作时 CLK 升至高电平）且 DDRM=0（SDR 模式）时，nCS 在执行首次 CLK 上升沿之前，保持一个 CLK 周期的低电平，在操作最后一次升高 CLK 边沿时的一个 CLK 周期后升至高电平，如下图所示。

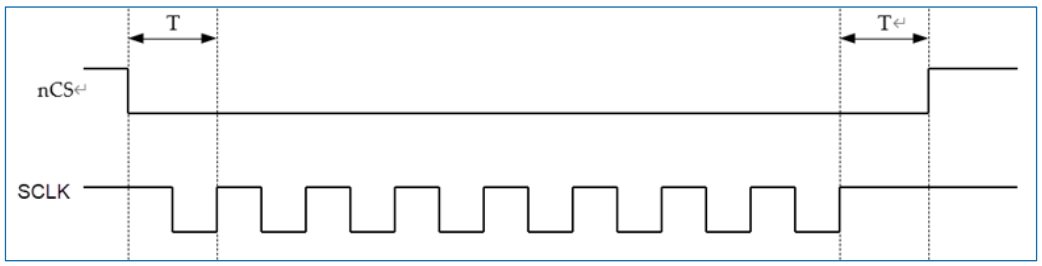


图 27-6 SDR 模式下 CKMODE=1 时的 nCS（T=CLK 周期）

当 CKMODE=1（模式 3）且 DDRM=1（DDR 模式）时，nCS 在执行首次 CLK 上升沿之前，保持一个 CLK 周期的低电平；在执行最后一个 CLK 上升沿之后，nCS 保持一个 CLK 周期的高电平，如下图所示。由于 DDR 操作必须伴随下降沿完成，当 nCS 变为高电平时，CLK 为低电平并在经过半个周期后恢复高电

平。

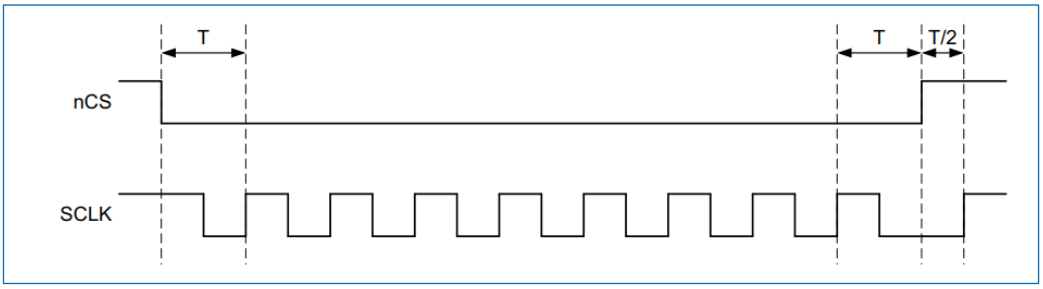


图 27-7 DDR 模式下 CKMODE=1 时的 nCS（T=CLK 周期）

如果在读取操作中 FIFO 一直满，或者在写入操作中 FIFO 一直空、操作会停止，CLK 保持低电平，直到固件为 FIFO 提供服务。如果在操作停止时发生中止，nCS 会在请求中止后上升，CLK 会在半个 CLK 周期后上升，如图 27-8 所示。

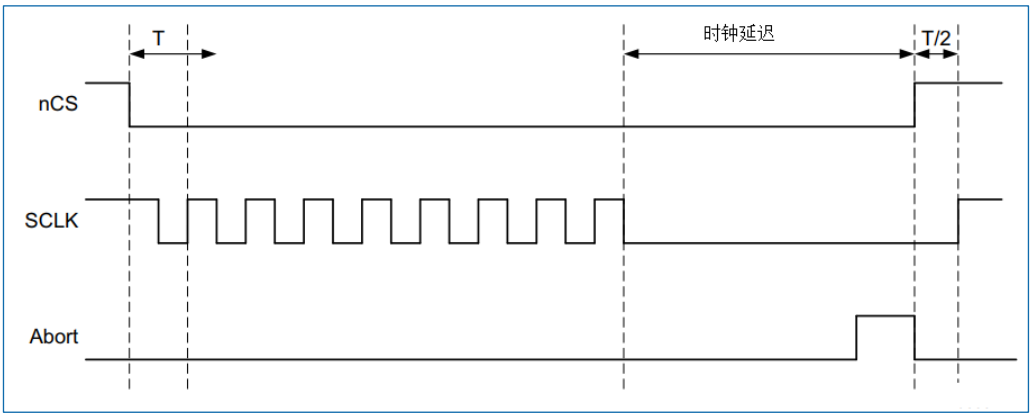


图 27-8 CKMODE=1 且发生中止时的 nCS（T=CLK 周期）

不处于双闪存模式（DFM=0）时，仅访问 FLASH1，因此 BK2_nCS 保持高电平。在双闪存模式下，BK2_nCS 与 BK1_nCS 的行为完全相同。因此，如果存在 FLASH2 并且应用程序始终处于双闪存模式，则 FLASH2 可使用 BK1_nCS，而 BK2_nCS 引脚输出可用于其他功能。

27.3 QSPI 中断

发生如下事件时可生成中断：

- 超时
- 状态匹配
- FIFO 阈值
- 传输完成
- 传输错误

可以使用单独的中断使能位以提高灵活性。

表 27-1 QSPI 中断请求

中断事件	事件标志	使能控制位
超时	TOF	TOIE
状态匹配	SMF	SMIE
FIFO 阈值	FTF	FTIE
传输完成	TCF	TCIE

中断事件	事件标志	使能控制位
传输错误	TEF	TEIE

27.4 QSPI 寄存器

基地址：0xA000 1000

空间大小：0x400

27.4.1 QSPI 控制寄存器（QSPI_CR）

偏移地址：0x0000

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRESCALER[7:0]								PMM	APMS	Res	TOIE	SMIE	FTIE	TCIE	TEIE
rw								rw	rw		rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res				FTHRES[3:0]				FSEL	DFM	Res	SSHIFT	TCEN	DMAEN	ABORT	EN
				rw				rw	rw		rw	rw	rw	rw	rw

位 31:24	PRESCALER[7:0]：时钟预分频器（Clock prescaler） 该字段定义基于 AHB 时钟生成 CLK 所用的分频系数（值+1）。 • 0：FCLK=FAHB，AHB 直接用作 QSPICLK（预分频器被旁路） • 1：FCLK=FAHB/2 • 2：FCLK=FAHB/3 • ... • 255：FCLK=FAHB/256 对于奇数时钟分频系数，CLK 的占空比并非 50%。时钟信号的高电平持续时间比低电平持续时间多一个周期。 仅可在 BUSY=0 时修改该字段。
位 23	PMM：轮询匹配模式（Polling match mode） 该位指示在自动轮询模式期间用来确定是否“匹配”的方法。 • 0：AND 匹配模式。如果从 Flash 接收的所有未屏蔽位均与匹配寄存器中的对应位相匹配，则 SMF 置 1。 • 1：OR 匹配模式。如果从 Flash 接收的任意一个未屏蔽位与匹配寄存器中的对应位相匹配，则 SMF 置 1。 仅可在 BUSY=0 时修改该位
位 22	APMS：自动轮询模式停止（Automatic poll mode stop） 该位确定在匹配后自动轮询是否停止。 • 0：仅通过中止或禁用 QSPI 停止自动轮询模式。 • 1：发生匹配时，自动轮询模式停止。 仅可在 BUSY=0 时修改该位。
位 21	Res：保留 必须保持复位值。
位 20	TOIE：超时中断使能（Time Out interrupt enable）

	该位使能超时中断。 • 0：禁止中断 • 1：使能中断
位 19	SMIE：状态匹配中断使能（Status match interrupt enable） 该位使能状态匹配中断。 • 0：禁止中断 • 1：使能中断
位 18	FTIE：FIFO 阈值中断使能（FIFO threshold interrupt enable） 该位使能 FIFO 阈值中断。 • 0：禁止中断 • 1：使能中断
位 17	TCIE：传输完成中断使能（Transfer complete interrupt enable） 该位使能传输完成中断。 • 0：禁止中断 • 1：使能中断
位 16	TEIE：传输错误中断使能（Transfer error interrupt enable） 该位使能传输错误中断。 • 0：禁止中断 • 1：使能中断
位 15:12	Res：保留 必须保持复位值。

位 11:8	FTHRES[3:0]：FIFO 阈值级别（FIFO threshold level）
	定义在间接模式下 FIFO 中将导致 FIFO 阈值标志（FTF，QSPI_SR[2]）置 1 的字节数阈值。
	• 在间接写入模式下（FMODE=00）：
	  ○ 0：如果 FIFO 中存在 1 个或更多空闲字节可供写入，则 FTF 置 1。
	  ○ 1：如果 FIFO 中存在 2 个或更多空闲字节可供写入，则 FTF 置 1。
	  ○ ...
	  ○ 15：如果 FIFO 中存在 16 个空闲字节可供写入，则 FTF 置 1。
	• 在间接读取模式下（FMODE=01）：
	  ○ 0：如果 FIFO 中存在 1 个或更多有效字节可供读取，则 FTF 置 1。
	  ○ 1：如果 FIFO 中存在 2 个或更多有效字节可供读取，则 FTF 置 1。
	  ○ ...
	  ○ 15：如果 FIFO 中存在 16 个有效字节可供读取，则 FTF 置 1。
	如果 DMAEN=1，则在更改 FTHRES 值前，必须禁止相应通道的 DMA 控制器。

位 7	FSEL：Flash 选择（Flash memory selection） 该位选择单闪存模式（DFM=0）下要寻址的 Flash。 • 0：选择 FLASH1 • 1：选择 FLASH2 仅可在 BUSY=0 时修改该位。在 DFM=1 时忽略该位。

位 6	DFM：双闪存模式（Dual-flash mode） 该位激活双闪存模式，同时使用两个外部 Flash 以将吞吐量和容量扩大一倍。 • 0：禁止双闪存模式 • 1：使能双闪存模式 仅可在 BUSY=0 时修改该位。
位 5	Res：保留 必须保持复位值。
位 4	SSHIFT：采样移位（Sample shift） 默认情况下，QSPI 在 Flash 驱动数据后过半个 CLK 周期开始采集数据。使用该位，可考虑外部信号延迟，推迟数据采集。 • 0：不发生移位 • 1：移位半个周期 在 DDR 模式下（DDRM=1），固件必须确保 SSHIFT=0。仅可在 BUSY=0 时修改该字段。
位 3	TCEN：超时计数器使能（Timeout counter enable） 该位仅在内存映射模式（FMODE=11）下有效。激活该位后，如果在一段时间（通过 TIMEOUT[15:0]（QSPI_LPTR）定义）内一直没有进行访问，将释放片选（nCS）（从而降低功耗）。使能超时计数器。 默认情况下，即便在很长时间内不访问 Flash，QSPI 也不会停止预取操作，之前的读取操作将保持激活状态并且 nCS 保持低电平。由于 nCS 保持低电平时，Flash 功耗增加，应用程序可能会激活超时计数器（TCEN=1,QSPI_CR[3]）。从而在 FIFO 中写满预取的数据后，若在 TIMEOUT[15:0]（QSPI_LPTR）个周期的时长内没有访问，则释放 nCS。 • 0：禁止超时计数器，在内存映射模式中进行访问后，片选（nCS）保持激活。 • 1：使能超时计数器，在内存映射模式下，Flash 持续不活动 TIMEOUT[15:0]个周期后释放片选（nCS）。 仅可在 BUSY=0 时修改该位。
位 2	DMAEN：DMA 使能（DMA enable） 在间接模式下，通过 QSPI_DR 寄存器可使用 DMA 输入或输出数据。FIFO 阈值标志 FTF 置 1 时，将触发 DMA 传输。 • 0：间接模式下禁止 DMA • 1：间接模式下使能 DMA
位 1	ABORT：中止请求（Abort request） 该位中止执行中的命令序列。在中止完成时自动复位。 该位可停止当前的传输。 在轮询模式或内存映射模式下，该位也用以复位 APM 位或 DM 位。 • 0：不请求中止 • 1：请求中止
位 0	EN：使能（Enable） 使能 QSPI。 • 0：禁止 QSPI • 1：使能 QSPI

27.4.2 QSPI 器件配置寄存器（QSPI_DCR）

偏移地址：0x0004

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res											FSIZE[4:0]				
											rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res					CSHT[2:0]			Res							CKMODE
					rw										rw

位 31:21	Res：保留 必须保持复位值。
位 20:16	FSIZE[4:0]：Flash 大小（Flash memory size） 该字段使用下面的公式定义了外部存储器的大小： Flash 中的字节数=2[FSIZE+1] FSIZE+1 是对 Flash 寻址所需的地址位数。 在间接模式下，Flash 容量最高可达 4GB（使用 32 位进行寻址），但在内存映射模式下的可寻址空间限制为 256MB。 如果 DFM=1，FSIZE 表示两个 Flash 容量的总和。 仅可在 BUSY=0 时修改该字段。
位 15:11	Res：保留 必须保持复位值。
位 10:8	CSHT[2:0]：片选高电平时间（Chip select high time） CSHT+1 定义片选（nCS）在发送至 Flash 的命令之间必须保持高电平的最少 CLK 周期数。 • 0：nCS 在 Flash 命令之间保持高电平至少 1 个周期 • 1：nCS 在 Flash 命令之间保持高电平至少 2 个周期 • ... • 7：nCS 在 Flash 命令之间保持高电平至少 8 个周期 仅可在 BUSY=0 时修改该字段。
位 7:1	Res：保留 必须保持复位值。
位 0	CKMODE：模式 0/模式 3（Mode0/mode3） 该位指示 CLK 在命令之间（nCS=1 时）的电平。 • 0：nCS 为高电平（片选释放）时，CLK 必须保持低电平。这称为模式 0。 • 1：nCS 为高电平（片选释放）时，CLK 必须保持高电平。这称为模式 3。 仅可在 BUSY=0 时修改该字段。

27.4.3 QSPI 状态寄存器（QSPI_SR）

偏移地址：0x0008

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res			FLEVEL[4:0]					Res		BUSY	TOF	SMF	FTF	TCF	TEF
			r							r	r	r	r	r	r

位 31:13	Res：保留 必须保持复位值。
位 12:8	FLEVEL[4:0]：FIFO 级别（FIFO level） 该字段给出 FIFO 中的有效字节数。FIFO 为空时 FLEVEL=0，写满时 FLEVEL=16。在内存映射模式和自动状态轮询模式下，FLEVEL 为零。
位 7:6	Res：保留 必须保持复位值。
位 5	BUSY：忙（Busy） 操作进行时，该位置 1。在对 Flash 的操作完成并且 FIFO 为空时，该位自动清零。
位 4	TOF：超时标志（Time out flag） 发生超时时该位置 1。向 CTOF 写入 1 可将该位清零。
位 3	SMF：状态匹配标志（Status match flag） 在自动轮询模式下，若未屏蔽的接收数据与匹配寄存器（QSPI_PSMAR）中的对应 位相匹配，则该位置 1。向 CSMF 写入 1 可将该位清零。
位 2	FTF：FIFO 阈值标志（FIFO threshold flag） 在间接模式下，若达到 FIFO 阈值，或从 Flash 读取完成后，FIFO 中留有数据时，该位置 1。只要阈值条件不再为“真”，该位就自动清零。在自动轮询模式下，每次读取状态寄存器时，该位即置 1；读取数据寄存器时，该位清零。
位 1	TCF：传输完成标志（Transfer complete flag） 在间接模式下，当传输的数据数量达到编程设定值，或在任何模式下传输中止时，该位置 1。向 CTCF 写入 1 时，该位清零。
位 0	TEF：传输错误标志（Transfer error flag） 在间接模式下访问无效地址时，该位置 1。向 CTEF 写入 1 可将该位清零。

27.4.4 QSPI 标志清零寄存器（QSPI_FCR）

偏移地址：0x000C

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res											CTOF	CSMF	Res	CTCF	CTEF
											w	w		w	w

位 31:5	Res：保留 必须保持复位值。

位 4	CTOF：清除超时标志（Clear timeout flag） 写入 1 可将 QSPI_SR 寄存器中的 TOF 标志清零位。
位 3	CSMF：清除状态匹配标志（Clear status match flag） 写入 1 可将 QSPI_SR 寄存器中的 SMF 标志清零。
位 2	Res：保留 必须保持复位值。
位 1	CTCF：清除传输完成标志（Clear transfer complete flag） 写入 1 可将 QSPI_SR 寄存器中的 TCF 标志清零。
位 0	CTEF：清除传输错误标志（Clear transfer error flag） 写入 1 可将 QSPI_SR 寄存器中的 TEF 标志清零。

27.4.5 QSPI 数据长度寄存器（QSPI_DLR）

偏移地址：0x0010

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DL[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DL[15:0]															
rw															

位 31:0	DL[31:0]：数据长度（Data length） 在间接模式和状态轮询模式下待检索的数据数量（值+1）。对状态轮询模式应使用不大于 3 的值（表示 4 字节）。 在间接模式下，所有位置 1 表示未定义长度，QSPI 将继续传输数据直到到达由 FSIZE 定义的存储器末尾。 • 0x0000_0000：传输 1 个字节 • 0x0000_0001：传输 2 个字节 • 0x0000_0002：传输 3 个字节 • 0x0000_0003：传输 4 个字节 • ... • 0xFFFF_FFFD：传输 4,294,967,294（4G-2）个字节 • 0xFFFF_FFFE：传输 4,294,967,295（4G-1）个字节 • 0xFFFF_FFFF：未定义长度--传输所有字节直到到达由 FSIZE 定义的 Flash 的结尾。如果 FSIZE=0x1F，则读取无限继续下去。 在双闪存模式（DFM=1）下，即使该位写入“0”，DL[0]也始终保持为“1”，因此保证了每次访问均传输偶数个字节。 该字段在内存映射模式（FMODE=10）下不起作用。 仅可在 BUSY=0 时写入该字段。

27.4.6 QSPI 通信配置寄存器（QSPI_CCR）

偏移地址：0x00014

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DDRM	DHHC	Res	SIOO	FMODE[1:0]		DMODE[1:0]		Res	DCYC[4:0]					ABSIZE[1:0]	
rw	rw		rw	rw		rw			rw					rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABMODE[1:0]		ADSIZE[1:0]		ADMODE[1:0]		IMODE[1:0]		INSTRUCTION[7:0]							
rw		rw		rw		rw		rw							

位 31	DDRM：双倍数据速率模式（Double data rate mode） 该位为地址、交替字节和数据阶段设置 DDR 模式： • 0：禁止 DDR 模式 • 1：使能 DDR 模式 仅可在 BUSY=0 时写入该字段。
位 30	DHHC：DDR 保持（DDR hold） DDR 模式下数据输出延迟 1/4 个 QSPI 输出时钟周期： • 0：使用模拟延迟来延迟数据输出 • 1：数据输出延迟 1/4 个 QSPI 输出时钟周期。 该特性仅在 DDR 模式下激活。 仅可在 BUSY=0 时写入该字段。
位 29	Res：保留 必须保持复位值。
位 28	SIOO：仅发送指令一次模式（Send instruction only once mode） 参见章节“27.2.11 指令仅发送一次”，IMODE=00 时，该位不起作用。 • 0：在每个事务中发送指令 • 1：仅为第一条命令发送指令 仅可在 BUSY=0 时写入该字段。
位 27:26	FMODE[1:0]：功能模式（Functional mode） 该字段定义 QSPI 操作的功能模式： • 00：间接写入模式 • 01：间接读取模式 • 10：自动轮询模式 • 11：内存映射模式 如果 DMAEN=1，则在更改 FMODE 值前，必须禁止相应通道的 DMA 控制器。 仅可在 BUSY=0 时写入该字段。
位 25:24	DMODE[1:0]：数据模式（Data mode） 该字段定义数据阶段的操作模式： • 00：无数据 • 01：单线传输数据 • 10：双线传输数据 • 11：四线传输数据 该字段还定义空指令阶段的操作模式。仅可在 BUSY=0 时写入该字段。

位 23	Res：保留 必须保持复位值。
位 22:18	DCYC[4:0]：空指令周期数（Number of dummy cycles） 该字段定义空指令阶段的持续时间。在 SDR 和 DDR 模式下，它指定 CLK 周期数（0-31）。 仅可在 BUSY=0 时写入该字段。
位 17:16	ABSIZE[1:0]：交替字节长度（Alternate bytes size） 该位定义交替字节长度： • 00：8 位交替字节 • 01：16 位交替字节 • 10：24 位交替字节 • 11：32 位交替字节 仅可在 BUSY=0 时写入该字段。
位 15:14	ABMODE[1:0]：交替字节模式（Alternate bytes mode） 该字段定义交替字节阶段的操作模式： • 00：无交替字节 • 01：单线传输交替字节 • 10：双线传输交替字节 • 11：四线传输交替字节 仅可在 BUSY=0 时写入该字段。
位 13:12	ADSIZE[1:0]：地址长度（Address size） 该位定义地址长度： • 00：8 位地址 • 01：16 位地址 • 10：24 位地址 • 11：32 位地址 仅可在 BUSY=0 时写入该字段。
位 11:10	ADMODE[1:0]：地址模式（Address mode） 该字段定义地址阶段的操作模式： • 00：无地址 • 01：单线传输地址 • 10：双线传输地址 • 11：四线传输地址 仅可在 BUSY=0 时写入该字段。
位 9:8	IMODE[1:0]：指令模式（Instruction mode） 该字段定义指令阶段的操作模式： • 00：无指令 • 01：单线传输指令 • 10：双线传输指令 • 11：四线传输指令 仅可在 BUSY=0 时写入该字段。

位 7:0	INSTRUCTION[7:0]：指令（Instruction） 指定要发送到外部 SPI 设备的指令。仅可在 BUSY=0 时写入该字段。

27.4.7 QSPI 地址寄存器（QSPI_AR）

偏移地址：0x0018

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRESS[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRESS[15:0]															
rw															

位 31:0	ADDRESS[31:0]：地址（Address） 指定要发送到外部 Flash 的地址。 BUSY=0 或 FMODE=11（内存映射模式）时，将忽略写入该字段。 在双闪存模式下，由于地址始终为偶地址，ADDRESS[0]自动保持为“0”。

27.4.8 QSPI 交替字节寄存器（QSPI_ABR）

偏移地址：0x001C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALTERNATE[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ALTERNATE[15:0]															
rw															

位 31:0	ALTERNATE[31:0]：交替字节（Alternate Bytes） 指定要在地址后立即发送到外部 SPI 设备的可选数据。 仅可在 BUSY=0 时写入该字段。

27.4.9 QSPI 数据寄存器（QSPI_DR）

偏移地址：0x0020

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw															

位 31:0	DATA[31:0]：数据（Data） 指定要与外部 SPI 设备交换的数据。 在间接写入模式下，写入该寄存器的数据在数据阶段发送到 Flash，在此之前则存储于 FIFO。如果 FIFO 太满，将暂停写入，直到 FIFO 具有足够的空间接受要写入的数据才继续。 在间接读模式下，读取该寄存器可获得（通过 FIFO）已从 Flash 接收的数据。如果 FIFO 所含字节数比读取操作要求的字节数少并且 BUSY=1，将暂停读取，直到足够的数据出现或传输完成（不分先

	后）才继续。 在自动轮询模式下，该寄存器包含最后从 Flash 读取的数据（未进行屏蔽）。支持对该寄存器进行字、半字以及字节访问。在间接写入模式下，字节写入将在 FIFO 中增加 1 个字节，半字写入增加 2 个，而字写入则增加 4 个。类似地，在间接读取模式下，字节读取将擦除 FIFO 中的 1 个字节，半字读取擦除 2 个，而字读取则擦除 4 个。间接模式下的访问必须与此寄存器的最低位对齐：字节读取必须读取 DATA[7:0]而半字读取必须读取 DATA[15:0]。

27.4.10 QSPI 轮询状态屏蔽寄存器（QSPI_PSMKR）

偏移地址：0x0024

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MASK31	MASK30	MASK29	MASK28	MASK27	MASK26	MASK25	MASK24	MASK23	MASK22	MASK21	MASK20	MASK19	MASK18	MASK17	MASK16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MASK15	MASK14	MASK13	MASK12	MASK11	MASK10	MASK9	MASK8	MASK7	MASK6	MASK5	MASK4	MASK3	MASK2	MASK1	MASK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 x (x=31..0)	MASKx：状态屏蔽位 x（Status mask bit） 对在轮询模式下接收的状态字节进行屏蔽。 对于位 n： • 0：屏蔽在自动轮询模式下所接收数据的位 n，在匹配逻辑中不考虑其值。 • 1：不屏蔽在自动轮询模式下所接收数据的位 n，在匹配逻辑中考虑其值。 仅可在 BUSY=0 时写入该字段。

27.4.11 QSPI 轮询状态匹配寄存器（QSPI_PSMAR）

偏移地址：0x0028

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MATCH31	MATCH30	MATCH29	MATCH28	MATCH27	MATCH26	MATCH25	MATCH24	MATCH23	MATCH22	MATCH21	MATCH20	MATCH19	MATCH18	MATCH17	MATCH16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MATCH15	MATCH14	MATCH13	MATCH12	MATCH11	MATCH10	MATCH9	MATCH8	MATCH7	MATCH6	MATCH5	MATCH4	MATCH3	MATCH2	MATCH1	MATCH0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 x (x=31..0)	MATCHx：状态匹配位 x（Status match bit） 该值将与屏蔽状态寄存器比较以进行匹配。 仅可在 BUSY=0 时写入该字段。

27.4.12 QSPI 轮询间隔寄存器（QSPI_PIR）

偏移地址：0x002C

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTERVAL[15:0]															
rw															

位 31:16	Res：保留 必须保持复位值。
位 15:0	INTERVAL[15:0]：轮询间隔（Polling interval） 自动轮询阶段读取操作之间的 CLK 周期数。仅可在 BUSY=0 时写入该字段。

27.4.13 QSPI 低功耗超时寄存器（QSPI_LPTR）

偏移地址：0x0030

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMEOUT[15:0]															
rw															

位 31:16	Res：保留 必须保持复位值。
位 15:0	TIMEOUT[15:0]：超时时长（Timeout period） 在内存映射模式下，每次访问结束后，QSPI 将预取后续字节并将其存放在 FIFO 中。该字段指示在 FIFO 写满后，QSPI 等待多少个 CLK 时钟周期才让 nCS 升至高电平将 Flash 置为低功耗状态。 仅可在 BUSY=0 时写入该字段。

27.4.14 QSPI 加密寄存器（QSPI_SECR）

偏移地址：0x0040

复位值：0x0000 0000

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

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

位 31:2	Res：保留 必须保持复位值。
位 1	DECRYPTEN：解密使能位（Decryption enable bit） 此位指示是否启用解密。 • 0：解密失能 • 1：解密使能
位 0	ENCRYPTEN：加密使能位（Encryption enable bit） 此位指示是否启用加密。

	• 0：加密失能 • 1：加密使能

28 数字相机接口（DCMI）

数字摄像头接口是一个同步并行接口，能够接收外部 8 位、10 位、12 位或 14 位 CMOS 摄像头模块发出的高速数据流。可支持不同的数据格式：YCbCr4:2:2/RGB565 逐行视频和压缩数据（JPEG）。

此接口适用于黑白摄像头、X24 和 X5 摄像头，并假定所有预处理（如调整大小）都在摄像头模块中执行。

28.1 DCMI 主要特性

DCMI 通过其同步并行数字接口，接收片外 CMOS 相机模块发送的高速图像视频数据。DCMI 具备以下特点：

- 8 位、10 位、12 位或 14 位并行接口
- 内嵌码/外部行同步和帧同步
- 连续模式或快照模式
- 裁剪功能
- 支持以下数据格式：
  - 8/10/12/14 位逐行视频：单色或原始拜尔格式
  - YCbCr4:2:2 逐行视频
  - RGB565 逐行视频
  - 压缩数据：JPEG
- 像素时钟和同步信号的极性可配
- 支持连续帧采样或单帧采样
- 支持图像自动裁剪
- 支持跳帧、跳行和跳像素点采样
- 8-word 深度数据接收 FIFO

28.2 DCMI 引脚

下表显示了 DCMI 引脚。

表 28-1 DCMI 引脚

名称	信号类型
D[0:13]	数据输入
HSYNC	水平同步（行同步）输入
VSYNC	垂直同步（场同步）输入
PIXCLK	像素时钟输入

28.3 DCMI 时钟

数字摄像头接口使用 PIXCLK（像素时钟）和 HCLK 这两个时钟域。当 PIXCLK 产生的信号稳定后，将在 HCLK 上升沿时对这些信号进行采样。HCLK 域产生一个使能信号用于指示摄像头发出的数据已稳定，可进行采样。PIXCLK 的最大周期必须大于 2.5 个 HCLK 周期。

28.4 DCMI 功能概述

数字摄像头接口是一个同步并行接口，可接收高速（可达 54 MByte/s）数据流。该接口包含多达 14 条数据线（D13-D0）和一条像素时钟线（PIXCLK）。像素时钟的极性可以编程，因此可以在像素时钟的上升沿或下降沿捕获数据。

这些数据被放到 32 位数据寄存器（DCMI_DR）中，然后通过通用 DMA 进行传输。图像缓冲区由 DMA 管理，而不是由摄像头接口管理。

从摄像头接收的数据可以按行/帧来组织（原始 YUB/RGB/拜尔模式），也可以是一系列 JPEG 图像。要使能 JPEG 图像接收，必须将 JPEG 位（DCMI_CR 寄存器的位 3）置 1。

数据流可由可选的 HSYNC（水平同步）信号和 VSYNC（垂直同步）信号硬件同步，或者通过数据流中嵌入的同步码同步。

下图显示了 DCMI 框图。

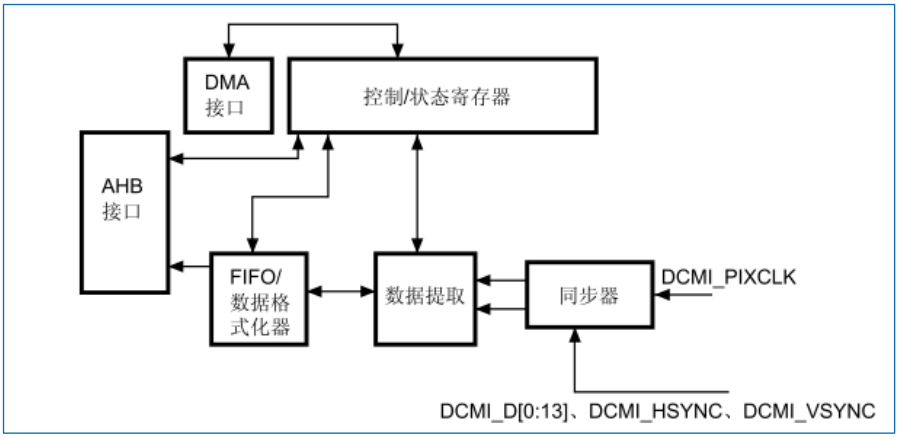


图 28-1 DCMI 框图

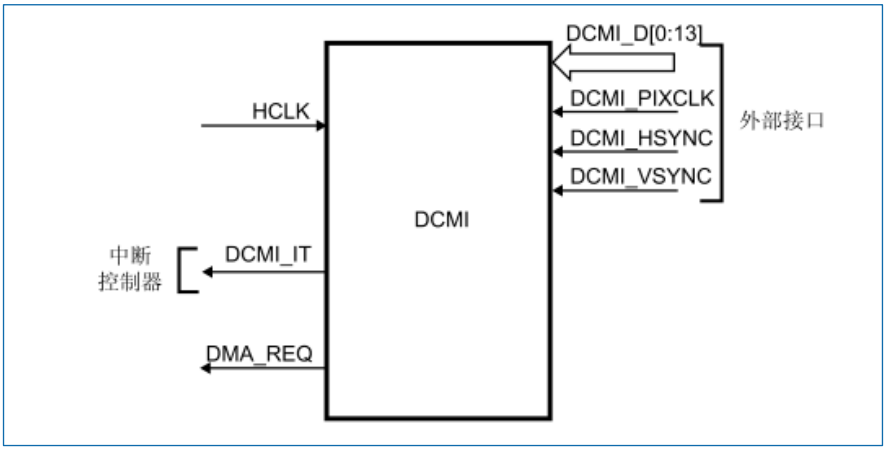


图 28-2 顶层框图

28.4.1 DMA 接口

当 DCMI_CR 寄存器中的 CAPTURE 位置 1 时，激活 DMA 接口。摄像头接口每次在其寄存器中收到一个完整的 32 位数据块时，都将触发一个 DMA 请求。

28.4.2 DCMI 物理接口

该接口由 11/13/15/17 个输入信号组成。仅支持从模式。

根据 DCMI_CR 寄存器中 EDM[1:0]位的设置，摄像头接口可以捕获 8 位、10 位、12 位或 14 位数据。如果使用的位数少于 14，则必须将未使用的输入引脚接地。

表 28-2 DCMI 信号

信号名称		信号说明
8 位 10 位 12 位 14 位	D[0..7] D[0..9] D[0..11] D[0..13]	Data
PIXCLK		像素时钟
HSYNC		水平同步/数据有效
VSYNC		垂直同步

数据与 PIXCLK 保持同步，并根据像素时钟的极性在像素时钟上升沿/下降沿发生变化。

HSYNC 信号指示行的开始/结束。

VSYNC 信号指示帧的开始/结束。

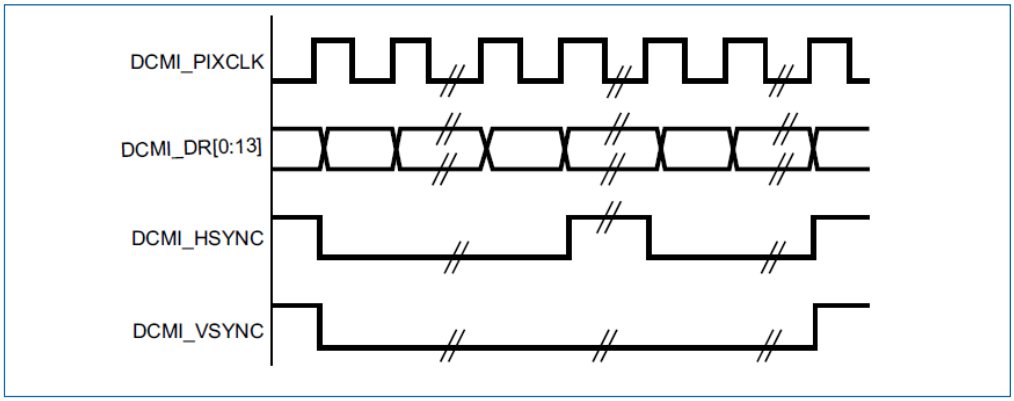


图 28-3 DCMI 信号波形

说明：

- DCMI_PIXCLK 的捕获沿为下降沿，DCMI_HSYNC 和 DCMI_VSYNC 的有效状态为 1。
- DCMI_HSYNC 和 DCMI_VSYNC 的状态可同时发生更改。

8 位数据

当 DCMI_CR 中的 EDM[1:0]编程为“00”时，接口将捕获其输入（D[0:7]）的 8 个 LSB，并将其存储为 8 位数据。D[13:8]输入则忽略。在此情况下，要捕获 32 位字，摄像头接口需要花费四个像素时钟周期。

捕获的第一个数据字节放置在 32 位字的 LSB 位置，捕获的第四个数据字节放置在 32 位字的 MSB 位置。下表列举了捕获到的数据字节在两个 32 位字中的位置排布。

表 28-3 捕获的数据字节在 32 位字（宽 8 位）中的位置排布

字节地址	31:24	23:16	15:8	7:0
0	D_{n+3}[7:0]	D_{n+2}[7:0]	D_{n+1}[7:0]	D_n[7:0]
4	D_{n+7}[7:0]	D_{n+6}[7:0]	D_{n+5}[7:0]	D_{n+4}[7:0]

10 位数据

当 DCMI_CR 中的 EDM[1:0]编程为“01”时，摄像头接口将捕获其输入 D[0..9]的 10 位数据，并将其

存储为 16 位字的 10 个最低有效位。DCMI_DR 寄存器中的其余最高有效位（位 11 到 15）将清零。因此，在此情况下，每两个像素时钟周期会生成一个 32 位数据字。

捕获的第一个数据放置在 32 位字的 LSB 位置，捕获的第二个数据放置在 32 位字的 MSB 位置，如下表中所示。

表 28-4 捕获的数据字节在 32 位字（宽 10 位）中的位置排布

字节地址	31:26	25:16	15:10	9:0
0	0	$D_{n+1[9:0]}$	0	D_n[9:0]
4	0	D_{n+3}[9:0]	0	D_{n+2}[9:0]

12 位数据

当 DCMI_CR 中的 EDM[1:0]编程为“10”时，摄像头接口将捕获其输入 D[0..11]的 12 位数据，并将其存储为 16 位字的 12 个最低有效位。其余最高有效位将清零。因此，在此情况下，每两个像素时钟周期会生成一个 32 位数据字。

捕获的第一个数据放置在 32 位字的 LSB 位置，捕获的第二个数据放置在 32 位字的 MSB 位置，如下表中所示。

表 28-5 捕获的数据字节在 32 位字（宽 12 位）中的位置排布

字节地址	31:28	27:16	15:12	11:0
0	0	D_{n+1}[11:0]	0	D_n[11:0]
4	0	D_{n+3}[11:0]	0	D_{n+2}[11:0]

14 位数据

当 DCMI_CR 中的 EDM[1:0]编程为“11”时，摄像头接口将捕获其输入 D[0..13]的 14 位数据，并将其存储为 16 位字的 14 个最低有效位。其余最高有效位将清零。因此，在此情况下，每两个像素时钟周期会生成一个 32 位数据字。

捕获的第一个数据放置在 32 位字的 LSB 位置，捕获的第二个数据放置在 32 位字的 MSB 位置，如下表中所示。

表 28-6 捕获的数据字节在 32 位字（宽 14 位）中的位置排布

字节地址	31:30	29:16	15:14	13:0
0	0	D_{n+1}[13:0]	0	D_n[13:0]
4	0	D_{n+3}[13:0]	0	D_{n+2}[13:0]

28.4.3 同步

数字摄像头接口支持内嵌码同步或硬件（HSYNC 和 VSYNC）同步。使用内嵌码同步时，由数字摄像头模块确保 0x00 和 0xFF 值仅用于同步（不用于数据中）。只有 8 位并行数据接口宽度支持内嵌码同步码（即，DCMI_CR 寄存器中的 EDM[1:0]位应为“00”）。

对于压缩数据，DCMI 仅支持硬件同步模式。这种情况下，VSYNC 指示图像的开始/结束，HSYNC 则用作“数据有效”信号。下图显示了相应的时序图。

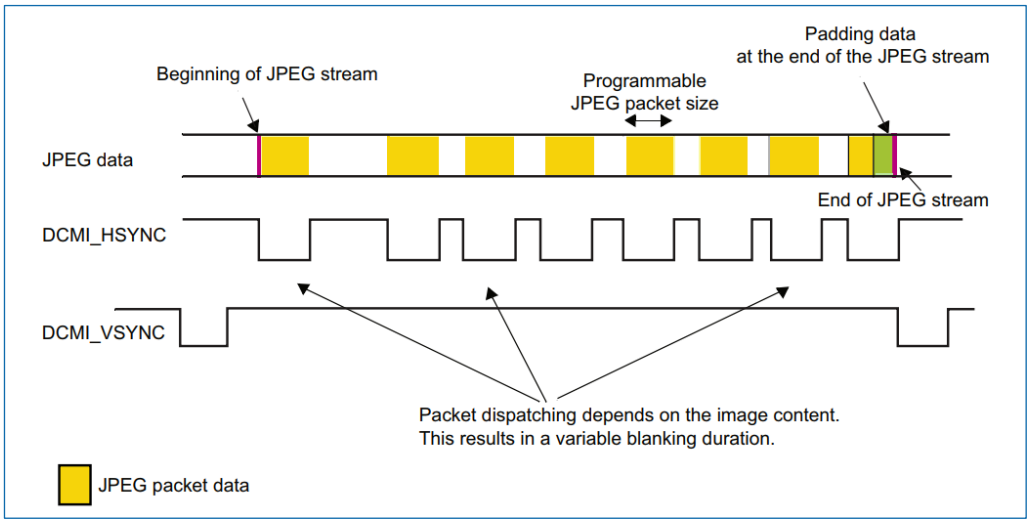


图 28-4 时序图

28.4.3.1 硬件同步模式

硬件同步模式下将使用两个同步信号（HSYNC/VSYNC）。根据摄像头模块/模式的不同，可能在水平/垂直同步期间内发送数据。由于系统会忽略 HSYNC/VSYNC 信号有效电平期间内接收的所有数据，HSYNC/VSYNC 信号相当于消隐信号。

为了正确地将图像传输到 DMA/RAM 缓冲区，数据传输将与 VSYNC 信号同步。选择硬件同步模式并启用捕获（DCMI_CR 中的 CAPTURE 位置 1）时，数据传输将与 VSYNC 信号的无效电平同步（开始下一帧时）。

之后传输便可以连续执行，由 DMA 将连续帧传输到多个连续的缓冲区或一个具有循环特性的缓冲区。为了允许 DMA 管理连续帧，每一帧结束时都将激活 VSIF（垂直同步中断标志）。

28.4.3.2 内嵌码同步模式

在此同步模式下，将使用数据流中嵌入的 32 位码来同步数据流。这些码使用数据中不再使用的值 0x00/0xFF。共有 4 种同步码类型，均采用 0xFF0000XY 格式。只有 8 位并行数据接口支持内嵌码同步（DCMI_CR 寄存器中的 EDM[1:0]位应编程为“00”）。对于其它数据宽度，此模式将造成无法预知的结果，因此不得使用。

注意：（在隔行扫描模式下）摄像头模块具有 8 个此类代码。因此，摄像头接口不支持隔行扫描模式（否则每帧数据的一半都会被丢弃）。

模式 2

四个内嵌码可表示以下事件

- 帧开始（FS）
- 帧结束（FE）
- 行开始（LS）
- 行结束（LE）

4 个同步码采用的格式 0xFF0000XY 中的 XY 值可编程（参见章节：DCMI 内嵌同步码寄存器（DCMI_ESCR））。

将 0xFF 编程为“帧结束”意味着所有除此之外的同步码都视为有效的帧结束同步码。在此模式下，一旦使能摄像头接口，将在首次出现帧结束（FE）同步码并且后接帧开始（FS）同步码之后开始捕获帧。

模式 1

摄像头模式 1 是另一种编码。此模式与 ITU656 兼容。这些同步码表示另一组事件：

- SAV（有效行）——行开始
- EAV（有效行）——行结束
- SAV（消隐）——帧间消隐期内的行开始
- EAV（消隐）——帧间消隐期内的行结束

可通过对同步码进行如下编程来支持此模式：

- FS≤0xFF
- FE≤0xFF
- LS≤SAV（有效）
- LE≤EAV（有效）

此外还针对帧/行开始和帧/行结束同步码实现了非屏蔽位功能。这样可以仅使用同步码中未被屏蔽的位进行比较。因此，可以选择一个位用于同步码的比较，来检测帧/行起始和帧/行结束。这意味着可以多个同步码表示帧/行的起始和结束，它们仅在未被屏蔽的位相同即可。

示例

FS=0xA5

FS 的非屏蔽码=0x10

这种情况下，只需要比较数据码的第 4 位来检测是否是 FS 信号。

28.4.4 捕获模式

此接口支持两种类型的捕获：快照（单帧）和连续采集。

快照模式（单帧）

此模式下只捕获单帧（DCMI_CR 寄存器中的 CM=“1”）。在 DCMI_CR 中的 CAPTURE 位置 1 后，该接口将等待系统检测帧开始，然后再对数据进行采样。收到完整的第一帧后，将自动禁止摄像头接口（DCMI_CR 中的 CAPTURE 位清零）。如果使能相应中断（FRAME_IE 位），将生成中断（IT_FRAME），中断标志 FRAME_MIS 置位。

如果发生溢出，这帧将丢失并且 CAPTURE 位清零。

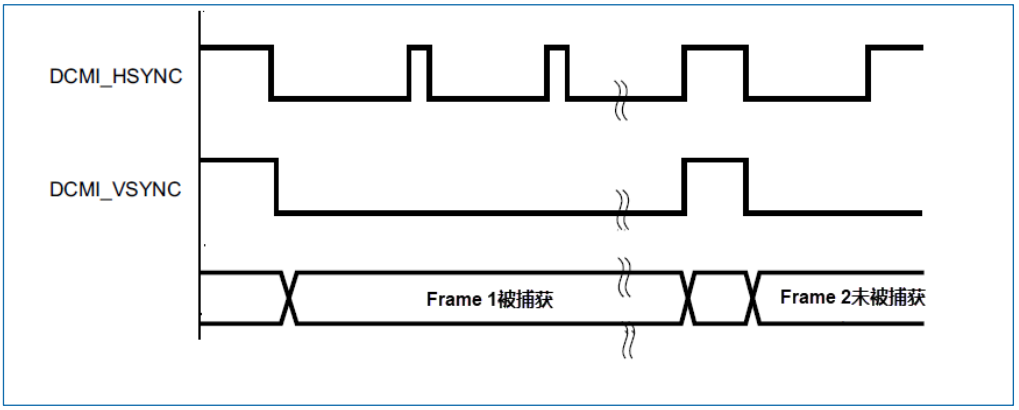


图 28-5 连续采集模式下的帧捕获波形

说明：

- 此例中，DCMI_HSYNC 和 DCMI_VSYNC 的有效状态为 1。
- DCMI_HSYNC 和 DCMI_VSYNC 的状态可同时发生更改。

连续采集模式

在连续采集模式（DCMI_CR 寄存器中的 CM=“0”）下，可以通过配置 DCMI_CR 中的 FCRC 位来选择采集所有图片，或每两帧采集一次图片，或每四帧采集一张图片，以此降低帧捕获率。

注意：在硬件同步模式下（DCMI_CR 中的 ESS=“0”），即使 DCMI_CR 中的 CAPTURE=“0”，也将生成 IT_VSYNC 中断（如果使能），中断标志位 VSYNC_MIS 置位。因此为进一步降低帧捕获率，IT_VSYNC 中断可与快照模式结合使用，用来统计 2 次捕获之间的帧数。这并不适用于内嵌码同步模式。

28.4.5 裁剪功能

摄像头接口可以使用裁剪功能从收到的图像中选择一个矩形窗口。起始（左上角）坐标和窗口大小（用像素时钟数表示的水平尺寸以及用行数表示的垂直尺寸）由两个 32 位寄存器（DCMI_CWSTRT 和 DCMI_CWSIZE）指定。窗口大小用像素时钟数（水平尺寸）和行数（垂直尺寸）表示。

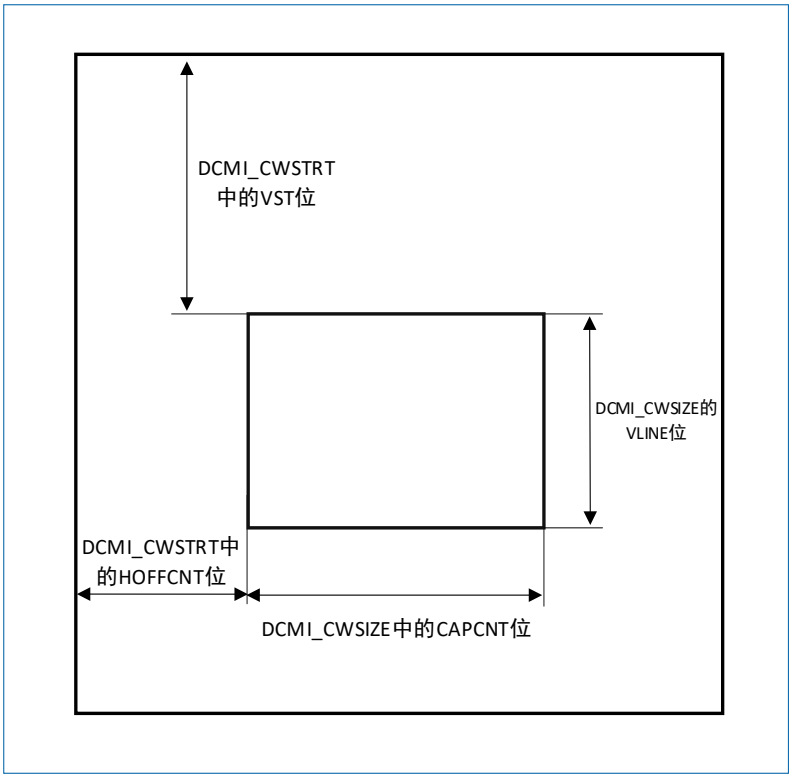


图 28-6 裁剪后窗口的坐标和大小

这些寄存器将捕获窗口的起点坐标指定为某一行号（在帧内，从 0 开始）和像素时钟数（在行内，从 0 开始），窗口大小则指定为行数和像素时钟数。CAPCNT 值只能是 4 的倍数（两个最低有效位强制为 0），才能通过 DMA 正确传输数据。

如果在捕获 DCMI_CWSIZE 寄存器中指定行数完成之前，VSYNC 信号已有效，那么捕获将停止，并且在中断使能时生成 IT_FRAME 中断。

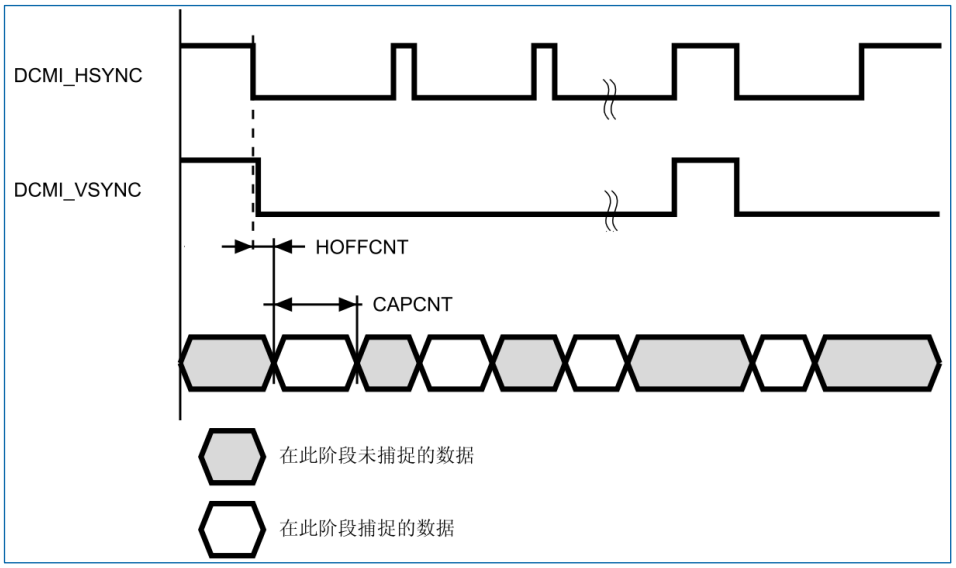


图 28-7 数据捕获波形

说明：

- 此例中，DCMI_HSYNC 和 DCMI_VSYNC 的有效状态为 1。
- DCMI_HSYNC 和 DCMI_VSYNC 的状态可同时发生更改。

28.4.6 JPEG 格式

要允许接收 JPEG 图像，必须将 DCMI_CR 寄存器中的 JPEG 位置 1。JPEG 图像不按行和帧存储，因此 VSYNC 信号用于启动捕获过程，而 HSYNC 则用作数据使能信号。行中包含的字节数可能不是 4 的倍数，因此处理此类情况时应十分谨慎，因为需要每次从捕获的数据形成一个完整的 32 位字时，才生成一个 DMA 请求。检测到帧结束但尚未凑成 32 位字待发送数据时，将使用“0”进行填充，并触发一个 DMA 请求。

裁剪功能和内嵌码同步不适用于 JPEG 格式。

28.4.7 FIFO

输入模式

为了对 AHB 上的数据传输率加以管理，在摄像头接口上实现了 4 个字深度的 FIFO。DCMI 配有一个简单的 FIFO 控制器，每次摄像头接口从 AHB 读取数据时读指针递增，每次摄像头接口向 FIFO 写入数据时写指针递增。因为没有溢出保护，如果数据传输率超过了 AHB 接口能够承受的速率，FIFO 中的数据就会被覆盖。

如果同步信号出错，或者 FIFO 发生溢出，FIFO 将复位，DCMI 接口将等待新的数据帧开始。

28.5 数据格式说明

28.5.1 数据格式

支持三种类型的数据：

- 8/10/12/14 位逐行视频：单色或原始拜尔格式
- YCbCr4:2:2 逐行视频
- RGB565 逐行视频。采用 16 位（5 位表示红色，6 位表示绿色，5 位表示蓝色）编码的像素需要两个时钟周期传输。

压缩数据：JPEG

对于 B&W、YCbCr 或 RGB 数据，最大输入大小为 2048×2048 像素。JPEG 压缩模式无限制。对于单色、RGB 和 YCbCr，帧缓冲区以光栅模式存储。使用 32 位字。仅支持小端对齐格式。

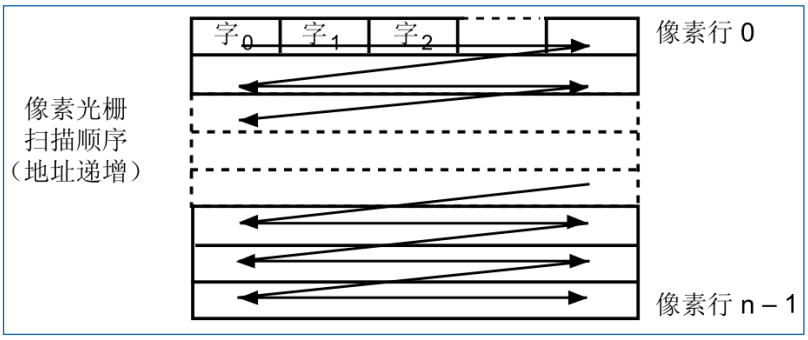


图 28-8 像素光栅扫描顺序

28.5.2 单色格式

特性：

- 光栅格式
- 每个像素 8 位

下表显示了数据的存储方式。

表 28-7 单色逐行视频格式的数据存储

字节地址	31:24	23:16	15:8	7:0
0	n+3	n+2	n+1	n
4	n+7	n+6	n+5	n+4

28.5.3 RGB 格式

特性：

- 光栅格式
- RGB
- 一个缓冲区交替存储 RGB 信号：BRGBRGBRG 等
- 对显示输出的优化

RGB 平面格式与标准的 OS 帧缓冲显示格式兼容。

仅支持 16BPP（每个像素 16 位）：RGB565（每 32 位字表示 2 个像素）。

不支持 24BPP（托盘化格式）和灰度格式。像素按照光栅扫描顺序进行存储，即从顶部像素行到底部像素行，从像素行的左侧到右侧。像素分量包括 R（红色）、G（绿色）和 B（蓝色）。所有分量的空间分辨率都相同（格式 4:4:4）。一帧数据中各个分量交替间隔存储。

下表显示了数据的存储方式。

表 28-8 以 RGB 逐行视频格式存储数据

字节地址	31:27	26:21	20:16	15:11	10:5	4:0
0	R_{n+1}	G_{n+1}	B_{n+1}	R_n	G_n	B_n
4	R_{n+4}	G_{n+3}	B_{n+3}	R_{n+2}	G_{n+2}	B_{n+2}

28.5.4 YCbCr 格式

特性：

- 光栅格式
- YCbCr4:2:2
- 一个缓冲区交替存储 Y、Cb 和 Cr：CbYCrYCbYCr 等

像素分量包括 Y（亮度）、Cb 和 Cr（蓝色色度和红色色度）。每个分量都采用 8 位进行编码。亮度和色度（交替）存储在一起，如下表中所示。

表 28-9 YCbCr 逐行视频格式下的数据存储

字节地址	31:24	23:16	15:8	7:0
0	Y_{n+1}	Cr_n	Y_n	Cb_n
4	Y_{n+3}	Cr_{n+2}	Y_{n+2}	Cb_{n+2}

28.5.5 YCbCr 格式——仅含 Y 分量

特性：

- 光栅格式
- YCbCr4:2:2
- 缓冲区仅包含 Y 分量信息——单色图像

像素分量包括 Y（亮度）、Cb 和 Cr（蓝色色度和红色色度）。在此模式下，将丢弃色度信息。仅存储每个像素采用 8 位进行编码的亮度分量，如下表所示。

结果为单色图像，其分辨率与原始 YCbCr 数据的分辨率相同。

表 28-10 YCbCr 逐行视频格式下的数据存储——Y 分量提取模式

字节地址	31:24	23:16	15:8	7:0
0	Y_{n+3}	Y_{n+2}	Y_{n+1}	Y_n
4	Y_{n+7}	Y_{n+6}	Y_{n+5}	Y_{n+4}

28.5.6 半分辨率图像提取

这是对先前接收模式的修改，适用于单色、RGB 或 Y 分量提取模式。此模式仅允许存储半分辨率图像。可通过 OELS 和 LSM 控制位进行选择。

28.6 DCMI 中断

有五种中断源。所有中断都可通过软件屏蔽。全局中断（IT_DCMI）是所有单个中断的逻辑或运算所得结果。下表列出了所有中断。

表 28-11 DCMI 中断

中断名称	中断事件
IT_LINE	表示行结束
IT_FRAME	表示帧捕获结束
IT_OVR	表示接收的数据发生溢出错误

中断名称	中断事件
IT_VSYNC	表示同步帧
IT_ERR	表示内嵌码同步帧检测期间检测到错误
IT_DCMI	以上中断的逻辑或结果

28.7 DCMI 寄存器

基地址：0x5005 0000

空间大小：0x400

所有 DCMI 寄存器都必须作为 32 位字访问，否则将发生总线错误。

28.7.1 DCMI 控制寄存器（DCMI_CR）

偏移地址：0x00

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HALFWD	Res										OELS	LSM	OEBS	BSM[1:0]	
rw											rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BENDIAN	ENABLE	Res		EDM[1:0]		FCRC[1:0]		VSPOL	HSPOL	PCKPOL	ESS	JPEG	CROP	CM	CAPTURE
rw	rw			rw		rw		rw	rw	rw	rw	rw	rw	rw	rw

位 31	HALFWD：半字（Half word） DCMI 产生 DMA 请求的字长配置。 • 0：每接收一个 32-bit word，DCMI 产生一次 DMA 请求。 • 1：每接收一个 16-bit half word，DCMI 产生一次 DMA 请求（DR[31:16]数据无效）。
位 30:21	Res：保留 必须保持复位值。
位 20	OELS：奇数/偶数行选择（行选择开始）（Odd/even line select, Line select start） 此位与 LSM 字段（LSM=1）一起使用 • 0：接口在帧开始后捕获第一行，同时丢弃第二行。 • 1：接口在帧开始后捕获第二行，同时丢弃第一行。
位 19	LSM：行选择模式（Line select mode） • 0：接口捕获所有接收到的行 • 1：接口每两行捕获一行
位 18	OEBS：奇数/偶数字节选择（字节选择开始）（Odd/even byte select, byte select start） 此位与 BSM 字段（BSM<>00）一起使用 • 0：接口从帧/行开始捕获第一个数据（字节或双字节），同时丢弃第二个字节。 • 1：接口从帧/行开始捕获第二个数据（字节或双字节），同时丢弃第一个字节。
位 17:16	BSM[1:0]：字节选择模式（Byte select mode） • 00：接口捕获所有接收到的数据

	• 01：接口对接收到的数据每隔一个字节进行捕获 • 10：接口每四个字节捕获一个字节 • 11：接口每四个字节捕获两个字节 *注意：此模式仅在 EDM[1:0]=00 时有效。对于所有其它 EDM 值，必须将此位字段编程为复位值。*
位 15	BENDIAN：8 位模式下，DR 寄存器读出的数据格式的顺序（Byte endian） • 0：8 位模式下，DR 寄存器读出的数据格式为 {Byte3,Byte2,Byte1,Byte0}。 • 1：8 位模式下，DR 寄存器读出的数据格式为 {Byte2,Byte3,Byte0,Byte1}。 *注意：只能在 8 位模式下设置此位。*
位 14	ENABLE：DCMI 使能（DCMI enable） • 0：禁止 DCMI • 1：使能 DCMI *注意：使能此位之前，应对 DCMI 配置寄存器进行适当的设置。*
位 13:12	Res：保留 必须保持复位值。
位 11:10	EDM[1:0]：扩展数据模式（Extended data mode） • 00：接口每个像素时钟捕获 8 位数据 • 01：接口每个像素时钟捕获 10 位数据 • 10：接口每个像素时钟捕获 12 位数据 • 11：接口每个像素时钟捕获 14 位数据
位 9:8	FCRC[1:0]：帧捕获率控制（Frame capture rate control） 这些位定义了帧捕获频率。仅在连续采集模式下有效。快照模式下将被忽略。 • 00：捕获所有帧 • 01：每隔一帧捕获一次（带宽降低 50%） • 10：每隔三帧捕获一次（带宽降低 75%） • 11：保留
位 7	VSPOL：垂直同步极性（Vertical synchronization polarity） 此位指示数据在并行接口上无效时 VSYNC 引脚的电平。 • 0：VSYNC 低电平有效 • 1：VSYNC 高电平有效
位 6	HSPOL：水平同步极性（Horizontal synchronization polarity） 此位指示数据在并行接口上无效时 HSYNC 引脚的电平。 • 0：HSYNC 低电平有效 • 1：HSYNC 高电平有效
位 5	PCKPOL：像素时钟极性（Pixel clock polarity） 此位用来配置像素时钟的捕获沿。 • 0：下降沿有效 • 1：上升沿有效

位 4	ESS：内嵌码同步选择（Embedded synchronization select） • 0：硬件同步，数据捕获（帧/行开始/停止）由 HSYNC/VSYNC 信号同步。 • 1：内嵌码同步，数据捕获由数据流中嵌入的同步码同步。 *注意：仅对 8 位并行数据有效。ESS 位置 1 时，将忽略 HSPOL/VSPOL。JPEG 模式下会禁止此位。*
位 3	JPEG：JPEG 格式（JPEG format） • 0：未经压缩的视频格式 • 1：此位用于 JPEG 数据传输。HSYNC 信号用作数据使能信号。此模式下无法使用裁剪和内嵌码同步功能（ESS 位）。
位 2	CROP：裁剪功能（Crop feature） • 0：捕获完整图像。这种情况下，图像帧包含的字节总数应该为 4 的倍数。 • 1：仅捕获剪裁寄存器所指定的窗口中的数据。如果窗口大小超出图片大小，则仅捕获图片大小。
位 1	CM：捕获模式（Capture mode） • 0：连续采集模式：收到的数据将通过 DMA 传输到目标存储区。缓冲区位置和模式（线性或循环缓冲区）由系统 DMA 控制。 • 1：快照模式（单帧）：一旦激活，接口将等待帧开始，然后通过 DMA 传输。帧结束时，将自动复位 CAPTURE 位。
位 0	CAPTURE：使能捕获（Capture enable） • 0：禁止捕获。 • 1：使能捕获。 摄像头接口等待第一帧开始，然后生成一个 DMA 请求以将收到的数据传输到目标存储器中。 在快照模式下，收到的第一帧结束时将自动使 CAPTURE 位清零。在连续采集模式下，如果在执行捕获操作时通过软件将此位清零，则帧结束后此位的清零才生效。 注意：使能此位之前，应对 DMA 控制器和所有 DCMI 配置寄存器进行适当的编程。

28.7.2 DCMI 状态寄存器（DCMI_SR）

偏移地址：0x04

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res													FNE	VSYNC	HSYNC
													r	r	r

位 31:3	Res：保留 必须保持复位值。
位 2	FNE：FIFO 非空（FIFO not empty） 此位指示 FIFO 的状态 • 0：FIFO 为空 • 1：FIFO 包含有效数据
位 1	VSYNC：垂直同步（Vertical synchronization）

	此位指示编程了适当极性的 VSYNC 引脚的状态。 使用内嵌码同步时，此位的含义如下： • 0：有效帧 • 1：在帧之间同步 如果使用内嵌码同步，则仅当 DCMI_CR 中的 CAPTURE 位置 1 时，此位才有意义。
位 0	HSYNC：水平同步（Horizontal synchronization） 此位指示编程了适当极性的 HSYNC 引脚的状态。 使用内嵌码同步时，此位的含义如下： • 0：有效行 • 1：在行之间同步 如果使用内嵌码同步，则仅当 DCMI_CR 中的 CAPTURE 位置 1 时，此位才有意义。

28.7.3 DCMI 裁剪窗口起点（DCMI_CWSTRT）

偏移地址：0x20

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res			VST[12:0]												
			rw												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res		HOFFCNT[13:0]													
		rw													

位 31:29	Res：保留 必须保持复位值。
位 28:16	VST[12:0]：窗口开始的行数（Vertical start line count） 图像捕获从此行开始。对之前行的数据不予捕获。 • 0x0000：行 1 • 0x0001：行 2 • 0x0002：行 3 • ....
位 15:14	Res：保留 必须保持复位值。
位 13:0	HOFFCNT[13:0]：窗口开始的水平方向位移（Horizontal offset count） 窗口行内，每行在捕获数据前需空出的像素时钟个数。

28.7.4 DCMI 原始中断状态寄存器（DCMI_RIS）

偏移地址：0x08

复位值：0x0000 0000

DCMI_RIS 用来指示原始中断状态，以只读方式访问。执行读操作时，此寄存器返回的是对应中断未被 DCMI_IER 寄存器屏蔽的原始状态。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res											LINE_RIS	VSYNC_RIS	ERR_RIS	OVR_RIS	FRAME_RIS
											r	r	r	r	r

位 31:5	Res：保留 必须保持复位值。
位 4	LINE_RIS：行原始中断状态（Line raw interrupt status） 当 HSYNC 信号从无效状态更改为有效状态时，此位置 1。即使行无效也会变成高电平。 如果使用内嵌码同步，则仅当 DCMI_CR 中的 CAPTURE 位置 1 时，此位才置 1。向 DCMI_ICR 中的 LINE_ISC 位写入“1”即可将此位清零。
位 3	VSYNC_RIS：VSYNC 原始中断状态（VSYNC raw interrupt status） 当 VSYNC 信号从无效状态更改为有效状态时，此位置 1。 如果使用内嵌码同步，则仅当 DCMI_CR 中的 CAPTURE 位置 1 时，此位才置 1。向 DCMI_ICR 中的 VSYNC_ISC 位写入“1”即可将此位清零。
位 2	ERR_RIS：同步错误原始中断状态（Synchronization error raw interrupt status） • 0：未检测到同步错误。 • 1：未按正确顺序接收内嵌码同步字符。 此位仅在内嵌码同步模式下有效。向 DCMI_ICR 中的 ERR_ISC 位写入“1”即可将此位清零。 *注意：此位仅适用于内嵌码同步模式。*
位 1	OVR_RIS：溢出原始中断状态（Overrun raw interrupt status） • 0：未发生数据缓冲区溢出。 • 1：发生数据缓冲区溢出，数据 FIFO 损坏。 向 DCMI_ICR 中的 OVR_ISC 位写入“1”即可将此位清零。
位 0	FRAME_RIS：捕获完成原始中断状态（Capture complete raw interrupt status） • 0：没有新的捕获数据。 • 1：已捕获一帧。 对一帧数据或裁剪窗口内的数据捕获完毕后，此位置 1。如果捕获裁剪窗口，则将在裁剪窗口最后一行的行结束时将此位置 1。即使捕获的帧为空（例如，窗口超出帧范围），此位也将置 1。 向 DCMI_ICR 中的 FRAME_ISC 位写入“1”即可将此位清零。

28.7.5 DCMI 中断使能寄存器（DCMI_IER）

偏移地址：0x0C

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res											LINE_IE	VSYNC_IE	ERR_IE	OVR_IE	FRAME_IE
											rw	rw	rw	rw	rw

位 31:5	Res：保留 必须保持复位值。

位 4	LINE_IE：使能行中断（Line interrupt enable） • 0：接收到行时不生成中断 • 1：完全接收到行时生成一个中断
位 3	VSYNC_IE：VSYNC 中断使能（VSYNC interrupt enable） • 0：不生成中断 • 1：VSYNC 每次从无效电平变为有效电平时都生成一个中断 VSYNC 信号的有效电平由 VSPOL 位定义。
位 2	ERR_IE：同步错误中断使能（Synchronization error interrupt enable） • 0：不生成中断 • 1：如果未按正确顺序接收内嵌码同步代码，则生成一个中断。 *注意：此位仅适用于内嵌码同步模式。*
位 1	OVR_IE：溢出中断使能（Overrun interrupt enable） • 0：不生成中断 • 1：如果 DMA 无法在收到新数据（32 位）之前传输上一个数据，则生成一个中断。
位 0	FRAME_IE：捕获完成中断使能（Capture complete interrupt enable） • 0：不生成中断 • 1：接收完成一帧数据或裁剪窗口内的数据（在裁剪模式下），生成一个中断。

28.7.6 DCMI 屏蔽中断状态寄存器（DCMI_MIS）

偏移地址：0x10

复位值：0x0000 0000

此 DCMI_MIS 寄存器是一个只读寄存器。执行读操作时，此寄存器将返回相应中断的当前屏蔽状态（取决于 DCMI_IER 中的值）。如果 DCMI_IER 中的相应使能位和 DCMI_RIS 中的相应位都置 1，则此寄存器中的对应位将置 1。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res											LINE_MIS	VSYNC_MIS	ERR_MIS	OVR_MIS	FRAME_MIS
											r	r	r	r	r

位 31:5	Res：保留 必须保持复位值。
位 4	LINE_MIS：行屏蔽中断状态（Line masked interrupt status） 此位指示屏蔽行中断的状态 • 0：接收到行时不生成中断 • 1：收到完整一行数据且 DCMI_IER 中的 LINE_IE 位置 1 时生成一个中断。
位 3	VSYNC_MIS：VSYNC 屏蔽中断状态（VSYNC masked interrupt status） 此位指示屏蔽 VSYNC 中断的状态 • 0：VSYNC 电平跳变时不生成中断 • 1：VSYNC 每次从无效电平变为有效电平且 DCMI_IER 中的 VSYNC_IE 位置 1

	时都生成一个中断。 VSYNC 信号的有效电平由 VSPOL 位定义。
位 2	ERR_MIS：同步错误屏蔽中断状态（Synchronization error masked interrupt status） 此位指示屏蔽同步错误中断 • 0：发生同步错误时不生成中断 • 1：如果未按正确顺序接收内嵌同步码且 DCMI_IER 中的 ERR_IE 位置 1，则生成一个中断。 *注意：此位仅适用于内嵌码同步模式。*
位 1	OVR_MIS：溢出屏蔽中断状态（Overrun masked interrupt status） 此位指示屏蔽溢出中断的状态 • 0：发生溢出时不生成中断 • 1：如果 DMA 无法在收到新数据（32 位）之前传输上一个数据且 DCMI_IER 中的 OVR_IE 位置 1，则生成一个中断。
位 0	FRAME_MIS：捕获完成屏蔽中断状态（Capture complete masked interrupt status） 此位指示屏蔽捕获完成中断的状态 • 0：完成捕获后不生成中断 • 1：接收完成一帧数据或裁剪窗口内的数据（在裁剪模式下），且 DCMI_IER 中的 FRAME_IE 位置 1 时生成一个中断。

28.7.7 DCMI 中断清零寄存器（DCMI_ICR）

偏移地址：0x14

复位值：0x0000 0000

DCMI_ICR 寄存器为只写寄存器。向此寄存器的某一位写入“1”可将 DCMI_RIS 和 DCMI_MIS 寄存器中的相应位清零。写入“0”则不会带来任何影响。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res											LINE_ISC	VSYNC_ISC	ERR_ISC	OVR_ISC	FRAME_ISC
											w	w	w	w	w

位 31:5	Res：保留 必须保持复位值。
位 4	LINE_ISC：线中断状态清零（Line interrupt status clear） 在此位中写入“1”可将 DCMI_RIS 寄存器中的 LINE_RIS 清零。
位 3	VSYNC_ISC：垂直同步中断状态清零（Vertical synch interrupt status clear） 在此位中写入“1”可将 DCMI_RIS 中的 VSYNC_RIS 位清零。
位 2	ERR_ISC：同步错误中断状态清零（Synchronization error interrupt status clear） 在此位中写入“1”可将 DCMI_RIS 中的 ERR_RIS 位清零。 *注意：此位仅适用于内嵌码同步模式。*

位 1	OVR_ISC：溢出中断状态清零（Overrun interrupt status clear） 在此位中写入“1”可将 DCMI_RIS 中的 OVR_RIS 位清零。
位 0	FRAME_ISC：捕获完成中断状态清零（Capture complete interrupt status clear） 在此位中写入“1”可将 DCMI_RIS 中的 FRAME_RIS 位清零。

28.7.8 DCMI 内嵌同步码寄存器（DCMI_ESCR）

偏移地址：0x18

复位值：0x0000 0000

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

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

位 31:24	FEC：帧结束分隔码（Frame end delimiter code） 此字节指定帧结束分隔码。由 4 个字节组成，格式为 0xFF0x000x00FEC。如果将 FEC 编程为 0xFF，所有未使用的其它代码（0xFF0000XY）都将视为帧结束分隔符。
位 23:16	LEC：行结束分隔码（Line end delimiter code） 此字节指定行结束分隔码。由 4 个字节组成，格式为 0xFF0x000x00LEC。
位 15:8	LSC：行开始分隔码（Line start delimiter code） 此字节指定行开始分隔码。由 4 个字节组成，格式为 0xFF0x000x00LSC。
位 7:0	FSC：帧开始分隔码（Frame start delimiter code） 此字节指定帧开始分隔码。由 4 个字节组成，格式为 0xFF0x000x00FSC。如果将 FSC 编程为 0xFF，将检测不到任何帧开始分隔符。但在 FEC 代码后第一次出现 LSC 时将视为帧分隔符的开始。

28.7.9 DCMI 内嵌码同步取消屏蔽寄存器（DCMI_ESUR）

偏移地址：0x1C

复位值：0x0000 0000

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

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

位 31:24	FEU：帧结束分隔符取消屏蔽（Frame end delimiter unmask） 此字节指定对帧结束分隔码的屏蔽。 • 0：将帧结束分隔符与收到的数据进行比较时，将屏蔽 DCMI_ESCR 中 FEC 字节中的相应位。 • 1：将帧结束分隔符与收到的数据进行比较时，将比较 DCMI_ESCR 中 FEC 字节中的相应位。
位 23:16	LEU：行结束分隔符取消屏蔽（Line end delimiter unmask） 此字节指定对行结束分隔码的屏蔽。 • 0：将行结束分隔符与收到的数据进行比较时，将屏蔽 DCMI_ESCR 中 LEC 字节中的相应位。

	• 1：将行结束分隔符与收到的数据进行比较时，将比较 DCMI_ESCR 中 LEC 字节中的相应位。
位 15:8	LSU：行开始分隔符取消屏蔽（Line start delimiter unmask） 此字节指定对行开始分隔码的屏蔽。 • 0：将行开始分隔符与收到的数据进行比较时，将屏蔽 DCMI_ESCR 中 LSC 字节中的相应位。 • 1：将行开始分隔符与收到的数据进行比较时，将比较 DCMI_ESCR 中 LSC 字节中的相应位。
位 7:0	FSU：帧开始分隔符取消屏蔽（Frame start delimiter unmask） 此字节指定对帧开始分隔码的屏蔽。 • 0：将帧开始分隔符与收到的数据进行比较时，将屏蔽 DCMI_ESCR 中 FSC 字节中的相应位。 • 1：将帧开始分隔符与收到的数据进行比较时，将比较 DCMI_ESCR 中 FSC 字节中的相应位。

28.7.10 DCMI 裁剪窗口大小（DCMI_CWSIZE）

偏移地址：0x24

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res		VLINE[13:0]													
		rw													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res		CAPCNT[13:0]													
		rw													

位 31:30	Res：保留 必须保持复位值。
位 29:16	VLINE[13:0]：垂直行计数（Vertical line count） 窗口内包含的行数。 • 0x0000：行 • 0x0001：2 行 • 0x0002：3 行 • ....
位 15:14	Res：保留 必须保持复位值。
位 13:0	CAPCNT[13:0]：捕获计数（Capture count） 窗口内要捕获的像素时钟数。其值应与表示不同并行接口宽度的字对齐数据相对应。 • 0x0000：1 个像素 • 0x0001：2 个像素 • 0x0002：3 个像素 • ....

28.7.11 DCMI 数据寄存器（DCMI_DR）

偏移地址：0x28

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Byte3								Byte2							

r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Byte1								Byte0							
r								r							

位 31:24	Byte3：存放待 DMA 搬移的数据（Data byte 3 for DMA transfer）
位 23:16	Byte2：存放待 DMA 搬移的数据（Data byte 2 for DMA transfer）
位 15:8	Byte1：存放待 DMA 搬移的数据（Data byte 1 for DMA transfer）
位 7:0	Byte0：存放待 DMA 搬移的数据（Data byte 0 for DMA transfer）

在发起 DMA 请求前，数字摄像头接口将所接收的数据按照 32 位的格式打包。一个 4 字深度的 FIFO 确保了足够长的时间以完成 DMA 传输，避免 DMA 溢出。

29 SDIO 接口（SDIO）

29.1 SDIO 主要功能

SD/SDIO MMC 卡主机模块（SDIO）在 AHB 外设总线和多媒体卡（MMC）、SD 存储卡、SDIO 卡和 CE-ATA 设备间提供了操作接口。

多媒体卡系统规格书由 MMCA 技术委员会发布，可以在多媒体卡协会的网站上（www.mmca.org）获得。CE-ATA 系统规格书可以在 CE-ATA 工作组的网站上（www.ce-ata.org）获得。

SDIO 的主要功能如下：

- 与多媒体卡系统规范 V4.2 全兼容。支持三种不同的数据总线模式：1 位（默认）、4 位和 8 位。
- 与多媒体卡系统规范 V4.1 及更早的版本全兼容（向前兼容）。
- 与 SD 存储卡规范 V2.0 全兼容。
- 与 SDI/O 卡规范 V2.0 全兼容：支持两种不同的数据总线模式：1 位（默认）和 4 位。
- 完全支持 CE-ATA 功能（与 CE-ATA 数字协议版本 1.1 全兼容）。
- 8 位总线模式下数据传输速率可达 48MHz。
- 数据和命令输出使能信号，用于控制外部双向驱动器。

注意：

1．SDIO 没有 SPI 兼容的通信模式。

2．在多媒体卡系统规范 V2.11 中，定义 SD 存储卡协议是多媒体卡协议的超集。只支持 I/O 模式的 SD 卡或复合卡中的 I/O 部分不能支持 SD 存储设备中很多需要的命令，这里有些命令在 SDI/O 设备中不起作用，如擦除命令，因此 SDIO 不支持这些命令。另外，SD 存储卡和 SDI/O 卡中有些命令是不同的，SDIO 也不支持这些命令。细节可以参考 SDI/O 卡规格书版本 1.0。使用现有的 MMC 命令机制，在 MMC 接口上可以实现 CE-ATA 的支持。SDIO 接口的电气和信号定义详见 MMC 参考资料。

多媒体卡/SD 总线将所有卡与控制器相连。

当前版本的 SDIO 在同一时间里只能支持一个 SD/SDIO/MMC4.2 卡，但可以支持多个 MMC 版本 4.1 或以前版本的卡。

29.2 SDIO 总线拓扑

总线上的通信是通过传送命令和数据实现。

在多媒体卡/SD/SDI/O 总线上的基本传输是命令/响应传输，这样的总线传输在命令或响应机制下实现信息直接交换；另外，某些操作还具有数据令牌。

在 SD/SDIO 存储器卡上传送的数据是以数据块的形式传输；在 MMC 上传送的数据是以数据块或数据流的形式传输；在 CE-ATA 设备上传送的数据也是以数据块的形式传输。

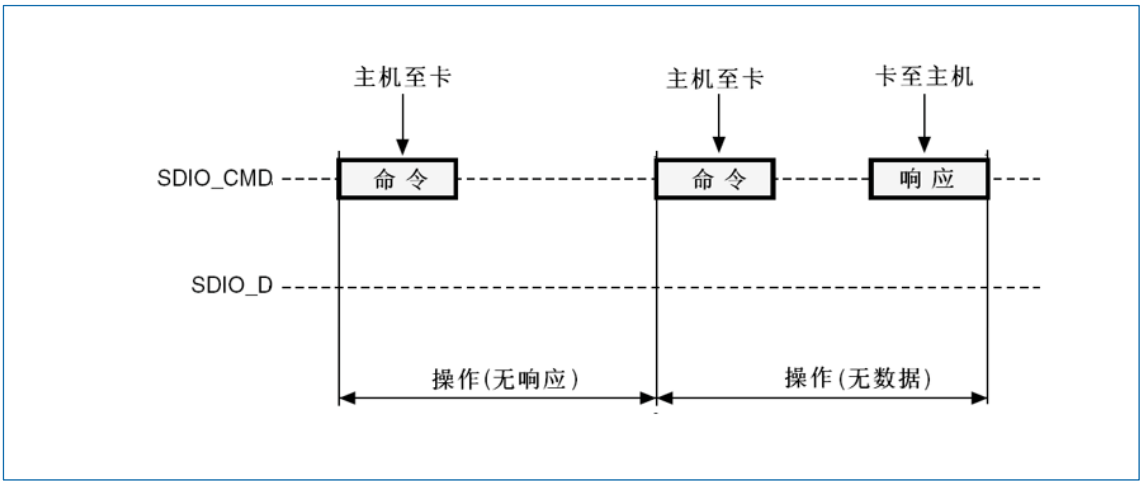


图 29-1 SDIO“无响应”和“无数据”操作

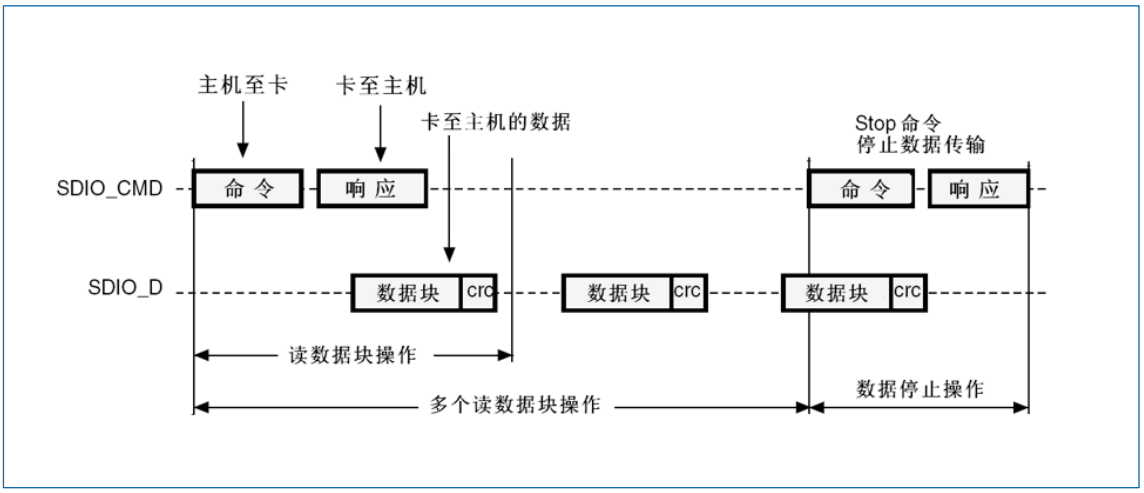


图 29-2 SDIO（多）数据块读操作

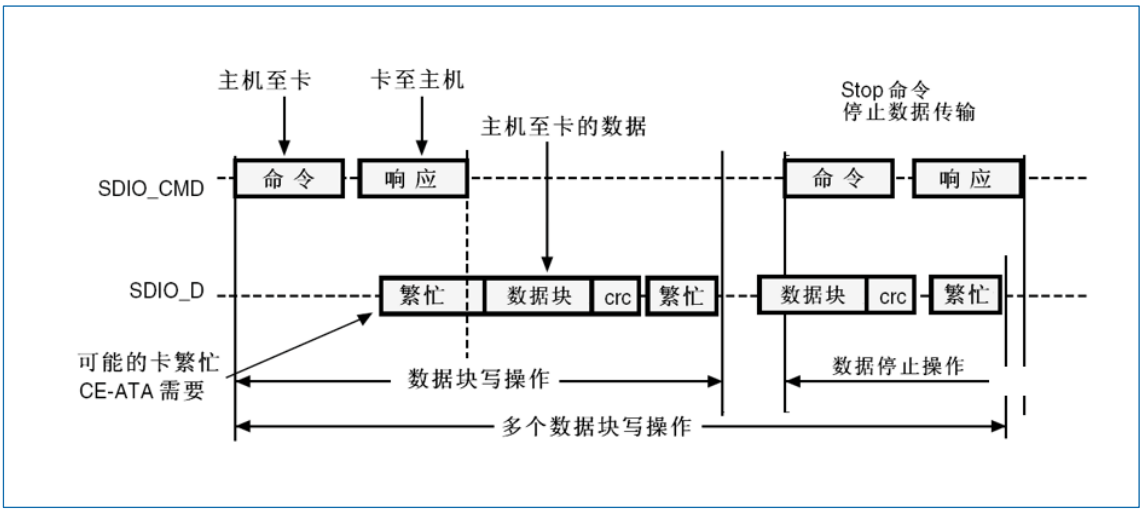


图 29-3 SDIO（多）数据块写操作

注意：当有 Busy（繁忙）信号时，SDIO（SDIO_D0 被拉低）将不会发送任何数据。

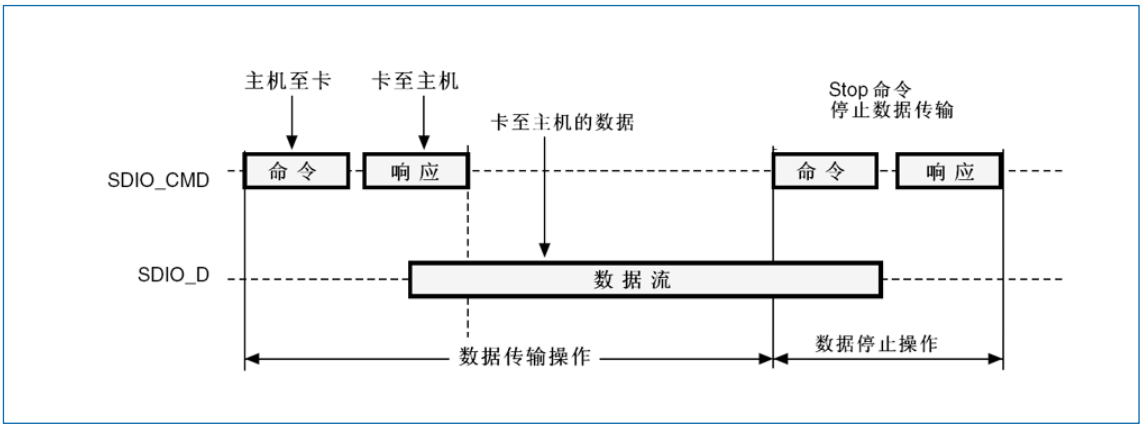


图 29-4 SDIO 连续读操作

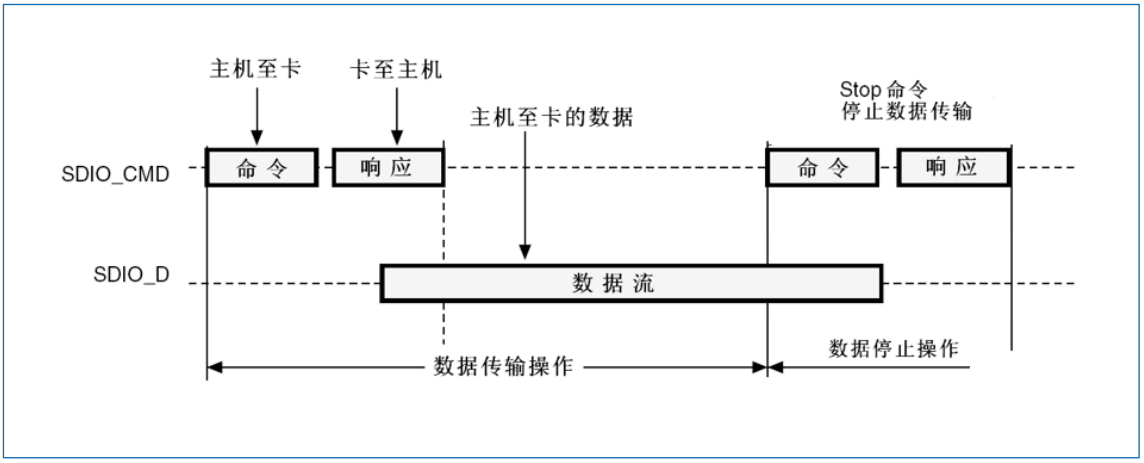


图 29-5 SDIO 连续写操作

29.3 SDIO 功能描述

SDIO 包含 2 个部分：

- SDIO 适配器模块：实现所有 MMC/SD/SDI/O 卡的相关功能，如时钟的产生、命令和数据的传送。
- AHB 总线接口：操作 SDIO 适配器模块中的寄存器，并产生中断和 DMA 请求信号。

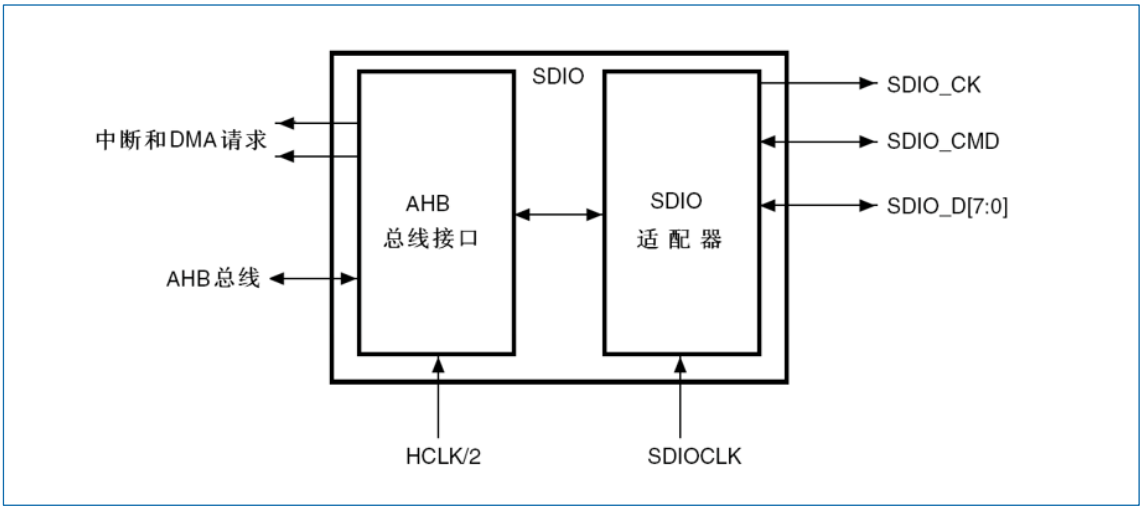


图 29-6 SDIO 框图

复位后默认 SDIO_D0 用于数据传输。初始化后主机可以改变数据总线的宽度。

如果一个多媒体卡接到了总线上，则 SDIO_D0、SDIO_D[3:0]或 SDIO_D[7:0]可以用于数据传输。MMC

版本 V3.31 和之前版本的协议只支持 1 位数据线，所以只能用 SDIO_D0。

如果一个 SD 或 SDI/O 卡接到了总线上，主机可以使用 SDIO_D0 或 SDIO_D[3:0]来配置数据传输。所有的数据线都工作在推挽模式。

SDIO_CMD 有两种操作模式：

- 用于初始化时的开漏模式（仅用于 MMC 版本 V3.31 或之前版本）。
- 用于命令传输的推挽模式（SD/SDI/O 卡和 MMC V4.2 版在初始化时也使用推挽驱动）。

SDIO_CK 是卡的时钟：每个时钟周期在命令和数据线上传输 1 位命令或数据。对于多媒体卡 V3.31 协议，时钟频率可以在 0 MHz 至 20 MHz 间变化；对于多媒体卡 V4.0/4.2 协议，时钟频率可以在 0 MHz 至 48 MHz 间变化；对于 SD 或 SDI/O 卡，时钟频率可以在 0 MHz 至 25 MHz 间变化。

SDIO 使用两个时钟信号：

- SDIO 适配器时钟（SDIOCLK=HCLK）
- AHB 总线时钟（PCLK2, 即 HCLK/2）

PCLK2 和 SDIO_CK 的时钟频率必须遵循以下条件：

Frequency(PCLK2) > ((3* Width)/32) * Frequency(SDMMC_CK)

下表适用于多媒体卡/SD/SDI/O 卡总线：

表 29-1 SD10 引脚定义

引脚	方向	说明
SDIO_CK	输出	多媒体卡/SD/SDIO 卡时钟。这是从主机至卡的时钟线。
SDIO_CMD	双向	多媒体卡/SD/SDIO 卡命令。这是双向的命令/响应信号线。
SDIO_D[7:0]	双向	多媒体卡/SD/SDIO 卡数据。这些是双向的数据总线。

29.3.1 SDIO 适配器

下图是简化的 SDIO 适配器框图：

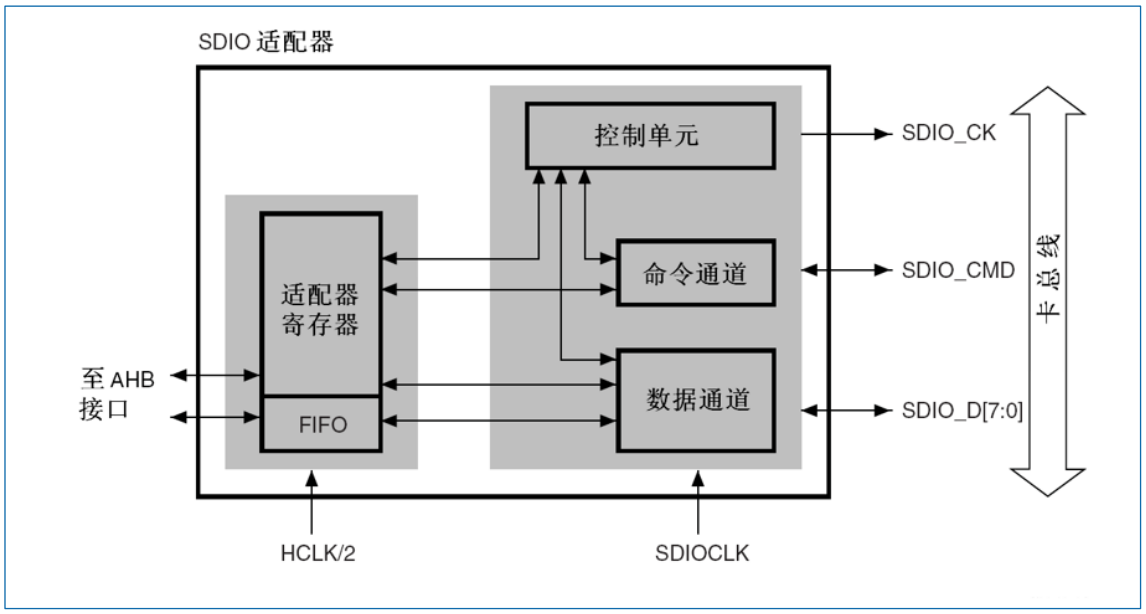


图 29-7 SDIO 适配器

SDIO 适配器是多媒体/加密数字存储卡总线的主设备（主机），用于连接一组多媒体卡或加密数字存储卡，它包含以下 5 个部分：

- 适配器寄存器模块
- 控制单元
- 命令通道
- 数据通道
- 数据 FIFO

注意：适配器寄存器和 FIFO 使用 AHB 总线一侧的时钟（HCLK/2），控制单元、命令通道和数据通道使用 SDIO 适配器一侧的时钟（SDIOCLK）。

29.3.1.1 适配器寄存器模块

适配器寄存器模块包含所有系统寄存器。该模块还产生清除多媒体卡中静态标记的信号，当在 SDIO 清除寄存器中的相应位写'1'时会产生清除信号。

29.3.1.2 控制单元

控制单元包含电源管理功能和为存储器卡提供的时钟分频。共有三种电源阶段：

- 电源关闭
- 电源启动
- 电源开启

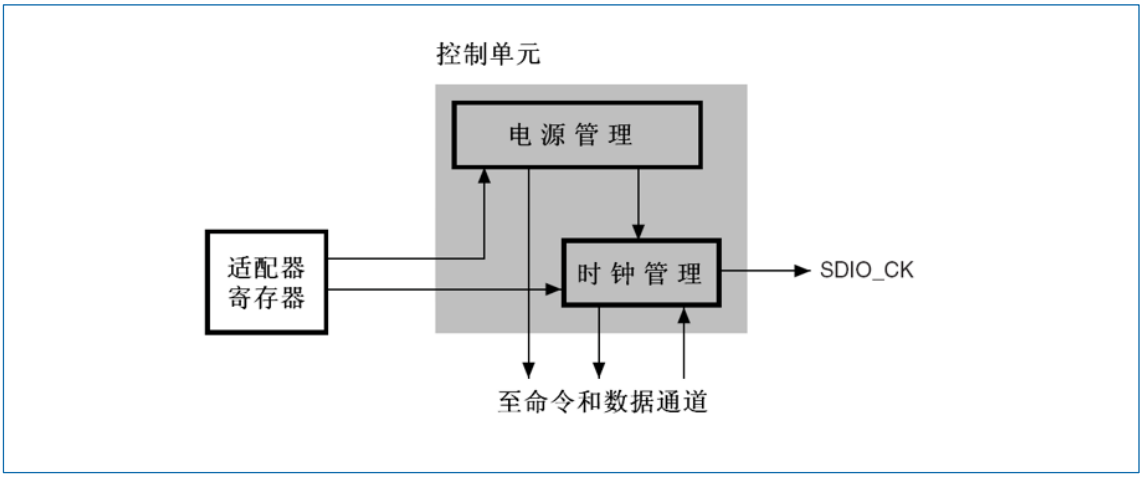


图 29-8 控制单元

上图为控制单元的框图，有电源管理和时钟管理子单元。在电源关闭和电源启动阶段，电源管理子单元会关闭卡总线上的输出信号。

时钟管理子单元产生和控制 SDIO_CK 信号。SDIO_CK 输出可以使用时钟分频或时钟旁路模式。

下述情况下没有时钟输出：

- 复位后
- 在电源关闭和电源启动阶段
- 当启动了省电模式并且卡总线处于空闲状态（命令通道和数据通道子单元都进入空闲阶段后的 8 个时钟周期）

29.3.1.3 命令通道

命令通道单元向卡发送命令并从卡接收响应。

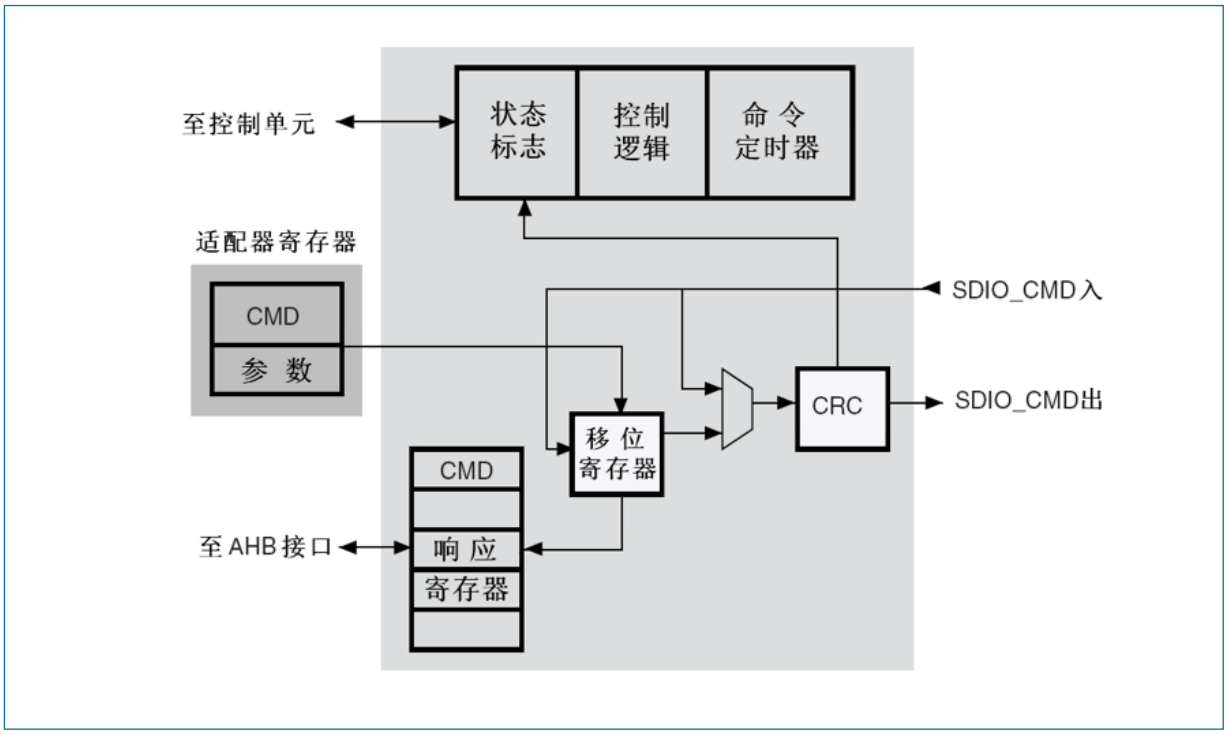


图 29-9 SDIO 适配器命令通道

命令通道状态机（CPSM）

当写入命令寄存器并设置了使能位，开始发送命令。命令发送完成时，命令通道状态机（CPSM）设置状态标志，并在不需要响应时进入空闲状态（见下图）。当收到响应后，接收到的 CRC 码将会与内部产生的 CRC 码比较，然后设置相应的状态标志。

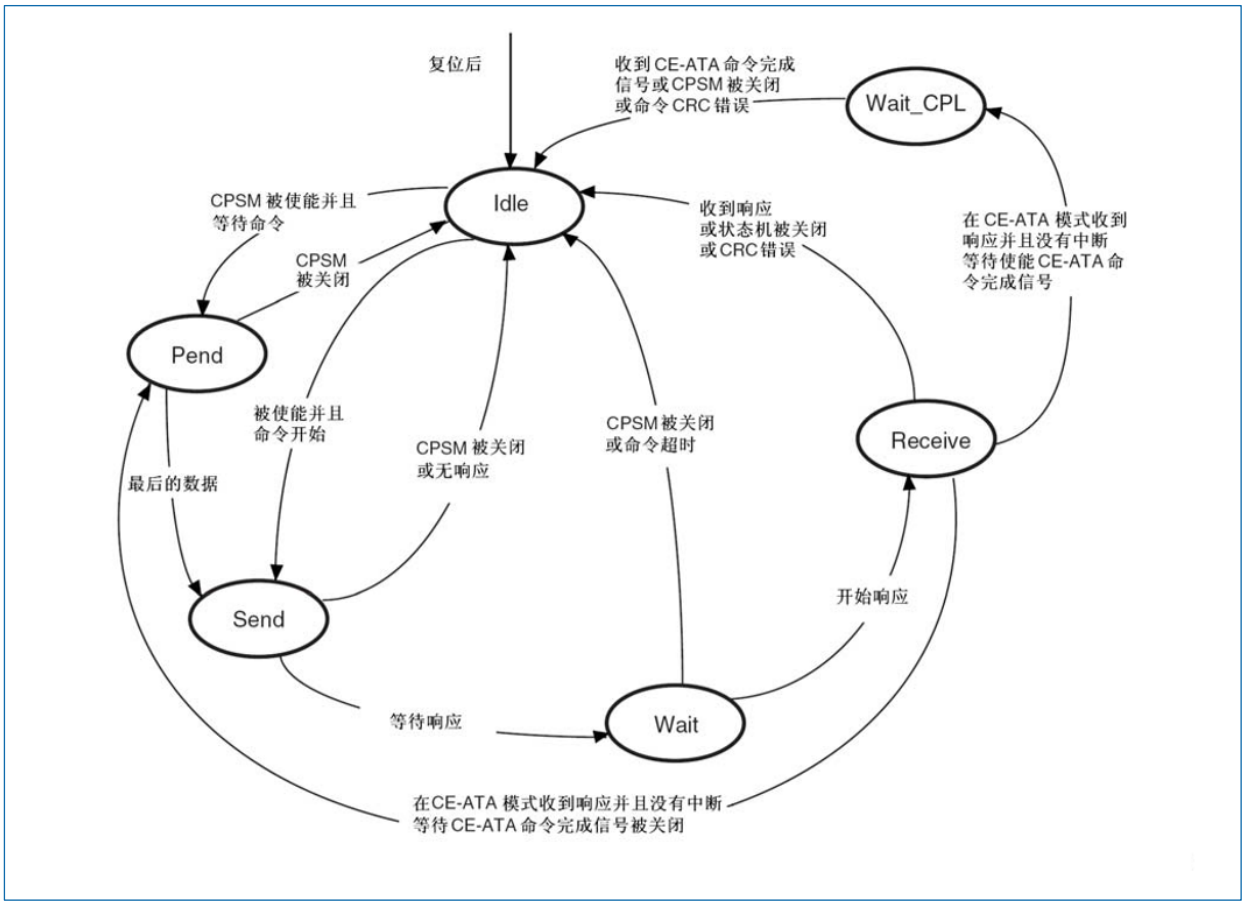


图 29-10 命令通道状态机（CPSM）

当进入等待（Wait）状态时，命令定时器开始运行；当 CPSM 进入接收（Receive）状态之前，产生了超时，则设置超时标志并进入空闲（Idle）状态。

注意：命令超时固定为 64 个 SDIO_CK 时钟周期。

- 如果在命令寄存器设置了中断位，则关闭定时器，CPSM 等待某一个卡发出的中断请求。
- 如果命令寄存器中设置挂起位，CPSM 进入挂起（Pend）状态并等待数据通道子单元发出的 CmdPend 信号；在检测到 CmdPend 信号时，CPSM 进入发送（Send）状态，这将触发数据计数器来发送停止命令的功能。

注意：CPSM 保持在空闲状态至少 8 个 SDIO_CK 周期，以满足 N_{CC} 和 N_{RC} 时序限制。N_{CC} 是两个主机命令间的最小间隔；N_{RC} 是主机命令与卡响应之间的最小间隔。

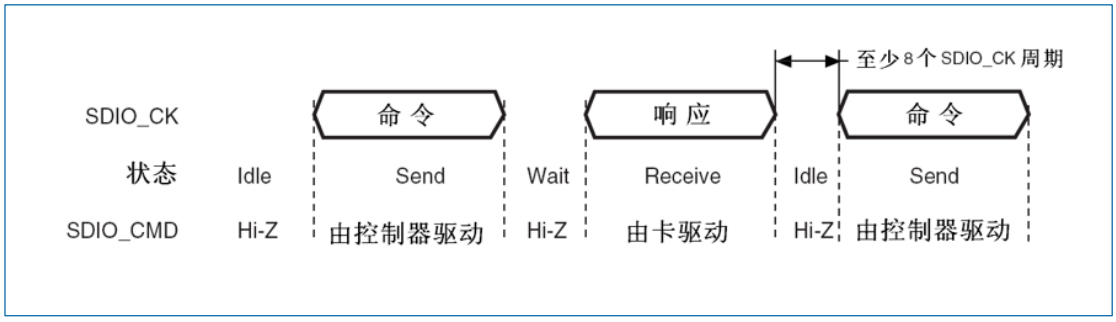


图 29-11 SDIO 命令传输

命令格式

- 命令：命令是用于开始一项操作。主机向一个指定的卡或所有的卡发出带地址的命令或广播命令（广播命令只适合于 MMC V3.31 或之前的版本）。命令在 CMD 线上串行传送。所有命令的长度固定为 48 位，下表给出了多媒体卡、SD 存储卡和 SDIO 卡上一般的命令格式。

  CE-ATA 命令是 MMC V4.2 命令的扩充，所以具有相同的格式。命令通道操作于半双工模式，这样命令和响应可以分别发送和接收。如果 CPSM 不处在发送状态，SDIO_CMD 输出处于高阻状态，如图 29-11 所示。SDIO_CMD 上的数据与 SDIO_CK 的上升沿同步。

表 29-2 命令格式

位	宽度	数值	说明
47	1	0	开始位
46	1	1	传输位
[45:40]	6	-	命令索引
[39:8]	32	-	参数
[7:1]	7	-	CRC7
0	1	1	结束位

- 响应：响应是由一个被指定地址的卡发送到主机，对于 MMCV3.31 或以前版本所有的卡同时发送响应；响应是对先前接收到命令的一个应答。响应在 CMD 线上串行传送。

SDIO 支持 2 种响应类型，2 种类型都有 CRC 错误检测：

- 48 位短响应
- 136 位长响应

注意：如果响应不包含 CRC（如 CMD1 的响应），设备驱动应该忽略 CRC 失败状态。

表 29-3 短响应格式

位	宽度	数值	说明
47	1	0	开始位
46	1	0	传输位
[45:40]	6	-	命令索引
[39:8]	32	-	参数
[7:1]	7	-	CRC7（或 1111111）
0	1	1	结束位

表 29-4 长响应格式

位	宽度	数值	说明
135	1	0	开始位
134	1	0	传输位
[133:128]	6	111111	保留
[127:1]	127	-	CID 或 CSD（包含内部 CRC7）
0	1	1	结束位

命令寄存器包含命令索引（发至卡的 6 位）和命令类型；命令本身决定了是否需要响应和响应的类型是 48 位还是 136 位（见 29.9.4 SDIO 命令寄存器（SDIO_CMD）节）。命令通道中的状态标志示于下表：

表 29-5 命令通道状态标志

标志	说明
CMDREND	响应的 CRC 正确
CCRCFAIL	响应的 CRC 错误
CMDSENT	命令（不需要响应的命令）已经送出
CTIMEOUT	响应超时
CMDACT	正在发送命令

CRC 发生器计算 CRC 码之前所有位的 CRC 校验和，包括开始位、发送位、命令索引和命令参数（或卡状态）。对于长响应格式，CRC 校验和计算的是 CID 或 CSD 的前 120 位；注意，长响应格式中的开始位、传输位和 6 个保留位不参与 CRC 计算。

CRC 校验和是一个 7 位的数值：CRC[6:0]=余数[（M（x）*x^7）/G（x）]

其中：

- G（x）=x^7+x^3+1
- 短响应格式时，M（x）=（开始位）*x^{39}+…+（CRC 前的最后一位）*x^0
- 长响应格式时，M（x）=（开始位）*x^{119}+…+（CRC 前的最后一位）*x^0

29.3.1.4 数据通道

数据通道子单元在主机与卡之间传输数据。下图是数据通道的框图。

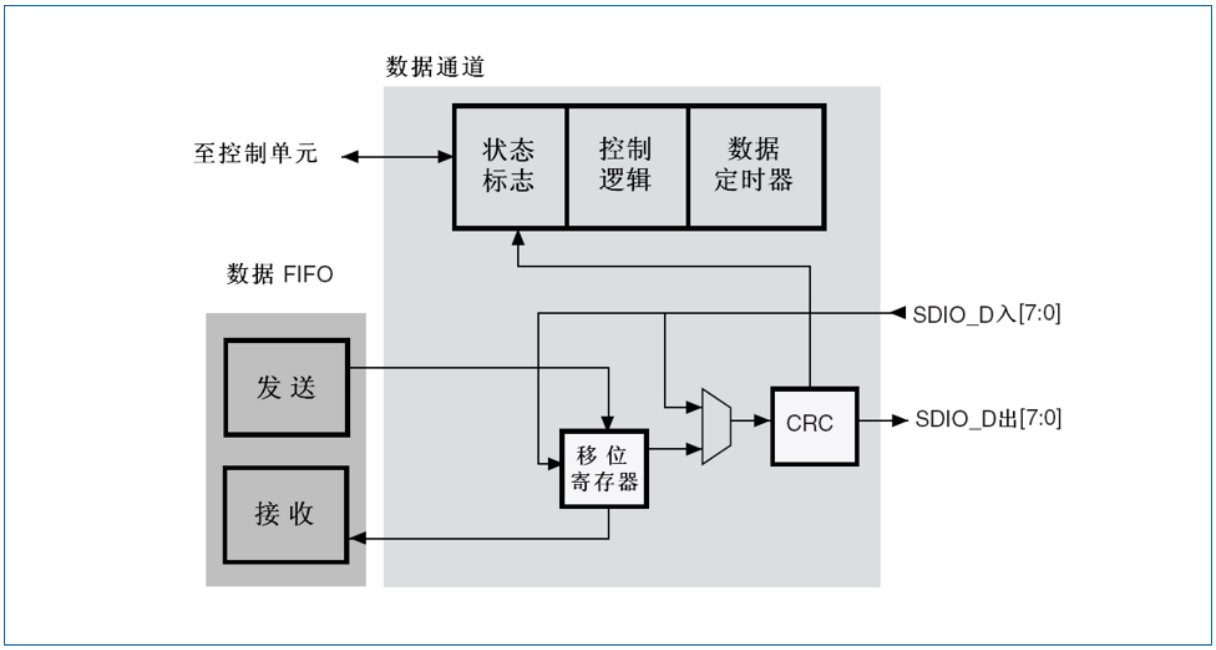


图 29-12 数据通道

时钟控制寄存器可以配置卡的数据总线宽度。

- 如果选择了 4 位总线模式，则每个时钟周期四条数据信号线（SDIO_D[3:0]）上将传输 4 位数据；
- 如果选择了 8 位总线模式，则每个时钟周期八条数据信号线（SDIO_D[7:0]）上将传输 8 位数据；
- 如果没有选择宽总线模式，则每个时钟周期只在 SDIO_D0 上传输 1 位数据。

根据传输的方向（发送或接收），使能时数据通道状态机（DPSM）将进入 Wait_S 或 Wait_R 状态：

- 发送：DPSM 进入 Wait_S 状态。如果发送 FIFO 中有数据，则 DPSM 进入发送状态，同时数据通道子单元开始向卡发送数据。
- 接收：DPSM 进入 Wait_R 状态并等待开始位；当收到开始位时，DPSM 进入接收状态，同时数据通道子单元开始从卡接收数据。

数据通道状态机（DPSM）

DPSM 工作在 SDIO_CK 频率，卡总线信号与 SDIO_CK 的上升沿同步。DPSM 有 6 个状态，如下图所示：

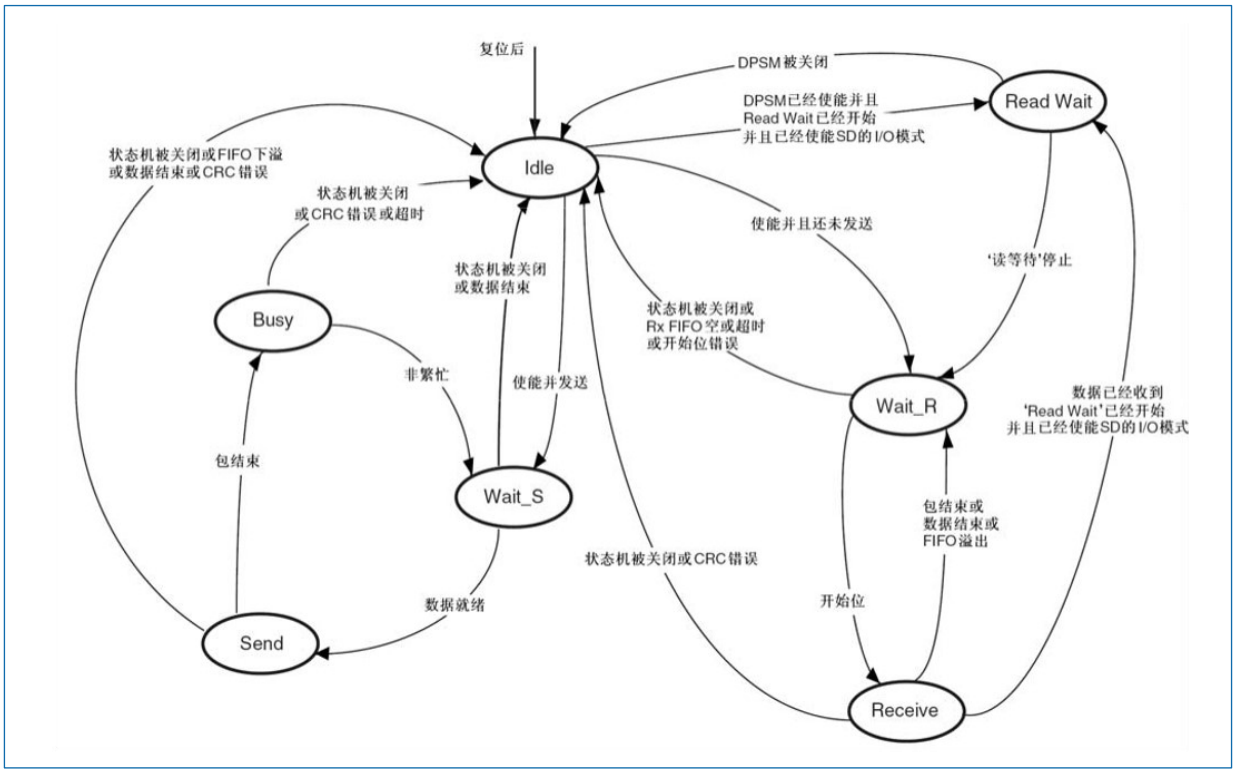


图 29-13 数据通道状态机（DPSM）

- 空闲（Idle）：数据通道不工作，SDIO_D[7:0]输出处于高阻状态。当写入数据控制寄存器并设置使能位时，DPSM 为数据计数器加载新的数值，并依据数据方向位进入 Wait_S 或 Wait_R 状态。
- Wait_R：如果数据计数器等于 0，当接收 FIFO 为空时 DPSM 进入到空闲（Idle）状态。如果数据计数器不等于 0，DPSM 等待 SDIO_D 上的开始位。如果 DPSM 在超时之前接收到一个开始位，它会进入接收（Receive）状态并加载数据块计数器。如果 DPSM 在检测到一个开始位前出现超时，或发生开始位错误，DPSM 将进入空闲状态并设置超时状态标志。
- 接收（Receive）：接收到的串行数据被组合为字节并写入数据 FIFO。根据数据控制寄存器中传输模式位的设置，数据传输模式可以是块传输或流传输：
  - 在块模式下，当数据块计数器达到 0 时，DPSM 等待接收 CRC 码，如果接收到的代码与内部产生的 CRC 码匹配，则 DPSM 进入 Wait_R 状态，否则设置 CRC 失败状态标志同时 DPSM 进入到空闲状态。
  - 在流模式下，当数据计数器不为 0 时，DPSM 接收数据；当计数器为 0 时，将移位寄存器中的剩余数据写入数据 FIFO，同时 DPSM 进入 Wait_R 状态。如果产生了 FIFO 上溢错误，DPSM 设置 FIFO 的错误标志并进入空闲状态。
- Wait_S：如果数据计数器为 0，DPSM 进入空闲状态；否则 DPSM 等待数据 FIFO 空标志消失后，进入发送状态。

注意：DPSM 会在 Wait_S 状态保持至少 2 个时钟周期，以满足 N_{WR} 的时序要求，N_{WR} 是接收到卡的响应至主机开始数据传输的间隔。

- 发送（Send）：DPSM 开始发送数据到卡设备。根据数据控制寄存器中传输模式位的设置，数据传输模式可以是块传输或流传输：
  - 在块模式下，当数据块计数器达到 0 时，DPSM 发送内部产生的 CRC 码，然后是结束位，并进入繁忙状态。
  - 在流模式下，当使能位为高同时数据计数器不为 0 时，DPSM 向卡设备发送数据，然后进入空闲状态。如果产生了 FIFO 下溢错误，DPSM 设置 FIFO 的错误标志并进入空闲状态。
- 繁忙（Busy）：DPSM 等待 CRC 状态标志：
  - 如果没有接收到正确的 CRC 状态，则 DPSM 进入空闲状态并设置 CRC 失败状态标志。

- 如果接收到正确的 CRC 状态，则当 SDIO_D0 不为低时（卡不繁忙）DPSM 进入 Wait_S 状态。
- 当 DPSM 处于繁忙状态时发生了超时，DPSM 则设置数据超时标志并进入空闲状态。
- 当 DPSM 处于 Wait_R 或繁忙状态时，数据定时器被使能，并能够产生数据超时错误：
  - 发送数据时，如果 DPSM 处于繁忙状态超过程序设置的超时间隔，则产生超时。
  - 接收数据时，如果未收完所有数据，并且 DPSM 处于 Wait_R 状态超过程序设置的超时间隔，则产生超时。

- 数据：数据可以从主机传送到卡，也可以反向传输。数据在数据线上传输。数据存储在一个 32 字的 FIFO 中，每个字为 32 位宽。

表 29-6 数据令牌格式

说明	开始位	数据	CRC16	结束位
块数据	0	-	有	1
流数据	0	-	无	1

29.3.1.5 数据 FIFO

数据 FIFO（先进先出）子单元是一个具有发送和接收单元的数据缓冲区。

FIFO 包含一个每字 32 位宽、共 32 个字的数据缓冲区，和发送与接收电路。因为数据 FIFO 工作在 AHB 时钟区域（HCLK/2），所有来自 SDIO 时钟区域（SDIOCLK）子单元的信号都进行了重新同步。

依据 TXACT 和 RXACT 标志，可以关闭 FIFO、使能发送或使能接收。TXACT 和 RXACT 由数据通道子单元设置而且是互斥的：

- 当 TXACT 有效时，发送 FIFO 代表发送电路和数据缓冲区
- 当 RXACT 有效时，接收 FIFO 代表接收电路和数据缓冲区

发送 FIFO

当使能了 SDIO 的发送功能，数据可以通过 AHB 接口写入发送 FIFO。通过 32 个连续的地址可以访问发送 FIFO。发送 FIFO 中有一个数据输出寄存器，包含读指针指向的数据字。当数据通道子单元对移位寄存器装载数值后，它移动读指针至下个数据并输出数据。如果未使能发送 FIFO，所有的状态标志均处于无效状态。当发送数据时，数据通道子单元设置 TXACT 为有效。

表 29-7 发送 FIF0 状态标志

标志	说明
TXFIFOF	当所有 32 个发送 FIFO 字都有有效的数据时，该标志为高。
TXFIFOE	当所有 32 个发送 FIFO 字都没有有效的数据时，该标志为高。
TXFIFOHE	当 8 个或更多发送 FIFO 字为空时，该标志为高。该标志可以作为 DMA 请求。
TXDAVL	当发送 FIFO 包含有效数据时，该标志为高。该标志的意思刚好与 TXFIFOE 相反。
TXUNDERR	当发生下溢错误时，该标志为高。写入 SDIO 清除寄存器时清除该标志。

接收 FIFO

当数据通道子单元接收到一个数据字，它会把数据写入 FIFO，写操作结束后，写指针自动加一；在另一端，有一个读指针始终指向 FIFO 中的当前数据。如果关闭了接收 FIFO，所有的状态标志会被清除，读写指针也被复位。在接收到数据时数据通道子单元设置 RXACT。下表列出了接收 FIFO 的状态标志。通

过 32 个连续的地址可以访问接收 FIFO。

表 29-8 接收 FIFO 状态标志

标志	说明
RXFIFOF	当所有 32 个接收 FIFO 字都有有效的数据时，该标志为高。
RXFIFOE	当所有 32 个接收 FIFO 字都没有有效的数据时，该标志为高。
RXFIFOHF	当 8 个或更多接收 FIFO 字有有效的数据时，该标志为高。该标志可以作为 DMA 请求。
RXDAVL	当接收 FIFO 包含有效数据时，该标志为高。该标志的意思刚好与 RTXFIFOE 相反。
RXOVERR	当发生上溢错误时，该标志为高。写入 SDIO 清除寄存器时清除该标志。

29.3.2 SDIOAHB 接口

AHB 接口产生中断和 DMA 请求，并访问 SDIO 适配器寄存器和数据 FIFO。它包含一个数据通道、寄存器译码器和中断/DMA 控制逻辑。

29.3.2.1 SDIO 中断

只要选中的状态标志中有一个为高，中断控制逻辑产生中断请求。有一个屏蔽寄存器用于选择可以产生中断的条件。如果设置了相应的屏蔽标志，则对应的状态标志可以产生中断。

29.3.2.2 SDIO/DMA 接口：在 SDIO 和存储器之间数据传输的过程

在下面的例子中，主机控制器使用 CMD24（WRITE_BLOCK）从主机传送 512 字节到 MMC 卡。DMA 控制器用于从存储器向 SDIO 的 FIFO 填充数据。

1. 执行卡识别过程
2. 提高 SDIO_CK 频率
3. 发送 CMD7 命令选择卡
4. 按下述步骤配置 DMA2：
   A. 使能 DMA2 控制器并清除所有挂起的中断。
   B. 设置 DMA2 通道 4 的源地址寄存器为存储器缓冲区的基地址，DMA2 通道 4 的目标地址寄存器为 SDIO_FIFO 寄存器的地址。
   C. 设置 DMA2 通道 4 控制寄存器（存储器递增，非外设递增，外设和源的数据宽度为字宽度）。
   D. 使能 DMA2 通道 4。
5. 发送 CMD24（WRITE_BLOCK），操作如下：
   A. 设置 SDIO 数据长度寄存器（SDIO 数据时钟寄存器应该在执行卡识别过程之前设置好）。
   B. 设置 SDIO 参数寄存器为卡中需要传送数据的地址。
   C. 设置 SDIO 命令寄存器：CMDINDEX 置为 24（WRITE_BLOCK）；WAITRESP 置为 1（SDIO 卡主机等待响应）；CPSMEN 置为 1（使能 SDIO 卡主机发送命令），保持其它域为它们的复位值。
   D. 等待 SDIO_STA[6]=CMDREND 中断，然后设置 SDIO 数据寄存器：DTEN 置为 1（使能 SDIO 卡主机发送数据）；DTDIR 置为 0（控制器至卡方向）；DTMODE 置为 0（块数据传送）；DMAEN 置为 1（使能 DMA）；DBLOCKSIZE 置为 9（512 字节）；其它域不用设置。
   E. 等待 SDIO_STA[10]=DBCKEND。

6. 查询 DMA 通道的使能状态寄存器，确认没有通道仍处于使能状态。

29.4 卡功能描述

29.4.1 卡识别模式

在卡识别模式，主机复位所有的卡、检测操作电压范围、识别卡并为总线上每个卡设置相对地址（RCA）。在卡识别模式下，所有数据通信只使用命令信号线（CMD）。

29.4.2 卡复位

GO_IDLE_STATE 命令（CMD0）是一个软件复位命令，它把多媒体卡和 SD 存储器置于空闲状态。IO_RW_DIRECT 命令（CMD52）复位 SDI/O 卡。上电后或执行 CMD0 后，所有卡的输出总线驱动器都处于高阻状态，同时所有卡都被初始化至一个默认的相对卡地址（RCA=0x0001）和默认的驱动器寄存器设置（最低的速度，最大的电流驱动能力）。

29.4.3 操作电压范围确认

所有的卡都可以使用任何规定范围内的电压与 SDIO 卡主机通信，可支持的最小和最大电压 VDD 数值由卡上的操作条件寄存器（OCR）定义。

卡存储了位于内部存储器中的识别号（CID）和卡特定数据（CSD），且仅能在数据传输 VDD 条件下传送这些信息。当 SDIO 卡主机模块与卡的 VDD 范围不一致时，卡将不能完成识别周期，也不能发送 CSD 数据；因此，在 VDD 范围不匹配时，SDIO 卡主机可以用下面几个特殊命令去识别和拒绝卡：SEND_OP_COND（CMD1）、SD_APP_OP_COND（SD 存储卡的 ACMD41）和 IO_SEND_OP_COND（SDI/O 卡的 CMD5）。SDIO 卡主机在执行这几个命令时会发送所需的 VDD 电压窗口。不能在指定的电压范围进行数据传输的卡，将从总线断开并进入非激活状态。

使用这些不包含电压范围作为操作数的命令，SDIO 卡主机能够查询每个卡并在确定公共的电压范围前，把不在此范围内的卡置于非激活状态。当 SDIO 卡主机能够选择公共的电压范围或用户需要知道卡是否能用时，SDIO 卡主机可以进行这样的查询。

29.4.4 卡识别过程

多媒体卡和 SD 卡的卡识别过程是有区别的；

- 对于多媒体卡，卡识别过程以时钟频率 F_{od} 开始，所有 SDIO_CMD 输出为开漏驱动，允许在这个过程中的卡的并行连接，识别过程如下：

1. 总线被激活。
2. SDIO 卡主机广播发送 SEND_OP_COND（CMD1）命令，并接收操作条件。
3. 得到的响应是所有卡的操作条件寄存器内容的“线与”。
4. 不兼容的卡会被置于非激活状态。
5. SDIO 卡主机广播发送 ALL_SEND_CID（CMD2）至所有激活的卡。
6. 所有激活的卡同时串行地发送它们的 CID 号，那些检测到输出的 CID 位与命令线上的数据不相符的卡会停止发送，并等待下一个识别周期。最终只有一个卡能够成功地传送完整的 CID 至 SDIO 卡主机并进入识别状态。
7. SDIO 卡主机发送 SET_RELATIVE_ADDR（CMD3）命令至这个卡，这个新的地址被称为相对卡地址（RCA），它比 CID 短，用于对卡寻址。至此，这个卡转入待机状态，并不再响应新的识别过程，同时它的输出驱动从开漏转变为推挽模式。

8. SDIO 卡主机重复上述步骤 5 至 7，直到收到超时条件。

- 对于 SD 卡而言，卡识别过程以时钟频率 F_{od} 开始，所有 SDIO_CMD 输出为推挽驱动而不是开漏驱动，识别过程如下：

1. 总线被激活。
2. SDIO 卡主机广播发送 SD_APP_OP_COND（ACMD41）命令。
3. 所有卡以操作条件寄存器的内容作为响应。
4. 不兼容的卡会被置于非激活状态。
5. SDIO 卡主机广播发送 ALL_SEND_CID（CMD2）至所有激活的卡。
6. 所有激活的卡发送回它们唯一卡识别号（CID）并进入识别状态。
7. SDIO 卡主机发送 SET_RELATIVE_ADDR（CMD3）命令和一个地址到一个激活的卡，这个新的地址被称为相对卡地址（RCA），它比 CID 短，用于对卡寻址。至此，这个卡转入待机状态。SDIO 卡主机可以再次发送该命令更改 RCA，该 RCA 将是最后一次的赋值。
8. SDIO 卡主机对所有激活的卡重复上述步骤 5 至 7。

- 对于 SDI/O 卡而言，卡识别过程如下：

1. 总线被激活。
2. SDIO 卡主机发送 IO_SEND_OP_COND（CMD5）命令。
3. 所有卡以操作条件寄存器的内容作为响应。
4. 不兼容的卡会被置于非激活状态。
5. SDIO 卡主机发送 SET_RELATIVE_ADDR（CMD3）命令和一个地址到一个激活的卡，这个新的地址被称为相对卡地址（RCA），它比 CID 短，用于对卡寻址。至此，这个卡转入待机状态。SDIO 卡主机可以再次发送该命令更改 RCA，卡的 RCA 将是最后一次的赋值。

29.4.5 写数据块

执行写数据块命令（CMD24-27）时，主机把一个或多个数据块从主机传送到卡中，同时在每个数据块的末尾传送一个 CRC 码。一个支持写数据块命令的卡始终能够接收由 WRITE_BL_LEN 定义的数据块。如果 CRC 校验错误，卡通过 SDIO_D 信号线指示错误，传送的数据被丢弃而不被写入，所有后续（在多块写模式下）传送的数据块将被忽略。

如果主机传送部分数据，而累计的数据长度未与数据块对齐，当不允许块错位（未设置 CSD 的参数 WRITE_BLK_MISALIGN），卡将在第一个错位的块之前检测到块错位错误（设置状态寄存器中的 ADDRESS_ERROR 错误位）。当主机试图写一个写保护区域时，写操作也会被中止，此时卡会设置 WP_VIOLATION 位。

设置 CID 和 CSD 寄存器不需要事先设置块长度，传送的数据也是通过 CRC 保护的。如果 CSD 或 CID 寄存器的部分是存储在 ROM 中，则这个不能更改的部分必须与接收缓冲区的对应部分相一致，如果有不一致之处，卡将报告一个错误同时不修改任何寄存器的内容。有些卡需要长的甚至不可预计的时间完成写一个数据块，在接收一个数据块并完成 CRC 检验后，卡开始写操作，如果它的写缓冲区已经满并且不能再从新的 WRITE_BLOCK 命令接受新的数据时，它会把 SDIO_D 信号线拉低。主机可以在任何时候使用 SEND_STATUS（CMD13）查询卡的状态，卡将返回当前状态。READY_FOR_DATA 状态位指示卡是否可以接受新的数据或写操作是否还在进行。主机可以使用 CMD7（选择另一个卡）取消选中某个卡，而把这个卡置于断开状态，这样可以释放 SDIO_D 信号线而不中断未完成的写操作；当重新选择一个卡，如果写操作仍然在进行并且写缓冲区仍不能使用，它会重新通过拉低 SDIO_D 信号线指示忙的状态。

29.4.6 读数据块

在读数据块模式下，数据传输的基本单元是数据块，它的大小在 CSD 中（READ_BL_LEN）定义。如果设置了 READ_BL_PARTIAL，同样可以传送较小的数据块，较小数据块的开始和结束地址完全包含在一个物理块中，由 READ_BL_LEN 定义物理块的大小。为保证数据传输的完整性，每个数据块后都附有一个 CRC 校验码。CMD17（READ_SINGLE_BLOCK）启动一次读数据块操作，在传输结束后卡返回到发送状态。

CMD18（READ_MULTIPLE_BLOCK）启动一次连续多个数据块的读操作。主机可以在多数据块读操作的任何时候中止操作，而不管操作的类型。发送停止传输命令即可中止操作。

如果在多数据块任一类型的读操作中，卡检测到错误（例如：越界、地址错位或内部错误），它将停止数据传输并保持在数据状态；此时主机必须发送停止传输命令中止操作。在停止传输命令的响应中报告读错误。

如果主机发送停止传输命令时，卡已经传输完一个确定数目的多个数据块操作中的最后一个数据块，因为此时卡已经不在数据状态，主机会得到一个非法命令的响应。如果主机传输部分数据块，而累计的数据长度不能与物理块对齐，同时不允许块错位，卡会在出现第一个未对齐的块时检测出一个块对齐错误，并在状态寄存器中设置 ADDRESS_ERROR 错误标志。

29.4.7 数据流操作，数据流写入和数据流读出（只适用于多媒体卡）

在数据流模式，数据按字节传输，同时每个数据块后没有 CRC。

数据流写（只适用于多媒体卡）

WRITE_DAT_UNTIL_STOP（CMD20）开始从 SDIO 卡主机至卡的数据传输，从指定的地址开始连续传输直到 SDIO 卡主机发出一个停止命令。如果允许部分数据块传输（设置了 CSD 参数 WRITE_BL_PARTIAL），则数据流可以在卡的地址空间中的任意地址开始和停止，否则数据流只能在数据块的边界开始和停止。因为传输的数据数目没有事先设定，不能使用 CRC 校验。如果发送数据时达到了存储器的最大地址，也 SDIO 卡主机没有发送停止命令，随后传输的数据会被丢弃。

数据流写操作的最大时钟频率可以通过下式计算：

$$\text{Maximumspeed=Min（TRANSPEED）}\frac{\text{（8x2}^{writebllen}\text{）（-NSAC）}}{\text{TAACxR2WFACTOR}}$$

其中：

- Maximumspeed=最大写频率
- TRANSPEED=最大数据传输率
- writebllen=最大写数据块长度
- NSAC=以 CLK 周期计算的数据读操作时间 2
- TAAC=数据读操作时间 1
- R2WFACTOR=写速度因子

如果主机试图使用更高的频率，卡可能不能处理数据并停止编程，同时在状态寄存器中设置 OVERRUN 错误位，丢弃所有随后传输的数据并（在接收数据状态）等待停止命令。如果主机试图

写入一个写保护区域，写操作将被中止，同时卡将设置 WP_VIOLATION 位。

数据流读（只适用于多媒体卡）

READ_DAT_UNTIL_STOP（CMD11）控制数据流数据传输。

这个命令要求卡从指定的地址发送数据，直到 SDIO 卡主机发送 STOP_TRANSMISSION（CMD12）命令。因为串行命令传输的延迟，停止命令的执行会有延迟，数据传送会在停止命令的结束位后停止。如果发送数据时达到了存储器的最大地址，SDIO 卡主机没有发送停止命令，随后传输的数据将是无效数

据。

数据流读操作的最大时钟频率可以通过下式计算：

$$\text{Maximumspeed=Min（TRANSPEED）}\frac{\text{（8x2}^{readbllen}\text{）（-NSAC）}}{\text{TAACxR2WFACTOR}}$$

其中：

- Maximumspeed=最大写频率
- TRANSPEED=最大数据传输率
- readbllen=最大读数据块长度
- NSAC=以 CLK 周期计算的数据读操作时间 2
- TAAC=数据读操作时间 1
- R2WFACTOR=写速度因子

如果主机试图使用高的频率，卡将不能处理数据传输，此时卡在状态寄存器中设置 UNDERRUN 错误位，中止数据传输并在数据状态等待停止命令。

29.4.8 擦除：成组擦除和扇区擦除

多媒体卡的擦除单位是擦除组，擦除组是以写数据块计算，写数据块是卡的基本写入单位。擦除组的大小是卡的特定参数，在 CSD 中定义。主机可以擦除一个连续范围的擦除组，开始擦除操作有三个步骤。

首先，主机使用 ERASE_GROUP_START（CMD35）命令定义连续范围的开始地址，然后使用 ERASE_GROUP_END（CMD36）命令定义连续范围的结束地址，最后发送擦除命令 ERASE（CMD38）开始擦除操作。擦除命令的地址域是以字节为单位的擦除组地址。卡会舍弃未与擦除组大小对齐的部分，把地址边界对齐到擦除组的边界。

如果未按照上述步骤收到了擦除命令，卡在状态寄存器中设置 ERASE_SEQ_ERROR 位，并重新等待第一个步骤。

如果收到了除 SEND_STATUS 和擦除命令之外的其它命令，卡在状态寄存器中设置 ERASE_RESET 位，解除擦除序列并执行新的命令。

如果擦除范围包含了写保护数据块，这些块不被擦除，只有未保护的块被擦除，同时卡在状态寄存器中设置 WP_ERASE_SKIP 状态位。在擦除过程中，卡拉低 SDIO_D 信号来指示擦除操作正在进行。实际的擦除时间可能很长，主机可以使用 CMD7 解除卡的选择。

29.4.9 宽总线选择和解除选择

可以通过 SET_BUS_WIDTH（ACMD6）命令选择或取消选择宽总线（4 位总线宽度）操作模式，上电后或 GO_IDLE_STATE（CMD0）命令后默认的总线宽度为 1 位。SET_BUS_WIDTH（ACMD6）命令仅在传输状态时有效，也就是说只有在使用 SELECT/DESELECT_CARD（CMD7）命令选择了卡后才能改变总线宽度。

29.4.10 保护管理

SDIO 卡主机模块支持三种保护方式：

- 内部卡写保护（卡内管理）
- 机械写保护开关（仅由 SDIO 卡主机模块管理）
- 密码管理的卡锁操作

内部卡的写保护

卡的数据可以被保护不被覆盖或擦除。在 CSD 中永久或临时地设置写保护位，生产厂商或内容提供

商可以永久地对整个卡施行写保护。对于支持在 CSD 中设置 WP_GRP_ENABLE 位从而提供一组扇区写保护的卡，部分数据可以被保护，写保护可以通过程序改变。写保护的基本单位是 CSD 参数 WP_GRP_SIZE 个扇区。SET_WRITE_PROT 和 CLR_WRITE_PROT 命令控制指定组的保护，SEND_WRITE_PROT 命令与单数据块读命令类似，卡送出一个包含 32 个写保护位（代表从指定地址开始的 32 个写保护组）的数据块，跟着一个 16 位的 CRC 码。写保护命令的地址域是一个以字节为单位的组地址。

卡将截断所有组大小以下的地址。

机械写保护开关

在卡的侧面有一个机械的滑动开关，允许用户设置或清除卡的写保护。当滑动开关被滑动到小窗口打开的位置时，卡处于写保护状态，当滑动开关被滑动到小窗口关闭的位置时，可以更改卡中内容。在卡的插槽上的对应部位也有一个开关告知 SDIO 卡主机模块，卡是否处于写保护状态。SDIO 卡主机模块负责卡的写保护，卡的内部电路不知道写保护开关的位置。

密码保护

密码保护功能允许 SDIO 卡主机模块使用密码对卡实行上锁或解锁。密码存储在 128 位的 PWD 寄存器中，它的长度设置在 8 位的 PWD_LEN 寄存器中。这些寄存器是不可挥发的，即掉电后它们的内容不丢失。已上锁的卡能够响应和执行相应的命令，即允许 SDIO 卡主机模块执行复位、初始化和查询状态等操作，但不允许访问卡中的数据。当设置了密码后（即 PWD_LEN 的数值不为 0），上电后卡自动处于上锁状态。正如 CSD 和 CID 寄存器写命令，上锁/解锁命令仅在传输状态下有效，在这个状态下，命令中没有地址参数，但卡已经被选中。卡的上锁/解锁命令具有单数据块写命令的结构和总线操作类型，传输的数据块包含所有命令所需要的信息（密码设置模式、PWD 内容和上锁/解锁指示）。在发送卡的上锁/解锁命令之前，命令数据块的长度由 SDIO 卡主机模块定义，命令结构示于表 29-2。

位的设置如下：

- ERASE：设置该位将执行强制擦除，所有其它位必须为 0，只发送命令字节。
- LOCK_UNLOCK：设置该位锁住卡，LOCK_UNLOCK 与 SET_PWD 可以同时设置，但不能与 CLR_PWD 同时设置。
- CLR_PWD：设置该位清除密码数据。
- SET_PWD：设置该位将密码数据保存至存储器。
- PWD_LEN：以字节为单位定义密码的长度。
- PWD：密码（依不同的命令，新的密码或正在使用的密码）。

以下几节列出了设置/清除密码、上锁/解锁和强制擦除的命令序列。

设置密码

1. 选择一个卡（SELECT/DESELECT_CARD，CMD7）。
2. 定义要在 8 位的卡上锁/解锁模式下发送的数据块长度（SET_BLOCKLEN，CMD16），8 位的 PWD_LEN，新密码的字节数目。当更换了密码后，发送命令的数据块长度必须同时考虑新旧密码的长度。
3. 以合适的数据块长度在数据线上发送 LOCK/UNLOCK（CMD42）命令，并包含 16 位的 CRC 码。数据块包含了操作模式（SET_PWD=1）、长度（PWD_LEN）和密码（PWD）。当更换了密码后，长度数值（PWD_LEN）包含了新旧两个密码的长度，PWD 域包含了旧的密码（正在使用的）和新的密码。
4. 当旧的密码匹配后，新的密码和它的长度被分别存储在 PWD 和 PWD_LEN 域。如果送出的旧密码与期望的密码（长度或内容）不吻合，则设置状态寄存器中的 LOCK_UNLOCK_FAILED 错误位，同时密码不变。

密码长度域（PWD_LEN）指示当前是否设置了密码，如果该域为非零，则表示使用了密码，卡在上

电时自动上锁。在不断电的情况下，如果设置了密码，可以通过设置 LOCK_UNLOCK 位或发送一个额外的上锁命令，立即锁住卡。

清除密码

1. 选择一个卡（SELECT/DESELECT_CARD，CMD7）。
2. 定义要在 8 位的卡上锁/解锁模式下发送的数据块长度（SET_BLOCKLEN，CMD16），8 位的 PWD_LEN，当前使用密码的字节数目。
3. 当密码匹配后，PWD 域被清除同时 PWD_LEN 被设为 0。如果送出的密码与期望的密码（长度或内容）不吻合，则设置状态寄存器中的 LOCK_UNLOCK_FAILED 错误位，同时密码不变。

卡上锁

1. 选择一个卡（SELECT/DESELECT_CARD，CMD7）。
2. 定义要在 8 位的卡上锁/解锁模式（见表 142 的字节 0）下发送的数据块长度（SET_BLOCKLEN，CMD16），8 位的 PWD_LEN，和当前密码的字节数目。
3. 以合适的数据块长度在数据线上发送 LOCK/UNLOCK（CMD42）命令，并包含 16 位的 CRC 码。数据块包含了操作模式（LOCK_UNLOCK=1）、长度（PWD_LEN）和密码（PWD）。
4. 当密码匹配后，卡被上锁并则设置状态寄存器中的 CARD_IS_LOCKED 状态位。如果送出的密码与期望的密码（长度或内容）不吻合，则设置状态寄存器中的 LOCK_UNLOCK_FAILED 错误位，同时上锁操作失败。

设置密码和为卡上锁可以在同一个操作序列中进行，此时 SDIO 卡主机模块按照前述的步骤设置密码，但在发送新密码命令的第 3 步需要设置 LOCK_UNLOCK 位。

如果曾经设置过密码（PWD_LEN 不为 0），卡会在上电复位时自动地上锁。对已经上锁的卡执行上锁操作或对没有密码的卡执行上锁操作会导致失败，并设置状态寄存器中的 LOCK_UNLOCK_FAILED 错误位。

卡解锁

1. 选择一个卡（SELECT/DESELECT_CARD，CMD7）。
2. 定义要在 8 位的卡上锁/解锁模式（见表 142 的字节 0）下发送的数据块长度（SET_BLOCKLEN，CMD16），8 位的 PWD_LEN，和当前密码的字节数目。
3. 以合适的数据块长度在数据线上发送 LOCK/UNLOCK（CMD42）命令，并包含 16 位的 CRC 码。数据块包含了操作模式（LOCK_UNLOCK=0）、长度（PWD_LEN）和密码（PWD）。
4. 当密码匹配后，卡锁被解除，同时状态寄存器中的 CARD_IS_LOCKED 位被清除。如果送出的密码与期望的密码（长度或内容）不吻合，则设置状态寄存器中的 LOCK_UNLOCK_FAILED 错误位，同时卡仍保持上锁状态。

解锁状态只在当前的供电过程中有效，只要不清除 PWD 域，下次上电后卡会被自动上锁。

试图对已经解了锁的卡执行解锁操作会导致操作失败，并设置状态寄存器中的 LOCK_UNLOCK_FAILED 错误位。

强制擦除

如果用户忘记了密码（PWD 的内容），可以在清除卡中的所有内容后使用卡。强制擦除操作擦除所有卡中的数据和密码。

1. 选择一个卡（SELECT/DESELECT_CARD，CMD7）。
2. 设置发送的数据块长度（SET_BLOCKLEN，CMD16）为 1，仅发送 8 位的卡上锁/解锁字节（见

表 29-22 的字节 0）。

3. 以合适的数据块长度在数据线上发送 LOCK/UNLOCK（CMD42）命令，并包含 16 位的 CRC 码。数据块包含了操作模式（ERASE=1）所有其它位为 0。

4. 当 ERASE 位是数据域中仅有的位时，卡中的所有内容将被擦除，包括 PWD 和 PWD_LEN 域，同时卡不再被上锁。如果有任何其它位不为 0，则设置状态寄存器中的 LOCK_UNLOCK_FAILED 错误位，卡中的数据保持不变，同时卡仍保持上锁状态。

试图对已经解了锁的卡执行擦除操作会导致操作失败，并设置状态寄存器中的 LOCK_UNLOCK_FAILED 错误位。

29.4.11 卡状态寄存器

响应格式 R1 包含了一个 32 位的卡状态域，这个域是用于向卡主机发送卡的状态信息（这些信息有可能存在本地的状态寄存器中）。除非特别说明，卡返回的状态始终是与之前的命令相关的。

下表定义了不同的状态信息。表中有关类型和清除条件域的缩写定义如下：

类型：

- E：错误位
- S：状态位
- R：检测位，并依据实际的命令响应而设置
- X：检测位，在命令的执行中设置。SDIO 卡主机通过发送状态命令读出这些位而查询卡的状态。

清除条件：

- A：依据卡的当前状态
- B：始终与之前的命令相关。接收到正确的命令即可清除（具有一个命令的延迟）。
- C：读即可清除

表 29-9 卡状态

位	名称	类型	数值	说明	清除条件
31	ADDRESS_OUT_OF_RANGE	ERX	‘0’=无错误 ‘1’=错误	命令中的地址参数超出了卡的允许范围。 一个多数据块或数据流读/写操作（即使从一个合法的地址开始）试图读或写超出卡的容量的部分。	C
30	ADDRESS_MISALIGN		‘0’=无错误 ‘1’=错误	命令中的地址参数（与当前的数据块长度对照）定义的第一个数据块未与卡的物理块对齐。 一个多数据块或数据流读/写操作（即使从一个合法的地址开始）试图读或写未与物理块对齐的数据块。	C
29	BLOCK_LEN_ERROR		‘0’=无错误 ‘1’=错误	SET_BLOCKLEN 命令的参数超出了卡的最大允许范围，或先前定义的数据块长度对于当前命令来说是非法的（例如：主机发出一个写命令，当前的块长度小于卡所允许的最小长度，同时又不允许写入部分数据块）。	C
28	ERASE_SEQ_ERROR		‘0’=无错误 ‘1’=错误	发送擦除命令的顺序错误。	C
27	ERASE_PARAM	EX	‘0’=无错误	擦除时选择了非法的擦除组。	C

位	名称	类型	数值	说明	清除条件
			‘1’=错误		
26	WP_VIOLATION	EX	‘0’=无错误 ‘1’=错误	试图对一个写保护的数据块编程。	C
25	CARD_IS_LOCKED	SR	‘0’=卡未锁 ‘1’=卡已锁	当设置了该位，表示卡已经被锁住。	A
24	LOCK_UNLOCK_FAILED	EX	‘0’=无错误 ‘1’=错误	在上锁/解锁中有命令的顺序错误或检测到密码错误。	C
23	COM_CRC_ERROR	ER	‘0’=无错误 ‘1’=错误	之前的命令中 CRC 校验错误。	B
22	ILLEGAL_COMMAND	ER	‘0’=无错误 ‘1’=错误	对于当前的卡状态，命令非法。	B
21	CARD_ECC_FAILED	EX	‘0’=成功 ‘1’=失败	卡的内部实施了 ECC 校验，但在更正数据时失败。	C
20	CC_ERROR	ER	‘0’=无错误 ‘1’=错误	（标准中未定义）卡内部发生错误，与主机的命令无关。	C
19	ERROR	EX	‘0’=无错误 ‘1’=错误	产生了与执行上一个主机命令相关的（标准中未定义）卡内部的错误（例如：读或写错误）。	C
18	保留	-	-	-	-
17	保留	-	-	-	-
16	CID/CSD_OVERWRITE	EX	‘0’=无错误 ‘1’=错误	可以是任何一个下述的错误： ● 已经写入了 CID 寄存器，不能覆盖 ● CSD 的只读部分与卡的内容不匹配 ● 试图进行拷贝或永久写保护的反向操作，即恢复原状或解除写保护。	C
15	WP_ERASE_SKIP	EX	‘0’=未保护 ‘1’=已保护	遇到已经存在的写保护数据块，仅有部分地址空间被擦除。	C
14	CARD_ECC_DISABLED	SX	‘0’=允许 ‘1’=不允许	执行命令时没有使用内部的 ECC。	A
13	ERASE_RESET		‘0’=清除 ‘1’=设置	因为收到一个擦除顺序之外的命令（非 CMD35、CMD36、CMD38 或 CMD13 命令），进入擦除过程的序列被中止。	C
12:9	CURRENT_STATE	SR	0=空闲 1=就绪 2=识别 3=待机 4=发送 5=数据 6=接收	当收到命令时卡内状态机的状态。如果命令的执行导致状态的变化，这个变化将会在下个命令的响应中反映出来。这四个位按十进制数 0 至 15 解释。	B

位	名称	类型	数值	说明	清除条件
			7=编程 8=断开 9=忙测试 10~15=保留		
8	READY_FOR_DATA	SR	‘0’=未就绪 ‘1’=就绪	与总线上的缓冲器空的信号相对应。	
7	SWITCH_ERROR	EX	‘0’=无错误 ‘1’=转换错	卡没有按照 SWITCH 命令的要求转换到希望的模式。	B
6	保留	-	-	-	-
5	APP_CMD	SR	‘0’=不允许 ‘1’=允许	卡期望 ACMD，或指示命令已经被解释为 ACMD 命令。	C
4	保留给 SDI/O 卡				
3	AKE_SEQ_ERROR	ER	‘0’=无错误 ‘1’=错误	验证的顺序有错误。	C
2	保留给与应用相关的命令。				
1:0	保留给生产厂家的测试模式。				

29.4.12 SD 状态寄存器

SD 状态包含与 SD 存储器卡特定功能相关的状态位和一些与未来应用相关的状态位，SD 状态的长度是一个 512 位的数据块。收到 ACMD13 命令（CMD55，然后是 CMD13）后，这个寄存器的内容被传送到 SDIO 卡主机。只有卡处于传输状态时（卡已被选择）才能发送 ACMD13 命令。

下表定义了不同的 SD 状态寄存器信息。表中有关类型和清除条件域的缩写定义如下：

类型：

- E：错误位
- S：状态位
- R：检测位，并依据实际的命令响应而设置
- X：检测位，在命令的执行中设置。SDIO 卡主机通过发送状态命令读出这些位而查询卡的状态。

清除条件：

- A：依据卡的当前状态
- B：始终与之前的命令相关。接收到正确的命令即可清除（具有一个命令的延迟）。
- C：读即可清除

表 29-10 SD 状态

位	名称	类型	数值	说明	清除条件
511:510	DAT_BUS_WIDTH	SR	‘00’=1（默认） ‘01’=保留	由 SET_BUS_WIDTH 命令定义的当前数据总线宽度。	A

位	名称	类型	数值	说明	清除条件
			‘10’=4 位宽 ‘11’=保留		
509	SECURED_MODE	SR	‘0’=未处于保密模式 ‘1’=处于保密模式	卡处于保密操作模式（详见“SD 保密规范”）。	A
508:496	保留				
495:480	SD_CARD_TYPE	SR	‘00xxh’=在物理规范版本 1.01 ~2.00 的 SD 存储器卡（'x'表示任意值）。已定义的卡有： '0000'=通用 SD 读写卡 '0001'=SDROM 卡	这个域的低 8 位可以在未来定义 SD 存储卡的不同变种（每个位可以用于定义不同的 SD 类型）。高 8 位可以用于定义那些不遵守当前的 SD 物理层规范的 SD 卡。	A
479:448	SIZE_OF_PROTECTED_AREA	SR	受保护的区域大小（见以下说明）	（见以下说明）	A
447:440	SPEED_CLASS	SR	卡的速度类型（见以下说明）	（见以下说明）	A
439:432	PERFORMANCE_MOVE	SR	以 1MB/秒为单位的传输性能（见以下说明）	（见以下说明）	A
431:428	AU_SIZE	SR	AU 的大小（见以下说明）	（见以下说明）	A
427:424	保留	-	-	-	-
423:408	ERASE_SIZE	SR	一次可以擦除的 AU 数目	（见以下说明）	A
407:402	ERASE_TIMEOUT	SR	擦除 UNIT_OF_ERASE_AU 指定的范围的超时数值	（见以下说明）	A
401:400	ERASE_OFFSET	SR	在擦除时增加的固定偏移数值	（见以下说明）	A
399:312	保留				
311:0	保留给生产厂商				

SIZE_OF_PROTECTED_AREA

标准容量卡和高容量卡设置该位的方式不同。

- 对于标准容量卡，受保护区域的容量由下式计算：

受保护区域=SIZE_OF_PROTECTED_AREA*MULT*BLOCK_LEN

其中，SIZE_OF_PROTECTED_AREA 的单位是 MULT*BLOCK_LEN。

- 对于高容量卡，受保护区域的容量由下式计算：

受保护区域=SIZE_OF_PROTECTED_AREA

其中，SIZE_OF_PROTECTED_AREA 的单位是字节。

SPEED_CLASS

这 8 位指示速度的类型和可以通过计算 PW/2 的数值（PW 是写的性能）。

表 29-11 速度类型代码

SPEED_CLASS	数值定义
00h	类型 0
01h	类型 2
02h	类型 4
03h	类型 6
04h~FFh	保留

PERFORMANCE_MOVE

这 8 位以 1MB/秒为单位指示移动性能（Pm）。如果卡不用 RU（纪录单位）移动数据，应该认为 Pm 是无穷大。设置这个域为 FFh 表示无穷大。

表 29-12 移动性能代码

PERFORMANCE_MOVE	数值定义
00h	未定义
01h	1MB/秒
02h	2MB/秒
……	……
FEh	254MB/秒
FFh	无穷大

AU_SIZE

这 4 位指示 AU 的长度，数值是 16K 字节为单位 2 的幂次的倍数。

表 29-13 AU_SIZE 代码

AU_SIZE	数值定义
00h	未定义
01h	16KB
02h	32KB
03h	64KB
04h	128KB
05h	256KB
06h	512KB
07h	1MB
08h	2MB
09h	4MB

AU_SIZE	数值定义
Ah~Fh	保留

依据卡的容量，最大的 AU 长度由下表定义。卡可以在 RU 长度和最大的 AU 长度之间设置任意的 AU 长度。

表 29-14 最大的 AU 长度

容量	16MB~64MB	128MB~256MB	512MB	1GB~32GB
最大的 AU 长度	512KB	1MB	2MB	4MB

ERASE_SIZE

这个 16 位域给出了 N_{ERASE}，当 N_{ERASE} 个 AU 被擦除时，ERASE_TIMEOUT 定义了超时时间。主机应该确定适当的一次操作中擦除的 AU 数目，这样主机可以显示擦除操作的进度。如果该域为 0，则不支持擦除的超时计算。

表 29-15 ERASE_SIZE 代码

ERASE_SIZE	数值定义
0000h	不支持擦除的超时计算
0001h	1 个 AU
0002h	2 个 AU
0003h	3 个 AU
……	……
FFFFh	65535 个 AU

ERASE_TIMEOUT

这 6 位给出了 T_{ERASE}，当 ERASE_SIZE 指示的多个 AU 被擦除时，这个数值给出了从偏移量算起的擦除超时。ERASE_TIMEOUT 的范围可以定义到最多 63 秒，卡的生产商可以根据具体实现选择合适的 ERASE_SIZE 与 ERASE_TIMEOUT 的组合，先确定 ERASE_TIMEOUT 再确定 ERASE_SIZE。

表 29-16 擦除超时代码

ERASE_TIMEOUT	数值定义
00	不支持擦除的超时计算
01	1 秒
02	2 秒
03	3 秒
……	……
63	63 秒

ERASE_OFFSET

这 2 位给出了 T_{OFFSET}，当 ERASE_SIZE 和 ERASE_TIMEOUT 同为 0 时这个数值没有意义。

表 29-17 擦除偏移代码

ERASE_OFFSET	数值定义
0	0 秒
1	1 秒
2	2 秒
3	3 秒

29.4.13 SD 的 I/O 模式

29.4.13.1 SD 的 I/O 中断

为了让 SDI/O 卡能够中断 MMC/SD 模块，在 SD 接口上有一个具有中断功能的引脚：第 8 脚，在 4 位 SD 模式下这个脚是 SDIO_D1，卡用它向 MMC/SD 模块提出中断申请。对于每一个卡或卡内的功能，中断功能是可选的。SDI/O 的中断是电平有效，即在被识别并得到 MMC/SD 模块的响应之前，中断信号线必须保持有效电平（低），在中断过程结束后保持无效电平（高）。在 MMC/SD 模块服务了中断请求后，通过一个 I/O 写操作，写入适当的位到 SDI/O 卡的内部寄存器，即可清除中断状态位。所有 SDI/O 卡的中断输出是低电平有效，MMC/SD 模块在所有数据线（SDIO/D[3:0]）上提供上拉电阻。MMC/SD 模块在中断阶段对第 8 脚（SDIO_D/IRQ）采样并进行中断检测，其它时间该信号线上的数值将被忽略。

存储器操作和 I/O 操作都具有中断阶段，单个数据块操作的中断阶段定义与多个数据块传输操作的中断阶段定义不同。

29.4.13.2 SD 的 I/O 暂停和恢复

在一个多功能的 SD I/O 卡或同时具有 I/O 和存储器功能的卡中，多个设备（I/O 和存储器）共用 MMC/SD 总线。为了使 MMC/SD 模块中的多个设备能够共用总线，SD I/O 卡和复合卡可以有选择地实现暂停/恢复的概念；如果一个卡支持暂停/恢复，MMC/SD 模块能够暂时地停止一个功能或存储器的数据传输操作（暂停），借此让出总线给具有更高优先级的其它功能或存储器，在这个具有更高优先级的传输完成后，再恢复原先暂停的传输。支持暂停/恢复的操作是可选的。在 MMC/SD 总线上执行暂停/恢复操作有下述步骤：

1. 确定 SDIO_D[3:0]信号线的当前功能。
2. 请求低优先级或慢的操作暂停。
3. 等待暂停操作完成，确认设备已暂停。
4. 开始高优先级的传输。
5. 等待高优先级的传输结束。
6. 恢复暂停的操作。

29.4.13.3 SDI/O 读等待（ReadWait）

可选的读等待（RW）操作只适用于 SD 卡的 1 位或 4 位模式。读等待操作允许 MMC/SD 模块在一个卡正在读多个寄存器（IO_RW_EXTENDED,CMD53）时，要求它暂时停止数据传输，同时允许 MMC/SD 模块发送命令到 SD I/O 设备中的其他功能。判断一个卡是否支持读等待协议，MMC/SD 模块应该检测卡的内部寄存器。读等待的时间与中断阶段相关。

29.4.14 命令与响应

应用相关命令和通用命令

SD 卡主机模块系统是用于提供一个适用于多种应用类型的标准接口，但同时又要兼顾特定用户和应用的功能，因此标准中定义了两类通用命令：应用相关命令（ACMD）和通用命令（GEN_CMD）。

当卡收到 APP_CMD（CMD55）命令时，卡期待下一个命令是应用相关命令。应用相关命令（ACMD）具有普通 MMC 相同的格式结构，并可以使用相同的 CMD 号码，因为它是出现在 APP_CMD（CMD55）后面，所以卡把它识别为 ACMD 命令。如果跟随 APP_CMD（CMD55）之后不是一个已经定义的应用相关命令，则认为它是一个标准命令；例如：有一个 SD_STATUS（ACMD13）应用相关命令，如果在紧随 APP_CMD（CMD55）之后收到 CMD13，它将被解释为 SD_STATUS（ACMD13）；但是如果卡在紧随 APP_CMD（CMD55）之后收到 CMD7，而这个卡没有定义 ACMD7，则它将被解释为一个标准的 CMD7（SELECT/DESELECT_CARD）命令。

如果要使用生产厂商自定义的 ACMD，SD 卡主机需要做以下操作：

1. 发送 APP_CMD（CMD55）命令卡送回响应给 MMC/SD 卡模块，指示设置了 APP_CMD 位并等待 ACMD 命令。
2. 发送指定的 ACMD 卡送回响应给 MMC/SD 卡模块，指示设置了 APP_CMD 位，收到的命令已经正确地按照 ACMD 命令解析；如果发送了一个非 ACMD 命令，卡将按照普通的多媒体卡命令处理同时清除卡中状态寄存器的 APP_CMD 位。

如果发送了一个非法的命令（不管是 ACMD 还是 CMD），将被按照标准的非法多媒体卡命令进行错误处理。

GEN_CMD 命令的总线操作过程，与单数据块读写命令（WRITE_BLOCK，CMD24 或 READ_SINGLE_BLOCK，CMD17）相同；这时命令的参数表示数据传输的方向而不是地址，数据块具有用户自定义的格式和意义。

发送 GEN_CMD（CMD56）命令之前，卡必须被选中（状态机处于传输状态），数据块的长度由 SET_BLOCKLEN（CMD16）定义。GEN_CMD（CMD56）命令的响应是 R1b 格式。

命令类型

应用相关命令和通用命令有四种不同的类型：

1. 广播命令（BC）：发送到所有卡，没有响应返回。
2. 带响应的广播命令（BCR）：发送到所有卡，同时收到从所有卡返回的响应。
3. 带寻址（点对点）的命令（AC）：发送到选中的卡，在 SDIO_D 信号线上不包括数据传输。
4. 带寻址（点对点）的数据传输命令（ADTC）：发送到选中的卡，在 SDIO_D 信号线上包含数据传输。

命令格式

命令格式参见表 29-2。

多媒体卡/SD 卡模块的命令

表 29-18 基于块传输的写命令

CMD 索引	类型	参数	响应格式	缩写	说明
CMD23	ac	[31:16]=0 [15:0]=数据块数目	R1	SET_BLOCK_COUNT	定义在随后的多块读或写命令中需要传输块的数目。
CMD24	adtc	[31:0]=数据地址	R1	WRITE_BLOCK	按照 SET_BLOCKLEN 命令选择的长度写一个块。
CMD25	adtc	[31:0]=数据地址	R1	WRITE_MULTIPLE_BLOCK	收到一个 STOP_TRANSMISSION 命令或达到了指定的块数目之前，连续地写数据块。
CMD26	adtc	[31:0]=填充位	R1	PROGRAM_CID	对卡的识别寄存器编程。对于每个卡只能发送一次这个命令。卡中有硬件机制防止多次的编程操作。通常该命令保留给生产厂商。
CMD27	adtc	[31:0]=填充位	R1	PROGRAM_CSD	对卡的 CSD 中可编程的位编程。
CMD28	ac	[31:0]=数据地址	R1b	SET_WRITE_PROT	如果卡有写保护功能，该命令设置指定组的写保护位。写保护特性设置在卡的特殊数据区（WP_GRP_SIZE）。
CMD29	ac	[31:0]=数据地址	R1b	CLR_WRITE_PROT	如果卡有写保护功能，该命令清除指定组的写保护位。
CMD30	adtc	[31:0]=写保护数据地址	R1	SEND_WRITE_PROT	如果卡有写保护功能，该命令要求卡发送写保护位的状态。
CMD31	保留				

表 29-19 基于块传输的写保护命令

CMD 索引	类型	参数	响应格式	缩写	说明
CMD28	ac	[31:0]=数据地址	R1b	SET_WRITE_PROT	如果卡具有写保护的功能，该命令设置指定组的写保护位。写保护的属性设置在卡的特定数据域（WP_GRP_SIZE）。
CMD29	ac	[31:0]=数据地址	R1b	CLR_WRITE_PROT	如果卡具有写保护的功能，该命令清除指定组的写保护位。
CMD30	adtc	[31:0]=写保护数据地址	R1	SEND_WRITE_PROT	如果卡具有写保护的功能，该命令要求卡发送写保护位的状态。
CMD31	保留				

表 29-20 擦除命令

CMD 索引	类型	参数	响应格式	缩写	说明
CMD32 …CMD34	保留。为了与旧版本的对媒体卡协议向后兼容，不能使用这些命令代码。				

CMD 索引	类型	参数	响应格式	缩写	说明
CMD35	ac	[31:0]=数据地址	R1	ERASE_GROUP_START	在选择的擦除范围内，设置第一个擦除组的地址。
CMD36	ac	[31:0]=数据地址	R1	ERASE_GROUP_END	在选择的连续擦除范围内，设置最后一个擦除组的地址。
CMD37	保留。为了与旧版本的对媒体卡协议向后兼容，不能使用这个命令代码。				
CMD38	ac	[31:0]=填充位	R1	ERASE	擦除之前选择的数据块。

表 29-21 I/O 模式命令

CMD 索引	类型	参数	响应格式	缩写	说明
CMD39	ac	[31:16]=RCA [15]=寄存器写标志 [14:8]=寄存器地址 [7:0]=寄存器数据	R4	FAST_IO	用于写和读 8 位（寄存器）数据域。该命令指定一个卡和寄存器，如果设置了写标志还提供写入的数据。R4 响应包含从指定寄存器读出的数据。该命令访问未在多媒体卡标准中定义的与应用相关的寄存器。
CMD40	bcr	[31:0]=填充位	R5	GO_IRQ_STATE	置系统于中断模式。
CMD41	保留。				

表 29-22 上锁命令

CMD 索引	类型	参数	响应格式	缩写	说明
CMD42	adtc	[31:0]=填充位	R1b	LOCK_UNLOCK	设置/清除密码或对卡上锁/解锁。数据块的长度由 SET_BLOCKLEN 命令设置。
CMD43 …CMD54	保留。				

表 29-23 应用相关命令

CMD 索引	类型	参数	响应格式	缩写	说明
CMD55	ac	[31:16]=RCA [15:0]=填充位	R1	APP_CMD	指示卡下一个命令是应用相关命令而不是一个标准命令。
CMD56	adtc	[31:1]=填充位 [0]=RD/WR			在通用或应用相关命令中，或者用于向卡中传输一个数据块，或者用于从卡中读取一个数据块。数据块的长度由 SET_BLOCKLEN 命令设置。
CMD57 …CMD59	保留。				
CMD60 …CMD63	保留给生产厂商。				

29.5 响应格式

所有的响应是通过 MCCMD 命令在 SDIO_CMD 信号线上传输。响应的传输总是从对应响应字的位串的最左面开始，响应字的长度与响应的类型相关。

一个响应总是有一个起始位（始终为 0），跟随着传输的方向位（卡=0）。下表中标示为 x 的数值表示一个可变的部分。除了 R3 响应类型，所有的响应都有 CRC 保护。每一个命令码字都有一个结束位（始终为 1）。

共有 8 种响应类型，它们的格式定义如下：

29.5.1 R1（普通响应命令）

代码长度=48 位。位 45:40 指示要响应的命令索引，它的数值介于 0 至 63 之间。卡的状态由 32 位进行编码。

表 29-24 R1 响应

位	域宽度	数值	说明
47	1	0	开始位
46	1	0	传输位
[45:40]	6	X(1)	命令索引
[39:8]	32	X(1)	卡状态
[7:1]	7	X(1)	CRC7
0	1	1	结束位

(1). X 表示不定值。

29.5.2 R1b

与 R1 格式相同，但可以选择在数据线上发送一个繁忙信号。收到这些命令后，依据收到命令之前的状态，卡可能变为繁忙。

29.5.3 R2（CID、CSD 寄存器）

代码长度=136 位。CID 寄存器的内容将作为 CMD2 和 CMD10 的响应发出。CSD 寄存器的内容将作为 CMD9 的响应发出。卡只送出 CID 和 CSD 的位[127…1]，在接收端这些寄存器的位 0 被响应的结束位所取代。卡通过拉低 MCDAT 指示它正在进行擦除操作；实际擦除操作的时间可能非常长，主机可以发送 CMD7 命令不选中这个卡。

表 29-25 R2 响应

位	域宽度	数值	说明
135	1	0	开始位
134	1	0	传输位
[133:128]	6	‘111111’	命令索引
[127:1]	127	X(1)	卡状态
0	1	1	结束位

(1). X 表示不定值。

29.5.4 R3（OCR 寄存器）

代码长度=48 位。OCR 寄存器的内容将作为 CMD1 的响应发出。电平代码的定义是：限制的电压窗口=低，卡繁忙=低。

表 29-26 R3 响应

位	域宽度	数值	说明
47	1	0	开始位
46	1	0	传输位
[45:40]	6	‘111111’	保留
[39:8]	32	X(1)	OCR 寄存器
[7:1]	7	‘1111111’	保留
0	1	1	结束位

(1). X 表示不定值。

29.5.5 R4（快速 I/O）

代码长度=48 位。参数域包含指定卡的 RCA、需要读出或写入寄存器的地址、和它的内容。

表 29-27 R4 响应

位		域宽度	数值	说明
47		1	0	开始位
46		1	0	传输位
[45:40]		6	‘100111’	保留
[39:8]参数域	[31:16]	16	X(1)	RCA
	[15:8]	8	X(1)	寄存器地址
	[7:0]	8	X(1)	读寄存器的内容
[7:1]		7	X(1)	CRC7
0		1	1	结束位

(1). X 表示不定值。

29.5.6 R4b

仅适合 SD I/O 卡：一个 SDIO 卡收到 CMD5 后将返回一个唯一的 SDIO 响应 R4。

表 29-28 R4b 响应

位	域宽度	数值	说明
47	1	0	开始位

位		域宽度	数值	说明
46		1	0	传输位
[45:40]		6	X(1)	保留
[39:8]参数域	39	16	X(1)	卡已就绪
	[38:36]	3	X(1)	I/O 功能数目
	35	1	X(1)	当前存储器
	[34:32]	3	X(1)	填充位
	[31:8]	24	X(1)	I/OORC
[7:1]		7	X(1)	保留
0		1	1	结束位

(1). X 表示不定值。

当一个 SD I/O 卡收到命令 CMD5，卡的 I/O 部分被使能并能够正常地响应所有后续的命令。I/O 卡的使能状态将保持到下一次复位、断电或收到 I/O 复位的 CMD52 命令。注意，一个只包含存储器功能的 SD 卡可以响应 CMD5 命令，它的正确响应可以是：当前存储器=1，I/O 功能数目=0。按照 SD 存储器卡规范版本 1.0 设计的只包含存储器功能的 SD 卡，可以检测到 CMD5 命令为一个非法命令并不响应它。可以处理 I/O 卡的主机将发送 CMD5 命令，如果卡返回响应 R4，则主机会依据 R4 响应中的数据确定卡的配置。

29.5.7 R5（中断请求）

仅适用于多媒体卡。代码长度=48 位。如果这个响应由主机产生，则参数中的 RCA 域为 0x0。

表 29-29 R5 响应

位		域宽度	数值	说明
47		1	0	开始位
46		1	0	传输位
[45:40]		6	‘101000’	CMD40
[39:8]参数域	[31:16]	16	X(1)	成功的卡或主机的 RCA[31:16]
	[15:0]	16	X(1)	未定义。可以作为中断数据。
[7:1]		7	X(1)	CRC7
0		1	1	结束位

(1). X 表示不定值。

29.5.8 R6（中断请求）

仅适用于 SD I/O 卡。这是一个存储器设备对 CMD3 命令的正常响应。

表 29-30 R6 响应

位		域宽度	数值	说明
47		1	0	开始位
46		1	0	传输位
[45:40]		6	‘101000’	CMD40
[39:8]参数域	[31:16]	16	X(1)	成功的卡或主机的 RCA[31:16]
	[15:0]	16	X(1)	未定义。可以作为中断数据。
[7:1]		7	X(1)	CRC7
0		1	1	结束位

(1). X 表示不定值。

当发送 CMD3 命令到只有 I/O 功能的卡时，卡的状态位[23:8]会改变；此时，响应中的 16 位将是只有 I/O 功能的 SD 卡中的数值：

- 位 15=COM_CRC_ERROR
- 位 14=ILLEGAL_COMMAND
- 位 13=ERROR
- 位[12:0]=保留

29.6 SDIO I/O 卡特定的操作

下述功能是 SD I/O 卡特定的操作:

- 由 SDIO_D2 信号线实现的 SDIO 读等待操作。
- 通过停止时钟实现的 SDIO 读等待操作。
- SDIO 暂停/恢复操作（写和读暂停）
- SDIO 中断

只有设置了 SDIO_DCTRL[11]位时，SDIO 才支持这些操作；但读暂停除外，因为它不需要特殊的硬件操作。

29.6.1 使用 SDIO_D2 信号线的 SDIOI/O 读等待操作

在收到第一个数据块之前即可以开始读等待过程，使能数据通道（设置 SDIO_DCTRL[0]位）、使能 SDIO 特定操作（设置 SDIO_DCTRL[11]位）、开始读等待（SDIO_DCTRL[10]=0 并且 SDIO_DCTRL[8]=1），同时数据传输方向是从卡至 SDIO 主机（SDIO_DCTRL[1]=1），DPSM 将直接从空闲进入读等待状态。在读等待状态时，2 个 SDIO_CK 时钟周期后，DPSM 驱动 SDIO_D2 为’0’，在此状态，如果设置 RWSTOP 位（SDIO_DCTRL[9]），则 DPSM 会在等待状态多停留 2 个 SDIO_CK 时钟周期，（根据 SDIO 规范）并在一个时钟周期中驱动 SDIO_D2 为’1’。然后 DPSM 开始等待从卡里接收数据。在接收数据块时，即使设置了开始读等待，DPSM 也不会进入读等待，读等待过程将在收到 CRC 后开始。必须清除 RWSTOP 才能开始新的读等待操作。在读等待期间，SDIO 主机可以在 SDIO_D1 上监测 SDIO 中断。

29.6.2 使用停止 SDIO_CK 的 SDIO 读等待操作

如果 SDIO 卡不能支持前述的读等待操作，SDIO 可以停止 SDIO_CK 进入读等待（按照章节：“29.6.1 使用 SDIO_D2 信号线的 SDIOI/O 读等待操作”介绍的方式设置 SDIO_DCTRL，但置 SDIO_DCTRL[10]=1），在接收当前数据块结束位之后的 2 个 SDIO_CK 周期后，DSPM 停止时钟，在设置了读等待开始位后恢复

时钟。

因为 SDIO_CK 停止了，可以向卡发送任何命令。在读等待期间，SDIO 主机可以在 SDIO_D1 上监测 SDIO 中断。

29.6.3 SDIO 暂停/恢复操作

在向卡发送数据时，SDIO 可以暂停写操作。设置 SDIO_CMD[11]位，这指示 CPSM 当前的命令是一个暂停命令。CPSM 分析响应，在从卡收到 ACK 时（暂停被接受），它确认在收到当前数据块的 CRC 后进入空闲状态。

硬件不会保存结束暂停操作之后，剩余的发送数据块数目。可以通过软件暂停写操作：在收到卡对暂停命令的 ACK 时，停止 DPSM（SDIO_DCTRL[0]=0），DPSM 即可进入空闲状态。

暂停读操作：DPSM 在 Wait_r 状态等待，在停止数据传输进入暂停之前，已经发送完成完整的数据包。随后应用程序继续读出 RxFIFO 直到 FIFO 变空，最后 DPSM 自动地进入空闲状态。

29.6.4 SDIO 中断

当设置了 SDIO_DCTRL[11]位，SDIO 主机在 SDIO_D1 信号线上监测 SDIO 中断。

29.7 CE-ATA 特定操作

下面是 CE-ATA 的特定操作：

- 送出命令完成信号能够关闭 CE-ATA 设备
- 从 CE-ATA 设备接收命令完成信号
- 使用状态位和/或中断，向 CPU 发送 CE-ATA 命令完成信号仅当设置了 SDIO_CMD[14]位时，即 SDIO 主机只对 CE-ATA 的 CMD61 命令支持这些操作。

29.7.1 命令完成指示关闭

如果未设置 SDIO_CMD[12]中的“允许 CMD 结束位”并且设置了 SDIO_CMD[13]中的“非中断使能位”，则在收到一个短响应后的 8 个位周期之后，发出命令完成关闭信号。

在命令移位寄存器中写入关闭序列“00001”并且在命令计数器中写入 43，则 CPSM 进入暂停状态。8 个周期后，一个触发将 CPSM 移至发送状态。当命令计数器达到 48 时，因为没有要等待的响应，CPSM 变为空闲状态。

29.7.2 命令完成指示使能

如果设置 SDIO_CMD[12]中的“允许 CMD 结束位”并且设置了 SDIO_CMD[13]中的“非中断使能位”，CPSM 在 Waitcpl 状态下等待命令完成信号。

当在 CMD 信号上收到’0’，CPSM 进入空闲状态。在个 7 位周期之内不能发送新命令。然后，在最后 5 个周期（上述 7 个周期之外），在推挽模式下 CMD 信号变为’1’。

29.7.3 CE-ATA 中断

命令完成是由状态位 SDIO_STA[23]通知 CPU，使用清除位 SDIO_ICR[23]可以清除该位。根据屏蔽位 SDIO_MASKx[23]的设置，SDIO_STA[23]状态位可以在每一个中断线上产生中断。

29.7.4 中止 CMD61

如果还未发送“命令完成指示关闭”信号，但需要中止 CMD61 命令，命令状态机必须被关闭。然后它变成空闲，并且可以发送 CMD12 命令。在此操作期间，不传送“命令完成指示关闭”信号。

29.8 硬件流控制

使用硬件流控制功能可以避免 FIFO 下溢（发送模式）和上溢（接收模式）错误。

操作过程是停止 SDIO_CK 并冻结 SDIO 状态机，在 FIFO 不能进行发送和接收数据时，数据传输暂停。只有由 SDIOCLK 驱动的状态机被冻结，AHB 接口还在工作。即使在流控制起作用时，仍然可以读出或写入 FIFO。

必须设置 SDIO_CLKCR[14]位为'1'，才能使能硬件流控制。复位后，硬件流控制功能关闭。

29.9 SDIO 寄存器

基地址：0x4001 8000

空间大小：0x400

设备通过可以在 AHB 上操作的 32 位控制寄存器与系统通信。必须以字（32 位）的方式操作这些外设寄存器。

29.9.1 SDIO 电源控制寄存器（SDIO_POWER）

偏移地址：0x00

复位值：0x00000000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res														PWRCTRL[1:0]	
														rw	

位 31:2	Res：保留 必须保持复位值。
位 1:0	PWRCTRL[1:0]：电源控制位（Power control） 该位域用于定义卡时钟的当前功能状态： • 00：电源关闭，卡的时钟停止 • 01：保留 • 10：保留的上电状态 • 11：上电状态，卡的时钟开启

注意：写数据后的 7 个 HCLK 时钟周期内，不能写入这个寄存器。

29.9.2 SDIO 时钟控制寄存器（SDIO_CLKCR）

偏移地址：0x04

复位值：0x00000000

SDIO_CLKCR 寄存器控制 SDIO_CK 输出时钟。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	HWFC_EN	NEGEDGE	WIDBUS[1:0]		BYPASS	PWRSAV	CLKEN	CLKDIV[7:0]							
	rw	rw	rw		rw	rw	rw	rw							

位 31:15	Res：保留

	必须保持复位值。
位 14	HWFC_EN：硬件流控制使能（HW flow control enable） • 0：关闭硬件流控制 • 1：使能硬件流控制 当使能硬件流控制后，关于 TXFIFOE 和 RXFIFOF 中断信号的意义参见章节“29.9.11 SDIO 状态寄存器（SDIO_STA）”。
位 13	NEGEDGE：SDIO_CK 相位选择位（SDIO_CK dephasing selection bit） • 0：在主时钟 SDIOCLK 的上升沿产生 SDIO_CK • 1：在主时钟 SDIOCLK 的下降沿产生 SDIO_CK
位 12:11	WIDBUS[1:0]：宽总线模式使能位（Wide bus mode enable bit） • 00：默认总线模式，使用 SDIO_D0。 • 01：4 位总线模式，使用 SDIO_D[3:0]。 • 10：8 位总线模式，使用 SDIO_D[7:0]。 • 11：保留
位 10	BYPASS：旁路时钟分频器（Clock divider bypass enable bit） • 0：关闭旁路：驱动 SDIO_CK 输出信号之前，依据 CLKDIV 数值对 SDIOCLK 分频。 • 1：使能旁路：SDIOCLK 直接驱动 SDIO_CK 输出信号。
位 9	PWRSAV：省电配置位（Power saving configuration bit） 为了省电，当总线为空闲时，设置 PWRSAV 位可以关闭 SDIO_CK 时钟输出。 • 0：始终输出 SDIO_CK。 • 1：仅在有总线活动时才输出 SDIO_CK。
位 8	CLKEN：时钟使能位（Clock enable bit） • 0：SDIO_CK 关闭 • 1：SDIO_CK 使能
位 7:0	CLKDIV[7:0]：时钟分频系数（Clock divide factor） 这个域定义了输入时钟（SDIOCLK）与输出时钟（SDIO_CK）间的分频系数： SDIO_CK 频率=SDIOCLK/[CLKDIV+2]。

29.9.3 SDIO 参数寄存器（SDIO_ARG）

偏移地址：0x08

复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CMDARG[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDARG[15:0]															
rw															

位 31:3	CMDARG[31:0]：命令参数（Command argument） 命令参数是发送到卡中命令的一部分，如果一个命令包含一个参数，必须在写命令到命令寄存 器之

	前加载这个寄存器。

29.9.4 SDIO 命令寄存器（SDIO_CMD）

偏移地址：0x0C

复位值：0x00000000

SDIO_CMD 寄存器包含命令索引和命令类型位。命令索引是作为命令的一部分发送到卡中。命令类型位控制命令通道状态机（CPSM）。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	ATACMD	NIEN	ENCMDCOMPL	SDIOSUSPEND	CPSMEN	WAITPEND	WAITINT	WAITRESP[1:0]		CMDINDEX[5:0]					
	rw	rw	rw	rw	rw	rw	rw	rw		rw					

位 31:15	Res：保留 必须保持复位值。
位 14	ATACMD：CE-ATA 命令（CE-ATA command） 如果设置该位，CPSM 转至 CMD61。
位 13	NIEN：不使能中断（Interrupt not enabled） 如果未设置该位，则使能 CE-ATA 设备的中断。
位 12	ENCMDCOMPL：使能 CMD 完成（Enable CMD completion） 如果设置该位，则使能命令完成信号。
位 11	SDIOSUSPEND：SDI/O 暂停命令（SDI/O suspend command） 如果设置该位，则将要发送的命令是一个暂停命令（只能用于 SDIO 卡）。
位 10	CPSMEN：命令通道状态机（CPSM）使能位（Command path state machine, CPSM Enable bit） 如果设置该位，则使能 CPSM。
位 9	WAITPEND：CPSM 等待数据传输结束（Cmd Pend 内部信号）（Cmd Pend internal signal, CPSM Waits for ends of data transfer） 如果设置该位，则 CPSM 在开始发送一个命令之前等待数据传输结束。
位 8	WAITINT：CPSM 等待中断请求（CPSM waits for interrupt request） 如果设置该位，则 CPSM 关闭命令超时控制并等待中断请求。
位 7:6	WAITRESP[1:0]：等待响应位（Wait for response bits） 这 2 位指示 CPSM 是否需要等待响应，如果需要等待响应，则指示响应类型。 • 00：无响应，期待 CMDSENT 标志。 • 01：短响应，期待 CMDREND 或 CCRCFAIL 标志。 • 10：无响应，期待 CMDSENT 标志。 • 11：长响应，期待 CMDREND 或 CCRCFAIL 标志。
位 5:0	CMDINDEX[5:0]：命令索引（Command index） 命令索引是作为命令的一部分发送到卡中。

注意：

1.写数据后的 7 个 HCLK 时钟周期内不能写入这个寄存器。

2.多媒体卡可以发送 2 种响应：48 位长的短响应，或 136 位长的长响应。SD 卡和 SDI/O 卡只能发送短响应，参数可以根据响应的类型而变化，软件将根据发送的命令区分响应的类型。CE-ATA 设备只发送短响应。

29.9.5 SDIO 命令响应寄存器（SDIO_RESPCMD）

偏移地址：0x10

复位值：0x00000000

SDIO_RESPCMD 寄存器包含最后收到的命令响应中的命令索引。如果传输的命令响应不包含命令索引（长响应或 OCR 响应），尽管它应该包含 111111b（响应中的保留域值），但 RESPCMD 域的内容未知。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res										RESPCMD[5:0]					
										r					

位 31:6	Res：保留 必须保持复位值。
位 5:0	RESPCMD[5:0]：响应的命令索引（Response command index） 只读位，包含最后收到的命令响应中的命令索引。

29.9.6 SDIO 响应 1/2/3/4 寄存器（SDIO_RESPx）（x=1..4）

偏移地址：0x14+4*（x-1）

复位值：0x00000000

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

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

位 31:0	CARDSTATUSx：卡状态位，见下表（Card state）

根据响应状态，卡的状态长度是 32 位或 127 位。总是先收到卡状态的最高位，SDIO_RESP3 寄存器的最低位始终为 0。

表 29-31 响应类型和 SDIO_RESPx 寄存器

寄存器	短响应	长响应
SDIO_RESP1	卡状态[31:0]	卡状态[127:96]
SDIO_RESP2	不用	卡状态[95:64]
SDIO_RESP3	不用	卡状态[63:32]
SDIO_RESP4	不用	卡状态[31:1]

29.9.7 SDIO 数据定时器寄存器（SDIO_DTIMER）

偏移地址：0x24

复位值：0x00000000

SDIO_DTIMER 寄存器包含以卡总线时钟周期为单位的数据超时时间。

一个计数器从 SDIO_DTIMER 寄存器加载数值，并在数据通道状态机（DPSM）进入 Wait_R 或繁忙状态时进行递减计数，当 DPSM 处在这些状态时，如果计数器减为 0，则设置超时标志。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATATIME[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATATIME[15:0]															
rw															

位 31:0	DATATIME[31:0]：数据超时时间（Data timeout time） 以卡总线时钟周期为单位的数据超时时间。

注意：在写入数据控制寄存器进行数据传输之前，必须先写入数据定时器寄存器和数据长度寄存器。

29.9.8 SDIO 数据长度寄存器（SDIO_DLEN）

偏移地址：0x28

复位值：0x00000000

SDIO_DLEN 寄存器包含需要传输的数据字节长度。当数据传输开始时，这个数值被加载到数据计数器中。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res							DATALENGTH[24:16]								
							rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATALENGTH[15:0]															
rw															

位 31:25	Res：保留 必须保持复位值。
位 24:0	DATALENGTH[24:0]：数据长度（Data length value） 要传输的数据字节数目。

注意：对于块数据传输，数据长度寄存器中的数值必须是数据块长度（见 SDIO_DCTRL）的倍数。在写入数据控制寄存器进行数据传输之前，必须先写入数据定时器寄存器和数据长度寄存器。

29.9.9 SDIO 数据控制寄存器（SDIO_DCTRL）

偏移地址：0x2C

复位值：0x00000000

SDIO_DCTRL 寄存器控制数据通道状态机（DPSM）。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res				SDIOEN	RWMOD	RWSTOP	RWSTART	DBLOCKSIZE[3:0]				DMAEN	DTMODE	DTDIR	DTEN
				rw	rw	rw	rw	rw				rw	rw	rw	rw

位 31:12	Res：保留 必须保持复位值。
位 11	SDIOEN：SDI/O 使能功能（SDI/O enable functions） 如果设置了该位，则 DPSM 执行 SDI/O 卡特定的操作。
位 10	RWMOD：读等待模式（Read wait mode） • 0：停止 SDIO_CK 控制读等待； • 1：使用 SDIO_D2 控制读等待。
位 9	RWSTOP：读等待停止（Read wait stop） • 0：如果设置了 RWSTART，执行读等待； • 1：如果设置了 RWSTART，停止读等待。
位 8	RWSTART：读等待开始（Read wait start） 设置该位开始读等待操作。
位 7:4	DBLOCKSIZE[3:0]：数据块长度（Data block size） 当选择了块数据传输模式，该域定义数据块长度： • 0000：块长度=20=1 字节 • 0001：块长度=21=2 字节 • 0010：块长度=22=4 字节 • 0011：块长度=23=8 字节 • 0100：（十进制 4）块长度=24=16 字节 • 0101：（十进制 5）块长度=25=32 字节 • 0110：（十进制 6）块长度=26=64 字节 • 0111：块长度=27=128 字节 • 1000：块长度=28=256 字节 • 1001：块长度=29=512 字节 • 1010：块长度=210=1024 字节 • 1011：块长度=211=2048 字节 • 1100：块长度=212=4096 字节 • 1101：块长度=213=8192 字节 • 1110：块长度=214=16384 字节 • 1111：保留
位 3	DMAEN：DMA 使能位（DMA enable bit） • 0：关闭 DMA • 1：使能 DMA
位 2	DTMODE：数据传输模式（Data transfer mode selection） • 0：块数据传输 • 1：流数据传输
位 1	DTDIR：数据传输方向（Data transfer direction selection）

	• 0：控制器至卡 • 1：卡至控制器
位 0	DTEN：数据传输使能位（Data transfer enable bit） 如果设置该位为 1，则开始数据传输。根据 DTSIR 方向位，DPSM 进入 Wait_S 或 Wait_R 状态，如果在传输的一开始就设置了 RWSTART 位，则 DPSM 进入读等待状态。不需要在数据传输结束后清除使能位，但必须更改 SDIO_DCTRL 以允许新的数据传输。

注意：写数据后的 7 个 HCLK 时钟周期内不能写入这个寄存器。

29.9.10 SDIO 数据计数寄存器（SDIO_DCOUNT）

偏移地址：0x30

复位值：0x00000000

当 DPSM 从空闲状态进入 Wait_R 或 Wait_S 状态时，SDIO_DCOUNT 寄存器从数据长度寄存器加载数值（见 SDIO_DLEN），在数据传输过程中，该计数器的数值递减直到减为 0，然后 DPSM 进入空闲状态并设置数据状态结束标志 DATAEND。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res							DATACOUNT[24:16]								
							r								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATACOUNT[15:0]															
r															

位 31:25	Res：保留 必须保持复位值。
位 24:0	DATACOUNT[24:0]：数据计数数值（Data count value） 读这个寄存器时返回待传输的数据字节数，写这个寄存器无作用。

注意：对于块数据传输，数据长度寄存器中的数值必须是数据块长度（见 SDIO_DCTRL）的倍数。在写入数据控制寄存器进行数据传输之前，必须先写入数据定时器寄存器和数据长度寄存器。

29.9.11 SDIO 状态寄存器（SDIO_STA）

偏移地址：0x34

复位值：0x00000000

SDIO_STA 是一个只读寄存器，它包含两类标志：

- 静态标志（位[23:22、10:0]）：写入 SDIO 中断清除寄存器（见 SDIO_ICR），可以清除这些位。
- 动态标志（位[21:11]）：这些位的状态变化根据它们对应的那部分逻辑而变化（例如：FIFO 满和空标志变高或变低随 FIFO 的数据写入变化）。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res								CEATAEND	SDIOIT	RXDVAL	TXDVAL	RXFIFOE	TXFIFOE	RXFIFOF	TXFIFOF
								r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXFIFOHF	TXFIFOHE	RXACT	TXACT	CMDACT	DBCKEND	STBITERR	DATAEND	CMDSENT	CMDREND	RXOVERR	TXUNDERR	DTIMEOUT	CTIMEOUT	DCRCFAIL	CCRCFAIL
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31:24	Res：保留

	必须保持复位值。
位 23	CEATAEND：在 CMD61 接收到 CE-ATA 命令完成信号（CE-ATA command completion signal received for CMD61）
位 22	SDIOIT：收到 SDIO 中断（SDIO interrupt received）
位 21	RXDVAL：在接收 FIFO 中的数据可用（Data available in receive FIFO）
位 20	TXDVAL：在发送 FIFO 中的数据可用（Data available in transmit FIFO）
位 19	RXFIFOE：接收 FIFO 空（Receive FIFO empty）
位 18	TXFIFOE：发送 FIFO 空（Transmit FIFO empty） 若使用了硬件流控制，当 FIFO 包含 2 个字时，TXFIFOE 信号变为有效。
位 17	RXFIFOF：接收 FIFO 满（Receive FIFO full） 若使用了硬件流控制，当 FIFO 还差 2 个字满时，RXFIFOF 信号变为有效。
位 16	TXFIFOF：发送 FIFO 满（Transmit FIFO full）
位 15	RXFIFOHF：接收 FIFO 半满（Receive FIFO half full） FIFO 中至少还有 8 个字。
位 14	TXFIFOHE：发送 FIFO 半空（Transmit FIFO half empty） FIFO 中至少还可以写入 8 个字。
位 13	RXACT：正在接收数据（Data receive in progress）
位 12	TXACT：正在发送数据（Data transmit in progress）
位 11	CMDACT：正在传输命令（Command transfer in progress）
位 10	DBCKEND：已发送/接收数据块（CRC 检测成功）（Data block sent/received, CRC check passed）
位 9	STBITERR：在宽总线模式，没有在所有数据信号上检测到起始位（Start bit not detected on all data signals in wide bus mode）
位 8	DATAEND：数据结束（数据计数器，SDIO_DCOUNT=0）（Data counter, SDIDCOUNT is zero）
位 7	CMDSENT：命令已发送（不需要响应）（Command sent, no response required）
位 6	CMDREND：已接收到响应（CRC 检测成功）（Command response）
位 5	RXOVERR：接收 FIFO 上溢错误（Received FIFO overrun error）
位 4	TXUNDERR：发送 FIFO 下溢错误（Transmit FIFO underrun error）
位 3	DTIMEOUT：数据超时（Data time out）
位 2	CTIMEOUT：命令响应超时（Command response time out）

	命令超时时间是一个固定的值，为 64 个 SDIO_CK 时钟周期。
位 1	DCRCFAIL：已发送/接收数据块（CRC 检测失败）（Data block sent/received）
位 0	CCRCFAIL：已收到命令响应（CRC 检测失败）（Command response received）

29.9.12 SDIO 清除中断寄存器（SDIO_ICR）

偏移地址：0x38

复位值：0x00000000

SDIO_ICR 是一个只写寄存器，在对应寄存器位写’1’将清除 SDIO_STA 状态寄存器中的对应位。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res					DBCKENDC	STBITERRC	DATAENDC	CMDSENTC	CMDRENDC	RXOVERRC	TXUNDERRC	DTIMEOUTC	CTIMEOUTC	DCRCFAILC	CCRCFAILC
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:24	Res：保留 必须保持复位值。
位 23	CEATAENDC：CEATAEND 标志清除位（CEATAEND flag clear bit） 软件设置该位以清除 CEATAEND 标志。
位 22	SDIOITC：SDIOIT 标志清除位（SDIOIT flag clear bit） 软件设置该位以清除 SDIOIT 标志。
位 21:11	Res：保留 必须保持复位值。
位 10	DBCKENDC：DBCKEND 标志清除位（DBCKEND flag clear bit） 软件设置该位以清除 DBCKEND 标志。
位 9	STBITERRC：STBITERR 标志清除位（STBITERR flag clear bit） 软件设置该位以清除 STBITERR 标志。
位 8	DATAENDC：DATAEND 标志清除位（DATAEND flag clear bit） 软件设置该位以清除 DATAEND 标志。
位 7	CMDSENTC：CMDSENT 标志清除位（CMDSENT flag clear bit） 软件设置该位以清除 CMDSENT 标志。
位 6	CMDRENDC：CMDREND 标志清除位（CMDREND flag clear bit） 软件设置该位以清除 CMDREND 标志。
位 5	RXOVERRC：RXOVERR 标志清除位（RXOVERR flag clear bit） 软件设置该位以清除 RXOVERR 标志。

位 4	TXUNDERRC：TXUNDERR 标志清除位（TXUNDERR flag clear bit） 软件设置该位以清除 TXUNDERR 标志。
位 3	DTIMEOUTC：DTIMEOUT 标志清除位（DTIMEOUT flag clear bit） 软件设置该位以清除 DTIMEOUT 标志。
位 2	CTIMEOUTC：CTIMEOUT 标志清除位（CTIMEOUT flag clear bit） 软件设置该位以清除 CTIMEOUT 标志。
位 1	DCRCFAILC：DCRCFAIL 标志清除位（DCRC FAIL flag clear bit） 软件设置该位以清除 DCRCFAIL 标志。
位 0	CCRCFAILC：CCRCFAIL 标志清除位（CCRC FAIL flag clear bit） 软件设置该位以清除 CCRCFAIL 标志。

29.9.13 SDIO 中断屏蔽寄存器（SDIO_MASK）

偏移地址：0x3C

复位值：0x00000000

在对应位置’1’，SDIO_MASK 中断屏蔽寄存器决定哪一个状态位产生中断。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res								CEATAENDIE	SDIOITIE	RXDVALIE	TXDVALIE	RXFIFOEIE	TXFIFOEIE	RXFIFOFIE	TXFIFOFIE
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXFIFOHFIE	TXFIFOHE	RXACTIE	TXACTIE	CMDACTIE	DBCKENDIE	STBITERRIE	DATAENDIE	CMDSENTIE	CMDRENDIE	RXOVERRIE	TXUNDERRIE	DTIMEOUTIE	CTIMEOUTIE	DCRCFAILIE	CCRCFAILIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:24	Res：保留 必须保持复位值。
位 23	CEATAENDIE：允许接收到 CE-ATA 命令完成信号产生中断（CE-ATA command completion signal received interrupt enable） 由软件设置/清除该位，允许/关闭在收到 CE-ATA 命令完成信号产生中断功能。 • 0：收到 CE-ATA 命令完成信号时不产生中断 • 1：收到 CE-ATA 命令完成信号时产生中断
位 22	SDIOITIE：允许 SDIO 模式中断已接收中断（SDIO mode interrupt received interrupt enable） 由软件设置/清除该位，允许/关闭 SDIO 模式中断已接收中断功能。 • 1：SDIO 模式中断已接收不产生中断 • 0：SDIO 模式中断已接收产生中断
位 21	RXDVALIE：接收 FIFO 中的数据有效产生中断（Data available in RxFIFO interrupt enable） 由软件设置/清除该位，允许/关闭接收 FIFO 中的数据有效中断。 • 0：接收 FIFO 中的数据有效不产生中断 • 1：接收 FIFO 中的数据有效产生中断
位 20	TXDVALIE：发送 FIFO 中的数据有效产生中断（Data available in RxFIFO interrupt enable）

	由软件设置/清除该位，允许/关闭发送 FIFO 中的数据有效中断。 • 0：发送 FIFO 中的数据有效不产生中断 • 1：发送 FIFO 中的数据有效产生中断
位 19	RXFIFOEIE：接收 FIFO 空产生中断（RxFIFO empty interrupt enable） 由软件设置/清除该位，允许/关闭接收 FIFO 空中断。 • 0：接收 FIFO 空不产生中断 • 1：接收 FIFO 空产生中断
位 18	TXFIFOEIE：发送 FIFO 空产生中断（TxFIFO empty interrupt enable） 由软件设置/清除该位，允许/关闭发送 FIFO 空中断。 • 0：发送 FIFO 空不产生中断 • 1：发送 FIFO 空产生中断
位 17	RXFIFOFIE：接收 FIFO 满产生中断（RxFIFO full interrupt enable） 由软件设置/清除该位，允许/关闭接收 FIFO 满中断。 • 0：接收 FIFO 满不产生中断 • 1：接收 FIFO 满产生中断
位 16	TXFIFOFIE：发送 FIFO 满产生中断（TxFIFO full interrupt enable） 由软件设置/清除该位，允许/关闭发送 FIFO 满中断。 • 0：发送 FIFO 满不产生中断 • 1：发送 FIFO 满产生中断
位 15	RXFIFOHFIE：接收 FIFO 半满产生中断（RxFIFO half full interrupt enable） 由软件设置/清除该位，允许/关闭接收 FIFO 半满中断。 • 0：接收 FIFO 半满不产生中断 • 1：接收 FIFO 半满产生中断
位 14	TXFIFOHE：发送 FIFO 半空产生中断（TxFIFO half empty interrupt enable） 由软件设置/清除该位，允许/关闭发送 FIFO 半空中断。 • 0：发送 FIFO 半空不产生中断 • 1：发送 FIFO 半空产生中断
位 13	RXACTIE：正在接收数据产生中断（Datareceive acting interrupt enable） 由软件设置/清除该位，允许/关闭正在接收数据中断。 • 0：正在接收数据不产生中断 • 1：正在接收数据产生中断
位 12	TXACTIE：正在发送数据产生中断（Datatransmit acting interrupt enable） 由软件设置/清除该位，允许/关闭正在发送数据中断。 • 0：正在发送数据不产生中断 • 1：正在发送数据产生中断
位 11	CMDACTIE：正在传输命令产生中断（Command acting interrupt enable） 由软件设置/清除该位，允许/关闭正在传输命令中断。 • 0：正在传输命令不产生中断

	• 1：正在传输命令产生中断
位 10	DBCKENDIE：数据块传输结束产生中断（Data block end interrupt enable） 由软件设置/清除该位，允许/关闭数据块传输结束中断。 • 0：数据块传输结束不产生中断 • 1：数据块传输结束产生中断
位 9	STBITERRIE：起始位错误产生中断（Start bit error interrupt enable） 由软件设置/清除该位，允许/关闭起始位错误中断。 • 0：起始位错误不产生中断 • 1：起始位错误产生中断
位 8	DATAENDIE：数据传输结束产生中断（Data end interrupt enable） 由软件设置/清除该位，允许/关闭数据传输结束中断。 • 0：数据传输结束不产生中断 • 1：数据传输结束产生中断
位 7	CMDSENTIE：命令已发送产生中断（Command sent interrupt enable） 由软件设置/清除该位，允许/关闭命令已发送中断。 • 0：命令已发送不产生中断 • 1：命令已发送产生中断
位 6	CMDRENDIE：接收到响应产生中断（Command response received interrupt enable） 由软件设置/清除该位，允许/关闭接收到响应中断。 • 0：接收到响应不产生中断 • 1：接收到响应产生中断
位 5	RXOVERRIE：接收 FIFO 上溢错误产生中断（RxFIFO overrun error interrupt enable） 由软件设置/清除该位，允许/关闭接收 FIFO 上溢错误中断。 • 0：接收 FIFO 上溢错误不产生中断 • 1：接收 FIFO 上溢错误产生中断
位 4	TXUNDERRIE：发送 FIFO 下溢错误产生中断（TxFIFO underrun error interrupt enable） 由软件设置/清除该位，允许/关闭发送 FIFO 下溢错误中断。 • 0：发送 FIFO 下溢错误不产生中断 • 1：发送 FIFO 下溢错误产生中断
位 3	DTIMEOUTIE：数据超时产生中断（Data timeout interrupt enable） 由软件设置/清除该位，允许/关闭数据超时中断。 • 0：数据超时不产生中断 • 1：数据超时产生中断
位 2	CTIMEOUTIE：命令超时产生中断（Command timeout interrupt enable） 由软件设置/清除该位，允许/关闭命令超时中断。 • 0：命令超时不产生中断 • 1：命令超时产生中断

位 1	DCRCFAILIE：数据块 CRC 检测失败产生中断（Data CRC fail interrupt enable） 由软件设置/清除该位，允许/关闭数据块 CRC 检测失败中断。 • 0：数据块 CRC 检测失败不产生中断 • 1：数据块 CRC 检测失败产生中断
位 0	CCRCFAILIE：命令 CRC 检测失败产生中断（Command CRC fail interrupt enable） 由软件设置/清除该位，允许/关闭命令 CRC 检测失败中断。 • 0：命令 CRC 检测失败不产生中断 • 1：命令 CRC 检测失败产生中断

29.9.14 SDIO FIFO 计数器寄存器（SDIO_FIFOCNT）

偏移地址：0x48

复位值：0x00000000

SDIO_FIFOCNT 寄存器包含还未写入 FIFO 或还未从 FIFO 读出的数据字数目。当在数据控制寄存器（SDIO_DCTRL）中设置了数据传输使能位 DTEN，并且 DPSM 处于空闲状态时，FIFO 计数器从数据长度寄存器（见 SDIO_DLEN）加载数值。如果数据长度未与字对齐（4 的倍数），则最后剩下的 1~3 个字节被当成一个字处理。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res							FIFOCOUNT[24:16]								
							r								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIFOCOUNT[15:0]															
r															

位 31:25	Res：保留 必须保持复位值。
位 24:0	FIFOCOUNT[24:0]：数据计数数值（Data count value） 读这个寄存器时返回待传输的数据字节数，写这个寄存器无作用。

注意：对于块数据传输，数据长度寄存器中的数值必须是数据块长度（见 SDIO_DCTRL）的倍数。在写入数据控制寄存器进行数据传输之前，必须先写入数据定时器寄存器和数据长度寄存器。

29.9.15 SDIO FIFO 数据寄存器（SDIO_FIFO）

偏移地址：0x80

复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIFODATA[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIFODATA[15:0]															
rw															

位 31:0	FIFODATA[31:0]：接收或发送 FIFO 数据（Receive and transmit FIFO data） FIFO 数据占据 32 个 32 位的字，地址为：（SDIO 基址+0x80）至（SDIO 基址+0xFC）

30 串行音频接口（SAI）

SAI 接口（串行音频接口）灵活性高、配置多样，可支持多种音频协议。该接口适用于许多立体声或单声道应用。例如，它可配置为支持 I2S 标准、LSB 或 MSB 对齐、PCM/DSP、TDM、AC’97 等协议和作为 SPDIF 发送端和 PDM 接收端。将音频模块配置为发送器时，SAI 接口可提供 SPDIF 输出。

SAI 通过两个完全独立的音频子模块来实现这种灵活性和可配置性。每个模块都有自己的时钟发生器和 I/O 线控制器。每个模块有其独立的时钟单元和协议处理单元，用户可根据需要配置为：协同工作或独立工作，主接口或从接口，接收端或发送端。

SAI 可以配置为主模式或配置为从模式。音频子模块既可作为接收器，又可作为发送器；既可与另一模块同步，又可以不同步。

30.1 SAI 主要特性

- 具有两个独立的音频子模块，子模块既可作为接收器，也可作为发送器，并带有自身的 FIFO。
- 每个音频子模块集成多达 8 个字（32 位）的 FIFO。
- 两个音频子模块间可以是同步或异步模式。
- 两个音频子模块的主/从配置相互独立。
- 当两个音频子模块都配置为以主模式工作时，每个子模块的时钟发生器采用独立的音频采样频率。
- 数据大小可配置：8 位、10 位、16 位、20 位、24 位或 32 位。
- 音频协议：I2S、LSB 或 MSB 对齐、PCM/DSP、TDM、AC’97、PDM。
- 支持 SPDIF 输出。
- 高达 16 个大小可配置的 Slot。
- 每帧的位数可配置。
- 帧同步有效电平可配置（偏移、位长、电平）。
- 可配置 Slot 中第一个有效位的位置。
- 支持 LSB 优先或 MSB 优先数据传输。
- 支持静音模式。
- 具有立体声/单声道音频帧功能。
- 通信时钟选通边沿可配置（SCK）。
- 错误标志及其对应相应中断（当中断使能时）
    - 上溢和下溢检测
    - 从模式下的帧同步信号提前检测
    - 从模式下的帧同步信号滞后检测
    - 接收时编码解码器未针对 AC’97 模式就绪
- 支持如下中断源（使能时）：
    - 错误
    - FIFO 请求
- DMA 接口分别为 SAIA 和 SAIB 提供独立的请求通道
- 支持的特征处理：双通道/单通道，μ_law/a_law 压缩解压等

30.2 SAI 功能说明

30.2.1 SAI 框图

SAI 框图如下图所示。

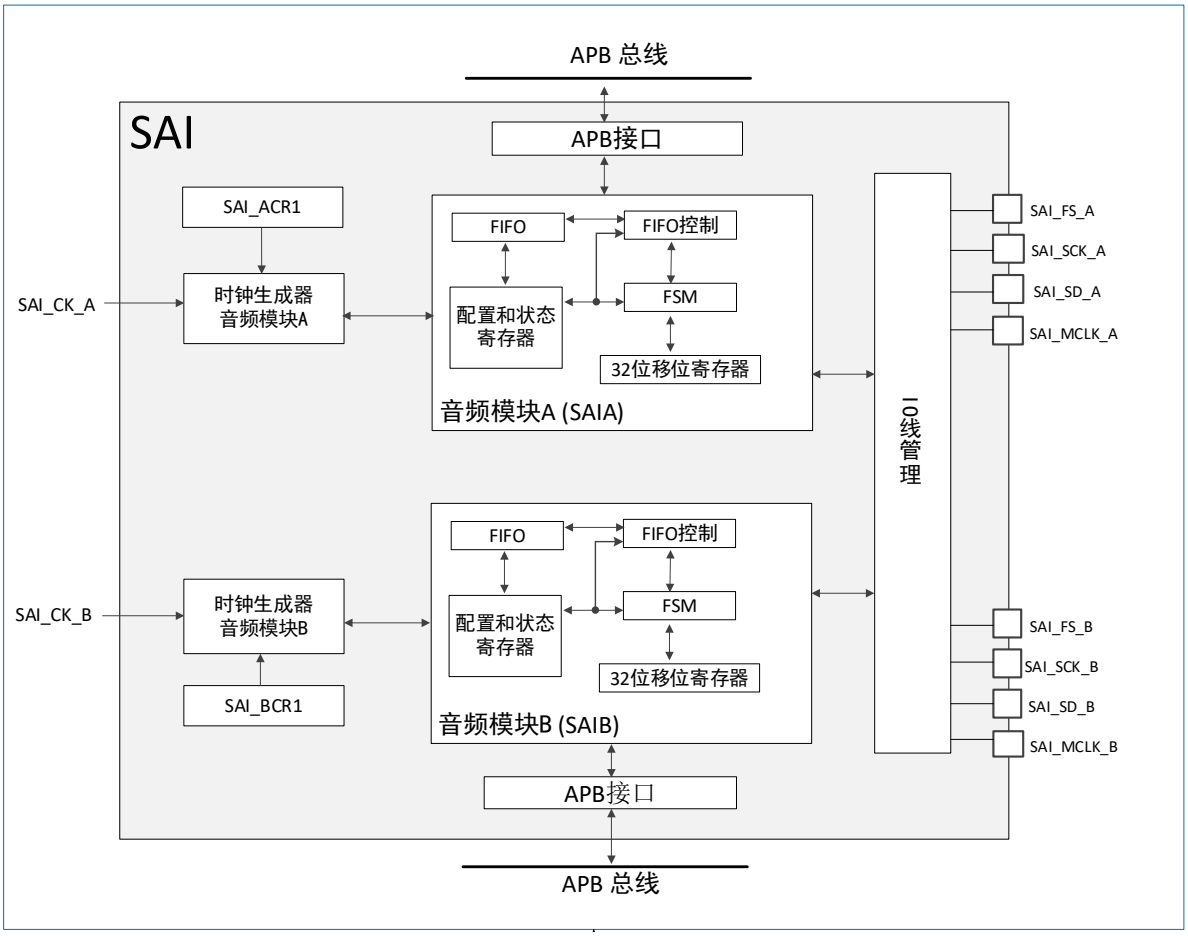


图 30-1 功能框图

SAI 主要由两个各自带有时钟发生器的音频子模块组成。每个音频模块集成一个 32 位移位寄存器，该寄存器由模块自身的功能状态机控制。数据存储和读取都是通过专用的 FIFO 来完成。FIFO 可通过 CPU 访问，也可通过 DMA 访问以减轻 CPU 的通信负担。每个音频子模块是独立的。这两个音频子模块可彼此同步。

I/O 线控制器管理 SAI 中指定音频模块的 4 个专用引脚（SD、SCK、FS、MCLK）。如果将两个子模块声明为同步模块，则可以共用其中某些引脚，从而留出一些引脚用作通用 I/O。MCLK 引脚是否可用作输出引脚取决于实际应用和解码要求以及音频模块是否配置为主模式。更多关于 SAI 引脚的信息，参见下表：

表 30-1 SAI 引脚说明

引脚名称	信号类别	描述
SAI_FS_A/B	输入/输出	音频模块 A/B 帧同步线
SAI_SCK_A/B	输入/输出	音频模块 A/B 位时钟
SAI_SD_A/B	输入/输出	音频模块 A/B 数据线
SAI_MCLK_A/B	输出	音频模块 A/B 主时钟

可配置功能状态机来处理多种音频协议。一些寄存器用于设置所需协议（音频帧波形发生器）。音频子模块在主模式或从模式下均可用作发送器或接收器。主模式意味着从 SAI 生成 SCK_x 位时钟，而从模式则意味着 SCK_x 位时钟来自另一外部或内部主器件。在特殊情况下，FS 信号方向与主模式或从模式定义不直接相关。在 AC'97 协议中，即使 SAI（链接控制器）设置为使用 SCK 时钟，FS 信号方向也会是 SAI 输出（从模式下也是如此）。

注意：为方便阅读此部分，符号 SAI_x 指 SAI_A 或 SAI_B，其中"x"代表 SAIA 子模块或 SAIB 子模块。

30.2.2 SAI 的主要模式

SAI 的每个音频子模块均可通过所选音频模块的 SAIx_CR1.MODE 位配置为主模式或从模式。

主模式

在主模式下，SAI 向连接的外部器件提供时钟信号：

- 位时钟和帧同步分别在引脚 SCK_x 和 FS_x 上输出。
- 如果需要，SAI 也可以在 MCLK_x 引脚上生成主时钟。

SCK_x、FS_x 和 MCLK_x 均配置为输出。

从模式

SAI 从外部器件接收时钟信号。

- 如果将 SAI 子模块配置为异步模式，则 SCK_x 引脚和 FS_x 引脚将被配置为输入。
- 如果将 SAI 子模块配置为与另一个 SAI 接口或第二个音频子模块同步运行，则会释放相应的 SCK_x 引脚和 FS_x 引脚以用作通用 I/O。

在从模式下，不使用 MCLK_x 引脚，可将其分配给其它功能。建议在使能主器件前先使能从器件。

配置和使能 SAI 模式

可通过相应音频模块的 SAIx_CR1.MODE 位将每个音频子模块独立定义为发送器或接收器。为此，SAIx_SD 引脚将分别被配置为输出或输入。

可使用两个不同的 MCLK 和 SCK 时钟频率配置同一 SAI 中的两个主音频模块。在这种情况下，必须将这两个音频模块配置为异步模式。

SAI 中的每个音频模块均通过 SAIx_CR1.SAIXEN 位使能。在从模式下，此位一经激活，发送器或接收器便会对时钟线（SCK_x）、数据线（SD_x）和同步线（FS_x）上的活动敏感。

在主 TX 模式下，即使 FIFO 中没有数据，使能音频模块也会立即为外部从模块产生位时钟，但 FS 信号的产生受 FIFO 中是否存在数据的控制。FIFO 接收到要发送的第一个数据后，此数据将输出到外部从模块。如果 FIFO 中没有要发送的数据，则随后将在音频帧中传送值 0，并会产生一个下溢标志。

在从模式下，使能音频模块时和检测到 SOF 位时开始音频帧。

在从 TX 模式下，使能音频模块后的第一个帧上不可能出现下溢事件，因为此时的强制操作顺序如下：

1. 通过软件或 DMA 写入 SAIx_DR。
2. 等待 FIFO 阈值（FLH）标志与 000b（FIFO 为空）不同。
3. 使能音频模块为从发送模式。

30.2.3 SAI 同步模式

将音频子模块 A 配置为与音频子模块 B 同步运行。在这种情况下，将共用位时钟和帧同步信号，以

减少通信时占用外部引脚的数量。配置为同步模式的音频模块将释放其 SCK_x、FS_x 和 MCLK_x 引脚以用作 GPIO，而配置为异步模式的音频模块将使用其 FS_x、SCK_x 和 MCLK_xI/O 引脚（如果该音频模块被配置为主模块）。

通常，同步模式下的音频模块可用于在全双工模式下配置 SAI。两个音频模块中的一个可配置为主模块，另一个为从模块；也可将两个均配置为从模块，其中一个模块为异步模块（SAIx_CR1 中的相应位 SYNCEN[1:0]设置为 00），另一个为同步模块（SAIx_CR1 中的相应位 SYNCEN[1:0]设置为 01）。

注意：由于存在内部重新同步阶段，因此 PCLK2 频率必须大于位时钟频率的二倍。

30.2.4 音频数据大小

通过配置 SAIx_CR1.DS[2:0]位，配置音频帧的数据大小。数据大小可以是 8 位、10 位、16 位、20 位、24 位或 32 位。在传输期间，将首先发送数据的 MSB 或 LSB，具体取决于 SAIx_CR1.LSBFIRST 位的配置。

30.2.5 帧同步

FS 信号用作音频帧（SOF）中的帧同步信号。此信号的波形完全可配置，以便在帧同步时，支持各种具有特殊规格的音频协议。该重新配置性通过寄存器 SAIx_FRCR 来实现。下图说明了这种灵活性。

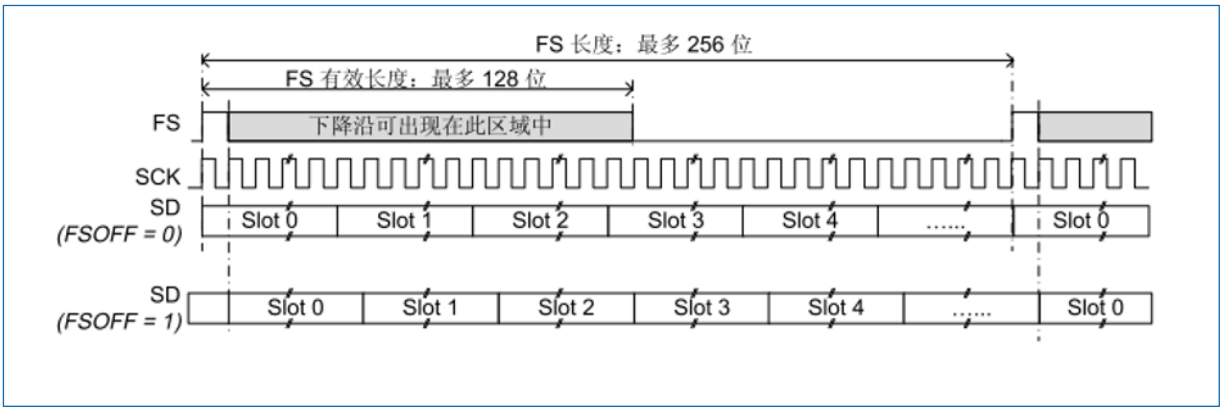


图 30-2 音频帧

AC’97 模式或 SPDIF 模式下（SAIx_CR1 寄存器中的位 PRTCFG[1:0]=10 或 PRTCFG[1:0]=01），帧同步信号的波形被强制配置为支持 AC’97 协议。SAIx_FRCR 寄存器值被忽略。

每个音频模块相互独立，因此均需要特定的配置。

30.2.5.1 帧长度

- 主模式

  将 SAIx_FRCR.FRL[7:0]字段置 1，可将音频帧的长度配置为最长 256 个位时钟周期。如果帧长度大于该帧声明的 Slot 数，则要发送的剩余位将用 0 填充，或者 SD 线将释放为高阻态，具体取决于 SAIx_CR2.TRIS 的状态（参见：“30.2.5.5FS 信号的作用”）。在接收模式下，剩余位被忽略。如果将位 NODIV 清零，（FRL+1）必须等于 2 的几次幂（从 8 到 256）以确保音频帧的每个位时钟周期都包含整数个 MCLK 脉冲。如果将位 NODIV 置 1，则（FRL+1）字段可采用从 8 到 256 的任意值。请参见：“30.2.7 SAI 时钟发生器”。

- 从模式

  音频帧的长度主要用于指定由外部主模块向从模块发送的每个音频帧的位时钟周期数。它主要用于从主模块中检测音频帧传输期间出现的提前或滞后帧同步信号。这种情况下会出现错误。有关详细信息，请参见：“30.2.13 错误标志”。在从模式下，SAIx_FRCR.FRL[7:0]位的配置不受任何限制。帧中的位数等于 FRL[7:0]+1。音频帧中要传输的最小位数为 8。

30.2.5.2 帧同步极性

SAIx_FRCR.FSPOL 位用于设置 FS 引脚的有效极性，通过该极性来启动帧。SOF 信号对边沿敏感。

在从模式下，音频模块等待一个有效帧来启动发送或接收。SOF 信号与此信号同步。只有通信期间未检测到预期 SOF 信号极性相同的 SOF 信号时，帧同步极性才有效（请参见：“30.2.13 错误标志”）。

在主模式下，每次音频帧完成时均会发送帧同步信号，直至 SAIx_CR1.SAIXEN 位清零。如果前一个音频帧结束时 FIFO 中不存在数据，则将按照“30.2.13 错误标志”所述管理下溢条件，但音频通信流不会中断。

30.2.5.3 帧同步有效电平长度

SAIx_FRCR 寄存器的 FSALL[6:0]位用于配置帧同步信号的有效电平的长度。该长度可设置为 1 到 128 个位时钟周期。

例如，有效长度在 I2S、LSB 或 MSB 对齐模式下为帧长的一半，在 PCM/DSP 或 TDM 模式下为 1 位。

30.2.5.4 帧同步偏移

基于应用中支持的音频协议（例如 I2S 标准协议和 MSB 对齐协议），可以在发送音频帧的最后一位或第一位时将帧同步信号置为有效。通过 SAIx_FRCR.FSOFF 位选择两个配置之一。

30.2.5.5 FS 信号的作用

FS 信号可以有不同的含义，具体取决于 FS 的功能。SAIx_FRCR.FSDEF 位用于选择 FS 信号的含义：

- 0：SOF 信号，例如 PCM/DSP、TDM、AC’97 和音频协议，
- 1：音频帧内的 SOF 信号和通道识别信号，例如 I2S、MSB 或 LSB 对齐协议。

当 FS 信号被视为帧内的 SOF 信号和通道识别信号时，声明的 Slot 数必须是一半用于左通道，一半用于右通道。如果半个音频帧上的位时钟周期数大于某个通道的专用 Slot 数，若 TRIS=0，则 SAIx_CR2 寄存器中的剩余位时钟周期将发送 0。否则若 TRIS=1，SD 线将释放为高阻态。接收时，直到通道发生变化，才会考虑剩余位时钟。

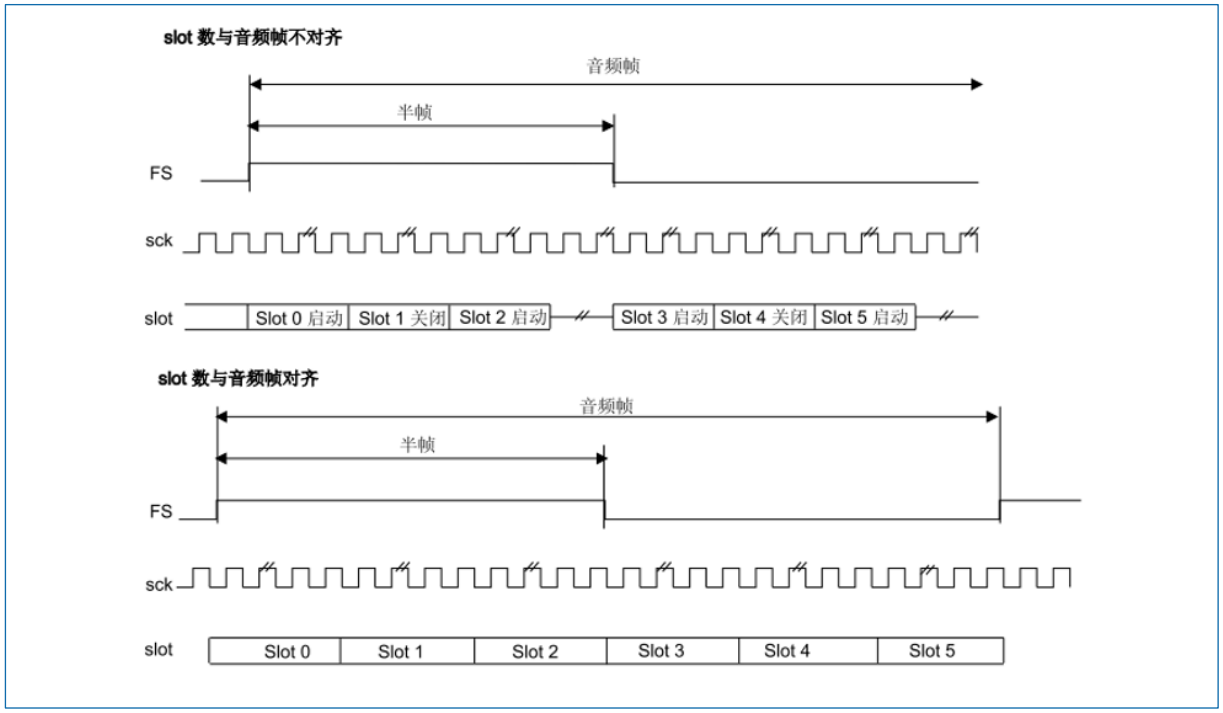


图 30-3 FS 的作用是 SOF 信号+通道识别信号（FSDEF=TRIS=1）

上图的说明：1.帧长度应为偶数。

如果 SAIx_FRCR 中的 FSDEF 位保持清零状态，则 FS 信号等效于 SOF 信号，如果 SAIx_SLOTR 中的 NBSLOT[3:0]位定义的 Slot 数乘以 SAIx_SLOTR 中的 SLOTSZ[1:0]位配置的 Slot 位数所得的结果小于帧大小（SAIx_FRCR.FRL[7:0]位），则：

- 如果 SAIx_CR2.TRIS=0，则最后一个 Slot 后的剩余位将强制为 0，直至发送器的帧结束，
- 如果 TRIS=1，则传输这些剩余位时，数据线将释放为高阻态。在接收模式下，这些位被丢弃。

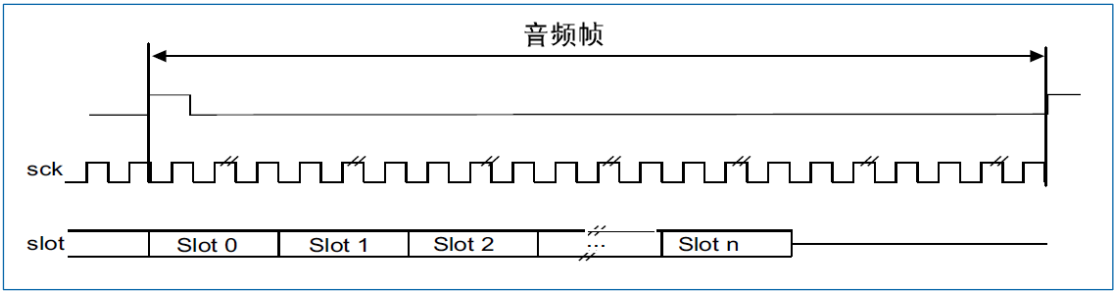


图 30-4 FS 的作用是 SOF 信号（FSDEF=0）

上图的说明：如果 TRIS=0，则 slot n 后的数据=0；如果 TRIS=1，则 slot n 后的 SD 输出释放为高阻态。

在发送模式下将音频模块配置为获取 SD 线上的 SPDIF 输出时，将不使用 FS 信号。相应的 FSI/O 会被释放，留作他用。

30.2.6 Slot 配置

Slot 是音频帧中的基本元素。音频帧中的 Slot 数等于 NBSLOT[3:0]+1。每个音频帧的最大 Slot 数固定为 16。

对于 AC’97 协议或 SPDIF 协议（位 PRTCFG[1:0]=10 或 PRTCFG[1:0]=01 时），Slot 数自动按照协议规范设置，NBSLOT[3:0]的值被忽略。

通过设置 SAIx_SLOTR 寄存器的 SLOTEN[15:0]位，可将各个 Slot 定义为有效 Slot 或无效 Slot。

传输一个无效 Slot 时，SD 数据线将强制为 0 或释放为高阻态，具体取决于 TRIS 位在发送模式下的配置。在接收模式下，从该 Slot 结束后接收的数据被忽略。因此，既不会有 FIFO 访问，也不会有与此无效 Slot 状态相关的 FIFO 读/写请求。

Slot 大小也可配置，如下图所示。通过将 SAIx_SLOTR.SLOTSZ[1:0]位置 1 来选择 Slot 大小。该大小适用于音频帧中的每个 Slot。

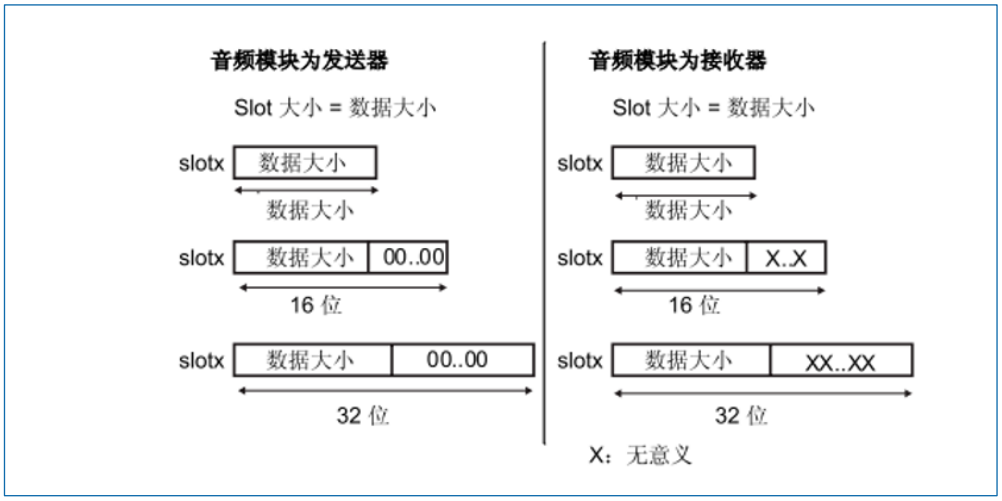


图 30-5 SAIx_SLOTR 中的 FBOFF=0 时的 Slot 大小配置

可以选择 Slot 内要传输的第一个数据位的位置，此偏移通过 SAIx_SLOTR.FBOFF[4:0]位配置。在发送

模式下，将从 Slot 开始时注入 0 值，直至到达此偏移位置。接收时，偏移阶段中的位被忽略。此特性适用于 LSB 对齐协议（如果偏移等于 Slot 大小减去数据大小）。

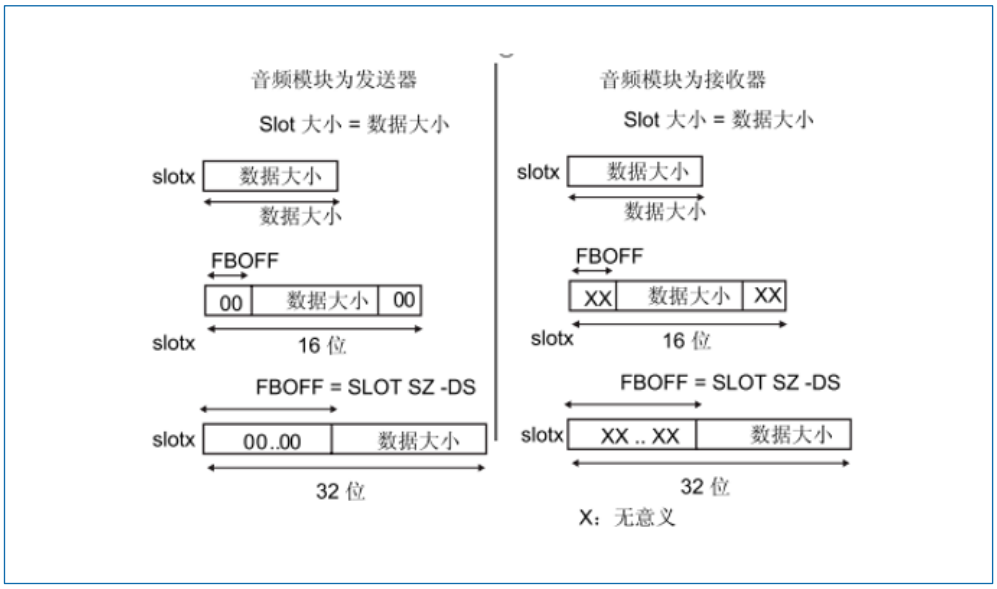


图 30-6 第一位偏移

要避免出现故障 SAI 行为，必须遵循以下条件：

- FBOFF≤（SLOTSZ-DS），
- DS≤SLOTSZ，
- NBSLOTxSLOTSZ≤FRL（帧长度），

SAIx_FRCR.FSDEF 位置 1 时，Slot 数必须为偶数。

在 AC’97 和 SPDIF 协议（位 PRTCFG[1:0]=10 或 PRTCFG[1:0]=01）中，Slot 大小按照“30.2.9 AC’97 链路控制器”中的定义自动设置。

有关 AC’97 模式下时钟发生器编程的详细信息，请参见：“30.2.9 AC’97 链路控制器”；有关 SPDIF 模式下时钟发生器编程的详细信息，请参见：“30.2.10 SPDIF 输出”。

30.2.7 SAI 时钟发生器

每个音频模块都有自己的时钟发生器，旨在使这两个子模块完全独立。这两个时钟发生器的功能没有任何区别。

将音频模块配置为主模块时，时钟发生器将提供通信时钟（位时钟 SCK_x）以及用于外部解码器的主时钟（MCLK_x）。

当音频模块定义为从模块时，时钟发生器将关闭。

下图给出了音频模块时钟发生器的架构。

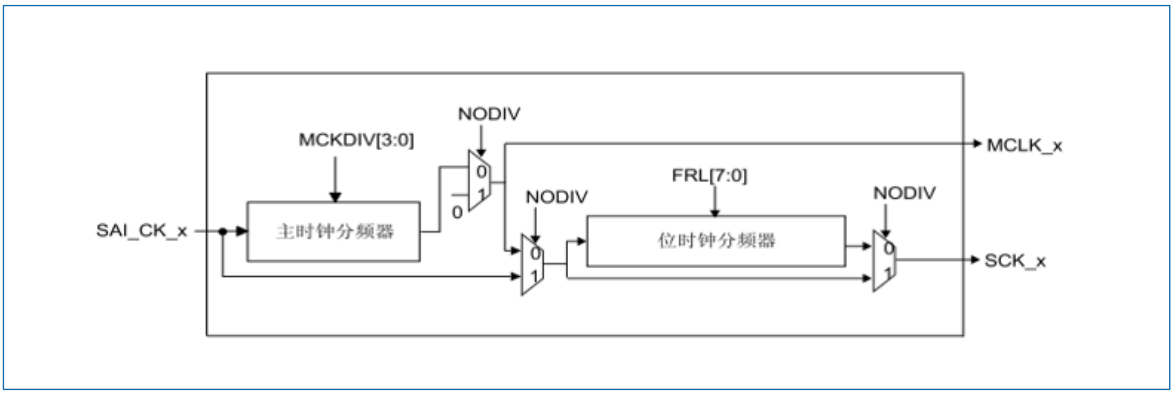


图 30-7 音频模块时钟发生器概览

注意：当 NODIV 置 1 时，如果 MCLK_x 引脚配置为 GPIO 外设中的 SAI 引脚，则其信号电平将置 0。

时钟发生器的时钟源来自芯片时钟控制器。SAI_CK_x 时钟等效于主时钟，可通过 MCKDIV[3:0]位为外部解码器分频：

- 如果 MCKDIV[3:0]不等于 0000，则 MCLK_x=SAI_CK_x/（MCKDIV[3:0]*2）
- 如果 MCKDIV[3:0]等于 0000，则 MCLK_x=SAI_CK_x

MCLK_x 信号仅在 TDM 模式中使用。

分频必须均匀，使 MCLK 输出上和 SCK_x 时钟上都保持 50%的占空比。如果位 MCKDIV[3:0]=0000，采用一分频可使 MCLK_x 等于 SAI_CK_x。

在 SAI 中，使用单一比率 MCLK/FS=256。大多数情况下，将遇到三个频率范围，如下表所示。

表 30-2 可能的音频采样范围示例

输入 SAI_CK_x 时钟频率	可获得的常见音频采样频率（F_S）	MCKDIV[3:0]
192kHz x 256	192kHz	MCKDIV[3:0]=0000
	96kHz	MCKDIV[3:0]=0001
	48kHz	MCKDIV[3:0]=0010
	16kHz	MCKDIV[3:0]=0110
	8kHz	MCKDIV[3:0]=1100
44.1kHz x 256	44.1kHz	MCKDIV[3:0]=0000
	22.05kHz	MCKDIV[3:0]=0001
	11.025kHz	MCKDIV[3:0]=0010
SAI_CK_x=MCLK（1）	MCLK	MCKDIV[3:0]=0000

(1). 当芯片时钟控制器选择一个外部时钟源（MCLK）而非 PLL 时钟时，会出现这种情况。

如果相应音频模块声明为主模块且 SAIx_CR1.NODIV=0，可通过 I/O 口为外部解码器生成主时钟。在从模式下，由于时钟发生器关闭，将忽略这 NODIV 中设置的值，且 MCLK_x I/O 引脚用作通用 I/O 使用。

位时钟通过主时钟导出。位时钟分频器按照以下公式设置位时钟（SCK_x）和主时钟（MCLK_x）间的分频系数：

SCK_x=MCLK_x*（FRL[7:0]+1）/256

其中：

- 256 是 MCLK_x 和音频采样频率之间的固定比率。
- FRL[7:0]是音频帧中的位时钟周期-1，在 SAIx_FRCR 寄存器中配置。

主模式下，（FRL[7:0]+1）必须等于 2 的几次幂（请参见：“30.2.5 帧同步”），以通过位时钟周期获得偶数个 MCLK_x 脉冲。位时钟（SCK_x）上将保证 50%的占空比。

SAI_CK_x 时钟还可等于位时钟频率。在这种情况下，SAIx_CR1.NODIV 位应置 1，MCKDIV 分频器内的值以及位时钟分频器内的值将被忽略。此时，每个帧的位数完全可配置，而无需等于 2 的几次幂。

SCK 上的位时钟选通边沿可通过 SAIx_CR1.CKSTR 位配置。

有关 SPDIF 模式下的时钟发生器编程的详细信息，请参见：“30.2.10 SPDIF 输出”。

30.2.8 内部 FIFO

SAI 中的每个音频模块都有自己的 FIFO。根据模块是定义为发送器还是接收器，可相应的读取或写入其 FIFO。因此只存在一个 FIFO 请求，它与 SAIx_SR.FREQ 位相关。

如果 SAIx_IM.FREQIE 位使能，将产生中断。这取决于：

- FIFO 阈值设置（SAIx_CR2 中的 FTH[2:0]位）
- 通信方向（发送器或接收器）。请参见：“30.2.8.1 在发送模式下产生中断”和“30.2.8.2 在接收模式下产生中断”。

30.2.8.1 在发送模式下产生中断

根据发送模式下的 FIFO 配置产生中断：

- 当 SAIx_CR2 寄存器中的 FIFO 阈值位配置为 FIFO 为空时（FTH[2:0]置为 000b），如果寄存器中没有数据（SAIx_SR.FLVL[2:0]位小于 001b），将产生中断（SAIx_SR.FREQ 位由硬件置 1）。当 FIFO 不再为空时（SAIx_SR.FLVL[2:0]位不是 000b），即 FIFO 中存储一个或多个数据时，此中断（SAIx_SR.FREQ 位）由硬件清零。
- 当 SAIx_CR2 寄存器中的 FIFO 阈值位配置为 FIFO 四分之一满时（FTH[2:0]置为 001b），如果不到四分之一的 FIFO 包含数据（SAIx_SR.FLVL[2:0]位小于 010b），将产生中断（SAIx_SR.FREQ 位由硬件置 1）。当至少四分之一的 FIFO 包含数据时（SAIx_SR.FLVL[2:0]位大于等于 010b），此中断（SAIx_SR.FREQ 位）由硬件清零。
- 当 SAIx_CR2 寄存器中的 FIFO 阈值位配置为 FIFO 半满时（FTH[2:0]置为 010b），如果不到一半的 FIFO 包含数据（SAIx_SR.FLVL[2:0]位小于 011b），将产生中断（SAIx_SR.FREQ 位由硬件置 1）。当至少一半的 FIFO 包含数据时（SAIx_SR.FLVL[2:0]位大于或等于 011b），此中断（SAIx_SR.FREQ 位）由硬件清零。
- 当 SAIx_CR2 寄存器中的 FIFO 阈值位配置为 FIFO 四分之三满时（FTH[2:0]置为 011b），如果不到四分之三的 FIFO 包含数据（SAIx_SR.FLVL[2:0]位小于 100b），将产生中断（SAIx_SR.FREQ 位由硬件置 1）。当至少四分之三的 FIFO 包含数据时（SAIx_SR.FLVL[2:0]位大于或等于 100b），此中断（SAIx_SR.FREQ 位）由硬件清零。
- 当 SAIx_CR2 寄存器中的 FIFO 阈值位配置为 FIFO 为满时（FTH[2:0]置为 100b），如果 FIFO 不满（SAIx_SR 中的 FLVL[2:0]位小于 101b），将产生中断（SAIx_SR.FREQ 位由硬件置 1）。当 FIFO 已满时（SAIx_SR 中的 FLVL[2:0]位等于值 101b），此中断（SAIx_SR.FREQ 位）由硬件清零。

30.2.8.2 在接收模式下产生中断

根据接收模式下的 FIFO 配置产生中断：

- 当 SAIx_CR2 寄存器中的 FIFO 阈值位配置为 FIFO 为空时（FTH[2:0]置为 000b），如果 SAIx_DR 寄存器中至少有一个数据（SAIx_SR 中的 FLVL[2:0]位大于或等于 001b），将产生中断（SAIx_SR.FREQ 位由硬件置 1）。当 FIFO 变为空时（SAIx_SR.FLVL[2:0]位等于 000b），即 FIFO 中未存储数据时，此中断（SAIx_SR.FREQ 位）由硬件清零。

- 当 SAIx_CR2 寄存器中的 FIFO 阈值位配置为 FIFO 四分之一满时（FTH[2:0]置为 001b），如果至少有四分之一的 FIFO 数据单元可用（SAIx_SR.FLVL[2:0]位大于或等于 010b），将产生中断（SAIx_SR.FREQ 位由硬件置 1）。当不到四分之一的 FIFO 数据单元可用时（SAIx_SR.FLVL[2:0]位小于 010b），此中断（SAIx_SR.FREQ 位）由硬件清零。
- 当 SAIx_CR2 寄存器中的 FIFO 阈值位配置为 FIFO 半满时（FTH[2:0]置为 010b），如果至少有一半的 FIFO 数据单元可用（SAIx_SR.FLVL[2:0]位大于或等于 011b），将产生中断（SAIx_SR.FREQ 位由硬件置 1）。当不到一半的 FIFO 数据单元可用时（SAIx_SR.FLVL[2:0]位小于 011b），此中断（SAIx_SR.FREQ 位）由硬件清零。
- 当 SAIx_CR2 寄存器中的 FIFO 阈值位配置为 FIFO 四分之三满时（FTH[2:0]置为 011b），如果至少有四分之三的 FIFO 数据单元可用（SAIx_SR.FLVL[2:0]位大于或等于 100b），将产生中断（SAIx_SR.FREQ 位由硬件置 1）。当不到四分之三的 FIFO 数据单元可用时（SAIx_SR.FLVL[2:0]位小于 100b），此中断（SAIx_SR.FREQ 位）由硬件清零。
- 当 SAIx_CR2 寄存器中的 FIFO 阈值位配置为 FIFO 满时（FTH[2:0]置为 100b），如果 FIFO 已满（SAIx_SR.FLVL[2:0]位等于 101b），将产生中断（SAIx_SR.FREQ 位由硬件置 1）。当 FIFO 不满时（SAIx_SR.FLVL[2:0]位小于 101b），此中断（SAIx_SR.FREQ 位）由硬件清零。

与中断的产生类似，如果 SAIx_CR1.DMAEN 位置 1，则 SAI 可使用 DMA。FREQ 位的有效机制与上述 FREQIE 的中断产生机制相同。

每个 FIFO 均是一个 8 字 FIFO。无论访问大小为何，每次对 FIFO 进行读写时，均针对 1 个字的 FIFO 单元进行操作。每个 FIFO 字包含一个音频 Slot。每次访问 SAIx_DR 寄存器后，FIFO 指针递增一个字。

数据应以右对齐方式写入 SAIx_DR。

将 SAIx_CR2.FFLUSH 位置 1 禁止 SAI 后，可重新初始化 FIFO 指针。如果 FFLUSH 在 SAI 使能情况下置 1，则 FIFO 中的数据将自动清除。

30.2.9 AC’97 链路控制器

SAI 可用作 AC’97 链路控制器。在此协议中：

- Slot 数和 Slot 大小固定。
- 帧同步信号定义完善并且波形固定。

要选择此协议，可将 SAIx_CR1.PRTCFG[1:0]位置为 10。选择 AC’97 模式时，只能使用 16 位或 20 位的数据大小，否则将无法保证 SAI 运行正常。

- 因此，NBSLOT[3:0]和 SLOTSZ[1:0]位将被忽略。
- Slot 数固定为 13。第一个 Slot 为 16 位宽，所有其它 Slot 均为 20 位宽（数据 Slot）。
- SAIx_SLOTR.FBOFF[4:0]位被忽略。
- SAIx_FRCR 寄存器被忽略。
- 未使用 MCLK。

无论采用主配置还是从配置，由于 AC’97 链路控制器驱动 FS 信号，异步模块发出的 FS 信号将自动配置为输出。

下图给出了 AC’97 音频帧的结构。

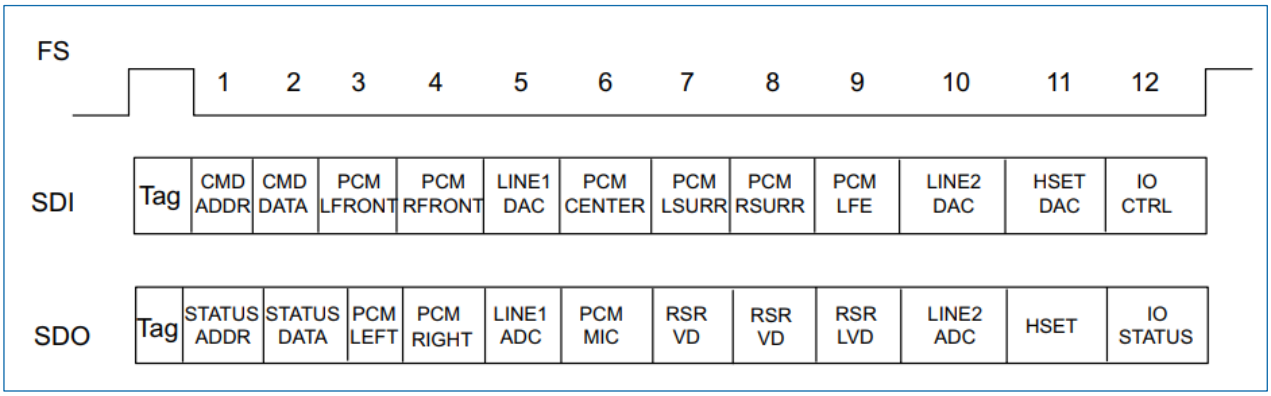


图 30-8 AC’97 音频帧

注意：在 AC’97 协议中，TAG 的位 2 将保留（始终为 0），因此无论 SAI FIFO 中写入何值，TAG 的位 2 均强制为 0。有关 TAG 表示的详细信息，请参见 AC’97 协议标准。

可将一个 SAI 用于 AC’97 点对点通信。

使用两个 SAI（对于具有两个嵌入式 SAI 的器件）可以控制三个外部 AC’97 解码器，如下图所示。

在 SAI1 中，必须将音频模块 A 声明为异步主发送器，而将音频模块 B 定义为从接收器，并在内部与音频模块 A 同步。

在从接收器模式下，同时针对音频模块 A 和音频模块 B 将 SAI2 配置为与外部 SAI 同步。

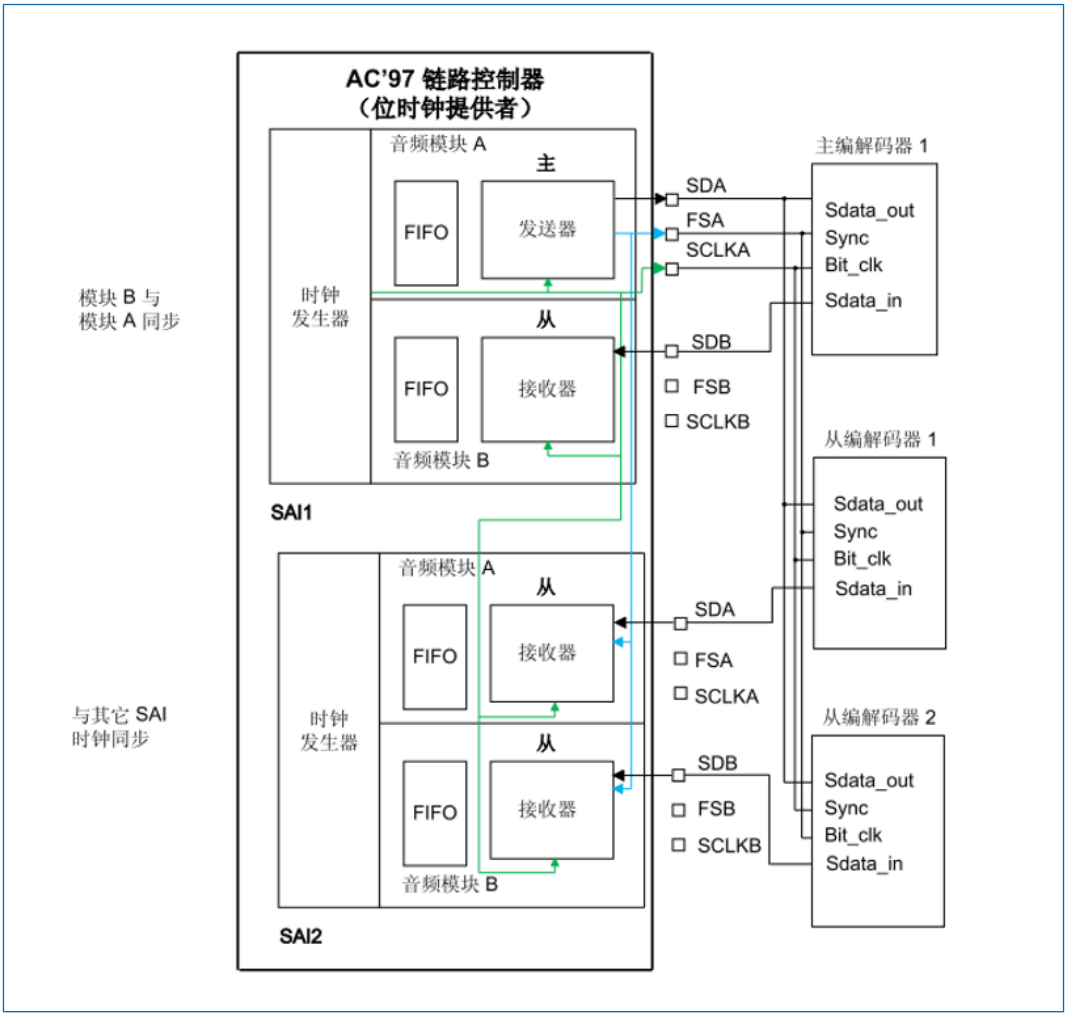


图 30-9 具有至少 2 个嵌入式 SAI（三个外部 AC’97 解码器）的器件上的典型 AC’97 配置示例

在接收器模式下，用作 AC’97 链路控制器的 SAI 无需 FIFO 请求，因此当 Slot0 中的编码解码器就绪

位解码为低电平时，FIFO 中不存储任何数据。如果 SAIx_IM.CNRDYIE 位使能，则 SAIx_SR.CNRDY 将置 1 并会产生一个中断。此标志专用于 AC’97 协议。

AC’97 模式下的时钟发生器编程

在 AC’97 模式下，帧长度固定为 256 位，其频率应设置为 48kHz。章节“30.2.7 SAI 时钟发生器”中给出的规则应在 FRL=255 时使用，以生成适当的帧速率（F_{FS_x}）。

30.2.10 SPDIF 输出

SPDIF 接口仅在发送模式下可用。它支持 IEC60958 音频标准。

要选择 SPDIF 模式，需将 SAIx_CR1.PRTCFG[1:0]位置为 01。

对于 SPDIF 协议：

- 仅使能 SD 数据线。
- 释放 FS、SCK、MCLK I/O 引脚。
- 为使能 SAI 的时钟发生器并管理 SD 线上的数据速率，将 MODE[1]位强制设置为 0 以选择主模式。
- 数据大小强制设置为 24 位。忽略在 SAIx_CR1 寄存器的 DS[2:0]位中设置的值。
- 必须配置时钟发生器以定义符号率，已知位时钟频率应为符号率的两倍。数据采用曼彻斯特协议进行编码。
- 忽略 SAIx_FRCR 和 SAIx_SLOTR 寄存器。在内部配置 SAI 使其符合 SPDIF 协议要求，如下图所示。

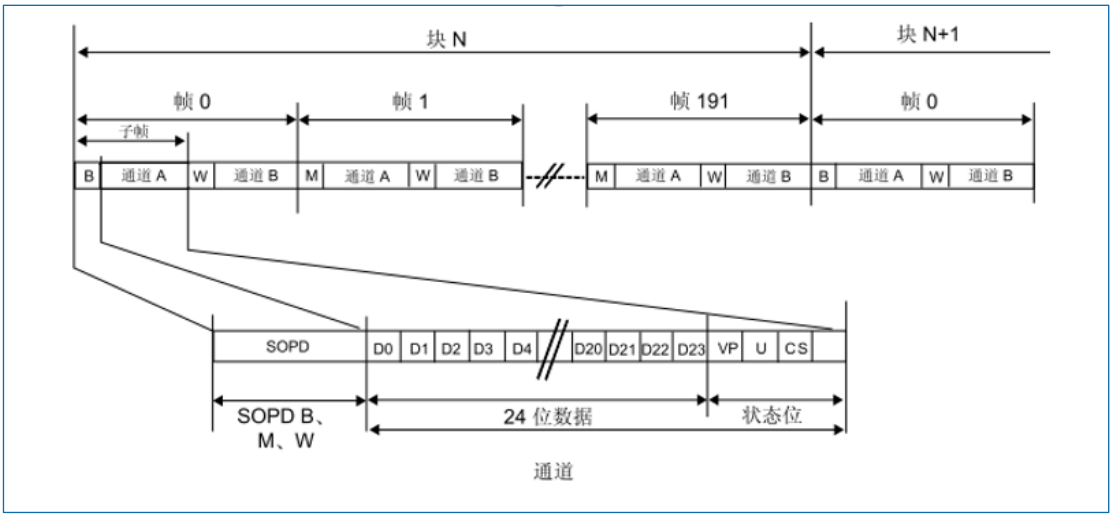


图 30-10 SPDIF 格式

SPDIF 块包含 192 个帧。每个帧由两个 32 位的子帧构成，通常一个子帧用于左通道，一个用于右通道。每个子帧包含一个 SOPD 模式（4 位），用于指定该子帧是否为块的开始（并用于识别通道 A），或者子帧是否标识块中某处的通道 A，或者子帧是否指代通道 B（请参见下表）。接下来的 28 位是通道信息，由 24 个数据位和 4 个状态位组成。

表 30-3 SOPD 模式

SOPD	报头编码		说明
	最后一位是 0	最后一位是 1	
B	11101000	00010111	块开始处的通道 A 数据
W	11100100	00011011	块中某处的通道 B 数据

SOPD	报头编码		说明
	最后一位是 0	最后一位是 1	
M	11100010	00011101	通道 A 数据

必须按如下方式填充 SAIx_DR 中存储的数据：

- SAIx_DR[26:24]包含通道状态位、用户位和有效性位。
- SAIx_DR[23:0]包含所考虑通道的 24 位数据。

如果数据大小为 20 位，应将数据映射到 SAIx_DR[23:4]上。如果数据大小为 16 位，应将数据映射到 SAIx_DR[23:8]上。SAIx_DR[23]始终代表 MSB。

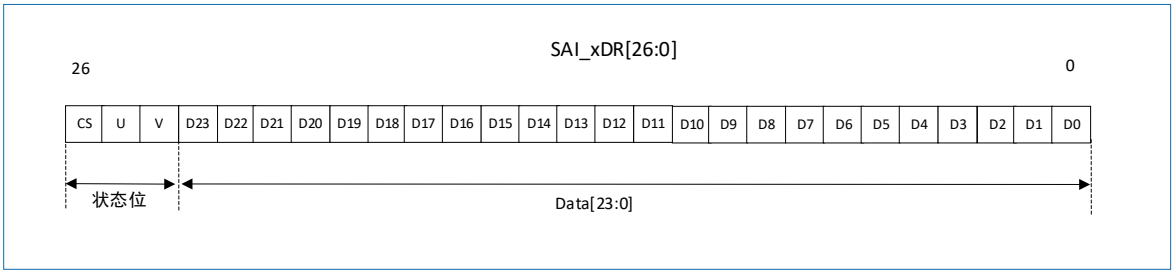


图 30-11 SAIx_DR 寄存器定序

注意：执行传输时，LSB 始终优先。

SAI 首先在块中发送每个子帧的适当报头。随后在 SD 线上发送 SAIx_DR（以曼彻斯特协议进行编码）。SAI 通过传输按下表所述计算的奇偶校验位来结束子帧。

表 30-4 奇偶校验位计算

SAIx_DR[26:0]	传输奇偶校验位 P 值
奇数个 0	0
奇数个 1	1

在 SPDIF 模式下，下溢错误标志是在 SAIx_SR 寄存器唯一用到的错误标志。因为 SAI 只能在发送模式下工作。因此，要从下溢中断或下溢状态位检测到的下溢错误中恢复，应按顺序执行以下步骤：

1. 如果已使用 DMA，则禁止 DMA 流（通过 DMA 外设）。
2. 禁止 SAI 并通过轮询 SAIx_CR1.SAIXEN 位确认已从物理上禁止外设。
3. 将 SAIx_CLRFR.COVRUNDR 写 1（清除 SAIx_SR.OVRUDR 标志）。
4. 通过将 SAIx_CR2.FFLUSH 位置 1 来刷新 FIFO。
5. 软件需要指向与新块开始（报头 B 的数据）相对应的后续数据的地址。如果已使用 DMA，应相应地更新 DMA 源起始地址指针。
6. 如果使用 DMA 来管理根据新的源起始地址来进行的传输数据，则必须再次使能 DMA 流（DMA 外设）。
7. 通过将 SAIx_CR1.SAIXEN 位置 1 再次使能 SAI。

SPDIF 发生器模式下的时钟发生器编程

对于 SPDIF 发生器，SAI 应提供与符号率两倍的位时钟。下表给出了符号率相对于音频采样率的通常示例。

表 30-5 音频采样频率与符号率（SHARK）

音频采样频率（F_S）	符号率
44.1kHz	2.8224MHz
48kHz	3.072MHz
96kHz	6.144MHz
192kHz	12.288MHz

通常，音频采样率（F_S）与位时钟速率（F_{SCK_X}）之间的关系由以下公式给出：

$$F_S = \frac{F_{SCK\_x}}{128}$$

F_{SCK_X} 通过如下公式获取：

$$F_{SCK\_x} = \frac{F_{SAI\_CK\_x}}{MCKDIV}$$

30.2.11 PDM

只有 SAIA 接口支持 PDM（pulse-density modulated）接口作为接收端，处理来自数字麦克风等设备的 1 比特信号流。

其功能特性如下：

- 使用两级滤波器，输出频率抽取倍数（64 倍、128 倍、256 倍）可配。
- 输出数据位宽 20 位，高位对齐 32 位 APB 总线。
- 支持 PDM 双通道输入和单通道输入。
- 共两组可独立使用的 PDM 接口。
- 音频信号频率范围 20Hz~20kHz，SNR≥93dB。

PDM 结构框图

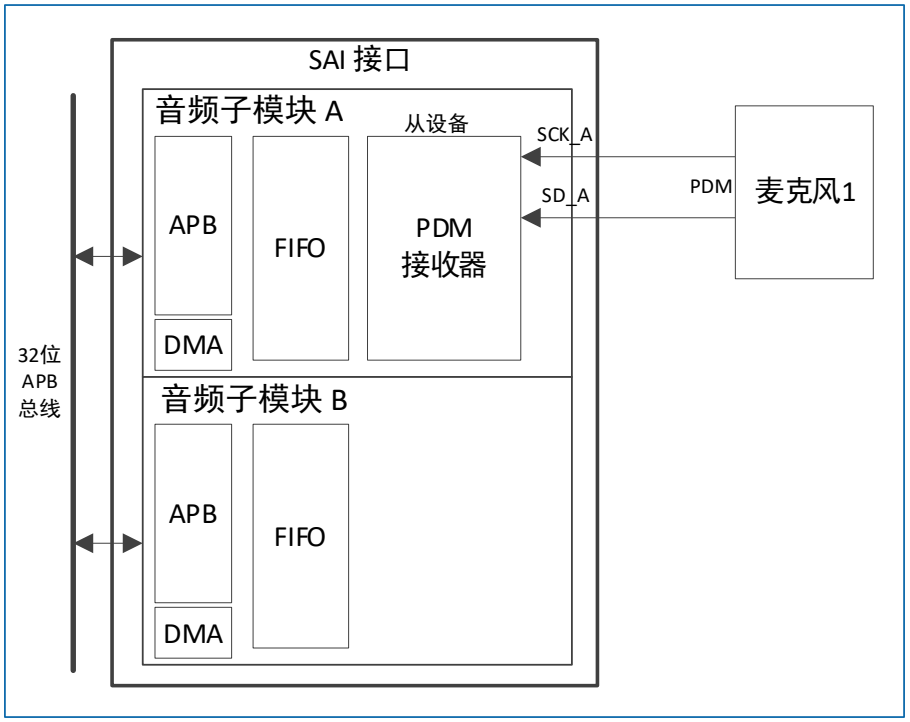


图 30-12 PDM 框图

PDM 模式配置

需要对 SAI 模块 A 进行如下表格的配置以支持 PDM 模式：

表 30-6 PDM 模式配置步骤

1	配置 PRTCFG[1:0]=11，选择 PDM 协议。
2	配置 NBSLOT=0，对应 1 个 slot，接收单声道 （配置 CKSTR=0，时钟下沿采集，接收右声道数据） （配置 CKSTR=1，时钟上沿采集，接收左声道数据） 配置 NBSLOT=1，对应 2 个 slot，接收双声道
3	配置 SLOTSZ=10，对应一个 slot 为 32 位位宽
4	配置 DS=101，数据位宽 20 位
5	配置 MODE=11，从接收器模式

时序图

PDM 支持双通道数据采集，时钟上沿采集左声道数据，时钟下沿采集右声道数据（时序图如见图 30-13）；也支持单通道数据采集，具体配置见上表（表 30-6）。

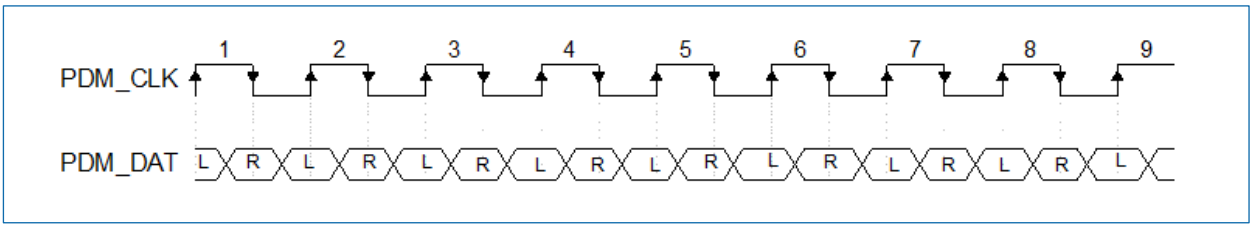


图 30-13 PDM 时序图（以双通道为例）

音频信号频率与过采样率：

可通过 SAI_ACR2 寄存器的 DEC[1:0]配置滤波器抽取倍数。音频信号频率与过采样率之间的对应关系入下表所示：

表 30-7 PDM 音频信号频率与过采样率

音频信号频率（kHz）	PDM 输入时钟率（MHz）		
	64 x 过采样率	128 x 过采样率	256 x 过采样率
8	-	-	2.048
16	-	2.048	4.096
32	2.048	4.096	8.192
44.1	2.8224	5.6448	11.2896
48	3.072	6.144	12.288
88.2	5.6448	11.2896	22.5792
96	6.144	12.288	24.576

源码补码转换功能

该 PDM 模式新增源码补码的转换功能：

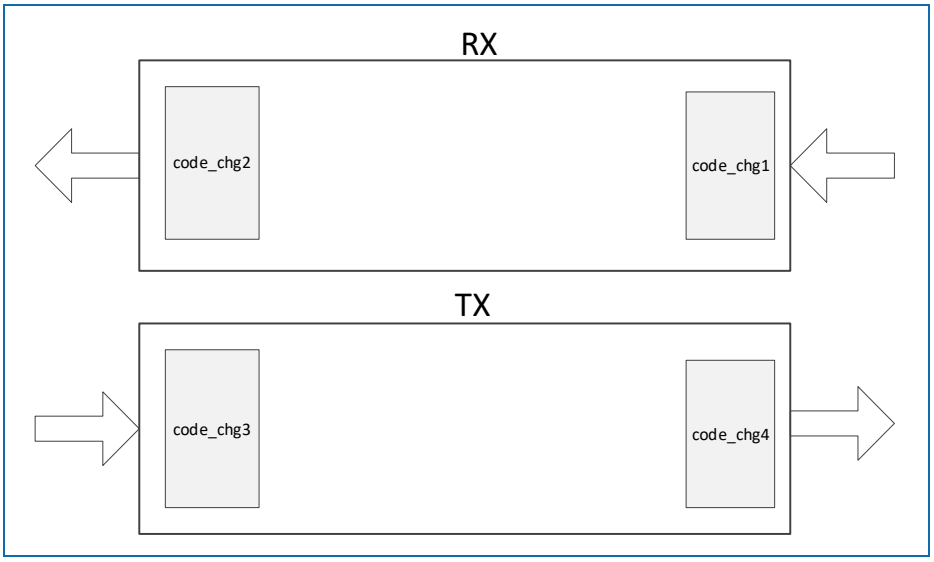


图 30-14 源码补码转换功能

源码补码转换功能可对以下类型数据进行预转换：

- 带符号位的源码
- 带符号位的补码
- 不带符号位的源码
- 不带符号位的补码

转换逻辑：

- 数据 1 和 2 之间可以互相转换；
- 数据 3 和 4 之间可以互相转换（有效位取反操作）。

总共 4 个转换逻辑单元（Code_chgx），处于 RX/TX 的接收和发送两头（见图 30-14），每个转换逻辑单元均可单独配置。详细的设置，参见“30.4.2 配置寄存器 2（SAIx_CR2）（x=A..B）”。

30.2.12 其他特殊功能

根据所选的音频协议，SAI 接口内嵌了一些特定的实用功能。这些功能可通过 SAIx_CR2 寄存器的特定位来访问。

30.2.12.1 静音模式

当音频子模块用作发送器或接收器时，可使用静音模式。

发送模式下的音频子模块

在发送模式下，可随时选择静音模式。静音模式对于全部音频帧均有效。在帧传输过程中将 SAIx_CR2 寄存器的 MUTE 置 1，将使能静音模式。

该静音模式位仅在帧结束时选通。当帧结束时置 1，静音模式将在新的音频帧开始时激活，并持续整个帧长度，直至此次帧结束；然后再次选通该位，以确保下一帧仍为静音帧。

如果通过 SAIx_SLOTR 寄存器的 NBSLOT[3:0]位设置的 Slot 数小于或等于 2，可指定静音模式下发送的值是否为 0 或该值是否为每个 Slot 的最后一个值。通过 SAIx_CR2.MUTEVAL 位进行选择。

如果在 SAIx_SLOTR 寄存器的 NBSLOT[3:0]位中设置的 Slot 数大于 2，由于在各 Slot 的每位上都发送值 0，因此 SAIx_CR2 中的 MUTEVAL 位没有意义。

在静音模式下，FIFO 指针仍递增，这意味着将丢弃 FIFO 中请求在静音模式下传输的数据。

接收模式下的音频子模块

在接收模式下，对于给定数量的连续音频帧（SAIx_CR2.MUTECNT[5:0]位），如果在音频帧所有声明的有效 Slot 接收到 0，则可检测到从外部发送器发来的静音模式。

检测到相应数量的静音帧时，SAIx_SR.MUTEDET 标志置 1 并会在 SAIx_CR2 中的 MUTEDETIE 位置 1 情况下产生中断。

在音频子模块禁止时或在有效 Slot 内至少接收到音频帧中的一个数据时，静音帧计数器清零。计数器达到 MUTECNT[5:0]位中指定的值时，仅产生一次中断。随后中断事件在计数器清零时重新初始化。

注意：静音模式不可用于 SPDIF 音频模块。

30.2.12.2 单声道/立体声模式

在发送模式下，如果 Slot 数等于 2（SAIx_SLOTR 中的 NBSLOT[3:0]=0001），则可寻址单声道模式，而无需在存储器中进行任何数据预处理。在这种情况下，由于发送时 Slot0 的数据被复制到数据 Slot1 中，FIFO 的访问时间将减少一半。

要使能单声道模式，需要：

1. 将 SAIx_CR1.MONO 位置。
2. 将 SAIx_SLOTR 中的 NBSLOT 置 1，并将 SLOTEN 设置 3。

在接收模式下，MONO 位可置 1 并且仅在 Slot 数等于 2（与发送模式下相同）时才有意义。当该位置 1 时，只有 Slot0 的数据将存储到 FIFO 中。Slot1 的数据由于被认为与前一个 Slot 的数据相同而被丢弃。如果接收模式下的数据流量是左声道数据和右声道数据明显不同的真立体声音频流，则 MONO 位没有意义。由软件完成从输出立体声文件到等效单声道文件的转换。

30.2.12.3 压扩模式

移动通信应用可能需要通过数据压扩算法处理待发送或待接收的数据。

应用可根据 SAIx_CR2.COMP[1:0]位（仅当选择 TDM 模式时使用），选择在 SD 串行输出线（压缩）发送数据前是否处理数据，以及是否在 SD 串行输入线（扩展）接收数据后扩展数据，如下图所示。所支持的两个压扩模式是 μ-Law 和 A-Law，二者是 CCITTG.711 推荐标准的一部分。

美国和日本采用的压扩标准是 μ-Law，该标准允许 14 位动态范围（SAIx_CR2.COMP[1:0]=10）。

欧洲压扩标准是 A-Law，该标准支持 13 位动态范围（SAIx_CR2.COMP[1:0]=11）。

可根据 1 的补码或 2 的补码表示来计算 μ-Law 或 A-Law 压扩标准，具体取决于 SAIx_CR2.CPL 位设置。

在 μ-Law 和 A-Law 标准中，数据将编码为采用 MSB 对齐的 8 位。压扩数据始终为 8 位宽。因此，当 SAI 音频模块使能（SAIx_CR1.SAIXEN=1）并且通过 COMP[1:0]位选择这两个压扩模式之一时，SAIx_CR1.DS[2:0]位将强制为 010。

如果无需压扩处理，则 COMP[1:0]位应保持清零。

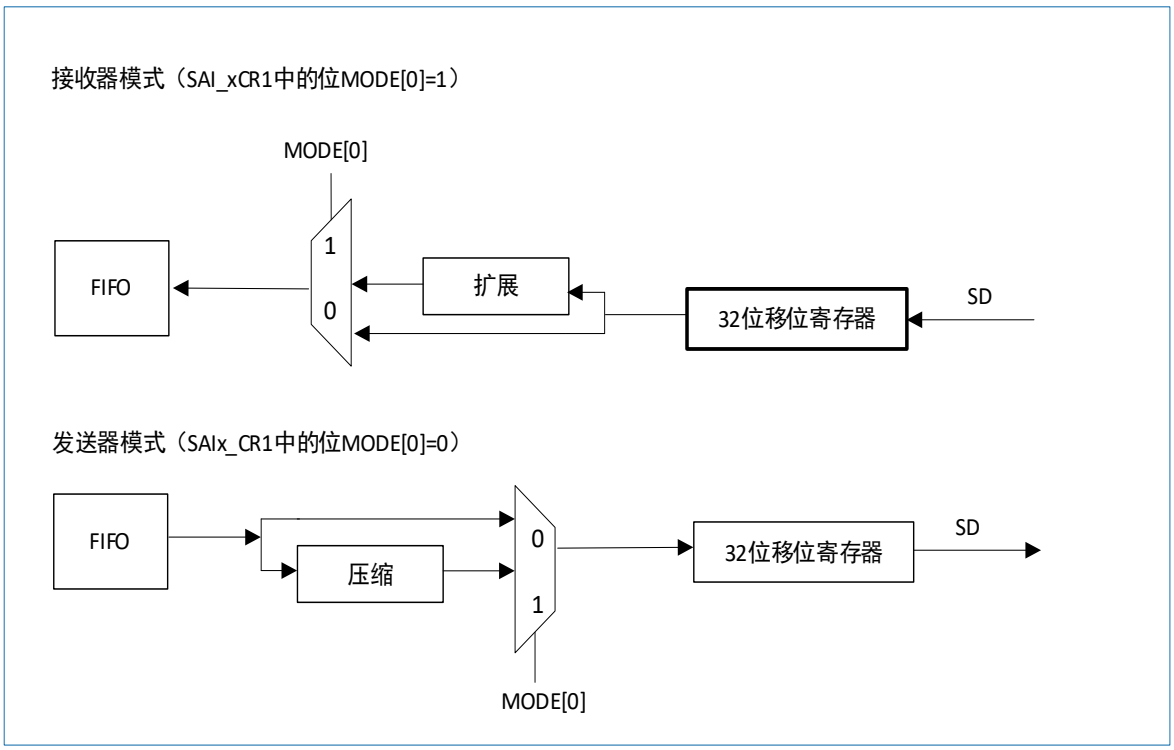


图 30-15 SAI 的音频模块中的数据压扩硬件

通过 SAIx_CR2 自动选择扩展模式和压缩模式：

- 如果 SAI 音频模块配置为发送器，且 SAIx_CR2.COMP[1]位置 1，将采用压缩模式。
- 如果 SAI 音频模块声明为接收器，将采用扩展算法。

无效 Slot 上的输出数据线管理

在发送模式下，当在数据线上发送无效 Slot 时，可选择 SD 线输出的行为（通过 TRIS 位实现）。

- 发送无效 Slot 时 SAI 将 SD 输出线强制为 0。
- 该输出线在最有一个数据位传输结束时释放为高阻态，为另一个连接此节点的发送器释放该数据线。

切记不要让两个发送器同时驱动同一个 SD 输出引脚，否则会导致短路。为确保存在发送间隙，如果数据低于 32 位，可通过在 SAIx_SLOTR 寄存器中设置 SLOTSZ[1:0]=10 将数据扩展到 32 位。随后，如果下一 Slot 声明为无效，则 SD 输出引脚将在有效 Slot 的 LSD 结束时（将数据扩展到 32 位的填 0 阶段）置为三态。

此外，如果 Slot 数乘以 Slot 大小所得结果小于帧长度，则在通过填 0 来补充音频帧结束时，SD 输出线置为三态。

下图说明了这些行为。

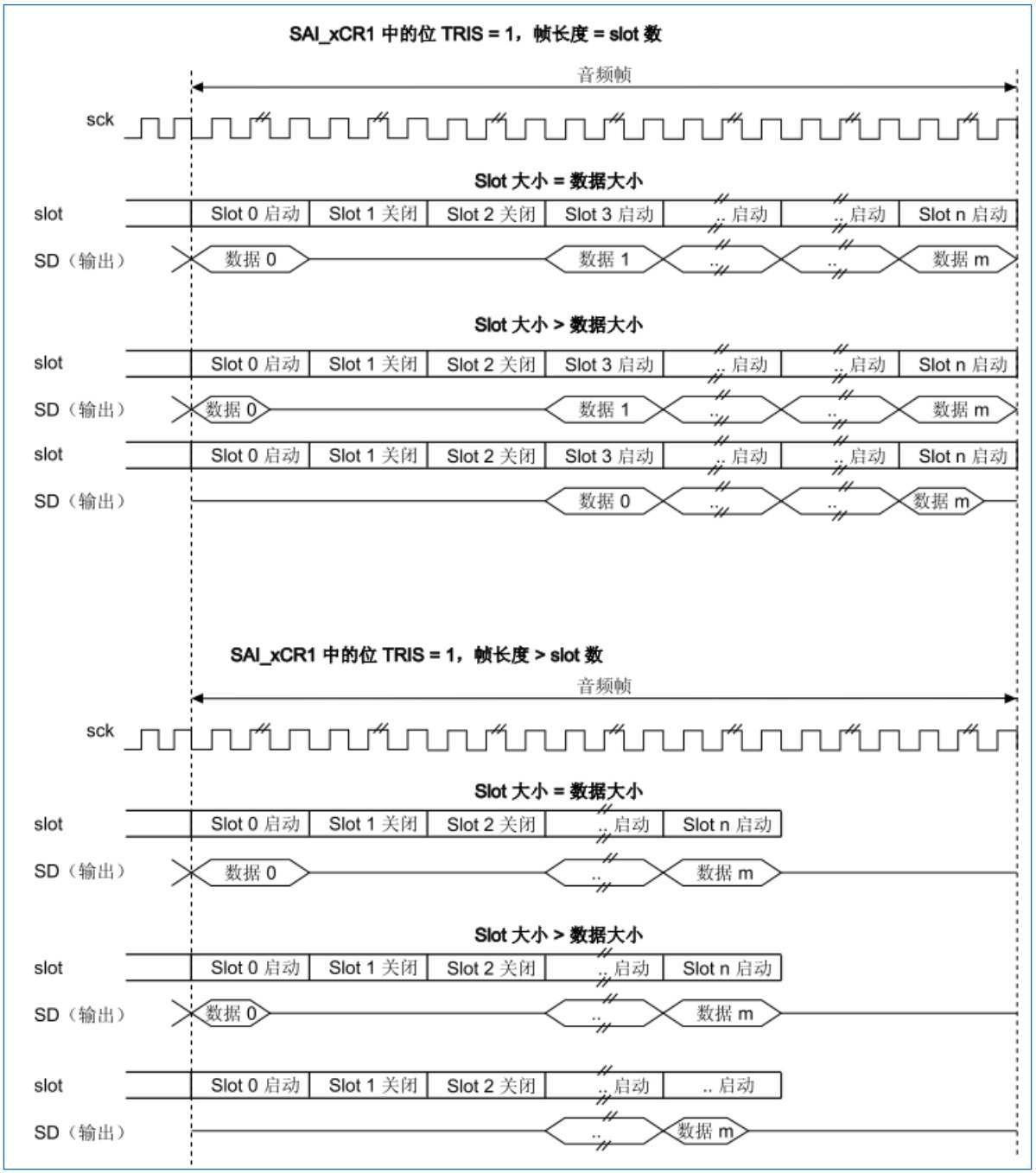


图 30-16 发送无效 Slot 时 SD 输出线上的三态策略

当所选音频协议使用 FS 信号作为 SOF 信号或通道识别信号（SAIx_FRCR.FSDEF=1）时，将按照下图管理三态模式（其中，SAIx_CR1.TRIS=1，FSDEF=1，半帧长大于 Slot 数/2 且 NBSLOT=6）。

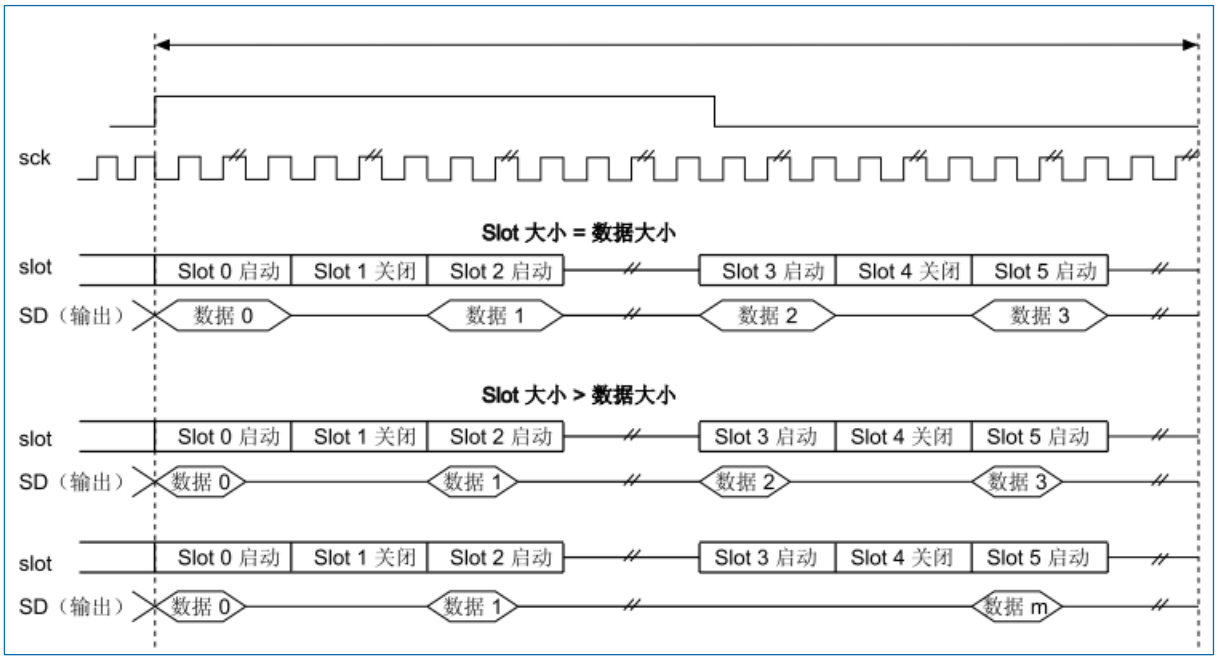


图 30-17 采用 I2S 等协议时输出数据线上的三态策略

如果 SAIx_CR2.TRIS 位清零，上面两幅图上的 SD 输出线上的所有高阻态将替换为使用值 0 驱动。

30.2.13 错误标志

SAI 使用以下错误标志：

- FIFO 上溢/下溢
- 帧同步提前检测
- 帧同步滞后检测
- 编解码器未就绪（仅限 AC’97）
- 主模式时钟配置错误

30.2.13.1 FIFO 上溢/下溢（OVRUDR）

FIFO 上溢/下溢位是 SAIx_SR.OVRUDR 位。由于音频模块既可作为接收器，又可作为发送器，并且指定 SAI 中的每个音频模块都具有自己的 SAIx_SR 寄存器，因此上溢或下溢错误共用同一位。

上溢

若音频模块配置为接收器，则在 FIFO 已满且无法再存储接收数据的情况下又收到音频帧数据时，将出现上溢情况。这种情况下，接收数据将丢失，SAIx_SR.OVRUDR 标志置 1；如果 SAIx_IM.OVRUDRIE 位置 1，还将生成中断。内部将记录发生上溢时的 Slot 编号。在 FIFO 释放出空间存储新数据之前，无法再存储更多数据。在 FIFO 释放了至少一个数据的空间时，SAI 音频模块接收器将从检测到上溢后内部记录的 Slot 编号开始接收来自新音频帧的新数据，这样可避免目标存储器中出现数据 Slot 不对齐的情况（参见图 30-18）。

SAIx_CLRFR.COVRUDR 位置 1 将清除 OVRUDR 标志。

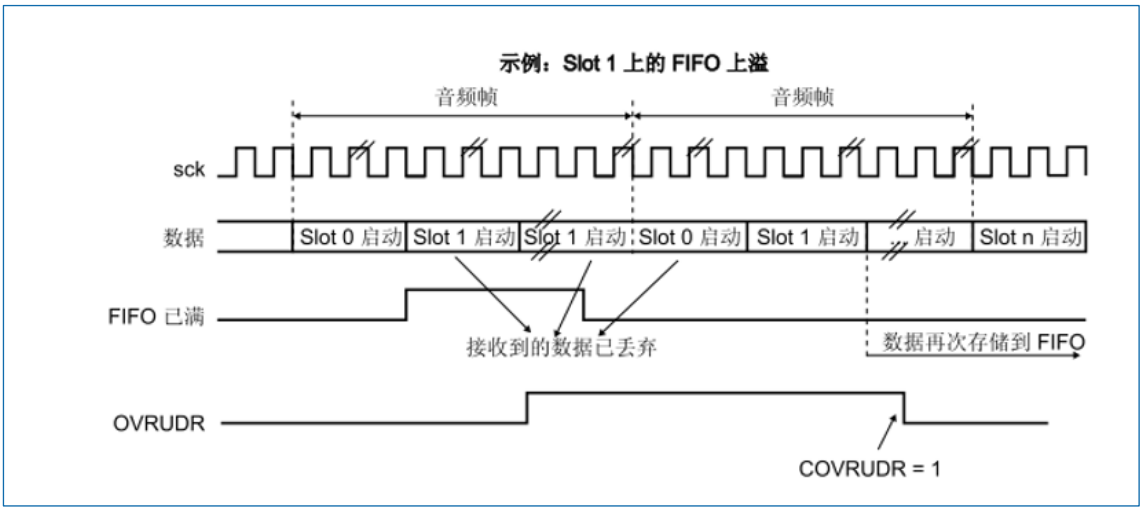


图 30-18 上溢错误检测

下溢

当 SAI 中的音频模块用作发送器时，如果需要发送数据时 FIFO 为空，则可能出现下溢。如果检测到下溢，则将存储发生事件的 Slot 编号并发送 MUTE 值（00），直到 FIFO 准备好发送与检测到下溢的 Slot 对应的数据（参见图 30-19）。这样可避免存储器指针与音频帧中的 Slot 之间发生同步失效。

下溢事件会使 SAIx_SR.OVRUDR 标志置 1，如果 SAIx_IM.OVRUDRIE 位置 1，还将生成中断。要清除该标志，可将 SAIx_CLRFR.COVRUDR 位置 1。

将音频子模块配置为主模式或从模式时，会发生下溢事件。

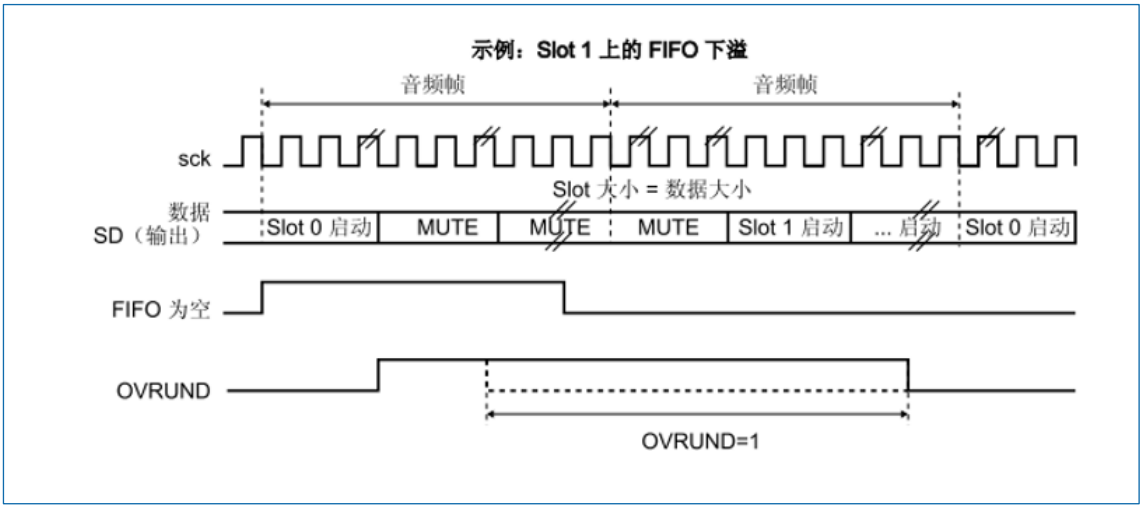


图 30-19 FIFO 下溢事件

30.2.13.2 帧同步提前检测（AFSDET）

AFSDET 标志仅在从模式下使用。主模式下不会使能该标志。由于帧长度、帧极性和帧偏移已定义且已知，该标志用于指示是否比预期更早检测到帧同步（FS）信号。

出现帧检测提前时，SAIx_SR.AFSDET 标志将置 1。

对 FS 提前不敏感的当前音频帧不受该检测影响。也就是说 FS 信号的“寄生”事件将被标记但不干扰当前音频帧。

如果 SAIx_IM.AFSDETIE 位置 1，将生成中断。要清除 AFSDET 标志，必须将 SAIx_CLRFR.CAFSDET 位置 1。

为了在出现帧检测提前错误后重新与主模块同步，需要执行以下四个步骤：

1. 通过复位 SAIx_CR1.SAIXEN 位禁止 SAI 模块。为确保禁止 SAI，读回 SAIXEN 位并检查其是否已设置为 0。
2. 通过 SAIx_CR2 寄存器的 FFLUSH 位刷新 FIFO。
3. 重新使能 SAI 外设（SAIXEN 位置 1）。
4. SAI 模块将等待 FS 使能以重新开始与主模块同步。

注意：AC’97 模式下不使能 SAIXEN 标志，原因是 SAI 音频模块作为链路控制器，即使在声明为从模块时也会生成 FS 信号。由于未使用 FS 信号，因此它在 SPDIF 模式下无意义。

30.2.13.3 帧同步滞后检测

只有当 SAI 音频模块用作从模块时，SAIx_SR.LFSDET 标志才可置 1。SAIx_FRCR 寄存器中，帧长度、帧极性和帧偏移配置均已知。

如果外部主模块未在预定时间发送 FS 信号生成过晚），LFSDET 标志将置 1，如果 SAIx_IM.LFSDETIE 位置 1，还将生成中断。

SAIx_CLRFR.CLFSDET 位置 1 时将清除 LFSDET 标志。

在检测到相应错误时帧同步滞后检测标志置 1，SAI 需要重新与主模块同步（参见章节：“30.2.13.2 帧同步提前检测（AFSDET）”）。

在噪声环境中，音频模块的状态机可能会错误检测到对 SCK 时钟的干扰，并将 SAI 数据移位到错误的帧位置。SAI 会检测到此事件，并将其报告为帧同步滞后检测错误。

如果外部主模块不是在连续模式下管理音频数据帧发送，则不会对帧造成破坏，大多数应用都不会出现这种情况。这种情况下 LFSDET 标志将置 1。

注意：AC’97 模式下不使能 LFSDET 标志，原因是 SAI 音频模块作为链路控制器，即使在声明为从模块时也会生成 FS 信号。由于协议未使用 FS 信号，因此该位在 SPDIF 模式下无意义。

30.2.13.4 编解码器未就绪（CNRDYAC’97）

仅当 SAI 音频模块配置为在 AC’97 模式下工作时（SAIx_CR1.PRTCFG[1:0]=10），SAIx_SR.CNRDY 标志才有意义。如果 SAIx_IM.CNRDYIE 位置 1，则在 CNRDY 标志置 1 时将生成中断。

在接收 AC’97 音频帧的 TAG0（slot0）期间，当编解码器未准备好进行通信时，将使能 CNRDY。这种情况下，在 TAG0 指示编解码器就绪之前，数据都不会自动存储到 FIFO，原因是编解码器未就绪。编解码器就绪后将捕获 SAIx_SLOTR 寄存器中定义的所有有效 Slot。

要清除 CNRDY 标志，必须将 SAIx_CLRFR.CCNRDY 位置 1。

30.2.13.5 主模式时钟配置错误（NODIV=0）

当音频模块在主模式下工作（MODE[1]=0）且 NODIV 位等于 0 时，如果满足以下条件，则只要使能 SAI，WCKCFG 标志便会置 1：

- （FRL+1）不是 2 的几次幂，

  并且
- （FRL+1）不在 8 和 256 之间。

MODE 位、NODIV 位和 SAIXEN 位属于 SAIx_CR1 寄存器，FRL 位属于 SAIx_FRCR 寄存器。

如果 WCKCFGIE 位置 1，则当 SAIx_SR.WCKCFG 标志置 1 时将生成中断。要清除该标志，可将 SAIx_CLRFR.CWCKCFG 位置 1。

当 WCKCFG 位置 1 时，音频模块会自动禁止，因此将对 SAIXEN 位执行硬件清零。

30.2.14 禁止 SAI

可随时通过清零 SAIx_CR1.SAIXEN 位禁止 SAI 音频模块。所有已开始的帧将在 SAI 停止工作前自动完成。SAIXEN 位将保持高电平，直到当前音频帧传输结束时 SAI 完全关闭。

如果 SAI 中有与另一个音频模块同步工作的音频模块，则必须先禁止以主模式工作的音频模块。

30.2.15 SAI DMA 接口

为减轻 CPU 负担和优化总线带宽，每个 SAI 音频模块都具有独立的 DMA 接口以便对 SAIx_DR 寄存器进行读/写操作（访问内部 FIFO）。每个音频子模块都有一个支持基本 DMA 请求/应答协议的 DMA 通道。

要为 DMA 传输配置音频子模块，可将 SAIx_CR1.DMAEN 位置 1。DMA 请求直接由 FIFO 控制器管理，具体取决于 FIFO 阈值（更多详细信息，请参见章节：“30.2.8 内部 FIFO”。DMA 传输方向与 SAI 音频子模块配置相关：

- 如果音频模块用作发送器，则音频模块的 FIFO 控制器将输出 DMA 请求以向 FIFO 加载 SAIx_DR 寄存器中写入的数据。
- 如果音频模块用作接收器，则 DMA 请求与来自 SAIx_DR 寄存器的读取操作相关。

按照下面的顺序将 SAI 接口配置为 DMA 模式：

1. 配置 SAI 和 FIFO 阈值以指定何时启动 DMA 请求。
2. 配置 SAIDMA 通道。
3. 使能 DMA。
4. 使能 SAI 接口。

注意：配置 SAI 模块前，必须禁止 SAIDMA 通道。

30.3 SAI 中断

SAI 支持 7 个中断源，如下表所示。

表 30-8 SAI 中断源

中断源	中断组	音频模块模式	中断使能	中断清零
FREQ	FREQ	主或从接收器或发送器	SAIx_IM.FREQIE	取决于： ● FIFO 阈值设置（SAIx_CR2 中的 FTH 位） ● 通信方向（发送器或接收器）更多详细信息，请参见“30.2.8 内部 FIFO”。
OVRUDR	ERROR	主或从接收器或发送器	SAIx_IM.OVRUDRIE	SAIx_CLRFR.COVRUDR = 1
AFSDET	ERROR	从（不适用于AC’97 模式和 SPDIF 模式）	SAIx_IM.AFSDETIE	SAIx_CLRFR.CAFSDET = 1
LFSDET	ERROR	从（不适用于AC’97 模式和 SPDIF 模式）	SAIx_IM.LFSDETIE	SAIx_CLRFR.CLFSDET = 1
CNRDY	ERROR	从（仅限AC’97 模式）	SAIx_IM .CNRDYIE	SAIx_CLRFR.CCNRDY = 1
MUTEDET	MUTE	主或从仅限接收模式	SAIx_IM.MUTEDETIE	SAIx_CLRFR.CMUTEDET = 1

中断源	中断组	音频模块模式	中断使能	中断清零
WCKCFG	ERROR	主模式且SAIx_CR1.NODIV = 0	SAIx_IM.WCKCFGIE	SAIx_CLRFR.CWCKCFG = 1

按照以下顺序使能中断：

1. 禁止 SAI 中断。
2. 配置 SAI。
3. 配置 SAI 中断源。
4. 使能 SAI。

30.4 SAI 寄存器

基地址：（SAIA, SAIB）=（0x4001 5800 , 0x4001 5824）

空间大小：（SAIA, SAIB）=（0x24, 0x24）

30.4.1 配置寄存器 1（SAIx_CR1）（x=A..B）

偏移地址：0x04

复位值：0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res						DEC[1:0]		MCKDIV[3:0]				NODIV	Res	DMAEN	SAIEN
						rw		rw				rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res		OUTDRIV	MONO	SYNCEN[1:0]		CKSTR	LSBFIRST	DS[2:0]			Res	PRTCFG[1:0]		MODE[1:0]	
		rw	rw	rw		rw	rw	rw				rw		rw	

位 31:26	Res：保留 必须保持复位值。
位 25:24	DEC[1:0]：滤波器抽取倍数（Decimation Filter） 用于配置 SAI 的滤波器抽取倍数 • 00：64 倍抽取 • 01：128 倍抽取 • 10：256 倍抽取 • 11：保留
位 23:20	MCKDIV[3:0]：主时钟分频器（Master clock divider） 这些位由软件置 1 和清零。当音频模块在从模式下工作时，这些位无意义。必须在音频模块禁止的情况下配置这些位。 • 0000：主时钟输入 1 分频。 • 其它：主时钟频率将根据以下公式计算： $F_{MCLK\_x}=\frac{FSAI\_x\_CK}{MCKDIV*2}$
位 19	NODIV：无分频器（No divider） 此位由软件置 1 和清零。

	• 0：使能主时钟发生器 • 1：在时钟发生器中不使用分频器（此时主/位时钟分频器不起作用）
位 18	Res：保留 必须保持复位值。
位 17	DMAEN：DMA 使能（DMA enable） 此位由软件置 1 和清零。 • 0：禁止 DMA • 1：使能 DMA 注意：在接收模式下，必须在 DMAEN 位置 1 前配置 MODE[1:0] 位，以避免 DMA 请求，原因是复位后音频模块将默认以发送模式工作。
位 16	SAIEN：音频模块使能（Audio block enable） 该位由软件置 1。 要关闭音频模块，必须由应用软件将该位编程为 0 并轮询该位，直到该位读回 0，这表示该模块已完全禁止。将该位置 1 前，先检查该位是否已置 0，否则不会执行此使能命令。 该位可用于控制 SAIx 音频模块的状态。如果在音频帧传输期间禁止该位，则正在进行的传输将继续完成，音频模块在该音频帧传输结束时被完全禁止。 • 0：禁止 SAIx 音频模块 • 1：使能 SAIx 音频模块。 *注意：当 SAIx 模块配置为主模式时，SAIx 的输入中必须有内核时钟，然后才能将 SAIXEN 位置 1。*
位 15:14	Res：保留 必须保持复位值。
位 13	OUTDRIV：输出驱动（Output drive） 此位由软件置 1 和清零。 • 0：当 SAIXEN 置 1 时驱动音频模块输出 • 1：在该位置 1 后立即驱动音频模块输出 注意：该位必须在音频模块配置后的使能前置 1。
位 12	MONO：单声道模式（Mono mode） 此位由软件置 1 和清零。仅当 Slot 数为 2 时该位才有意义。如果选择了单声道模式，则当音频模块用作发送器时，Slot 0 的数据将复制到 Slot1 上。在接收模式下，将丢弃 Slot1 并仅存储从 Slot 0 接 收的数据。更多详细信息，请参见单声道 / 立体声模式一节。 • 0：立体声模式 • 1：单声道模式
位 11:10	SYNCEN[1:0]：同步使能（Synchronization enable） 这些位将由软件置 1 和清零。必须在音频子模块禁止的情况下配置这些位。 • 00：音频子模块处于异步模式。 • 01：音频子模块与另一个内部音频子模块同步。这种情况下，必须将该音频子模块配置为从模式。 • 10：音频子模块与外部 SAI 的嵌入式外设同步。这种情况下，应将音频子模块配置为从模式。 • 11：保留 *注意：使能 SPDIF 模式后，应将音频子模块配置为异步。*

位 9	CKSTR：时钟选通边沿（Clock strobing edge） 此位由软件置 1 和清零。必须在音频模块禁止的情况下配置该位。SPDIF 音频协议下该位没有意义。 • 0：在 SCK 上升沿更改 SAI 生成的信号，而在 SCK 下降沿对 SAI 接收的信号进行采样。 • 1：在 SCK 下降沿更改 SAI 生成的信号，而在 SCK 上升沿对 SAI 接收的信号进行采样。
位 8	LSBFIRST：最低有效位优先（Least significant bit first）。 此位由软件置 1 和清零。必须在音频模块禁止的情况下配置该位。AC’97 音频协议下该位没有意义，原因是传输 AC’97 数据时，数据的 MSB 位优先。SPDIF 音频协议下该位没有意义，原因是传输 SPDIF 数据时，数据的 LSB 位优先。 • 0：优先传输数据的 MSB • 1：优先传输数据的 LSB
位 7:5	DS[2:0]：数据大小（Data size）。 这些位将由软件置 1 和清零。选择 SPDIF 协议时（位 PRTCFG[1:0]），将忽略这些位，原因是在这种情况下帧和数据大小是固定的。通过 COMP[1:0] 位选择压扩模式时，将忽略 DS[1:0]，原因是算法已将数据大小固定为 8 位。 必须在音频模块禁止的情况下配置这些位。 • 000：保留 • 001：保留 • 010：8 位 • 011：10 位 • 100：16 位 • 101：20 位 • 110：24 位 • 111：32 位
位 4	Res：保留 必须保持复位值。
位 3:2	PRTCFG[1:0]：协议配置（Protocol configuration） 这些位将由软件置 1 和清零。必须在音频模块禁止的情况下配置这些位。 • 00：自由协议。通过设置大部分配置寄存器位以及帧配置寄存器，自由协议允许用户使用音频模块 这一强大的配置功能来处理特定的音频协议（如 I2S、LSB/MSB 对齐、TDM、PCM/DSP 等）。 • 01：SPDIF 协议 • 10：AC’97 协议 • 11：PDM 协议
位 1:0	MODE[1:0]：SAIx 音频模块模式（SAIx audio block mode） 这些位将由软件置 1 和清零。必须在 SAIx 音频模块禁止的情况下配置这些位。 • 00：主发送器 • 01：主接收器 • 10：从发送器 • 11：从接收器 注意：如果将音频模块配置为 SPDIF 模式，则将强制设置主发送模式（MODE[1:0] = 00）。在主发送模式下，音频模块将立即开始生成 FS 和时钟。

30.4.2 配置寄存器 2（SAIx_CR2）（x=A..B）

偏移地址：0x08

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CODE_CHG4[1:0]		CODE_CHG3[1:0]		CODE_CHG2[1:0]		CODE_CHG1[1:0]		Res							
rw		rw		rw		rw									

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP[1:0]		CPL	MUTECNT[5:0]						MUTEVAL	MUTE	TRIS	FFLUSH	FTH[2:0]		
rw		rw	rw						rw	rw	rw	rw	rw		

位 25+2(y-1):24+2(y-1) (y=4..1)	CODE_CHGy[1:0]：源码补码转换逻辑单元（Source Complement conversion control） • 00：不转换 • 10：不转换 • 01：将数据按不带符号位转换（不带符号位的源码补码之间互转一次） • 11：将数据按带符号位转换（带符号位的源码补码之间互转一次）
位 23:16	Res：保留 必须保持复位值。
位 15:14	COMP[1:0]：压扩模式（Companding mode） 这些位将由软件置 1 和清零。μ-Law 和 A-Law 算法是 CCITT G.711 建议的一部分，要使用何种补码 类型取决于 CPL 位。 数据扩展还是数据压缩由 MODE[0] 位的状态确定。 如果将音频模块配置为发送器，则应用数据压缩。 如果将音频模块配置为接收器，则自动应用数据扩展。 更多详细信息，请参见压扩模式一节。 • 00：不支持压扩算法 • 01：保留 • 10：μ-Law 算法 • 11：A-Law 算法 注意：选择任意一个音频协议时都能使用压扩模式。
位 13	CPL：补码位（Complement bit） 此位由软件置 1 和清零。 该位定义用于压扩模式的补码类型。 • 0：1 的补码表示 • 1：2 的补码表示 *注意：仅当压扩模式为μ-Law 算法或 A-Law 算法时该位才有效。*
位 12:7	MUTECNT[5:0]：静音计数器（Mute counter） 这些位将由软件置 1 和清零。这些位仅用于接收模式。 这些位中所设置的值将与接收模式下检测到的连续静音帧数量进行比较。当静音帧数量与该值相等时，MUTEDET 标志置 1，并且在 MUTEDETIE 位置 1 的情况下，还将生成中断。 更多详细信息，请参见静音模式一节。
位 6	MUTEVAL：静音值（Mute value） 该位由软件置 1 和清零。必须在使能音频模块（SAIxEN） 前写入。仅当音频模块用作发送

	器、Slot 数小于或等于 2 并且 MUTE 位置 1 时，该位才有意义。 如果声明了 2 个以上的 Slot，则无论 MUTEVAL 位的值为何，静音模式下发送的位值都将等于 0。 如果 Slot 数小于或等于 2 且 MUTEVAL = 1，则为每个 Slot 发送的 MUTE 值将是上一帧期间发送的值。 更多详细信息，请参见静音模式一节。 • 0：静音模式期间发送位置 0 • 1：静音模式期间发送上一个值 注意：该位对 SPDIF 音频模块无意义，从而也不使用。
位 5	MUTE：静音（Mute） 此位由软件置 1 和清零。仅当音频模块用作发送器时该位才有意义。Slot 数小于或等于 2 时，MUTE 值与 MUTEVAL 值相关； Slot 数大于 2 时，MUTE 值等于 0。 更多详细信息，请参见静音模式一节。 • 0：禁止静音模式 • 1：使能静音模式 注意：该位对 SPDIF 音频模块无意义，从而也不使用。
位 4	TRIS：数据线的三态管理（Tristate management on data line） 此位由软件置 1 和清零。仅当将音频模块配置为发送器时该位才有意义。当音频模块配置为 SPDIF 模式时不使用该位。应在 SAI 禁止时配置此位。 更多详细信息，请参见无效 Slot 上的输出数据线管理一节。 • 0：Slot 无效时，SD 输出线仍由 SAI 驱动。 • 1：SD 输出线将在上一个有效 Slot（下一个 Slot 无效）的最后一个数据位传输结束时释放（高阻态）。
位 3	FFLUSH：FIFO 刷新（FIFO flush） 该位由软件置 1，始终读为 0。应在 SAI 禁止时配置此位。 • 0：禁止 FIFO 刷新。 • 1：FIFO 刷新。将此位编程为 1 可触发 FIFO 刷新。所有的内部 FIFO 指针（读和写）将清零。这种情况下，仍存留在 FIFO 中的数据丢失（发送或接收数据不会继续丢失）。刷新 SAI 前，必须禁止 DMA 数据流/中断。
位 2:0	FTH[2:0]：FIFO 阈值（FIFO threshold）。 此位由软件置 1 和清零。 • 000：FIFO 为空 • 001：¼ FIFO 满 • 010：½ FIFO 满 • 011：¾ FIFO 满 • 100：FIFO 已满 • 101：保留 • 110：保留 • 111：保留

30.4.3 帧配置寄存器（SAIx_FRCR）（x=A..B）

偏移地址：0x0C

复位值：0x0000 0007

该寄存器对于 AC’97 和 SPDIF 音频协议无意义。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res													FSOFF	FSPOL	FSDEF
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	FSALL[6:0]							FRL[7:0]							
	rw							rw							

位 31:19	Res：保留 必须保持复位值。
位 18	FSOFF：帧同步偏移（Frame synchronization offset） 此位由软件置 1 和清零。该位对 AC’97 或 SPDIF 音频模块配置无意义，从而也不使用。必须在音频模块禁止的情况下配置该位。 • 0：在 Slot 0 的第一位上使能 FS。 • 1：在 Slot 0 第一位的前一位上使能 FS。
位 17	FSPOL：帧同步极性（Frame synchronization polarity） 此位由软件置 1 和清零。该位用于配置 FS 信号上的 SOF 电平。该位对 AC’97 或 SPDIF 音频模块 配置无意义，从而也不使用。 必须在音频模块禁止的情况下配置该位。 • 0：FS 为低电平有效（下降沿） • 1：FS 为高电平有效（上升沿）
位 16	FSDEF：帧同步定义（Frame synchronization definition） 此位由软件置 1 和清零。 • 0：FS 信号为 SOF 信号 • 1：FS 信号为 SOF 信号 + 通道识别信号 此位置 1 时，SAIx_SLOTR 寄存器中定义的 Slot 数必须为偶数。这意味着有半数 Slot 将用于左通道，其它 Slot 用于右通道（例如，对于 I2S 或 MSB/LSB 对齐等协议，该位必须置 1）。 此位对 AC’97 或 SPDIF 音频模块配置无意义，从而也不使用。必须在音频模块禁止的情况下配置该位。
位 15	Res：保留 必须保持复位值。
位 14:8	FSALL[6:0]：帧同步有效电平长度（Frame synchronization active level length） 这些位将由软件置 1 和清零。这些位用于指定音频帧中 FS 信号的有效电平长度，以位时钟数（SCK）+1（FSALL[6:0]+1）表示这些位对 AC’97 或 SPDIF 音频模块配置无意义，从而也不使用。 必须在音频模块禁止的情况下配置这些位。
位 7:0	FRL[7:0]：帧长度（Frame length） 这些位将由软件置 1 和清零。这些位用于定义以 SCK 时钟周期数表示的音频帧长度：帧中的位数等于 FRL[7:0]+1。 音频帧中发送的位数必须大于或等于 8，否则音频模块将出现操作异常。数据大小为 8 位且在 SAIx_SLOTR 寄存器的 NBSLOT[4:0]中只定义了一个 Slot（NBSLOT[3:0]=0000）时便属于这种情况。 在主模式下，如果使用主时钟（MCLK_x 引脚上提供），则帧长度应为 8 到 256 之间的一个等于 2 的几次幂的数。不使用主时钟（NODIV=1）时，建议将帧长度编程为 8 到 256 之间的值。这些位对 AC’97 或 SPDIF 音频模块配置无意义，从而也不使用。

30.4.4 Slot 寄存器（SAIx_SLOTR）（x=A..B）

偏移地址：0x10

复位值：0x0000 0000

该寄存器对于AC'97 和SPDIF 音频协议无意义。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SLOTEN[15:0]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res				NBSLOT[3:0]				SLOTSZ[1:0]		Res	FBOFF[4:0]				
				rw				rw			rw				

位 31:16	SLOTEN[15:0]：Slot 使能（Slot enable） 这些位将由软件置 1 和清零。 每个 SLOTEN 位都对应于从 0 到 15 的一个 Slot 位置（最多 16 个 Slot）。 • 0：无效 Slot • 1：有效 Slot 必须在音频模块禁止的情况下使能 Slot。 在 AC’97 或 SPDIF 模式下会忽略这些位。
位 15:12	Res：保留 必须保持复位值。
位 11:8	NBSLOT[3:0]：音频帧中的 Slot 数（Number of slots in an audio frame） 这些位将由软件置 1 和清零。 此位字段中设置的值表示音频帧中的 Slot 数 - 1（包括无效 Slot 数）。Slot 数最大值为 16。 SAIx_FRCR.FSDEF 位置 1 时，Slot 数应为偶数。 必须在音频模块禁止的情况下配置 Slot 数。 在 AC’97 或 SPDIF 模式下会忽略这些位。
位 7:6	SLOTSZ[1:0]：Slot 大小（Slot size）。 此位由软件置 1 和清零。 Slot 大小必须大于或等于数据大小。如果不满足该条件，SAI 的行为将不确定。 有关如何驱动 SD 线的信息，请参见无效 Slot 上的输出数据线管理一节。 必须在音频模块禁止的情况下配置这些位。 在 AC’97 或 SPDIF 模式下会忽略这些位。 • 00：Slot 大小与数据大小（在 SAIx_CR1 寄存器的 DS[3:0] 位中指定）相当。 • 01：16 位 • 10：32 位 • 11：保留
位 5	Res：保留 必须保持复位值。
位 4:0	FBOFF[4:0]：第一个位偏移（First bit offset）。 这些位将由软件置 1 和清零。 此位字段中设置的值定义 Slot 中第一个数据传输位的位置。它表示一个偏移值。在发送模式下，此数据字段以外的位将强制清零。在接收模式下，将丢弃额外接收的位。 必须在音频模块禁止的情况下配置这些位。 在 AC’97 或 SPDIF 模式下会忽略这些位。

30.4.5 中断屏蔽寄存器（SAIx_IM）（x=A..B）

偏移地址：0x14

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res									LFSDETIE	AFSDETIE	CNRDYIE	FREQIE	WCKCFGIE	MUTEDETIE	OVRUDRIE
									rw	rw	rw	rw	rw	rw	rw

位 31:7	Res：保留 必须保持复位值。
位 6	LFSDETIE：帧同步滞后检测中断使能（Late frame synchronization detection interrupt enable）。 此位由软件置 1 和清零。 • 0：禁止中断 • 1：使能中断 该位置 1 时，若 SAIx_SR.LFSDET 位置 1，则生成中断。 该位对于 AC’97、SPDIF 模式无意义；若音频模块为主模块，该位也无意义。
位 5	AFSDETIE：帧同步提前检测中断使能（Anticipated frame synchronization detection interrupt enable）。 此位由软件置 1 和清零。 • 0：禁止中断 • 1：使能中断 该位置 1 时，若 SAIx_SR.AFSDET 位置 1，则生成中断。 该位对于 AC’97、SPDIF 模式无意义；若音频模块为主模块，该位也无意义。
位 4	CNRDYIE：编解码器未就绪中断使能（Codec not ready interrupt enable）。 此位由软件置 1 和清零。 • 0：禁止中断 • 1：使能中断 若使能该中断，音频模块将在 AC’97 帧的 Slot 0（tag0） 中检测连接到该线路的编解码器是否就绪。 如果未就绪，SAIx_SR.CNRDY 标志置 1，并生成中断。 仅当通过 PRTCFG[1:0] 位选择了 AC’97 模式且音频模块用作接收器时，该位才有意义。
位 3	FREQIE：FIFO 请求中断使能（FIFO request interrupt enable） 此位由软件置 1 和清零。 • 0：禁止中断 • 1：使能中断 该位置 1 时，若 SAIx_SR.FREQ 位置 1，则生成中断。 在接收模式下，必须在 FREQIE 位置 1 前配置 MODE 位，以避免寄生中断，原因是复位后音频模块将默认用作发送器。
位 2	WCKCFGIE：时钟配置错误中断使能（Wrong clock configuration interrupt enable） 此位由软件置 1 和清零。 • 0：禁止中断

	• 1：使能中断 仅在将音频模块配置为主模式（MODE[1] = 0） 且 NODIV = 0 时才执行此位。 该位在 SAIx_SR.WCKCFG 标志置 1 时生成中断。 注意： 该位仅用于 TDM 模式，其它模式下没有意义。
位 1	MUTEDETIE：静音检测中断使能（Mute detection interrupt enable） 此位由软件置 1 和清零。 • 0：禁止中断 • 1：使能中断 该位置 1 时，若 SAI_ASR.MUTEDET 位置 1，则生成中断。 仅当音频模块配置为以发送模式工作时该位才有意义。
位 0	OVRUDRIE：上溢/下溢中断使能（Overrun/underrun interrupt enable） 此位由软件置 1 和清零。 • 0：禁止中断 • 1：使能中断 该位置 1 时，若 SAI_ASR.OVRUDR 位置 1，则生成中断。

30.4.6 状态寄存器（SAIx_SR）（x=A..B）

偏移地址：0x018

复位值：0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res													FLVL[2:0]		
													r		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res									LFSDET	AFSDET	CNRDY	FREQ	WCKCFG	MUTEDET	OVRUDR
									r	r	r	r	r	r	r

位 31:19	Res：保留 必须保持复位值。
位 18:16	FLVL[2:0]：FIFO 阈值标志（FIFO level threshold） 该位为只读。FIFO 阈值标志只通过硬件管理，其设置取决于 SAI 模块的配置（发送器或接收器 模式）。 • 如果将 SAI 模块配置为发送器： ○ 000：FIFO 为空 ○ 001：FIFO <= ¼满，非空 ○ 010：¼ < FIFO <= ½满 ○ 011：½ < FIFO <= ¾满 ○ 100：¾ < FIFO 满，但未满 ○ 101：FIFO 已满 • 如果将 SAI 模块配置为接收器： ○ 000：FIFO 为空 ○ 001：FIFO < ¼满，但非空 ○ 010：¼ <= FIFO < ½满 ○ 011：½ =< FIFO < ¾满

	○ 100：¾ =< FIFO 满，但未满 ○ 101：FIFO 已满
位 15:7	Res：保留 必须保持复位值。
位 6	LFSDET：帧同步滞后检测（Late frame synchronization detection） 该位为只读。 • 0：无错误。 • 1：帧同步信号未在正确的时刻出现。 仅当音频模块配置为以从模式工作时，此标志才能置 1。 不适用于 AC’97 或 SPDIF 模式。 它可在 SAIx_IM.LFSDETIE 位置 1 时生成中断。 在软件将 SAIx_CLRFR.CLFSDET 位置 1 时清除该标志。
位 5	AFSDET：帧同步提前检测（Anticipated frame synchronization detection） 该位为只读。 • 0：无错误。 • 1：提前检测到帧同步信号。仅当音频模块配置为以从模式工作时，此标志才能置 1。 不适用于 AC’97 或 SPDIF 模式。 它可在 SAIx_IM.AFSDETIE 位置 1 时生成中断。 在软件将 SAIx_CLRFR.CAFSDET 位置 1 时清除该标志。
位 4	CNRDY：编解码器未就绪（Codec not ready） 该位为只读。 • 0：外部 AC’97 编解码器已就绪 • 1：外部 AC’97 编解码器未就绪 仅当在 SAIx_CR1 寄存器中选择了 AC’97 音频模块并且音频模块配置为接收器模式时，才使用 该位。 它可在 SAIx_IM.CNRDYIE 位置 1 时生成中断。 在软件将 SAIx_CLRFR.CCNRDY 位置 1 时清除该标志。
位 3	FREQ：FIFO 请求（FIFO request） 该位为只读。 • 0：无 FIFO 请求。 • 1：FIFO 请求读取或写入 SAIx_DR。请求内容取决于音频模块的配置： ○ 如果模块配置为发送模式，则 FIFO 请求与向 SAIx_DR 中写入相关。 ○ 如果音频模块配置为接收模式，则 FIFO 请求与从 SAIx_DR 中读取相关。 此标志可在 SAIx_IM.FREQIE 位置 1 时生成中断。
位 2	WCKCFG：时钟配置错误标志（Wrong clock configuration flag） 该位为只读。 • 0：时钟配置正确 • 1：时钟配置不符合章节“30.2.5 帧同步”中定义的帧长度规范（SAIx_FRCR 寄存器中 FRL[7:0] 位的配置）。 该位仅在音频模块工作在主模式（MODE[1] = 0） 下且 NODIV = 0 时使用。 它可在 SAIx_IM.WCKCFGIE 位置 1 时生成中断。 在软件将 SAIx_CLRFR.CWCKCFG 位置 1 时清除该标志。

位 1	MUTEDET：静音检测（Mute detection） 该位为只读。 • 0：SD 输入线上未检测到 MUTE 值 • 1：在 SD 输入线上检测到指定数量的连续音频帧中的 MUTE 值（0 值） 如果在指定音频帧的每个 Slot 或在一定数量（在 SAIx_CR2.MUTECNT 位中设置）的连续音频帧中接收到连续的 0 值，则该标志置 1。 它可在 SAIx_IM.MUTEDETIE 位置 1 时生成中断。 在软件将 SAIx_CLRFR.CMUTEDET 位置 1 时清除该标志。
位 0	OVRUDR：上溢/下溢错误标志位（Overrun/underrun） 该位为只读。 • 0：无上溢/下溢错误。 • 1：检测到上溢/下溢错误。仅当音频模块分别被配置为接收器和发送器时才会发生上溢和下溢情况。它可在 SAIx_IM.OVRUDRIE 位置 1 时生成中断。 在软件将 SAIx_CLRFR.COVRUDR 位置 1 时清除该标志。

30.4.7 清除标志寄存器（SAIx_CLRFR）（x=A..B）

偏移地址：0x01C

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res									CLFSDET	CAFSDET	CCNRDY	Res	CWCKCFG	CMUTEDET	COVRUDR
									w	w	w		w	w	w

位 31:7	Res：保留 必须保持复位值。
位 6	CLFSDET：清除帧同步滞后检测标志（Clear late frame synchronization detection flag） 该位为只写位。 将此位编程为 1 可清除 SAIx_SR.LFSDET 标志。 该位不适用于 AC’97 或 SPDIF 模式。 读取该位将始终返回值 0。
位 5	CAFSDET：清除帧同步提前检测标志（Clear anticipated frame synchronization detection flag） 该位为只写位。 将此位编程为 1 可清除 SAIx_SR.AFSDET 标志。不适用于 AC’97 或 SPDIF 模式。 读取该位将始终返回值 0。
位 4	CCNRDY：清除编解码器未就绪标志（Clear codec not ready flag） 该位为只写位。 将此位编程为 1 可清除 SAIx_SR.CNRDY 标志。 仅当在 SAIx_CR1 寄存器中选择了 AC’97 音频协议时，才使用该位。 读取该位将始终返回值 0。
位 3	Res：保留

	必须保持复位值。
位 2	CWCKCFG：清除时钟配置错误标志（Clear wrong clock configuration flag） 该位为只写位。 将此位编程为 1 可清除 SAIx_SR.WCKCFG 标志。 仅当音频模块设置为主模块时（MODE[1] = 0）且 SAIx_CR1.NODIV = 0 时，才使用 此位。 读取该位将始终返回值 0。
位 1	CMUTEDET：静音检测标志（Mute detection flag） 该位为只写位。 将此位编程为 1 可清除 SAIx_SR.MUTEDET 标志。读取该位将始终返回值 0。
位 0	COVRUDR：清除上溢/下溢标志（Clear overrun/underrun） 该位为只写位。 将此位编程为 1 可清除 SAIx_SR.OVRUDR 标志。 读取该位将始终返回值 0。

30.4.8 数据寄存器（SAIx_DR）（x=A..B）

偏移地址：0x020

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw															

位 31:0	DATA[31:0]：数据（Data） 若 FIFO 未满，写入该寄存器可向 FIFO 加载数据。 若 FIFO 非空，读取该寄存器可清空 FIFO。

31 高级加密标准硬件加速器（AES）

AES 硬件加速器通过 AES 算法对数据进行加密和解密。加密处理器完全兼容如下标准：联邦信息处理标准出版物（FIPS PUB 197，2001 年 11 月 26 日）规定的高级加密标准（AES）。

加速器使用长度为 128 位、192 位或 256 位的密钥对 128 位数据块进行加密和解密，还可执行密钥生成。为了最大限度地减少使用相同密钥处理多个数据块 CPU 或 DMA 执行的写操作，会将加密或解密密钥存储在内部寄存器中。

默认使用电子密码本（ECB）模式。此外，硬件还支持密码块链接（CBC）和计数器（CTR）模式链接算法。

AES 支持对传入和传出数据进行 DMA 传输（需要 2 个 DMA 通道）。

支持使用 TRNG 输出的随机数来随机化 AES 运算模块，以加强 AES 的安全性。

31.1 AES 主要特性

- 使用 AES Rijndael 模块密码算法进行加密/解密。
- AES 加密解密算法符合 NIST FIPS 197 标准。
- 内部 256 位寄存器用于存储加密或生成密钥（8 个 32 位寄存器）。
- 支持电子密码本（ECB）模式、密码块链接（CBC）模式和计数器（CTR）模式
- 解密密钥生成
- 128 位数据块处理
- 128 位、192 位或者 256 位密钥长度可配
- 加解密计算时间：
  - 128 位密钥：57 个时钟周期
  - 192 位密钥：67 个时钟周期
  - 256 位密钥：77 个时钟周期
- 解密密钥准备时间：
  - 128 位密钥：57 个时钟周期
  - 192 位密钥：67 个时钟周期
  - 256 位密钥：77 个时钟周期
- 1 个 32 位输入缓冲器和 1 个 32 位输出缓冲器
  - 缓冲器寄存器的访问仅支持 32 位数据宽度。
- 采用自动数据流控制，支持直接存储器访问 DMA（使用 2 个通道，分别用于传入数据和传出数据）。
- 一个 128 位寄存器用于初始化向量（AES 配置为 CBC 模式时）或 32 位计数器初始化（AES 配置为 CTR 模式时）
- 支持 AES 时钟随机化，和 TRNG 模块配合，随机化 AES 时钟
- 更方便的中断现场保护和中断现场恢复

31.2 AES 功能说明

下图显示了 AES 加速器的框图。

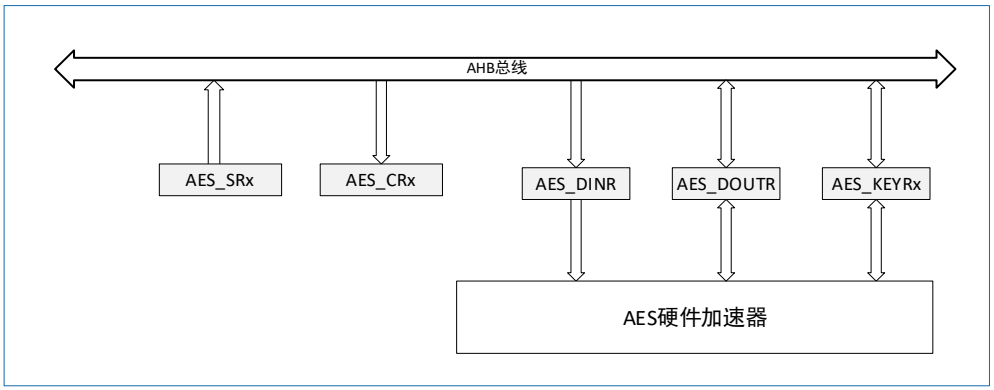


图 31-1 AES 系统框图

AES 加速器密钥使用长度为 128 位、192 位或 256 位可配；被加/解密的数据块基本单元长度为 128 位（可以有 N 个这样的数据块单元）。如果选择 CBC 或 CTR 链接模式，还会处理初始化向量。

它提供 4 种工作模式：

- 模式 1：使用 AES_KEYRx 寄存器中存储的密钥进行加密。
- 模式 2：ECB 或 CBC 模式的解密密钥准备。在进行模式 3 的 ECB 或 CBC 解密运算之前，需要先使用模式 2 把解密密钥准备好。在模式 2 执行完成后解密密钥自动存储在 AES_KEYRx 中。
- 模式 3：使用存储在 AES_KEYRx 中的密钥进行解密。
- 模式 4：使用存储在 AES_KEYRx 中的密钥进行单次 ECB 或 CBC 解密运算（自动完成解密密钥准备）。

注意：模式 2 和模式 4 只用于 ECB 或 CBC 解密。

通过 AES_CR.MODE[1:0]位选择工作模式。仅当 AES 已禁止（AES_CR.EN=0）时，才允许切换模式。使能 AES 之前，必须存储 KEY 寄存器（AES_KEYRx）。

要选择将 ECB、CBC 或 CTR 中的一种模式用于加密解决方案，必须在 AES 已禁止（AES_CR.EN=0）时写入 AES_CR 寄存器的 CHMOD[1:0]位和 AES_IVRx 寄存器（AES_IVR 寄存器仅用于 CBC 和 CTR 链接模式）。

使能（位 EN=1）后，AES 处于输入阶段，等待软件针对模式 1、模式 3 或模式 4 将输入数据（4 个字）写入 AES_DINR 寄存器。根据所选模式，数据对应于明文消息或者是密文消息。在对 AES_DINR 寄存器的两次连续写入之间自动插入一个等待周期，以便将密钥与数据交替发送到 AES 处理器。

对于模式 2，AES_CR.EN 位置 1 后立即开始密钥生成过程。在使能 AES 之前，需要将加密的密钥加载到 AES_KEYRx 寄存器中。密钥生成过程结束时（CFF 标志置 1），会将生成密钥存储在 AES_KEYRx 寄存器中，并由硬件禁止 AES。在该模式下，不得在 AES 已使能的情况下读取 AES_KEYRx 寄存器，必须等到 CCF（计算完成标志）被硬件置 1 后才能读取 AES_KEYRx 寄存器。

计算阶段完成后，会立即将 AES_SR 寄存器中的状态标志 CCF 置 1。如果 AES_CR.CCFIE=1，可产生中断。随后，软件可从 AES_DOUTR 寄存器（对于模式 1、模式 3、模式 4）或 AES_KEYRx 寄存器（若选择模式 2）中读回数据。

在输入和输出阶段，软件必须连续读取或写入数据字节（处于模式 2 时除外），但 AES 允许每个读取或写入操作之间存在延迟（示例：如果在此时为另一中断提供服务）。

如果检测到非预期的读取或写入操作，则会将 AES_SR 寄存器中的 RDERR 和 WRERR 标志置 1。如果 AES_CR.ERRIE 位置 1，可产生中断。检测到错误后，AES 不会禁止，而是会继续照常处理。

可随时通过将 AES_xCR.EN 位清零来重新初始化 AES。随后，可通过将 EN 置 1 重新启动 AES，此时，AES 会等待第一个输入数据字节写入（处于模式 2 下时除外，此时在 EN 位置 1 后，会立即从 AES_KEYRx 寄存器中存储的值开始密钥生成过程）。

31.3 加密和生成密钥

AES_KEYRx 寄存器用于存储加密或解密密钥。这 8 个寄存器使用小端模式进行组织：必须将密钥的 32 位 LSB（最低有效位）载入寄存器 AES_KEYR0 中，以此类推，必须将 256 位密钥的 32 位 MSB（最高有效位）载入 AES_KEYR7。

AES 禁止时（AES_CR.EN=0），必须将加密或解密密钥存储在这些寄存器中。密钥的字节顺序是固定的。

在模式 2（密钥生成）中，需要将加密密钥载入 AES_KEYRx 中。然后必须使能 AES。计算阶段结束时，生成密钥会自动存储在 AES_KEYRx 寄存器中，并覆盖之前的加密密钥。生成密钥可用时，将由硬件禁止 AES。

在模式 4（密钥生成+解密）中，AES_KEYRx 寄存器只包含加密密钥。生成密钥会在内部计算，不会对这些寄存器执行任何写操作。

31.4 AES 链接算法

AES 硬件支持三种算法，可在 AES 禁止（位 EN=0）的情况下通过 AES_CR.CHMOD[1:0]位进行选择：

- 电子密码本（ECB）
- 密码块连接（CBC）
- 计数器模式（CTR）

31.4.1 电子密码本（ECB）

此模式为默认模式。此模式不使用 AES_IVR 寄存器，也不会进行链接操作。消息会划分到各个块中，并对各个块分别进行独立加解密。

图 31-2 和图 31-3 分别介绍了用于加密和解密的电子密码本算法的原理。

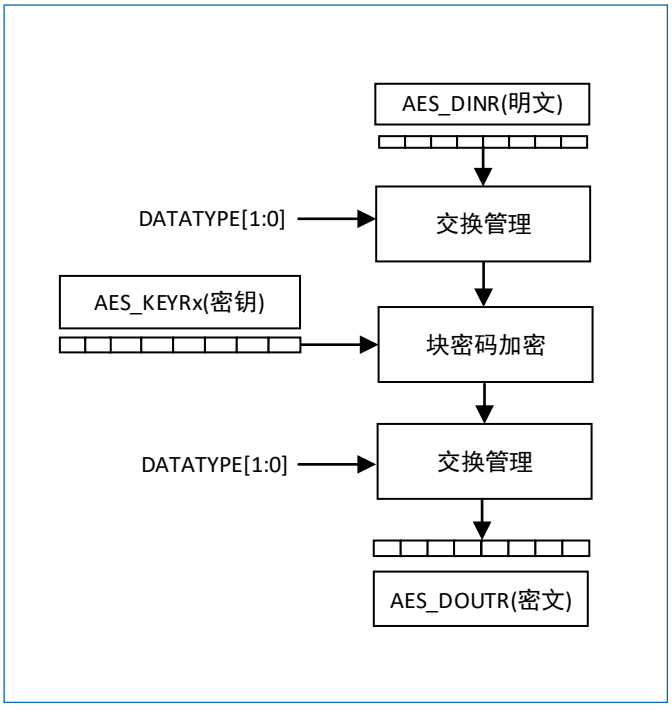


图 31-2 ECB 加密模式

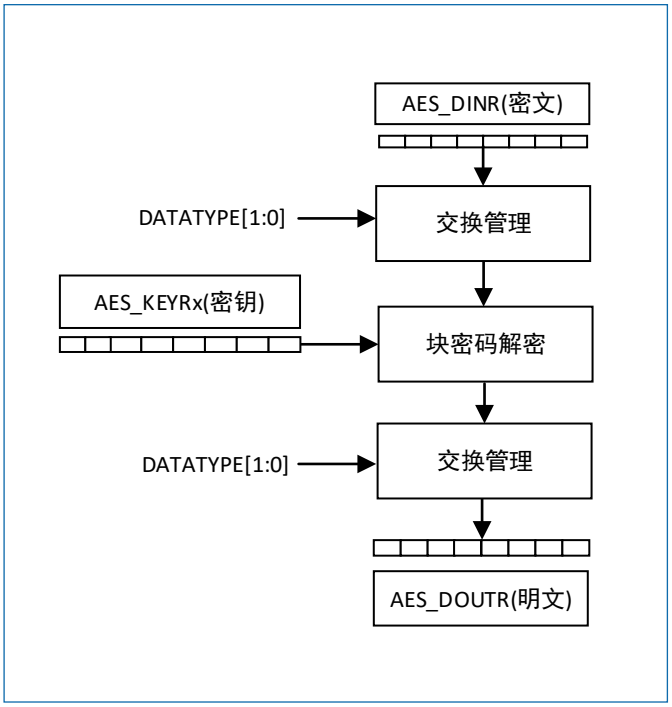


图 31-3 ECB 解密模式

31.4.2 密码块链接（CBC）

在密码块链接（CBC）模式下，会先将每个明文块与之前的密码文本块进行异或运算，然后再进行加密。为了确保每条消息都是唯一的，会在第一个块处理过程中使用初始化向量（AES_IVRx）。

图 31-4 和图 31-5 分别介绍了密码块链接加解密算法的原理。

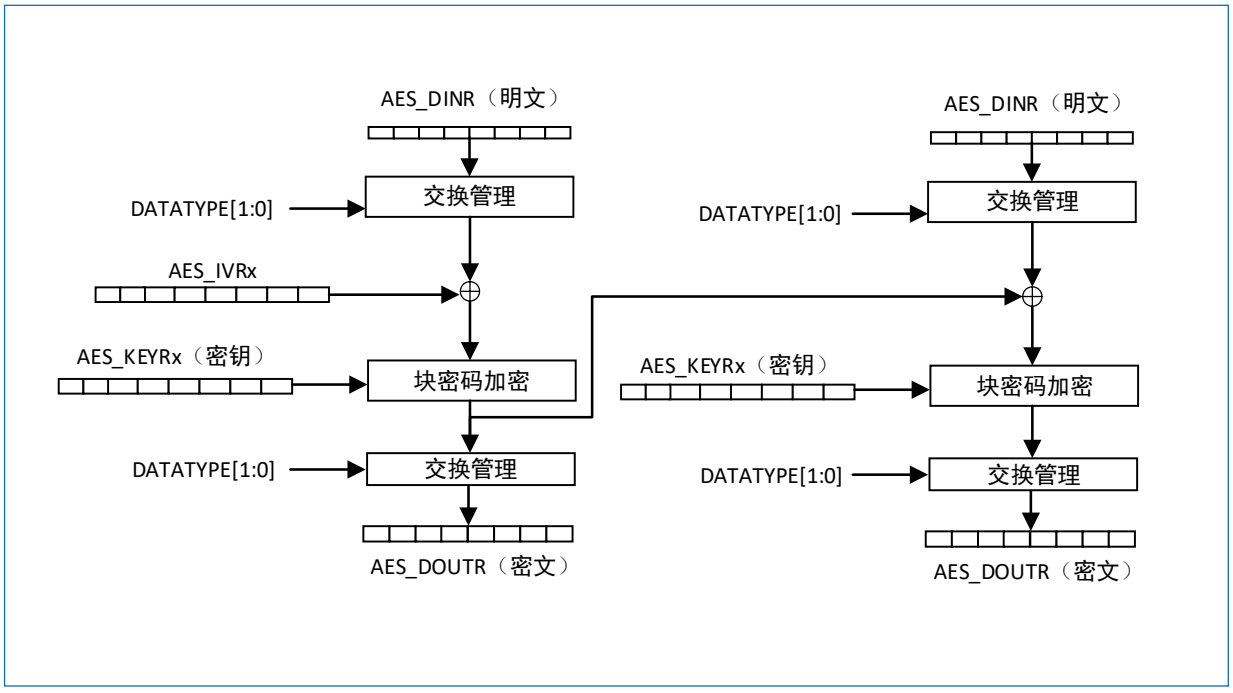


图 31-4 CBC 模式加密

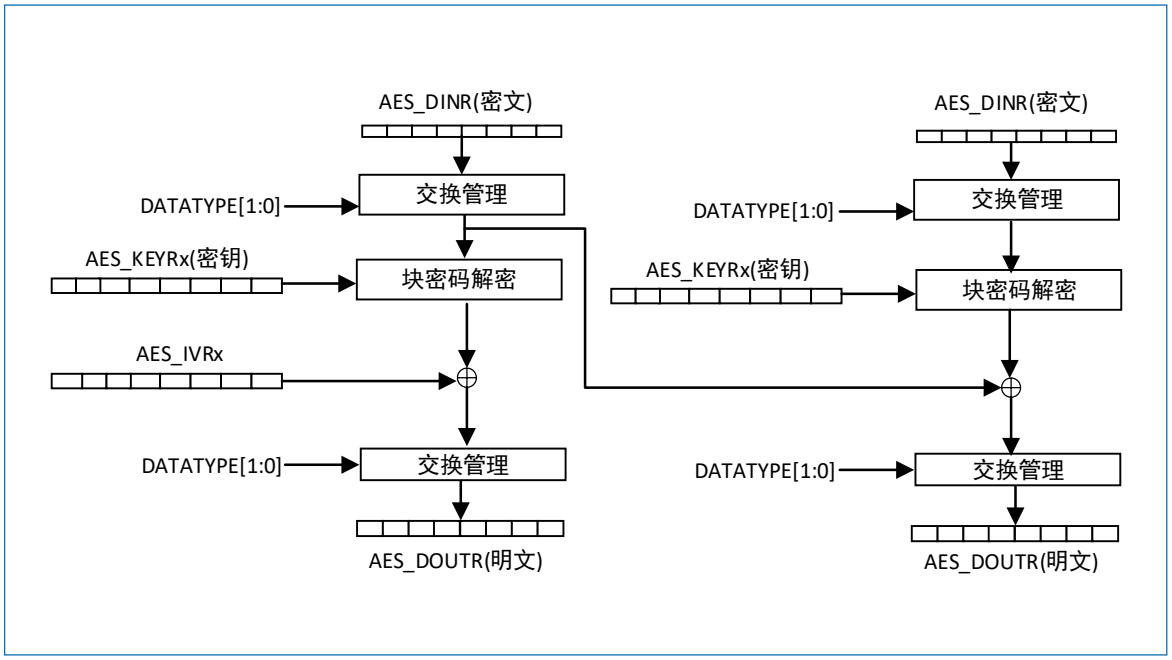


图 31-5 CBC 模式解密

需要注意的是，在密码块链接（CBC）算法下，解密只能选择：工作模式 2+工作模式 3 的组合。如果选择工作模式 4，则只有第一次单次解密是正确的，后面都会是错误的。

CBC 解密配置步骤如下：

1. 选择 CBC 算法；
2. 选择模式 2：密钥生成；
3. 使能 AES；
4. 等到 CCF 标记置位之后，清除该标记；
5. 关闭 AES，重新配置工作模式 3：解密；
6. 使能 AES；
7. 输入解密数据，开始运算。

31.4.3 计数器模式（CTR）

CTR 模式使用 AES 算法生成密钥流，还会使用 32 位计数器与密码文本或明文进行异或运算。

图 31-6 和图 31-7 分别介绍了计数器模式加解密算法的原理。

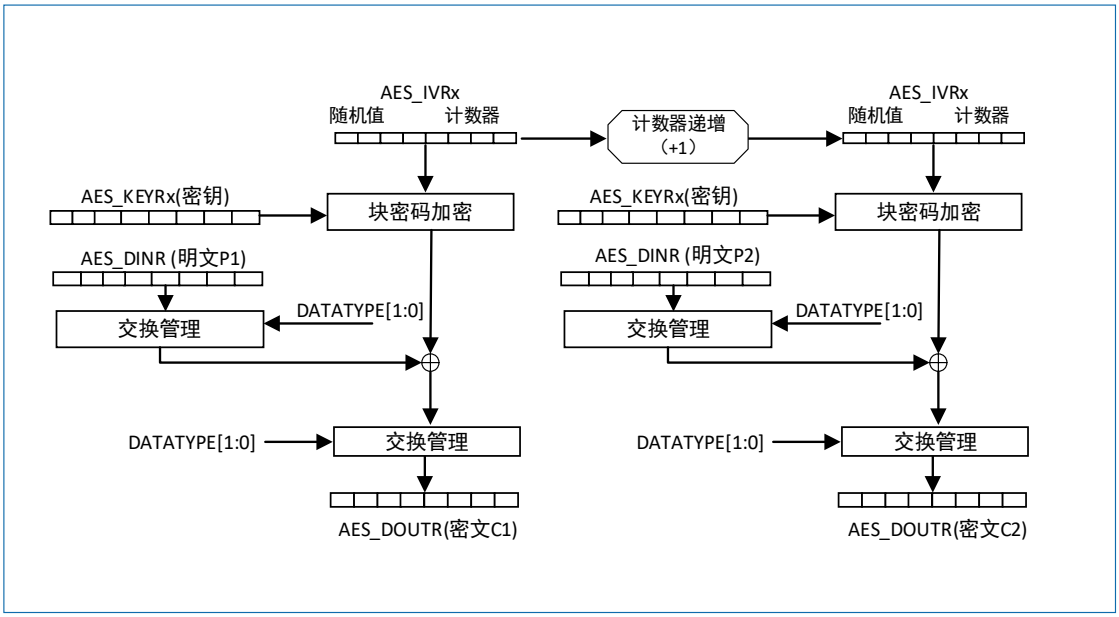


图 31-6 CTR 模式加密

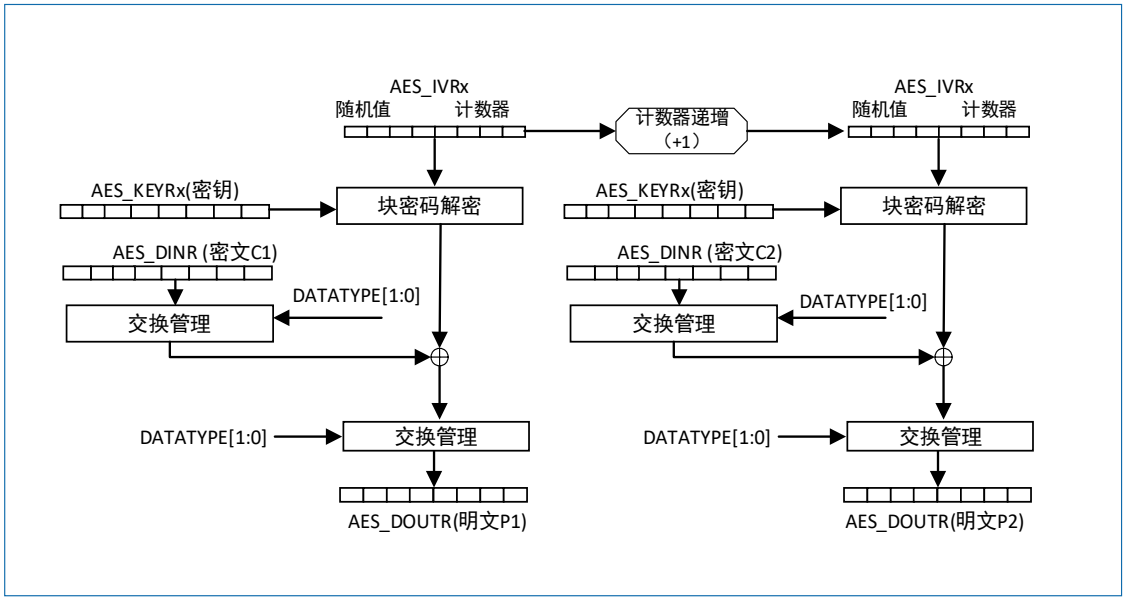


图 31-7 CTR 模式解密

可通过 AES_IVRx 寄存器获取随机值和 32 位计数器，其构成表 31-1 所示：

表 31-1 32 位计数器 + 随机值构成

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
Nonce[31:0]	Nonce[63:32]	Nonce[96:64]	32-bit counter=0x0001

如上表所示 AES_IVR0 是一个 32 位的计数器，并且初始值为 1。

在计数器模式下，对于要处理的每个模块，计数器都会在初始值的基础上递增，以确保该序列是长时间内不会重复的唯一序列。该计数器属于 32 位计数器，这意味会对 AES 禁止时保存的初始化值保留随机消息。只有 128 位初始化向量的 32 位 LSB 才能代表计数器。与 CBC 模式（仅在处理第一个数据库块时使用一次 AES_IVRx 寄存器）相反，在计数器模式下，处理每个数据块时都会使用 AES_IVRx 寄存器。

在计数器模式下，不区分加密和解密，因为 AES 始终加密计数器的值来产生密钥流，然后和数据异或（加密时和明文异或，解密是和密文异或）。

注意：仅当AES 已禁止（位EN=0）时才可向AES_IVRx 寄存器执行写入，这可确保良好的AES 行为。

在 AES 使能的情况下读取此寄存器会返回值 0x00000000。

在 AES 已禁止的情况下读取此寄存器会返回最新计数器值（用于管理挂起模式）。

在 CTR 模式下，MODE[1:0]设置成 11,10,或 00 都是加密模式，不能设置为 01。

31.5 数据交换

128 位的数据可以通过 4 次连续 32 位写 AES_DINR 寄存器而写入。写的时候高 32 位[127:96]先写入，低 32 位[31:0]最后写入。

128 位的运算结果可以通过 4 次连续读 AES_DOUTR 寄存器读出。读的时候先得到的是结果的高 32 位[127:96]，最后得到的是结果的最低 32 位[31:0]。

数据交换

AES_CR.DATATYPE 位提供的不同交换模式来处理写入到 AES_DINR 的数据。AES_DINR 的数据交换处理在进行 AES 运算之前完成。在 AES 运算完成后，把结果进行数据交换处理之后再存入 AES_DOUTR 寄存器。

注意：交换操作仅与 AES_DOUTR 和 AES_DINR 寄存器有关。AES_KEYRx 和 AES_IVRx 寄存器不受影响。

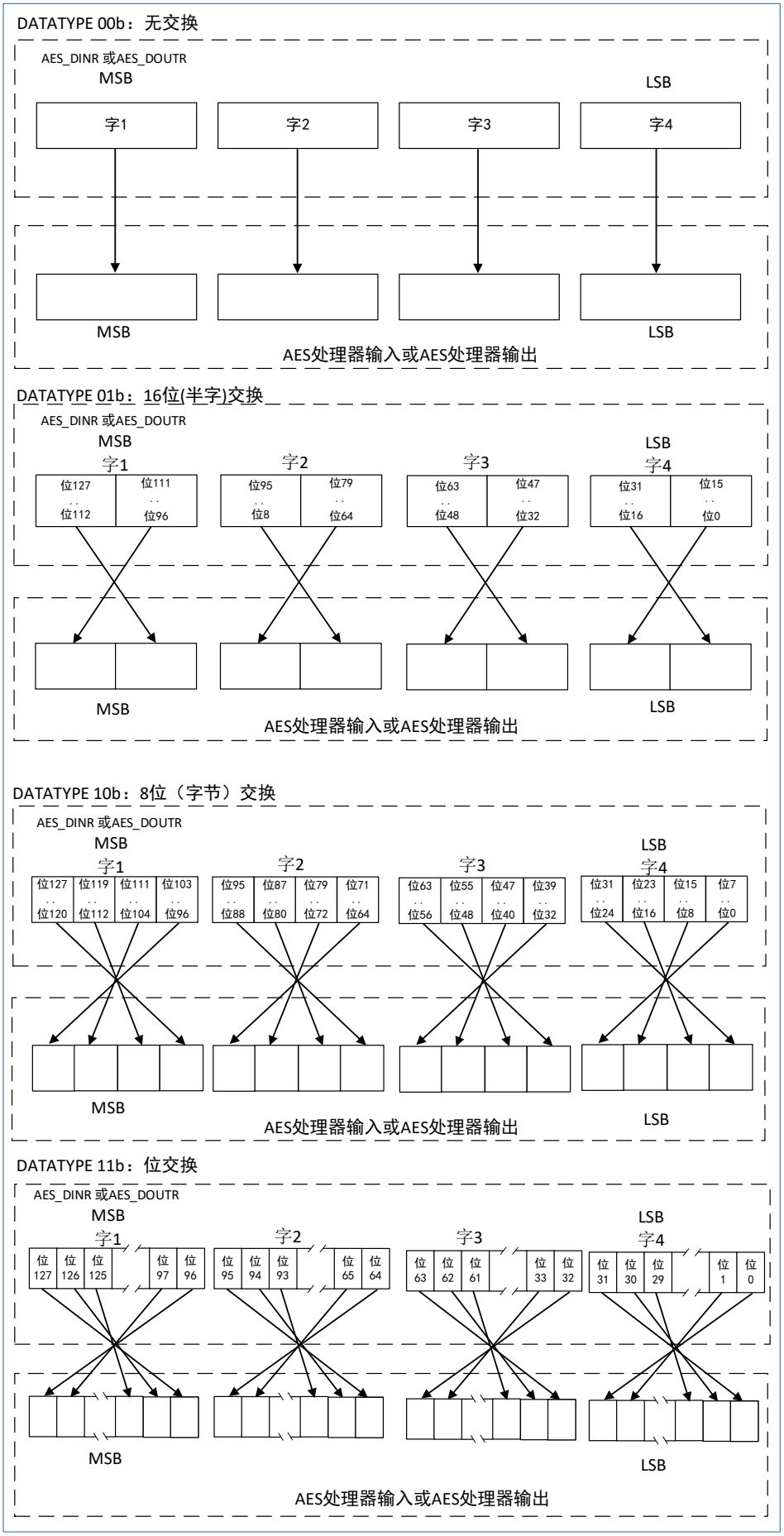


图 31-8 根据数据类型构建 128 位块

31.6 工作模式

31.6.1 模式 1：加密

1. 通过将 AES_CR.EN 位置 0 禁止 AES。
2. 通过在 AES_CR 寄存器中编程 MODE[1:0]=00 来配置模式 1，并通过对 CHMOD[1:0]位进行编程来选择需要执行哪种类型的链接算法。
3. 如果选择 CTR 或 CBC 模式，则写入 AES_KEYRx 寄存器（加密密钥）和 AES_IVRx 寄存器。如果选择 ECB 模式，则不使用 AES_IVRx 寄存器。
4. 通过将 AES_CR.EN 位置 1 使能 AES。
5. 对 AES_DINR 寄存器执行 4 次写入操作，以输入明文（MSB 优先），如图 31-9 所示。
6. 等待 AES_SR.CCF 标志置 1。
7. 对 AES_DOUTR 寄存器执行 4 次读取操作，以获取密文（MSB 优先），如图 31-9 所示。
8. 重复执行步骤 5、6、7，处理使用相同加密密钥的所有块。

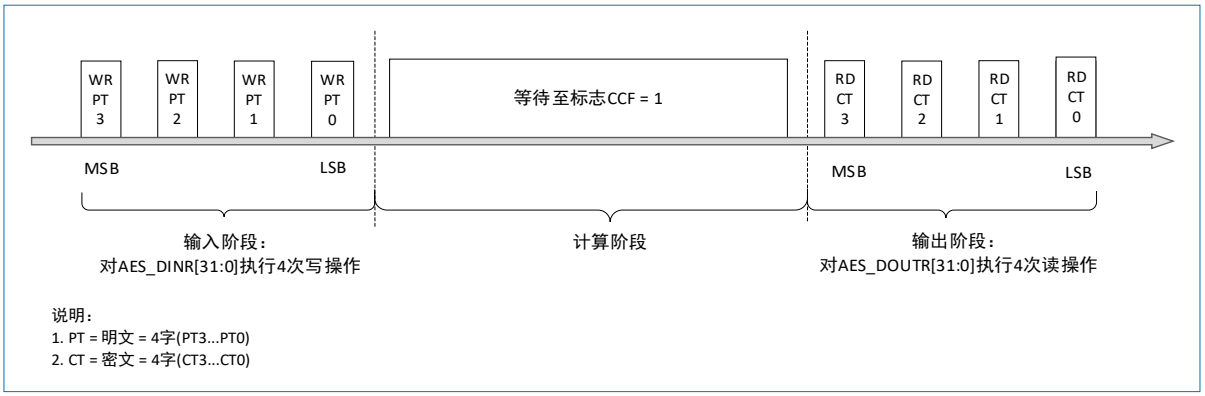


图 31-9 模式 1：加密

31.6.2 模式 2：密钥生成

1. 通过将 AES_CR.EN 位置 0 以禁止 AES。
2. 通过在 AES_CR 寄存器中编程 MODE[1:0]=01 配置模式 2。请注意，CHMOD[1:0]位在这种情况下没有意义，因为该密钥生成模式与所选链接算法无关。
3. 将加密密钥写入到 AES_KEYRx 寄存器中，以获取生成密钥。（写入到 AES_IVRx 中不起作用。）
4. 通过将 AES_CR.EN 位置 1 使能 AES。
5. 等待 AES_SR.CCF 标志置 1。
6. 生成密钥会自动放入 AES_KEYRx 寄存器。如有需要，可读取 AES_KEYRx 寄存器获得解密密钥。AES 由硬件禁止。
7. 若需重新开始计算生成密钥，请重复步骤 3、4、5、6。

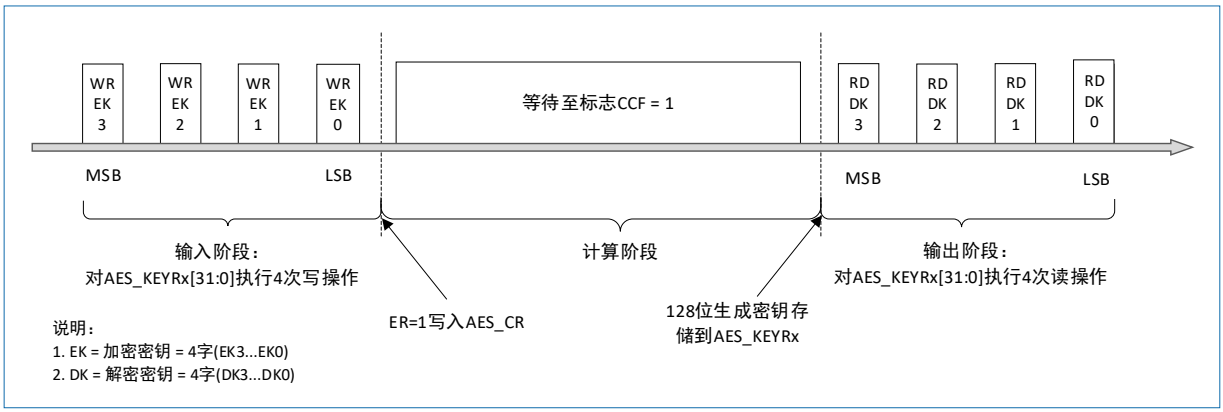


图 31-10 模式 2：密钥生成

31.6.3 模式 3：解密

1. 通过将 AES_CR.EN 位置 0 以禁止 AES。
2. 通过在 AES_CR 寄存器中编程 MODE[3:0]=10 来配置模式 3。
3. 将解密密钥写入 AES_KEYRx 寄存器（如果已使用模式 2，密钥生成将生成密钥存储到 AES_KEYRx 寄存器中，则可略过此步）。如果选择 CTR 或 CBC 模式，则写入 AES_IVRx 寄存器。如果选择 ECB 模式，则不使用 AES_IVRx 寄存器。
4. 通过将 AES_CR.EN 位置 1 以使能 AES。
5. 对 AES_DINR 寄存器执行 4 次写入操作，以输入密文（MSB 优先），如下图。
6. 等待 AES_SR.CCF 标志置 1。
7. 对 AES_DOUTR 寄存器进行 4 次读取操作，以获取明文（MSB 优先），如下图。
8. 重复执行步骤 5、6、7，处理使用相同加密密钥（存储在 AES_KEYRx 寄存器中）的块。

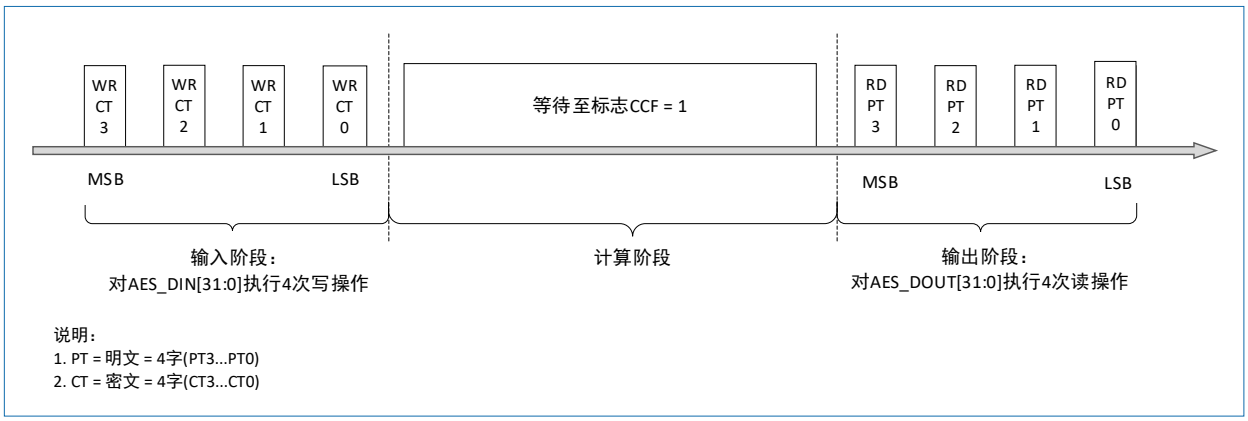


图 31-11 模式 3：解密

31.6.4 模式 4：单次解密

1. 通过将 AES_CR.EN 位置 0 以禁止 AES。
2. 通过在 AES_CR 寄存器中编程 MODE[1:0]=11 配置模式 4。如果 AES 配置为 CTR 模式，则禁止使用此模式。如果软件写入 MODE[1:0] = 11 和 CHMOD[1:0]=10，则会强制进入 CTR 解密模式。*（注意：禁止配置 CHMOD[1:0]=01（选择 CBC 模式），否则只有第一次单次解密是正确的，后面都会是错误的。）*
3. 将加密密钥写入 AES_KEYRx 寄存器。

4. 通过将 AES_CR.EN 位置 1 使能 AES。

5. 对 AES_DINR 寄存器执行 4 次写入操作，以输入密文（MSB 优先），如下图所示。

6. 等待 AES_SR.CCF 标志置 1。

7. 对 AES_DOUTR 寄存器进行 4 次读取操作，以获取明文（MSB 优先），如下图所示。

8. 重复执行步骤 5、6、7，处理所有使用相同加密密钥的块。

说明：运行单次解密运算后，AES_KEYRx 会被替换为解密密钥，因此，在调用单次解密运算时，每次都需要重新输入密钥到 AES_KEYRx 寄存器，即使本次使用的密钥和上一次使用的密钥相同。

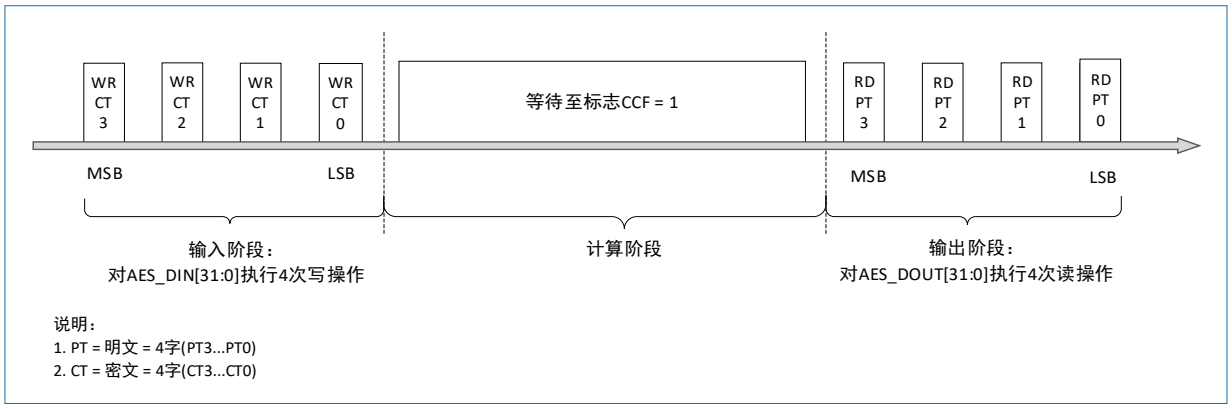


图 31-12 模式 4：密钥生成和解密

31.7 AES DMA 接口

AES 加速器提供连接 DMA 控制器的接口。

必须对 DMA 进行配置才能传输字。

可将 AES 关联到两个不同的 DMA 请求通道：

- 输入的 DMA 请求通道：如果将 AES_CR.DMAINEN 位置 1，每次 AES 要将字写入 AES_DINR 寄存器时，都会在 INPUT 阶段发起 DMA 请求（AES_IN）。DMA 通道必须配置为存储器到外设模式，数据大小为 32 位。
- 输出的 DMA 请求通道：如果将 DMAOUTEN 位置 1，每次 AES 要从 AES_DOUTR 寄存器读取字时，都会在 OUTPUT 阶段发起 DMA 请求（AES_OUT）。DMA 通道必须配置为外设到存储器模式，数据大小为 32 位。

AES 会为每个阶段使能四个 DMA 请求，如下图 31-13 和图 31-14 对此进行了介绍。

DMA 请求会一直产生，直到 AES 被禁用。因此，128 位数据块处理结束时的数据输出阶段之后，AES 会自动切换到下一个数据块的新数据输入阶段（如果存在）。

说明：对于模式 2（密钥生成），可通过软件使用 CPU 访问 AES_KEYRx 寄存器。不会为此提供 DMA 通道。因此，AES_CR.DMAINEN 位和 DMAOUTEN 位在该模式下不起作用。不支持在模式 4（单次解密）下使用 DMA。

当 DMAOUTEN=1 时，CCF 标志不起作用。该情况下软件不需要读取该标志。如果应用需要禁止 AES 来取消 DMA 管理并为数据输入或数据输出阶段使用 CPU 访问，该位可能保持为 1，必须由软件清零。

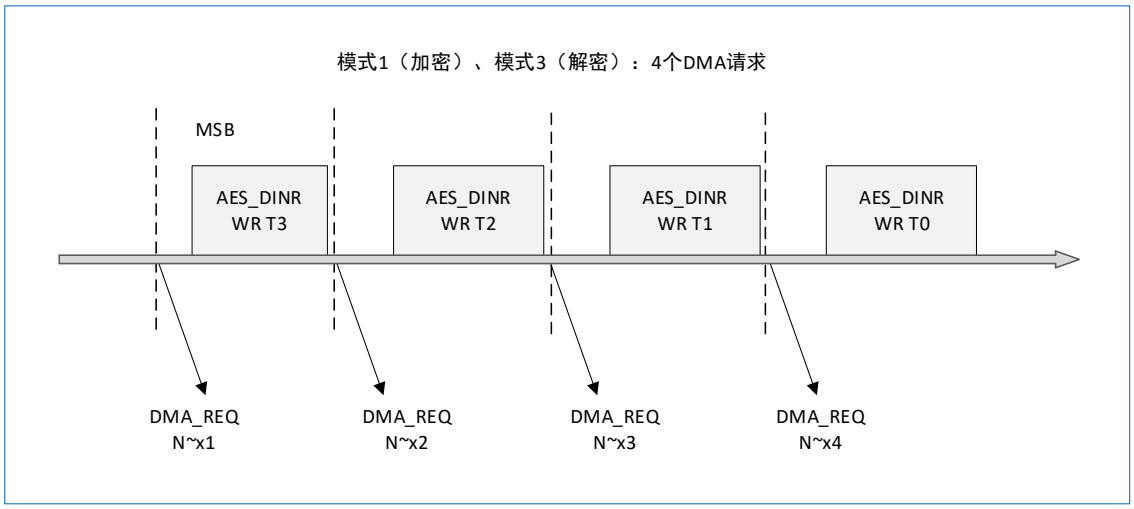


图 31-13 输入阶段的 DMA 请求和数据传输（AES_IN）

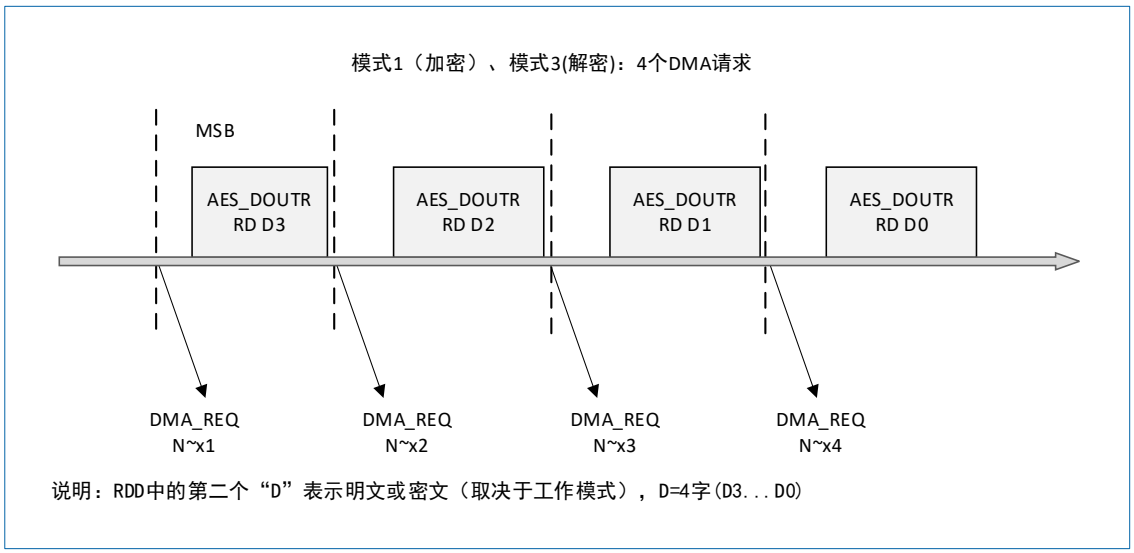


图 31-14 输出阶段的 DMA 请求和数据传输（AES_OUT）

31.8 错误标志

如果在计算阶段或输入阶段检测到未预期的读取操作，则会将 AES_SR.RDERR 标志置 1。

如果在输出阶段或计算阶段检测到未预期的写入操作，则会将 AES_SR.WRERR 标志置 1。

可通过将 AES_CR 寄存器中的相应位置 1 的方式将标志清零（CCFC 位可将 CCF 标志清零，ERRC 位可将 WERR 和 RDERR 标志清零）。

如果之前已将 AES_CR.ERRIE 位置 1，则当其中一个错误标志置 1 时，可生成中断。

如果检测到错误，则 AES 不会被硬件禁止并将继续照常进行处理。

31.9 处理时间

下表列出了每种工作模式下处理块（128 位）需要的时间。

表 31-2 处理时间（时钟周期数）

工作模式		输入阶段	计算阶段	输出阶段	总计
模式 1：加密	128 位密钥	8	57	4	69
	192 位密钥	10	67	4	81

工作模式		输入阶段	计算阶段	输出阶段	总计
模式 2：解密密钥生成	256 位密钥	12	77	4	93
	128 位密钥	4	23	-	27
	192 位密钥	6	27	-	33
	256 位密钥	8	31	-	39
模式 3：解密	128 位密钥	4	57	4	65
	192 位密钥	6	67	4	77
	256 位密钥	8	77	4	89
模式 4：单次解密	128 位密钥	8	82	4	94
	192 位密钥	10	96	4	110
	256 位密钥	12	110	4	126

31.10 AES 中断

AES 中断如表 31-3 所示。

表 31-3 AES 中断请求

中断事件	事件标志	使能控制位
AES 计算完成标志	CCF	CCFIE
AES 读取错误标志	RDERR	ERRIE
AES 写入错误标志	WRERR	ERRIE

每一个中断请求源可以从 AES_SR 寄存器读取获得。

31.11 挂起与上下文恢复

AES 挂起

在 AES 运算的过程中，如果高优先级的中断发生，并且该中断服务程序也需要使用 AES 时，需要把当前的 AES 挂起。为了保证计算的正确性，需要在一个计算完成后才挂起 AES。

1. 保存 AES.CR1/CR2/SR1/SR2 的值，并压栈。
2. 如果使用了 DMA，清除 AES.DMAINEN。
3. 等待 AES_SR.CCF 变高。
4. 如果使用了 DMA，清除 AES_CR.DMAOUTEN。DMAOUTEN 清除后，硬件依然会完成 DMA 读 DOUT 的操作。如果没有使用 DMA，连续读 4 次 AES_DOUTR 寄存器并把结果压栈。
5. 写 AES_CR.CCFC 寄存器清除 CCF 标志。
6. 把 AES_CR.EN 清零，使 AES 关闭。
7. 连续 4 次读取 AES_DINR 并压栈。

8. 读取 AES_CR、AES_CR2、AES_KEYRx 寄存器并压栈。

上下文恢复

1. 把 AES_CR.EN 位清零禁用 AES；
2. 恢复 AES_KEYRx 寄存器的配置；
3. 恢复 AES_IVRx 寄存器的配置；
4. 恢复 AES_DOUTR 寄存器的配置；
5. 恢复 AES_DINR 寄存器的配置；
6. 恢复 AES_CR、AES_CR2 寄存器的配置；
7. 把 AES_CR2.CCF_SET 位和 AES_CR2.INT_RESUME 位置 1 以恢复上文。

31.12 AES 寄存器

基地址：0x5006 0000

空间大小：0x400

31.12.1 AES 控制寄存器（AES_CR）

偏移地址：0x00

复位值：0x0000 0000

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

1	1	1	12	11	10	9	8	7	6	5	4	3	2	1	0
Res			DMAOUTEN	DMAINEN	ERRIE	CCFIE	ERRC	CCFC	CHMOD[1:0]		MODE[1:0]		DATATYPE[1:0]		EN
			rw	rw	rw	rw	rw	rw	rw		rw		rw		rw

位 31:13	Res：保留 必须保持复位值。
位 12	DMAOUTEN：数据输出阶段使能 DMA 管理（Enable DMA management of data output phase） • 0：禁止 DMA（在数据输出阶段） • 1：使能 DMA（在数据输出阶段） 如果 DMAOUTEN 位置 1，则会在模式 1、3 下为输出数据阶段生成 DMA 请求。该位在模式 2（解密密钥生成）和模式 4（单次解密）下不起作用。模式 4（单次解密）下不支持 DMA 功能。
位 11	DMAINEN：数据输入阶段使能 DMA 管理（Enable DMA management of data input phase） • 0：禁止 DMA（在数据输入阶段） • 1：使能 DMA（在数据输入阶段） 如果 DMAINEN 位置 1，则会在模式 1、3 下为输入数据阶段生成 DMA 请求。该位在模式 2（解密密钥生成）和模式 4（单次解密）下不起作用。模式 4（单次解密）下不支持 DMA 功能。
位 10	ERRIE：错误中断使能（Error interrupt enable） • 0：禁止错误中断 • 1：使能错误中断 如果 RDERR 或 WRERR 中至少有一个标志置 1，则会生成中断。

位 9	CCFIE：CCF 标志中断使能（CCF flag interrupt enable） • 0：禁止 CCF 中断 • 1：使能 CCF 中断 如果 CCF 标志置 1，则会生成中断。
位 8	ERRC：错误清零（Error clear） 向该位写入 1 会将 RDERR 和 WRERR 标志清零。 该位始终读为 0。
位 7	CCFC：计算完成标志清零（Computation Complete Flag Clear） 向该位写入 1 时会将 CCF 标志清零。 该位始终读为 0。
位 6:5	CHMOD[1:0]：AES 链接模式（AES chaining mode） • 00：电子密码本（ECB） • 01：密码块链接（CBC） • 10：计数器模式（CTR） • 11：保留 仅可在 AES 已禁用的情况下更改 AES 链接模式。禁止在 AES 已使能的情况下写入这些位，以免出现不可预期的 AES 行为。
位 4:3	MODE[1:0]：AES 工作模式（AES operating mode） • 00：模式 1：加密 • 01：模式 2：解密密钥生成 • 10：模式 3：解密 • 11：模式 4：单次密钥生成+解密 仅可在 AES 已禁用的情况下更改工作模式。禁止在 AES 已使能的情况下写入该位域，以免出现不可预期的 AES 行为。
位 2:1	DATATYPE[1:0]：数据类型选择（Data type selection） • 00：32 位数据无交换。 • 01：16 位数据或半字。在字中交换每个半字。例如，如果写入 AES_DINR 寄存器的四个 32 位数据之一为 0x764356AB，则提供给加密块的值为 0x56AB7643。 • 10：8 位数据或字节。在字中交换所有字节。例如，如果写入 AES_DINR 寄存器的四个 32 位数据之一为 0x764356AB，则提供给加密块的值为 0xAB564376。 • 11：位数据在字中交换所有位。例如，如果写入 AES_DINR 寄存器的四个 32 位数据之一为 0x764356AB，则提供给加密块的值为 0xD56AC26E。 仅可在 AES 已禁用的情况下更改数据类型选择。禁止在 AES 已使能的情况下写入这些位，以免出现不可预期的 AES 行为。 DATATYPE 会同时影响写入 DINR 的数据和保存到 DOUTR 的数据。
位 0	EN：AES 使能（AES enable） • 0：禁 AES • 1：使能 AES 可随时通过将该位清零的方式重新初始化 AES：EN 置 1 后，AES 准备好开始处理新块。如果在模式 2（解密密钥生成）下完成 AES 计算，则由硬件清零该位。

31.12.2 AES 控制寄存器 2（AES_CR2）

偏移地址：0x80

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res									CCF_SET	INT_RESUME	RANDOM_CLK_EN	Res		KEY_SIZE[1:0]	
									rw	rw	rw			rw	

位 31:7	Res：保留 必须保持复位值。
位 6	CCF_SET：强制置位 ASE_SR.CCF（Force to set bit AES_SR.CCF） 该位用于在中断返回使恢复上下文。 • 0：不起作用 • 1：把 AES_SR.CCF 置 1 读该位始终读为 0。
位 5	INT_RESUME：指示 AES 从中断服务程序恢复（Indicate the AES to resume from the interrupt service program） 如果在 AES 进行 CBC 运算时被中断，同时中断服务程序也使用了 AES，则在中断退出时需要软件置位 INT_RESUME 寄存器。因为 CBC 模式被中断后，恢复时硬件需要区分当前是否为第一个 block，如果软件不置位 INT_RESUME，则 CBC 模式中断恢复后会计算出错。 当 INT_RESUME 被置 1 后，会被硬件清零，当 AES 开始恢复执行运算会立即清除 INT_RESUME 位。
位 4	RANDOM_CLK_EN：TRNG 输出的随机数来随机化 AES 时钟（Random number of TRNG output to randomize the AES clock） • 0：不随机化 AES 时钟。 • 1：使用 TRNG 输出的随机数来随机化 AES 时钟，加强 AES 的安全性。
位 3:2	Res：保留 必须保持复位值。
位 1:0	KEY_SIZE[1:0]：密钥长度配置（Key size） • 00：128 位密钥长度 • 01：192 位密钥长度 • 10：256 位密钥长度 • 11：无效

31.12.3 AES 状态寄存器（AES_SR）

偏移地址：0x04

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res													WRERR	RDERR	CCF
													r	r	r

位 31:3	Res：保留 必须保持复位值。
位 2	WRERR：写入错误标志（Write error flag） 如果检测到 AES_DINR 寄存器发生了非预期的写入操作（计算阶段或数据输出阶段），则由硬件将该位置 1。如果 AES_CR.ERRIE 位之前已置 1，将生成中断。该位对 AES 无影响，即使 WRERR 置 1，AES 也会继续运行。 通过将 AES_CR.ERRC 位置 1，该位由软件清零。 • 0：未检测到写入错误 • 1：检测到写入错误
位 1	RDERR：读取错误标志（Read error flag） 如果检测到 AES_DOUTR 寄存器发生了非预期的读取操作（计算阶段或数据输入阶段），则会通过硬件将该位置 1。如果 AES_CR.ERRIE 位之前已置 1，将生成中断。该位对 AES 无影响，即使 RDERR 置 1，AES 也会继续运行。 通过将 AES_CR.ERRC 位置 1，该位由软件清零。 • 0：未检测到读取错误 • 1：检测到读取错误
位 0	CCF：计算完成标志（Computation complete flag） 当计算完成时，该位由硬件置 1。如果 AES_CR.CCFIE 位之前已置 1，将生成中断。 也可以通过向 AES_CR2.CCF_SET 写 1，置位 CCF 标志寄存器。 通过向 AES_CR.CCFC 位写入 1 清零。 • 0：计算未完成 • 1：计算已完成

31.12.4 AES 状态寄存器 2（AES_SR2）

偏移地址：0x94

复位值：0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res											FST_CAL	INPUT_CN[1:0]		Res	
											r	r			

位 31:5	Res：保留 必须保持复位值。
位 4	FST_CAL：AES_EN 使能后，第一个运算的状态。（The state of the first operation after the AES_EN enable） 状态由硬件更新，当 AES_EN 为 0 时，把 FST_CAL 置 0；如果 AES_EN 使能后，第一个运算还未完成，硬件把 FST_CAL 置 1，完成第一个运算后硬件会自动把 FST_CAL 清零。 • 0：CR.AES_EN 使能后的第一个运算已经完成或者 AES_EN 无效。 • 1：CR.AES_EN 使能后的第一个运算还未完成。
位 3:2	INPUT_CN[1:0]：状态由硬件更新，标识已经输入了多少个字的数据到 DINR。（The Number of words already entered into the DINR register）

	• 00：表示还未输入或者已经完成输入 4 个字的数据 • 01：表示输入了 1 个字到 DINR • 10：表示输入了 2 个字到 DINR • 11：表示输入了 3 个字到 DINR 从中断恢复的时候，可以通过 INPUT_CNT 的值来保证恢复现场。
位 1:0	Res：保留 必须保持复位值。

31.12.5 AES 数据输入寄存器（AES_DINR）

偏移地址：0x08

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DINR[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DINR[15:0]															
rw															

位 31:0	DINR[31:0]：数据输入寄存器（Data input register） 输入阶段必须对该寄存器执行 4 次写入操作： • 在模式 1（加密）下，必须从 MSB 到 LSB 写入 4 个代表明文的字。 • 在模式 2（密钥生成）下，不会使用该寄存器。 • 在模式 3（解密）和模式 4（单次密钥生成+解密）下，必须从 MSB 到 LSB 写入 4 个代表密文的字。 说明：该寄存器必须通过 32 位数据宽度进行访问。

31.12.6 AES 数据输出寄存器（AES_DOUTR）

偏移地址：0x0C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DOUTR[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DOUTR[15:0]															
rw															

位 31:0	DOUTR[31:0]：数据输出寄存器（Data output register） 将 CCF 标志（计算完成标志）置 1 后，对该数据寄存器读取 4 次可得到 128 位输出结果： • 在模式 1（加密）下，必须从 MSB 到 LSB 读取 4 个代表密码文本的字。 • 在模式 2（密钥生成）下，不需要读取该寄存器。 • 在模式 3（加密）和模式 4（单次密钥生成+解密）下，读取的 4 个字代表从 MSB 到 LSB 的明文。 说明：该寄存器必须通过 32 位数据宽度进行访问。

31.12.7 AES 密钥寄存器 0（AES_KEYR0）（key[31:0]）

偏移地址：0x10

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEYR0[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEYR0[15:0]															
rw															

位 31:0	KEYR0[31:0]：AES 密钥寄存器 0（LSB of AES key register 0, key[31:0]） 必须在将 AES_CR.EN 位置 1 之前写入该寄存器。 在模式 1（加密）、模式 2（密钥生成）和模式 4（单次密钥生成+解密）下，要写入的值是从 MSB 开始的加密密钥，也就是 Key[255:224]。 在模式 3（解密）下，要写入的值是从 MSB 开始的解密密钥，也就是 Key[255:224]。如果在该解密模式下向寄存器写入加密密钥，则在 AES 使能前读取该寄存器将返回加密值。在 CCF 标志置 1 之后读取该寄存器将返回生成密钥。 在 AES 已使能时读取该寄存器会返回不可预测的值。 说明：在模式 4（单次生成密钥+解密）下，该寄存器不包含生成密钥，但始终包含加密密钥值。

31.12.8 AES 密钥寄存器 1（AES_KEYR1）（Key[63:32]）

偏移地址：0x14

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEYR1[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEYR1[15:0]															
rw															

位 31:0	KEYR1[31:0]：AES 密钥寄存器 1（AES key register1, key[63:32]） 请参见 AES_KEYR0 的说明。

31.12.9 AES 密钥寄存器 2（AES_KEYR2）（Key[95:64]）

偏移地址：0x18

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEYR2[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEYR2[15:0]															
rw															

位 31:0	KEYR2[31:0]：AES 密钥寄存器 2（AES key register2, key[95:64]） 请参见 AES_KEYR0 的说明。

31.12.10 AES 密钥寄存器 3（AES_KEYR3）（key[127:96]）

偏移地址：0x1C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEYR3[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEYR3[15:0]															
rw															

位 31:0	KEYR3[31:0]：AES 密钥寄存器 3（AES key register3, key[127:96]） 请参见 AES_KEYR0 的说明。

31.12.11 AES 密钥寄存器 4（AES_KEYR4）（key[159:128]）

偏移地址：0x84

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEYR4[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEYR4[15:0]															
rw															

位 31:0	KEYR4[31:0]：AES 密钥寄存器 4（AES key register4, key[159:128]） 请参见 AES_KEYR0 的说明。

31.12.12 AES 密钥寄存器 5（AES_KEYR5）（key[191:160]）

偏移地址：0x88

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEYR5[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEYR5[15:0]															
rw															

位 31:0	KEYR5[31:0]：AES 密钥寄存器 5（AES key register 5, key[191:160]） 请参见 AES_KEYR0 的说明。

31.12.13 AES 密钥寄存器 6（AES_KEYR6）（key[223:192]）

偏移地址：0x8C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEYR6[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEYR6[15:0]															
rw															

位 31:0	KEYR6[31:0]：AES 密钥寄存器 6（AES key register 6, key[223:192]） 请参见 AES_KEYR0 的说明。

31.12.14 AES 密钥寄存器 7（AES_KEYR7）（key[255:224]）

偏移地址：0x90

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEYR7[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEYR7[15:0]															
rw															

位 31:0	KEYR7[31:0]：AES 密钥寄存器 7（MSB of AES key register 7, key[255:224]） 请参见 AES_KEYR0 的说明。

31.12.15 AES 初始化向量寄存器 0（AES_IVR0）（IVR[31:0]）

偏移地址：0x20

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVR0[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVR0[15:0]															
rw															

位 31:0	IVR0[31:0]：初始化向量寄存器 0（Initialization vector register 0） 必须在将 AES_CR.EN 位置 1 之前写入该寄存器。只有 CBC 或者 CTR 模式才会用到 IVRx 寄存器。 在 CTR 模式下，每次 AES 运算后 IVR0 的值会加 1。

31.12.16 AES 初始化向量寄存器 1（AES_IVR1）（IVR[63:32]）

偏移地址：0x24

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVR1[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVR1[15:0]															
rw															

位 31:0	IVR1[31:0]：初始化向量寄存器 1（Initialization vector register 1）

	必须在将 AES_CR.EN 位置 1 之前写入该寄存器。只有 CBC 或者 CTR 模式才会用到 IVRx 寄存器。

31.12.17 AES 初始化向量寄存器 2（AES_IVR2）（IVR[95:64]）

偏移地址：0x28

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVR2[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVR2[15:0]															
rw															

位 31:0	IVR2[31:0]：初始化向量寄存器 2（Initialization vector register 2） 必须在将 AES_CR.EN 位置 1 之前写入该寄存器。只有 CBC 或者 CTR 模式才会用到 IVRx 寄存器。

31.12.18 AES 初始化向量寄存器 3（AES_IVR3）（IVR[127:96]）

偏移地址：0x2C

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVR3[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVR3[15:0]															
rw															

位 31:0	IVR3[31:0]：初始化向量寄存器 3（Initialization vector register 3） 必须在将 AES_CR.EN 位置 1 之前写入该寄存器。只有 CBC 或者 CTR 模式才会用到 IVRx 寄存器。

32 散列处理器（HASH）

散列处理器完全兼容安全散列算法（SHA-256）。散列处理器可计算最多（2^{64}-1）位的消息，且得出经 FIPS（联邦信息处理标准）认可的 256 位长度的摘要。

32.1 HASH 主要特性

- 适合于数据验证应用，符合以下标准：
  - FIPS PUB 180-2（联邦信息处理标准出版物 180-2）
  - 安全散列标准规范（SHA-256）
- 快速计算 SHA-256。
- 支持 AHB 从外设。
- 可自动填充来完成输入位串，从而适应模数为 512（16×32 位）的消息摘要计算。
- 连续消息块的摘要所对应的多个 32 位字加到一起以形成整个消息的摘要。
  - 32 位数据字用于输入数据，支持字、半字、字节和位串表示法（仅采用小端模式数据表示法）。
  - 可自动交换，以兼容大端模式 SHA-256 计算标准（采用小端模式输入位串表示法）。
- 数据流自动控制，支持直接存储器访问（DMA）。

注意：SHA-256 算法中定义的填充是，指在 bx1 中添加一个位，然后在 bx0 中添加 N 位，以使总长度与 448%512 同余（512-64=448（长度））。然后，将使用 64 位整数（也就是原始消息长度的二进制表示）来补全消息。对于此散列处理器，用于输入消息的量是 32 位字，因此必须在消息的末尾处附加信息，也就是最后输入的 32 位字中有效位的数量。

32.2 HASH 功能说明

下图显示了散列处理器的框图。

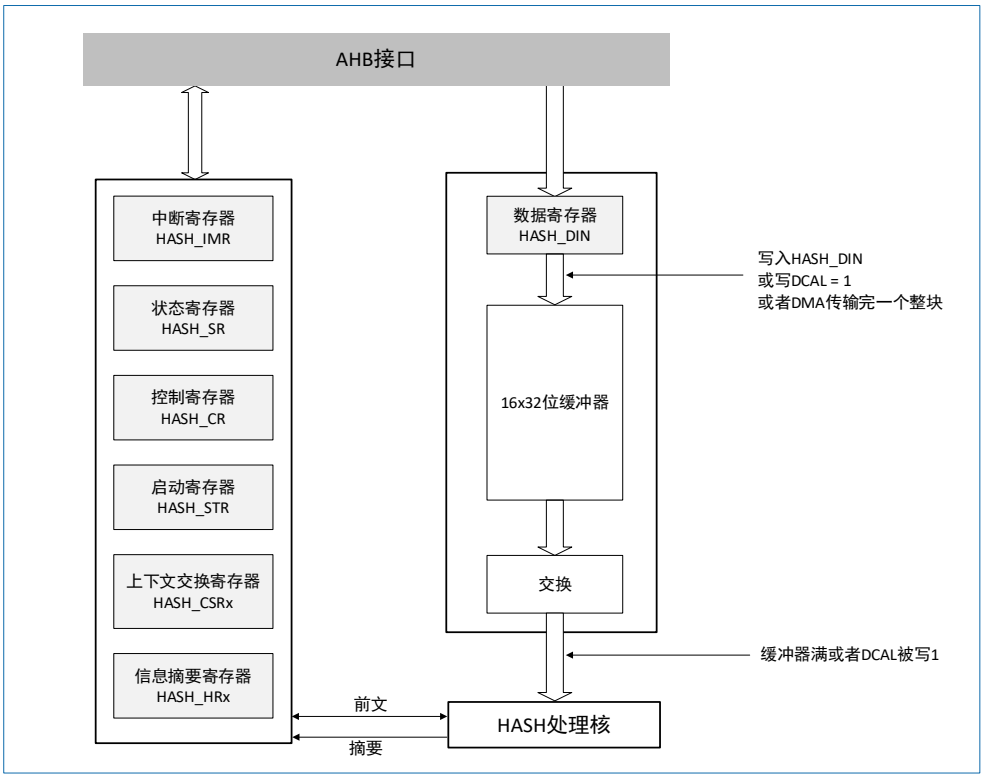


图 32-1 HASH 的框图

针对计算消息或数据文件的压缩表示，FIPSPUB180-2 标准详细说明了 SHA-256 安全散列算法。输入中提供的任何消息长度低于 2^{64} 位时，SHA-256 将生成一个 256 位的输出位串，称为消息摘要。然后可以通过数字签名算法来处理此消息摘要，以便生成或验证消息的签名。对消息摘要而不是对消息签名进行处理可提高流程的效率，因为消息摘要通常比消息要小得多。数字签名的创建程序和数字签名的验证程序必须使用相同散列算法。

SHA-256 安全可靠，因为要找出某个给定消息摘要对应的消息，或找出两个生成相同消息摘要的不同消息，在计算层面无法实现。对传输中的消息进行任何更改都极有可能产生不同的消息摘要，从而导致签名验证失败。有关 SHA-256 算法的详细信息，请参见 2002 年 8 月 1日颁布的 FIPSPUB180-2（联邦信息处理标准出版物 180-2）。

目前该标准的实施采用小端模式输入数据约定。例如，C 位串“abc”在存储器中必须表示为 24 位十六进制值 0x434241。

要由散列处理器处理的消息或数据文件应视为位串。消息长度为消息中的位数（空消息的长度为 0）。可以将该位串的 32 位视为构成了一个 32 位字。请注意，FIPSPUB180-2 标准约定如下：位串从左到右增长，位可以分组为字节（8 位）或字（32 位）（但某些实施还使用半字（16 位），并使用大端模式字节（半字）排序）。此约定主要是对于填充很重要（参见“32.2.4 消息填充”）。

32.2.1 处理的持续时间

计算消息的中间块（计算过程中用到的临时数据块）所花费的时间是为 50 个 HCLK 时钟周期。

必须将上述时间与模块的 16 字装入到处理器所需要的时间（对于 512 位块、至少为 16 个时钟周期）相加，作为处理的持续时间。该持续时间取决于最后一个块的长度以及密钥的大小。与处理中间块相比，此时间可能按 1 到 2.5 倍系数增加。

32.2.2 数据类型

通过将数据写入到 HASH_DIN 寄存器，数据将一次性输入到散列处理器 32 位（字）中。但原始位串可能是按照字节、半字或字来组织，甚至可能表示为位。由于系统存储器组织方式为小端模式，而 SHA-256 计算方式为大端模式，根据原始位串的分组方式，散列处理器将自动执行位、字节或半字交换操作。

使用散列控制寄存器（HASH_CR）中的 DATATYPE 位字段配置待处理的数据类型。消息的最低有效位必须在输入到散列处理器的第一个字中的位 0（右侧），位串的第 32 位必须在输入到散列处理器的第二个字中的位 0，依此类推。

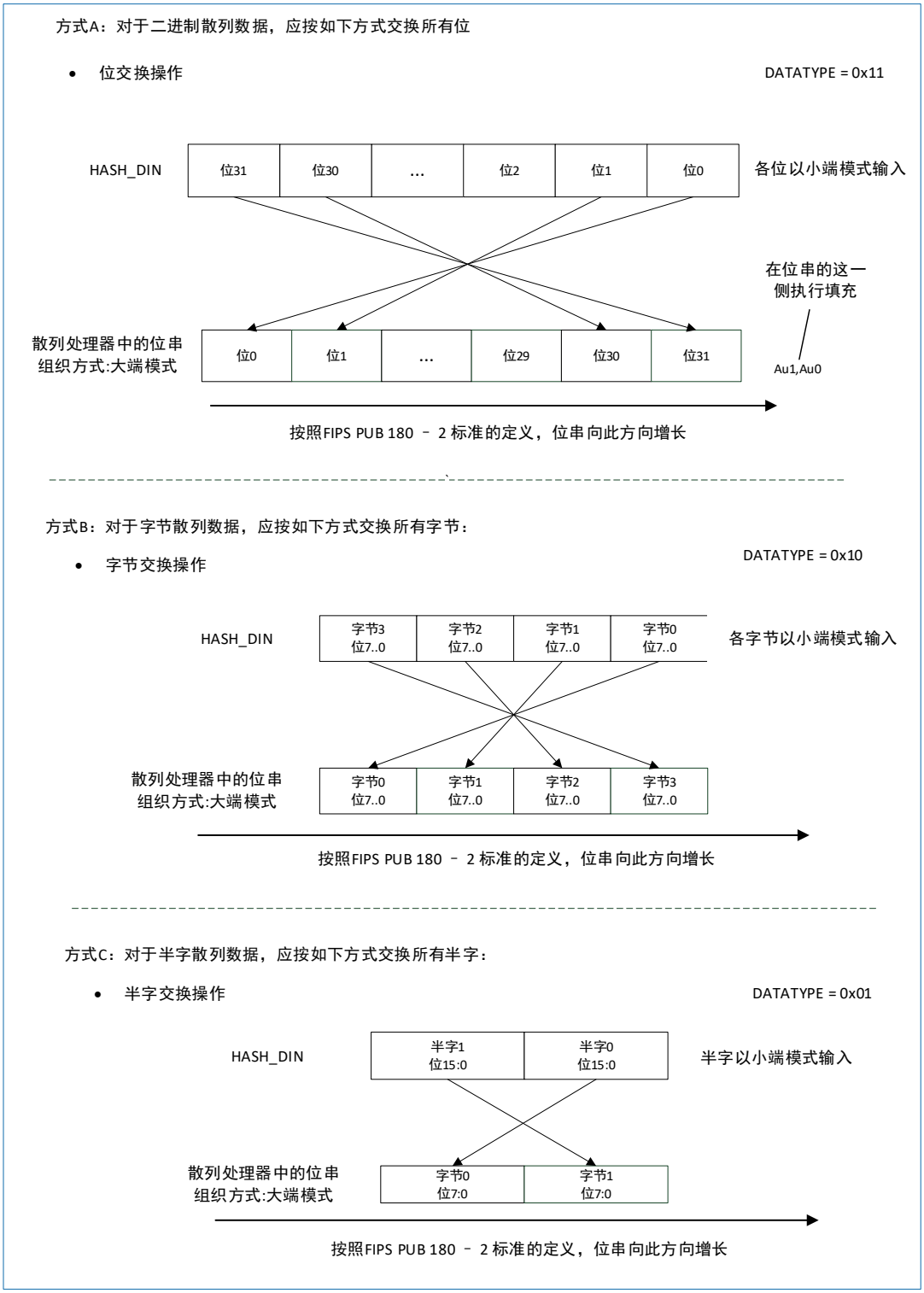


图 32-2 位、字节、半字交换

32.2.3 消息摘要计算

散列在计算消息摘要时，将按顺序处理 512 位的块。这样，DMA 或 CPU 每次将 16x32 位字（=512 位）写入到散列处理器时，散列将自动开始计算消息摘要。此操作称为“部分摘要计算”。

要处理的消息以写入到 HASH_DIN 寄存器的 32 位字来输入到外设。每次在寄存器中写入新数据时，HASH_DIN 寄存器的当前内容将传输到输入 FIFO（INFIFO）。HASH_DIN 和输入 FIFO 构成 17 字长度的 FIFO（名为 IN 缓冲器）。

只有在块的最后一个值已进入 INFIFO 之后，才能开始处理块。外设必须获取有关 HASH_DIN 寄存器

是否包含消息的最后一位的信息。可能发生两种情况：

- 未使用 DMA 时：
  - 如果是部分摘要计算，则方式是将额外的一个字（实际上是下一个块的第一个字）写入到 HASH_DIN 寄存器。然后，软件必须等到处理器再次就绪（当 DINIS=1 时），然后才能在 HASH_DIN 中写入新数据。
  - 如果是最终摘要计算（输入的最后一个块），则方式是将 DCAL 位写入到 1。
- 使用 DMA 时：
  - 将使用 DMA 控制器发送的信息来自动解释 HASH_DIN 寄存器的存储信息。
  - 当最后的模块通过 DMA 传送到 HASH_DIN，系统将 DCAL 位自动置 1 并启动最终摘要。
  - 将使用 DMA 控制器发送的信息来自动解释 HASH_DIN 寄存器的存储信息。

这一过程（数据输入和部分摘要计算）将继续，直至原始消息的最后一位写入到 HASH_DIN 寄存器。因为消息的长度（位数）可以是任何整数值，因此写入到散列处理器的最后一个字的有效位数在 1 到 32 之间。这个最后一个字的有效位数必须写到 HASH_STR 寄存器的 NBLW 位域中，以便消息填充能够在最终消息摘要计算之前得到正确处理。

此操作完成后，将 HASH_STR 的 DCAL 位置 1 将使散列处理器开始处理消息的最后一个输入块。此处理包括：

- 自动执行消息填充操作：此操作的目的是使填充的消息的总长度变为 512 的倍数。散列处理器在计算消息摘要时，将按顺序处理 512 位的块。
- 计算最终消息摘要。

如果使能 DMA，DMA 会在传输最后一个数据字时向散列处理器提供信息。然后将自动执行填充和摘要计算，就如同 DCAL 已写为 1 一样。

32.2.4 消息填充

消息填充的方法是：在原始消息的结尾添加一个“1”，后跟 M 个“0”以及一个 64 位整数（其数值等于消息长度 L（位数）），目的是生成一个长度为 512 的填充消息块。经过这三个填充操作后生成的消息长度为 L+1+M+64（512 位的整数倍）。

“1”将添加到 HASH_DIN 寄存器中写入的最后一个字，位的位置由 NBLW 位字段定义，而其余更高的位将被清零（“0”）。

示例：我们假定原始消息是采用 ASCII 二进制编码格式的“abc”，长度 L=24。

字节 0	字节 1	字节 2	字节 3
01100001	01100010	01100011	UUUUUUUU

NBLW 必须以值 24 加载：“1”在位串中的位 24 处追加（在以上位串中从左到右开始计数）。这对应于 HASH_DIN 寄存器中的位 31（小端模式约定）：01100001011000100110001111UUUUUUU。

由于 L=24，因此以上位串中的位数是 25，并将追加 423 个“0”，现在为 448。这将生成（十六进制，大端模式）：

61626380	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000		

将追加双字表示法格式的 L 值，即 0000000000000018。因此，十六进制格式的最终填充消息为：

61626380	00000000	00000000	00000000

00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000018

如果散列编程为使用小端模式字节输入格式，则以上消息必须通过以下步骤来输入：

1. 0xUU636261 写入到 HASH_DIN 寄存器（其中的“U”意味着无关）。
2. 0x18 写入到 HASH_STR 寄存器（写入到 HASH_DIN 寄存器的最后一个字中的有效位数是 24，因为原始消息长度为 24 位）。
3. 0x10 写入到 HASH_STR 寄存器以启动消息填充和摘要计算。当 NBLW≠0x00 时，消息填充会将“1”放入到 HASH_DIN 寄存器中（位的位置由 NBLW 值定义），并且在位位置[31:（NBLW+1）]处插入“0”。当 NBLW=0x00 时，消息填充使用值 0x00000001 插入一个新字。然后将添加一个全零字（0x00000000）以及双字表示法格式的消息长度，从而形成 16x32 位字的块。
4. 将执行散列计算，并且随后在 HASH_HRx 寄存器（x=0..7）中消息摘要可供 SHA-256 算法使用。例如：

   H0=0xA9993E36

   H1=0x4706816A

   H2=0xBA3E2571

   H3=0x7850C26C

   H4=0x9CD0D89D

32.2.5 散列运算

当 HASH_CR 的 MODE 位为 0 时，将 HASH_CR 寄存器的 INIT 置 1 将选择散列函数（SHA-256）。同时 INIT 位置 1 时，使用 ALGO 位来选择算法（SHA-256）。

随后，通过逐字将消息写入到 HASH_DIN 寄存器来发送消息。写入了 512 位的块（即 16 个字）后，部分摘要计算将在写入下一个块的第一个数据之后启动。对于 SHA-256 算法，散列处理器在 50 个周期内保持运行。

随后可以重复此过程，直至消息的最后一个字。如果使用了 DMA 传输，请参见 “11 DMA 控制器”。否则，如果消息长度不是 512 位的整倍数，则必须写入 HASH_STR 寄存器以启动最终摘要的计算。

计算完成之后，可以从 HASH_HR0-HASH_HR7 寄存器中读取摘要。

32.2.6 前文交换

可以中断散列/HMAC 进程以执行优先级更高的其它处理，并可在稍后当优先级更高的任务完成后再执行被中断的进程。要这样做，必须将被中断的任务的上下文从散列寄存器保存到存储器，然后从存储器恢复到散列寄存器。

以下说明由软件或 DMA 控制数据流的过程。

软件加载数据的过程

仅当目前未在处理任何块时，才能保存上下文。即，必须等待 DINIS=1（最后的块已处理并且输入 FIFO 为空）或 NBW≠0（FIFO 未满，并且未在进行任何处理）。

- 保存上下文：将以下寄存器的内容存储到存储器中：
  - HASH_IMR
  - HASH_STR
  - HASH_CR

- HASH_CSR0 到 HASH_CSR37

- 恢复前文：在高优先级任务完成后，可以恢复上下文。请遵循以下顺序：

1. 将存储器中保存的值写入以下寄存器中：HASH_IMR、HASH_STR 和 HASH_CR。
2. 通过将 HASH_CR.INIT 位置 1 来初始化散列处理器。
3. 将存储器中保存的值写入 HASH_CSR0 到 HASH_CSR37 寄存器。
4. 现在便可以从之前的中断点重新开始处理。

DMA 加载数据的过程

在此情况下，无法预测 DMA 传输是否在进行，也无法预测处理是否在进行。因此，必须停止 DMA 传输，然后等到散列就绪才能中断消息处理。

- 中断一个消息处理
  - 将 DMAE 位清零，以禁止 DMA 接口。
  - 等待直至当前 DMA 传输完成（等待 HASH_SR.DMAES=0）。请注意，块不一定已完全传输到散列。
  - 在 DMA 控制器中禁止对应的通道。
  - 等到散列处理器就绪（未在处理任何块），也就是等待 DINIS=1。
- 上下文保存阶段和上下文恢复阶段与以上相同（请参见软件加载数据的过程）。

  重新配置 DMA 控制器以使其传输消息的结尾。现在，通过将 DMAE 位置 1 便可从之前的中断点重新开始处理。

注意：如果在两个块之间进行上下文交换（最后一个块已完全处理并且下一个块尚未推入到 INFIFO，HASH_CR 寄存器中 NBW=000），则不必保存和恢复 HASH_CSR22 到 HASH_CSR37 寄存器。

32.2.7 散列中断

散列处理器可生成两个独立的可屏蔽中断源。这两个源连接到同一个中断向量。

通过更改 HASH_IMR 寄存器中的屏蔽位，可单独使能或禁止各个中断源。将相应的屏蔽位置 1 以使能中断。

可以从 HASH_SR 寄存器中读取各个中断源的状态。

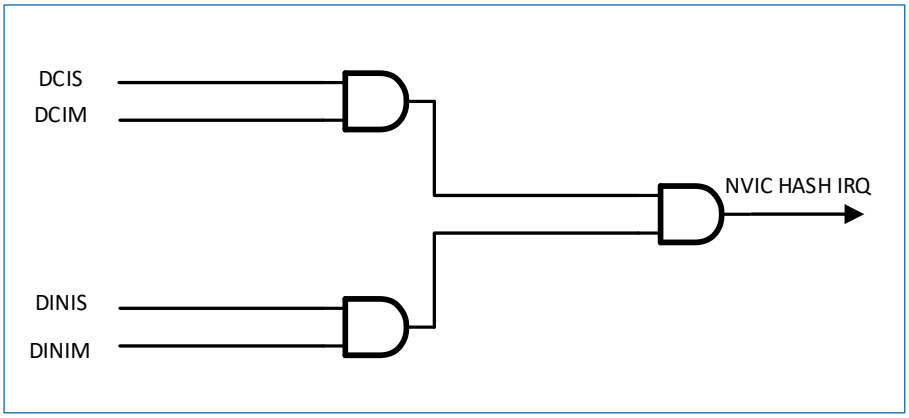


图 32-3 HASH 中断向量的生成

32.3 HASH 寄存器

基地址：0x5006 0400

空间大小：0x400

散列内核与多个控制和状态寄存器以及五个消息摘要寄存器相关联。所有这些寄存器都只能通过字来访问，否则会产生 AHB 错误。

32.3.1 HASH 控制寄存器（HASH_CR）

偏移地址：0x00

复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res													ALGO1	Res	
													rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res			DINNE	NBW[3:0]				ALGO0	MODE	DATATYPE[1:0]		DMAE	INIT	Res	
			r	r				rw	rw	rw		rw	w		

位 31:19	Res：保留 必须保持复位值。
位 18	ALGO1：算法选择 1（Algorithm selection bit1） ALGO1 和 ALGO0（位 7）配合使用，以选择 HASH 模式。 • 11：选择 HASH256 • 其他：保留 *注意：仅当 INIT 位置 1 并且 MODE=1 时此选择才有效。在计算期间更改该位不起作用。*
位 17:13	Res：保留 必须保持复位值。
位 12	DINNE：DIN 非空（DIN not empty） 当 HASH_DIN 寄存器包含有效数据时（也就是在至少写入一次之后），该位将置 1。当 INIT 位（初始化）或 DCAL 位（上一个消息处理已完成）写入到 1 时，会将该位清零。 • 0：数据输入缓冲器中不存在任何数据。 • 1：输入缓冲器至少包含一个字的数据，在异步协议里这一位使能和禁止 FSMC 使用等待信号。

位 11:8	NBW[3:0]：已推入的字数（Number of words already pushed）
	该位域反映了消息中已推入到 INFIFO 的字数。
	如果对 HASH_DIN 寄存器执行了写访问且 DINNE=1，则 NBW 递增（+1）。当 INIT 位写入 1 或者当摘要计算启动时（DCAL 写入到 1 或者 DMA 传输结束），NBW 将变为 0000。
	• 如果未使用 DMA：
	  ○ 0000 且 DINNE=0：没有任何字推入到 DIN 缓冲器（缓冲器为空，HASH_DIN 寄存器和 INFIFO 均为空）。
	  ○ 0000 且 DINNE=1：1 个字已推入到 DIN 缓冲器（HASH_DIN 寄存器包含 1 个字，INFIFO 为空）。
	  ○ 0001：2 个字已推入到 DIN 缓冲器（HASH_DIN 寄存器和 INFIFO 分别包含 1 个字）。
	  ○ ...
	  ○ 1111：16 个字已推入到 DIN 缓冲器。
	• 如果使用了 DMA，则 NBW 刚好是已推入到 INFIFO 的字数。

位 7	ALGO0：算法选择 0（Algorithm selection bit 0） 该位和 ALGO1 配合使用，请参见 ALGO1 的用法。
位 6	MODE：模式选择（Mode selection） 该位针对所选算法选择散列。

	• 0：选择散列模式 • 1：保留 *注意：仅当 INIT 置 1 时，此选择才有效。在计算期间更改该位不起作用。*
位 5:4	DATATYPE[1:0]：数据类型选择（Data type selection） 该位域定义了在 HASH_DIN 寄存器中输入的数据格式。 • 00：32 位数据 • 写入到 HASH_DIN 的数据将由散列处理直接使用，不会重新排序。 • 01：16 位数据或半字 • 写入到 HASH_DIN 的数据被视为 2 个半字，并且在交换之后由散列处理使用。 • 10：8 位数据或字节 • 写入到 HASH_DIN 的数据被视为 4 个字节，并且在交换之后由散列处理使用。 • 11：位数据或位串 • 写入到 HASH_DIN 的数据被视为 32 位（位串的第一位处于位置 0），并且在交换之后由散列处理使用（位串的第一位处于位置 31）。
位 3	DMAE：DMA 使能（DMA enable） • 0：禁止 DMA 传输。 • 1：使能 DMA 传输。散列内核可以接收数据时发送 DMA 请求。 *注意：* *1、当 DMA 发出 DMA 端子计数信号时（传输消息的最后一个数据时），将由硬件将该位清零。* *当 INIT 位写入到 1 时，不会将该位清零。* *p2、如果该位写入到 0，而已经向 DMA 请求了 DMA 传输，则会将 DMAE 清零但是不会中止当前传输。DMA 接口继续在内部使能，直到传输完成或 INIT 写入到 1。*
位 2	INIT：初始化消息摘要计算（Initialize message digest calculation） 将该位写入到 1 将会复位散列处理器内核，以使散列可以计算新消息的消息摘要。 若在该位中写入 0 则不会产生影响。读取该位将始终返回 0。
位 1:0	Res：保留 必须保持复位值。

32.3.2 HASH 数据输入寄存器（HASH_DIN）

偏移地址：0x04

复位值：0x00000000

HASH_DIN 是数据输入寄存器，其宽度为 32 位。它用于以 512 位的块来输入消息。写入 HASH_DIN 寄存器时，在 AHB 数据总线上呈现的值将被“推入”散列内核，并且寄存器获取 AHB 数据总线上呈现的新值。必须之前已在 HASH_CR 寄存器中配置了 DATATYPE 位才能获取正确的消息表示法。

16 字的块写入到 HASH_DIN 寄存器后，将启动中间摘要计算。有两种启动方式：

- 如果未使用 DMA，将新数据写入到 HASH_DIN 寄存器来启动，即写入下一个块的第一个字（中间摘要计算）。
- 如果使用了 DMA，则自动启动。
- 最后一个块写入到 HASH_DIN 寄存器之后，将启动最终摘要计算（包括填充）。有两种启动方式：
- 通过在 HASH_STR 寄存器中将 DCAL 位写入 1 来启动（最终摘要计算）。
- 如果使用了 DMA 并且 MDMAT 位设置为“0”，则自动启动。

当摘要计算（中间或最终）正在进行时，将扩展对 HASH_DIN 寄存器的所有新的写访问（通过在 AHB 总线上插入等待状态），直至散列计算完成。

读取 HASH_DIN 寄存器时，将访问该位置中写入的最后一个字（复位后为零）。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAIN[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAIN[15:0]															
rw															

位 31:0	DATAIN[31:0]：数据输入（Data input） • 读取该寄存器：返回寄存器的当前内容。 • 写入该寄存器：当前寄存器内容推入到 INFIFO，并且寄存器获取在 AHB 数据总线上呈现的新值。

32.3.3 HASH 启动寄存器（HASH_STR）

偏移地址：0x08

复位值：0x00000000

HASH_STR 寄存器有两个功能：

- 用于定义散列处理器中输入的消息的最后一个字的有效位数（也就是写入到 HASH_DIN 寄存器的最后一个数据中有效的最低有效位的数量）。
- 通过将 DCAL 位写入到 1，启动对消息中最后一个块的处理。

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res							DCAL	Res			NBLW[4:0]				
							r				rw				

位 31:9	Res：保留 必须保持复位值。
位 8	DCAL：摘要计算（Digest calculation） 将该位写 1 将开始使用先前写入的 NBLW 值来填充消息，并且由于 INIT 位已被写 1，因此将使用写入到 INFIFO 的所有数据字开始计算最终消息摘要。读取该位将返回 0。
位 7:5	Res：保留 必须保持复位值。
位 4:0	NBLW[4:0]：在散列处理器的位串结构中，消息的最后一个字中的有效位数（Number of last words） 如果写入该位域并且 DCAL 为“0”，则获取 AHB 数据总线上的值： • 0x00：对于在散列处理器的位串结构中写入的最后一个数据，所有 32 位（在数据交换后）均有效。 • 0x01：对于在散列处理器的位串结构中写入的最后一个数据，仅位[31]（在数据交换后）有效。 • 0x02：对于在散列处理器的位串结构中写入的最后一个数据，仅位[31:30]（在数据交换后）有效。 • 0x03：对于在散列处理器的位串结构中写入的最后一个数据，仅位[31:29]（在数据交换后）有效。 • ... • 0x1F：对于在散列处理器的位串结构中写入的最后一个数据，仅位[0]（在数据交换后）有效。

	如果当 DCAL 为“1”时写入该位域，则位字段不会更改。读取该位域将返回写入到 NBLW 的最后一个值。 *注意：必须在设置 DCAL 位之前配置该位域，否则该位域不会有效。特别请注意，配置 NBLW 和设置 DCAL 不能同时进行。*

32.3.4 HASH 摘要寄存器

这些寄存器包含消息摘要结果，其命名方式为 H0 到 H7。

如果当散列内核在计算中间摘要或最终消息摘要时（也就是在 DCAL 位已经写入到 1 时），对这些寄存器之一进行读访问，则读取将被拖延直到散列计算完成。

注意：

1.H0、H1、H2、H3 和 H4 映射到两个区域。

2.当开始对一个新的位做摘要计算（INIT=1 可以实现），这些寄存器为复位值。

3.当启动新位流的摘要计算时（通过将 INIT 位写入到 1），这些寄存器（HASH_HR0-7）将呈现其复位值。

32.3.4.1 HASH 摘要寄存器 0（HASH_HR0）

偏移地址：0x0C 和 0x310

复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H0[31:16]															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
H0[15:0]															
r															

位 31:0	H0[31:0]：消息摘要结果（Message digest result） 用于存放消息摘要结果。

32.3.4.2 HASH 摘要寄存器 1（HASH_HR1）

偏移地址：0x10 和 0x314

复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H1[31:16]															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
H1[15:0]															
r															

位 31:0	H1[31:0]：消息摘要结果（Message digest result） 用于存放消息摘要结果。

32.3.4.3 HASH 摘要寄存器 2（HASH_HR2）

偏移地址：0x14 和 0x318

复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H2[31:16]															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
H2[15:0]															
r															

位 31:0	H2[31:0]：消息摘要结果（Message digest result） 用于存放消息摘要结果。

32.3.4.4 HASH 摘要寄存器 3（HASH_HR3）

偏移地址：0x18 和 0x31C

复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H3[31:16]															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
H3[15:0]															
r															

位 31:0	H3[31:0]：消息摘要结果（Message digest result） 用于存放消息摘要结果。

32.3.4.5 HASH 摘要寄存器 4（HASH_HR4）

偏移地址：0x1C 和 0x320

复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H4[31:16]															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
H4[15:0]															
r															

位 31:0	H4[31:0]：消息摘要结果（Message digest result） 用于存放消息摘要结果。

32.3.4.6 HASH 摘要寄存器 5（HASH_HR5）

偏移地址：0x324

复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H5[31:16]															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
H5[15:0]															
r															

位 31:0	H5[31:0]：消息摘要结果（Message digest result） 用于存放消息摘要结果。

32.3.4.7 HASH 摘要寄存器 6（HASH_HR6）

偏移地址：0x328

复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H6[31:16]															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
H6[15:0]															
r															

位 31:0	H6[31:0]：消息摘要结果（Message digest result） 用于存放消息摘要结果。

32.3.4.8 HASH 摘要寄存器 7（HASH_HR7）

偏移地址：0x32C

复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H7[31:16]															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
H7[15:0]															
r															

位 31:0	H7[31:0]：消息摘要结果（Message digest result） 用于存放消息摘要结果。

32.3.5 HASH 中断使能寄存器（HASH_IMR）

偏移地址：0x20

复位值：0x00000000

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

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

位 31:2	Res：保留 必须保持复位值。
位 1	DCIE：摘要计算完成中断使能（Digest calculation completion interrupt enable） • 0：禁止摘要计算完成中断。 • 1：使能摘要计算完成中断。

位 0	DINIE：数据输入中断使能（Data input interrupt enable） • 0：禁止数据输入中断。 • 1：使能数据输入中断。

32.3.6 HASH 状态寄存器（HASH_SR）

偏移地址：0x24

复位值：0x00000001

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res												BUSY	DMAS	DCIS	DINIS
												r	r	rc_w0	rc_w0

位 31:4	Res：保留 必须保持复位值。
位 3	BUSY：忙碌位（Busy bit） • 0：当前未处理任何块。 • 1：散列内核正在处理某个数据块。
位 2	DMAS：DMA 状态（DMA status） 该位提供有关 DMA 接口活动的信息。该位与 DMAE 一起设置，在 DMAE=0 且未在进行任何 DMA 传输时，该位将被清零。没有任何中断与该位相关联。 • 0：禁止 DMA 接口（DMAE=0）并且未在进行任何传输。 • 1：使能 DMA 接口（DMAE=1）或者某个传输在进行。
位 1	DCIS：摘要计算完成中断状态（Digest calculation completion interrupt status） 该位是在摘要就绪时（也就是整个消息已处理完毕时），由硬件设置。要将该位清零可写入 0，或者向 HASH_CR.INIT 位写入 1。 • 0：HASH_HRx 寄存器中无任何摘要。 • 1：摘要计算已完成，摘要存储于 HASH_HRx 寄存器中。如果 HASH_IMR 寄存器中将 DCIE 位置 1，则产生中断。
位 0	DINIS：数据输入中断状态（Data input interrupt status） 该位是在输入缓冲器准备好获取新块时（16 个位置空闲），由硬件置 1。将该位清零则写入 0 或者写 HASH_DIN 寄存器。 • 0：输入缓冲器中的空闲位置少于 16 个。 • 1：可以将一个新块输入到输入缓冲器中。 如果 HASH_IMR 寄存器中将 DINIE 位置 1，则产生中断。

32.3.7 HASH 上下文交换寄存器（HASH_CSRx）（x=0..37）

偏移地址：0x0F8+x*0x4

复位值：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HASH_CSR[31:16]															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HASH_CSR[15:0]															
r															

位 31:0	HASH_CSR[31:0]：上下文交换（HASH context swap） 这些寄存器包含散列处理器的完整内部寄存器状态。如果高优先级的任务需要使用散列处理器，而该处理器正在执行其他任务，则必须使用上述寄存器进行上下文交换。 发生此类事件时，必须读取 HASH_CSRx 寄存器，并且读取值必须保存到系统存储器空间中。然后有优先权的任务便可以使用散列处理器，在散列计算完成时，可以从存储器中读取保存的上下文并回写到 HASH_CSRx 寄存器中。

33 真随机数发生器（TRNG）

TRNG 是一个以连续模拟噪声为基础的随机数发生器，在主机读数时提供一个 32 位的随机数。

TRNG 已通过 FIPS PUB 140-2 测试，成功率达 99%。

33.1 TRNG 主要特性

- 提供由模拟发生器产生的 32 位随机数。
- 两个连续随机数的间隔为 40 个 TRNG_CLK 时钟信号周期。
- 通过监察 TRNG 熵来标识异常行为（产生稳定值，或产生稳定的值序列）。
- 可禁止 TRNG 模块以降低功耗。
- TRNG 时钟为 AHB 时钟。
- 使能 TRNG 后，需要等待大约 1500 个时钟周期，再读取 TRNG_DR 寄存器。

33.2 TRNG 功能说明

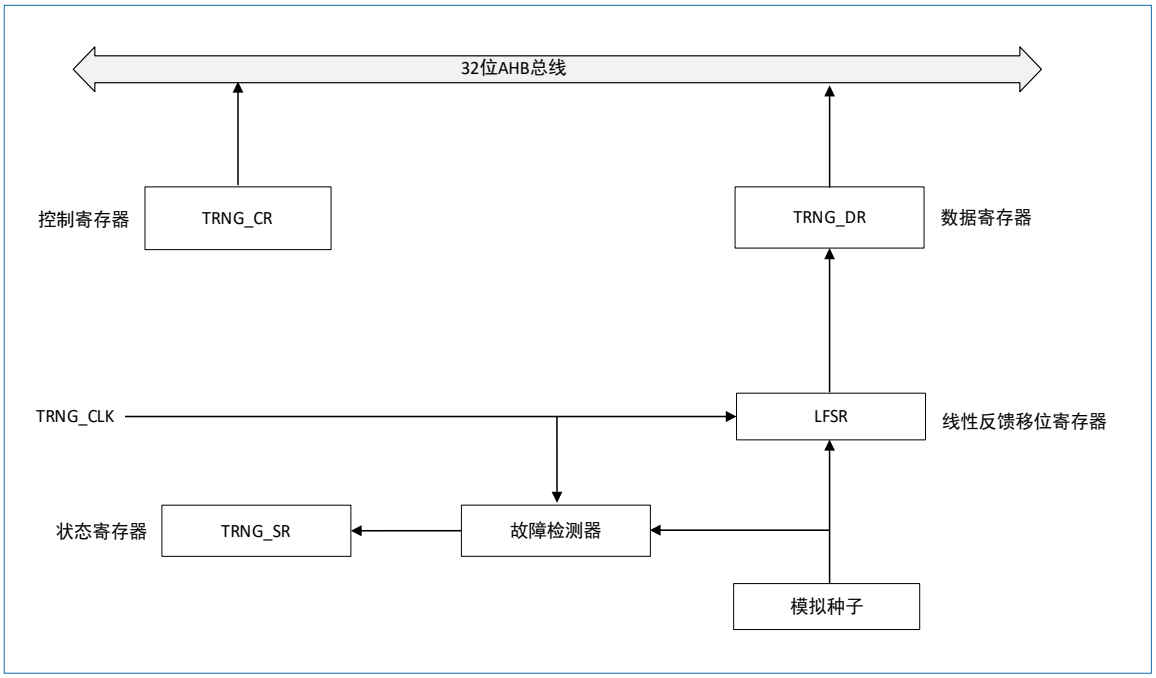


图 33-1 TRNG 框图

上图说明：LFSR 为寄存器为内部寄存器，用户不可见。

随机数发生器采用模拟电路实现。此电路产生馈入线性反馈移位寄存器（LFSR，此寄存器为内部寄存器，用户不可见）的种子，用于生成 32 位随机数。

该模拟电路由几个环形振荡器组成，振荡器的输出进行异或运算以产生种子。产生的种子（1 位随机序列）输出到 LFSR 移位寄存器，进行运算，生成 32 位随机数，详见图 33-2。

LFSR 由专用时钟（TRNG_CLK）按恒定频率提供时钟信息，因此随机数质量与 HCLK 频率无关。当将大量种子引入 LFSR 后，LFSR 的内容会传入数据寄存器（TRNG_DR）。

同时，系统会监视模拟种子和专用时钟 TRNG_CLK。状态位（TRNG_SR 寄存器中）指示何时在种子上出现异常序列，或何时 TRNG_CLK 时钟频率过低。

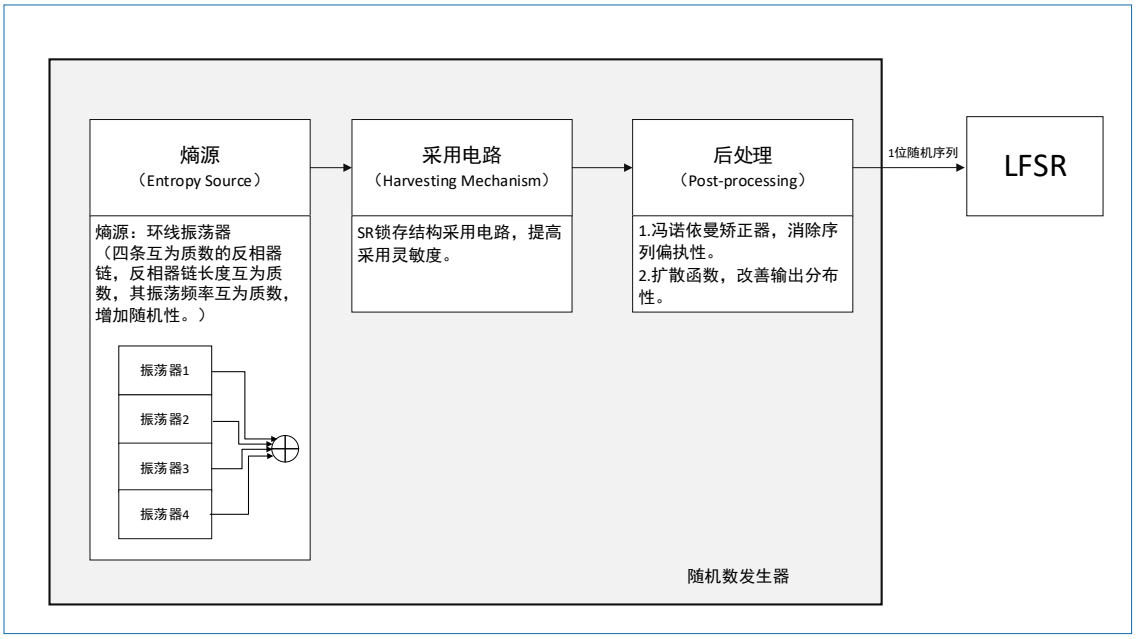


图 33-2 模拟种子的产生示意图

33.3 TRNG 的运行

要运行 TRNG，请按以下步骤操作：

1. 通过将 TRNG_CR.RNGEN 位置 1，使能随机数产生。这会激活模拟部分、LFSR 和错误检测器。

2. 检查随机数已准备就绪（TRNG_SR.DRDY 位为 1）。然后即可读取 TRNG_DR 寄存器中的内容。

按照 FIPSPUB（联邦信息处理标准出版物）140-2 的要求，将 RNGEN 位置 1 后产生的第一个随机数不应使用，但应保存起来，与产生的下一个随机数进行比较。随后产生的每个随机数都需要与产生的上一个随机数进行比较。如果任何一对进行比较的数字相等，则测试失败（连续随机数发生器测试）。

33.4 种子错误管理

出现种子错误时，只要 SECS 位为 1，就会中断随机数产生。此时 TRNG_DR 寄存器中有可用随机数，但不能使用，因为它可能没有足够的熵。应执行以下操作：将 RNGEN 位清零并置 1，以便重新初始化和重新启动 TRNG。

33.5 TRNG 寄存器

基地址：0x5006 0800

空间大小：0x400

TRNG 与控制寄存器、数据寄存器和状态寄存器相关联。必须通过字（32 位）进行访问。

33.5.1 TRNG 控制寄存器（TRNG_CR）

偏移地址：0x00

复位值：0x00000000

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

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

位 31:3	Res：保留 必须保持复位值。
位 2	RNGEN：随机数发生器使能（Random number generator enable） • 0：禁止随机数发生器，模拟随机源电源和逻辑时钟关闭。 • 1：使能随机数发生器。
位 1:0	Res：保留 必须保持复位值。

33.5.2 TRNG 状态寄存器（TRNG_SR）

偏移地址：0x04

复位值：0x00000000

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

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

位 31:3	Res：保留 必须保持复位值。

位 2	SECS：种子错误当前状态（Seed error current status）
	• 0：目前未检测到错误序列。
	• 1：检测到以下错误序列之一。
	  ○ 超过 64 个连续位具有相同值（0 或 1）。
	  ○ 超过 32 个连续交替的 0 和 1（0101010101...01）。

位 1	Res：保留 必须保持复位值。
位 0	DRDY：数据就绪（Data ready） • 0：TRNG_DR 寄存器尚未有效，无可用随机数据。 • 1：TRNG_DR 寄存器包含有效随机数据。 *说明：读取 TRNG_DR 寄存器后，该位恢复到 0，直到计算出新的有效值。*

33.5.3 TRNG 数据寄存器（TRNG_DR）

偏移地址：0x08

复位值：0x00000000

TRNG_DR 寄存器是只读寄存器，提供 32 位随机数值供读取。读取后，此寄存器在最多 40 个 TRNG_CLK 时钟周期后，提供新的随机数值。在读取 RNDATA 值之前，软件必须检查 DRDY 位是否置 1。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RNDATA[31:16]															
r															

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

RNDATA[15:0]
r

位 31:0	RNDATA[31:0]：随机数据（32 位）（Random data） DRDY=1 时，更新该数据。

34 算术运算协处理器（COALU）

该协处理器（COALU）在原有 ARM Cortex-M3 运算能力的基础上，能进一步提高芯片的运算能力，使芯片能适用更多的运算场景。该 COALU 在 ARM Cortex-M3 指令的基础上扩充了定点运算和增加了浮点运算。

34.1 COALU 特性

- 定点运算：
  - 64/32 位开方运算
  - 64/32 位除法运算
  - 饱和运算
    - 乘累加饱和运算
    - 32 位饱和加减运算
    - SIMD 饱和加减运算
    - 数据饱和化
  - SIMD 加减运算
  - 加减取半运算
  - 扩展加减运算
  - 乘累加运算和 SIMD 乘累加运算
- 浮点运算
  - 浮点加法运算
  - 浮点减法运算
  - 浮点乘法运算
  - 浮点除法运算
  - 浮点开方运算
  - 浮点乘累加运算
  - 浮点数定点数相互转换运算（包括 32 位定点数和浮点数相互转换、64 位定点数和浮点数相互转换）

34.2 运算指令

表 34-1 运算指令代码及运行周期（定点运算）

运算类型	ISA_CODE[9:0]	指令	描述	可能更新的标志位	运行周期数
定点开方	0001_000000	SQRT	32 为定点数开方运算：RSLT0=sqrt(OP0)	-	2~17
	0001_000001	LSQRT	64 位定点数开方运算：RSLT0=sqrt(OP1, OP0)	-	2~33
64 位定点除法	0010_000000	SLDIV	带符号数除法，64 位被除数，32 位除数：{RSLT1,RSLT0}={OP3,OP2}/OP0; RSLT2={OP3,OP2}%OP0	SR.DZINT/ SR.DIVOFINT	2~33
	0010_000001	ULDIV	无符号数除法，64 位被除数，32 位除数：	SR.DZINT/	2~33

运算类型	ISA_CODE[9:0]	指令	描述	可能更新的标志位	运行周期数
			{RSLT1,RSLT0}={OP3,OP2}/OP0; RSLT2={OP3,OP2}%OP0	SR.DIVOFINT	
SIMD 加减	0011_000000	SADD16	把操作数分解成两个 16 位带符号数，然后相加： RSLT0[31:16]=OP1[31:16]+OP0[31:16], RSLT0[15:0]=OP1[15:0]+OP0[15:0]	本指令可能会更新 SIMDFR 状态寄存器除 Q0~Q3 的其他位。	1
	0011_000001	SADD8	把操作数分解成 4 个 8 位带符号数，然后相加： RSLT0[31:24]=OP1[31:24]+OP0[31:24], RSLT0[23:16]=OP1[23:16]+OP0[23:16], RSLT0[15:8]=OP1[15:8]+OP0[15:8], RSLT0[7:0]=OP1[7:0]+OP0[7:0]	本指令可能会更新 SIMDFR 状态寄存器除 Q0~Q3 的其他位。	1
	0011_000010	SSUB16	把操作数分解成两个 16 位带符号数，然后相减： RSLT0[31:16]=OP1[31:16]-OP0[31:16], RSLT0[15:0]=OP1[15:0]-OP0[15:0]	本指令可能会更新 SIMDFR 状态寄存器中除 Q0~Q3 的其他位。	1
	0011_000011	SSUB8	把操作数分解成 4 个 8 位带符号数，然后相减： RSLT0[31:24]=OP1[31:24]-OP0[31:24, RSLT0[23:16]=OP1[23:16]-OP0[23:16], RSLT0[15:8]=OP1[15:8]-OP0[15:8], RSLT0[7:0]=OP1[7:0]-OP0[7:0];	本指令可能会更新 SIMDFR 状态寄存器中除 Q0~Q3 的其他位。	1
	0011_000100	SASX	把操作数分解成两个 16 位带符号数，然后错位加减： RSLT0[31:16]=OP1[31:16]+OP0[15:0], RSLT0[15:0]=OP1[15:0]-OP0[31:16];	本指令可能会更新 SIMDFR 状态寄存器中除 Q0~Q3 的其他位。	1
	0011_000101	SSAX	把操作数分解成两个 16 位带符号数，然后错位减加： RSLT0[31:16]=OP1[31:16]-OP0[15:0], RSLT0[15:0]=OP1[15:0]+OP0[31:16];	本指令可能会更新 SIMDFR 状态寄存器中除 Q0~Q3 的其他位。	1
	0011_000110	UADD16	把操作数分解成两个 16 位无符号数，然后相加： RSLT0[31:16]=OP1[31:16]+OP0[31:16], RSLT0[15:0]=OP1[15:0]+OP0[15:0];	本指令可能会更新 SIMDFR 状态寄存器中除 Q0~Q3 的其他位。	1
	0011_000111	UADD8	把操作数分解成 4 个 8 位无符号数，然后相加： RSLT0[31:24]=OP1[31:24]+OP0[31:24], RSLT0[23:16]=OP1[23:16]+OP0[23:16], RSLT0[15:8]=OP1[15:8]+OP0[15:8], RSLT0[7:0]=OP1[7:0]+OP0[7:0];	本指令可能会更新 SIMDFR 状态寄存器中除 Q0~Q3 的其他位。	1
	0011_001000	USUB16	把操作数分解成两个 16 位无符号数，然后	本指令可能会	1

运算类型	ISA_CODE[9:0]	指令	描述	可能更新的标志位	运行周期数
			相减： RSLT0[31:16]=OP1[31:16]-OP0[31:16], RSLT0[15:0]=OP1[15:0]-OP0[15:0]	更新 SIMDFR 状态寄存器中除 Q0~Q3 的其他位。	
	0011_001001	USUB8	把操作数分解成 4 个 8 位无符号数，然后相减： RSLT0[31:24]=OP1[31:24]-OP0[31:24], RSLT0[23:16]=OP1[23:16]-OP0[23:16], RSLT0[15:8]=OP1[15:8]-OP0[15:8], RSLT0[7:0]=OP1[7:0]-OP0[7:0]	本指令可能会更新 SIMDFR 状态寄存器中除 Q0~Q3 的其他位。	1
	0011_001010	UASX	把操作数分解成两个 16 位无符号数，然后错位加减： RSLT0[31:16]=OP1[31:16]+OP0[15:0], RSLT0[15:0]=OP1[15:0]-OP0[31:16]	本指令可能会更新 SIMDFR 状态寄存器中除 Q0~Q3 的其他位。	1
	0011_001011	USAX	把操作数分解成两个 16 位无符号数，然后错位减加： RSLT0[31:16]=OP1[31:16]-OP0[15:0], RSLT0[15:0]=OP1[15:0]+OP0[31:16];	本指令可能会更新 SIMDFR 状态寄存器中除 Q0~Q3 的其他位。	1
SIMD 饱和加减运算	0100_000000	QADD16	把操作数分解成两个 16 位带符号数饱和加： RSLT0[31:16]=OP1[31:16]+OP0[31:16], RSLT0[15:0]=OP1[15:0]+OP0[15:0];	本指令可能会更新 SIMDFR 状态寄存器。	1
	0100_000001	QADD8	把操作数分解成 4 个 8 位带符号数饱和加： RSLT0[31:24]=OP1[31:24]+OP0[31:24], RSLT0[23:16]=OP1[23:16]+OP0[23:16], RSLT0[15:8]=OP1[15:8]+OP0[15:8], RSLT0[7:0]=OP1[7:0]+OP0[7:0]	本指令可能会更新 SIMDFR 状态寄存器。	1
	0100_000010	QSUB16	把操作数分解成两个 16 位带符号数饱和减： RSLT0[31:16]=OP1[31:16]-OP0[31:16], RSLT0[15:0]=OP1[15:0]-OP0[15:0]	本指令可能会更新 SIMDFR 状态寄存器。	1
	0100_000011	QSUB8	把操作数分解成 4 个 8 位带符号数饱和减： RSLT0[31:24]=OP1[31:24]-OP0[31:24], RSLT0[23:16]=OP1[23:16]-OP0[23:16], RSLT0[15:8]=OP1[15:8]-OP0[15:8], RSLT0[7:0]=OP1[7:0]-OP0[7:0]	本指令可能会更新 SIMDFR 状态寄存器。	1
	0100_000100	QASX	把操作数分解成两个 16 位带符号数，错位饱和加减： RSLT0[31:16]=OP1[31:16]+OP0[15:0], RSLT0[15:0]=OP1[15:0]-OP0[31:16]	本指令可能会更新 SIMDFR 状态寄存器。	1
	0100_000101	QSAX	把操作数分解成两个 16 位带符号数，错位	本指令可能会	1

运算类型	ISA_CODE[9:0]	指令	描述	可能更新的标志位	运行周期数
			饱和减加： RSLT0[31:16]=OP1[31:16]-OP0[15:0], RSLT0[15:0]=OP1[15:0]+OP0[31:16];	更新 SIMDFR 状态寄存器。	
	0100_000110	UQADD16	把操作数分解成两个 16 位无符号数饱和加： RSLT0[31:16]=OP1[31:16]+OP0[31:16], RSLT0[15:0]=OP1[15:0]+OP0[15:0];	本指令可能会更新 SIMDFR 状态寄存器。	1
	0100_000111	UQADD8	把操作数分解成 4 个 8 位无符号数饱和加： RSLT0[31:24]=OP1[31:24]+OP0[31:24], RSLT0[23:16]=OP1[23:16]+OP0[23:16], RSLT0[15:8]=OP1[15:8]+OP0[15:8], RSLT0[7:0]=OP1[7:0]+OP0[7:0];	本指令可能会更新 SIMDFR 状态寄存器。	1
	0100_001000	UQSUB16	把操作数分解成两个 16 位无符号数饱和减： RSLT0[31:16]=OP1[31:16]-OP0[31:16], RSLT0[15:0]=OP1[15:0]-OP0[15:0];	本指令可能会更新 SIMDFR 状态寄存器。	1
	0100_001001	UQSUB8	把操作数分解成 4 个 8 位无符号数饱和减： RSLT0[31:24]=OP1[31:24]-OP0[31:24], RSLT0[23:16]=OP1[23:16]-OP0[23:16], RSLT0[15:8]=OP1[15:8]-OP0[15:8], RSLT0[7:0]=OP1[7:0]-OP0[7:0];	本指令可能会更新 SIMDFR 状态寄存器。	1
	0100_001010	UQASX	把操作数分解成两个 16 位无符号数，错位饱和加减： RSLT0[31:16]=OP1[31:16]+OP0[15:0], RSLT0[15:0]=OP1[15:0]-OP0[31:16];	本指令可能会更新 SIMDFR 状态寄存器。	1
	0100_001011	UQSAX	把操作数分解成两个 16 位无符号数，错位饱和减加： RSLT0[31:16]=OP1[31:16]-OP0[15:0], RSLT0[15:0]=OP1[15:0]+OP0[31:16];	本指令可能会更新 SIMDFR 状态寄存器。	1
32 位加减饱和运算	0101_000001	QADD	32 位带符号数饱和加 RSLT0=OP1[31:0]+OP0[31:0]	FR.Q/FR.V/FR.N / SR.QFINT/ SR.OFINT	1
	0101_000010	QSUB	32 位带符号数饱和减 RSLT0=OP1[31:0]-OP0[31:0]	FR.Q/FR.V/FR.N / SR.QFINT/ SR.OFINT	1
	0101_000011	QDADD	32 位被加数乘 2 并饱和化后和加数带符号饱和加 RSLT0=OP1[31:0]*2+OP0[31:0]	FR.Q/FR.V/FR.N / SR.QFINT/ SR.OFINT	1
	0101_000100	QDSUB	32 位减数乘 2 并饱和化后和被减数做带符号饱和减法 RSLT0=OP1[31:0]-OP0[31:0]*2	FR.Q/FR.V/FR.N / SR.QFINT/ SR.OFINT	1

运算类型	ISA_CODE[9:0]	指令	描述	可能更新的标志位	运行周期数
数据饱和变换	0110_000001	SSAT16	把操作数分解成两个 16 位带符号数，然后饱和到设定位： RSLT0[31:16]=OP0[31:16]饱和到 OP1 设定位 RSLT0[15:0]=OP0[15:0]饱和到 OP1 设定位	-	1
	0110_000010	USAT16	把操作数分解成两个 16 位无符号数，然后饱和到设定位： RSLT0[31:16]=OP0[31:16]饱和到 OP1 设定位 RSLT0[15:0]=OP0：w[15:0]饱和到 OP1 设定位	-	1
加减取半运算	0111_000000	SHADD16	把操作数分解成两个 16 位带符号数相加，然后右移 1 位： RSLT0[31:16]={OP1[31:16]+OP0[31:16]} >> 1, RSLT0[15:0]={OP1[15:0]+OP0[15:0]} >> 1;	本指令可能会更新 SIMDFR 状态寄存器中除 Q0~Q3 的其他位。	1
	0111_000001	SHADD8	把操作数分解成 4 个 8 位带符号数相加，然后右移 1 位： RSLT0[31:24]={OP1[31:24]+OP0[31:24]} >> 1, RSLT0[23:16]={OP1[23:16]+OP0[23:16]} >> 1, RSLT0[15:8]={OP1[15:8]+OP0[15:8]} >>1, RSLT0[7:0]={OP1[7:0]+OP0[7:0]} >> 1;	本指令可能会更新 SIMDFR 状态寄存器中除 Q0~Q3 的其他位。	1
	0111_000010	SHSUB16	把操作数分解成两个 16 位带符号数相减，然后右移 1 位： RSLT0[31:16]={OP1[31:16]-OP0[31:16]} >> 1, RSLT0[15:0]={OP1[15:0]-OP0[15:0]} >> 1	本指令可能会更新 SIMDFR 状态寄存器中除 Q0~Q3 的其他位。	1
	0111_000011	SHSUB8	把操作数分解成 4 个 8 位带符号数相减，然后右移 1 位： RSLT0[31:24]={OP1[31:24]-OP0[31:24]} >> 1, RSLT0[23:16]={OP1[23:16]-OP0[23:16]} >> 1, RSLT0[15:8]={OP1[15:8]-OP0[15:8]} >> 1, RSLT0[7:0]={OP1[7:0]-OP0[7:0]} >> 1;	本指令可能会更新 SIMDFR 状态寄存器中除 Q0~Q3 的其他位。	1
	0111_000100	SHASX	把操作数分解成两个 16 位带符号数错位加减，然后右移 1 位： RSLT0[31:16]={OP1[31:16]+OP0[15:0]} >> 1, RSLT0[15:0]={OP1[15:0]-OP0[31:16]} >> 1	本指令可能会更新 SIMDFR 状态寄存器中除 Q0~Q3 的其他位。	1
	0111_000101	SHSAX	把操作数分解成两个 16 位带符号数错位减加，然后右移 1 位： RSLT0[31:16]={OP1[31:16]-OP0[15:0]} >> 1, RSLT0[15:0]={OP1[15:0]+OP0[31:16]} >> 1	本指令可能会更新 SIMDFR 状态寄存器中除 Q0~Q3 的其他位。	1
	0111_000110	UHADD16	把操作数分解成两个 16 位无符号数相加，然后右移 1 位： RSLT0[31:16]={OP1[31:16]+OP0[31:16]} >> 1, RSLT0[15:0]={OP1[15:0]+OP0[15:0]} >> 1	本指令可能会更新 SIMDFR 状态寄存器中除 Q0~Q3 的其他位。	1

运算类型	ISA_CODE[9:0]	指令	描述	可能更新的标志位	运行周期数
	0111_000111	UHADD8	把操作数分解成 4 个 8 位无符号数相加，然后右移 1 位： RSLT0[31:24]={OP1[31:24]+OP0[31:24]} >> 1, RSLT0[23:16]={OP1[23:16]+OP0[23:16]} >> 1, RSLT0[15:8]={OP1[15:8]+OP0[15:8]} >>1, RSLT0[7:0]={OP1[7:0]+OP0[7:0]} >> 1	本指令可能会更新 SIMDFR 状态寄存器中除 Q0~Q3 的其他位。	1
	0111_001000	UHSUB16	把操作数分解成两个 16 位无符号数相减，然后右移 1 位： RSLT0[31:16]={OP1[31:16]-OP0[31:16]} >> 1, RSLT0[15:0]={OP1[15:0]-OP0[15:0]} >> 1	本指令可能会更新 SIMDFR 状态寄存器中除 Q0~Q3 的其他位。	1
	0111_001001	UHSUB8	把操作数分解成 4 个 8 位无符号数相减，然后右移 1 位： RSLT0[31:24]={OP1[31:24]-OP0[31:24]} >> 1, RSLT0[23:16]={OP1[23:16]-OP0[23:16]} >> 1, RSLT0[15:8]={OP1[15:8]-OP0[15:8]} >> 1, RSLT0[7:0]={OP1[7:0]-OP0[7:0]} >> 1	本指令可能会更新 SIMDFR 状态寄存器中除 Q0~Q3 的其他位。	1
	0111_001010	UHASX	把操作数分解成两个 16 位无符号数错位加减，然后右移 1 位： RSLT0[31:16]={OP1[31:16]+OP0[15:0]} >> 1, RSLT0[15:0]={OP1[15:0]-OP0[31:16]} >> 1	本指令可能会更新 SIMDFR 状态寄存器中除 Q0~Q3 的其他位。	1
	0111_001011	UHSAX	把操作数分解成两个 16 位无符号数错位减加，然后右移 1 位： RSLT0[31:16]={OP1[31:16]-OP0[15:0]} >> 1, RSLT0[15:0]={OP1[15:0]+OP0[31:16]} >> 1	本指令可能会更新 SIMDFR 状态寄存器中除 Q0~Q3 的其他位。	1
乘累加	1000_000000	SMLALBB	把被乘数和乘数的低 16 位做带符号乘法，然后和一个 64bit 带符号数累加。 {RSLT1, RSLT0} = OP1[15:0]*OP0[15:0]+{OP3,OP2} 可能会更新 FR.V、FR.N 和 SR.OFINT	FR.V/FR.N/ SR.OFINT	1
	1000_000001	SMLALBT	把被乘数的低 16 位和乘数的高 16 位做带符号乘法，然后和一个 64bit 带符号数累加。 {RSLT1, RSLT0}= OP1[15:0]*OP0[31:16]+{OP3,OP2}	FR.V/FR.N/ SR.OFINT	1
	1000_000010	SMLALTB	把被乘数的高 16 位和乘数的低 16 位做带符号乘法，然后和一个 64bit 带符号数累加。 {RSLT1, RSLT0}= OP1[31:16]*OP0[15:0]+{OP3,OP2}	FR.V/FR.N/ SR.OFINT	1
	1000_000011	SMLALTT	把被乘数的高 16 位和乘数的高 16 位做带符号乘法，然后和一个 64bit 带符号数累加。 {RSLT1, RSLT0}=	FR.V/FR.N/ SR.OFINT	1

运算类型	ISA_CODE[9:0]	指令	描述	可能更新的标志位	运行周期数
			OP1[31:16]*OP0[31:16]+{OP3,OP2}		
	1000_000100	SMLAW	把被乘数和乘数做带符号数乘法，然后累加。 RSLT0=OP1*OP0+OP2	FR.V/FR.N/ SR.OFINT	1
	1000_000101	SMLALD	把被乘数和乘数的高低 16 位分别做带符号乘法，然后和一个 64bit 带符号数累加。 {RSLT1, RSLT0}= OP1[31:16]*OP0[31:16]+OP1[15:0]*OP0[15:0]+{OP3,OP2}	FR.V/FR.N/ SR.OFINT	2
	1000_000110	SMLALDX	把被乘数和乘数的高低 16 位交叉做带符号乘法，然后和一个 64bit 带符号数累加。 {RSLT1, RSLT0} = OP1[31:16]*OP0[15:0]+OP1[15:0]*OP0[31:16]+{OP3,OP2}	FR.V/FR.N/ SR.OFINT	2
	1000_000111	SMLSLD	把被乘数和乘数的高低 16 位分别做带符号乘法，然后乘积相减，再和一个 64bit 带符号数累加。 {RSLT1,RSLT0}=OP1[15:0]*OP0[15:0]-OP1[31:16]*OP0[31:16]+{OP3,OP2}	FR.V/FR.N/ SR.OFINT	2
	1000_001000	SMLSLDX	把被乘数和乘数的高低 16 位交叉做带符号乘法，然后乘积相减，再和一个 64bit 带符号数累加。 {RSLT1,RSLT0}=OP1[15:0]*OP0[31:16]-OP1[31:16]*OP0[15:0]+{OP3,OP2}	FR.V/FR.N/ SR.OFINT	2
	1000_001001	SMMLA	把被乘数和乘数做带符号乘法，然后选取高 32 位结果，再和一个 32bit 带符号数相加。 RSLT1={OP1*OP0}[63:32]+OP3	FR.V/FR.N/ SR.OFINT	1
	1000_001010	SMMLAR	把被乘数和乘数做带符号乘法，然后选取高 32 位舍入后的结果，再和一个 32bit 带符号数累加。 RSLT1={OP1*OP0+32'h80000000}[63:32]+OP3	FR.V/FR.N/ SR.OFINT	1
	1000_001011	SMMLS	把被乘数和乘数做带符号乘法，然后选取高 32 位结果，再减去一个 32bit 带符号数。 RSLT1={OP1*OP0}[63:32]-OP3	FR.V/FR.N/ SR.OFINT	1
	1000_001100	SMMLSR	把被乘数和乘数做带符号乘法，然后选取高 32 位舍入后的结果，再减去一个 32bit 带符号数。 RSLT1={OP1*OP0+32'h80000000}[63:32]-OP3	FR.V/FR.N/ SR.OFINT	1
	1000_001101	UMAAL	把被乘数和乘数做 32 位带符号乘法，并与{OP3, OP2}相加得到 64 位结果。 {RSLT1,RSLT0}=OP1*OP0+{OP3,OP2}	FR.V/FR.N/ SR.OFINT	1

运算类型	ISA_CODE[9:0]	指令	描述	可能更新的标志位	运行周期数
	1000_001110	SMMAAA	把两组 32 位带符号被乘数和乘数相乘，然后和一个 32 位带符号数累加。 RSLT0=OP4*OP3+OP1*OP0+OP2	FR.V/FR.N/SR.OFINT	2
饱和乘累加	1001_000000	QSMLALBB	把被乘数和乘数的低 16 位做带符号乘法，然后和一个 64bit 带符号数累加；结果饱和为 64 位。 {RSLT1, RSLT0}= OP1[15:0]*OP0[15:0]+{OP3,OP2}	FR.Q/FR.V/FR.N/SR.QFINT/SR.OFINT	1
	1001_000001	QSMLALBT	把被乘数的低 16 位和乘数的高 16 位做带符号乘法，然后和一个 64bit 带符号数累加；结果饱和为 64 位。 {RSLT1, RSLT0}= OP1[15:0]*OP0[31:16]+{OP3,OP2}	FR.Q/FR.V/FR.N/SR.QFINT/SR.OFINT	1
	1001_000010	QSMLALTB	把被乘数的高 16 位和乘数的低 16 位做带符号乘法，然后和一个 64bit 带符号数累加；结果饱和为 64 位。 {RSLT1, RSLT0}= OP1[31:16]*OP0[15:0]+{OP3,OP2}	FR.Q/FR.V/FR.N/SR.QFINT/SR.OFINT	1
	1001_000011	QSMLALTT	把被乘数的高 16 位和乘数的高 16 位做带符号乘法，然后和一个 64bit 带符号数累加；结果饱和为 64 位 {RSLT1, RSLT0}= OP1[31:16]*OP0[31:16]+{OP3,OP2}	FR.Q/FR.V/FR.N/SR.QFINT/SR.OFINT	1
	1001_000100	QSMLAW	把被乘数和乘数做带符号数乘法，然后累加；结果饱和为 32 位 RSLT0=OP1*OP0+OP2	FR.Q/FR.V/FR.N/SR.QFINT/SR.OFINT	1
	1001_000101	QSMLALD	把被乘数和乘数的高低 16 位分别做带符号乘法，然后和一个 64bit 带符号数累加；结果饱和为 64 位 {RSLT1, RSLT0}= OP1[31:16]*OP0[31:16]+OP1[15:0]*OP0[15:0]+{OP3,OP2}	FR.Q/FR.V/FR.N/SR.QFINT/SR.OFINT	2
	1001_000110	QSMLALDX	把被乘数和乘数的高低 16 位交叉做带符号乘法，然后和一个 64bit 带符号数累加；结果饱和为 64 位。 {RSLT1, RSLT0}= OP1[31:16]*OP0[15:0]+OP1[15:0]*OP0[31:16]+{OP3,OP2}	FR.Q/FR.V/FR.N/SR.QFINT/SR.OFINT	2
	1001_000111	QSMLSLD	把被乘数和乘数的高低 16 位分别做带符号乘法，然后乘积相减，再和一个 64bit 带符号数累加；结果饱和为 64 位。 {RSLT1, RSLT0}= OP1[15:0]*OP0[15:0]- OP1[31:16]*OP0[31:16]+{OP3,OP2}	FR.Q/FR.V/FR.N/SR.QFINT/SR.OFINT	2
	1001_001000	QSMLSLD	把被乘数和乘数的高低 16 位交叉做带符号	FR.Q/FR.V/	2

运算类型	ISA_CODE[9:0]	指令	描述	可能更新的标志位	运行周期数
		X	乘法，然后乘积相减，再和一个 64bit 带符号数累加；结果饱和为 64 位。 {RSLT1, RSLT0}= OP1[15:0]*OP0[32:16]-OP1[31:16]*OP0[15:0]+{OP3,OP2}	FR.N/SR.QFINT/SR.OFINT	
	1001_001001	QSMMLA	把被乘数和乘数做带符号乘法，然后选取高 32 位结果，再和一个 32bit 带符号数相加；结果饱和为 32 位。 RSLT1={OP1*OP0}[63:32]+OP3	FR.Q/FR.V/FR.N/SR.QFINT/SR.OFINT	1
	1001_001010	QSMMLAR	把被乘数和乘数做带符号乘法，然后选取高 32 位舍入后的结果，再和一个 32bit 带符号数累加；结果饱和为 32 位。 RSLT1={OP1*OP0+32'h80000000}[63:32]+OP3	FR.Q/FR.V/FR.N/SR.QFINT/SR.OFINT	1
	1001_001011	QSMMLS	把被乘数和乘数做带符号乘法，然后选取高 32 位结果，再减去一个 32bit 带符号数；结果饱和为 32 位。 RSLT1={OP1*OP0}[63:32]-OP3	FR.Q/FR.V/FR.N/SR.QFINT/SR.OFINT	1
	1001_001100	QSMMLSR	把被乘数和乘数做带符号乘法，然后选取高 32 位舍入后的结果，再减去一个 32bit 带符号数；结果饱和为 32 位 RSLT1={OP1*OP0+32'h80000000}[63:32]-OP3	FR.Q/FR.V/FR.N/SR.QFINT/SR.OFINT	1
	1001_001101	QUMAAL	把被乘数和乘数做 32 位带符号乘法，并与{OP4, OP3}相加得到 64 位结果{RSLT1,RSLT0}=OP1*OP0+{OP3,OP2}	FR.Q/FR.V/FR.N/SR.QFINT/SR.OFINT	1
	1001_001110	QSMMAAA	把两组 32 位带符号被乘数和乘数相乘，然后和一个 32 位带符号数累加，乘法结果和累加结果都饱和化为 32 位 RSLT0=OP4*OP2+OP1*OP0+OP2	FR.Q/FR.V/FR.N/SR.QFINT/SR.OFINT	2

表 34-2 运算指令代码及运行周期（浮点运算）

运算类型	ISA_CODE[9:0]	指令	描述	标志位更新	运行周期数
浮点运算	1010_000000	VADD	浮点数加法： RSLT0=OP1+OP0	FR.FV/FR.FZ/FR.FN/SR.IOINT	1~2
	1010_000001	VSUB.F32	浮点数减法： RSLT0=OP1-OP0	FR.FV/FR.FZ/FR.FN/SR.IOINT	1~2
	1010_000010	VMUL.F32	浮点数乘法： RSLT0=OP2*OP1	FR.FV/FR.FZ/FR.FN/SR.IOINT	1~2
	1010_000011	VDIV.F32	浮点数除法： RSLT0=OP1/OP0	FR.FV/FR.FZ/FR.FN/SR.IOINT/SR.DZINT	14

运算类型	ISA_CODE[9:0]	指令	描述	标志位更新	运行周期数
	1010_000100	VSQRT.F32	浮点数开方： RSLT0=sqrt（OP0）	FR.FV/FR.FZ/SR.IOINT	15
	1010_000101	VLSQRT.F32	高精度浮点开方（把尾数补零扩展为 48 位后开方，提高指数为奇数的开方精度）： RSLT0=sqrt（OP0）	FR.FV/FR.FZ/SR.IOINT	26
	1010_001001	VCVTR.S32.F32	把浮点数转换为 32 位带符号定点数，舍入方式由 CR.RMODE[1:0]决定： RSLT0=OP0	FR.FV/FR.FZ/FR.FN/SR.IOINT	1~2
	1010_001010	VCVT.F32.S32	把 32 位带符号定点数转换为浮点数： RSLT0=OP0	FR.FV/FR.FZ	1~2
	1010_001111	VMLA.F32	浮点数乘累加，对乘法运算的结果进行舍入后再进行加法： RSLT0=OP2+OP1*OP0	FR.FV/FR.FZ/FR.FN/SR.IOINT	4
	1010_010000	VMLS.F32	浮点数乘减，对乘法运算的结果进行舍入后再进行减法： RSLT0=OP2-OP1*OP0	FR.FV/FR.FZ/FR.FN/SR.IOINT	4
	1010_010010	VNMLA.F32	浮点乘累加并取负数： RSLT0=-（OP2+OP1*OP0）	FR.FV/FR.FZ/FR.FN/SR.IOINT	4
	1010_010011	VNMLS.F32	浮点乘累减并取负数： RSLT0=-（OP2-OP1*OP0）	FR.FV/FR.FZ/FR.FN/SR.IOINT	4
	1010_010100	VNMUL	浮点乘法并取负数： RSLT0=-OP1*OP0	FR.FV/FR.FZ/FR.FN/SR.IOINT	1~2
	1010_010101	VCVTR.S64.F32	把浮点数转换为 64 位带符号定点数，舍入方式由 CR.RMODE[1:0]决定： {RSLT1, RSLT0}=OP0	FR.FV/FR.FZ/FR.FN/SR.IOINT	1~2
	1010_010110	VCVT.F32.S64	把 64 位带符号定点数转换为浮点数： RSLT0={OP1, OP0}	FR.FV/FR.FZ	1~2

34.3 COALU 寄存器

基地址：0x4003 0000

空间大小：0x400

所有寄存器只支持字访问。

注：COALU 中不同运算需要的操作数寄存器和结果寄存器数量各不相同，最多可能需要 5 个操作数寄存器、3 个结果寄存器。操作数寄存器分别标识为 OP0~OP5，结果寄存器分别标识为 RSLT0~RSLT3。

可以通过配置 COALU_LINK0 或 COALU_LINK1 寄存器把每个操作数寄存器和结果寄存器映射到任意一个通用寄存器。

在一次运算中选择 COALU_LINK0 或者是 COALU_LINK1 中的配置完成映射由 COALU_CR.LK 位选择。如果 COALU_CR.LK 为

0，选择 COALU_LINK0 中的配置完成映射，否则由 COALU_LINK1 中的配置完成映射。

34.3.1 控制寄存器（COALU_CR）

偏移地址：0x00

复位值：0x0000 0000

本寄存器控制 COALU 协处理器的具体操作。当 COALU_SR.BSY 为 1 时访问本寄存器会暂停（Stall）AHB 总线。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res			DIVOFIE	DZIE	IOIE	QFIE	OFIE	Res		RMODE[1:0]		Res		START	DFS
			rw	rw	rw	rw	rw			rw				w	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LK	FAMW	Res				ISA_CODE[9:0]									
rw	rw					rw									

位 31:29	Res：保留 必须保持复位值。
位 28	DIVOFIE：除法溢出中断使能（Divide overrun interrupt enable） • 0：无作用 • 1：使能除法溢出中断
位 27	DZIE：除数为 0 中断使能（Divisor zero interrupt enable） • 0：无作用 • 1：使能除 0 中断
位 26	IOIE：非法操作中断使能（Illegal operation interrupt enable） • 0：无作用 • 1：使能非法操作中断
位 25	QFIE：饱和异常中断使能（Saturation error interrupt enable） • 0：无作用 • 1：使能饱和异常中断
位 24	OFIE：溢出异常中断使能（Overrun error interrupt enable） • 0：无作用 • 1：使能溢出中断
位 23:22	Res：保留 必须保持复位值。
位 21:20	RMODE[1:0]：浮点运算进位控制（Carry mode） • 00：舍入到最近（RN）模式（如果 GRS>100，在 23 位尾数加 1；如果 GRS<100，丢弃；如果 GSR=100，则如果 23 位尾数的最低位为 1，加 1，如果 23 位尾数的最低位为 0，丢弃）。 • 01：舍入到正无穷（RP）模式（如果结果为正并且 GRS 不等于 0，尾数加 1）。 • 10：舍入到负无穷（RM）模式（如果结果为负并且 GRS 不等于 0，尾数加 1）。 • 11：舍入到零（RZ）模式（丢弃 GRS）。
位 19:18	Res：保留

	必须保持复位值。
位 17	START：运算开始（Start） 软件写 1 开始运算，在 DFS 为 1 时有用。 软件读时始终返回值 0。
位 16	DFS：禁止快速开始（Disable Fast Start） 默认情况下，软件往 OP0 写数后立即开始运算，当 DFS 为 1 后，需要软件往 START 写 1 后才开始运算。
位 15	LK：操作数连接寄存器选择信号（Link selection） • 0：当前指令选择 COALU_LINK0 中的配置映射通用寄存器。 • 1：当前指令选择 COALU_LINK1 中的配置映射通用寄存器。
位 14	FAMW：浮点数减法、浮点乘法运算、浮点定点数转换 wait 控制（Wait control） • 1：浮点数加减法、浮点乘法、浮点定点数转换运算在 2 个时钟周期内完成。 • 0：浮点数加减法、浮点乘法、浮点定点数转换运算在 1 个时钟周期内完成。 当 HCLK<60MHz 时可以配置 FAMW 为 0，否则需要配置 FAMW 为 1。
位 13:10	Res：保留 必须保持复位值。
位 9:0	ISA_CODE[9:0]：指令代码（Instruction code） 指明 COALU 完成何种运算操作。 详细的运用参“34.2 运算指令”。

34.3.2 操作数连接寄存器（COALU_LINKx）（x=0..1）

偏移地址：0x04+0x04*x

复位值：0x7654 3210; 0xBA98 7650

通过配置本寄存器可以把运算结果直接映射到操作数寄存器，作为下一次运算的输入操作数。当 COALU_SR.BSY 为 1 时访问本寄存器会暂停 AHB 总线。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSLT2_LK[3:0]				RSLT1_LK[3:0]				RSLT0_LK[3:0]				OP4_LK[3:0]			
rw				rw				rw				rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OP3_LK[3:0]				OP2_LK[3:0]				OP1_LK[3:0]				OP0_LK[3:0]			
rw				rw				rw				rw			

位 31:28	RSLT2_LK[3:0]：RSLT2 映射配置（RSLT2 remap configuration） • 0000：把 RSLT2 映射到 COALU_R0 • 0001：把 RSLT2 映射到 COALU_R1 • … • 1011：把 RSLT2 映射到 COALU_R11 • 其余：禁止使用
位 27:24	RSLT1_LK[3:0]：RSLT1 映射配置（RSLT1 remap configuration） • 0000：把 RSLT1 映射到 COALU_R0 • 0001：把 RSLT1 映射到 COALU_R1

	• … • 1011：把 RSLT1 映射到 COALU_R11 • 其余：禁止使用
位 23:20	RSLT0_LK[3:0]：RSLT0 映射配置（RSLT0 remap configuration） • 0000：把 RSLT0 映射到 COALU_R0 • 0001：把 RSLT0 映射到 COALU_R1 • … • 1011：把 RSLT0 映射到 COALU_R11 • 其余：禁止使用
位 19:16	OP4_LK[3:0]：OP4 映射配置（OP4 remap configuration） • 0000 ：把 OP4 映射到 COALU_R0 • 0001：把 OP4 映射到 COALU_R1 • … • 1011：把 OP4 映射到 COALU_R11 • 其余：禁止使用
位 15:12	OP3_LK[3:0]：OP3 映射配置（OP3 remap configuration） • 0000 ：把 OP3 映射到 COALU_R0 • 0001：把 OP3 映射到 COALU_R1 • … • 1011：把 OP3 映射到 COALU_R11 • 其余：禁止使用
位 11:8	OP2_LK[3:0]：OP2 映射配置（OP2 remap configuration） • 0000 ：把 OP2 映射到 COALU_R0 • 0001：把 OP2 映射到 COALU_R1 • … • 1011：把 OP2 映射到 COALU_R11 • 其余：禁止使用
位 7:4	OP1_LK[3:0]：OP1 映射配置（OP1 remap configuration） • 0000 ：把 OP1 映射到 COALU_R0 • 0001：把 OP1 映射到 COALU_R1 • … • 1011： 把 OP1 映射到 COALU_R11 • 其余：禁止使用
位 3:0	OP0_LK[3:0]：OP0 映射配置（OP0 remap configuration） • 0000 ：把 OP0 映射到 COALU_R0 • 0001：把 OP0 映射到 COALU_R1 • … • 1011：把 OP0 映射到 COALU_R11 • 其余：禁止使用

注：可以把一个通用寄存器同时映射到多个 OP 寄存器和 RSLT 寄存器。

比如执行指令：RSLT0=OP1*OP0+OP2，如果把 R0 映射到 OP2，OP1 和 RSLT0，把 R1 映射到 OP0，则执行指令后 R0=R0*R1+R0。

34.3.3 状态寄存器（COALU_SR）

偏移地址：0x0C

复位值：0x0000 0000

通过配置本寄存器可以把运算结果直接映射到操作数寄存器，作为下一次运算的输入操作数。软件可以轮询 BSY 位判断运算是否结束。当 BSY 为 1 时，写本寄存器的其他位会暂停 AHB 总线。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res											DIV_OFINT	DZINT	IOINT	QFINT	OFINT
											rw	rw	rw	rw	rw

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

位 31:21	Res：保留 必须保持复位值。
位 20	DIV_OFINT：除法溢出中断标志（Divide overrun interrupt flag） 硬件根据运算数据置 1 或清零该位。 软件往该位写 0 可清零，软件不能置 1 该位。 带符号除法中，如果被除数为 0x8000_0000_0000_0000，除数为 0xffff_ffff，这种情况下会置位 DIV_OFINT。这时所得的商为 0x8000_0000_0000_0000，余数为 0。
位 19	DZINT：除数为 0 中断标志（Divisor zero interrupt flag） 硬件根据运算数据置 1 或清零该位。 软件往该位写 0 可清零，软件不能置 1 该位被 0 除累积异常中断标志（浮点运算或普通运算都会更新）。
位 18	IOINT：非法操作中断标志（Illegal operation interrupt flag） 硬件根据运算数据置 1 或清零该位。 软件往该位写 0 可清零，软件不能置 1 该位非法操作异常中断标志（浮点数操作专用，结果是 NaN 时置位 IOINT）。
位 17	QFINT：饱和异常中断标志（Saturation error interrupt flag） 硬件根据运算数据置 1 或清零该位。 软件往该位写 0 可清零，软件不能置 1 该位饱和异常中断标志，标识运算结果被饱和
位 16	OFINT：溢出异常中断标志（Overrun error interrupt flag） 硬件根据运算数据置 1 或清零该位。 软件往该位写 0 可清零，软件不能置 1 该位溢出异常中断标志（浮点运算溢出或普通运算溢出）。
位 15:1	Res：保留 必须保持复位值。
位 0	BSY：忙标志（Busy） • 0：表示协处理器当前空闲

	• 1：表示协处理中运算正在进行 BSY 信号由硬件置位和清零，不可被软件改写。

如果 COALU_SR 中的 xxINT 位被置 1，同时 COALU_CR 中其对应的中断使能位也为 1，则发送 COALU 中断请求。

34.3.4 标志寄存器（COALU_FR）

偏移地址：0x10

复位值：0x00000000

通过配置本寄存器可以把运算结果直接映射到操作数寄存器，作为下一次运算的输入操作数。当 COALU_SR.BSY 为 1 时访问本寄存器会暂停 AHB 总线。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res												FV	Res	FZ	FN
												rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res											Q	V	Res		N
											rw	rw			rw

位 31:20	Res：保留 必须保持复位值。
位 19	FV：浮点运算溢出标志（Floating-point operation overflow flag） 硬件根据运算数据置 1 或清零该位。 软件往该位写 0 可清零，软件不能置 1 该位。
位 18	Res：保留 必须保持复位值。
位 17	FZ：浮点运算结果等于 0 标志（The result of a floating-point operation equals to 0 flag） 硬件根据运算数据置 1 或清零该位。 软件往该位写 0 可清零，软件不能置 1 该位。
位 16	FN：浮点运算结果为负标志（The result of a floating-point operation is negative flag） 硬件根据运算数据置 1 或清零该位。 软件往该位写 0 可清零，软件不能置 1 该位。
位 15:5	Res：保留 必须保持复位值。
位 4	Q：饱和标志（Saturated flag） 该位标识运算结果被饱和。 硬件根据运算数据置 1 或清零该位。 软件往该位写 0 可清零，软件不能置 1 该位。
位 3	V：溢出标志（Overflow flag） 该位标识运算结果发生溢出。 硬件根据运算数据置 1 或清零该位。 软件往该位写 0 可清零，软件不能置 1 该位。

位 2:1	Res：保留 必须保持复位值。
位 0	N：负标志（Negative flag） 该位标识运算结果为负数。 硬件根据运算数据置 1 或清零该位。 软件往该位写 0 可清零，软件不能置 1 该位。

注意：

不是所有运算都会更新 Q，V，N 标志。

当一次运算还没有完成时，如果读标志寄存器会 halt 总线直到运算完成。

软件可以把标志位清零，不能置 1。

34.3.5 SIMD 指令标志寄存器（COALU_SIMDFR）

偏移地址：0x14

复位值：0x00000000

通过配置本寄存器可以把运算结果直接映射到操作数寄存器，作为下一次运算的输入操作数。当 COALU_SR.BSY 为 1 时访问本寄存器会暂停 AHB 总线。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res												GE3	GE2	GE1	GE0
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Q3	Q2	Q1	Q0	V3	V2	V1	V0	Z3	Z2	Z1	Z0	N3	N2	N1	N0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:20	Res：保留 必须保持复位值。
位 19	GE3：SIMD 指令运算结果第 3 段大于 0 标志（The 3rd segment of result of a SIMD operation greater than 0 flag）
位 18	GE2：SIMD 指令运算结果第 2 段大于 0 标志（The 2nd segment of result of a SIMD operation greater than 0 flag）
位 17	GE1：SIMD 指令运算结果第 1 段大于 0 标志（The 1st segment of result of a SIMD operation greater than 0 flag）
位 16	GE0：SIMD 指令运算结果第 0 段大于 0 标志（The zero segment of result of a SIMD operation greater than 0 flag）
位 15	Q3：SIMD 指令运算结果第 3 段饱和标志（The 3rd segment of result of a SIMD operation saturated flag）
位 14	Q2：SIMD 指令运算结果第 2 段饱和标志（The 2nd segment of result of a SIMD operation saturated flag）
位 13	Q1：SIMD 指令运算结果第 1 段饱和标志（The 1st segment of result of a SIMD operation saturated flag）
位 12	Q0：SIMD 指令运算结果第 0 段饱和标志（The zero segment of result of a SIMD operation saturated flag）
位 11	V3：SIMD 指令运算结果第 3 段溢出标志（The 3rd segment of result of a SIMD operation Overflow flag）

位 10	V2：SIMD 指令运算结果第 2 段溢出标志（The 2nd segment of result of a SIMD operation Overflow flag）
位 9	V1：SIMD 指令运算结果第 1 段溢出标志（The 1st segment of result of a SIMD operation Overflow flag）
位 8	V0：SIMD 指令运算结果第 0 段溢出标志（The zero segment of result of a SIMD operation Overflow flag）
位 7	Z3：SIMD 指令运算结果第 3 段为零标志（The 3rd segment of result of a SIMD operation equals to 0 flag）
位 6	Z2：SIMD 指令运算结果第 2 段为零标志（The 2nd segment of result of a SIMD operation equals to 0 flag）
位 5	Z1：SIMD 指令运算结果第 1 段为零标志（The 1st segment of result of a SIMD operation equals to 0 flag）
位 4	Z0：SIMD 指令运算结果第 0 段为零标志（The zero segment of result of a SIMD operation equals to 0 flag）
位 3	N3：SIMD 指令运算结果第 3 段为负标志（The 3rd segment of result of a SIMD operation is negative flag）
位 2	N2：SIMD 指令运算结果第 2 段为负标志（The 2nd segment of result of a SIMD operation is negative flag）
位 1	N1：SIMD 指令运算结果第 1 段为负标志（The 1st segment of result of a SIMD operation is negative flag）
位 0	N0：SIMD 指令运算结果第 0 段为负标志（The zero segment of result of a SIMD operation is negative flag）

注意：

不是所有运算都会更新标志位。

若 SIMD 指令为半字运算，则 x3，x2 两个位表示运算结果的高半字状态，并且 x3 和 x2 位相等。

若 SIMD 指令为字节运算，则 x3，x2，x1，x0 每个 bit 分别结果每个字节的状态。

软件可以把标志位清零，不能置 1。

34.3.6 通用寄存器 x（COALU_Rx）(x=0..11)

偏移地址：0x20+4*x

复位值：0x00000000

当 COALU_SR.BSY 为 1 时访问本寄存器会暂停 AHB 总线。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw															

位 31:0	DATA[31:0]：数据（32 位）（Data register） 用户操作寄存器，可以软件写入 32 位数据，也可以硬件更新数据。 用户把待运算的数据写到通用寄存器，然后运算的结果也可以存储到通用寄存器。

35 器件电子签名（UID）

电子签名存放在闪存存储器模块的系统存储区域，可以通过 JTAG/SWD 或者 CPU 读取。它所包含的芯片识别信息在出厂时编写，用户固件或者外部设备可以读取电子签名，用以自动匹配不同配置的 HK32F39A 微控制器。

35.1 存储器容量寄存器

35.1.1 闪存容量寄存器

基地址：0x1FFF F7E0

只读，它的内容在出厂时编写

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F_SIZE[15:0]															
r															

位 15:0	F_SIZE[15:0]：闪存存储器容量（Flash size） 以 K 字节为单位指示产品中闪存存储器容量。例：0x0080=128K 字节

35.1.2 UID 寄存器（96 位）

基地址：0x1FFF F7E8

产品唯一的身份标识非常适合：

- 用来作为序列号（例如 USB 字符序列号或者其他的终端应用）
- 用来作为密码，在编写闪存时，将此唯一标识与软件加解密算法结合使用，提高代码在闪存存储器内的安全性。
- 用来激活带安全机制的自举过程。

96 位的产品唯一身份标识所提供的参考号码对任意一个航顺 32 位微控制器，在任何情况下都是唯一的。用户无论在何种情况下，都不能修改这个身份标识。这个 96 位的产品唯一身份标识，按照用户不同的用法，可以以字节（8 位）为单位读取，也可以以半字（16 位）或者全字（32 位）读取。

35.1.2.1 UID0 寄存器（U_ID0）

偏移地址：0x00

复位值：0xXXXXXXXX

说明：此处 X 表示出厂前编程设置。

U_ID0 为 UID 的 0 到 15 位。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID0[15:0]															
r															

位 15:0	U_ID0[15:0]：唯一身份标志 15:0 位（Unique ID bits）

35.1.2.2 UID1 寄存器（U_ID1）

偏移地址：0x02

复位值：0xXXXXXXXX

说明：此处 X 表示出厂前编程设置。

U_ID1 为 UID 的 16 到 31 位。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID1[15:0]															

r

位 15:0	U_ID1[15:0]：唯一身份标志 31:16 位。（Unique ID bits） 这个域的数值也预留作为未来的其它功能。

35.1.2.3 UID2 寄存器（U_ID2）

偏移地址：0x04

复位值：0xXXXXXXXX

说明：此处 X 表示出厂前编程设置。

U_ID2 为 UID 的 32 到 63 位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
U_ID2[31:16]															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID2[15:0]															
r															

位 31:0	U_ID2[31:0]：唯一身份标志 63:32 位（Unique ID bits）

35.1.2.4 UID3 寄存器（U_ID3）

偏移地址：0x08

复位值：0xXXXXXXXX

说明：此处 X 表示出厂前编程设置。

U_ID3 为 UID 的 64 到 95 位。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
U_ID3[31:16]															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID3[15:0]															
r															

位 31:0	U_ID3[31:0]：唯一身份标志 95:64 位（Unique ID bits）

36 调试支持（DBG）

36.1 概况

器件使用 Cortex®-M3 内核，该内核内含硬件调试模块，支持复杂的调试操作。硬件调试模块允许内核在取指（指令断点）或访问数据（数据断点）时停止。内核停止时，内核的内部状态和系统的外部状态都是可以查询的。完成查询后，内核和外设可以被复原，程序将继续执行。

当器件连接到调试器并开始调试时，调试器将使用内核的硬件调试模块进行调试操作。

支持两种调试接口：

- 串行接口
- JTAG 调试接口

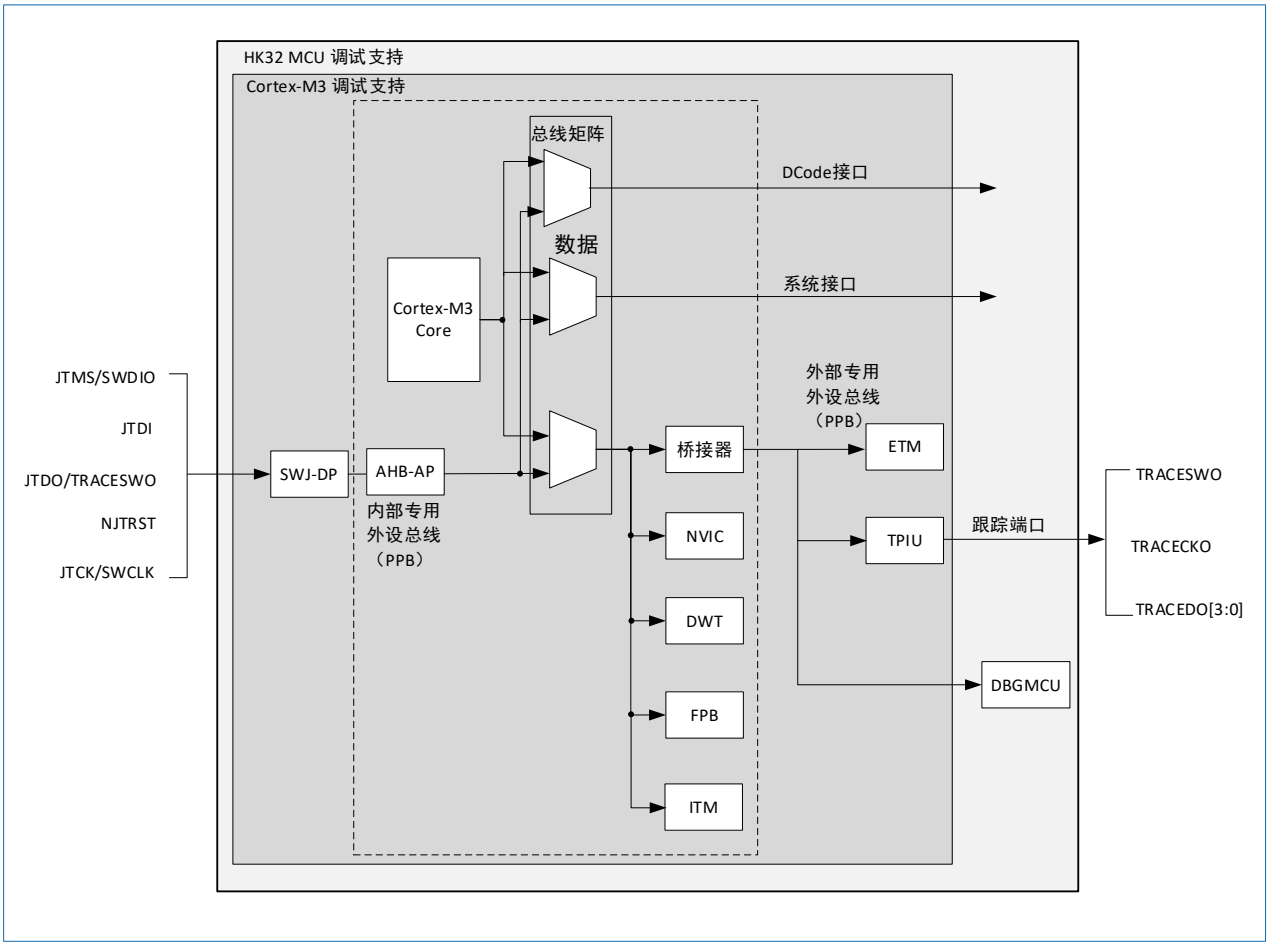


图 36-1 微控制器级别和 Cortex®-M3 级别的调试框图

注意：Cortex®-M3 内核内含的硬件调试模块是 ARM®CoreSight 开发工具集的子集。

ARM®Cortex®-M3 内核提供集成的片上调试功能。它由以下部分组成：

- SWJ-DP：串行/JTAG 调试端口
- AHP-AP：AHB 访问端口
- ITM：执行跟踪单元
- FPB：闪存指令断点
- DWT：数据触发
- TPIU：跟踪单元接口（仅支持 SWV 异步模式（TRACESWO 会被使用））

它还包含专用于 HK32F39A 的调试特性：

- 灵活的调试引脚分配
- MCU 调试盒（支持低电源模式，控制外设时钟等）

注意：更多 ARMCortex®-M3 内核的调试功能信息，请参考 Cortex®-M3（r1p1 版）技术参考手册（TRM）和 CoreSight 开发工具集（r1p0 版）TRM。

36.2 ARM®参考文献

- Cortex®-M3（r1p1 版）技术参考手册（TRM）
- ARM®调试接口 V5
- ARM®CoreSight 开发工具集（r1p0 版）技术参考手册

36.3 SWJ（Serial wire and JTAG）调试端口

器件内核集成了串行/JTAG 调试接口（SWJ-DP）。这是标准的 ARM®CoreSight 调试接口，包括 JTAG-DP 接口（5 个引脚）和 SW-DP 接口（2 个引脚）。

- JTAG 调试接口（JTAG-DP）为 AHP-AP 模块提供 5 针标准 JTAG 接口。
- 串行调试接口（SW-DP）为 AHP-AP 模块提供 2 针（时钟＋数据）接口。

在 SWJ-DP 接口中，SW-DP 接口的 2 个引脚和 JTAG 接口的 5 个引脚中的一些是复用的。

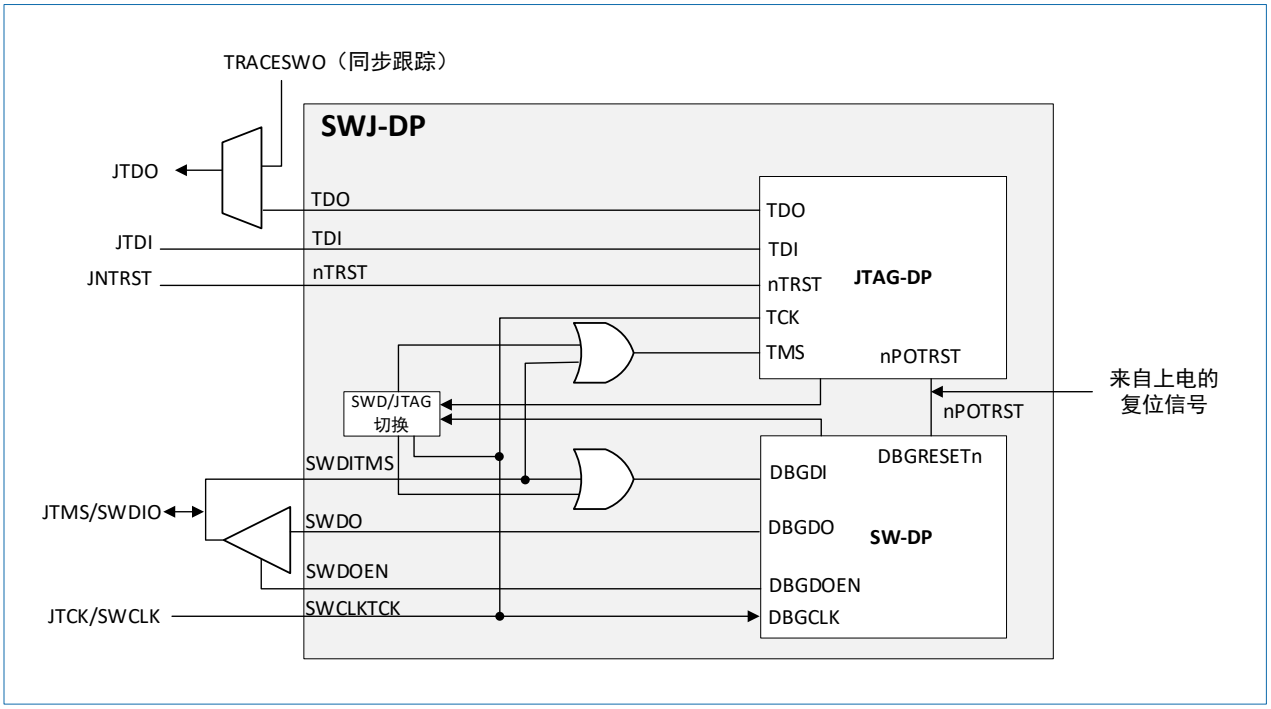


图 36-2 SWJ 调试端口

上面的图显示异步跟踪输出脚（TRACESWO）和 TDO 是复用的，因此异步跟踪功能只能在 SW-DP 调试接口上实现，不能在 JTAG-DP 调试接口上实现。

36.3.1 JTAG-DP 和 SW-DP 切换的机制

JTAG 调试接口是默认的调试接口。

如果调试器想要切换到 SW-DP，必须在 TMS/TCK 上输出一指定的 JTAG 序列（分别映射到 SWDIO 和 SWCLK），该序列禁止 JTAG-DP，并激活 SW-DP。该方法可以只通过 SWCLK 和 SWDIO 两个引脚来激活 SW-DP 接口。

指定的序列是：

1. 输出超过 50 个 TCK 周期的 TMS（SWDIO）＝1 信号
2. 输出 16 个 TMS（SWDIO）信号 0111100111100111（MSB）
3. 输出超过 50 个 TCK 周期的 TMS（SWDIO）＝1 信号

36.4 引脚分布和调试端口脚

不同封装的微控制器有不同的有效引脚数。因此，某些与引脚相关的功能可能随封装而不同。

36.4.1 SWJ 调试端口脚

器件的 5 个普通 I/O 口可用作 SWJ-DP 接口引脚。这些引脚在所有的封装里都存在。

表 36-1 SWJ 调试端口引脚

SWJ-DP 端口引脚名称	JTAG 调试接口		SW 调试接口		引脚分配
	类型	描述	类型	调试功能	
JTMS/SWDIO	输入	JTAG 模式选择	输入/输出	串行数据输入/输出	PA13
JTCK/SWCLK	输入	JTAG 时钟	输入	串行时钟	PA14
JTDI	输入	JTAG 数据输入	-	-	PA15
JTDO/TRACESWO	输出	JTAG 数据输出	-	跟踪时为 TRACESWO 信号	PB3
JNTRST	输入	JTAG 模块复位	-	-	PB4

36.4.2 灵活的 SWJ-DP 脚分配

复位（SYSRESETn 或 PORESETn）以后，属于 SWJ-DP 的所有 5 个引脚都立即被初始化为可被调试器使用的专用引脚（注意，并没有初始化跟踪输出脚，除非调试器对此脚进行定义）。

然而，器件可以用复用重映射和调试 I/O 配置寄存器（AFIO_MAPR）寄存器（详见章节：“10.4.2 复用重映射和调试 I/O 配置寄存器（AFIO_MAPR）”）来禁止 SWJ-DP 接口的部分或所有引脚的功能，这些专用引脚将被释放以用作普通 I/O 口。此寄存器被映射到和 Cortex®-M3 系统总线相连接的 APB 桥上。对此寄存器的设置将由用户代码而不是调试器完成。

3 个控制位用来配置 SWJ-DP 接口的引脚，这 3 个位在系统复位时复位。

- AFIO_MAPR（器件中的地址是 0x40010004）
  - 读：APB，无等待状态。
  - 写：APB，如果 AHB-APB 桥的写缓冲器满了，则一个等待状态位 26:24=SWJ_CFG[2:0]由软件置位和复位。

这 3 位用来设置分配给 SWJ 调试接口的专用引脚数目，目的是在使用不同的调试接口时能释放尽可能多的引脚用作普通 I/O 口。

复位后的初始值是 000（所有引脚都设置为 JTAG-DP 接口专用引脚），同时只能置位 3 个位中的一个（禁止同时设置一个以上的位）。

表 36-2 灵活的 SWJ_DP 引脚分配

SWJ-CFG[2:0]	配置为调试专用的引脚	SWJ 接口的 I/O 口分配				
		PA13/JTMS/SWDIO	PA14/JTCK/SWCLK	PA15/JTDI	PB3/JTDO	PB4/JNTRST
000	所有的 SWJ 引脚（JTAG-DP+SW-DP）复位状态	专用	专用	专用	专用	专用
001	所有的 SWJ 引脚（JTAG-DP+SW-DP）除了 JNTRST 引脚	专用	专用	专用	专用	释放
010	JTAG-DP 接口禁止，SW-DP 接口允许	专用	专用	释放		
100	JTAG-DP 接口和 SW-DP 接口都禁止	释放				
其他	禁止					

注意：

当 APB 桥的写缓冲区满了的时侯，在写 AFIO_MAPR 寄存器时需要多用一个 APB 周期。这是因为 JTAGSW 脚的释放需要 2 个 APB 周期，以保证输入内核的 nTRST 和 TCK 信号的平稳。

周期 1：输入 1/0 的 JTAGSW 信号到内核（nTRST，TDI 和 TMS 为 1，TCK 为 0）。

周期 2：GPI/O 控制器获得 SWJTAG I/O 引脚的控制信号（如对方向，上拉/下拉，施密特触发等的控制）。

36.4.3 JTAG 脚上的内部上拉和下拉

保证 JTAG 的输入引脚不是悬空的是非常必要的，因为它们直接连接到 D 触发器控制着调试模式。必须特别注意 SWCLK/TCK 引脚，因为它们直接连接到一些 D 触发器的时钟端。

为了避免任何未受控制的 I/O 电平，器件在 JTAG 输入脚上嵌入了内部上拉和下拉。

- JINTRST：内部上拉
- JTDI：内部上拉
- JTMS/SWDIO：内部上拉
- TCK/SWCLK：内部下拉

一旦 JTAG I/O 被用户代码释放，GPIO 控制器再次取得控制。这些 I/O 口的状态将恢复到复位时的状态：

- JNTRST：带上拉的输入
- JTDI：带上拉的输入
- JTMS/SWDIO：带上拉的输入
- JICK/SWCLK：带下拉的输入
- JTDO：浮动输入

软件可以把这些 I/O 口作为普通的 I/O 口使用。

注意：JTAGEEE 标准建议对 TDI，TMS 和 nTRST 上拉，而对 TCK 没有特别的建议。但在该芯片中，JTCK 引脚带有下拉。内嵌的上拉和下拉使芯片不再需要外加外部电阻。

36.4.4 利用串行接口并释放不用的调试脚作为普通 I/O 口

为了利用串行调试接口来释放一些普通 I/O 口，用户软件必须在复位后设置 SWJ_CFG=010，从而释放 PA15，PB3 和 PB4 用做普通 I/O 口。

在调试时，调试器进行以下操作：

- 在系统复位时，所有 SWJ 引脚被分配为专用引脚（JTAG-DP+SW-DP）。
- 在系统复位状态下，调试器发送指定 JTAG 序列，从 JTAG-DP 切换到 SW-DP。
- 仍然在系统复位状态下，调试器在复位地址处设置断点。
- 释放复位信号，内核停止在复位地址处。
- 从这里开始，所有的调试通信将使用 SW-DP 接口，其他 JTAG 引脚可以由用户代码改配为普通 I/O 口。

对于用户软件设计，应注意：

在复位后，这些专用引脚仍然处于带上拉的输入（nTRST，TMS，TDI），带下拉的输入（TCK），和输出（TDO）状态，并持续一段时间，直到用户代码释放这些引脚。当这些引脚被配置成专用引脚时（JTAG 或者 SW 或者 TRACE），修改相应的普通 I/O 口配置寄存器是无效的。

36.5 JTAGTAP 连接

器件内部串联了两个 JTAGTAP。边界扫描 TAP 专门用来进行测试（IR 寄存器为 5 位宽）和 Cortex®-M3TAP（IR 寄存器有 4 位宽）。

为了访问 Cortex®-M3 TAP 对芯片进行调试，必须：

1. 首先，必须将 BYPASS 指令移位输入 TMCTAP。
2. 其次，在移位输入 IR 时，每个扫描链包含 9 位（=5+4），对于不用的 TAP，必须输入 BYPASS 指令。
3. 移位输入数据时，不用的 TAP 处于 BYPASS 模式下，因此数据扫描链需要额外添加一位。

注意：重要：一旦使用了指定的 JTAG 序列选择了串行调试接口，TMC TAP 自动被禁止（JTMS 被强制为高）。

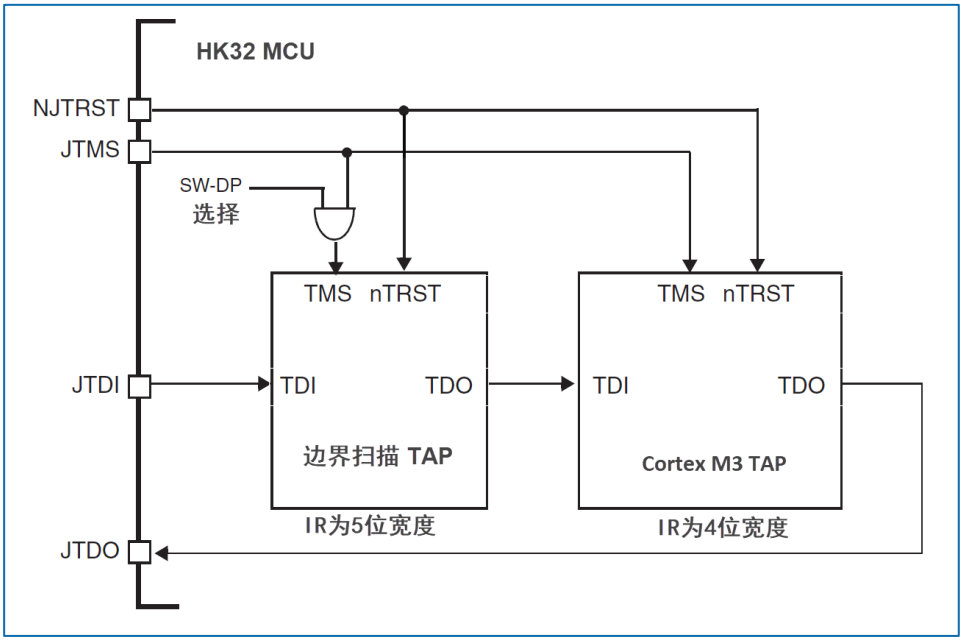


图 36-3 JTAGTAP 连接

36.6 ID 代码和锁定机制

在器件微控制器内部有多个 ID 编码。强烈建议工具设计者使用映射在外部 PPB 存储器上地址为 0xE0042000 的 MCU DEVICE ID 来锁定调试器。

36.6.1 微控制器设备 ID 编码

微控制器内含一个 MCUID 编码。这个 ID 定义了 MCU 的部件号和硅片版本。它是 DBG_MCU 的一个组成部分，并且映射到外部 PPB 总线上（详见章节：“36.15 MCU 调试模块（MCUDBG）”）。使用 JTAG 调试口（4~5 个引脚）或 SW 调试口（2 个引脚）或通过用户代码都可以访问此编码。即使当 MCU 处于系统复位状态下这个编码也可以被访问。MCU 器件 ID 代码寄存器，请参见“36.16.1 微控制器 ID 代码寄存器（DBGMCU_IDCODE）”。

36.6.2 边界扫描 TAP

36.6.2.1 JTAGID 编码

器件的边界扫描 TAP 集成了 JTAG ID 编码：

- 0x06410041：版本 A
- 0x16410041：版本 B 和版本 Z

36.6.3 Cortex-M3 TAP

ARM® Cortex®-M3 的 TAP 有一个 JTAGID 编码。这个 ID 编码是 ARM®默认的，且没有被修改过，只能通过 JTAG 调试口访问。此编码是 0x3BA00477（对应于 Cortex®-M3r1p1）。

调试器/编程工具应该只通过 DEV_ID（11:0）来识别芯片。

36.6.4 Cortex-M3 JEDEC-106 ID 代码

ARM®的 Cortex®-M3 有一个 JEDEC-106ID 编码。它位于映射到内部 PPB 总线地址为 0xE00FF000-0xE00FFFFF 的 4KB ROM 表中。

36.7 JTAG 调试端口

标准的 JTAG 状态机是通过一个 4 位的指令寄存器（IR）和 5 个数据寄存器实现的（详见 Cortex®-*M3 r1p1 Technical Reference Manual*）。

表 36-3 JTAG 调试端口数据寄存器

IR（3:0）	数据寄存器	描述
1110	IDCODE[32 位]	ID 编码寄存器 0x3BA00477（ARM®Cortex-M3 r1p1-01rel0 ID 编码）
1010	DPACC[32 位]	调试接口寄存器 初始化调试端口，并允许访问调试接口寄存器 ● 输入数据时： ○ 位 34:3=DATA[31:0]：对应写操作的 32 位数据位 ○ 位 2:1=A[3:2]：调试接口寄存器的 2 位地址值 ○ 位 0=RnW：读操作（1）或写操作（0） ● 输出数据时： ○ 位 34:3=DATA[31:0]：前一次读操作的 32 位数据结果 ○ 位 2:0=ACK[2:0]：3 位的应答

IR（3:0）	数据寄存器	描述
		-010＝成功/失败 -001＝等待 -其他＝未定义 ○ A（3:2）的定义请参考表 36-4
1011	APACC[35 位]	存取接口寄存器 初始化存取接口并允许访问存取接口寄存器 ● 输入数据时： ○ 位 34:3=DATA[31:0]：对应写操作的 32 位数据位 ○ 位 2:1=A[3:2]：2 位地址（AP 寄存器的部分地址） ○ 位 0=RnW：读操作（1）或写操作（0） ● 输出数据时： ○ 位 34:3=DATA[31:0]：前一次读操作的 32 位数据结果 ○ 位 2:0=ACK[2:0]：3 位的应答 -010＝成功/失败 -001＝等待 -其他＝未定义 关于 AP 寄存器请参考 AHB-AP 章节，这些寄存器的地址： 由以下部分组成： ● 移位值 A[3:2] ● DPSELECT 寄存器的当前值
1000	ABORT[35 位]	中止寄存器 ● 位 31:1 未定义 ● 位 0＝DAPABORT：写 1 产生一个 DAP 中止

表 36-4 由 A[3:2]定义的 32 位调试接口寄存器地址

地址	A（3:2）值	描述
0x0	00	未定义
0x4	01	DPCTRL/STAT 寄存器，用于： ● 请求一个系统或调试的上电操作 ● 配置 AP 访问的操作模式 ● 控制比较，校验操作 ● 读取一些状态位（溢出，上电响应）
0x8	10	DPSELECT 寄存器 用来选择当前的访问端口和有效的 4 字长寄存器窗口 ● 位 31:24：APSEL 选择当前 AP ● 位 23:8：未定义 ● 位 7:4：APBANKSEL：在当前 AP 上选择 4 字长寄存器窗口 ● 位 3:0：未定义
0xC	11	DPRDBUFF 寄存器：用来使调试器获得前一次操作的最终结果（不用再请求一个新的 JTAG-DP 操作）

36.8 SW 调试端口

36.8.1 SW 协议介绍

此同步串行协议使用 2 个引脚：

- SWCLK：从主机到目标的时钟信号
- SWDIO：双向数据信号

协议允许读写 2 个寄存器组（DPACC 和 APACC 寄存器组）。

数据位按 LSB 传输。

由于 SWDIO 为双向口，该引脚需有上拉（ARM®建议使用 100kΩ 电阻）。

按协议每次 SWDIO 方向改变时，需插入一个转换时间。在该期间内主机和目标都不驱动此信号线。转换时间的默认值是 1 个比特，但可以通过配置 SWCLK 频率来调节。

36.8.2 SW 协议序列

每个序列由 3 个阶段组成：

1. 主机发送包请求（8 位）
2. 目标发送确认响应（3 位）
3. 主机或目标发送数据（33 位）

表 36-5 请求包（8 位）

比特位	名称	描述
0	起始	必须为 1
1	APnDP	● 0：访问 DP ● 1：访问 AP
2	RnW	● 0：写请求 ● 1：读请求
4:3	A（3:2）	DP 或 AP 寄存器的地址（请参考 0）
5	Parity	前面比特位的校验位
6	Stop	0
7	Park	不能由主机驱动，由于有上拉，目标永远读为 1

有关 DPACC 和 APACC 寄存器描述的详细资料，请参考 Cortex®-M3 r1p1 技术参考手册。包请求后总是跟一个（默认为 1 位）转换时间，此时主机和目标都不驱动线路。

表 36-6 ACK 定义（3 位）

比特位	名称	描述
0..2	ACK	● 001：失败 ● 010：等待 ● 100：成功

当 ACK 为失败或等待，或者是一个回复读操作的 ACK，此 ACK 后有一个转换时间。

表 36-7 传输数据（33 位）

比特位	名称	描述
0..31	WDATA/RDATA	写或读的数据
32	Parity	32 位数据的奇偶校验位

读操作的数据传输操作后有一个转换时间。

36.8.3 SW-DP 状态机（Reset，idlestates，IDcode）

SW-DP 状态机有一个内部 ID 编码用来识别 SW-DP，它遵守 JEP-106 标准。此 ID 编码是 ARM®默认的编码，值为 0x1BA01477（对应于 Cortex-M3 r1p1）。

注意：在调试器读这个 ID 编码之前，SW-DP 的状态机是不工作的。

- SW-DP 状态机在以下条件下将处于复位状态：
  - 上电复位。
  - DP 从 JTAG 切换到 SWD 后。
  - 线路有超过 50 个周期的高电平。
- SW-DP 状态机在此条件下处于空闲状态：复位后，线路有至少 2 个周期的低电平。
- 当状态机处于 RESET 状态后，必须首先进入 IDLE 状态，并执行一个读 DP-SWID 寄存器的操作。否则，调试器在执行其他传输时，只能获得一个失败的 ACK 响应。

更详细的 SW-DP 状态机资料请参考 Cortex-M3 r1p1 技术参考手册和 Core Sight Design Kit r1p0 技术参考手册。

36.8.4 DP 和 AP 读/写访问

- 对 DP 的读操作没有延迟：调试器将直接获得数据（如果“ACK＝OK”），或者等待（如果“ACK=WAIT”）。
- 对 AP 的读操作具有延迟。即前一次读操作的结果只能在下一次操作时获得。如果下一次的操作不是对 AP 的访问，则必须读 DP-RDBUFF 寄存器来获得上一次读操作的结果。
- DP-CTRL/STAT 寄存器的 READOK 标志位会在每次 AP 读操作和 RDBUFF 读操作后更新，以通知调试器 AP 的读操作是否成功。
- SW-DP 具有写缓冲区（DP 和 AP 都有写缓冲），这使得其他传输在进行时，仍然可以接受写操作。如果写缓冲区满，调试器将获得一个“WAIT”的 ACK 响应。读 IDCODE 寄存器，读 CTRL/STAT 寄存器和写 ABORT 寄存器操作在写缓冲区满时仍被接受。
- 由于 SWCLK 和 HCLK 的异步性，需要在写操作后（在奇偶校验位后）插入 2 个额外的 SWCLK 周期，以确保内部写操作正确完成。这两个额外的时钟周期需要在线路为低时插入（IDLE 状态下）。这个操作步骤在写 CTRL/STAT 寄存器以提出一个上电请求时尤其重要，否则下一个操作（在内核上电后才有效的操作）会立即执行，这将导致失败。

36.8.5 SW-DP 寄存器

当 APnDP=0 时，可以访问以下这些寄存器。

表 36-8 SW-DP 寄存器

A（3:2）	读/写	SELECT 寄存器的 CTRLSEL 位	寄存器	描述
00	读		IDCODE	固定为 0x1BA01477（用于识别 SW-DP）。

A（3:2）	读/写	SELECT 寄存器的 CTRLSEL 位	寄存器	描述
00	写		ABORT	
01	读/写	0	DP-CTRL/STAT	请求一个系统或调试的上电操作； 配置 AP 访问的操作模式； 控制比较，校验操作； 读取一些状态位（溢出，上电响应）。
01	读/写	1	WIRE CONTROL	配置串行通信物理层协议（如转换时间长度等）。
10	读		READ RESEND	允许从一个错误的调试传输中恢复数据而不用重复最初的 AP 传输。
10	写		SELECT	选择当前的访问端口和有效的 4 字长寄存器窗口。
11	读/写		READ BUFFER	由于 AP 的访问具有传递性（当前 AP 读操作的结果会在下次 AP 传输时传出）,因此这个寄存器非常必要。这个寄存器会从 AP 捕获上一次读操作的数据结果，因此可以获得数据而不必再启动一个新的 AP 传输。

36.8.6 SW-AP 寄存器

当 APnDP=1 时，可以访问以下这些寄存器。

AP 寄存器的访问地址由以下两部分组成：

- A[3:2]的值。
- DPSELECT 寄存器的当前值。

36.9 对于 JTAG-DP 或 SWDP 都有效的 AHB-AP（AHB 访问端口）

功能：

- 系统访问是独立于处理器状态的
- JTAG-DP 和 SW-DP 都可以访问 AHB-AP
- AHB-AP 是总线矩阵的 AHB 主设备。因此，它可以访问所有的数据总线（Dcode 总线，System 总线，内部和外部 PPB 总线），只有 ICode 总线除外。
- 支持位寻址的传输
- 旁路 FPB 的 AHB-AP 传输

32 位 AHP-AP 寄存器的地址是 6-位宽（最多 64 个字或 256 个字节），由以下部分组成：

- 位[8:4]=DPSELECT 寄存器的位[7:4]APBANKSEL
- 位[3:2]=35 位 SW-DP 包请求中的 A（3:2）。

Cortex®-M3 的 AHB-AP 有 9 个 32 位的寄存器。

表 36-9 Cortex®-M3 AHB-AP 寄存器

偏移地址	寄存器名	描述
0x00	AHB-AP Control and Status Word	配置 AHB 接口的传输特性（长度，地址自加模式，当前传输状态，特权模式等）。
0x04	AHB-AP Transfer Address	-

偏移地址	寄存器名	描述
0x0C	AHB-AP Data Read/Write	-
0x10	AHB-AP Banked Data0	直接访问 4 个相连的字而不用重写访问地址。
0x14	AHB-AP Banked Data1	
0x18	AHB-AP Banked Data2	
0x1C	AHB-AP Banked Data3	
0xF8	AHB-AP Debug ROM Address	调试接口的基地址。
0xFC	AHB-AP ID Register	-

更多信息请参考 Cortex®-M3 r1p1 技术参考手册。

36.10 内核调试

通过操作内核调试寄存器可以实行对内核的调试。对这些寄存器的访问通过先进高性能总线（AHB-AP）进行。处理器可以通过内部私有外设总线（PPB）直接访问这些寄存器。

它包括 4 个寄存器。

表 36-10 内核调试寄存器

寄存器	描述
DHCSR	32 位的调试控制和状态寄存器此寄存器提供内核状态信息，允许内核进入调试模式，和提供单步功能。
DCRSR	17 位的内核寄存器调试选择寄存器此寄存器选择需要进行读写操作的内核寄存器。
DCRDR	32 位的内核寄存器调试数据寄存器此寄存器存放由 DCRSR 选择的内核寄存器读出的或需要写入的数据。
DEMCR	32 位异常调试和监视控制寄存器此寄存器提供向量传输和监视调试控制功能。TRCENA 位启动 TRACE 功能。

注意：这些寄存器在系统复位时不复位，仅在上电复位时复位。

更多详细资料请参考 Cortex®-M3 r1p1 技术参考手册。

为了在复位后立即使内核进入调试状态，需要：

- 使能调试和异常监视控制寄存器的位 0（VC_CORRESET）。
- 使能调试控制和状态寄存器的位 0（C_DEBUGEN）。

36.11 调试器主机在系统复位下的连接能力

器件的复位系统由下列复位源组成：

- POR（上电复位），在每次上电时发起一次复位
- 内部看门狗复位
- 软件复位
- 外部复位

Cortex®-M3 将调试部分的复位（通常是 PORRESETn）和其他复位（SYSRESETn）区分开。因此，当

内核处于系统复位状态时，调试器可以连接到内核，配置内核调试寄存器，使能调试允许位，这样操作使内核在系统复位被释放时立即进入调试状态而不执行任何指令。同样的，可以在内核处于复位状态下时配置调试特性。

注意：强烈建议调试器在系统复位时连接内核（在复位向量处设置断点）。

36.12 Flash 补丁和断点单元

FPB（Flash patch breakpoint）单元：

- 实现硬件断点。
- 用系统区域的代码和数据取代代码区域的代码和数据。此特性可以用来纠正代码区域内的软件错误。

软件补丁功能和硬件断点功能不能同时使用。

FPB 由以下部分组成：

- 2 个内容比较器，用来比较代码区域取得的内容并重映射到系统区域的相关地址。
- 6 个指令比较器，用来比较代码区域的指令。这些比较器可用来实现软件补丁或者硬件断点功能。

36.13 数据观察点触发（DWT）

DWT（Data watchpoint trigger）模块由以下比较器组成，它们分别是：

- 一个硬件数据比较器
- 一个 PC 值取样器
- 一个数据地址取样器

DWT 还可用来获取某些侧面的信息。通过一些计数器可以获得以下数据：

- 时钟周期
- 分支指令
- 存取单元操作
- 睡眠周期
- CPI（每条指令的执行时间）
- 中断开销

36.14 指令跟踪微单元（ITM）

36.14.1 概述

ITM（Instrumentation trace macrocell）是一应用驱动的跟踪源，它支持 *printf* 类的调试手段来跟踪操作系统（OS）和应用事件，并发布判定的系统信息。ITM 以包的形式发布跟踪信息，它由以下部分组成：

- 软件跟踪：软件可以通过直接写 ITM 激发寄存器来发布包信息。
- 硬件跟踪：ITM 会发布由 DWT 产生的信息包。
- 时间戳：时间戳被发布到相应的包上。ITM 包含一个 21 位的计数器以产生时间戳。Cortex-M3 的时钟或串行线观测器的位时钟率给计数器提供时钟。

由 ITM 发送的信息包输出到跟踪端口接口单元（TPIU），TPIU 再添加一些额外的包（参考 TPIU），然后输出完整的包序列给调试器。

用户在设置或使用 ITM 之前，必须先使能异常调试和监视控制寄存器的 TRCEN 位。

36.14.2 时间戳包，同步和溢出包

时间戳包包含了时间戳信息，普通的控制和同步信息。它使用一个 21 位的时间戳计数器（及可能的预分频器），此计数器在每个时间戳包发放时复位。计数器的时钟可以是 CPU 时钟也可以是 SWV 时钟。

同步包为 0x80_00_00_00_00_00，按 000000000080 发送给 TPIU（LSB 在前）。同步包是时间戳包的控制信号。

它也在每个 DWT 触发时发送，因此 DWT 必须配置为触发 ITM：必须设置 DWT 控制寄存器的位 0（CYCCNTENA）。此外，也必须设置 ITM 跟踪控制寄存器的位 2（SYNCENA）。

注意：如果 SYNENA 位没有被置起，DWT 产生给 TPIU 的同步触发，将只发送 TPIU 同步包而不发送 ITM 同步包。

溢出包是一个特殊的时间戳包，该包指示数据已经被写但是 FIFO 已满。

表 36-11 主要的 ITM 寄存器

地址	寄存器	描述
@E0000FB0	ITM Lock Access	写入 0xC5ACCE55 允许写其他 ITM 寄存器
@E0000E80	ITM Trace Control	• 位[31:24]：总是 0 • 位 23：busy • 位[22:16]：7 位的 ATBID 用以识别跟踪数据源 • 位[15:10]：总是 0 • 位[9:8]：时间戳的预分频 • 位[7:5]：未定义 • 位 4：使能 SWV 功能即时间戳计数器使用 SWV 时钟 • 位 3：使能 DWT 的激发功能 • 位 2：此位必须设为 1 来使能 DWT 的产生同步触发功能，以使 TPIU 能够发送同步包 • 位 1：时间戳使能 • 位 0：ITM 的全局使能位
@E0000E40	ITM Trace Privilege	• 位 3：置'1'使能跟踪端口[31:24] • 位 2：置'1'使能跟踪端口[23:16] • 位 1：置'1'使能跟踪端口[15:8] • 位 0：置'1'使能跟踪端口[7:0]
@E0000E00	ITM Trace Enable	每个比特位使能相应的触发端口产生跟踪
@E0000000-E000007C	Stimulus Port Registers 0-31	向选中的产生跟踪的触发端口（32 个）写 32 位数据

关于配置的例子：

向 TUIU 输出一个简单值：

- 配置 TPIU 并使能 I/IO_TRACEN 以使 MCU 分配 TRACE 的引脚（参见：“跟踪引脚分配”和“调试 MCU 配置寄存器”）；
- 向 ITM Lock Access 寄存器写入 0xC5ACCE55，以允许写其他 ITM 寄存器；
- 向 Trace Control 寄存器写入 0x00010005，使能 TPIU 的同步包并使能整个 ITM 功能，寄存器中的 ATBID 为 0x01；
- 向 ITM Trace Enable 寄存器写入 0x1，以使能触发端口 0；

- 向 ITM Trace Privilege 寄存器写入 0x1，关闭对触发端口 7:0 的屏蔽；
- 把需要输出的值写入触发端口 0 寄存器，这个步骤可以通过软件完成（使用 *printf* 功能）。

36.15 MCU 调试模块（MCUDBG）

MCU 调试模块协助调试器提供以下功能：

- 低功耗模式
- 在断点时提供定时器、看门狗、I2C 和 CAN 的时钟控制
- 对跟踪脚分配的控制

36.15.1 低功耗模式的调试支持

使用 WFI 和 WFE 可以进入低功耗模式。

MCU 支持多种低功耗模式，分别可以关闭 CPU 时钟，或降低 CPU 的能耗。

内核不允许在调试期间关闭 FCLK 或 HCLK。这些时钟对于调试操作是必要的，因此在调试期间，它们必须工作。MCU 使用一种特殊的方式，允许用户在低功耗模式下调试代码。

为实现这一功能，调试器必须先设置一些配置寄存器来改变低功耗模式的特性。

- 在睡眠模式下，调试器必须先置位 DBGMCU_CR 寄存器的 DBG_SLEEP 位。这将为 HCLK 提供与 FCLK（由代码配置的系统时钟）相同的时钟。
- 在停机/待机模式下，调试器必须先置位 DBGMCU_CR 寄存器的 DBG_STOP/DBG_STANDBY 位。这将激活内部 RC 振荡器，在停止模式下为 FCLK 和 HCLK 提供时钟。

36.15.2 支持定时器、看门狗、CAN 和 I2C 的调试

在产生断点时，有必要根据定时器和看门狗的不同用途选择计数器的工作模式：

- 在产生断点时，计数器继续计数。这在输出 PWM 控制电机时常要用到。
- 在产生断点时，计数器停止计数。这对于看门狗的计数器是必需的。

对于 CAN，用户可以选择在断点期间阻止接收寄存器的更新。对于 I2C，用户可以选择在断点期间阻止 SMBUS 超时。

36.16 DBGMCU 寄存器

基地址：0xE0042000

空间大小：0x400

此寄存器允许在调试状态下下配置 MCU。包括：

- 支持低功耗模式
- 支持定时器和看门狗的计数器
- 支持 CAN 通信
- 分配跟踪引脚

DBGMCU_CR 寄存器被映射到外部 PPB 总线，基地址为 0xE004 2000。寄存器由 PORESET 异步复位（不被系统复位所复位）。当内核处于复位状态下时，调试器可写该寄存器。如果调试器不支持这些特性，用户软件仍可写这些寄存器。

36.16.1 微控制器 ID 代码寄存器（DBGMCU_IDCODE）

偏移地址：0x00

复位值：0x1003 6414

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV_ID[15:0]															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEV_ID[11:0]															
r															

位 31:16	REV_ID[15:0]：版本识别（Revision identifier） 该域标识产品的版本：0x1003
位 15:0	DEV_ID[11:0]：设备识别（Device identifier） 这个部分指示了设备编码。 对于 HK32F39A 微控制器：设备编码为 0x6414。

36.16.2 调试 MCU 配置寄存器（DBGMCU_CR）

偏移地址：0x04

复位值：0x0000 0000

注：该寄存器只能被 POR 复位，不被系统复位所复位，只支持 32 位访问

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res										DBG_CAN2_STOP	DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM8_STOP	DBG_I2C2_SMBUS_TIMEOUT
										rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBG_I2C1_SMBUS_TIMEOUT	DBG_CAN1_STOP	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP	DBG_TIM1_STOP	DBG_WWDG_STOP	DBG_IWDG_STOP	TRACE_MODE[1:0]		TRACE_IOEN	Res		DBG_STANDBY	DBG_STOP	DBG_SLEEP
rw	rw	rw	rw	rw	rw	rw	rw	rw		rw			rw	rw	rw

位 31:22	Res：保留 必须保持复位值。
位 21	DBG_CAN2_STOP：CORE halt 后，调试 CAN2 停止（Debug CAN2 stopped when core is halted） ● 0：与正常模式一样 ● 1：CAN2 接受寄存器停止
位 20	DBG_TIM7_STOP：当内核停止时停止定时器计数器（Debug TIM17 stopped when core is halted） • 0：当内核停止时，仍然向相关定时器的计数器提供时钟，定时器输出工作正常 • 1：当内核停止时，切断相关定时器的计数器的时钟，同时关闭定时器的输出（就好像对某一暂停事件的紧急响应，停止定时器）
位 19	DBG_TIM6_STOP：当内核停止时停止定时器计数器（Debug TIM6 stopped when core is halted） • 0：当内核停止时，仍然向相关定时器的计数器提供时钟，定时器输出工作正常 • 1：当内核停止时，切断相关定时器的计数器的时钟，同时关闭定时器的输出（就好像对某一暂停事件的紧急响应，停止定时器）
位 18	DBG_TIM5_STOP：当内核停止时停止定时器计数器（Debug TIM5 stopped when core is halted） • 0：当内核停止时，仍然向相关定时器的计数器提供时钟，定时器输出工作正常。 • 1：当内核停止时，切断相关定时器的计数器的时钟，同时关闭定时器的输出（就好像对某一暂停事件的紧急响应，停止定时器）。

位 17	DBG_TIM8_STOP：当内核停止时停止定时器计数器（Debug TIM8 stopped when core is halted） • 0：当内核停止时，仍然向相关定时器的计数器提供时钟，定时器输出工作正常。 • 1：当内核停止时，切断相关定时器的计数器的时钟，同时关闭定时器的输出（就好像对某一暂停事件的紧急响应，停止定时器）。
位 16	DBG_I2C2_SMBUS_TIMEOUT：当内核停止时停止 SMBUS 超时模式（I2C2 SMBUS timeout mode stopped when core is halted） • 0：与正常模式操作相同 • 1：冻结 SMBUS 的超时控制
位 15	DBG_I2C1_SMBUS_TIMEOUT：当内核停止时停止 SMBUS 超时模式（I2C1 SMBUS timeout mode stopped when core is halted） • 0：与正常模式操作相同 • 1：冻结 SMBUS 的超时控制
位 14	DBG_CAN1_STOP：当内核进入调试状态时，CAN 停止运行（Debug CAN stopped when core is halted） • 0：CAN 仍然正常运行 • 1：CAN 的接收寄存器不继续接收数据
位 13	DBG_TIM4_STOP：当内核进入调试状态时计数器停止工作（Debug TIM4 stopped when core is halted） • 0：选中定时器的计数器仍然正常工作 • 1：选中定时器的计数器停止工作
位 12	DBG_TIM3_STOP：当内核进入调试状态时计数器停止工作（Debug TIM3 stopped when core is halted） • 0：选中定时器的计数器仍然正常工作 • 1：选中定时器的计数器停止工作
位 11	DBG_TIM2_STOP：当内核进入调试状态时计数器停止工作（Debug TIM2 stopped when core is halted） • 0：选中定时器的计数器仍然正常工作 • 1：选中定时器的计数器停止工作
位 10	DBG_TIM1_STOP：当内核进入调试状态时计数器停止工作（Debug TIM1 stopped when core is halted） • 0：选中定时器的计数器仍然正常工作 • 1：选中定时器的计数器停止工作
位 9	DBG_WWDG_STOP：当内核进入调试状态时窗口看门狗停止工作（Debug window watch dog stopped when core is halted ） • 0：窗口看门狗计数器仍然正常工作 • 1：窗口看门狗计数器停止工作
位 8	DBG_IWDG_STOP：当内核进入调试状态时独立看门狗停止工作（Debug independent watch dog stopped when core is halted） • 0：独立看门狗计数器仍然正常工作 • 1：独立看门狗计数器停止工作
位 7:6	TRACE_MODE[1:0]：跟踪引脚模式选择（Trace pin assignment control） 和 TRACE_IOEN 一起适用控制跟踪引脚分配。 • 当 TRACE_IOEN＝0 时 TRACE_MODE＝xx：不分配跟踪引脚（默认状态）。

	• 当 TRACE_IOEN=1 时
	  ○ TRACE_MODE=00：跟踪引脚使用异步模式。
	  ○ TRACE_MODE=01：跟踪引脚使用同步模式，并且数据长度为 1。
	  ○ TRACE_MODE=10：跟踪引脚使用同步模式，并且数据长度为 2。
	  ○ TRACE_MODE=11：跟踪引脚使用同步模式，并且数据长度为 4。

位 5	TRACE_IOEN：跟踪引脚分配使能（Trace pin assignment enable） • 0：不分配跟踪引脚 • 1：分配跟踪引脚
位 4:3	Res：保留 必须保持复位值。
位 2	DBG_STANDBY：调试待机模式（Debug Standby mode） • 0：（FCLK 关，HCLK 关）整个数字电路部分都断电 从软件的观点看，退出 STANDBY 模式与复位是一样的（除了一些状态位指示了微控制器刚从 STANDBY 状态退出）。 • 1：（FCLK 开，HCLK 开）数字电路部分不下电 FCLK 和 HCLK 时钟由内部 RL 振荡器提供时钟。另外，微控制器通过产生系统复位来退出 STANDBY 模式和复位是一样的。
位 1	DBG_STOP：调试停机模式（Debug Stop mode） • 0：（FCLK 关，HCLK 关） 在停止模式时，时钟控制器禁止一切时钟（包括 HCLK 和 FCLK）。当从 STOP 模式退出时，时钟的配置和复位之后的配置一样（微控制器由 8MHz 的内部 RC 振荡器（HSI）提供时钟）。因此，软件必须重新配置时钟控制系统启动 PLL，晶振等。 • 1：（FCLK 开，HCLK 开） 在停止模式时，FCLK 和 HCLK 时钟由内部 RC 振荡器提供。当退出停止模式时，软件必须重新配置时钟系统启动 PLL，晶振等（与配置此位为 0 时的操作一样）。
位 0	DBG_SLEEP：调试睡眠模式（Debug Sleep mode） • 0：（FCLK 开，HCLK 关） 在睡眠模式时，FCLK 由原先已配置好的系统时钟提供，HCLK 则关闭。由于睡眠模式不会复位已配置好的时钟系统，因此从睡眠模式退出时，软件不需要重新配置时钟系统。 • 1：（FCLK 开，HCLK 开） 在睡眠模式时，FCLK 和 HCLK 时钟都由原先配置好的系统时钟提供。

36.17 跟踪端口接口单元（TPIU）

36.17.1 导言

TPIU（Trace port interface unit）在片上数据跟踪和 ITM 之间担当桥梁的作用。

输出的数据流封装成跟踪源 ID，然后被追踪端口分析器采集。

内核嵌入了一个简单的专门为低价调试所设计的 TPIU（由一个特殊版本的 CoreSight TPIU 组成）。

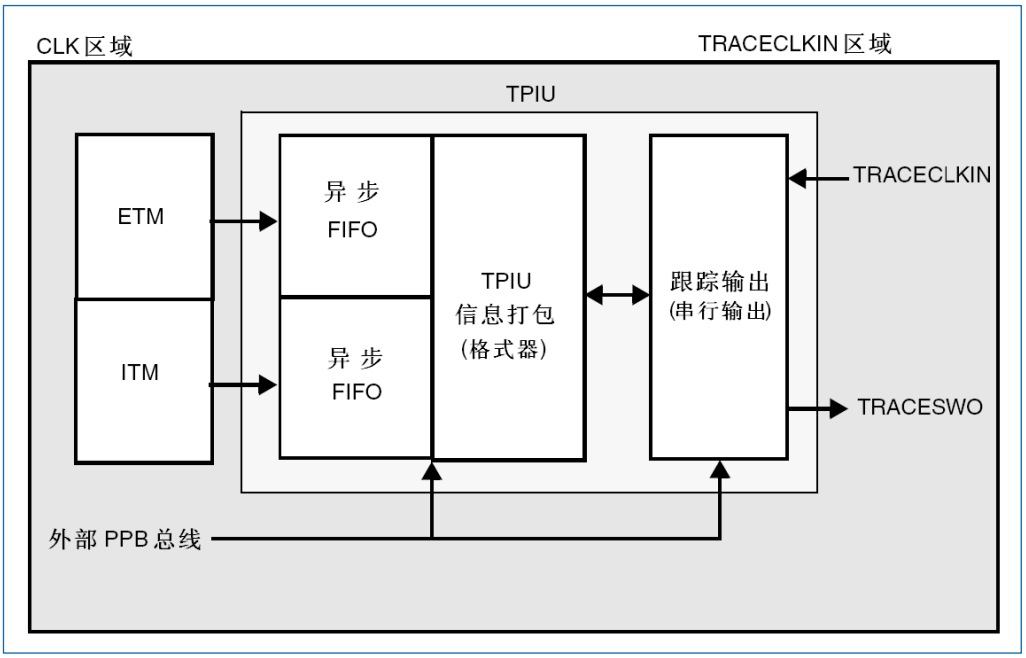


图 36-4 TPIU 框图

36.17.2 跟踪引脚分配

异步模式

异步模式需要 1 个额外的引脚并且存在于所有的封装。此模式仅在串行调试接口有效（不支持 JTAG 调试接口）。

表 36-12 异步跟踪引脚分配

TPUI 引脚	同步跟踪模式		引脚分配
	类型	描述	
TRACESWO	输出	异步跟踪数据输出	PB3

TPUI 跟踪引脚分配

这些引脚在默认状态下不是专用引脚。可以通过设置 MCU 调试模块配置寄存器的 TRACE_IOEN 和 TRACE_MODE 位分配这些引脚。必须由调试器完成设置。

此外，由跟踪的配置决定分配的引脚数（异步）。

异步模式：需要 1 个额外引脚

调试器需要设置 MCU 调试模块配置寄存器的 TRACE_IOEN 和 TRACE_MODE[1:0]位来分配跟踪引脚。默认时，跟踪脚是不分配的。

此寄存器被映射到外部 PPB 并且被 PORESET 所复位（系统复位不复位此寄存器）。调试器可以在系统复位的状态下写该寄存器。

表 36-13 灵活的跟踪引脚分配

DBGMCU_CR 寄存器		引脚用途	跟踪引脚分配
TRACE_IOEN	TRACE_MODE[1:0]		PB3/JTDO/TRACESWO
0	XX	无跟踪 （默认状态）	释放（1）

DBGMCU_CR 寄存器		引脚用途	跟踪引脚分配
TRACE_IOEN	TRACE_MODE[1:0]		PB3/JTDO/TRACESWO
1	00	异步跟踪	TRACESWO
1	01	保留	释放（1）
1	10	保留	
1	11	保留	

(1) 使用串行调试接口时，此引脚被释放，使用 JTAG 调试接口时，此引脚用作 JTDO。

注意：TUIP 的输入时钟 TRACECLKIN 默认接地。所以在比特位 TRACE_IOEN 被置位后，HCLK 需要两个时钟周期。

然后，调试器可以通过写 TPIU 的 SPP_R 寄存器的 PROTOCOL[1:0]位来配置跟踪模式。

- PROTOCOL=01 或 10：串行模式（曼彻斯特或 NRZ 编码）。默认状态为 01 然后通过写 TPIU 的 CPSPS_R（Current Sync Port Size）寄存器的位[3:0]来配置跟踪端口的大小。
- 0x1：1 个引脚（默认）
- 0x2：2 个引脚
- 0x8：4 个引脚

36.17.3 TPUI 格式器

协议格式器输出 16 个字节组成的帧：

- 7 个数据字节
- 8 个多用途字节，由以下部分组成：
  - 1 位（LSB）用来区分数据字节（0）和 ID 字节（1）。
  - 7 位（MSB）可以作为数据或跟踪源 ID 的变化。
- 1 个辅助字节，其中的每个位都对应于 8 个多用途字节中的一个：
  - 如果对应的多用途字节是数据字节，那么这个位是数据的比特 0 位。
  - 如果对应字节是 ID 字节，这个位表明 ID 变化何时生效。

注意：更多信息，请参考 ARM® CoreSight Architecture Specification v1.0（ARM® IHI0029B）

36.17.4 TPUI 帧异步包

TPUI 会产生两种类型的同步包：

- 帧同步包（或全字同步包）该包为 0x7F_FF_FF_FF（LSB 先发），这个序列只有在 0x7F 没有被作为 ID 源编码的时候才能使用。该包在帧之间周期性地输出。在连续模式里，一旦同步帧被发现，TPA 必须抛弃所有这些帧。
- 半字同步包该包为：0x7F_FF（LSB 先发）。它在帧之间或帧内周期性的输出。这些包只存在于连续模式中，并且使能 TPA 检测 IDLE 模式下的 TRACE 口（无 TRACE 被捕捉）。当被 TPA 检测到时，必须将其抛弃。

36.17.5 同步帧包的发送

由于内核的 TPIU 内没有同步计数寄存器，因此同步的触发只能由 DWT 产生。参见 DWT Control Register 寄存器（SYNCTAP[11:10]位）和 DWT Current PC Sampler Cycle Count 寄存器。

TPUI 帧同步包（0x7F_FF_FF_FF）在下列情况时被发送：

- 在每个 TPIU 复位释放后。复位信号同步于 TRACECLKIN 时钟的上升沿释放，这意味着当 DBGMCU_CFG 寄存器的 TRACE_IOEN 位被置位时，同步包就被发送。这种情况下，包 0x7F_FF_FF_FF 后面不跟任何格式的包。
- 在每个 DWT 触发时（假设已事先设置好 DWT），有以下两种情况：
  - 如果 ITM 的 SYNENA 位=0，只发送字 0x7F_FF_FF_FF，后面不跟任何格式的数据包。
  - 如果 ITM 的 SYNENA 位=1，ITM 同步包将跟在（0x80_00_00_00_00_00）后面，由 TPUI 编排格式（加上跟踪源 ID）。

36.17.6 同步模式

跟踪输出数据的引脚数可以为 4 个，2 个或者 1 个，由 TRACED[3:0]配置。

配置时钟输出到调试器（TRACECKO）TRACECLKIN 在内部被驱动，并仅当使用 TRACE 时和 HCLK 相连接。

注意：在此类同步模式中，不需要提供稳定的时钟频率。

TRACE 的 I/O 端口（包括 TRACECKO）由 TRACLKIN 的上升沿驱动（等同于 HCLK）。因此，TRACECKO 的输出频率等于 HCLK/2。

36.17.7 异步模式

调试模块提供一个低成本的，只使用一个引脚的跟踪数据输出功能，即使用异步输出引脚 TRACESWO。但显然，这样的输出数据带宽是有限的。TRACESWO 引脚与 JTDO 引脚复用，只在 SW-DP 调试接口有效，因此 HK32F39A 的所有封装都提供这种功能。

异步模式需要 TRACECLKIN 引脚有平稳的频率提供。对标准的 UART（NRZ）捕捉机制来说，需要 5%的正确度。曼彻斯特编码可放宽到 10%。

36.17.8 TRACECLKIN 在 HK32F39A 内部的连接

TRACECLKIN 输入在 HK32F39A 内部与 HCLK 相连接。这意味着在使用异步跟踪模式时，应用程序应限制使用时间帧保证 CPU 频率的稳定。

当使用异步跟踪功能时需注意：

HK32F39A 微控制器的初时时钟是内部 RC 振荡器，此振荡器在复位状态下的频率与复位后的频率不同。这是由于 RC 校准在复位状态下使用初始值，而这个值在复位释放后会更新。因此，不应该在系统复位状态下激活跟踪端口分析器（Trace Port Analyzer）的跟踪功能（置位 TRACE_IOEN）。因为在复位状态下的同步帧包的比特宽度与复位后的包不同。

36.17.9 TPIU 寄存器

只有当 DEMCR 寄存器的 TRCENA 位被置位时，TPIUAPB 寄存器才可以被读写。否则寄存器读出值为 0（这一位的输出使能 TPIU 的 PCLK）。

表 36-14 重要的 TPIU 寄存器

地址	寄存器	描述
0xE0040004	Current port size	跟踪端口的长度： ● 位 0：端口长度为 1 ● 位 1：端口长度为 2 ● 位 2：端口长度为 3，不支持 ● 位 3：端口长度为 4

地址	寄存器	描述
		仅有一位必须被置位。 默认状态下，端口长度为 1（0x00000001）
0xE00400F0	Selected pin protocol	跟踪端口协议的选择： 位 1:0＝ ● 00：同步跟踪模式 ● 01：串行输出－曼彻斯特编码（默认值） ● 10：串行输出－NRZ ● 11：未定义
0xE0040304	Formatter and flush control	● 位[31:9]：总是 0 ● 位 8：TrigIn：总是 1，指示触发器 ● 位[7:4]：总是 0 ● 位[3:2]：总是 0 ● 位 1：EnFCont： ○ 同步模式下（SelectPinProtocol 寄存器的位[1:0]为 00），此位强制为 1，连续模式下格式器被自动使能。 ○ 异步模式下（SelectPinProtocol 寄存器的位[1:0]不为 00），此比特可以被置位或复位来选择是否使能格式器。 ● 位 0：总是 0 默认值为 0x102 *注意：在同步模式下，由于 TRACECTL 信号没有外部引脚，因此格式器会在连续模式下自动使能。这意味着格式器会插入一些控制包来识别跟踪包的源。*
0xE0040300	Formatter and flush status	没有在 Cortex®-M3 中使用，读出值始终为 0x00000008

36.17.10 配置的例子

- 设置 Debug Exception and Monitor Control 寄存器的 TRCENA 位；
- 在 TPIU Current Port Size 寄存器中写入期望值（默认是 0x1，指示端口长度为 1bit）；
- 向 TPIU Formatter and Flush Control 寄存器中写入 0x102（默认值）；
- 写 TPIU Select Pin Protocol 寄存器，选择异步模式。
- 向 DBGMCU Control 寄存器写入 0x20（置位 IO_TRACEN），为异步模式分配 TRACE 的 I/O 口。此时 TPIU 将发出一个同步包（FF_FF_FF_7F）；
- 配置 ITM 并且写 ITM Stimulus 寄存器输出数据。

37 缩略语与术语

37.1 寄存器描述中的缩略语

缩写	全称	中文描述
r	read-only	只读
w	write-only	只写
rc_w0	read/clear this field by writing ‘0'	可读；可通过对该位域写 0 清除。 对该位域写 1，该位域值无变化。
rc_w1	read/clear this field by writing ‘1'	可读；可通过对该位域写 1 清除。 对该位域写 0，该位域值无变化。
rs	read/set	可读写，与 rw 有区别，通常设置该位域为 1 时启动某种硬件动作；当完成硬件动作后，该位域会被硬件自动清 0。
rw	read/write	可读写该位或指定位。
t	toggle	软件只能通过写 1 来翻转此位，写 0 无效。

37.2 缩略语

缩写	全称	中文描述
AHB	Advanced High-Performance Bus	高级高性能总线
APB	Advanced Peripheral Bus	外围总线
GPIO	General Purpose Input Output	通用输入输出
HSI	High-Speed Internal（Clock Signal）	高速内部（时钟信号）
IAP	In-Application Programming	在线应用编程
ICP	In Circuit Programing	在电路编程
LSI	Low-Speed Internal（Clock Signal）	低速内部（时钟信号）
MCU	Microcontroller Unit	微控制单元
OBL	Option Byte Loader	选项字节装载器
SWD	Serial Wire Debug	内核集成的调试口，它是基于 SWD 协议的 2 线调试接口。

37.3 术语

名称	中文描述
Byte	字节，8 位数据长度。
Halfword	半字，16 位的数据或指令长度。
Option byte	选项字节，保存在 Flash 中的 MCU 配置字节。
Word	字，32 位的数据或指令长度。

38 重要提示

航顺芯片和其他航顺商标均为深圳市航顺芯片技术研发有限公司的商标。本文档提及的其他商标或注册商标，由各自的所有人持有。

在未经深圳市航顺芯片技术研发有限公司同意下，不得以任何形式或途径修改本公司产品规格和数据表中的任何部分以及子部份。深圳市航顺芯片技术研发有限公司在以下方面保留权利：修改数据单和/或产品、停产任一产品或者终止服务不做通知；建议顾客获取最新版本的相关信息，在下定订单前进行核实以确保信息的及时性和完整性。所有的产品都依据订单确认时所提供的销售合同条款出售，条款内容包括保修范围、知识产权和责任范围。

深圳市航顺芯片技术研发有限公司保证在销售期间，产品的性能按照本公司的标准保修。公司认为有必要维持此项保修，会使用测试和其他质量控制技术。除了政府强制规定外，其他仪器的测量表没有必要进行特殊测试。

顾客认可本公司的产品的设计、生产的目的不涉及与生命保障相关或者用于其他危险的活动或者环境的其他系统或产品中。出现故障的产品会导致人身伤亡、财产或环境的损伤（统称高危活动）。人为在高危活动中使用本公司产品，本公司据此不作保修，并且不对顾客或者第三方负有责任。

深圳市航顺芯片技术研发有限公司将会提供与现在一样的技术支持、帮助、建议和信息，（全部包括关于购买的电路板或其他应用程序的设计，开发或调试）。特此声明，对于所有的技术支持、可销性或针对特定用途，及在支持技术无误下，电路板和其他应用程序可以操作或运行的，本公司将不作任何有关此类支持技术的担保，并对您在使用这项支持服务不负任何法律责任。

深圳市航顺芯片技术研发有限公司

联系电话：0755-83247667

网址：www.hsxp-hk.com